



Red Hat Enterprise Linux 8

네트워킹 구성 및 관리

Red Hat Enterprise Linux 8에서 네트워킹 구성 및 관리 가이드

Red Hat Enterprise Linux 8 네트워크 구성 및 관리

Red Hat Enterprise Linux 8에서 네트워크 구성 및 관리 가이드

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

법적 공지

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Configuring_and_managing_networking.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

초록

이 문서에서는 Red Hat Enterprise Linux 8에서 네트워킹을 관리하는 방법을 설명합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체	11
RED HAT 문서에 관한 피드백 제공	12
1장. 일관된 네트워크 인터페이스 장치 이름 지정	13
1.1. 네트워크 인터페이스 장치 이름 지정 계층 구조	13
1.2. 네트워크 장치 이름 변경 작동 방식	14
1.3. 설명된 X86_64 플랫폼에서 예측 가능한 네트워크 인터페이스 장치 이름	14
1.4. SYSTEM Z 플랫폼에서 예측 가능한 네트워크 인터페이스 장치 이름 설명	16
1.5. 설치 중에 일관된 인터페이스 장치 이름 지정 비활성화	16
1.6. 설치된 시스템에서 일관된 인터페이스 장치 이름 지정 비활성화	17
1.7. 이더넷 인터페이스의 접두사 사용자 지정	20
1.8. UDEV 규칙을 사용하여 사용자 정의 네트워크 인터페이스 이름 할당	21
1.9. SYSTEMD 링크 파일을 사용하여 사용자 정의 네트워크 인터페이스 이름 할당	23
1.10. 추가 리소스	24
2장. NETWORKMANAGER 시작하기	25
2.1. NETWORKMANAGER 사용의 이점	25
2.2. NETWORKMANAGER 연결을 관리하는 데 사용할 수 있는 유틸리티 및 애플리케이션의 개요	26
2.3. 수동으로 생성된 IFCFG 파일을 NETWORKMANAGER에 로드	26
3장. 특정 장치를 무시하도록 NETWORKMANAGER 구성	29
3.1. NETWORKMANAGER에서 UNMANAGED로 영구적으로 장치 구성	29
3.2. NETWORKMANAGER에서 일시적으로 관리되지 않는 장치로 구성	30
4장. 텍스트 기반 인터페이스를 사용하여 네트워크 연결 관리	32
4.1. NMTUI 유틸리티 시작	32
4.2. NMTUI를 사용하여 연결 프로필 추가	33
4.3. NMTUI를 사용하여 수정된 연결에 변경 사항 적용	35
5장. NMCLI 시작하기	38
5.1. NMCLI의 다양한 출력 형식	38
5.2. NMCLI에서 탭 완료 사용	38
5.3. 자주 사용되는 NMCLI 명령	39
6장. GNOME GUI를 사용하여 네트워킹 설정 시작하기	41
6.1. GNOME SHELL 네트워크 연결 아이콘을 사용하여 네트워크에 연결	41
7장. NMSTATE 소개	43
7.1. PYTHON 애플리케이션에서 LIBNMSTATE 라이브러리 사용	43
7.2. NMSTATECTL을 사용하여 현재 네트워크 구성 업데이트	44
7.3. 추가 리소스	45
8장. 이더넷 연결 구성	46
8.1. NMCLI를 사용하여 정적 이더넷 연결 구성	46
8.2. NMCLI 대화형 편집기를 사용하여 정적 이더넷 연결 구성	50
8.3. NMSTATECTL을 사용하여 정적 이더넷 연결 구성	54
8.4. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성	57
8.5. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성	59
8.6. NMCLI를 사용하여 동적 이더넷 연결 구성	62
8.7. NMCLI 대화형 편집기를 사용하여 동적 이더넷 연결 구성	64
8.8. NMSTATECTL을 사용하여 동적 이더넷 연결 구성	68
8.9. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성	69
8.10. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성	71

8.11. CONTROL-CENTER를 사용하여 이더넷 연결 구성	73
8.12. NM-CONNECTION-EDITOR를 사용하여 이더넷 연결 구성	77
8.13. NETWORKMANAGER의 DHCP 클라이언트 변경	82
8.14. NETWORKMANAGER 연결의 DHCP 동작 구성	83
8.15. 인터페이스 이름별로 단일 연결 프로필을 사용하여 여러 이더넷 인터페이스 구성	84
8.16. PCI ID를 사용하여 여러 이더넷 인터페이스에 대한 단일 연결 프로파일 구성	86
9장. 와이파이 연결 관리	88
9.1. 무선 규제 도메인 설정	88
9.2. NMCLI를 사용하여 WI-FI 연결 구성	88
9.3. 컨트롤 센터를 사용하여 WI-FI 연결 구성	90
9.4. NMCLI를 사용하여 WI-FI 네트워크에 연결	95
9.5. NMCLI를 사용하여 숨겨진 WI-FI 네트워크에 연결	96
9.6. GNOME GUI를 사용하여 WI-FI 네트워크에 연결	97
9.7. NMCLI를 사용하여 기존 WI-FI 연결에서 802.1X 네트워크 인증 구성	98
10장. VLAN 태그 지정 구성	101
10.1. NMCLI 명령을 사용하여 VLAN 태그 지정 구성	101
10.2. RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 구성	103
10.3. NM-CONNECTION-EDITOR를 사용하여 VLAN 태그 구성	106
10.4. NMSTATECTL을 사용하여 VLAN 태그 지정 구성	110
10.5. RHEL 시스템 역할을 사용하여 VLAN 태그 지정 구성	112
11장. VXLAN을 사용하여 VM의 가상 계층 2 도메인 만들기	115
11.1. VXLAN의 이점	115
11.2. 호스트에서 이더넷 인터페이스 구성	116
11.3. VXLAN이 연결된 네트워크 브리지 만들기	117
11.4. 기존 브리지를 사용하여 LIBVIRT에서 가상 네트워크 만들기	119
11.5. VXLAN을 사용하도록 가상 머신 구성	120
12장. 네트워크 브리지 구성	123
12.1. NMCLI 명령을 사용하여 네트워크 브리지 구성	123
12.2. RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성	128
12.3. NM-CONNECTION-EDITOR를 사용하여 네트워크 브릿지 구성	130
12.4. NMSTATECTL을 사용하여 네트워크 브릿지 구성	135
13장. 네트워크 팀ING 구성	139
13.1. 네트워크 팀ING 이해	139
13.2. 컨트롤러 및 포트 인터페이스의 기본 동작 이해	140
13.3. 네트워크 팀ING 및 본딩 기능 비교	140
13.4. TEAMD 서비스, 러너 및 링크-감시자 이해	142
13.5. TEAMD 서비스 설치	143
13.6. NMCLI 명령을 사용하여 네트워크 팀 구성	143
13.7. RHEL 웹 콘솔을 사용하여 네트워크 팀 구성	148
13.8. NM-CONNECTION-EDITOR를 사용하여 네트워크 팀 구성	152
14장. 네트워크 본딩 구성	158
14.1. 네트워크 본딩 이해	158
14.2. 컨트롤러 및 포트 인터페이스의 기본 동작 이해	159
14.3. 네트워크 팀ING 및 본딩 기능 비교	159
14.4. 연결 모드에 따라 업스트림 스위치 구성	161
14.5. NMCLI 명령을 사용하여 네트워크 본딩 구성	161
14.6. RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성	165
14.7. NM-CONNECTION-EDITOR를 사용하여 네트워크 본딩 구성	169
14.8. NMSTATECTL을 사용하여 네트워크 본딩 구성	173

14.9. RHEL 시스템 역할을 사용하여 네트워크 본딩 구성	176
14.10. VPN을 중단하지 않고 이더넷과 무선 연결을 전환할 수 있는 네트워크 본딩 만들기	178
15장. VPN 연결 구성	183
15.1. CONTROL-CENTER를 사용한 VPN 연결 구성	183
15.2. NM-CONNECTION-EDITOR를 사용하여 VPN 연결 구성	189
15.3. IPSEC 연결 속도를 높이기 위해 ESP 하드웨어 오프로드 자동 감지 및 사용 구성	193
15.4. IPSEC 연결을 가속화하도록 본딩에 ESP 하드웨어 오프로드 구성	195
16장. IP 터널 구성	198
16.1. IPV4 패킷에서 IPV4 트래픽을 캡슐화하도록 NMCLI를 사용하여 IPIP 터널 구성	199
16.2. IPV4 패킷에서 계층-3 트래픽을 캡슐화하도록 NMCLI를 사용하여 GRE 터널 구성	202
16.3. IPV4를 통해 이더넷 프레임 전송하도록 GRE-TAP 터널 구성	206
16.4. 추가 리소스	210
17장. RHEL에서 레거시 네트워크 스크립트 지원	211
17.1. 레거시 네트워크 스크립트 설치	211
18장. 포트 미러링	212
18.1. NMCLI를 사용하여 네트워크 인터페이스 미러링	212
19장. 모든 MAC 주소에서 트래픽을 허용하도록 네트워크 장치 구성	214
19.1. IPROUTE2를 사용하여 모든 트래픽을 허용하도록 네트워크 장치 임시 구성	214
19.2. NMCLI를 사용하여 모든 트래픽을 수락하도록 네트워크 장치를 영구적으로 설정	215
19.3. NMSTATECTL을 사용하여 모든 트래픽을 수락하도록 네트워크 네트워크 장치를 영구적으로 구성	216
20장. FREERADIUS 백엔드와 함께 HOSTAPD를 사용하여 LAN 클라이언트에 대한 802.1X 네트워크 인증 서비스 설정	219
20.1. 사전 요구 사항	219
20.2. 인증기에서 브리지 설정	219
20.3. FREERADIUS의 인증서 요구 사항	221
20.4. 테스트 목적으로 FREERADIUS 서버에 인증서 세트 생성	221
20.5. EAP를 사용하여 네트워크 클라이언트를 안전하게 인증하도록 FREERADIUS 구성	224
20.6. 유선 네트워크에서 HOSTAPD를 인증자로 구성	229
20.7. FREERADIUS 서버 또는 인증자를 대상으로 EAP-TTLS 인증 테스트	231
20.8. FREERADIUS 서버 또는 인증자를 대상으로 EAP-TLS 인증 테스트	234
20.9. HOSTAPD 인증 이벤트를 기반으로 트래픽 차단 및 허용	236
21장. 802.1X 표준을 사용하여 파일 시스템에 저장된 인증서로 RHEL 클라이언트를 네트워크에 인증	240
21.1. NMCLI를 사용하여 기존 이더넷 연결에서 802.1X 네트워크 인증 구성	240
21.2. NMSTATECTL을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성	242
21.3. RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성	244
22장. 기본 게이트웨이 설정 관리	248
22.1. NMCLI를 사용하여 기존 연결에서 기본 게이트웨이 설정	248
22.2. NMCLI 대화형 모드를 사용하여 기존 연결에서 기본 게이트웨이 설정	249
22.3. NM-CONNECTION-EDITOR를 사용하여 기존 연결에서 기본 게이트웨이 설정	251
22.4. CONTROL-CENTER를 사용하여 기존 연결에서 기본 게이트웨이 설정	254
22.5. NMSTATECTL을 사용하여 기존 연결에서 기본 게이트웨이 설정	256
22.6. 시스템 역할을 사용하여 기존 연결에서 기본 게이트웨이 설정	257
22.7. 레거시 네트워크 스크립트를 사용할 때 기존 연결에서 기본 게이트웨이 설정	259
22.8. NETWORKMANAGER가 여러 기본 게이트웨이를 관리하는 방법	260
22.9. 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NETWORKMANAGER 구성	261
22.10. 여러 기본 게이트웨이로 인해 예기치 않은 라우팅 동작 수정	262
23장. 정적 경로 구성	265

23.1. 정적 경로가 필요한 네트워크의 예	265
23.2. NMCLI 명령을 사용하여 정적 경로를 구성하는 방법	268
23.3. NMCLI 명령을 사용하여 정적 경로 구성	270
23.4. CONTROL-CENTER를 사용하여 정적 경로 구성	271
23.5. NM-CONNECTION-EDITOR를 사용하여 정적 경로 구성	273
23.6. NMCLI 대화형 모드를 사용하여 정적 경로 구성	274
23.7. NMSTATECTL을 사용하여 정적 경로 구성	276
23.8. RHEL 시스템 역할을 사용하여 정적 경로 구성	277
23.9. 레거시 네트워크 스크립트를 사용할 때 KEY-VALUE-FORMAT에 정적 경로 구성 파일 생성	280
23.10. 레거시 네트워크 스크립트를 사용할 때 IP-COMMAND-FORMAT에 정적 경로 구성 파일 생성	281
24장. 대체 경로를 정의하도록 정책 기반 라우팅 구성	284
24.1. NETWORKMANAGER를 사용하여 특정 서브넷의 트래픽을 다른 기본 게이트웨이로 라우팅	284
24.2. 레거시 네트워크 스크립트를 사용할 때 정책 기반 라우팅과 관련된 구성 파일의 개요	289
24.3. 레거시 네트워크 스크립트를 사용하여 특정 서브넷의 트래픽을 다른 기본 게이트웨이로 라우팅	290
25장. 더미 인터페이스 생성	298
25.1. NMCLI를 사용하여 IPV4 및 IPV6 주소가 모두 있는 더미 인터페이스 만들기	298
26장. NMSTATE-AUTOCONF를 사용하여 LLDP를 사용하여 네트워크 상태 자동 구성	299
26.1. NMSTATE-AUTOCONF를 사용하여 네트워크 인터페이스 자동 구성	299
27장. LLDP를 사용하여 네트워크 구성 문제 디버그	303
27.1. LLDP 정보를 사용하여 잘못된 VLAN 구성 디버깅	303
28장. 수동으로 키 파일 형식으로 NETWORKMANAGER 프로파일 생성	306
28.1. NETWORKMANAGER 프로파일의 키 파일 형식	306
28.2. 키 파일 형식으로 NETWORKMANAGER 프로파일 생성	307
29장. NETCONSOLE을 사용하여 네트워크에서 커널 메시지 기록	310
29.1. 커널 메시지를 원격 호스트에 기록하도록 NETCONSOLE 서비스 구성	310
30장. SYSTEMD 네트워크 대상 및 서비스	312
30.1. 네트워크 및 네트워크 온라인 SYSTEMD 대상의 차이점	312
30.2. NETWORKMANAGER-WAIT-ONLINE 개요	312
30.3. 네트워크를 시작한 후 시작하도록 SYSTEMD 서비스 구성	313
31장. LINUX 트래픽 제어	315
31.1. 대기열 분야의 개요	315
31.2. RHEL에서 사용 가능한 QDISC	316
31.3. TC 유틸리티를 사용하여 네트워크 인터페이스의 QDISC 검사	318
31.4. 기본 QDISC 업데이트	318
31.5. TC 유틸리티를 사용하여 네트워크 인터페이스의 현재 QDISK를 일시적으로 설정	320
31.6. NETWORKMANAGER를 사용하여 네트워크 인터페이스의 현재 QDISK 영구 설정	320
32장. MULTIPATH TCP 시작하기	322
32.1. MPTCP 이점	322
32.2. MPTCP 지원을 사용하도록 RHEL 준비	322
32.3. IPROUTE2를 사용하여 MPTCP 애플리케이션에 대한 여러 경로 구성 및 활성화	326
32.4. MPTCP 하위 흐름 모니터링	329
32.5. 커널에서 MULTIPATH TCP 비활성화	333
33장. DNS 서버 순서 구성	335
33.1. NETWORKMANAGER가 /ETC/RESOLV.CONF에서 DNS 서버를 주문하는 방법	335
DNS 우선순위 매개변수의 기본값	336
유효한 DNS 우선순위 값:	336

33.2. NETWORKMANAGER 전체 기본 DNS 서버 우선순위 값 설정	337
33.3. NETWORKMANAGER 연결의 DNS 우선 순위 설정	338
34장. IFCFG 파일로 IP 네트워킹 구성	340
34.1. IFCFG 파일을 사용하여 정적 네트워크 설정으로 인터페이스 구성	340
34.2. IFCFG 파일을 사용하여 동적 네트워크 설정으로 인터페이스 구성	341
34.3. IFCFG 파일을 사용하여 시스템 전체 및 개인 연결 프로필 관리	342
35장. 특정 연결에 대해 NETWORKMANAGER를 사용하여 IPV6 비활성화	343
35.1. NMCLI를 사용하여 연결에서 IPV6 비활성화	343
36장. /ETC/RESOLV.CONF 파일 수동 구성	345
36.1. NETWORKMANAGER 구성에서 DNS 처리 비활성화	345
36.2. DNS 설정을 수동으로 구성하려면 /ETC/RESOLV.CONF를 심볼릭 링크로 교체	346
37장. RX 링 버퍼 모니터링 및 튜닝	348
37.1. 삭제된 패킷 수 표시	348
37.2. RX 링 버퍼를 늘려 높은 패킷 드롭 속도를 줄입니다.	349
38장. 802.3 링크 설정 구성	351
38.1. 자동 협상 이해	351
38.2. NMCLI 유틸리티를 사용하여 802.3 링크 설정 구성	351
39장. ETHTOOL 오프로드 기능 구성	353
39.1. NETWORKMANAGER에서 지원하는 오프로드 기능	353
39.2. NETWORKMANAGER를 사용하여 ETHTOOL 오프로드 기능 구성	357
39.3. RHEL 시스템 역할을 사용하여 ETHTOOL 기능 설정	358
40장. ETHTOOL 통합 설정 구성	362
40.1. NETWORKMANAGER에서 지원하는 통합 설정	362
40.2. NETWORKMANAGER를 사용하여 ETHTOOL 통합 설정 구성	363
40.3. RHEL 시스템 역할을 사용하여 ETHTOOL COALESCE 설정 설정	364
41장. MACSEC을 사용하여 동일한 물리적 네트워크에서 계층 2 트래픽 암호화	368
41.1. NMCLI를 사용하여 MACSEC 연결 구성	369
41.2. 추가 리소스	371
42장. 다른 도메인에 다른 DNS 서버 사용	372
42.1. 선택한 DNS 서버로 특정 도메인의 DNS 요청 전송	372
43장. IPVLAN 시작하기	375
43.1. IPVLAN 개요	375
43.2. IPVLAN 모드	375
43.3. MACVLAN 개요	376
43.4. IPVLAN 및 MACVLAN 비교	376
43.5. IPROUTE2를 사용하여 IPVLAN 장치 생성 및 구성	376
44장. 다른 인터페이스에서 동일한 IP 주소 재사용	379
44.1. 다른 인터페이스에서 동일한 IP 주소 영구적으로 재사용	379
44.2. 다른 인터페이스에서 동일한 IP 주소 일시적으로 재사용	381
44.3. 추가 리소스	383
45장. 분리된 VRF 네트워크 내에서 서비스 시작	384
45.1. VRF 장치 구성	384
45.2. 분리된 VRF 네트워크 내에서 서비스 시작	386
46장. 시스템의 라우팅 프로토콜 설정	389

46.1. PVCROUTING 소개	389
46.2. COLLECTDROUTING 설정	391
46.3. COLLECTDR의 설정 수정	392
46.4. 특정 데몬의 구성 수정	393
47장. 기본 네트워크 설정 테스트	394
47.1. PING 유틸리티를 사용하여 다른 호스트에 대한 IP 연결 확인	394
47.2. 호스트 유틸리티를 사용하여 이름 확인	394
48장. NETWORKMANAGER를 사용하여 DHCPCLIENT 종료 후크 실행	396
48.1. NETWORKMANAGER 디스패치 도구 스크립트의 개념	396
48.2. DHCPCLIENT 종료 후크를 실행하는 NETWORKMANAGER 디스패치 스크립트 생성	397
49장. NETWORKMANAGER 디버깅 소개	399
49.1. 디버깅 수준 및 도메인	399
49.2. NETWORKMANAGER 로그 수준 설정	399
49.3. NMCLI를 사용하여 런타임 시 로그 수준 일시적으로 설정	401
49.4. NETWORKMANAGER 로그 보기	402
50장. 네트워크 패킷 캡처	403
50.1. XDPDUMP를 사용하여 XDP 프로그램에서 삭제한 패킷을 포함하여 네트워크 패킷 캡처	403
50.2. 추가 리소스	404
51장. FIREWALLD 사용 및 구성	405
51.1. FIREWALLD 시작하기	405
51.1.1. firewalld, nftables 또는 iptables를 사용하는 경우	405
51.1.2. 영역	406
51.1.3. 사전 정의된 서비스	408
51.1.4. firewalld 시작	408
51.1.5. firewalld 중지	409
51.1.6. 영구 firewalld 구성 확인	409
51.2. FIREWALLD의 현재 상태 및 설정 보기	410
51.2.1. firewalld의 현재 상태 보기	410
51.2.2. GUI를 사용하여 허용된 서비스 보기	410
51.2.3. CLI를 사용하여 firewalld 설정 보기	411
51.3. FIREWALLD를 사용하여 네트워크 트래픽 제어	412
51.3.1. CLI를 사용하여 긴급한 경우 모든 트래픽 비활성화	412
51.3.2. CLI를 사용하여 사전 정의된 서비스로 트래픽 제어	413
51.3.3. GUI를 사용하여 사전 정의된 서비스로 트래픽 제어	414
51.3.4. 새 서비스 추가	415
51.3.5. GUI를 사용하여 포트 열기	416
51.3.6. GUI를 사용하여 프로토콜로 트래픽 제어	417
51.3.7. GUI를 사용하여 소스 포트 열기	417
51.4. CLI를 사용하여 포트 제어	418
51.4.1. 포트 열기	418
51.4.2. 포트 닫기	419
51.5. 시스템 역할을 사용하여 포트 구성	419
51.6. FIREWALLD 영역 작업	422
51.6.1. 영역 나열	422
51.6.2. 특정 영역에 대한 firewalld 설정 수정	422
51.6.3. 기본 영역 변경	423
51.6.4. 영역에 네트워크 인터페이스 할당	423
51.6.5. nmcli를 사용하여 연결에 영역 할당	424
51.6.6. ifcfg 파일에서 네트워크 연결에 수동으로 영역 할당	424

51.6.7. 새 영역 생성	425
51.6.8. 영역 구성 파일	425
51.6.9. 영역 대상을 사용하여 들어오는 트래픽에 대한 기본 동작 설정	426
51.7. 소스에 따라 영역을 사용하여 들어오는 트래픽 관리	427
51.7.1. 소스 추가	427
51.7.2. 소스 제거	428
51.7.3. 소스 포트 추가	428
51.7.4. 소스 포트 제거	429
51.7.5. 특정 도메인에만 서비스를 허용하려면 영역 및 소스를 사용합니다.	429
51.8. 영역 간 전달된 트래픽 필터링	430
51.8.1. 정책 오브젝트와 영역 간의 관계	431
51.8.2. 우선순위를 사용하여 정책 정렬	431
51.8.3. 정책 오브젝트를 사용하여 로컬 호스트 컨테이너와 호스트에 물리적으로 연결된 네트워크 간 트래픽 필터링	432
51.8.4. 정책 오브젝트의 기본 대상 설정	433
51.9. FIREWALLD를 사용하여 NAT 구성	433
51.9.1. 다양한 NAT 유형: 마스크레이드, 소스 NAT, 대상 NAT, 리디렉션	434
51.9.2. IP 주소 마스크레이딩 구성	434
51.10. 포트 전달	435
51.10.1. 리디렉션할 포트 추가	436
51.10.2. 동일한 머신에서 TCP 포트 80을 포트 88으로 리디렉션	436
51.10.3. 리디렉션된 포트 제거	437
51.10.4. 동일한 머신의 포트 88으로 전달된 TCP 포트 80 제거	437
51.11. ICMP 요청 관리	438
51.11.1. ICMP 요청 나열 및 차단	438
51.11.2. GUI를 사용하여 ICMP 필터 구성	441
51.12. FIREWALLD를 사용하여 IP 세트 설정 및 제어	441
51.12.1. CLI를 사용하여 IP 세트 옵션 구성	441
51.13. 리치 규칙 우선순위 지정	444
51.13.1. 우선순위 매개변수가 규칙을 다른 체인으로 구성하는 방법	444
51.13.2. 리치 규칙의 우선 순위 설정	445
51.14. 방화벽 잠금 구성	445
51.14.1. CLI를 사용하여 잠금 구성	445
51.14.2. CLI를 사용하여 잠금 허용 목록 옵션 구성	446
51.14.3. 설정 파일을 사용하여 잠금 허용 목록 옵션 구성	449
51.15. FIREWALLD 영역 내의 다양한 인터페이스 또는 소스 간 트래픽 전달 활성화	450
51.15.1. 기본 타겟이 ACCEPT로 설정된 인트라 영역 전달과 영역의 차이점	450
51.15.2. 영역 내 전달을 사용하여 이더넷 및 Wi-Fi 네트워크 간의 트래픽 전달	450
51.16. ANSIBLE에서 RHEL 시스템 역할을 사용하여 FIREWALLD 설정 구성	452
51.16.1. 방화벽 RHEL 시스템 역할 소개	452
51.16.2. 한 로컬 포트에서 다른 로컬 포트에 들어오는 트래픽 전달	453
51.16.3. 시스템 역할을 사용하여 포트 구성	456
51.16.4. firewalld RHEL 시스템 역할을 사용하여 teamd firewalld 영역 구성	458
51.17. 추가 리소스	460
52장. NFTABLES 시작하기	462
52.1. IPTABLES에서 NFTABLES로 마이그레이션	462
52.1.1. firewalld, nftables 또는 iptables를 사용하는 경우	462
52.1.2. iptables 및 ip6tables 규칙 세트를 nftables로 변환	463
52.1.3. 단일 iptables 및 ip6tables 규칙을 nftables로 변환	464
52.1.4. 일반적인 iptables 및 nftables 명령 비교	465
52.1.5. 추가 리소스	466
52.2. NFTABLES 스크립트 작성 및 실행	466

52.2.1. 지원되는 nftables 스크립트 형식	467
52.2.2. nftables 스크립트 실행 중	468
52.2.3. nftables 스크립트에서 주석 사용	469
52.2.4. nftables 스크립트에서 변수 사용	470
52.2.5. nftables 스크립트에 파일 포함	471
52.2.6. 시스템이 부팅될 때 nftables 규칙을 자동으로 로드	471
52.3. NFTABLES 테이블, 체인 및 규칙 생성 및 관리	472
52.3.1. 표준 체인 우선 순위 값 및 텍스트 이름	473
52.3.2. nftables 규칙 세트 표시	474
52.3.3. nftables 테이블 생성	475
52.3.4. nftables 체인 생성	476
52.3.5. nftables 체인 끝에 규칙 추가	477
52.3.6. nftables 체인의 시작 부분에 규칙 삽입	478
52.3.7. nftables 체인의 특정 위치에 규칙 삽입	479
52.4. NFTABLES를 사용하여 NAT 구성	480
52.4.1. 다양한 NAT 유형: 마스크레이드, 소스 NAT, 대상 NAT, 리디렉션	481
52.4.2. nftables를 사용하여 마스크레이딩 구성	482
52.4.3. nftables를 사용하여 소스 NAT 구성	483
52.4.4. nftables를 사용하여 대상 NAT 구성	484
52.4.5. nftables를 사용하여 리디렉션 구성	485
52.5. NFTABLES 명령의 세트 사용	486
52.5.1. nftables에서 익명 세트 사용	486
52.5.2. nftables에서 명명된 세트 사용	487
52.5.3. 추가 리소스	488
52.6. NFTABLES 명령에서 확인 맵 사용	488
52.6.1. nftables에서 익명 맵 사용	489
52.6.2. nftables에서 명명된 맵 사용	490
52.6.3. 추가 리소스	493
52.7. NFTABLES를 사용하여 포트 전달 구성	493
52.7.1. 들어오는 패킷이 다른 로컬 포트에 전달	493
52.7.2. 특정 로컬 포트에서 들어오는 패킷을 다른 호스트로 전달	494
52.8. NFTABLES를 사용하여 연결 양 제한	495
52.8.1. nftables를 사용하여 연결 수 제한	495
52.8.2. 1분 내에 새로 들어오는 TCP 연결을 10개 이상 시도하는 IP 주소 차단	496
52.9. NFTABLES 규칙 디버깅	497
52.9.1. 카운터를 사용하여 규칙 생성	497
52.9.2. 기존 규칙에 카운터 추가	498
52.9.3. 기존 규칙과 일치하는 패킷 모니터링	499
52.10. NFTABLES 규칙 세트 백업 및 복원	500
52.10.1. nftables 규칙 세트를 파일에 백업	500
52.10.2. 파일에서 nftables 규칙 세트 복원	501
52.11. 추가 리소스	501
53장. 고성능 트래픽 필터링에 XDP-FILTER를 사용하여 로더 공격 방지	502
53.1. XDP-FILTER 규칙과 일치하는 네트워크 패킷 삭제	502
53.2. XDP-FILTER 규칙과 일치하는 네트워크 패킷을 제외하고 모든 네트워크 패킷 삭제	504
54장. DPDK 시작하기	508
54.1. DPDK 패키지 설치	508
54.2. 추가 리소스	508
55장. RHEL에서 EBPf 네트워킹 기능 이해	509
55.1. RHEL의 네트워킹 EBPf 기능 개요	509
XDP	510

AF_XDP	511
트래픽 제어	512
소켓 필터	512
컨트롤 그룹	512
스트림 구문	513
SO_REUSEPORT 소켓 선택	514
흐름 분산 섹터	514
TCP 혼잡 제어	514
캡슐화가 있는 경로	515
소켓 조회	515
55.2. 네트워크 카드별 XDP 기능 개요	515
56장. BPF 컴파일러 컬렉션을 사용한 네트워크 추적	518
56.1. BCC 소개	518
56.2. BCC-TOOLS 패키지 설치	518
56.3. 커널의 허용 대기열에 추가된 TCP 연결 표시	519
56.4. 나가는 TCP 연결 시도 추적	520
56.5. 나가는 TCP 연결의 대기 시간 측정	520
56.6. 커널에서 삭제한 TCP 패킷 및 세그먼트에 대한 세부 정보 표시	521
56.7. TCP 세션 추적	522
56.8. TCP 재전송 추적	523
56.9. TCP 상태 변경 정보 표시	524
56.10. 특정 서브넷에 전송된 TCP 트래픽 요약 및 집계	525
56.11. IP 주소 및 포트를 통한 네트워크 처리량 표시	526
56.12. 설정된 TCP 연결 추적	527
56.13. IPV4 및 IPV6 추적 시도를 수신 대기	528
56.14. 소프트 인터럽트의 서비스 시간 요약	529
56.15. 추가 리소스	529
57장. TIPC 시작하기	530
57.1. TIPC의 아키텍처	530
57.2. 시스템이 부팅될 때 TIPSC 모듈 로드	531
57.3. TIPC 네트워크 생성	532
57.4. 추가 리소스	533
58장. NM-CLOUD-SETUP을 사용하여 공용 클라우드에서 네트워크 인터페이스 자동 구성	535
58.1. NM-CLOUD-SETUP 구성 및 사전 배포	535

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 [CTO Chris Wright의 메시지](#)를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 이를 개선할 수 있는지 알려 주십시오.

- 특정 문구에 대한 간단한 주석은 다음과 같습니다.
 1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **Feedback** (피드백) 버튼이 있는지 확인합니다.
 2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
 3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
 4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성하십시오.
 1. [Bugzilla](#) 웹 사이트로 이동하십시오.
 2. Component로 **Documentation**을 선택하십시오.
 3. **Description** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
 4. **Submit Bug**를 클릭하십시오.

1장. 일관된 네트워크 인터페이스 장치 이름 지정

Red Hat Enterprise Linux는 네트워크 인터페이스에 일관되고 예측 가능한 장치 이름을 지정하는 방법을 제공합니다. 이러한 기능은 네트워크 인터페이스를 찾고 차별화하는 데 도움이 됩니다.

커널은 고정 접두사와 커널이 네트워크 장치를 초기화할 때 증가하는 번호를 연결하여 네트워크 인터페이스에 이름을 할당합니다. 예를 들어 **eth0** 은 시작 시 프로브 중인 첫 번째 장치를 나타냅니다. 그러나 이러한 이름은 새시의 레이블과 반드시 일치하지는 않습니다. 여러 네트워크 어댑터가 있는 최신 서버 플랫폼은 이러한 인터페이스의 비정형 및 반향식 명명에 직면할 수 있습니다. 이는 시스템 보드에 내장된 네트워크 어댑터와 추가 기능 어댑터 모두에 영향을 미칩니다.

Red Hat Enterprise Linux에서 **udev** 장치 관리자는 다양한 명명 체계를 지원합니다. 기본적으로 **udev** 는 펌웨어, 토폴로지 및 위치 정보를 기반으로 고정 이름을 할당합니다. 여기에는 다음과 같은 이점이 있습니다.

- 장치 이름은 완전히 예측 가능합니다.
- 하드웨어를 추가하거나 제거하는 경우에도 장치 이름은 고정된 상태로 유지됩니다. 재조정이 발생하지 않기 때문입니다.
- 결함 있는 하드웨어를 원활하게 교체할 수 있습니다.

1.1. 네트워크 인터페이스 장치 이름 지정 계층 구조

Red Hat Enterprise Linux의 기본 장치 이름 지정이 일관된 경우 **udev** 장치 관리자는 다음 체계를 기반으로 장치 이름을 생성합니다.

스키마	설명	예제
1	장치 이름에는 온보드 장치에 대한 펌웨어 또는 BIOS 제공 인덱스 번호가 포함됩니다. 이 정보를 사용할 수 없거나 적용할 수 없는 경우 udev 는 스키마 2를 사용합니다.	eno1
2	장치 이름에는 펌웨어 또는 BIOS에서 제공하는 PCI Express(PCIe) 핫 플러그 슬롯 번호가 포함됩니다. 이 정보를 사용할 수 없거나 적용할 수 없는 경우 udev 는 스키마 3을 사용합니다.	ens1
3	장치 이름은 하드웨어의 커넥터의 실제 위치를 포함합니다. 이 정보를 사용할 수 없거나 적용할 수 없는 경우 udev 는 스키마 5를 사용합니다.	enp2s0
4	장치 이름에는 MAC 주소가 포함됩니다. Red Hat Enterprise Linux는 기본적으로 이 스키마를 사용하지 않지만 관리자는 선택적으로 사용할 수 있습니다.	enx525400d5e0fb
5	예측할 수 없는 기존의 커널 이름 지정 체계. udev 가 다른 체계를 적용할 수 없는 경우, 장치 관리자는 이 체계를 사용합니다.	eth0

기본적으로 Red Hat Enterprise Linux는 **/usr/lib/systemd/network/99-default.link** 파일의 **NamePolicy** 설정을 기반으로 장치 이름을 선택합니다. **NamePolicy** 의 값 순서는 중요합니다. Red Hat Enterprise Linux는 파일에 지정된 첫 번째 장치 이름과 **udev** 를 생성합니다.

커널 장치의 이름을 변경하기 위해 **udev** 규칙을 수동으로 구성한 경우 해당 규칙이 우선합니다.

1.2. 네트워크 장치 이름 변경 작동 방식

Red Hat Enterprise Linux에서는 기본적으로 일관된 장치 명명이 활성화됩니다. **udev** 장치 관리자는 장치의 이름을 변경하기 위해 다른 규칙을 처리합니다. 다음 목록에서는 이러한 규칙을 처리하는 **udev** 순서와 해당 규칙이 담당하는 작업을 설명합니다.

1. **/usr/lib/udev/rules.d/60-net.rules** 파일은 **/etc/sysconfig/network-scripts/IKE-*** 파일에서 **/lib/udev/rename_device** 도우미에서 **HWADDR** 매개변수를 검색하는 것을 정의합니다. 변수에 설정된 값이 인터페이스의 MAC 주소와 일치하면 도우미 유틸리티에서 파일의 **DEVICE** 매개변수에 설정된 이름으로 인터페이스의 이름을 변경합니다.
2. **/usr/lib/udev/rules.d/71-biosdevname.rules** 파일은 **biosdevname** 유틸리티에서 이전 단계에서 이름이 변경되지 않은 경우 이름 지정 정책에 따라 인터페이스 이름을 변경한다는 것을 정의합니다.
3. **/usr/lib/udev/rules.d/75-net-description.rules** 파일은 **udev** 가 네트워크 인터페이스 장치를 검사함을 정의하고 다음 단계에서 처리할 **udev**-internal 변수의 속성을 설정합니다. 이러한 속성 중 일부는 정의되지 않을 수 있습니다.
4. **/usr/lib/udev/rules.d/80-net-setup-link.rules** 파일에서 **net_setup_link udev** 를 호출한 다음 정책을 적용합니다. 다음은 **/usr/lib/systemd/network/99-default.link** 파일에 저장된 기본 정책입니다.

```
[Link]
NamePolicy=kernel database onboard slot path
MACAddressPolicy=persistent
```

이 정책에서 커널이 영구 이름을 사용하는 경우 **udev** 는 인터페이스의 이름을 바꾸지 않습니다. 커널이 영구 이름을 사용하지 않는 경우 **udev** 는 인터페이스의 이름을 **udev** 의 하드웨어 데이터베이스에서 제공한 이름으로 변경합니다. 이 데이터베이스를 사용할 수 없는 경우 Red Hat Enterprise Linux는 위에서 설명한 메커니즘으로 대체됩니다.

또는 MAC(Media Access Control) 주소 기반 인터페이스 이름의 경우 이 파일의 **NamePolicy** 매개변수를 **mac** 로 설정합니다.

5. **/usr/lib/udev/rules.d/80-net-setup-link.rules** 파일은 **udev** -internal 매개변수에 따라 다음 순서로 인터페이스 이름을 변경함을 정의합니다.
 - a. **ID_NET_NAME_ONBOARD**
 - b. **ID_NET_NAME_SLOT**
 - c. **ID_NET_NAME_PATH**

하나의 매개 변수가 설정되지 않은 경우 **udev** 는 다음 매개 변수를 사용합니다. 매개 변수를 설정하지 않으면 인터페이스의 이름이 변경되지 않습니다.

3단계와 4단계는 [네트워크 인터페이스 장치 명명 계층 구조](#)에 설명된 1-4단계를 구현합니다.

추가 리소스

- [이더넷 인터페이스의 접두사 사용자 지정](#)
- **NamePolicy** 매개변수에 대한 자세한 내용은 **systemd.link(5)** 매뉴얼 페이지를 참조하십시오.

1.3. 설명된 X86_64 플랫폼에서 예측 가능한 네트워크 인터페이스 장치 이름

일관된 네트워크 장치 이름 기능이 활성화되면 **udev** 장치 관리자는 다른 기준에 따라 장치 이름을 생성합니다. 이 섹션에서는 Red Hat Enterprise Linux를 x86_64 플랫폼에 설치할 때의 이름 지정 체계에 대해 설명합니다.

인터페이스 이름은 인터페이스 유형에 따라 2자 접두사로 시작됩니다.

- 이더넷용 EN
- 무선 LAN(WLAN)
- 무선 WW(WW for wireless wide area network)

또한 다음 중 하나가 **udev** 장치 관리자가 적용하는 스키마를 기반으로 위에서 언급한 접두사 중 하나에 추가됩니다.

- **o<on-board_index_number>**
- **s<hot_plug_slot_index_number>[f<function>][d<device_id>]**

모든 다중 기능 **PCI** 장치에는 함수 0 장치를 포함하여 장치 이름에 [f <function>] 번호가 있습니다.

- **x<MAC_address>**
- **[P<domain_number>]p<bus>s<slot>[f<function>][d<device_id>]**

[P <domain_number>] 부분은 **PCI** 지리적 위치를 정의합니다. 이 부분은 도메인 번호가 0 이 아닌 경우에만 설정됩니다.

- **[P<domain_number>]p<bus>s<slot>[f<function>][u<usb_port>][...][c<config>][i<interface>]**

USB 장치의 경우 허브의 포트 번호 전체 체인이 구성됩니다. 이름이 최대(15자)보다 길면 이름을 내보내지 않습니다. 체인에 **USB** 장치가 여러 개 있는 경우 **udev** 는 **USB** 구성 설명자(c1) 및 **USB** 인터페이스 설명자(i0)의 기본값을 표시하지 않습니다.

1.4. SYSTEM Z 플랫폼에서 예측 가능한 네트워크 인터페이스 장치 이름 설명

일관된 네트워크 장치 이름 기능이 활성화되면 **System z** 플랫폼의 **udev** 장치 관리자는 버스 ID를 기반으로 장치의 이름을 생성합니다. 버스 ID는 **s390** 채널 하위 시스템에서 장치를 식별합니다.

채널 명령 단어(CCW) 장치의 경우 버스 ID는 선행 **0.n** 접두사가 있는 장치 번호입니다. 여기서 **n** 은 하위 채널 세트 ID입니다.

이더넷 인터페이스의 이름은 (예: **enccw0.0.1234**)입니다. 직렬 라인 인터넷 프로토콜 (**SLIP**) 채널-토 채널 (**CTC**) 네트워크 장치 (예: **slccw0.0.1234**).

사용 가능한 네트워크 장치 및 버스 ID를 표시하려면 **znetconf -c** 또는 **lscss -a** 명령을 사용합니다.

1.5. 설치 중에 일관된 인터페이스 장치 이름 지정 비활성화

이 섹션에서는 설치 중에 일관된 인터페이스 장치 이름을 비활성화하는 방법을 설명합니다.



주의

Red Hat은 일관된 장치 이름을 비활성화하지 않을 것을 권장하며 네트워크 인터페이스가 두 개 이상인 호스트에서 이 기능을 지원하지 않습니다. 일관된 장치 이름 지정을 비활성화하면 다양한 종류의 문제가 발생할 수 있습니다. 예를 들어 시스템에 다른 네트워크 인터페이스 카드를 추가하면 **eth0** 과 같은 커널 장치 이름 할당이 더 이상 수정되지 않습니다. 결과적으로 재부팅 후 커널은 장치의 이름을 다르게 지정할 수 있습니다.

절차

1. **Red Hat Enterprise Linux 8 설치 미디어 부팅.**
2. 부팅 관리자에서 **Install Red Hat Enterprise Linux 8** 을 선택하고 **Tab** 키를 눌러 항목을 편집합니다.

3.

net.ifnames=0 매개변수를 커널 명령줄에 추가합니다.

```
vmlinux... net.ifnames=0
```

4.

Enter 를 눌러 설치를 시작합니다.

추가 리소스

•

RHEL 7 및 RHEL 8에서 net.ifnames=0을 설정하는 것이 안전합니까?

•

RHEL 7에서 커널 NIC 이름을 사용할 때 RHEL 8로 인플레이스 업그레이드를 수행하는 방법

1.6. 설치된 시스템에서 일관된 인터페이스 장치 이름 지정 비활성화

이 섹션에서는 이미 설치된 **RHEL** 시스템에서 일관된 인터페이스 장치 이름 지정을 비활성화하는 방법을 설명합니다.



주의

Red Hat은 일관된 장치 이름을 비활성화하지 않을 것을 권장하며 네트워크 인터페이스가 두 개 이상인 호스트에서 이 기능을 지원하지 않습니다. 일관된 장치 명명을 비활성화하면 다른 종류의 문제가 발생할 수 있습니다. 예를 들어 시스템에 다른 네트워크 인터페이스 카드를 추가하면 **eth0** 과 같은 커널 장치 이름 할당이 더 이상 수정되지 않습니다. 결과적으로 재부팅 후 커널은 장치의 이름을 다르게 지정할 수 있습니다.

사전 요구 사항

•

시스템은 일관된 인터페이스 장치 이름을 사용합니다. 이는 기본값입니다.

절차

1.

/etc/default/grub 파일을 편집하고 **net.ifnames=0** 매개 변수를 **GRUB_CMDLINE_LINUX** 변수에 추가합니다.

```
GRUB_CMDLINE_LINUX="... net.ifnames=0"
```

2.

grub.cfg 파일을 다시 빌드합니다.

•

UEFI 부팅 모드가 있는 시스템에서 다음을 수행합니다.

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

•

기존 부팅 모드가 있는 시스템에서 다음을 수행합니다.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

3.

현재 프로필 이름과 관련 장치 이름을 표시합니다.

```
# nmcli -f NAME,DEVICE,FILENAME connection show  
NAME          DEVICE  FILENAME  
System enp1s0 enp1s0 /etc/sysconfig/network-scripts/ifcfg-enp1s0  
System enp7s0 enp7s0 /etc/NetworkManager/system-  
connections/enp7s0.nmconnection
```

각 장치와 연결된 프로필 이름 및 구성 파일을 확인합니다.

4.

모든 연결 프로필에서 **HWADDR** 매개변수를 제거합니다.

```
# sed -i '/^HWADDR=/d' /etc/sysconfig/network-scripts/ifcfg-enp1s0  
/etc/NetworkManager/system-connections/enp7s0.nmconnection
```

5.

이더넷 장치와 연결된 **MAC** 주소를 표시합니다.

```
# ip link show  
...  
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state  
UP mode DEFAULT group default qlen 1000  
    link/ether 00:53:00:c5:98:1c brd ff:ff:ff:ff:ff:ff  
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state  
UP mode DEFAULT group default qlen 1000  
    link/ether 00:53:00:b6:87:c6 brd ff:ff:ff:ff:ff:ff
```

6.

호스트를 재부팅합니다.

```
# reboot
```

7.

재부팅 후 이더넷 장치를 표시하고 **MAC** 주소를 기반으로 새 인터페이스 이름을 식별합니다.

```
# ip link show
```

```
...
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
```

```
link/ether 00:53:00:b6:87:c6 brd ff:ff:ff:ff:ff:ff
```

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
```

```
link/ether 00:53:00:c5:98:1c brd ff:ff:ff:ff:ff:ff
```

현재 출력을 이전 출력과 비교하면 다음을 수행하십시오.

•

인터페이스 **enp7s0** (MAC 주소 **00:53:b6:87:c6**)은 이제 **eth0** 로 이름이 지정됩니다.

•

인터페이스 **enp1s0** (MAC 주소 **00:53:00:c5:98:1c**)은 이제 **eth1** 로 이름이 지정됩니다.

8.

구성 파일의 이름을 변경합니다.

```
# mv /etc/NetworkManager/system-connections/enp7s0.nmconnection
```

```
/etc/NetworkManager/system-connections/eth0.nmconnection
```

```
# mv /etc/sysconfig/network-scripts/ifcfg-enp1s0 /etc/sysconfig/network-scripts/ifcfg-eth1
```

9.

NetworkManager를 다시 로드합니다.

```
# nmcli connection reload
```

10.

구성 파일에 프로필 이름이 설정되지 않은 경우 **NetworkManager**는 기본값을 사용합니다. 연결 이름을 변경하고 다시 로드한 후 현재 프로필 이름을 확인하려면 다음을 입력합니다.

```
# nmcli -f NAME,DEVICE,FILENAME connection show
```

```
NAME      FILENAME
```

```
System enp7s0 /etc/NetworkManager/system-connections/eth0.nmconnection
```

```
System enp1s0 /etc/sysconfig/network-scripts/ifcfg-eth1
```

다음 단계에서 프로필 이름이 필요합니다.

11.

NetworkManager 연결 프로필의 이름을 변경하고 각 프로필의 인터페이스 이름을 업데이트합니다.

```
# nmcli connection modify "System enp7s0" connection.id eth0 connection.interface-name eth0
# nmcli connection modify "System enp1s0" connection.id eth1 connection.interface-name eth1
```

12.

NetworkManager 연결을 다시 활성화합니다.

```
# nmcli connection up eth0
# nmcli connection up eth1
```

1.7. 이더넷 인터페이스의 접두사 사용자 지정

Red Hat Enterprise Linux를 설치하는 동안 이더넷 인터페이스 이름의 접두사를 사용자 지정할 수 있습니다.



중요

Red Hat은 이미 배포된 시스템에서 **prefixdevname** 유틸리티를 사용하여 접두사 사용자 지정을 지원하지 않습니다.

RHEL 설치 후 **udev** 서비스는 이더넷 장치 < prefix > . < index > . < index > . 예를 들어 접두사 네트워크, **RHEL** 이름 이더넷 인터페이스 **net 0,net1** 등을 선택하는 경우.

사전 요구 사항

- 설정할 접두사는 다음 요구 사항을 충족합니다.
 - **ASCII** 문자로 구성됩니다.
 - 영숫자 문자열입니다.

- 이 문자는 16자보다 짧습니다.
- **eth,eno,ens, em** 과 같이 네트워크 인터페이스 이름 지정에 사용되는 다른 잘 알려진 접두사와 충돌하지 않습니다.

절차

1. **Red Hat Enterprise Linux** 설치 미디어를 부팅합니다.
2. 부팅 관리자에서 다음을 수행합니다.
 - a. **Install Red Hat Enterprise Linux <version>** 항목을 선택하고 **Tab** 을 눌러 항목을 편집합니다.
 - b. 커널 옵션에 **net.ifnames.prefix= <prefix >**를 추가합니다.
 - c. **Enter** 를 눌러 설치 프로그램을 시작합니다.
3. **Red Hat Enterprise Linux** 설치.

검증

- 설치 후 이더넷 인터페이스를 표시합니다.

```
# ip link show
...
2: net0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:53:00:c5:98:1c brd ff:ff:ff:ff:ff:ff
3: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:53:00:c2:39:9e brd ff:ff:ff:ff:ff:ff
...
```

1.8. UDEV 규칙을 사용하여 사용자 정의 네트워크 인터페이스 이름 할당

udev 장치 관리자는 인터페이스 이름을 사용자 지정하는 일련의 규칙을 지원합니다.

절차

1.

모든 네트워크 인터페이스 및 **MAC** 주소를 표시합니다.

```
# ip link list
```

```
enp6s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether b4:96:91:14:ae:58 brd ff:ff:ff:ff:ff:ff
enp6s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether b4:96:91:14:ae:5a brd ff:ff:ff:ff:ff:ff
enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 00:90:fa:6a:7d:90 brd ff:ff:ff:ff:ff:ff
```

2.

다음 콘텐츠를 사용하여 `/etc/udev/rules.d/70-custom-ifnames.rules` 를 만듭니다.

```
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="b4:96:91:14:ae:58",ATTR{type}=="1",NAME="provider0"
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="b4:96:91:14:ae:5a",ATTR{type}=="1",NAME="provider1"
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:90:fa:6a:7d:90",ATTR{type}=="1",NAME="dmz"
```

이러한 규칙은 네트워크 인터페이스의 **MAC** 주소와 일치하고 **NAME** 속성에 지정된 이름으로 이름을 바꿉니다. 이 예에서 **ATTR{type}** 매개변수 값 1 은 인터페이스가 이더넷 유형임을 정의합니다.

검증

1.

시스템을 재부팅합니다.

```
# reboot
```

2.

각 **MAC** 주소의 인터페이스 이름이 규칙 파일의 **NAME** 매개변수에 설정된 값과 일치하는지 확인합니다.

```
# ip link show
```

```
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
```

```

mode DEFAULT group default qlen 1000
  link/ether b4:96:91:14:ae:58 brd ff:ff:ff:ff:ff:ff
  altname enp6s0f0
provider1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
  link/ether b4:96:91:14:ae:5a brd ff:ff:ff:ff:ff:ff
  altname enp6s0f1
dmz: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
  link/ether 00:90:fa:6a:7d:90 brd ff:ff:ff:ff:ff:ff
  altname enp4s0f0

```

추가 리소스

- `udev(7)` 도움말 페이지
- `udevadm(8)` 도움말 페이지
- `kernel-doc` 패키지에서 제공하는 `/usr/src/kernels/ <kernel_version>/include/uapi/linux/if_arp.h`

1.9. SYSTEMD 링크 파일을 사용하여 사용자 정의 네트워크 인터페이스 이름 할당

네트워크 인터페이스를 `provider0` 로 변경하여 이름 지정 체계를 생성합니다.

절차

1. 모든 인터페이스 이름과 **MAC** 주소를 표시합니다.

```
# ip link show
```

```

enp6s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
  link/ether b4:96:91:14:ae:58 brd ff:ff:ff:ff:ff:ff
enp6s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
  link/ether b4:96:91:14:ae:5a brd ff:ff:ff:ff:ff:ff
enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
  link/ether 00:90:fa:6a:7d:90 brd ff:ff:ff:ff:ff:ff

```

2. **MAC** 주소 `b4:96:91:14:ae:58` 을 `provider0` 으로 이름을 지정하려면 다음 콘텐츠를 사용하여 `/etc/systemd/network/70-custom-ifnames.link` 파일을 생성합니다.

```
[Match]
MACAddress=b4:96:91:14:ae:58
```

```
[Link]
Name=provider0
```

이 링크 파일은 **MAC** 주소와 일치하고 네트워크 인터페이스의 이름을 **Name** 매개변수에 설정된 이름으로 변경합니다.

검증

1. 시스템을 재부팅합니다.

```
# reboot
```

2. 링크 파일에 지정한 **MAC** 주소가 있는 장치가 **provider0**에 할당되었는지 확인합니다.

```
# ip link show
```

```
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
    link/ether b4:96:91:14:ae:58 brd ff:ff:ff:ff:ff:ff
```

추가 리소스

- [systemd.link\(5\) man page](#)

1.10. 추가 리소스

- [udev 장치 관리자에 대한 자세한 내용은 udev \(7\) 도움말 페이지를 참조하십시오.](#)

2장. NETWORKMANAGER 시작하기

기본적으로 **RHEL**은 **NetworkManager**를 사용하여 네트워크 구성 및 연결을 관리합니다.

2.1. NETWORKMANAGER 사용의 이점

NetworkManager 사용의 주요 이점은 다음과 같습니다.

- 네트워크 구성 및 상태를 쿼리하고 제어할 수 있는 **D-Bus**를 통해 **API** 제공. 이러한 방식으로 여러 애플리케이션에서 네트워크를 확인하고 구성하여 동기화 및 최신 네트워킹 상태를 보장할 수 있습니다. 예를 들어 웹 브라우저를 통해 서버를 모니터링하고 구성하는 **RHEL** 웹 콘솔은 **NetworkManager D-BUS** 인터페이스를 사용하여 네트워킹과 **Gnome GUI**, **nmcli** 및 **nm-connection-editor** 툴을 구성합니다. 이러한 도구 중 하나에서 수행된 각 변경 사항은 다른 도구에서 감지됩니다.
- 네트워크 관리의 용이성: **NetworkManager** 는 네트워크 연결이 작동하는지 확인합니다. 시스템에 네트워크 구성이 없지만 네트워크 장치가 있음을 감지하면 **NetworkManager** 에서 임시 연결을 생성하여 연결을 제공합니다.
- 사용자에게 간편한 연결 설정 제공: **NetworkManager** 는 **GUI**, **nmtui**, **nmcli** 등 다양한 도구를 통해 관리를 제공합니다.
- 구성 유연성 지원. 예를 들어 **NetworkManager** 는 **Automate** 인터페이스를 구성하고 사용 가능한 네트워크를 스캔하고 표시합니다. 인터페이스를 선택할 수 있으며 **NetworkManager** 는 재부팅 프로세스 후 자동 연결을 제공하는 필수 자격 증명을 표시합니다. **NetworkManager** 는 네트워크 별칭, **IP** 주소, 정적 경로, **DNS** 정보 및 **VPN** 연결을 구성할 수 있으며 다양한 연결별 매개 변수를 구성할 수 있습니다. 요구 사항을 반영하도록 구성 옵션을 수정할 수 있습니다.
- 재부팅 프로세스 후 장치 상태를 유지하고 다시 시작하는 동안 관리 모드로 설정된 인터페이스를 대신합니다.
- 명시적으로 관리되지 않지만 사용자 또는 다른 네트워크 서비스에 의해 수동으로 제어되는 장치 처리.

추가 리소스

- **RHEL 8** 웹 콘솔을 사용하여 시스템 관리.

2.2. NETWORKMANAGER 연결을 관리하는 데 사용할 수 있는 유틸리티 및 애플리케이션의 개요

다음 유틸리티 및 애플리케이션을 사용하여 **NetworkManager** 연결을 관리할 수 있습니다.

- **nmcli**: 연결을 관리하는 명령줄 유틸리티입니다.
- **nmtui**: 쿼리 기반 텍스트 사용자 인터페이스(TUI). 이 애플리케이션을 사용하려면 **NetworkManager-tui** 패키지를 설치합니다.
- **nm-connection-editor**: **NetworkManager** 관련 작업의 GUI(그래픽 사용자 인터페이스). 이 애플리케이션을 시작하려면 **GNOME** 세션의 터미널에 **nm-connection-editor** 를 입력합니다.
- **control-center**: 데스크탑 사용자를 위해 **GNOME** 셸에서 제공하는 GUI. 이 애플리케이션은 **nm-connection-editor** 보다 적은 기능을 지원합니다.
- **GNOME** 셸의 네트워크 연결 아이콘: 이 아이콘은 네트워크 연결 상태를 나타내고 사용 중인 연결 유형의 시각적 표시자 역할을 합니다.

추가 리소스

- [텍스트 기반 인터페이스를 사용하여 네트워크 연결 관리](#)
- [nmcli 시작하기](#)

2.3. 수동으로 생성된 IFCFG 파일을 NETWORKMANAGER에 로드

Red Hat Enterprise Linux에서 **ifcfg** 파일을 편집하는 경우 **NetworkManager** 는 변경 사항을 자동으로 인식하지 못하므로 변경 사항을 확인하라는 메시지가 표시됩니다. 도구 중 하나를 사용하여 **NetworkManager** 프로파일 설정을 업데이트하는 경우 **NetworkManager** 는 해당 프로파일을 다시 연결할 때까지 이러한 변경 사항을 구현하지 않습니다. 예를 들어 편집기를 사용하여 구성 파일이 변경된 경우 **NetworkManager** 는 구성 파일을 다시 읽어야 합니다.



중요

NetworkManager는 키 파일 형식으로 저장된 프로필을 지원합니다. 그러나 **NetworkManager API**를 사용하여 프로필을 만들거나 업데이트하는 경우 **NetworkManager**는 **ifcfg** 형식을 사용합니다.

향후 주요 **RHEL** 릴리스에서는 키 파일 형식이 기본값입니다. 구성 파일을 수동으로 생성하고 관리하려면 키 파일 형식을 사용하는 것이 좋습니다. 자세한 내용은 [키 파일 형식으로 NetworkManager 프로필 수동 생성](#)을 참조하십시오.

`/etc/sysconfig/` 디렉토리는 구성 파일 및 스크립트의 위치입니다. 대부분의 네트워크 구성 정보는 `/etc/NetworkManager/` 하위 디렉토리에 저장되는 VPN, 모바일 광대역 및 PPPoE 구성을 제외하고 여기에 저장됩니다. 예를 들어, 인터페이스 관련 정보는 `/etc/sysconfig/network-scripts/` 디렉터리에 있는 `ifcfg` 파일에 저장됩니다.

VPN, 모바일 광대역 및 PPPoE 연결에 대한 정보는 /etc/NetworkManager/system-connections/ 에 저장됩니다.

절차

1. 새 구성 파일을 로드하려면 다음을 수행합니다.

```
# nmcli connection load /etc/sysconfig/network-scripts/ifcfg-connection_name
```

2. **NetworkManager에 이미 로드된 연결 파일을 업데이트한 경우 다음을 입력합니다.**

```
# nmcli connection up connection_name
```

추가 리소스

- *NetworkManager(8) 도움말 페이지*
- *NetworkManager.conf(5) 도움말 페이지*
- */usr/share/doc/initscripts/sysconfig.txt*

- ***ifcfg(8)* 도움말 페이지**

3장. 특정 장치를 무시하도록 NETWORKMANAGER 구성

기본적으로 **NetworkManager**는 **lo (loopback)** 장치를 제외한 모든 장치를 관리합니다. 그러나 **NetworkManager**가 이러한 장치를 무시하도록 특정 장치를 **Unmanaged**로 설정할 수 있습니다. 이 설정을 사용하면 스크립트를 사용하여 이러한 장치를 수동으로 관리할 수 있습니다.

3.1. NETWORKMANAGER에서 UNMANAGED로 영구적으로 장치 구성

인터페이스 이름, **MAC** 주소 또는 장치 유형과 같은 여러 기준에 따라 장치를 관리되지 않은 상태로 구성할 수 있습니다. 이 절차에서는 **NetworkManager**에서 **enp1s0** 인터페이스를 **Unmanaged**로 설정하는 방법을 설명합니다.

네트워크 장치를 임시로 비관리형으로 구성하려면 **NetworkManager**에서 수동으로 장치를 비관리형으로 구성을 참조하십시오.

절차

1. 선택 사항: 비관리로 설정하려는 장치를 식별하는 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE TYPE    STATE    CONNECTION
enp1s0 ethernet disconnected --
...
```

2. 다음 콘텐츠를 사용하여 **/etc/NetworkManager/conf.d/99-unmanaged-devices.conf** 파일을 생성합니다.

```
[keyfile]
unmanaged-devices=interface-name:enp1s0
```

여러 장치를 관리되지 않음으로 설정하려면 **unmanaged-devices** 매개변수의 항목을 **Semic**로 분리합니다.

```
[keyfile]
unmanaged-devices=interface-name:interface_1;interface-name:interface_2;...
```

3. **NetworkManager** 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

검증 단계

- 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE TYPE   STATE   CONNECTION
enp1s0 ethernet unmanaged --
...
```

enp1s0 장치 옆에 관리되지 않는 상태는 **NetworkManager**가 이 장치를 관리하지 않음을 나타냅니다.

추가 리소스

- **NetworkManager.conf(5)** 도움말 페이지의 장치 목록 형식 섹션입니다.

3.2. NETWORKMANAGER에서 일시적으로 관리되지 않는 장치로 구성

인터페이스 이름, **MAC** 주소 또는 장치 유형과 같은 여러 기준에 따라 장치를 관리되지 않은 상태로 구성할 수 있습니다. 이 절차에서는 **NetworkManager**에서 **enp1s0** 인터페이스를 **Unmanaged**로 일시적으로 설정하는 방법을 설명합니다.

예를 들어 테스트용으로 이 방법을 사용합니다. 네트워크 장치를 관리되지 않음으로 영구적으로 구성하려면 **NetworkManager**에서 **영구적으로 장치를 비관리형으로** 구성을 참조하십시오.

절차

1. 선택 사항: 비관리로 설정하려는 장치를 식별하는 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE TYPE   STATE   CONNECTION
enp1s0 ethernet disconnected --
...
```

2. **enp1s0** 장치를 **Unmanaged** 상태로 설정합니다.

```
# nmcli device set enp1s0 managed no
```

검증 단계

- 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE TYPE    STATE    CONNECTION
enp1s0 ethernet unmanaged --
...
```

enp1s0 장치 옆에 관리되지 않는 상태는 **NetworkManager**가 이 장치를 관리하지 않음을 나타냅니다.

추가 리소스

- **NetworkManager.conf(5)** 도움말 페이지의 장치 목록 형식 섹션

4장. 텍스트 기반 인터페이스를 사용하여 네트워크 연결 관리

nmtui 애플리케이션은 **NetworkManager** 를 위한 텍스트 사용자 인터페이스(TUI)입니다. 다음 섹션에서는 **nmtui** 를 사용하여 네트워크 인터페이스를 구성하는 방법을 설명합니다.



참고

nmtui 애플리케이션은 일부 연결 유형을 지원하지 않습니다. 특히 **802.1X** 인증이 필요한 **VPN** 연결 또는 이더넷 연결을 추가하거나 수정할 수 없습니다.

4.1. NMTUI 유틸리티 시작

이 절차에서는 **NetworkManager** 텍스트 사용자 인터페이스인 **nmtui** 를 시작하는 방법을 설명합니다.

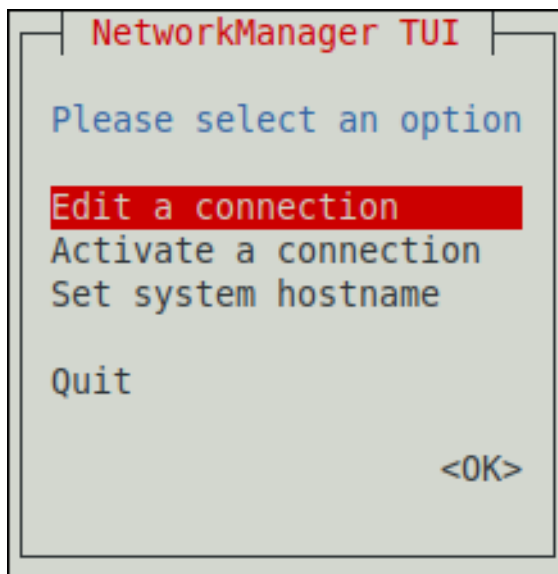
사전 요구 사항

- **NetworkManager-tui** 패키지가 설치됩니다.

절차

1. **nmtui** 를 시작하려면 다음을 입력합니다.

```
# nmtui
```



2. 탐색하려면 다음을 수행합니다.

- 커서를 사용하거나 탭을 눌러 전달되고 **ShiftTab** 키를 눌러 옵션을 뒤로 이동합니다.
- **Enter** 를 사용하여 옵션을 선택합니다.
- 스페이스 표시줄을 사용하여 확인란의 상태를 전환합니다.

4.2. NMTUI를 사용하여 연결 프로필 추가

nmtui 애플리케이션은 **NetworkManager**에 텍스트 사용자 인터페이스를 제공합니다. 다음 절차에서는 새 연결 프로필을 추가하는 방법을 설명합니다.

사전 요구 사항

- **NetworkManager-tui** 패키지가 설치됩니다.

절차

1. **NetworkManager** 텍스트 사용자 인터페이스 유틸리티를 시작합니다.

```
# nmtui
```
2. **Edit a connection** 메뉴 항목을 선택하고 **Enter** 키를 누릅니다.
3. **Add(추가)** 단추를 선택하고 **Enter** 키를 누릅니다.
4. **이더넷** 을 선택하고 **Enter** 를 누릅니다.
5. 필드에 연결 세부 정보를 입력합니다.

Edit Connection

Profile name

enpls0

Device

enpls0 (52:54:00:DF:55:D1)

= ETHERNET
<Show>

= IPv4 CONFIGURATION
<Manual>
<Hide>

Addresses

192.0.2.1/24

<Remove>

<Add...>

Gateway

192.0.2.254

DNS servers

192.0.2.254

<Remove>

<Add...>

Search domains <Add...>

Routing (No custom routes) <Edit...>

[] Never use this network for default route

[] Ignore automatically obtained routes

[] Ignore automatically obtained DNS parameters

[] Require IPv4 addressing for this connection

= IPv6 CONFIGURATION
<Manual>
<Hide>

Addresses

2001:db8:1::1/64

<Remove>

<Add...>

Gateway

2001:db8:1::fffe

DNS servers

2001:db8:1::fffe

<Remove>

<Add...>

Search domains <Add...>

Routing (No custom routes) <Edit...>

[] Never use this network for default route

[] Ignore automatically obtained routes

[] Ignore automatically obtained DNS parameters

[] Require IPv6 addressing for this connection

[X] Automatically connect

[X] Available to all users

<Cancel>
<OK>

6. **OK(확인)** 를 선택하여 변경 사항을 저장합니다.
7. **Back (뒤로)**을 선택하여 메인 메뉴로 돌아갑니다.
8. **connecting a connection** 을 선택하고 **Enter** 키를 누릅니다.
9. 새 연결 항목을 선택하고 **Enter** 키를 눌러 연결을 활성화합니다.

10. **Back**(뒤로)을 선택하여 주 메뉴로 돌아갑니다.
11. **Quit** 을 선택합니다.

검증 단계

1. 장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp1s0    ethernet connected Example-Connection
```

2. 연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:     b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:     802-3-ethernet
connection.interface-name: enp1s0
...
```

디스크의 구성이 장치의 구성과 일치하지 않는 경우 **NetworkManager**를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 **NetworkManager** 서비스를 다시 시작한 후 **NetworkManager**에서 연결을 복제합니다.

추가 리소스

- [기본 네트워크 설정 테스트](#)
- [nmtui\(1\) 도움말 페이지](#)

4.3. NMTUI를 사용하여 수정된 연결에 변경 사항 적용

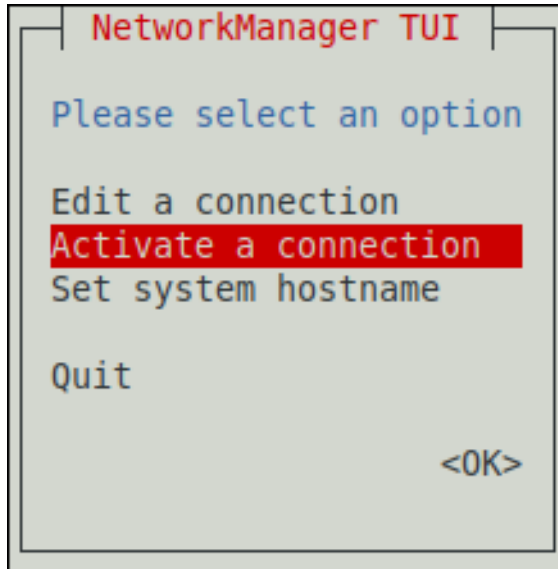
nmtui 연결을 수정한 후 연결을 다시 활성화해야 합니다. **nmtui** 의 연결을 반응하면 일시적으로 연결을 비활성화합니다.

사전 요구 사항

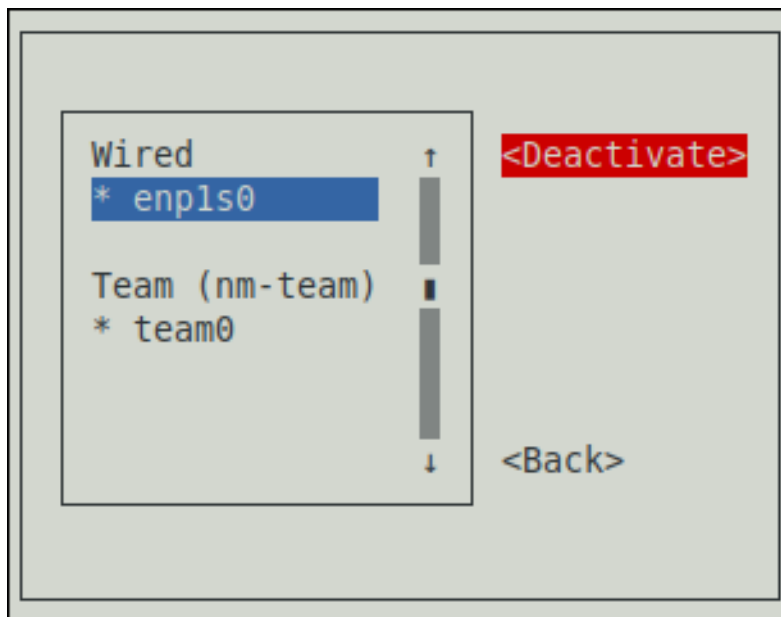
- 연결 프로필에 자동 연결 설정이 활성화되지 않았습니다.

절차

1. 메인 메뉴에서 **upgrade a connection** 메뉴 항목을 선택합니다.



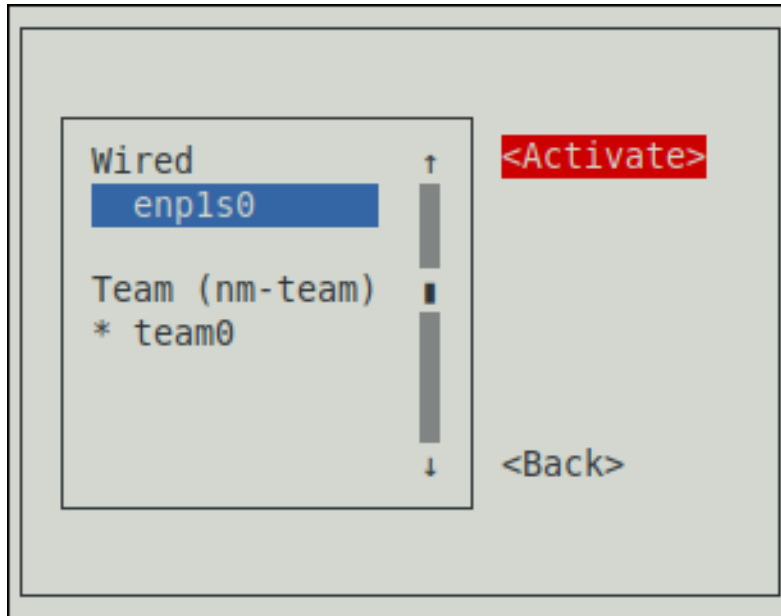
2. 수정된 연결을 선택합니다.
3. 오른쪽에서 비활성화 버튼을 선택하고 **Enter** 키를 누릅니다.



4. 연결을 다시 선택합니다.

5.

오른쪽에서 **Vault** 버튼을 선택하고 **Enter** 키를 누릅니다.



5장. NMCLI 시작하기

이 섹션에서는 **nmcli** 유틸리티에 대한 일반 정보를 설명합니다.

5.1. NMCLI의 다양한 출력 형식

nmcli 유틸리티는 **nmcli** 명령의 출력을 수정하기 위해 다양한 옵션을 지원합니다. 이러한 옵션을 사용하면 필요한 정보만 표시할 수 있습니다. 이렇게 하면 스크립트의 출력 처리를 간소화합니다.

기본적으로 **nmcli** 유틸리티는 표와 같은 형식으로 출력을 표시합니다.

```
# nmcli device
DEVICE TYPE    STATE    CONNECTION
enp1s0 ethernet connected enp1s0
lo    loopback unmanaged --
```

f 옵션을 사용하면 특정 열을 사용자 지정 순서로 표시할 수 있습니다. 예를 들어 **DEVICE** 및 **STATE** 열만 표시하려면 다음을 입력합니다.

```
# nmcli -f DEVICE,STATE device
DEVICE STATE
enp1s0 connected
lo    unmanaged
```

t 옵션을 사용하면 콜론으로 구분된 형식으로 출력의 개별 필드를 표시할 수 있습니다.

```
# nmcli -t device
enp1s0:ethernet:connected:enp1s0
lo:loopback:unmanaged:
```

스크립트의 출력을 처리할 때 **-f** 및 **-t** 를 결합하여 콜론으로 구분된 형식으로 특정 필드만 표시할 수 있습니다.

```
# nmcli -f DEVICE,STATE -t device
enp1s0:connected
lo:unmanaged
```

5.2. NMCLI에서 탭 완료 사용

bash-completion 패키지가 호스트에 설치된 경우 **nmcli** 유틸리티는 탭 완료를 지원합니다. 이를 통해 옵션 이름을 자동 완성하고 가능한 옵션과 값을 식별할 수 있습니다.

예를 들어 **nmcli con** 를 입력하고 **Tab** 을 누른 경우 셸은 **nmcli** 연결 로 명령을 자동으로 완료합니다.

완료의 경우 입력한 옵션 또는 값이 고유해야 합니다. 고유하지 않은 경우 **nmcli** 는 모든 가능성을 표시합니다. 예를 들어 **nmcli connection d** 를 입력하고 **Tab** 을 누른 경우 명령은 가능한 옵션으로 **command delete** 및 **down** 명령을 표시합니다.

탭 완료를 사용하여 연결 프로필에 설정할 수 있는 모든 속성을 표시할 수도 있습니다. 예를 들어 **nmcli connection modify connection_name** 을 입력하고 **Tab** 을 눌러 사용 가능한 속성의 전체 목록을 표시합니다.

5.3. 자주 사용되는 NMCLI 명령

다음은 자주 사용되는 **nmcli** 명령에 대한 개요입니다.

-

연결 프로파일 목록을 표시하려면 다음을 입력합니다.

```
# nmcli connection show
NAME UUID TYPE DEVICE
enp1s0 45224a39-606f-4bf7-b3dc-d088236c15ee ethernet enp1s0
```

-

특정 연결 프로필의 설정을 표시하려면 다음을 입력합니다.

```
# nmcli connection show connection_name
connection.id: enp1s0
connection.uuid: 45224a39-606f-4bf7-b3dc-d088236c15ee
connection.stable-id: --
connection.type: 802-3-ethernet
...
```

-

연결 속성을 수정하려면 다음을 입력합니다.

```
# nmcli connection modify connection_name property value
```

여러 속성 값조합을 명령에 전달하는 경우 단일 명령을 사용하여 여러속성 을 수정할 수 있습니다.

- 네트워크 장치 목록, 해당 상태 및 장치를 사용하는 연결 프로필을 표시하려면 다음을 입력합니다.

```
# nmcli device
DEVICE TYPE    STATE    CONNECTION
enp1s0 ethernet connected enp1s0
enp8s0 ethernet disconnected --
enp7s0 ethernet unmanaged --
...
```

- 연결을 활성화하려면 다음을 입력합니다.

```
# nmcli connection up connection_name
```

- 연결을 비활성화하려면 다음을 입력합니다.

```
# nmcli connection down connection_name
```

6장. GNOME GUI를 사용하여 네트워킹 설정 시작하기

GNOME에서 다음과 같은 방법으로 네트워크 연결을 관리하고 구성할 수 있습니다.

- 데스크탑의 오른쪽 상단에 있는 **GNOME Shell** 네트워크 연결 아이콘
- **GNOME control-center** 응용 프로그램
- **GNOME nm-connection-editor** 응용 프로그램

6.1. GNOME SHELL 네트워크 연결 아이콘을 사용하여 네트워크에 연결

GNOME GUI를 사용하는 경우 **GNOME Shell** 네트워크 연결 아이콘을 사용하여 네트워크에 연결할 수 있습니다.

사전 요구 사항

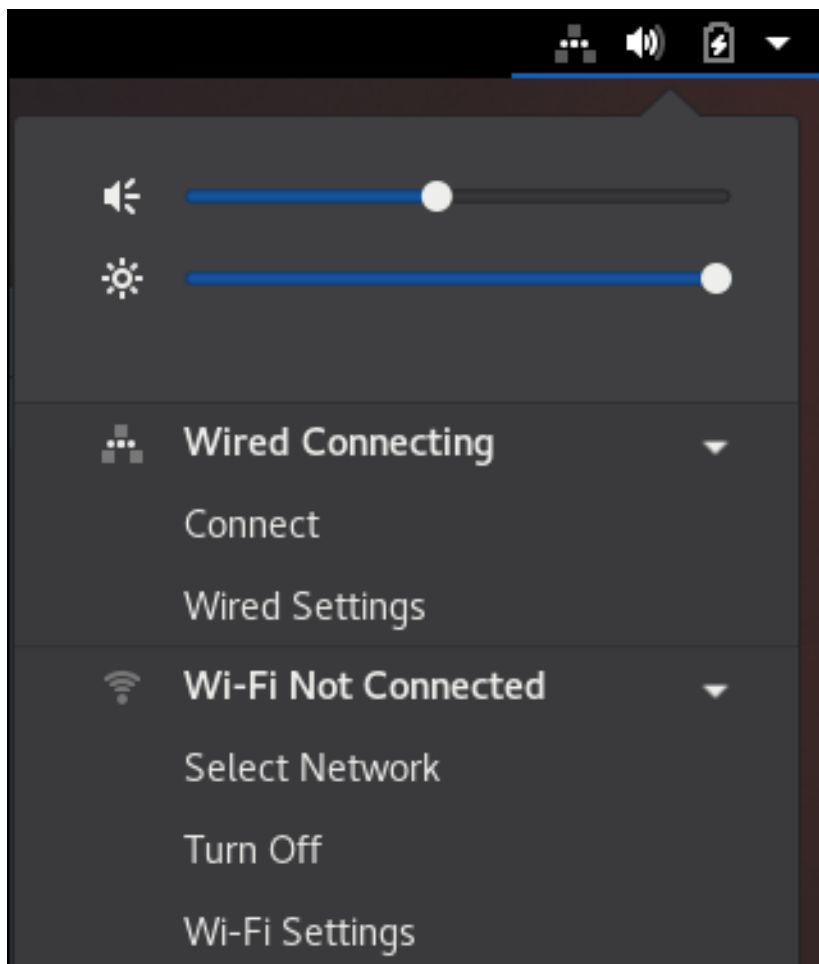
- **GNOME** 패키지 그룹이 설치되어 있습니다.
- **GNOME**에 로그인했습니다.
- 네트워크에 고정 IP 주소 또는 802.1x 구성과 같은 특정 구성이 필요한 경우 연결 프로필이 이미 생성되었습니다.

절차

1. 데스크탑의 오른쪽 상단에 있는 네트워크 연결 아이콘을 클릭합니다.



2. 연결 유형에 따라 **Wired** 또는 **Wi-Fi** 항목을 선택합니다.



- 유선 연결의 경우 네트워크에 연결할 연결을 선택합니다.
- **Wi-Fi** 연결의 경우 네트워크 선택을 클릭하고 연결할 네트워크를 선택한 후 암호를 입력합니다.

7장. NMSTATE 소개

NMState는 선언적 네트워크 관리자 API입니다. **nmstate** 패키지는 **RHEL**에서 **NetworkManager**를 관리하기 위해 **libnmstate Python** 라이브러리와 명령줄 유틸리티인 **nmstatectl** 을 제공합니다. **Nmstate**를 사용하는 경우 **YAML** 또는 **JSON** 형식의 지침을 사용하여 예상 네트워킹 상태를 설명합니다.

Nmstate를 사용하면 많은 이점이 있습니다. 예를 들면 다음과 같습니다.

- **RHEL** 네트워크 기능 관리를 위한 안정적이고 확장 가능한 인터페이스 제공
- 호스트 및 클러스터 수준에서 원자성 및 트랜잭션 작업 지원
- 대부분의 속성의 부분 편집을 지원하고 지침에 지정되지 않은 기존 설정을 유지합니다.
- 관리자가 자체 플러그인을 사용할 수 있도록 플러그인 지원 제공

7.1. PYTHON 애플리케이션에서 LIBNMSTATE 라이브러리 사용

libnmstate Python 라이브러리를 사용하면 개발자가 자체 애플리케이션에서 **Nmstate**를 사용할 수 있습니다.

라이브러리를 사용하려면 소스 코드로 가져옵니다.

```
import libnmstate
```

이 라이브러리를 사용하려면 **nmstate** 패키지를 설치해야 합니다.

예 7.1. libnmstate 라이브러리를 사용하여 네트워크 상태 쿼리

다음 **Python** 코드는 **libnmstate** 라이브러리를 가져오고 사용 가능한 네트워크 인터페이스 및 해당 상태를 표시합니다.

```
import json
import libnmstate
```

```
from libnmstate.schema import Interface

net_state = libnmstate.show()
for iface_state in net_state[Interface.KEY]:
    print(iface_state[Interface.NAME] + ": "
          + iface_state[Interface.STATE])
```

7.2. NMSTATECTL을 사용하여 현재 네트워크 구성 업데이트

nmstatectl 유틸리티를 사용하여 파일에 하나 또는 모든 인터페이스의 현재 네트워크 구성을 저장할 수 있습니다. 그런 다음 이 파일을 사용하여 다음을 수행할 수 있습니다.

- 구성을 수정하고 동일한 시스템에 적용합니다.
- 파일을 다른 호스트에 복사하고 동일한 또는 수정된 설정을 사용하여 호스트를 구성합니다.

이 절차에서는 **enp1s0** 인터페이스의 설정을 파일에 내보내고 구성을 수정한 다음 호스트에 설정을 적용하는 방법을 설명합니다.

사전 요구 사항

- **nmstate** 패키지가 설치되어 있습니다.

절차

1. **enp1s0** 인터페이스의 설정을 **~/network-config.yml** 파일로 내보냅니다.

```
# nmstatectl show enp1s0 > ~/network-config.yml
```

이 명령은 **enp1s0**의 구성을 **YAML** 형식으로 저장합니다. 출력을 **JSON** 형식으로 저장하려면 **--json** 옵션을 명령에 전달합니다.

인터페이스 이름을 지정하지 않으면 **nmstatectl**에서 모든 인터페이스의 구성을 내보냅니다.

2. 텍스트 편집기를 사용하여 **~/network-config.yml** 파일을 수정하여 구성을 업데이트합니다.

3. *~/network-config.yml* 파일의 설정을 적용합니다.

```
# nmstatectl apply ~/network-config.yml
```

설정을 **JSON** 형식으로 내보낸 경우 명령에 **--json** 옵션을 전달합니다.

7.3. 추가 리소스

- */usr/share/doc/nmstate/README.md*
- */usr/share/doc/nmstate/examples/*

8장. 이더넷 연결 구성

이 섹션에서는 정적 및 동적 IP 주소를 사용하여 이더넷 연결을 구성하는 방법에 대해 설명합니다.

8.1. NMCLI를 사용하여 정적 이더넷 연결 구성

다음 절차에서는 **nmcli** 유틸리티를 사용하여 다음 설정으로 이더넷 연결을 추가하는 방법을 설명합니다.

- 정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 192.0.2.254
- IPv6 기본 게이트웨이 - 2001:db8:1::fffe
- IPv4 DNS 서버 - 192.0.2.200
- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com

절차

1. 이더넷 연결에 대한 새 **NetworkManager** 연결 프로필을 추가합니다.

```
# nmcli connection add con-name Example-Connection ifname enp7s0 type ethernet
```

추가 단계는 생성한 **Example-Connection** 연결 프로필을 수정합니다.

2.

IPv4 주소를 설정합니다.

```
# nmcli connection modify Example-Connection ipv4.addresses 192.0.2.1/24
```

3.

IPv6 주소를 설정합니다.

```
# nmcli connection modify Example-Connection ipv6.addresses 2001:db8:1::1/64
```

4.

IPv4 및 IPv6 연결 방법을 *manual* 로 설정합니다.

```
# nmcli connection modify Example-Connection ipv4.method manual
# nmcli connection modify Example-Connection ipv6.method manual
```

5.

IPv4 및 IPv6 기본 게이트웨이를 설정합니다.

```
# nmcli connection modify Example-Connection ipv4.gateway 192.0.2.254
# nmcli connection modify Example-Connection ipv6.gateway 2001:db8:1::fffe
```

6.

IPv4 및 IPv6 DNS 서버 주소를 설정합니다.

```
# nmcli connection modify Example-Connection ipv4.dns "192.0.2.200"
# nmcli connection modify Example-Connection ipv6.dns "2001:db8:1::ffbb"
```

여러 DNS 서버를 설정하려면 공백으로 구분되어 따옴표로 묶으십시오.

7.

IPv4 및 IPv6 연결에 대한 DNS 검색 도메인을 설정합니다.

```
# nmcli connection modify Example-Connection ipv4.dns-search example.com
# nmcli connection modify Example-Connection ipv6.dns-search example.com
```

8.

연결 프로필을 활성화합니다.

```
# nmcli connection up Example-Connection
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/13)
```

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp7s0    ethernet connected Example-Connection
```

2.

연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...
```

3.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:

```
# ping 2001:db8:1::2
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

•

원격 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

○

명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::fff3
```

4.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

문제 해결 단계

1.

연결에 실패하거나 네트워크 인터페이스가 **up** 및 **down** 상태 간에 전환되는 경우:

•

네트워크 케이블이 호스트 및 스위치에 플러그인되어 있는지 확인합니다.

•

링크 실패가 이 호스트에만 있는지 또는 서버가 연결된 동일한 스위치에 연결된 다른 호스트에 있는지 확인합니다.

•

네트워크 케이블과 네트워크 인터페이스가 예상대로 작동하는지 확인합니다. 하드웨어 진단 단계를 수행하고 결함 케이블 및 네트워크 인터페이스 카드를 교체합니다.

•

디스크의 구성이 장치의 구성과 일치하지 않는 경우 **NetworkManager**를 시작하거나 다

시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 **NetworkManager** 서비스를 다시 시작한 후 **NetworkManager**에서 연결을 복제합니다.

추가 리소스

- **`nm-settings(5)`, `nmcli` 및 `nmcli(1)` 도움말 페이지**
- **기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 **NetworkManager** 구성**

8.2. NMCLI 대화형 편집기를 사용하여 정적 이더넷 연결 구성

다음 절차에서는 **nmcli** 대화형 모드를 사용하여 다음 설정으로 이더넷 연결을 추가하는 방법을 설명합니다.

- **정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크**
- **정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크**
- **IPv4 기본 게이트웨이 - 192.0.2.254**
- **IPv6 기본 게이트웨이 - 2001:db8:1::fffe**
- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - `example.com`**

절차

1. 이더넷 연결에 대한 새 **NetworkManager** 연결 프로필을 추가하고 대화형 모드를 시작하려면 다음을 입력합니다.

```
# nmcli connection edit type ethernet con-name Example-Connection
```

2.

네트워크 인터페이스를 설정합니다.

```
nmcli> set connection.interface-name enp7s0
```

3.

IPv4 주소를 설정합니다.

```
nmcli> set ipv4.addresses 192.0.2.1/24
```

4.

IPv6 주소를 설정합니다.

```
nmcli> set ipv6.addresses 2001:db8:1::1/64
```

5.

IPv4 및 IPv6 연결 방법을 *manual* 로 설정합니다.

```
nmcli> set ipv4.method manual
nmcli> set ipv6.method manual
```

6.

IPv4 및 IPv6 기본 게이트웨이를 설정합니다.

```
nmcli> set ipv4.gateway 192.0.2.254
nmcli> set ipv6.gateway 2001:db8:1::fffe
```

7.

IPv4 및 IPv6 DNS 서버 주소를 설정합니다.

```
nmcli> set ipv4.dns 192.0.2.200
nmcli> set ipv6.dns 2001:db8:1::ffbb
```

여러 DNS 서버를 설정하려면 공백으로 구분되어 따옴표로 묶으십시오.

8.

IPv4 및 IPv6 연결에 대한 DNS 검색 도메인을 설정합니다.

```
nmcli> set ipv4.dns-search example.com
nmcli> set ipv6.dns-search example.com
```

9.

연결을 저장하고 활성화합니다.

```
nmcli> save persistent
Saving the connection with 'autoconnect=yes'. That might result in an immediate
activation of the connection.
Do you still want to save? (yes/no) [yes] yes
```

10.

대화형 모드를 종료합니다.

```
nmcli> quit
```

검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp7s0    ethernet connected Example-Connection
```

2.

연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id:  --
connection.type:     802-3-ethernet
connection.interface-name: enp7s0
...
```

3.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 IP 주소를 **ping** 합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:


```
# ping 2001:db8:1::2
```

명령이 실패하면 **IP** 및 서브넷 설정을 확인합니다.

-

원격 서브넷의 **IP** 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

-

명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::fff3
```

4.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

문제 해결 단계

1.

연결에 실패하거나 네트워크 인터페이스가 **up** 및 **down** 상태 간에 전환되는 경우:

- 네트워크 케이블이 호스트 및 스위치에 플러그인되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 서버가 연결된 동일한 스위치에 연결된 다른 호스트에 있는지 확인합니다.
- 네트워크 케이블과 네트워크 인터페이스가 예상대로 작동하는지 확인합니다. 하드웨어 진단 단계를 수행하고 결함 케이블 및 네트워크 인터페이스 카드를 교체합니다.

디스크의 구성이 장치의 구성과 일치하지 않는 경우 **NetworkManager**를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 **NetworkManager** 서비스를 다시 시작한 후 **NetworkManager**에서 연결을 복제합니다.

추가 리소스

- **nm-settings(5)** 도움말 페이지
- **nmcli(1)** 도움말 페이지
- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 **NetworkManager** 구성

8.3. NMSTATECTL을 사용하여 정적 이더넷 연결 구성

이 절차에서는 **nmstatectl** 유틸리티를 사용하여 다음 설정으로 **enp7s0** 장치에 대한 이더넷 연결을 구성하는 방법을 설명합니다.

- 정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크
- /64 서브넷 마스크가 포함된 정적 IPv6 주소 - 2001:db8:1::1
- IPv4 기본 게이트웨이 - 192.0.2.254
- IPv6 기본 게이트웨이 - 2001:db8:1::fffe

- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**

nmstatectl 유틸리티를 사용하면 구성을 설정한 후 구성 파일과 결과가 일치합니다. 문제가 발생하면 **nmstatectl** 에서 변경 사항을 자동으로 롤백하여 시스템을 잘못된 상태로 유지하지 않습니다.

절차는 **YAML** 형식으로 인터페이스 구성을 정의합니다. 또는 **JSON** 형식으로 구성을 지정할 수도 있습니다.

사전 요구 사항

- **nmstate** 패키지가 설치되어 있습니다.

절차

1. 다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/create-ethernet-profile.yml**)을 만듭니다.

```
---
interfaces:
- name: enp7s0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    address:
    - ip: 192.0.2.1
      prefix-length: 24
    dhcp: false
  ipv6:
    enabled: true
    address:
    - ip: 2001:db8:1::1
      prefix-length: 64
    autoconf: false
    dhcp: false
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-address: 192.0.2.254
```

```

    next-hop-interface: enp7s0
  - destination: ::/0
    next-hop-address: 2001:db8:1::fffe
    next-hop-interface: enp7s0
dns-resolver:
  config:
    search:
      - example.com
    server:
      - 192.0.2.200
      - 2001:db8:1::ffbb

```

2.

시스템에 설정을 적용합니다.

```
# nmstatectl apply ~/create-ethernet-profile.yml
```

검증 단계

1.

장치 및 연결 상태를 표시합니다.

```

# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp7s0    ethernet connected enp7s0

```

2.

연결 프로파일의 모든 설정을 표시합니다.

```

# nmcli connection show enp7s0
connection.id:      enp7s0
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id:  --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...

```

3.

연결 설정을 **YAML** 형식으로 표시합니다.

```
# nmstatectl show enp7s0
```

추가 리소스

•

[nmstatectl\(8\) 도움말 페이지](#)

- `/usr/share/doc/nmstate/examples/`

8.4. 인터페이스 이름으로 **RHEL** 시스템 역할을 사용하여 정적 이더넷 연결 구성

다음 절차에서는 **Ansible** 플레이북을 실행하여 다음 설정으로 **enp7s0** 인터페이스에 대한 이더넷 연결을 원격으로 추가하는 방법을 설명합니다.

- 정적 **IPv4** 주소 - **192.0.2.1** 에 **/24** 서브넷 마스크
- 정적 **IPv6** 주소 - **2001:db8:1::1** 에 **/64** 서브넷 마스크
- **IPv4** 기본 게이트웨이 - **192.0.2.254**
- **IPv6** 기본 게이트웨이 - **2001:db8:1::fffe**
- **IPv4** **DNS** 서버 - **192.0.2.200**
- **IPv6** **DNS** 서버 - **2001:db8:1::ffbb**
- **DNS** 검색 도메인 - **example.com**

Ansible 제어 노드에서 이 절차를 실행합니다.

사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

•

호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1.

플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2.

다음 콘텐츠를 사용하여 **~/ethernet-static-IP.yml** 플레이북을 생성합니다.

```
---
- name: Configure an Ethernet connection with static IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 192.0.2.1/24
            - 2001:db8:1::1/64
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        state: up
```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-static-IP.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)
- [ansible-playbook\(1\) 도움말 페이지](#)

8.5. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 특정 장치 경로와 일치하는 장치의 고정 IP 주소와 함께 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다.

다음 명령을 사용하여 장치 경로를 식별할 수 있습니다.

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

이 절차에서는 **PCI ID 0000:00:0[1-3].0** 표현식과 일치하는 장치로 다음 설정을 설정하지만 **0000:00:02.0** 은 아닙니다.

- 정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 192.0.2.254

- **IPv6 기본 게이트웨이 - 2001:db8:1::fffe**
- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**

Ansible 제어 노드에서 이 절차를 실행합니다.

사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-dynamic-IP.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network
```



```
vars:
  network_connections:
    - name: example
      match:
        path:
          - pci-0000:00:0[1-3].0
          - &!pci-0000:00:02.0
        type: ethernet
        autoconnect: yes
      ip:
        address:
          - 192.0.2.1/24
          - 2001:db8:1::1/64
        gateway4: 192.0.2.254
        gateway6: 2001:db8:1::fffe
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
      state: up
```

이 예제의 **match** 매개 변수는 **Ansible**이 **PCI ID 0000:00:0[1-3].0** 과 일치하는 장치에 플레이를 적용하지만 **0000:00:00:02.0**이 아닌 장치에 플레이를 적용하도록 정의합니다. 사용할 수 있는 특수 수정자 및 와일드카드에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 파일에서 일치하는 매개 변수 설명을 참조하십시오.

3.

플레이북을 실행합니다.

-

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

- **`/usr/share/ansible/roles/rhel-system-roles.network/README.md` file**
- **`ansible-playbook(1)` 도움말 페이지**

8.6. NMCLI를 사용하여 동적 이더넷 연결 구성

이 절차에서는 `nmcli` 유틸리티를 사용하여 동적 이더넷 연결을 추가하는 방법을 설명합니다. 이 설정을 사용하면 **NetworkManager**에서 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다.

사전 요구 사항

- **DHCP 서버는 네트워크에서 사용할 수 있습니다.**

절차

1. 이더넷 연결에 대한 새 **NetworkManager** 연결 프로필을 추가합니다.

```
# nmcli connection add con-name Example-Connection ifname enp7s0 type ethernet
```

2. 필요한 경우 **Example-Connection** 프로필을 사용할 때 호스트 이름 **NetworkManager**가 **DHCP** 서버로 보냅니다.

```
# nmcli connection modify Example-Connection ipv4.dhcp-hostname Example
ipv6.dhcp-hostname Example
```

3. 필요한 경우 **Example-Connection** 프로필을 사용할 때 클라이언트 ID **NetworkManager**가 **IPv4 DHCP** 서버로 보냅니다.

```
# nmcli connection modify Example-Connection ipv4.dhcp-client-id client-ID
```

IPv6에 대해 `dhcp-client-id` 매개 변수가 없습니다. **IPv6**에 대한 식별자를 생성하려면 `dhclient` 서비스를 구성합니다.

검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE   TYPE    STATE    CONNECTION
enp7s0   ethernet connected Example-Connection
```

2.

연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...
```

3.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:

```
# ping 2001:db8:1::2
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

•

원격 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

○

명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::ff3
```

4.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

추가 리소스

- **dhclient(8)** 도움말 페이지
- **nm-settings(5)**
- **nmcli(1)** 도움말 페이지
- [NetworkManager 서비스를 다시 시작한 후 NetworkManager에서 연결을 중복합니다.](#)

8.7. NMCLI 대화형 편집기를 사용하여 동적 이더넷 연결 구성

이 절차에서는 **nmcli** 유틸리티의 대화형 편집기를 사용하여 동적 이더넷 연결을 추가하는 방법을 설명합니다. 이 설정을 사용하면 **NetworkManager**에서 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다.

사전 요구 사항

- **DHCP** 서버는 네트워크에서 사용할 수 있습니다.

절차

1. 이더넷 연결에 대한 새 **NetworkManager** 연결 프로필을 추가하고 대화형 모드를 시작하려면 다음을 입력합니다.

```
# nmcli connection edit type ethernet con-name Example-Connection
```

2. 네트워크 인터페이스를 설정합니다.

```
nmcli> set connection.interface-name enp7s0
```

3. 필요한 경우 **Example-Connection** 프로필을 사용할 때 호스트 이름 **NetworkManager**가 **DHCP** 서버로 보냅니다.

```
nmcli> set ipv4.dhcp-hostname Example
nmcli> set ipv6.dhcp-hostname Example
```

4. 필요한 경우 **Example-Connection** 프로필을 사용할 때 클라이언트 ID **NetworkManager**가 **IPv4 DHCP** 서버로 보냅니다.

```
nmcli> set ipv4.dhcp-client-id client-ID
```

IPv6에 대해 **dhcp-client-id** 매개 변수가 없습니다. **IPv6**에 대한 식별자를 생성하려면 **dhclient** 서비스를 구성합니다.

5. 연결을 저장하고 활성화합니다.

```
nmcli> save persistent
Saving the connection with 'autoconnect=yes'. That might result in an immediate
activation of the connection.
Do you still want to save? (yes/no) [yes] yes
```

6.

대화형 모드를 종료합니다.

```
nmcli> quit
```

검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp7s0    ethernet connected Example-Connection
```

2.

연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...
```

3.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:

```
# ping 2001:db8:1::2
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

•

원격 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

○

명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::ff3
```

4.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

추가 리소스

- **dhclient(8)** 도움말 페이지
- **nm-settings(5)**
- **nmcli(1)** 도움말 페이지

•

NetworkManager 서비스를 다시 시작한 후 **NetworkManager**에서 연결을 중복합니다.

8.8. NMSTATECTL을 사용하여 동적 이더넷 연결 구성

이 절차에서는 **nmstatectl** 유틸리티를 사용하여 **enp7s0** 장치에 동적 이더넷을 추가하는 방법을 설명합니다. 이 절차의 설정을 통해 **NetworkManager**는 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다.

nmstatectl 유틸리티를 사용하면 구성을 설정한 후 구성 파일과 결과가 일치합니다. 문제가 발생하면 **nmstatectl** 에서 변경 사항을 자동으로 롤백하여 시스템을 잘못된 상태로 유지하지 않습니다.

절차는 **YAML** 형식으로 인터페이스 구성을 정의합니다. 또는 **JSON** 형식으로 구성을 지정할 수도 있습니다.

사전 요구 사항

•

nmstate 패키지가 설치되어 있습니다.

절차

1.

다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/create-ethernet-profile.yml**)을 만듭니다.

```
---
interfaces:
- name: enp7s0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    auto-dns: true
    auto-gateway: true
    auto-routes: true
    dhcp: true
  ipv6:
    enabled: true
    auto-dns: true
    auto-gateway: true
    auto-routes: true
    autoconf: true
    dhcp: true
```


2. 시스템에 설정을 적용합니다.

```
# nmstatectl apply ~/create-ethernet-profile.yml
```

검증 단계

1. 장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE  TYPE  STATE  CONNECTION
enp7s0  ethernet connected enp7s0
```

2. 연결 프로파일의 모든 설정을 표시합니다.

```
# nmcli connection show enp7s0
connection.id:      enp7s0_
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...
```

3. 연결 설정을 **YAML** 형식으로 표시합니다.

```
# nmstatectl show enp7s0
```

추가 리소스

- [nmstatectl\(8\) 도움말 페이지](#)
- [/usr/share/doc/nmstate/examples/](#)

8.9. 인터페이스 이름으로 **RHEL** 시스템 역할을 사용하여 동적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 **enp7s0** 인터페이스에 대해 동적 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다. 이 설정을 사용하면 네트워크 연결이 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다. **Ansible** 제어 노드에서 이 절차를 실행합니다.

사전 요구 사항

- **DHCP** 서버는 네트워크에서 사용할 수 있습니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-dynamic-IP.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          dhcp4: yes
          auto6: yes
          state: up
```

3. 플레이북을 실행합니다.

- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file
- [ansible-playbook\(1\)](#) 도움말 페이지

8.10. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 특정 장치 경로와 일치하는 장치에 대한 동적 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다. 동적 IP 설정을 사용하면 네트워크 연결이 **DHCP** 서버에서 이 연결에 대한 IP 설정을 요청합니다. **Ansible** 제어 노드에서 이 절차를 실행합니다.

다음 명령을 사용하여 장치 경로를 식별할 수 있습니다.

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

사전 요구 사항

- **DHCP** 서버는 네트워크에서 사용할 수 있습니다.

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-dynamic-IP.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: example
        match:
          path:
            - pci-0000:00:0[1-3].0
            - &!pci-0000:00:02.0
          type: ethernet
          autoconnect: yes
          ip:
            dhcp4: yes
            auto6: yes
            state: up
```

이 예제의 **match** 매개 변수는 Ansible이 **PCI ID 0000:00:0[1-3].0** 과 일치하는 장치에 플레이를 적용하지만 **0000:00:00:02.0**이 아닌 장치에 플레이를 적용하도록 정의합니다. 사용할 수 있는 특수 수정자 및 와일드카드에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 파일에서 일치하는 매개변수 설명을 참조하십시오.

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

•

/usr/share/ansible/roles/rhel-system-roles.network/README.md file

•

ansible-playbook(1) 도움말 페이지

8.11. CONTROL-CENTER를 사용하여 이더넷 연결 구성

이더넷 연결은 실제 또는 가상 시스템에서 가장 자주 사용되는 연결 유형입니다. 이 섹션에서는 **GNOME control-center** 에서 이 연결 유형을 구성하는 방법을 설명합니다.

control-center 는 **nm-connection-editor** 애플리케이션 또는 **nmcli** 유틸리티만큼 많은 구성 옵션을 지원하지 않습니다.

사전 요구 사항

•

서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.

- **GNOME이 설치되어 있습니다.**

절차

1. 슈퍼 키를 누르고 설정 을 입력한 후 **Enter** 키를 누릅니다.
2. 왼쪽 탐색에서 **Network** 를 선택합니다.
3. **Wired** 항목 옆에 있는 **+** 버튼을 클릭하여 새 프로필을 생성합니다.
4. 선택 사항: **Identity (ID)** 탭에서 연결의 이름을 설정합니다.
5. **IPv4** 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 수동 방법을 선택하고 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

The screenshot shows the 'New Profile' window with the following configuration:

- Identity** tab: (Empty)
- IPv4** tab:
 - IPv4 Method:** ☒ Automatic (DHCP), ☐ Link-Local Only, ☒ Manual, ☐ Disable
 - Addresses:**

Address	Netmask	Gateway
192.0.2.1	24	192.0.2.254
 - DNS:** Automatic ☒ ON. Text box: 192.0.2.1
- IPv6** tab: (Empty)
- Security** tab: (Empty)

6. **IPv6** 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 수동 방법을 선택하고 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

New Profile

Cancel Add

Identity IPv4 **IPv6** Security

IPv6 Method

☐ Automatic
 ☐ Automatic, DHCP only
☐ Link-Local Only
 ☒ Manual
☐ Disable

Addresses

Address	Prefix	Gateway	
2001:db8:1::1	64	2001:db8:1::fff3	✕
			✕

DNS Automatic **ON**

2001:db8:1::fffd

7. **Add(추가)** 버튼을 클릭하여 연결을 저장합니다. **GNOME control-center** 는 연결을 자동으로 활성화합니다.

검증 단계

1. 장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE   TYPE    STATE    CONNECTION
enp7s0   ethernet connected Example-Connection
```

2. 연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show Example-Connection
connection.id:      Example-Connection
connection.uuid:    b6cdfa1c-e4ad-46e5-af8b-a75f06b79f76
connection.stable-id: --
connection.type:    802-3-ethernet
connection.interface-name: enp7s0
...
```

3.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:

```
# ping 2001:db8:1::2
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

•

원격 서브넷의 IP 주소를 **ping**합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

○

명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::fffe
```

4.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

문제 해결 단계

1.

연결에 실패하거나 네트워크 인터페이스가 **up** 및 **down** 상태 간에 전환되는 경우:

- 네트워크 케이블이 호스트 및 스위치에 플러그인되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 서버가 연결된 동일한 스위치에 연결된 다른 호스트에 있는지 확인합니다.
- 네트워크 케이블과 네트워크 인터페이스가 예상대로 작동하는지 확인합니다. 하드웨어 진단 단계를 수행하고 결함 케이블 및 네트워크 인터페이스 카드를 교체합니다.

추가 리소스

- 연결에 기본 게이트웨이가 없는 경우 기본 게이트웨이를 **제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager** 구성을 참조하십시오.

8.12. NM-CONNECTION-EDITOR를 사용하여 이더넷 연결 구성

이더넷 연결은 실제 또는 가상 서버에서 가장 자주 사용되는 연결 유형입니다. 이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 이 연결 유형을 구성하는 방법을 설명합니다.

사전 요구 사항

- 서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.
- **GNOME**이 설치되어 있습니다.

절차

1. 터미널을 열고 다음을 입력합니다.


```
$ nm-connection-editor
```
2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. 이더넷 연결 유형을 선택하고 생성을 클릭합니다.
4. 일반 탭에서 다음을 수행합니다.
 - a. 시스템이 부팅되거나 **NetworkManager** 서비스를 다시 시작할 때 이 연결을 자동으로 활성화하려면 다음을 수행합니다.
 - i. 우선 순위가 지정된 연결을 자동으로 선택합니다.
 - ii. 선택 사항: 우선순위를 사용하여 자동으로 **Connect** 옆에 있는 우선순위 값을 변경합니다.

동일한 장치에 여러 연결 프로필이 있는 경우 **NetworkManager**는 하나의 프로필만 활성화합니다. 기본적으로 **NetworkManager**는 자동 연결이 활성화된 마지막으로 사용되는 프로필을 활성화합니다. 그러나 프로필에 우선 순위 값을 설정하면 **NetworkManager**가 우선 순위가 가장 높은 프로필을 활성화합니다.
 - b. 연결 프로필을 생성한 사용자만 프로필을 사용할 수 있어야 하는 경우 모든 사용자가 이 네트워크에 연결할 수 있는 확인란을 선택합니다.

The screenshot shows the 'Editing Example connection 1' window with the 'General' tab selected. The 'Connection name' field contains 'Example connection 1'. Below the tabs, there are several settings: 'Connect automatically with priority' is checked with a value of 100; 'All users may connect to this network' is checked; 'Automatically connect to VPN' is unchecked; and 'Metered connection' is set to 'Automatic'.

5.

이더넷 탭에서 장치를 선택하고 선택적으로 이더넷 관련 설정을 추가로 선택합니다.

The screenshot shows the 'Editing Ethernet connection 1' window with the 'Ethernet' tab selected. The 'Connection name' field contains 'Ethernet connection 1'. The 'Device' dropdown is set to 'enp1s0 (52:54:00:6B:74:BE)'. The 'Cloned MAC address' field is empty. The 'MTU' is set to 'automatic'. Under 'Wake on LAN', 'Default' is selected, and 'Phy', 'Unicast', 'Multicast', 'Ignore', 'Broadcast', 'Arp', and 'Magic' are unchecked. The 'Wake on LAN password' field is empty. 'Link negotiation' is set to 'Ignore'. 'Speed' is set to '100 Mb/s' and 'Duplex' is set to 'Full'.

6.

IPv4 Settings 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

Method: Manual

Addresses

Address	Netmask	Gateway
192.0.2.1	24	192.0.2.254

DNS servers: 192.0.2.1

7.

IPv6 Settings 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

Method: Manual

Addresses

Address	Prefix	Gateway
2001:db8:1::1	64	2001:db8:1::fff3

DNS servers: 2001:db8:1::fffd

8.

연결을 저장합니다.

9.

nm-connection-editor 를 닫습니다.

검증 단계

1.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

•

동일한 서브넷의 **IP** 주소를 **ping** 합니다.

IPv4의 경우:

```
# ping 192.0.2.3
```

IPv6의 경우:

```
# ping 2001:db8:1::2
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

- 원격 서브넷의 IP 주소를 ping합니다.

IPv4의 경우:

```
# ping 198.162.3.1
```

IPv6의 경우:

```
# ping 2001:db8:2::1
```

- 명령이 실패하면 기본 게이트웨이를 ping하여 설정을 확인합니다.

IPv4의 경우:

```
# ping 192.0.2.254
```

IPv6의 경우:

```
# ping 2001:db8:1::fff3
```

- 호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 DNS 설정을 확인합니다.

추가 리소스

- 연결에 기본 게이트웨이가 없는 경우 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 **NetworkManager** 구성을 참조하십시오.

8.13. NETWORKMANAGER의 DHCP 클라이언트 변경

기본적으로 **NetworkManager**는 내부 **DHCP** 클라이언트를 사용합니다. 그러나 기본 제공 클라이언트에서 제공하지 않는 기능이 있는 **DHCP** 클라이언트가 필요한 경우 **dhclient**를 사용하도록 **NetworkManager**를 구성할 수도 있습니다.

RHEL은 **dhcpcd**를 제공하지 않으므로 **NetworkManager**는 이 클라이언트를 사용할 수 없습니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/NetworkManager/conf.d/dhcp-client.conf** 파일을 생성합니다.

```
[main]
dhcp=dhclient
```

dhcp 매개 변수를 **internal** (기본값) 또는 **dhclient**로 설정할 수 있습니다.

2. **dhcp** 매개 변수를 **dhclient**로 설정하면 **dhcp-client** 패키지를 설치합니다.

```
# yum install dhcp-client
```

3. **NetworkManager**를 다시 시작합니다.

```
# systemctl restart NetworkManager
```

restart는 임시로 모든 네트워크 연결을 중단합니다.

검증

- **/var/log/messages** 로그 파일에서 다음과 유사한 항목을 검색합니다.

```
Apr 26 09:54:19 server NetworkManager[27748]: <info> [1650959659.8483] dhcp-init:
Using DHCP client 'dhclient'
```

이 로그 항목은 **NetworkManager**가 **dhclient**를 **DHCP** 클라이언트로 사용하는지 확인합니다.

추가 리소스

- **NetworkManager.conf(5) 매뉴얼 페이지**

8.14. NETWORKMANAGER 연결의 DHCP 동작 구성

DHCP(Dynamic Host Configuration Protocol) 클라이언트는 클라이언트가 네트워크에 연결할 때마다 **DHCP** 서버의 동적 IP 주소 및 해당 구성 정보를 요청합니다.

DHCP 서버에서 IP 주소를 검색하도록 연결을 구성하면 **NetworkManager**에서 **DHCP** 서버에서 IP 주소를 요청합니다. 기본적으로 클라이언트는 이 요청이 완료될 때까지 45초 동안 기다립니다. **DHCP** 연결이 시작되면 **dhcp** 클라이언트는 **DHCP** 서버에서 IP 주소를 요청합니다.

사전 요구 사항

- **DHCP**를 사용하는 연결이 호스트에 구성되어 있습니다.

절차

1.

ipv4.dhcp-timeout 및 **ipv6.dhcp-timeout** 속성을 설정합니다. 예를 들어 두 옵션을 모두 30초로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify connection_name ipv4.dhcp-timeout 30 ipv6.dhcp-timeout 30
```

또는 성공할 때까지 IP 주소 요청 및 갱신을 중지하지 않도록 **NetworkManager**가 중단되도록 매개 변수를 **infinity**로 설정합니다.

2.

선택 사항: **NetworkManager**가 시간 초과 전에 IPv4 주소를 수신하지 못하는 경우 동작을 구성합니다.

```
# nmcli connection modify connection_name ipv4.may-fail value
```

ipv4.may-fail 옵션을 다음과 같이 설정하는 경우 다음을 수행합니다.

- 예, 연결 상태는 **IPv6** 구성에 따라 다릅니다.
 - **IPv6** 구성이 활성화되어 성공하면 **NetworkManager**는 **IPv6** 연결을 활성화하고 더 이상 **IPv4** 연결을 활성화하지 않습니다.
 - **IPv6** 구성이 비활성화되거나 구성되지 않은 경우 연결이 실패합니다.
 - 아니요, 연결이 비활성화됩니다. 이 경우 다음을 수행합니다.
 - 연결의 자동 연결 속성이 활성화된 경우 **NetworkManager**는 **autoconnect-retries** 속성에 설정된 대로 연결을 여러 번 다시 활성화합니다. 기본값은 4 입니다.
 - 연결이 계속 **DHCP** 주소를 가져올 수 없는 경우 자동 활성화에 실패합니다. 5분 후에 자동 연결 프로세스가 다시 시작되어 **DHCP** 서버에서 **IP** 주소를 가져옵니다.
3. 선택 사항: **NetworkManager**가 시간 초과 전에 **IPv6** 주소를 수신하지 못하는 경우 동작을 구성합니다.

```
# nmcli connection modify connection_name ipv6.may-fail value
```

추가 리소스

- **nm-settings(5)** 도움말 페이지

8.15. 인터페이스 이름별로 단일 연결 프로필을 사용하여 여러 이더넷 인터페이스 구성

대부분의 경우 하나의 연결 프로필에는 하나의 네트워크 장치 설정이 포함됩니다. 그러나 연결 프로필에서 인터페이스 이름을 설정할 때 **NetworkManager**는 와일드카드도 지원합니다. 동적 **IP** 주소 할당이 있는 이더넷 네트워크 간에 호스트 로밍하는 경우 이 기능을 사용하여 여러 이더넷 인터페이스에 사용할 수 있는 단일 연결 프로필을 만들 수 있습니다.

사전 요구 사항

- 네트워크에서 **DHCP** 사용 가능
- 호스트에 여러 이더넷 어댑터가 있습니다.
- 호스트에 연결 프로필이 없습니다.

절차

1. **enp** 로 시작하는 모든 인터페이스 이름에 적용되는 연결 프로필을 추가합니다.

```
#nmcli connection add con-name Example connection.multi-connect multiple
match.interface-name enp* type ethernet
```

검증 단계

1. 단일 연결 프로필의 모든 설정을 표시합니다.

```
#nmcli connection show Example
```

```
connection.id:           Example
...
connection.multi-connect: 3 (multiple)
match.interface-name:    `enp*`
...
```

3 은 연결 프로필의 네트워크 인터페이스 수가 아니라 연결 프로필에서 활성 상태인 인터페이스 수를 나타냅니다. 연결 프로필은 **match.interface-name** 매개변수의 패턴과 일치하는 모든 장치를 사용하므로 연결 프로필에는 동일한 **UUID(Universally Unique Identifier)**가 있습니다.

2. 연결 상태를 표시합니다.

```
#nmcli connection show
```

```
NAME                UUID                TYPE  DEVICE
...
Example 6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1 ethernet enp7s0
Example 6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1 ethernet enp8s0
Example 6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1 ethernet enp9s0
```

추가 리소스

- [nmcli\(1\) 도움말 페이지](#)
- [nm-settings\(5\) 도움말 페이지](#)

8.16. PCI ID를 사용하여 여러 이더넷 인터페이스에 대한 단일 연결 프로파일 구성

PCI ID는 시스템에 연결된 장치의 고유 식별자입니다. 연결 프로파일은 **PCI ID** 목록을 기반으로 일치하는 인터페이스를 통해 여러 장치를 추가합니다. 이 절차를 사용하여 여러 장치 **PCI ID**를 단일 연결 프로파일에 연결할 수 있습니다.

사전 요구 사항

- 네트워크에서 **DHCP** 서버를 사용할 수 있습니다.
- 호스트에 여러 이더넷 어댑터가 있습니다.
- 시스템에 연결 프로파일 없습니다.

절차

1. 장치 경로를 식별합니다. 예를 들어 **enp** 으로 시작하는 모든 인터페이스의 장치 경로를 표시하려면 다음을 입력합니다.

```
#udevadm info /sys/class/net/enp* | grep ID_PATH=
```

```
...
```

```
E: ID_PATH=pci-0000:07:00.0
```

```
E: ID_PATH=pci-0000:08:00.0
```

2. **0000:00:0[7-8].0** 표현식과 일치하는 모든 **PCI ID**에 적용되는 연결 프로파일을 추가합니다.

```
#nmcli connection add type ethernet connection.multi-connect multiple match.path  
"pci-0000:07:00.0 pci-0000:08:00.0" con-name Example
```

검증 단계

1.

연결 상태를 표시합니다.

```
#nmcli connection show
```

```
NAME    UUID                                TYPE    DEVICE
```

```
...
```

```
Example  9cee0958-512f-4203-9d3d-b57af1d88466 ethernet enp7s0
```

```
Example  9cee0958-512f-4203-9d3d-b57af1d88466 ethernet enp8s0
```

```
...
```

2.

연결 프로파일의 모든 설정을 표시하려면 다음을 수행합니다.

```
#nmcli connection show Example
```

```
connection.id:      Example
```

```
...
```

```
connection.multi-connect: 3 (multiple)
```

```
match.path:         pci-0000:07:00.0,pci-0000:08:00.0
```

```
...
```

이 연결 프로파일은 `match.path` 매개변수의 패턴과 일치하는 **PCI ID**가 있는 모든 장치를 사용하므로 연결 프로파일의 **UUID(Universally Unique Identifier)**가 동일합니다.

추가 리소스

- [`nmcli\(1\)` 도움말 페이지](#)
- [`nm-settings\(5\)` 도움말 페이지](#)

9장. 와이파이 연결 관리

이 섹션에서는 **Wi-Fi** 연결을 구성하고 관리하는 방법을 설명합니다.

9.1. 무선 규제 도메인 설정

Red Hat Enterprise Linux에서 **crda** 패키지에는 지정된 관할지에 대한 무선 규제 규칙을 커널에 제공하는 중앙 규제 도메인 에이전트가 포함되어 있습니다. 특정 **udev** 스크립트에서 사용하며 **udev** 스크립트를 디버깅하지 않는 한 수동으로 실행하면 안 됩니다. 커널은 새로운 규제 도메인 변경 시 **udev** 이벤트를 보내 **crda**를 실행합니다. 규제 도메인 변경은 **Linux** 무선 하위 시스템(**IEEE-802.11**)에서 트리거합니다. 이 하위 시스템은 **compliance.bin** 파일을 사용하여 규제 데이터베이스 정보를 유지합니다.

setregdomain 유틸리티는 시스템의 규제 도메인을 설정합니다. **Setregdomain**은 인수를 사용하지 않으며 일반적으로 관리자가 수동으로 수행하는 대신 **udev**와 같은 시스템 스크립트를 통해 호출됩니다. 국가 코드 조치가 실패하면 시스템 관리자가 **/etc/sysconfig/regdomain** 파일에서 **COUNTRY** 환경 변수를 정의할 수 있습니다.

추가 리소스

- [setregdomain\(1\) 도움말 페이지](#)
- [CRDA\(8\) 도움말 페이지](#)
- [regulatory.bin\(5\) 도움말 페이지](#)
- [IW\(8\) 도움말 페이지](#)

9.2. NMCLI를 사용하여 WI-FI 연결 구성

다음 절차에서는 **nmcli**를 사용하여 **Wi-fi** 연결 프로필을 구성하는 방법을 설명합니다.

사전 요구 사항

- 설치할 **nmcli** 유틸리티입니다.

- 라디오가 **on**(기본값)인지 확인합니다.

```
$ nmcli radio wifi on
```

절차

1.

정적 IP 구성으로 **Wi-Fi** 연결 프로필을 생성하려면 다음을 수행합니다.

```
$ nmcli con add con-name MyCafe ifname wlan0 type wifi ssid MyCafe ip4
192.0.2.101/24 gw4 192.0.2.1
```

2.

DNS 서버를 설정합니다. 예를 들어 **192.0.2.1** 을 **DNS** 서버로 설정하려면 다음을 수행합니다.

```
$ nmcli con modify con-name MyCafe ipv4.dns "192.0.2.1"
```

3.

선택적으로 **DNS** 검색 도메인을 설정합니다. 예를 들어 검색 도메인을 **example.com** 으로 설정하려면 다음을 수행합니다.

```
$ nmcli con modify con-name MyCafe ipv4.dns-search "example.com"
```

4.

특정 속성을 확인하려면 **example mtu**:

```
$ nmcli connection show id MyCafe | grep mtu
802-11-wireless.mtu: auto
```

5.

설정 속성을 변경하려면 다음을 수행합니다.

```
$ nmcli connection modify id MyCafe wireless.mtu 1350
```

6.

변경 사항을 확인하려면 다음을 수행합니다.

```
$ nmcli connection show id MyCafe | grep mtu
802-11-wireless.mtu: 1350
```

검증 단계

1.

ping 유틸리티를 사용하여 이 호스트가 다른 호스트에 패킷을 보낼 수 있는지 확인합니다.

- 동일한 서브넷의 IP 주소를 **ping**합니다. 예를 들면 다음과 같습니다.

```
# ping 192.0.2.103
```

명령이 실패하면 IP 및 서브넷 설정을 확인합니다.

- 원격 서브넷의 IP 주소를 **ping**합니다. 예를 들면 다음과 같습니다.

```
# ping 198.51.16.3
```

- 명령이 실패하면 기본 게이트웨이를 **ping**하여 설정을 확인합니다.

```
# ping 192.0.2.1
```

2.

호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들면 다음과 같습니다.

```
# host client.example.com
```

명령이 시간 초과되었거나 서버에 연결할 수 없는 경우와 같은 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

추가 리소스

- **nm-settings(5)** 도움말 페이지
- **NetworkManager**는 **NetworkManager** 서비스를 다시 시작한 후 연결을 중복 합니다.

9.3. 컨트롤 센터를 사용하여 Wi-Fi 연결 구성

Wi-Fi에 연결하면 현재 네트워크 연결에 따라 네트워크 설정이 미리 채워집니다. 즉, 인터페이스가 네트워크에 연결할 때 설정이 자동으로 탐지됩니다.

이 절차에서는 **control-center**를 사용하여 수동으로 **Wi-Fi** 설정을 구성하는 방법을 설명합니다.

절차

1. **Super** 키를 눌러 **Activities Overview** (활동 개요)를 입력하고 **Wi-Fi** 를 입력한 다음 **Enter** 를 누릅니다. 왼쪽 메뉴 항목에 사용 가능한 네트워크 목록이 표시됩니다.
2. 편집할 **Wi-Fi** 연결 이름 오른쪽에 있는 기어바퀴 아이콘을 선택하면 편집하는 연결 대화 상자가 표시됩니다. **Details**(세부 정보) 메뉴 창에 추가 구성을 만들 수 있는 연결 세부 정보가 표시됩니다.

옵션

- a. 자동으로 연결을 선택하면 **NetworkManager** 가 사용 가능한 것을 탐지할 때마다 **NetworkManager** 가 이 연결에 자동으로 연결됩니다. **NetworkManager** 가 자동으로 연결되지 않도록 하려면 확인란을 지웁니다. 확인란이 명확하면 네트워크 연결 아이콘의 메뉴에서 해당 연결을 수동으로 선택하여 연결되도록 해야 합니다.
- b. 다른 사용자가 연결을 사용할 수 있도록 하려면 **Make available to other users** (다른 사용자가 사용할 수 있도록 만들기) 확인란을 선택합니다.
- c. 배경 데이터 사용 제한 옵션을 변경하여 백그라운드 데이터 사용을 제어할 수도 있습니다. **You can also control the background data usage by changing the Restrict background data usage option.**



참고

Wi-Fi 연결을 삭제하려면 **Forget Connection** (연결 허용) 빨간색 상자를 클릭합니다.

3. **Identity(ID)** 메뉴 항목을 선택하여 기본 구성 옵션을 확인합니다.

SSID - 액세스 지점(AP)의 **SSID**(Service Set Identifier)입니다.

BSSID - **BSSID**(기본 서비스 세트 식별자)는 인프라 모드에서 연결하는 특정 무선 액세스 지점의 하드웨어 주소라고도 하는 **MAC** 주소입니다. 이 필드는 기본적으로 비어 있으며 **BSSID** 를 지정하지 않고도 **SSID** 로 무선 액세스 지점에 연결할 수 있습니다. **BSSID**를 지정하면 시스템이 특정 액세스 지점에만 연결되도록 강제 적용합니다. 애드혹 네트워크의 경우 애드혹 네트워크가 생성될 때 **mac80211** 하위 시스템에서 **BSSID** 가 임의로 생성됩니다. **NetworkManager** 에 의해 표시되지 않습니다.

MAC 주소 - MAC 주소를 특정 무선 어댑터와 특정 연결(또는 연결)과 연결할 수 있습니다.

복제 주소 - 실제 하드웨어 주소 대신 사용할 복제된 MAC 주소입니다. 필요 없는 한 비워 두십시오.

4.

추가 IP 주소 구성에서는 **IPv4** 및 **IPv 6** 메뉴 항목을 선택합니다.

기본적으로 **IPv4** 및 **IPv 6** 는 모두 현재 네트워크 설정에 따라 자동 구성으로 설정됩니다. 즉, 인터페이스가 네트워크에 연결할 때 로컬 IP 주소, **DNS** 주소 및 기타 설정과 같은 주소가 자동으로 탐지됩니다. **DHCP** 서버가 이 네트워크의 IP 구성을 할당하는 경우 충분하지만 **IPv4** 및 **IPv 6** 설정에서 정적 구성을 제공할 수도 있습니다. **IPv4** 및 **IPv6** 메뉴 항목에는 다음 설정이 표시됩니다.

- **IPv4 방법**
 - 자동(**DHCP**) - 연결 중인 네트워크가 라우터 알립(**RA**)을 사용하거나 **DHCP** 서버를 사용하여 동적 IP 주소를 할당하는 경우 이 옵션을 선택합니다. **Details** (세부 정보) 메뉴 항목에 할당된 IP 주소를 볼 수 있습니다.
 - 링크-로컬 전용 - 연결 중인 네트워크에 **DHCP** 서버가 없고 IP 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. 임의의 주소는 **RFC 3927** 에 따라 접두사 **169.254/16** 과 같이 할당됩니다.
 - 수동 - IP 주소를 수동으로 할당하려면 이 옵션을 선택합니다.
 - **disable** - 이 연결에 대해 **IPv4** 가 비활성화되어 있습니다.

- **DNS**

Automatic (자동)이 있고 이 연결에 **DNS** 서버를 할당하는 **DHCP** 서버를 사용할 수 없는 경우 이를 **OFF** (꺼 짐)로 전환하여 IP를 심프로 구분하는 **DNS** 서버의 IP 주소를 입력합니다.

- **라우트**

Routes(경로) 섹션에서 **Automatic(자동)** 이 **ON** 이면 라우터 알림(RA) 또는 **DHCP**의 경로가 사용되지만 정적 경로도 추가할 수 있습니다. **OFF** 인 경우 정적 경로만 사용됩니다.

- **Address (주소)** - 원격 네트워크, 하위 네트워크 또는 호스트의 **IP** 주소를 입력합니다.
- **넷마스크** - 위에 입력한 **IP** 주소의 넷마스크 또는 접두사 길이입니다.
- **gateway** - 원격 네트워크, 서브넷 또는 위에서 입력한 호스트로 향하는 게이트웨이의 **IP** 주소입니다.
- **지표** - 네트워크 비용, 이 경로에 제공할 기본 설정 값입니다. 더 낮은 값은 더 높은 값보다 우선합니다.

- 이 연결은 네트워크의 리소스에만 사용하십시오.

이 확인란을 선택하여 연결이 기본 경로가 되지 않도록 합니다.

또는 **Wi-Fi** 연결에서 **IPv6** 설정을 구성하려면 **IPv6** 메뉴 항목을 선택합니다.

- **IPv6 방법**

- **auto** - 이 옵션을 선택하여 하드웨어 주소 및 라우터 알림(RA)을 기반으로 자동 상태 비저장 구성을 생성하도록 **IPv6** 상태 비저장 주소 자동 구성(SLAAC)을 사용합니다.
- **Automatic, DHCP only** - 이 옵션을 선택하여 RA를 사용하지 않고 **DHCPv6**의 정보를 직접 요청하여 상태 저장 구성을 생성합니다.
- **링크-로컬 전용** - 연결 중인 네트워크에 **DHCP** 서버가 없고 **IP** 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. **RFC 4862**에 따라 접두사 **FE80::0**이 있는 임의 주소가 할당됩니다.
- **수동** - **IP** 주소를 수동으로 할당하려면 이 옵션을 선택합니다.

○

disable - 이 연결에 대해 IPv6 가 비활성화되어 있습니다.

●

DNS, Routes, 네트워크 필드의 리소스에만 이 연결을 사용하는 것은 IPv4 설정에서 일반적입니다.

5.

Wi-Fi 연결에서 보안 설정을 구성하려면 **Security** (보안) 메뉴 항목을 선택합니다.



주의

암호화 없이 또는 안전하지 않은 **WEP** 또는 **WPA** 표준만 지원하는 **Wi-Fi** 네트워크에 연결하지 마십시오.

다음과 같은 구성 옵션을 사용할 수 있습니다.

●

보안

○

none - 암호화가 비활성화되어 있으며 데이터는 네트워크를 통해 일반 텍스트로 전송됩니다.

○

WEP 40/128비트 키 - IEEE 802.11 표준의 **WEP**(유선 동등 개인 정보 보호). 단일 사전 공유 키()를 사용합니다.

○

WEP 128비트 암호 - WEP 키를 파생하기 위한 암호의 **MD5** 해시.

○

동적 **WEP(802.1X)** - **WEP** 키는 동적으로 변경됩니다.

○

LEAP - Cisco 시스템의 경량 확장 인증 프로토콜.

○

WPA 및 **WPA2** 개인용 - **WPA(Fi-Fi Protected Access)**의 초안 **IEEE 802.11i** 표

준에서 제공. **WPA2(WPA2) Wi-Fi Protected Access 2(WPA2)**. 개인 모드는 사전 공유 키(WPA-)를 사용합니다.

- **WPA 및 WPA2 Enterprise - IEEE 802.1X 네트워크 액세스 제어를 제공하기 위해 RADIUS 인증 서버와 함께 사용하기 위한 WPA 및 WPA 2.**

- **WPA3 개인용 - Wi-Fi 보호 액세스 3(WPA3) 개인은 사전 공유 키(PSK) 대신 Simultaneous Authentication of Equals(SAE)를 사용하여 사전 공격을 방지합니다. WPA3는 완벽한 전달 보안을 사용합니다.**

- **password - 인증 프로세스에 사용할 암호를 입력합니다.**

6. 구성을 완료하고 나면 **Apply (적용)** 버튼을 클릭하여 저장합니다.



참고

더하기 버튼을 클릭하여 새 연결을 추가하면 **NetworkManager** 가 해당 연결에 대한 새 구성 파일을 만든 다음 기존 연결을 편집하는 데 사용되는 동일한 대화 상자를 엽니다. 이러한 대화 상자의 차이점은 기존 연결 프로필에 **Details** 메뉴 항목이 있다는 것입니다.

9.4. NMCLI를 사용하여 WI-FI 네트워크에 연결

다음 절차에서는 **nmcli** 유틸리티를 사용하여 무선 연결에 연결하는 방법을 설명합니다.

사전 요구 사항

- 설치할 **nmcli** 유틸리티입니다.
- 라디오가 **on**(기본값)인지 확인합니다.

```
$ nmcli radio wifi on
```

절차

1. 사용 가능한 **Wi-Fi** 연결 목록을 새로 고치려면 다음을 수행합니다.

```
$ nmcli device wifi rescan
```

2.

사용 가능한 **Wi-Fi** 액세스 지점을 보려면 다음을 수행합니다.

```
$ nmcli dev wifi list
```

```
IN-USE SSID   MODE CHAN RATE   SIGNAL BARS SECURITY
...
MyCafe  Infra 3    405 Mbit/s 85   WPA1 WPA2
```

3.

nmcli 를 사용하여 **Wi-Fi** 연결에 연결하려면 다음을 수행합니다.

```
$ nmcli dev wifi connect SSID-Name password wireless-password
```

예를 들면 다음과 같습니다.

```
$ nmcli dev wifi connect MyCafe password wireless-password
```

Wi-Fi 상태를 비활성화하려면 다음을 수행합니다.

```
$ nmcli radio wifi off
```

9.5. NMCLI를 사용하여 숨겨진 **Wi-Fi** 네트워크에 연결

모든 액세스 지점에 식별할 **Service Set Identifier(SSID)**가 있습니다. 그러나 액세스 지점은 **SSID**를 브로드캐스트하지 않도록 구성할 수 있습니다. 이 경우 숨겨지고 **NetworkManager**의 사용 가능 네트워크 목록에 표시되지 않습니다.

다음 절차에서는 **nmcli** 도구를 사용하여 숨겨진 네트워크에 연결하는 방법을 설명합니다.

사전 요구 사항

- 설치할 **nmcli** 유틸리티입니다.
- **SSID** 및 **Wi-Fi** 연결의 암호를 알아야 합니다.

- 라디오가 **on**(기본값)인지 확인합니다.

```
$ nmcli radio wifi on
```

절차

- 숨겨진 **SSID**에 연결합니다.

```
$ nmcli dev wifi connect SSID_Name password wireless_password hidden yes
```

9.6. GNOME GUI를 사용하여 WI-FI 네트워크에 연결

이 절차에서는 무선 네트워크에 연결하여 인터넷에 액세스하는 방법을 설명합니다.

절차

1. 화면의 오른쪽 상단에 있는 **GNOME Shell network connection**(GNOME 셸 네트워크 연결 아이콘) 메뉴를 엽니다.
2. **Wi-Fi Not Connected** 를 선택합니다.
3. **Select Network** (네트워크 선택) 옵션을 클릭합니다.
4. 연결할 네트워크 이름을 클릭한 다음 **Connect** (연결)를 클릭합니다.

네트워크가 표시되지 않으면 네트워크가 숨겨질 수 있습니다.
5. 네트워크를 암호 또는 암호화 키로 보호해야 하는 경우 암호를 입력하고 연결을 클릭합니다.

암호를 모르는 경우 **Wi-Fi** 네트워크의 관리자에게 문의하십시오.
6. 연결에 성공하면 네트워크 이름이 연결 아이콘 메뉴에 표시되고 무선 표시기가 화면의 오른쪽 상단에 있습니다.

추가 요소:

- 제어 센터를 사용하여 **Wi-Fi 연결 구성**

9.7. NMCLI를 사용하여 기존 WI-FI 연결에서 802.1X 네트워크 인증 구성

nmcli 유틸리티를 사용하여 네트워크에 자체적으로 인증하도록 클라이언트를 구성할 수 있습니다. 이 절차에서는 **wlp1s0**이라는 기존 **NetworkManager** Wi-Fi 연결 프로파일에서 **MSCHAPv2**(Microsoft Challenge-Handshake Authentication Protocol) 버전을 사용하여 **PEAP**(Protected Extensible Authentication Protocol) 인증을 구성하는 방법을 설명합니다.

사전 요구 사항

1. 네트워크에는 **802.1X** 네트워크 인증이 있어야 합니다.
2. **Wi-Fi 연결 프로파일**은 **NetworkManager**에 있으며 유효한 **IP** 구성이 있습니다.
3. 클라이언트가 인증자의 인증서를 확인해야 하는 경우 인증 기관(**CA**) 인증서를 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 저장해야 합니다.
4. **wpa_supplicant** 패키지가 설치되어 있습니다.

절차

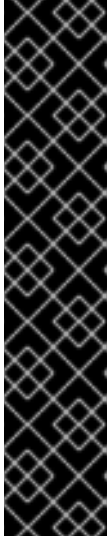
1. **Wi-Fi 보안 모드**를 **wpa-eap**, 확장 가능 인증 프로토콜(**EAP**)을 **peap** 으로, 내부 인증 프로토콜을 **mschapv2** 로, 사용자 이름으로 설정합니다.

```
# nmcli connection modify wlp1s0 wireless-security.key-mgmt wpa-eap 802-1x.eap
peap 802-1x.phase2-auth mschapv2 802-1x.identity user_name
```

단일 명령에서 **wireless-security.key-backend**, **802-1x.eap**, **802-1x.phase2-auth** **.identity** 매개변수를 설정해야 합니다.

2. 선택적으로 구성에 암호를 저장합니다.

```
# nmcli connection modify wlp1s0 802-1x.password password
```



중요

기본적으로 **NetworkManager**는 **root** 사용자만 읽을 수 있는 **/etc/sysconfig/network-scripts/keys-connection_name** 파일에 암호를 일반 텍스트로 저장합니다. 그러나 구성 파일의 일반 텍스트 암호는 보안 위험이 될 수 있습니다.

보안을 강화하려면 **802-1x.password-flags** 매개 변수를 **0x1** 로 설정합니다. 이 설정을 사용하면 **GNOME** 데스크탑 환경 또는 **nm-applet** 이 실행 중인 서버에서 **NetworkManager**가 이러한 서비스에서 암호를 검색합니다. 다른 경우에는 **NetworkManager**에서 암호를 묻는 메시지를 표시합니다.

3.

클라이언트가 인증자 인증서를 확인해야 하는 경우 연결 프로필의 **802-1x.ca-cert** 매개변수를 **CA** 인증서 경로로 설정합니다.

```
# nmcli connection modify wlp1s0 802-1x.ca-cert /etc/pki/ca-trust/source/anchors/ca.crt
```



참고

보안상의 이유로 **Red Hat**은 인증자 인증서를 사용하여 클라이언트가 인증자 ID를 검증할 수 있도록 하는 것이 좋습니다.

4.

연결 프로필을 활성화합니다.

```
# nmcli connection up wlp1s0
```

검증 단계

-

네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

추가 리소스

-

[와이파이 연결 관리](#)

-

nm-settings(5) 매뉴얼 페이지의 **802-1x** 설정 섹션

- ***nmcli(1) 도움말 페이지***

10장. VLAN 태그 지정 구성

이 섹션에서는 **VLAN(Virtual Local Area Network)**을 구성하는 방법에 대해 설명합니다. **VLAN**은 실제 네트워크 내의 논리적 네트워크입니다. **VLAN** 인터페이스는 인터페이스를 통과하면 **VLAN ID**로 패킷에 태그를 지정하고, 패킷 반환 태그를 제거합니다.

이더넷, 본딩, 팀 또는 브리지 장치와 같은 다른 인터페이스 위에 **VLAN** 인터페이스를 만듭니다. 이 인터페이스를 상위 인터페이스 라고 합니다.

10.1. NMCLI 명령을 사용하여 VLAN 태그 지정 구성

이 섹션에서는 **nmcli** 유틸리티를 사용하여 **VLAN(Virtual Local Area Network)** 태그를 구성하는 방법을 설명합니다.

사전 요구 사항

- 가상 **VLAN** 인터페이스의 상위로 사용하려는 인터페이스는 **VLAN** 태그를 지원합니다.
- 본딩 인터페이스 상단에 **VLAN**을 구성하는 경우:
 - 본딩의 포트가 설정되어 있습니다.
 - 본딩이 **fail_over_mac=follow** 옵션으로 구성되지 않았습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽은 여전히 잘못된 소스 **MAC** 주소로 전송됩니다.
 - 본딩은 일반적으로 **DHCP** 서버 또는 **IPv6** 자동 구성에서 **IP** 주소를 가져올 것으로 예상되지 않습니다. 본딩을 생성하는 동안 **ipv4.method=disable** 및 **ipv6.method=ignore** 옵션을 설정하여 확인합니다. 그렇지 않으면 일정 시간 후에 **DHCP** 또는 **IPv6** 자동 구성이 실패하면 인터페이스가 중단될 수 있습니다.
- 호스트가 연결된 스위치는 **VLAN** 태그를 지원하도록 구성됩니다. 자세한 내용은 스위치 설명서를 참조하십시오.

절차

1.

네트워크 인터페이스를 표시합니다.

```
# nmcli device status
DEVICE  TYPE    STATE    CONNECTION
enp1s0  ethernet disconnected enp1s0
bridge0 bridge   connected bridge0
bond0   bond    connected bond0
...
```

2.

VLAN 인터페이스를 만듭니다. 예를 들어 **enp1s0** 을 상위 인터페이스로 사용하고 **VLAN ID 10** 으로 패킷을 태그하는 **vlan10** 이라는 **VLAN** 인터페이스를 생성하려면 다음을 입력합니다.

```
# nmcli connection add type vlan con-name vlan10 ifname vlan10 vlan.parent enp1s0
vlan.id 10
```

VLAN은 0 에서 4094 사이의 범위 내에 있어야 합니다.

3.

기본적으로 **VLAN** 연결은 상위 인터페이스에서 최대 전송 단위(MTU)를 상속합니다. 선택적으로 다른 **MTU** 값을 설정합니다.

```
# nmcli connection modify vlan10 ethernet.mtu 2000
```

4.

VLAN 장치의 **IP** 설정을 구성합니다. 이 **VLAN** 장치를 다른 장치의 포트로 사용하려면 이 단계를 건너뛰니다.

a.

IPv4 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **vlan10** 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify vlan10 ipv4.addresses '192.0.2.1/24'
# nmcli connection modify vlan10 ipv4.gateway '192.0.2.254'
# nmcli connection modify vlan10 ipv4.dns '192.0.2.253'
# nmcli connection modify vlan10 ipv4.method manual
```

b.

IPv6 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **vlan10** 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify vlan10 ipv6.addresses '2001:db8:1::1/32'
# nmcli connection modify vlan10 ipv6.gateway '2001:db8:1::fffe'
# nmcli connection modify vlan10 ipv6.dns '2001:db8:1::fffd'
# nmcli connection modify vlan10 ipv6.method manual
```

5.

연결을 활성화합니다.

```
# nmcli connection up vlan10
```

검증 단계

•

설정을 확인합니다.

```
# ip -d addr show vlan10
4: vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 52:54:00:72:2f:6e brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
    gso_max_size 65536 gso_max_segs 65535
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute vlan10
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::1/32 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

추가 리소스

•

[기본 네트워크 설정 테스트.](#)

•

[기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성.](#)

•

[nmcli-examples\(7\) 도움말 페이지](#)

•

[nm-settings\(5\) 매뉴얼 페이지의 vlan 설정 섹션](#)

10.2. RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 구성

이 섹션에서는 RHEL 웹 콘솔을 사용하여 네트워크 브릿지를 구성하는 방법을 설명합니다.

사전 요구 사항

•

가상 VLAN 인터페이스의 상위로 사용하려는 인터페이스는 VLAN 태그를 지원합니다.

- - 본딩 인터페이스 상단에 **VLAN**을 구성하는 경우:
 - 본딩의 포트가 설정되어 있습니다.
 - 본딩이 **fail_over_mac=follow** 옵션으로 구성되지 않았습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽은 여전히 잘못된 소스 **MAC** 주소로 전송됩니다.
 - 본딩은 일반적으로 **DHCP** 서버 또는 **IPv6** 자동 구성에서 **IP** 주소를 가져올 것으로 예상되지 않습니다. 본딩을 생성하여 **IPv4** 및 **IPv6** 프로토콜을 비활성화하여 확인합니다. 그렇지 않으면 일정 시간 후에 **DHCP** 또는 **IPv6** 자동 구성이 실패하면 인터페이스가 중단될 수 있습니다.
- 호스트가 연결된 스위치는 **VLAN** 태그를 지원하도록 구성됩니다. 자세한 내용은 스위치 설명서를 참조하십시오.

절차

1. 화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.
2. **Interfaces (interface)** 섹션에서 **Add VLAN** 을 클릭합니다.
3. 상위 장치를 선택합니다.
4. **VLAN ID**를 입력합니다.
5. **VLAN** 장치 이름을 입력하거나 자동으로 생성된 이름을 유지합니다.

VLAN settings
✕

Parent
enp1s0

VLAN ID
10

Name
enp1s0.10

Apply
Cancel

6.

Apply(적용)를 클릭합니다.

7.

기본적으로 **VLAN** 장치는 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

a.

Interfaces (interface) 섹션에서 **VLAN** 장치의 이름을 클릭합니다.

b.

구성할 프로토콜 옆에 있는 편집 을 클릭합니다.

c.

주소 옆에 있는 **Manual** 을 선택하고 **IP** 주소, 접두사 및 기본 게이트웨이를 입력합니다.

d.

DNS 섹션에서 + 버튼을 클릭하고 **DNS** 서버의 **IP** 주소를 입력합니다. 이 단계를 반복하여 여러 **DNS** 서버를 설정합니다.

e.

DNS 검색 도메인 섹션에서 + 버튼을 클릭하고 검색 도메인을 입력합니다.

f.

인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 해당 경로를 구성합니다.

IPv4 settings

Addresses

Manual

+

Address	Prefix length or netmask	Gateway
192.0.2.1	24	192.0.2.254

DNS

Automatic

+

Server

192.0.2.253

-

DNS search domains

Automatic

+

Search domain

example.com

-

Routes

Automatic

+

Apply

Cancel

g.

적용을 클릭합니다.

검증

•

화면 왼쪽의 탐색에서 **Networking** 탭을 선택하고 인터페이스에 들어오고 나가는 트래픽이 있는지 확인합니다.

Interfaces			
Add bond Add team Add bridge Add VLAN			
Name	IP address	Sending	Receiving
enp1s0.10	192.0.2.1/24	1.11 Mbps	61.2 Mbps

10.3. NM-CONNECTION-EDITOR를 사용하여 VLAN 태그 구성

이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 **VLAN**(Virtual Local Area Network) 태그를 구성하는 방법을 설명합니다.

사전 요구 사항

- 가상 **VLAN** 인터페이스의 상위로 사용하려는 인터페이스는 **VLAN** 태그를 지원합니다.
- 본딩 인터페이스 상단에 **VLAN**을 구성하는 경우:
 - 본딩의 포트가 설정되어 있습니다.
 - 본딩이 **fail_over_mac=follow** 옵션으로 구성되지 않았습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽은 여전히 잘못된 소스 **MAC** 주소로 전송됩니다.
- 호스트가 연결되어 **VLAN** 태그를 지원하도록 구성되어 있습니다. 자세한 내용은 스위치 설명서를 참조하십시오.

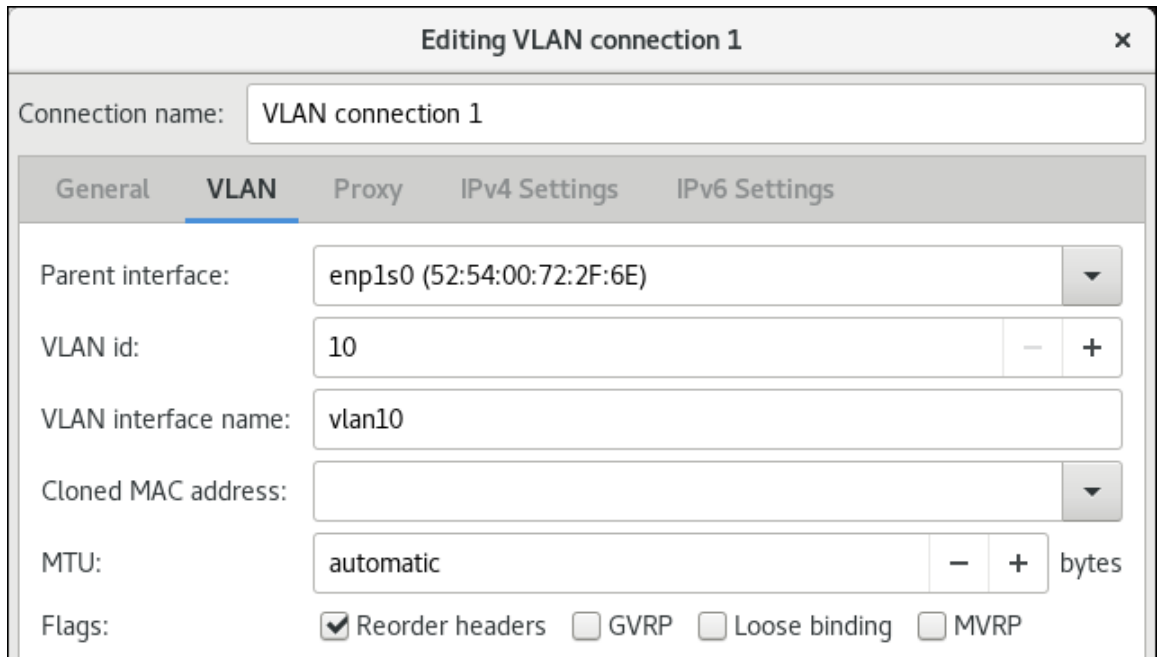
절차

1. 터미널을 열고 **nm-connection-editor** 를 입력합니다.


```
$ nm-connection-editor
```
2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. **VLAN** 연결 유형을 선택하고 생성을 클릭합니다.
4. **VLAN** 탭에서:
 - a. 상위 인터페이스를 선택합니다.
 - b. **VLAN ID**를 선택합니다. **VLAN**은 0 에서 4094 사이의 범위 내에 있어야 합니다.
 - c. 기본적으로 **VLAN** 연결은 상위 인터페이스에서 최대 전송 단위(**MTU**)를 상속합니다. 선택적으로 다른 **MTU** 값을 설정합니다.

d.

선택적으로 **VLAN** 인터페이스의 이름과 추가 **VLAN** 관련 옵션을 설정합니다.



The screenshot shows a window titled "Editing VLAN connection 1". At the top, there's a tab bar with "General", "VLAN", "Proxy", "IPv4 Settings", and "IPv6 Settings". The "VLAN" tab is active. Below the tabs, the "Connection name:" field contains "VLAN connection 1". The main settings area includes:

- Parent interface:** A dropdown menu showing "enp1s0 (52:54:00:72:2F:6E)".
- VLAN id:** A text input field with "10", flanked by minus and plus buttons.
- VLAN interface name:** A text input field with "vlan10".
- Cloned MAC address:** A dropdown menu that is currently empty.
- MTU:** A text input field with "automatic", flanked by minus and plus buttons, with the unit "bytes" to the right.
- Flags:** A row of four checkboxes: "Reorder headers" (checked), "GVRP" (unchecked), "Loose binding" (unchecked), and "MVRP" (unchecked).

5.

VLAN 장치의 **IP** 설정을 구성합니다. 이 **VLAN** 장치를 다른 장치의 포트로 사용하려면 이 단계를 건너뛰십시오.

a.

IPv4 Settings 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

Editing VLAN connection 1

Connection name: VLAN connection 1

General VLAN Proxy **IPv4 Settings** IPv6 Settings

Method: Manual

Addresses

Address	Netmask	Gateway
192.0.2.1	24	192.0.2.254

DNS servers: 192.0.2.253

b.

IPv6 Settings 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

Editing VLAN connection 1

Connection name: VLAN connection 1

General VLAN Proxy IPv4 Settings **IPv6 Settings**

Method: Manual

Addresses

Address	Prefix	Gateway
2001:db8:1::1	64	2001:db8:1::fff3

DNS servers: 2001:db8:1::fffd

6.

Save(저장)를 클릭하여 **VLAN** 연결을 저장합니다.

7.

nm-connection-editor 를 닫습니다.

검증 단계

1.

설정을 확인합니다.

```
# ip -d addr show vlan10
4: vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 52:54:00:d5:e0:fb brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
    gso_max_size 65536 gso_max_segs 65535
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute vlan10
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::1/32 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

추가 리소스

- [기본 네트워크 설정 테스트.](#)
- [기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성.](#)

10.4. NMSTATECTL을 사용하여 VLAN 태그 지정 구성

이 섹션에서는 `nmstatectl` 유틸리티를 사용하여 이더넷 연결을 사용하는 ID 10으로 VLAN을 구성하는 방법을 설명합니다. 하위 장치로 VLAN 연결에는 IP, 기본 게이트웨이 및 DNS 구성이 포함됩니다.

환경에 따라 YAML 파일을 적절하게 조정합니다. 예를 들어 VLAN에서 브리지 또는 본딩 장치를 사용하려면 VLAN에서 사용하는 포트의 `base-iface` 속성 및 `type` 속성을 조정하십시오.

사전 요구 사항

- 이더넷 장치를 VLAN의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- `nmstate` 패키지가 설치되어 있습니다.

절차

1.

다음 콘텐츠를 사용하여 **YAML 파일(예: ~/create-vlan.yml)**을 생성합니다.

```
---
interfaces:
- name: vlan10
  type: vlan
  state: up
  ipv4:
    enabled: true
    address:
      - ip: 192.0.2.1
        prefix-length: 24
    dhcp: false
  ipv6:
    enabled: true
    address:
      - ip: 2001:db8:1::1
        prefix-length: 64
    autoconf: false
    dhcp: false
  vlan:
    base-iface: enp1s0
    id: 10
- name: enp1s0
  type: ethernet
  state: up

routes:
  config:
    - destination: 0.0.0.0/0
      next-hop-address: 192.0.2.254
      next-hop-interface: vlan10
    - destination: ::/0
      next-hop-address: 2001:db8:1::fffe
      next-hop-interface: vlan10

dns-resolver:
  config:
    search:
      - example.com
    server:
      - 192.0.2.200
      - 2001:db8:1::ffbb
```

2.

시스템에 설정을 적용합니다.

```
# nmstatectl apply ~/create-vlan.yml
```

검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE   TYPE   STATE   CONNECTION
vlan10   vlan   connected vlan10
```

2.

연결 프로파일의 모든 설정을 표시합니다.

```
# nmcli connection show vlan10
connection.id:      vlan10
connection.uuid:    1722970f-788e-4f81-bd7d-a86bf21c9df5
connection.stable-id: --
connection.type:    vlan
connection.interface-name: vlan10
...
```

3.

연결 설정을 **YAML** 형식으로 표시합니다.

```
# nmstatectl show vlan0
```

추가 리소스

- [nmstatectl\(8\) 도움말 페이지](#)
- [/usr/share/doc/nmstate/examples/](#)

10.5. RHEL 시스템 역할을 사용하여 VLAN 태그 지정 구성

네트워크 **RHEL** 시스템 역할을 사용하여 **VLAN** 태그를 구성할 수 있습니다. 다음 절차에서는 이더넷 연결 및 이 이더넷 연결 상단에 **ID 10** 이 있는 **VLAN**을 추가하는 방법을 설명합니다. 하위 장치로 **VLAN** 연결에는 **IP**, 기본 게이트웨이 및 **DNS** 구성이 포함됩니다.

환경에 따라 그에 따라 플레이를 조정합니다. 예를 들면 다음과 같습니다.

- 본딩과 같은 다른 연결의 포트로 **VLAN**을 사용하려면 **ip** 속성을 생략하고 하위 구성에서 **IP** 구성을 설정합니다.
-

VLAN에서 팀, 브리지 또는 본딩 장치를 사용하려면 VLAN에서 사용하는 포트의 `interface_name` 및 `type` 속성을 조정합니다.

사전 요구 사항

- **`ansible` 및 `rhel-system-roles` 패키지는 제어 노드에 설치됩니다.**
- **플레이북을 실행할 때 `root`와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 `sudo` 권한이 있습니다.**

절차

1. **플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 IP 또는 이름을 `/etc/ansible/hosts` Ansible 인벤토리 파일에 추가합니다.**

```
node.example.com
```

2. **다음 콘텐츠를 사용하여 `~/vlan-ethernet.yml` 플레이북을 생성합니다.**

```
---
- name: Configure a VLAN that uses an Ethernet connection
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      # Add an Ethernet profile for the underlying device of the VLAN
      - name: enp1s0
        type: ethernet
        interface_name: enp1s0
        autoconnect: yes
        state: up
        ip:
          dhcp4: no
          auto6: no

      # Define the VLAN profile
      - name: enp1s0.10
        type: vlan
        ip:
          address:
            - "192.0.2.1/24"
            - "2001:db8:1::1/64"
```

```

gateway4: 192.0.2.254
gateway6: 2001:db8:1::fffe
dns:
- 192.0.2.200
- 2001:db8:1::ffbb
dns_search:
- example.com
vlan_id: 10
parent: enp1s0
state: up

```

VLAN 프로파일의 상위 속성은 **enp1s0** 장치 상단에서 작동하도록 **VLAN**을 구성합니다.

3.

플레이북을 실행합니다.

-

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/vlan-ethernet.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/vlan-ethernet.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

-

[/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file

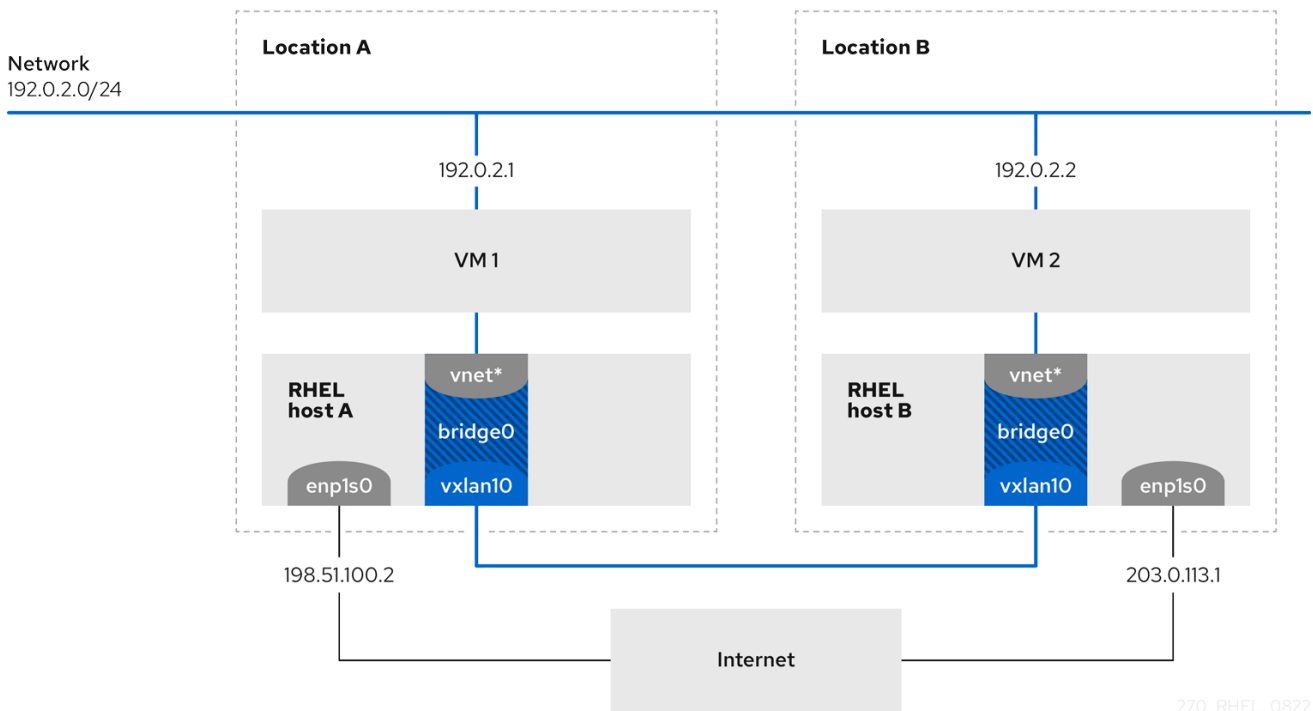
-

[ansible-playbook\(1\)](#) 도움말 페이지

11장. VXLAN을 사용하여 VM의 가상 계층 2 도메인 만들기

VXLAN(Virtual extensible LAN)은 **UDP** 프로토콜을 사용하여 **IP** 네트워크를 통해 계층 2 트래픽을 터널링하는 네트워킹 프로토콜입니다. 예를 들어 다른 호스트에서 실행 중인 특정 **VM(가상 머신)**은 **VXLAN** 터널을 통해 통신할 수 있습니다. 호스트는 다른 서버넷에 있거나 전 세계 다른 데이터 센터에 있을 수 있습니다. **VM**의 관점에서 동일한 **VXLAN**의 다른 **VM**은 동일한 계층 2 도메인 내에 있습니다.

이 문서에서는 **VM**에 표시되지 않는 **RHEL** 호스트에서 **VXLAN**을 구성하는 방법을 설명합니다.



270_RHEL_0822

이 예에서 **RHEL-host-A** 및 **RHEL-host-B**는 브리지 **br0** 을 사용하여 각 호스트에서 **VM**의 가상 네트워크를 **MTU 10** 이라는 **VXLAN**으로 연결합니다. 이 구성으로 인해 **VXLAN**은 **VM**에 표시되지 않으며 **VM**에는 특별한 구성이 필요하지 않습니다. 나중에 더 많은 **VM**을 동일한 가상 네트워크에 연결하면 **VM**이 동일한 가상 계층 2 도메인의 자동으로 멤버가 됩니다.



중요

일반 계층-2 트래픽과 마찬가지로 **VXLAN**의 데이터는 암호화되지 않습니다. 보안상의 이유로 **VPN** 또는 다른 유형의 암호화된 연결을 통해 **VXLAN**을 사용하십시오.

11.1. VXLAN의 이점

VXLAN(가상 확장 가능 LAN)은 다음과 같은 주요 이점을 제공합니다.

- **VXLAN은 24비트 ID를 사용합니다. 따라서 최대 16,777,216개의 격리된 네트워크를 생성할 수 있습니다. 예를 들어 VLAN(가상 LAN)은 4,096개의 격리된 네트워크만 지원합니다.**
- **VXLAN은 IP 프로토콜을 사용합니다. 이를 통해 트래픽을 라우팅하고 사실상 동일한 계층 2 도메인 내의 다른 네트워크 및 위치에서 시스템을 실행할 수 있습니다.**
- **대부분의 터널 프로토콜과 달리 VXLAN은 지점 간 네트워크일 뿐만 아니라 지점 간 네트워크일 뿐입니다. VXLAN은 다른 끝점의 IP 주소를 동적으로 배우거나 정적으로 구성된 전달 항목을 사용할 수 있습니다.**
- **특정 네트워크 카드에서는 UDP 터널 관련 오프로드 기능을 지원합니다.**

추가 리소스

- `/usr/share/doc/kernel-doc - <kernel_version> /Documentation/networking/vxlan.rst`

11.2. 호스트에서 이더넷 인터페이스 구성

RHEL VM 호스트를 이더넷에 연결하려면 네트워크 연결 프로필을 만들고, IP 설정을 구성하고, 프로필을 활성화합니다.

RHEL 호스트 둘 다에서 이 절차를 실행하고 그에 따라 IP 주소 구성을 조정합니다.

사전 요구 사항

- **호스트가 이더넷 호스트에 연결되어 있습니다.**

절차

1. **NetworkManager에 새 이더넷 연결 프로필을 추가합니다.**

```
# nmcli connection add con-name Example ifname enp1s0 type ethernet
```

2. **IPv4 설정을 구성합니다.**

■


```
# nmcli connection modify Example ipv4.addresses 198.51.100.2/24 ipv4.method
manual ipv4.gateway 198.51.100.254 ipv4.dns 198.51.100.200 ipv4.dns-search
example.com
```

네트워크에서 **DHCP**를 사용하는 경우 이 단계를 건너뛰니다.

3.

예제 연결을 활성화합니다.

```
# nmcli connection up Example
```

검증

1.

장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE    TYPE    STATE    CONNECTION
enp1s0    ethernet connected Example
```

2.

원격 네트워크에서 호스트를 **ping**하여 **IP** 설정을 확인합니다.

```
# ping RHEL-host-B.example.com
```

해당 호스트에 네트워크를 구성하기 전에 다른 **VM** 호스트를 **ping**할 수 없습니다.

추가 리소스

-

nm-settings(5)

11.3. VXLAN이 연결된 네트워크 브리지 만들기

VM(가상 머신)에서 볼 수 없는 **VXLAN**(가상 확장 가능 **LAN**)을 만들려면 호스트에서 브리지를 만들고 **VXLAN**을 브리지에 연결합니다. **NetworkManager**를 사용하여 브리지와 **VXLAN**을 둘 다 만듭니다. 일반적으로 호스트에서 **vnet*** 라는 **VM**의 트래픽 액세스 포인트(**TAP**) 장치를 브리지에 추가하지 않습니다. **libvirt** 서비스는 **VM**이 시작될 때 동적으로 추가합니다.

RHEL 호스트 모두에서 이 절차를 실행하고 그에 따라 **IP** 주소를 조정합니다.

절차

1.

브리지 **br0** 을 만듭니다.

```
# nmcli connection add type bridge con-name br0 ifname br0 ipv4.method disabled
ipv6.method disabled
```

이 명령은 계층 2에서 작동하므로 브리지 장치에 **IPv4** 및 **IPv6** 주소를 설정하지 않습니다.

2.

VXLAN 인터페이스를 만들고 **br0** 에 연결합니다.

```
# nmcli connection add type vxlan slave-type bridge con-name br0-vxlan10 ifname
vxlan10 id 10 local 198.51.100.2 remote 203.0.113.1 master br0
```

이 명령은 다음 설정을 사용합니다.

- **id 10:** **VXLAN** 식별자를 설정합니다.
- **local 198.51.100.2:** 발신 패킷의 소스 **IP** 주소를 설정합니다.
- **remote 203.0.113.1:** **VXLAN** 장치 전달 데이터베이스에서 대상 링크 계층 주소를 알 수 없는 경우 나가는 패킷에 사용할 유니캐스트 또는 멀티캐스트 **IP** 주소를 설정합니다.
- **마스터 br0:** 이 **VXLAN** 연결을 **br0** 연결에서 포트로 만들도록 설정합니다.
- **ipv4.method 비활성화 및 ipv6.method:** 브리지에서 **IPv4** 및 **IPv6**를 비활성화합니다.

기본적으로 **NetworkManager**는 **8472** 를 대상 포트로 사용합니다. 대상 포트가 다른 경우 **destination-port <port_number>** 옵션을 명령에 전달합니다.

3.

br0 연결 프로필을 활성화합니다.

```
# nmcli connection up br0
```

4.

로컬 방화벽에서 들어오는 **UDP** 연결에 대해 포트 **8472** 를 엽니다.

```
# firewall-cmd --permanent --add-port=8472/udp
# firewall-cmd --reload
```

검증

•

전달 테이블을 표시합니다.

```
# bridge fdb show dev vxlan10
2a:53:bd:d5:b3:0a master br0 permanent
00:00:00:00:00:00 dst 203.0.113.1 self permanent
...
```

추가 리소스

•

nm-settings(5)

11.4. 기존 브리지를 사용하여 **LIBVIRT**에서 가상 네트워크 만들기

VM(가상 머신)에서 연결된 **VXLAN(가상 확장 가능 LAN)**이 있는 **br0** 브릿지를 사용하려면 먼저 이 브리지를 사용하는 **libvirtd** 서비스에 가상 네트워크를 추가합니다.

사전 요구 사항

•

libvirt 패키지를 설치했습니다.

•

libvirtd 서비스를 시작하고 활성화했습니다.

•

RHEL에서 **VXLAN**을 사용하여 **br0** 장치를 구성했습니다.

절차

1.

다음 콘텐츠를 사용하여 **~/vxlan10-bridge.xml** 파일을 만듭니다.

```
<network>
  <name>vxlan10-bridge</name>
  <forward mode="bridge" />
```

```
<bridge name="br0" />
</network>
```

2. `~/vxlan10-bridge.xml` 파일을 사용하여 `libvirt` 에 새 가상 네트워크를 만듭니다.

```
# virsh net-define ~/vxlan10-bridge.xml
```

3. `~/vxlan10-bridge.xml` 파일을 제거합니다.

```
# rm ~/vxlan10-bridge.xml
```

4. `rhcos 10-bridge` 가상 네트워크를 시작합니다.

```
# virsh net-start vxlan10-bridge
```

5. `libvirtd` 서비스가 시작될 때 자동으로 시작하도록 `644310-bridge` 가상 네트워크를 구성합니다.

```
# virsh net-autostart vxlan10-bridge
```

검증

- 가상 네트워크 목록을 표시합니다.

```
# virsh net-list
Name           State Autostart Persistent
-----
vxlan10-bridge active yes      yes
...
```

추가 리소스

- `virsh(1)` 도움말 페이지

11.5. VXLAN을 사용하도록 가상 머신 구성

호스트의 **VXLAN**(가상 확장 가능 LAN)이 연결된 브릿지 장치를 사용하도록 VM을 구성하려면 해당 네트워크를 사용하도록 기존 VM을 사용하여 새 VM을 만들거나 기존 VM의 설정을 업데이트합니다.

RHEL 호스트에서 다음 절차를 수행하십시오.

사전 요구 사항

- **libvirt** 에서 **644 310-bridge** 가상 네트워크를 구성했습니다.

절차

- 새 VM을 생성하고 **KnativeServing 10-bridge** 네트워크를 사용하도록 구성하려면 VM을 생성할 때 **--network network:octets10-bridge** 옵션을 **virt-install** 명령에 전달합니다.

```
# virt-install ... --network network:vxlan10-bridge
```

- 기존 VM의 네트워크 설정을 변경하려면 다음을 수행합니다.

- a. VM의 네트워크 인터페이스를 **#161 10-bridge** 가상 네트워크에 연결합니다.

```
# virt-xml VM_name --edit --network network=vxlan10-bridge
```

- b. VM을 종료하고 다시 시작합니다.

```
# virsh shutdown VM_name
# virsh start VM_name
```

검증

1. 호스트에 VM의 가상 네트워크 인터페이스를 표시합니다.

```
# virsh domiflist VM_name
Interface Type Source Model MAC
-----
vnet1 bridge vxlan10-bridge virtio 52:54:00:c5:98:1c
```

2. **dhcp 10-bridge** 브리지에 연결된 인터페이스를 표시합니다.

```
# ip link show master vxlan10-bridge
18: vxlan10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master br0 state UNKNOWN mode DEFAULT group default qlen 1000
```

```
link/ether 2a:53:bd:d5:b3:0a brd ff:ff:ff:ff:ff:ff
19: vnet1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master br0 state UNKNOWN mode DEFAULT group default qlen 1000
link/ether 52:54:00:c5:98:1c brd ff:ff:ff:ff:ff:ff
```

libvirt 서비스는 브리지 구성을 동적으로 업데이트합니다. **KnativeServing 10-bridge** 네트워크를 사용하는 VM을 시작하면 호스트의 해당 **vnet*** 장치가 브리지 포트에 표시됩니다.

3.

ARP(Address resolution protocol) 요청을 사용하여 VM이 동일한 **VXLAN**에 있는지 확인합니다.

a.

동일한 **VXLAN**에서 두 개 이상의 VM을 시작합니다.

b.

하나의 VM에서 **ARP** 요청을 다른 VM으로 보냅니다.

```
# arping -c 1 192.0.2.2
ARPING 192.0.2.2 from 192.0.2.1 enp1s0
Unicast reply from 192.0.2.2 [52:54:00:c5:98:1c] 1.450ms
Sent 1 probe(s) (0 broadcast(s))
Received 1 response(s) (0 request(s), 0 broadcast(s))
```

명령에 **회신**이 표시되면 VM이 동일한 계층 2 도메인에 있고, 이 경우 **VXLAN**입니다.

arping 유틸리티를 사용하도록 **iputils** 패키지를 설치합니다.

추가 리소스

- [virt-install\(1\) 도움말 페이지](#)
- [virt-xml\(1\) 도움말 페이지](#)
- [virsh\(1\) 도움말 페이지](#)
- [arping\(8\) 도움말 페이지](#)

12장. 네트워크 브리지 구성

네트워크 브리지는 **MAC** 주소 테이블을 기반으로 네트워크 간에 트래픽을 전달하는 링크 계층 장치입니다. 브리지는 네트워크 트래픽을 수신 대기하여 **MAC** 주소 테이블을 빌드하여 각 네트워크에 연결된 호스트를 학습합니다. 예를 들어 **Red Hat Enterprise Linux** 호스트의 소프트웨어 브릿지를 사용하여 하드웨어 브릿지 또는 가상화 환경에서 가상 시스템(**VM**)을 호스트와 동일한 네트워크에 통합할 수 있습니다.

브리지에는 브리지가 연결해야 하는 각 네트워크에 네트워크 장치가 필요합니다. 브리지를 구성할 때 브리지는 **controller** 라고 하며 포트를 사용하는 장치입니다.

다음과 같은 다양한 유형의 장치에 브리지를 생성할 수 있습니다.

- 물리적 및 가상 이더넷 장치
- 네트워크 본드
- 네트워크 팀
- VLAN 장치

무선 시간의 효율적인 사용을 위해 **Wi-Fi**에서 3 주소 프레임 사용을 지정하는 **IEEE 802.11** 표준으로 인해 **Ad-Hoc** 또는 **Infrastructure** 모드에서 작동하는 **Wi-Fi** 네트워크를 통한 브리지를 구성할 수 없습니다.

12.1. NMCLI 명령을 사용하여 네트워크 브리지 구성

이 섹션에서는 **nmcli** 유틸리티를 사용하여 네트워크 브리지를 구성하는 방법을 설명합니다.

사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 브리지 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.

- 팀, 본딩 또는 **VLAN** 장치를 브리지 포트로 사용하려면 이러한 장치를 생성하는 동안 브리지를 만들거나 예 설명된 대로 미리 생성할 수 있습니다.
 - **nmcli** 명령을 사용하여 네트워크 팀 구성
 - **nmcli** 명령을 사용하여 네트워크 본딩 구성
 - **nmcli** 명령을 사용하여 **VLAN** 태그 지정 구성

절차

1. 브리지 인터페이스를 만듭니다.

```
# nmcli connection add type bridge con-name bridge0 ifname bridge0
```

이 명령은 **bridge0** 이라는 브리지를 생성하고 다음을 입력합니다.

2. 네트워크 인터페이스를 표시하고 브리지에 추가할 인터페이스의 이름을 확인합니다.

```
# nmcli device status
DEVICE TYPE    STATE    CONNECTION
enp7s0 ethernet disconnected --
enp8s0 ethernet disconnected --
bond0 bond      connected bond0
bond1 bond      connected bond1
...
```

이 예제에서는 다음을 수행합니다.

- **enp7s0** 및 **enp8s0** 은 구성되지 않습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 연결 프로필을 추가합니다.
- **bond0** 및 **bond1** 에는 기존 연결 프로필이 있습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 프로필을 수정합니다.

3.

인터페이스를 브리지에 할당합니다.

a.

브리지에 할당하려는 인터페이스가 구성되지 않은 경우 새 연결 프로필을 생성합니다.

```
# nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
ifname enp7s0 master bridge0
# nmcli connection add type ethernet slave-type bridge con-name bridge0-port2
ifname enp8s0 master bridge0
```

이 명령은 **enp7s0** 및 **enp8s0**에 대한 프로필을 생성하여 **bridge0** 연결에 추가합니다.

b.

기존 연결 프로필을 브리지에 할당하려면 **bridge0**에 이러한 연결의 **master** 매개변수를 설정합니다.

```
# nmcli connection modify bond0 master bridge0
# nmcli connection modify bond1 master bridge0
```

이러한 명령은 **bond0** 및 **bond1**이라는 기존 연결 프로필을 **bridge0** 연결에 할당합니다.

4.

브리지의 IP 설정을 구성합니다. 이 브리지를 다른 장치의 포트로 사용하려면 이 단계를 건너칩니다.

a.

IPv4 설정을 구성합니다. 예를 들어 **bridge0** 연결의 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정하려면 다음을 입력합니다.

```
# nmcli connection modify bridge0 ipv4.addresses '192.0.2.1/24'
# nmcli connection modify bridge0 ipv4.gateway '192.0.2.254'
# nmcli connection modify bridge0 ipv4.dns '192.0.2.253'
# nmcli connection modify bridge0 ipv4.dns-search 'example.com'
# nmcli connection modify bridge0 ipv4.method manual
```

b.

IPv6 설정을 구성합니다. 예를 들어 **bridge0** 연결의 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정하려면 다음을 입력합니다.

```
# nmcli connection modify bridge0 ipv6.addresses '2001:db8:1::1/64'
# nmcli connection modify bridge0 ipv6.gateway '2001:db8:1::fffe'
# nmcli connection modify bridge0 ipv6.dns '2001:db8:1::fffd'
# nmcli connection modify bridge0 ipv6.dns-search 'example.com'
# nmcli connection modify bridge0 ipv6.method manual
```

5.

선택 사항: 브리지의 추가 속성을 구성합니다. 예를 들어 **bridge0**의 스페닝 Tree Protocol(STP) 우선 순위를 **16384**로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify bridge0 bridge.priority '16384'
```

기본적으로 **STP**가 활성화됩니다.

6.

연결을 활성화합니다.

```
# nmcli connection up bridge0
```

7.

포트가 연결되어 있고 **CONNECTION** 열에 포트의 연결 이름이 표시되는지 확인합니다.

```
# nmcli device
DEVICE  TYPE    STATE    CONNECTION
...
enp7s0  ethernet connected bridge0-port1
enp8s0  ethernet connected bridge0-port2
```

연결 포트를 활성화하면 **NetworkManager**도 브리지를 활성화하지만 다른 포트도 활성화되지 않습니다. 브리지가 활성화된 경우 **Red Hat Enterprise Linux**가 모든 포트를 자동으로 활성화하도록 구성할 수 있습니다.

a.

브리지 연결의 **connection.autoconnect-slaves** 매개 변수를 활성화합니다.

```
# nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

b.

브리지를 다시 활성화합니다.

```
# nmcli connection up bridge0
```

검증 단계

•

ip 유틸리티를 사용하여 특정 브릿지의 포트인 이더넷 장치의 링크 상태를 표시합니다.

```
# ip link show master bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master bridge0 state UP mode DEFAULT group default qlen 1000
```

```
link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master bridge0 state UP mode DEFAULT group default qlen 1000
link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```

•

브리지 유틸리티를 사용하여 모든 브리지 장치의 포트인 이더넷 장치의 상태를 표시합니다.

```
# bridge link show
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
listening priority 32 cost 100
5: enp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
forwarding priority 32 cost 100
6: enp11s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1
state blocking priority 32 cost 100
...
```

특정 이더넷 장치의 상태를 표시하려면 **bridge link show dev ethernet_device_name** 명령을 사용합니다.

추가 리소스

•

[기본 네트워크 설정 테스트](#)

•

[기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성](#)

•

[nmcli-examples\(7\) 도움말 페이지](#)

•

[nm- settings\(5\) 도움말 페이지의 브릿지 설정 섹션](#)

•

[nm -settings\(5\) 매뉴얼 페이지의 bridge-port 설정 섹션](#)

•

[bridge\(8\) 도움말 페이지](#)

•

[NetworkManager 서비스를 다시 시작한 후 NetworkManager에서 연결을 중복합니다.](#)

-

[vlan 정보를 사용하여 브리지를 구성하는 방법은 무엇입니까?](#)

12.2. RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성

이 섹션에서는 **RHEL** 웹 콘솔을 사용하여 네트워크 브릿지를 구성하는 방법을 설명합니다.

사전 요구 사항

-

서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.

-

이더넷 장치를 브리지 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.

-

팀, 본딩 또는 **VLAN** 장치를 브리지 포트로 사용하려면 이러한 장치를 생성하는 동안 브리지를 만들거나 예 설명된 대로 미리 생성할 수 있습니다.

-

[RHEL 웹 콘솔을 사용하여 네트워크 팀 구성](#)

-

[RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성](#)

-

[RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 구성](#)

절차

- 1.

화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.

- 2.

Interfaces (interface) 섹션에서 **Add bridge** (브리지 추가)를 클릭합니다.

- 3.

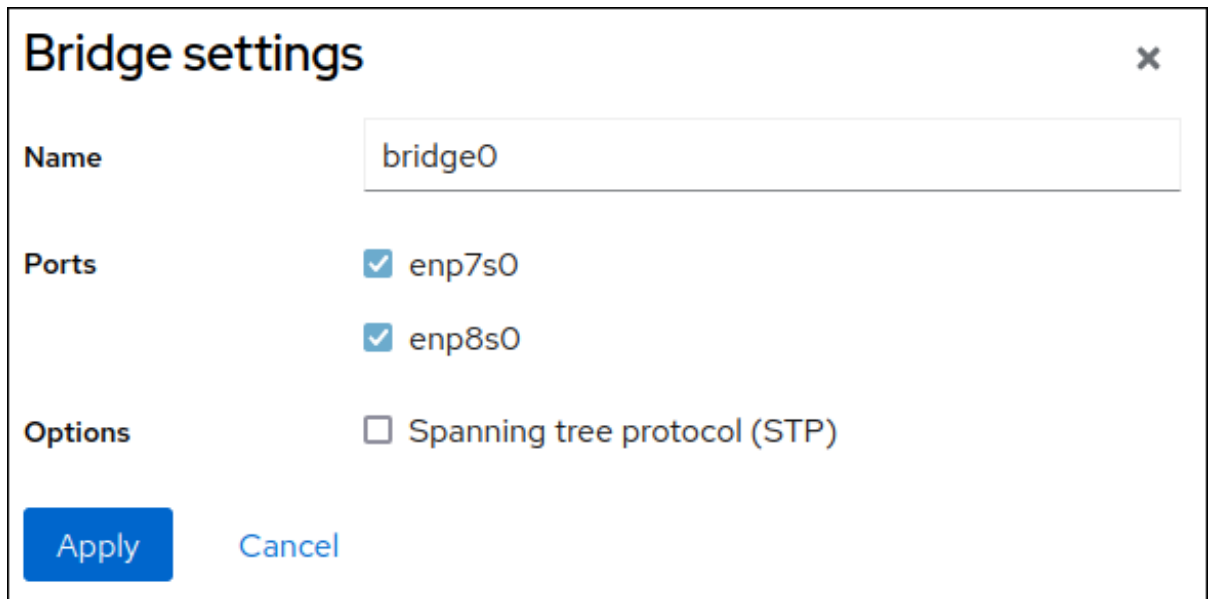
만들 브릿지 장치의 이름을 입력합니다.

- 4.

브리지의 포트여야 하는 인터페이스를 선택합니다.

5.

선택 사항: **Spanning 트리 프로토콜(STP)** 기능을 활성화하여 브리지 루프 및 브로드캐스트 방사능을 방지합니다.



The image shows a 'Bridge settings' dialog box with a close button (X) in the top right corner. It contains three sections: 'Name' with a text input field containing 'bridge0'; 'Ports' with two checked checkboxes for 'enp7s0' and 'enp8s0'; and 'Options' with an unchecked checkbox for 'Spanning tree protocol (STP)'. At the bottom, there are two buttons: 'Apply' (highlighted in blue) and 'Cancel'.

6.

Apply(적용)를 클릭합니다.

7.

기본적으로 브리지는 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

a.

Interfaces (interface) 섹션에서 브리지 이름을 클릭합니다.

b.

구성할 프로토콜 옆에 있는 편집 을 클릭합니다.

c.

주소 옆에 있는 **Manual** 을 선택하고 **IP** 주소, 접두사 및 기본 게이트웨이를 입력합니다.

d.

DNS 섹션에서 **+** 버튼을 클릭하고 **DNS** 서버의 **IP** 주소를 입력합니다. 이 단계를 반복하여 여러 **DNS** 서버를 설정합니다.

e.

DNS 검색 도메인 섹션에서 **+** 버튼을 클릭하고 검색 도메인을 입력합니다.

f.

인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 해당 경로를 구성합니다.

IPv4 settings

Addresses

Manual

+

Address	Prefix length or netmask	Gateway
192.0.2.1	24	192.0.2.254

DNS

Automatic

+

Server

192.0.2.253

-

DNS search domains

Automatic

+

Search domain

example.com

-

Routes

Automatic

+

Apply

Cancel

g.

적용을 클릭합니다.

검증

1.

화면 왼쪽의 탐색에서 **Networking** 탭을 선택하고 인터페이스에 들어오고 나가는 트래픽이 있는지 확인합니다.

Interfaces			
Add bond Add team Add bridge Add VLAN			
Name	IP address	Sending	Receiving
bridge0	192.0.2.1/24	1.11 Mbps	61.2 Mbps

12.3. NM-CONNECTION-EDITOR를 사용하여 네트워크 브릿지 구성

이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 브리지를 구성하는 방법을 설명합니다.

nm-connection-editor 는 브리지에 새 포트만 추가할 수 있습니다. 기존 연결 프로필을 포트에 사용하려면 **nmcli** 명령을 사용하여 네트워크 브릿지 구성에 설명된 대로 **nmcli** 유틸리티를 사용하여 브리지를 생성합니다.

사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 브리지 포트에 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 본딩 또는 **VLAN** 장치를 브리지 포트에 사용하려면 해당 장치가 아직 구성되지 않았는지 확인합니다.

절차

1. 터미널을 열고 **nm-connection-editor** 를 입력합니다.

```
$ nm-connection-editor
```

2. **+** 버튼을 클릭하여 새 연결을 추가합니다.

3. 브리지 연결 유형을 선택하고 만들기 를 클릭합니다.

4. 브리지 탭에서 다음을 수행합니다.

- a. 선택 사항: 인터페이스 이름 필드에 브리지 인터페이스의 이름을 설정합니다.

- b. **Add(추가)** 버튼을 클릭하여 네트워크 인터페이스의 새 연결 프로필을 만들고 프로필을 브리지에 포트에 추가합니다.

- i. 인터페이스의 연결 유형을 선택합니다. 예를 들어 유선 연결을 위해 이더넷 을 선택합니다.

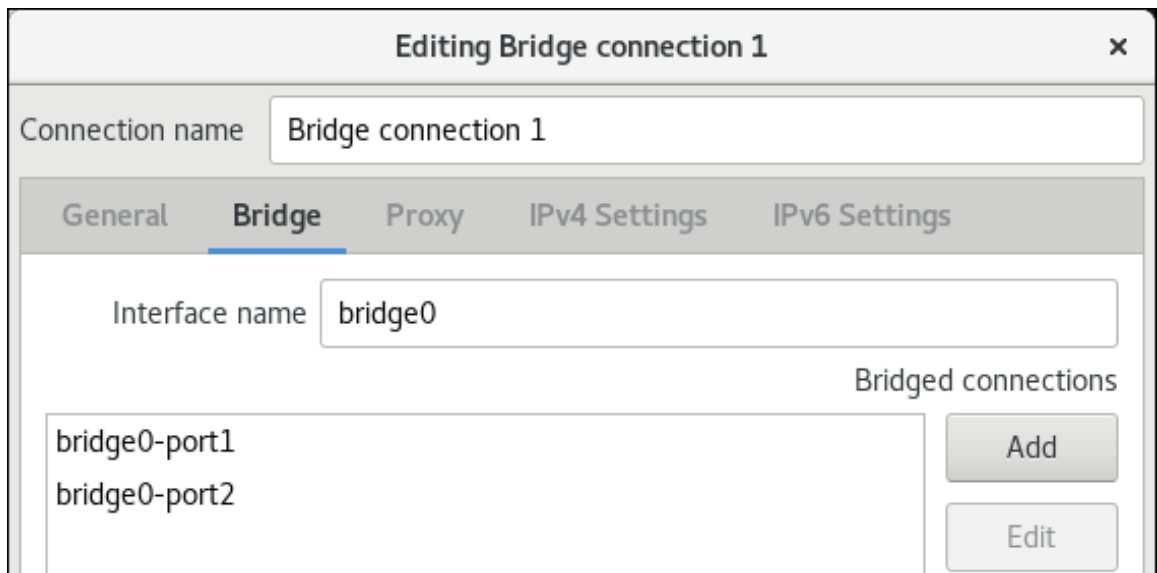
- ii.

선택적으로 포트 장치의 연결 이름을 설정합니다.
- iii.

이더넷 장치에 대한 연결 프로필을 생성하는 경우 이더넷 탭을 열고 **Device (장치)** 필드에서 선택하고, 브릿지에 포트에 추가할 네트워크 인터페이스를 선택합니다. 다른 장치 유형을 선택한 경우 적절하게 구성합니다.
- iv.

저장을 클릭합니다.
- c.

브릿지에 추가할 각 인터페이스에 대해 이전 단계를 반복합니다.



5.

선택 사항: **STP(Spanning Tree Protocol)** 옵션과 같은 추가 브리지 설정을 구성합니다.
6.

브리지의 **IP** 설정을 구성합니다. 이 브리지를 다른 장치의 포트에 사용하려면 이 단계를 건너뛹니다.

 - a.

IPv4 Settings 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

Editing Bridge connection 1 ✕

Connection name

General Bridge Proxy **IPv4 Settings** IPv6 Settings

Method Manual ▾

Addresses

Address	Netmask	Gateway
192.0.2.1	24	192.0.2.254

Add
Delete

DNS servers

Search domains

b.

IPv6 Settings 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

Editing Bridge connection 1 ✕

Connection name

General Bridge Proxy IPv4 Settings **IPv6 Settings**

Method Manual ▾

Addresses

Address	Prefix	Gateway
2001:db8:1::1	64	2001:db8:1::fff3

Add
Delete

DNS servers

Search domains

7. *브리지 연결을 저장합니다.*
8. ***nm-connection-editor** 를 닫습니다.*

검증 단계

- *ip* 유틸리티를 사용하여 특정 브리지의 포트인 이더넷 장치의 링크 상태를 표시합니다.
- ```
ip link show master bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```
- *브리지 유틸리티를 사용하여 모든 브리지 장치에 포트인 이더넷 장치의 상태를 표시합니다.*

```
bridge link show
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
listening priority 32 cost 100
5: enp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
forwarding priority 32 cost 100
6: enp11s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1
state blocking priority 32 cost 100
...
```

특정 이더넷 장치의 상태를 표시하려면 **bridge link show dev ethernet\_device\_name** 명령을 사용합니다.

#### 추가 리소스

- [nm-connection-editor를 사용하여 네트워크 본딩 구성](#)
- [nm-connection-editor를 사용하여 네트워크 팀 구성](#)

- ***nm-connection-editor를 사용하여 VLAN 태그 구성***
- ***기본 네트워크 설정 테스트***
- ***기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성***
- ***vlan 정보를 사용하여 브리지를 구성하는 방법은 무엇입니까?***

#### 12.4. NMSTATECTL을 사용하여 네트워크 브릿지 구성

이 섹션에서는 **nmstatectl** 유틸리티를 사용하여 다음 설정으로 **Linux** 네트워크 브리지 **bridge0** 을 구성하는 방법을 설명합니다.

- ***브리지의 네트워크 인터페이스인 enp1s0 및 enp7s0***
- ***STP(Spanning Tree Protocol): 활성화됨***
- ***정적 IPv4 주소: /24 서브넷 마스크가 있는 192.0.2.1***
- ***정적 IPv6 주소: 2001:db8:1::1 및 /64 서브넷 마스크***
- ***IPv4 기본 게이트웨이: 192.0.2.254***
- ***IPv6 기본 게이트웨이: 2001:db8:1::fffe***
- ***IPv4 DNS 서버: 192.0.2.200***
- ***IPv6 DNS 서버: 2001:db8:1::ffbb***

- **DNS 검색 도메인: `example.com`**

#### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 브리지의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 본딩 또는 **VLAN** 장치를 브리지에서 포트로 사용하려면 포트 목록에서 인터페이스 이름을 설정하고 해당 인터페이스를 정의합니다.
- **nmstate** 패키지가 설치되어 있습니다.

#### 절차

1. 다음 콘텐츠를 사용하여 **YAML** 파일(예: `~/create-bridge.yml`)을 만듭니다.

```

interfaces:
- name: bridge0
 type: linux-bridge
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
 bridge:
 options:
 stp:
 enabled: true
 port:
 - name: enp1s0
 - name: enp7s0
- name: enp1s0
```

```

type: ethernet
state: up
- name: enp7s0
 type: ethernet
 state: up

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: bridge0
 - destination: ::0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: bridge0
dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb

```

2.

시스템에 설정을 적용합니다.

```
nmstatectl apply ~/create-bridge.yml
```

## 검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
nmcli device status
DEVICE TYPE STATE CONNECTION
bridge0 bridge connected bridge0
```

2.

연결 프로파일의 모든 설정을 표시합니다.

```
nmcli connection show bridge0
connection.id: bridge0
connection.uuid: e2cc9206-75a2-4622-89cf-1252926060a9
connection.stable-id: --
connection.type: bridge
connection.interface-name: bridge0
...
```

3.

연결 설정을 **YAML** 형식으로 표시합니다.

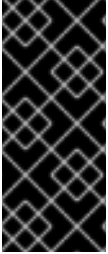
**# nmstatectl show bridge0**

추가 리소스

- ***nmstatectl(8) 도움말 페이지***
- ***/usr/share/doc/nmstate/examples/***
- ***vlan 정보를 사용하여 브리지를 구성하는 방법은 무엇입니까?***

## 13장. 네트워크 티밍 구성

이 섹션에서는 네트워크 티밍의 기초, 본딩과 티밍의 차이점, **Red Hat Enterprise Linux**에서 네트워크 팀을 구성하는 방법에 대해 설명합니다.



### 중요

네트워크 티밍은 **Red Hat Enterprise Linux 9**에서 더 이상 사용되지 않습니다. 서버를 **RHEL**의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 [네트워크 본딩 구성](#)을 참조하십시오.

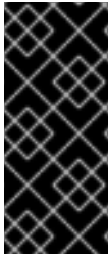
다음과 같은 다양한 유형의 장치에서 네트워크 팀을 생성할 수 있습니다.

- 물리적 및 가상 이더넷 장치
- 네트워크 본드
- 네트워크 브리지
- VLAN 장치

### 13.1. 네트워크 티밍 이해

네트워크 티밍은 네트워크 인터페이스를 결합하거나 집계하여 더 높은 처리량 또는 중복성을 갖춘 논리적 인터페이스를 제공하는 기능입니다.

네트워크 티밍은 커널 드라이버를 사용하여 패킷 흐름의 빠른 처리와 기타 작업을 위한 사용자 공간 라이브러리 및 서비스를 구현합니다. 이렇게 하면 네트워크 티밍이 부하 분산 및 중복 요구 사항을 위한 확장 가능하고 확장 가능한 솔루션입니다.



## 중요

장애 조치 메커니즘과 같은 특정 네트워크 터밍 기능은 네트워크 스위치 없이 직접 케이블 연결을 지원하지 않습니다. 자세한 내용은 [교차 케이블 사용으로 직접 연결에서 본딩이 지원됩니까?](#)

### 13.2. 컨트롤러 및 포트 인터페이스의 기본 동작 이해

**NetworkManager** 서비스를 사용하여 팀 또는 본딩 포트 인터페이스를 관리하거나 해결할 때 다음과 같은 기본 동작을 고려하십시오.

- 컨트롤러 인터페이스를 시작하면 포트 인터페이스가 자동으로 시작되지 않습니다.
- 포트 인터페이스를 시작하면 항상 컨트롤러 인터페이스가 시작됩니다.
- 컨트롤러 인터페이스를 중지하면 포트 인터페이스도 중지됩니다.
- 포트가 없는 컨트롤러는 정적 IP 연결을 시작할 수 있습니다.
- 포트 없는 컨트롤러는 **DHCP** 연결을 시작할 때 포트를 대기합니다.
- 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어로 포트를 추가하면 완료됩니다.
- 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어 없이 포트를 추가할 때 계속 대기합니다.

### 13.3. 네트워크 터밍 및 본딩 기능 비교

네트워크 팀 및 네트워크 본딩에서 지원되는 기능에 대해 알아보십시오.

| 기능           | 네트워크 본딩 | 네트워크 팀 |
|--------------|---------|--------|
| 브로드캐스트 Tx 정책 | 있음      | 있음     |



| 기능                        | 네트워크 본딩      | 네트워크 팀  |
|---------------------------|--------------|---------|
| 라운드 로빈 Tx 정책              | 있음           | 있음      |
| active-backup Tx 정책       | 있음           | 있음      |
| LACP (802.3ad) 지원         | 예 (활성 전용)    | 있음      |
| 해시 기반 Tx 정책               | 있음           | 있음      |
| 사용자는 해시 함수를 설정할 수 있습니다    | 없음           | 있음      |
| TX 로드 밸런싱 지원(TLB)         | 있음           | 있음      |
| LACP 해시 포트 선택             | 있음           | 있음      |
| LACP 지원에 대한 로드 밸런싱        | 없음           | 있음      |
| ethtool 링크 모니터링           | 있음           | 있음      |
| ARP 링크 모니터링               | 있음           | 있음      |
| NS/NA (IPv6) 링크 모니터링      | 없음           | 있음      |
| 포트 업/다운 지연                | 있음           | 있음      |
| 포트 우선 순위 및 고정 (기본" 옵션 개선) | 없음           | 있음      |
| 개별 포트별 링크 모니터링 설정         | 없음           | 있음      |
| 여러 링크 모니터링 설정             | 제한됨          | 있음      |
| 잠금 없는 Tx/Rx 경로            | 아니요 (rwlock) | 예 (RCU) |
| VLAN 지원                   | 있음           | 있음      |
| 사용자 공간 런타임 제어             | 제한됨          | 있음      |
| 사용자 공간의 논리                | 없음           | 있음      |
| 확장성                       | 하드           | 쉬움      |
| 모듈식 설계                    | 없음           | 있음      |
| 성능 오버헤드                   | 낮음           | 매우 낮음   |

| 기능                | 네트워크 본딩 | 네트워크 팀 |
|-------------------|---------|--------|
| D-Bus 인터페이스       | 없음      | 있음     |
| 다중 장치 스택          | 있음      | 있음     |
| LLDP를 사용하는 제로 구성  | 없음      | (계획 중) |
| NetworkManager 지원 | 있음      | 있음     |

#### 13.4. TEAMD 서비스, 러너 및 링크-감시자 이해

팀 서비스인 **teamd** 는 팀 드라이버의 인스턴스 한 개를 제어합니다. 이 드라이버 인스턴스는 하드웨어 장치 드라이버의 인스턴스를 추가하여 네트워크 인터페이스 팀을 구성합니다. 팀 드라이버는 네트워크 인터페이스(예: **team0** )를 커널에 제공합니다.

**teamd** 서비스는 모든 터밍 방법에 대한 공통 논리를 구현합니다. 이러한 함수는 라운드 로빈과 같은 다양한 로드 공유 및 백업 방법에 고유하며 러너라고도 하는 별도의 코드 단위로 구현됩니다. 관리자는 **JavaScript Object Notation (JSON)** 형식으로 러너를 지정하고, 인스턴스를 생성할 때 **JSON** 코드가 **teamd** 인스턴스로 컴파일됩니다. 또는 **NetworkManager** 를 사용하는 경우 **team.runner** 매개변수에서 러너를 설정하고 **NetworkManager** 는 해당 **JSON** 코드를 자동으로 만들 수 있습니다.

사용 가능한 러너는 다음과 같습니다.

- **broadcast:** 모든 포트에서 데이터를 전송합니다.
- **roundrobin:** 는 모든 포트에서 차례로 데이터를 전송합니다.
- **activebackup:** 다른 포트는 백업으로 유지되는 동안 하나의 포트에서 데이터를 전송합니다.
- **loadbalance:** 활성 Tx 부하 분산 및 **BPF(Berkeley Packet Filter)** 기반 Tx 포트 선택기를 사용하여 모든 포트에서 데이터를 전송합니다.
- **랜덤:** 임의로 선택한 포트에서 데이터를 전송합니다.

•

**lACP: 802.3ad LACP**(링크 집계 제어 프로토콜)를 구현합니다.

**teamd** 서비스는 링크 감시자를 사용하여 하위 장치의 상태를 모니터링합니다. 다음 **link-watchers**를 사용할 수 있습니다.

•

**ethtool: libteam** 라이브러리는 **ethtool** 유틸리티를 사용하여 링크 상태 변경을 감시합니다. 기본 **link-watcher**입니다.

•

**arp\_ping: libteam** 라이브러리는 **arp\_ping** 유틸리티를 사용하여 **Address Resolution Protocol(ARP)**를 사용하여 **far-end** 하드웨어 주소의 존재를 모니터링합니다.

•

**nsna\_ping: IPv6** 연결에서 **libteam** 라이브러리는 **IPv6 Neighbor** 검색 프로토콜의 **Neighbor Advertisement** 및 **Neighbor Solicitation** 기능을 사용하여 이웃 인터페이스의 존재를 모니터링합니다.

각 러너는 **lACP**를 제외하고 모든 링크 감시자를 사용할 수 있습니다. 이 러너는 **ethtool** 링크 감시자만 사용할 수 있습니다.

### 13.5. TEAMD 서비스 설치

**NetworkManager**에서 네트워크 팀을 구성하려면 **NetworkManager**의 **teamd** 서비스 및 팀 플러그인이 필요합니다. 둘 다 기본적으로 **Red Hat Enterprise Linux**에 설치됩니다. 이 섹션에서는 패키지를 제거하는 경우 필수 패키지를 설치하는 방법에 대해 설명합니다.

#### 사전 요구 사항

•

활성 **Red Hat** 서브스크립션이 호스트에 할당됩니다.

#### 절차

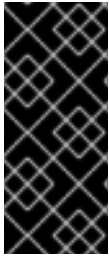
•

**teamd** 및 **NetworkManager-team** 패키지를 설치합니다.

```
yum install teamd NetworkManager-team
```

### 13.6. NMCLI 명령을 사용하여 네트워크 팀 구성

이 섹션에서는 **nmcli** 유틸리티를 사용하여 네트워크 팀을 구성하는 방법을 설명합니다.



#### 중요

네트워크 팀은 **Red Hat Enterprise Linux 9**에서 더 이상 사용되지 않습니다. 서버를 **RHEL**의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 [네트워크 본딩 구성](#) 을 참조하십시오.

#### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 팀의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치하고 스위치에 연결해야 합니다.
- 본딩, 브리지 또는 **VLAN** 장치를 팀의 포트로 사용하려면 팀을 생성하는 동안 이러한 장치를 만들거나 예 설명된 대로 미리 생성할 수 있습니다.
  - [nmcli 명령을 사용하여 네트워크 본딩 구성](#)
  - [nmcli 명령을 사용하여 네트워크 브리지 구성](#)
  - [nmcli 명령을 사용하여 VLAN 태그 지정 구성](#)

#### 절차

1. 팀 인터페이스를 생성합니다.

```
nmcli connection add type team con-name team0 ifname team0 team.runner activebackup
```

이 명령은 **activebackup** 러너를 사용하는 **team0** 이라는 네트워크 팀을 생성합니다.

2. 선택적으로 링크 감시자를 설정합니다. 예를 들어 **team0** 연결 프로필에서 **ethtool** 링크 감시자를 설정하려면 다음을 수행합니다.

```
nmcli connection modify team0 team.link-watchers "name=ethtool"
```

링크 감시자는 다양한 매개 변수를 지원합니다. 링크 감시자의 매개 변수를 설정하려면 이름 속성에 공백을 구분하여 지정합니다. **name** 속성은 따옴표로 묶어야 합니다. 예를 들어 **ethtool** 링크 감시자를 사용하고 **delay-up** 매개 변수를 2500 밀리초 (2.5 초)로 설정하려면 다음을 수행하십시오.

```
nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2500"
```

여러 링크 감시자와 각 링크 감시자를 특정 매개 변수로 설정하려면 링크 감시자를 쉼표로 구분해야 합니다. 다음 예제에서는 **delay-up** 매개 변수와 **source-host** 및 **target-host** 매개 변수를 사용하여 **arp\_ping** 링크 감시자를 사용하여 **ethtool** 링크 감시자를 설정합니다.

```
nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2,
name=arp_ping source-host=192.0.2.1 target-host=192.0.2.2"
```

3.

네트워크 인터페이스를 표시하고 팀에 추가할 인터페이스의 이름을 확인합니다.

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp7s0 ethernet disconnected --
enp8s0 ethernet disconnected --
bond0 bond connected bond0
bond1 bond connected bond1
...
```

이 예제에서는 다음을 수행합니다.

- 

**enp7s0** 및 **enp8s0** 은 구성되지 않습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 연결 프로필을 추가합니다. 연결에 할당되지 않은 팀의 이더넷 인터페이스만 사용할 수 있습니다.

- 

**bond0** 및 **bond1** 에는 기존 연결 프로필이 있습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 프로필을 수정합니다.

4.

포트 인터페이스를 팀에 할당합니다.

a.

팀에 할당하려는 인터페이스가 구성되지 않은 경우 해당 인터페이스에 대한 새 연결 프

로필을 생성합니다.

```
nmcli connection add type ethernet slave-type team con-name team0-port1
nmcli connection add type ethernet slave-type team con-name team0-port2
```

. 이러한 명령은 **enp7s0** 및 **enp8s0** 에 대한 프로필을 생성하여 **team0** 연결에 추가합니다.

b.

팀에 기존 연결 프로필을 할당하려면 **team0** 에 이러한 연결의 **master** 매개변수를 설정합니다.

```
nmcli connection modify bond0 master team0
nmcli connection modify bond1 master team0
```

이러한 명령은 **bond0** 및 **bond1** 이라는 기존 연결 프로필을 **team0** 연결에 할당합니다.

5.

팀의 **IP** 설정을 구성합니다. 이 팀을 다른 장치의 포트로 사용하려면 이 단계를 건너뛰니다.

a.

**IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 **team0** 연결에 설정하려면 다음을 입력합니다.

```
nmcli connection modify team0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify team0 ipv4.gateway '192.0.2.254'
nmcli connection modify team0 ipv4.dns '192.0.2.253'
nmcli connection modify team0 ipv4.dns-search 'example.com'
nmcli connection modify team0 ipv4.method manual
```

b.

**IPv6** 설정을 구성합니다. 예를 들어 **team0** 연결의 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정하려면 다음을 입력합니다.

```
nmcli connection modify team0 ipv6.addresses '2001:db8:1::1/64'
nmcli connection modify team0 ipv6.gateway '2001:db8:1::fffe'
nmcli connection modify team0 ipv6.dns '2001:db8:1::fffd'
nmcli connection modify team0 ipv6.dns-search 'example.com'
nmcli connection modify team0 ipv6.method manual
```

6.

연결을 활성화합니다.

```
nmcli connection up team0
```

### 검증 단계

- 팀의 상태를 표시합니다.

```
teamdctl team0 state
setup:
 runner: activebackup
ports:
 enp7s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
 enp8s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
runner:
 active port: enp7s0
```

이 예제에서는 두 포트가 모두 **up**입니다.

### 추가 리소스

- [기본 네트워크 설정 테스트](#)
- [기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성](#)
- [teamd 서비스, 러너 및 link-watchers 이해](#)
- [nmcli-examples\(7\) 도움말 페이지](#)
- [nm-settings\(5\) 매뉴얼 페이지의 team 섹션](#)

- **teamd.conf(5) man page**

### 13.7. RHEL 웹 콘솔을 사용하여 네트워크 팀 구성

이 섹션에서는 RHEL 웹 콘솔을 사용하여 네트워크 팀을 구성하는 방법을 설명합니다.



#### 중요

네트워크 팀은 **Red Hat Enterprise Linux 9**에서 더 이상 사용되지 않습니다. 서버를 **RHEL**의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 [네트워크 본딩 구성](#)을 참조하십시오.

#### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 팀의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치하고 스위치에 연결해야 합니다.
- 본딩, 브리지 또는 **VLAN** 장치를 팀의 포트로 사용하려면 다음에 설명된 대로 미리 생성합니다.
  - [RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성](#)
  - [RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성](#)
  - [RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 구성](#)

#### 절차

1. 화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.
2. 인터페이스 섹션에서 팀 추가를 클릭합니다.



3.        생성할 팀 장치의 이름을 입력합니다.

4.        팀의 포트여야 하는 인터페이스를 선택합니다.

5.        팀의 러너를 선택합니다.

로드 밸런싱 또는 **802.3ad LACP** 를 선택하면 웹 콘솔에 추가 필드 밸런서 가 표시됩니다.

6.        링크 감시자를 설정합니다.

- **Ethtool** 을 선택하는 경우 링크를 설정하고 지연을 연결하십시오.
- **ARP ping** 또는 **NSNA ping** 을 설정하는 경우 **ping** 간격 및 **ping** 대상을 설정합니다.

## Team settings ✕

Name

team0

Ports

☒ enp7s0  
☒ enp8s0

Runner

Active backup ▼

Link watch

Ethtool ▼

Link up delay

0

Link down delay

0

Apply

Cancel

7.

**Apply(적용)**를 클릭합니다.

8.

기본적으로 팀은 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

a.

**Interfaces (interface)** 섹션에서 팀의 이름을 클릭합니다.

b.

구성할 프로토콜 옆에 있는 **편집** 을 클릭합니다.

c.

주소 옆에 있는 **Manual** 을 선택하고 **IP** 주소, 접두사 및 기본 게이트웨이를 입력합니다.

d.

**DNS** 섹션에서 **+** 버튼을 클릭하고 **DNS** 서버의 **IP** 주소를 입력합니다. 이 단계를 반복하여 여러 **DNS** 서버를 설정합니다.

e.

**DNS 검색 도메인** 섹션에서 **+** 버튼을 클릭하고 검색 도메인을 입력합니다.

f.

인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 해당 경로를 구성합니다.

IPv4 settings
×

Addresses
Manual
+

| Address   | Prefix length or netmask | Gateway     |
|-----------|--------------------------|-------------|
| 192.0.2.1 | 24                       | 192.0.2.254 |

DNS
Automatic
+

Server
192.0.2.253
-

DNS search domains
Automatic
+

Search domain
example.com
-

Routes
Automatic
+

Apply Cancel

g.

적용을 클릭합니다.

## 검증

1.

화면 왼쪽의 탐색에서 **Networking** 탭을 선택하고 인터페이스에 들어오고 나가는 트래픽이 있는지 확인합니다.

| Interfaces                                         |              |           |           |
|----------------------------------------------------|--------------|-----------|-----------|
| <div> Add bond Add team Add bridge Add VLAN </div> |              |           |           |
| Name                                               | IP address   | Sending   | Receiving |
| team0                                              | 192.0.2.1/24 | 1.11 Mbps | 61.2 Mbps |

2.

팀의 상태를 표시합니다.

```
teamdctl team0 state
setup:
 runner: activebackup
ports:
 enp7s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
 enp8s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
runner:
 active port: enp7s0
```

이 예제에서는 두 포트가 모두 **up**입니다.

추가 리소스

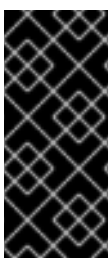


네트워크 팀 러너

### 13.8. NM-CONNECTION-EDITOR를 사용하여 네트워크 팀 구성

이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 팀을 구성하는 방법을 설명합니다.

**nm-connection-editor** 는 팀에 새 포트만 추가할 수 있습니다. 기존 연결 프로필을 포트에 사용하려면 **nmcli** 명령을 사용하여 네트워크 팀 구성에 설명된 대로 **nmcli** 유틸리티를 사용하여 팀을 생성합니다.



중요

네트워크 팀은 **Red Hat Enterprise Linux 9**에서 더 이상 사용되지 않습니다. 서버를 **RHEL**의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 [네트워크 본딩 구성](#) 을 참조하십시오.

## 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 팀의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 본딩 또는 **VLAN** 장치를 팀의 포트로 사용하려면 해당 장치가 아직 구성되지 않았는지 확인합니다.

## 절차

1. 터미널을 열고 **nm-connection-editor** 를 입력합니다.

**\$ nm-connection-editor**

2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. 팀 연결 유형을 선택하고 생성을 클릭합니다.
4. 팀 탭에서 다음을 수행합니다.
  - a. 선택 사항: 인터페이스 이름 필드에 팀 인터페이스의 이름을 설정합니다.
  - b. **Add(추가)** 버튼을 클릭하여 네트워크 인터페이스의 새 연결 프로필을 추가하고 프로필을 팀에 포트로 추가합니다.
    - i. 인터페이스의 연결 유형을 선택합니다. 예를 들어 유선 연결을 위해 이더넷 을 선택합니다.
    - ii. 선택 사항: 포트에 대한 연결 이름을 설정합니다.
    - iii. 이더넷 장치에 대한 연결 프로필을 생성하는 경우 이더넷 탭을 열고 장치 필드에서 팀에 포트로 추가할 네트워크 인터페이스를 선택합니다. 다른 장치 유형을 선택한 경우

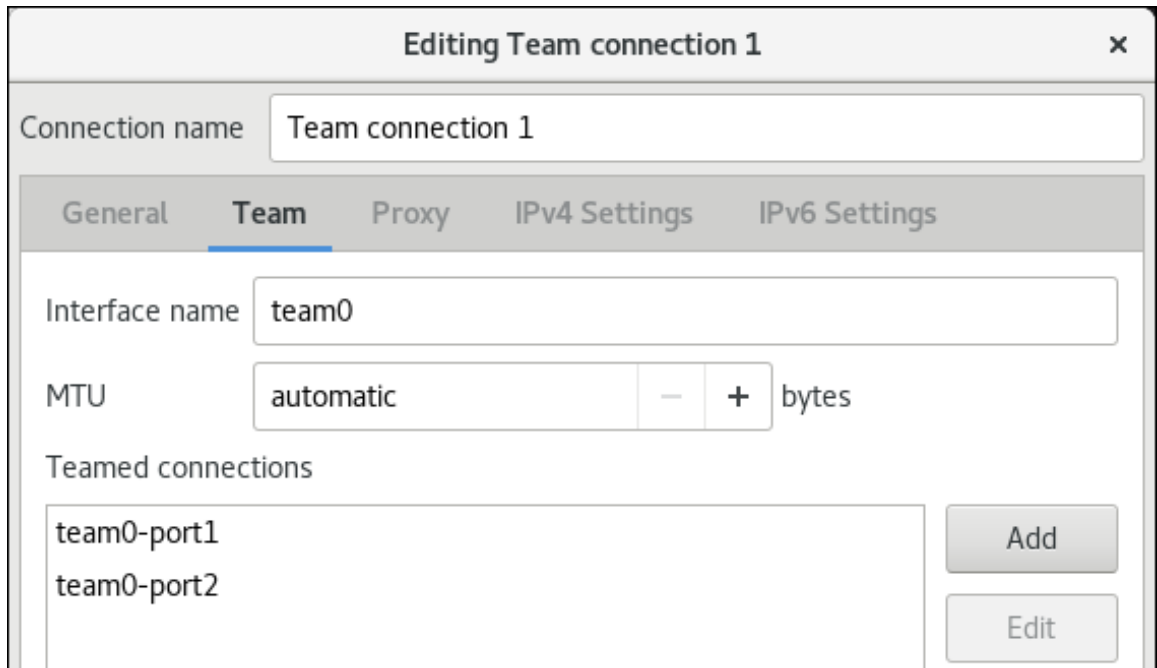
적절하게 구성합니다. 연결에 할당되지 않은 팀의 이더넷 인터페이스만 사용할 수 있습니다.

iv.

저장을 클릭합니다.

c.

팀에 추가할 각 인터페이스에 대해 이전 단계를 반복합니다.



d.

**Advanced(고급)** 버튼을 클릭하여 고급 옵션을 팀 연결에 설정합니다.

i.

**Runner** 탭에서 러너를 선택합니다.

ii.

**Link Watcher** 탭에서 링크 감시자 및 선택적 설정을 설정합니다.

iii.

**OK(확인)**를 클릭합니다.

5.

팀의 **IP** 설정을 구성합니다. 이 팀을 다른 장치의 포트로 사용하려면 이 단계를 건너뛰니다.

a.

**IPv4 Settings** 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마

스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

Editing Team connection 1

Connection name Team connection 1

General Team Proxy **IPv4 Settings** IPv6 Settings

Method Manual

**Addresses**

| Address   | Netmask | Gateway     |
|-----------|---------|-------------|
| 192.0.2.1 | 24      | 192.0.2.254 |

Add

Delete

DNS servers 192.0.2.253

Search domains example.com

b.

**IPv6 Settings** 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

Editing Team connection 1

Connection name: Team connection 1

General Team Proxy IPv4 Settings **IPv6 Settings**

Method: Manual

Addresses

| Address       | Prefix | Gateway          |
|---------------|--------|------------------|
| 2001:db8:1::1 | 64     | 2001:db8:1::fff3 |

Add Delete

DNS servers: 2001:db8:1::fffd

Search domains: example.com

6. 팀 연결을 저장합니다.
7. ***nm-connection-editor*** 를 닫습니다.

#### 검증 단계

- 팀의 상태를 표시합니다.

```
teamdctl team0 state
setup:
 runner: activebackup
ports:
 enp7s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
 enp8s0
 link watches:
 link summary: up
 instance[link_watch_0]:
```



```
name: ethtool
link: up
down count: 0
runner:
active port: enp7s0
```

#### 추가 리소스

- [nm-connection-editor](#)를 사용하여 네트워크 본딩 구성
- [nm-connection-editor](#)를 사용하여 네트워크 팀 구성
- [nm-connection-editor](#)를 사용하여 VLAN 태그 구성
- [기본 네트워크 설정 테스트](#)
- [기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성](#)
- [teamd 서비스, 러너 및 link-watchers 이해](#)
- [NetworkManager 서비스를 다시 시작한 후 NetworkManager에서 연결을 중복합니다.](#)

## 14장. 네트워크 본딩 구성

이 섹션에서는 네트워크 본딩의 기초, 본딩과 터밍의 차이점, Red Hat Enterprise Linux에서 네트워크 본딩을 구성하는 방법에 대해 설명합니다.

다음과 같은 다양한 유형의 장치에 본딩을 생성할 수 있습니다.

- 물리적 및 가상 이더넷 장치
- 네트워크 브리지
- 네트워크 팀
- VLAN 장치

### 14.1. 네트워크 본딩 이해

네트워크 본딩은 네트워크 인터페이스를 결합하거나 집계하여 더 높은 처리량 또는 중복성을 갖춘 논리적 인터페이스를 제공하는 방법입니다.

**active-backup**, **balance-tlb** 및 **balance-alb** 모드에는 네트워크 스위치의 특정 구성이 필요하지 않습니다. 그러나 다른 본딩 모드에서는 링크를 집계하도록 스위치를 구성해야 합니다. 예를 들어, Cisco 스위치는 0, 2 및 3 모드의 경우, 4 모드인 경우 LACP(Link Aggregation Control Protocol) 및 InstallPlan Channel 이 필요합니다.

자세한 내용은 스위치 및 [Linux 이더넷 본딩 드라이버 HOWTO 설명서](#)를 참조하십시오.

#### 중요

장애 조치 메커니즘과 같은 특정 네트워크 본딩 기능은 네트워크 스위치 없이 직접 케이블 연결을 지원하지 않습니다. 자세한 내용은 교차 케이블 사용으로 직접 연결에서 지원되는 **ls bonding**을 참조하십시오. <https://access.redhat.com/solutions/202583> KCS 솔루션.

## 14.2. 컨트롤러 및 포트 인터페이스의 기본 동작 이해

**NetworkManager** 서비스를 사용하여 팀 또는 본딩 포트 인터페이스를 관리하거나 해결할 때 다음과 같은 기본 동작을 고려하십시오.

- 컨트롤러 인터페이스를 시작하면 포트 인터페이스가 자동으로 시작되지 않습니다.
- 포트 인터페이스를 시작하면 항상 컨트롤러 인터페이스가 시작됩니다.
- 컨트롤러 인터페이스를 중지하면 포트 인터페이스도 중지됩니다.
- 포트가 없는 컨트롤러는 정적 IP 연결을 시작할 수 있습니다.
- 포트 없는 컨트롤러는 **DHCP** 연결을 시작할 때 포트를 대기합니다.
- 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어로 포트를 추가하면 완료됩니다.
- 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어 없이 포트를 추가할 때 계속 대기합니다.

## 14.3. 네트워크 팀 및 본딩 기능 비교

네트워크 팀 및 네트워크 본딩에서 지원되는 기능에 대해 알아보십시오.

| 기능                  | 네트워크 본딩   | 네트워크 팀 |
|---------------------|-----------|--------|
| 브로드캐스트 Tx 정책        | 있음        | 있음     |
| 라운드 로빈 Tx 정책        | 있음        | 있음     |
| active-backup Tx 정책 | 있음        | 있음     |
| LACP (802.3ad) 지원   | 예 (활성 전용) | 있음     |
| 해시 기반 Tx 정책         | 있음        | 있음     |

| 기능                        | 네트워크 본딩      | 네트워크 팀  |
|---------------------------|--------------|---------|
| 사용자는 해시 함수를 설정할 수 있습니다    | 없음           | 있음      |
| TX 로드 밸런싱 지원(TLB)         | 있음           | 있음      |
| LACP 해시 포트 선택             | 있음           | 있음      |
| LACP 지원에 대한 로드 밸런싱        | 없음           | 있음      |
| ethtool 링크 모니터링           | 있음           | 있음      |
| ARP 링크 모니터링               | 있음           | 있음      |
| NS/NA (IPv6) 링크 모니터링      | 없음           | 있음      |
| 포트 업/다운 지연                | 있음           | 있음      |
| 포트 우선 순위 및 고정 (기본" 옵션 개선) | 없음           | 있음      |
| 개별 포트별 링크 모니터링 설정         | 없음           | 있음      |
| 여러 링크 모니터링 설정             | 제한됨          | 있음      |
| 잠금 없는 Tx/Rx 경로            | 아니요 (rwlock) | 예 (RCU) |
| VLAN 지원                   | 있음           | 있음      |
| 사용자 공간 런타임 제어             | 제한됨          | 있음      |
| 사용자 공간의 논리                | 없음           | 있음      |
| 확장성                       | 하드           | 쉬움      |
| 모듈식 설계                    | 없음           | 있음      |
| 성능 오버헤드                   | 낮음           | 매우 낮음   |
| D-Bus 인터페이스               | 없음           | 있음      |
| 다중 장치 스택                  | 있음           | 있음      |
| LLDP를 사용하는 제로 구성          | 없음           | (계획 중)  |
| NetworkManager 지원         | 있음           | 있음      |

#### 14.4. 연결 모드에 따라 업스트림 스위치 구성

다음 표에서는 본딩 모드에 따라 업스트림 스위치에 적용해야 하는 설정을 설명합니다.

| 본딩 모드             | 스위치 설정                               |
|-------------------|--------------------------------------|
| 0 - balance-rr    | 정적 Etherchannel 활성화 필요 (LACP 협상이 아님) |
| 1 - active-backup | 자율 포트 필요                             |
| 2 - balance-xor   | 정적 Etherchannel 활성화 필요 (LACP 협상이 아님) |
| 3 - broadcast     | 정적 Etherchannel 활성화 필요 (LACP 협상이 아님) |
| 4 - 802.3ad       | LACP 협상 Etherchannel 활성화 필요          |
| 5 - balance-tlb   | 자율 포트 필요                             |
| 6 - balance-alb   | 자율 포트 필요                             |

스위치에서 이러한 설정을 구성하려면 스위치 설명서를 참조하십시오.

#### 14.5. NMCLI 명령을 사용하여 네트워크 본딩 구성

이 섹션에서는 `nmcli` 명령을 사용하여 네트워크 본딩을 구성하는 방법을 설명합니다.

##### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 본딩의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 브리지 또는 VLAN 장치를 본딩의 포트로 사용하려면 본딩을 생성하는 동안 이러한 장치를 만들거나 예 설명된 대로 미리 생성할 수 있습니다.
  - [nmcli 명령을 사용하여 네트워크 팀 구성](#)

- **nmcli 명령을 사용하여 네트워크 브리지 구성**
- **nmcli 명령을 사용하여 VLAN 태그 지정 구성**

## 절차

1.

본딩 인터페이스를 생성합니다.

```
nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup"
```

이 명령은 **active-backup** 모드를 사용하는 **bond0** 이라는 본딩을 생성합니다.

모니터링 간격을 추가로 설정하려면 **miimon=간격** 옵션을 **bond.options** 속성에 추가합니다. 예를 들어 동일한 명령을 사용하지만, **MII** 모니터링 간격을 **1000** 밀리초(1초)로 설정하려면 다음을 입력합니다.

```
nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup,miimon=1000"
```

2.

네트워크 인터페이스를 표시하고 본딩에 추가할 인터페이스 이름을 기록합니다.

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp7s0 ethernet disconnected --
enp8s0 ethernet disconnected --
bridge0 bridge connected bridge0
bridge1 bridge connected bridge1
...
```

이 예제에서는 다음을 수행합니다.

- **enp7s0** 및 **enp8s0** 은 구성되지 않습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 연결 프로필을 추가합니다.
- **bridge0** 및 **bridge1** 에는 기존 연결 프로필이 있습니다. 이러한 장치를 포트로 사용하려면 다음 단계에서 프로필을 수정합니다.

3.

본딩에 인터페이스를 할당합니다.

a.

본딩에 할당하려는 인터페이스가 구성되지 않은 경우 해당 인터페이스에 대한 새 연결 프로필을 생성합니다.

```
nmcli connection add type ethernet slave-type bond con-name bond0-port1
ifname enp7s0 master bond0
nmcli connection add type ethernet slave-type bond con-name bond0-port2
ifname enp8s0 master bond0
```

이 명령은 **enp7s0** 및 **enp8s0** 에 대한 프로필을 생성하여 **bond0** 연결에 추가합니다.

b.

본딩에 기존 연결 프로필을 할당하려면 **bond0** 에 이러한 연결의 **master** 매개변수를 설정합니다.

```
nmcli connection modify bridge0 master bond0
nmcli connection modify bridge1 master bond0
```

이러한 명령은 **bridge0** 및 **bridge1** 이라는 기존 연결 프로필을 **bond0** 연결에 할당합니다.

4.

본딩의 IP 설정을 구성합니다. 이 본딩을 다른 장치의 포트로 사용하려면 이 단계를 건너뛰니다.

a.

**IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 **bond0** 연결로 설정하려면 다음을 입력합니다.

```
nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify bond0 ipv4.gateway '192.0.2.254'
nmcli connection modify bond0 ipv4.dns '192.0.2.253'
nmcli connection modify bond0 ipv4.dns-search 'example.com'
nmcli connection modify bond0 ipv4.method manual
```

b.

**IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 **bond0** 연결로 설정하려면 다음을 입력합니다.

```
nmcli connection modify bond0 ipv6.addresses '2001:db8:1::1/64'
nmcli connection modify bond0 ipv6.gateway '2001:db8:1::fffe'
nmcli connection modify bond0 ipv6.dns '2001:db8:1::fffd'
```

```
nmcli connection modify bond0 ipv6.dns-search 'example.com'
nmcli connection modify bond0 ipv6.method manual
```

5.

연결을 활성화합니다.

```
nmcli connection up bond0
```

6.

포트가 연결되어 있고 **CONNECTION** 열에 포트의 연결 이름이 표시되는지 확인합니다.

```
nmcli device
DEVICE TYPE STATE CONNECTION
...
enp7s0 ethernet connected bond0-port1
enp8s0 ethernet connected bond0-port2
```

연결 포트를 활성화하면 **NetworkManager**는 본딩을 활성화하지만 다른 포트도 활성화되지 않습니다. 본딩이 활성화된 경우 **Red Hat Enterprise Linux**가 모든 포트를 자동으로 활성화하도록 구성할 수 있습니다.

a.

본딩 연결의 **connection.autoconnect-slaves** 매개변수를 활성화합니다.

```
nmcli connection modify bond0 connection.autoconnect-slaves 1
```

b.

브리지를 다시 활성화합니다.

```
nmcli connection up bond0
```

## 검증 단계

1.

호스트에서 네트워크 케이블을 일시적으로 제거합니다.

소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. **nmcli**와 같은 연결을 비활성화하는 틀은 포트 구성 변경 사항을 처리하고 실제 링크 실패 이벤트가 아닌 본딩 드라이버의 기능만 표시합니다.

2.

본딩 상태를 표시합니다.

```
cat /proc/net/bonding/bond0
```



## 추가 리소스

- [기본 네트워크 설정 테스트](#)
- [기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성.](#)
- [nmcli-examples\(7\) 도움말 페이지](#)
- [네트워크 본딩 문서](#)

## 14.6. RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성

이 섹션에서는 **RHEL** 웹 콘솔을 사용하여 네트워크 본딩을 구성하는 방법을 설명합니다.

### 사전 요구 사항

- **RHEL** 웹 콘솔에 로그인되어 있습니다.
- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 본딩의 멤버로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 브리지 또는 **VLAN** 장치를 본딩의 멤버로 사용하려면 다음에 설명된 대로 사전에 생성합니다.
  - [RHEL 웹 콘솔을 사용하여 네트워크 팀 구성](#)
  - [RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성](#)
  - [RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 구성](#)

## 절차

1. 화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.
2. **Interfaces (interface)** 섹션에서 **Add bond** 를 클릭합니다.
3. 생성할 본딩 장치의 이름을 입력합니다.
4. 본딩의 멤버여야 하는 인터페이스를 선택합니다.
5. 본딩의 모드를 선택합니다.  
  
**Active backup** 을 선택하면 웹 콘솔에 원하는 활성 장치를 선택할 수 있는 추가 기본 필드가 표시됩니다.
6. 링크 모니터링 모드를 설정합니다. 예를 들어, **Adaptive** 로드 밸런싱 모드를 사용하는 경우 **ARP** 로 설정합니다.
7. 선택 사항: 모니터링 간격, 링크 up 지연, 링크 다운 지연 설정을 조정합니다. 일반적으로 문제 해결 목적으로만 기본값을 변경합니다.

## Bond settings

Name

bond0

Interfaces

☒ enp7s0  
☒ enp8s0

MAC

Mode

Active backup

Primary

enp7s0

Link monitoring

MII (recommended)

Monitoring interval

100

Link up delay

0

Link down delay

0

Apply

Cancel

8.

**Apply(적용)**를 클릭합니다.

9.

기본적으로 본딩은 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

a.

**Interfaces (interface)** 섹션에서 본딩 이름을 클릭합니다.

b.

구성할 프로토콜 옆에 있는 편집 을 클릭합니다.

c.

주소 옆에 있는 **Manual** 을 선택하고 IP 주소, 접두사 및 기본 게이트웨이를 입력합니다.

d.

**DNS** 섹션에서 + 버튼을 클릭하고 DNS 서버의 IP 주소를 입력합니다. 이 단계를 반복하여 여러 DNS 서버를 설정합니다.

e.

**DNS** 검색 도메인 섹션에서 + 버튼을 클릭하고 검색 도메인을 입력합니다.

f.

인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 해당 경로를 구성합니다.

**IPv4 settings** [X]

Addresses Manual +

| Address   | Prefix length or netmask | Gateway     |
|-----------|--------------------------|-------------|
| 192.0.2.1 | 24                       | 192.0.2.254 |

DNS Automatic +

| Server      |
|-------------|
| 192.0.2.253 |

DNS search domains Automatic +

| Search domain |
|---------------|
| example.com   |

Routes Automatic +

Apply Cancel

g.

적용을 클릭합니다.

1.

화면 왼쪽의 탐색에서 **Networking** 탭을 선택하고 인터페이스에 들어오고 나가는 트래픽이 있는지 확인합니다.

| Interfaces                                         |              |           |           |
|----------------------------------------------------|--------------|-----------|-----------|
| <div> Add bond Add team Add bridge Add VLAN </div> |              |           |           |
| Name                                               | IP address   | Sending   | Receiving |
| bond0                                              | 192.0.2.1/24 | 1.11 Mbps | 61.2 Mbps |

2.

호스트에서 네트워크 케이블을 일시적으로 제거합니다.

소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. 웹 콘솔과 같은 연결을 비활성화하는 틀은 본딩 드라이버만 사용하여 멤버 구성 변경 사항을 처리하고 실제 링크 실패 이벤트가 아닌 멤버 구성 변경 사항을 처리하는 기능을 보여줍니다.

3.

본딩 상태를 표시합니다.

```
cat /proc/net/bonding/bond0
```

#### 14.7. NM-CONNECTION-EDITOR를 사용하여 네트워크 본딩 구성

이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 본딩을 구성하는 방법을 설명합니다.

**nm-connection-editor** 는 본딩에 새 포트만 추가할 수 있습니다. 기존 연결 프로필을 포트로 사용하려면 **nmcli** 명령을 사용하여 네트워크 본딩 구성에 설명된 대로 **nmcli** 유틸리티를 사용하여 본딩을 생성합니다.

##### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 본딩의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 본딩 또는 **VLAN** 장치를 본딩 포트로 사용하려면 해당 장치가 아직 구성되지 않았는지 확인합니다.

## 절차

1. 터미널을 열고 **`nm-connection-editor`** 를 입력합니다.
 

```
$ nm-connection-editor
```
2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. **Bond** 연결 유형을 선택하고 만들기 를 클릭합니다.
4. **Bond** 탭에서:
  - a. 선택 사항: 인터페이스 이름 필드에 본딩 인터페이스의 이름을 설정합니다.
  - b. **Add(추가)** 버튼을 클릭하여 네트워크 인터페이스를 본딩에 포트로 추가합니다.
    - i. 인터페이스의 연결 유형을 선택합니다. 예를 들어 유선 연결을 위해 이더넷 을 선택합니다.
    - ii. 선택 사항: 포트에 대한 연결 이름을 설정합니다.
    - iii. 이더넷 장치에 대한 연결 프로필을 만드는 경우 이더넷 탭을 열고 **Device (장치)** 필드에서 선택하고, 본딩에 포트로 추가할 네트워크 인터페이스를 선택합니다. 다른 장치 유형을 선택한 경우 적절하게 구성합니다. 구성되지 않은 본딩에서는 이더넷 인터페이스 만 사용할 수 있습니다.
    - iv. 저장을 클릭합니다.
  - c. 본딩에 추가할 각 인터페이스에 대해 이전 단계를 반복합니다.

Editing Bond connection 1

Connection name: Bond connection 1

General | **Bond** | Proxy | IPv4 Settings | IPv6 Settings

Interface name: bond0

Bonded connections

|             |     |
|-------------|-----|
| bond0-port1 | Add |
| bond0-port2 |     |

Edit

d. 선택 사항: **MII( Media Independent Interface)** 모니터링 간격과 같은 기타 옵션을 설정합니다.

5. 본딩의 **IP** 설정을 구성합니다. 이 본딩을 다른 장치의 포트로 사용하려면 이 단계를 건너뛰니다.

a. **IPv4 Settings** 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

**Editing Bond connection 1** [X]

Connection name: Bond connection 1

General | Bond | Proxy | **IPv4 Settings** | IPv6 Settings

Method: Manual

**Addresses**

| Address   | Netmask | Gateway     |
|-----------|---------|-------------|
| 192.0.2.1 | 24      | 192.0.2.254 |

[Add] [Delete]

DNS servers: 192.0.2.253

Search domains: example.com

b.

**IPv6 Settings** 탭에서 **IPv6** 설정을 구성합니다. 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 설정합니다.

**Editing Bond connection 1** [X]

Connection name: Bond connection 1

General | Bond | Proxy | IPv4 Settings | **IPv6 Settings**

Method: Manual

**Addresses**

| Address       | Prefix | Gateway          |
|---------------|--------|------------------|
| 2001:db8:1::1 | 64     | 2001:db8:1::fff3 |

[Add] [Delete]

DNS servers: 2001:db8:1::fffd

Search domains: example.com



6. **Save(저장)**를 클릭하여 본딩 연결을 저장합니다.

7. **nm-connection-editor** 를 닫습니다.

#### 검증 단계

1. 호스트에서 네트워크 케이블을 일시적으로 제거합니다.

소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. **nmcli**와 같은 연결을 비활성화하는 톨은 포트 구성 변경 사항을 처리하고 실제 링크 실패 이벤트가 아닌 본딩 드라이버의 기능만 표시합니다.

2. 본딩 상태를 표시합니다.

```
cat /proc/net/bonding/bond0
```

#### 추가 리소스

- [기본 네트워크 설정 테스트.](#)
- [기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성.](#)
- [nm-connection-editor를 사용하여 네트워크 팀 구성](#)
- [nm-connection-editor를 사용하여 네트워크 브릿지 구성](#)
- [nm-connection-editor를 사용하여 VLAN 태그 구성](#)

#### 14.8. NMSTATECTL을 사용하여 네트워크 본딩 구성

이 섹션에서는 **nmstatectl** 유틸리티를 사용하여 다음 설정으로 네트워크 **bond0** 을 구성하는 방법을 설명합니다.

- 본당의 네트워크 인터페이스인 **enp1s0** 및 **enp7s0**
- 모드: **active-backup**
- 정적 **IPv4** 주소: **/24** 서브넷 마스크가 있는 **192.0.2.1**
- 정적 **IPv6** 주소: **2001:db8:1::1** 및 **/64** 서브넷 마스크
- **IPv4** 기본 게이트웨이: **192.0.2.254**
- **IPv6** 기본 게이트웨이: **2001:db8:1::fffe**
- **IPv4** DNS 서버: **192.0.2.200**
- **IPv6** DNS 서버: **2001:db8:1::ffbb**
- DNS 검색 도메인: **example.com**

#### 사전 요구 사항

- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.
- 이더넷 장치를 본당의 포트로 사용하려면 물리적 또는 가상 이더넷 장치를 서버에 설치해야 합니다.
- 팀, 브리지 또는 **VLAN** 장치를 본당에서 포트로 사용하려면 포트 목록에서 인터페이스 이름을 설정하고 해당 인터페이스를 정의합니다.
- **nmstate** 패키지가 설치되어 있습니다.

## 절차

1.

다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/create-bond.yml**)을 생성합니다.

```

interfaces:
- name: bond0
 type: bond
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
 link-aggregation:
 mode: active-backup
 port:
 - enp1s0
 - enp7s0
- name: enp1s0
 type: ethernet
 state: up
- name: enp7s0
 type: ethernet
 state: up

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: bond0
 - destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: bond0

dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb
```

2.

시스템에 설정을 적용합니다.

```
nmstatectl apply ~/create-bond.yml
```

## 검증 단계

1.

장치 및 연결 상태를 표시합니다.

```
nmcli device status
DEVICE TYPE STATE CONNECTION
bond0 bond connected bond0
```

2.

연결 프로파일의 모든 설정을 표시합니다.

```
nmcli connection show bond0
connection.id: bond0
connection.uuid: 79cbc3bd-302e-4b1f-ad89-f12533b818ee
connection.stable-id: --
connection.type: bond
connection.interface-name: bond0
...
```

3.

연결 설정을 **YAML** 형식으로 표시합니다.

```
nmstatectl show bond0
```

## 추가 리소스

•

[nmstatectl\(8\) 도움말 페이지](#)

•

[/usr/share/doc/nmstate/examples/](#)

## 14.9. RHEL 시스템 역할을 사용하여 네트워크 본딩 구성

네트워크 **RHEL** 시스템 역할을 사용하여 네트워크 본딩을 구성할 수 있습니다. 다음 절차에서는 두 이더넷 장치를 사용하고 **IPv4** 및 **IPv6** 주소, 기본 게이트웨이 및 **DNS** 구성을 설정하는 **active-backup** 모드로 본딩을 구성하는 방법을 설명합니다.



### 참고

**Linux** 본딩 포트에가 아니라 본딩에서 **IP** 구성을 설정합니다.

## 사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.

## 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/bond-ethernet.yml** 플레이북을 생성합니다.

```

- name: Configure a network bond that uses two Ethernet ports
 hosts: node.example.com
 become: true
 tasks:
 - include_role:
 name: rhel-system-roles.network

 vars:
 network_connections:
 # Define the bond profile
 - name: bond0
 type: bond
 interface_name: bond0
 ip:
 address:
 - "192.0.2.1/24"
 - "2001:db8:1::1/64"
 gateway4: 192.0.2.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 192.0.2.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
 bond:
 mode: active-backup
 state: up
```

```
Add an Ethernet profile to the bond
```

```
- name: bond0-port1
 interface_name: enp7s0
 type: ethernet
 controller: bond0
 state: up
```

```
Add a second Ethernet profile to the bond
```

```
- name: bond0-port2
 interface_name: enp8s0
 type: ethernet
 controller: bond0
 state: up
```

3.

플레이북을 실행합니다.

- 

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/bond-ethernet.yml
```

- 

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/bond-ethernet.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- 

**/usr/share/ansible/roles/rhel-system-roles.network/README.md** file

- 

**ansible-playbook(1)** 도움말 페이지

#### 14.10. VPN을 중단하지 않고 이더넷과 무선 연결을 전환할 수 있는 네트워크 본딩 만들기

워크스테이션을 회사 네트워크에 연결하는 **RHEL** 사용자는 일반적으로 **VPN**을 사용하여 원격 리소스

에 액세스합니다. 그러나 예를 들어, 워크스테이션이 이더넷과 **Wi-Fi** 연결 간에 전환되는 경우(예: 이더넷 연결을 사용하여 보조 스테이션에서 랩톱을 해제하면 **VPN** 연결이 중단됩니다. 이 문제를 방지하려면 **active-backup** 모드에서 이더넷 및 **Wi-Fi** 연결을 사용하는 네트워크 본딩을 만들 수 있습니다.

#### 사전 요구 사항

- 호스트에는 이더넷 및 **Wi-Fi** 장치가 포함되어 있습니다.
- 이더넷 및 **Wi-Fi NetworkManager** 연결 프로필이 생성되었으며 두 연결이 독립적으로 작동합니다.

이 절차에서는 다음 연결 프로필을 사용하여 **bond0** 이라는 네트워크 본딩을 생성합니다.

- **enp11s0u1** 이더넷 장치와 관련된 **Docking\_station**
- **wlp1s0** **Wi-Fi** 장치와 연결된 **Wi-Fi**

#### 절차

1. **active-backup** 모드로 본딩 인터페이스를 만듭니다.

```
nmcli connection add type bond con-name bond0 ifname bond0 bond.options "mode=active-backup"
```

이 명령은 인터페이스 및 연결 프로필 **bond0** 의 이름을 모두 지정합니다.

2. 본딩의 **IPv4** 설정을 구성합니다.

- 네트워크의 **DHCP** 서버가 **IPv4** 주소를 호스트에 할당하는 경우 작업이 필요하지 않습니다.

- 로컬 네트워크에 정적 **IPv4** 주소가 필요한 경우 주소, 네트워크 마스크, 기본 게이트웨이, **DNS** 서버 및 **DNS** 검색 도메인을 **bond0** 연결로 설정합니다.

```
nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify bond0 ipv4.gateway '192.0.2.254'
```

```
nmcli connection modify bond0 ipv4.dns '192.0.2.253'
nmcli connection modify bond0 ipv4.dns-search 'example.com'
nmcli connection modify bond0 ipv4.method manual
```

3.

본당의 IPv6 설정을 구성합니다.

•

네트워크의 라우터 또는 DHCP 서버가 IPv6 주소를 호스트에 할당하는 경우 작업이 필요하지 않습니다.

•

로컬 네트워크에 정적 IPv6 주소가 필요한 경우 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 DNS 검색 도메인을 bond0 연결로 설정합니다.

```
nmcli connection modify bond0 ipv6.addresses '2001:db8:1::1/64'
nmcli connection modify bond0 ipv6.gateway '2001:db8:1::fffe'
nmcli connection modify bond0 ipv6.dns '2001:db8:1::fffd'
nmcli connection modify bond0 ipv6.dns-search 'example.com'
nmcli connection modify bond0 ipv6.method manual
```

4.

연결 프로필을 표시합니다.

```
nmcli connection show
NAME UUID TYPE DEVICE
Docking_station 256dd073-fecc-339d-91ae-9834a00407f9 ethernet enp11s0u1
Wi-Fi 1f1531c7-8737-4c60-91af-2d21164417e8 wifi wlp1s0
...
```

다음 단계에서 연결 프로필의 이름과 이더넷 장치 이름이 필요합니다.

5.

이더넷 연결의 연결 프로필을 본당에 할당합니다.

```
nmcli connection modify Docking_station master bond0
```

6.

Wi-Fi 연결의 연결 프로필을 본당에 할당합니다.

```
nmcli connection modify Wi-Fi master bond0
```

7.

Wi-Fi 네트워크에서 MAC 필터링을 사용하여 허용 목록의 MAC 주소만 네트워크에 액세스할 수 있는 경우 NetworkManager가 활성 포트의 MAC 주소를 본당에 동적으로 할당하도록 구성합니다.



```
nmcli connection modify bond0 +bond.options fail_over_mac=1
```

이 설정을 사용하면 이더넷 및 Wi-Fi 장치의 MAC 주소 대신 Wi-Fi 장치의 MAC 주소만 허용 목록에 설정해야 합니다.

8.

이더넷 연결과 연결된 장치를 본딩의 기본 장치로 설정합니다.

```
nmcli con modify bond0 +bond.options "primary=enp11s0u1"
```

이 설정을 사용하면 본딩에서 이더넷 연결을 항상 사용합니다.

9.

**bond0** 장치가 활성화되면 **NetworkManager**가 포트를 자동으로 활성화하도록 설정합니다.

```
nmcli connection modify bond0 connection.autoconnect-slaves 1
```

10.

**bond0** 연결을 활성화합니다.

```
nmcli connection up bond0
```

#### 검증 단계

•

현재 활성 장치, 본딩 상태 및 해당 포트를 표시합니다.

```
cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: enp11s0u1 (primary_reselect always)
Currently Active Slave: enp11s0u1
MII Status: up
MII Polling Interval (ms): 1
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: enp11s0u1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:53:00:59:da:b7
Slave queue ID: 0
```

**Slave Interface: wlp1s0**  
**MII Status: up**  
**Speed: Unknown**  
**Duplex: Unknown**  
**Link Failure Count: 2**  
**Permanent HW addr: 00:53:00:b3:22:ba**  
**Slave queue ID: 0**

#### 추가 리소스

- [이더넷 연결 구성](#)
- [와이파이 연결 관리](#)
- [네트워크 본딩 구성](#)

## 15장. VPN 연결 구성

이 섹션에서는 가상 사설 네트워크(VPN) 연결을 구성하는 방법에 대해 설명합니다.

VPN은 인터넷을 통해 로컬 네트워크에 연결하는 방법입니다. Libreswan 에서 제공하는 IPsec 은 VPN 을 만드는 데 선호되는 방법입니다. Libreswan 은 VPN을 위한 사용자 공간 IPsec 구현입니다. VPN은 인터넷과 같은 중간 네트워크에서 터널을 설정하여 LAN과 다른 원격 LAN 간의 통신을 가능하게 합니다. 보안상의 이유로 VPN 터널은 항상 인증 및 암호화를 사용합니다. 암호화 작업의 경우 Libreswan 은 NSS 라이브러리를 사용합니다.

### 15.1. CONTROL-CENTER를 사용한 VPN 연결 구성

다음 절차에서는 **control-center** 를 사용하여 VPN 연결을 구성하는 방법을 설명합니다.

사전 요구 사항

- **NetworkManager-libreswan-gnome** 패키지가 설치되어 있습니다.

절차

1. **Super** 키를 눌러 **Settings** 을 입력하고 **Enter** 를 눌러 **control-center** 애플리케이션을 엽니다.
2. 왼쪽에서 **네트워크** 항목을 선택합니다.
3. **+** 아이콘을 클릭합니다.
4. **VPN** 을 선택합니다.
5. **Identity** 메뉴 항목을 선택하여 기본 구성 옵션을 확인합니다.

일반

**gateway** - 원격 VPN 게이트웨이의 이름 또는 IP 주소입니다.

## 인증

### 유형

- ***IKEv2(인증서)***- 클라이언트는 인증서를 통해 인증됩니다. 더 안전합니다(기본값).
- ***IKEv1(XAUTH)*** - 클라이언트는 사용자 이름과 암호 또는 사전 공유 키()로 인증됩니다.

다음 구성 설정은 고급 섹션에서 사용할 수 있습니다.

그림 15.1. VPN 연결의 고급 옵션

IPsec Advanced Options

✕

Identification

Domain:

Security

Phase1 Algorithms:

Phase2 Algorithms:

☐ Disable PFS

Phase1 Lifetime:

Phase2 Lifetime:

☐ Disable rekeying

Connectivity

Remote Network:

☐ narrowing

Enable fragmentation

Enable MOBIKE

Apply



## 주의

**gnome-control-center** 애플리케이션을 사용하여 **IPsec** 기반 **VPN** 연결을 구성할 때 고급 대화 상자는 구성을 표시하지만 변경은 허용하지 않습니다. 결과적으로 사용자는 고급 **IPsec** 옵션을 변경할 수 없습니다. 고급 속성의 구성을 수행하는 대신 **nm-connection-editor** 또는 **nmcli** 도구를 사용합니다.

## ID

•

**domain** - 필요한 경우 **Domain Name**(도메인 이름)을 입력합니다.

## 보안

•

**1 단계 알고리즘 - Libre s wan** 매개 변수에 해당합니다. - 암호화된 채널을 인증하고 설정하는 데 사용할 알고리즘을 입력합니다.

•

**단계2 알고리즘 - esp Libreswan** 매개 변수에 해당합니다. **IPsec** 협상에 사용할 알고리즘을 입력합니다.

**Disable PFS (PFS 비활성화)** 필드를 확인하여 **PFS(PFS 전달)**를 지원하지 않는 이전 서버와의 호환성을 확인하기 위해 **PFS(전달 보안)**를 해제합니다.

•

**1 단계 라이프사이클 - the ikelifetime Libreswan** 매개 변수에 해당합니다. - 트래픽을 암호화하는 데 사용되는 키가 유효한지 확인합니다.

•

**2 단계 라이프 사이클 - salifetime Libreswan** 매개 변수에 해당합니다. - 특정 연결 인스턴스가 만료되기 전에 지속되어야 하는 시간입니다.

보안상의 이유로 암호화 키를 수시로 변경해야 합니다.

•

**원격 네트워크 - VPN**을 통해 도달해야 하는 대상 사설 네트워크인 **rightsubnet Libreswan** 매개 변수에 해당합니다.

좁히기를 활성화하려면 좁히는 필드를 확인합니다. **IKEv2** 협상에서만 유효합니다.

- 조각화 사용 - 조각화 허용 여부에 따라 **fragmentation Libreswan** 매개변수에 해당합니다. 유효한 값은 **yes** (기본값) 또는 **no** 입니다.
- **Mobike - mobike Libreswan** 매개변수에 해당합니다. **Mobility** 및 **Multihoming Protocol(MOBIKE, RFC 4555)**를 허용하여 연결을 처음부터 다시 시작할 필요 없이 엔드포인트를 마이그레이션하는 연결을 활성화할 수 있습니다. 이는 유선, 무선 또는 모바일 데이터 연결을 전환하는 모바일 장치에서 사용됩니다. 값은 **no** (기본값) 또는 **yes** 입니다.

6.

**IPv4** 메뉴 항목을 선택합니다.

#### IPv4 방법

- **Automatic (DHCP)** - 연결하는 네트워크가 **DHCP** 서버를 사용하여 동적 **IP** 주소를 할당하는 경우 이 옵션을 선택합니다.
- **링크-로컬 전용** - 연결 중인 네트워크에 **DHCP** 서버가 없고 **IP** 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. 임의의 주소는 **RFC 3927**에 따라 접두사 **169.254/16**과 같이 할당됩니다.
- **수동** - **IP** 주소를 수동으로 할당하려면 이 옵션을 선택합니다.
- **disable** - 이 연결에 대해 **IPv4**가 비활성화되어 있습니다.

#### DNS

**DNS** 섹션에서 **Automatic** (자동)이 **ON** (자동)인 경우 이를 **OFF** (꺼짐)로 전환하여 **IP**를 쉽게 구분하는 데 사용할 **DNS** 서버의 **IP** 주소를 입력합니다.

#### 라우트

**Routes**(경로) 섹션에서 **Automatic**(자동)이 **ON**이면 **DHCP**의 경로가 사용되지만 정적

경로도 추가할 수 있습니다. **OFF** 인 경우 정적 경로만 사용됩니다.

- **Address (주소)** - 원격 네트워크 또는 호스트의 IP 주소를 입력합니다.
- **넷마스크** - 위에 입력한 IP 주소의 넷마스크 또는 접두사 길이입니다.
- **Gateway(게이트웨이)** - 위에서 입력한 원격 네트워크 또는 호스트를 만드는 게이트웨이의 IP 주소입니다.
- **지표** - 네트워크 비용, 이 경로에 제공할 기본 설정 값입니다. 더 낮은 값은 더 높은 값보다 우선합니다.

이 연결은 네트워크의 리소스에만 사용하십시오.

이 확인란을 선택하여 연결이 기본 경로가 되지 않도록 합니다. 이 옵션을 선택하면 연결을 통해 자동으로 학습되었거나 여기에 수동으로 입력한 경로만 연결을 통해 라우팅됩니다.

7.

**VPN 연결에서 IPv6 설정을 구성하려면 IPv6 메뉴 항목을 선택합니다.**

#### IPv6 방법

- **auto** - 이 옵션을 선택하여 하드웨어 주소 및 라우터 알럼(RA)을 기반으로 자동 상태 비저장 구성을 생성하도록 IPv6 상태 비저장 주소 자동 구성(SLAAC)을 사용합니다.
- **Automatic, DHCP only** - 이 옵션을 선택하여 RA를 사용하지 않고 DHCPv6의 정보를 직접 요청하여 상태 저장 구성을 생성합니다.
- **링크-로컬 전용** - 연결 중인 네트워크에 DHCP 서버가 없고 IP 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. RFC 4862에 따라 접두사 FE80::0이 있는 임의 주소가 할당됩니다.
- **수동** - IP 주소를 수동으로 할당하려면 이 옵션을 선택합니다.



- **disable** - 이 연결에 대해 **IPv6** 가 비활성화되어 있습니다.

**DNS, 경로, 네트워크 리소스**에만 이 연결을 사용하는 것은 **IPv4** 설정에서 일반적입니다.

8. **VPN** 연결을 편집한 후 **Add** (추가) 버튼을 클릭하여 구성을 사용자 지정하거나 **Apply**(적용) 버튼을 클릭하여 기존에 대해 저장합니다.
9. **프로필을 ON** 으로 전환하여 **VPN** 연결을 활성 상태로 전환합니다.

#### 추가 리소스

- **nm-settings-libreswan(5)**

### 15.2. NM-CONNECTION-EDITOR를 사용하여 VPN 연결 구성

다음 절차에서는 **nm-connection-editor** 를 사용하여 **VPN** 연결을 구성하는 방법을 설명합니다.

#### 사전 요구 사항

- **NetworkManager-libreswan-gnome** 패키지가 설치되어 있습니다.
- **IKEv2(Internet Key Exchange version 2)** 연결을 구성하는 경우:
  - 인증서를 **IPsec** 네트워크 보안 서비스(**NSS**) 데이터베이스로 가져옵니다.
  - **NSS** 데이터베이스의 인증서는 잘 알려져 있습니다.

#### 절차

1. 터미널을 열고 다음을 입력합니다.

```
$ nm-connection-editor
```

2.

**+ 버튼을 클릭하여 새 연결을 추가합니다.**

3.

**IPsec 기반 VPN 연결 유형을 선택하고 생성을 클릭합니다.**

4.

**VPN 탭에서:**

a.

**VPN 게이트웨이의 호스트 이름 또는 IP 주소를 게이트웨이 필드에 입력하고 인증 유형을 선택합니다. 인증 유형에 따라 다른 추가 정보를 입력해야 합니다.**

•

**IKEv2(Certifiate)** 는 더 안전한 인증서를 사용하여 클라이언트를 인증합니다. 이 설정을 사용하려면 **IPsec NSS** 데이터베이스의 인증서가 필요합니다.

•

**IKEv1(XAUTH)** 은 사용자 이름 및 암호(사전 공유 키)를 사용하여 사용자를 인증합니다. 이 설정을 사용하려면 다음 값을 입력해야 합니다.

○

사용자 이름

○

암호

○

그룹 이름

○

**Secret**

b.

원격 서버가 **IKE** 교환에 대한 로컬 식별자를 지정하는 경우 원격 **ID** 필드에 정확한 문자열을 입력합니다. 원격 서버에서 **Libreswan**을 실행하면 이 값은 서버의 **leftid** 매개 변수에 설정됩니다.

Editing VPN connection 1

Connection name: VPN connection 1

General VPN Proxy IPv4 Settings

**General**

Gateway: vpn.example.com

**Authentication**

Type: IKEv2 (Certificate)

Certificate name: cert\_name\_in\_IPSec\_DB

Remote ID:

Advanced...

c.

선택적으로 **Advanced** (고급) 버튼을 클릭하여 추가 설정을 구성합니다. 다음 설정을 구성할 수 있습니다.

•

**ID**

○

**domain** - 필요한 경우 도메인 이름을 입력합니다.

•

**보안**

○

**1 단계** 알고리즘은 **Libre swan** 매개 변수에 해당합니다. 암호화된 채널을 인증하고 설정하는 데 사용할 알고리즘을 입력합니다.

○

**2 단계** 알고리즘은 **esp Libreswan** 매개 변수에 해당합니다. **IPsec** 협상에 사용할 알고리즘을 입력합니다.

**Disable PFS (PFS 비활성화)** 필드를 확인하여 **PFS(PFS 전달)**를 지원하지 않는 이전 서버와의 호환성을 확인하기 위해 **PFS(전달 보안)**를 해제합니다.

○

**1 단계 라이프 사이클**은 **ikelifetime Libreswan** 매개 변수에 해당합니다. 이 매개 변수는 트래픽을 암호화하는 데 사용되는 키가 유효한 기간을 정의합니다.

○

**2 단계 라이프 타임**은 **salifetime Libreswan** 매개 변수에 해당합니다. 이 매개 변수는 보안 연결이 유효한 기간을 정의합니다.

●

### 연결

○

원격 네트워크는 **rightsubnet Libreswan** 매개 변수에 해당하며 **VPN**을 통해 연결할 대상 사설 원격 네트워크를 정의합니다.

좁히기를 활성화하려면 좁히는 필드를 확인합니다. **IKEv2** 협상에서만 유효합니다.

○

조각화 활성화는 **fragmentation Libreswan** 매개 변수에 해당하며 **IKE** 조각화를 허용할지 여부를 정의합니다. 유효한 값은 **yes** (기본값) 또는 **no**입니다.

○

**Mobike**는 **mobike Libreswan** 매개 변수에 해당합니다. 매개 변수는 연결을 처음부터 재시작할 필요 없이 해당 엔드포인트를 마이그레이션하기 위한 연결을 활성화할 수 있도록 **MOBIKE(Mobility and Multihoming Protocol)(RFC 4555)**를 허용할지 여부를 정의합니다. 이는 유선, 무선 또는 모바일 데이터 연결을 전환하는 모바일 장치에서 사용됩니다. 값은 **no** (기본값) 또는 **yes**입니다.

5.

**IPv4 Settings** 탭에서 **IP** 할당 방법을 선택하고 선택적으로 추가 정적 주소, **DNS** 서버, 검색 도메인 및 경로를 설정합니다.

Editing VPN connection 1

Connection name: VPN connection 1

General VPN Proxy **IPv4 Settings**

Method: Automatic (VPN)

**Additional static addresses**

| Address | Netmask | Gateway |
|---------|---------|---------|
|         |         |         |

Add Delete

Additional DNS servers:

Additional search domains:

Routes...

6. 연결을 저장합니다.

7. **nm-connection-editor** 를 닫습니다.



참고

+ 버튼을 클릭하여 새 연결을 추가하면 **NetworkManager** 가 해당 연결에 대한 새 구성 파일을 만든 다음 기존 연결을 편집하는 데 사용되는 동일한 대화 상자를 엽니다. 이러한 대화 상자의 차이점은 기존 연결 프로필에 **Details** 메뉴 항목이 있다는 것입니다.

추가 리소스



**nm-settings-libreswan(5)** 도움말 페이지

### 15.3. IPSEC 연결 속도를 높이기 위해 ESP 하드웨어 오프로드 자동 감지 및 사용 구성

하드웨어로 캡슐화된 보안 페이로드(ESP)를 오프로드하면 이더넷을 통해 IPsec 연결을 가속화할 수 있습니다. 기본적으로 **Libreswan**은 하드웨어가 이 기능을 지원하는지 감지하여 ESP 하드웨어 오프로드

를 활성화합니다. 이 절차에서는 기능을 비활성화하거나 명시적으로 사용하도록 설정한 경우 자동 탐지를 활성화하는 방법을 설명합니다.

#### 사전 요구 사항

- 네트워크 카드는 **ESP** 하드웨어 오프로드를 지원합니다.
- 네트워크 드라이버는 **ESP** 하드웨어 오프로드를 지원합니다.
- **IPsec** 연결이 구성되고 작동합니다.

#### 절차

1. **ESP** 하드웨어 오프로드 지원의 자동 탐지를 사용해야 하는 연결의 `/etc/ipsec.d/` 디렉토리에 서 **Libreswan** 구성 파일을 편집합니다.
2. 연결 설정에 **nic-offload** 매개변수가 설정되어 있지 않은지 확인합니다.
3. **nic-offload** 를 제거한 경우 **ipsec** 서비스를 다시 시작합니다.

```
systemctl restart ipsec
```

#### 검증

네트워크 카드가 **ESP** 하드웨어 오프로드 지원을 지원하는 경우 다음 단계에 따라 결과를 확인하십시오.

1. **IPsec** 연결에 사용하는 이더넷 장치의 **tx\_ipsec** 및 **rx\_ipsec** 카운터를 표시합니다.

```
ethtool -S enp1s0 | egrep "_ipsec"
tx_ipsec: 10
rx_ipsec: 10
```

2. **IPsec** 터널을 통해 트래픽을 전송합니다. 예를 들어 원격 **IP** 주소를 **ping** 합니다.

```
ping -c 5 remote_ip_address
```

3.

이더넷 장치의 **tx\_ipsec** 및 **rx\_ipsec** 카운터를 다시 표시합니다.

```
ethtool -S enp1s0 | egrep "_ipsec"
tx_ipsec: 15
rx_ipsec: 15
```

카운터 값이 증가하면 **ESP** 하드웨어 오프로드가 작동합니다.

추가 리소스

- [IPsec을 사용하여 VPN 구성](#)

#### 15.4. IPSEC 연결을 가속화하도록 본딩에 **ESP** 하드웨어 오프로드 구성

하드웨어로 **ESB(Security Payload)**를 오프로드하면 **IPsec** 연결 속도가 빨라집니다. 네트워크 본딩을 장애 조치의 이유로 사용하는 경우 **ESP** 하드웨어 오프로드를 구성하는 절차와 일반 이더넷 장치를 사용하는 절차가 다릅니다. 예를 들어 이 시나리오에서는 본딩에 대한 오프로드 지원을 활성화하고 커널은 설정을 본딩 포트에 적용합니다.

사전 요구 사항

- 본딩의 모든 네트워크 카드는 **ESP** 하드웨어 오프로드를 지원합니다.
- 네트워크 드라이버는 본딩 장치에서 **ESP** 하드웨어 오프로드를 지원합니다. **RHEL**에서는 **ixgbe** 드라이버만 이 기능을 지원합니다.
- 본딩이 구성되고 작동합니다.
- 본딩에서는 **active-backup** 모드를 사용합니다. 본딩 드라이버는 이 기능에 대해 다른 모드를 지원하지 않습니다.
- **IPsec** 연결이 구성되고 작동합니다.

절차

1.

네트워크 본딩에서 **ESP** 하드웨어 오프로드 지원을 활성화합니다.

```
nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

이 명령을 사용하면 **bond0** 연결에서 **ESP** 하드웨어 오프로드를 지원할 수 있습니다.

2.

**bond0** 연결을 다시 활성화합니다.

```
nmcli connection up bond0
```

3.

**ESP** 하드웨어 오프로드를 사용해야 하는 연결의 **/etc/ipsec.d/** 디렉토리에서 **Libreswan** 구성 파일을 편집하고 **nic-offload=yes** 문을 연결 항목에 추가합니다.

```
conn example
...
nic-offload=yes
```

4.

**ipsec** 서비스를 다시 시작하십시오.

```
systemctl restart ipsec
```

## 검증

1.

본딩의 활성 포트를 표시합니다.

```
grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

2.

활성 포트의 **tx\_ipsec** 및 **rx\_ipsec** 카운터를 표시합니다.

```
ethtool -S enp1s0 | egrep "_ipsec"
tx_ipsec: 10
rx_ipsec: 10
```

3.

**IPsec** 터널을 통해 트래픽을 전송합니다. 예를 들어 원격 **IP** 주소를 **ping**합니다.

```
ping -c 5 remote_ip_address
```



4.

활성 포트의 **tx\_ipsec** 및 **rx\_ipsec** 카운터를 다시 표시합니다.

```
ethtool -S enp1s0 | egrep "_ipsec"
tx_ipsec: 15
rx_ipsec: 15
```

카운터 값이 증가하면 **ESP** 하드웨어 오프로드가 작동합니다.

추가 리소스

- [네트워크 본딩 구성](#)
- [네트워크 보안 설명서의 IPsec을 사용하여 VPN 구성 섹션](#)
- [네트워크 보안 문서의 IPsec을 사용하여 VPN 구성 장.](#)

## 16장. IP 터널 구성

**VPN과 유사하게 IP 터널은 인터넷과 같은 세 번째 네트워크를 통해 두 개의 네트워크를 직접 연결합니다. 그러나 일부 터널 프로토콜이 암호화를 지원하지는 않습니다.**

터널을 설정하는 두 네트워크의 라우터에는 최소한 두 개의 인터페이스가 필요합니다.

- 로컬 네트워크에 연결된 인터페이스 한 개
- 터널이 설정된 네트워크에 연결된 하나의 인터페이스입니다.

터널을 설정하려면 원격 서브넷에서 **IP** 주소를 사용하여 두 라우터 모두에서 가상 인터페이스를 만듭니다.

**NetworkManager**는 다음 **IP** 터널을 지원합니다.

- **GRE(Generic Routing Encapsulation)**
- 일반 라우팅 **Encapsulation over IPv6 (IP6GRE)**
- **GRETap(Generic Routing Encapsulation Terminal Access Point)**
- 일반 라우팅 캡슐화 터미널 액세스 지점 **over IPv6 (IP6GRETap)**
- **IPv4 over IPv4 (IPIP)**
- **IPv6를 통한 IPv4 (IPIP6)**
- **IPv6 over IPv6 (IP6IP6)**

## SIT(Simple Internet transition)

유형에 따라 이러한 터널은 **OSI(Open Systems Interconnection)** 모델의 계층 2 또는 3에서 작동합니다.

### 16.1. IPV4 패킷에서 IPV4 트래픽을 캡슐화하도록 NMCLI를 사용하여 IPIP 터널 구성

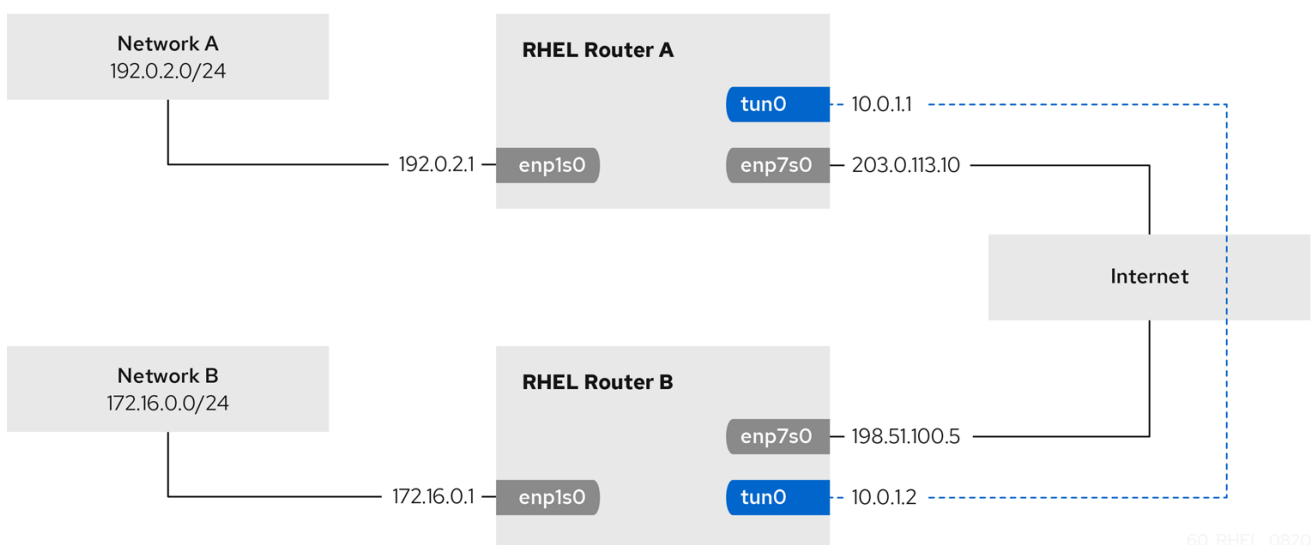
**IPIP(IP IP)** 터널은 **OSI** 계층 3에서 작동하며 **RFC 2003**에 설명된 대로 **IPv4** 패킷의 **IPv4** 트래픽을 캡슐화합니다.

#### 중요

**IPIP** 터널을 통해 전송된 데이터는 암호화되지 않습니다. 보안상의 이유로 터널은 이미 암호화된 데이터에만 사용합니다(예: **HTTPS**와 같은 다른 프로토콜).

**IPIP** 터널은 유니캐스트 패킷만 지원합니다. 멀티 캐스트를 지원하는 **IPv4** 터널이 필요한 경우 **IPv4** 패킷에서 계층-3 트래픽을 캡슐화하도록 **nmcli**를 사용하여 **GRE** 터널 구성을 참조하십시오.

이 절차에서는 다음 다이어그램에 표시된 대로 두 **RHEL** 라우터 간에 **IPIP** 터널을 생성하여 인터넷을 통해 두 개의 내부 서브넷을 연결하는 방법을 설명합니다.



사전 요구 사항

- 각 **RHEL** 라우터에는 로컬 서브넷에 연결된 네트워크 인터페이스가 있습니다.
- 각 **RHEL** 라우터에는 인터넷에 연결된 네트워크 인터페이스가 있습니다.
- 터널을 통해 전송하려는 트래픽은 **IPv4** 유니캐스트입니다.

## 절차

1.

네트워크 **A**의 **RHEL** 라우터에서:

a.

**tun0** 이라는 **IPIP** 터널 인터페이스를 만듭니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode ipip con-name tun0 ifname
tun0 remote 198.51.100.5 local 203.0.113.10
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 **IP** 주소를 설정합니다.

b.

**IPv4** 주소를 **tun0** 장치로 설정합니다.

```
nmcli connection modify tun0 ipv4.addresses '10.0.1.1/30'
```

두 개의 사용 가능한 **IP** 주소가 있는 **/30** 서브넷은 터널에 충분합니다.

c.

수동 **IPv4** 구성을 사용하도록 **tun0** 연결을 구성합니다.

```
nmcli connection modify tun0 ipv4.method manual
```

d.

트래픽을 **172.16.0.0/24** 네트워크로 라우팅하는 고정 경로를 라우터 **B** 라우터의 터널 **IP**로 추가합니다.

```
nmcli connection modify tun0 +ipv4.routes "172.16.0.0/24 10.0.1.2"
```

e.

**tun0** 연결을 활성화합니다.

```
nmcli connection up tun0
```

f.

패킷 전달 활성화:

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

네트워크 B의 RHEL 라우터에서:

a.

tun0 이라는 IPIP 터널 인터페이스를 만듭니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode ipip con-name tun0 ifname
tun0 remote 203.0.113.10 local 198.51.100.5
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 IP 주소를 설정합니다.

b.

IPv4 주소를 tun0 장치로 설정합니다.

```
nmcli connection modify tun0 ipv4.addresses '10.0.1.2/30'
```

c.

수동 IPv4 구성을 사용하도록 tun0 연결을 구성합니다.

```
nmcli connection modify tun0 ipv4.method manual
```

d.

192.0.2.0/24 네트워크로 트래픽을 라우팅하는 정적 경로를 라우터 A 라우터의 터널 IP로 추가합니다.

```
nmcli connection modify tun0 +ipv4.routes "192.0.2.0/24 10.0.1.1"
```

e.

tun0 연결을 활성화합니다.

```
nmcli connection up tun0
```

f.

패킷 전달 활성화:

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

### 검증 단계

- 

각 **RHEL** 라우터에서 다른 라우터의 내부 인터페이스의 **IP** 주소를 **ping**합니다.

a.

라우터 **A**에서 **ping 172.16.0.1**:

```
ping 172.16.0.1
```

b.

라우터 **B**에서 **192.0.2.1 ping**:

```
ping 192.0.2.1
```

### 추가 리소스

- 

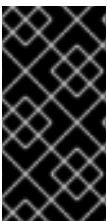
**nmcli man** 페이지

- 

**nm -settings(5)** 매뉴얼 페이지의 **ip-tunnel** 설정 섹션

## 16.2. IPV4 패킷에서 계층-3 트래픽을 캡슐화하도록 NMCLI를 사용하여 GRE 터널 구성

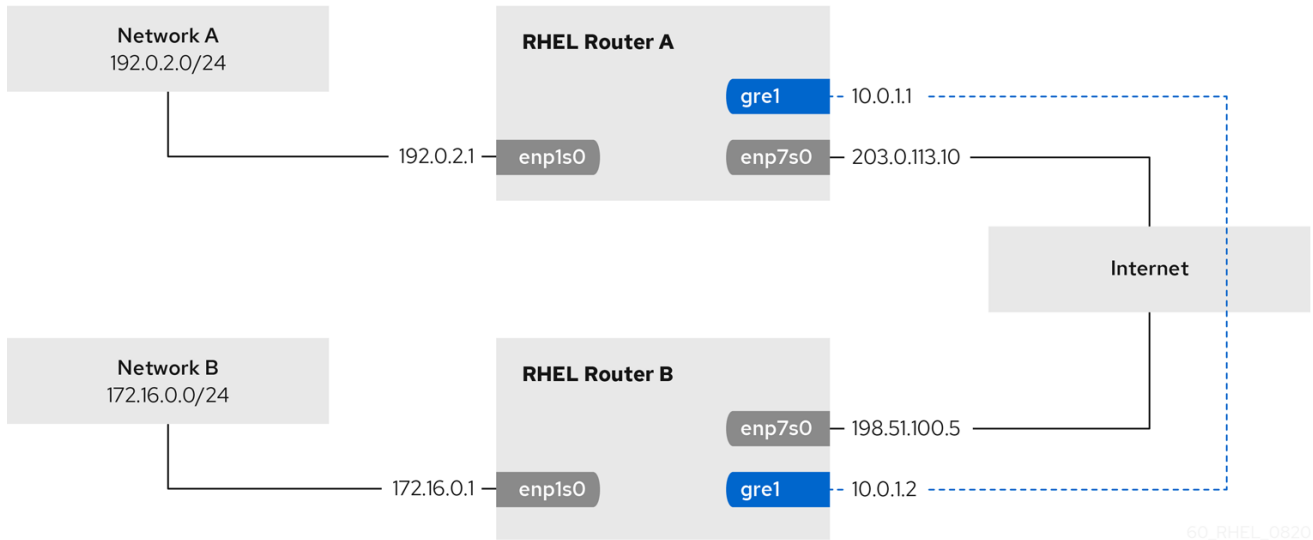
**GRE(Generic Routing Encapsulation)** 터널은 [RFC 2784](#)에 설명된 대로 **IPv4** 패킷의 계층-3 트래픽을 캡슐화합니다. **GRE** 터널은 모든 계층 3 프로토콜을 유효한 이더넷 유형으로 캡슐화할 수 있습니다.



### 중요

**GRE** 터널을 통해 전송되는 데이터는 암호화되지 않습니다. 보안상의 이유로 터널은 이미 암호화된 데이터에만 사용됩니다(예: **HTTPS**와 같은 다른 프로토콜).

이 절차에서는 다음 다이어그램에 표시된 대로 두 **RHEL** 라우터 간에 **GRE** 터널을 만들어 인터넷을 통해 두 개의 내부 서브넷을 연결하는 방법을 설명합니다.



## 참고

**gre0** 장치 이름은 예약되어 있습니다. 장치에 **gre1** 또는 다른 이름을 사용합니다.

## 사전 요구 사항

- 각 **RHEL** 라우터에는 로컬 서브넷에 연결된 네트워크 인터페이스가 있습니다.
- 각 **RHEL** 라우터에는 인터넷에 연결된 네트워크 인터페이스가 있습니다.

## 절차

1.

네트워크 **A**의 **RHEL** 라우터에서:

a.

**gre1** 이라는 **GRE** 터널 인터페이스를 만듭니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode gre con-name gre1 ifname gre1 remote 198.51.100.5 local 203.0.113.10
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 **IP** 주소를 설정합니다.

b.

**IPv4** 주소를 **gre1** 장치로 설정합니다.

```
nmcli connection modify gre1 ipv4.addresses '10.0.1.1/30'
```

두 개의 사용 가능한 IP 주소가 있는 /30 서브넷은 터널에 충분합니다.

c.

수동 IPv4 구성을 사용하도록 gre1 연결을 구성합니다.

```
nmcli connection modify gre1 ipv4.method manual
```

d.

트래픽을 172.16.0.0/24 네트워크로 라우팅하는 고정 경로를 라우터 B 라우터의 터널 IP로 추가합니다.

```
nmcli connection modify gre1 +ipv4.routes "172.16.0.0/24 10.0.1.2"
```

e.

gre1 연결을 활성화합니다.

```
nmcli connection up gre1
```

f.

패킷 전달 활성화:

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

네트워크 B의 RHEL 라우터에서:

a.

gre1 이라는 GRE 터널 인터페이스를 만듭니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode gre con-name gre1 ifname
gre1 remote 203.0.113.10 local 198.51.100.5
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 IP 주소를 설정합니다.

b.

IPv4 주소를 gre1 장치로 설정합니다.

```
nmcli connection modify gre1 ipv4.addresses '10.0.1.2/30'
```



c.

수동 **IPv4** 구성을 사용하도록 **gre1** 연결을 구성합니다.

```
nmcli connection modify gre1 ipv4.method manual
```

d.

**192.0.2.0/24** 네트워크로 트래픽을 라우팅하는 정적 경로를 라우터 **A** 라우터의 터널 **IP**로 추가합니다.

```
nmcli connection modify gre1 +ipv4.routes "192.0.2.0/24 10.0.1.1"
```

e.

**gre1** 연결을 활성화합니다.

```
nmcli connection up gre1
```

f.

패킷 전달 활성화:

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

## 검증 단계

1.

각 **RHEL** 라우터에서 다른 라우터의 내부 인터페이스의 **IP** 주소를 **ping**합니다.

a.

라우터 **A**에서 **ping 172.16.0.1**:

```
ping 172.16.0.1
```

b.

라우터 **B**에서 **192.0.2.1 ping**:

```
ping 192.0.2.1
```

## 추가 리소스

•

**nmcli man** 페이지

•

**nm -settings(5)** 매뉴얼 페이지의 **ip-tunnel** 설정 섹션

### 16.3. IPv4를 통해 인터넷 프레임을 전송하도록 GRE-TAP 터널 구성

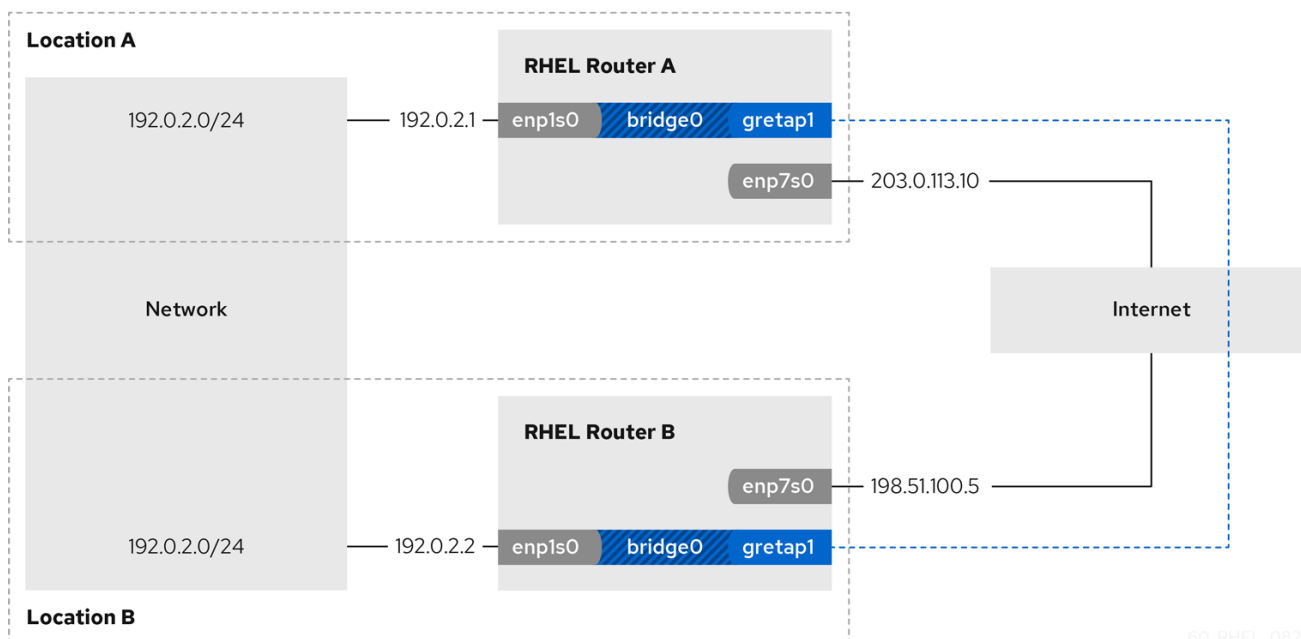
**GRE-TAP(Generic Routing Encapsulation Terminal Access Point)** 터널은 OSI 수준 2에서 작동하고 [RFC 2784](#)에 설명된 대로 IPv4 패킷의 인터넷 트래픽을 캡슐화합니다.



#### 중요

**GRE-TAP** 터널을 통해 전송되는 데이터는 암호화되지 않습니다. 보안상의 이유로 **VPN** 또는 다른 암호화된 연결을 통해 터널을 설정하십시오.

다음 절차에서는 다음 다이어그램에 표시된 대로 브리지를 사용하여 두 개의 네트워크를 연결하는 두 **RHEL** 라우터 간에 **GRE-TAP** 터널을 만드는 방법을 설명합니다.



60\_RHEL\_0820



#### 참고

**gretap0** 장치 이름은 예약되어 있습니다. **gretap1** 또는 장치에 다른 이름을 사용합니다.

#### 사전 요구 사항

- 

각 **RHEL** 라우터에는 로컬 네트워크에 연결된 네트워크 인터페이스가 있으며 인터페이스에는 IP 구성이 할당되지 않습니다.

•

각 **RHEL** 라우터에는 인터넷에 연결된 네트워크 인터페이스가 있습니다.

## 절차

1.

네트워크 **A**의 **RHEL** 라우터에서:

a.

**bridge0** 이라는 브리지 인터페이스를 만듭니다.

```
nmcli connection add type bridge con-name bridge0 ifname bridge0
```

b.

브리지의 **IP** 설정을 구성합니다.

```
nmcli connection modify bridge0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify bridge0 ipv4.method manual
```

c.

로컬 네트워크에 연결된 인터페이스에 대한 새 연결 프로필을 브리지에 추가합니다.

```
nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
ifname enp1s0 master bridge0
```

d.

**GRETAP** 터널 인터페이스의 새 연결 프로필을 브리지에 추가합니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode gretap slave-type bridge
con-name bridge0-port2 ifname gretap1 remote 198.51.100.5 local 203.0.113.10
master bridge0
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 **IP** 주소를 설정합니다.

e.

선택 사항: 필요하지 않은 경우 스페닝 트리 프로토콜 (**STP**)을 비활성화합니다.

```
nmcli connection modify bridge0 bridge.stp no
```

기본적으로 **STP**가 활성화되어 연결을 사용할 수 있기 전에 지연이 발생합니다.

f.

**bridge0** 연결을 활성화하면 브리지의 포트를 자동으로 활성화하도록 설정합니다.

```
nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

g.

**bridge0** 연결을 활성화합니다.

```
nmcli connection up bridge0
```

2.

네트워크 **B**의 **RHEL** 라우터에서:

a.

**bridge0** 이라는 브리지 인터페이스를 만듭니다.

```
nmcli connection add type bridge con-name bridge0 ifname bridge0
```

b.

브리지의 **IP** 설정을 구성합니다.

```
nmcli connection modify bridge0 ipv4.addresses '192.0.2.2/24'
nmcli connection modify bridge0 ipv4.method manual
```

c.

로컬 네트워크에 연결된 인터페이스에 대한 새 연결 프로필을 브리지에 추가합니다.

```
nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
ifname enp1s0 master bridge0
```

d.

**GRETAP** 터널 인터페이스의 새 연결 프로필을 브리지에 추가합니다.

```
nmcli connection add type ip-tunnel ip-tunnel.mode gretap slave-type bridge
con-name bridge0-port2 ifname gretap1 remote 203.0.113.10 local 198.51.100.5
master bridge0
```

원격 및 로컬 매개 변수는 원격 및 로컬 라우터의 공용 **IP** 주소를 설정합니다.

e.

선택 사항: 필요하지 않은 경우 스페닝 트리 프로토콜 (**STP**)을 비활성화합니다.

```
nmcli connection modify bridge0 bridge.stp no
```

f.

**bridge0** 연결을 활성화하면 브리지의 포트를 자동으로 활성화하도록 설정합니다.

```
nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

g.

**bridge0** 연결을 활성화합니다.

```
nmcli connection up bridge0
```

### 검증 단계

1.

두 라우터 모두에서 **enp1s0** 및 **gretap1** 연결이 연결되어 있고 **CONNECTION** 열에 포트의 연결 이름이 표시되는지 확인합니다.

```
nmcli device
nmcli device
DEVICE TYPE STATE CONNECTION
...
bridge0 bridge connected bridge0
enp1s0 ethernet connected bridge0-port1
gretap1 iptunnel connected bridge0-port2
```

2.

각 **RHEL** 라우터에서 다른 라우터의 내부 인터페이스의 **IP** 주소를 **ping**합니다.

a.

**A** 라우터에서 **ping 192.0.2.2**:

```
ping 192.0.2.2
```

b.

라우터 **B**에서 **192.0.2.1 ping**:

```
ping 192.0.2.1
```

### 추가 리소스

•

**nmcli man** 페이지

•

**nm -settings(5)** 매뉴얼 페이지의 **ip-tunnel** 설정 섹션

## 16.4. 추가 리소스

- ***ip-link(8) 도움말 페이지***

## 17장. RHEL에서 레거시 네트워크 스크립트 지원

기본적으로 RHEL은 NetworkManager를 사용하여 네트워크 연결을 구성하고 관리하고 /usr/sbin/ifup 및 /usr/sbin/ifdown 스크립트는 NetworkManager를 사용하여 /etc/sysconfig/network-scripts/ 디렉토리의 ifcfg 파일을 처리합니다.



### 중요

레거시 스크립트는 RHEL 8에서 더 이상 사용되지 않으며 향후 주요 RHEL 버전에서 제거됩니다. 예를 들어 이전 버전에서 RHEL 8로 업그레이드된 레거시 네트워크 스크립트를 계속 사용하는 경우 Red Hat은 구성을 NetworkManager로 마이그레이션하는 것이 좋습니다.

### 17.1. 레거시 네트워크 스크립트 설치

NetworkManager를 사용하지 않고 네트워크 구성을 처리하는 더 이상 사용되지 않는 네트워크 스크립트가 필요한 경우 설치할 수 있습니다. 이 경우 /usr/sbin/ifup 및 /usr/sbin/ifdown 스크립트는 네트워크 구성을 관리하는 더 이상 사용되지 않는 셸 스크립트에 연결됩니다.

#### 절차

- **network-scripts** 패키지를 설치합니다.

```
yum install network-scripts
```

## 18장. 포트 미러링

네트워크 관리자는 포트 미러링을 사용하여 한 네트워크 장치에서 다른 네트워크 장치로 통신 중인 인바운드 및 아웃바운드 네트워크 트래픽을 복제할 수 있습니다. 관리자는 포트 미러링을 사용하여 네트워크 트래픽을 모니터링하고 네트워크 데이터를 수집합니다.

- 네트워크 문제를 디버깅하고 네트워크 흐름 조정
- 네트워크 트래픽을 검사 및 분석하여 네트워크 문제 해결
- 침입 감지

### 18.1. NMCLI를 사용하여 네트워크 인터페이스 미러링

**NetworkManager**를 사용하여 포트 미러링을 구성할 수 있습니다. 다음 절차에서는 트래픽 제어(tc) 규칙을 추가하고 **enp1s0** 네트워크 인터페이스에 필터링하여 **enp1s0**에서 **enp7s0**에 네트워크 트래픽을 미러링합니다.

#### 사전 요구 사항

- 네트워크 트래픽을 미러링할 네트워크 인터페이스입니다.

#### 절차

1. 네트워크 트래픽을 미러링할 네트워크 연결 프로필을 추가합니다.

```
nmcli connection add type ethernet ifname enp1s0 con-name enp1s0 autoconnect no
```

2. **10: handle**을 사용하여 송신(outgoing) 트래픽에 대해 **prio qdisc**를 **enp1s0**에 연결합니다.

```
nmcli connection modify enp1s0 +tc.qdisc "root prio handle 10:"
```

자식 없이 **prio qdisc**를 연결하면 필터를 첨부할 수 있습니다.



3.

**ffff: handle**를 사용하여 **Ingress** 트래픽에 **qdisc** 를 추가합니다.

```
nmcli connection modify enp1s0 +tc.qdisc "ingress handle ffff:"
```

4.

수신 및 송신 **qdiscs** 의 패킷과 일치하고 **enp7s0** 에 미러링하려면 다음 필터를 추가합니다.

```
nmcli connection modify enp1s0 +tc.tfilter "parent ffff: matchall action mirred egress mirror dev enp7s0"
```

```
nmcli connection modify enp1s0 +tc.tfilter "parent 10: matchall action mirred egress mirror dev enp7s0"
```

**matchall** 필터는 모든 패킷과 일치하며 **mirred** 작업은 패킷을 대상으로 리디렉션합니다.

5.

연결을 활성화합니다.

```
nmcli connection up enp1s0
```

## 검증 단계

1.

**tcpdump** 유틸리티를 설치합니다.

```
yum install tcpdump
```

2.

대상 장치에 미러링된 트래픽을 표시합니다(**enp7s0**).

```
tcpdump -i enp7s0
```

## 추가 리소스

- 

[tcpdump를 사용하여 네트워크 패킷을 캡처하는 방법](#)

## 19장. 모든 MAC 주소에서 트래픽을 허용하도록 네트워크 장치 구성

네트워크 장치는 일반적으로 컨트롤러가 수신하도록 프로그래밍된 패킷을 가로채고 읽습니다. 가상 스위치 또는 포트 그룹 수준에서 모든 MAC 주소의 트래픽을 수락하도록 네트워크 장치를 구성할 수 있습니다.

이 네트워크 모드를 사용하여 다음을 수행할 수 있습니다.

- 네트워크 연결 문제 진단
- 보안상의 이유로 네트워크 활동을 모니터링합니다.
- 네트워크의 개인 데이터 전송 또는 침입을 가로칩니다.

이 섹션에서는 `iproute2`, `nmcli` 또는 `nmstatectl` 유틸리티를 사용하여 모든 MAC 주소의 트래픽을 수락하도록 네트워크 장치를 구성하는 방법을 설명합니다. InfiniBand 를 제외한 모든 종류의 네트워크 장치에 대해 이 모드를 활성화할 수 있습니다.

### 19.1. IPROUTE2를 사용하여 모든 트래픽을 허용하도록 네트워크 장치 임시 구성

다음 절차에서는 MAC 주소와 관계없이 모든 트래픽을 허용하도록 네트워크 장치를 구성하는 방법을 설명합니다. `iproute2` 유틸리티를 사용하여 변경한 사항은 시스템이 재부팅된 후 일시적으로 손실됩니다.

#### 절차

1.

선택 사항: 모든 트래픽을 수신하려는 네트워크 인터페이스를 표시합니다.

```
ip a
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel
state DOWN group default qlen 1000
 link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
2: bond0: <NO-CARRIER,BROADCAST,MULTICAST,MASTER,UP> mtu 1500 qdisc
noqueue state DOWN group default qlen 1000
 link/ether 6a:fd:16:b0:83:5c brd ff:ff:ff:ff:ff:ff
3: wlp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
...
```

2.

이 속성을 활성화하거나 비활성화하려면 장치를 수정합니다.

•

**enp1s0**에 대해 **accept-all-mac-addresses** 모드를 활성화하려면 다음을 수행합니다.

```
ip link set enp1s0 promisc on
```

•

**enp1s0**에 대해 **accept-all-mac-addresses** 모드를 비활성화하려면 다음을 수행합니다.

```
ip link set enp1s0 promisc off
```

#### 검증 단계

•

**accept-all-mac-addresses** 모드가 활성화되어 있는지 확인하려면 다음을 수행하십시오.

```
ip link show enp1s0
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc fq_codel state DOWN mode DEFAULT group default qlen 1000
link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
```

장치 설명의 **PROMISC** 플래그는 모드가 활성화되었음을 나타냅니다.

## 19.2. NMCLI를 사용하여 모든 트래픽을 수락하도록 네트워크 장치를 영구적으로 설정

다음 절차에서는 **nmcli** 명령을 사용하여 **MAC** 주소와 관계없이 트래픽을 수락하도록 네트워크 장치를 구성하는 방법을 설명합니다.

#### 절차

1.

선택 사항: 모든 트래픽을 수신하려는 네트워크 인터페이스를 표시합니다.

```
ip a
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
2: bond0: <NO-CARRIER,BROADCAST,MULTICAST,MASTER,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
link/ether 6a:fd:16:b0:83:5c brd ff:ff:ff:ff:ff:ff
```

```
3: wlp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
...
```

없는 경우 새 연결을 만들 수 있습니다.

2.

이 속성을 활성화 또는 비활성화하려면 네트워크 장치를 수정합니다.

- **enp1s0**에 대해 **ethernet.accept-all-mac-addresses** 모드를 활성화하려면 다음을 수행합니다.

```
nmcli connection modify enp1s0 ethernet.accept-all-mac-addresses yes
```

- **enp1s0**에 대해 **accept-all-mac-addresses** 모드를 비활성화하려면 다음을 수행합니다.

```
nmcli connection modify enp1s0 ethernet.accept-all-mac-addresses no
```

3.

변경 사항을 적용하려면 연결을 다시 활성화합니다.

```
nmcli connection up enp1s0
```

검증 단계

- **ethernet.accept-all-mac-addresses** 모드가 활성화되어 있는지 확인하려면 다음을 수행하십시오.

```
nmcli connection show enp1s0
...
802-3-ethernet.accept-all-mac-addresses:1 (true)
```

**802-3-ethernet.accept-all-mac-addresses: true**는 모드가 활성화되었음을 나타냅니다.

### 19.3. NMSTATECTL을 사용하여 모든 트래픽을 수락하도록 네트워크 네트워크 장치를 영구적으로 구성

다음 절차에서는 **nmstatectl** 유틸리티를 사용하여 **MAC** 주소에 관계없이 모든 트래픽을 수락하도록 네트워크 장치를 구성하는 방법을 설명합니다.

## 사전 요구 사항

- **nmstate** 패키지가 설치되어 있습니다.
- 장치를 구성하는 데 사용한 **.yaml** 파일을 사용할 수 있습니다.

## 절차

1. **enp1s0** 연결의 기존 **enp1s0.yaml** 파일을 편집하고 다음 콘텐츠를 추가합니다.

```

interfaces:
 - name: enp1s0
 type: ethernet
 state: up
 accept-all-mac-address: true
```

2. 네트워크 설정을 적용합니다.

```
nmstatectl apply ~/enp1s0.yaml
```

## 검증 단계

- **802-3-ethernet.accept-all-mac-addresses** 모드가 활성화되어 있는지 확인하려면 다음을 수행하십시오.

```
nmstatectl show enp1s0
interfaces:
 - name: enp1s0
 type: ethernet
 state: up
 accept-all-mac-addresses: true
...
```

**802-3-ethernet.accept-all-mac-addresses: true** 는 모드가 활성화되었음을 나타냅니다.

## 추가 리소스

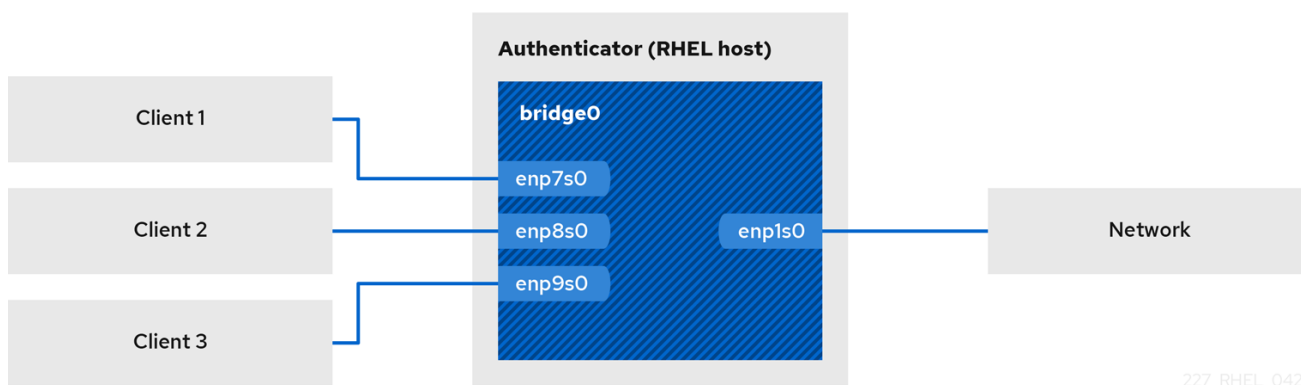
- **nmstatectl** 에 대한 자세한 내용은 **nmstatectl(8)** 매뉴얼 페이지를 참조하십시오.

- 자세한 구성 예제는 **`/usr/share/doc/nmstate/examples/`** 디렉토리를 참조하십시오.

## 20장. FREERADIUS 백엔드와 함께 HOSTAPD를 사용하여 LAN 클라이언트에 대한 802.1X 네트워크 인증 서비스 설정

**IEEE 802.1X** 표준은 무단 클라이언트로부터 네트워크를 보호하는 보안 인증 및 권한 부여 방법을 정의합니다. **hostapd** 서비스 및 **FreeRADIUS**를 사용하면 네트워크에서 네트워크 액세스 제어(NAC)를 제공할 수 있습니다.

이 설명서에서 **RHEL** 호스트는 다른 클라이언트를 기존 네트워크와 연결하는 브리지 역할을 합니다. 그러나 **RHEL** 호스트는 인증된 클라이언트만 네트워크에 대한 액세스 권한을 부여합니다.



227\_RHEL\_0422

### 20.1. 사전 요구 사항

- **FreeRADIUS의 깔끔한 설치.**

**freeradius** 패키지가 이미 설치된 경우 **/etc/raddb/** 디렉토리를 제거한 다음 패키지를 다시 설치합니다. **/etc/raddb/** 디렉토리의 권한과 심볼릭 링크가 다르므로 **yum reinstall** 명령을 사용하여 패키지를 다시 설치하지 마십시오.

### 20.2. 인증기에서 브리지 설정

네트워크 브리지는 **MAC** 주소 테이블에 따라 호스트와 네트워크 간에 트래픽을 전달하는 링크 계층 장치입니다. **RHEL**을 **802.1X** 인증기로 설정하는 경우 인증 및 **LAN** 인터페이스를 브리지에 수행할 인터페이스를 모두 추가합니다.

#### 사전 요구 사항

- 서버에는 여러 이더넷 인터페이스가 있습니다.

## 절차

1.

*브리지 인터페이스를 만듭니다.*

```
nmcli connection add type bridge con-name br0 ifname br0
```

2.

*이더넷 인터페이스를 브리지에 할당합니다.*

```
nmcli connection add type ethernet slave-type bridge con-name br0-port1 ifname enp1s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port2 ifname enp7s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port3 ifname enp8s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port4 ifname enp9s0 master br0
```

3.

*브리지를 활성화하여 LAN(EAPOL) 패킷을 통해 확장 가능한 인증 프로토콜을 전달합니다.*

```
nmcli connection modify br0 group-forward-mask 8
```

4.

*포트를 자동으로 활성화하도록 연결을 구성합니다.*

```
nmcli connection modify br0 connection.autoconnect-slaves 1
```

5.

*연결을 활성화합니다.*

```
nmcli connection up br0
```

## 검증

1.

*특정 브릿지의 포트에 해당하는 이더넷 장치의 링크 상태를 표시합니다.*

```
ip link show master br0
3: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master br0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
...
```



2.

**br0** 장치에서 **EAPOL** 패킷 전달이 활성화되어 있는지 확인합니다.

```
cat /sys/class/net/br0/bridge/group_fwd_mask
0x8
```

명령이 **0x8** 을 반환하면 전달이 활성화됩니다.

추가 리소스

•

**nm-settings(5)** 도움말 페이지

### 20.3. FREERADIUS의 인증서 요구 사항

안전한 **FreeRADIUS** 서비스의 경우 다른 목적으로 **TLS** 인증서가 필요합니다.

•

서버에 대한 암호화된 연결을 위한 **TLS** 서버 인증서입니다. 신뢰할 수 있는 **CA**(인증 기관)를 사용하여 인증서를 발행합니다.

서버 인증서를 사용하려면 **EKU**(Extended Key usage) 필드를 **TLS** 웹 서버 인증으로 설정해야 합니다.

•

확장된 인증 프로토콜 전송 계층 보안(**EAP-TLS**)에 대해 동일한 **CA**에서 발행한 클라이언트 인증서. **EAP-TLS**는 인증서 기반 인증을 제공하며 기본적으로 활성화됩니다.

클라이언트 인증서를 사용하려면 **EKU** 필드를 **TLS** 웹 클라이언트 인증으로 설정해야 합니다.

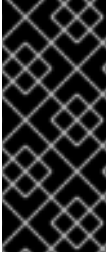


주의

연결을 보호하려면 회사의 **CA**를 사용하거나 고유한 **CA**를 생성하여 **FreeRADIUS**에 대한 인증서를 발행하십시오. 공용 **CA**를 사용하는 경우 이를 통해 사용자를 인증하고 **EAP-TLS**에 대한 클라이언트 인증서를 발행할 수 있습니다.

### 20.4. 테스트 목적으로 FREERADIUS 서버에 인증서 세트 생성

테스트 목적으로 **freeradius** 패키지는 **/etc/raddb/certs/** 디렉터리에 스크립트 및 구성 파일을 설치하여 고유한 **CA**(인증 기관) 및 발급 인증서를 생성합니다.



#### 중요

기본 구성을 사용하는 경우 이러한 스크립트에서 생성된 인증서는 **60일** 후에 만료되고 키는 안전하지 않은 암호("whatever")를 사용합니다. 그러나 **CA**, 서버 및 클라이언트 구성을 사용자 지정할 수 있습니다.

절차를 수행한 후 이 문서의 뒷부분에 필요한 다음 파일이 생성됩니다.

- **/etc/raddb/certs/ca.pem: CA 인증서**
- **/etc/raddb/certs/server.key: 서버 인증서의 개인 키**
- **/etc/raddb/certs/server.pem: 서버 인증서**
- **/etc/raddb/certs/client.key: 클라이언트 인증서의 개인 키**
- **/etc/raddb/certs/client.pem: 클라이언트 인증서**

#### 사전 요구 사항

- **freeradius** 패키지를 설치했습니다.

#### 절차

1. **/etc/raddb/certs/ 디렉터리로 변경합니다.**

```
cd /etc/raddb/certs/
```

2. 선택 사항: **CA** 구성을 사용자 지정합니다.

```

...
[req]
default_bits = 2048
input_password = ca_password
output_password = ca_password
...
[certificate_authority]
countryName = US
stateOrProvinceName = North Carolina
localityName = Raleigh
organizationName = Example Inc.
emailAddress = admin@example.org
commonName = "Example Certificate Authority"
...

```

3.

선택 사항: 서버 구성을 사용자 지정합니다.

```

...
[CA_default]
default_days = 730
...
[req]
distinguished_name = server
default_bits = 2048
input_password = key_password
output_password = key_password
...
[server]
countryName = US
stateOrProvinceName = North Carolina
localityName = Raleigh
organizationName = Example Inc.
emailAddress = admin@example.org
commonName = "Example Server Certificate"
...

```

4.

선택 사항: 클라이언트 구성을 사용자 지정합니다.

```

...
[CA_default]
default_days = 365
...
[req]
distinguished_name = client
default_bits = 2048
input_password = password_on_private_key
output_password = password_on_private_key
...
[client]
countryName = US
stateOrProvinceName = North Carolina

```

```

localityName = Raleigh
organizationName = Example Inc.
emailAddress = user@example.org
commonName = user@example.org
...

```

5.

인증서를 생성합니다.

```
make all
```

6.

`/etc/raddb/certs/server.pem` 파일의 그룹을 `radiusd` 로 변경합니다.

```
chgrp radiusd /etc/raddb/certs/server.pem*
```

추가 리소스

•

`/etc/raddb/certs/README.md`

## 20.5. EAP를 사용하여 네트워크 클라이언트를 안전하게 인증하도록 FREERADIUS 구성

**freeradius**는 확장 가능한 인증 프로토콜(EAP)의 다양한 방법을 지원합니다. 그러나 보안 네트워크의 경우 이 문서에서는 다음과 같은 보안 EAP 인증 방법만 지원하도록 **FreeRADIUS**를 구성하는 방법을 설명합니다.

•

**EAP-TLS**(전송 계층 보안)는 보안 TLS 연결을 사용하여 인증서를 사용하여 클라이언트를 인증합니다. EAP-TLS를 사용하려면 각 네트워크 클라이언트에 대해 TLS 클라이언트 인증서와 서버의 서버 인증서가 필요합니다. 동일한 CA(인증 기관)에서 인증서를 발급해야 합니다. 사용하는 모든 클라이언트 인증서가 **FreeRADIUS** 서버에 대해 인증할 수 있으므로 항상 자체 CA를 사용하여 인증서를 생성합니다.

•

**EAP-TTLS**(터널링된 전송 계층 보안)는 보안 TLS 연결을 사용하고 암호 인증 프로토콜(PAP) 또는 챌린지 핸드셰이크 인증 프로토콜(CHAP)과 같은 메커니즘을 사용하여 클라이언트를 인증합니다. EAP-TTLS를 사용하려면 TLS 서버 인증서가 필요합니다.

•

**EAP-PEAP**(보안 확장 가능 인증 프로토콜)는 보안 TLS 연결을 외부 인증 프로토콜로 사용하여 터널을 설정합니다. 인증자는 **RADIUS** 서버의 인증서를 인증합니다. 그런 다음 공급자는 **Microsoft challenge** 핸드셰이크 인증 프로토콜 버전 2 (MS-CHAPv2) 또는 기타 방법을 사용하여 암호화된 터널을 통해 인증합니다.



## 참고

기본 **FreeRADIUS** 구성 파일은 문서 역할을 하며 모든 매개변수와 지시문을 설명합니다. 특정 기능을 비활성화하려면 구성 파일에서 해당 부분을 제거하는 대신 주석 처리하십시오. 이를 통해 구성 파일 및 포함된 문서의 구조를 유지할 수 있습니다.

## 사전 요구 사항

- **freeradius** 패키지를 설치했습니다.
- **/etc/raddb/** 디렉토리의 구성 파일은 변경되지 않으며 **freeradius** 패키지에서 제공하는 것과 같습니다.
- 서버에 다음 파일이 있습니다.
  - **FreeRADIUS** 호스트의 TLS 개인 키: **/etc/raddb/certs/server.key**
  - **FreeRADIUS** 호스트의 TLS 서버 인증서: **/etc/raddb/certs/server.pem**
  - **TLS CA** 인증서: **/etc/raddb/certs/ca.pem**

파일을 다른 위치에 저장하거나 이름이 다른 경우 **/etc/raddb/mods-available/eap** 파일의 **private\_key\_file**, **certificate\_file** 및 **ca\_file** 매개변수를 설정합니다.

## 절차

1. **Diffie-Hellman (DH)** 매개 변수가 있는 **/etc/raddb/certs/dh** 가 없으면 새로 만듭니다. 예를 들어 2048비트 소수로 **DH** 파일을 만들려면 다음을 입력합니다.

```
openssl dhparam -out /etc/raddb/certs/dh 2048
```

보안상의 이유로 2048비트 소수보다 작은 **DH** 파일을 사용하지 마십시오. 비트 수에 따라 파일 작성에 몇 분이 걸릴 수 있습니다.

2. **TLS** 개인 키, 서버 인증서, **CA** 인증서 및 **DH** 매개변수로 파일에 대한 보안 권한을 설정합니다

다.

```
chmod 640 /etc/raddb/certs/server.key /etc/raddb/certs/server.pem
/etc/raddb/certs/ca.pem /etc/raddb/certs/dh
chown root:radiusd /etc/raddb/certs/server.key /etc/raddb/certs/server.pem
/etc/raddb/certs/ca.pem /etc/raddb/certs/dh
```

3.

`/etc/raddb/mods-available/eap` 파일을 편집합니다.

a.

`private_key_password` 매개변수에 개인 키의 암호를 설정합니다.

```
eap {
 ...
 tls-config tls-common {
 ...
 private_key_password = key_password
 ...
 }
}
```

b.

환경에 따라 `eap` 지시문의 `default_eap_type` 매개변수를 사용하는 기본 **EAP** 유형으로 설정합니다.

```
eap {
 ...
 default_eap_type = tls
 ...
}
```

보안 환경의 경우 `tls,tls` 또는 `peap` 만 사용하십시오.

c.

`md5` 지시문을 주석 처리하여 안전하지 않은 **EAP-MD5** 인증 방법을 비활성화합니다.

```
eap {
 ...
 # md5 {
 # }
 ...
}
```

기본 구성 파일에서 기타 비보안 **EAP** 인증 방법은 기본적으로 주석 처리됩니다.

4.

`/etc/raddb/sites-available/default` 파일을 편집하고 **eap** 이외의 모든 인증 방법을 주석 처리합니다.

```
authenticate {
 ...
 # Auth-Type PAP {
 # pap
 # }

 # Auth-Type CHAP {
 # chap
 # }

 # Auth-Type MS-CHAP {
 # mschap
 # }

 # mschap

 # digest
 ...
}
```

이렇게 하면 **EAP**만 활성화되며 일반 텍스트 인증 방법을 비활성화합니다.

5.

`/etc/raddb/clients.conf` 파일을 편집합니다.

a.

**localhost** 및 **localhost\_ipv6** 클라이언트 지시문에서 보안 암호를 설정합니다.

```
client localhost {
 ipaddr = 127.0.0.1
 ...
 secret = client_password
 ...
}

client localhost_ipv6 {
 ipv6addr = ::1
 secret = client_password
}
```

b.

네트워크 인증기와 같은 **RADIUS** 클라이언트가 원격 호스트의 **FreeRADIUS** 서비스에 액세스할 수 있어야 하는 경우 이에 대해 해당 클라이언트 지시문을 추가합니다.

```
client hostapd.example.org {
 ipaddr = 192.0.2.2/32
```

```
secret = client_password
}
```

**ipaddr** 매개변수는 **IPv4** 및 **IPv6** 주소를 허용하고 선택적 **CIDR(Classless inter-domain routing)** 표기법을 사용하여 범위를 지정할 수 있습니다. 그러나 이 매개 변수에는 하나의 값만 설정할 수 있습니다. 예를 들어 **IPv4** 및 **IPv6** 주소에 대한 액세스 권한을 부여하려면 두 개의 클라이언트 지시문을 추가합니다.

클라이언트 지시문에 대해 설명적인 이름(예: **hostname**) 또는 **IP** 범위 사용 위치를 설명하는 단어를 사용합니다.

6.

**EAP-TTLS** 또는 **EAP-PEAP**를 사용하려면 사용자를 **/etc/raddb/users** 파일에 추가하십시오.

```
example_user Cleartext-Password := "user_password"
```

인증서 기반 인증(**EAP-TLS**)을 사용해야 하는 사용자의 경우 항목을 추가하지 마십시오.

7.

구성 파일을 확인합니다.

```
radiusd -XC
...
Configuration appears to be OK
```

8.

**radiusd** 서비스를 활성화하고 시작합니다.

```
systemctl enable --now radiusd
```

## 검증

•

**FreeRADIUS** 서버 또는 인증자를 대상으로 **EAP-TTLS** 인증 테스트

•

**FreeRADIUS** 서버 또는 인증자를 대상으로 **EAP-TLS** 인증 테스트

## 문제 해결

1.

**radiusd** 서비스를 중지합니다.



```
systemctl stop radiusd
```

2.

디버그 모드에서 서비스를 시작합니다.

```
radiusd -X
```

```
...
```

```
Ready to process requests
```

3.

확인 섹션에서 참조하는 **FreeRADIUS** 호스트에서 인증 테스트를 수행합니다.

다음 단계

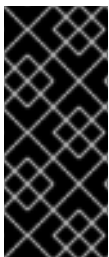
•

사용하지 않는 인증 방법 및 기타 기능을 비활성화합니다.

## 20.6. 유선 네트워크에서 HOSTAPD를 인증자로 구성

호스트 액세스 포인트 데몬(**hostapd**) 서비스는 **802.1X** 인증을 제공하기 위해 유선 네트워크에서 인증자 역할을 할 수 있습니다. 이를 위해 **hostapd** 서비스에는 클라이언트를 인증하는 **RADIUS** 서버가 필요합니다.

**hostapd** 서비스는 통합 **RADIUS** 서버를 제공합니다. 그러나 테스트 목적으로만 통합 **RADIUS** 서버를 사용하십시오. 프로덕션 환경의 경우 다양한 인증 방법 및 액세스 제어와 같은 추가 기능을 지원하는 **FreeRADIUS** 서버를 사용하십시오.



중요

**hostapd** 서비스가 트래픽 플레인과 상호 작용하지 않습니다. 서비스는 인증자 역할을 합니다. 예를 들어 **hostapd** 제어 인터페이스를 사용하여 인증 이벤트 결과에 따라 트래픽을 허용하거나 거부하는 스크립트 또는 서비스를 사용합니다.

사전 요구 사항

•

**hostapd** 패키지가 설치되어 있어야 합니다.

•

**FreeRADIUS** 서버가 구성되었으며 클라이언트를 인증할 준비가 되었습니다.

절차

1.

다음 콘텐츠를 사용하여 `/etc/hostapd/hostapd.conf` 파일을 생성합니다.

```
General settings of hostapd
=====

Control interface settings
ctrl_interface=/var/run/hostapd
ctrl_interface_group=wheel

Enable logging for all modules
logger_syslog=-1
logger_stdout=-1

Log level
logger_syslog_level=2
logger_stdout_level=2

Wired 802.1X authentication
=====

Driver interface type
driver=wired

Enable IEEE 802.1X authorization
ieee8021x=1

Use port access entry (PAE) group address
(01:80:c2:00:00:03) when sending EAPOL frames
use_pae_group_addr=1

Network interface for authentication requests
interface=br0

RADIUS client configuration
=====

Local IP address used as NAS-IP-Address
own_ip_addr=192.0.2.2

Unique NAS-Identifier within scope of RADIUS server
nas_identifier=hostapd.example.org

RADIUS authentication server
auth_server_addr=192.0.2.1
auth_server_port=1812
auth_server_shared_secret=client_password

RADIUS accounting server
acct_server_addr=192.0.2.1
acct_server_port=1813
acct_server_shared_secret=client_password
```

이 구성에 사용된 매개변수에 대한 자세한 내용은 `/usr/share/doc/hostapd/hostapd.conf` 예제 설정 파일에서 해당 설명을 참조하십시오.

2.

**hostapd** 서비스를 활성화하고 시작합니다.

```
systemctl enable --now hostapd
```

#### 검증

- 다음 내용을 참조하십시오.
  - [FreeRADIUS 서버 또는 인증자를 대상으로 EAP-TTLS 인증 테스트](#)
  - [FreeRADIUS 서버 또는 인증자를 대상으로 EAP-TLS 인증 테스트](#)

#### 문제 해결

1.

**hostapd** 서비스를 중지합니다.

```
systemctl stop hostapd
```

2.

디버그 모드에서 서비스를 시작합니다.

```
hostapd -d /etc/hostapd/hostapd.conf
```

3.

확인 섹션에서 참조하는 **FreeRADIUS** 호스트에서 인증 테스트를 수행합니다.

#### 추가 리소스

- [hostapd.conf\(5\) man page](#)
- [/usr/share/doc/hostapd/hostapd.conf](#)

터널링된 전송 계층 보안(EAP-TTLS)을 통한 확장 가능한 인증 프로토콜(EAP)을 사용하여 인증이 예상대로 작동하는지 테스트하려면 다음 절차를 실행하십시오.

- **FreeRADIUS 서버를 설정한 후**
- **hostapd 서비스를 802.1X 네트워크 인증에 대한 인증 도구로 설정한 후.**

이 프로세스에서 사용된 테스트 유틸리티의 출력은 EAP 통신에 대한 추가 정보를 제공하고 문제를 더 버깅하는 데 도움이 됩니다.

#### 사전 요구 사항

- 인증하려는 경우 다음을 수행합니다.
  - **FreeRADIUS 서버:**
    - **hostapd 패키지에서 제공하는 Multus ol\_test 유틸리티가 설치됩니다.**
    - 이 절차를 실행하는 클라이언트는 **FreeRADIUS 서버의 클라이언트 데이터베이스**에 권한이 있습니다.
  - 동일한 이름의 패키지에서 제공하는 **wpa\_supplicant** 유틸리티인 인증자가 설치됩니다.
- **/etc/pki/tls/certs/ca.pem 파일에 CA(인증 기관) 인증서를 저장하셨습니다.**

#### 절차

1. 다음 콘텐츠를 사용하여 **/etc/wpa\_supplicant/wpa\_supplicant-TTLS.conf** 파일을 생성합니다.

```
ap_scan=0

network={
 eap=TTLS
```

```

eapol_flags=0
key_mgmt=IEEE8021X

Anonymous identity (sent in unencrypted phase 1)
Can be any string
anonymous_identity="anonymous"

Inner authentication (sent in TLS-encrypted phase 2)
phase2="auth=PAP"
identity="example_user"
password="user_password"

CA certificate to validate the RADIUS server's identity
ca_cert="/etc/pki/tls/certs/ca.pem"
}

```

2.

인증하려면 다음을 수행합니다.

•

**FreeRADIUS** 서버는 다음을 입력합니다.

```

eapol_test -c /etc/wpa_supplicant/wpa_supplicant-TTLS.conf -a 192.0.2.1 -s
client_password
...
EAP: Status notification: remote certificate verification (param=success)
...
CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
...
SUCCESS

```

**a** 옵션은 **FreeRADIUS** 서버의 IP 주소를 정의하고, **-s** 옵션은 **FreeRADIUS** 서버의 클라이언트 구성에서 명령을 실행하는 호스트의 암호를 지정합니다.

•

인증자가 다음과 같이 입력합니다.

```

wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant-TTLS.conf -D wired -i
enp0s31f6
...
enp0s31f6: CTRL-EVENT-EAP-SUCCESS EAP authentication completed
successfully
...

```

**i** 옵션은 **wpa\_supplicant** 가 LAN(EAPOL) 패킷을 통해 확장된 인증 프로토콜을 전송하는 네트워크 인터페이스 이름을 지정합니다.

자세한 디버깅 정보를 보려면 명령에 **-d** 옵션을 전달합니다.

추가 리소스

- `/usr/share/doc/wpa_supplicant/wpa_supplicant.conf`

## 20.8. FREERADIUS 서버 또는 인증자를 대상으로 EAP-TLS 인증 테스트

**EAP(extendable authentication protocol)** 전송 계층 보안(**EAP-TLS**)을 사용하여 인증이 예상대로 작동하는지 테스트하려면 다음 절차를 실행하십시오.

- **FreeRADIUS** 서버를 설정한 후
- **hostapd** 서비스를 **802.1X** 네트워크 인증에 대한 인증 도구로 설정한 후.

이 프로세스에서 사용된 테스트 유틸리티의 출력은 **EAP** 통신에 대한 추가 정보를 제공하고 문제를 디버깅하는 데 도움이 됩니다.

사전 요구 사항

- 인증하려는 경우 다음을 수행합니다.
  - **FreeRADIUS** 서버:
    - **hostapd** 패키지에서 제공하는 **Multus ol\_test** 유틸리티가 설치됩니다.
    - 이 절차를 실행하는 클라이언트는 **FreeRADIUS** 서버의 클라이언트 데이터베이스에 권한이 있습니다.
  - 동일한 이름의 패키지에서 제공하는 **wpa\_supplicant** 유틸리티인 인증자가 설치됩니다.
- `/etc/pki/tls/certs/ca.pem` 파일에 **CA**(인증 기관) 인증서를 저장하셨습니다.

- 클라이언트 인증서를 발급한 **CA**는 **FreeRADIUS** 서버의 서버 인증서를 발급한 것과 동일합니다.
- 클라이언트 인증서를 `/etc/pki/tls/certs/client.pem` 파일에 저장했습니다.
- 클라이언트의 개인 키를 `/etc/pki/tls/private/client.key`에 저장했습니다.

## 절차

1. 다음 콘텐츠를 사용하여 `/etc/wpa_supplicant/wpa_supplicant-TLS.conf` 파일을 생성합니다.

```
ap_scan=0

network={
 eap=TLS
 eapol_flags=0
 key_mgmt=IEEE8021X

 identity="user@example.org"
 client_cert="/etc/pki/tls/certs/client.pem"
 private_key="/etc/pki/tls/private/client.key"
 private_key_passwd="password_on_private_key"

 # CA certificate to validate the RADIUS server's identity
 ca_cert="/etc/pki/tls/certs/ca.pem"
}
```

2. 인증하려면 다음을 수행합니다.

- **FreeRADIUS** 서버는 다음을 입력합니다.

```
eapol_test -c /etc/wpa_supplicant/wpa_supplicant-TLS.conf -a 192.0.2.1 -s
client_password
...
EAP: Status notification: remote certificate verification (param=success)
...
CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
...
SUCCESS
```

**a** 옵션은 **FreeRADIUS** 서버의 IP 주소를 정의하고, **-s** 옵션은 **FreeRADIUS** 서버의 클라이언트 구성에서 명령을 실행하는 호스트의 암호를 지정합니다.

- 인증자가 다음과 같이 입력합니다.

```
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant-TLS.conf -D wired -i
enp0s31f6
...
enp0s31f6: CTRL-EVENT-EAP-SUCCESS EAP authentication completed
successfully
...
```

*i* 옵션은 **wpa\_supplicant** 가 LAN(EAPOL) 패킷을 통해 확장된 인증 프로토콜을 전송하는 네트워크 인터페이스 이름을 지정합니다.

자세한 디버깅 정보를 보려면 명령에 **-d** 옵션을 전달합니다.

#### 추가 리소스

- [\*\*/usr/share/doc/wpa\\_supplicant/wpa\\_supplicant.conf\*\*](#)

## 20.9. HOSTAPD 인증 이벤트를 기반으로 트래픽 차단 및 허용

**hostapd** 서비스가 트래픽 플레인과 상호 작용하지 않습니다. 서비스는 인증자 역할을 합니다. 그러나 인증 이벤트 결과에 따라 트래픽을 허용하고 거부하는 스크립트를 작성할 수 있습니다.



#### 중요

이 절차는 지원되지 않으며 엔터프라이즈급 솔루션이 아닙니다. **hostapd\_cli** 에서 검색한 이벤트를 평가하여 트래픽을 차단하거나 허용하는 방법만 보여줍니다.

**802-1x-tr-backend systemd** 서비스가 시작되면 **RHEL**은 LAN (EAPOL) 패킷에 대해 수신 대기 중인 포트의 모든 트래픽을 차단하고 **hostapd\_cli** 유틸리티를 사용하여 **hostapd** 제어 인터페이스에 연결합니다. **/usr/local/bin/802-1x-tr-backend** 스크립트는 이벤트를 평가합니다. **hostapd\_cli** 에서 수신한 다양한 이벤트에 따라 스크립트에서 **MAC** 주소에 대한 트래픽을 허용하거나 차단합니다. **802-1x-tr-backend** 서비스가 중지되면 모든 트래픽이 자동으로 다시 허용됩니다.

**hostapd** 서버에서 다음 절차를 수행합니다.



## 사전 요구 사항

- **hostapd** 서비스가 구성되어 있으며 서비스는 클라이언트를 인증할 준비가 되어 있습니다.

## 절차

1.

다음 콘텐츠를 사용하여 **/usr/local/bin/802-1x-tr-management** 파일을 만듭니다.

```
#!/bin/sh

if ["$1" == "xblock_all"]
then

 nft delete table bridge tr-mgmt-br0 2>/dev/null || true
 nft -f - << EOF
table bridge tr-mgmt-br0 {
 set allowed_macs {
 type ether_addr
 }

 chain accesscontrol {
 ether saddr @allowed_macs accept
 ether daddr @allowed_macs accept
 drop
 }

 chain forward {
 type filter hook forward priority 0; policy accept;
 meta ibrname "br0" jump accesscontrol
 }
}
EOF
 echo "802-1x-tr-mgmt Blocking all traffic through br0. Traffic for given host will be
allowed after 802.1x authentication"

elif ["$1" == "xallow_all"]
then

 nft delete table bridge tr-mgmt-br0
 echo "802-1x-tr-mgmt Allowed all forwarding again"

fi

case ${2:-NOTANEVENT} in

 AP-STA-CONNECTED | CTRL-EVENT-EAP-SUCCESS | CTRL-EVENT-EAP-
 SUCCESS2)
 nft add element bridge tr-mgmt-br0 allowed_macs { $3 }
 echo "$1: Allowed traffic from $3"
 ;;

 AP-STA-DISCONNECTED | CTRL-EVENT-EAP-FAILURE)
```

```
nft delete element bridge tr-mgmt-br0 allowed_macs { $3 }
echo "802-1x-tr-mgmt $1: Denied traffic from $3"
;;
```

```
esac
```

2.

다음 콘텐츠를 사용하여 `/etc/systemd/system/802-1x-tr-backend@.service` systemd 서비스 파일을 만듭니다.

```
[Unit]
Description=Example 802.1x traffic management for hostapd
After=hostapd.service
After=sys-devices-virtual-net-%i.device

[Service]
Type=simple
ExecStartPre=/bin/sh -c '/usr/sbin/tc qdisc del dev %i ingress > /dev/null 2>&1'
ExecStartPre=/bin/sh -c '/usr/sbin/tc qdisc del dev %i clsact > /dev/null 2>&1'
ExecStartPre=/usr/sbin/tc qdisc add dev %i clsact
ExecStartPre=/usr/sbin/tc filter add dev %i ingress pref 10000 protocol 0x888e
matchall action ok index 100
ExecStartPre=/usr/sbin/tc filter add dev %i ingress pref 10001 protocol all matchall
action drop index 101
ExecStart=/usr/sbin/hostapd_cli -i %i -a /usr/local/bin/802-1x-tr-mgmt
ExecStopPost=/usr/sbin/tc qdisc del dev %i clsact

[Install]
WantedBy=multi-user.target
```

3.

`systemd`를 다시 로드합니다.

```
systemctl daemon-reload
```

4.

`hostapd` 인터페이스 이름이 수신 대기 상태로 `802-1x-tr-backend` 서비스를 활성화하고 시작합니다.

```
systemctl enable --now 802-1x-tr-mgmt@br0.service
```

## 검증

•

네트워크로 인증할 수 있습니다. 다음 내용을 참조하십시오.

○

[FreeRADIUS 서버 또는 인증자를 대상으로 EAP-TTLS 인증 테스트](#)

- 

**FreeRADIUS 서버 또는 인증자를 대상으로 EAP-TLS 인증 테스트**

추가 리소스

- 

***systemd.service(5) man page***

**21장. 802.1X 표준을 사용하여 파일 시스템에 저장된 인증서로 RHEL 클라이언트를 네트워크에 인증**

관리자는 **IEEE 802.1X** 표준을 기반으로 포트 기반 **NAT(Network Access Control)**를 자주 사용하여 네트워크를 권한 없는 **LAN** 및 **Wi-Fi** 클라이언트로부터 보호합니다. 이 섹션의 절차에서는 네트워크 인증을 구성하는 다양한 옵션에 대해 설명합니다.

**21.1. NMCLI를 사용하여 기존 이더넷 연결에서 802.1X 네트워크 인증 구성**

**nmcli** 유틸리티를 사용하여 네트워크에 자체적으로 인증하도록 클라이언트를 구성할 수 있습니다. 이 절차에서는 네트워크에 인증하기 위해 **enp1s0**이라는 기존 **NetworkManager** 이더넷 연결 프로필에서 **TLS** 인증을 구성하는 방법을 설명합니다.

**사전 요구 사항**

- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 이더넷 연결 프로필은 **NetworkManager**에 있으며 유효한 **IP** 구성이 있습니다.
- 클라이언트에는 **TLS** 인증에 필요한 다음 파일이 있습니다.
  - 저장된 클라이언트 키는 **/etc/pki/tls/private/client.key** 파일에 있으며, **root** 사용자만 파일을 소유하고 읽을 수 있습니다.
  - 클라이언트 인증서는 **/etc/pki/tls/certs/client.crt** 파일에 저장됩니다.
  - **CA(인증 기관)** 인증서는 **/etc/pki/tls/certs/ca.crt** 파일에 저장됩니다.
- **wpa\_supplicant** 패키지가 설치되어 있습니다.

**절차**

1. 확장 가능 인증 프로토콜(**EAP**)을 **tls**로 설정하고 클라이언트 인증서 및 키 파일에 대한 경로를 설정합니다.

```
nmcli connection modify enp1s0 802-1x.eap tls 802-1x.client-cert
/etc/pki/tls/certs/client.crt 802-1x.private-key /etc/pki/tls/certs/certs/client.key
```

단일 명령에서 **802-1x.eap**, **802-1x.client-cert** 및 **802-1x.private-key** 매개변수를 설정해야 합니다.

2.

**CA** 인증서의 경로를 설정합니다.

```
nmcli connection modify enp1s0 802-1x.ca-cert /etc/pki/tls/certs/ca.crt
```

3.

인증서에 사용된 사용자의 **ID**를 설정합니다.

```
nmcli connection modify enp1s0 802-1x.identity user@example.com
```

4.

선택적으로 구성에 암호를 저장합니다.

```
nmcli connection modify enp1s0 802-1x.private-key-password password
```

#### 중요

기본적으로 **NetworkManager**는 **root** 사용자만 읽을 수 있는 **/etc/sysconfig/network-scripts/keys-connection\_name** 파일에 암호를 일반 텍스트로 저장합니다. 그러나 구성 파일의 일반 텍스트 암호는 보안 위험이 될 수 있습니다.

보안을 강화하려면 **802-1x.password-flags** 매개 변수를 **0x1**로 설정합니다. 이 설정을 사용하면 **GNOME** 데스크탑 환경 또는 **nm-applet**이 실행 중인 서버에서 **NetworkManager**가 이러한 서비스에서 암호를 검색합니다. 다른 경우에는 **NetworkManager**에서 암호를 묻는 메시지를 표시합니다.

5.

연결 프로필을 활성화합니다.

```
nmcli connection up enp1s0
```

#### 검증 단계

- 

네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

## 추가 리소스

- [이더넷 연결 구성](#)
- [nm- settings\(5\) 매뉴얼 페이지의 802-1x 설정 섹션](#)
- [nmcli\(1\) 도움말 페이지](#)

## 21.2. NMSTATECTL을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성

**nmstate** 유틸리티를 사용하여 802.1X 표준을 사용하여 클라이언트를 인증하는 이더넷 연결을 만들 수 있습니다. 이 절차에서는 다음 설정으로 **enp1s0** 인터페이스에 이더넷 연결을 추가하는 방법을 설명합니다.

- 정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 192.0.2.254
- IPv6 기본 게이트웨이 - 2001:db8:1::fffe
- IPv4 DNS 서버 - 192.0.2.200
- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com
- TLS EAP(Extensible Authentication Protocol)를 사용한 802.1x 네트워크 인증



## 참고

**nmstate 라이브러리는 TLS EAP 메서드만 지원합니다.**

## 사전 요구 사항

- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 관리 노드는 **NetworkManager**를 사용합니다.
- 클라이언트에는 **TLS** 인증에 필요한 다음 파일이 있습니다.
  - 저장된 클라이언트 키는 **/etc/pki/tls/private/client.key** 파일에 있으며, **root** 사용자만 파일을 소유하고 읽을 수 있습니다.
  - 클라이언트 인증서는 **/etc/pki/tls/certs/client.crt** 파일에 저장됩니다.
  - CA(인증 기관) 인증서는 **/etc/pki/tls/certs/ca.crt** 파일에 저장됩니다.

## 절차

1. 다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/create-ethernet-profile.yml**)을 만듭니다.

```

interfaces:
- name: enp1s0
 type: ethernet
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
```

```

dhcp: false
802.1x:
 ca-cert: /etc/pki/tls/certs/ca.crt
 client-cert: /etc/pki/tls/certs/client.crt
 eap-methods:
 - tls
 identity: client.example.org
 private-key: /etc/pki/tls/private/client.key
 private-key-password: password
routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: enp1s0
 - destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: enp1s0
dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb

```

2.

시스템에 설정을 적용합니다.

```
nmstatectl apply ~/create-ethernet-profile.yml
```

## 검증

•

네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

### 21.3. RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 802.1X 표준을 사용하여 클라이언트를 인증하는 이더넷 연결 생성을 자동화할 수 있습니다. 이 절차에서는 **Ansible** 플레이북을 실행하여 **enp1s0** 인터페이스에 대한 이더넷 연결을 원격으로 추가하는 방법을 설명합니다.

•

정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크

•

정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크



- **IPv4 기본 게이트웨이 - 192.0.2.254**
- **IPv6 기본 게이트웨이 - 2001:db8:1::fffe**
- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**
- **TLS EAP(Extensible Authentication Protocol)를 사용한 802.1x 네트워크 인증**

**Ansible** 제어 노드에서 이 절차를 실행합니다.

#### 사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 관리 노드에 대한 적절한 **sudo** 권한이 있어야 합니다.
- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 관리 노드는 **NetworkManager**를 사용합니다.
- **TLS** 인증에 필요한 다음 파일은 제어 노드에 있습니다.
  - 클라이언트 키는 **/srv/data/client.key** 파일에 저장됩니다.

- 클라이언트 인증서는 **/srv/data/client.crt** 파일에 저장됩니다.
- CA(인증 기관) 인증서는 **/srv/data/ca.crt** 파일에 저장됩니다.

## 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/enable-802.1x.yml** 플레이북을 생성합니다.

```

- name: Configure an Ethernet connection with 802.1X authentication
 hosts: node.example.com
 become: true
 tasks:
 - name: Copy client key for 802.1X authentication
 copy:
 src: "/srv/data/client.key"
 dest: "/etc/pki/tls/private/client.key"
 mode: 0600

 - name: Copy client certificate for 802.1X authentication
 copy:
 src: "/srv/data/client.crt"
 dest: "/etc/pki/tls/certs/client.crt"

 - name: Copy CA certificate for 802.1X authentication
 copy:
 src: "/srv/data/ca.crt"
 dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

 - include_role:
 name: rhel-system-roles.network
 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 192.0.2.1/24
 - 2001:db8:1::1/64
 gateway4: 192.0.2.254
 gateway6: 2001:db8:1::fffe
 dns:
```

```

- 192.0.2.200
- 2001:db8:1::ffbb
dns_search:
- example.com
ieee802_1x:
identity: user_name
eap: tls
private_key: "/etc/pki/tls/private/client.key"
private_key_password: "password"
client_cert: "/etc/pki/tls/certs/client.crt"
ca_cert: "/etc/pki/ca-trust/source/anchors/ca.crt"
domain_suffix_match: example.com
state: up

```

3.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/enable-802.1x.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

#### 추가 리소스

•

**/usr/share/ansible/roles/rhel-system-roles.network/README.md** file

•

**/usr/share/ansible/roles/rhel-system-roles.network/README.md** file

•

**ansible-playbook(1)** 도움말 페이지

## 22장. 기본 게이트웨이 설정 관리

기본 게이트웨이는 다른 경로가 패킷의 대상과 일치하지 않을 때 네트워크 패킷을 전달하는 라우터입니다. 로컬 네트워크에서 기본 게이트웨이는 일반적으로 인터넷에 가까운 하나의 홉에 있는 호스트입니다.

### 22.1. NMCLI를 사용하여 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 에서 설명한 대로 연결을 생성할 때 기본 게이트웨이를 설정합니다(예: **nmcli** 를 사용하여 정적 인터넷 연결 구성).

이 섹션에서는 **nmcli** 유틸리티를 사용하여 이전에 생성된 연결에서 기본 게이트웨이를 설정하거나 업데이트하는 방법을 설명합니다.

#### 사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.
- 사용자가 물리적 콘솔에 로그인된 경우 사용자 권한만으로 충분합니다. 그렇지 않으면 사용자에게 **root** 권한이 있어야 합니다.

#### 절차

1. 기본 게이트웨이의 IP 주소를 설정합니다.

예를 들어 예제 연결에 대한 기본 게이트웨이의 IPv4 주소를 **192.0.2.1** 로 설정하려면 다음을 수행합니다.

```
$ sudo nmcli connection modify example ipv4.gateway "192.0.2.1"
```

예를 들어 **2001:db8:1::1** 에 대한 예제 연결에서 기본 게이트웨이의 IPv6 주소를 설정하려면 다음을 수행합니다.

```
$ sudo nmcli connection modify example ipv6.gateway "2001:db8:1::1"
```

2. 변경 사항을 적용하려면 네트워크 연결을 다시 시작합니다. 예를 들어 명령줄을 사용하여 예제 연결을 다시 시작하려면 다음을 수행합니다.

■

## \$ sudo nmcli connection up example



주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

3.

선택적으로 경로가 활성화 상태인지 확인합니다.

IPv4 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -4 route
```

```
default via 192.0.2.1 dev example proto static metric 100
```

IPv6 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -6 route
```

```
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

추가 리소스



[nmcli를 사용하여 정적 이더넷 연결 구성](#)

## 22.2. NMCLI 대화형 모드를 사용하여 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 에서 설명한 대로 연결을 생성할 때 기본 게이트웨이를 설정합니다(예: [nmcli 대화형 편집기를 사용하여 동적 이더넷 연결 구성](#)).

이 섹션에서는 **nmcli** 유틸리티의 대화형 모드를 사용하여 이전에 생성된 연결에서 기본 게이트웨이를 설정하거나 업데이트하는 방법을 설명합니다.

사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 **IP** 주소를 구성해야 합니다.
- 사용자가 물리적 콘솔에 로그인된 경우 사용자 권한만으로 충분합니다. 그렇지 않으면 사용자에게 **root** 권한이 있어야 합니다.

## 절차

1. 필요한 연결에 대한 **nmcli** 대화형 모드를 엽니다. 예를 들어 **example** 연결에 대해 **nmcli** 대화형 모드를 여는 방법은 다음과 같습니다.

```
$ sudo nmcli connection edit example
```

2. 기본 게이트웨이를 설정합니다.

예를 들어 예제 연결에 대한 기본 게이트웨이의 **IPv4** 주소를 **192.0.2.1** 로 설정하려면 다음을 수행합니다.

```
nmcli> set ipv4.gateway 192.0.2.1
```

예를 들어 **2001:db8:1::1** 에 대한 예제 연결에서 기본 게이트웨이의 **IPv6** 주소를 설정하려면 다음을 수행합니다.

```
nmcli> set ipv6.gateway 2001:db8:1::1
```

3. 선택적으로 기본 게이트웨이가 올바르게 설정되었는지 확인합니다.

```
nmcli> print
...
ipv4.gateway: 192.0.2.1
...
ipv6.gateway: 2001:db8:1::1
...
```

4. 설정을 저장합니다.

```
nmcli> save persistent
```

5.

변경 사항을 적용하려면 네트워크 연결을 다시 시작하십시오.

```
nmcli> activate example
```



주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

6.

**nmcli** 대화형 모드를 그대로 둡니다.

```
nmcli> quit
```

7.

선택적으로 경로가 활성화 상태인지 확인합니다.

**IPv4** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

**IPv6** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

추가 리소스



[\*\*nmcli\*\* 대화형 편집기를 사용하여 정적 인터넷 연결 구성](#)

## 22.3. NM-CONNECTION-EDITOR를 사용하여 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 이 섹션에서는 **nm-connection-editor** 애플리케이션을 사용하여 이전에 생성된 연결에서 기본 게이트웨이를 설정하거나 업

데이트하는 방법을 설명합니다.

#### 사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.

#### 절차

1. 터미널을 열고 **nm-connection-editor** 를 입력합니다.

```
$ nm-connection-editor
```

2. 수정할 연결을 선택하고 기어바퀴 아이콘을 클릭하여 기존 연결을 편집합니다.
3. **IPv4 기본 게이트웨이를 설정합니다.** 예를 들어, 연결에 대한 기본 게이트웨이의 **IPv4** 주소를 **192.0.2.1** 로 설정하려면 다음을 수행합니다.
  - a. **IPv4 Settings** 탭을 엽니다.
  - b. 게이트웨이 의 주소가 내에 있는 **IP** 범위 옆에 있는 게이트웨이 필드에 주소를 입력합니다.

| Addresses   |         |           |
|-------------|---------|-----------|
| Address     | Netmask | Gateway   |
| 192.0.2.123 | 24      | 192.0.2.1 |

4. **IPv6 기본 게이트웨이를 설정합니다.** 예를 들어 **2001:db8:1::1** 에 대한 연결에 있는 기본 게이트웨이의 **IPv6** 주소를 설정하려면 다음을 수행합니다.
  - a. **IPv6** 탭을 엽니다.
  - b. 게이트웨이 의 주소가 내에 있는 **IP** 범위 옆에 있는 게이트웨이 필드에 주소를 입력합니다.



| Addresses     |        |               |
|---------------|--------|---------------|
| Address       | Prefix | Gateway       |
| 2001:db8:1::5 | 64     | 2001:db8:1::1 |

5. **OK(확인)**를 클릭합니다.

6. **저장**을 클릭합니다.

7. 변경 사항을 적용하려면 네트워크 연결을 다시 시작합니다. 예를 들어 명령줄을 사용하여 예제 연결을 다시 시작하려면 다음을 수행합니다.

```
$ sudo nmcli connection up example
```



주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

8. 선택적으로 경로가 활성화 상태인지 확인합니다.

**IPv4** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

**IPv6** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

## 추가 리소스

- [nm-connection-editor를 사용하여 이더넷 연결 구성](#)

**22.4. CONTROL-CENTER를 사용하여 기존 연결에서 기본 게이트웨이 설정**

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 이 섹션에서는 **control-center** 애플리케이션을 사용하여 이전에 생성된 연결에서 기본 게이트웨이를 설정하거나 업데이트하는 방법을 설명합니다.

## 사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.
- 연결의 네트워크 구성은 **control-center** 애플리케이션에서 열려 있습니다.

## 절차

1. **IPv4 기본 게이트웨이를 설정합니다.** 예를 들어, 연결에 대한 기본 게이트웨이의 **IPv4** 주소를 **192.0.2.1** 로 설정하려면 다음을 수행합니다.
  - a. **IPv4** 탭을 엽니다.
  - b. 게이트웨이 의 주소가 내에 있는 **IP** 범위 옆에 있는 게이트웨이 필드에 주소를 입력합니다.

| Addresses   |               |           |
|-------------|---------------|-----------|
| Address     | Netmask       | Gateway   |
| 192.0.2.123 | 255.255.255.0 | 192.0.2.1 |

2. **IPv6 기본 게이트웨이를 설정합니다.** 예를 들어 **2001:db8:1::1** 에 대한 연결에 있는 기본 게이트웨이의 **IPv6** 주소를 설정하려면 다음을 수행합니다.

a.

**IPv6 탭을 엽니다.**

b.

게이트웨이 의 주소가 내에 있는 **IP 범위** 옆에 있는 **게이트웨이** 필드에 주소를 입력합니다.

| Addresses     |        |               |
|---------------|--------|---------------|
| Address       | Prefix | Gateway       |
| 2001:db8:1::5 | 64     | 2001:db8:1::1 |

3.

**Apply(적용)**를 클릭합니다.

4.

네트워크 창에서 다시 돌아가서 연결 버튼을 **off** 로 전환하고 다시 켜서 변경 사항을 적용합니다.

**주의**

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

5.

선택적으로 경로가 활성화 상태인지 확인합니다.

**IPv4** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

**IPv6** 기본 게이트웨이를 표시하려면 다음을 수행합니다.

```
$ ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

-

## 추가 리소스

- [control-center](#)를 사용하여 이더넷 연결 구성

## 22.5. NMSTATECTL을 사용하여 기존 연결에서 기본 게이트웨이 설정

**nmstatectl** 유틸리티를 사용하여 네트워크 연결의 기본 게이트웨이를 설정할 수 있습니다. 이 절차에서는 기존 **enp1s0** 연결의 기본 게이트웨이를 **192.0.2.1**로 설정하는 방법을 설명합니다.

## 사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.
- **enp1s0** 인터페이스가 구성되어 있으며 기본 게이트웨이의 IP 주소는 이 인터페이스의 IP 구성 서브넷 내에 있습니다.
- **nmstate** 패키지가 설치되어 있습니다.

## 절차

1. 다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/set-default-gateway.yml**)을 만듭니다.

```

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.1
 next-hop-interface: enp1s0
```

2. 시스템에 설정을 적용합니다.

```
nmstatectl apply ~/set-default-gateway.yml
```

## 추가 리소스

- **nmstatectl**에 대한 자세한 내용은 **nmstatectl(8)** 매뉴얼 페이지를 참조하십시오.

•

자세한 구성 예제는 `/usr/share/doc/nmstate/examples/` 디렉토리를 참조하십시오.

## 22.6. 시스템 역할을 사용하여 기존 연결에서 기본 게이트웨이 설정

네트워크 **RHEL** 시스템 역할을 사용하여 기본 게이트웨이를 설정할 수 있습니다.



### 중요

네트워크 **RHEL** 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 값과 일치하지 않는 경우 시스템 역할이 동일한 이름으로 기존 연결 프로필을 덮어씁니다. 따라서 예를 들어 IP 구성이 이미 있는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

•

정적 IPv4 주소 - 198.51.100.20 에 /24 서브넷 마스크

•

정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크

•

IPv4 기본 게이트웨이 - 198.51.100.254

•

IPv6 기본 게이트웨이 - 2001:db8:1::fffe

•

IPv4 DNS 서버 198.51.100.200

•

IPv6 DNS 서버 - 2001:db8:1::ffbb

•

DNS 검색 도메인 - example.com

사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

## 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-connection.yml** 플레이북을 만듭니다.

```

- name: Configure an Ethernet connection with static IP and default gateway
 hosts: node.example.com
 become: true
 tasks:
 - include_role:
 name: rhel-system-roles.network

 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 198.51.100.20/24
 - 2001:db8:1::1/64
 gateway4: 198.51.100.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 198.51.100.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
 state: up
```

3. 플레이북을 실행합니다.
- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

■

```
ansible-playbook -u root ~/ethernet-connection.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/ethernet-connection.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

#### 추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)
- [ansible-playbook\(1\) 도움말 페이지](#)

### 22.7. 레거시 네트워크 스크립트를 사용할 때 기존 연결에서 기본 게이트웨이 설정

다음 절차에서는 레거시 네트워크 스크립트를 사용할 때 기본 게이트웨이를 구성하는 방법을 설명합니다. 이 예제에서는 **enp1s0** 인터페이스를 통해 연결할 수 있는 기본 게이트웨이를 **192.0.2.1**로 설정합니다.

#### 사전 요구 사항

- **NetworkManager** 패키지가 설치되어 있지 않거나 **NetworkManager** 서비스가 비활성화됩니다.
- **network-scripts** 패키지가 설치되어 있습니다.

#### 절차

1. **/etc/sysconfig/network-scripts/ifcfg-enp1s0** 파일에서 **GATEWAY** 매개 변수를 **192.0.2.1**로 설정합니다.

```
GATEWAY=192.0.2.1
```

2.

`/etc/sysconfig/network-scripts/route-enp0s1` 파일에 **default** 항목을 추가합니다.

```
default via 192.0.2.1
```

3.

네트워크를 재시작합니다.

```
systemctl restart network
```

## 22.8. NETWORKMANAGER가 여러 기본 게이트웨이를 관리하는 방법

예를 들어 대체 이유와 같은 경우 호스트에 여러 기본 게이트웨이를 설정합니다. 그러나 비동기 라우팅 문제를 방지하려면 동일한 프로토콜의 각 기본 게이트웨이에 별도의 메트릭 값이 필요합니다. **RHEL**은 지표가 가장 낮은 기본 게이트웨이에 대한 연결만 사용합니다.

다음 명령을 사용하여 연결의 **IPv4** 및 **IPv6** 게이트웨이 모두에 대한 지표를 설정할 수 있습니다.

```
nmcli connection modify connection-name ipv4.route-metric value ipv6.route-metric value
```

### 중요

라우팅 문제를 방지하기 위해 여러 연결 프로필에서 동일한 프로토콜에 대해 동일한 메트릭 값을 설정하지 마십시오.

지표 값 없이 기본 게이트웨이를 설정하면 **NetworkManager**에서 인터페이스 유형에 따라 지표 값을 자동으로 설정합니다. 따라서 **NetworkManager**는 이 네트워크 유형의 기본값을 활성화한 첫 번째 연결에 할당하고 활성화되는 순서대로 동일한 유형의 서로 연결에 증분된 값을 설정합니다. 예를 들어 기본 게이트웨이가 있는 이더넷 연결이 두 개 있는 경우 **NetworkManager**는 경로에 있는 **100** 개의 지표를 먼저 활성화하는 연결의 기본 게이트웨이로 설정합니다. 두 번째 연결의 경우 **NetworkManager**는 **101** 을 설정합니다.

다음은 자주 사용되는 네트워크 유형과 해당 기본 메트릭에 대한 개요입니다.

| 연결 유형 | 기본 메트릭 값 |
|-------|----------|
| VPN   | 50       |



| 연결 유형      | 기본 메트릭 값 |
|------------|----------|
| 이더넷        | 100      |
| MACsec     | 125      |
| InfiniBand | 150      |
| 본딩         | 300      |
| 팀          | 350      |
| VLAN       | 400      |
| 브릿지        | 425      |
| TUN        | 450      |
| Wi-Fi      | 600      |
| IP 터널      | 675      |

#### 추가 리소스

- [대체 경로를 정의하도록 정책 기반 라우팅 구성](#)
- [다중 경로 TCP 시작하기](#)

### 22.9. 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NETWORKMANAGER 구성

**NetworkManager**가 특정 프로필을 사용하지 않고 기본 게이트웨이를 제공하도록 구성할 수 있습니다. 기본 게이트웨이에 연결되지 않은 연결 프로필은 다음 절차를 따르십시오.

#### 사전 요구 사항

- 기본 게이트웨이에 연결되지 않은 연결에 대한 **NetworkManager** 연결 프로필이 있습니다.

#### 절차

1. 연결이 동적 IP 구성을 사용하는 경우 **NetworkManager**가 IPv4 및 IPv6 연결의 기본 경로로 연결을 사용하지 않도록 구성합니다.

```
nmcli connection modify connection_name ipv4.never-default yes ipv6.never-default yes
```

**ipv4.never-default** 및 **ipv6.never-default** 를 **yes** 로 설정하면 연결 프로파일에서 해당 프로토콜에 대한 기본 게이트웨이의 IP 주소가 자동으로 제거됩니다.

2.

연결을 활성화합니다.

```
nmcli connection up connection_name
```

검증 단계

- 

**ip -4 route** 및 **ip -6 route** 명령을 사용하여 RHEL이 IPv4 및 IPv6 프로토콜의 기본 경로에 네트워크 인터페이스를 사용하지 않는지 확인합니다.

## 22.10. 여러 기본 게이트웨이로 인해 예기치 않은 라우팅 동작 수정

다중 경로 TCP를 사용하는 경우 호스트에 여러 기본 게이트웨이가 필요한 경우와 같은 몇 가지 시나리오만 있습니다. 대부분의 경우 예기치 않은 라우팅 동작 또는 비동기 라우팅 문제를 방지하기 위해 단일 기본 게이트웨이만 구성합니다.



참고

트래픽을 다른 인터넷 공급자로 라우팅하려면 여러 기본 게이트웨이 대신 정책 기반 라우팅을 사용합니다.

사전 요구 사항

- 

호스트는 **NetworkManager**를 사용하여 네트워크 연결(기본값)을 관리합니다.

- 

호스트에는 여러 네트워크 인터페이스가 있습니다.

- 

호스트에는 여러 기본 게이트웨이가 구성되어 있습니다.

절차

1.

라우팅 테이블을 표시합니다.

•

IPv4의 경우 다음을 입력합니다.

```
ip -4 route
default via 192.0.2.1 dev enp1s0 proto static metric 101
default via 198.51.100.1 dev enp7s0 proto static metric 102
...
```

•

IPv6의 경우 다음을 입력합니다.

```
ip -6 route
default via 2001:db8:1::1 dev enp1s0 proto static metric 101 pref medium
default via 2001:db8:2::1 dev enp7s0 proto static metric 102 pref medium
...
```

기본로 시작하는 항목은 기본 경로를 나타냅니다. 이러한 항목의 인터페이스 이름은 **dev** 옆에 표시됩니다.

2.

다음 명령을 사용하여 이전 단계에서 식별한 인터페이스를 사용하는 **NetworkManager** 연결을 표시합니다.

```
nmcli -f GENERAL.CONNECTION,IP4.GATEWAY,IP6.GATEWAY device show enp1s0
GENERAL.CONNECTION: Corporate-LAN
IP4.GATEWAY: 192.168.122.1
IP6.GATEWAY: 2001:db8:1::1

nmcli -f GENERAL.CONNECTION,IP4.GATEWAY,IP6.GATEWAY device show enp7s0
GENERAL.CONNECTION: Internet-Provider
IP4.GATEWAY: 198.51.100.1
IP6.GATEWAY: 2001:db8:2::1
```

이 예에서 **Corporate-LAN** 및 **Internet-Provider** 라는 프로파일에는 기본 게이트웨이가 설정되어 있습니다. 로컬 네트워크에서 기본 게이트웨이는 일반적으로 인터넷에 더 가까운 호스트이므로 이 절차의 나머지 부분에서는 **Corporate-LAN** 의 기본 게이트웨이가 올바르지 않다고 가정합니다.

3.

**NetworkManager**에서 **Corporate-LAN** 연결을 IPv4 및 IPv6 연결의 기본 경로로 사용하지 않도록 구성합니다.

```
nmcli connection modify Corporate-LAN ipv4.never-default yes ipv6.never-default yes
```

**ipv4.never-default** 및 **ipv6.never-default** 를 **yes** 로 설정하면 연결 프로파일에서 해당 프로토콜에 대한 기본 게이트웨이의 IP 주소가 자동으로 제거됩니다.

4.

**Corporate-LAN** 연결을 활성화합니다.

```
nmcli connection up Corporate-LAN
```

#### 검증 단계

- 

**IPv4** 및 **IPv6** 라우팅 테이블을 표시하고 각 프로토콜에 하나의 기본 게이트웨이만 사용할 수 있는지 확인합니다.

- 

**IPv4**의 경우 다음을 입력합니다.

```
ip -4 route
default via 192.0.2.1 dev enp1s0 proto static metric 101
...
```

- 

**IPv6**의 경우 다음을 입력합니다.

```
ip -6 route
default via 2001:db8:1::1 dev enp1s0 proto static metric 101 pref medium
...
```

#### 추가 리소스

- 

[대체 경로를 정의하도록 정책 기반 라우팅 구성](#)

- 

[다중 경로 TCP 시작하기](#)

## 23장. 정적 경로 구성

라우팅을 사용하면 상호 연결된 네트워크 간에 트래픽을 전송하고 수신할 수 있습니다. 대규모 환경에서는 일반적으로 관리자는 라우터가 다른 라우터에 대해 동적으로 배울 수 있도록 서비스를 구성합니다. 소규모 환경에서는 트래픽이 한 네트워크에서 다음 네트워크로 도달할 수 있도록 정적 경로를 구성하는 경우가 많습니다.

이러한 모든 조건이 적용되는 경우 여러 네트워크 간에 작동하는 통신을 수행하기 위해 정적 경로가 필요합니다.

- 트래픽은 여러 네트워크를 전달해야 합니다.
- 기본 게이트웨이를 통한 전용 트래픽 흐름으로는 충분하지 않습니다.

**23.1절. “정적 경로가 필요한 네트워크의 예”** 시나리오와 정적 경로를 구성하지 않을 때 다양한 네트워크 간 트래픽 흐름을 설명합니다.

### 23.1. 정적 경로가 필요한 네트워크의 예

모든 IP 네트워크가 하나의 라우터를 통해 직접 연결되어 있지 않기 때문에 이 예제에는 고정 경로가 필요합니다. 정적 경로가 없으면 일부 네트워크는 서로 통신할 수 없습니다. 또한 일부 네트워크의 트래픽은 한 방향으로만 이동합니다.



참고

이 예제의 네트워크 토폴로지는 인위적이며 정적 라우팅의 개념을 설명하는 데만 사용됩니다. 프로덕션 환경에서는 권장되는 토폴로지가 아닙니다.

이 예제의 모든 네트워크 간 통신의 경우 홉 라우터 2 (203.0.113.10)를 사용하여 Raleigh (198.51.100.0/24)로 고정 경로를 구성합니다. 다음 홉의 IP 주소는 데이터 센터 네트워크(203.0.113.0/24)의 라우터 2 중 하나입니다.

다음과 같이 정적 경로를 구성할 수 있습니다.

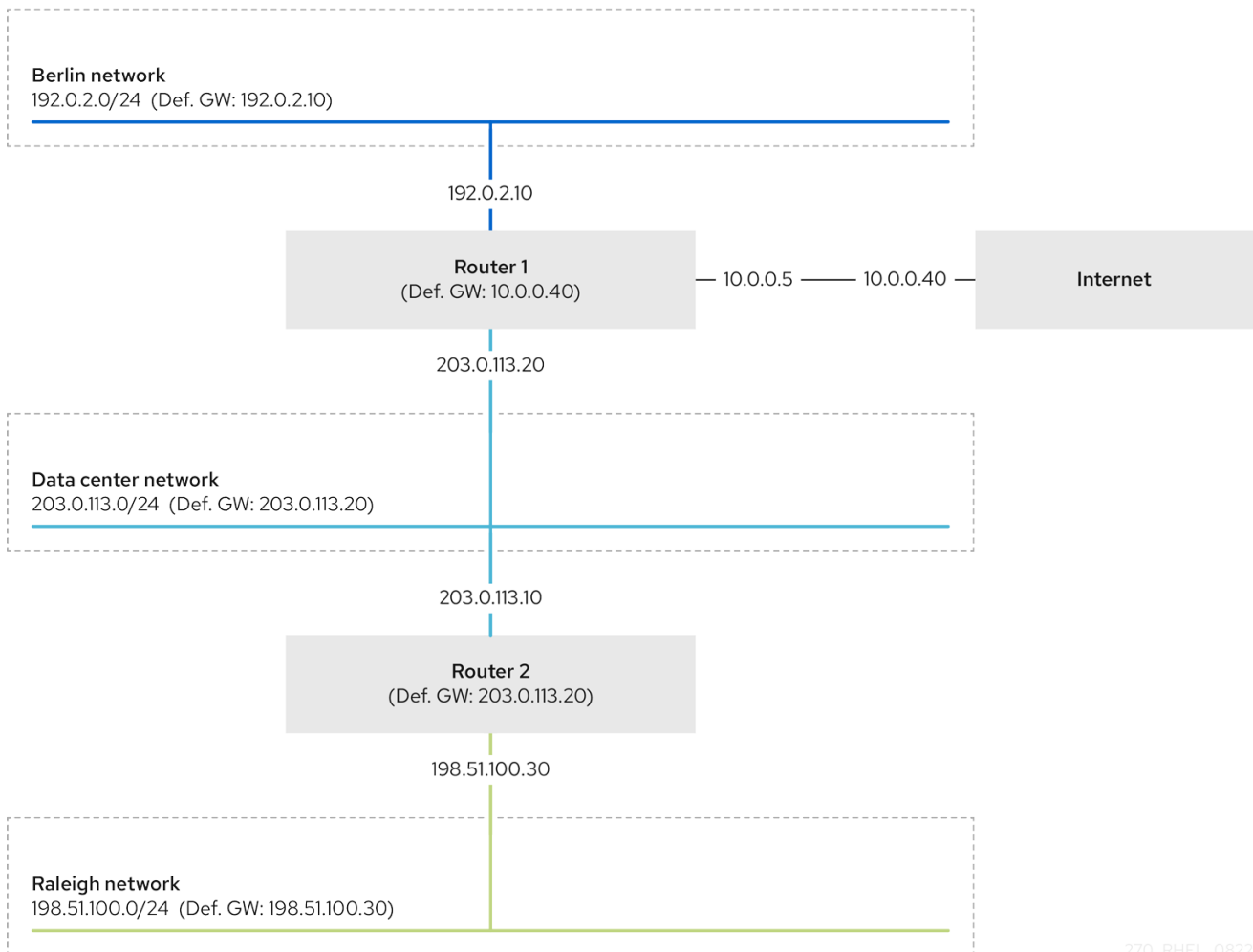
- 단순화된 구성의 경우 이 정적 경로를 라우터 1에서만 설정합니다. 그러나 데이터 센터

(203.0.113.0/24)에서 호스트를 통해 트래픽을 라우터 1을 통해 항상 라우터 1을 통해 Raleigh(198.51.100.0/24)에 전송하므로 라우터 1의 트래픽이 증가합니다.

- 

보다 복잡한 구성의 경우 데이터 센터의 모든 호스트에서 이 정적 경로를 구성합니다 (203.0.113.0/24). 그러면 이 서브넷의 모든 호스트는 Raleigh(198.51.100.0/24)에 더 가까운 라우터 2(203.0.113.10)로 직접 트래픽을 보냅니다.

트래픽 흐름 또는 그렇지 않은 네트워크 트래픽의 자세한 내용은 다이어그램 아래에 있는 설명을 참조하십시오.



270\_RHEL\_0822

필요한 정적 경로가 구성되지 않은 경우, 다음에서는 통신이 작동하고 작동하지 않는 상황에 대해 설명합니다.

- 

**NetNamespace 네트워크의 호스트 (192.0.2.0/24):**

- 직접 연결되어 있으므로 동일한 서브넷의 다른 호스트와 통신할 수 있습니다.
- 라우터 1은 **controlPlane** 네트워크(192.0.2.0/24)에 있고 인터넷으로 연결되는 기본 게이트웨이가 있기 때문에 인터넷과 통신할 수 있습니다.
- **Router 1**은 **jenkinsfile**(192.0.2.0/24) 및 데이터 센터(203.0.113.0/24) 네트워크 둘 다에 인터페이스가 있기 때문에 데이터 센터 네트워크(203.0.113.0/24)와 통신할 수 있습니다.
- 라우터 1에는 이 네트워크에 인터페이스가 없으므로 **Raleigh** 네트워크(198.51.100.0/24)와 통신할 수 없습니다. 따라서 라우터 1은 트래픽을 자체 기본 게이트웨이(인터넷)로 전송합니다.
- 데이터 센터 네트워크의 호스트 (203.0.113.0/24):
  - 직접 연결되어 있으므로 동일한 서브넷의 다른 호스트와 통신할 수 있습니다.
  - 기본 게이트웨이가 라우터 1로 설정되어 있고 라우터 1에는 네트워크, 데이터 센터(203.0.113.0/24) 및 인터넷에 인터페이스가 있으므로 인터넷과 통신할 수 있습니다.
  - 기본 게이트웨이가 라우터 1로 설정되어 있고 라우터 1에는 데이터 센터(203.0.113.0/24) 및 **ENV**(192.0.2.0/24) 네트워크 둘 다에 인터페이스가 있으므로 **Galera** 네트워크(192.0.2.0/24)와 통신할 수 있습니다.
  - 데이터 센터 네트워크에 인터페이스가 없기 때문에 **Raleigh** 네트워크(198.51.100.0/24)와 통신할 수 없습니다. 따라서 데이터 센터의 호스트(203.0.113.0/24)는 트래픽을 기본 게이트웨이(**Router 1**)로 보냅니다. 라우터 1에는 **Raleigh** 네트워크(198.51.100.0/24)에 인터페이스가 없으며 결과적으로 라우터 1은 이 트래픽을 자체 기본 게이트웨이(인터넷)로 전송합니다.
- **Raleigh** 네트워크 (198.51.100.0/24)의 호스트:
  - 직접 연결되어 있으므로 동일한 서브넷의 다른 호스트와 통신할 수 있습니다.
  - 인터넷의 호스트와 통신할 수 없습니다. 라우터 2는 기본 게이트웨이 설정으로 인해 트

래픽을 라우터 1로 전송합니다. 라우터 1의 실제 동작은 역방향 경로 필터(**rp\_filter**) 시스템 제어(**sysctl**) 설정에 따라 달라집니다. 기본적으로 **RHEL**에서 라우터 1은 나가는 트래픽을 인터넷으로 라우팅하는 대신 삭제합니다. 그러나 구성된 동작에 관계없이 정적 경로 없이 통신을 수행할 수 없습니다.

○

데이터 센터 네트워크(**203.0.113.0/24**)와 통신할 수 없습니다. 기본 게이트웨이 설정으로 인해 나가는 트래픽이 라우터 2를 통해 대상에 도달합니다. 그러나 데이터 센터 네트워크(**203.0.113.0/24**)의 호스트가 기본 게이트웨이(**Router 1**)로 응답을 전송하므로 패킷에 대한 응답이 발신자에 도달하지 않습니다. 그런 다음 라우터 1은 트래픽을 인터넷으로 전송합니다.

○

**controlPlane** 네트워크(**192.0.2.0/24**)와 통신할 수 없습니다. 라우터 2는 기본 게이트웨이 설정으로 인해 트래픽을 라우터 1로 전송합니다. 라우터 1의 실제 동작은 **rp\_filter sysctl** 설정에 따라 다릅니다. 기본적으로 **RHEL**에서 **Router 1**은 **outgoing** 트래픽을 **[2]192.0.2.0/24** 네트워크(**192.0.2.0/24**)로 전송하는 대신 나가는 트래픽을 삭제합니다. 그러나 구성된 동작에 관계없이 정적 경로 없이 통신을 수행할 수 없습니다.



#### 참고

정적 경로를 구성하는 것 외에도 두 라우터에서 **IP** 전달을 활성화해야 합니다.

#### 추가 리소스

●

[net.ipv4.conf.all.rp\\_filter](#)를 서버에 설정한 경우 서버를 **ping**할 수 없는 이유는 무엇입니까?

●

[IP 전달 활성화](#)

## 23.2. NMCLI 명령을 사용하여 정적 경로를 구성하는 방법

정적 경로를 구성하려면 다음 구문과 함께 **nmcli** 유틸리티를 사용합니다.

```
$ nmcli connection modify connection_name ipv4.routes "ip[/prefix] [next_hop] [metric] [attribute=value] [attribute=value] ..."
```

명령은 다음 경로 특성을 지원합니다.

●

**cwnd=n**: 패킷 수에 정의된 혼잡 창(**CWND**) 크기를 설정합니다.



- **lock-cwnd=true/false:** 커널이 **CWND** 값을 업데이트할 수 있는지 여부를 정의합니다.
- **lock-mtu=true/false:** 커널이 **MTU** 검색을 라우팅하도록 **MTU**를 업데이트할 수 있는지 여부를 정의합니다.
- **lock-window=true/false:** 커널이 **TCP** 패킷의 최대 창 크기를 업데이트할 수 있는지 여부를 정의합니다.
- **mtu=n:** 대상의 경로와 함께 사용할 **MTU**(최대 전송 단위)를 설정합니다.
- **onlink=true/false:** 인터페이스 접두사와 일치하지 않는 경우에도 다음 홉이 이 링크에 직접 연결되는지 여부를 정의합니다.
- **scope=n:** **IPv4** 경로의 경우 이 속성은 경로 접두사에서 다루는 대상의 범위를 설정합니다. 값을 정수(0-255)로 설정합니다.
- **src=address:** 경로 접두사에서 다루는 대상으로 트래픽을 보낼 때 선호하는 소스 주소를 설정합니다.
- **table=table\_id:** 경로를 추가해야 하는 테이블의 **ID**를 설정합니다. 이 매개변수를 생략하면 **NetworkManager**는 기본 테이블을 사용합니다.
- **tos=n:** 서비스 유형(**TOS**) 키를 설정합니다. 값을 정수(0-255)로 설정합니다.
- **type=value:** 경로 유형을 설정합니다. **NetworkManager**는 유니캐스트, 로컬, 블랙홀, 도달할 수 없는, 금지, 경로 유형을 **throw** 하는 것을 지원합니다. 기본값은 유니캐스트입니다.
- **window=n:** **TCP**가 이러한 대상에 알리는 최대 창 크기를 바이트 단위로 측정합니다.

**ipv4.routes** 하위 명령을 사용하는 경우 **nmcli** 는 이 매개변수의 모든 현재 설정을 재정의합니다.

경로를 추가하려면 다음을 수행합니다.

```
$ nmcli connection modify connection_name +ipv4.routes "..."
```

마찬가지로 특정 경로를 제거하려면 다음을 수행합니다.

```
$ nmcli connection modify connection_name -ipv4.routes "..."
```

### 23.3. NMCLI 명령을 사용하여 정적 경로 구성

**nmcli connection modify** 명령을 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

이 섹션의 절차에서는 **198.51.100.1** 에서 실행되는 게이트웨이를 사용하는 **192.0.2.0/24** 네트워크에 경로를 추가하는 방법을 설명합니다.

#### 사전 요구 사항

- 네트워크가 구성되었습니다
- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- 사용자가 물리적 콘솔에 로그인된 경우 사용자 권한만으로 충분합니다. 그렇지 않으면 명령에 **root** 권한이 필요합니다.

#### 절차

1. 예제 연결에 정적 경로를 추가합니다.

```
$ sudo nmcli connection modify example +ipv4.routes "192.0.2.0/24 198.51.100.1"
```

한 단계에서 여러 경로를 설정하려면 개별 경로를 쉼표로 구분하여 명령으로 전달합니다. 예를 들어 **192.0.2.0/24** 및 **203.0.113.0/24** 네트워크에 경로를 추가하려면 모두 **198.51.100.1** 게이트웨이를 통해 라우팅됩니다.

```
$ sudo nmcli connection modify example +ipv4.routes "192.0.2.0/24 198.51.100.1, 203.0.113.0/24 198.51.100.1"
```

2.

선택적으로 경로가 구성에 올바르게 추가되었는지 확인합니다.

```
$ nmcli connection show example
```

```
...
```

```
ipv4.routes: { ip = 192.0.2.1/24, nh = 198.51.100.1 }
```

```
...
```

3.

네트워크 연결을 다시 시작합니다.

```
$ sudo nmcli connection up example
```



주의

연결을 다시 시작하면 해당 인터페이스에서 연결이 잠시 중단됩니다.

4.

선택적으로 경로가 활성화 상태인지 확인합니다.

```
$ ip route
```

```
...
```

```
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

추가 리소스



**nmcli(1) 도움말 페이지**

## 23.4. CONTROL-CENTER를 사용하여 정적 경로 구성

**GNOME**에서 **control-center** 를 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

이 섹션의 절차에서는 **198.51.100.1** 에서 실행되는 게이트웨이를 사용하는 **192.0.2.0/24** 네트워크에 경로를 추가하는 방법을 설명합니다.

사전 요구 사항

- 네트워크가 구성되어 있습니다.
- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- 연결의 네트워크 구성은 **control-center** 애플리케이션에서 열립니다. [nm-connection-editor](#)를 사용하여 이더넷 연결 구성을 참조하십시오.

## 절차

1. **IPv4** 탭을 엽니다.
2. 선택적으로 **IPv4** 탭의 **Routes** 섹션에서 **On** 버튼을 클릭하여 정적 경로만 사용하도록 자동 경로를 비활성화합니다. 자동 경로가 활성화되면 **Red Hat Enterprise Linux**는 **DHCP** 서버에서 수신한 정적 경로와 경로를 사용합니다.
3. 주소, 넷마스크, 게이트웨이 및 선택적으로 메트릭 값을 입력합니다.

| Routes    |         |              | Automatic <input type="checkbox"/> OFF |                                  |
|-----------|---------|--------------|----------------------------------------|----------------------------------|
| Address   | Netmask | Gateway      | Metric                                 |                                  |
| 192.0.2.0 | 24      | 198.51.100.1 |                                        | <input type="button" value="x"/> |

4. **Apply(적용)**를 클릭합니다.
5. 네트워크 창에서 다시 돌아가서 연결 버튼을 **off**로 전환하고 다시 커서 변경 사항을 적용합니다.



주의

연결을 다시 시작하면 해당 인터페이스에서 연결이 잠시 중단됩니다.

6.

선택적으로 경로가 활성화 상태인지 확인합니다.

```
$ ip route
```

```
...
```

```
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

### 23.5. NM-CONNECTION-EDITOR를 사용하여 정적 경로 구성

**nm-connection-editor** 애플리케이션을 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

이 섹션의 절차에서는 예제 연결에 연결할 수 있는 **198.51.100.1** 에서 실행되는 게이트웨이를 사용하는 **192.0.2.0/24** 네트워크에 경로를 추가하는 방법을 설명합니다.

#### 사전 요구 사항

- 네트워크가 구성되어 있습니다.
- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.

#### 절차

1.

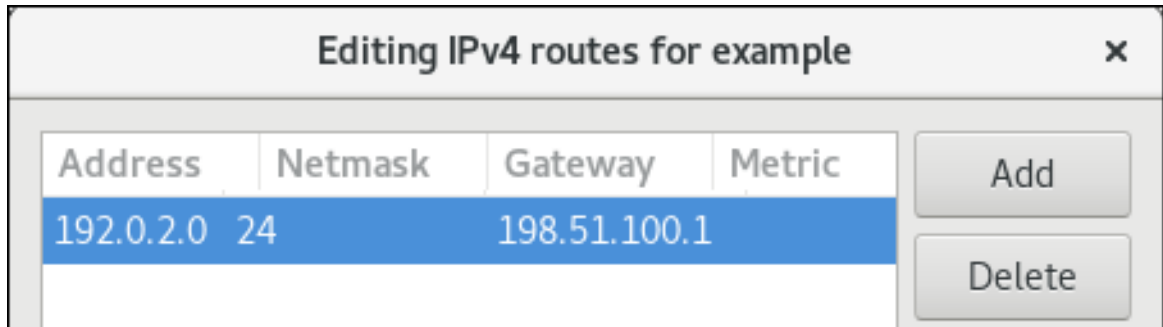
터미널을 열고 **nm-connection-editor** 를 입력합니다.

```
$ nm-connection-editor
```

2.

예제 연결을 선택하고 톱니바퀴 아이콘을 클릭하여 기존 연결을 편집합니다.

3. **IPv4** 탭을 엽니다.
4. **Routes(경로)** 버튼을 클릭합니다.
5. **Add(추가)** 버튼을 클릭하고 주소, 넷마스크, 게이트웨이 및 지포 값을 선택적으로 입력합니다.



6. **OK(확인)**를 클릭합니다.
7. **저장**을 클릭합니다.
8. 변경 사항을 적용하려면 네트워크 연결을 다시 시작합니다. 예를 들어 명령줄을 사용하여 예제 연결을 다시 시작하려면 다음을 수행합니다.

```
$ sudo nmcli connection up example
```

9. 선택적으로 경로가 활성 상태인지 확인합니다.

```
$ ip route
```

```
...
```

```
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

### 23.6. NMCLI 대화형 모드를 사용하여 정적 경로 구성

**nmcli** 유틸리티의 대화형 모드를 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

이 섹션의 절차에서는 예제 연결에 연결할 수 있는 **198.51.100.1** 에서 실행되는 게이트웨이를 사용하는 **192.0.2.0/24** 네트워크에 경로를 추가하는 방법을 설명합니다.

#### 사전 요구 사항

- 네트워크가 구성되었습니다
- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- 사용자가 물리적 콘솔에 로그인된 경우 사용자 권한만으로 충분합니다. 그렇지 않으면 명령에 **root** 권한이 필요합니다.

#### 절차

1. 예제 연결에 대한 **nmcli** 대화형 모드를 엽니다.

```
$ sudo nmcli connection edit example
```

2. 정적 경로를 추가합니다.

```
nmcli> set ipv4.routes 192.0.2.0/24 198.51.100.1
```

3. 선택적으로 경로가 구성에 올바르게 추가되었는지 확인합니다.

```
nmcli> print
...
ipv4.routes: { ip = 192.0.2.1/24, nh = 198.51.100.1 }
...
```

**ip** 속성은 라우팅할 네트워크와 게이트웨이 **nh** 속성이 표시됩니다(다음 참조).

4. 설정을 저장합니다.

```
nmcli> save persistent
```

5.

네트워크 연결을 다시 시작합니다.

```
nmcli> activate example
```



주의

연결을 다시 시작하면 현재 이 연결을 사용하는 모든 연결이 일시적으로 중단됩니다.

6.

**nmcli** 대화형 모드를 그대로 둡니다.

```
nmcli> quit
```

7.

선택적으로 경로가 활성화 상태인지 확인합니다.

```
$ ip route
```

```
...
```

```
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

### 23.7. NMSTATECTL을 사용하여 정적 경로 구성

**nmstatectl** 유틸리티를 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

이 섹션의 절차에서는 **198.51.100.1** 에서 실행되는 게이트웨이를 사용하는 **192.0.2.0/24** 네트워크에 경로를 추가하는 방법을 설명합니다. 이 프로세스는 **enp1s0** 인터페이스를 통해 연결할 수 있습니다.

사전 요구 사항

•

**enp1s0** 네트워크 인터페이스가 구성되어 있습니다.

•

정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.



- **nmstate** 패키지가 설치되어 있습니다.

## 절차

1. 다음 콘텐츠를 사용하여 **YAML** 파일(예: `~/add-static-route-to-enp1s0.yml`)을 생성합니다.

```

routes:
 config:
 - destination: 192.0.2.0/24
 next-hop-address: 198.51.100.1
 next-hop-interface: enp1s0
```

2. 시스템에 설정을 적용합니다.

```
nmstatectl apply ~/add-static-route-to-enp1s0.yml
```

## 추가 리소스

- **nmstatectl(8)** 도움말 페이지
- `/usr/share/doc/nmstate/examples/`

## 23.8. RHEL 시스템 역할을 사용하여 정적 경로 구성

네트워크 **RHEL** 시스템 역할을 사용하여 정적 경로를 구성할 수 있습니다.

### 중요

네트워크 **RHEL** 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 값과 일치하지 않는 경우 시스템 역할이 동일한 이름으로 기존 연결 프로필을 덮어씁니다. 따라서 예를 들어 IP 구성이 이미 있는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp7s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20** 에 /24 서브넷 마스크
- 정적 IPv6 주소 - **2001:db8:1::1** 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - **198.51.100.254**
- IPv6 기본 게이트웨이 - **2001:db8:1::fffe**
- IPv4 DNS 서버 **198.51.100.200**
- IPv6 DNS 서버 - **2001:db8:1::ffbb**
- DNS 검색 도메인 - **example.com**
- 정적 경로:
  - **192.0.2.0/24** 게이트웨이 **198.51.100.1**
  - **203.0.113.0/24** 게이트웨이 **198.51.100.2**

#### 사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

#### 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.

**node.example.com**

2.

다음 콘텐츠를 사용하여 `~/add-static-routes.yml` 플레이북을 생성합니다.

```

- name: Configure an Ethernet connection with static IP and additional routes
 hosts: node.example.com
 become: true
 tasks:
 - include_role:
 name: rhel-system-roles.network

 vars:
 network_connections:
 - name: enp7s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 198.51.100.20/24
 - 2001:db8:1::1/64
 gateway4: 198.51.100.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 198.51.100.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
 route:
 - network: 192.0.2.0
 prefix: 24
 gateway: 198.51.100.1
 - network: 203.0.113.0
 prefix: 24
 gateway: 198.51.100.2
 state: up
```

3.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/add-static-routes.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/add-static-routes.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

#### 검증 단계

- 라우팅 테이블을 표시합니다.

```
ip -4 route
default via 198.51.100.254 dev enp7s0 proto static metric 100
192.0.2.0/24 via 198.51.100.1 dev enp7s0 proto static metric 100
203.0.113.0/24 via 198.51.100.2 dev enp7s0 proto static metric 100
...
```

#### 추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file
- [ansible-playbook\(1\)](#) 도움말 페이지

### 23.9. 레거시 네트워크 스크립트를 사용할 때 KEY-VALUE-FORMAT에 정적 경로 구성 파일 생성

다음 절차에서는 **NetworkManager** 대신 레거시 네트워크 스크립트를 사용할 때 **192.0.2.0/24** 네트워크에 대한 **IPv4** 경로에 대한 라우팅 구성 파일을 수동으로 생성하는 방법을 설명합니다. 이 예에서 **IP** 주소가 **198.51.100.1** 인 해당 게이트웨이는 **enp1s0** 인터페이스를 통해 연결할 수 있습니다.

이 절차의 예제에서는 **key-value-format**의 구성 항목을 사용합니다.

#### 참고

레거시 네트워크 스크립트는 정적 **IPv4** 경로에 대해서만 **key-value-format**을 지원합니다. **IPv6** 경로의 경우 **ip-command-format**을 사용합니다. 레거시 네트워크 스크립트를 사용할 때 **ip-command-format**에서 정적 경로 구성 파일 생성을 참조하십시오.

#### 사전 요구 사항

- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- **NetworkManager** 패키지가 설치되어 있지 않거나 **NetworkManager** 서비스가 비활성화됩니다.
- **network-scripts** 패키지가 설치되어 있습니다.

## 절차

1. 정적 IPv4 경로를 `/etc/sysconfig/network-scripts/route-enp0s1` 파일에 추가합니다.

```
ADDRESS0=192.0.2.0
NETMASK0=255.255.255.0
GATEWAY0=198.51.100.1
```

- **ADDRESS0** 변수는 첫 번째 라우팅 항목의 네트워크를 정의합니다.
- **NETMASK0** 변수는 첫 번째 라우팅 항목의 넷마스크를 정의합니다.
- **GATEWAY0** 변수는 첫 번째 라우팅 항목에 대한 원격 네트워크 또는 호스트에 대한 게이트웨이의 IP 주소를 정의합니다.

정적 경로를 여러 개 추가하는 경우 변수 이름의 수를 늘립니다. 각 경로의 변수는 순차적으로 번호가 지정되어야 합니다. 예를 들어 **ADDRESS0,ADDRESS1,ADDRESS3** 등입니다.

2. 네트워크를 재시작합니다.

```
systemctl restart network
```

## 추가 리소스

- `/usr/share/doc/network-scripts/sysconfig.txt`

다음 절차에서는 레거시 네트워크 스크립트를 사용할 때 다음 정적 경로에 대한 라우팅 구성 파일을 수동으로 생성하는 방법을 설명합니다.

- **192.0.2.0/24 네트워크에 대한 IPv4 경로입니다. IP 주소가 198.51.100.1 인 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.**
- **2001:db8:1::/64 네트워크에 대한 IPv6 경로입니다. IP 주소가 2001:db8:2::1 인 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.**

이 절차의 예제에서는 **ip-command-format**의 설정 항목을 사용합니다.

#### 사전 요구 사항

- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- **NetworkManager** 패키지가 설치되어 있지 않거나 **NetworkManager** 서비스가 비활성화됩니다.
- **network-scripts** 패키지가 설치되어 있습니다.

#### 절차

1. 정적 IPv4 경로를 **/etc/sysconfig/network-scripts/route-enp0s1** 파일에 추가합니다.

```
192.0.2.0/24 via 198.51.100.1 dev enp0s1
```

2. 정적 IPv6 경로를 **/etc/sysconfig/network-scripts/route6-enp0s1** 파일에 추가합니다.

```
2001:db8:1::/64 via 2001:db8:2::1 dev enp0s1
```

3. 네트워크를 재시작합니다.

```
systemctl restart network
```

#### 추가 리소스

•

레거시 네트워크 스크립트 구성에 대한 자세한 내용은 `/usr/share/doc/network-scripts/sysconfig.txt` 파일을 참조하십시오.

## 24장. 대체 경로를 정의하도록 정책 기반 라우팅 구성

기본적으로 **RHEL**의 커널은 라우팅 테이블을 사용하여 대상 주소를 기반으로 네트워크 패킷을 전달할 위치를 결정합니다. 정책 기반 라우팅을 사용하면 복잡한 라우팅 시나리오를 구성할 수 있습니다. 예를 들어 소스 주소, 패킷 메타데이터 또는 프로토콜과 같은 다양한 기준에 따라 패킷을 라우팅할 수 있습니다.

이 섹션에서는 **NetworkManager**를 사용하여 정책 기반 라우팅을 구성하는 방법에 대해 설명합니다.



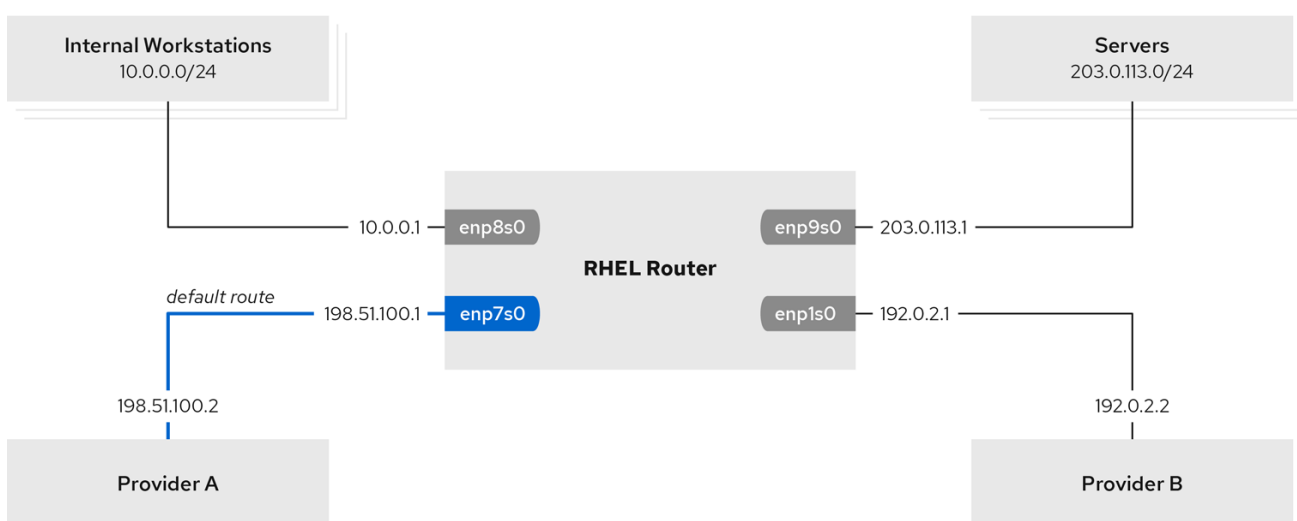
### 참고

**NetworkManager**를 사용하는 시스템에서는 **nmcli** 유틸리티만 라우팅 규칙 설정 및 특정 테이블에 경로를 할당할 수 있습니다.

### 24.1. NETWORKMANAGER를 사용하여 특정 서브넷의 트래픽을 다른 기본 게이트웨이로 라우팅

이 섹션에서는 기본적으로 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 **A**로 라우팅하는 라우터로 **RHEL**을 구성하는 방법에 대해 설명합니다. **RHEL**은 정책 기반 라우팅을 사용하여 내부 워크스테이션 서브넷에서 받은 트래픽을 프로바이더 **B**로 라우팅합니다.

이 절차에서는 다음 네트워크 토폴로지를 가정합니다.



60\_RHEL\_0120



- 시스템은 **NetworkManager** 를 사용하여 기본값인 네트워크를 구성합니다.
- 프로시저에서 설정할 **RHEL** 라우터에는 네 개의 네트워크 인터페이스가 있습니다.
  - **enp7s0** 인터페이스는 공급자 **A**의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 IP는 **198.51.100.2** 입니다. 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp1s0** 인터페이스는 공급자 **B**의 네트워크에 연결되어 있습니다. 공급자의 네트워크의 게이트웨이 IP는 **192.0.2.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp8s0** 인터페이스는 내부 워크스테이션이 있는 **10.0.0.0/24** 서브넷에 연결됩니다.
  - **enp9s0** 인터페이스는 회사 서버를 사용하여 **203.0.113.0/24** 서브넷에 연결됩니다.
- 내부 워크스테이션 서브넷의 호스트는 기본 게이트웨이로 **10.0.0.1** 을 사용합니다. 이 절차에서는 이 IP 주소를 라우터의 **enp8s0** 네트워크 인터페이스에 할당합니다.
- 서버 서브넷의 호스트는 기본 게이트웨이로 **203.0.113.1** 을 사용합니다. 이 절차에서는 라우터의 **enp9s0** 네트워크 인터페이스에 이 IP 주소를 할당합니다.
- **firewalld** 서비스가 활성화되고 활성화됩니다.

## 절차

1.

**A** 공급자에 네트워크 인터페이스를 구성합니다.

```
nmcli connection add type ethernet con-name Provider-A ifname enp7s0
 ipv4.method manual ipv4.addresses 198.51.100.1/30 ipv4.gateway 198.51.100.2
 ipv4.dns 198.51.100.200 connection.zone external
```

**nmcli connection add** 명령은 **NetworkManager** 연결 프로필을 생성합니다. 다음 목록은 명령의 옵션을 설명합니다.

- **type ethernet:** 연결 유형이 이더넷임을 정의합니다.
- **con-name connection\_name:** 프로필 이름을 설정합니다. 혼동을 피하려면 의미 있는 이름을 사용합니다.
- **ifname network\_device:** 네트워크 인터페이스를 설정합니다.
- **ipv4.method manual:** 정적 IP 주소를 구성할 수 있습니다.
- **ipv4.addresses IP\_address/subnet\_mask:** IPv4 주소 및 서브넷 마스크를 설정합니다.
- **ipv4.gateway IP\_address:** 기본 게이트웨이 주소를 설정합니다.
- **ipv4.dns IP\_of\_DNS\_server:** DNS 서버의 IPv4 주소를 설정합니다.
- **connection.zone firewalld\_zone:** 정의된 **firewalld** 영역에 네트워크 인터페이스를 할당합니다. **firewalld** 는 외부 영역에 할당된 인터페이스에 대해 자동으로 마스커레이딩을 활성화합니다.

2.

공급자 **B**에 네트워크 인터페이스를 구성합니다.

```
nmcli connection add type ethernet con-name Provider-B ifname enp1s0
ipv4.method manual ipv4.addresses 192.0.2.1/30 ipv4.routes "0.0.0.0/0 192.0.2.2
table=5000" connection.zone external
```

이 명령은 **ipv4.gateway** 대신 **ipv4.routes** 매개변수를 사용하여 기본 게이트웨이를 설정합니다. 이 연결은 기본값과 다른 라우팅 테이블(5000)에 이 연결에 대한 기본 게이트웨이를 할당해야 합니다. 연결이 활성화되면 **NetworkManager**에서 이 새 라우팅 테이블을 자동으로 생성합니다.

3.

내부 워크스테이션 서브넷에 네트워크 인터페이스를 구성합니다.

```
nmcli connection add type ethernet con-name Internal-Workstations ifname enp8s0
ipv4.method manual ipv4.addresses 10.0.0.1/24 ipv4.routes "10.0.0.0/24 table=5000"
ipv4.routing-rules "priority 5 from 10.0.0.0/24 table 5000" connection.zone trusted
```

■

이 명령은 **ipv4.routes** 매개변수를 사용하여 **ID 5000** 이 있는 라우팅 테이블에 정적 경로를 추가합니다. **10.0.0.0/24** 서브넷의 고정 경로는 로컬 네트워크 인터페이스의 IP를 **provider B(192.0.2.1)**로 다음 홉으로 사용합니다.

또한 이 명령은 **ipv4.routing-rules** 매개변수를 사용하여 **10.0.0.0/24** 서브넷에서 테이블 **5000** 으로 트래픽을 라우팅하는 우선 순위 **5** 가 있는 라우팅 규칙을 추가합니다. 낮은 값은 우선 순위가 높습니다.

**ipv4.routing-rules** 매개변수의 구문은 **ip rule add** 명령과 동일합니다. 단, **ipv4.routing-rules** 에는 항상 우선 순위를 지정해야 합니다.

4.

서버 서브넷에 네트워크 인터페이스를 구성합니다.

```
nmcli connection add type ethernet con-name Servers ifname enp9s0 ipv4.method
manual ipv4.addresses 203.0.113.1/24 connection.zone trusted
```

#### 검증 단계

1.

내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

a.

**traceroute** 패키지를 설치합니다.

```
yum install traceroute
```

b.

**traceroute** 유틸리티를 사용하여 인터넷의 호스트에 대한 경로를 표시합니다.

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.337 ms 0.260 ms 0.223 ms
 2 192.0.2.1 (192.0.2.1) 0.884 ms 1.066 ms 1.248 ms
 ...
```

명령 출력은 라우터가 공급자 **B**의 네트워크인 **192.0.2.1** 을 통해 패킷을 전송함을 표시합니다.

2.

서버 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

a.

**traceroute** 패키지를 설치합니다.

```
yum install traceroute
```

b.

**traceroute** 유틸리티를 사용하여 인터넷의 호스트에 대한 경로를 표시합니다.

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 203.0.113.1 (203.0.113.1) 2.179 ms 2.073 ms 1.944 ms
 2 198.51.100.2 (198.51.100.2) 1.868 ms 1.798 ms 1.549 ms
 ...
```

명령 출력은 라우터가 공급자 A의 네트워크인 198.51.100.2 를 통해 패킷을 전송함을 표시합니다.

## 문제 해결 단계

**RHEL** 라우터에서 다음을 수행합니다.

1.

규칙 목록을 표시합니다.

```
ip rule list
0: from all lookup local
5: from 10.0.0.0/24 lookup 5000
32766: from all lookup main
32767: from all lookup default
```

기본적으로 **RHEL**에는 로컬, 기본, 기본 테이블에 대한 규칙이 포함되어 있습니다.

2.

표 5000 에 경로를 표시합니다.

```
ip route list table 5000
0.0.0.0/0 via 192.0.2.2 dev enp1s0 proto static metric 100
10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
```

3.

인터페이스 및 방화벽 영역을 표시합니다.

```
firewall-cmd --get-active-zones
external
```

```

interfaces: enp1s0 enp7s0
trusted
interfaces: enp8s0 enp9s0

```

4.

외부 영역에 마스커레이딩이 활성화되어 있는지 확인합니다.

```

firewall-cmd --info-zone=external
external (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0 enp7s0
sources:
services: ssh
ports:
protocols:
masquerade: yes
...

```

추가 리소스

- **nm- settings(5) 매뉴얼 페이지의 IPv4 설정 섹션**
- **nm- settings(5) 도움말 페이지의 연결 설정 섹션**
- **nmcli(1) 도움말 페이지의 연결 관리 명령 섹션**
- **[RHEL에서 NetworkManager를 사용하여 정책 기반 라우팅을 설정할 수 있습니까?](#)**

## 24.2. 레거시 네트워크 스크립트를 사용할 때 정책 기반 라우팅과 관련된 구성 파일의 개요

**NetworkManager** 대신 레거시 네트워크 스크립트를 사용하여 네트워크를 구성하는 경우 정책 기반 라우팅을 구성할 수도 있습니다.

참고

**network-scripts** 패키지에서 제공하는 레거시 네트워크 스크립트를 사용하여 네트워크 구성은 **RHEL 8**에서 더 이상 사용되지 않습니다. **NetworkManager**를 사용하여 정책 기반 라우팅을 구성하는 것이 좋습니다. 예를 들어 **NetworkManager**를 사용하여 특정 서브넷에서 다른 기본 게이트웨이로 트래픽 라우팅을 참조하십시오.

레거시 네트워크 스크립트를 사용하는 경우 다음 구성 파일이 정책 기반 라우팅에 관련되어 있습니다.

- **/etc/sysconfig/network-scripts/route-interface:** 이 파일은 IPv4 경로를 정의합니다. **table** 옵션을 사용하여 라우팅 테이블을 지정합니다. 예를 들면 다음과 같습니다.

```
192.0.2.0/24 via 198.51.100.1 table 1
203.0.113.0/24 via 198.51.100.2 table 2
```

- **/etc/sysconfig/network-scripts/route6-interface:** 이 파일은 IPv6 경로를 정의합니다.

- **/etc/sysconfig/network-scripts/rule-interface:** 이 파일은 커널이 특정 라우팅 테이블로 트래픽을 라우팅하는 IPv4 소스 네트워크에 대한 규칙을 정의합니다. 예를 들면 다음과 같습니다.

```
from 192.0.2.0/24 lookup 1
from 203.0.113.0/24 lookup 2
```

- **/etc/sysconfig/network-scripts/rule6-interface:** 이 파일은 커널이 특정 라우팅 테이블로 트래픽을 라우팅하는 IPv6 소스 네트워크에 대한 규칙을 정의합니다.

- **/etc/iproute2/rt\_tables:** 이 파일은 특정 라우팅 테이블을 참조하기 위해 숫자 대신 이름을 사용하려면 매핑을 정의합니다. 예를 들면 다음과 같습니다.

```
1 Provider_A
2 Provider_B
```

추가 리소스

- **ip-route(8) 도움말 페이지**
- **IP-rule(8) 도움말 페이지**

### 24.3. 레거시 네트워크 스크립트를 사용하여 특정 서브넷의 트래픽을 다른 기본 게이트웨이로 라우팅

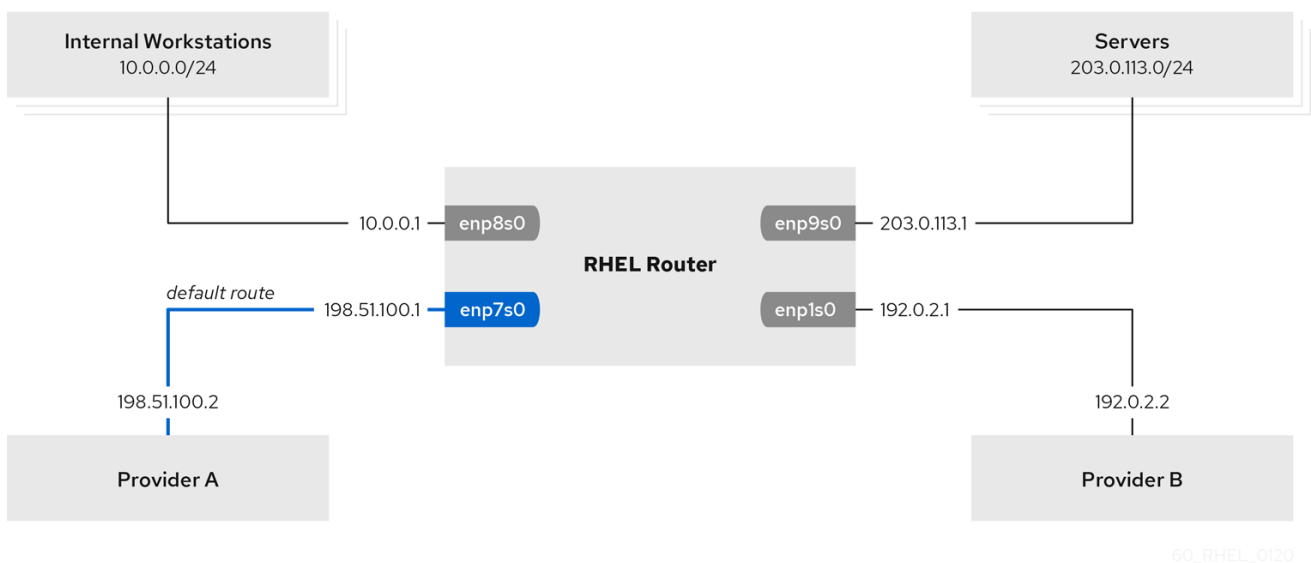
이 섹션에서는 기본적으로 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 **A**로 라우팅하는 라우터로 **RHEL**을 구성하는 방법에 대해 설명합니다. **RHEL**은 정책 기반 라우팅을 사용하여 내부 워크스테이션 서브넷에서 받은 트래픽을 프로바이더 **B**로 라우팅합니다.

## 중요

**network-scripts** 패키지에서 제공하는 레거시 네트워크 스크립트를 사용하여 네트워크 구성은 **RHEL 8**에서 더 이상 사용되지 않습니다. 호스트에서 **NetworkManager** 대신 레거시 네트워크 스크립트를 사용하는 경우에만 이 섹션의 절차를 따릅니다.

**NetworkManager**를 사용하여 네트워크 설정을 관리하는 경우 **NetworkManager**를 사용하여 특정 서브넷에서 다른 기본 게이트웨이로 트래픽 라우팅을 참조하십시오.

이 절차에서는 다음 네트워크 토폴로지를 가정합니다.



## 참고

레거시 네트워크 스크립트는 구성 파일을 알파벳순으로 처리합니다. 따라서 인터페이스에 다른 인터페이스의 규칙 및 경로에 사용되는 인터페이스가 종속 인터페이스에 필요한 경우 시작되도록 구성 파일의 이름을 지정해야 합니다. 올바른 순서를 달성하기 위해 이 절차에서는 **ifcfg-\***, **route-\***, **rules-\*** 파일의 번호를 사용합니다.

## 사전 요구 사항

- **NetworkManager** 패키지가 설치되어 있지 않거나 **NetworkManager** 서비스가 비활성화됩니다.
- **network-scripts** 패키지가 설치되어 있습니다.

- 프로시저에서 설정할 **RHEL** 라우터에는 네 개의 네트워크 인터페이스가 있습니다.
  - **enp7s0** 인터페이스는 공급자 **A**의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 IP는 **198.51.100.2** 입니다. 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp1s0** 인터페이스는 공급자 **B**의 네트워크에 연결되어 있습니다. 공급자의 네트워크의 게이트웨이 IP는 **192.0.2.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp8s0** 인터페이스는 내부 워크스테이션이 있는 **10.0.0.0/24** 서브넷에 연결됩니다.
  - **enp9s0** 인터페이스는 회사 서버를 사용하여 **203.0.113.0/24** 서브넷에 연결됩니다.
- 내부 워크스테이션 서브넷의 호스트는 기본 게이트웨이로 **10.0.0.1** 을 사용합니다. 이 절차에서는 이 IP 주소를 라우터의 **enp8s0** 네트워크 인터페이스에 할당합니다.
- 서버 서브넷의 호스트는 기본 게이트웨이로 **203.0.113.1** 을 사용합니다. 이 절차에서는 라우터의 **enp9s0** 네트워크 인터페이스에 이 IP 주소를 할당합니다.
- **firewalld** 서비스가 활성화되고 활성화됩니다.

## 절차

1.

다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/ifcfg-1\_Provider-A** 파일을 만들어 네트워크 인터페이스의 구성을 공급자 **A**에 추가합니다.

```
TYPE=Ethernet
IPADDR=198.51.100.1
PREFIX=30
GATEWAY=198.51.100.2
DNS1=198.51.100.200
DEFROUTE=yes
NAME=1_Provider-A
DEVICE=enp7s0
ONBOOT=yes
ZONE=external
```



다음 목록에서는 구성 파일에서 사용되는 매개변수를 설명합니다.

- **TYPE=이더넷:** 연결 유형이 이더넷임을 정의합니다.
- **IPADDR=IP\_address:** IPv4 주소를 설정합니다.
- **PREFIX=subnet\_mask:** 서브넷 마스크를 설정합니다.
- **GATEWAY=IP\_address:** 기본 게이트웨이 주소를 설정합니다.
- **DNS1=IP\_of\_DNS\_server:** DNS 서버의 IPv4 주소를 설정합니다.
- **DEFROUTE=yes|no:** 연결이 기본 경로인지 여부를 정의합니다.
- **NAME=connection\_name:** 연결 프로필의 이름을 설정합니다. 혼동을 피하려면 의미 있는 이름을 사용합니다.
- **DEVICE=network\_device:** 네트워크 인터페이스를 설정합니다.
- **ONBOOT=yes:** RHEL이 시스템이 부팅될 때 이 연결을 시작하도록 정의합니다.
- **ZONE=firewalld\_zone:** 정의된 firewalld 영역에 네트워크 인터페이스를 할당합니다. firewalld 는 외부 영역에 할당된 인터페이스에 대해 자동으로 마스커레이딩을 활성화합니다.

2.

네트워크 인터페이스의 구성을 공급자 **B**에 추가합니다.

a.

다음 콘텐츠를 사용하여 `/etc/sysconfig/network-scripts/ifcfg-2_Provider-B` 파일을 만듭니다.

```
TYPE=Ethernet
IPADDR=192.0.2.1
```

```
PREFIX=30
DEFROUTE=no
NAME=2_Provider-B
DEVICE=enp1s0
ONBOOT=yes
ZONE=external
```

이 인터페이스의 구성 파일에는 기본 게이트웨이 설정이 포함되어 있지 않습니다.

b.

**2\_Provider-B** 연결의 게이트웨이를 별도의 라우팅 테이블에 할당합니다. 따라서 다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/route-2\_Provider-B** 파일을 만듭니다.

```
0.0.0.0/0 via 192.0.2.2 table 5000
```

이 항목은 이 게이트웨이를 통해 라우팅된 모든 서브넷의 게이트웨이 및 트래픽을 테이블 **5000**으로 할당합니다.

3.

내부 워크스테이션 서브넷에 대한 네트워크 인터페이스에 대한 구성을 생성합니다.

a.

다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/ifcfg-3\_Internal-Workstations** 파일을 만듭니다.

```
TYPE=Ethernet
IPADDR=10.0.0.1
PREFIX=24
DEFROUTE=no
NAME=3_Internal-Workstations
DEVICE=enp8s0
ONBOOT=yes
ZONE=internal
```

b.

내부 워크스테이션 서브넷의 라우팅 규칙 구성을 추가합니다. 따라서 다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/rule-3\_Internal-Workstations** 파일을 만듭니다.

```
pri 5 from 10.0.0.0/24 table 5000
```

이 구성은 모든 트래픽을 **10.0.0.0/24** 서브넷에서 테이블 **5000**으로 라우팅하는 우선 순위 **5**가 있는 라우팅 규칙을 정의합니다. 낮은 값은 우선 순위가 높습니다.

c.

다음 내용이 포함된 **/etc/sysconfig/network-scripts/route-3\_Internal-Workstations**

파일을 만들어 **ID 5000** 이 있는 라우팅 테이블에 정적 경로를 추가합니다.

```
10.0.0.0/24 via 192.0.2.1 table 5000
```

이 고정 경로는 **RHEL**이 **10.0.0.0/24** 서브넷에서 로컬 네트워크 인터페이스의 **IP**로 트래픽을 공급자 **B(192.0.2.1)**로 전송함을 정의합니다. 이 인터페이스는 테이블 **5000** 을 라우팅하고 다음 홉으로 사용됩니다.

4.

다음 콘텐츠와 함께 **/etc/sysconfig/network-scripts/ifcfg-4\_Servers** 파일을 만들어 네트워크 인터페이스의 구성을 서버 서브넷에 추가합니다.

```
TYPE=Ethernet
IPADDR=203.0.113.1
PREFIX=24
DEFROUTE=no
NAME=4_Servers
DEVICE=enp9s0
ONBOOT=yes
ZONE=internal
```

5.

네트워크를 재시작합니다.

```
systemctl restart network
```

## 검증 단계

1.

내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

a.

**traceroute** 패키지를 설치합니다.

```
yum install traceroute
```

b.

**traceroute** 유틸리티를 사용하여 인터넷의 호스트에 대한 경로를 표시합니다.

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.337 ms 0.260 ms 0.223 ms
 2 192.0.2.1 (192.0.2.1) 0.884 ms 1.066 ms 1.248 ms
 ...
```

명령 출력은 라우터가 공급자 **B**의 네트워크인 **192.0.2.1** 을 통해 패킷을 전송함을 표시합니다.

2.

서버 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

a.

**traceroute** 패키지를 설치합니다.

```
yum install traceroute
```

b.

**traceroute** 유틸리티를 사용하여 인터넷의 호스트에 대한 경로를 표시합니다.

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 203.0.113.1 (203.0.113.1) 2.179 ms 2.073 ms 1.944 ms
 2 198.51.100.2 (198.51.100.2) 1.868 ms 1.798 ms 1.549 ms
 ...
```

명령 출력은 라우터가 공급자 **A**의 네트워크인 **198.51.100.2** 를 통해 패킷을 전송함을 표시합니다.

## 문제 해결 단계

**RHEL** 라우터에서 다음을 수행합니다.

1.

규칙 목록을 표시합니다.

```
ip rule list
0: from all lookup local
5: from 10.0.0.0/24 lookup 5000
32766: from all lookup main
32767: from all lookup default
```

기본적으로 **RHEL**에는 로컬, 기본, 기본 테이블에 대한 규칙이 포함되어 있습니다.

2.

표 **5000** 에 경로를 표시합니다.

```
ip route list table 5000
default via 192.0.2.2 dev enp1s0
10.0.0.0/24 via 192.0.2.1 dev enp1s0
```

3.

인터페이스 및 방화벽 영역을 표시합니다.

```
firewall-cmd --get-active-zones
external
 interfaces: enp1s0 enp7s0
internal
 interfaces: enp8s0 enp9s0
```

4.

외부 영역에 마스커레이딩이 활성화되어 있는지 확인합니다.

```
firewall-cmd --info-zone=external
external (active)
 target: default
 icmp-block-inversion: no
 interfaces: enp1s0 enp7s0
 sources:
 services: ssh
 ports:
 protocols:
 masquerade: yes
 ...
```

#### 추가 리소스

- [레거시 네트워크 스크립트를 사용할 때 정책 기반 라우팅과 관련된 구성 파일의 개요](#)
- [ip-route\(8\) 도움말 페이지](#)
- [IP-rule\(8\) 도움말 페이지](#)
- [/usr/share/doc/network-scripts/sysconfig.txt](#)

## 25장. 더미 인터페이스 생성

**Red Hat Enterprise Linux** 사용자는 디버깅 및 테스트 목적으로 더미 네트워크 인터페이스를 만들고 사용할 수 있습니다. 더미 인터페이스는 실제로 전송하지 않고 패킷을 라우팅할 장치를 제공합니다. **NetworkManager**에서 관리하는 루프백과 같은 장치를 추가로 생성할 수 있으며 비활성 **SLIP(Serial Line Internet Protocol)** 주소가 로컬 프로그램의 실제 주소처럼 표시되도록 할 수 있습니다.

### 25.1. NMCLI를 사용하여 IPV4 및 IPV6 주소가 모두 있는 더미 인터페이스 만들기

다양한 설정을 사용하여 더미 인터페이스를 만들 수 있습니다. 다음 절차에서는 **IPv4** 및 **IPv6** 주소를 모두 사용하여 더미 인터페이스를 만드는 방법을 설명합니다. 더미 인터페이스를 만든 후 **NetworkManager**는 자동으로 기본 공용 방화벽 영역에 할당합니다.



#### 참고

**IPv4** 또는 **IPv6** 주소 없이 더미 인터페이스를 구성하려면 **ipv4.method** 및 **ipv6.method** 매개 변수를 비활성화 로 설정합니다. 그렇지 않으면 **IP** 자동 구성이 실패하고 **NetworkManager**는 연결을 비활성화하고 더미 장치를 제거합니다.

#### 절차

1.

정적 **IPv4** 및 **IPv6** 주소를 사용하는 **dummy0**이라는 더미 인터페이스를 생성하려면 다음을 입력합니다.

```
nmcli connection add type dummy ifname dummy0 ipv4.method manual
ipv4.addresses 192.0.2.1/24 ipv6.method manual ipv6.addresses 2001:db8:2::1/64
```

2.

선택 사항: 더미 인터페이스를 보려면 다음을 입력합니다.

```
nmcli connection show
NAME UUID TYPE DEVICE
enp1s0 db1060e9-c164-476f-b2b5-caec62dc1b05 ethernet ens3
dummy-dummy0 aaf6eb56-73e5-4746-9037-eed42caa8a65 dummy dummy0
```

#### 추가 리소스



**nm-settings(5)** 도움말 페이지

## 26장. NMSTATE-AUTOCONF를 사용하여 LLDP를 사용하여 네트워크 상태 자동 구성

네트워크 장치는 **LLDP**(링크 계층 검색 프로토콜)를 사용하여 **LAN**에서 **ID**, 기능 및 인접 항목을 알릴 수 있습니다. **nmstate-autoconf** 유틸리티는 이 정보를 사용하여 로컬 네트워크 인터페이스를 자동으로 구성할 수 있습니다.

### 중요

**nmstate-autoconf** 유틸리티는 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(**SLA**)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있으며 프로덕션에는 사용하지 않는 것이 좋습니다. 이러한 미리보기를 통해 향후 제품 기능에 조기에 액세스할 수 있어 고객이 개발 과정에서 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능에 대한 지원 범위에 대한 자세한 내용은 **Red Hat** 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

### 26.1. NMSTATE-AUTOCONF를 사용하여 네트워크 인터페이스 자동 구성

**nmstate-autoconf** 유틸리티는 **LLDP**를 사용하여 로컬 장치를 구성하기 위해 스위치에 연결된 인터페이스의 **VLAN** 설정을 식별합니다.

이 절차에서는 다음 시나리오와 **LLDP**를 사용하여 **VLAN** 설정을 브로드캐스트하는 것으로 가정합니다.

- **RHEL** 서버의 **enp1s0** 및 **enp2s0** 인터페이스는 **VLAN ID 100** 및 **VLAN** 이름 **prod-net** 로 구성된 포트를 전환하도록 연결됩니다.
- **RHEL** 서버의 **enp3s0** 인터페이스는 **VLAN ID 200** 및 **VLAN** 이름 **RWX -net** 으로 구성된 스위치 포트에 연결됩니다.

그런 다음 **nmstate-autoconf** 유틸리티를 사용하여 이 정보를 사용하여 서버에 다음 인터페이스를 생성합니다.

- **bond100** - **enp1s0** 및 **enp2s0** 포트가 있는 본딩 인터페이스입니다.

- **prod-net - bond100** 상단에 **VLAN ID 100** 이 있는 **VLAN** 인터페이스입니다.
- **Mgmt-net - VLAN ID가 200인 enp3s0** 상단에 있는 **VLAN** 인터페이스

**LLDP**가 동일한 **VLAN ID**를 브로드캐스트하는 다른 스위치 포트에 여러 네트워크 인터페이스를 연결하면 **nmstate-autoconf** 가 이러한 인터페이스와의 본딩을 생성하고 그 위에 공통 **VLAN ID**를 구성합니다.

#### 사전 요구 사항

- **nmstate** 패키지가 설치되어 있습니다.
- **LLDP**는 네트워크 스위치에서 활성화됩니다.
- 이더넷 인터페이스가 작동 중입니다.

#### 절차

1. 이더넷 인터페이스에서 **LLDP**를 활성화합니다.
  - a. 다음 콘텐츠를 사용하여 **YAML** 파일(예: **~/enable-lldp.yml**)을 만듭니다.

```
interfaces:
 - name: enp1s0
 type: ethernet
 lldp:
 enabled: true
 - name: enp2s0
 type: ethernet
 lldp:
 enabled: true
 - name: enp3s0
 type: ethernet
 lldp:
 enabled: true
```

- b. 시스템에 설정을 적용합니다.



```
nmstatectl apply ~/enable-lldp.yml
```

2.

LLDP를 사용하여 네트워크 인터페이스를 구성합니다.

a.

선택 사항인 시험 실행을 시작하여 **nmstate-autoconf**가 생성하는 **YAML** 구성을 표시하고 확인합니다.

```
nmstate-autoconf -d enp1s0,enp2s0,enp3s0

interfaces:
- name: prod-net
 type: vlan
 state: up
 vlan:
 base-iface: bond100
 id: 100
- name: mgmt-net
 type: vlan
 state: up
 vlan:
 base-iface: enp3s0
 id: 200
- name: bond100
 type: bond
 state: up
 link-aggregation:
 mode: balance-rr
 port:
 - enp1s0
 - enp2s0
```

b.

**nmstate-autoconf**를 사용하여 **LLDP**에서 수신한 정보를 기반으로 구성을 생성하고 해당 설정을 시스템에 적용합니다.

```
nmstate-autoconf enp1s0,enp2s0,enp3s0
```

다음 단계

•

네트워크에 **IP** 설정을 제공하는 **DHCP** 서버가 인터페이스에 없는 경우 수동으로 구성합니다. 자세한 내용은 다음을 참조하십시오.

○

[이더넷 연결 구성](#)

- 

### 네트워크 본딩 구성

#### 검증

1. 개별 인터페이스의 설정을 표시합니다.

```
nmstatectl show <interface_name>
```

#### 추가 리소스

- [\*\*\*nmstate-autoconf\(8\)\*\*\* 도움말 페이지](#)

## 27장. LLDP를 사용하여 네트워크 구성 문제 디버그

**LLDP(연결 계층 검색 프로토콜)**를 사용하여 토폴로지에서 네트워크 구성 문제를 디버깅할 수 있습니다. 즉, **LLDP**는 다른 호스트 또는 라우터 및 스위치와 구성 불일치를 보고할 수 있습니다.

### 27.1. LLDP 정보를 사용하여 잘못된 VLAN 구성 디버깅

특정 **VLAN**을 사용하도록 스위치 포트를 구성하고 호스트에서 이러한 **VLAN** 패킷을 수신하지 않는 경우 **LLDP(Link Layer Discovery Protocol)**를 사용하여 문제를 디버깅할 수 있습니다. 패킷을 수신하지 않는 호스트에서 다음 절차를 수행하십시오.

#### 사전 요구 사항

- **nmstate** 패키지가 설치되어 있습니다.
- 이 스위치는 **LLDP**를 지원합니다.
- **LLDP**는 인접 장치에서 사용할 수 있습니다.

#### 절차

1. 다음 콘텐츠를 사용하여 **~/enable-LLDP-enp1s0.yml** 파일을 만듭니다.

```
interfaces:
 - name: enp1s0
 type: ethernet
 lldp:
 enabled: true
```

2. **~/enable-LLDP-enp1s0.yml** 파일을 사용하여 **enp1s0** 인터페이스에서 **LLDP**를 활성화합니다.

```
nmstatectl apply ~/enable-LLDP-enp1s0.yml
```

3. **LLDP** 정보를 표시합니다.

```
nmstatectl show enp1s0
- name: enp1s0
```

```

type: ethernet
state: up
ipv4:
 enabled: false
 dhcp: false
ipv6:
 enabled: false
 autoconf: false
 dhcp: false
lldp:
 enabled: true
 neighbors:
 - type: 5
 system-name: Summit300-48
 - type: 6
 system-description: Summit300-48 - Version 7.4e.1 (Build 5)
 05/27/05 04:53:11
 - type: 7
 system-capabilities:
 - MAC Bridge component
 - Router
 - type: 1
 _description: MAC address
 chassis-id: 00:01:30:F9:AD:A0
 chassis-id-type: 4
 - type: 2
 _description: Interface name
 port-id: 1/1
 port-id-type: 5
 - type: 127
 ieee-802-1-vlans:
 - name: v2-0488-03-0505
 vid: 488
 oui: 00:80:c2
 subtype: 3
 - type: 127
 ieee-802-3-mac-phy-conf:
 autoneg: true
 operational-mau-type: 16
 pmd-autoneg-cap: 27648
 oui: 00:12:0f
 subtype: 1
 - type: 127
 ieee-802-1-ppvids:
 - 0
 oui: 00:80:c2
 subtype: 2
 - type: 8
 management-addresses:
 - address: 00:01:30:F9:AD:A0
 address-subtype: MAC
 interface-number: 1001
 interface-number-subtype: 2
 - type: 127
 ieee-802-3-max-frame-size: 1522
 oui: 00:12:0f

```

```

 subtype: 4
 mac-address: 82:75:BE:6F:8C:7A
 mtu: 1500

```

4.

출력을 확인하여 설정이 예상 구성과 일치하는지 확인합니다. 예를 들어 스위치에 연결된 인터페이스의 **LLDP** 정보는 이 호스트가 연결된 스위치 포트가 **VLAN ID 448** 을 사용하도록 연결되었음을 나타냅니다.

```

- type: 127
 ieee-802-1-vlans:
 - name: v2-0488-03-0505
 vid: 488

```

**enp1s0** 인터페이스의 네트워크 구성이 다른 **VLAN ID**를 사용하는 경우 적절하게 변경합니다.

추가 리소스

**VLAN 태그 지정 구성**

## 28장. 수동으로 키 파일 형식으로 NETWORKMANAGER 프로파일 생성

**NetworkManager**는 키 파일 형식으로 저장된 프로파일을 지원합니다. 그러나 기본적으로 **nmcli**, **network RHEL System Role** 또는 **nmstate API**와 같은 **NetworkManager** 유틸리티를 사용하여 프로파일을 관리하는 경우 **NetworkManager**는 **ifcfg** 형식의 프로파일을 계속 사용합니다.

다음 주요 **RHEL** 릴리스에서 키 파일 형식이 기본값입니다.

### 28.1. NETWORKMANAGER 프로파일의 키 파일 형식

**NetworkManager**는 연결 프로파일을 디스크에 저장할 때 **INI** 스타일 키 파일 형식을 사용합니다.

키 파일 형식의 인터넷 연결 프로파일의 예

```
[connection]
id=example_connection
uuid=82c6272d-1ff7-4d56-9c7c-0eb27c300029
type=ethernet
autoconnect=true

[ipv4]
method=auto

[ipv6]
method=auto

[ethernet]
mac-address=00:53:00:8f:fa:66
```

각 섹션은 **nm-settings(5)** 및 **nm-settings-keyfile(5)** 도움말 페이지에 설명된 대로 **NetworkManager** 설정 이름에 해당합니다. 섹션의 각 키-값 쌍은 도움말 페이지의 설정 사양에 나열된 속성 중 하나입니다.

**NetworkManager** 키 파일의 대부분의 변수에는 일대일 매핑이 있습니다. 즉, **NetworkManager** 속성은 동일한 이름의 변수 및 동일한 형식으로 키 파일에 저장됩니다. 그러나 주로 키 파일 구문을 읽기 쉽게 만드는 예외가 있습니다. 이러한 예외 목록은 **nm-settings-keyfile(5)** 매뉴얼 페이지를 참조하십시오.



## 중요

보안상의 이유로 연결 프로필에는 개인 키 및 암호와 같은 중요한 정보가 포함될 수 있으므로 **NetworkManager**는 **root**가 소유한 구성 파일만 사용하며 루트에서만 읽고 쓸 수 있습니다.

연결 프로필의 목적에 따라 다음 디렉토리 중 하나에 저장합니다.

- **/etc/NetworkManager/system-connections/**: 편집할 수 있는 사용자가 생성한 영구 프로필의 일반 위치입니다. **NetworkManager**는 자동으로 **/etc/NetworkManager/system-connections/**에 복사합니다.
- **/run/NetworkManager/system-connections/**: 시스템을 재부팅할 때 자동으로 제거되는 임시 프로필의 경우.
- **/usr/lib/NetworkManager/system-connections/**: 변경 불가능한 프로필의 경우. **NetworkManager API**를 사용하여 이러한 프로필을 편집하는 경우 **NetworkManager**는 이 프로필을 영구 스토리지 또는 임시 스토리지에 복사합니다.

**NetworkManager**는 디스크에서 프로파일을 자동으로 다시 로드하지 않습니다. 키file 형식으로 연결 프로필을 생성하거나 업데이트하는 경우 **nmcli connection reload** 명령을 사용하여 **NetworkManager**에 변경 사항을 알립니다.

## 28.2. 키 파일 형식으로 NETWORKMANAGER 프로파일 생성

이 섹션에서는 키 파일 형식으로 **NetworkManager** 연결 프로파일을 수동으로 생성하는 방법에 대한 일반적인 절차에 대해 설명합니다.



## 참고

구성 파일을 수동으로 생성하거나 업데이트하면 예기치 않거나 작동하지 않는 네트워크 구성이 발생할 수 있습니다. **NetworkManager** 유틸리티(예: **nmcli**, **network RHEL System Role** 또는 **nmstate API**)를 사용하여 **NetworkManager** 연결을 관리하는 것이 좋습니다.

## 절차

1.

이더넷과 같은 하드웨어 인터페이스에 대한 프로필을 생성하는 경우 이 인터페이스의 **MAC**

주소를 표시합니다.

```
ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
 link/ether 00:53:00:8f:fa:66 brd ff:ff:ff:ff:ff:ff
```

2.

연결 프로필을 만듭니다. 예를 들어 **DHCP**를 사용하는 이더넷 장치의 연결 프로필의 경우 다음 콘텐츠를 사용하여 `/etc/NetworkManager/system-connections/example.nmconnection` 파일을 만듭니다.

```
[connection]
id=example_connection
type=ethernet
autoconnect=true

[ipv4]
method=auto

[ipv6]
method=auto

[ethernet]
mac-address=00:53:00:8f:fa:66
```



참고

모든 파일 이름은 **.nmconnection** 접미사와 함께 사용할 수 있습니다. 그러나 나중에 **nmcli** 명령을 사용하여 연결을 관리하는 경우 이 연결을 참조할 때 **id** 변수에 설정된 연결 이름을 사용해야 합니다. **id** 변수를 생략하면 **.nmconnection** 없이 파일 이름을 사용하여 이 연결을 참조합니다.

3.

**root** 사용자만 읽고 업데이트할 수 있도록 구성 파일에 대한 권한을 설정합니다.

```
chown root:root /etc/NetworkManager/system-connections/example.nmconnection
chmod 600 /etc/NetworkManager/system-connections/example.nmconnection
```

4.

연결 프로필을 다시 로드합니다.

```
nmcli connection reload
```

5.

**NetworkManager**가 구성 파일에서 프로필을 읽는지 확인합니다.



```
nmcli -f NAME,UUID,FILENAME connection
NAME UUID FILENAME
example-connection 86da2486-068d-4d05-9ac7-957ec118afba
/etc/NetworkManager/system-connections/example.nmconnection
...
```

명령이 새로 추가한 연결을 표시하지 않으면 파일 권한 및 파일에서 사용한 구문이 올바른지 확인합니다.

6.

선택 사항: 프로파일에서 자동 연결 변수를 **false** 로 설정하면 연결을 활성화합니다.

```
nmcli connection up example_connection
```

검증

1.

연결 프로파일을 표시합니다.

```
nmcli connection show example_connection
```

2.

인터페이스의 IP 설정을 표시합니다.

```
ip address show enp1s0
```

추가 리소스

- 

**nm-settings-keyfile (5)**

## 29장. NETCONSOLE을 사용하여 네트워크에서 커널 메시지 기록

**netconsole** 커널 모듈과 **same-named** 서비스를 사용하여 디스크에 로깅이 실패하거나 직렬 콘솔을 사용할 수 없는 경우 커널 메시지를 네트워크를 통해 로깅하여 커널을 디버깅할 수 있습니다.

### 29.1. 커널 메시지를 원격 호스트에 기록하도록 NETCONSOLE 서비스 구성

**netconsole** 커널 모듈을 사용하여 커널 메시지를 원격 시스템 로그 서비스에 기록할 수 있습니다.

#### 사전 요구 사항

- **rsyslog** 와 같은 시스템 로그 서비스는 원격 호스트에 설치됩니다.
- 원격 시스템 로그 서비스는 이 호스트에서 들어오는 로그 항목을 수신하도록 구성되어 있습니다.

#### 절차

1. **netconsole-service** 패키지를 설치합니다.

```
yum install netconsole-service
```

2. **/etc/sysconfig/netconsole** 파일을 편집하고 **SYSLOGADDR** 매개 변수를 원격 호스트의 IP 주소로 설정합니다.

```
SYSLOGADDR=192.0.2.1
```

3. **netconsole** 서비스를 활성화하고 시작합니다.

```
systemctl enable --now netconsole
```

#### 검증 단계

- 원격 시스템 로그 서버의 **/var/log/messages** 파일을 표시합니다.

#### 추가 리소스

- 

원격 로깅 솔루션 구성

### 30장. SYSTEMD 네트워크 대상 및 서비스

**NetworkManager**는 시스템 부팅 프로세스 중에 네트워크를 구성합니다. 그러나 루트 디렉터리가 **iSCSI** 장치에 저장된 경우와 같이 원격 루트(/)로 부팅하는 경우 **RHEL**을 시작하기 전에 네트워크 설정이 초기 **RAM** 디스크(**initrd**)에 적용됩니다. 예를 들어, 네트워크 구성이 **rd.neednet=1**을 사용하여 커널 명령줄에 지정되거나 원격 파일 시스템을 마운트하도록 구성이 지정된 경우 네트워크 설정이 **initrd**에 적용됩니다.

이 섹션에서는 네트워크 설정을 적용하는 동안 사용되는 **network**, **network-online** 및 **NetworkManager-wait-online** 서비스와 같은 다양한 대상과 **network-online** 서비스가 시작된 후 시작되도록 **systemd** 서비스를 구성하는 방법에 대해 설명합니다.

#### 30.1. 네트워크 및 네트워크 온라인 SYSTEMD 대상의 차이점

**systemd**는 네트워크 및 **network-online** 대상 장치를 유지 관리합니다. **NetworkManager-wait-online.service**와 같은 특수 단위에는 **WantedBy=network-online.target** 및 **Before=network-online.target** 매개 변수가 있습니다. 활성화된 경우 이러한 장치는 **network-online.target**으로 시작하고 일부 네트워크 연결 형식이 설정될 때까지 대상에 도달할 대상을 지연합니다. 네트워크가 연결될 때까지 **network-online** 대상을 지연시킵니다.

**network-online** 대상은 서비스를 시작하여 추가 실행에 상당한 지연이 추가되었습니다. **systemd**는 **Wants** 및 이 대상 단위의 매개 변수에 대한 종속성을 \$ 네트워크 기능을 참조하는 **Linux Standard Base(LSB)** 헤더가 있는 모든 **SysV(System V) init** 스크립트 서비스 장치에 자동으로 추가합니다. **LSB** 헤더는 **init** 스크립트의 메타데이터입니다. 이를 사용하여 종속성을 지정할 수 있습니다. 이는 **systemd** 대상과 유사합니다.

네트워크 대상은 부팅 프로세스의 실행을 크게 지연시키지 않습니다. 네트워크 대상에 도달한다는 것은 네트워크 설정을 담당하는 서비스가 시작되었음을 의미합니다. 그러나 네트워크 장치가 구성되었음을 의미하지는 않습니다. 이 대상은 시스템을 종료하는 동안 중요합니다. 예를 들어 부팅 중에 네트워크 대상 다음에 정렬된 서비스가 있는 경우 종료 중에 이 종속성이 반전됩니다. 네트워크가 서비스가 중지될 때까지 연결이 끊어지지 않습니다. 원격 네트워크 파일 시스템의 모든 마운트 단위는 자동으로 네트워크-온라인 대상 장치를 시작하고 그 후에 자체적으로 주문합니다.

#### 참고

네트워크 온라인 대상 장치는 시스템이 시작되는 동안에만 유용합니다. 시스템 부팅이 완료되면 이 대상은 네트워크의 온라인 상태를 추적하지 않습니다. 따라서 **network-online**을 사용하여 네트워크 연결을 모니터링할 수 없습니다. 이 대상은 일회성 시스템 시작 개념을 제공합니다.

#### 30.2. NETWORKMANAGER-WAIT-ONLINE 개요

동기 레거시 네트워크 스크립트는 모든 구성 파일을 반복하여 장치를 설정합니다. 모든 네트워크 관련 구성을 적용하고 네트워크가 온라인 상태인지 확인합니다.

**NetworkManager-wait-online** 서비스는 네트워크를 구성할 때까지 시간 초과를 기다립니다. 이 네트워크 구성에는 이더넷 장치 연결, Wi-Fi 장치 검사 등이 포함됩니다. **NetworkManager**는 자동으로 시작되도록 구성된 적합한 프로필을 자동으로 활성화합니다. **DHCP** 시간 제한 또는 유사한 이벤트로 인해 자동 활성화 프로세스가 실패하면 **NetworkManager**가 장기간 사용하지 않을 수 있습니다. 구성에 따라 **NetworkManager**는 동일한 프로필 또는 다른 프로필 활성화를 다시 시도합니다.

시작이 완료되면 모든 프로필이 연결이 끊긴 상태이거나 성공적으로 활성화됩니다. 자동 연결을 위해 프로필을 구성할 수 있습니다. 다음은 연결이 활성화된 것으로 간주되는 시기를 정의하는 매개변수의 몇 가지 예입니다.

- **connection.wait-device-timeout** - 드라이버에서 장치를 감지하도록 시간 초과를 설정합니다.
- **ipv4.may-fail** 및 **ipv6.may-fail** - 하나의 IP 주소 제품군으로 활성화를 설정하거나 특정 주소 제품군에 구성을 완료했는지 여부를 설정합니다.
- **ipv4.gateway-ping-timeout** - delays activation.

추가 리소스

- **nm-settings(5)** 도움말 페이지

### 30.3. 네트워크를 시작한 후 시작하도록 **SYSTEMD** 서비스 구성

**Red Hat Enterprise Linux**는 **/usr/lib/systemd/system/** 디렉터리에 **systemd** 서비스 파일을 설치합니다. 이 절차에서는 **/etc/systemd/system/service\_name.service.d/**에 있는 서비스 파일의 드롭인 스니펫을 생성하여 **/usr/lib/systemd/system/**의 서비스 파일과 함께 사용하여 네트워크가 온라인 상태가 된 후 특정 서비스를 시작합니다. 드롭인 스니펫의 설정이 **/usr/lib/systemd/system/**의 서비스 파일에 있는 설정과 겹치는 경우 우선순위가 더 높습니다.

절차

1. 편집기에서 서비스 파일을 열려면 다음을 입력합니다.

```
systemctl edit service_name
```

2.

다음을 입력하고 변경 사항을 저장합니다.

```
[Unit]
After=network-online.target
```

3.

**systemd** 서비스를 다시 로드합니다.

```
systemctl daemon-reload
```

## 31장. LINUX 트래픽 제어

**Linux**는 패킷 전송의 관리 및 조작을 위한 도구를 제공합니다. **MTC(Linux 트래픽 제어)** 하위 시스템은 네트워크 트래픽을 정책화, 분류, 구성 및 스케줄링하는 데 도움이 됩니다. 또한 **TC**는 필터와 작업을 사용하여 분류 중에 패킷 콘텐츠를 무시합니다. **TC** 하위 시스템은 **TC** 아키텍처의 기본 요소인 대기열 원칙(**qdisc**)을 사용하여 이를 달성합니다.

스케줄링 메커니즘은 다른 큐를 입력하거나 종료하기 전에 패킷을 정렬하거나 다시 정렬합니다. 가장 일반적인 스케줄러는 첫 번째 **FIFO(First-In-First-Out)** 스케줄러입니다. **qdiscs** 작업을 임시로 **the tc** 유틸리티를 사용하거나 **NetworkManager**를 사용하여 영구적으로 수행할 수 있습니다.

이 섹션에서는 대기열 분야에 대해 설명하고 **RHEL**의 기본 **qdisc**를 업데이트하는 방법을 설명합니다.

### 31.1. 대기열 분야의 개요

큐잉 정제(**qdiscs**)는 네트워크 인터페이스의 트래픽 전송 일정을 정복하고 나중에 스케줄링하는 데 도움이 됩니다. **qdisc**에는 두 가지 작업이 있습니다.

- 나중에 전송하기 위해 패킷을 대기열에 추가할 수 있도록 **Enqueue** 요청
- 즉시 전송을 위해 대기 중인 패킷 중 하나를 선택할 수 있도록 대기열을 비활성화합니다.

모든 **qdisc**에는 1 또는 **abcd**와 같은 연결된 콜론을 사용하여 처리 라는 16비트 16진수 식별 번호가 있습니다. 이 수를 **qdisc** 주요 번호라고 합니다. **qdisc**에 클래스가 있는 경우 식별자는 마이너 앞에 페이지 번호를 가진 두 숫자의 쌍으로 구성됩니다. **<major>:<minor>** (예: **abcd:1**). 마이너 숫자의 번호 지정 체계는 **qdisc** 유형에 따라 다릅니다. 때때로 숫자가 체계적이기 때문에 첫 번째 클래스에 **<major>:1**, 두 번째 **<major>:2** 등이 있습니다. 일부 **qdiscs**는 사용자가 클래스를 만들 때 클래스의 마이너 번호를 임의로 설정할 수 있도록 허용합니다.

### 다양한 qdiscs

다양한 유형의 **qdisc**가 존재하며 네트워크 인터페이스로 패킷을 전송하는데 도움이 됩니다. **root**, **parent** 또는 자식 클래스를 사용하여 **qdiscs**를 구성할 수 있습니다. 하위 항목을 연결할 수 있는 지점을 클래스라고 합니다. **qdisc**의 클래스는 유연하며 항상 여러 하위 클래스 또는 단일 자식 클래스를 포함할 수 있습니다. 풍부한 **qdisc** 자체를 포함하는 클래스에 대해 금지는 없으므로 복잡한 트래픽 제어 시나리오가 용이해집니다.

비정상적인 **qdiscs**는 패킷을 자체적으로 저장하지 않습니다. 대신 **qdisc**와 관련된 기준에 따라 해당 자식 중 하나에 요청을 큐에 추가 및 큐에 추가합니다. 결국 이 재귀 패킷 통과는 패킷이 저장되는

위치(또는 분리 시 선택)가 종료됩니다.

### classless qdiscs

일부 **qdisc**에는 하위 클래스가 포함되어 있으며 클래스가 없는 **qdiscs** 라고 합니다. **classless qdiscs** 는 클래스화된 **qdiscs** 에 비해 더 적은 사용자 지정이 필요합니다. 일반적으로 인터페이스를 인터페이스에 연결하는 것만으로도 충분합니다.

### 추가 리소스

- **tc(8) man page**
- **Process-actions.8** 도움말 페이지

## 31.2. RHEL에서 사용 가능한 QDISC

각 **qdisc** 는 고유한 네트워킹 관련 문제를 해결합니다. 다음은 **RHEL**에서 사용 가능한 **qdisc**의 목록입니다. 다음 **qdisc** 중 하나를 사용하여 네트워킹 요구 사항에 따라 네트워크 트래픽을 셰이핑할 수 있습니다.

표 31.1. RHEL에서 사용 가능한 스케줄러

| qdisc 이름                                           | 에 포함                        | 오프로드 지원 |
|----------------------------------------------------|-----------------------------|---------|
| 비동기 전송 모드(ATM)                                     | <b>kernel-modules-extra</b> |         |
| 클래스 기반 큐링                                          | <b>kernel-modules-extra</b> |         |
| 크레딧 기반 셰퍼                                          | <b>kernel-modules-extra</b> | 있음      |
| 응답하지 않는 흐름(CHOKE)을 위해 을 선택하고 응답하지 않는 흐름에 대해 계속합니다. | <b>kernel-modules-extra</b> |         |
| 제어된 지연 (CoDel)                                     | <b>kernel-core</b>          |         |
| deficit Round Robin (DRR)                          | <b>kernel-modules-extra</b> |         |
| 차별화된 서비스 마커 (DSANG)                                | <b>kernel-modules-extra</b> |         |
| 향상된 전송 선택 (ETS)                                    | <b>kernel-modules-extra</b> | 있음      |
| 공정 대기열 (FQ)                                        | <b>kernel-core</b>          |         |



| qdisc 이름                                        | 에 포함                 | 오프로드 지원 |
|-------------------------------------------------|----------------------|---------|
| 공정 큐리 제어 지연 (FQ_CODEL)                          | kernel-core          |         |
| GRED(Commonized Random Early Detection)         | kernel-modules-extra |         |
| HSFC(Hierarchical Fair Service Curve)           | kernel-core          |         |
| HHF(High-Hitter Filter)                         | kernel-core          |         |
| 계층 구조 토큰 버킷 (HTB)                               | kernel-core          |         |
| INGRESS                                         | kernel-core          | 있음      |
| 다중 대기열 우선 순위 (MQPRIO)                           | kernel-modules-extra | 있음      |
| Multiqueue (MULTIQ)                             | kernel-modules-extra | 있음      |
| 네트워크 에뮬레이터 (NETEM)                              | kernel-modules-extra |         |
| PPIE(Proportional Integral-controller Enhanced) | kernel-core          |         |
| PLUG                                            | kernel-core          |         |
| 빠른 공정 대기열 (QFQ)                                 | kernel-modules-extra |         |
| RED( 임의 초기 탐지)                                  | kernel-modules-extra | 있음      |
| SFB(Stochastic Fair Blue)                       | kernel-modules-extra |         |
| Stochastic Fairness Queueing(SFQ)               | kernel-core          |         |
| 토큰 버킷 필터 (TBF)                                  | kernel-core          | 있음      |
| Trivial Link Equalizer (TEQL)                   | kernel-modules-extra |         |



### 중요

**qdisc** 오프로드에는 **NIC**에서 하드웨어 및 드라이버 지원이 필요합니다.

### 추가 리소스

- **tc(8), cbq, cbs, cbs,choke,CoDel,drp,fq,htb,mqprio,netem,원형,sfb, pfifo,tc-red,sfq,tbf, prio** 매뉴얼 페이지.

### 31.3. TC 유틸리티를 사용하여 네트워크 인터페이스의 QDISC 검사

기본적으로 **Red Hat Enterprise Linux** 시스템은 **fq\_codel qdisc** 를 사용합니다. 이 절차에서는 **qdisc** 카운터를 검사하는 방법을 설명합니다.

#### 절차

1.

선택 사항: 현재 **qdisc** 보기:

```
tc qdisc show dev enp0s1
```

2.

현재 **qdisc** 카운터를 검사합니다.

```
tc -s qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
Sent 1008193 bytes 5559 pkt (dropped 233, overlimits 55 requeues 77)
backlog 0b 0p requeues 0
....
```

- 삭제됨 - 모든 대기열이 가득 차 있기 때문에 패킷이 삭제된 횟수
- **Overlimits** - 구성된 링크 용량이 채워지는 횟수
- **sent - dequeues** 수

### 31.4. 기본 QDISC 업데이트

현재 **qdisc** 로 네트워킹 패킷 손실이 발생한 경우 네트워크 요구 사항에 따라 **qdisc** 를 변경할 수 있습니다. 네트워크 요구 사항을 충족하는 **qdisc** 을 선택할 수 있습니다. 다음 절차에서는 **Red Hat Enterprise Linux**에서 기본 **qdisc** 를 변경하는 방법을 설명합니다.

#### 절차

1.

현재 기본 **qdisc** 보기:

```
sysctl -a | grep qdisc
net.core.default_qdisc = fq_codel
```

2.

현재 이더넷 연결의 **qdisc** 를 확인합니다.

```
tc -s qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
maxpacket 0 drop_overlimit 0 new_flow_count 0 ecn_mark 0
new_flows_len 0 old_flows_len 0
```

3.

기존 **qdisc** 를 업데이트합니다.

```
sysctl -w net.core.default_qdisc=pfifo_fast
```

4.

변경 사항을 적용하려면 네트워크 드라이버를 다시 로드합니다.

```
rmmod NETWORKDRIVERNAME
```

```
modprobe NETWORKDRIVERNAME
```

5.

네트워크 인터페이스를 시작합니다.

```
IP 링크 enp0s1 up
```

#### 검증 단계

•

이더넷 연결의 **qdisc** 를 확인합니다.

```
tc -s qdisc show dev enp0s1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
Sent 373186 bytes 5333 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
....
```

## 추가 리소스

•

[Red Hat Enterprise Linux에서 sysctl 변수 설정 방법](#)**31.5. TC 유틸리티를 사용하여 네트워크 인터페이스의 현재 QDISK를 일시적으로 설정**

기본 **qdisc**를 변경하지 않고 현재 **qdisc**를 업데이트할 수 있습니다. 다음 절차에서는 **Red Hat Enterprise Linux**에서 현재 **qdisc**를 변경하는 방법을 설명합니다.

## 절차

1.

선택 사항: 현재 **qdisc** 보기:

```
tc -s qdisc show dev enp0s1
```

2.

현재 **qdisc**를 업데이트합니다.

```
Process qdisc dev enp0s1 루트 htb를 대체
```

## 검증 단계

•

업데이트된 현재 **qdisc** 보기:

```
tc -s qdisc show dev enp0s1
qdisc htb 8001: root refcnt 2 r2q 10 default 0 direct_packets_stat 0 direct_qlen 1000
Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
```

**31.6. NETWORKMANAGER를 사용하여 네트워크 인터페이스의 현재 QDISK 영구 설정**

**NetworkManager** 연결의 현재 **qdisc** 값을 업데이트할 수 있습니다.

## 절차

1.

선택 사항: 현재 **qdisc** 보기:

```
tc qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2
```

2.

현재 **qdisc** 를 업데이트합니다.

```
nmcli connection modify enp0s1 tc.qdiscs 'root pfifo_fast'
```

3.

선택 사항: 기존 **qdisc** 에 다른 **qdisc** 를 추가하려면 **+tc.qdisc** 옵션을 사용합니다.

```
nmcli connection modify enp0s1 +tc.qdisc 'ingress handle ffff:'
```

4.

변경 사항을 활성화합니다.

```
nmcli connection up enp0s1
```

#### 검증 단계

•

현재 **qdisc** 네트워크 인터페이스를 표시합니다.

```
tc qdisc show dev enp0s1
qdisc pfifo_fast 8001: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc ingress ffff: parent ffff:fff1 -----
```

#### 추가 리소스

•

**nm-settings(5)** 도움말 페이지

## 32장. MULTIPATH TCP 시작하기

다중 경로 TCP(MPTCP)는 TCP(Transmission Control Protocol)의 확장입니다. IP(인터넷 프로토콜)를 사용하여 호스트는 대상에 패킷을 보낼 수 있습니다. TCP는 인터넷을 통해 안정적으로 데이터를 제공하고 네트워크 부하에 대응하여 대역폭을 자동으로 조정합니다.

이 섹션에서는 다음을 수행하는 방법에 대해 설명합니다.

- 새 MPTCP 연결 만들기
- 서버가 MPTCP를 사용하도록 설정
- 커널에서 MPTCP 비활성화

또한 MPTCP 사용의 이점도 포함합니다.

### 32.1. MPTCP 이점

다중 경로 TCP(MPTCP) 설계로 연결 안정성이 향상됩니다. MPTCP 용어에서 링크는 경로로 간주됩니다.

MPTCP의 장점은 다음과 같습니다.

- 이를 통해 연결이 여러 네트워크 인터페이스를 동시에 사용할 수 있습니다.
- 연결이 링크 속도에 바인딩된 경우 여러 링크를 사용하면 연결 처리량이 증가할 수 있습니다. 연결이 CPU에 바인딩된 경우 여러 링크를 사용하면 연결이 느려집니다.
- 오류를 연결하는 복원성이 증가합니다.

### 32.2. MPTCP 지원을 사용하도록 RHEL 준비

**RHEL**에서 **MPTCP** 지원은 기본적으로 비활성화되어 있습니다. 이 기능을 지원하는 애플리케이션에서 사용할 수 있도록 **MPTCP**를 활성화합니다. 또한 해당 애플리케이션에 기본적으로 **TCP** 소켓이 있는 경우 **MPTCP** 소켓을 강제로 사용하도록 사용자 공간 애플리케이션을 구성해야 합니다.

이 절차에서는 **sysctl** 툴을 사용하여 **MPTCP** 지원을 활성화하고 **SystemTap** 스크립트를 사용하여 애플리케이션 시스템 전체에 **MPTCP**를 사용하도록 **RHEL**을 준비하는 방법을 설명합니다.

### 사전 요구 사항

다음 패키지가 설치되어 있습니다.

- **kernel-debuginfo**
- **kernel-debuginfo-common**
- **systemtap**
- **systemtap-devel**
- **kernel-devel**
- **iperf3**

### 절차

1. 커널에서 **MPTCP** 소켓을 활성화합니다.

```
echo "net.mptcp.enabled=1" > /etc/sysctl.d/90-enable-MPTCP.conf
sysctl -p /etc/sysctl.d/90-enable-MPTCP.conf
```

2. 커널에서 **MPTCP**가 활성화되어 있는지 확인합니다.

```
sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 1
```

3.

다음 콘텐츠를 사용하여 **MPtcp-app.stap** 파일을 만듭니다.

```
#!/usr/bin/env stap

%{
#include <linux/in.h>
#include <linux/ip.h>
%}

/* according to [1], RSI contains 'type' and RDX
 * contains 'protocol'.
 * [1] https://github.com/torvalds/linux/blob/master/arch/x86/entry/entry_64.S#L79
 */

function mptcpify () %{
 if (CONTEXT->kregs->si == SOCK_STREAM &&
 (CONTEXT->kregs->dx == IPPROTO_TCP ||
 CONTEXT->kregs->dx == 0)) {
 CONTEXT->kregs->dx = IPPROTO_MPTCP;
 STAP_RETVALUE = 1;
 } else {
 STAP_RETVALUE = 0;
 }
}%}

probe kernel.function("__sys_socket") {
 if (mptcpify() == 1) {
 printf("command %16s mptcpified\n", execname());
 }
}
```

4.

**TCP** 대신 **MPTCP** 소켓을 생성하도록 사용자 공간 애플리케이션을 강제 적용합니다.

```
stap -vg mptcp-app.stap
```

참고: 이 작업은 명령 후 시작된 모든 **TCP** 소켓에 영향을 미칩니다. 위의 명령을 **Ctrl+C** 와 중단한 후에도 애플리케이션은 **TCP** 소켓을 계속 사용합니다.

5.

또는 **MPTCP** 사용량이 특정 애플리케이션만 사용하도록 허용하려면 다음 콘텐츠를 사용하여 **MP tcp-app.stap** 파일을 수정할 수 있습니다.

```
#!/usr/bin/env stap

%{
#include <linux/in.h>
#include <linux/ip.h>
%}
```



```

/* according to [1], RSI contains 'type' and RDX
 * contains 'protocol'.
 * [1] https://github.com/torvalds/linux/blob/master/arch/x86/entry/entry_64.S#L79
 */

function mptcpify () %{
 if (CONTEXT->kregs->si == SOCK_STREAM &&
 (CONTEXT->kregs->dx == IPPROTO_TCP ||
 CONTEXT->kregs->dx == 0)) {
 CONTEXT->kregs->dx = IPPROTO_MPTCP;
 STAP_RETVALUE = 1;
 } else {
 STAP_RETVALUE = 0;
 }
}%

probe kernel.function("__sys_socket") {
 cur_proc = execname()
 if ((cur_proc == @1) && (mptcpify() == 1)) {
 printf("command %16s mptcpified\n", cur_proc);
 }
}

```

6.

대체 방법을 선택하는 경우 **TCP** 대신 **iperf3** 툴을 강제로 사용할 수 있습니다. 이렇게 하려면 다음 명령을 입력합니다.

```
stap -vg mptcp-app.stap iperf3
```

7.

**MP tcp-app.stap** 스크립트가 커널 프로브를 설치한 후 커널 **dmesg** 출력에 다음 경고가 나타납니다.

```

dmesg
...
[1752.694072] Kprobes globally unoptimized
[1752.730147] stap_1ade3b3356f3e68765322e26dec00c3d_1476: module_layout:
kernel tainted.
[1752.732162] Disabling lock debugging due to kernel taint
[1752.733468] stap_1ade3b3356f3e68765322e26dec00c3d_1476: loading out-of-tree
module taints kernel.
[1752.737219] stap_1ade3b3356f3e68765322e26dec00c3d_1476: module verification
failed: signature and/or required key missing - tainting kernel
[1752.737219] stap_1ade3b3356f3e68765322e26dec00c3d_1476 (mptcp-app.stap):
systemtap: 4.5/0.185, base: ffffffff0550000, memory:
224data/32text/57ctx/65638net/367alloc kb, probes: 1

```

8.

**iperf3** 서버를 시작합니다.

```
iperf3 -s
```

**Server listening on 5201**

9.

클라이언트를 서버에 연결합니다.

```
iperf3 -c 127.0.0.1 -t 3
```

10.

연결이 설정된 후 **ss** 출력을 확인하여 하위 흐름별 상태를 확인합니다.

```
ss -nti '(dport :5201)'
```

```
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
ESTAB 0 0 127.0.0.1:41842 127.0.0.1:5201
cubic wscale:7,7 rto:205 rtt:4.455/8.878 ato:40 mss:21888 pmtu:65535 rcvmss:536
advms:65483 cwnd:10 bytes_sent:141 bytes_acked:142 bytes_received:4 segs_out:8
segs_in:7 data_segs_out:3 data_segs_in:3 send 393050505bps lastsnd:2813
lastrcv:2772 lastack:2772 pacing_rate 785946640bps delivery_rate 10944000000bps
delivered:4 busy:41ms rcv_space:43690 rcv_ssthresh:43690 minrtt:0.008 tcp-ulp-
mptcp flags:Mmec token:0000(id:0)/2ff053ec(id:0) seq:3e2cbea12d7673d4 sfseq:3
ssnoff:ad3d00f4 maplen:2
```

11.

**nstat MPTcp\*** 명령을 사용하여 **MPTCP** 카운터를 확인합니다.

```
nstat MPTcp*
```

```
#kernel
MPTcpExtMPCapableSYNRX 2 0.0
MPTcpExtMPCapableSYNTAX 2 0.0
MPTcpExtMPCapableSYNACKRX 2 0.0
MPTcpExtMPCapableACKRX 2 0.0
```

추가 리소스

- [RHEL 시스템용 debuginfo 패키지를 다운로드하거나 설치하려면 어떻게 해야 하나요?](#)
- [TCP\(7\) 도움말 페이지](#)
- [mptcpize\(8\) 도움말 페이지](#)

### 32.3. IPROUTE2를 사용하여 MPTCP 애플리케이션에 대한 여러 경로 구성 및 활성화

각 **MPTCP** 연결은 일반 **TCP**와 유사한 단일 하위 흐름을 사용합니다. **MPTCP** 이점을 활용하려면 각 **MPTCP** 연결에 대한 최대 하위 흐름 수에 대한 더 높은 제한을 지정하고 해당 하위 흐름을 생성하도록 추가 엔드포인트를 구성합니다.

**MPTCP**는 동일한 소켓에 대해 아직 혼합 **IPv6** 및 **IPv4** 엔드포인트를 지원하지 않습니다. 동일한 주소 제품군에 속하는 엔드포인트를 사용합니다.

#### 사전 요구 사항

- **iperf3** 패키지가 설치됨
- 서버 네트워크 인터페이스 설정:
  - **enp4s0: 192.0.2.1/24**
  - **enp1s0: 198.51.100.1/24**
- 클라이언트 네트워크 인터페이스 설정:
  - **enp4s0f0: 192.0.2.2/24**
  - **enp4s0f1: 198.51.100.2/24**

#### 절차

1. 서버에서 연결당 하위 흐름 제한을 1로 설정합니다.

```
ip mptcp limits set subflow 1
```

초기 항목을 제외하고 각 연결에 보유할 수 있는 최대 추가 하위 흐름 수를 설정합니다.

2. 클라이언트의 연결당 및 추가 하위 흐름 제한을 1로 설정합니다.

```
ip mptcp limits set subflow 1 add_addr_accepted 1
```

3.

**IP 주소 198.51.100.1 을 서버의 새 MPTCP 엔드포인트로 추가합니다.**

```
ip mptcp endpoint add 198.51.100.1 dev enp1s0 signal
```

중요

플래그에 대해 다음 값을 하위 흐름, 백업, 신호로 설정할 수 있습니다. 플래그를 로 설정합니다.

- **3방향 핸드셰이크가 완료된 후 ADD\_ADDR 패킷을 보냅니다.**
- **subflow 클라이언트가 MP\_JOIN SYN 을 보냅니다.**
- **백업, 끝점을 백업 주소로 설정**

4.

**iperf3 서버를 시작합니다.**

```
iperf3 -s
```

**Server listening on 5201**

5.

**클라이언트를 서버에 연결합니다.**

```
iperf3 -c 192.0.2.1 -t 3
```

## 검증 단계

1.

**연결이 설정되었는지 확인합니다.**

```
ss -nti '(sport :5201)'
```

2.

**연결 및 IP 주소 제한을 확인합니다.**

```
ip mptcp limit show
```

3.

새로 추가된 끝점을 확인합니다.

```
ip mptcp endpoint show
```

4.

서버에서 **nstat MPTcp\*** 명령을 사용하여 **MPTCP** 카운터를 확인합니다.

```
nstat MPTcp*
```

```
#kernel
```

|                        |   |     |
|------------------------|---|-----|
| MPTcpExtMPCapableSYNRX | 2 | 0.0 |
| MPTcpExtMPCapableACKRX | 2 | 0.0 |
| MPTcpExtMPJoinSynRx    | 2 | 0.0 |
| MPTcpExtMPJoinAckRx    | 2 | 0.0 |
| MPTcpExtEchoAdd        | 2 | 0.0 |

추가 리소스

- [IP-mptcp\(8\) 도움말 페이지](#)
- [mptcpize\(8\) 도움말 페이지](#)

### 32.4. MPTCP 하위 흐름 모니터링

다중 경로 **TCP(MPTCP)** 소켓의 라이프 사이클은 복잡할 수 있습니다. 기본 **MPTCP** 소켓이 생성되고, **MPTCP** 경로가 검증되고, 하나 이상의 하위 흐름이 생성되고 결국 제거됩니다. 마지막으로 **MPTCP** 소켓이 종료됩니다.

**MPTCP** 프로토콜을 사용하면 **iproute** 패키지에서 제공하는 **ip** 유틸리티를 사용하여 소켓 및 하위 흐름 생성 및 삭제와 관련된 **MPTCP** 관련 이벤트를 모니터링할 수 있습니다. 이 유틸리티는 **netlink** 인터페이스를 사용하여 **MPTCP** 이벤트를 모니터링합니다.

다음 절차에서는 **MPTCP** 이벤트를 모니터링하는 방법을 설명합니다. 이를 위해 **MPTCP** 서버 애플리케이션을 시뮬레이션하고 클라이언트는 이 서비스에 연결됩니다. 이 예제의 관련 클라이언트는 다음 인터페이스 및 **IP** 주소를 사용합니다.

- 서버: 192.0.2.1

- **클라이언트(Ethernet 연결): 192.0.2.2**
- **클라이언트(WiFi 연결): 192.0.2.3**

이 예제를 단순화하기 위해 모든 인터페이스가 동일한 서브넷에 있습니다. 이는 필수 조건이 아닙니다. 그러나 라우팅이 올바르게 구성되어 클라이언트가 두 인터페이스를 통해 서버에 연결할 수 있어야 합니다.

#### 사전 요구 사항

- 이더넷과 WiFi가 있는 랩탑과 같은 두 개의 네트워크 인터페이스가 있는 **RHEL** 클라이언트
- 클라이언트는 두 인터페이스를 통해 서버에 연결할 수 있습니다.
- **RHEL** 서버
- 클라이언트 및 서버 모두 **RHEL 8.6** 이상을 실행합니다.

#### 절차

1. 클라이언트와 서버 둘 다에서 추가 하위 흐름 제한을 1로 설정합니다.

```
ip mptcp limits set add_addr_accepted 0 subflows 1
```

2. 서버에서 **MPTCP** 서버 애플리케이션을 시뮬레이션하려면 **TCP** 소켓 대신 **MPTCP** 소켓을 적용하여 수신 대기 모드에서 **netcat (nc)**을 시작합니다.

```
nc -l -k -p 12345
```

**-k** 옵션을 사용하면 첫 번째 수락된 연결 후 **nc**가 리스너를 닫지 않습니다. 이는 하위 흐름의 모니터링을 입증하는 데 필요합니다.

3. 클라이언트에서 다음을 수행합니다.

a.

가장 낮은 메트릭을 사용하여 인터페이스를 식별합니다.

```
ip -4 route
192.0.2.0/24 dev enp1s0 proto kernel scope link src 192.0.2.2 metric 100
192.0.2.0/24 dev wlp1s0 proto kernel scope link src 192.0.2.3 metric 600
```

**enp1s0** 인터페이스에는 **wlp1s0** 보다 낮은 메트릭이 있습니다. 따라서 **RHEL**은 기본적으로 **enp1s0** 을 사용합니다.

b.

첫 번째 터미널에서 모니터링을 시작합니다.

```
ip mptcp monitor
```

c.

두 번째 터미널에서 서버에 대한 **MPTCP** 연결을 시작합니다.

```
nc 192.0.2.1 12345
```

**RHEL**은 **enp1s0** 인터페이스와 연결된 **IP** 주소를 이 연결의 소스로 사용합니다.

모니터링 터미널에서 'ip mptcp 모니터' 명령이 이제 로그를 기록합니다.

```
[CREATED] token=63c070d2 remid=0 locid=0 saddr4=192.0.2.2
daddr4=192.0.2.1 sport=36444 dport=12345
```

토큰은 **MPTCP** 소켓을 고유 **ID**로 식별하며 나중에 동일한 소켓에서 **MPTCP** 이벤트의 상관 관계를 파악할 수 있습니다.

d.

서버에 대한 **nc** 연결이 실행되는 터미널에서 **Enter** 키를 누릅니다. 이 첫 번째 데이터 패킷은 연결을 완전히 설정합니다. 데이터가 전송되지 않으면 연결이 설정되지 않습니다.

모니터링 터미널에서 **ip MPtcp** 모니터 가 이제 로그입니다.

```
[ESTABLISHED] token=63c070d2 remid=0 locid=0 saddr4=192.0.2.2
daddr4=192.0.2.1 sport=36444 dport=12345
```

e.

선택 사항: 서버의 포트 **12345** 에 대한 연결을 표시합니다.

```
ss -taunp | grep ":12345"
tcp ESTAB 0 0 192.0.2.2:36444 192.0.2.1:12345
```

이 시점에서 서버에 대한 하나의 연결만 설정됩니다.

f.

세 번째 터미널에서 다른 끝점을 생성합니다.

```
ip mptcp endpoint add dev wlp1s0 192.0.2.3 subflow
```

이 명령은 이 명령에서 클라이언트의 **WiFi** 인터페이스 이름 및 **IP** 주소를 설정합니다.

모니터링 터미널에서 **ip MPtcp** 모니터 가 이제 로그인합니다.

```
[SF_ESTABLISHED] token=63c070d2 remid=0 locid=2 saddr4=192.0.2.3
daddr4=192.0.2.1 sport=53345 dport=12345 backup=0 ifindex=3
```

**locid** 필드는 새 하위 흐름의 로컬 주소 **ID**를 표시하고 연결에 **NAT**(네트워크 주소 변환)를 사용하는 경우에도 이 하위 흐름을 식별합니다. **saddr4** 필드는 **ip mptcp endpoint add** 명령의 엔드포인트의 **IP** 주소와 일치합니다.

g.

선택 사항: 서버의 포트 **12345** 에 대한 연결을 표시합니다.

```
ss -taunp | grep ":12345"
tcp ESTAB 0 0 192.0.2.2:36444 192.0.2.1:12345
tcp ESTAB 0 0 192.0.2.3%wlp1s0:53345 192.0.2.1:12345
```

이제 명령으로 다음 두 개의 연결을 표시합니다.

•

소스 주소 **192.0.2.2** 와의 연결은 이전에 설정한 첫 번째 **MPTCP** 하위 흐름에 해당합니다.

•

소스 주소 **192.0.2.3** 이 있는 **wlp1s0** 인터페이스를 통한 하위 흐름의 연결입니다.



h.

세 번째 터미널에서 끝점을 삭제합니다.

```
ip mptcp endpoint delete id 2
```

**ip MP tcp** 모니터 출력의 **locid** 필드의 **ID**를 사용하거나 **ip MP tcp endpoint show** 명령을 사용하여 엔드포인트 **ID**를 검색합니다.

모니터링 터미널에서 **ip MPtcp** 모니터 가 이제 로그입니다.

```
[SF_CLOSED] token=63c070d2 remid=0 locid=2 saddr4=192.0.2.3
daddr4=192.0.2.1 sport=53345 dport=12345 backup=0 ifindex=3
```

i.

**nc** 클라이언트가 있는 첫 번째 터미널에서 **Ctrl+C** 세션을 종료합니다.

모니터링 터미널에서 **ip MPtcp** 모니터 가 이제 로그입니다.

```
[CLOSED] token=63c070d2
```

추가 리소스

- [IP-mptcp\(1\) 매뉴얼 페이지](#)
- [NetworkManager가 여러 기본 게이트웨이를 관리하는 방법](#)

### 32.5. 커널에서 MULTIPATH TCP 비활성화

다음 절차에서는 커널에서 **MPTCP** 옵션을 비활성화하는 방법을 설명합니다.

절차

- **MP tcp.enabled** 옵션을 비활성화합니다.

```
echo "net.mptcp.enabled=0" > /etc/sysctl.d/90-enable-MPTCP.conf
sysctl -p /etc/sysctl.d/90-enable-MPTCP.conf
```

검증 단계

- 

커널에서 **MP tcp.enabled** 가 비활성화되었는지 확인합니다.

```
sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 0
```

### 33장. DNS 서버 순서 구성

대부분의 애플리케이션은 **glibc** 라이브러리의 **getaddrinfo()** 함수를 사용하여 **DNS** 요청을 해결합니다. 기본적으로 **glibc** 는 모든 **DNS** 요청을 **/etc/resolv.conf** 파일에 지정된 첫 번째 **DNS** 서버로 전송합니다. 이 서버가 응답하지 않는 경우 **Red Hat Enterprise Linux**는 이 파일에서 다음 서버를 사용합니다.

이 섹션에서는 **DNS** 서버 순서를 사용자 지정하는 방법에 대해 설명합니다.

#### 33.1. NETWORKMANAGER가 /ETC/RESOLV.CONF에서 DNS 서버를 주문하는 방법

**NetworkManager**는 다음 규칙에 따라 **/etc/resolv.conf** 파일에서 **DNS** 서버를 주문합니다.

- 하나의 연결 프로파일만 있는 경우 **NetworkManager**는 해당 연결에 지정된 **IPv4** 및 **IPv6** **DNS** 서버의 순서를 사용합니다.
- 여러 연결 프로파일이 활성화되면 **NetworkManager**는 **DNS** 우선 순위 값을 기반으로 **DNS** 서버를 주문합니다. **DNS** 우선순위를 설정하는 경우 **NetworkManager**의 동작은 **dns** 매개변수에 설정된 값에 따라 달라집니다. **/etc/NetworkManager/NetworkManager.conf** 파일의 **[main]** 섹션에서 이 매개변수를 설정할 수 있습니다.

○

**DNS=default** 또는 **dns** 매개변수가 설정되지 않은 경우:

**NetworkManager**는 각 연결의 **ipv4.dns-priority** 및 **ipv6.dns-priority** 매개 변수를 기반으로 다른 연결에서 **DNS** 서버를 정렬합니다.

값을 설정하지 않거나 **ipv4.dns-priority** 및 **ipv6.dns-priority** 를 0 으로 설정하면 **NetworkManager**는 글로벌 기본값을 사용합니다. **DNS 우선 순위 매개 변수의 기본값** 을 참조하십시오.

○

**dns=dnsmasq** or **dns=systemd-resolved**:

이러한 설정 중 하나를 사용하는 경우 **NetworkManager**는 **dnsmasq** 의 **127.0.0.1** 또는 **127.0.0.53** 을 **/etc/resolv.conf** 파일에서 이름 서버로 설정합니다.

**dnsmasq** 및 **systemd** 확인 서비스는 **NetworkManager** 연결에 설정된 검색 도메인에 대한 쿼리를 해당 연결에 지정된 **DNS** 서버로 전달하고 쿼리를 다른 도메인에 기본 경로와의

연결로 전달합니다. 여러 연결에 동일한 검색 도메인이 설정되면 **dnsmasq** 및 **systemd** 에서 이 도메인에 대한 **systemd** 확인 쿼리는 우선 순위가 가장 낮은 연결의 **DNS** 서버에 설정됩니다.

### DNS 우선순위 매개변수의 기본값

**NetworkManager**는 연결에 다음과 같은 기본값을 사용합니다.

- 50 VPN 연결의 경우
- 기타 연결의 경우 100

유효한 **DNS** 우선순위 값:

글로벌 기본값 및 연결별 **ipv4.dns-priority** 및 **ipv6.dns-priority** 매개변수를 -2147483647 과 2147483647 사이의 값으로 설정할 수 있습니다.

- 더 낮은 값은 우선 순위가 높습니다.
- 음수 값은 더 큰 값이 있는 다른 구성을 제외하고 특별한 영향을 미칩니다. 예를 들어 우선 순위 값이 음수인 연결이 하나 이상 있으면 **NetworkManager**는 우선 순위가 가장 낮은 연결 프로필에 지정된 **DNS** 서버만 사용합니다.
- 여러 연결의 우선 순위가 동일한 경우 **NetworkManager**는 다음 순서로 **DNS**에 우선순위를 부여합니다.
  - a. VPN 연결
  - b. 활성 기본 경로와 연결. 활성 기본 경로는 가장 낮은 메트릭이 있는 기본 경로입니다.

### 추가 리소스

- **nm-settings(5)** 도움말 페이지의 **ipv4** 및 **ipv6** 섹션에 있는 **dns-priority** 매개변수 설명
- [다른 도메인에 다른 DNS 서버 사용](#)

### 33.2. NETWORKMANAGER 전체 기본 DNS 서버 우선순위 값 설정

NetworkManager는 연결에 다음과 같은 DNS 우선 순위 기본값을 사용합니다.

- 50 VPN 연결의 경우
- 기타 연결의 경우 100

이 섹션에서는 이러한 시스템 전체 기본값을 IPv4 및 IPv6 연결의 사용자 지정 기본값으로 재정의하는 방법에 대해 설명합니다.

#### 절차

1. `/etc/NetworkManager/NetworkManager.conf` 파일을 편집합니다.

- a. `[connection]` 섹션이 없는 경우 해당 섹션을 추가합니다.

```
[connection]
```

- b. 사용자 정의 기본값을 `[connection]` 섹션에 추가합니다. 예를 들어 IPv4 및 IPv6의 새 기본값을 모두 200 으로 설정하려면 다음을 추가합니다.

```
ipv4.dns-priority=200
ipv6.dns-priority=200
```

매개 변수를 -2147483647 과 2147483647 사이의 값으로 설정할 수 있습니다. 매개 변수를 0 으로 설정하면 기본 제공 기본값( VPN 연결의 경우 50 및 기타 연결의 경우 100 )이 활성화됩니다.

2. NetworkManager 서비스를 다시 로드합니다.

```
systemctl reload NetworkManager
```

추가 리소스

•

**NetworkManager.conf(5) 매뉴얼 페이지의 연결 섹션****33.3. NETWORKMANAGER 연결의 DNS 우선 순위 설정**

이 섹션에서는 **NetworkManager**가 **/etc/resolv.conf** 파일을 만들거나 업데이트할 때 **DNS** 서버 순서를 정의하는 방법을 설명합니다.

**DNS** 우선순위 설정은 서로 다른 **DNS** 서버가 구성된 여러 연결이 있는 경우에만 의미가 있습니다. 여러 **DNS** 서버가 구성된 연결만 있는 경우 연결 프로필에서 **DNS** 서버를 선호하는 순서로 수동으로 설정합니다.

**사전 요구 사항**

•

시스템에는 여러 개의 **NetworkManager** 연결이 구성되어 있습니다.

•

시스템에는 **/etc/NetworkManager/NetworkManager.conf** 파일에 **dns** 매개 변수가 설정되지 않았거나 매개 변수가 기본값으로 설정됩니다.

**절차**

1.

선택적으로 사용 가능한 연결을 표시합니다.

```
nmcli connection show
NAME UUID TYPE DEVICE
Example_con_1 d17ee488-4665-4de2-b28a-48befab0cd43 ethernet enp1s0
Example_con_2 916e4f67-7145-3ffa-9f7b-e7cada8f6bf7 ethernet enp7s0
...
```

2.

**ipv4.dns-priority** 및 **ipv6.dns-priority** 매개 변수를 설정합니다. 예를 들어 **Example\_con\_1** 연결에 대해 두 매개 변수를 모두 **10** 으로 설정하려면 다음을 수행합니다.

```
nmcli connection modify Example_con_1 ipv4.dns-priority 10 ipv6.dns-priority 10
```

3.

선택적으로 다른 연결에 대해 이전 단계를 반복합니다.

4.

업데이트한 연결을 다시 활성화합니다.

■

---

```
nmcli connection up Example_con_1
```

검증 단계

- */etc/resolv.conf* 파일의 내용을 표시하여 **DNS** 서버 순서가 올바른지 확인합니다.

```
cat /etc/resolv.conf
```

### 34장. IFCFG 파일로 IP 네트워킹 구성

이 섹션에서는 **ifcfg** 파일을 편집하여 네트워크 인터페이스를 수동으로 구성하는 방법을 설명합니다.

#### 중요

**NetworkManager**는 키 파일 형식으로 저장된 프로필을 지원합니다. 그러나 **NetworkManager API**를 사용하여 프로필을 생성하거나 업데이트할 때 **NetworkManager**는 기본적으로 **ifcfg** 형식을 사용합니다.

향후 주요 **RHEL** 릴리스에서는 키 파일 형식이 기본값입니다. 구성 파일을 수동으로 생성하고 관리하려면 키 파일 형식을 사용하는 것이 좋습니다. 자세한 내용은 [키 파일 형식으로 NetworkManager 프로필 수동 생성](#)을 참조하십시오.

인터페이스 구성(**ifcfg**) 파일은 개별 네트워크 장치에 대한 소프트웨어 인터페이스를 제어합니다. 시스템이 부팅되면 이러한 파일을 사용하여 가져올 인터페이스와 구성 방법을 결정합니다. 이러한 파일의 이름은 일반적으로 **ifcfg-name**입니다. 여기서 접미사 이름은 구성 파일이 제어하는 장치의 이름을 나타냅니다. 관례적으로 **ifcfg** 파일의 접미사는 구성 파일 자체의 **DEVICE** 지시문에서 제공한 문자열과 동일합니다.

#### 34.1. IFCFG 파일을 사용하여 정적 네트워크 설정으로 인터페이스 구성

다음 절차에서는 **ifcfg** 파일을 사용하여 네트워크 인터페이스를 구성하는 방법을 설명합니다.

#### 절차

- **ifcfg** 파일을 사용하여 정적 네트워크 설정으로 인터페이스를 구성하려면 이름이 **enp1s0**인 인터페이스의 경우 포함된 **/etc/sysconfig/network-scripts/** 디렉터리에 **ifcfg-enp1s0**이라는 파일을 만듭니다.

○

**IPv4** 구성의 경우:

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
GATEWAY=10.0.1.1
```



○

*IPv6 구성의 경우:*

```

DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=2001:db8:1::2/64

```

추가 리소스

●

*기본 네트워크 설정 테스트*

●

*nm-settings-ifcfg-rh(5) man page*

### 34.2. IFCFG 파일을 사용하여 동적 네트워크 설정으로 인터페이스 구성

다음 절차에서는 *ifcfg* 파일을 사용하여 동적 네트워크 설정으로 네트워크 인터페이스를 구성하는 방법을 설명합니다.

절차

1.

*ifcfg* 파일을 사용하여 동적 네트워크 설정으로 **em1** 이라는 인터페이스를 구성하려면 다음을 포함하는 */etc/sysconfig/network-scripts/* 디렉터리에 *ifcfg-em1* 이라는 이름의 파일을 만듭니다.

```

DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes

```

2.

전송할 인터페이스를 구성하려면 다음을 수행합니다.

●

DHCP 서버에 다른 호스트 이름, *ifcfg* 파일에 다음 행을 추가합니다.

```
DHCP_HOSTNAME=hostname
```

●

DHCP 서버에 다른 FQDN(정규화된 도메인 이름)을 *ifcfg* 파일에 추가합니다.

```
DHCP_FQDN=fully.qualified.domain.name
```



## 참고

이러한 설정 중 하나만 사용할 수 있습니다. **DHCP\_HOSTNAME** 및 **DHCP\_FQDN** 을 모두 지정하는 경우 **DHCP\_FQDN** 만 사용됩니다.

3.

특정 **DNS** 서버를 사용하도록 인터페이스를 구성하려면 **ifcfg** 파일에 다음 행을 추가합니다.

```
PEERDNS=no
DNS1=ip-address
DNS2=ip-address
```

여기서 **ip-address** 는 **DNS** 서버의 주소입니다. 이로 인해 네트워크 서비스가 지정된 **DNS** 서버로 **/etc/resolv.conf** 를 업데이트합니다. 하나의 **DNS** 서버 주소만 필요하며 다른 하나는 선택 사항입니다.

### 34.3. IFCFG 파일을 사용하여 시스템 전체 및 개인 연결 프로필 관리

이 절차에서는 시스템 전체 및 개인 연결 프로필을 관리하도록 **ifcfg** 파일을 구성하는 방법을 설명합니다.

#### 절차

권한은 **ifcfg** 파일의 **USERS** 지시문에 해당합니다. **USERS** 지시문이 없으면 모든 사용자가 네트워크 프로필을 사용할 수 있습니다.

•

예를 들어 다음 행을 사용하여 **ifcfg** 파일을 수정합니다. 그러면 나열된 사용자만 연결을 사용할 수 있습니다.

```
USERS="joe bob alice"
```

### 35장. 특정 연결에 대해 NETWORKMANAGER를 사용하여 IPV6 비활성화

이 섹션에서는 **NetworkManager**를 사용하여 네트워크 인터페이스를 관리하는 시스템에서 **IPv6** 프로토콜을 비활성화하는 방법을 설명합니다. **IPv6** 를 비활성화하면 **NetworkManager**가 커널에 해당 **sysctl** 값을 자동으로 설정합니다.



#### 참고

커널 튜닝 가능 항목 또는 커널 부팅 매개 변수를 사용하여 **IPv6**를 비활성화하는 경우 시스템 구성에 추가 고려 사항을 지정해야 합니다. 자세한 내용은 [RHEL에서 IPv6 프로토콜을 비활성화하거나 활성화하는 방법](#) 문서를 참조하십시오.

#### 사전 요구 사항



시스템은 **NetworkManager**를 사용하여 **Red Hat Enterprise Linux**의 기본 네트워크 인터페이스인 네트워크 인터페이스를 관리합니다.

#### 35.1. NMCLI를 사용하여 연결에서 IPV6 비활성화

다음 절차에서는 **nmcli** 유틸리티를 사용하여 **IPv6** 프로토콜을 비활성화하는 방법을 설명합니다.

#### 절차

1. 선택적으로 네트워크 연결 목록을 표시합니다.

```
nmcli connection show
NAME UUID TYPE DEVICE
Example 7a7e0151-9c18-4e6f-89ee-65bb2d64d365 ethernet enp1s0
...
```

2. 연결의 **ipv6.method** 매개 변수를 **disabled** 로 설정합니다.

```
nmcli connection modify Example ipv6.method "disabled"
```

3. 네트워크 연결을 다시 시작합니다.

```
nmcli connection up Example
```

## 검증 단계

1.

장치의 IP 설정을 표시하려면 **ip address show** 명령을 입력합니다.

```
ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
 link/ether 52:54:00:6b:74:be brd ff:ff:ff:ff:ff:ff
 inet 192.0.2.1/24 brd 192.10.2.255 scope global noprefixroute enp1s0
 valid_lft forever preferred_lft forever
```

**inet6** 항목이 표시되지 않으면 장치에서 **IPv6** 가 비활성화됩니다.

2.

**/proc/sys/net/ipv6/conf/enp1s0/disable\_ipv6** 파일에 이제 **1** 값이 포함되어 있는지 확인합니다.

```
cat /proc/sys/net/ipv6/conf/enp1s0/disable_ipv6
1
```

값 **1** 은 **IPv6** 가 장치에 대해 비활성화되어 있음을 의미합니다.

## 36장. /ETC/RESOLV.CONF 파일 수동 구성

기본적으로 RHEL(Red Hat Enterprise Linux) 8의 NetworkManager는 활성 NetworkManager 연결 프로필의 DNS 설정으로 /etc/resolv.conf 파일을 동적으로 업데이트합니다. 이 섹션에서는 /etc/resolv.conf 에서 DNS 설정을 수동으로 구성하기 위해 이 기능을 비활성화하는 방법에 대한 다양한 옵션에 대해 설명합니다.

### 36.1. NETWORKMANAGER 구성에서 DNS 처리 비활성화

이 섹션에서는 /etc/resolv.conf 파일을 수동으로 구성하기 위해 NetworkManager 구성에서 DNS 처리를 비활성화하는 방법을 설명합니다.

#### 절차

1. 루트 사용자로 텍스트 편집기를 사용하여 다음 콘텐츠를 사용하여 /etc/NetworkManager/conf.d/90-dns-none.conf 파일을 생성합니다.

```
[main]
dns=none
```

2. NetworkManager 서비스를 다시 로드합니다.

```
systemctl reload NetworkManager
```



#### 참고

서비스를 다시 로드한 후 NetworkManager는 더 이상 /etc/resolv.conf 파일을 업데이트하지 않습니다. 그러나 파일의 마지막 내용은 보존됩니다.

3. 선택적으로 혼동을 피하기 위해 /etc/resolv.conf 에서 NetworkManager 주석에 의해 Generated 를 제거합니다.

#### 검증 단계

1. /etc/resolv.conf 파일을 편집하고 설정을 수동으로 업데이트합니다.
2. NetworkManager 서비스를 다시 로드합니다.

```
systemctl reload NetworkManager
```

3. `/etc/resolv.conf` 파일을 표시합니다.

```
cat /etc/resolv.conf
```

DNS 처리를 비활성화한 경우 **NetworkManager**에서 수동으로 구성된 설정을 재정의하지 않았습니다.

#### 추가 리소스

- **NetworkManager.conf(5)** 도움말 페이지의 **dns** 매개변수 설명

### 36.2. DNS 설정을 수동으로 구성하려면 `/ETC/RESOLV.CONF`를 심볼릭 링크로 교체

`/etc/resolv.conf`가 심볼릭 링크인 경우 **NetworkManager**는 DNS 구성을 자동으로 업데이트하지 않습니다. 이 섹션에서는 `/etc/resolv.conf`를 DNS 구성으로 대체 파일에 대한 심볼릭 링크로 교체하는 방법을 설명합니다.

#### 사전 요구 사항

- **rc-manager** 옵션은 파일로 설정되지 않습니다. 확인하려면 **NetworkManager --print-config** 명령을 사용합니다.

#### 절차

1. `/etc/resolv.conf.manually-configured`와 같은 파일을 만들고 환경에 대한 DNS 구성을 추가합니다. 원래 `/etc/resolv.conf`와 동일한 매개변수 및 구문을 사용합니다.

2. `/etc/resolv.conf` 파일을 제거합니다.

```
rm /etc/resolv.conf
```

3. `/etc/resolv.conf.manually-configured`를 참조하는 `/etc/resolv.conf`라는 심볼릭 링크를 만듭니다.

```
ln -s /etc/resolv.conf.manually-configured /etc/resolv.conf
```

### 추가 리소스

- ***resolv.conf(5) man page***
- ***NetworkManager.conf(5) 도움말 페이지의 rc-manager 매개 변수***

### 37장. RX 링 버퍼 모니터링 및 튜닝

수신(RX) 링 버퍼는 장치 드라이버와 NIC(네트워크 인터페이스 카드) 간에 공유 버퍼이며 장치 드라이버가 처리할 수 있을 때까지 들어오는 패킷을 저장합니다.

패킷 드롭 속도로 인해 애플리케이션이 보고되는 경우 이더넷 장치 RX 링 버퍼의 크기를 늘릴 수 있습니다.

- 데이터 손실
- 클러스터 펜스,
- 성능 저하,
- 시간 제한
- 백업 실패.

이 섹션에서는 삭제된 패킷 수를 식별하고 RX 링 버퍼를 늘려 높은 패킷 드롭 속도를 줄이는 방법을 설명합니다.

#### 37.1. 삭제된 패킷 수 표시

**ethtool** 유틸리티를 사용하면 관리자가 네트워크 드라이버 설정을 쿼리, 구성 또는 제어할 수 있습니다.

RX 링 버퍼가 소진되면 **ethtool -S interface\_name**의 출력에서 "차단" 또는 "drop"과 같은 카운터가 증가합니다. 폐기된 패킷은 커널이 패킷을 처리할 수 있는 것보다 사용 가능한 버퍼가 더 빨리 채워져 있음을 나타냅니다.

이 절차에서는 **ethtool**을 사용하여 드롭 카운터를 표시하는 방법을 설명합니다.

절차



- **enp1s0** 인터페이스에 대한 드롭 카운터를 보려면 다음을 입력합니다.

```
$ ethtool -S enp1s0
```

### 37.2. RX 링 버퍼를 늘려 높은 패킷 드롭 속도를 줄입니다.

**ethtool** 유틸리티는 높은 패킷 드롭 속도를 줄이기 위해 **RX** 버퍼를 늘리는 데 도움이 됩니다.

#### 절차

1. 최대 **RX** 링 버퍼 크기를 보려면 다음을 수행합니다.

```
ethtool -g enp1s0
Ring parameters for enp1s0:
Pre-set maximums:
RX: 4080
RX Mini: 0
RX Jumbo: 16320
TX: 255
Current hardware settings:
RX: 255
RX Mini: 0
RX Jumbo: 0
TX: 255
```

2. 설정 최대 섹션의 값이 현재 하드웨어 설정 섹션에서보다 큰 경우 **RX** 링 버퍼를 늘립니다.

- **enp1s0** 장치의 **RX** 링 버퍼를 **4080** 으로 임시로 변경하려면 다음을 입력합니다.

```
ethtool -G enp1s0 rx 4080
```

- **RX** 링 버퍼를 영구적으로 변경하려면 **NetworkManager** 디스패치 스크립트를 생성합니다.

자세한 내용은 [How to make NIC ethtool settings persistent](#)(부팅 시 자동으로 적용) 문서를 작성하고 디스패치 스크립트를 생성하는 방법을 참조하십시오.



### 중요

네트워크 인터페이스 카드에서 사용하는 드라이버에 따라 링 버퍼에서 변경하면 곧 네트워크 연결을 중단할 수 있습니다.

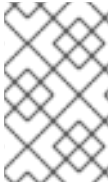
### 추가 리소스

- [ifconfig 및 ip 명령이 RHEL7에서 패킷 삭제를 보고합니다.](#)
- [0.05% 패킷 드롭율에 대해 우려해야 합니까?](#)
- [ethtool\(8\) 도움말 페이지](#)

## 38장. 802.3 링크 설정 구성

### 38.1. 자동 협상 이해

자동 협상은 **IEEE 802.3u Fast Ethernet** 프로토콜의 기능입니다. 장치 포트를 대상으로 하여 링크를 통해 정보 교환을 위한 속도, 중복 모드 및 흐름 제어의 최적의 성능을 제공합니다. 자동 협상 프로토콜을 사용하면 이더넷을 통해 데이터 전송의 최적의 성능을 얻을 수 있습니다.



#### 참고

자동 협상의 최대 성능을 활용하려면 링크의 양쪽에서 동일한 구성을 사용합니다.

### 38.2. NMCLI 유틸리티를 사용하여 802.3 링크 설정 구성

이더넷 연결의 **802.3** 링크 설정을 구성하려면 다음 구성 매개변수를 수정합니다.

- **802-3-ethernet.auto-negotiate**
- **802-3-ethernet.speed**
- **802-3-ethernet.duplex**

#### 절차

1. 연결의 현재 설정을 표시합니다.

```
nmcli connection show Example-connection
...
802-3-ethernet.speed: 0
802-3-ethernet.duplex: --
802-3-ethernet.auto-negotiate: no
...
```

문제가 있는 경우 매개변수를 재설정해야 하는 경우 이러한 값을 사용할 수 있습니다.

2.

**speed** 및 **duplex** 링크 설정을 설정합니다.

```
nmcli connection modify Example-connection 802-3-ethernet.auto-negotiate no 802-3-ethernet.speed 10000 802-3-ethernet.duplex full
```

이 명령은 자동 협상을 비활성화하고 연결 속도를 **10000 Mbit** 전체 **duplex**로 설정합니다.

3.

연결을 다시 활성화합니다.

```
nmcli connection up Example-connection
```

## 검증

•

**ethtool** 유틸리티를 사용하여 이더넷 인터페이스 **enp1s0**의 값을 확인합니다.

```
ethtool enp1s0
```

Settings for enp1s0:

```
...
Advertised auto-negotiation: No
...
Speed: 10000Mb/s
Duplex: Full
Auto-negotiation: off
...
Link detected: yes
```

## 추가 리소스

•

네트워크 인터페이스 속도는 **100Mbps**이고 **1Gbps**여야 합니다.

•

**nm-settings(5)** 도움말 페이지

## 39장. ETHTOOL 오프로드 기능 구성

네트워크 인터페이스 카드는 **TOE(TCP 오프로드 엔진)**를 사용하여 특정 작업을 네트워크 컨트롤러로 오프로드하여 네트워크 처리량을 향상시킬 수 있습니다.

이 섹션에서는 오프로드 기능을 설정하는 방법에 대해 설명합니다.

### 39.1. NETWORKMANAGER에서 지원하는 오프로드 기능

**NetworkManager**를 사용하여 다음 **ethtool** 오프로드 기능을 설정할 수 있습니다.

- **`ethtool.feature-esp-hw-offload`**
- **`ethtool.feature-esp-tx-csum-hw-offload`**
- **`ethtool.feature-fcoe-mtu`**
- **`ethtool.feature-gro`**
- **`ethtool.feature-gso`**
- **`ethtool.feature-highdma`**
- **`ethtool.feature-hw-tc-offload`**
- **`ethtool.feature-l2-fwd-offload`**
- **`ethtool.feature-loopback`**
- **`ethtool.feature-lro`**

- *ethtool.feature-macsec-hw-offload*
- *ethtool.feature-ntuple*
- *ethtool.feature-rx*
- *ethtool.feature-rx-all*
- *ethtool.feature-rx-fcs*
- *ethtool.feature-rx-gro-hw*
- *ethtool.feature-rx-gro-list*
- *ethtool.feature-rx-udp\_tunnel-port-offload*
- *ethtool.feature-rx-udp-gro-forwarding*
- *ethtool.feature-rx-vlan-filter*
- *ethtool.feature-rx-vlan-stag-filter*
- *ethtool.feature-rx-vlan-stag-hw-parse*
- *ethtool.feature-rxhash*
- *ethtool.feature-rxvlan*

- *ethtool.feature-sg*
- *ethtool.feature-tls-hw-record*
- *ethtool.feature-tls-hw-rx-offload*
- *ethtool.feature-tls-hw-tx-offload*
- *ethtool.feature-tso*
- *ethtool.feature-tx*
- *ethtool.feature-tx-checksum-fcoe-crc*
- *ethtool.feature-tx-checksum-ip-generic*
- *ethtool.feature-tx-checksum-ipv4*
- *ethtool.feature-tx-checksum-ipv6*
- *ethtool.feature-tx-checksum-sctp*
- *ethtool.feature-tx-esp-segmentation*
- *ethtool.feature-tx-fcoe-segmentation*
- *ethtool.feature-tx-gre-csum-segmentation*

- *ethtool.feature-tx-gre-segmentation*
- *ethtool.feature-tx-gso-list*
- *ethtool.feature-tx-gso-partial*
- *ethtool.feature-tx-gso-robust*
- *ethtool.feature-tx-ipxip4-segmentation*
- *ethtool.feature-tx-ipxip6-segmentation*
- *ethtool.feature-tx-nocache-copy*
- *ethtool.feature-tx-scatter-gather*
- *ethtool.feature-tx-scatter-gather-fraglist*
- *ethtool.feature-tx-sctp-segmentation*
- *ethtool.feature-tx-tcp-ecn-segmentation*
- *ethtool.feature-tx-tcp-mangleid-segmentation*
- *ethtool.feature-tx-tcp-segmentation*
- *ethtool.feature-tx-tcp6-segmentation*



- `ethtool.feature-tx-tunnel-remcsum-segmentation`
- `ethtool.feature-tx-udp-segmentation`
- `ethtool.feature-tx-udp_tnl-csum-segmentation`
- `ethtool.feature-tx-udp_tnl-segmentation`
- `ethtool.feature-tx-vlan-stag-hw-insert`
- `ethtool.feature-txvlan`

개별 오프로드 기능에 대한 자세한 내용은 **ethtool** 유틸리티 문서 및 커널 설명서를 참조하십시오.

### 39.2. NETWORKMANAGER를 사용하여 ETHTOOL 오프로드 기능 구성

이 섹션에서는 **NetworkManager**를 사용하여 **ethtool** 오프로드 기능을 활성화 및 비활성화하는 방법과 **NetworkManager** 연결 프로파일에서 기능의 설정을 제거하는 방법에 대해 설명합니다.

#### 절차

1.

예를 들어 **RX** 오프로드 기능을 활성화하고 **enp1s0** 연결 프로파일에서 **TX** 오프로드를 비활성화하려면 다음을 입력합니다.

```
nmcli con modify enp1s0 ethtool.feature-rx on ethtool.feature-tx off
```

이 명령은 **RX** 오프로드를 명시적으로 활성화하고 **TX** 오프로드를 비활성화합니다.

2.

이전에 활성화되었거나 비활성화한 오프로드 기능 설정을 제거하려면 **feature**의 매개변수를 무시 하도록 설정합니다. 예를 들어 **TX** 오프로드의 구성을 제거하려면 다음을 입력합니다.

```
nmcli con modify enp1s0 ethtool.feature-tx ignore
```

3.

네트워크 프로필을 다시 활성화합니다.

```
nmcli connection up enp1s0
```

## 검증 단계

•

ethtool -k 명령을 사용하여 네트워크 장치의 현재 오프로드 기능을 표시합니다.

```
ethtool -k network_device
```

## 추가 리소스

•

[NetworkManager에서 지원하는 오프로드 기능](#)

## 39.3. RHEL 시스템 역할을 사용하여 ETHTOOL 기능 설정

네트워크 RHEL 시스템 역할을 사용하여 NetworkManager 연결의 ethtool 기능을 구성할 수 있습니다.



## 중요

네트워크 RHEL 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 값과 일치하지 않는 경우 시스템 역할이 동일한 이름으로 기존 연결 프로필을 덮어씁니다. 따라서 IP 구성이 이미 있는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 enp1s0 연결 프로필을 생성하거나 업데이트합니다.

•

정적 IPv4 주소 - 198.51.100.20 에 /24 서브넷 마스크

•

정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크

•

IPv4 기본 게이트웨이 - 198.51.100.254

- **IPv6 기본 게이트웨이 - 2001:db8:1::fffe**
- **IPv4 DNS 서버 198.51.100.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**
- **ethtool 기능:**
  - **GRO(Generic receive offload): 비활성화**
  - **GSO(Generic segmentation offload): 활성화됨**
  - **TX SCTP(스트림 제어 전송 프로토콜) 분할: 비활성화**

#### 사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

#### 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.
 

**node.example.com**
2. 다음 콘텐츠를 사용하여 **~/configure-ethernet-device-with-ethtool-features.yml** 플레이북을 만듭니다.

```

- name: Configure an Ethernet connection with ethtool features
 hosts: node.example.com
 become: true
 tasks:
 - include_role:
 name: rhel-system-roles.network

 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 198.51.100.20/24
 - 2001:db8:1::1/64
 gateway4: 198.51.100.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 198.51.100.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
 ethtool:
 features:
 gro: "no"
 gso: "yes"
 tx_sctp_segmentation: "no"
 state: up

```

3.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/configure-ethernet-device-with-ethtool-features.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-with-ethtool-features.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사

---

용자로 관리 호스트에 연결합니다.

#### 추가 리소스

- ***/usr/share/ansible/roles/rhel-system-roles.network/README.md file***
- ***ansible-playbook(1) 도움말 페이지***

## 40장. ETHTOOL 통합 설정 구성

인터럽트 결합을 사용하면 시스템은 네트워크 패킷을 수집하고 여러 패킷에 대한 단일 인터럽트를 생성합니다. 이렇게 하면 하드웨어 인터럽트가 1개인 커널로 전송되는 데이터의 양이 증가하여 인터럽트 부하가 감소하고 처리량이 극대화됩니다.

이 섹션에서는 **ethtool** 통합 설정을 설정하는 다양한 옵션을 제공합니다.

### 40.1. NETWORKMANAGER에서 지원하는 통합 설정

**NetworkManager**를 사용하여 다음 **ethtool coalesce** 설정을 설정할 수 있습니다.

- **coalesce-adaptive-rx**
- **coalesce-adaptive-tx**
- **coalesce-pkt-rate-high**
- **coalesce-pkt-rate-low**
- **coalesce-rx-frames**
- **coalesce-rx-frames-high**
- **coalesce-rx-frames-irq**
- **coalesce-rx-frames-low**
- **coalesce-rx-usecs**

- *coalesce-rx-usecs-high*
- *coalesce-rx-usecs-irq*
- *coalesce-rx-usecs-low*
- *coalesce-sample-interval*
- *coalesce-stats-block-usecs*
- *coalesce-tx-frames*
- *coalesce-tx-frames-high*
- *coalesce-tx-frames-irq*
- *coalesce-tx-frames-low*
- *coalesce-tx-usecs*
- *coalesce-tx-usecs-high*
- *coalesce-tx-usecs-irq*
- *coalesce-tx-usecs-low*

## 40.2. NETWORKMANAGER를 사용하여 ETHTOOL 통합 설정 구성

이 섹션에서는 **NetworkManager** 연결 프로파일에서 설정을 제거하는 방법과 **NetworkManager** 연결 프

로필을 사용하여 **ethtool** 을 설정하는 방법에 대해 설명합니다.

#### 절차

1. 예를 들어 **enp1s0** 연결 프로필에서 **128** 로 지연할 수신된 패킷의 최대 수를 설정하려면 다음을 입력합니다.

```
nmcli connection modify enp1s0 ethtool.coalesce-rx-frames 128
```

2. 석탄 설정을 제거하려면 설정을 무시 하도록 설정합니다. 예를 들어 **ethtool.coalesce-rx-frames** 설정을 제거하려면 다음을 입력합니다.

```
nmcli connection modify enp1s0 ethtool.coalesce-rx-frames ignore
```

3. 네트워크 프로필을 다시 활성화하려면 다음을 수행합니다.

```
nmcli connection up enp1s0
```

#### 검증 단계

1. 네트워크 장치의 현재 오프로드 기능을 표시하려면 **ethtool -c** 명령을 사용합니다.

```
ethtool -c network_device
```

#### 추가 리소스

- [NetworkManager에서 지원하는 결합 설정](#)

### 40.3. RHEL 시스템 역할을 사용하여 ETHTOOL COALESCE 설정 설정

네트워크 **RHEL** 시스템 역할을 사용하여 **NetworkManager** 연결의 **ethtool coalesce** 설정을 구성할 수 있습니다.





## 중요

네트워크 **RHEL** 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 값과 일치하지 않는 경우 시스템 역할이 동일한 이름으로 기존 연결 프로필을 덮어씁니다. 따라서 IP 구성이 이미 있는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20** 에 /24 서브넷 마스크
- 정적 IPv6 주소 - **2001:db8:1::1** 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - **198.51.100.254**
- IPv6 기본 게이트웨이 - **2001:db8:1::fffe**
- IPv4 DNS 서버 **198.51.100.200**
- IPv6 DNS 서버 - **2001:db8:1::ffbb**
- DNS 검색 도메인 - **example.com**
- **ethtool Coesce** 설정:
  - **RX 프레임: 128**
  - **TX 프레임: 128**

사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

## 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/configure-ethernet-device-with-ethtool-coalesce-settings.yml** 플레이북을 만듭니다.

```

- name: Configure an Ethernet connection with ethtool coalesce settings
 hosts: node.example.com
 become: true
 tasks:
 - include_role:
 name: rhel-system-roles.network

 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 198.51.100.20/24
 - 2001:db8:1::1/64
 gateway4: 198.51.100.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 198.51.100.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
 ethtool:
 coalesce:
 rx_frames: 128
 tx_frames: 128
 state: up
```

3.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/configure-ethernet-device-with-ethtoolcoalesce-
settings.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-
with-ethtoolcoalesce-settings.yml
```

**--ask-become-pass** 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

•

[/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)

•

[ansible-playbook\(1\)](#) 도움말 페이지

## 41장. MACSEC을 사용하여 동일한 물리적 네트워크에서 계층 2 트래픽 암호화

**MACsec**을 사용하여 두 장치 간(**point-to-point**) 간의 통신을 보호할 수 있습니다. 예를 들어 분기 사무실은 중앙 사무실과 함께 **Metro-Ethernet** 연결을 통해 연결되며, 사무실을 연결하는 두 호스트에서 **MACsec**을 구성하여 보안을 강화할 수 있습니다.

**Media Access Control Security(MACsec)**는 다음을 포함하여 이더넷 링크를 통해 다양한 트래픽 유형을 보호하는 계층 2 프로토콜입니다.

- **DHCP**(동적 호스트 구성 프로토콜)
- 주소 확인 프로토콜 (**ARP**)
- 인터넷 프로토콜 버전 4 / 6 (**IPv4 / IPv6**) 및
- **TCP** 또는 **UDP**와 같은 **IP**를 통한 트래픽

**MACsec**은 기본적으로 **GCM-AES-128** 알고리즘을 사용하여 **LAN**의 모든 트래픽을 암호화하고 인증하며, 사전 공유 키를 사용하여 참가자 호스트 간 연결을 설정합니다. 사전 공유 키를 변경하려면 **MACsec**을 사용하는 네트워크의 모든 호스트에서 **NM** 구성을 업데이트해야 합니다.

**MACsec** 연결은 이더넷 네트워크 카드, **VLAN** 또는 터널 장치와 같은 이더넷 장치를 상위로 사용합니다. **MACsec** 장치에서만 암호화된 연결을 사용하는 다른 호스트와 통신하도록 **IP** 구성을 설정하거나 상위 장치에서 **IP** 구성을 설정할 수도 있습니다. 후자의 경우, 암호화되지 않은 연결을 사용하고 **MACsec** 장치를 사용하여 암호화된 연결에 **MACsec** 장치를 사용하여 다른 호스트와 통신하는 데 상위 장치를 사용할 수 있습니다.

**MACsec**에는 특별한 하드웨어가 필요하지 않습니다. 예를 들어 호스트와 스위치 간의 트래픽만 암호화하려는 경우를 제외하고 모든 스위치를 사용할 수 있습니다. 이 시나리오에서는 스위치에서 **MACsec**도 지원해야 합니다.

즉, **MACsec**을 구성하는 일반적인 두 가지 방법이 있습니다.

- 호스트 대 호스트 및

다른 호스트로 전환한 후 다른 호스트로 전환하는 호스트



중요

**MACsec은 동일한 (물리적 또는 가상) LAN에 있는 호스트 간에만 사용할 수 있습니다.**

#### 41.1. NMCLI를 사용하여 MACSEC 연결 구성

**nmcli** 유틸리티를 사용하여 **MACsec**을 사용하도록 이더넷 인터페이스를 구성할 수 있습니다. 이 절차에서는 이더넷을 통해 연결된 두 호스트 간에 **MACsec** 연결을 생성하는 방법을 설명합니다.

##### 절차

1.

**MACsec**을 구성하는 첫 번째 호스트에서 다음을 수행합니다.

•

사전 공유 키에 대해 **CAK**(연결 연결 키) 및 연결 연결 키 이름(**CKN**)을 만듭니다.

a.

**16바이트 16진수 CAK**를 생성합니다.

```
dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
50b71a8ef0bd5751ea76de6d6c98c03a
```

b.

**32바이트 16진수 CKN**을 생성합니다.

```
dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

2.

**MACsec** 연결을 통해 연결하려는 두 호스트 모두에서 다음을 수행합니다.

3.

**MACsec** 연결을 생성합니다.

```
nmcli connection add type macsec con-name macsec0 ifname macsec0
connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-
cak 50b71a8ef0bd5751ea76de6d6c98c03a macsec.mka-ckn
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

이전 단계에서 생성된 **CAK** 및 **CKN**을 **macsec.mka-cak** 및 **macsec.mka-ckn** 매개변수의 사용합니다. **MACsec** 보호 네트워크의 모든 호스트에서 값이 동일해야 합니다.

4.

**MACsec 연결에서 IP 설정을 구성합니다.**

a.

**IPv4 설정을 구성합니다.** 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **macsec0** 연결로 설정하려면 다음을 입력합니다.

```
nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
'192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
```

b.

**IPv6 설정을 구성합니다.** 예를 들어 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **macsec0** 연결로 설정하려면 다음을 입력합니다.

```
nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
'2001:db8:1::1/32' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd'
```

5.

**연결을 활성화합니다.**

```
nmcli connection up macsec0
```

## 검증 단계

1.

**트래픽이 암호화되었는지 확인합니다.**

```
tcpdump -nn -i enp1s0
```

2.

**선택 사항: 암호화되지 않은 트래픽을 표시합니다.**

```
tcpdump -nn -i macsec0
```

3.

**MACsec 통계를 표시합니다.**

```
ip macsec show
```

4.

**각 유형의 보호 유형에 대한 개별 카운터 표시: 무결성 전용 (encrypt off) 및 암호화 (encrypt on)**

**# ip -s macsec show**

#### 41.2. 추가 리소스

- [\*\*MACsec: 네트워크 트래픽 블로그 암호화에 다른 솔루션.\*\*](#)

## 42장. 다른 도메인에 다른 DNS 서버 사용

기본적으로 **RHEL(Red Hat Enterprise Linux)**은 모든 **DNS** 요청을 **/etc/resolv.conf** 파일에 지정된 첫 번째 **DNS** 서버로 보냅니다. 이 서버가 응답하지 않으면 **RHEL**은 이 파일에서 다음 서버를 사용합니다.

하나의 **DNS** 서버가 모든 도메인을 확인할 수 없는 환경에서 관리자는 특정 도메인의 **DNS** 요청을 선택한 **DNS** 서버로 보내도록 **RHEL**을 구성할 수 있습니다. 예를 들어 **example.com** 및 다른 **DNS** 서버에 대한 쿼리를 확인하도록 하나의 **DNS** 서버를 구성하여 **example.net**에 대한 쿼리를 확인할 수 있습니다. 다른 모든 **DNS** 요청의 경우 **RHEL**은 기본 게이트웨이와 연결에 구성된 **DNS** 서버를 사용합니다.

### 중요

**systemd-resolved** 서비스는 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(**SLA**)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있으며 프로덕션에는 사용하지 않는 것이 좋습니다. 이러한 미리보기를 통해 향후 제품 기능에 조기에 액세스할 수 있어 고객이 개발 과정에서 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능에 대한 지원 범위에 대한 자세한 내용은 **Red Hat** 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

### 42.1. 선택한 DNS 서버로 특정 도메인의 DNS 요청 전송

이 섹션에서는 특정 도메인에 대한 **DNS** 쿼리를 선택한 **DNS** 서버로 전송하도록 **systemd-resolved** 서비스 및 **NetworkManager**를 구성합니다.

이 섹션의 절차를 완료하면 **RHEL**은 **/etc/resolv.conf** 파일에서 **systemd-resolved**에서 제공하는 **DNS** 서비스를 사용합니다. **systemd-resolved** 서비스는 포트 53 IP 주소 127.0.0.53에서 수신 대기하는 **DNS** 서비스를 시작합니다. 서비스는 **DNS** 요청을 **NetworkManager**에 지정된 해당 **DNS** 서버로 동적으로 라우팅합니다.

### 참고

**127.0.0.53** 주소는 네트워크가 아닌 로컬 시스템에서만 연결할 수 있습니다.

### 사전 요구 사항



시스템에는 여러 개의 **NetworkManager** 연결이 구성되어 있습니다.



- **DNS 서버 및 검색 도메인은 특정 도메인을 확인하는 데 사용되는 NetworkManager 연결에 구성됩니다.**

예를 들어 VPN 연결에 지정된 DNS 서버가 **example.com** 도메인에 대한 쿼리를 확인하는 경우 VPN 연결 프로필에는 다음이 있어야 합니다.

- **example.com**을 확인할 수 있는 DNS 서버 구성
- 검색 도메인을 **ipv4.dns-search** 및 **ipv6.dns-search** 매개변수에서 **example.com** 으로 구성

### 절차

1. **systemd-resolved** 서비스를 시작하고 활성화합니다.

```
systemctl --now enable systemd-resolved
```

2. **/etc/NetworkManager/NetworkManager.conf** 파일을 편집하고 **[main]** 섹션에서 다음 항목을 설정합니다.

```
dns=systemd-resolved
```

3. **NetworkManager** 서비스를 다시 로드합니다.

```
systemctl reload NetworkManager
```

### 검증 단계

1. **/etc/resolv.conf** 파일의 **nameserver** 항목이 **127.0.0.53** 을 참조하는지 확인합니다.

```
cat /etc/resolv.conf
nameserver 127.0.0.53
```

2. **systemd-resolved** 서비스가 로컬 IP 주소 **127.0.0.53** 의 포트 **53**에서 수신 대기하는지 확인합니다.

```
ss -tulpn | grep "127.0.0.53"
```

```
udp UNCONN 0 0 127.0.0.53%lo:53 0.0.0.0:* users:(("systemd-
resolve",pid=1050,fd=12))
tcp LISTEN 0 4096 127.0.0.53%lo:53 0.0.0.0:* users:(("systemd-
resolve",pid=1050,fd=13))
```

#### 추가 리소스

- ***NetworkManager.conf(5)*** 도움말 페이지의 ***dns*** 매개 변수 설명

## 43장. IPVLAN 시작하기

이 문서에서는 **IPVLAN** 드라이버를 설명합니다.

### 43.1. IPVLAN 개요

**IPVLAN**은 컨테이너 환경에서 호스트 네트워크에 액세스하는 데 사용할 수 있는 가상 네트워크 장치의 드라이버입니다. **IPVLAN**은 호스트 네트워크 내부에 생성된 **IPVLAN** 장치 수에 관계없이 외부 네트워크에 단일 **MAC** 주소를 노출합니다. 즉, 사용자는 여러 컨테이너에 여러 **IPVLAN** 장치를 가질 수 있으며 해당 스위치는 단일 **MAC** 주소를 읽습니다. **IPVLAN** 드라이버는 로컬 스위치가 관리할 수 있는 총 **MAC** 주소 수에 제약을 줄 때 유용합니다.

### 43.2. IPVLAN 모드

**IPVLAN**에는 다음 모드를 사용할 수 있습니다.

- 

#### L2 모드

**IPVLAN L2** 모드에서 가상 장치는 **ARP**(주소 확인 프로토콜) 요청을 수신하고 응답합니다. **netfilter** 프레임워크는 가상 장치를 보유한 컨테이너 내부에서만 실행됩니다. 컨테이너화된 트래픽의 기본 네임스페이스에서 **netfilter** 체인이 실행되지 않습니다. **L2** 모드를 사용하면 우수한 성능을 제공하지만 네트워크 트래픽에 대한 제어는 줄어듭니다.

- 

#### L3 모드

**L3** 모드에서 가상 장치는 **L3** 이상의 트래픽만 처리합니다. 가상 장치는 **ARP** 요청에 응답하지 않으며 사용자는 관련 피어에서 **IPVLAN IP** 주소에 대한 주력 항목을 수동으로 구성해야 합니다. 관련 컨테이너의 송신 트래픽은 기본 네임스페이스의 **netfilter POSTROUTING** 및 **OUTPUT** 체인에 배치되고 수신 트래픽이 **L2** 모드와 동일한 방식으로 스레드됩니다. **L3** 모드를 사용하면 좋은 제어가 가능하지만 네트워크 트래픽 성능이 저하됩니다.

- 

#### L3S 모드

**L3S** 모드에서는 가상 장치에서 관련 컨테이너의 송신 및 수신 트래픽 모두 기본 네임스페이스의 **netfilter** 체인에 배치된다는 점을 제외하고 **L3** 모드와 동일한 방식으로 처리합니다. **L3S** 모드는 **L3** 모드와 유사한 방식으로 작동하지만 네트워크 제어 기능이 향상됩니다.



## 참고

**L3 및 L3 S 모드의 경우 IPVLAN 가상 장치는 브로드캐스트 및 멀티캐스트 트래픽을 수신하지 않습니다.**

### 43.3. MACVLAN 개요

**MACVLAN** 드라이버를 사용하면 각 **NIC**에서 고유한 **MAC** 주소로 식별되는 여러 가상 네트워크 장치를 생성할 수 있습니다. 물리적 **NIC**에 있는 패킷은 대상의 **MAC** 주소를 통해 관련 **MACVLAN** 장치로 분리됩니다. **MACVLAN** 장치는 캡슐화 수준을 추가하지 않습니다.

### 43.4. IPVLAN 및 MACVLAN 비교

다음 표는 **MACVLAN**과 **IPVLAN**의 주요 차이점을 보여줍니다.

| MACVLAN                                                                                                    | IPVLAN                                                       |
|------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| 각 <b>MACVLAN</b> 장치에 대해 <b>MAC</b> 주소를 사용합니다. 스위치에서 <b>MAC</b> 테이블의 <b>MAC</b> 주소 덮어쓰기로 인해 연결이 끊어질 수 있습니다. | <b>IPVLAN</b> 장치 수를 제한하지 않는 단일 <b>MAC</b> 주소를 사용합니다.         |
| 글로벌 네임스페이스의 Netfilter 규칙은 하위 네임스페이스의 <b>MACVLAN</b> 장치에 대한 트래픽에 영향을 줄 수 없습니다.                              | <b>L3 모드 및 L3S 모드</b> 에서 <b>IPVLAN</b> 장치로의 트래픽을 제어할 수 있습니다. |

**IPVLAN** 및 **MACVLAN** 모두 캡슐화가 필요하지 않습니다.

### 43.5. IPROUTE2를 사용하여 IPVLAN 장치 생성 및 구성

다음 절차에서는 **iproute2**를 사용하여 **IPVLAN** 장치를 설정하는 방법을 보여줍니다.

#### 절차

1.

**IPVLAN** 장치를 생성하려면 다음 명령을 입력합니다.

```
ip link add link real_NIC_device name IPVLAN_device type ipvlan mode l2
```

**NIC**(네트워크 인터페이스 컨트롤러)는 컴퓨터를 네트워크에 연결하는 하드웨어 구성 요소입니다.

니다.

#### 예 43.1. IPVLAN 장치 생성

```
ip link add link enp0s31f6 name my_ipvlan type ipvlan mode l2
ip link
47: my_ipvlan@enp0s31f6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state
DOWN mode DEFAULT group default qlen 1000 link/ether e8:6a:6e:8a:a2:44 brd
ff:ff:ff:ff:ff:ff
```

2.

인터페이스에 IPv4 또는 IPv6 주소를 할당하려면 다음 명령을 입력합니다.

```
ip addr add dev IPVLAN_device IP_address/subnet_mask_prefix
```

3.

L3 모드 또는 L3S 모드에서 IPVLAN 장치를 구성하는 경우 다음과 같이 설정합니다.

a.

원격 호스트에서 원격 피어의 인접 설정을 구성합니다.

```
ip neigh add dev peer_device IPVLAN_device_IP_address lladdr MAC_address
```

여기서 **MAC\_address** 는 IPVLAN 장치가 기반으로 하는 실제 NIC의 MAC 주소입니다.

b.

다음 명령을 사용하여 L3 모드의 IPVLAN 장치를 구성합니다.

```
ip route add dev <real_NIC_device> <peer_IP_address/32>
```

L3S 모드의 경우 :

```
ip route add dev real_NIC_device peer_IP_address/32
```

여기서 IP 주소는 원격 피어의 주소를 나타냅니다.

4.

활성 IPVLAN 장치를 설정하려면 다음 명령을 입력합니다.

```
ip link set dev IPVLAN_device up
```

5.

**IPVLAN** 장치가 활성 상태인지 확인하려면 원격 호스트에서 다음 명령을 실행합니다.

```
ping IP_address
```

**IP\_address** 에서 **IPVLAN** 장치의 **IP** 주소를 사용하는 위치입니다.

## 44장. 다른 인터페이스에서 동일한 IP 주소 재사용

관리자는 가상 라우팅 및 전달(VRF)을 사용하면 동일한 호스트에서 동시에 여러 라우팅 테이블을 사용할 수 있습니다. 이를 위해 VRF는 계층 3에서 네트워크를 분할합니다. 이를 통해 관리자는 VRF 도메인당 별도의 독립적인 라우팅 테이블을 사용하여 트래픽을 격리할 수 있습니다. 이 기술은 운영 체제에서 다른 VLAN 태그를 사용하여 동일한 물리적 미디어를 공유하는 트래픽을 격리하는 계층 2에서 네트워크를 분할하는 VLAN(가상 LAN)과 유사합니다.

계층 2에서 VRF를 파티셔닝하는 한 가지 이점은 라우팅이 관련된 피어 수를 고려하는 것이 더 효율적이라는 것입니다.

Red Hat Enterprise Linux는 각 VRF 도메인에 가상 vrf 장치를 사용하고 기존 네트워크 장치를 VRF 장치에 추가하여 VRF 도메인에 경로를 추가합니다. 원래 장치에 연결된 주소와 경로는 VRF 도메인 내에서 이동됩니다.

각 VRF 도메인은 서로 격리됩니다.

### 44.1. 다른 인터페이스에서 동일한 IP 주소 영구적으로 재사용

이 절차에서는 VRF 기능을 사용하여 한 서버의 여러 인터페이스에서 동일한 IP 주소를 영구적으로 사용하는 방법을 설명합니다.

#### 중요

원격 피어가 동일한 IP 주소를 재사용하면서 두 VRF 인터페이스에 모두 연결할 수 있도록 하려면 네트워크 인터페이스는 다른 브로드캐스트 도메인에 속해야 합니다. 네트워크의 브로드캐스트 도메인은 노드 집합으로, 누구나 전송하는 브로드캐스트 트래픽을 수신합니다. 대부분의 구성에서 동일한 스위치에 연결된 모든 노드는 동일한 브로드캐스트 도메인에 속합니다.

#### 사전 요구 사항

- root 사용자로 로그인했습니다.
- 네트워크 인터페이스는 구성되지 않습니다.

#### 절차

1.

첫 번째 **VRF** 장치를 생성하고 구성합니다.

a.

**VRF** 장치에 대한 연결을 만들어 라우팅 테이블에 할당합니다. 예를 들어 **1001** 라우팅 테이블에 할당된 **vrf0** 이라는 **VRF** 장치를 생성하려면 다음을 수행합니다.

```
nmcli connection add type vrf ifname vrf0 con-name vrf0 table 1001 ipv4.method disabled ipv6.method disabled
```

b.

**vrf0** 장치를 활성화합니다.

```
nmcli connection up vrf0
```

c.

방금 만든 **VRF**에 네트워크 장치를 할당합니다. 예를 들어, **enp1s0** 이더넷 장치를 **vrf0** **VRF** 장치에 추가하고 **IP** 주소와 서브넷 마스크를 **enp1s0** 에 할당하려면 다음을 입력합니다.

```
nmcli connection add type ethernet con-name vrf.enp1s0 ifname enp1s0 master vrf0 ipv4.method manual ipv4.address 192.0.2.1/24
```

d.

**vrf.enp1s0** 연결을 활성화합니다.

```
nmcli connection up vrf.enp1s0
```

2.

다음 **VRF** 장치를 생성하고 구성합니다.

a.

**VRF** 장치를 생성하고 라우팅 테이블에 할당합니다. 예를 들어 **1002** 라우팅 테이블에 할당된 **vrf1** 이라는 **VRF** 장치를 생성하려면 다음을 입력합니다.

```
nmcli connection add type vrf ifname vrf1 con-name vrf1 table 1002 ipv4.method disabled ipv6.method disabled
```

b.

**vrf1** 장치를 활성화합니다.

```
nmcli connection up vrf1
```

c.

방금 만든 **VRF**에 네트워크 장치를 할당합니다. 예를 들어, **enp7s0** 이더넷 장치를 **vrf1** **VRF** 장치에 추가하고 **IP** 주소와 서브넷 마스크를 **enp7s0** 에 할당하려면 다음을 입력합니다.



```
nmcli connection add type ethernet con-name vrf.enp7s0 ifname enp7s0 master
vrf1 ipv4.method manual ipv4.address 192.0.2.1/24
```

d.

**vrf.enp7s0** 장치를 활성화합니다.

```
nmcli connection up vrf.enp7s0
```

#### 44.2. 다른 인터페이스에서 동일한 IP 주소 일시적으로 재사용

이 섹션의 절차에서는 **VRF**(가상 라우팅 및 전달) 기능을 사용하여 한 서버의 여러 인터페이스에서 동일한 IP 주소를 일시적으로 사용하는 방법을 설명합니다. 시스템을 재부팅한 후 구성이 일시적이고 손실되므로 이 절차는 테스트 목적으로만 사용하십시오.

##### 중요

원격 피어가 동일한 IP 주소를 재사용하면서 두 **VRF** 인터페이스에 모두 연결할 수 있도록 하려면 네트워크 인터페이스는 다른 브로드캐스트 도메인에 속해야 합니다. 네트워크의 브로드캐스트 도메인은 브로드캐스트 트래픽을 수신하는 노드 집합입니다. 대부분의 구성에서 동일한 스위치에 연결된 모든 노드는 동일한 브로드캐스트 도메인에 속합니다.

##### 사전 요구 사항

- **root** 사용자로 로그인했습니다.
- 네트워크 인터페이스는 구성되지 않습니다.

##### 절차

1.

첫 번째 **VRF** 장치를 생성하고 구성합니다.

a.

**VRF** 장치를 생성하고 라우팅 테이블에 할당합니다. 예를 들어 **1001** 라우팅 테이블에 할당된 **blue** 라는 **VRF** 장치를 생성하려면 다음을 수행합니다.

```
ip link add dev blue type vrf table 1001
```

b.

파란색 장치를 활성화합니다.

```
ip link set dev blue up
```

c.

네트워크 장치를 **VRF** 장치에 할당합니다. 예를 들어, **Blue VRF** 장치에 **enp1s0** 이더넷 장치를 추가하려면 다음을 수행합니다.

```
ip link set dev enp1s0 master blue
```

d.

**enp1s0** 장치를 활성화합니다.

```
ip link set dev enp1s0 up
```

e.

**IP** 주소 및 서브넷 마스크를 **enp1s0** 장치에 할당합니다. 예를 들어, **192.0.2.1/24** 로 설정하려면 다음을 수행합니다.

```
ip addr add dev enp1s0 192.0.2.1/24
```

2.

다음 **VRF** 장치를 생성하고 구성합니다.

a.

**VRF** 장치를 생성하고 라우팅 테이블에 할당합니다. 예를 들어 **1002** 라우팅 테이블에 할당된 **red** 라는 **VRF** 장치를 생성하려면 다음을 수행합니다.

```
ip link add dev red type vrf table 1002
```

b.

빨간색 장치를 활성화합니다.

```
ip link set dev red up
```

c.

네트워크 장치를 **VRF** 장치에 할당합니다. 예를 들어, **red VRF** 장치에 **enp7s0** 이더넷 장치를 추가하려면 다음을 수행합니다.

```
ip link set dev enp7s0 master red
```

d.

**enp7s0** 장치를 활성화합니다.

```
ip link set dev enp7s0 up
```

e.

**Blue VRF 도메인에서 enp1s0 에 사용한 것과 동일한 IP 주소 및 서브넷 마스크를 enp7s0 장치에 할당합니다.**

```
ip addr add dev enp7s0 192.0.2.1/24
```

3.

선택적으로, 위에서 설명한 대로 추가 **VRF** 장치를 생성합니다.

#### 44.3. 추가 리소스

- 

**kernel- doc** 패키지의 `/usr/share/doc/kernel-doc-  
<kernel_version>/Documentation/networking/vrf.txt`

## 45장. 분리된 VRF 네트워크 내에서 서비스 시작

가상 라우팅 및 전달(VRF)을 사용하면 운영 체제의 기본 라우팅 테이블과 다른 라우팅 테이블을 사용하여 격리된 네트워크를 생성할 수 있습니다. 그런 다음 서비스 및 애플리케이션을 시작하여 해당 라우팅 테이블에 정의된 네트워크에만 액세스할 수 있습니다.

### 45.1. VRF 장치 구성

VRF(가상 라우팅 및 전달)를 사용하려면 VRF 장치를 생성하고 실제 또는 가상 네트워크 인터페이스를 연결하고 라우팅 정보를 연결합니다.



#### 주의

원격으로 잠길 수 없도록 하려면 로컬 콘솔에서 이 절차를 수행하거나 VRF 장치에 할당하지 않으려는 네트워크 인터페이스를 통해 원격으로 수행합니다.

#### 사전 요구 사항

- 로컬로 로그인하거나 VRF 장치에 할당하려는 네트워크와 다른 인터페이스를 사용하여 로그인됩니다.

#### 절차

1. 동일한 이름의 가상 장치로 **vrf0** 연결을 만들고 라우팅 테이블 **1000**에 연결합니다.

```
nmcli connection add type vrf ifname vrf0 con-name vrf0 table 1000 ipv4.method disabled ipv6.method disabled
```

2. **enp1s0** 장치를 **vrf0** 연결에 추가하고 IP 설정을 구성합니다.

```
nmcli connection add type ethernet con-name enp1s0 ifname enp1s0 master vrf0 ipv4.method manual ipv4.address 192.0.2.1/24 ipv4.gateway 192.0.2.254
```

이 명령은 **enp1s0** 연결을 **vrf0** 연결 포트에 생성합니다. 이 구성으로 인해 라우팅 정보는 **vrf0** 장치와 연결된 라우팅 테이블( **1000** )에 자동으로 할당됩니다.

3.

격리된 네트워크에 고정 경로가 필요한 경우 다음을 수행합니다.

a.

정적 경로를 추가합니다.

```
nmcli connection modify enp1s0 +ipv4.routes "198.51.100.0/24 192.0.2.2"
```

이렇게 하면 198.51.100.0/24 네트워크에 라우터로 192.0.2.2 를 사용하는 경로가 추가됩니다.

b.

연결을 활성화합니다.

```
nmcli connection up enp1s0
```

### 검증

1.

vrf0 과 연결된 장치의 IP 설정을 표시합니다.

```
ip -br addr show vrf vrf0
enp1s0 UP 192.0.2.15/24
```

2.

VRF 장치 및 관련 라우팅 테이블을 표시합니다.

```
ip vrf show
Name Table

vrf0 1000
```

3.

기본 라우팅 테이블을 표시합니다.

```
ip route show
default via 192.168.0.1 dev enp1s0 proto static metric 100
```

4.

라우팅 테이블 1000 을 표시합니다.

```
ip route show table 1000
default via 192.0.2.254 dev enp1s0 proto static metric 101
broadcast 192.0.2.0 dev enp1s0 proto kernel scope link src 192.0.2.1
192.0.2.0/24 dev enp1s0 proto kernel scope link src 192.0.2.1 metric 101
```

```
local 192.0.2.1 dev enp1s0 proto kernel scope host src 192.0.2.1
broadcast 192.0.2.255 dev enp1s0 proto kernel scope link src 192.0.2.1
198.51.100.0/24 via 192.0.2.2 dev enp1s0 proto static metric 101
```

기본 항목은 이 라우팅 테이블을 사용하는 서비스가 기본 라우팅 테이블에 있는 기본 게이트웨이가 아닌 기본 게이트웨이로 **192.0.2.254** 를 사용함을 나타냅니다.

5.

**vrf0** 과 연결된 네트워크에서 **traceroute** 유틸리티를 실행하여 유틸리티가 테이블 **1000** 의 경로를 사용하는지 확인합니다.

```
ip vrf exec vrf0 traceroute 203.0.113.1
traceroute to 203.0.113.1 (203.0.113.1), 30 hops max, 60 byte packets
 1 192.0.2.254 (192.0.2.254) 0.516 ms 0.459 ms 0.430 ms
...
```

첫 번째 홉은 시스템의 기본 라우팅 테이블에서 기본 게이트웨이가 아닌 라우팅 테이블 **1000** 에 할당되는 기본 게이트웨이입니다.

#### 추가 리소스

- 

**ip-vrf(8)**

## 45.2. 분리된 VRF 네트워크 내에서 서비스 시작

분리된 가상 라우팅 및 전달(VRF) 네트워크 내에서 시작하도록 **Apache HTTP** 서버와 같은 서비스를 구성할 수 있습니다.



#### 중요

서비스는 동일한 **VRF** 네트워크에 있는 로컬 **IP** 주소에만 바인딩할 수 있습니다.

#### 사전 요구 사항

- 

**vrf0** 장치를 구성했습니다.

- 

**vrf0** 장치와 연결된 인터페이스에 할당된 **IP** 주소에서만 수신 대기하도록 **Apache HTTP Server**를 구성했습니다.

## 설치

1. **httpd systemd** 서비스의 내용을 표시합니다.

```
systemctl cat httpd
...
[Service]
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
...
```

분리된 VRF 네트워크 내에서 동일한 명령을 실행하려면 이후 단계에서 **ExecStart** 매개변수의 콘텐츠가 필요합니다.

2. **/etc/systemd/system/httpd.service.d/** 디렉터리를 만듭니다.

```
mkdir /etc/systemd/system/httpd.service.d/
```

3. 다음 콘텐츠를 사용하여 **/etc/systemd/system/httpd.service.d/override.conf** 파일을 만듭니다.

```
[Service]
ExecStart=
ExecStart=/usr/sbin/ip vrf exec vrf0 /usr/sbin/httpd $OPTIONS -DFOREGROUND
```

**ExecStart** 매개변수를 재정의하려면 먼저 설정을 해제한 다음 표시된 대로 새 값으로 설정해야 합니다.

4. **systemd**를 다시 로드합니다.

```
systemctl daemon-reload
```

5. **httpd** 서비스를 다시 시작합니다.

```
systemctl restart httpd
```

## 검증

1. **httpd** 프로세스의 프로세스 ID(PID)를 표시합니다.

```
pidof -c httpd
1904 ...
```

2.

**PID**에 대한 **VRF** 연결을 표시합니다. 예를 들면 다음과 같습니다.

```
ip vrf identify 1904
vrf0
```

3.

**vrf0** 장치와 연결된 모든 **PID**를 표시합니다.

```
ip vrf pids vrf0
1904 httpd
...
```

추가 리소스

- [ip-vrf\(8\)](#)



## 46장. 시스템의 라우팅 프로토콜 설정

이 섹션에서는 시스템에 필요한 라우팅 프로토콜을 활성화하고 설정하는 방법을 설명합니다.

### 46.1. PVCROUTING 소개

**Free Range Routing ( Free Range Routing )**은 AppStream 리포지토리에서 사용할 수 있는 **frr** 패키지에서 제공하는 라우팅 프로토콜 스택입니다.

**GAR**은 이전 **RHEL** 버전에서 사용된 **Quagga** 를 대체합니다. 따라서, **pvc R** 은 여러 **IPv4** 및 **IPv6** 라우팅 프로토콜을 지원하는 **TCP/IP** 기반 라우팅 서비스를 제공합니다.

지원되는 프로토콜은 다음과 같습니다.

- **BGP(Bound Gateway Protocol)**
- **중간 시스템에서 중간 시스템(IS-IS)**
- **OSPF(Open shortest Path First)**
- **프로토콜 독립 멀티 캐스트 (PIM)**
- **라우팅 정보 프로토콜 (RIP)**
- **라우팅 정보 프로토콜 차세대RIPng(RIPng)**
- **EIGRP(Enhanced Interior Gateway Routing Protocol)**
- **Next Hop Resolution Protocol (NHRP)**

- 양방향 전달 탐지 (**BFD**)
- 정책 기반 라우팅 (**PBR**)

**collectdR**은 다음 서비스의 컬렉션입니다.

- **zebra**
- **bgpd**
- **isisd**
- **ospfd**
- **ospf6d**
- **pimd**
- **ripd**
- **ripngd**
- **eigrpd**
- **nhrpd**
- **bfdd**

- **pbrd**
- **staticd**
- **fabriCD**

**frr**가 설치된 경우 시스템은 전용 라우터 역할을 할 수 있으며 라우팅 프로토콜을 사용하여 내부 또는 외부 네트워크의 다른 라우터와 라우팅 정보를 교환할 수 있습니다.

## 46.2. COLLECTDRROUTING 설정

이 섹션에서는 무료 범위 라우팅(**FRRouting** 또는 **FRR**)을 설정하는 방법을 설명합니다.

### 사전 요구 사항

- **frr** 패키지가 시스템에 설치되어 있는지 확인합니다.

```
yum install frr
```

### 절차

1. **/etc/frr/daemons** 구성 파일을 편집하고 시스템에 필요한 데몬을 활성화합니다.

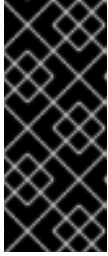
예를 들어, 가져온 데몬을 활성화하려면 다음 행을 포함합니다.

```
ripd=yes
```



주의

**zebra** 데몬은 **zebra=yes**를 항상 활성화해야 합니다.



## 중요

기본적으로 `/etc/frr/daemons`에는 모든 데몬에 대한 `[daemon_name]=no` 항목이 포함되어 있습니다. 따라서 모든 데몬이 비활성화되어 있고 시스템을 새로 설치한 후 **PUR**을 시작하면 효과가 없습니다.

2.

**frr** 서비스를 시작합니다.

```
systemctl start frr
```

3.

선택적으로 부팅 시 자동으로 시작되도록 **HFSR**을 설정할 수도 있습니다.

```
systemctl enable frr
```

### 46.3. COLLECTDR의 설정 수정

이 섹션에서는 다음을 설명합니다.

- **fa R**을 설정한 후 추가 데몬을 활성화하는 방법
- **collectd R**을 설정한 후 데몬을 비활성화하는 방법

#### 사전 요구 사항

- **collectd Routing** 설정에 설명된 대로가 설정되어 있습니다.

#### 절차

1.

`/etc/frr/daemons` 구성 파일을 편집하고 필요한 데몬의 행을 **no** 대신 **yes**로 수정합니다.

예를 들어, 가져온 데몬을 활성화하려면 다음을 수행합니다.

```
ripd=yes
```

2. **frr** 서비스를 다시 로드합니다.

```
systemctl reload frr
```

#### 46.4. 특정 데몬의 구성 수정

기본 구성을 사용하면 **pvc R**의 모든 라우팅 데몬이 일반 라우터로만 작동할 수 있습니다.

데몬의 추가 구성에 다음 절차를 사용하십시오.

##### 절차

1. **/etc/frr/** 디렉토리 내에서 필요한 데몬에 대한 구성 파일을 생성하고 파일 이름을 다음과 같이 지정합니다.

```
[daemon_name].conf
```

예를 들어 **eigrpd** 데몬을 추가로 구성하려면 언급된 디렉토리에 **eigrpd.conf** 파일을 생성합니다.

2. 필요한 콘텐츠로 새 파일을 채웁니다.

특정 **GAR** 데몬의 설정 예제는 **/usr/share/doc/frr/** 디렉토리를 참조하십시오.

3. **frr** 서비스를 다시 로드합니다.

```
systemctl reload frr
```

## 47장. 기본 네트워크 설정 테스트

이 섹션에서는 기본 네트워크 테스트를 수행하는 방법에 대해 설명합니다.

### 47.1. PING 유틸리티를 사용하여 다른 호스트에 대한 IP 연결 확인

**ping** 유틸리티는 **ICMP** 패킷을 원격 호스트로 보냅니다. 이 기능을 사용하여 다른 호스트에 대한 IP 연결이 작동하는지 테스트할 수 있습니다.

#### 절차

- 기본 게이트웨이와 같은 동일한 서브넷에서 호스트의 IP 주소를 **ping**합니다.

```
ping 192.0.2.3
```

명령이 실패하면 기본 게이트웨이 설정을 확인합니다.

- 원격 서브넷에서 호스트의 IP 주소를 **ping**합니다.

```
ping 198.162.3.1
```

명령이 실패하면 기본 게이트웨이 설정을 확인하고 게이트웨이가 연결된 네트워크 간에 패킷을 전달하는지 확인합니다.

### 47.2. 호스트 유틸리티를 사용하여 이름 확인

다음 절차에서는 **Red Hat Enterprise Linux**에서 이름 확인을 확인하는 방법을 설명합니다.

#### 절차

- 호스트 유틸리티를 사용하여 이름 확인이 작동하는지 확인합니다. 예를 들어 **client.example.com** 호스트 이름을 IP 주소로 확인하려면 다음을 입력합니다.

```
host client.example.com
```

연결 시간 초과 또는 서버에 도달할 수 없는 등 명령이 오류를 반환하는 경우 **DNS** 설정을 확인합니다.

## 48장. NETWORKMANAGER를 사용하여 DHCPCLIENT 종료 후크 실행

**NetworkManager** 디스패치기 스크립트를 사용하여 **dhclient** 종료 후크를 실행할 수 있습니다.

### 48.1. NETWORKMANAGER 디스패치 도구 스크립트의 개념

**NetworkManager-dispatcher** 서비스는 네트워크 이벤트가 발생하면 알파벳순으로 사용자 제공 스크립트를 실행합니다. 이러한 스크립트는 일반적으로 셸 스크립트이지만 실행 가능한 스크립트 또는 애플리케이션일 수 있습니다. 예를 들어 디스패치 스크립트를 사용하여 **NetworkManager**로 관리할 수 없는 네트워크 관련 설정을 조정할 수 있습니다.

다음 디렉터리에 디스패치 스크립트를 저장할 수 있습니다.

- **/etc/NetworkManager/dispatcher.d/**: 디스패치 스크립트의 일반적인 위치는 **root** 사용자가 편집할 수 있습니다.
- **/usr/lib/NetworkManager/dispatcher.d/**: 사전 배포된 변경 불가능한 디스패치 스크립트의 경우.

보안상의 이유로 **NetworkManager-dispatcher** 서비스는 다음 조건이 충족된 경우에만 스크립트를 실행합니다.

- 스크립트는 **root** 사용자가 소유합니다.
- 스크립트는 루트 에서만 읽고 쓸 수 있습니다.
- 스크립트에 **setuid** 비트가 설정되어 있지 않습니다.

**NetworkManager-dispatcher** 서비스는 두 개의 인수를 사용하여 각 스크립트를 실행합니다.

1. 작업이 수행한 장치의 인터페이스 이름입니다.



2.

인터페이스가 활성화되면 와 같은 동작입니다.

**NetworkManager(8)** 도움말 페이지의 **systemtap** 스크립트 섹션에서는 스크립트에서 사용할 수 있는 작업 및 환경 변수에 대한 개요를 제공합니다.

**NetworkManager-dispatcher** 서비스는 한 번에 하나의 스크립트를 실행하지만 기본 **NetworkManager** 프로세스에서 비동기적으로 실행됩니다. 스크립트가 대기열에 저장되면 이후 이벤트가 더 이상 사용되지 않는 경우에도 서비스는 항상 해당 스크립트가 실행됩니다. 그러나 **NetworkManager-dispatcher** 서비스는 이전 스크립트의 종료를 기다리지 않고 즉시 `/etc/NetworkManager/dispatcher.d/no-wait.d/`의 파일을 참조하는 심볼릭 링크인 스크립트를 실행합니다.

추가 리소스

•

**NetworkManager(8)** 도움말 페이지의 **Dispatcher** 스크립트 섹션

## 48.2. DHCPCLIENT 종료 후크를 실행하는 NETWORKMANAGER 디스패치 스크립트 생성

이 섹션에서는 IPv4 주소를 DHCP 서버에서 할당하거나 업데이트할 때 `/etc/dhcp/dhclient-exit-hooks.d/` 디렉터리에 저장된 **dhclient** 종료 후크를 실행하는 **NetworkManager** 디스패치 스크립트를 작성하는 방법을 설명합니다.

사전 요구 사항

•

**dhclient** 종료 후크는 `/etc/dhcp/dhclient-exit-hooks.d/` 디렉터리에 저장됩니다.

절차

1.

다음 콘텐츠를 사용하여 `/etc/NetworkManager/dispatcher.d/12-dhclient-down` 파일을 생성합니다.

```
#!/bin/bash
Run dhclient.exit-hooks.d scripts

if [-n "$DHCP4_DHCP_LEASE_TIME"]; then
 if ["$2" = "dhcp4-change"] || ["$2" = "up"]; then
 if [-d /etc/dhcp/dhclient-exit-hooks.d]; then
 for f in /etc/dhcp/dhclient-exit-hooks.d/*.sh ; do
 if [-x "$f"]; then
 . "$f"
 fi
 fi
 fi
 fi
fi
```

```
done
fi
fi
fi
```

2. *root* 사용자를 파일의 소유자로 설정합니다.

```
chown root:root /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

3. *root* 사용자만 실행할 수 있도록 권한을 설정합니다.

```
chmod 0700 /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

4. SELinux 컨텍스트를 복원하십시오.

```
restorecon /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

#### 추가 리소스

- *NetworkManager(8)* 도움말 페이지의 *Dispatcher* 스크립트 섹션.

## 49장. NETWORKMANAGER 디버깅 소개

전체 또는 특정 도메인의 로그 수준을 늘리면 **NetworkManager**가 수행하는 작업에 대한 세부 정보를 기록할 수 있습니다. 관리자는 이 정보를 사용하여 문제를 해결할 수 있습니다. **NetworkManager**는 로깅 정보를 생성하기 위해 다양한 수준 및 도메인을 제공합니다. **/etc/NetworkManager/NetworkManager.conf** 파일은 **NetworkManager**의 기본 구성 파일입니다. 로그는 저널에 저장됩니다.

이 섹션에서는 **NetworkManager**에 디버그 로깅을 활성화하고 다양한 로깅 수준 및 도메인을 사용하여 로깅 세부 정보 양을 구성하는 방법에 대해 설명합니다.

### 49.1. 디버깅 수준 및 도메인

**levels** 및 **domains** 매개변수를 사용하여 **NetworkManager**의 디버깅을 관리할 수 있습니다. 수준은 상세 정보 표시 수준을 정의하는 반면 도메인은 지정된 심각도(레벨)로 로그를 기록하기 위해 메시지의 카테고리(카테고리)를 정의합니다.

| 로그 수준 | 설명                                 |
|-------|------------------------------------|
| OFF   | NetworkManager에 대한 메시지를 기록하지 않음    |
| ERR   | 중요한 오류만 기록합니다.                     |
| WARN  | 작업을 반영할 수 있는 경고 로그                 |
| INFO  | 상태 및 작업 추적에 유용한 다양한 정보 메시지를 기록합니다. |
| DEBUG | 디버깅 목적으로 자세한 로깅 사용                 |
| TRACE | DEBUG 수준보다 더 자세한 로깅 사용             |

후속 수준은 이전 수준의 모든 메시지를 기록합니다. 예를 들어 로그 수준을 **INFO**로 설정하면 **ERR** 및 **WARN** 로그 수준에 포함된 메시지도 로깅합니다.

추가 리소스

- [NetworkManager.conf\(5\) 도움말 페이지](#)

### 49.2. NETWORKMANAGER 로그 수준 설정

기본적으로 모든 로그 도메인은 **INFO** 로그 수준을 기록하도록 설정됩니다. 디버그 로그를 수집하기 전에 속도 제한을 비활성화합니다. 속도 제한이 있는 경우 **systemd-journald** 는 짧은 시간에 너무 많은 메시지가 있는 경우 메시지를 삭제합니다. 이는 로그 수준이 **TRACE** 일 때 발생할 수 있습니다.

이 절차에서는 속도 제한을 비활성화하고 모든 (**ALL**) 도메인에 대해 디버그 로그를 기록할 수 있습니다.

## 절차

1.

속도 제한을 비활성화하려면 **/etc/systemd/journald.conf** 파일을 편집하고 **[Journal]** 섹션의 **RateLimitBurst** 매개변수의 주석을 제거하고 값을 **0** 으로 설정합니다.

```
RateLimitBurst=0
```

2.

**systemd-journald** 서비스를 다시 시작합니다.

```
systemctl restart systemd-journald
```

3.

다음 콘텐츠를 사용하여 **/etc/NetworkManager/conf.d/95-nm-debug.conf** 파일을 생성합니다.

```
[logging]
domains=ALL:TRACE
```

**domains** 매개 변수에는 쉼표로 구분된 여러 **domain:level** 쌍을 포함할 수 있습니다.

4.

**NetworkManager** 서비스를 다시 시작합니다.

```
systemctl restart NetworkManager
```

## 검증

•

**systemd** 저널을 쿼리하여 **NetworkManager** 장치의 저널 항목을 표시합니다.

```
journalctl -u NetworkManager
```

```
...
```

```
Jun 30 15:24:32 server NetworkManager[164187]: <debug> [1656595472.4939] active-connection[0x5565143c80a0]: update activation type from assume to managed
```

```

Jun 30 15:24:32 server NetworkManager[164187]: <trace> [1656595472.4939]
device[55b33c3bdb72840c] (enp1s0): sys-iface-state: assume -> managed
Jun 30 15:24:32 server NetworkManager[164187]: <trace> [1656595472.4939]
l3cfg[4281fdf43e356454,ifindex=3]: commit type register (type "update", source
"device", existing a369f23014b9ede3) -> a369f23014b9ede3
Jun 30 15:24:32 server NetworkManager[164187]: <info> [1656595472.4940] manager:
NetworkManager state is now CONNECTED_SITE
...

```

### 49.3. NMCLI를 사용하여 런타임 시 로그 수준 일시적으로 설정

**nmcli** 를 사용하여 런타임 시 로그 수준을 변경할 수 있습니다. 그러나 설정 파일을 사용하여 디버깅을 활성화하고 **NetworkManager**를 다시 시작하는 것이 좋습니다. **.conf** 파일을 사용하여 디버깅 수준 및 도메인을 업데이트하면 부팅 문제를 디버그하고 초기 상태의 모든 로그를 캡처하는 데 도움이 됩니다.

#### 절차

1.

선택 사항: 현재 로깅 설정을 표시합니다.

```

nmcli general logging
LEVEL DOMAINS
INFO
PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVIC
E,OLPC,
WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS_PROPS,TEAM,CON
CHECK,DC
B,DISPATCH

```

2.

로깅 수준 및 도메인을 수정하려면 다음 옵션을 사용합니다.

- 

모든 도메인의 로그 수준을 동일한 **LEVEL** 로 설정하려면 다음을 입력합니다.

```
nmcli general logging level LEVEL domains ALL
```

- 

특정 도메인의 수준을 변경하려면 다음을 입력합니다.

```
nmcli general logging level LEVEL domains DOMAINS
```

이 명령을 사용하여 로깅 수준을 업데이트하면 기타 모든 도메인에 대한 로깅이 비활성화됩니다.

- 특정 도메인의 수준을 변경하고 다른 모든 도메인의 수준을 유지하려면 다음을 입력합니다.

```
nmcli general logging level KEEP domains DOMAIN:LEVEL,DOMAIN:LEVEL
```

#### 49.4. NETWORKMANAGER 로그 보기

문제 해결을 위해 **NetworkManager** 로그를 볼 수 있습니다.

##### 절차

- 로그를 보려면 다음을 입력합니다.

```
journalctl -u NetworkManager -b
```

##### 추가 리소스

- **NetworkManager.conf(5)** 도움말 페이지.
- **journalctl** 도움말 페이지.

## 50장. 네트워크 패킷 캡처

네트워크 문제 및 통신을 디버깅하기 위해 네트워크 패킷을 캡처할 수 있습니다. 다음 섹션에서는 네트워크 패킷 캡처에 대한 지침과 추가 정보를 제공합니다.

### 50.1. XDPDUMP를 사용하여 XDP 프로그램에서 삭제한 패킷을 포함하여 네트워크 패킷 캡처

**xdpdump** 유틸리티는 네트워크 패킷을 캡처합니다. **tcpdump** 유틸리티와 달리 **xdpdump** 는 이 작업에 대해 확장된 **Berkeley Packet Filter(eBPF)** 프로그램을 사용합니다. 이를 통해 **xdpdump** 는 **Express Data Path(XDP)** 프로그램에서 삭제된 패킷을 캡처할 수 있습니다. **tcpdump** 와 같은 사용자 공간 유틸리티는 삭제된 패키지와 **XDP** 프로그램에서 수정한 원본 패킷을 캡처할 수 없습니다.

**xdpdump** 를 사용하여 이미 인터페이스에 연결된 **XDP** 프로그램을 디버깅할 수 있습니다. 따라서 유틸리티는 **XDP** 프로그램이 시작되기 전에 패킷을 캡처하고 완료한 후에 캡처할 수 있습니다. 후자의 경우 **xdpdump** 도 **XDP** 작업을 캡처합니다. 기본적으로 **xdpdump** 는 **XDP** 프로그램의 항목에 들어오는 패킷을 캡처합니다.

#### 중요

**AMD** 및 **Intel** 64비트 이외의 아키텍처에서는 **xdpdump** 유틸리티가 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(**SLA**)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있으며 프로덕션에는 사용하지 않는 것이 좋습니다. 이러한 미리보기를 통해 향후 제품 기능에 조기에 액세스할 수 있어 고객이 개발 과정에서 기능을 테스트하고 피드백을 제공할 수 있습니다.

[기술 프리뷰 기능에 대한 지원](#) 범위에 대한 자세한 내용은 **Red Hat** 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

**xdpdump** 에는 패킷 필터 또는 디코딩 기능이 없습니다. 그러나 패킷 디코딩을 위해 **tcpdump** 와 함께 사용할 수 있습니다.

이 절차에서는 **enp1s0** 인터페이스의 모든 패킷을 캡처하여 **/root/capture.pcap** 파일에 기록하는 방법을 설명합니다.

#### 사전 요구 사항



**XDP** 프로그램을 지원하는 네트워크 드라이버입니다.

- **XDP 프로그램이 enp1s0 인터페이스에 로드됩니다.** 프로그램이 로드되지 않은 경우 **xdpdump** 는 이전 버전과의 호환성을 위해 **tcpdump** 가 수행하는 유사한 방식으로 패킷을 캡처합니다.

## 절차

1. **enp1s0 인터페이스에서 패킷을 캡처하여 /root/capture.pcap 파일에 기록하려면 다음을 입력합니다.**

```
xdpdump -i enp1s0 -w /root/capture.pcap
```

2. 패킷 캡처를 중지하려면 **CtrlC**를 누릅니다.

## 추가 리소스

- **xdpdump(8) 도움말 페이지**
- 개발자이고 **xdpdump** 의 소스 코드에 관심이 있는 경우 **Red Hat** 고객 포털에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치합니다.

## 50.2. 추가 리소스

- **tcpdump로 네트워크 패킷을 캡처하는 방법은 무엇입니까?**



## 51장. FIREWALLD 사용 및 구성

방화벽은 시스템을 외부로부터 원하지 않는 트래픽으로부터 보호하는 방법입니다. 사용자가 방화벽 규칙 집합을 정의하여 호스트 시스템에서 들어오는 네트워크 트래픽을 제어할 수 있습니다. 이러한 규칙은 들어오는 트래픽을 정렬하고 차단하거나 이를 통해 허용하는 데 사용됩니다.

**firewalld**는 **D-Bus** 인터페이스와 함께 동적 사용자 지정 호스트 기반 방화벽을 제공하는 방화벽 서비스 데몬입니다. 동적이기 때문에 규칙이 변경될 때마다 방화벽 데몬을 다시 시작할 필요 없이 규칙을 생성, 변경 및 삭제할 수 있습니다.

**firewalld**는 트래픽 관리를 간소화하는 영역과 서비스의 개념을 사용합니다. 영역은 사전 정의된 규칙 세트입니다. 네트워크 인터페이스 및 소스를 영역에 할당할 수 있습니다. 허용되는 트래픽은 컴퓨터가 연결된 네트워크에 따라 달라지며 이 네트워크가 할당된 보안 수준입니다. 방화벽 서비스는 특정 서비스에 대해 들어오는 트래픽을 허용하기 위해 필요한 모든 설정을 포괄하고 영역 내에서 적용되는 사전 정의된 규칙입니다.

서비스는 네트워크 통신에 하나 이상의 포트 또는 주소를 사용합니다. 방화벽은 포트를 기반으로 통신을 필터링합니다. 서비스에 대한 네트워크 트래픽을 허용하려면 포트를 열어야 합니다. **firewalld**는 명시적으로 **open**으로 설정되지 않은 포트의 모든 트래픽을 차단합니다. **trusted**와 같은 일부 영역에서 기본적으로 모든 트래픽을 허용합니다.

**nftables** 백엔드가 있는 **firewalld**는 **--direct** 옵션을 사용하여 사용자 지정 **nftables** 규칙을 **firewalld**에 전달하는 것을 지원하지 않습니다.

### 51.1. FIREWALLD 시작하기

이 섹션에서는 **firewalld**에 대한 정보를 제공합니다.

#### 51.1.1. firewalld, nftables 또는 iptables를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld:** 간단한 방화벽 사용 사례에 **firewalld** 유틸리티를 사용합니다. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables:** **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 등의 복잡하고 성능 중요한 방화벽을 설정합니다.

•

**iptables:** Red Hat Enterprise Linux의 **iptables** 유틸리티는 레거시 백엔드 대신 **the nf\_tables** 커널 API를 사용합니다. **the nf\_tables** API는 **iptables** 명령을 사용하는 스크립트가 여전히 Red Hat Enterprise Linux에서 작동하도록 이전 버전과의 호환성을 제공합니다. 새로운 방화벽 스크립트의 경우 **nftables** 를 사용하는 것이 좋습니다.



#### 중요

서로 다른 방화벽 서비스가 서로 영향을 미치지 않도록 하려면 **RHEL** 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

### 51.1.2. 영역

**firewalld** 는 사용자가 해당 네트워크 내에서 인터페이스와 트래픽을 배치하기로 결정한 신뢰 수준에 따라 네트워크를 다른 영역으로 분리하는 데 사용할 수 있습니다. 연결은 하나의 영역에만 포함될 수 있지만 영역은 많은 네트워크 연결에 사용할 수 있습니다.

**NetworkManager** 는 인터페이스 영역의 **firewalld** 를 알립니다. 다음을 사용하여 인터페이스에 영역을 할당할 수 있습니다.

•

**NetworkManager**

•

**firewall-config** 도구

•

**firewall-cmd** 명령줄 도구

•

**RHEL** 웹 콘솔

세 번째 방법은 적절한 **NetworkManager** 구성 파일만 편집할 수 있습니다. 웹 콘솔, **firewall-cmd** 또는 **firewall-config** 를 사용하여 인터페이스의 영역을 변경하는 경우 요청은 **NetworkManager** 로 전달되며 **firewalld** 에서 처리되지 않습니다.

사전 정의된 영역은 **/usr/lib/firewalld/zones/** 디렉토리에 저장되며 사용 가능한 모든 네트워크 인터페이스에 즉시 적용할 수 있습니다. 이러한 파일은 수정된 후에만 **/etc/firewalld/zones/** 디렉토리에 복사됩니다. 사전 정의된 영역의 기본 설정은 다음과 같습니다.

블록

들어오는 네트워크 연결은 IPv4의 경우 IPv4 및 icmp6-adm-prohibited의 icmp-host-prohibited 메시지와 함께 거부됩니다. 시스템 내에서 시작된 네트워크 연결만 가능합니다.

#### dmz

내부 네트워크에 대한 제한된 액세스 권한으로 공개적으로 액세스할 수 있는 비유형 영역에 있는 컴퓨터. 선택한 연결만 허용됩니다.

#### drop

들어오는 네트워크 패킷은 알림 없이 삭제됩니다. 나가는 네트워크 연결만 가능합니다.

#### external

특히 라우터의 경우 마스커레이딩이 활성화된 외부 네트워크에서 사용합니다. 네트워크의 다른 컴퓨터는 컴퓨터가 손상되지 않도록 신뢰하지 않습니다. 선택한 연결만 허용됩니다.

#### 홈

주로 네트워크의 다른 컴퓨터를 신뢰하는 가정에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

#### internal

네트워크의 다른 컴퓨터를 대부분 신뢰하는 경우 내부 네트워크에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

#### public

네트워크의 다른 컴퓨터를 신뢰하지 않는 공공 부문의 경우. 선택한 연결만 허용됩니다.

#### trusted

모든 네트워크 연결이 허용됩니다.

#### work

네트워크의 다른 컴퓨터를 대부분 신뢰하는 직장에서 사용할 수 있습니다. 선택한 연결만 허용됩니다.

이러한 영역 중 하나는 기본 영역으로 설정됩니다. 인터페이스 연결이 NetworkManager에 추가되면 기본 영역에 할당됩니다. 설치 시 firewalld의 기본 영역이 퍼블릭 영역으로 설정됩니다. 기본 영역은 변경할 수 있습니다.



## 참고

네트워크 영역 이름은 쉽게 설명해야 하며 사용자가 합리적인 결정을 신속하게 내릴 수 있도록 해야 합니다. 보안 문제를 방지하려면 기본 영역 구성을 검토하고 요구 사항 및 위험 평가에 따라 불필요한 서비스를 비활성화합니다.

## 추가 리소스

- **firewalld.zone(5)** 도움말 페이지.

## 51.1.3. 사전 정의된 서비스

서비스는 로컬 포트, 프로토콜, 소스 포트 및 대상의 목록일 뿐만 아니라 서비스가 활성화된 경우 자동으로 로드되는 방화벽 도우미 모듈 목록입니다. 서비스를 사용하면 사용자는 포트 열기, 프로토콜 정의, 패킷 전달 활성화 등의 여러 작업을 한 단계로 수행할 수 있으므로 사용자는 시간을 절약할 수 있습니다.

서비스 구성 옵션과 일반 파일 정보는 **firewalld.service(5)** 도움말 페이지에 설명되어 있습니다. 서비스는 다음과 같은 형식으로 이름이 지정된 개별 XML 구성 파일인 **service-name.xml** 을 통해 지정됩니다. 프로토콜 이름은 **firewalld** 의 서비스 또는 애플리케이션 이름보다 우선합니다.

그래픽 **firewall-config** 도구, **firewall-cmd** 및 **firewall-offline-cmd** 를 사용하여 서비스를 추가하고 제거할 수 있습니다.

또는 **/etc/firewalld/services/** 디렉터리에서 XML 파일을 편집할 수도 있습니다. 사용자가 서비스를 추가하거나 변경하지 않은 경우 **/etc/firewalld/services/** 에 해당 XML 파일이 없습니다. 서비스를 추가하거나 변경하려면 **/usr/lib/firewalld/services/** 디렉터리의 파일을 템플릿으로 사용할 수 있습니다.

## 추가 리소스

- **firewalld.service(5)** 도움말 페이지

## 51.1.4. firewalld 시작

## 절차

1. **firewalld** 를 시작하려면 **root** 로 다음 명령을 입력합니다:

```
systemctl unmask firewalld
systemctl start firewalld
```

2.

시스템 시작 시 **firewalld** 가 자동으로 시작되도록 하려면 **root** 로 다음 명령을 입력합니다.

```
systemctl enable firewalld
```

### 51.1.5. firewalld 중지

#### 절차

1.

**firewalld** 를 중지하려면 **root** 로 다음 명령을 입력합니다:

```
systemctl stop firewalld
```

2.

시스템을 시작할 때 **firewalld** 가 자동으로 시작되지 않도록 하려면 다음을 수행합니다.

```
systemctl disable firewalld
```

3.

**firewalld D-Bus** 인터페이스에 액세스하고 다른 서비스에 **firewalld** 가 필요한 경우에도 **firewalld** 가 시작되지 않도록 하려면 :

```
systemctl mask firewalld
```

### 51.1.6. 영구 firewalld 구성 확인

예를 들어, **firewalld** 구성 파일을 수동으로 편집한 후 관리자가 변경 사항이 올바른지 확인하려고 합니다. 이 섹션에서는 **firewalld** 서비스의 영구 구성을 확인하는 방법에 대해 설명합니다.

#### 사전 요구 사항

•

**firewalld** 서비스가 실행 중입니다.

#### 절차

1.

**firewalld** 서비스의 영구 구성을 확인합니다.

```
firewall-cmd --check-config
success
```

영구 구성이 유효한 경우 명령에서 성공이 반환됩니다. 다른 경우 명령은 다음과 같은 추가 세부 정보와 함께 오류를 반환합니다.

```
firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
```

## 51.2. FIREWALLD의 현재 상태 및 설정 보기

이 섹션에서는 **firewalld**의 현재 상태, 허용된 서비스 및 현재 설정에 대한 정보를 다룹니다.

### 51.2.1. firewalld의 현재 상태 보기

방화벽 서비스 **firewalld**는 기본적으로 시스템에 설치됩니다. **firewalld CLI** 인터페이스를 사용하여 서비스가 실행 중인지 확인합니다.

#### 절차

1. 서비스를 상태를 보려면 다음을 수행합니다.

```
firewall-cmd --state
```

2. 서비스 상태에 대한 자세한 내용은 **systemctl status** 하위 명령을 사용합니다.

```
systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
Docs: man:firewalld(1)
Main PID: 705 (firewalld)
Tasks: 2 (limit: 4915)
CGroup: /system.slice/firewalld.service
└─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

### 51.2.2. GUI를 사용하여 허용된 서비스 보기

그래픽 **firewall-config** 도구를 사용하여 서비스 목록을 보려면 **Super** 키를 눌러 **Activities Overview**(활동 개요)를 입력하고 방화벽 을 입력한 다음 **Enter** 키를 누릅니다. **firewall-config** 도구가 나

타납니다. 이제 **Services** (서비스) 탭에서 서비스 목록을 볼 수 있습니다.

명령줄을 사용하여 그래픽 방화벽 구성 도구를 시작할 수 있습니다.

사전 요구 사항

- **firewall-config** 패키지를 설치했습니다.

절차

- 명령줄을 사용하여 그래픽 방화벽 구성 도구를 시작하려면 다음을 수행합니다.

```
$ firewall-config
```

**Firewall Configuration**(방화벽 구성) 창이 열립니다. 이 명령은 일반 사용자로 실행할 수 있지만, 관리자 암호를 묻는 메시지가 표시되는 경우가 있습니다.

### 51.2.3. CLI를 사용하여 **firewalld** 설정 보기

**CLI** 클라이언트를 사용하면 현재 방화벽 설정을 다양한 볼 수 있습니다. **list -all** 옵션은 **firewalld** 설정에 대한 전체 개요를 표시합니다.

**firewalld** 는 영역을 사용하여 트래픽을 관리합니다. **zone** 옵션으로 영역을 지정하지 않으면 활성 네트워크 인터페이스 및 연결에 할당된 기본 영역에서 명령이 효과적입니다.

절차

- 기본 영역에 대한 모든 관련 정보를 나열하려면 다음을 수행합니다.

```
firewall-cmd --list-all
public
target: default
icmp-block-inversion: no
interfaces:
sources:
services: ssh dhcpv6-client
ports:
protocols:
masquerade: no
forward-ports:
```

**source-ports:**  
**icmp-blocks:**  
**rich rules:**

•

설정을 표시할 영역을 지정하려면 **--zone=zone-name** 인수를 **firewall-cmd --list-all** 명령에 추가합니다. 예를 들면 다음과 같습니다.

```
firewall-cmd --list-all --zone=home
home
target: default
icmp-block-inversion: no
interfaces:
sources:
services: ssh mdns samba-client dhcpv6-client
... [trimmed for clarity]
```

•

서비스 또는 포트와 같은 특정 정보에 대한 설정을 보려면 특정 옵션을 사용합니다. 명령 도움말 페이지를 사용하여 **firewalld** 도움말 페이지를 참조하거나 옵션 목록을 가져옵니다.

```
firewall-cmd --help
```

•

현재 영역에서 허용되는 서비스를 보려면 다음을 수행합니다.

```
firewall-cmd --list-services
ssh dhcpv6-client
```

#### 참고

**CLI** 도구를 사용하여 특정 하위 파트의 설정을 나열하는 경우 해석하기 어려울 수 있습니다. 예를 들어 **SSH** 서비스를 허용하고 **firewalld** 는 서비스에 필요한 포트(22)를 엽니다. 나중에 허용된 서비스를 나열하는 경우 목록에 **SSH** 서비스가 표시되지만 열려 있는 포트를 나열하는 경우 아무것도 표시되지 않습니다. 따라서 **--list-all** 옵션을 사용하여 전체 정보를 받는 것이 좋습니다.

## 51.3. FIREWALLD를 사용하여 네트워크 트래픽 제어

이 섹션에서는 **firewalld** 를 사용하여 네트워크 트래픽을 제어하는 방법에 대해 설명합니다.

### 51.3.1. CLI를 사용하여 긴급한 경우 모든 트래픽 비활성화

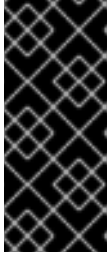
시스템 공격과 같은 긴급 상황에서는 모든 네트워크 트래픽을 비활성화하고 공격자를 차단할 수 있습니다.



## 절차

1. 네트워킹 트래픽을 즉시 비활성화하려면 **에서 패닉 모드를 전환합니다.**

```
firewall-cmd --panic-on
```



## 중요

패닉 모드를 활성화하면 모든 네트워킹 트래픽이 중지됩니다. 따라서 시스템에 대한 물리적 액세스 권한이 있거나 직렬 콘솔을 사용하여 로그인하는 경우에만 사용해야 합니다.

2. 패닉 모드를 끄면 방화벽을 영구 설정으로 되돌립니다. 패닉 모드를 끄려면 다음을 입력합니다.

```
firewall-cmd --panic-off
```

## 검증

- 패닉 모드가 켜져 있는지 또는 꺼져 있는지 여부를 확인하려면 다음을 사용합니다.

```
firewall-cmd --query-panic
```

## 51.3.2. CLI를 사용하여 사전 정의된 서비스로 트래픽 제어

트래픽을 제어하는 가장 간단한 방법은 사전 정의된 서비스를 **firewalld**에 추가하는 것입니다. 그러면 필요한 모든 포트가 열리고 서비스 정의 파일에 따라 다른 설정을 수정합니다.

## 절차

1. 서비스가 아직 허용되지 않은지 확인합니다.

```
firewall-cmd --list-services
ssh dhcpv6-client
```

2. 사전 정의된 모든 서비스를 나열합니다.

```
firewall-cmd --get-services
```

```
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp
dhcpv6 dhcpv6-client dns docker-registry ...
[trimmed for clarity]
```

3.

허용된 서비스에 서비스를 추가합니다.

```
firewall-cmd --add-service=<service-name>
```

4.

새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

### 51.3.3. GUI를 사용하여 사전 정의된 서비스로 트래픽 제어

다음 절차에서는 그래픽 사용자 인터페이스를 사용하여 사전 정의된 서비스로 네트워크 트래픽을 제어하는 방법을 설명합니다.

#### 사전 요구 사항

•

**firewall-config** 패키지를 설치했습니다

#### 절차

1.

사전 정의된 또는 사용자 지정 서비스를 활성화하거나 비활성화하려면 다음을 수행합니다.

a.

**firewall-config** 도구를 시작하고 서비스를 구성할 네트워크 영역을 선택합니다.

b.

영역 탭을 선택한 다음 아래의 서비스 탭을 선택합니다.

c.

신뢰할 수 있는 각 서비스 유형의 확인란을 선택하거나 선택한 영역에서 서비스를 차단하는 확인란을 선택 취소합니다.

2.

서비스를 편집하려면 다음을 수행합니다.

- a. **firewall-config** 도구를 시작합니다.
- b. **Configuration** (구성)이라는 레이블이 지정된 메뉴에서 **Permanent** (영구)를 선택합니다. **Services** (서비스) 창의 아래쪽에 추가 아이콘 및 메뉴 버튼이 표시됩니다.
- c. 구성할 서비스를 선택합니다.

**Ports** (포트), **Protocols** (프로토콜) 및 **Source Port** (소스 포트) 탭을 사용하면 선택한 서비스에 대한 포트, 프로토콜 및 소스 포트를 추가, 변경 및 제거할 수 있습니다. **modules** 탭은 **Netfilter** 도우미 모듈을 구성하는 데 사용됩니다. **Destination** (대상) 탭에서는 특정 대상 주소 및 인터넷 프로토콜(IP4 또는 IPv6)으로 트래픽을 제한할 수 있습니다.



참고

런타임 모드에서 서비스 설정을 변경할 수 없습니다.

#### 51.3.4. 새 서비스 추가

그래픽 **firewall-config** 도구, **firewall-cmd** 및 **firewall-offline-cmd** 를 사용하여 서비스를 추가하고 제거할 수 있습니다. 또는 **/etc/firewalld/services/** 에서 **XML** 파일을 편집할 수도 있습니다. 서비스가 추가 또는 변경되지 않은 경우 **/etc/firewalld/services/** 에 해당 **XML** 파일이 없습니다. 서비스를 추가하거나 변경하려면 **/usr/lib/firewalld/services/** 파일을 템플릿으로 사용할 수 있습니다.



참고

서비스 이름은 영숫자여야 하며, 추가로 **\_** (마이크로어) 및 **-** (대시)만 포함할 수 있습니다.

#### 절차

터미널에서 새 서비스를 추가하려면 **firewalld** 가 활성 상태가 아닌 경우 **firewall-cmd** 또는 **firewall-offline-cmd** 를 사용합니다.

1. 다음 명령을 입력하여 새롭고 빈 서비스를 추가합니다.

```
$ firewall-cmd --new-service=service-name --permanent
```

2.

로컬 파일을 사용하여 새 서비스를 추가하려면 다음 명령을 사용합니다.

```
$ firewall-cmd --new-service-from-file=service-name.xml --permanent
```

추가 **--name=service-name** 옵션을 사용하여 서비스 이름을 변경할 수 있습니다.

3.

서비스 설정이 변경되는 즉시 서비스의 업데이트된 사본이 **/etc/firewalld/services/**에 배치됩니다.

**root**로 다음 명령을 입력하여 서비스를 수동으로 복사할 수 있습니다.

```
cp /usr/lib/firewalld/services/service-name.xml /etc/firewalld/services/service-name.xml
```

**firewalld**는 첫 번째 위치에 있는 **/usr/lib/firewalld/services**에서 파일을 로드합니다. 파일이 **/etc/firewalld/services**에 배치되고 유효한 경우 파일이 **/usr/lib/firewalld/services**에서 일치하는 파일을 재정의합니다. **/usr/lib/firewalld/services**의 재정의된 파일은 **/etc/firewalld/services**에서 일치하는 파일을 제거하거나 **firewalld**가 서비스의 기본값을 로드하라는 요청을 받은 즉시 사용됩니다. 이는 영구 환경에만 적용됩니다. 런타임 환경에서도 이러한 폴백을 가져오려면 다시 로드해야 합니다.

### 51.3.5. GUI를 사용하여 포트 열기

방화벽을 통한 트래픽을 특정 포트에 허용하기 위해 **GUI**에서 포트를 열 수 있습니다.

#### 사전 요구 사항

•

**firewall-config** 패키지를 설치했습니다

#### 절차

1.

**firewall-config** 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.

2.

**Ports(포트)** 탭을 선택하고 오른쪽에 있는 **Add(추가)** 버튼을 클릭합니다. **Port and Protocol(포트 및 프로토콜)** 창이 열립니다.

3.

허용할 포트의 포트 번호 또는 범위를 입력합니다.

4. 목록에서 **tcp** 또는 **udp** 를 선택합니다.

### 51.3.6. GUI를 사용하여 프로토콜로 트래픽 제어

특정 프로토콜을 사용하여 방화벽을 통한 트래픽을 허용하려면 **GUI**를 사용할 수 있습니다.

#### 사전 요구 사항

- **firewall-config** 패키지를 설치했습니다

#### 절차

1. **firewall-config** 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.
2. **Protocols**(프로토콜) 탭을 선택하고 오른쪽에 있는 **Add**(추가) 버튼을 클릭합니다.  
**Protocol**(프로토콜) 창이 열립니다.
3. 목록에서 프로토콜을 선택하거나 **Other Protocol**(기타 프로토콜) 확인란을 선택하고 필드에 프로토콜을 입력합니다.

### 51.3.7. GUI를 사용하여 소스 포트 열기

특정 포트에서 방화벽을 통한 트래픽을 허용하려면 **GUI**를 사용할 수 있습니다.

#### 사전 요구 사항

- **firewall-config** 패키지를 설치했습니다

#### 절차

1. **firewall-config** 도구를 시작하고 변경할 설정의 네트워크 영역을 선택합니다.
2. **Source Port**(소스 포트) 탭을 선택하고 오른쪽에 있는 **Add**(추가) 단추를 클릭합니다.  
**Source Port**(소스 포트) 창이 열립니다.

3.

허용할 포트의 포트 번호 또는 범위를 입력합니다. 목록에서 **tcp** 또는 **udp** 를 선택합니다.

#### 51.4. CLI를 사용하여 포트 제어

포트는 운영 체제에서 네트워크 트래픽을 수신 및 구분하고 이에 따라 시스템 서비스로 전달할 수 있는 논리적 장치입니다. 일반적으로 포트에서 수신 대기하는 데몬으로 표시됩니다. 즉, 이 포트에 들어오는 모든 트래픽을 기다립니다.

일반적으로 시스템 서비스는 예약된 표준 포트에서 수신 대기합니다. 예를 들어 **httpd** 데몬은 포트 **80** 에서 수신 대기합니다. 그러나 기본적으로 시스템 관리자는 보안 또는 기타 이유로 다른 포트에서 수신 대기하도록 데몬을 구성합니다.

##### 51.4.1. 포트 열기

열려 있는 포트를 통해 시스템은 보안 위험을 나타내는 외부에서 액세스할 수 있습니다. 일반적으로 포트를 닫고 특정 서비스에 필요한 경우에만 포트를 엽니다.

#### 절차

현재 영역에서 열려 있는 포트 목록을 가져오려면 다음을 수행합니다.

1.

허용되는 모든 포트를 나열합니다.

```
firewall-cmd --list-ports
```

2.

포트를 허용된 포트에 추가하여 들어오는 트래픽에 대해 엽니다.

```
firewall-cmd --add-port=port-number/port-type
```

포트 유형은 **tcp**, **udp**, **sctp** 또는 **dccp** 입니다. 유형은 네트워크 통신 유형과 일치해야 합니다.

3.

새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

포트 유형은 **tcp,udp,sctp** 또는 **dccp** 입니다. 유형은 네트워크 통신 유형과 일치해야 합니다.

#### 51.4.2. 포트 닫기

열려 있는 포트가 더 이상 필요하지 않은 경우 **firewalld** 에서 해당 포트를 닫습니다. 포트를 열린 상태로 유지하면 보안 위험이 있으므로 사용하지 않는 즉시 모든 불필요한 포트를 종료하는 것이 좋습니다.

##### 절차

포트를 종료하려면 허용된 포트 목록에서 제거합니다.

1. 허용되는 모든 포트를 나열합니다.

```
firewall-cmd --list-ports
```



##### 주의

이 명령은 포트에 열린 포트 목록만 제공합니다. 서비스로 열린 열려 있는 포트는 볼 수 없습니다. 따라서 **--list-ports** 대신 **--list-all** 옵션을 사용해야 합니다.

2. 포트를 허용된 포트에서 제거하여 들어오는 트래픽에 대해 종료합니다.

```
firewall-cmd --remove-port=port-number/port-type
```

3. 새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

#### 51.5. 시스템 역할을 사용하여 포트 구성

**RHEL(Red Hat Enterprise Linux) firewalld** 시스템 역할을 사용하여 들어오는 트래픽을 위해 로컬 방화벽에서 포트를 열거나 닫고 재부팅 시 새 구성을 유지할 수 있습니다. 이 예제에서는 **HTTPS** 서비스에

들어오는 트래픽을 허용하도록 기본 영역을 구성하는 방법을 설명합니다.

**Ansible** 제어 노드에서 이 절차를 실행합니다.

#### 사전 요구 사항

- **firewalld** 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

#### 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 인벤토리되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** **Ansible** 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/adding-and-removing-ports.yml** 플레이북을 만듭니다.

```

- name: Allow incoming HTTPS traffic to the local host
 hosts: node.example.com
 become: true

 tasks:
 - include_role:
 name: linux-system-roles.firewall

 vars:
```



```

firewall:
- port: 443/tcp
 service: http
 state: enabled
 runtime: true
 permanent: true

```

**permanent: true** 옵션을 사용하면 재부팅 시 새 설정이 유지됩니다.

3.

플레이북을 실행합니다.

- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/adding-and-removing-ports.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/adding-and-removing-ports.yml
```

**ask -become-pass** 옵션을 사용하면 **ansible-playbook** 명령이 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

## 검증

1.

관리 노드에 연결합니다.

```
$ ssh user_name@node.example.com
```

2.

**HTTPS** 서비스와 연결된 **443/tcp** 포트가 열려 있는지 확인합니다.

```
$ sudo firewall-cmd --list-ports
443/tcp
```

## 추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md`
- `ansible-playbook(1)` 도움말 페이지

## 51.6. FIREWALLD 영역 작업

영역은 들어오는 트래픽을 보다 투명하게 관리하는 개념을 나타냅니다. 영역은 네트워킹 인터페이스에 연결되거나 다양한 소스 주소가 할당됩니다. 각 영역에 대해 개별적으로 방화벽 규칙을 관리하므로 복잡한 방화벽 설정을 정의하고 트래픽에 적용할 수 있습니다.

### 51.6.1. 영역 나열

다음 절차에서는 명령줄을 사용하여 영역을 나열하는 방법을 설명합니다.

#### 절차

1. 시스템에서 사용할 수 있는 영역을 보려면 다음을 수행합니다.

```
firewall-cmd --get-zones
```

`firewall-cmd --get-zones` 명령은 시스템에서 사용할 수 있는 모든 영역을 표시하지만 특정 영역에 대한 세부 정보는 표시하지 않습니다.

2. 모든 영역에 대한 자세한 정보를 보려면 다음을 수행합니다.

```
firewall-cmd --list-all-zones
```

3. 특정 영역에 대한 자세한 정보를 보려면 다음을 수행합니다.

```
firewall-cmd --zone=zone-name --list-all
```

### 51.6.2. 특정 영역에 대한 `firewalld` 설정 수정

`cli`를 사용하여 사전 정의된 서비스로 트래픽 제어 및 `cli`를 사용하여 포트 제어는 현재 작업 영역의 범위에서 서비스를 추가하거나 포트를 수정하는 방법을 설명합니다. 때로는 다른 영역에 규칙을 설정해야

합니다.

절차

- 다른 영역에서 작업하려면 **--zone=zone-name** 옵션을 사용합니다. 예를 들어 영역 **public** 에서 **SSH** 서비스를 허용하려면 다음을 수행합니다.

```
firewall-cmd --add-service=ssh --zone=public
```

### 51.6.3. 기본 영역 변경

시스템 관리자는 구성 파일의 네트워킹 인터페이스에 영역을 할당합니다. 인터페이스가 특정 영역에 할당되지 않은 경우 기본 영역에 할당됩니다. **firewalld** 서비스를 다시 시작하면 **firewalld** 가 기본 영역에 대한 설정을 로드하여 활성화합니다.

절차

기본 영역을 설정하려면 다음을 수행합니다.

1. 현재 기본 영역을 표시합니다.

```
firewall-cmd --get-default-zone
```

2. 새 기본 영역을 설정합니다.

```
firewall-cmd --set-default-zone zone-name
```



참고

이 절차에 따라 설정은 **--permanent** 옵션이 없어도 영구적인 설정입니다.

### 51.6.4. 영역에 네트워크 인터페이스 할당

사용 중인 인터페이스의 영역을 변경하여 다양한 영역에 대해 다양한 규칙 집합을 정의한 다음 설정을 빠르게 변경할 수 있습니다. 여러 인터페이스를 사용하면 각 인터페이스를 통해 들어오는 트래픽을 구분 하도록 특정 영역을 설정할 수 있습니다.

절차

특정 인터페이스에 영역을 할당하려면 다음을 수행합니다.

1.

활성 영역과 여기에 할당된 인터페이스를 나열합니다.

```
firewall-cmd --get-active-zones
```

2.

인터페이스를 다른 영역에 할당합니다.

```
firewall-cmd --zone=zone_name --change-interface=interface_name --permanent
```

#### 51.6.5. nmcli를 사용하여 연결에 영역 할당

다음 절차에서는 **nmcli** 유틸리티를 사용하여 **NetworkManager** 연결에 **firewalld** 영역을 추가하는 방법을 설명합니다.

##### 절차

1.

**NetworkManager** 연결 프로필에 영역을 할당합니다.

```
nmcli connection modify profile connection.zone zone_name
```

2.

연결을 활성화합니다.

```
nmcli connection up profile
```

#### 51.6.6. ifcfg 파일에서 네트워크 연결에 수동으로 영역 할당

**NetworkManager** 에서 연결을 관리하는 경우 사용하는 영역을 알고 있어야 합니다. 모든 네트워크 연결에 대해 이동식 장치가 있는 컴퓨터의 위치에 따라 다양한 방화벽 설정의 유연성을 제공하는 영역을 지정할 수 있습니다. 따라서 회사 또는 홈과 같은 다양한 위치에 대해 영역 및 설정을 지정할 수 있습니다.

##### 절차

•

연결 영역을 설정하려면 **/etc/sysconfig/network-scripts/ifcfg-connection\_name** 파일을 편집하고 이 연결에 영역을 할당하는 행을 추가합니다.

```
ZONE=zone_name
```

### 51.6.7. 새 영역 생성

사용자 지정 영역을 사용하려면 새 영역을 생성하고 사전 정의된 영역처럼 사용합니다. 새 영역에는 **--permanent** 옵션이 필요합니다. 그렇지 않으면 명령이 작동하지 않습니다.

#### 절차

1. 새 영역을 생성합니다.

```
firewall-cmd --permanent --new-zone=zone-name
```

2. 새 영역이 영구 설정에 추가되었는지 확인합니다.

```
firewall-cmd --get-zones
```

3. 새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

### 51.6.8. 영역 구성 파일

영역은 영역 구성 파일을 사용하여 만들 수도 있습니다. 이 방법은 새 영역을 만들어야 할 때 유용할 수 있지만 다른 영역의 설정을 재사용하고 약간만 변경하려고 합니다.

**firewalld** 영역 구성 파일에는 영역에 대한 정보가 포함되어 있습니다. **XML** 파일 형식의 영역 설명, 서비스, 포트, 프로토콜, **icmp-blocks**, **masquerade**, **forward-ports** 및 리치 언어 규칙입니다. 파일 이름은 현재 **zone-name**의 길이를 17자로 제한하는 **zone-name.xml** 이어야 합니다. 영역 구성 파일은 **/usr/lib/firewalld/zones/** 및 **/etc/firewalld/zones/** 디렉터리에 있습니다.

다음 예제에서는 **TCP** 및 **UDP** 프로토콜 모두에 대해 하나의 서비스(**SSH**) 및 하나의 포트 범위를 허용하는 구성을 보여줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
 <short>My Zone</short>
 <description>Here you can describe the characteristic features of the zone.</description>
 <service name="ssh"/>
```

```
<port protocol="udp" port="1025-65535"/>
<port protocol="tcp" port="1025-65535"/>
</zone>
```

해당 영역의 설정을 변경하려면 섹션을 추가하거나 제거하여 포트, 서비스 등을 추가합니다.

#### 추가 리소스

- [firewalld.zone](#) 도움말 페이지

### 51.6.9. 영역 대상을 사용하여 들어오는 트래픽에 대한 기본 동작 설정

모든 영역에 대해 추가로 지정되지 않은 들어오는 트래픽을 처리하는 기본 동작을 설정할 수 있습니다. 이러한 동작은 영역의 대상을 설정하여 정의됩니다. 네 가지 옵션이 있습니다.

- **수락:** 특정 규칙에 의해 허용되지 않는 패킷을 제외하고 들어오는 모든 패킷을 허용합니다.
- **거부:** 특정 규칙에서 허용하는 패킷을 제외한 들어오는 모든 패킷을 거부합니다. **firewalld** 가 패킷을 거부하면 소스 시스템에 거부에 대한 정보가 표시됩니다.
- **DROP:** 특정 규칙에서 허용하는 패킷을 제외하고 들어오는 모든 패킷을 삭제합니다. **firewalld** 가 패킷을 삭제하면 소스 시스템에 패킷 삭제에 대한 정보가 표시되지 않습니다.
- **default: REJECT** 와 유사한 동작이지만 특정 시나리오에서 특별한 의미가 있습니다. 자세한 내용은 **firewall-cmd(1)** 도움말 페이지의 **Adapt** 및 **Query Zones** 및 **Policies** 섹션을 참조하십시오.

#### 절차

영역의 대상을 설정하려면 다음을 수행합니다.

1. 특정 영역에 대한 정보를 나열하여 기본 대상을 확인합니다.

```
firewall-cmd --zone=zone-name --list-all
```

2. 영역에 새 대상을 설정합니다.

```
firewall-cmd --permanent --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

## 추가 리소스

- [firewall-cmd\(1\) 도움말 페이지](#)

### 51.7. 소스에 따라 영역을 사용하여 들어오는 트래픽 관리

영역을 사용하여 소스에 따라 들어오는 트래픽을 관리할 수 있습니다. 이를 통해 들어오는 트래픽을 정렬하고 다른 영역을 통해 라우팅하여 해당 트래픽으로 연결할 수 있는 서비스를 허용하거나 허용하지 않도록 할 수 있습니다.

소스를 영역에 추가하면 영역이 활성화되고 해당 소스에서 들어오는 트래픽이 이를 통해 전송됩니다. 지정된 소스의 트래픽에 적절하게 적용되는 각 영역에 대해 다른 설정을 지정할 수 있습니다. 하나의 네트워크 인터페이스만 있는 경우에도 더 많은 영역을 사용할 수 있습니다.

#### 51.7.1. 소스 추가

들어오는 트래픽을 특정 영역으로 라우팅하려면 소스를 해당 영역에 추가합니다. 소스는 IP 주소이거나 클래스 없는 도메인 간 라우팅(CIDR) 표기법의 IP 마스크일 수 있습니다.



#### 참고

중복되는 네트워크 범위를 사용하여 여러 영역을 추가하는 경우 영역 이름으로 영숫자로 정렬되며 첫 번째 영역만 고려됩니다.

- 현재 영역에 소스를 설정하려면 다음을 수행합니다.

```
firewall-cmd --add-source=<source>
```

- 특정 영역의 소스 IP 주소를 설정하려면 다음을 수행합니다.

```
firewall-cmd --zone=zone-name --add-source=<source>
```

다음 절차에서는 신뢰할 수 있는 영역의 192.168.2.15 에서 들어오는 모든 트래픽을 허용합니다.

### 절차

1. 사용 가능한 모든 영역을 나열합니다.

```
firewall-cmd --get-zones
```

2. 영구 모드에서 소스 IP를 신뢰할 수 있는 영역에 추가합니다.

```
firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. 새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

#### 51.7.2. 소스 제거

영역에서 소스를 제거하면 소스에서 들어오는 트래픽이 줄어듭니다.

### 절차

1. 필수 영역에 허용된 소스를 나열합니다.

```
firewall-cmd --zone=zone-name --list-sources
```

2. 영역에서 소스를 영구적으로 제거합니다.

```
firewall-cmd --zone=zone-name --remove-source=<source>
```

3. 새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

#### 51.7.3. 소스 포트 추가

원본 포트에 따라 트래픽을 정렬할 수 있도록 하려면 **--add-source-port** 옵션을 사용하여 소스 포트를 지정합니다. 이 옵션을 **--add-source** 옵션과 결합하여 트래픽을 특정 IP 주소 또는 IP 범위로 제한할 수도



있습니다.

#### 절차

- 소스 포트를 추가하려면 다음을 수행합니다.

```
firewall-cmd --zone=zone-name --add-source-port=<port-name>/<tcp/udp/sctp/dccp>
```

#### 51.7.4. 소스 포트 제거

소스 포트를 제거하면 원본 포트에 따라 트래픽 정렬을 비활성화합니다.

#### 절차

- 소스 포트를 제거하려면 다음을 수행합니다.

```
firewall-cmd --zone=zone-name --remove-source-port=<port-name>/<tcp/udp/sctp/dccp>
```

#### 51.7.5. 특정 도메인에만 서비스를 허용하려면 영역 및 소스를 사용합니다.

특정 네트워크의 트래픽이 시스템에서 서비스를 사용하도록 허용하려면 영역 및 소스를 사용합니다. 다음 절차에서는 다른 트래픽이 차단되는 동안 **192.0.2.0/24** 네트워크에서 **HTTP** 트래픽만 허용합니다.



#### 주의

이 시나리오를 구성할 때 기본 타겟이 있는 영역을 사용합니다. 타겟이 **ACCEPT**로 설정된 영역을 사용하면 **192.0.2.0/24**의 트래픽에 대해 모든 네트워크 연결이 수락되기 때문에 보안 위험이 있습니다.

#### 절차

1. 사용 가능한 모든 영역을 나열합니다.

```
firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

2.

IP 범위를 내부 영역에 추가하여 소스에서 시작된 트래픽을 영역을 통해 라우팅합니다.

```
firewall-cmd --zone=internal --add-source=192.0.2.0/24
```

3.

http 서비스를 내부 영역에 추가합니다.

```
firewall-cmd --zone=internal --add-service=http
```

4.

새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

## 검증

•

내부 영역이 활성화되어 있고 서비스가 허용되는지 확인합니다.

```
firewall-cmd --zone=internal --list-all
internal (active)
target: default
icmp-block-inversion: no
interfaces:
sources: 192.0.2.0/24
services: cockpit dhcpv6-client mdns samba-client ssh http
...
```

## 추가 리소스

•

[firewalld.zones\(5\) 도움말 페이지](#)

## 51.8. 영역 간 전달된 트래픽 필터링

정책 오브젝트를 사용하면 정책에서 유사한 권한이 필요한 다른 ID를 그룹화할 수 있습니다. 트래픽 방향에 따라 정책을 적용할 수 있습니다.

정책 개체 기능은 **firewalld**에서 정방향 및 출력 필터링을 제공합니다. 다음은 다른 영역 간 트래픽을 필터링하여 로컬 호스팅 VM에 액세스하여 호스트를 연결할 수 있도록 **firewalld**를 사용하는 방법을 설명합

니다.

### 51.8.1. 정책 오브젝트와 영역 간의 관계

정책 개체를 사용하면 사용자가 서비스, 포트, 리치 규칙과 같은 **firewalld**의 기본 규칙을 정책에 연결할 수 있습니다. 정책 오브젝트를 상태 저장 및 단방향 방식으로 영역 간에 전달하는 트래픽에 적용할 수 있습니다.

```
firewall-cmd --permanent --new-policy myOutputPolicy
```

```
firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST
```

```
firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

**HOST** 및 **ALL** 은 수신 및 송신 영역 목록에 사용되는 심볼릭 영역입니다.

- **HOST(호스트)** 심볼릭 영역을 사용하면 **firewalld**를 실행하는 호스트의 대상이거나 에서 시작된 트래픽에 대한 정책을 사용할 수 있습니다.
- 모든 심볼릭 영역은 모든 현재 및 향후 영역에 정책을 적용합니다. 모든 심볼릭 영역은 모든 영역에 대해 와일드카드 역할을 합니다.

### 51.8.2. 우선순위를 사용하여 정책 정렬

여러 정책이 동일한 트래픽 집합에 적용할 수 있으므로 적용할 수 있는 정책에 대한 우선 순위 순서를 생성하려면 우선 순위를 사용해야 합니다.

정책을 정렬하는 우선 순위를 설정하려면 다음을 수행합니다.

```
firewall-cmd --permanent --policy mypolicy --set-priority -500
```

위의 예에서 **-500** 은 우선 순위가 낮지만 우선 순위가 높습니다. 따라서 **-500**은 **-100** 전에 실행됩니다. 우선 순위가 높은 값은 더 낮은 값보다 우선합니다.

다음 규칙은 정책 우선 순위에 적용됩니다.

- 우선순위가 음수인 정책은 영역의 규칙보다 먼저 적용됩니다.
- 우선순위가 양당하는 정책은 영역의 규칙 다음에 적용됩니다.
- 우선 순위 0은 예약되므로 사용할 수 없습니다.

### 51.8.3. 정책 오브젝트를 사용하여 로컬 호스트 컨테이너와 호스트에 물리적으로 연결된 네트워크 간 트래픽 필터링

정책 오브젝트 기능을 사용하면 사용자가 컨테이너 및 가상 시스템 트래픽을 필터링할 수 있습니다.

#### 절차

1. 새 정책을 만듭니다.

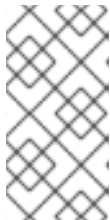
```
firewall-cmd --permanent --new-policy podmanToHost
```

2. 모든 트래픽을 차단합니다.

```
firewall-cmd --permanent --policy podmanToHost --set-target REJECT
```

```
firewall-cmd --permanent --policy podmanToHost --add-service dhcp
```

```
firewall-cmd --permanent --policy podmanToHost --add-service dns
```



#### 참고

기본적으로 호스트에 대한 모든 트래픽을 차단한 다음 호스트에 필요한 서비스를 선택적으로 여는 것이 좋습니다.

3. 정책과 함께 사용할 수신 영역을 정의합니다.

```
firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

4. 정책과 함께 사용할 송신 영역을 정의합니다.

```
firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

검증

- 정책에 대한 정보를 확인합니다.

```
firewall-cmd --info-policy podmanToHost
```

#### 51.8.4. 정책 오브젝트의 기본 대상 설정

정책에 대해 **--set-target** 옵션을 지정할 수 있습니다. 다음 대상을 사용할 수 있습니다.

- **ACCEPT** - 패킷 수락
- **DROP** - 원하지 않는 패킷을 삭제합니다
- **REJECT** - ICMP 응답이 포함된 원하지 않는 패킷 거부
- **CONTINUE(기본값)** - 패킷은 다음 정책과 영역의 규칙이 적용됩니다.

```
firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

검증

- 정책에 대한 정보 확인

```
firewall-cmd --info-policy mypolicy
```

#### 51.9. FIREWALLD를 사용하여 NAT 구성

**firewalld** 를 사용하면 다음 **NAT**(네트워크 주소 변환) 유형을 구성할 수 있습니다.

- **마스커레이딩**

- **SNAT(소스 NAT)**
- **대상 NAT(DNAT)**
- **리디렉션**

### 51.9.1. 다양한 NAT 유형: 마스커레이드, 소스 NAT, 대상 NAT, 리디렉션

다음은 다양한 NAT(네트워크 주소 변환) 유형입니다.

#### 마스커레이딩 및 소스 NAT (SNAT)

이러한 NAT 유형 중 하나를 사용하여 패킷의 소스 IP 주소를 변경합니다. 예를 들어 인터넷 서비스 공급자는 10.0.0.0/8 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하여 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 해당 범위에서 패킷의 소스 IP 주소를 공용 IP 주소로 매핑합니다.

마스커레이딩과 SNAT 모두 매우 유사합니다. 차이점은 다음과 같습니다.

- 마스커레이딩은 나가는 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 나가는 인터페이스에서 동적 IP 주소를 사용하는 경우 마스커레이딩을 사용합니다.
- SNAT는 패킷의 소스 IP 주소를 지정된 IP로 설정하고 나가는 인터페이스의 IP를 동적으로 조회하지 않습니다. 따라서 SNAT는 마스커레이딩보다 빠릅니다. 나가는 인터페이스에서 고정 IP 주소를 사용하는 경우 SNAT를 사용합니다.

#### 대상 NAT(DNAT)

이 NAT 유형을 사용하여 들어오는 패킷의 대상 주소와 포트를 다시 작성합니다. 예를 들어 웹 서버가 개인 IP 범위의 IP 주소를 사용하므로 인터넷에서 직접 액세스할 수 없는 경우 라우터에 DNAT 규칙을 설정하여 들어오는 트래픽을 이 서버로 리디렉션할 수 있습니다.

#### 리디렉션

이 유형은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 DNAT의 특별한 사례입니다. 예를 들어 서비스가 표준 포트와 다른 포트에서 실행되는 경우 표준 포트에서 들어오는 트래픽을 이 특정 포트로 리디렉션할 수 있습니다.

### 51.9.2. IP 주소 마스커레이딩 구성

다음 절차에서는 시스템에서 IP 마스커레이딩을 활성화하는 방법을 설명합니다. IP 마스커레이딩은 인터넷에 액세스할 때 게이트웨이 뒤의 개별 시스템을 숨깁니다.

## 절차

1.

IP 마스커레이드가 활성화되어 있는지 확인하려면 (예: 외부 영역의 경우) 다음 명령을 root 로 입력합니다.

```
firewall-cmd --zone=external --query-masquerade
```

이 명령은 활성화된 경우 종료 상태 0 으로 yes 를 출력합니다. 그렇지 않으면 종료 상태 1 로 no 를 출력합니다. zone(영역)을 생략하면 기본 영역이 사용됩니다.

2.

IP 마스커레이딩을 사용하려면 root 로 다음 명령을 입력합니다.

```
firewall-cmd --zone=external --add-masquerade
```

3.

이 설정을 영구적으로 설정하려면 명령에 --permanent 옵션을 전달합니다.

4.

IP 마스커레이딩을 비활성화하려면 root 로 다음 명령을 입력합니다.

```
firewall-cmd --zone=external --remove-masquerade
```

이 설정을 영구적으로 만들려면 --permanent 옵션을 명령에 전달합니다.

## 51.10. 포트 전달

이 메서드를 사용하여 포트를 리디렉션하는 것은 IPv4 기반 트래픽에서만 작동합니다. IPv6 리디렉션 설정의 경우 리치 규칙을 사용해야 합니다.

외부 시스템으로 리디렉션하려면 마스커레이딩을 활성화해야 합니다. 자세한 내용은 [IP 주소 마스커레이딩 구성을 참조하십시오](#).



## 참고

로컬 전달을 구성한 호스트에서 리디렉션된 포트를 통해 서비스에 액세스할 수 없습니다.

### 51.10.1. 리디렉션할 포트 추가

**firewalld** 를 사용하면 시스템의 특정 포트에 도달하는 들어오는 트래픽이 선택한 다른 내부 포트 또는 다른 시스템의 외부 포트에 전달되도록 포트 리디렉션을 설정할 수 있습니다.

#### 사전 요구 사항

- 한 포트에서 다른 포트 또는 다른 주소로 트래픽을 리디렉션하기 전에 패킷이 도착하는 포트, 사용되는 프로토콜, 리디렉션하려는 세 가지 사항을 알아야 합니다.

#### 절차

1.

포트를 다른 포트에 리디렉션하려면 다음을 수행합니다.

```
firewall-cmd --add-forward-port=port=port-number:proto=tcp|udp|sctp|dccp:toport=port-number
```

2.

다른 IP 주소에서 다른 포트에 포트를 리디렉션하려면 다음을 수행합니다.

a.

전달할 포트를 추가합니다.

```
firewall-cmd --add-forward-port=port=port-number:proto=tcp|udp:toport=port-number:toaddr=IP
```

b.

마스커레이드를 활성화합니다.

```
firewall-cmd --add-masquerade
```

### 51.10.2. 동일한 머신에서 TCP 포트 80을 포트 88으로 리디렉션

단계에 따라 TCP 포트 80을 포트 88으로 리디렉션합니다.



**절차**

1. 포트 80을 TCP 트래픽의 포트 88로 리디렉션합니다.

```
firewall-cmd --add-forward-port=port=80:proto=tcp:toport=88
```

2. 새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

3. 포트가 리디렉션되는지 확인합니다.

```
firewall-cmd --list-all
```

**51.10.3. 리디렉션된 포트 제거**

다음 절차에서는 리디렉션된 포트를 제거하는 방법을 설명합니다.

**절차**

1. 리디렉션된 포트를 제거하려면 다음을 수행합니다.

```
firewall-cmd --remove-forward-port=port=port-number:proto=<tcp/udp>:toport=port-number:toaddr=<IP>
```

2. 다른 주소로 리디렉션되는 전달된 포트를 제거하려면 다음을 수행합니다.

- a. 전달된 포트를 제거합니다.

```
firewall-cmd --remove-forward-port=port=port-number:proto=<tcp/udp>:toport=port-number:toaddr=<IP>
```

- b. 마스커레이드를 비활성화합니다.

```
firewall-cmd --remove-masquerade
```

**51.10.4. 동일한 머신의 포트 88으로 전달된 TCP 포트 80 제거**

다음 절차에서는 포트 리디렉션을 제거하는 방법을 설명합니다.

#### 절차

1.

리디렉션된 포트를 나열합니다.

```
~]# firewall-cmd --list-forward-ports
port=80:proto=tcp:toport=88:toaddr=
```

2.

방화벽에서 리디렉션된 포트를 제거합니다.

```
~]# firewall-cmd --remove-forward-port=port=80:proto=tcp:toport=88:toaddr=
```

3.

새 설정을 영구적으로 설정합니다.

```
~]# firewall-cmd --runtime-to-permanent
```

### 51.11. ICMP 요청 관리

**ICMP( Internet Control Message Protocol )**는 다양한 네트워크 장치에서 오류 메시지 및 연결 문제를 나타내는 운영 정보를 전송하는 데 사용하는 지원 프로토콜입니다(예: 요청된 서비스를 사용할 수 없음). **ICMP**는 시스템 간에 데이터를 교환하는 데 사용되지 않으므로 **TCP** 및 **UDP**와 같은 전송 프로토콜과 다릅니다.

안타깝게도 **ICMP** 메시지, 특히 에코 요청 및 에코를 사용하여 네트워크에 대한 정보를 공개하고 다양한 유형의 사기 활동에 이러한 정보를 오용할 수 있습니다. 따라서 **firewalld**를 사용하면 **ICMP** 요청을 차단하여 네트워크 정보를 보호할 수 있습니다.

#### 51.11.1. ICMP 요청 나열 및 차단

##### ICMP 요청 나열

**ICMP** 요청은 **/usr/lib/firewalld/icmptypes/** 디렉토리에 있는 개별 **XML** 파일에 설명됩니다. 이러한 파일을 읽고 요청에 대한 설명을 확인할 수 있습니다. **firewall-cmd** 명령은 **ICMP** 요청 조작을 제어합니다.

- 

사용 가능한 모든 **ICMP** 유형을 나열하려면 다음을 수행합니다.

```
firewall-cmd --get-icmp-types
```

•

ICMP 요청은 IPv4, IPv6 또는 두 프로토콜 모두에서 사용할 수 있습니다. ICMP 요청이 사용된 프로토콜을 보려면 다음을 수행합니다.

```
firewall-cmd --info-icmp-type=<icmp-type>
```

•

ICMP 요청 상태는 요청이 현재 차단되었거나 그렇지 않은 경우 **no** 인 경우 **yes** 를 표시합니다. ICMP 요청이 현재 차단되었는지 확인하려면 다음을 실행합니다.

```
firewall-cmd --query-icmp-block=<icmp-type>
```

### ICMP 요청 차단 또는 차단 해제

서버에서 ICMP 요청을 차단하면 일반적으로 수행하는 정보를 제공하지 않습니다. 그러나 이는 정보가 전혀 제공되지 않음을 의미하지는 않습니다. 클라이언트는 특정 ICMP 요청이 차단되고 있음을 수신합니다(거부됨). ICMP 요청을 차단하면 특히 IPv6 트래픽에서 통신 문제가 발생할 수 있으므로 주의해서 고려해야 합니다.

•

ICMP 요청이 현재 차단되었는지 확인하려면 다음을 실행합니다.

```
firewall-cmd --query-icmp-block=<icmp-type>
```

•

ICMP 요청을 차단하려면 다음을 수행합니다.

```
firewall-cmd --add-icmp-block=<icmp-type>
```

•

ICMP 요청에 대한 블록을 제거하려면 다음을 수행합니다.

```
firewall-cmd --remove-icmp-block=<icmp-type>
```

### 정보를 제공하지 않고 ICMP 요청 차단

일반적으로 ICMP 요청을 차단하면 클라이언트가 차단하고 있음을 알 수 있습니다. 따라서 실시간 IP 주소를 스니핑하는 잠재적인 공격자는 IP 주소가 온라인 상태임을 계속 확인할 수 있습니다. 이 정보를 완전히 숨기려면 ICMP 요청을 모두 삭제해야 합니다.

•

모든 ICMP 요청을 차단하고 삭제하려면 다음을 수행합니다.

•

영역의 대상을 **DROP** 으로 설정합니다.

```
firewall-cmd --permanent --set-target=DROP
```

이제 명시적으로 허용된 트래픽을 제외하고 **ICMP** 요청을 포함한 모든 트래픽이 삭제됩니다.

특정 **ICMP** 요청을 차단 및 삭제하고 다른 **ICMP** 요청을 허용하려면 다음을 수행합니다.

1.

영역의 대상을 **DROP** 으로 설정합니다.

```
firewall-cmd --permanent --set-target=DROP
```

2.

**ICMP** 블록 인버전을 추가하여 모든 **ICMP** 요청을 한 번에 차단합니다.

```
firewall-cmd --add-icmp-block-inversion
```

3.

허용하려는 **ICMP** 요청에 대한 **ICMP** 블록을 추가합니다.

```
firewall-cmd --add-icmp-block=<icmptype>
```

4.

새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

블록 **inversion** 은 **ICMP** 요청의 설정을 변환하므로 이전에 차단되지 않은 모든 요청이 **DROP** 으로 변경되므로 영역의 대상이 차단되므로 차단됩니다. 차단된 요청은 차단되지 않습니다. 즉, 요청 차단을 해제하려면 차단 명령을 사용해야 합니다.

블록 인버전을 완전히 허용 설정으로 되돌리려면 다음을 수행합니다.

1.

영역의 대상을 기본값 또는 **ACCEPT** 로 설정합니다.

```
firewall-cmd --permanent --set-target=default
```

2.

ICMP 요청에 대해 추가된 모든 블록을 제거합니다.

```
firewall-cmd --remove-icmp-block=<icmptype>
```

3.

ICMP 블록 인버전을 제거합니다.

```
firewall-cmd --remove-icmp-block-inversion
```

4.

새 설정을 영구적으로 설정합니다.

```
firewall-cmd --runtime-to-permanent
```

### 51.11.2. GUI를 사용하여 ICMP 필터 구성

•

ICMP 필터를 활성화하거나 비활성화하려면 **firewall-config** 도구를 시작하고 메시지가 필터링될 네트워크 영역을 선택합니다. **ICMP Filter(ICMP 필터)** 탭을 선택하고 필터링할 각 ICMP 메시지 유형의 확인란을 선택합니다. 확인란을 지워 필터를 비활성화합니다. 이 설정은 방향에 따라 설정되며 기본값은 모든 것을 허용합니다.

•

ICMP 필터를 전환하려면 오른쪽에 있는 **Invert Filter (검색 필터)** 확인란을 클릭합니다. 표시된 ICMP 유형만 허용되고 다른 모든 유형이 거부됩니다. **DROP** 대상을 사용하는 영역에서는 해당 대상이 삭제됩니다.

### 51.12. FIREWALLD를 사용하여 IP 세트 설정 및 제어

**firewalld** 에서 지원하는 IP 세트 유형 목록을 보려면 **root**로 다음 명령을 입력합니다.

```
~]# firewall-cmd --get-ipset-types
hash:ip hash:ip,mark hash:ip,port hash:ip,port,ip hash:ip,port,net hash:mac hash:net hash:net,iface
hash:net,net hash:net,port hash:net,port,net
```

#### 51.12.1. CLI를 사용하여 IP 세트 옵션 구성

IP 세트는 **firewalld** 영역에서 소스로, 리치 규칙의 소스로도 사용할 수 있습니다. **Red Hat Enterprise Linux**에서 선호되는 방법은 직접 규칙에 **firewalld** 로 만든 IP 세트를 사용하는 것입니다.

•

영구 환경에서 **firewalld** 로 알려진 IP 세트를 나열하려면 다음 명령을 **root**로 사용하십시오.

```
firewall-cmd --permanent --get-ipsets
```

- 새 IP 세트를 추가하려면 영구 환경을 **root** 로 사용하여 다음 명령을 사용하십시오.

```
firewall-cmd --permanent --new-ipset=test --type=hash:net
success
```

이전 명령은 이름 **test** 및 IPv4 의 **hash:net** 유형으로 새 IP 집합을 생성합니다. IPv6 에 사용할 IP 세트를 만들려면 **--option=family=inet6** 옵션을 추가합니다. 런타임 환경에서 새 설정을 적용하려면 **firewalld** 를 다시 로드합니다.

- 다음 명령을 사용하여 **root** 로 새 IP 세트를 나열합니다.

```
firewall-cmd --permanent --get-ipsets
test
```

- IP 세트에 대한 자세한 정보를 얻으려면 **root** 로 다음 명령을 사용하십시오.

```
firewall-cmd --permanent --info-ipset=test
test
type: hash:net
options:
entries:
```

현재 IP 세트에는 항목이 없습니다.

- 테스트 IP 세트에 항목을 추가하려면 **root** 로 다음 명령을 사용하십시오.

```
firewall-cmd --permanent --ipset=test --add-entry=192.168.0.1
success
```

이전 명령은 IP 주소 **192.168.0.1** 을 IP 집합에 추가합니다.

- IP 세트의 현재 항목 목록을 가져오려면 다음 명령을 **root** 로 사용하십시오.

```
firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

**IP 주소 목록이 포함된 파일을 생성합니다. 예를 들면 다음과 같습니다.**

```
cat > iplist.txt <<EOL
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
EOL
```

**IP 집합의 IP 주소 목록이 있는 파일에는 행당 항목이 포함되어야 합니다. 해시, 세미콜론 또는 빈 행으로 시작하는 행은 무시됩니다.**

**iplist.txt 파일의 주소를 추가하려면 root 로 다음 명령을 사용하십시오.**

```
firewall-cmd --permanent --ipset=test --add-entries-from-file=iplist.txt
success
```

**IP 세트의 확장 항목 목록을 보려면 root 로 다음 명령을 사용하십시오.**

```
firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

**IP 세트에서 주소를 제거하고 업데이트된 항목 목록을 확인하려면 다음 명령을 루트로 사용합니다.**

```
firewall-cmd --permanent --ipset=test --remove-entries-from-file=iplist.txt
success
firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

**IP 세트를 영역에 소스로 추가하여 영역으로 설정된 IP 세트에 나열된 주소에서 들어오는 모든 트래픽을 처리할 수 있습니다. 예를 들어 테스트 IP 세트를 드롭 영역에 소스로 추가하여 테스트 IP 세트에 나열된 모든 항목에서 들어오는 모든 패킷을 삭제하려면 root 로 다음 명령을 사용합니다.**

```
firewall-cmd --permanent --zone=drop --add-source=ipset:test
success
```

소스의 **ipset**: 접두사는 소스 가 IP 주소 또는 주소 범위가 아닌 IP 세트임을 표시합니다.

IP 집합의 생성 및 제거만 영구 환경으로 제한되며, **--permanent** 옵션을 사용하지 않고 런타임 환경에서 다른 모든 IP 집합 옵션을 사용할 수도 있습니다.



#### 주의

Red Hat은 **firewalld** 를 통해 관리되지 않는 IP 세트를 사용하지 않는 것이 좋습니다. 이러한 IP 세트를 사용하려면 세트를 참조하려면 영구적인 직접 규칙이 필요하며 이러한 IP 세트를 생성하려면 사용자 지정 서비스를 추가해야 합니다. 이 서비스는 **firewalld** 를 시작하기 전에 시작해야 합니다. 그렇지 않으면 **firewalld** 가 이러한 세트를 사용하여 직접 규칙을 추가할 수 없습니다. **/etc/firewalld/direct.xml** 파일을 사용하여 영구적인 직접 규칙을 추가할 수 있습니다.

### 51.13. 리치 규칙 우선순위 지정

기본적으로 리치 규칙은 규칙 동작을 기반으로 구성됩니다. 예를 들어 거부 규칙은 허용 규칙보다 우선합니다. 리치 규칙의 **priority** 매개 변수는 관리자가 리치 규칙과 실행 순서를 세부적으로 제어할 수 있습니다.

#### 51.13.1. 우선순위 매개변수가 규칙을 다른 체인으로 구성하는 방법

리치 규칙에서 우선순위 매개변수를 **-32768** 및 **32767** 사이의 숫자로 설정할 수 있으며 더 낮은 값이 우선합니다.

**firewalld** 서비스는 우선 순위 값을 기반으로 다른 체인으로 규칙을 구성합니다.

- **0보다 낮은 우선 순위:** 규칙이 **\_pre** 접미사가 있는 체인으로 리디렉션됩니다.
- **우선순위 0:** 규칙이 **\_post** 접미사가 있는 체인으로 리디렉션됩니다.
- **우선 순위는 0:** 작업을 기반으로 하는 규칙은 **\_log**, **\_deny** 또는 **\_allow** 를 사용하여 체인으로 리디렉션됩니다.



이러한 하위 체인 내에서 **firewalld** 는 우선 순위 값을 기반으로 규칙을 정렬합니다.

### 51.13.2. 리치 규칙의 우선 순위 설정

절차에서는 **priority** 매개 변수를 사용하여 다른 규칙에서 허용되거나 거부되지 않는 모든 트래픽을 기록하는 리치 규칙을 생성하는 방법의 예를 설명합니다. 이 규칙을 사용하여 예기치 않은 트래픽에 플래그를 지정할 수 있습니다.

#### 절차

1.

우선 순위가 매우 낮은 리치 규칙을 추가하여 다른 규칙과 일치하지 않는 모든 트래픽을 기록합니다.

```
firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"'
```

명령은 로그 항목 수를 분당 5 개로 추가로 제한합니다.

2.

선택적으로 이전 단계에서 명령을 생성한 **nftables** 규칙을 표시합니다.

```
nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
 chain filter_IN_public_post {
 log prefix "UNEXPECTED: " limit rate 5/minute
 }
}
```

### 51.14. 방화벽 잠금 구성

로컬 애플리케이션 또는 서비스는 방화벽 구성이 **root** (예: **libvirt**)로 실행되는 경우 변경할 수 있습니다. 이 기능을 사용하면 관리자가 방화벽 구성을 잠글 수 있으므로 애플리케이션이 없거나 잠금 허용 목록에 추가된 애플리케이션만 방화벽 변경 사항을 요청할 수 있습니다. 기본적으로 잠금 설정이 비활성화됩니다. 활성화된 경우 로컬 애플리케이션 또는 서비스에서 방화벽에 대한 원하지 않는 구성 변경 사항이 없는지 확인할 수 있습니다.

#### 51.14.1. CLI를 사용하여 잠금 구성

다음 절차에서는 명령줄을 사용하여 잠금을 활성화하거나 비활성화하는 방법을 설명합니다.

•

잠금이 활성화되었는지 여부를 쿼리하려면 **root** 로 다음 명령을 사용하십시오.

```
firewall-cmd --query-lockdown
```

잠금이 활성화된 경우 명령이 종료 상태 **0** 으로 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 로 **no** 를 출력합니다.

•

잠금을 활성화하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --lockdown-on
```

•

잠금을 비활성화하려면 **root** 로 다음 명령을 사용하십시오.

```
firewall-cmd --lockdown-off
```

#### 51.14.2. CLI를 사용하여 잠금 허용 목록 옵션 구성

잠금 허용 목록에는 명령, 보안 컨텍스트, 사용자 및 사용자 ID가 포함될 수 있습니다. 허용 목록의 명령 항목이 별표 "\*"로 끝나면 해당 명령으로 시작하는 모든 명령줄이 일치합니다. "\*"가 없으면 인수를 포함한 절대 명령이 일치해야 합니다.

•

컨텍스트는 실행 중인 애플리케이션 또는 서비스의 보안(**SELinux**) 컨텍스트입니다. 실행 중인 애플리케이션의 컨텍스트를 가져오려면 다음 명령을 사용합니다.

```
$ ps -e --context
```

이 명령은 실행 중인 모든 애플리케이션을 반환합니다. 출력을 **grep** 도구를 통해 파이프하여 관심 있는 애플리케이션을 가져옵니다. 예를 들면 다음과 같습니다.

```
$ ps -e --context | grep example_program
```

•

허용 목록에 있는 모든 명령행을 나열하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --list-lockdown-whitelist-commands
```

•

허용 목록에 명령을 추가하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

•

허용 목록에서 명령을 제거하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

•

명령이 허용 목록에 있는지 쿼리하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --query-lockdown-whitelist-command='/usr/bin/python3 -Es /usr/bin/command'
```

이 명령은 **true**인 경우 종료 상태 **0** 으로 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 로 **no** 를 출력합니다.

•

허용 목록에 있는 모든 보안 컨텍스트를 나열하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --list-lockdown-whitelist-contexts
```

•

허용 목록에 컨텍스트 컨텍스트를 추가하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --add-lockdown-whitelist-context=context
```

•

허용 목록에서 컨텍스트 컨텍스트를 제거하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --remove-lockdown-whitelist-context=context
```

•

컨텍스트 컨텍스트 가 허용 목록에 있는지 쿼리하려면 **root** 로 다음 명령을 입력합니다.

```
firewall-cmd --query-lockdown-whitelist-context=context
```

종료 상태 **0** (**true**인 경우)을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 에서 **no** 를 출력합니다.

- 허용 목록에 있는 모든 사용자 **ID**를 나열하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --list-lockdown-whitelist-uids
```
- 허용 목록에 사용자 **ID uid** 를 추가하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --add-lockdown-whitelist-uid=uid
```
- 허용 목록에서 사용자 **ID uid** 를 제거하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --remove-lockdown-whitelist-uid=uid
```
- 사용자 **ID uid** 가 허용 목록에 있는지 쿼리하려면 다음 명령을 입력합니다.  

```
$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

종료 상태 **0 (true인 경우)**을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 **1** 에서 **no** 를 출력합니다.
- 허용 목록에 있는 모든 사용자 이름을 나열하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --list-lockdown-whitelist-users
```
- 허용 목록에 사용자 이름 사용자를 추가하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --add-lockdown-whitelist-user=user
```
- 허용 목록에서 사용자 이름 사용자를 제거하려면 **root** 로 다음 명령을 입력합니다.  

```
firewall-cmd --remove-lockdown-whitelist-user=user
```
- 사용자 이름 사용자가 허용 목록에 있는지 쿼리하려면 다음 명령을 입력합니다.  

```
$ firewall-cmd --query-lockdown-whitelist-user=user
```

종료 상태 0 (true인 경우)을 사용하여 **yes** 를 출력합니다. 그렇지 않으면 종료 상태 1 에서 **no** 를 출력합니다.

### 51.14.3. 설정 파일을 사용하여 잠금 허용 목록 옵션 구성

기본 허용 목록 구성 파일에는 **NetworkManager** 컨텍스트와 **libvirt** 의 기본 컨텍스트가 포함되어 있습니다. 사용자 ID 0도 목록에 있습니다.

+ **allowlist** 구성 파일은 **/etc/firewalld/** 디렉토리에 저장됩니다.

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist>
 <selinux context="system_u:system_r:NetworkManager_t:s0"/>
 <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
 <user id="0"/>
</whitelist>
```

다음은 사용자 ID가 815 인 사용자에게 대해 **firewall-cmd** 유틸리티에 대한 모든 명령을 활성화하는 허용 목록 구성 파일의 예입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist>
 <command name="/usr/libexec/platform-python -s /bin/firewall-cmd*"/>
 <selinux context="system_u:system_r:NetworkManager_t:s0"/>
 <user id="815"/>
 <user name="user"/>
</whitelist>
```

이 예에서는 사용자 ID 와 사용자 이름을 모두 표시하지만 하나의 옵션만 필요합니다. **Python**은 인터프리터이며 명령줄 앞에 추가됩니다. 다음과 같은 특정 명령을 사용할 수도 있습니다.

```
/usr/bin/python3 /bin/firewall-cmd --lockdown-on
```

이 예제에서는 **--lockdown-on** 명령만 허용됩니다.

**Red Hat Enterprise Linux**에서 모든 유틸리티는 **/usr/bin/** 디렉토리에 배치되고 **/bin/** 디렉토리는 **/usr/bin/** 디렉토리에 연결되어 있습니다. 즉, **root** 로 입력될 때 **firewall-cmd** 의 경로를 **/bin/firewall-cmd** 로 해석할 수 있지만 **/usr/bin/firewall-cmd** 를 사용할 수 있습니다. 모든 새 스크립트는 새 위치를 사용해야 합니다. 그러나 **root** 로 실행되는 스크립트가 **/bin/firewall-cmd** 경로를 사용하도록 작성된 경우

기존에 루트가 아닌 사용자에게만 사용되는 `/usr/bin/firewall-cmd` 경로 외에 허용 목록에 명령 경로를 추가해야 합니다.

명령의 **name** 특성 끝에 있는 **\***는 이 문자열로 시작하는 모든 명령이 일치함을 의미합니다. **\***가 없는 경우 인수를 포함한 절대 명령이 일치해야 합니다.

### 51.15. FIREWALLD 영역 내의 다양한 인터페이스 또는 소스 간 트래픽 전달 활성화

내부 영역 전달은 **firewalld** 영역 내의 인터페이스 또는 소스 간 트래픽 전달을 활성화하는 **firewalld** 기능입니다.

#### 51.15.1. 기본 타겟이 **ACCEPT**로 설정된 인트라 영역 전달과 영역의 차이점

영역 내 전달이 활성화되면 단일 **firewalld** 영역 내 트래픽이 인터페이스 또는 소스에서 다른 인터페이스 또는 소스로 이동할 수 있습니다. 영역은 인터페이스 및 소스의 신뢰 수준을 지정합니다. 신뢰 수준이 동일하면 인터페이스 또는 소스 간의 통신이 가능합니다.

**firewalld**의 기본 영역에서 영역 전달을 활성화하는 경우 현재 기본 영역에 추가된 인터페이스와 소스에만 적용됩니다.

**firewalld**의 신뢰할 수 있는 영역은 **ACCEPT**로 설정된 기본 대상을 사용합니다. 이 영역은 전달된 모든 트래픽을 허용하며, 영역 내 전달은 적용되지 않습니다.

다른 기본 대상 값과 마찬가지로, 전달된 트래픽은 기본적으로 삭제되며 신뢰할 수 있는 영역을 제외한 모든 표준 영역에 적용됩니다.

#### 51.15.2. 영역 내 전달을 사용하여 이더넷 및 Wi-Fi 네트워크 간의 트래픽 전달

영역 내 전달을 사용하여 동일한 **firewalld** 영역 내의 인터페이스와 소스 간에 트래픽을 전달할 수 있습니다. 예를 들어, 이 기능을 사용하여 **enp1s0**에 연결된 이더넷 네트워크와 **wlp0s 20**에 연결된 Wi-Fi 네트워크 간에 트래픽을 전달합니다.

#### 절차

1.

커널에서 패킷 전달을 활성화합니다.

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
```

```
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

내부 영역 전달을 활성화하려는 인터페이스는 내부 영역과 다른 영역에 할당되지 않은지 확인합니다.

```
firewall-cmd --get-active-zones
```

3.

현재 인터페이스가 내부가 아닌 영역에 할당된 경우 인터페이스를 다시 할당합니다.

```
firewall-cmd --zone=internal --change-interface=interface_name --permanent
```

4.

**enp1s0** 및 **wlp0s20** 인터페이스를 **internal** 영역에 추가합니다.

```
firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5.

지역 내 전달을 활성화합니다.

```
firewall-cmd --zone=internal --add-forward
```

## 검증

다음 확인 단계에서는 **nmap-ncat** 패키지가 두 호스트 모두에 설치되어 있어야 합니다.

1.

영역 전달을 활성화한 호스트의 **enp1s0** 인터페이스와 동일한 네트워크에 있는 호스트에 로그인합니다.

2.

**ncat** 으로 **echo** 서비스를 시작하여 연결을 테스트합니다.

```
ncat -e /usr/bin/cat -l 12345
```

3.

**wlp0s20** 인터페이스와 동일한 네트워크에 있는 호스트에 로그인합니다.

4.

**enp1s0** 과 동일한 네트워크에 있는 호스트에서 실행 중인 에코 서버에 연결합니다.

```
ncat <other host> 12345
```

5.

내용을 입력하고 **Enter** 키를 누르고 텍스트가 다시 전송되었는지 확인합니다.

추가 리소스

- [firewalld.zones\(5\) 도움말 페이지](#)

## 51.16. ANSIBLE에서 RHEL 시스템 역할을 사용하여 FIREWALLD 설정 구성

**Ansible** 방화벽 시스템 역할을 사용하여 한 번에 여러 클라이언트에서 **firewalld** 서비스 설정을 구성할 수 있습니다. 이 해결 방법:

- 효율적인 입력 설정을 사용하는 인터페이스를 제공합니다.
- 의도한 모든 **firewalld** 매개 변수를 한 곳에 보관합니다.

제어 노드에서 **firewall** 역할을 실행한 후 시스템 역할은 관리 노드에 **firewalld** 매개 변수를 즉시 적용하고 재부팅 시 일관되게 유지되도록 합니다.



### 중요

**RHEL** 채널을 통해 제공되는 **RHEL** 시스템 역할은 기본 **AppStream** 리포지토리의 **RPM** 패키지로 **RHEL** 고객이 사용할 수 있습니다. **RHEL** 시스템 역할은 **Ansible Automation Hub**를 통해 **Ansible** 서브스크립션을 사용하는 고객에 대한 컬렉션으로도 사용할 수 있습니다.

### 51.16.1. 방화벽 RHEL 시스템 역할 소개

**RHEL** 시스템 역할은 **Ansible** 자동화 유틸리티에 대한 콘텐츠 집합입니다. **Ansible** 자동화 유틸리티와 함께 이 콘텐츠는 여러 시스템을 원격으로 관리할 수 있는 일관된 구성 인터페이스를 제공합니다.

**firewalld** 서비스의 자동 구성을 위해 **RHEL** 시스템 역할의 **rhel-system-roles.firewall** 역할이 도입되었습니다. **rhel-system-roles** 패키지에는 이 시스템 역할과 참조 문서도 포함됩니다.



자동화된 방식으로 하나 이상의 시스템에 **firewalld** 매개 변수를 적용하려면 플레이북에서 방화벽 시스템 역할 변수를 사용합니다. 플레이북은 텍스트 기반 **YAML** 형식으로 작성된 하나 이상의 플레이 목록입니다.

인벤토리 파일을 사용하여 **Ansible**에서 구성할 시스템 세트를 정의할 수 있습니다.

**firewall** 역할을 사용하면 다양한 **firewalld** 매개 변수를 구성할 수 있습니다. 예를 들면 다음과 같습니다.

- 영역.
- 패킷을 허용해야 하는 서비스입니다.
- 포트에 대한 트래픽 액세스 권한 부여, 거부 또는 삭제
- 영역의 포트 또는 포트 범위 전달.

#### 추가 리소스

- [/usr/share/doc/rhel-system-roles/firewall/ 디렉터리의 README.md 및 README.html 파일](#)
- [플레이북 작업](#)
- [인벤토리 빌드 방법](#)

#### 51.16.2. 한 로컬 포트에서 다른 로컬 포트로 들어오는 트래픽 전달

**rhel-system-roles.firewall** 역할을 사용하면 여러 관리 호스트에 지속 효과를 사용하여 **firewalld** 매개 변수를 원격으로 구성할 수 있습니다.

#### 사전 요구 사항

- **RHEL** 서브스크립션을 통해 제어 노드에 **ansible-core** 및 **rhel-system-roles** 패키지를 설치

했습니다.

- 관리 호스트의 인벤토리는 제어 시스템에 있으며 **Ansible**은 이러한 호스트에 연결할 수 있습니다.
- 관리 호스트에서 **Ansible** 플레이북을 실행할 수 있는 권한이 있습니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 호스트에서 적절한 **sudo** 권한이 있습니다.
- 인벤토리 파일은 플레이북에서 작업을 수행해야 하는 호스트를 나열합니다. 이 절차의 플레이북은 **testinservers** 그룹의 호스트에서 실행됩니다.

#### 중요

**RHEL 8.0 - 8.5**는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible,ansible-playbook; docker** 및 **podman** 과 같은 커넥터, 플러그인 및 모듈의 전체 세트와 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 가져오고 설치하는 방법에 대한 자세한 내용은 **Red Hat Ansible Engine**을 다운로드하고 설치하는 방법을 참조하십시오.

**RHEL 8.6** 이상에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모 기본 제공 **Ansible** 플러그인 세트를 포함하는 **Ansible Core( ansible-core RPM로 제공)**를 도입했습니다. **AppStream** 리포지토리는 제한된 지원 범위가 있는 **ansible-core** 를 제공합니다. **RHEL 9 AppStream**에 포함된 **ansible-core** 패키지에 대한 지원 범위를 검토하여 자세한 내용을 확인할 수 있습니다.

#### 절차

1.

`~/port_forwarding.yml` 파일을 만들고 다음 내용을 추가합니다.

```

- name: Forward incoming traffic on port 8080 to 443
 hosts: testingservers

 tasks:
 - include_role:
 name: rhel-system-roles.firewall
```

```
vars:
 firewall:
 - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

이 파일은 플레이북을 나타내며 일반적으로 인벤토리 파일에서 선택한 특정 관리 호스트에 대해 실행되는 플레이 라는 정렬된 작업 목록을 포함합니다. 이 경우 플레이북은 관리 호스트의 **testingservers** 그룹에 대해 실행됩니다.

플레이의 **hosts** 키는 플레이가 실행되는 호스트를 지정합니다. 이 키의 값 또는 값을 관리 호스트의 개별 이름으로 제공하거나 인벤토리 파일에 정의된 호스트 그룹으로 제공할 수 있습니다.

**tasks** 섹션에는 **vars** 섹션에 언급된 매개 변수와 값을 구성하기 위해 어떤 시스템 역할을 구성할지 지정하는 **include\_role** 키가 있습니다.

**vars** 섹션에는 **firewall** 이라는 역할 변수가 포함되어 있습니다. 이 변수는 사전 값 목록이며 관리 호스트에서 **firewalld** 에 적용할 매개 변수를 지정합니다. **example** 역할은 포트 **8080**으로 들어오는 트래픽을 포트 **443**으로 전달합니다. 이 설정은 즉시 적용되며 재부팅 시에도 지속됩니다.

2.

선택적으로 플레이북의 구문이 올바른지 확인합니다.

```
ansible-playbook --syntax-check ~/port_forwarding.yml
```

```
playbook: port_forwarding.yml
```

이 예는 플레이북의 성공적인 확인을 보여줍니다.

3.

**Playbook**을 실행합니다.

```
ansible-playbook ~/port_forwarding.yml
```

검증

- 

관리 호스트에서 다음을 수행합니다.

- 

호스트를 재시작하여 재부팅 후에도 **firewalld** 설정이 아직 있는지 확인합니다.

```
reboot
```

○

**firewalld** 설정을 표시합니다.

```
firewall-cmd --list-forward-ports
```

#### 추가 리소스

- [RHEL 시스템 역할 시작하기](#)
- [/usr/share/doc/rhel-system-roles/firewall/ 디렉터리의 README.html 및 README.md 파일](#)
- [인벤토리 빌드](#)
- [Ansible 구성](#)
- [플레이북 작업](#)
- [변수 사용](#)
- [역할](#)

#### 51.16.3. 시스템 역할을 사용하여 포트 구성

**RHEL(Red Hat Enterprise Linux) firewalld** 시스템 역할을 사용하여 들어오는 트래픽을 위해 로컬 방화벽에서 포트를 열거나 닫고 재부팅 시 새 구성을 유지할 수 있습니다. 이 예제에서는 **HTTPS** 서비스에 들어오는 트래픽을 허용하도록 기본 영역을 구성하는 방법을 설명합니다.

**Ansible** 제어 노드에서 이 절차를 실행합니다.

#### 사전 요구 사항

- **firewalld** 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액

세스 및 권한.

- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

## 절차

1. 플레이북에서 명령을 실행할 호스트가 아직 인벤토리되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/adding-and-removing-ports.yml** 플레이북을 만듭니다.

```

- name: Allow incoming HTTPS traffic to the local host
 hosts: node.example.com
 become: true

 tasks:
 - include_role:
 name: linux-system-roles.firewall

 vars:
 firewall:
 - port: 443/tcp
 service: http
 state: enabled
 runtime: true
 permanent: true
```

**permanent: true** 옵션을 사용하면 재부팅 시 새 설정이 유지됩니다.

3.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u root ~/adding-and-removing-ports.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
ansible-playbook -u user_name --ask-become-pass ~/adding-and-removing-ports.yml
```

**ask -become-pass** 옵션을 사용하면 **ansible-playbook** 명령이 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

## 검증

1.

관리 노드에 연결합니다.

```
$ ssh user_name@node.example.com
```

2.

**HTTPS** 서비스와 연결된 **443/tcp** 포트가 열려 있는지 확인합니다.

```
$ sudo firewall-cmd --list-ports
443/tcp
```

## 추가 리소스

•

[/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)

•

[ansible-playbook\(1\) 도움말 페이지](#)

### 51.16.4. firewalld RHEL 시스템 역할을 사용하여 teamd firewalld 영역 구성

시스템 관리자는 **RHEL firewalld** 시스템 역할을 사용하여 영역에 대한 **HTTPS** 트래픽을 허용하도록 **enp1s0** 인터페이스에서 **dmz** 영역을 구성할 수 있습니다. 이렇게 하면 외부 사용자가 웹 서버에 액세스할 수 있습니다.

#### 사전 요구 사항

- **VPN** 시스템 역할로 구성하려는 시스템인 하나 이상의 관리형 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.
- 관리형 노드를 나열하는 인벤토리 파일입니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 관리형 노드는 **NetworkManager** 를 사용하여 네트워크를 구성합니다.

#### 절차

1. 다음 콘텐츠를 사용하여 **~/configuring-a-dmz-using-the-firewall-system-role.yml** 플레이북을 생성합니다.

```

- name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
 hosts: node.example.com
 become: true

 tasks:
 - include_role:
 name: linux-system-roles.firewall

 vars:
 firewall:
 - zone: dmz
 interface: enp1s0
 service: https
```

```
state: enabled
runtime: true
permanent: true
```

2.

플레이북을 실행합니다.

•

**root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
$ ansible-playbook -u root ~/configuring-a-dmz-using-the-firewall-system-role.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
$ ansible-playbook -u user_name --ask-become-pass ~/configuring-a-dmz-using-the-firewall-system-role.yml
```

**ask -become-pass** 옵션을 사용하면 **ansible-playbook** 명령이 **-u user\_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

**-u user\_name** 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

## 검증

•

관리 노드에서 **dmz** 영역에 대한 자세한 정보를 확인합니다.

```
firewall-cmd --zone=dmz --list-all
dmz (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0
sources:
services: https ssh
ports:
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
```

## 51.17. 추가 리소스



- [\*firewalld\(1\) 도움말 페이지\*](#)
- [\*firewalld.conf\(5\) 도움말 페이지\*](#)
- [\*firewall-cmd\(1\) 도움말 페이지\*](#)
- [\*firewall-config\(1\) 도움말 페이지\*](#)
- [\*firewall-offline-cmd\(1\) 도움말 페이지\*](#)
- [\*firewalld.icmptype\(5\) 도움말 페이지\*](#)
- [\*firewalld.ipset\(5\) 도움말 페이지\*](#)
- [\*firewalld.service\(5\) 도움말 페이지\*](#)
- [\*firewalld.zone\(5\) 도움말 페이지\*](#)
- [\*firewalld.direct\(5\) 도움말 페이지\*](#)
- [\*firewalld.lockdown-whitelist\(5\)\*](#)
- [\*firewalld.richlanguage\(5\)\*](#)
- [\*firewalld.zones\(5\) 도움말 페이지\*](#)
- [\*firewalld.dbus\(5\) 도움말 페이지\*](#)

## 52장. NFTABLES 시작하기

**nftables** 프레임워크는 패킷 분류 기능을 제공합니다. 가장 주목할 만한 기능은 다음과 같습니다.

- 선형 처리 대신 기본 제공 조회 테이블
- IPv4 및 IPv6 프로토콜 모두를 위한 단일 프레임워크
- 전체 규칙 세트를 가져오기, 업데이트 및 저장하는 대신 모두 원자적으로 적용되는 규칙
- 규칙 세트(nfttrace) 및 모니터링 추적 이벤트(nft의)에서 디버깅 및 추적 지원
- 프로토콜별 확장 없이 보다 일관되고 컴팩트한 구문
- 타사 애플리케이션용 Netlink API

**nftables** 프레임워크는 테이블을 사용하여 체인을 저장합니다. 체인에는 작업을 수행하기 위한 개별 규칙이 포함되어 있습니다. **libnftnl** 라이브러리는 **libmnl** 라이브러리를 통해 **nftables Netlink API**와의 낮은 수준의 상호 작용에 사용할 수 있습니다.

규칙 세트 변경의 효과를 표시하려면 **nft list ruleset** 명령을 사용합니다. 이러한 툴은 테이블, 체인, 규칙, 세트 및 기타 오브젝트를 **nftables** 규칙 세트에 추가하므로 **nft flush ruleset** 명령과 같은 **nftables** 규칙 세트 작업이 이전에 분리된 레거시 명령을 사용하여 설치된 규칙 세트에 영향을 줄 수 있습니다.

### 52.1. IPTABLES에서 NFTABLES로 마이그레이션

방화벽 구성에서 **iptables** 규칙을 계속 사용하는 경우 **iptables** 규칙을 **nftables** 로 마이그레이션할 수 있습니다.

#### 52.1.1. firewalld, nftables 또는 iptables를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld:** 간단한 방화벽 사용 사례에 **firewalld** 유틸리티를 사용합니다. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables:** **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 등의 복잡하고 성능 중요한 방화벽을 설정합니다.
- **iptables:** Red Hat Enterprise Linux의 **iptables** 유틸리티는 레거시 백엔드 대신 **the nf\_tables** 커널 API를 사용합니다. **the nf\_tables** API는 **iptables** 명령을 사용하는 스크립트가 여전히 Red Hat Enterprise Linux에서 작동하도록 이전 버전과의 호환성을 제공합니다. 새로운 방화벽 스크립트의 경우 **nftables** 를 사용하는 것이 좋습니다.



#### 중요

서로 다른 방화벽 서비스가 서로 영향을 미치지 않도록 하려면 **RHEL** 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

### 52.1.2. iptables 및 ip6tables 규칙 세트를 nftables로 변환

**iptables-restore-translate** 및 **ip6tables-restore-translate** 유틸리티를 사용하여 **iptables** 및 **ip6tables** 규칙 세트를 **nftables** 로 변환합니다.

#### 사전 요구 사항

- **nftables** 및 **iptables** 패키지가 설치됩니다.
- 시스템에는 **iptables** 및 **ip6tables** 규칙이 구성되어 있습니다.

#### 절차

1. **iptables** 및 **ip6tables** 규칙을 파일에 작성합니다.

```
iptables-save >/root/iptables.dump
ip6tables-save >/root/ip6tables.dump
```

2. 덤프 파일을 **nftables** 명령으로 변환합니다.

```
iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-
from-iptables.nft
ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-
from-ip6tables.nft
```

3.

및 필요한 경우 생성된 **nftables** 규칙을 수동으로 업데이트합니다.

4.

생성된 파일을 로드할 **nftables** 서비스를 활성화하려면 `/etc/sysconfig/nftables.conf` 파일에 다음을 추가합니다.

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5.

**iptables** 서비스를 중지하고 비활성화합니다.

```
systemctl disable --now iptables
```

사용자 지정 스크립트를 사용하여 **iptables** 규칙을 로드한 경우 스크립트가 더 이상 자동으로 시작되지 않고 재부팅하여 모든 테이블을 플러시해야 합니다.

6.

**nftables** 서비스를 활성화하고 시작합니다.

```
systemctl enable --now nftables
```

## 검증

•

**nftables** 규칙 세트를 표시합니다.

```
nft list ruleset
```

## 추가 리소스

•

시스템 부팅 시 **nftables** 규칙 자동 로딩

### 52.1.3. 단일 **iptables** 및 **ip6tables** 규칙을 **nftables**로 변환

Red Hat Enterprise Linux는 **iptables-translate** 및 **ip6tables-translate** 유틸리티를 제공하여 **iptables** 또는 **ip6tables** 규칙을 **nftables**에 해당하는 규칙으로 변환합니다.

### 사전 요구 사항

- **nftables** 패키지가 설치되어 있어야 합니다.

### 절차

- **iptables** 또는 **ip6tables** 대신 **iptables-translate** 또는 **ip6tables-translate** 유틸리티를 사용하여 해당 **nftables** 규칙을 표시합니다. 예를 들면 다음과 같습니다.

```
iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

일부 확장 기능에는 해당 지원이 누락되어 있는 경우도 있습니다. 이 경우 유틸리티는 # 기호가 앞에 오는 **untranslated** 규칙을 출력합니다. 예를 들면 다음과 같습니다.

```
iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

### 추가 리소스

- **iptables-translate --help**

#### 52.1.4. 일반적인 **iptables** 및 **nftables** 명령 비교

다음은 일반적인 **iptables** 및 **nftables** 명령을 비교한 것입니다.

- 모든 규칙 나열:

| iptables             | nftables            |
|----------------------|---------------------|
| <b>iptables-save</b> | <b>nft</b> 목록 규칙 세트 |

- 특정 테이블 및 체인 나열:

| iptables                             | nftables                                  |
|--------------------------------------|-------------------------------------------|
| <b>iptables -L</b>                   | <b>nft</b> 목록 테이블 <b>ip filter</b>        |
| <b>iptables -L INPUT</b>             | <b>nft</b> 목록 체인 <b>IP 필터 INPUT</b>       |
| <b>iptables -t nat -L PREROUTING</b> | <b>nft</b> 목록 체인 <b>ip nat PREROUTING</b> |

**nft** 명령은 테이블 및 체인을 미리 생성하지 않습니다. 이는 사용자가 수동으로 생성한 경우에만 존재합니다.

예제: **firewalld**에서 생성한 규칙 나열

```
nft list table inet firewalld
nft list table ip firewalld
nft list table ip6 firewalld
```

### 52.1.5. 추가 리소스

- [iptables: 두 가지 변형과 nftables의 관계](#)

## 52.2. NFTABLES 스크립트 작성 및 실행

**nftables** 프레임워크는 방화벽 규칙을 유지 관리하기 위해 셸 스크립트를 사용하는 주요 이점을 가져오는 기본 스크립팅 환경을 제공합니다. 스크립트 실행은 **atomic**입니다. 즉, 시스템이 전체 스크립트를 적용하거나 오류가 발생할 경우 실행을 방지합니다. 이렇게 하면 방화벽이 항상 일관된 상태에 있습니다.

또한 **nftables** 스크립트 환경에서는 관리자가 다음을 수행할 수 있습니다.

- 코멘트 추가
- 변수 정의

•

다른 규칙 세트 파일 포함

이 섹션에서는 이러한 기능을 사용하는 방법과 **nftables** 스크립트를 만들고 실행하는 방법을 설명합니다.

**nftables** 패키지를 설치하면 **Red Hat Enterprise Linux**에서 **/etc/nftables/** 디렉터리에 **\*.nft** 스크립트를 자동으로 생성합니다. 이러한 스크립트에는 다양한 용도로 테이블과 빈 체인을 생성하는 명령이 포함되어 있습니다.

### 52.2.1. 지원되는 **nftables** 스크립트 형식

**nftables** 스크립팅 환경은 다음 형식으로 스크립트를 지원합니다.

•

**nft list ruleset** 명령과 동일한 형식으로 스크립트를 작성할 수 있습니다. 규칙 세트는 다음과 같습니다.

```
#!/usr/sbin/nft -f

Flush the rule set
flush ruleset

table inet example_table {
 chain example_chain {
 # Chain for incoming packets that drops all packets that
 # are not explicitly allowed by any rule in this chain
 type filter hook input priority 0; policy drop;

 # Accept connections to port 22 (ssh)
 tcp dport ssh accept
 }
}
```

•

**nft** 명령과 동일한 구문을 명령에 사용할 수 있습니다.

```
#!/usr/sbin/nft -f

Flush the rule set
flush ruleset

Create a table
add table inet example_table

Create a chain for incoming packets that drops all packets
```

```
that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0 ; policy
drop ; }

Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

### 52.2.2. nftables 스크립트 실행 중

**nft** 유틸리티로 전달하거나 스크립트를 직접 실행하여 **nftables** 스크립트를 실행할 수 있습니다.

#### 사전 요구 사항

- 이 섹션의 절차에서는 `/etc/nftables/example_firewall.nft` 파일에 **nftables** 스크립트를 저장했다고 가정합니다.

#### 절차

- nft** 유틸리티에 전달하여 **nftables** 스크립트를 실행하려면 다음을 입력합니다.

```
nft -f /etc/nftables/example_firewall.nft
```

- 직접 **nftables** 스크립트를 실행하려면 다음을 수행합니다.

a.

한 번만 필요한 단계 :

i.

스크립트가 다음 **shebang** 시퀀스로 시작되는지 확인합니다.

```
#!/usr/sbin/nft -f
```

중요

**f** 매개변수를 생략하면 **nft** 유틸리티가 스크립트를 읽지 않고 표시됩니다. 오류: 구문 오류, 예기치 않은 줄 바꿈, 예상 문자열.

ii.

선택 사항: 스크립트 소유자를 **root** 로 설정합니다.



```
chown root /etc/nftables/example_firewall.nft
```

iii.

소유자에 대해 스크립트를 실행 파일로 만듭니다.

```
chmod u+x /etc/nftables/example_firewall.nft
```

b.

스크립트를 실행합니다.

```
/etc/nftables/example_firewall.nft
```

출력이 표시되지 않으면 시스템에서 스크립트를 성공적으로 실행했습니다.



중요

**nft** 가 스크립트를 성공적으로 실행하는 경우에도 잘못 배치된 규칙, 누락된 매개 변수 또는 스크립트의 기타 문제로 인해 방화벽이 예상대로 작동하지 않을 수 있습니다.

추가 리소스

- [chown\(1\) 도움말 페이지](#)
- [chmod\(1\) 도움말 페이지](#)
- [시스템 부팅 시 nftables 규칙 자동 로딩](#)

### 52.2.3. nftables 스크립트에서 주석 사용

**nftables** 스크립팅 환경은 # 문자 오른쪽에 있는 모든 것을 주석으로 해석합니다.

#### 예 52.1. nftables 스크립트의 주석

주석은 명령 옆에뿐만 아니라 행의 시작 부분에서 시작할 수 있습니다.

```
...
Flush the rule set
```

```
flush ruleset
```

```
add table inet example_table # Create a table
```

```
...
```

#### 52.2.4. nftables 스크립트에서 변수 사용

**nftables** 스크립트에서 변수를 정의하려면 **define** 키워드를 사용합니다. 단일 값과 익명 세트를 변수에 저장할 수 있습니다. 더 복잡한 시나리오의 경우 세트 또는 확인 맵을 사용합니다.

단일 값이 있는 변수

다음 예제에서는 값이 **enp1s0** 인 **INET\_DEV** 라는 변수를 정의합니다.

```
define INET_DEV = enp1s0
```

\$ 기호 다음에 변수 이름을 작성하여 스크립트에서 변수를 사용할 수 있습니다.

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

익명 세트를 포함하는 변수

다음 예제에서는 익명 세트를 포함하는 변수를 정의합니다.

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

\$ 기호 다음에 변수 이름을 작성하여 스크립트에서 변수를 사용할 수 있습니다.

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```



참고

중괄호는 변수가 집합을 나타내고 있음을 나타내기 때문에 규칙에서 사용할 때 특수한 의미 체계가 있습니다.

추가 리소스

- **nftables 명령의 세트 사용**
- **nftables 명령에서 verdict 맵 사용**

### 52.2.5. nftables 스크립트에 파일 포함

**nftables** 스크립팅 환경을 사용하면 관리자가 **include** 문을 사용하여 다른 스크립트를 포함할 수 있습니다.

절대 경로 또는 상대 경로가 없는 파일 이름만 지정하는 경우 **nftables**에는 **Red Hat Enterprise Linux**에서 **/etc**로 설정된 기본 검색 경로의 파일이 포함됩니다.

#### 예 52.2. 기본 검색 디렉토리의 파일 포함

기본 검색 디렉터리에서 파일을 포함하려면 다음을 수행합니다.

```
include "example.nft"
```

#### 예 52.3. 디렉토리의 \*.nft 파일 모두 포함

**/etc/nftables/rulesets/** 디렉토리에 저장된 \*.nft로 끝나는 모든 파일을 포함시키려면 다음을 수행하십시오.

```
include "/etc/nftables/rulesets/*.nft"
```

**include** 문은 점으로 시작하는 파일과 일치하지 않습니다.

### 추가 리소스

- **nft(8) 도움말 페이지의 include files 섹션**

### 52.2.6. 시스템이 부팅될 때 nftables 규칙을 자동으로 로드

**nftables systemd** 서비스는 **/etc/sysconfig/nftables.conf** 파일에 포함된 방화벽 스크립트를 로드함

니다. 이 섹션에서는 시스템이 부팅될 때 방화벽 규칙을 로드하는 방법을 설명합니다.

#### 사전 요구 사항

- **nftables** 스크립트는 **/etc/nftables/** 디렉터리에 저장됩니다.

#### 절차

1. **/etc/sysconfig/nftables.conf** 파일을 편집합니다.
  - **nftables** 패키지를 설치할 때 **/etc/nftables/**에 생성된 \*.nft 스크립트를 개선하는 경우 이러한 스크립트에 대한 **include** 문의 주석을 제거합니다.
  - 스크립트를 처음부터 작성하는 경우 **include** 문을 추가하여 이러한 스크립트를 포함합니다. 예를 들어 **nftables** 서비스가 시작될 때 **/etc/nftables/example.nft** 스크립트를 로드하려면 다음을 추가합니다.

```
include "/etc/nftables/example.nft"
```

2. 선택적으로 시스템을 재부팅하지 않고 방화벽 규칙을 로드하도록 **nftables** 서비스를 시작합니다.

```
systemctl start nftables
```

3. **nftables** 서비스를 활성화합니다.

```
systemctl enable nftables
```

#### 추가 리소스

- 지원되는 **nftables** 스크립트 형식

### 52.3. NFTABLES 테이블, 체인 및 규칙 생성 및 관리

이 섹션에서는 **nftables** 규칙 세트를 표시하는 방법과 이를 관리하는 방법을 설명합니다.

### 52.3.1. 표준 체인 우선 순위 값 및 텍스트 이름

체인을 만들 때 우선 순위는 정수 값 또는 동일한 후크 값을 통과하는 체인이 있는 순서를 지정하는 표준 이름을 설정할 수 있습니다.

이름 및 값은 기본 체인을 등록할 때 **xtables** 에서 사용하는 우선 순위에 따라 정의됩니다.



참고

**nft list chains** 명령은 기본적으로 텍스트 우선 순위 값을 표시합니다. 명령에 **-y** 옵션을 전달하여 숫자 값을 볼 수 있습니다.

#### 예 52.4. 텍스트 값을 사용하여 우선순위 설정

다음 명령은 표준 우선 순위 값 **50** 을 사용하여 **example\_table**에 **example\_chain** 이라는 체인을 생성합니다.

```
nft add chain inet example_table example_chain { type filter hook input priority 50 \;
policy accept \; }
```

우선순위는 표준 값이므로 텍스트 값을 사용할 수도 있습니다.

```
nft add chain inet example_table example_chain { type filter hook input priority security
\; policy accept \; }
```

표 52.1. 표준 우선 순위 이름, 제품군 및 후크 호환성 매트릭스

| 이름     | 현재의  | 제품군                        | 후크         |
|--------|------|----------------------------|------------|
| raw    | -300 | ip, ip6, inet              | 모두         |
| mangle | -150 | ip, ip6, inet              | 모두         |
| dstnat | -100 | ip, ip6, inet              | PREROUTING |
| filter | 0    | ip, ip6, inet, arp, netdev | 모두         |
| 보안     | 50   | ip, ip6, inet              | 모두         |

| 이름     | 현재의 | 제품군           | 후크          |
|--------|-----|---------------|-------------|
| srcnat | 100 | ip, ip6, inet | POSTROUTING |

모든 제품군은 동일한 값을 사용하지만 브리지 제품군은 다음과 같은 값을 사용합니다.

표 52.2. 브리지 제품군의 표준 우선 순위 이름 및 후크 호환성

| 이름     | 현재의  | 후크          |
|--------|------|-------------|
| dstnat | -300 | PREROUTING  |
| filter | -200 | 모두          |
| 아웃     | 100  | 출력 결과       |
| srcnat | 300  | POSTROUTING |

추가 리소스

- [nft\(8\) 도움말 페이지의 체인 섹션](#)

### 52.3.2. nftables 규칙 세트 표시

**nftables**의 규칙 세트에는 테이블, 체인 및 규칙이 포함됩니다. 이 섹션에서는 규칙 집합을 표시하는 방법을 설명합니다.

절차

- 규칙 세트를 표시하려면 다음을 입력합니다.

```
nft list ruleset
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport http accept
 tcp dport ssh accept
 }
}
```



## 참고

기본적으로 **nftables** 는 테이블을 사전 생성하지 않습니다. 그 결과 테이블이 없는 호스트에 규칙 집합을 표시하면 **nft list ruleset** 명령은 출력이 표시되지 않습니다.

### 52.3.3. nftables 테이블 생성

**nftables** 의 테이블은 체인, 규칙, 세트 및 기타 오브젝트 컬렉션이 포함된 네임 스페이스입니다. 이 섹션에서는 테이블을 만드는 방법을 설명합니다.

각 테이블에는 주소 제품군이 정의되어 있어야 합니다. 테이블의 주소 제품군은 테이블 프로세스 유형을 정의합니다. 테이블을 만들 때 다음 주소 제품군 중 하나를 설정할 수 있습니다.

- **ip**: 일치하는 **IPv4** 패킷 만 일치합니다. 주소 제품군을 지정하지 않으면 기본값입니다.
- **ip6**: **IPv6** 패킷 만 일치합니다.
- **inet**: **IPv4** 및 **IPv6** 패킷과 일치합니다.
- **arp**: **IPv4 ARP(Address Resolution Protocol)** 패킷과 일치합니다.
- **bridge**: 는 브리지 장치를 통과하는 패킷과 일치합니다.
- **netdev**: 수신에서 패킷 일치.

## 절차

1. **nft add table** 명령을 사용하여 새 테이블을 만듭니다. 예를 들어 **IPv4** 및 **IPv6** 패킷을 처리하는 **example\_table** 이라는 테이블을 생성하려면 다음을 수행합니다.

```
nft add table inet example_table
```

2. 선택적으로 규칙 세트의 모든 테이블을 나열합니다.

```
nft list tables
table inet example_table
```

#### 추가 리소스

- [nft\(8\) 도움말 페이지의 주소 제품군 섹션](#)
- [nft\(8\) 도움말 페이지의 Tables 섹션](#)

#### 52.3.4. nftables 체인 생성

체인은 규칙을 위한 컨테이너입니다. 다음 두 가지 규칙 유형이 있습니다.

- **기본 체인:** 기본 체인을 네트워킹 스택의 패킷 진입점으로 사용할 수 있습니다.
- **일반 체인:** 일반 체인을 건너뛰기 대상으로 사용하고 규칙을 더 효과적으로 구성할 수 있습니다.

이 절차에서는 기존 테이블에 기본 체인을 추가하는 방법을 설명합니다.

#### 사전 요구 사항

- 새 체인을 추가할 테이블입니다.

#### 절차

1. **nft add chain** 명령을 사용하여 새 체인을 만듭니다. 예를 들어 **example\_table** 에 **example\_chain** 이라는 체인을 생성하려면 다음을 수행합니다.

```
nft add chain inet example_table example_chain { type filter hook input priority 0 ;
policy accept ; }
```

#### 중요

셸이 세미콜론을 명령의 끝으로 해석하지 않도록 하려면 세미콜론 앞에 **\** 이스케이프 문자 앞에 추가합니다.



이 체인은 들어오는 패킷을 필터링합니다. **priority** 매개변수는 **nftables**가 동일한 후크 값이 있는 체인을 처리하는 순서를 지정합니다. 우선순위 값이 더 높은 값보다 우선합니다. **policy** 매개 변수는 이 체인의 규칙에 대한 기본 작업을 설정합니다. 원격으로 서버에 로그인하고 기본 정책을 종료하도록 설정한 경우 다른 규칙에서 원격 액세스를 허용하지 않는 경우 즉시 연결이 끊어집니다.

2.

선택적으로 모든 체인을 표시합니다.

```
nft list chains
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 }
}
```

추가 리소스

- **nft(8)** 도움말 페이지의 주소 제품군 섹션
- **nft(8)** 도움말 페이지의 체인 섹션

### 52.3.5. nftables 체인 끝에 규칙 추가

이 섹션에서는 기존 **nftables** 체인의 끝에 규칙을 추가하는 방법을 설명합니다.

사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

절차

1.

새 규칙을 추가하려면 **nft add rule** 명령을 사용합니다. 예를 들어 **port 22**에서 **TCP** 트래픽을 허용하는 **example\_table**의 **example\_chain**에 규칙을 추가하려면 다음을 수행합니다.

```
nft add rule inet example_table example_chain tcp dport 22 accept
```

포트 번호 대신 서비스 이름을 지정할 수도 있습니다. 이 예제에서는 포트 번호 **22** 대신 **ssh**를 사용할 수 있습니다. 서비스 이름은 **/etc/services** 파일의 항목에 따라 포트 번호로 확인됩니다.

다.

2.

필요한 경우 모든 체인과 해당 규칙을 **example\_table**에 표시합니다.

```
nft list table inet example_table
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 ...
 tcp dport ssh accept
 }
}
```

추가 리소스

- **nft(8)** 도움말 페이지의 주소 제품군 섹션
- **nft(8)** 도움말 페이지의 **Rules** 섹션

### 52.3.6. nftables 체인의 시작 부분에 규칙 삽입

이 섹션에서는 기존 **nftables** 체인의 시작 부분에 규칙을 삽입하는 방법을 설명합니다.

사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

절차

1.

새 규칙을 삽입하려면 **nft** 삽입 규칙 명령을 사용합니다. 예를 들어 포트 **22**에서 **TCP** 트래픽을 허용하는 **example\_table**의 **example\_chain**에 규칙을 삽입하려면 다음을 수행합니다.

```
nft insert rule inet example_table example_chain tcp dport 22 accept
```

또는 포트 번호 대신 서비스 이름을 지정할 수 있습니다. 이 예제에서는 포트 번호 **22** 대신 **ssh**를 사용할 수 있습니다. 서비스 이름은 **/etc/services** 파일의 항목에 따라 포트 번호로 확인됩니다.

2.

필요한 경우 모든 체인과 해당 규칙을 **example\_table** 에 표시합니다.

```
nft list table inet example_table
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport ssh accept
 ...
 }
}
```

추가 리소스

- **nft(8)** 도움말 페이지의 주소 제품군 섹션
- **nft(8)** 도움말 페이지의 **Rules** 섹션

### 52.3.7. nftables 체인의 특정 위치에 규칙 삽입

이 섹션에서는 **nftables** 체인의 기존 규칙 전후에 규칙을 삽입하는 방법을 설명합니다. 이렇게 하면 올바른 위치에 새 규칙을 배치할 수 있습니다.

사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

절차

1.

**nft -a list ruleset** 명령을 사용하여 핸들을 포함하여 모든 체인과 해당 규칙을 **example\_table** 에 표시합니다.

```
nft -a list table inet example_table
table inet example_table { # handle 1
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport 22 accept # handle 2
 tcp dport 443 accept # handle 3
 tcp dport 389 accept # handle 4
 }
}
```

**a**를 사용하면 프로세스가 표시됩니다. 새 규칙을 다음 단계에서 배치하려면 이 정보가 필요

합니다.

2.

**example\_table** 의 **example\_chain** 체인에 새 규칙을 삽입합니다.

- 

핸들 3 이전의 포트 636 에 **TCP** 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.

```
nft insert rule inet example_table example_chain position 3 tcp dport 636 accept
```

- 

핸들 3 이후 포트 80 에서 **TCP** 트래픽을 허용하는 규칙을 추가하려면 다음을 입력합니다.

```
nft add rule inet example_table example_chain position 3 tcp dport 80 accept
```

3.

필요한 경우 모든 체인과 해당 규칙을 **example\_table** 에 표시합니다.

```
nft -a list table inet example_table
table inet example_table { # handle 1
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport 22 accept # handle 2
 tcp dport 636 accept # handle 5
 tcp dport 443 accept # handle 3
 tcp dport 80 accept # handle 6
 tcp dport 389 accept # handle 4
 }
}
```

추가 리소스

- 

**nft(8)** 도움말 페이지의 주소 제품군 섹션

- 

**nft(8)** 도움말 페이지의 **Rules** 섹션

## 52.4. NFTABLES를 사용하여 NAT 구성

**nftables** 를 사용하면 다음 **NAT**(네트워크 주소 변환) 유형을 구성할 수 있습니다.

- **마스커레이딩**
- **SNAT(소스 NAT)**
- **대상 NAT(DNAT)**
- **리디렉션**



중요

**iifname** 및 **oifname** 매개변수에서 실제 인터페이스 이름만 사용할 수 있으며 대체 이름 (**lname**)은 지원되지 않습니다.

#### 52.4.1. 다양한 NAT 유형: 마스커레이드, 소스 NAT, 대상 NAT, 리디렉션

다음은 다양한 NAT(네트워크 주소 변환) 유형입니다.

##### 마스커레이딩 및 소스 NAT (SNAT)

이러한 NAT 유형 중 하나를 사용하여 패킷의 소스 IP 주소를 변경합니다. 예를 들어 인터넷 서비스 공급자는 10.0.0.0/8 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하며 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 해당 범위에서 패킷의 소스 IP 주소를 공용 IP 주소로 매핑합니다.

마스커레이딩과 SNAT 모두 매우 유사합니다. 차이점은 다음과 같습니다.

- 마스커레이딩은 나가는 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 나가는 인터페이스에서 동적 IP 주소를 사용하는 경우 마스커레이딩을 사용합니다.
- SNAT는 패킷의 소스 IP 주소를 지정된 IP로 설정하고 나가는 인터페이스의 IP를 동적으로 조회하지 않습니다. 따라서 SNAT는 마스커레이딩보다 빠릅니다. 나가는 인터페이스에서 고정 IP 주소를 사용하는 경우 SNAT를 사용합니다.

##### 대상 NAT(DNAT)

이 NAT 유형을 사용하여 들어오는 패킷의 대상 주소와 포트를 다시 작성합니다. 예를 들어 웹 서버가 개인 IP 범위의 IP 주소를 사용하므로 인터넷에서 직접 액세스할 수 없는 경우 라우터에 DNAT 규

칙을 설정하여 들어오는 트래픽을 이 서버로 리디렉션할 수 있습니다.

## 리디렉션

이 유형은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 **DNAT**의 특별한 사례입니다. 예를 들어 서비스가 표준 포트와 다른 포트에서 실행되는 경우 표준 포트에서 들어오는 트래픽을 이 특정 포트로 리디렉션할 수 있습니다.

### 52.4.2. **nftables**를 사용하여 마스커레이딩 구성

마스커레이딩을 사용하면 라우터가 인터페이스를 통해 전송된 패킷의 소스 **IP**를 인터페이스의 **IP** 주소로 동적으로 변경할 수 있습니다. 즉, 인터페이스에 새 **IP**가 할당되면 **nftables**에서 소스 **IP**를 교체할 때 새 **IP**를 자동으로 사용합니다.

다음 절차에서는 호스트가 **ens3** 인터페이스를 통해 나가는 패킷의 소스 **IP**를 **ens3**에 설정된 **IP**로 바꾸는 방법을 설명합니다.

## 절차

1. 테이블을 생성합니다.

```
nft add table nat
```

2. 테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

## 중요

이전 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에는 이 체인이 들어오는 패킷 응답과 일치해야 합니다.

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 인터페이스에서 나가는 패킷과 일치하는 사후 라우팅 체인에 규칙을 추가합니다.

```
nft add rule nat postrouting oifname "ens3" masquerade
```

### 52.4.3. nftables를 사용하여 소스 NAT 구성

라우터에서 **SNAT(Source NAT)**를 사용하면 인터페이스를 통해 전송된 패킷의 IP를 특정 IP 주소로 변경할 수 있습니다.

다음 절차에서는 **ens3** 인터페이스를 통해 나가는 패킷의 소스 IP를 **192.0.2.1**로 바꾸는 방법을 설명합니다.

#### 절차

1. 테이블을 생성합니다.

```
nft add table nat
```

2. 테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

#### 중요

사후 라우팅 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에서 이 체인이 나가는 패킷 응답과 일치해야 합니다.

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3**을 통해 나가는 패킷의 소스 IP를 **192.0.2.1**로 대체하는 사후 체인에 규칙을 추가합니다.

```
nft add rule nat postrouting oifname "ens3" snat to 192.0.2.1
```

#### 추가 리소스

●

특정 로컬 포트에서 수신 패킷을 다른 호스트로 전달

#### 52.4.4. nftables를 사용하여 대상 NAT 구성

대상 NAT를 사용하면 라우터의 트래픽을 인터넷에서 직접 액세스할 수 없는 호스트로 리디렉션할 수 있습니다.

다음 절차에서는 수신되는 트래픽을 **192.0.2.1 IP** 주소가 있는 호스트로 라우터의 포트 **80** 및 **443** 으로 리디렉션하는 방법을 설명합니다.

##### 절차

1.

테이블을 생성합니다.

```
nft add table nat
```

2.

테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



##### 중요

사후 라우팅 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크에서 이 체인이 나가는 패킷 응답과 일치해야 합니다.

셀이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3.

포트 **80** 및 **443** 으로 전송된 **ens3** 인터페이스에서 들어오는 트래픽을 **192.0.2.1 IP**가 있는 호스트로 리디렉션하는 사전 라우팅 체인에 규칙을 추가합니다.

```
nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4.

환경에 따라 **SNAT** 또는 **마스커레이딩** 규칙을 추가하여 소스 주소를 변경합니다.



a.

**ens3** 인터페이스가 동적 IP 주소를 사용하는 경우 마스커레이딩 규칙을 추가합니다.

```
nft add rule nat postrouting oifname "ens3" masquerade
```

b.

**ens3** 인터페이스에서 정적 IP 주소를 사용하는 경우 **SNAT** 규칙을 추가합니다. 예를 들어 **ens3** 에서 **198.51.100.1** IP 주소를 사용하는 경우:

```
nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
```

추가 리소스

•

다른 **NAT** 유형: 마스커레이딩, 소스 **NAT**, 대상 **NAT**, 리디렉션

#### 52.4.5. nftables를 사용하여 리디렉션 구성

리디렉션 기능은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 대상 네트워크 주소 변환 (**DNAT**)의 특별한 사례입니다.

다음 절차에서는 로컬 호스트의 포트 **22** 로 전송되고 전달된 트래픽을 포트 **2222** 로 리디렉션하는 방법을 설명합니다.

절차

1.

테이블을 생성합니다.

```
nft add table nat
```

2.

앞에 있는 체인을 표에 추가합니다.

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않으려면 -- 옵션을 **nft** 명령에 전달해야 합니다.

3.

포트 **22** 에서 수신되는 트래픽을 포트 **2222** 로 리디렉션 하는 사전 라우팅 체인에 규칙을 추가합니다.

```
nft add rule nat prerouting tcp dport 22 redirect to 2222
```

## 추가 리소스



다른 **NAT** 유형: [마스커레이딩](#), [소스 NAT](#), [대상 NAT](#), [리디렉션](#)

## 52.5. NFTABLES 명령의 세트 사용

**nftables** 프레임워크는 기본적으로 세트를 지원합니다. 예를 들어 규칙이 여러 **IP** 주소, 포트 번호, 인터페이스 또는 기타 일치 기준에 일치해야 하는 경우 세트를 사용할 수 있습니다.

### 52.5.1. nftables에서 익명 세트 사용

익명 세트에는 규칙에 직접 사용하는 { 22, 80, 443 } 와 같은 중괄호로 묶인 쉼표로 구분된 값이 포함됩니다. 또한 **IP** 주소 또는 기타 일치 기준에 대해 익명 세트를 사용할 수도 있습니다.

익명 세트의 단점은 세트를 변경하려는 경우 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 **nftables**에서 [명명된 세트를 사용하도록](#)에 설명된 대로 **named sets**를 사용합니다.

## 사전 요구 사항



**inet** 제품군의 **example\_chain** 체인과 **example\_table** 테이블이 있습니다.

## 절차

- 1.

예를 들어 **example\_table**의 **example\_chain**에 규칙을 추가하여 포트 22, 80 및 443으로 들어오는 트래픽을 허용하는 규칙을 추가하려면 다음을 수행합니다.

```
nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

- 2.

필요한 경우 모든 체인과 해당 규칙을 **example\_table**에 표시합니다.

```
nft list table inet example_table
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport { ssh, http, https } accept
 }
}
```

### 52.5.2. nftables에서 명명된 세트 사용

**nftables** 프레임워크는 변경 이름이 지정된 세트를 지원합니다. 명명된 집합은 테이블 내의 여러 규칙에 사용할 수 있는 요소의 목록 또는 범위입니다. 익명 세트에 비해 또 다른 이점은 세트를 사용하는 규칙을 교체하지 않고 명명된 집합을 업데이트할 수 있다는 것입니다.

명명된 세트를 만들 때 세트에 포함된 요소 유형을 지정해야 합니다. 다음 유형을 설정할 수 있습니다.

- **IPv4** 주소 또는 범위를 포함하는 세트(예: **192.0.2.1** 또는 **192. 0.2.0/24**)의 **ipv4\_addr**.
- **IPv 6** 주소 또는 범위를 포함하는 세트의 **ipv6\_addr** (예: **2001:db8:1::1** 또는 **2001:db8:1::1/64**).
- **MAC**(메시지 액세스 제어) 주소 목록이 포함된 세트(예: **52:54:00:6b:66:42**)의 경우 **ether\_addr** 입니다.
- **inet\_proto**: **tcp** 와 같은 인터넷 프로토콜 유형 목록이 포함된 집합.
- **ssh** 와 같은 인터넷 서비스 목록이 포함된 세트의 **inet\_service**.
- 패킷 표시 목록이 포함된 집합의 표시. 패킷 표시는 양의 **32비트** 정수 값일 수 있습니다(**0~2 147483647**).

#### 사전 요구 사항

- **example\_chain** 체인과 **example\_table** 테이블이 있습니다.

#### 절차

1. 빈 세트를 만듭니다. 다음 예제에서는 **IPv4** 주소 세트를 생성합니다.

- 여러 개별 **IPv4** 주소를 저장할 수 있는 세트를 생성하려면 다음을 수행합니다.

```
nft add set inet example_table example_set { type ipv4_addr \;
```

•

IPv4 주소 범위를 저장할 수 있는 세트를 생성하려면 다음을 수행합니다.

```
nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
```



중요

셸이 세미콜론을 명령의 끝으로 해석하지 않도록 하려면 세미콜론을 백슬래시로 이스케이프해야 합니다.

2.

선택적으로 집합을 사용하는 규칙을 생성합니다. 예를 들어 다음 명령은 **example\_table**의 **example\_chain**에 규칙을 추가하여 **example\_set**의 IPv4 주소에서 모든 패킷을 삭제합니다.

```
nft add rule inet example_table example_chain ip saddr @example_set drop
```

**example\_set**이 여전히 비어 있으므로 현재 규칙이 적용되지 않습니다.

3.

**example\_set**에 IPv4 주소 추가 :

•

개별 IPv4 주소를 저장하는 세트를 생성하는 경우 다음을 입력합니다.

```
nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
```

•

IPv4 범위를 저장하는 세트를 생성하는 경우 다음을 입력합니다.

```
nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
```

IP 주소 범위를 지정하면 위 예제에서 **192.0.2.0/24**와 같은 CIDR(Classless Inter-Domain Routing) 표기법을 사용할 수도 있습니다.

### 52.5.3. 추가 리소스

•

**nft(8)** 도움말 페이지의 **Sets** 섹션

## 52.6. NFTABLES 명령에서 확인 맵 사용

사전이라고도 하는 **Verdict** 맵을 사용하면 일치 기준을 작업에 매핑하여 패킷 정보를 기반으로 **nft** 가 작업을 수행할 수 있습니다.

### 52.6.1. nftables에서 익명 맵 사용

익명 맵은 규칙에서 직접 사용하는 **{ match\_criteria : action }** 문입니다. 문에는 쉼표로 구분된 여러 맵핑이 포함될 수 있습니다.

익명 맵의 단점은 맵을 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 **nftables**에서 명명된 맵을 사용하도록 설명된 대로 이름이 지정된 맵을 사용합니다.

이 예제에서는 익명 맵을 사용하여 들어오는 **TCP** 및 **UDP** 패킷을 별도로 계산하기 위해 **IPv4** 및 **IPv6** 프로토콜의 **TCP** 및 **UDP** 패킷을 서로 다른 체인으로 라우팅하는 방법을 설명합니다.

#### 절차

1.

**example\_table** 을 생성합니다.

```
nft add table inet example_table
```

2.

**example\_table** 에 **tcp\_packets** 체인을 생성합니다.

```
nft add chain inet example_table tcp_packets
```

3.

이 체인의 트래픽을 계산하는 **tcp\_packets** 에 규칙을 추가합니다.

```
nft add rule inet example_table tcp_packets counter
```

4.

**example\_table**에 **udp\_packets** 체인을 만듭니다.

```
nft add chain inet example_table udp_packets
```

5.

이 체인의 트래픽을 계산하는 **udp\_packets** 에 규칙을 추가합니다.

```
nft add rule inet example_table udp_packets counter
```

6.

들어오는 트래픽의 체인을 만듭니다. 예를 들어, **example\_table**에서 들어오는 트래픽을 필터링하는 **incoming\_traffic** 이라는 체인을 생성하려면 다음을 실행합니다.

```
nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7.

익명 맵이 있는 규칙을 **ingress\_traffic** 에 추가합니다.

```
nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

익명 맵은 패킷을 구분하고 프로토콜을 기반으로 다른 카운터 체인으로 전송합니다.

8.

트래픽 카운터를 나열하려면 **example\_table** 을 표시합니다.

```
nft list table inet example_table
table inet example_table {
 chain tcp_packets {
 counter packets 36379 bytes 2103816
 }

 chain udp_packets {
 counter packets 10 bytes 1559
 }

 chain incoming_traffic {
 type filter hook input priority filter; policy accept;
 ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
 }
}
```

**tcp\_packets** 및 **udp\_packets** 체인의 카운터는 수신된 패킷과 바이트 수를 모두 표시합니다.

### 52.6.2. nftables에서 명명된 맵 사용

**nftables** 프레임워크는 명명된 맵을 지원합니다. 테이블 내에서 이러한 맵을 여러 규칙에 사용할 수 있습니다. 익명 맵에 비해 또 다른 이점은 해당 맵을 사용하는 규칙을 교체하지 않고도 명명된 맵을 업데이트할 수 있다는 것입니다.

명명된 맵을 생성할 때 요소 유형을 지정해야 합니다.

- 일치하는 부분에 **IPv4** 주소가 포함된 맵에 대한 **ipv4\_addr** (예: 192.0.2.1 ).
- 일치하는 부분에 **IPv 6** 주소가 포함된 맵에 대한 **ipv6\_addr** (예: 2001:db8:1::1).
- 일치하는 **MAC**(메시지 액세스 제어) 주소가 포함된 맵(예: 52:54:00:6b:66:42 )의 경우 **ether\_addr** 입니다.
- **inet\_proto**: 일치하는 부분이 인터넷 프로토콜 유형이 포함된 맵(예: tcp ).
- **inet\_service** 에 일치하는 인터넷 서비스 이름 포트 번호가 포함된 맵(예: ssh 또는 22 )의 경우 **inet\_service**입니다.
- 일치하는 부분에 패킷 표시가 포함된 맵에 대한 표시. 패킷 표시는 양의 **32비트** 정수 값일 수 있습니다(0~2 147483647).
- 일치하는 부분이 카운터 값이 포함된 맵의 카운터입니다. 카운터 값은 임의의 **64비트** 정수 값일 수 있습니다.
- 일치하는 맵에 할당량 값이 포함된 맵 할당량입니다. 할당량 값은 임의의 **64비트** 정수 값일 수 있습니다.

예제에서는 소스 **IP** 주소를 기반으로 들어오는 패킷을 허용하거나 삭제하는 방법을 설명합니다. 이름 지정된 맵을 사용하면 **IP** 주소 및 작업이 맵에 동적으로 저장되는 동안 이 시나리오를 구성하는 데 단일 규칙만 필요합니다. 이 절차에서는 맵에서 항목을 추가하고 제거하는 방법도 설명합니다.

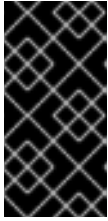
## 절차

1. 테이블 만들기. 예를 들어 **IPv4** 패킷을 처리하는 **example\_table** 이라는 테이블을 생성하려면 다음을 수행합니다.

```
nft add table ip example_table
```

2. 체인을 만듭니다. 예를 들어 **example\_table** 에 **example\_chain** 이라는 체인을 생성하려면 다음을 수행합니다.

```
nft add chain ip example_table example_chain { type filter hook input priority 0 ; }
```



### 중요

셸이 세미콜론을 명령의 끝으로 해석하지 않도록 하려면 세미콜론을 백슬래시로 이스케이프해야 합니다.

3.

빈 맵을 만듭니다. 예를 들어 IPv4 주소에 대한 맵을 생성하려면 다음을 수행합니다.

```
nft add map ip example_table example_map { type ipv4_addr : verdict ; }
```

4.

맵을 사용하는 규칙을 만듭니다. 예를 들어 다음 명령은 **example\_table** 의 **example\_chain** 에 규칙을 추가하여 **example\_map** 에 모두 정의된 IPv4 주소에 작업을 적용합니다.

```
nft add rule example_table example_chain ip saddr vmap @example_map
```

5.

IPv4 주소 및 해당 작업을 **example\_map** 에 추가:

```
nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

이 예제에서는 IPv4 주소의 작업 매핑을 정의합니다. 위에서 만든 규칙과 함께 방화벽은 192.0.2.1에서 패킷을 수락하고 192.0.2.2에서 패킷을 삭제합니다.

6.

선택적으로 다른 IP 주소 및 **action** 문을 추가하여 맵을 개선합니다.

```
nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7.

원하는 경우 맵에서 항목을 제거합니다.

```
nft delete element ip example_table example_map { 192.0.2.1 }
```

8.

선택적으로 규칙 세트를 표시합니다.

```
nft list ruleset
table ip example_table {
 map example_map {
 type ipv4_addr : verdict
```



```

elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
}

chain example_chain {
 type filter hook input priority filter; policy accept;
 ip saddr vmap @example_map
}
}

```

### 52.6.3. 추가 리소스

- **nft(8)** 도움말 페이지의 **Maps** 섹션

## 52.7. NFTABLES를 사용하여 포트 전달 구성

관리자는 포트 전달을 통해 특정 대상 포트에 전송된 패킷을 다른 로컬 또는 원격 포트에 전달할 수 있습니다.

예를 들어 웹 서버에 공용 IP 주소가 없는 경우 방화벽의 포트 80 및 443에서 수신되는 패킷을 웹 서버로 전달하는 방화벽에 포트 전달 규칙을 설정할 수 있습니다. 이 방화벽 규칙을 사용하면 인터넷의 사용자가 방화벽의 IP 또는 호스트 이름을 사용하여 웹 서버에 액세스할 수 있습니다.

### 52.7.1. 들어오는 패킷이 다른 로컬 포트에 전달

이 섹션에서는 포트 8022에서 들어오는 IPv4 패킷을 로컬 시스템의 포트 22로 전달하는 방법을 설명합니다.

#### 절차

1. **ip address family**를 사용하여 **nat**라는 테이블을 생성합니다.

```
nft add table ip nat
```

2. 테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

```
nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
```



## 참고

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않도록 하려면 **-- 옵션**을 **nft** 명령에 전달합니다.

3.

포트 **8022** 에서 들어오는 패킷을 로컬 포트 **22** 로 리디렉션 하는 사전 라우팅 체인에 규칙을 추가합니다.

```
nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

### 52.7.2. 특정 로컬 포트에서 들어오는 패킷을 다른 호스트로 전달

대상 네트워크 주소 변환(**DNAT**) 규칙을 사용하여 로컬 포트에서 들어오는 패킷을 원격 호스트로 전달할 수 있습니다. 이를 통해 인터넷 사용자는 개인 **IP** 주소가 있는 호스트에서 실행되는 서비스에 액세스할 수 있습니다.

절차에서는 로컬 포트 **443** 에서 들어오는 **IPv4** 패킷을 **192.0.2.1 IP** 주소를 사용하여 원격 시스템의 동일한 포트 번호로 전달하는 방법을 설명합니다.

#### 사전 요구 사항

•

패킷을 전달해야 하는 시스템에서 **root** 사용자로 로그인했습니다.

#### 절차

1.

**ip address family**를 사용하여 **nat** 라는 테이블을 생성합니다.

```
nft add table ip nat
```

2.

테이블에 **prerouting** 및 **postrouting** 체인을 추가합니다.

```
nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain ip nat postrouting { type nat hook postrouting priority 100 \; }
```



## 참고

셸이 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 않도록 하려면 -- 옵션을 **nft** 명령에 전달합니다.

3.

포트 **443** 에서 들어오는 패킷을 **192.0.2.1**의 동일한 포트로 리디렉션하는 이전 체인에 규칙을 추가합니다.

```
nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4.

나가는 트래픽을 마스커레이드하는 체인에 규칙을 추가합니다.

```
nft add rule ip nat postrouting daddr 192.0.2.1 masquerade
```

5.

패킷 전달 활성화:

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

## 52.8. NFTABLES를 사용하여 연결 양 제한

**nftables** 를 사용하여 연결 수를 제한하거나 지정된 커넥션 양을 설정하려는 **IP** 주소를 차단하여 너무 많은 시스템 리소스를 사용하지 않도록 할 수 있습니다.

### 52.8.1. nftables를 사용하여 연결 수 제한

**nft** 유틸리티의 **ct count** 매개 변수를 사용하면 관리자가 연결 수를 제한할 수 있습니다. 절차에서는 들어오는 연결을 제한하는 방법에 대한 기본적인 예를 설명합니다.

#### 사전 요구 사항

•

**example\_table**의 기본 **example\_chain** 이 있습니다.

#### 절차

1.

**IPv4** 주소에 대한 동적 세트를 생성합니다.

```
nft add set inet example_table example_meter { type ipv4_addr; flags dynamic ;}
```

2.

**IPv4** 주소에서 **SSH** 포트 (**22**)에 대한 동시 연결만 허용하는 규칙을 추가하고 동일한 **IP**의 모든 추가 연결을 거부합니다.

```
nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

3.

선택적으로 이전 단계에서 생성한 세트를 표시합니다.

```
nft list set inet example_table example_meter
table inet example_table {
 meter example_meter {
 type ipv4_addr
 size 65535
 elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2 }
 }
}
```

**elements** 항목은 현재 규칙과 일치하는 주소를 표시합니다. 이 예제에서 요소에는 **SSH** 포트에 대한 활성 연결이 있는 **IP** 주소가 나열됩니다. 출력에는 활성 연결 또는 연결이 거부된 경우 출력에 표시되지 않습니다.

### 52.8.2. 1분 내에 새로 들어오는 TCP 연결을 10개 이상 시도하는 IP 주소 차단

이 섹션에서는 1분 이내에 10개 이상의 **IPv4 TCP** 연결을 설정하는 호스트를 일시적으로 차단하는 방법에 대해 설명합니다.

#### 절차

1.

**ip address family**를 사용하여 **filter** 테이블을 생성합니다.

```
nft add table ip filter
```

2.

**filter** 테이블에 입력 체인을 추가합니다.

```
nft add chain ip filter input { type filter hook input priority 0 ; }
```

3.

소스 주소에서 1분 내에 10개 이상의 **TCP** 연결을 설정하는 모든 패킷을 삭제하는 규칙을 추가합니다.

```
nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip
saddr timeout 5m limit rate over 10/minute } drop
```

**timeout 5m** 매개 변수는 미터가 오래된 항목으로 채워지지 않도록 5분 후에 **nftables** 가 자동으로 항목을 제거함을 정의합니다.

## 검증

- 미터의 콘텐츠를 표시하려면 다음을 입력합니다.

```
nft list meter ip filter ratemeter
table ip filter {
 meter ratemeter {
 type ipv4_addr
 size 65535
 flags dynamic,timeout
 elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
 }
}
```

## 52.9. NFTABLES 규칙 디버깅

**nftables** 프레임워크는 패킷이 규칙과 일치하는 경우 규칙을 디버깅하기 위한 다양한 옵션을 제공합니다. 이 섹션에서는 이러한 옵션에 대해 설명합니다.

### 52.9.1. 카운터를 사용하여 규칙 생성

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다. 이 섹션에서는 카운터를 사용하여 새 규칙을 만드는 방법을 설명합니다.

- 기존 규칙에 카운터를 추가하는 절차에 대한 자세한 내용은 기존 규칙에 [카운터 추가를 참조하십시오](#).

## 사전 요구 사항

- 규칙을 추가할 체인이 있습니다.

## 절차

- 1.

**counter** 매개 변수를 사용하여 체인에 새 규칙을 추가합니다. 다음 예제에서는 포트 22에서 TCP 트래픽을 허용하고 이 규칙과 일치하는 패킷과 트래픽을 계산하는 카운터가 포함된 규칙을 추가합니다.

```
nft add rule inet example_table example_chain tcp dport 22 counter accept
```

2.

카운터 값을 표시하려면 다음을 수행합니다.

```
nft list ruleset
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport ssh counter packets 6872 bytes 105448565 accept
 }
}
```

### 52.9.2. 기존 규칙에 카운터 추가

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다. 이 섹션에서는 기존 규칙에 카운터를 추가하는 방법을 설명합니다.

- 

카운터를 사용하여 새 규칙을 추가하는 절차에 대한 자세한 내용은 [카운터를 사용하여 규칙 생성을 참조하십시오](#).

#### 사전 요구 사항

- 

카운터를 추가할 규칙입니다.

#### 절차

1.

핸들을 포함하여 체인에 규칙을 표시합니다.

```
nft --handle list chain inet example_table example_chain
table inet example_table {
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport ssh accept # handle 4
 }
}
```

2.

규칙을 **counter** 매개 변수로 교체하여 카운터를 추가합니다. 다음 예제에서는 이전 단계에 표시된 규칙을 교체하고 카운터를 추가합니다.

```
nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter
accept
```

3.

카운터 값을 표시하려면 다음을 수행합니다.

```
nft list ruleset
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport ssh counter packets 6872 bytes 105448565 accept
 }
}
```

### 52.9.3. 기존 규칙과 일치하는 패킷 모니터링

**nft monitor** 명령과 함께 **nftables**의 추적 기능을 사용하면 관리자가 규칙과 일치하는 패킷을 표시할 수 있습니다. 이 절차에서는 이 규칙과 일치하는 패킷을 모니터링하고 규칙에 대한 추적을 활성화하는 방법을 설명합니다.

#### 사전 요구 사항

•

카운터를 추가할 규칙입니다.

#### 절차

1.

핸들을 포함하여 체인에 규칙을 표시합니다.

```
nft --handle list chain inet example_table example_chain
table inet example_table {
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport ssh accept # handle 4
 }
}
```

2.

규칙을 대체하여 추적 기능을 추가하지만 메타 **nftrace set 1** 매개변수로 바꿉니다. 다음 예제에서는 이전 단계에 표시된 규칙을 교체하고 추적을 활성화합니다.

```
nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace
set 1 accept
```

3.

**nft monitor** 명령을 사용하여 추적을 표시합니다. 다음 예제에서는 명령의 출력을 필터링하여 **inet example\_table example\_chain** 이 포함된 항목만 표시합니다.

```
nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr
52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip
dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728
tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nfttrace set 1
accept (verdict accept)
...
```



주의

추적이 활성화된 규칙 수와 일치하는 트래픽 양에 따라 **nft monitor** 명령은 많은 출력을 표시할 수 있습니다. **grep** 또는 기타 유틸리티를 사용하여 출력을 필터링합니다.

## 52.10. NFTABLES 규칙 세트 백업 및 복원

이 섹션에서는 파일에 **nftables** 규칙을 백업하고 파일에서 규칙을 복원하는 방법을 설명합니다.

관리자는 규칙과 함께 파일을 사용하여 규칙을 다른 서버로 전송할 수 있습니다.

### 52.10.1. nftables 규칙 세트를 파일에 백업

이 섹션에서는 파일로 설정된 **nftables** 규칙을 백업하는 방법에 대해 설명합니다.

절차

- 

**nftables** 규칙을 백업하려면 다음을 수행합니다.

- 

**nft** 목록 **ruleset** 형식으로 생성된 형식으로 다음을 수행하십시오.

```
nft list ruleset > file.nft
```



- **JSON 형식으로 다음을 수행합니다.**

```
nft -j list ruleset > file.json
```

### 52.10.2. 파일에서 **nftables** 규칙 세트 복원

이 섹션에서는 **nftables** 규칙 세트를 복원하는 방법을 설명합니다.

#### 절차

- **nftables** 규칙을 복원하려면 다음을 수행합니다.
- 복원할 파일이 **nft** 목록 규칙 세트에 의해 생성되거나 **nft** 명령을 직접 포함하는 경우:

```
nft -f file.nft
```

- 복원할 파일이 **JSON** 형식인 경우:

```
nft -j -f file.json
```

### 52.11. 추가 리소스

- [Red Hat Enterprise Linux 8에서 nftables 사용](#)
- [iptables 뒤에는 무엇이 있습니까? 후자의 경우, 물론 nftables](#)
- [firewalld: 미래는 nftables입니다.](#)

## 53장. 고성능 트래픽 필터링에 XDP-FILTER를 사용하여 로더 공격 방지

**nftables** 와 같은 패킷 필터와 비교하여 **XDP(Trans Express Data Path)**는 네트워크 패킷을 처리하고 네트워크 인터페이스에서 바로 삭제합니다. 따라서 **XDP**는 방화벽 또는 기타 애플리케이션에 도달하기 전에 패키지에 대한 다음 단계를 결정합니다. 결과적으로 **XDP** 필터는 적은 리소스가 필요하며, **DDoS(Distributed DoS)** 공격으로부터 보호하기 위해 기존 패킷 필터보다 훨씬 높은 속도로 네트워크 패킷을 처리할 수 있습니다. 예를 들어, 테스트를 수행하는 동안 **Red Hat**은 단일 코어에서 초당 2천 6백만 네트워크 패킷을 삭제했으며 이는 동일한 하드웨어에서 **nftable**의 드롭 속도보다 훨씬 높습니다.

**xdp-filter** 유틸리티는 **XDP**를 사용하여 들어오는 네트워크 패킷을 허용하거나 삭제합니다. 특정 항목으로 또는 특정 항목의 트래픽을 필터링하는 규칙을 생성할 수 있습니다.

- **IP 주소**
- **MAC 주소**
- **포트**

**xdp-filter** 에 훨씬 높은 패킷 처리 속도가 있더라도 **nftables** 와 같은 기능은 없습니다. **XDP**를 사용하여 패킷 필터링을 시연하기 위해 **xdp-filter** 를 개념적 유틸리티로 고려하십시오. 또한 자체 **XDP** 애플리케이션을 작성하는 방법을 더 잘 이해할 수 있도록 유틸리티 코드를 사용할 수 있습니다.

### 중요

**AMD** 및 **Intel** 64비트 이외의 아키텍처에서 **xdp-filter** 유틸리티는 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(**SLA**)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있으며 프로덕션에는 사용하지 않는 것이 좋습니다. 이러한 미리보기를 통해 향후 제품 기능에 조기에 액세스할 수 있어 고객이 개발 과정에서 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능에 대한 지원 범위에 대한 자세한 내용은 **Red Hat** 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

### 53.1. XDP-FILTER 규칙과 일치하는 네트워크 패킷 삭제

이 섹션에서는 **xdp-filter** 를 사용하여 네트워크 패킷을 삭제하는 방법을 설명합니다.

- 특정 대상 포트로
- 특정 IP 주소에서
- 특정 MAC 주소

**xdp-filter**의 **allow** 정책은 모든 트래픽이 허용되도록 정의하고 필터는 특정 규칙과 일치하는 네트워크 패킷만 삭제합니다. 예를 들어 삭제하려는 패킷의 소스 IP 주소를 알고 있는 경우 이 방법을 사용합니다.

#### 사전 요구 사항

- **xdp-tools** 패키지가 설치됩니다.
- **XDP** 프로그램을 지원하는 네트워크 드라이버입니다.

#### 절차

1. **xdp-filter**를 로드하여 **enp1s0**과 같은 특정 인터페이스에서 수신되는 패킷을 처리합니다.

```
xdp-filter load enp1s0
```

기본적으로 **xdp-filter**는 허용 정책을 사용하며 유틸리티는 모든 규칙과 일치하는 트래픽만 삭제합니다.

선택적으로 **tcp,ipv4**, 또는 **ethernet**과 같은 특정 기능만 활성화하려면 **-f** 기능 옵션을 사용합니다. 모든 기능 대신 필요한 기능만 로드하면 패키지 처리 속도가 증가합니다. 여러 기능을 사용하려면 쉼표로 구분합니다.

명령이 오류와 함께 실패하면 네트워크 드라이버에서 **XDP** 프로그램을 지원하지 않습니다.

2. 해당 패킷과 일치하는 패킷 삭제에 규칙을 추가합니다. 예를 들면 다음과 같습니다.

- 포트 **22** 로 들어오는 패킷을 삭제하려면 다음을 입력합니다.

```
xdp-filter port 22
```

이 명령은 **TCP** 및 **UDP** 트래픽과 일치하는 규칙을 추가합니다. 특정 프로토콜과 일치하려면 **-p protocol** 옵션을 사용합니다.

- 192.0.2.1에서 들어오는 패킷을 삭제하려면 다음을 입력합니다.

```
xdp-filter ip 192.0.2.1 -m src
```

**xdp-filter** 는 IP 범위를 지원하지 않습니다.

- MAC 주소 **00:53:00:AA:07:BE** 에서 들어오는 패킷을 삭제하려면 다음을 입력합니다.

```
xdp-filter ether 00:53:00:AA:07:BE -m src
```

#### 검증 단계

- 삭제 및 허용된 패킷에 대한 통계를 표시하려면 다음 명령을 사용합니다.

```
xdp-filter status
```

#### 추가 리소스

- **XDP-filter(8)** 도움말 페이지
- **xdp-filter** 코드에 관심이 있으며 개발자인 경우 **Red Hat Customer Portal**에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치하십시오.

## 53.2. XDP-FILTER 규칙과 일치하는 네트워크 패킷을 제외하고 모든 네트워크 패킷 삭제

이 섹션에서는 **xdp-filter** 를 사용하여 네트워크 포트만 허용하는 방법에 대해 설명합니다.

- 특정 대상 포트로 이동
- 특정 IP 주소에서 특정 IP 주소로
- 특정 MAC 주소에서 특정 MAC 주소로

이렇게 하려면 필터가 특정 규칙과 일치하는 패킷을 제외한 모든 네트워크 패킷을 삭제하도록 정의하는 **xdp-filter**의 거부 정책을 사용합니다. 예를 들어 삭제하려는 패킷의 소스 IP 주소를 모르는 경우 이 방법을 사용합니다.



#### 주의

인터페이스에서 **xdp-filter**를 로드할 때 거부 하도록 기본 정책을 설정하면 커널이 특정 트래픽을 허용하는 규칙을 생성할 때까지 이 인터페이스에서 모든 패킷을 즉시 삭제합니다. 시스템에서 잠금을 방지하려면 로컬로 명령을 입력하거나 다른 네트워크 인터페이스를 통해 호스트에 연결합니다.

#### 사전 요구 사항

- **xdp-tools** 패키지가 설치됩니다.
- 로컬로 로그인하거나 트래픽을 필터링하지 않으려는 네트워크 인터페이스를 사용합니다.
- **XDP** 프로그램을 지원하는 네트워크 드라이버입니다.

#### 절차

1. **xdp-filter**를 로드하여 **enp1s0**과 같은 특정 인터페이스에서 패킷을 처리합니다.

```
xdp-filter load enp1s0 -p deny
```

선택적으로 **tcp,ipv4**, 또는 **ethernet**과 같은 특정 기능만 활성화하려면 **-f** 기능 옵션을 사용합

니다. 모든 기능 대신 필요한 기능만 로드하면 패키지 처리 속도가 증가합니다. 여러 기능을 사용하려면 쉼표로 구분합니다.

명령이 오류와 함께 실패하면 네트워크 드라이버에서 **XDP** 프로그램을 지원하지 않습니다.

2.

패킷을 허용하는 규칙을 추가합니다. 예를 들면 다음과 같습니다.

- 

패킷을 **22** 로 허용하려면 다음을 입력합니다.

```
xdp-filter port 22
```

이 명령은 **TCP** 및 **UDP** 트래픽과 일치하는 규칙을 추가합니다. 특정 프로토콜만 일치시키려면 명령에 **-p** 프로토콜 옵션을 전달합니다.

- 

**192.0.2.1** 에 패킷을 허용하려면 다음을 입력합니다.

```
xdp-filter ip 192.0.2.1
```

**xdp-filter** 는 **IP** 범위를 지원하지 않습니다.

- 

패킷이 **MAC** 주소 **00:53:00:AA:07:BE** 으로 허용하려면 다음을 입력합니다.

```
xdp-filter ether 00:53:00:AA:07:BE
```



중요

**xdp-filter** 유틸리티는 상태 저장 패킷 검사를 지원하지 않습니다. 이를 위해서는 **-m mode** 옵션을 사용하여 모드를 설정하지 않거나 시스템이 수신하는 들어오는 트래픽을 나가는 트래픽에 응답하도록 명시적 규칙을 추가해야 합니다.

검증 단계

- 

삭제 및 허용된 패킷에 대한 통계를 표시하려면 다음 명령을 사용합니다.

```
xdp-filter status
```

### 추가 리소스

- ***XDP-filter(8)*** 도움말 페이지.
- 개발자이고 ***xdp-filter*** 코드에 관심이 있는 경우 ***Red Hat Customer Portal***에서 해당 소스 ***RPM(SRPM)***을 다운로드하여 설치하십시오.

## 54장. DPDK 시작하기

**DPDK**(데이터 플레인 개발 키트)는 사용자 공간에서 패키지 처리를 가속화하는 라이브러리 및 네트워크 드라이버를 제공합니다.

관리자는 예를 들어 가상 머신에서 **SR-IOV(Single Root I/O Virtualization)**를 사용하여 대기 시간을 줄이고 I/O 처리량을 늘리기 위해 **DPDK**를 사용합니다.



참고

**Red Hat은 실험적 DPDK API를 지원하지 않습니다.**

### 54.1. DPDK 패키지 설치

이 섹션에서는 **dpdk** 패키지를 설치하는 방법에 대해 설명합니다.

사전 요구 사항

- **Red Hat Enterprise Linux가 설치되어 있어야 합니다.**
- 유효한 서브스크립션이 호스트에 할당됩니다.

절차

- **yum** 유틸리티를 사용하여 **dpdk** 패키지를 설치합니다.

```
yum install dpdk
```

### 54.2. 추가 리소스

- [네트워크 어댑터 Fast Datapath 기능 지원 매트릭스](#)



## 55장. RHEL에서 EBPF 네트워킹 기능 이해

**eBPF(Extended Berkeley Packet Filter)**는 커널 공간에서 코드를 실행할 수 있는 커널 내 가상 시스템입니다. 이 코드는 제한된 기능 집합만 액세스할 수 있는 제한된 샌드박스 환경에서 실행됩니다.

네트워킹에서는 **eBPF**를 사용하여 커널 패킷 처리를 보완하거나 교체할 수 있습니다. 사용하는 후크에 따라 **eBPF** 프로그램에는 다음이 포함됩니다. 예를 들면 다음과 같습니다.

- 패킷 데이터 및 메타 데이터에 대한 읽기 및 쓰기 액세스
- 소켓 및 경로를 찾을 수 있습니다.
- 소켓 옵션을 설정할 수 있습니다
- 패킷을 리디렉션할 수 있습니다

### 55.1. RHEL의 네트워킹 EBPF 기능 개요

확장 **eBPF(extended Berkeley Packet Filter)** 네트워킹 프로그램을 **RHEL**의 다음 후크에 연결할 수 있습니다.

- **XDP(Express Express Data Path):** 커널 네트워킹 스택이 처리하기 전에 수신된 패킷에 대한 초기 액세스를 제공합니다.
- **Direct-action** 플래그가 있는 **TC eBPF category:** 수신 및 송신에서 강력한 패킷 처리를 제공합니다.
- **Control Groups 버전 2 (cgroup v2):** 제어 그룹의 프로그램에서 수행하는 소켓 기반 작업을 필터링하고 재정의할 수 있습니다.
- **소켓 필터링:** 소켓에서 수신된 패킷 필터링을 활성화합니다. 이 기능은 **cBPF(Certified Berkeley Packet Filter)**에서도 제공되었지만 **eBPF** 프로그램을 지원하도록 확장되었습니다.

- 스트림 구문 분석기: 스트림을 개별 메시지로 분할하고 필터링하고 소켓으로 리디렉션할 수 있습니다.
- **SO\_REUSEPORT** 소켓 선택: **reuseport** 소켓 그룹에서 수신 소켓을 프로그래밍 방식으로 선택할 수 있습니다.
- 흐름 분산 장치: 커널이 특정 상황에서 패킷 헤더를 구문 분석하는 방법을 재정의할 수 있습니다.
- **TCP** 혼잡 제어 콜백: 사용자 지정 **TCP** 혼잡 제어 알고리즘을 구현할 수 있습니다.
- 캡슐화가 있는 경로: 사용자 지정 터널 캡슐화를 생성할 수 있습니다.

Red Hat은 RHEL에서 사용 가능하며 여기에 설명된 모든 **eBPF** 기능을 지원하지 않습니다. 자세한 내용 및 개별 후크의 지원 상태는 [RHEL 8 릴리스 노트](#) 및 다음 개요를 참조하십시오.

## XDP

**BPF\_PROG\_TYPE\_XDP** 유형의 프로그램을 네트워크 인터페이스에 연결할 수 있습니다. 그런 다음 커널은 커널 네트워크 스택 처리를 시작하기 전에 수신된 패킷에서 프로그램을 실행합니다. 따라서 빠른 패킷 삭제와 같은 특정 상황에서 빠른 패킷 전달을 통해 분산된 서비스 거부(DDoS) 공격 및 로드 밸런싱 시나리오에 대한 빠른 패킷 리디렉션을 방지할 수 있습니다.

**XDP**를 다양한 형태의 패킷 모니터링 및 샘플링에도 사용할 수 있습니다. 커널을 사용하면 **XDP** 프로그램에서 패킷을 수정하고 커널 네트워크 스택에 추가 처리를 위해 패킷을 전달할 수 있습니다.

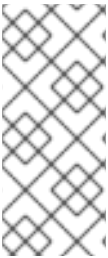
사용 가능한 **XDP** 모드는 다음과 같습니다.

- 네이티브(드라이버) **XDP**: 커널은 패킷 수신 중에 가능한 가장 빠른 지점에서 프로그램을 실행합니다. 현재 커널은 패킷을 구문 분석하지 않았으므로 커널에서 제공하는 메타데이터를 사용할 수 없습니다. 이 모드에서는 네트워크 인터페이스 드라이버가 **XDP**를 지원해야 하지만 일부 드라이버가 이 기본 모드를 지원하지는 않습니다.
- 일반 **XDP**: 커널 네트워크 스택은 처리 초기에 **XDP** 프로그램을 실행합니다. 이 시점에서 커널 데이터 구조가 할당되었으며 패킷이 미리 처리되었습니다. 패킷을 삭제하거나 리디렉션해야 하

는 경우 네이티브 모드에 비해 상당한 오버헤드가 필요합니다. 그러나 일반 모드에서는 네트워크 인터페이스 드라이버가 지원할 필요가 없으며 모든 네트워크 인터페이스에서 작동합니다.

- 오프로드된 **XDP**: 커널은 호스트 **CPU** 대신 네트워크 인터페이스에서 **XDP** 프로그램을 실행합니다. 이 작업에는 특정 하드웨어가 필요하며 이 모드에서 특정 **eBPF** 기능만 사용할 수 있습니다.

**RHEL**에서 **libxdp** 라이브러리를 사용하여 모든 **XDP** 프로그램을 로드합니다. 이 라이브러리를 사용하면 시스템 제어 **XDP** 사용을 사용할 수 있습니다.



#### 참고

현재 **XDP** 프로그램에 대한 몇 가지 시스템 구성 제한 사항이 있습니다. 예를 들어 수신 인터페이스에서 특정 하드웨어 오프로드 기능을 비활성화해야 합니다. 또한 기본 모드를 지원하는 모든 드라이버에서 일부 기능을 사용할 수 있는 것은 아닙니다.

**RHEL 8.5**에서 **Red Hat**은 다음 조건이 모두 적용되는 경우에만 **XDP** 기능을 지원합니다.

- **AMD** 또는 **Intel 64비트** 아키텍처에서 **XDP** 프로그램을 로드합니다.
- **libxdp** 라이브러리를 사용하여 프로그램을 커널에 로드합니다.
- **XDP** 프로그램은 **XDP** 하드웨어 오프로딩을 사용하지 않습니다.

또한 **Red Hat**은 지원되지 않는 기술 프리뷰로 **XDP** 기능을 다음과 같이 사용합니다.

- **AMD** 및 **Intel 64비트**가 아닌 아키텍처에 **XDP** 프로그램 로드. **libxdp** 라이브러리는 **AMD** 및 **Intel 64비트** 이외의 아키텍처에서는 사용할 수 없습니다.
- **XDP** 하드웨어 오프로딩.

#### AF\_XDP

패킷을 지정된 **AF\_XDP** 소켓으로 필터링 및 리디렉션하는 **XDP** 프로그램을 사용하면 **AF\_XDP** 프로토콜 제품군에서 사용자 공간으로 패킷을 빠르게 복사할 수 있습니다.

**RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.**

## 트래픽 제어

트래픽 제어(tc) 하위 시스템은 다음과 같은 유형의 **eBPF** 프로그램을 제공합니다.

- **BPF\_PROG\_TYPE\_SCHED\_CLS**
- **BPF\_PROG\_TYPE\_SCHED\_ACT**

이러한 유형을 사용하면 **eBPF**에서 사용자 지정 **tc** 분류기 및 **tc** 작업을 작성할 수 있습니다. **tc** 에코시스템과 함께 이는 강력한 패킷 처리 기능을 제공하며 여러 컨테이너 네트워킹 오케스트레이션 솔루션의 핵심 요소입니다.

대부분의 경우 **classifier**만 **direct-action** 플러그와 마찬가지로 **eBPF** 분류기에서 동일한 **eBPF** 프로그램에서 직접 작업을 실행할 수 있습니다. **clsact Queueing Discipline(qdisc)**은 수신 측에서 이를 활성화하도록 설계되었습니다.

흐름 **dissector eBPF** 프로그램을 사용하면 꽃 과 같은 다른 **qdiscs** 및 **tc constructors**의 작동에 영향을 미칠 수 있습니다.

**tc** 용 **eBPF**는 **RHEL 8.2** 이상에서 완전히 지원됩니다.

## 소켓 필터

소켓에서 수신된 패킷 필터링을 위해 여러 유틸리티에서 **CBPF(Certified Berkeley Packet Filter)**를 사용하거나 사용했습니다. 예를 들어 **tcpdump** 유틸리티를 사용하면 사용자가 표현식을 지정할 수 있으며, **which tcpdump**는 **cBPF** 코드로 변환됩니다.

**cBPF**의 대안으로 커널은 동일한 목적을 위해 **BPF\_PROG\_TYPE\_SOCKET\_FILTER** 유형의 **eBPF** 프로그램을 허용합니다.

**RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.**

## 컨트롤 그룹

RHEL에서는 **cgroup**에 연결할 수 있는 여러 유형의 **eBPF** 프로그램을 사용할 수 있습니다. 지정된 **cgroup**의 프로그램이 작업을 수행할 때 커널은 이 프로그램을 실행합니다. **cgroup** 버전 2만 사용할 수 있습니다.

RHEL에서는 다음 네트워킹 관련 **cgroup eBPF** 프로그램을 사용할 수 있습니다.

- **BPF\_PROG\_TYPE SOCK\_OPS:** 커널은 다양한 **TCP** 이벤트에서 이 프로그램을 호출합니다. 프로그램은 사용자 지정 **TCP** 헤더 옵션을 포함하여 커널 **TCP** 스택의 동작을 조정할 수 있습니다.
- **BPF\_PROG\_TYPE CGROUP SOCK\_ADDR:** 커널은 연결, **bind**, **sendto**, **recvmsg**, **getpeername**, **getsockname** 작업 중에 이 프로그램을 호출합니다. 이 프로그램을 통해 **IP** 주소 및 포트를 변경할 수 있습니다. 이는 **eBPF**에서 소켓 기반 **NAT**(네트워크 주소 변환)를 구현할 때 유용합니다.
- **BPF\_PROG\_TYPE CGROUP SOCKOPT:** 커널은 **setsockopt** 및 **getsockopt** 작업 중에 이 프로그램을 호출하고 옵션을 변경할 수 있습니다.
- **BPF\_PROG\_TYPE CGROUP SOCK:** 커널은 소켓 생성, 소켓 해제 및 주소 바인딩 중에 이 프로그램을 호출합니다. 이러한 프로그램을 사용하여 작업을 허용하거나 거부하거나 통계에 대한 소켓 생성을 검사하는 데만 사용할 수 있습니다.
- **BPF\_PROG\_TYPE CGROUP SKB:** 이 프로그램은 수신 및 송신에서 개별 패킷을 필터링하고 패킷을 수락하거나 거부할 수 있습니다.
- **BPF\_PROG\_TYPE CGROUP SYSCTL:** 이 프로그램을 사용하면 시스템 제어(**sysctl**)에 대한 액세스를 필터링할 수 있습니다.
- **BPF\_CGROUP\_INET4\_GETPEERNAME, BPF\_CGROUP\_INET6\_GETPEERNAME, BPF\_CGROUP\_INET4\_GETSOCKNAME, BPF\_CGROUP\_INET\_GETSOCKNAME:** 이러한 프로그램을 사용하면 **getsockname** 및 **getpeername** 시스템 호출의 결과를 덮어쓸 수 있습니다. 이는 **eBPF**에서 소켓 기반 **NAT**(네트워크 주소 변환)를 구현할 때 유용합니다.

RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.

스트림 구문 분석기는 특수 **eBPF** 맵에 추가된 소켓 그룹에서 작동합니다. 그런 다음 **eBPF** 프로그램은 커널이 해당 소켓을 수신하거나 보내는 패킷을 처리합니다.

**RHEL**에서는 다음 스트림 구문 분석기 **eBPF** 프로그램을 사용할 수 있습니다.

- **BPF\_PROG\_TYPE\_SK\_SKB:** **eBPF** 프로그램은 소켓에서 수신한 패킷을 개별 메시지로 구문 분석하고 커널에 해당 메시지를 삭제하거나 그룹의 다른 소켓으로 보내도록 지시합니다.
- **BPF\_PROG\_TYPE\_SK\_MSG:** 이 프로그램은 송신 메시지를 필터링합니다. **eBPF** 프로그램은 패킷을 개별 메시지로 구문 분석하고 패킷을 승인 또는 거부합니다.

**RHEL 8.5**에서 **Red Hat**은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.

#### **SO\_REUSEPORT** 소켓 선택

이 소켓 옵션을 사용하여 여러 소켓을 동일한 **IP** 주소 및 포트에 바인딩할 수 있습니다. **eBPF**가 없으면 커널은 연결 해시를 기반으로 수신 소켓을 선택합니다. **BPF\_PROG\_TYPE\_SK\_REUSEPORT** 프로그램을 사용하면 수신 소켓 선택을 완전히 프로그래밍할 수 있습니다.

**RHEL 8.5**에서 **Red Hat**은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.

#### 흐름 분산 섹터

커널이 전체 프로토콜 디코드를 통과하지 않고 패킷 헤더를 처리해야 하는 경우 차단됩니다. 예를 들어 **tc** 하위 시스템, 다중 경로 라우팅에서 또는 패킷 해시를 계산할 때 발생합니다. 이 경우 커널은 패킷 헤더를 구문 분석하고 패킷 헤더의 정보로 내부 구조를 채웁니다. **BPF\_PROG\_TYPE\_FLOW\_DISSECTOR** 프로그램을 사용하여 이 내부 구문 분석을 교체할 수 있습니다. **RHEL**의 **eBPF**에서 **IPv4** 및 **IPv6**을 통해서만 **TCP** 및 **UDP**를 차단할 수 있습니다.

**RHEL 8.5**에서 **Red Hat**은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.

#### **TCP** 혼잡 제어

**struct tcp\_congestion\_oops** 콜백을 구현하는 **BPF\_PROG\_TYPE\_STRUCT\_OPS** 프로그램을 사용하여 사용자 지정 **TCP** 혼잡 제어 알고리즘을 작성할 수 있습니다. 이러한 방식으로 구현된 알고리즘을 시스템에서 기본 제공 커널 알고리즘과 함께 사용할 수 있습니다.

**RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.**

#### 캡슐화가 있는 경로

다음 **eBPF** 프로그램 유형 중 하나를 터널 캡슐화 특성으로 라우팅 테이블의 경로에 연결할 수 있습니다.

- **BPF\_PROG\_TYPE\_LWT\_IN**
- **BPF\_PROG\_TYPE\_LWT\_OUT**
- **BPF\_PROG\_TYPE\_LWT\_XMIT**

이러한 **eBPF** 프로그램의 기능은 특정 터널 구성으로 제한되며 일반 캡슐화 또는 캡슐화 해제 솔루션을 생성할 수 없습니다.

**RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.**

#### 소켓 조회

바인딩 시스템 호출의 제한을 우회하려면 **BPF\_PROG\_TYPE\_SK\_LOOKUP** 유형의 **eBPF** 프로그램을 사용하십시오. 이러한 프로그램은 새로 들어오는 **TCP** 연결에 대해 수신 대기 소켓 또는 **UDP** 패킷에 대해 연결되지 않은 소켓을 선택할 수 있습니다.

**RHEL 8.5에서 Red Hat은 이 기능을 지원되지 않는 기술 프리뷰로 제공합니다.**

## 55.2. 네트워크 카드별 XDP 기능 개요

다음은 **XDP** 사용 가능 네트워크 카드 및 해당 카드와 함께 사용할 수 있는 **XDP** 기능에 대한 개요입니다.

| 네트워크 카드                        | 드라이버       | Basic | 리디렉션 | 대상  | HW 오프로드 | 0-copy |
|--------------------------------|------------|-------|------|-----|---------|--------|
| Amazon Elastic Network Adapter | <b>ena</b> | 제공됨   | 제공됨  | 제공됨 | 아니요     | 아니요    |

| 네트워크 카드                                        | 드라이버             | Basic   | 리디렉션    | 대상                 | HW 오피로드 | O-copy  |
|------------------------------------------------|------------------|---------|---------|--------------------|---------|---------|
| Broadcom NetXtreme-C/E<br>10/25/40/50 기가비트 이더넷 | <b>bnxt_en</b>   | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [a]<br>[b] | 아니<br>요 | 아니<br>요 |
| Cavium Thunder Virtual 함수                      | <b>nicvf</b>     | 제공<br>됨 | 아니<br>요 | 아니<br>요            | 아니<br>요 | 아니<br>요 |
| Intel® 이더넷 컨트롤러 710 제품군                        | <b>i40e</b>      | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [a]<br>[b] | 아니<br>요 | 제공<br>됨 |
| Intel® Ethernet Connection E800 시리즈            | <b>ice</b>       | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [a]<br>[b] | 아니<br>요 | 제공<br>됨 |
| Intel® PCI Express 기가비트 어댑터                    | <b>igb</b>       | 제공<br>됨 | 제공<br>됨 | 예 [a]              | 아니<br>요 | 아니<br>요 |
| Intel® 10GbE PCI Express 어댑터                   | <b>ixgbe</b>     | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [a]<br>[b] | 아니<br>요 | 제공<br>됨 |
| Intel® 10GbE PCI Express 가상 기능 이더넷             | <b>ixgbev</b>    | 제공<br>됨 | 아니<br>요 | 아니<br>요            | 아니<br>요 | 아니<br>요 |
| Mellanox Technologies 1/10/40Gbit 이더넷          | <b>mlx4_en</b>   | 제공<br>됨 | 아니<br>요 | 아니<br>요            | 아니<br>요 | 아니<br>요 |
| Mellanox 5세대 네트워크 어댑터<br>(ConnectX 시리즈)        | <b>mlx5_core</b> | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [b]        | 아니<br>요 | 제공<br>됨 |
| Netronome® NFP4000/NFP6000 NIC                 | <b>nfp</b>       | 제공<br>됨 | 아니<br>요 | 아니<br>요            | 제공<br>됨 | 아니<br>요 |
| Qlogic QED 25/40/100Gb 이더넷 NIC                 | <b>qede</b>      | 제공<br>됨 | 제공<br>됨 | 제공<br>됨            | 아니<br>요 | 아니<br>요 |
| Solarflare<br>SFC9000/SFC9100/EF100-family     | <b>sfc</b>       | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 [b]        | 아니<br>요 | 아니<br>요 |
| Microsoft Hyper-V 가상 네트워크                      | <b>hv_netvsc</b> | 제공<br>됨 | 아니<br>요 | 아니<br>요            | 아니<br>요 | 아니<br>요 |



| 네트워크 카드                                                                                                                                  | 드라이버              | Basic   | 리디렉션    | 대상                                                 | HW 오프로드 | 0-copy  |
|------------------------------------------------------------------------------------------------------------------------------------------|-------------------|---------|---------|----------------------------------------------------|---------|---------|
| 범용 TUN/TAP 장치                                                                                                                            | <b>tun</b>        | 제공<br>됨 | 제공<br>됨 | 제공<br>됨                                            | 아니<br>요 | 아니<br>요 |
| 가상 이더넷 쌍 장치                                                                                                                              | <b>veth</b>       | 제공<br>됨 | 제공<br>됨 | 제공<br>됨                                            | 아니<br>요 | 아니<br>요 |
| QEMU Virtio 네트워크                                                                                                                         | <b>virtio_net</b> | 제공<br>됨 | 제공<br>됨 | 제공<br>됨 <a href="#">[a]</a><br><a href="#">[b]</a> | 아니<br>요 | 아니<br>요 |
| <p><a href="#">[a]</a> XDP 프로그램이 인터페이스에 로드된 경우에만 해당합니다.</p> <p><a href="#">[b]</a> 가장 큰 CPU 인덱스와 같거나 크거나 같은 다수의 XDP TX 큐가 할당되어야 합니다.</p> |                   |         |         |                                                    |         |         |

**범례:**

- **기본: 기본 반환 코드 지원: *DROP,PASS,ABORTED, TX*.**
- **리디렉션: *REDIRECT* 반환 코드를 지원합니다.**
- **대상: *REDIRECT* 반환 코드의 대상이 될 수 있습니다.**
- **HW 오프로드: *XDP* 하드웨어 오프로드 지원.**
- **0 복사: *AF\_XDP* 프로토콜 제품군에 대한 제로 복사 모드를 지원합니다.**

## 56장. BPF 컴파일러 컬렉션을 사용한 네트워크 추적

이 섹션에서는 **BCC(BPF Compiler Collection)**의 개념, **BCC**를 설치하는 방법 및 **bcc-tools** 패키지에 제공하는 사전 생성된 스크립트를 사용하여 다양한 네트워크 추적 작업을 수행하는 방법에 대해 설명합니다. 이러한 모든 스크립트는 **--ebpf** 매개변수를 지원하여 유틸리티가 커널에 업로드하는 **eBPF** 코드를 표시합니다. 코드를 사용하여 **eBPF** 스크립트를 작성하는 방법에 대해 자세히 알아볼 수 있습니다.

### 56.1. BCC 소개

**BCC(BPF Compiler Collection)**는 **eBPF(extended Berkeley Packet Filter)** 프로그램을 쉽게 생성할 수 있는 라이브러리입니다. **eBPF** 프로그램의 주요 유틸리티는 오버헤드나 보안 문제가 발생하지 않고 **OS** 성능 및 네트워크 성능을 분석하는 것입니다.

**BCC**는 사용자가 **eBPF**의 심층적인 기술 세부 사항을 알고 있어야 하는 필요성을 없애고, 미리 생성된 **eBPF** 프로그램을 사용하는 **bcc-tools** 패키지와 같이 많은 즉시 사용 가능한 시작점을 제공합니다.



#### 참고

**eBPF** 프로그램은 디스크 **I/O**, **TCP** 연결 및 프로세스 생성과 같은 이벤트에서 트리거됩니다. 커널의 안전한 가상 시스템에서 실행되므로 커널이 충돌하거나 반복하거나 응답하지 않게 되는 프로그램이 발생할 가능성이 낮습니다.

### 56.2. BCC-TOOLS 패키지 설치

이 섹션에서는 **BCC(BPF Compiler Collection)** 라이브러리도 종속성으로 설치하는 **bcc-tools** 패키지를 설치하는 방법에 대해 설명합니다.

#### 절차

1.

**bcc-tools** 를 설치합니다.

```
yum install bcc-tools
```

**BCC** 툴은 **/usr/share/bcc/tools/** 디렉토리에 설치됩니다.

2.

선택적으로 툴을 검사합니다.

```
ll /usr/share/bcc/tools/
...
-rwxr-xr-x. 1 root root 4198 Dec 14 17:53 dcsnoop
-rwxr-xr-x. 1 root root 3931 Dec 14 17:53 dcstat
-rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
-rw-r--r--. 1 root root 7105 Dec 14 17:53 deadlock_detector.c
drwxr-xr-x. 3 root root 8192 Mar 11 10:28 doc
-rwxr-xr-x. 1 root root 7588 Dec 14 17:53 execsnoop
-rwxr-xr-x. 1 root root 6373 Dec 14 17:53 ext4dist
-rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
...
```

위의 목록의 **doc** 디렉터리에는 각 툴에 대한 문서가 포함되어 있습니다.

### 56.3. 커널의 허용 대기열에 추가된 TCP 연결 표시

커널이 **TCP 3-way** 핸드셰이크에서 **ACK** 패킷을 수신하면 커널은 연결 상태가 **meanBLISHED**로 변경된 후 **SYN** 대기열에서 수락 대기열로 이동합니다. 따라서 이 큐에는 성공적인 **TCP** 연결만 표시됩니다.

**tcpaccept** 유틸리티는 **eBPF** 기능을 사용하여 커널이 수락 대기열에 추가하는 모든 연결을 표시합니다. 유틸리티는 패킷을 캡처하여 필터링하는 대신 커널의 **accept()** 함수를 추적하므로 가볍습니다. 예를 들어 일반적인 문제 해결에 **tcpaccept**를 사용하여 서버가 수락한 새 연결을 표시합니다.

#### 절차

1. 커널 허용 대기열 추적을 시작하려면 다음 명령을 입력합니다.

```
/usr/share/bcc/tools/tcpaccept
PID COMM IP RADDR RPORT LADDR LPORT
843 sshd 4 192.0.2.17 50598 192.0.2.1 22
1107 ns-slapd 4 198.51.100.6 38772 192.0.2.1 389
1107 ns-slapd 4 203.0.113.85 38774 192.0.2.1 389
...
```

커널이 연결을 수락할 때마다 **tcpaccept**는 연결 세부 정보를 표시합니다.

2. **Ctrl +C**를 눌러 추적 프로세스를 중지합니다.

#### 추가 리소스

- [tcpaccept\(8\) 도움말 페이지](#)
- [/usr/share/bcc/tools/doc/tcpaccept\\_example.txt](#)

#### 56.4. 나가는 TCP 연결 시도 추적

**tcpconnect** 유틸리티는 **eBPF** 기능을 사용하여 나가는 **TCP** 연결 시도를 추적합니다. 유틸리티 출력에는 실패한 연결도 포함되어 있습니다.

**tcpconnect** 유틸리티는 추적(예: 패킷을 캡처하여 필터링) 대신 커널의 **connect()** 함수를 추적하므로 가볍습니다.

##### 절차

1. 다음 명령을 입력하여 나가는 모든 연결을 표시하는 추적 프로세스를 시작합니다.

```
/usr/share/bcc/tools/tcpconnect
PID COMM IP SADDR DADDR DPORT
31346 curl 4 192.0.2.1 198.51.100.16 80
31348 telnet 4 192.0.2.1 203.0.113.231 23
31361 isc-worker00 4 192.0.2.1 192.0.2.254 53
...
```

커널이 발신 연결을 처리할 때마다 **tcpconnect** 는 연결의 세부 정보를 표시합니다.

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

##### 추가 리소스

- [TCPConnect\(8\) 도움말 페이지](#)
- [/usr/share/bcc/tools/doc/tcpconnect\\_example.txt](#)

#### 56.5. 나가는 TCP 연결의 대기 시간 측정

**TCP** 연결 대기 시간은 연결을 설정하는 데 걸리는 시간입니다. 여기에는 일반적으로 애플리케이션 런타임이 아닌 커널 **TCP/IP** 처리와 네트워크 왕복 시간이 포함됩니다.

**tcpconnl**at 유틸리티는 **eBPF** 기능을 사용하여 전송된 **SYN** 패킷과 수신된 응답 패킷 사이의 시간을 측정합니다.

## 절차

1. 나가는 연결의 대기 시간 측정을 시작합니다.

```
/usr/share/bcc/tools/tcpconnl
PID COMM IP SADDR DADDR DPORT LAT(ms)
32151 isc-worker00 4 192.0.2.1 192.0.2.254 53 0.60
32155 ssh 4 192.0.2.1 203.0.113.190 22 26.34
32319 curl 4 192.0.2.1 198.51.100.59 443 188.96
...
```

커널이 발신 연결을 처리할 때마다 **tcpconnl**at 는 커널이 응답 패킷을 수신한 후 연결 세부 정보를 표시합니다.

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

## 추가 리소스

- **tcpconnl(8)** 도움말 페이지
- `/usr/share/bcc/tools/doc/tcpconnl_example.txt`

## 56.6. 커널에서 삭제한 TCP 패킷 및 세그먼트에 대한 세부 정보 표시

**tcpdrop** 유틸리티를 사용하면 관리자가 커널이 삭제한 **TCP** 패킷 및 세그먼트에 대한 세부 정보를 표시할 수 있습니다. 이 유틸리티를 사용하여 원격 시스템이 타이머 기반 재전송을 보낼 수 있는 삭제된 패킷의 속도를 디버깅합니다. 삭제된 패킷 및 세그먼트의 비율이 서버의 성능에 영향을 줄 수 있습니다.

리소스를 많이 사용하는 패킷을 캡처하고 필터링하는 대신 **tcpdrop** 유틸리티는 **eBPF** 기능을 사용하여 커널에서 직접 정보를 검색합니다.

## 절차

1.

삭제된 **TCP** 패킷 및 세그먼트에 대한 세부 정보를 표시하려면 다음 명령을 입력합니다.

```
/usr/share/bcc/tools/tcpdrop
TIME PID IP SADDR:SPORT > DADDR:DPORT STATE (FLAGS)
13:28:39 32253 4 192.0.2.85:51616 > 192.0.2.1:22 CLOSE_WAIT (FIN|ACK)
b'tcp_drop+0x1'
b'tcp_data_queue+0x2b9'
...

13:28:39 1 4 192.0.2.85:51616 > 192.0.2.1:22 CLOSE (ACK)
b'tcp_drop+0x1'
b'tcp_rcv_state_process+0xe2'
...
```

커널이 **TCP** 패킷 및 세그먼트를 삭제할 때마다 **tcpdrop** 은 삭제된 패키지를 유발하는 커널 스택 추적을 포함하여 연결의 세부 정보를 표시합니다.

2.

**Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

추가 리소스

- [tcpdrop\(8\) 도움말 페이지](#)
- [/usr/share/bcc/tools/doc/tcpdrop\\_example.txt](#)

## 56.7. TCP 세션 추적

**tcplife** 유틸리티는 **eBPF**를 사용하여 열려 있고 닫히는 **TCP** 세션을 추적하고 출력 행을 출력하여 각 세션을 요약합니다. 관리자는 **tcplife** 를 사용하여 전송된 트래픽 양을 식별할 수 있습니다.

이 섹션의 예제에서는 다음 정보를 검색하기 위해 포트 **22 (SSH)**에 대한 연결을 표시하는 방법을 설명합니다.

- 로컬 프로세스 ID(PID)
- 로컬 프로세스 이름

- 로컬 IP 주소 및 포트 번호
- 원격 IP 주소 및 포트 번호
- 수신 및 전송된 트래픽의 양(KB)입니다.
- 연결이 활성화된 시간(밀리초)

## 절차

1. 다음 명령을 입력하여 로컬 포트 22 에 대한 연결 추적을 시작합니다.

```
/usr/share/bcc/tools/tcplife -L 22
PID COMM LADDR LPORT RADDR RPORT TX_KB RX_KB MS
19392 sshd 192.0.2.1 22 192.0.2.17 43892 53 52 6681.95
19431 sshd 192.0.2.1 22 192.0.2.245 43902 81 249381 7585.09
19487 sshd 192.0.2.1 22 192.0.2.121 43970 6998 7 16740.35
...
```

연결이 닫힐 때마다 **tcplife** 에 연결 세부 정보가 표시됩니다.

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

## 추가 리소스

- **tcplife(8)** 도움말 페이지
- `/usr/share/bcc/tools/doc/tcplife_example.txt`

## 56.8. TCP 재전송 추적

**tcpretrans** 유틸리티는 로컬 및 원격 IP 주소 및 포트 번호와 같은 **TCP** 재전송에 대한 세부 정보와 재전송 시 **TCP** 상태를 표시합니다.

유틸리티는 **eBPF** 기능을 사용하므로 오버헤드가 매우 낮습니다.

## 절차

1.

다음 명령을 사용하여 **TCP** 재전송 세부 정보 표시를 시작합니다.

```
/usr/share/bcc/tools/tcpretrans
TIME PID IP LADDR:LPORT T> RADDR:RPORT STATE
00:23:02 0 4 192.0.2.1:22 R> 198.51.100.0:26788 ESTABLISHED
00:23:02 0 4 192.0.2.1:22 R> 198.51.100.0:26788 ESTABLISHED
00:45:43 0 4 192.0.2.1:22 R> 198.51.100.0:17634 ESTABLISHED
...
```

커널이 **TCP retransmit** 함수를 호출할 때마다 **tcpretrans** 는 연결 세부 정보를 표시합니다.

2.

**Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

## 추가 리소스

- 

**tcpretrans(8)** 도움말 페이지

- 

**/usr/share/bcc/tools/doc/tcpretrans\_example.txt**

## 56.9. TCP 상태 변경 정보 표시

**TCP** 세션 중에 **TCP** 상태가 변경됩니다. **tcpstates** 유틸리티는 **eBPF** 함수를 사용하여 이러한 상태 변경 사항을 추적하고 각 상태의 기간을 포함한 세부 정보를 출력합니다. 예를 들어 **tcpstates** 를 사용하여 연결이 초기화 상태에서 너무 많은 시간을 소비하는지 확인합니다.

## 절차

1.

다음 명령을 사용하여 **TCP** 상태 변경 추적을 시작합니다.

```
/usr/share/bcc/tools/tcpstates
SKADDR C-PID C-COMM LADDR LPORT RADDR RPORT OLDSTATE ->
NEWSTATE MS
ffff9cd377b3af80 0 swapper/1 0.0.0.0 22 0.0.0.0 0 LISTEN -> SYN_RECV
0.000
ffff9cd377b3af80 0 swapper/1 192.0.2.1 22 192.0.2.45 53152 SYN_RECV ->
```



```

ESTABLISHED 0.067
ffff9cd377b3af80 818 sssd_nss 192.0.2.1 22 192.0.2.45 53152 ESTABLISHED ->
CLOSE_WAIT 65636.773
ffff9cd377b3af80 1432 sshd 192.0.2.1 22 192.0.2.45 53152 CLOSE_WAIT ->
LAST_ACK 24.409
ffff9cd377b3af80 1267 pulseaudio 192.0.2.1 22 192.0.2.45 53152 LAST_ACK ->
CLOSE 0.376
...

```

연결에서 상태를 변경할 때마다 **tcpstates** 는 업데이트된 연결 세부 정보와 함께 새 행을 표시합니다.

여러 연결이 동시에 상태를 변경하는 경우 첫 번째 열(**SKADDR**)의 소켓 주소를 사용하여 어떤 항목이 동일한 연결에 속하는지 확인합니다.

2.

**Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

추가 리소스

- **tcpstates(8)** 도움말 페이지
- `/usr/share/bcc/tools/doc/tcpstates_example.txt`

## 56.10. 특정 서브넷에 전송된 TCP 트래픽 요약 및 집계

**tcpsubnet** 유틸리티는 로컬 호스트가 서브넷에 전송하는 **IPv4 TCP** 트래픽을 요약하고 집계하고 고정 간격으로 출력을 표시합니다. 유틸리티는 **eBPF** 기능을 사용하여 오버헤드를 줄이기 위해 데이터를 수집 및 요약합니다.

기본적으로 **tcpsubnet** 은 다음 서브넷의 트래픽을 요약합니다.

- **127.0.0.1/32**
- **10.0.0.0/8**
- **172.16.0.0/12**

- **192.0.2.0/24/16**
- **0.0.0.0/0**

마지막 서브넷(0.0.0.0/0)은 **catch-all** 옵션입니다. **tcpsubnet** 유틸리티는 이 **catch-all** 항목의 처음 4개와 다른 서브넷 트래픽 수를 계산합니다.

이 절차에 따라 **192.0.2.0/24** 및 **198.51.100.0/24** 서브넷에 대한 트래픽을 계산합니다. 다른 서브넷으로의 트래픽은 **0.0.0.0/0 catch-all** 서브넷 항목에서 추적됩니다.

### 절차

1. **192.0.2.0/24, 198.51.100.0/24** 및 기타 서브넷으로 전송되는 트래픽 양을 모니터링합니다.

```
/usr/share/bcc/tools/tcpsubnet 192.0.2.0/24,198.51.100.0/24,0.0.0.0/0
Tracing... Output every 1 secs. Hit Ctrl-C to end
[02/21/20 10:04:50]
192.0.2.0/24 856
198.51.100.0/24 7467
[02/21/20 10:04:51]
192.0.2.0/24 1200
198.51.100.0/24 8763
0.0.0.0/0 673
...
```

이 명령은 지정된 서브넷의 초당 한 번씩 트래픽을 바이트로 표시합니다.

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

### 추가 리소스

- **tcpsubnet(8)** 도움말 페이지
- **/usr/share/bcc/tools/doc/tcpsubnet.txt**

## 56.11. IP 주소 및 포트를 통한 네트워크 처리량 표시

**tcptop** 유틸리티는 호스트에서 전송하는 **TCP** 트래픽을 표시하고 킬로바이트로 수신합니다. 보고서는 활성화된 **TCP** 연결만 자동으로 새로 고치며 포함합니다. 유틸리티는 **eBPF** 기능을 사용하므로 오버헤드가 매우 낮습니다.

## 절차

1. 전송 및 수신 트래픽을 모니터링하려면 다음을 입력합니다.

```
/usr/share/bcc/tools/tcptop
13:46:29 loadavg: 0.10 0.03 0.01 1/215 3875

PID COMM LADDR RADDR RX_KB TX_KB
3853 3853 192.0.2.1:22 192.0.2.165:41838 32 102626
1285 sshd 192.0.2.1:22 192.0.2.45:39240 0 0
...
```

명령의 출력에는 활성 **TCP** 연결만 포함됩니다. 로컬 또는 원격 시스템이 연결을 종료하면 더 이상 연결이 출력에 표시되지 않습니다.

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

## 추가 리소스

- **tcptop(8)** 도움말 페이지
- `/usr/share/bcc/tools/doc/tcptop.txt`

## 56.12. 설정된 TCP 연결 추적

**tcptracer** 유틸리티는 **TCP** 연결을 연결, 수락 및 종료하는 커널 기능을 추적합니다. 유틸리티는 **eBPF** 기능을 사용하므로 오버헤드가 매우 낮습니다.

## 절차

1. 다음 명령을 사용하여 추적 프로세스를 시작합니다.

```
/usr/share/bcc/tools/tcptracer
Tracing TCP established connections. Ctrl-C to end.
T PID COMM IP SADDR DADDR SPORT DPORT
```

```
A 1088 ns-slapd 4 192.0.2.153 192.0.2.1 0 65535
A 845 sshd 4 192.0.2.1 192.0.2.67 22 42302
X 4502 sshd 4 192.0.2.1 192.0.2.67 22 42302
...
```

커널이 연결을 연결, 수락 또는 닫을 때마다 **tcptracer** 는 연결 세부 정보를 표시합니다.

2.

**Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

추가 리소스

- [tcptracer\(8\) 도움말 페이지](#)
- [/usr/share/bcc/tools/doc/tcptracer\\_example.txt file](#)

### 56.13. IPV4 및 IPV6 추적 시도를 수신 대기

**solisten** 유틸리티는 모든 **IPv4** 및 **IPv6 listen** 시도를 추적합니다. 결과적으로 오류가 발생하거나 연결을 수락하지 않는 수신 프로그램을 포함하여 수신 시도를 추적합니다. 유틸리티는 프로그램이 **TCP** 연결을 수신 대기하려고 할 때 커널이 호출하는 함수를 추적합니다.

절차

1.

다음 명령을 입력하여 모든 수신 **TCP** 시도를 표시하는 추적 프로세스를 시작합니다.

```
/usr/share/bcc/tools/solisten
PID COMM PROTO BACKLOG PORT ADDR
3643 nc TCPv4 1 4242 0.0.0.0
3659 nc TCPv6 1 4242 2001:db8:1::1
4221 redis-server TCPv6 128 6379 ::
4221 redis-server TCPv4 128 6379 0.0.0.0
....
```

2.

**Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

추가 리소스

- [solisten 매뉴얼 페이지](#)

- **`/usr/share/bcc/tools/doc/solisten_example.txt` file.**

#### 56.14. 소프트 인터럽트의 서비스 시간 요약

**softirqs** 유틸리티는 소프트 인터럽트(**soft IRQ**)를 서비스하는 데 소요되는 시간을 요약하고, 이번에는 총 또는 히스토그램 배포로 표시됩니다. 이 유틸리티는 안정적인 추적 메커니즘인 **irq:softirq\_enter** 및 **irq:softirq\_exit** 커널 추적 지점을 사용합니다.

##### 절차

1. 다음 명령을 입력하여 추적 소프트 **irq** 이벤트 시간을 시작합니다.

```
/usr/share/bcc/tools/softirqs
Tracing soft irq event time... Hit Ctrl-C to end.
^C
SOFTIRQ TOTAL_usecs
tasklet 166
block 9152
net_rx 12829
rcu 53140
sched 182360
timer 306256
```

2. **Ctrl +C** 를 눌러 추적 프로세스를 중지합니다.

##### 추가 리소스

- **`softirqs` 도움말 페이지**
- **`/usr/share/bcc/tools/doc/softirqs_example.txt`**
- **`MPstat(1)` 도움말 페이지**

#### 56.15. 추가 리소스

- **`/usr/share/doc/bcc/README.md` file**

## 57장. TIPC 시작하기

**VPC(Transparent Inter-process Communication)**는 클러스터 전체 운영을 위한 **IPC(Inter-process Communication)** 서비스입니다.

고가용성 동적 클러스터 환경에서 실행 중인 애플리케이션에는 특별한 요구 사항이 있습니다. 클러스터의 노드 수는 다를 수 있으며, 라우터가 실패할 수 있으며, 부하 분산 고려 사항으로 인해 기능을 클러스터의 다른 노드로 이동할 수 있습니다. **TIPC**는 애플리케이션 개발자가 이러한 상황을 처리하기 위한 노력을 최소화하고 이를 정확하고 최적으로 처리할 수 있는 가능성을 극대화합니다. 또한 **TIPC**는 **TCP**와 같은 일반 프로토콜보다 더 효율적이고 내결함성 통신을 제공합니다.

### 57.1. TIPC의 아키텍처

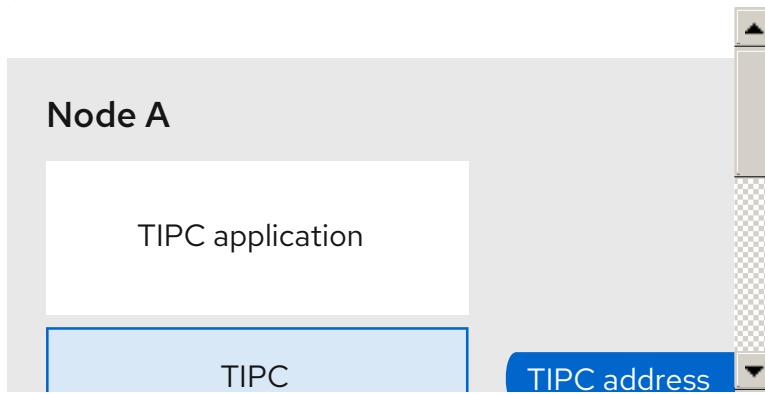
**TIPC**는 **TIPC**와 패킷 전송 서비스(베어러)를 사용하는 애플리케이션 간 계층이며 전송, 네트워크 및 신호 연결 계층의 수준을 확장합니다. 그러나 **TIPC**는 다른 전송 프로토콜을 전달자로 사용할 수 있으므로, 예를 들어 **TCP** 연결이 **TIPC** 신호 링크의 전달자 역할을 할 수 있습니다.

**TIPC**는 다음 전달자를 지원합니다.

- 이더넷
- InfiniBand
- UDP 프로토콜

**TIPC**는 모든 **TIPC** 통신의 엔드포인트인 **TIPC** 포트 간에 안정적으로 메시지를 전송합니다.

다음은 **TIPC** 아키텍처 다이어그램입니다.



## 57.2. 시스템이 부팅될 때 TIPSC 모듈 로드

**TIPC** 프로토콜을 사용하려면 먼저 **tips c** 커널 모듈을 로드합니다. 이 섹션에서는 **RHEL**이 시스템을 부팅할 때 이 모듈을 자동으로 로드하도록 구성하는 방법에 대해 설명합니다.

### 절차

1. 다음 콘텐츠를 사용하여 `/etc/modules-load.d/tipc.conf` 파일을 만듭니다.

```
tipc
```

2. **systemd-modules-load** 서비스를 다시 시작하여 시스템을 재부팅하지 않고 모듈을 로드합니다.

```
systemctl start systemd-modules-load
```

### 검증 단계

1. 다음 명령을 사용하여 **RHEL**에서 **tips c** 모듈을 로드 했는지 확인합니다.

```
lsmod | grep tipc
tipc 311296 0
```

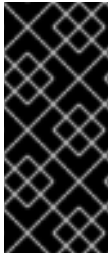
명령에서 **tips c** 모듈에 대한 항목이 표시되지 않으면 **RHEL**에서 로드하지 못했습니다.

### 추가 리소스

- [modules-load.d\(5\) man page](#)

### 57.3. TIPC 네트워크 생성

이 섹션에서는 **TIPC** 네트워크를 만드는 방법에 대해 설명합니다.



#### 중요

명령은 **TIPC** 네트워크를 임시로만 구성합니다. 노드에서 **TIPC**를 영구적으로 구성하려면 스크립트에서 이 절차의 명령을 사용하고 시스템이 부팅될 때 해당 스크립트를 실행하도록 **RHEL**을 구성합니다.

#### 사전 요구 사항



**tipsc** 모듈이 로드되었습니다. 자세한 내용은 [시스템 부팅 시 tipc 모듈 로드](#)를 참조하십시오.

#### 절차

1.

선택 사항: **UUID** 또는 노드의 호스트 이름과 같은 고유한 노드 **ID**를 설정합니다.

```
tipc node set identity host_name
```

**ID**는 최대 **16**개의 문자와 숫자로 구성된 모든 고유한 문자열일 수 있습니다.

이 단계 이후에는 **ID**를 설정하거나 변경할 수 없습니다.

2.

전달자를 추가합니다. 예를 들어 이더넷을 미디어로 사용하고 **enp0s1** 장치를 물리적 베어러 장치로 사용하려면 다음을 입력합니다.

```
tipc bearer enable media eth device enp1s0
```

3.

선택 사항: 중복성과 성능을 높이기 위해 이전 단계의 명령을 사용하여 추가 전달자를 연결합니다. 동일한 미디어에서 최대 **3**개의 전달자를 구성할 수 있지만 **2**개를 초과할 수는 없습니다.

4.

**TIPC** 네트워크에 참여해야 하는 각 노드에서 이전 단계를 모두 반복합니다.

#### 검증 단계



1.

클러스터 구성원의 링크 상태를 표시합니다.

```
tipc link list
broadcast-link: up
5254006b74be:enp1s0-525400df55d1:enp1s0: up
```

이 출력은 노드 **5254006b74be**의 베어러 **enp1s0**과 노드 **525400df55d1**의 베어러 **enp1s0** 간의 연결이 되었음을 나타냅니다.

2.

TIPC 게시 테이블을 표시합니다.

```
tipc nametable show
Type Lower Upper Scope Port Node
0 1795222054 1795222054 cluster 0 5254006b74be
0 3741353223 3741353223 cluster 0 525400df55d1
1 1 1 node 2399405586 5254006b74be
2 3741353223 3741353223 node 0 5254006b74be
```

- 서비스 유형 0이 있는 두 항목은 두 노드가 이 클러스터의 멤버임을 나타냅니다.
- 서비스 유형 1이 포함된 항목은 기본 제공 토폴로지 서비스 추적 서비스를 나타냅니다.
- 서비스 유형 2가 포함된 항목은 발행 노드에서 표시된 대로 링크를 표시합니다. 범위 제한 **3741353223**은 10진수 형식으로 피어 끝점의 주소(노드 ID를 기반으로 하는 고유한 32비트 해시 값)를 나타냅니다.

#### 추가 리소스

- [tipc-bearer\(8\) 매뉴얼 페이지](#)
- [tipc-namespace\(8\) 도움말 페이지](#)

#### 57.4. 추가 리소스

- 다른 전달자 수준 프로토콜을 사용하여 전송 미디어를 기반으로 노드 간 통신을 암호화하는 것이 좋습니다. 예를 들면 다음과 같습니다.

- **MACsec:** [MACsec을 사용하여 계층 2 트래픽 암호화](#) 참조
- **IPsec:** [IPsec을 사용하여 VPN 구성](#) 참조
- **TIPC**를 사용하는 방법에 대한 예를 들어 `git clone git://git.code.sf.net/p/tipcutils` 명령을 사용하여 업스트림 **GIT** 리포지토리를 복제합니다. 이 리포지토리에는 **TIPC** 기능을 사용하는 데모 및 테스트 프로그램의 소스 코드가 포함되어 있습니다. 이 리포지토리는 **Red Hat**에서 제공되지 않습니다.
- **kernel-doc** 패키지에서 제공하는 `/usr/share/doc/kernel-doc- <kernel_version>/Documentation/output/networking/tipc.html`

## 58장. NM-CLOUD-SETUP을 사용하여 공용 클라우드에서 네트워크 인터페이스 자동 구성

일반적으로 VM(가상 머신)에는 DHCP에서 구성할 수 있는 하나의 인터페이스만 있습니다. 그러나 일부 VM에는 DHCP에서 구성할 수 없는 하나의 인터페이스에 여러 네트워크 인터페이스, IP 주소 및 IP 서브넷이 있을 수 있습니다. 또한 관리자는 시스템이 실행되는 동안 네트워크를 재구성할 수 있습니다. **nm-cloud-setup** 유틸리티는 클라우드 서비스 공급자의 메타데이터 서버에서 구성 정보를 자동으로 검색하고 퍼블릭 클라우드에서 VM의 네트워크 구성을 업데이트합니다.

### 58.1. NM-CLOUD-SETUP 구성 및 사전 배포

공용 클라우드에서 네트워크 인터페이스를 활성화하고 구성하려면 **nm-cloud-setup** 을 타이머 및 서비스로 실행합니다. 다음 절차에서는 **Amazon EC2**에 **nm-cloud-setup** 을 사용하는 방법을 설명합니다.



#### 참고

**Red Hat Enterprise Linux** 온 디맨드 및 **AWS** 골든 이미지에서는 **nm-cloud-setup** 이 이미 활성화되어 있지 않으며 아무 작업도 필요하지 않습니다.

#### 사전 요구 사항

- 네트워크 연결이 있습니다.
- 연결에서는 **DHCP**를 사용합니다.

기본적으로 **NetworkManager**는 **DHCP**를 사용하는 연결 프로필을 생성합니다. **/etc/NetworkManager/NetworkManager.conf** 에서 **no-auto-default** 매개변수를 설정하여 프로필이 생성되지 않은 경우 이 초기 연결을 수동으로 생성합니다.

#### 절차

1. **nm-cloud-setup** 패키지를 설치합니다.
 

```
yum install NetworkManager-cloud-setup
```
2. **nm-cloud-setup** 서비스에 대한 **snap-in** 파일을 생성하고 실행합니다.
  - a. 다음 명령을 사용하여 **snap-in** 파일 편집을 시작합니다.

```
systemctl edit nm-cloud-setup.service
```

구성 설정을 적용하려면 서비스를 명시적으로 시작하거나 시스템을 재부팅하는 것이 중요합니다.

b.

**systemd snap-in** 파일을 사용하여 **nm-cloud-setup** 에서 클라우드 공급자를 구성합니다. 예를 들어 **Amazon EC2**를 사용하려면 다음을 입력합니다.

```
[Service]
Environment=NM_CLOUD_SETUP_EC2=yes
```

다음 환경 변수를 설정하여 사용할 클라우드를 사용하도록 설정할 수 있습니다.

- **Microsoft Azure**의 경우 **NM\_CLOUD\_SETUP\_AZURE**
- **AWS(Amazon EC2)**용 **NM\_CLOUD\_SETUP\_EC2**
- **GCP(Google Cloud Platform)**용 **NM\_CLOUD\_SETUP\_GCP**
- **Alibaba Cloud (Alibaba Cloud)**의 경우 **NM\_CLOUD\_SETUP\_ALIYUN (Alibaba Cloud)**

c.

파일을 저장하고 편집기를 종료합니다.

3.

**systemd** 구성을 다시 로드합니다.

```
systemctl daemon-reload
```

4.

**nm-cloud-setup** 서비스를 활성화하고 시작합니다.

```
systemctl enable --now nm-cloud-setup.service
```

5.

**nm-cloud-setup** 타이머를 활성화하고 시작합니다.

**# `systemctl enable --now nm-cloud-setup.timer`**

#### 추가 리소스

- ***nm-cloud-setup(8) 매뉴얼 페이지***
- ***[이더넷 연결 구성](#)***