Red Hat Enterprise Linux 8

네트워킹 구성 및 관리

네트워크 인터페이스, 방화벽 및 고급 네트워킹 기능 관리

Last Updated: 2024-06-26

Red Hat Enterprise Linux 8 네트워킹 구성 및 관리

네트워크 인터페이스, 방화벽 및 고급 네트워킹 기능 관리

법적 공지

초록

RHEL(Red Hat Enterprise Linux)의 네트워킹 기능을 사용하여 조직의 네트워크 및 보안 요구 사항을 충족하도록 호스트를 구성할 수 있습니다. 예를 들면 다음과 같습니다. 본딩, VLAN, 브리지, 터널 및 기타 네트워크 유형을 구성하여 호스트를 네트워크에 연결할 수 있습니다. 로컬 호스트 및 전체 네트워크에 대해 성능이 중요한 방화벽을 구축할 수 있습니다. RHEL에는 firewalld 서비스, nftables 프레임워크 및 XDP(Express Data Path)와 같은 패킷 필터링 소프트웨어가 포함되어 있습니다. RHEL은 정책 기반 라우팅 및 MPTCP(Multipath TCP)와 같은 고급 네트워킹 기능도 지원합니다.

차례

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

Jira를 통해 피드백 제출 (등록 필요)

1. Jira 웹 사이트에 로그인합니다.
2. 상단 탐색 모음에서 **생성** 을 클릭합니다.
3. **요약** 필드에 설명 제목을 입력합니다.
4. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

1장. 일관된 네트워크 인터페이스 이름 구현

udev 장치 관리자는 Red Hat Enterprise Linux에서 일관된 장치 이름을 구현합니다. 장치 관리자는 다양한 이름 지정 체계를 지원하며 기본적으로 펌웨어, 토폴로지 및 위치 정보를 기반으로 고정 이름을 할당합니다.

장치 이름을 일관되게 지정하지 않으면 Linux 커널은 고정 접두사와 인덱스를 결합하여 네트워크 인터페이스에 이름을 할당합니다. 커널이 네트워크 장치를 초기화할 때 인덱스가 증가합니다. 예를 들어 **eth0** 은 시작 시 프로브되는 첫 번째 이더넷 장치를 나타냅니다. 다른 네트워크 인터페이스 컨트롤러를 시스템에 추가하는 경우 재부팅 후 장치가 다른 순서로 초기화될 수 있으므로 커널 장치 이름 할당은 더 이상 수정되지 않습니다. 이 경우 커널은 장치의 이름을 다르게 지정할 수 있습니다.

이 문제를 해결하기 위해 **udev** 는 일관된 장치 이름을 할당합니다. 여기에는 다음과 같은 이점이 있습니다.

- 장치 이름은 재부팅 시 안정적입니다.
- 하드웨어를 추가하거나 제거하는 경우에도 장치 이름은 고정됩니다.
- 결함 있는 하드웨어를 원활하게 교체할 수 있습니다.
- 네트워크 이름 지정은 상태 비저장이며 명시적 구성 파일이 필요하지 않습니다.

주의

일반적으로 Red Hat은 일관된 장치 이름이 비활성화된 시스템을 지원하지 않습니다. 예외에 대한 자세한 내용은 net.ifnames=0 솔루션을 설정하는 것이 안전합니다 .

1.1. UDEV 장치 관리자의 네트워크 인터페이스 이름 변경 방법

네트워크 인터페이스에 대한 일관된 이름 지정 체계를 구현하기 위해 **udev** 장치 관리자는 다음 규칙 파일을 나열 순서대로 처리합니다.

1. 선택 사항: **/usr/lib/udev/rules.d/60-net.rules**
   /usr/lib/udev/rules.d/60-net.rules 파일은 더 이상 사용되지 않는 **/usr/lib/udev/rename_device** 도우미 유틸리티에서 **/etc/sysconfig/network-scripts/ifcfg-*** 파일에서 **HWADDR** 매개 변수를 검색하도록 정의합니다. 변수에 설정된 값이 인터페이스의 MAC 주소와 일치하는 경우 도우미 유틸리티는 인터페이스의 이름을 **ifcfg** 파일의 **DEVICE** 매개변수에 설정된 이름으로 바꿉니다.

   시스템이 NetworkManager 연결 프로필만 키 파일 형식으로 사용하는 경우 **udev** 는 이 단계를 건너뜁니다.

2. Dell 시스템에서만: **/usr/lib/udev/rules.d/71-biosdevname.rules**
   이 파일은 **biosdevname** 패키지가 설치된 경우에만 존재하며 규칙 파일은 **biosdevname** 유틸리티에서 이전 단계에서 이름이 변경되지 않은 경우 이름 지정 정책에 따라 인터페이스 이름을 변경하도록 정의합니다.

참고

Dell 시스템에만 **biosdevname** 을 설치하고 사용합니다.

3. **/usr/lib/udev/rules.d/75-net-description.rules**
   이 파일은 **udev** 가 네트워크 인터페이스를 검사하는 방법을 정의하고 **udev**-internal 변수에서 속성을 설정합니다. 그런 다음 이러한 변수는 **/usr/lib/udev/rules.d/80-net-setup-link.rules** 파일을 통해 다음 단계에서 처리됩니다. 일부 속성은 정의되지 않을 수 있습니다.

4. **/usr/lib/udev/rules.d/80-net-setup-link.rules**
   이 파일은 **udev** 서비스의 빌드된 **net_setup_link** 를 호출하고 **udev** 는 **/usr/lib/systemd/network/99-default.link** 파일의 **NamePolicy** 매개변수의 정책 순서에 따라 인터페이스의 이름을 바꿉니다. 자세한 내용은 네트워크 인터페이스 이름 지정 정책을 참조하십시오.

   정책이 적용되지 않으면 **udev** 는 인터페이스의 이름을 바꾸지 않습니다.

추가 리소스

- 주요 RHEL 버전 솔루션 간에 systemd 네트워크 인터페이스 이름이 다른 이유는 무엇입니까?

1.2. 네트워크 인터페이스 이름 지정 정책

기본적으로 **udev** 장치 관리자는 **/usr/lib/systemd/network/99-default.link** 파일을 사용하여 인터페이스 이름을 변경할 때 적용할 장치 이름 지정 정책을 결정합니다. 이 파일의 **NamePolicy** 매개변수는 **udev** 정책에서 사용하는 정책과 순서를 정의합니다.

```
NamePolicy=kernel database onboard slot path
```

다음 표에서는 **NamePolicy** 매개변수에서 지정한 대로 가장 먼저 일치하는 정책을 기반으로 **udev** 의 다양한 작업을 설명합니다.

정책	설명	이름 예
kernel	커널에서 장치 이름이 예측 가능함을 나타내는 경우 **udev** 는 이 장치의 이름을 바꾸지 않습니다.	**lo**
데이터베이스	이 정책은 **udev** 하드웨어 데이터베이스의 매핑에 따라 이름을 할당합니다. 자세한 내용은 **hwdb(7)** 도움말 페이지를 참조하십시오.	**idrac**
온보드	장치 이름에는 온보드 장치에 대한 펌웨어 또는 BIOS 제공 인덱스 번호가 포함됩니다.	**eno1**
slot	장치 이름은 펌웨어 또는 BIOS가 제공하는 PCI Express(PCIe) 핫플러그 슬롯 인덱스 번호를 통합합니다.	**ens1**
path	장치 이름은 하드웨어의 커넥터의 실제 위치를 포함합니다.	**enp1s0**
mac	장치 이름에는 MAC 주소가 포함됩니다. 기본적으로 Red Hat Enterprise Linux는 이 정책을 사용하지 않지만 관리자는 이를 활성화할 수 있습니다.	**enx525400d5e0fb**

추가 리소스

- udev 장치 관리자의 네트워크 인터페이스 이름 변경 방법

- **systemd.link(5)** man page

1.3. 네트워크 인터페이스 이름 지정 체계

udev 장치 관리자는 장치 드라이버가 일관된 장치 이름을 생성하기 위해 제공하는 특정 안정적인 인터페이스 속성을 사용합니다.

새로운 **udev** 버전이 특정 인터페이스에 대한 이름을 생성하는 방법을 변경하는 경우 Red Hat은 새 스키마 버전을 추가하고 **systemd.net-naming-scheme(7)** 도움말 페이지의 세부 정보를 문서화합니다. 기본적으로 RHEL(Red Hat Enterprise Linux) 8은 RHEL의 최신 마이너 버전을 설치하거나 업데이트하더라도 **rhel-8.0** 이름 지정 스키마를 사용합니다.

기본값 이외의 스키마를 사용하려면 네트워크 인터페이스 이름 지정 스키마를 전환할 수 있습니다.

다양한 장치 유형 및 플랫폼의 이름 지정 체계에 대한 자세한 내용은 **systemd.net-naming-scheme(7)** 매뉴얼 페이지를 참조하십시오.

1.4. 다른 네트워크 인터페이스 이름 지정 체계로 전환

기본적으로 RHEL(Red Hat Enterprise Linux) 8은 RHEL의 최신 마이너 버전을 설치하거나 업데이트하더라도 **rhel-8.0** 이름 지정 스키마를 사용합니다. 기본 이름 지정 스키마는 대부분의 시나리오에 적합하지만 다른 스키마 버전으로 전환해야 하는 이유가 있을 수 있습니다. 예를 들면 다음과 같습니다.

- 새로운 체계는 슬롯 번호와 같은 추가 속성을 인터페이스 이름에 추가하는 경우 장치를 더 잘 식별하는 데 도움이 될 수 있습니다.
- 새로운 체계는 **udev** 가 커널 할당 장치 이름(**eth***)으로 대체되지 않도록 할 수 있습니다. 이는 드라이버에서 두 개 이상의 인터페이스에 고유한 특성을 제공하지 않는 경우 고유 이름을 생성하는 경우에 발생합니다.

사전 요구 사항

- 서버의 콘솔에 액세스할 수 있습니다.

절차

1. 네트워크 인터페이스를 나열합니다.

```
# ip link show
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

   인터페이스의 MAC 주소를 기록합니다.

2. 선택 사항: 네트워크 인터페이스의 **ID_NET_NAMING_SCHEME** 속성을 표시하여 RHEL에서 현재 사용하는 이름 지정 스키마를 식별합니다.

```
# udevadm info --query=property --property=ID_NET_NAMING_SCHEME
/sys/class/net/eno1'
ID_NET_NAMING_SCHEME=rhel-8.0
```

   이 속성은 **lo** 루프백 장치에서 사용할 수 없습니다.

3. **net.naming-scheme= <*scheme*>** 옵션을 설치된 모든 커널의 명령줄에 추가합니다. 예를 들면 다음과 같습니다.

```
# grubby --update-kernel=ALL --args=net.naming-scheme=rhel-8.4
```

4. 시스템을 재부팅합니다.

```
# reboot
```

5. 기록한 MAC 주소에 따라 다른 이름 지정 체계로 인해 변경된 네트워크 인터페이스의 새 이름을 확인합니다.

```
# ip link show
2: eno1np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

스키마를 전환한 후 이 예제의 **udev** 이름은 MAC 주소가 **00:00:5e:00:53:1a eno1np0** 인 장치이며 이전에 **eno1** 이라는 이름이 지정되었습니다.

6. 이전 이름의 인터페이스를 사용하는 NetworkManager 연결 프로필을 식별합니다.

```
# nmcli -f device,name connection show
DEVICE  NAME
eno1  example_profile
...
```

7. 연결 프로필의 **connection.interface-name** 속성을 새 인터페이스 이름으로 설정합니다.

```
# nmcli connection modify example_profile connection.interface-name "eno1np0"
```

8. 연결 프로필을 다시 활성화합니다.

```
# nmcli connection up example_profile
```

검증

- 네트워크 인터페이스의 **ID_NET_NAMING_SCHEME** 속성을 표시하여 RHEL에서 사용하는 이름 지정 체계를 식별합니다.

```
# udevadm info --query=property --property=ID_NET_NAMING_SCHEME
/sys/class/net/eno1np0'
ID_NET_NAMING_SCHEME=_rhel-8.4
```

추가 리소스

- 네트워크 인터페이스 이름 지정 체계

1.5. IBM Z 플랫폼에서 예측 가능한 ROCE 장치 이름 확인

RHEL(Red Hat Enterprise Linux) 8.7 이상에서 **udev** 장치 관리자는 다음과 같이 IBM Z에서 RoCE 인터페이스의 이름을 설정합니다.

- 호스트가 장치에 고유한 ID(UID)를 적용하는 경우 **udev** 는 UID를 기반으로 하는 일관된 장치 이름을 할당합니다(예: **eno <*UID_in_decimal*>**).
- 호스트가 장치에 UID를 적용하지 않으면 해당 동작은 설정에 따라 다릅니다.
  - 기본적으로 **udev** 는 장치에 예측할 수 없는 이름을 사용합니다.
  - **net.naming-scheme=rhel-8.7** 커널 명령줄 옵션을 설정하면 **udev** 는 장치의 함수 식별자(FID)를 기반으로 하는 일관된 장치 이름을 할당합니다(예: **ens <*FID_in_decimal*>**).

다음과 같은 경우 IBM Z에서 RoCE 인터페이스에 예측 가능한 장치 이름을 수동으로 구성합니다.

- 호스트는 RHEL 8.6 이상을 실행하고 장치에 대해 UID를 적용하고 RHEL 8.7 이상으로 업데이트할 계획입니다.
  RHEL 8.7 이상으로 업데이트한 후 **udev** 는 일관된 인터페이스 이름을 사용합니다. 그러나 업데이트 전에 예측할 수 없는 장치 이름을 사용한 경우 NetworkManager 연결 프로필은 이러한 이름을 계속 사용하고 영향을 받는 프로필을 업데이트할 때까지 활성화하지 못합니다.
- 호스트는 RHEL 8.7 이상을 실행하고 UID를 적용하지 않으며 RHEL 9로 업그레이드할 계획입니다.

udev 규칙 또는 **systemd** 링크 파일을 사용하여 인터페이스 이름을 수동으로 변경하려면 먼저 예측 가능한 장치 이름을 결정해야 합니다.

사전 요구 사항

- RoCE 컨트롤러가 시스템에 설치되어 있습니다.
- **sysfsutils** 패키지가 설치되어 있습니다.

절차

1. 사용 가능한 네트워크 장치를 표시하고 RoCE 장치의 이름을 확인합니다.

```
# ip link show
...
2: enP5165p0s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
UP mode DEFAULT group default qlen 1000
...
```

2. **/sys/** 파일 시스템의 장치 경로를 표시합니다.

```
# systool -c net -p
Class = "net"

  Class Device = "enP5165p0s0"
  Class Device path = "/sys/devices/pci142d:00/142d:00:00.0/net/enP5165p0s0"
    Device = "142d:00:00.0"
    Device path = "/sys/devices/pci142d:00/142d:00:00.0"
```

   다음 단계의 **장치 경로** 필드에 표시된 경로를 사용합니다.

3. **< *device_path*> */uid*_id_unique** 파일의 값을 표시합니다. 예를 들면 다음과 같습니다.

```
# cat /sys/devices/pci142d:00/142d:00:00.0/uid_id_unique
```

표시된 값은 UID 고유성이 적용되었는지 여부를 나타내며 이후 단계에서 이 값이 필요합니다.

4. 고유 식별자를 확인합니다.

   - UID 고유성이 적용되는 경우 (**1**) < ***device_path***> **/uid** 파일에 저장된 UID를 표시합니다. 예를 들면 다음과 같습니다.

     ```
     # cat /sys/devices/pci142d:00/142d:00:00.0/uid
     ```

   - UID 고유성이 강제 적용되지 않은 경우(**0**) < ***device_path*> /*function*_id** 파일에 저장된 FID를 표시합니다. 예를 들면 다음과 같습니다.

     ```
     # cat /sys/devices/pci142d:00/142d:00:00.0/function_id
     ```

   명령의 출력은 16진수의 UID 및 FID 값을 표시합니다.

5. 16진수 식별자를 10진수로 변환합니다. 예를 들면 다음과 같습니다.

   ```
   # printf "%d\n" 0x00001402
   5122
   ```

6. 예측 가능한 장치 이름을 확인하려면 UID 고유성이 적용되었는지 여부에 따라 10진수 형식으로 식별자를 추가합니다.

   - UID 고유성이 적용되는 경우 **eno** 접두사(예: **eno5122**)에 식별자를 추가합니다.
   - UID 고유성이 적용되지 않으면 **ens** 접두사에 식별자를 추가합니다(예: **ens5122**).

다음 단계

- 다음 방법 중 하나를 사용하여 인터페이스 이름을 예측 가능한 이름으로 변경합니다.
  - udev 규칙을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성
  - systemd 링크 파일을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성

추가 리소스

- IBM 설명서: 네트워크 인터페이스 이름
- **systemd.net-naming-scheme(7)** 도움말 페이지

1.6. 설치 중에 이더넷 인터페이스의 접두사 사용자 정의

이더넷 인터페이스에 기본 device-naming 정책을 사용하지 않으려면 RHEL(Red Hat Enterprise Linux) 설치 중에 사용자 지정 장치 접두사를 설정할 수 있습니다.

중요

Red Hat은 RHEL 설치 중에 접두사를 설정하는 경우에만 사용자 지정된 이더넷 접두사가 있는 시스템을 지원합니다. 이미 배포된 시스템에서 **prefixdevname** 유틸리티를 사용하는 것은 지원되지 않습니다.

설치 중에 장치 접두사를 설정하면 **udev** 서비스는 설치 후 이더넷 인터페이스에 < ***prefix><index***> 형식을 사용합니다. 예를 들어 접두사 **net** 을 설정하면 서비스에서 **net0** 이름,**net1** 등을 이더넷 인터페이스에 할당합니다.

udev 서비스는 사용자 지정 접두사에 인덱스를 추가하고 알려진 이더넷 인터페이스의 인덱스 값을 유지합니다. 인터페이스를 추가하면 **udev** 에서 이전에 할당한 인덱스 값보다 큰 인덱스 값을 새 인터페이스에 할당합니다.

사전 요구 사항

- 접두사는 ASCII 문자로 구성됩니다.
- 접두사는 영숫자 문자열입니다.
- 접두사는 16자보다 짧습니다.
- 접두사는 **eth**,**eno**,**ens** 및 **em** 과 같은 잘 알려진 다른 네트워크 인터페이스 접두사와 충돌하지 않습니다.

절차

1. Red Hat Enterprise Linux 설치 미디어를 부팅합니다.
2. 부팅 관리자에서 다음 단계를 수행합니다.
   a. **Install Red Hat Enterprise Linux < *version*>** 항목을 선택합니다.
   b. **Tab** 을 눌러 항목을 편집합니다.
   c. 커널 옵션에 **net.ifnames.prefix= &*lt;prefix*>**를 추가합니다.
   d. **Enter** 를 눌러 설치 프로그램을 시작합니다.
3. Red Hat Enterprise Linux 설치.

검증

- 인터페이스 이름을 확인하려면 네트워크 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 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

추가 리소스

- 표준 RHEL 8 설치 수행

1.7. UDEV 규칙을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성

udev 규칙을 사용하여 조직의 요구 사항을 반영하는 사용자 지정 네트워크 인터페이스 이름을 구현할 수 있습니다.

절차

1. 이름을 변경할 네트워크 인터페이스를 확인합니다.

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

   인터페이스의 MAC 주소를 기록합니다.

2. 인터페이스의 장치 유형 ID를 표시합니다.

```
# cat /sys/class/net/enp1s0/type
1
```

3. **/etc/udev/rules.d/70-persistent-net.rules** 파일을 생성하고 이름을 변경할 각 인터페이스에 대한 규칙을 추가합니다.

```
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="<MAC_address>",ATTR{type}=="<
device_type_id>",NAME="<new_interface_name>"
```

중요

부팅 프로세스 중에 일관된 장치 이름이 필요한 경우 **70-persistent-net.rules** 만 파일 이름으로 사용합니다. RAM 디스크 이미지를 다시 생성하는 경우 **dracut** 유틸리티는 이 이름의 파일을 **initrd** 이미지에 추가합니다.

   예를 들어 다음 규칙을 사용하여 MAC 주소 **00:00:5e:53:1a로 인터페이스의** 이름을 **provider0** 으로 변경합니다.

```
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:00:5e:00:53:1a",ATTR{type}=="
1",NAME="provider0"
```

4. 선택 사항: **initrd** RAM 디스크 이미지를 다시 생성합니다.

```
# dracut -f
```

   이 단계는 RAM 디스크에 네트워킹 기능이 필요한 경우에만 필요합니다. 예를 들어 루트 파일 시스템이 iSCSI와 같은 네트워크 장치에 저장된 경우입니다.

5. 이름을 바꿀 인터페이스를 사용하는 NetworkManager 연결 프로필을 식별합니다.

```
# nmcli -f device,name connection show
DEVICE  NAME
enp1s0  example_profile
...
```

6. 연결 프로필에서 **connection.interface-name** 속성을 설정 해제합니다.

```
# nmcli connection modify example_profile connection.interface-name ""
```

7. 임시로 새 인터페이스 이름과 이전 인터페이스 이름과 일치하도록 연결 프로필을 구성합니다.

```
# nmcli connection modify example_profile match.interface-name "provider0 enp1s0"
```

8. 시스템을 재부팅합니다.

```
# reboot
```

9. 링크 파일에 지정한 MAC 주소가 있는 장치의 이름이 **provider0** 으로 변경되었는지 확인합니다.

```
# ip link show
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

10. 새 인터페이스 이름만 일치하도록 연결 프로필을 구성합니다.

```
# nmcli connection modify example_profile match.interface-name "provider0"
```

이제 연결 프로필에서 이전 인터페이스 이름을 삭제했습니다.

11. 연결 프로필을 다시 활성화합니다.

```
# nmcli connection up example_profile
```

추가 리소스

- **udev(7)** man page

1.8. SYSTEMD 링크 파일을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성

systemd 링크 파일을 사용하여 조직의 요구 사항을 반영하는 사용자 지정 네트워크 인터페이스 이름을 구현할 수 있습니다.

사전 요구 사항

- 다음 조건 중 하나를 충족해야 합니다. NetworkManager는 이 인터페이스를 관리하지 않거나 해당 연결 프로파일에서는 키 파일 형식을 사용합니다.

절차

1. 이름을 변경할 네트워크 인터페이스를 확인합니다.

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

인터페이스의 MAC 주소를 기록합니다.

2. 아직 없는 경우 **/etc/systemd/network/** 디렉터리를 만듭니다.

```
# mkdir -p /etc/systemd/network/
```

3. 이름을 바꿀 각 인터페이스에 대해 다음 콘텐츠를 사용하여 **/etc/systemd/network/** 디렉터리에 **70-*.link** 파일을 생성합니다.

```
[Match]
MACAddress=<MAC_address>

[Link]
Name=<new_interface_name>
```

중요

70 접두사가 있는 파일 이름을 사용하여 **udev** 규칙 기반 솔루션과 일치하는 파일 이름을 유지합니다.

예를 들어 다음 내용으로 **/etc/systemd/network/70- provider0.link** 파일을 생성하여 인터페이스의 이름을 **00:00:5e:00:53:1a** 로 변경합니다.

```
[Match]
MACAddress=00:00:5e:00:53:1a

[Link]
Name=provider0
```

4. 선택 사항: **initrd** RAM 디스크 이미지를 다시 생성합니다.

```
# dracut -f
```

이 단계는 RAM 디스크에 네트워킹 기능이 필요한 경우에만 필요합니다. 예를 들어 루트 파일 시스템이 iSCSI와 같은 네트워크 장치에 저장된 경우입니다.

5. 이름을 바꿀 인터페이스를 사용하는 NetworkManager 연결 프로필을 식별합니다.

```
# nmcli -f device,name connection show
DEVICE  NAME
enp1s0  example_profile
...
```

6. 연결 프로필에서 **connection.interface-name** 속성을 설정 해제합니다.

```
# nmcli connection modify example_profile connection.interface-name ""
```

7. 임시로 새 인터페이스 이름과 이전 인터페이스 이름과 일치하도록 연결 프로필을 구성합니다.

```
# nmcli connection modify example_profile match.interface-name "provider0 enp1s0"
```

8. 시스템을 재부팅합니다.

```
# reboot
```

9. 링크 파일에 지정한 MAC 주소가 있는 장치의 이름이 **provider0** 으로 변경되었는지 확인합니다.

```
# ip link show
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

10. 새 인터페이스 이름만 일치하도록 연결 프로필을 구성합니다.

```
# nmcli connection modify example_profile match.interface-name "provider0"
```

이제 연결 프로필에서 이전 인터페이스 이름을 삭제했습니다.

11. 연결 프로필을 다시 활성화합니다.

```
# nmcli connection up example_profile
```

추가 리소스

- **systemd.link(5)** man page

1.9. SYSTEMD 링크 파일을 사용하여 네트워크 인터페이스에 대체 이름 할당

대체 인터페이스 이름을 지정하면 커널에서 네트워크 인터페이스에 추가 이름을 할당할 수 있습니다. 네트워크 인터페이스 이름이 필요한 명령에서 일반 인터페이스 이름과 동일한 방식으로 이러한 대체 이름을 사용할 수 있습니다.

사전 요구 사항

- 대체 이름으로 ASCII 문자를 사용해야 합니다.
- 대체 이름은 128자 이상이어야 합니다.

절차

1. 네트워크 인터페이스 이름과 해당 MAC 주소를 표시합니다.

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

대체 이름을 할당할 인터페이스의 MAC 주소를 기록합니다.

2. 아직 없는 경우 **/etc/systemd/network/** 디렉터리를 만듭니다.

```
# mkdir -p /etc/systemd/network/
```

3. 대체 이름이 필요한 각 인터페이스에 대해 다음 콘텐츠를 사용하여 **/etc/systemd/network/** 디렉터리에 ***.link** 파일을 생성합니다.

```
[Match]
MACAddress=<MAC_address>

[Link]
AlternativeName=<alternative_interface_name_1>
AlternativeName=<alternative_interface_name_2>
AlternativeName=<alternative_interface_name_n>
```

예를 들어 다음 콘텐츠를 사용하여 **/etc/systemd/network/70-altname.link** 파일을 생성하여 MAC 주소 **00:00:5e:53:1a** 가 있는 인터페이스에 **공급자** 를 다른 이름으로 할당합니다.

```
[Match]
MACAddress=00:00:5e:00:53:1a

[Link]
AlternativeName=provider
```

4. **initrd** RAM 디스크 이미지를 다시 생성합니다.

```
# dracut -f
```

5. 시스템을 재부팅합니다.

```
# reboot
```

검증

- 대체 인터페이스 이름을 사용합니다. 예를 들어 대체 이름 **공급자가** 있는 장치의 IP 주소 설정을 표시합니다.

```
# ip address show provider
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
    altname provider
    ...
```

추가 리소스

- 인터페이스 이름 지정 체계에서 AlternativeNamesPolicy는 무엇입니까?

2장. 이더넷 연결 구성

NetworkManager는 호스트에 설치된 각 이더넷 어댑터에 대한 연결 프로필을 생성합니다. 기본적으로 이 프로필은 IPv4 및 IPv6 연결에 DHCP를 사용합니다. 자동으로 생성된 이 프로필을 수정하거나 다음 경우 새 프로필을 추가합니다.

- 네트워크에는 고정 IP 주소 구성과 같은 사용자 지정 설정이 필요합니다.
- 서로 다른 네트워크 간에 호스트가 순환되므로 여러 프로필이 필요합니다.

Red Hat Enterprise Linux는 관리자에게 이더넷 연결을 구성하는 다양한 옵션을 제공합니다. 예를 들면 다음과 같습니다.

- **nmcli** 를 사용하여 명령줄에서 연결을 구성합니다.
- **nmtui** 를 사용하여 텍스트 기반 사용자 인터페이스에서 연결을 구성합니다.
- GNOME 설정 메뉴 또는 **nm-connection-editor** 애플리케이션을 사용하여 그래픽 인터페이스에서 연결을 구성합니다.
- **nmstatectl** 을 사용하여 Nmstate API를 통한 연결을 구성합니다.
- RHEL 시스템 역할을 사용하여 하나 이상의 호스트에서 연결 구성을 자동화합니다.

참고

Microsoft Azure 클라우드에서 실행되는 호스트에서 이더넷 연결을 수동으로 구성하려면 **cloud-init** 서비스를 비활성화하거나 클라우드 환경에서 검색된 네트워크 설정을 무시하도록 구성합니다. 그렇지 않으면 **cloud-init** 가 수동으로 구성된 다음 네트워크 설정을 재부팅할 때 재정의됩니다.

2.1. NMCLI를 사용하여 이더넷 연결 구성

이더넷을 통해 호스트를 네트워크에 연결하는 경우 **nmcli** 유틸리티를 사용하여 명령줄에서 연결의 설정을 관리할 수 있습니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.

절차

1. NetworkManager 연결 프로필을 나열합니다.

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

   기본적으로 NetworkManager는 호스트의 각 NIC에 대한 프로필을 생성합니다. 이 NIC를 특정 네트워크에만 연결하려는 경우 자동으로 생성된 프로필을 조정합니다. 이 NIC를 다른 설정으로 네트워크에 연결하려는 경우 각 네트워크에 대한 개별 프로필을 생성합니다.

2. 추가 연결 프로필을 생성하려면 다음을 입력합니다.

```
# nmcli connection add con-name <connection-name> ifname <device-name> type ethernet
```

기존 프로필을 수정하려면 이 단계를 건너뜁니다.

3. 선택 사항: 연결 프로필의 이름을 변경합니다.

```
# nmcli connection modify "Wired connection 1" connection.id "Internal-LAN"
```

프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

4. 연결 프로필의 현재 설정을 표시합니다.

```
# nmcli connection show Internal-LAN
...
connection.interface-name:    enp1s0
connection.autoconnect:       yes
ipv4.method:                  auto
ipv6.method:                  auto
...
```

5. IPv4 설정을 구성합니다.

- DHCP를 사용하려면 다음을 입력합니다.

```
# nmcli connection modify Internal-LAN ipv4.method auto
```

ipv4.method 가 이미 **auto** (기본값)로 설정된 경우 이 단계를 건너뜁니다.

- 정적 IPv4 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 다음을 입력합니다.

```
# nmcli connection modify Internal-LAN ipv4.method manual ipv4.addresses 192.0.2.1/24 ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com
```

6. IPv6 설정을 구성합니다.

- SLAAC(상태 비저장 주소 자동 구성)를 사용하려면 다음을 입력합니다.

```
# nmcli connection modify Internal-LAN ipv6.method auto
```

ipv6.method 가 이미 **auto** (기본값)로 설정된 경우 이 단계를 건너뜁니다.

- 정적 IPv6 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 다음을 입력합니다.

```
# nmcli connection modify Internal-LAN ipv6.method manual ipv6.addresses 2001:db8:1::fffe/64 ipv6.gateway 2001:db8:1::fffe ipv6.dns 2001:db8:1::ffbb ipv6.dns-search example.com
```

7. 프로필의 다른 설정을 사용자 지정하려면 다음 명령을 사용합니다.

```
# nmcli connection modify <connection-name> <setting> <value>
```

값을 따옴표로 묶거나 spaces로 묶습니다.

8. 프로필을 활성화합니다.

```
# nmcli connection up Internal-LAN
```

검증

1. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 **이름 서버** 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

문제 해결

- 네트워크 커넥터가 호스트와 스위치에 연결되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 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 NetworkManager를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 NetworkManager 서비스 솔루션을 다시 시작한 후 NetworkManager 연결 중복을 참조하십시오.

추가 리소스

- **nm-settings(5) 도움말** 페이지

2.2. NMCLI 대화형 편집기를 사용하여 이더넷 연결 구성

이더넷을 통해 호스트를 네트워크에 연결하는 경우 **nmcli** 유틸리티를 사용하여 명령줄에서 연결의 설정을 관리할 수 있습니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.

절차

1. NetworkManager 연결 프로필을 나열합니다.

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

   기본적으로 NetworkManager는 호스트의 각 NIC에 대한 프로필을 생성합니다. 이 NIC를 특정 네트워크에만 연결하려는 경우 자동으로 생성된 프로필을 조정합니다. 이 NIC를 다른 설정으로 네트워크에 연결하려는 경우 각 네트워크에 대한 개별 프로필을 생성합니다.

2. 대화형 모드에서 **nmcli** 를 시작합니다.
   - 추가 연결 프로필을 생성하려면 다음을 입력합니다.

```
# nmcli connection edit type ethernet con-name "<connection-name>"
```

   - 기존 연결 프로필을 수정하려면 다음을 입력합니다.

```
# nmcli connection edit con-name "<connection-name>"
```

3. 선택 사항: 연결 프로필의 이름을 변경합니다.

```
nmcli> set connection.id Internal-LAN
```

   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

   nmcli 가 이름의 따옴표 부분을 만들지 않도록 공백이 포함된 ID를 설정하는 데 따옴표를 사용하지 마십시오. 예를 들어 예제 연결을 ID로 설정하려면 **set connection.id Example Connection** 을 입력합니다.

4. 연결 프로필의 현재 설정을 표시합니다.

```
nmcli> print
...
connection.interface-name:     enp1s0
connection.autoconnect:        yes
ipv4.method:                   auto
ipv6.method:                   auto
...
```

5. 새 연결 프로필을 생성하는 경우 네트워크 인터페이스를 설정합니다.

```
nmcli> set connection.interface-name enp1s0
```

6. IPv4 설정을 구성합니다.

   - DHCP를 사용하려면 다음을 입력합니다.

     ```
     nmcli> set ipv4.method auto
     ```

     ipv4.method 가 이미 **auto** (기본값)로 설정된 경우 이 단계를 건너뜁니다.

   - 정적 IPv4 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 다음을 입력합니다.

     ```
     nmcli> ipv4.addresses 192.0.2.1/24
     Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
     nmcli> ipv4.gateway 192.0.2.254
     nmcli> ipv4.dns 192.0.2.200
     nmcli> ipv4.dns-search example.com
     ```

7. IPv6 설정을 구성합니다.

   - SLAAC(상태 비저장 주소 자동 구성)를 사용하려면 다음을 입력합니다.

     ```
     nmcli> set ipv6.method auto
     ```

     ipv6.method 가 이미 **auto** (기본값)로 설정된 경우 이 단계를 건너뜁니다.

   - 정적 IPv6 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 다음을 입력합니다.

     ```
     nmcli> ipv6.addresses 2001:db8:1::fffe/64
     Do you also want to set 'ipv6.method' to 'manual'? [yes]: yes
     nmcli> ipv6.gateway 2001:db8:1::fffe
     nmcli> ipv6.dns 2001:db8:1::ffbb
     nmcli> ipv6.dns-search example.com
     ```

8. 연결을 저장하고 활성화합니다.

```
nmcli> save persistent
```

9. 대화형 모드를 종료합니다.

```
nmcli> quit
```

검증

1. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 **이름 서버** 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

문제 해결

- 네트워크 커넥터가 호스트와 스위치에 연결되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 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 NetworkManager를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 NetworkManager 서비스 솔루션을 다시 시작한 후 NetworkManager 연결 중복을 참조하십시오.

추가 리소스

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

2.3. NMTUI를 사용하여 이더넷 연결 구성

이더넷을 통해 호스트를 네트워크에 연결하는 경우 **nmtui** 애플리케이션을 사용하여 텍스트 기반 사용자 인터페이스에서 연결의 설정을 관리할 수 있습니다. **nmtui** 를 사용하여 새 프로필을 만들고 그래픽 인터페이스 없이 호스트에서 기존 프로필을 업데이트합니다.

참고

nmtui 에서 :

- 커서 키를 사용하여 이동합니다.
- 버튼을 선택하고 **Enter** 를 누릅니다.
- **Space** 를 사용하여 확인란을 선택하고 지웁니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.

절차

1. 연결에 사용할 네트워크 장치 이름을 모르는 경우 사용 가능한 장치를 표시합니다.

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

2. start **nmtui**:

```
# nmtui
```

3. **Edit a connection** 을 선택하고 **Enter** 를 누릅니다.
4. 새 연결 프로필을 추가하거나 기존 프로필을 수정할지 선택합니다.
   - 새 프로필을 생성하려면 다음을 수행합니다.
     i. **추가를** 누릅니다.
     ii. 네트워크 유형 목록에서 **이더넷** 을 선택하고 **Enter** 키를 누릅니다.
   - 기존 프로필을 수정하려면 목록에서 프로필을 선택하고 **Enter** 키를 누릅니다.
5. 선택 사항: 연결 프로필의 이름을 업데이트합니다.
   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

6. 새 연결 프로필을 생성하는 경우 장치 필드에 네트워크 **장치** 이름을 입력합니다.

7. 환경에 따라 **IPv4 구성 및 IPv6 구성** 영역에서 IP 주소 설정을 적절하게 구성합니다. 이를 위해 다음 영역 옆에 있는 버튼을 누른 후 다음을 선택합니다.

   - **비활성화됨** (이 연결에 IP 주소가 필요하지 않은 경우).
   - **자동으로** DHCP 서버가 이 NIC에 IP 주소를 동적으로 할당하는 경우입니다.
   - **수동**: 네트워크에 고정 IP 주소 설정이 필요한 경우입니다. 이 경우 추가 필드를 작성해야 합니다.

     i. 추가 필드를 표시하도록 구성할 프로토콜 옆에 **Show** 를 누릅니다.

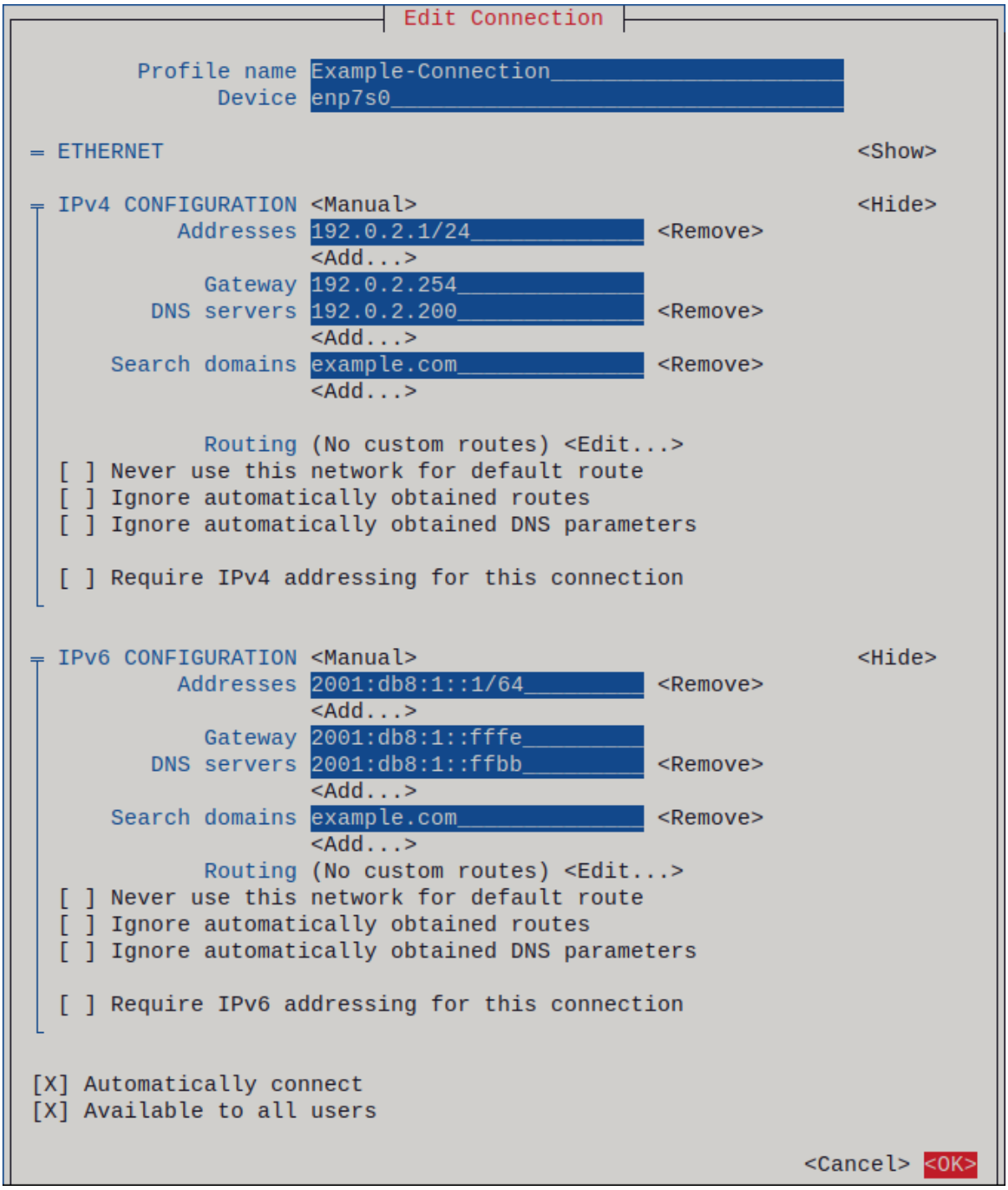
     ii. **주소** 옆에 있는 **추가** 를 클릭하고 CIDR(Classless Inter-Domain Routing) 형식으로 IP 주소와 서브넷 마스크를 입력합니다.
     서브넷 마스크를 지정하지 않으면 NetworkManager는 IPv4 주소에 대해 **/32** 서브넷 마스크와 IPv6 주소에 대해 **/64** 를 설정합니다.

     iii. 기본 게이트웨이의 주소를 입력합니다.

     iv. **DNS 서버** 옆에 있는 **추가** 를 클릭하고 DNS 서버 주소를 입력합니다.

     v. **검색 도메인** 옆에 있는 **추가** 를 클릭하고 DNS 검색 도메인을 입력합니다.

그림 2.1. 고정 IP 주소 설정을 사용하는 이더넷 연결의 예



8. **OK** 를 눌러 새 연결을 만들고 자동으로 활성화합니다.
9. **다시** 키를 눌러 기본 메뉴로 돌아갑니다.
10. **Quit** 를 선택하고 **Enter** 를 눌러 **nmtui** 애플리케이션을 종료합니다.

검증

1. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
```

```
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

   여러 연결 프로필이 동시에 활성화된 경우 **이름 서버** 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

문제 해결

- 네트워크 커넥터가 호스트와 스위치에 연결되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 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 NetworkManager를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 NetworkManager 서비스 솔루션을 다시 시작한 후 NetworkManager 연결 중복을 참조하십시오.

추가 리소스

- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성
- DNS 서버 순서 구성

2.4. CONTROL-CENTER를 사용하여 이더넷 연결 구성

이더넷을 통해 호스트를 네트워크에 연결하는 경우 GNOME 설정 메뉴를 사용하여 그래픽 인터페이스로 연결의 설정을 관리할 수 있습니다.

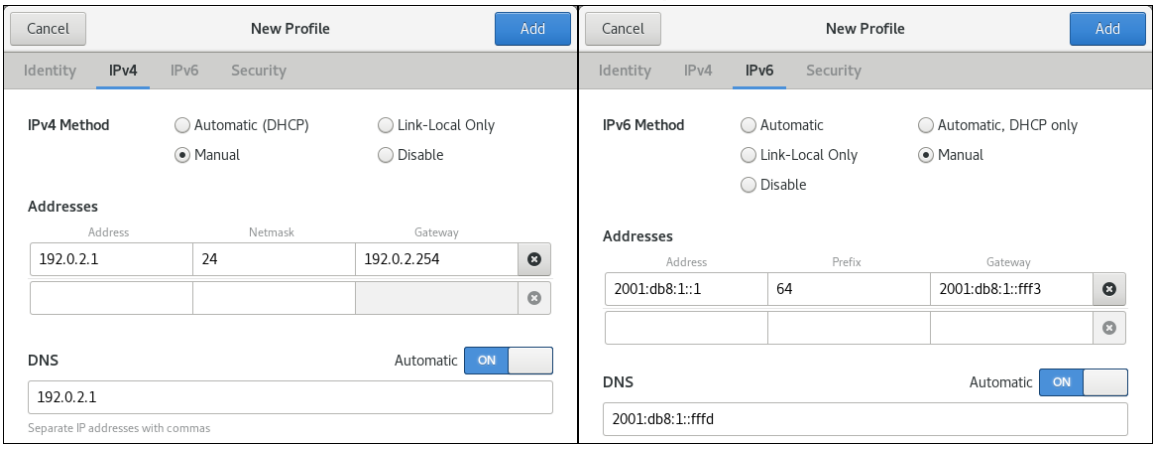
control-center 는 **nm-connection-editor** 애플리케이션 또는 **nmcli** 유틸리티만큼 많은 구성 옵션을 지원하지 않습니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.
- GNOME이 설치되어 있습니다.

절차

1. **Super** 키를 눌러 **Settings** 을 입력하고 **Enter** 를 누릅니다.
2. 왼쪽의 탐색에서 네트워크를 선택합니다.
3. **새 연결 프로필을 추가하거나 기존 프로필을 수정할지 선택합니다.**
   - **새 프로필을 생성하려면 이더넷 항목 옆에 있는+ 버튼을 클릭합니다.**
   - **기존 프로필을 수정하려면 프로필 항목 옆에 있는 장비 아이콘을 클릭합니다.**
4. **선택 사항: ID 탭에서 연결 프로필의 이름을 업데이트합니다.**
   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.
5. **환경에 따라 그에 따라 IPv4 및 IPv6 탭의 IP 주소 설정을 구성합니다.**
   - **DHCP 또는 IPv6 SLAAC(상태 비저장 주소 자동 구성)를 사용하려면 자동(DHCP)을 메서드(기본값) 로 선택합니다.**
   - **고정 IP 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 Manual 을 메서드로 선택하고 탭의 필드를 작성합니다.**



6. **연결 프로필을 추가하거나 수정할지 여부에 따라 추가 또는 적용 버튼을 클릭하여 연결을 저장합니다.**
   GNOME control-center 가 연결을 자동으로 활성화합니다.

검증

1. **NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 이름 서버 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

문제 해결 단계

- 네트워크 커넥터가 호스트와 스위치에 연결되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 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 NetworkManager를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 NetworkManager 서비스 솔루션을 다시 시작한 후 NetworkManager 연결 중복을참조하십시오.

2.5. NM-CONNECTION-EDITOR를 사용하여 이더넷 연결 구성

이더넷을 통해 호스트를 네트워크에 연결하는 경우 nm-connection-editor 애플리케이션을 사용하여 그래픽 인터페이스로 연결의 설정을 관리할 수 있습니다.

사전 요구 사항

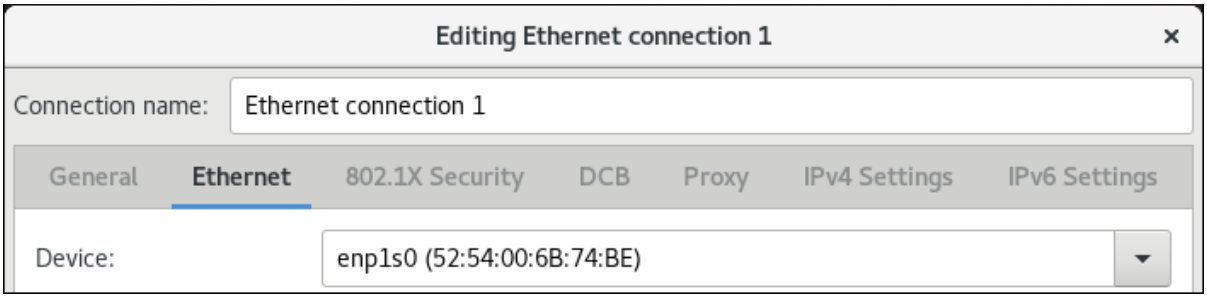
- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.
- GNOME이 설치되어 있습니다.

절차

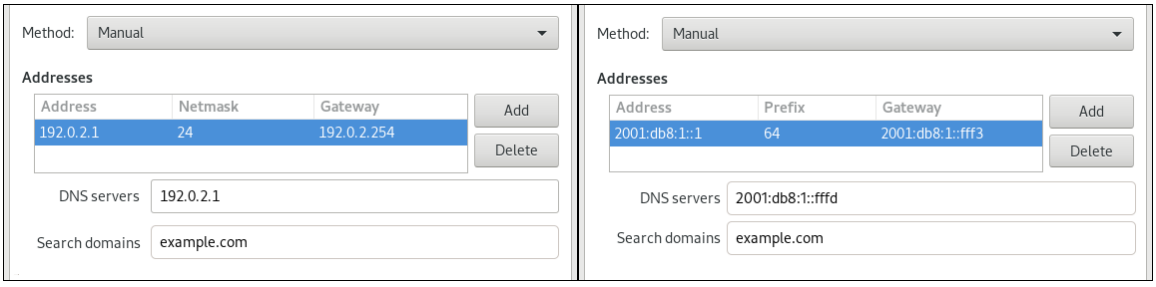
1. 터미널을 열고 다음을 입력합니다.

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

2. 새 연결 프로필을 추가하거나 기존 프로필을 수정할지 선택합니다.
   - 새 프로필을 생성하려면 다음을 수행합니다.
     i. **+** 버튼을 클릭합니다.
     ii. 연결 유형으로 이더넷 을 선택하고 생성 을 클릭합니다.
   - 기존 프로필을 수정하려면 프로필 항목을 두 번 클릭합니다.
3. 선택 사항: 연결 이름 필드에 있는 프로필 이름을 업데이트합니다.
   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.
4. 새 프로필을 생성하는 경우 이더넷 탭에서 장치를 선택합니다.



5. 환경에 따라 그에 따라 IPv4 설정 및 IPv6 설정 탭에서 IP 주소 설정을 구성합니다.
   - DHCP 또는 IPv6 SLAAC(상태 비저장 주소 자동 구성)를 사용하려면 자동(DHCP)을 메서드(기본값) 로 선택합니다.
   - 고정 IP 주소, 네트워크 마스크, 기본 게이트웨이, DNS 서버 및 검색 도메인을 설정하려면 **Manual** 을 메서드로 선택하고 탭의 필드를 작성합니다.



6. 저장을 클릭합니다.
7. nm-connection-editor 를 종료합니다.

검증

1. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 이름 서버 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

문제 해결 단계

- 네트워크 커넥터가 호스트와 스위치에 연결되어 있는지 확인합니다.
- 링크 실패가 이 호스트에만 있는지 또는 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 NetworkManager를 시작하거나 다시 시작하면 장치 구성이 반영되는 메모리 내 연결이 생성됩니다. 자세한 내용과 이 문제를 방지하는 방법은 NetworkManager 서비스 솔루션을 다시 시작한 후 NetworkManager 연결 중복을참조하십시오.

추가 리소스

- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성
- DNS 서버 순서 구성

2.6. NMSTATECTL을 사용하여 고정 IP 주소로 이더넷 연결 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 이더넷 연결을 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 **nmstatectl** 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.
- **nmstate** 패키지가 설치되어 있습니다.

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
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
```

이러한 설정은 다음 설정을 사용하여 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다.

- /**24** 서브넷 마스크가 있는 정적 IPv4 주소 -**192.0.2.1**
- 정적 IPv6 주소 - **2001:db8:1::1** (/**64** 서브넷 마스크 포함)
- 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**

2. 시스템에 설정을 적용합니다.

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

검증

1. 현재 상태를 YAML 형식으로 표시합니다.

```
# nmstatectl show enp1s0
```

2. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

3. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

4. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

5. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 이름 서버 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

추가 리소스

- **nmstatectl(8)** man page
- **/usr/share/doc/nmstate/examples/** 디렉터리

2.7. 인터페이스 이름으로 네트워크 RHEL 시스템 역할을 사용하여 고정 IP 주소로 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 이더넷 연결을 원격으로 구성할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.
- 관리형 노드는 NetworkManager를 사용하여 네트워크를 구성합니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: enp1s0
            interface_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
            state: up
```

이러한 설정은 다음 설정을 사용하여 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 -192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

2.8. 장치 경로와 함께 네트워크 RHEL 시스템 역할을 사용하여 고정 IP 주소로 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 이더넷 연결을 원격으로 구성할 수 있습니다.

다음 명령을 사용하여 장치 경로를 확인할 수 있습니다.

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

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.
- 서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.
- 관리형 노드는 NetworkManager를 사용하여 네트워크를 구성합니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP
      ansible.builtin.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
```

이러한 설정은 다음 설정을 사용하여 이더넷 연결 프로필을 정의합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 -192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
  이 예제의 match 매개변수는 Ansible이 PCI ID 0000:0[1-3].0 과 일치하는 장치에 플레이를 적용하지만 0000:00:02.0 과 일치하지 않음을 정의합니다. 사용할 수 있는 특수 수정자 및 와일드카드에 대한 자세한 내용은 /usr/share/ansible/roles/rhel-system-roles.network/README.md 파일의 match 매개변수 설명을 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

2.9. NMSTATECTL을 사용하여 동적 IP 주소로 이더넷 연결 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 이더넷 연결을 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 **nmstatectl** 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

사전 요구 사항

- 물리적 또는 가상 이더넷 NIC(네트워크 인터페이스 컨트롤러)가 서버 구성에 있습니다.
- DHCP 서버는 네트워크에서 사용할 수 있습니다.
- **nmstate** 패키지가 설치되어 있습니다.

절차

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

```
---
interfaces:
- name: enp1s0
  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
```

이러한 설정은 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다. 연결은 DHCP 서버와 IPv6 상태 비저장 주소 자동 구성(SLAAC)에서 IPv4 주소, IPv6 주소, 기본 게이트웨이, 경로, DNS 서버 및 검색 도메인을 검색합니다.

2. 시스템에 설정을 적용합니다.

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

검증

1. 현재 상태를 YAML 형식으로 표시합니다.

```
# nmstatectl show enp1s0
```

2. NIC의 IP 설정을 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 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

3. IPv4 기본 게이트웨이를 표시합니다.

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

4. IPv6 기본 게이트웨이를 표시합니다.

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

5. DNS 설정을 표시합니다.

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

여러 연결 프로필이 동시에 활성화된 경우 이름 서버 항목의 순서는 이러한 프로필의 DNS 우선 순위 값과 연결 유형에 따라 달라집니다.

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

```
# ping <host-name-or-IP-address>
```

추가 리소스

- **nmstatectl(8)** man page
- **/usr/share/doc/nmstate/examples/** 디렉터리

2.10. 인터페이스 이름으로 네트워크 RHEL 시스템 역할을 사용하여 동적 IP 주소로 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 이더넷 연결을 원격으로 구성할 수 있습니다. 동적 IP 주소 설정과의 연결의 경우 NetworkManager는 DHCP 서버에서 연결에 대한 IP 설정을 요청합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.
- DHCP 서버는 네트워크에서 사용 가능
- 관리형 노드는 NetworkManager를 사용하여 네트워크를 구성합니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/**playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with dynamic IP
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: enp1s0
            interface_name: enp1s0
            type: ethernet
            autoconnect: yes
            ip:
              dhcp4: yes
              auto6: yes
            state: up
```

이러한 설정은 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다. 연결은 DHCP 서버와 IPv6 상태 비저장 주소 자동 구성(SLAAC)에서 IPv4 주소, IPv6 주소, 기본 게이트웨이, 경로, DNS 서버 및 검색 도메인을 검색합니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

2.11. 장치 경로에서 네트워크 RHEL 시스템 역할을 사용하여 동적 IP 주소로 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 이더넷 연결을 원격으로 구성할 수 있습니다. 동적 IP 주소 설정과의 연결의 경우 NetworkManager는 DHCP 서버에서 연결에 대한 IP 설정을 요청합니다.

다음 명령을 사용하여 장치 경로를 확인할 수 있습니다.

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

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.
- 서버 구성에 물리적 또는 가상 이더넷 장치가 있습니다.
- DHCP 서버는 네트워크에서 사용할 수 있습니다.
- 관리 호스트는 NetworkManager를 사용하여 네트워크를 구성합니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with dynamic IP
      ansible.builtin.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
```

이러한 설정은 이더넷 연결 프로필을 정의합니다. 연결은 DHCP 서버와 IPv6 상태 비저장 주소 자동 구성(SLAAC)에서 IPv4 주소, IPv6 주소, 기본 게이트웨이, 경로, DNS 서버 및 검색 도메인을 검색합니다.

match 매개 변수는 Ansible이 PCI ID0000:00:0[1-3].0 과 일치하는 장치에 플레이를 적용하지만 0000:00:02.0 이 아닌 장치에 플레이를 적용하도록 정의합니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

2.12. 인터페이스 이름으로 단일 연결 프로필을 사용하여 여러 이더넷 인터페이스 구성

대부분의 경우 하나의 연결 프로필에는 하나의 네트워크 장치의 설정이 포함됩니다. 그러나 NetworkManager는 연결 프로필에서 인터페이스 이름을 설정할 때 와일드카드도 지원합니다. 동적 IP 주소 할당을 사용하는 이더넷 네트워크 간에 호스트가 로밍하는 경우 이 기능을 사용하여 여러 이더넷 인터페이스에 사용할 수 있는 단일 연결 프로필을 만들 수 있습니다.

사전 요구 사항

- 서버 구성에 여러 개의 물리적 또는 가상 이더넷 장치가 있습니다.
- DHCP 서버는 네트워크에서 사용할 수 있습니다.
- 호스트에 연결 프로필이 없습니다.

절차

1. **enp**: enp로 시작하는 모든 인터페이스 이름에 적용되는 연결 프로파일을 추가합니다.

```
# nmcli connection add con-name "Wired connection 1" connection.multi-connect
multiple match.interface-name enp* type ethernet
```

검증

1. 단일 연결 프로필의 모든 설정을 표시합니다.

```
# nmcli connection show "Wired connection 1"
connection.id:                          Wired connection 1
...
connection.multi-connect:               3 (multiple)
match.interface-name:                   enp*
...
```

3 은 연결 프로파일에서 동시에 활성 상태인 인터페이스의 수를 나타내며 연결 프로파일의 네트워크 인터페이스 수는 아닙니다. 연결 프로필은 **match.interface-name** 매개변수의 패턴과 일치하는 모든 장치를 사용하므로 연결 프로파일에 동일한 UBI(Universally Unique Identifier)가 있습니다.

2. 연결 상태를 표시합니다.

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
...
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp7s0
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp8s0
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp9s0
```

추가 리소스

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

2.13. PCI ID를 사용하여 여러 이더넷 인터페이스에 대해 단일 연결 프로필 구성

PCI ID는 시스템에 연결된 장치의 고유 식별자입니다. 연결 프로필은 PCI ID 목록에 따라 인터페이스를 일치시켜 여러 장치를 추가합니다. 이 절차를 사용하여 여러 장치 PCI ID를 단일 연결 프로필에 연결할 수 있습니다.

사전 요구 사항

- 서버 구성에 여러 개의 물리적 또는 가상 이더넷 장치가 있습니다.
- DHCP 서버는 네트워크에서 사용할 수 있습니다.
- 호스트에 연결 프로필이 없습니다.

절차

1. 장치 경로를 식별합니다. 예를 들어 **enp** 로 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 **:0[7-8].0** 표현식과 일치하는 모든 **PCI ID**에 적용되는 연결 프로필을 추가합니다.

```
# nmcli connection add type ethernet connection.multi-connect multiple match.path
"pci-0000:07:00.0 pci-0000:08:00.0" con-name "Wired connection 1"
```

검증

1. 연결 상태를 표시합니다.

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1   9cee0958-512f-4203-9d3d-b57af1d88466  ethernet  enp7s0
```

```
Wired connection 1   9cee0958-512f-4203-9d3d-b57af1d88466  ethernet  enp8s0
...
```

2. 연결 프로필의 모든 설정을 표시하려면 다음을 수행합니다.

```
# nmcli connection show "Wired connection 1"
connection.id:                Wired connection 1
...
connection.multi-connect:     3 (multiple)
match.path:                   pci-0000:07:00.0,pci-0000:08:00.0
...
```

이 연결 프로필은 **match.path** 매개변수의 패턴과 일치하는 PCI ID가 있는 모든 장치를 사용하므로 연결 프로파일에는 동일한 UUID(Universally Unique Identifier)가 있습니다.

추가 리소스

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

3장. 네트워크 본딩 구성

네트워크 본딩은 물리적 및 가상 네트워크 인터페이스를 결합하거나 집계하여 처리량 또는 중복성이 높은 논리 인터페이스를 제공하는 방법입니다. 본딩에서 커널은 모든 작업을 독점적으로 처리합니다. 이더넷 장치 또는 VLAN과 같은 다양한 유형의 장치에 본딩을 생성할 수 있습니다.

Red Hat Enterprise Linux는 관리자에게 팀 장치를 구성하는 다양한 옵션을 제공합니다. 예를 들면 다음과 같습니다.

- **nmcli** 를 사용하여 명령줄을 사용하여 본딩 연결을 구성합니다.
- RHEL 웹 콘솔을 사용하여 웹 브라우저를 사용하여 본딩 연결을 구성합니다.
- **nmtui** 를 사용하여 텍스트 기반 사용자 인터페이스에서 본딩 연결을 구성합니다.
- **nm-connection-editor** 애플리케이션을 사용하여 그래픽 인터페이스에서 본딩 연결을 구성합니다.
- **nmstatectl** 을 사용하여 Nmstate API를 통한 본딩 연결을 구성합니다.
- RHEL 시스템 역할을 사용하여 하나 이상의 호스트에서 본딩 구성을 자동화합니다.

3.1. 컨트롤러 및 포트 인터페이스의 기본 동작 이해

NetworkManager 서비스를 사용하여 팀 또는 본딩 포트 인터페이스를 관리하거나 해결할 때 다음 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.

3.2. 본딩 모드에 따른 업스트림 스위치 구성

사용하려는 본딩 모드에 따라 스위치에서 포트를 구성해야 합니다.

본딩 모드	스위치 설정
0 - **balance-rr**	LACP(Link Aggregation Control Protocol) 협상이 아닌 정적 EtherChannel이 활성화되어 있어야 합니다.
1 - **active-backup**	스위치에 구성이 필요하지 않습니다.

본딩 모드	스위치 설정
2 - balance-xor	LACP-negotiated가 아닌 정적 EtherChannel이 활성화되어 있어야 합니다.
3 - broadcast	LACP-negotiated가 아닌 정적 EtherChannel이 활성화되어 있어야 합니다.
4 - 802.3ad	LACP-negotiated EtherChannel이 활성화되어 있어야 합니다.
5 - balance-tlb	스위치에 구성이 필요하지 않습니다.
6 - balance-alb	스위치에 구성이 필요하지 않습니다.

스위치를 구성하는 방법에 대한 자세한 내용은 스위치 설명서를 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 bonding을 참조하십시오. https://access.redhat.com/solutions/202583 **KCS 솔루션.**

3.3. NMCLI를 사용하여 네트워크 본딩 구성

명령줄에서 네트워크 본딩을 구성하려면 nmcli 유틸리티를 사용합니다.

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 에 설명된 대로 미리 생성할 수 있습니다.**
  - nmcli를 사용하여 네트워크 팀 구성
  - nmcli를 사용하여 네트워크 브리지 구성
  - nmcli를 사용하여 VLAN 태그 지정 설정

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.

미디어 독립 인터페이스(MII) 모니터링 간격을 추가로 설정하려면 miimon=interval 옵션을 bond.options 속성에 추가합니다. 예를 들면 다음과 같습니다.

```
# 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
```

이러한 명령은 **enp7s0** 및 **enp8s0** 에 대한 프로필을 생성하여 **bond0** 연결에 추가합니다.

b. 기존 연결 프로필을 본딩에 할당하려면 다음을 수행합니다.

i. 이러한 연결의 **master** 매개변수를 **bond0** 으로 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.

ii. 연결을 다시 활성화합니다.

```
# nmcli connection up bridge0
# nmcli connection up bridge1
```

4. IPv4 설정을 구성합니다.

- 이 본딩 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify bond0 ipv4.method disabled
```

- DHCP를 사용하려면 작업이 필요하지 않습니다.

- 정적 IPv4 주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를 **bond0** 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search 'example.com' ipv4.method
manual
```

5. IPv6 설정을 구성합니다.

- 이 본딩 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify bond0 ipv6.method disabled
```

- SLAAC(stateless address autoconfiguration)를 사용하려면 작업이 필요하지 않습니다.
- 정적 IPv6 주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를 **bond0** 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify bond0 ipv6.addresses '2001:db8:1::1/64' ipv6.gateway
'2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd' ipv6.dns-search 'example.com'
ipv6.method manual
```

6. 선택 사항: 본딩 포트에서 매개변수를 설정하려면 다음 명령을 사용합니다.

```
# nmcli connection modify bond0-port1 bond-port.<parameter> <value>
```

7. 연결을 활성화합니다.

```
# nmcli connection up bond0
```

8. 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 NetworkManager는 본딩을 활성화하지만 다른 포트는 활성화하지 않습니다. 본딩이 활성화되면 Red Hat Enterprise Linux가 모든 포트를 자동으로 사용하도록 설정할 수 있습니다.

a. bond의 **connection.autoconnect-slaves** 매개변수를 활성화합니다.

```
# nmcli connection modify bond0 connection.autoconnect-slaves 1
```

b. 브리지를 다시 활성화합니다.

```
# nmcli connection up bond0
```

검증

1. 네트워크 장치 중 하나에서 네트워크 케이블을 일시적으로 제거하고 본딩의 다른 장치가 트래픽을 처리하는지 확인합니다.
   소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. **nmcli** 와 같은 연결을 비활성화하는 툴에는 포트 구성 변경 사항을 처리하는 본딩 드라이버 기능만 표시하고 실제 링크 실패 이벤트가 아닙니다.

2. 본딩 상태를 표시합니다.

```
# cat /proc/net/bonding/bond0
```

3.4. RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성

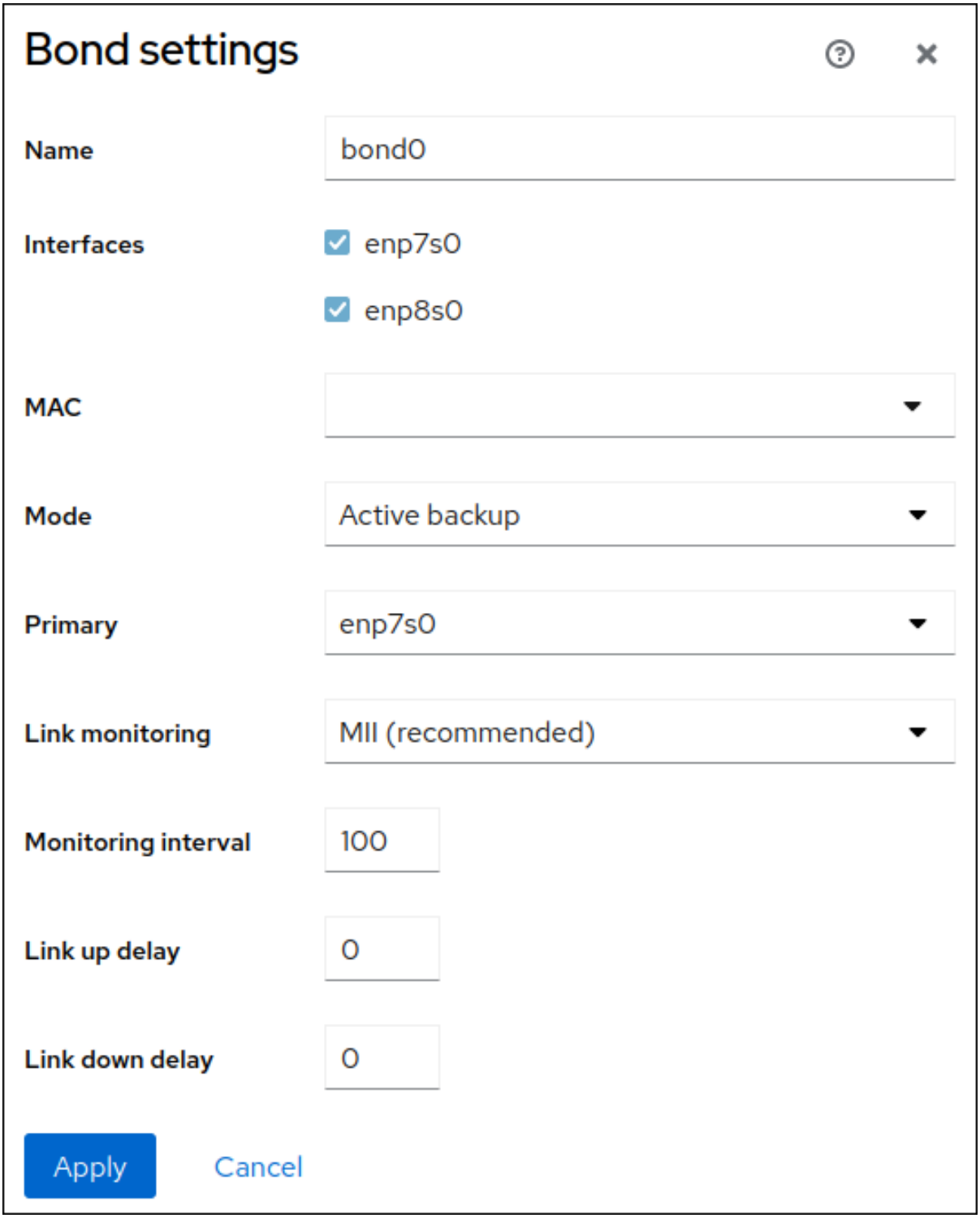
웹 브라우저 기반 인터페이스를 사용하여 네트워크 설정을 관리하려면 RHEL 웹 콘솔을 사용하여 네트워크 본딩을 구성합니다.

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 팀, 브리지 또는 VLAN 장치를 사전에 만듭니다.
  - RHEL 웹 콘솔을 사용하여 네트워크 팀 구성
  - RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성
  - RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 설정

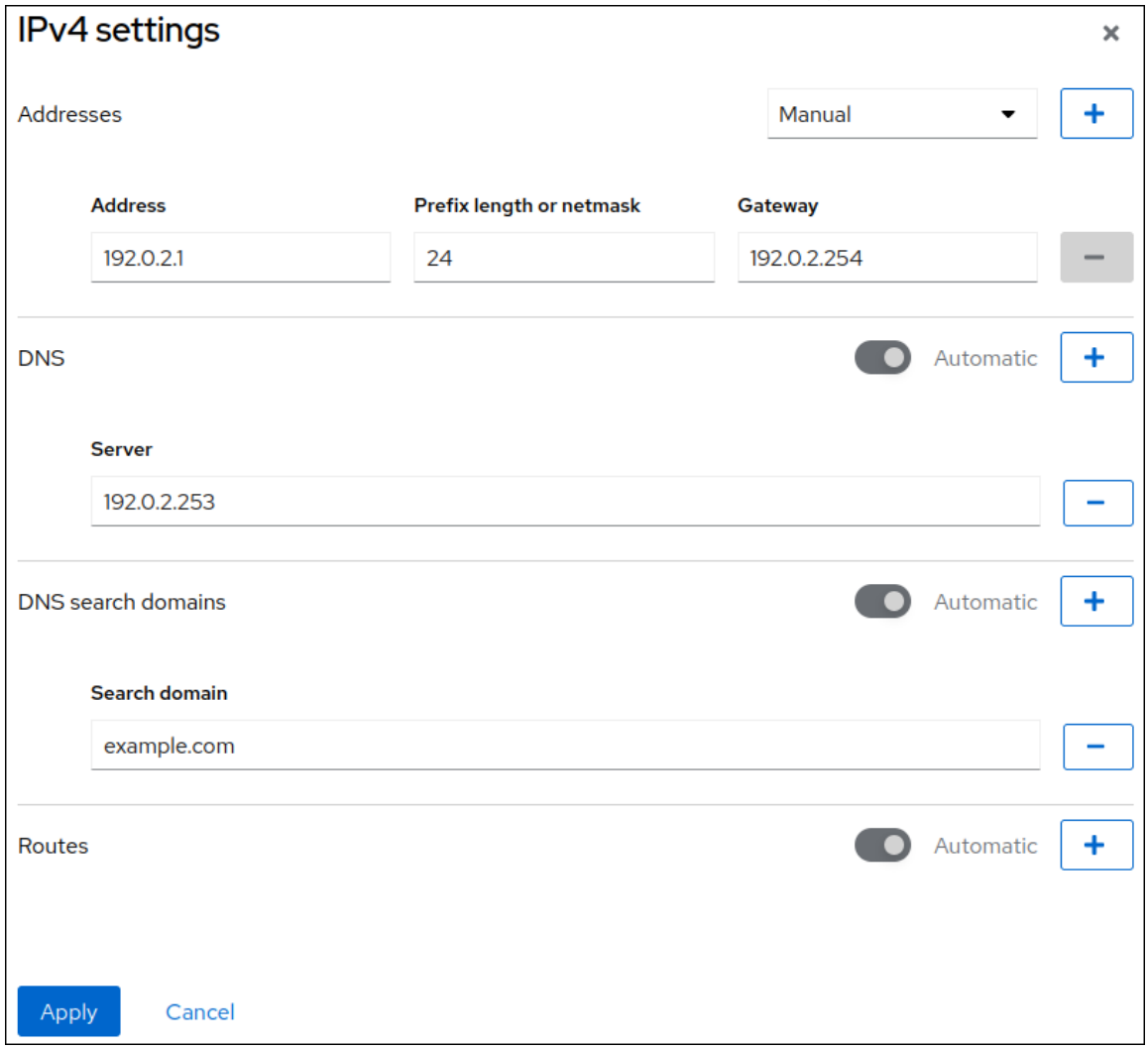
절차

1. 화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.
2. 인터페이스 섹션에서 본딩 추가 를 클릭합니다.
3. 생성할 본딩 장치의 이름을 입력합니다.
4. 본딩의 멤버여야 하는 인터페이스를 선택합니다.
5. 본딩 모드를 선택합니다.
   활성 백업을 선택하면 웹 콘솔에 기본 활성화된 장치를 선택할 수 있는 추가 필드가 표시됩니다.
6. 링크 모니터링 모드를 설정합니다. 예를 들어 Adaptive 로드 밸런싱 모드를 사용하는 경우 이를 **ARP** 로 설정합니다.
7. 선택 사항: 모니터링 간격을 조정하고, 지연 링크를 연결하며, 지연 설정을 링크 다운합니다. 일반적으로 문제 해결 목적으로만 기본값을 변경합니다.



8. **Apply**(적용)를 클릭합니다.
9. 기본적으로 본딩에서는 동적 IP 주소를 사용합니다. 고정 IP 주소를 설정하려면 다음을 수행합니다.
   a. Interfaces 섹션에서 본딩 이름을 클릭합니다.
   b. 구성할 프로토콜 옆에 있는 편집을 클릭합니다.
   c. 주소 옆에 있는수동 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.

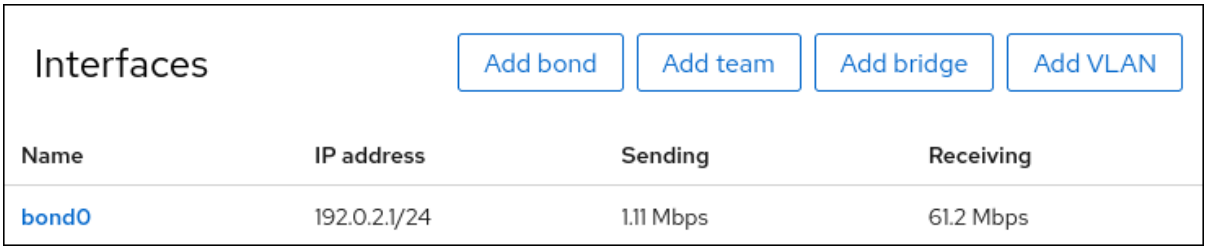
f. 인터페이스에 정적 경로가 필요한 경우 Routes 섹션에서 구성합니다.



g. 적용을클릭합니다.

검증

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



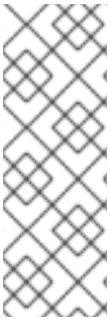
2. 네트워크 장치 중 하나에서 네트워크 케이블을 일시적으로 제거하고 본딩의 다른 장치가 트래픽을 처리하는지 확인합니다.
소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. 웹 콘솔과 같은 연결을 비활성화하는 툴에는 멤버 구성 변경 사항을 처리하는 본딩 드라이버 기능만 표시되고 실제 링크 실패 이벤트가 아님.

3. 본딩 상태를 표시합니다.

```
# cat /proc/net/bonding/bond0
```

3.5. NMTUI를 사용하여 네트워크 본딩 구성

nmtui 애플리케이션은 NetworkManager에 대한 텍스트 기반 사용자 인터페이스를 제공합니다.nmtui 를 사용하여 그래픽 인터페이스 없이 호스트에서 네트워크 본딩을 구성할 수 있습니다.



참고

nmtui 에서 :

- **커서 키를 사용하여 이동합니다.**
- **버튼을 선택하고 Enter 를 누릅니다.**
- **Space 를 사용하여 확인란을 선택하고 지웁니다.**

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.**

절차

1. **네트워크 본딩을 구성할 네트워크 장치 이름을 모르는 경우 사용 가능한 장치를 표시합니다.**

```
# nmcli device status
DEVICE    TYPE      STATE          CONNECTION
enp7s0    ethernet  unavailable    --
enp8s0    ethernet  unavailable    --
...
```

2. **start nmtui:**

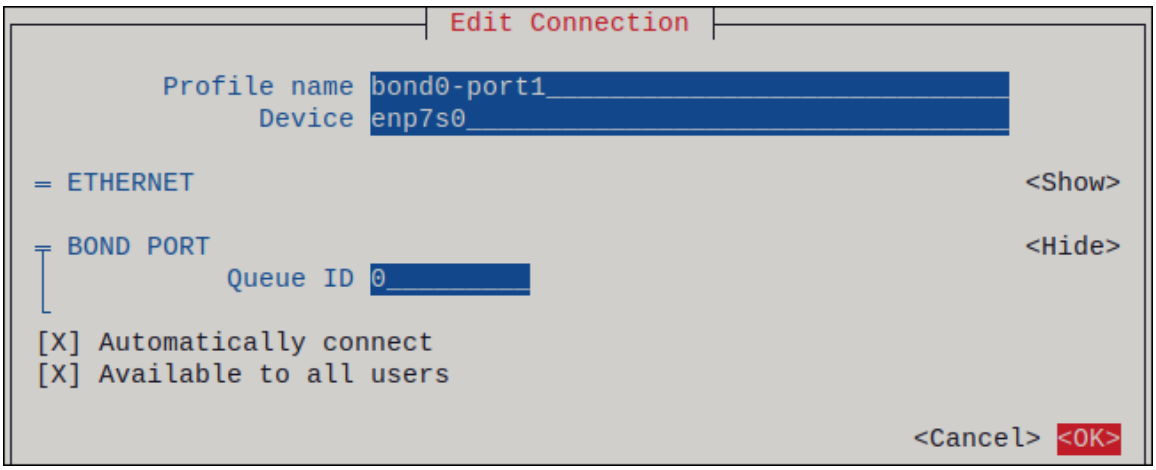
```
# nmtui
```

3. **Edit a connection을 선택하고Enter 를 누릅니다.**
4. **추가를 누릅니다.**
5. **네트워크 유형 목록에서 Bond 를 선택하고Enter 를 누릅니다.**
6. **선택 사항: 생성할 NetworkManager 프로필의 이름을 입력합니다.**
   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.
7. **장치 필드에 생성할 본딩 장치 이름을 입력합니다.**
8. **생성할 본딩에 포트를 추가합니다.**
   a. **Slaves 목록 옆에 있는Add 를 누릅니다.**
   b. **본딩에 포트로 추가할 인터페이스 유형을 선택합니다(예: 이더넷).**
   c. **선택 사항: 이 본딩 포트에 대해 생성할 NetworkManager 프로필의 이름을 입력합니다.**

d. 장치의 장치 이름을 장치 필드에 입력합니다.

e. OK 를 눌러 본딩 설정으로 창으로 돌아갑니다.

그림 3.1. 이더넷 장치를 본딩에 포트로 추가



f. 이 단계를 반복하여 더 많은 포트를 본딩에 추가합니다.

9. 본딩 모드를 설정합니다. 설정한 값에 따라 **nmtui** 는 선택한 모드와 관련된 설정에 대한 추가 필드를 표시합니다.

10. 환경에 따라 그에 따라 IPv4 구성 및 IPv6 구성 영역에서 IP 주소 설정을 구성합니다. 이를 위해 다음 영역 옆에 있는 버튼을 누른 후 다음을 선택합니다.

- 본딩에 IP 주소가 필요하지 않은 경우 비활성화됨
- DHCP 서버 또는 SLAAC(상태 비저장 주소 자동 구성)가 IP 주소를 본딩에 동적으로 할당하는 경우 자동입니다.
- 네트워크에 고정 IP 주소 설정이 필요한 경우 수동. 이 경우 추가 필드를 작성해야 합니다.

i. 추가 필드를 표시하도록 구성할 프로토콜 옆에 **Show** 를 누릅니다.

ii. 주소 옆에 있는 추가 를 클릭하고 **CIDR(Classless Inter-Domain Routing)** 형식으로 **IP** 주소와 서브넷 마스크를 입력합니다.

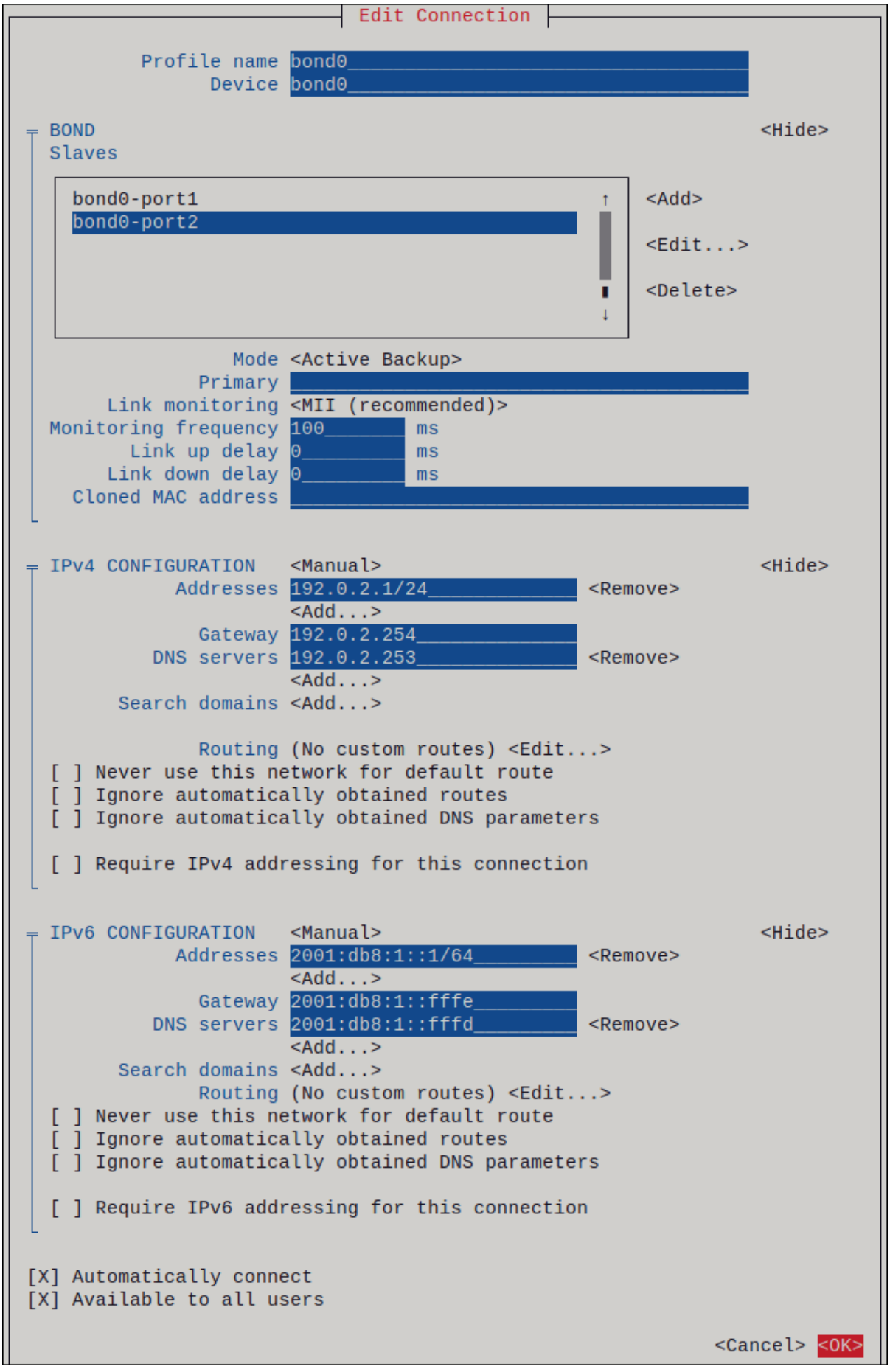
서브넷 마스크를 지정하지 않으면 **NetworkManager**는 **IPv4** 주소에 대해 **/32** 서브넷 마스크와 **IPv6** 주소에 대해 **/64** 를 설정합니다.

iii. 기본 게이트웨이의 주소를 입력합니다.

iv. **DNS 서버 옆에 있는 추가 를 클릭하고 DNS 서버 주소를 입력합니다.**

v. **검색 도메인 옆에 있는 추가 를 클릭하고 DNS 검색 도메인을 입력합니다.**

그림 3.2. 고정 IP 주소 설정을 사용한 본딩 연결 예



11. **OK** 를 눌러 새 연결을 만들고 자동으로 활성화합니다.

12. 다시 키를 눌러 기본 메뉴로 돌아갑니다.

13. **Quit** 를 선택하고 **Enter** 를 눌러 **nmtui** 애플리케이션을 종료합니다.

검증

1. 네트워크 장치 중 하나에서 네트워크 케이블을 일시적으로 제거하고 본딩의 다른 장치가 트래픽을 처리하는지 확인합니다.

   소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. **nmcli** 와 같은 연결을 비활성화하는 툴에는 포트 구성 변경 사항을 처리하는 본딩 드라이버 기능만 표시하고 실제 링크 실패 이벤트가 아닙니다.

2. 본딩 상태를 표시합니다.

```
# cat /proc/net/bonding/bond0
```

3.6. NM-CONNECTION-EDITOR를 사용하여 네트워크 본딩 구성

그래픽 인터페이스와 함께 **Red Hat Enterprise Linux**를 사용하는 경우 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 본딩을 구성할 수 있습니다.

nm-connection-editor 는 새 포트만 본딩에 추가할 수 있습니다. 기존 연결 프로필을 포트로 사용하려면 **nmcli** 를 사용하여 네트워크 본딩 구성에 설명된 **nmcli** 유틸리티를 사용하여 본딩 을 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 확

인합니다.

절차

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

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

2. **+** 버튼을 클릭하여 새 연결을 추가합니다.

3. 연결 유형을 선택하고 만들기 를 클릭합니다.

4. 본딩 탭에서 다음을 수행합니다.

   a. 선택 사항: 인터페이스 이름 필드에 본딩 인터페이스의 이름을 설정합니다.

   b. **Add**(추가) 버튼을 클릭하여 네트워크 인터페이스를 본딩에 포트로 추가합니다.

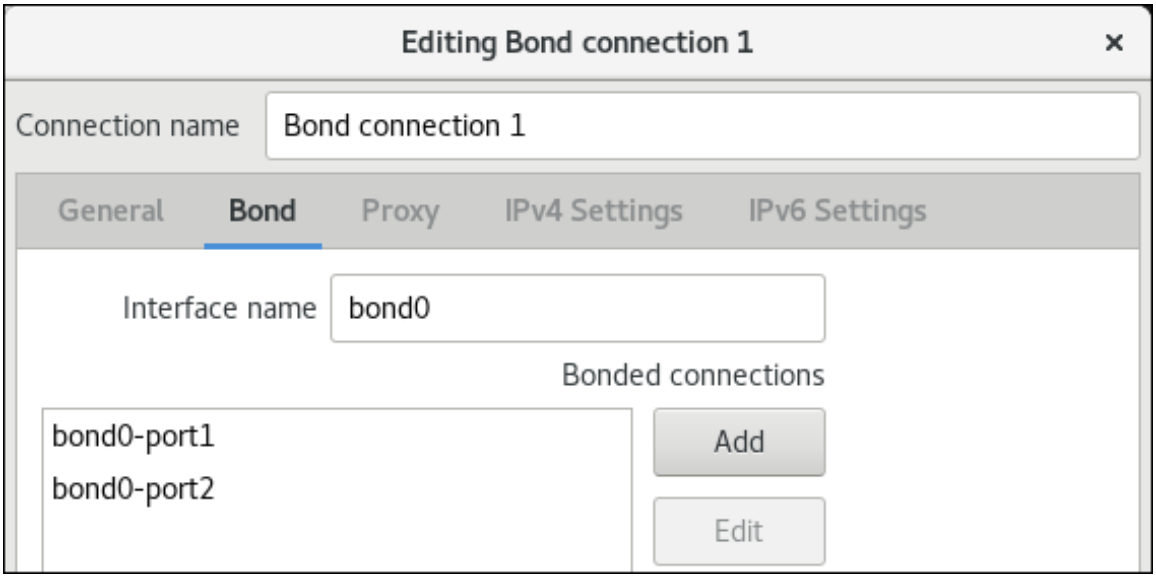
      i. 인터페이스의 연결 유형을 선택합니다. 예를 들어 유선 연결로 이더넷 을 선택합니다.

      ii. 선택 사항: 포트에 대한 연결 이름을 설정합니다.

      iii. 이더넷 장치에 대한 연결 프로필을 생성하는 경우 이더넷 탭을 열고 장치 필드에서 본딩에 포트로 추가할 네트워크 인터페이스를 선택합니다. 다른 장치 유형을 선택한 경우 적절하게 구성합니다. 구성되지 않은 본딩에서는 이더넷 인터페이스만 사용할 수 있습니다.

      iv. 저장을 클릭합니다.

   c. 본딩에 추가할 각 인터페이스에 대해 이전 단계를 반복합니다.



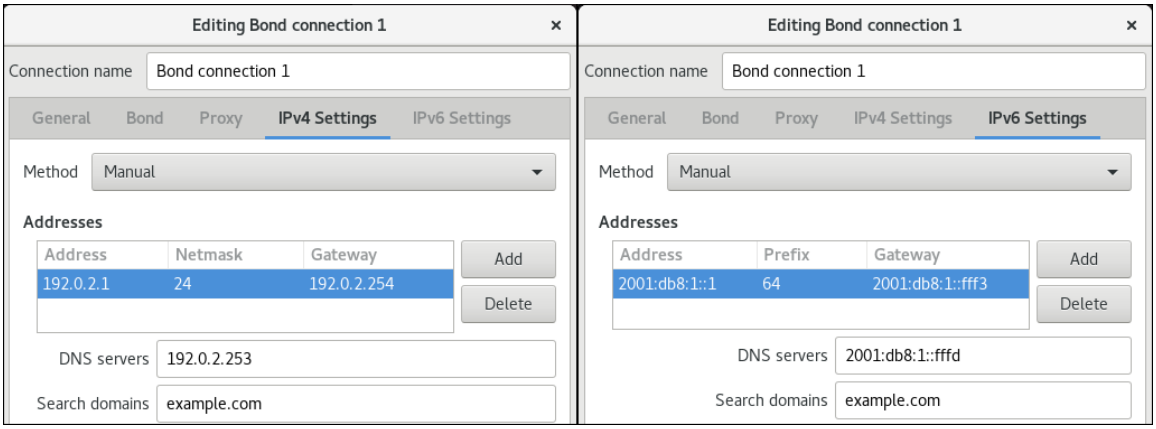
d.

선택 사항: **MII(Media Independent Interface)** 모니터링 간격과 같은 기타 옵션을 설정합니다.

5.

IPv4 설정 및 **IPv6 설정** 탭에서 **IP** 주소 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 **Method** 필드를 **Disabled** 로 설정합니다.
- **DHCP**를 사용하려면 **Method** 필드를 기본값인 **Automatic(DHCP)** 으로 둡니다.
- 고정 **IP** 설정을 사용하려면 **Method** 필드를 **Manual** 로 설정하고 그에 따라 필드를 작성합니다.



6. 저장을 클릭합니다.

7. nm-connection-editor 를 종료합니다.

검증

1. 네트워크 장치 중 하나에서 네트워크 케이블을 일시적으로 제거하고 본딩의 다른 장치가 트래픽을 처리하는지 확인합니다.

   소프트웨어 유틸리티를 사용하여 오류 이벤트를 올바르게 연결하는 방법은 없습니다. nmcli 와 같은 연결을 비활성화하는 툴에는 포트 구성 변경 사항을 처리하는 본딩 드라이버 기능만 표시하고 실제 링크 실패 이벤트가 아닙니다.

2. 본딩 상태를 표시합니다.

```
# cat /proc/net/bonding/bond0
```

추가 리소스

- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성
- nm-connection-editor를 사용하여 네트워크 팀 구성
- nm-connection-editor를 사용하여 네트워크 브리지 구성
- nm-connection-editor를 사용하여 VLAN 태그 지정 설정

3.7. NMSTATECTL을 사용하여 네트워크 본딩 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 네트워크 본딩을 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 nmstatectl 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

환경에 따라 YAML 파일을 적절하게 조정합니다. 예를 들어 본딩의 이더넷 어댑터와 다른 장치를 사용하려면 본딩에서 사용하는 포트의 base-iface 특성 및 유형 속성을 조정합니다.

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 본딩에서 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
```

이러한 설정은 다음 설정으로 네트워크 본딩을 정의합니다.

- 본딩의 네트워크 인터페이스 **enp1s0** 및 **enp7s0**
- 모드: **active-backup**
- 정적 **IPv4** 주소: **192.0.2.1** 및 **/24** 서브넷 마스크
- 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

2. 시스템에 설정을 적용합니다.

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

검증

1. 장치 및 연결 상태를 표시합니다.

```
# nmcli device status
DEVICE      TYPE      STATE      CONNECTION
bond0       bond      connected  bond0
```

2. 연결 프로필의 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) man page
- /usr/share/doc/nmstate/examples/ 디렉터리

3.8. 네트워크 RHEL 시스템 역할을 사용하여 네트워크 본딩 구성

네트워크 RHEL 시스템 역할을 사용하여 네트워크 본딩을 원격으로 구성할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.

절차

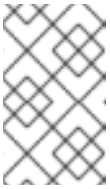
1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a network bond that uses two Ethernet ports
      ansible.builtin.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
```

이러한 설정은 다음 설정으로 네트워크 본딩을 정의합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
- 본딩 포트 - enp7s0 및 enp8s0
- 본딩 모드 - active-backup



참고

Linux 본딩 포트에가 아니라 본딩에서 IP 구성을 설정합니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

3.9. VPN을 중단하지 않고 이더넷과 무선 연결을 전환할 수 있는 네트워크 본딩 만들기

워크스테이션을 회사 네트워크에 연결하는 RHEL 사용자는 일반적으로 VPN을 사용하여 원격 리소스에 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 plug-Fi 연결을 사용하는 네트워크 본딩을 생성할 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 sha-Fi** 장치와 연결된 **hub-Fi**

절차

1. **active-backup** 모드로 본딩 인터페이스를 생성합니다.

```
# 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. 본딩의 **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 본딩에 할당합니다.

```
# nmcli connection modify Docking_station master bond0
```

6. **Wi-Fi** 연결의 연결 프로필을 본딩에 할당합니다.

```
# nmcli connection modify Wi-Fi master bond0
```

7. **Wi-Fi** 네트워크에서 **MAC** 필터링을 사용하여 허용 목록의 **MAC** 주소만 네트워크에 액세스할 수 있는 경우 **NetworkManager**가 활성 포트의 **MAC** 주소를 본딩에 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 구성합니다.

```
# nmcli connection modify bond0 connection.autoconnect-slaves 1
```

10. **bond0 연결을 활성화합니다.**

```
# nmcli connection up bond0
```

검증

- 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
- wi-Fi 연결 관리
- 네트워크 본딩 구성

3.10. 다양한 네트워크 본딩 모드

Linux 본딩 드라이버는 링크 집계를 제공합니다. 본딩은 단일 논리 본딩 인터페이스를 제공하기 위해 여러 네트워크 인터페이스를 병렬로 집계하는 프로세스입니다. 본딩된 인터페이스의 작업은 모드라고도 하는 본딩 정책에 따라 달라집니다. 다양한 모드는 로드 밸런싱 또는 핫란드 서비스를 제공합니다.

다음과 같은 모드가 있습니다.

balance-rr (Mode 0)

balance-rr 는 사용 가능한 첫 번째 포트에서 마지막 포트로 패킷을 순차적으로 전송하는 라운드 로빈 알고리즘을 사용합니다. 이 모드는 로드 밸런싱 및 내결함성을 제공합니다.

이 모드에서는 포트 집계 그룹(ECDHEChannel 또는 유사한 포트 그룹화)을 전환해야 합니다. Channel은 하나의 논리 이더넷 링크로 여러 개의 물리적 이더넷 링크를 그룹화하는 포트 링크 집계 기술입니다.

이 모드의 단점은 많은 워크로드에는 적합하지 않으며 TCP 처리량 또는 정렬된 패킷 전달이 필요한 경우입니다.

active-backup (Mode 1)

active-backup 에서는 본딩에서 하나의 포트만 활성 상태임을 결정하는 정책을 사용합니다. 이 모드는 내결함성을 제공하며 스위치 구성이 필요하지 않습니다.

활성 포트가 실패하면 대체 포트가 활성화됩니다. 본딩은 외부 주소 확인 프로토콜(ARP) 응답을 네트워크에 보냅니다. 무상 ARP는 ARP 프레임의 수신자가 전달 테이블을 업데이트하도록 강제 적용합니다. Active-backup 모드는 무차별 ARP를 전송하여 호스트의 연결을 유지하기 위한 새로운 경로를 발표합니다.

기본 옵션은 본딩 인터페이스의 기본 포트를 정의합니다.

balance-xor(Mode 2)

balance-xor 는 선택한 전송 해시 정책을 사용하여 패킷을 보냅니다. 이 모드는 로드 밸런싱, 내결함성을 제공하며, FlexVolume채널 또는 유사한 포트 그룹화를 설정하기 위해 스위치 구성이 필요합니다.

패킷 전송 및 밸런스 전송을 변경하려면 이 모드는 xmit_hash_policy 옵션을 사용합니다. 인터페이스의 트래픽 소스 또는 대상에 따라 인터페이스에 추가 로드 밸런싱 구성이 필요합니다. 설명

xmit_hash_policy bonding 매개변수를 참조하십시오.

브로드캐스트(Mode 3)

broadcast 는 모든 인터페이스에서 모든 패킷을 전송하는 정책을 사용합니다. 이 모드는 내결함성을 제공하며,ECDHEChannel 또는 유사한 포트 그룹화를 설정하기 위해 스위치 구성이 필요합니다.

이 모드의 단점은 많은 워크로드에는 적합하지 않으며 TCP 처리량 또는 정렬된 패킷 전달이 필요한 경우입니다.

802.3ad (Mode 4)

802.3ad 는 동일한 이름의 IEEE 표준 동적 링크 집계 정책을 사용합니다. 이 모드는 내결함성을 제공합니다. 이 모드에서는LACP(Link Aggregation Control Protocol) 포트 그룹화를 설정하려면 스위치 구성이 필요합니다.

이 모드에서는 동일한 속도와 모호한 설정을 공유하는 집계 그룹을 생성하고 활성 집계기의 모든 포트를 사용합니다. 인터페이스의 트래픽 소스 또는 대상에 따라 이 모드에는 추가 로드 밸런싱 구성이 필요합니다.

기본적으로 발신 트래픽에 대한 포트 선택 사항은 전송 해시 정책에 따라 다릅니다. 전송 해시 정책의 xmit_hash_policy 옵션을 사용하여 포트 선택 및 전송 밸런싱을 변경합니다.

802.3ad 와 Balance-xor 의 차이점은 규정 준수입니다. 802.3ad 정책은 포트 집계 그룹 간에 LACP를 협상합니다. 설명 xmit_hash_policy bonding 매개변수를참조하십시오.

balance-tlb (Mode 5)

balance-tlb 는 전송 로드 밸런싱 정책을 사용합니다. 이 모드에서는 스위치 지원이 필요하지 않은 내결함성, 로드 밸런싱 및 채널 본딩을 설정합니다.

활성 포트는 들어오는 트래픽을 수신합니다. 활성 포트가 실패하는 경우 다른 포트가 실패한 포트의 MAC 주소를 인수합니다. 발신 트래픽을 처리하는 인터페이스를 결정하려면 다음 모드 중 하나를 사용합니다.

- 값 0: 해시 배포 정책을 사용하여 부하 분산 없이 트래픽을 분산합니다.
- 값 1: 로드 밸런싱을 사용하여 각 포트에 트래픽 배포

본딩 옵션 **tlb_dynamic_lb=0** 을 사용하면 이 본딩 모드에서는 **xmit_hash_policy** 본딩 옵션을 사용하여 전송의 균형을 조정합니다. 기본 옵션은 본딩 인터페이스의 기본 포트를 정의합니다.

설명 **xmit_hash_policy bonding** 매개변수를 참조하십시오.

balance-alb (Mode 6)

balance-alb 는 조정형 로드 밸런싱 정책을 사용합니다. 이 모드는 내결함성, 로드 밸런싱을 제공하며 특별한 스위치 지원이 필요하지 않습니다.

이 모드에는 **IPv4** 및 **IPv6** 트래픽에 대한 **balance-transmit** 부하 분산(**balance-tlb**) 및 수신 로드 밸런싱이 포함됩니다. 본딩은 로컬 시스템에서 보낸 **ARP** 응답을 가로채고 본딩에서 포트 중 하나의 소스 하드웨어 주소를 덮어씁니다. **ARP** 협상은 수신 부하 분산을 관리합니다. 따라서 다른 포트는 서버에 다른 하드웨어 주소를 사용합니다.

기본 옵션은 본딩 인터페이스의 기본 포트를 정의합니다. 본딩 옵션 **tlb_dynamic_lb=0** 을 사용하면 이 본딩 모드에서는 **xmit_hash_policy** 본딩 옵션을 사용하여 전송의 균형을 조정합니다. 설명 **xmit_hash_policy bonding** 매개변수를 참조하십시오.

추가 리소스

- **/usr/share/doc/kernel-doc-<*version*>/Documentation/networking/bonding.rst** 는 **kernel-doc** 패키지에서 제공하는
- **/usr/share/doc/kernel-doc-<*version*>/Documentation/networking/bonding.txt** 는 **kernel-doc** 패키지에서 제공하는
- 가상 시스템 게스트 또는 컨테이너가 연결되는 브릿지와 함께 사용할 때 어떤 본딩 모드가 작동합니까?
- "xmit_hash_policy" bonding 매개변수의 다양한 정책 값은 어떻게 계산됩니까?

3.11. XMIT_HASH_POLICY BONDING 매개변수

xmit_hash_policy 로드 밸런싱 매개변수는 **balance-xor,802.3ad,balance-alb, balance-tlb** 모드에서 노드 선택에 대한 전송 해시 정책을 선택합니다. **tlb_dynamic_lb** 매개변수가 **0** 인 경우에만 모드 **5** 및 **6**에

적용됩니다. 이 매개변수의 가능한 값은 layer2 , layer2 +3,layer3+4,encap2+3,encap3+4, vlan+srcmac 입니다.

자세한 내용은 표를 참조하십시오.

정책 또는 네트워크 계층	Layer2	Layer2+3	Layer3+4	encap2+3	encap3+4	VLAN+src mac
사용	소스 및 대상 MAC 주소 및 이더넷 프로토콜 유형의 XOR	소스 및 대상 MAC 주소의 XOR	소스 및 대상 포트 및 IP 주소 XOR	지원되는 터널 내의 소스 및 대상 MAC 주소 및 IP 주소 (예: VXLAN(Virtual Extensible LAN)) XOR. 이 모드는 **skb_flow_dissect()** 함수를 사용하여 헤더 필드를 가져옵니다.	지원되는 터널 내의 소스 및 대상 포트 및 IP 주소 (예: VXLAN)의 XOR. 이 모드는 **skb_flow_dissect()** 함수를 사용하여 헤더 필드를 가져옵니다.	VLAN ID 및 소스 MAC 공급 업체 및 소스 MAC 장치의 XOR
트래픽 배치	동일한 기본 네트워크 인터페이스에서 특정 네트워크 피어로의 모든 트래픽	동일한 기본 네트워크 인터페이스의 특정 IP 주소로의 모든 트래픽	동일한 기본 네트워크 인터페이스의 특정 IP 주소 및 포트에 대한 모든 트래픽			
기본 선택	동일한 브로드캐스트 도메인에 이 시스템과 여러 다른 시스템 간에 네트워크 트래픽이 있는 경우	이 시스템과 여러 다른 시스템 간의 네트워크 트래픽이 기본 게이트웨이를 통과하는 경우	이 시스템과 다른 시스템 간의 네트워크 트래픽이 동일한 IP 주소를 사용하지만 여러 포트를 통과하는 경우	캡슐화된 트래픽은 소스 시스템과 여러 IP 주소를 사용하는 여러 다른 시스템 간의 트래픽입니다.	캡슐화된 트래픽은 여러 포트 번호를 사용하는 소스 시스템과 기타 시스템 간 트래픽입니다.	본딩에서 여러 컨테이너 또는 VM(가상 머신)의 네트워크 트래픽을 브리지 네트워크와 같은 외부 네트워크에 직접 노출하고 모드 2 또는 모드 4의 스위치를 구성할 수 없습니다.

보조 선택	네트워크 트래픽이 대부분 이 시스템과 기본 게이트웨이 뒤의 여러 다른 시스템 사이인 경우	네트워크 트래픽이 이 시스템과 다른 시스템 사이에 있는 경우				
compliant	802.3ad	802.3ad	802.3ad는 아닙니다.			
기본 정책	구성이 제공되지 않는 경우 기본 정책입니다.	IP가 아닌 트래픽의 경우 수식이 **layer2** 전송 정책과 동일합니다.	IP가 아닌 트래픽의 경우 수식이 **layer2** 전송 정책과 동일합니다.			

4장. 네트워크 티밍 구성

네트워크 팀은 물리적 및 가상 네트워크 인터페이스를 결합하거나 집계하여 처리량 또는 중복성이 높은 논리 인터페이스를 제공하는 방법입니다. 네트워크 티밍은 작은 커널 모듈을 사용하여 패킷 흐름에 대한 빠른 처리 및 기타 작업에 대한 사용자 공간 서비스를 구현합니다. 이렇게 하면 네트워크 티밍이 부하 분산 및 중복 요구 사항을 위한 확장 가능하고 확장 가능한 솔루션입니다.

Red Hat Enterprise Linux는 관리자에게 팀 장치를 구성하는 다양한 옵션을 제공합니다. 예를 들면 다음과 같습니다.

- **nmcli** 를 사용하여 명령줄을 사용하여 팀 연결을 구성합니다.
- **RHEL** 웹 콘솔을 사용하여 웹 브라우저를 사용하여 팀 연결을 구성합니다.
- **nm-connection-editor** 애플리케이션을 사용하여 그래픽 인터페이스에서 팀 연결을 구성합니다.

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.

4.1. 컨트롤러 및 포트 인터페이스의 기본 동작 이해

NetworkManager 서비스를 사용하여 팀 또는 본딩 포트 인터페이스를 관리하거나 해결할 때 다음 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.

4.2. TEAMD 서비스, 러너 및 링크-감시자 이해

팀 서비스인 **teamd** 는 팀 드라이버의 인스턴스 하나를 제어합니다. 이 드라이버 인스턴스는 하드웨어 장치 드라이버의 인스턴스를 추가하여 네트워크 인터페이스 팀을 구성합니다. 팀 드라이버는 **team0** 과 같은 네트워크 인터페이스를 커널에 제공합니다.

teamd 서비스는 모든 팀 작업에 공통 논리를 구현합니다. 이러한 함수는 라운드 로빈과 같은 다양한 로드 공유 및 백업 방법에 고유하며 실행자라고 하는 별도의 코드 단위로 구현 됩니다. 관리자는 **JSON(JavaScript Object Notation)** 형식으로 **runners**를 지정하고, 인스턴스를 생성할 때 **JSON** 코드가 **teamd** 인스턴스로 컴파일됩니다. 또는 **NetworkManager** 를 사용하는 경우 **team.runner** 매개변수에서 **runner**를 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.

- 임의의: 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-watchers를 사용할 수 있습니다.

- ethtool: libteam 라이브러리는 ethtool 유틸리티를 사용하여 링크 상태 변경 사항을 확인합니다. 기본 link-watcher입니다.

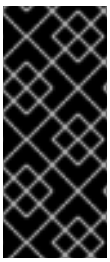
- arp_ping: libteam 라이브러리는 arp_ping 유틸리티를 사용하여 ARP(Address Resolution Protocol)를 사용하여 원격 하드웨어 주소의 존재를 모니터링합니다.

- nsna_ping: IPv6 연결에서 libteam 라이브러리는 IPv6 Neighbor Discovery 프로토콜의 Neighbor Advertisement 및 Neighbor Discovery 기능을 사용하여 블랜딩 인터페이스의 존재를 모니터링합니다.

각 실행기에서는 lacp 를 제외하고 모든 링크 감시자를 사용할 수 있습니다. 이 실행기에서는 ethtool 링크 감시자만 사용할 수 있습니다.

4.3. NMCLI를 사용하여 네트워크 팀 구성

명령줄에서 네트워크 팀을 구성하려면 nmcli 유틸리티를 사용합니다.



중요

네트워크 티밍은 Red Hat Enterprise Linux 9에서 더 이상 사용되지 않습니다. 서버를 RHEL의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 네트워크 본딩 구성 을 참조하십시오.

사전 요구 사항

- teamd 및 NetworkManager-team 패키지가 설치됩니다.

- 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 에 설명된 대로 미리 생성할 수 있습니다.

  - nmcli를 사용하여 네트워크 본딩 구성

  - nmcli를 사용하여 네트워크 브리지 구성

  - nmcli를 사용하여 VLAN 태그 지정 설정

절차

1. 팀 인터페이스를 생성합니다.

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

이 명령은 **activebackup** 실행기를 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 매개변수를 설정하려면 **name** 속성에 공백을 지정합니다. **name** 속성은 따옴표로 묶어야 합니다. 예를 들어 **ethtool** 링크 감시자를 사용하여 **delay-up** 매개변수를 **2500** 밀리초(2.5초)로 설정하려면 다음을 수행합니다.

```
# 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 구

분해야 합니다. 다음 예제에서는 **source-host** 및 **target-host** 매개변수를 사용하여 **delay-up** 매개변수 및 **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 프로필을 생성합니다.

```
# nmcli connection add type ethernet slave-type team con-name team0-port1
ifname enp7s0 master team0
# nmcli connection add type ethernet slave--type team con-name team0-port2
ifname enp8s0 master team0
```

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

.

b.
기존 연결 프로필을 팀에 할당하려면 다음을 수행합니다.

i.
이러한 연결의 **master** 매개변수를 **team0** 으로 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.

ii.
연결을 다시 활성화합니다.

```
# nmcli connection up bond0
# nmcli connection up bond1
```

5.
IPv4 설정을 구성합니다.

- 이 팀 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify team0 ipv4.method disabled
```

- **DHCP**를 사용하려면 작업이 필요하지 않습니다.

- 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **team0** 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify team0 ipv4.addresses '192.0.2.1/24' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search 'example.com' ipv4.method
manual
```

6.
IPv6 설정을 구성합니다.

- 이 팀 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify team0 ipv6.method disabled
```

- SLAAC(stateless address autoconfiguration)를 사용하려면 작업이 필요하지 않습니다.

- 정적 IPv6 주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를 team0 연결로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify team0 ipv6.addresses '2001:db8:1::1/64' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd' ipv6.dns-search 'example.com' ipv6.method manual
```

7. 연결을 활성화합니다.

```
# nmcli connection up team0
```

검증

- 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 up입니다.

추가 리소스

- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성
- teamd 서비스, 러너 및 link-watchers 이해
- nm-settings(5) 도움말 페이지
- teamd.conf(5) man page

4.4. RHEL 웹 콘솔을 사용하여 네트워크 팀 구성

웹 브라우저 기반 인터페이스를 사용하여 네트워크 설정을 관리하려면 RHEL 웹 콘솔을 사용하여 네트워크 팀을 구성합니다.

중요

네트워크 티밍은 Red Hat Enterprise Linux 9에서 더 이상 사용되지 않습니다. 서버를 RHEL의 향후 버전으로 업그레이드하려는 경우 커널 본딩 드라이버를 대안으로 사용하는 것이 좋습니다. 자세한 내용은 네트워크 본딩 구성 을 참조하십시오.

사전 요구 사항

- teamd 및 NetworkManager-team 패키지가 설치됩니다.
- 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 팀 포트로 사용하려면 다음에 설명된 대로 미리 만듭니다.
  - RHEL 웹 콘솔을 사용하여 네트워크 본딩 구성

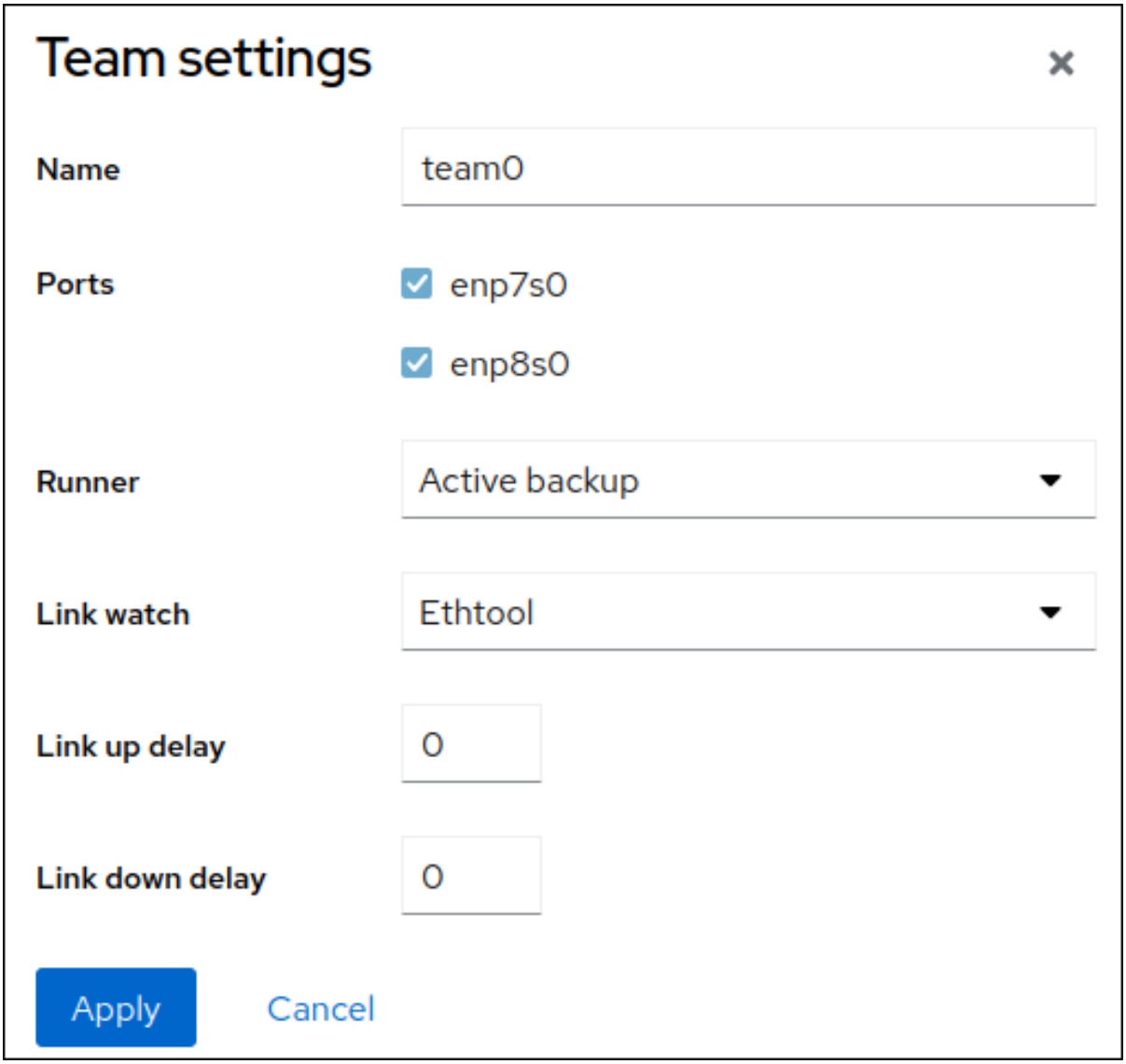
- RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성
- RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 설정

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 **runner**를 선택합니다.

   로드 밸런싱 또는 **802.3ad LACP** 를 선택하면 웹 콘솔에 추가 필드 밸런서 가 표시됩니다.
6. 링크 감시자를 설정합니다.

   - **Ethtool, additionally**를 선택하면 링크를 설정하고 지연을 연결합니다.
   - **ARP ping** 또는 **NSNA ping** 을 추가로 설정하는 경우 **ping** 간격 및 **ping** 대상을 설정합니다.



7. **Apply(적용)**를 클릭합니다.

8. 기본적으로 팀은 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

   a. **Interfaces** 섹션에서 팀 이름을 클릭합니다.

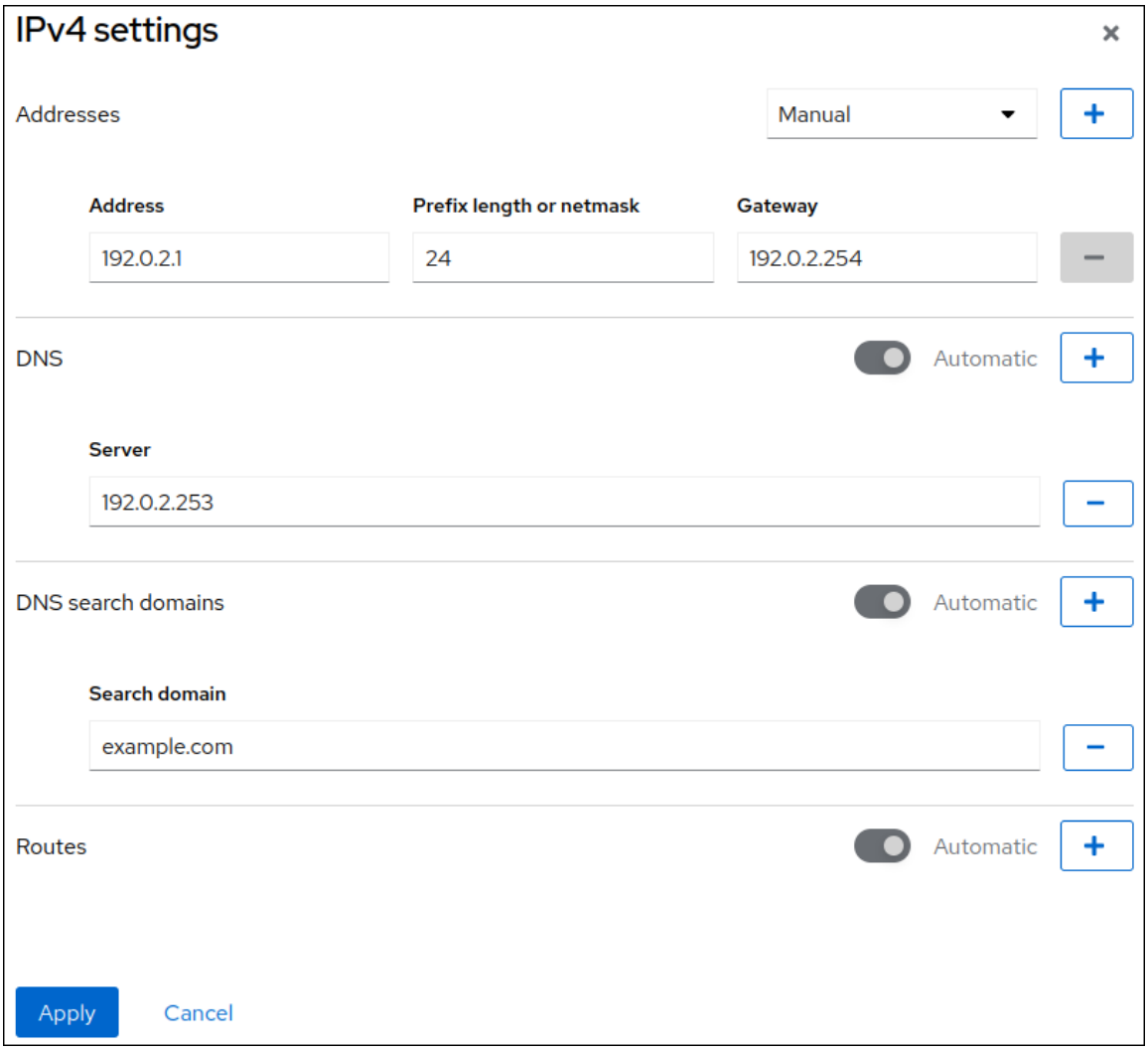
   b. 구성할 프로토콜 옆에 있는 편집 을 클릭합니다.

   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.

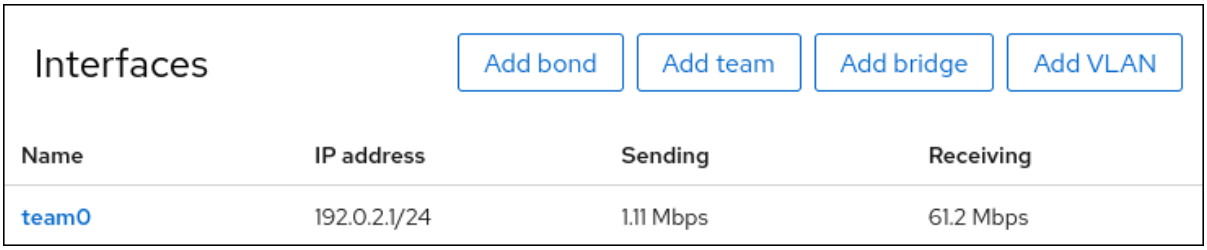
f. 인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 구성합니다.



g. 적용을클릭합니다.

검증

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



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 실행기

4.5. NM-CONNECTION-EDITOR를 사용하여 네트워크 팀 구성

그래픽 인터페이스와 함께 **Red Hat Enterprise Linux**를 사용하는 경우 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 팀을 구성할 수 있습니다.

nm-connection-editor 는 팀에 새 포트만 추가할 수 있습니다. 기존 연결 프로필을 포트로 사용하려면 **nmcli** 를 사용하여 네트워크 팀 구성에 설명된 대로 **nmcli** 유틸리티를 사용하여 팀을 생성합니다.

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.

사전 요구 사항

- teamd 및 NetworkManager-team 패키지가 설치됩니다.
- 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.

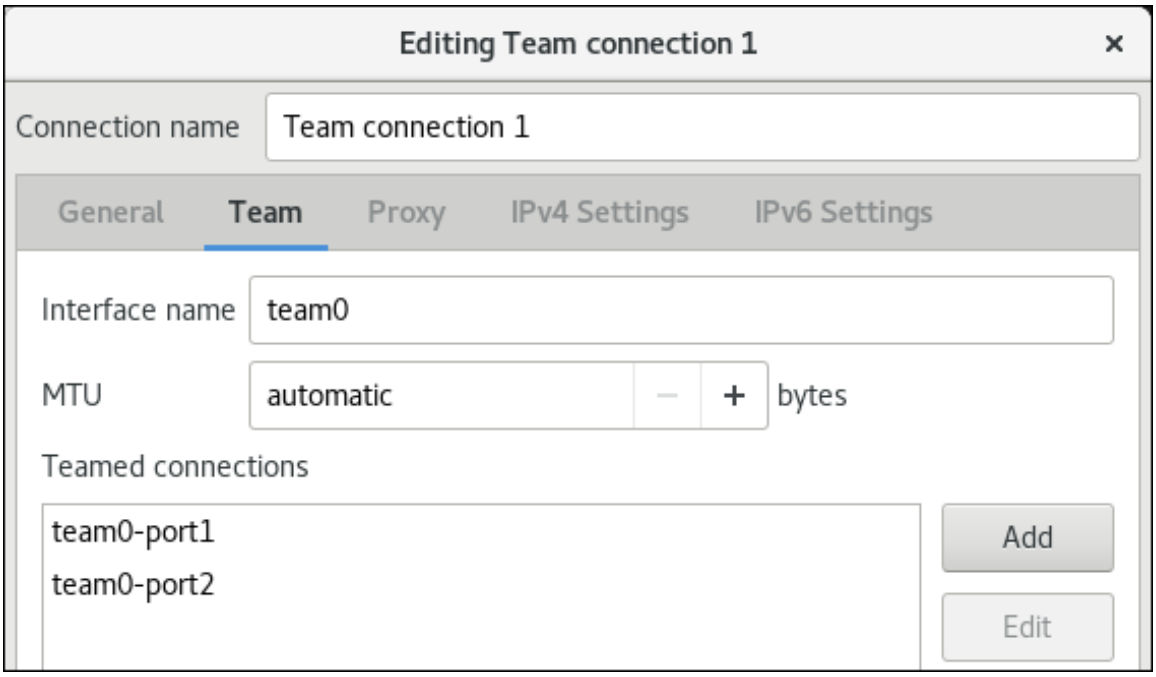
```
$ nm-connection-editor
```

2. + 버튼을 클릭하여 새 연결을 추가합니다.
3. 팀 연결 유형을 선택하고 생성을 클릭합니다.
4. 팀 탭에서 다음을 수행합니다.
   a. 선택 사항: Interface name 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 연결로 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 **link watcher** 및 해당 설정의 선택적 설정을 설정합니다.

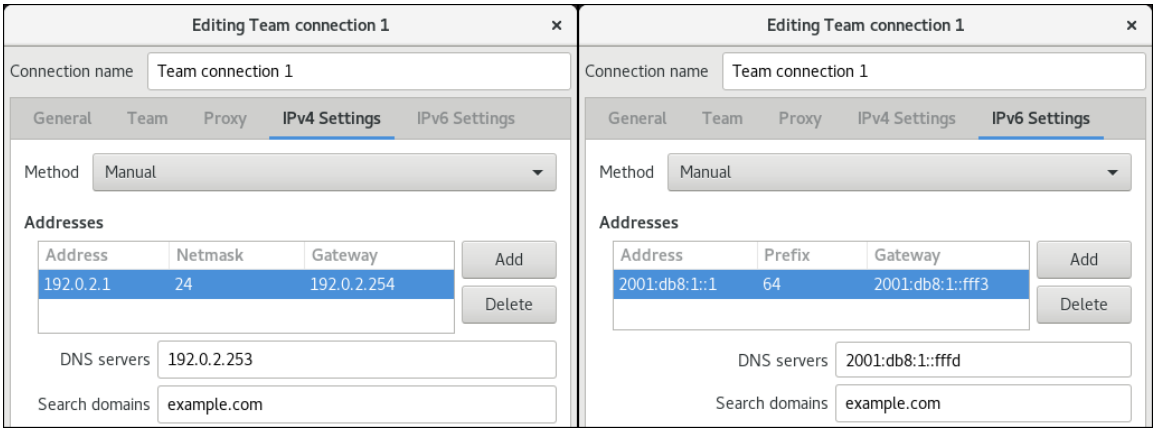
iii. **OK**(확인)를 클릭합니다.

5. **IPv4** 설정 및 **IPv6** 설정 탭에서 **IP** 주소 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 **Method** 필드를 **Disabled** 로 설정합니다.

- **DHCP**를 사용하려면 **Method** 필드를 기본값인 **Automatic(DHCP)** 으로 둡니다.

- 고정 **IP** 설정을 사용하려면 **Method** 필드를 **Manual** 로 설정하고 그에 따라 필드를 작성합니다.



6. 저장을 클릭합니다.

7. **nm-connection-editor** 를 종료합니다.

검증

- 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-editor를 사용하여 네트워크 본딩 구성
- nm-connection-editor를 사용하여 네트워크 팀 구성
- nm-connection-editor를 사용하여 VLAN 태그 지정 설정
- 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.

5장. VLAN 태그 지정 구성

VLAN(Virtual Local Area Network)은 물리적 네트워크 내의 논리적 네트워크입니다. VLAN 인터페이스는 인터페이스를 통과하면 VLAN ID로 패킷에 태그를 지정하고, 패킷 반환 태그를 제거합니다. 이더넷, 본딩, 팀 또는 브리지 장치와 같은 다른 인터페이스의 상단에 VLAN 인터페이스를 생성합니다. 이러한 인터페이스를 부모 인터페이스 라고 합니다.

Red Hat Enterprise Linux는 관리자가 VLAN 장치를 구성하는 다양한 옵션을 제공합니다. 예를 들면 다음과 같습니다.

- nmcli 를 사용하여 명령줄을 사용하여 VLAN 태그 지정을 구성합니다.
- RHEL 웹 콘솔을 사용하여 웹 브라우저를 사용하여 VLAN 태그를 구성합니다.
- nmtui 를 사용하여 텍스트 기반 사용자 인터페이스에서 VLAN 태그를 구성합니다.
- nm-connection-editor 애플리케이션을 사용하여 그래픽 인터페이스에서 연결을 구성합니다.
- nmstatectl 을 사용하여 Nmstate API를 통한 연결을 구성합니다.
- RHEL 시스템 역할을 사용하여 하나 이상의 호스트에서 VLAN 구성을 자동화합니다.

5.1. NMCLI를 사용하여 VLAN 태그 구성

nmcli 유틸리티를 사용하여 명령줄에서 VLAN(Virtual Local Area Network) 태그를 구성할 수 있습니다.

사전 요구 사항

- 가상 VLAN 인터페이스의 상위로 사용하려는 인터페이스는 VLAN 태그를 지원합니다.
- 본딩 인터페이스 상단에 VLAN을 구성하는 경우:

- 본딩의 포트가 설정되어 있습니다.

- 본딩은 **fail_over_mac=follow** 옵션으로 구성되지 않습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽이 잘못된 소스 **MAC** 주소로 전송됩니다.

- 일반적으로 본딩은 **DHCP** 서버 또는 **IPv6** 자동 구성에서 **IP** 주소를 가져오지 않습니다. 본딩을 생성하는 동안 **ipv4.method=disable** 및 **ipv6.method=ignore** 옵션을 설정하여 확인합니다. 그렇지 않으면 잠시 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** 을 사용하여 패킷 태그를 지정하는 **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. IPv4 설정을 구성합니다.

- 이 VLAN 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify vlan10 ipv4.method disabled
```

- DHCP를 사용하려면 작업이 필요하지 않습니다.

- 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' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.method manual
```

5. IPv6 설정을 구성합니다.

- 이 VLAN 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify vlan10 ipv6.method disabled
```

- SLAAC(stateless address autoconfiguration)를 사용하려면 작업이 필요하지 않습니다.

- 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' ipv6.gateway
'2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd' ipv6.method manual
```

6. 연결을 활성화합니다.

```
# nmcli connection up vlan10
```

검증

- 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

- **nm-settings(5)** 도움말 페이지

5.2. RHEL 웹 콘솔을 사용하여 VLAN 태그 지정 설정

웹 브라우저 기반 인터페이스를 사용하여 네트워크 설정을 관리하려면 **RHEL** 웹 콘솔을 사용하여 **VLAN** 태그를 구성합니다.

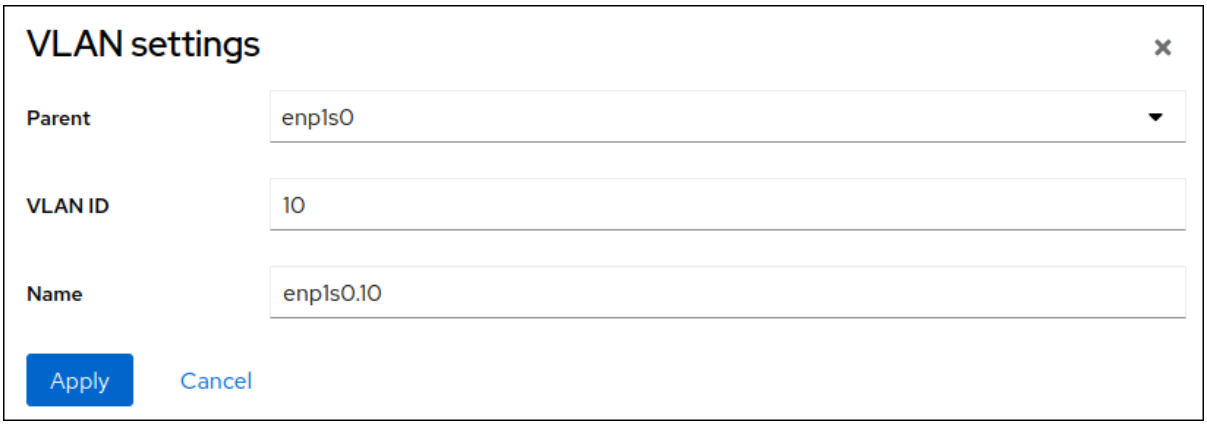
사전 요구 사항

- 가상 **VLAN** 인터페이스의 상위로 사용하려는 인터페이스는 **VLAN** 태그를 지원합니다.
- 본딩 인터페이스 상단에 **VLAN**을 구성하는 경우:
  - 본딩의 포트가 설정되어 있습니다.
  - 본딩은 **fail_over_mac=follow** 옵션으로 구성되지 않습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽이 잘못된 소스 **MAC** 주소로 전송됩니다.
  - 일반적으로 본딩은 **DHCP** 서버 또는 **IPv6** 자동 구성에서 **IP** 주소를 가져오지 않습니다. **IPv4** 및 **IPv6** 프로토콜을 비활성화하여 본딩을 만듭니다. 그렇지 않으면 잠시 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. 인터페이스 섹션에서 **VLAN** 추가를 클릭합니다.
3. 상위 장치를 선택합니다.
4. **VLAN ID**를 입력합니다.
5. **VLAN** 장치의 이름을 입력하거나 자동으로 생성된 이름을 유지합니다.



6. **Apply**(적용)를 클릭합니다.
7. 기본적으로 **VLAN** 장치는 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

   a. 인터페이스 섹션에서 **VLAN** 장치의 이름을 클릭합니다.

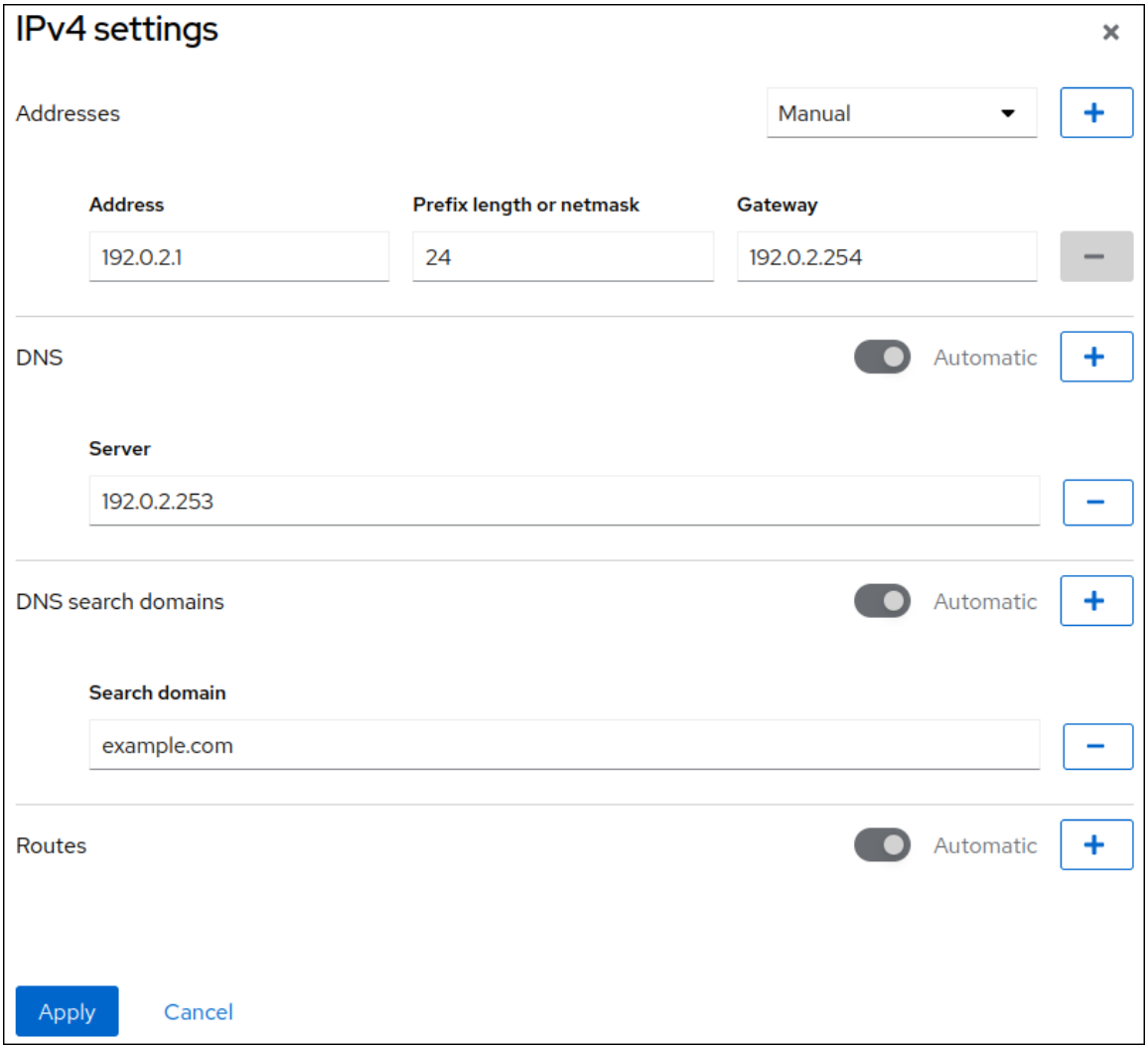
   b. 구성할 프로토콜 옆에 있는 편집 을 클릭합니다.

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.

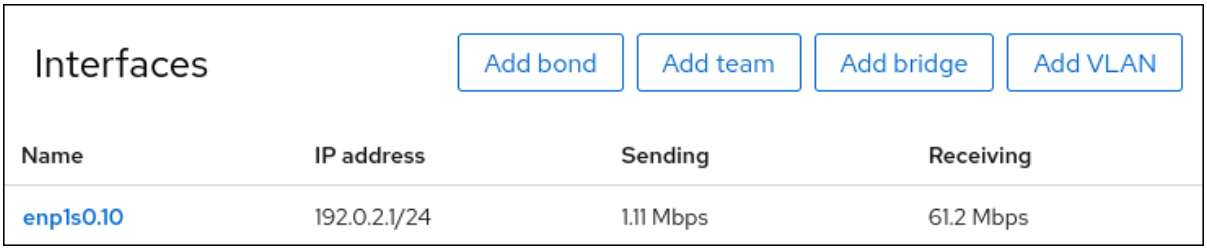
f. 인터페이스에 정적 경로가 필요한 경우 **Routes** 섹션에서 구성합니다.



g. 적용을클릭합니다.

검증

- 화면 왼쪽의 탐색에서 **Networking** 탭을 선택하고 인터페이스에서 수신 및 발신 트래픽이 있는지 확인합니다.



5.3. NMTUI를 사용하여 VLAN 태그 구성

nmtui 애플리케이션은 **NetworkManager**에 대한 텍스트 기반 사용자 인터페이스를 제공합니다. **nmtui** 를 사용하여 그래픽 인터페이스 없이 호스트에서 **VLAN** 태그를 구성할 수 있습니다.

참고

nmtui 에서 :

- 커서 키를 사용하여 이동합니다.
- 버튼을 선택하고 **Enter** 를 누릅니다.
- **Space** 를 사용하여 확인란을 선택하고 지웁니다.

사전 요구 사항

- 가상 **VLAN** 인터페이스의 상위로 사용하려는 인터페이스는 **VLAN** 태그를 지원합니다.
- 본딩 인터페이스 상단에 **VLAN**을 구성하는 경우:
  - 본딩의 포트가 설정되어 있습니다.
  - 본딩은 **fail_over_mac=follow** 옵션으로 구성되지 않습니다. **VLAN** 가상 장치는 **MAC** 주소를 상위의 새 **MAC** 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽은 여전히 잘못된 소스 **MAC** 주소와 함께 전송됩니다.
  - 일반적으로 본딩은 **DHCP** 서버 또는 **IPv6** 자동 구성에서 **IP** 주소를 가져오지 않습니다. 본딩을 생성하는 동안 **ipv4.method=disable** 및 **ipv6.method=ignore** 옵션을 설정하여 확인

합니다. 그렇지 않으면 잠시 후에 **DHCP** 또는 **IPv6** 자동 구성이 실패하면 인터페이스가 중단될 수 있습니다.

- 호스트가 연결된 스위치는 **VLAN** 태그를 지원하도록 구성되어 있습니다. 자세한 내용은 스위치 설명서를 참조하십시오.

절차

1. **VLAN** 태그를 구성할 네트워크 장치 이름을 모르는 경우 사용 가능한 장치를 표시합니다.

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

2. **start nmtui:**

```
# nmtui
```

3. **Edit a connection** 을 선택하고 **Enter** 를 누릅니다.

4. 추가를 누릅니다.

5. 네트워크 유형 목록에서 **VLAN** 을 선택하고 **Enter** 키를 누릅니다.

6. 선택 사항: 생성할 **NetworkManager** 프로필의 이름을 입력합니다.

   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

7. 장치 필드에 생성할 **VLAN** 장치 이름을 입력합니다.

8. **VLAN** 태그를 부모 필드에 구성할 장치의 이름을 입력합니다.

9. **VLAN ID**를 입력합니다. **ID**는 **0** 에서 **4094** 사이의 범위 내에 있어야 합니다.

10. 환경에 따라 그에 따라 **IPv4** 구성 및 **IPv6** 구성 영역에서 **IP** 주소 설정을 구성합니다. 이를 위해 다음 영역 옆에 있는 버튼을 누른 후 다음을 선택합니다.

- 비활성화됨 (이 **VLAN** 장치에 **IP** 주소가 필요하지 않거나 다른 장치의 포트로 사용하려는 경우).
- **DHCP** 서버 또는 **SLAAC**(상태 비저장 주소 자동 구성)가 **VLAN** 장치에 **IP** 주소를 동적으로 할당하는 경우 자동입니다.
- 네트워크에 고정 **IP** 주소 설정이 필요한 경우 수동. 이 경우 추가 필드를 작성해야 합니다.

  i. 추가 필드를 표시하도록 구성할 프로토콜 옆에 **Show** 를 누릅니다.

  ii. 주소 옆에 있는 추가 를 클릭하고 **CIDR(Classless Inter-Domain Routing)** 형식으로 **IP** 주소와 서브넷 마스크를 입력합니다.

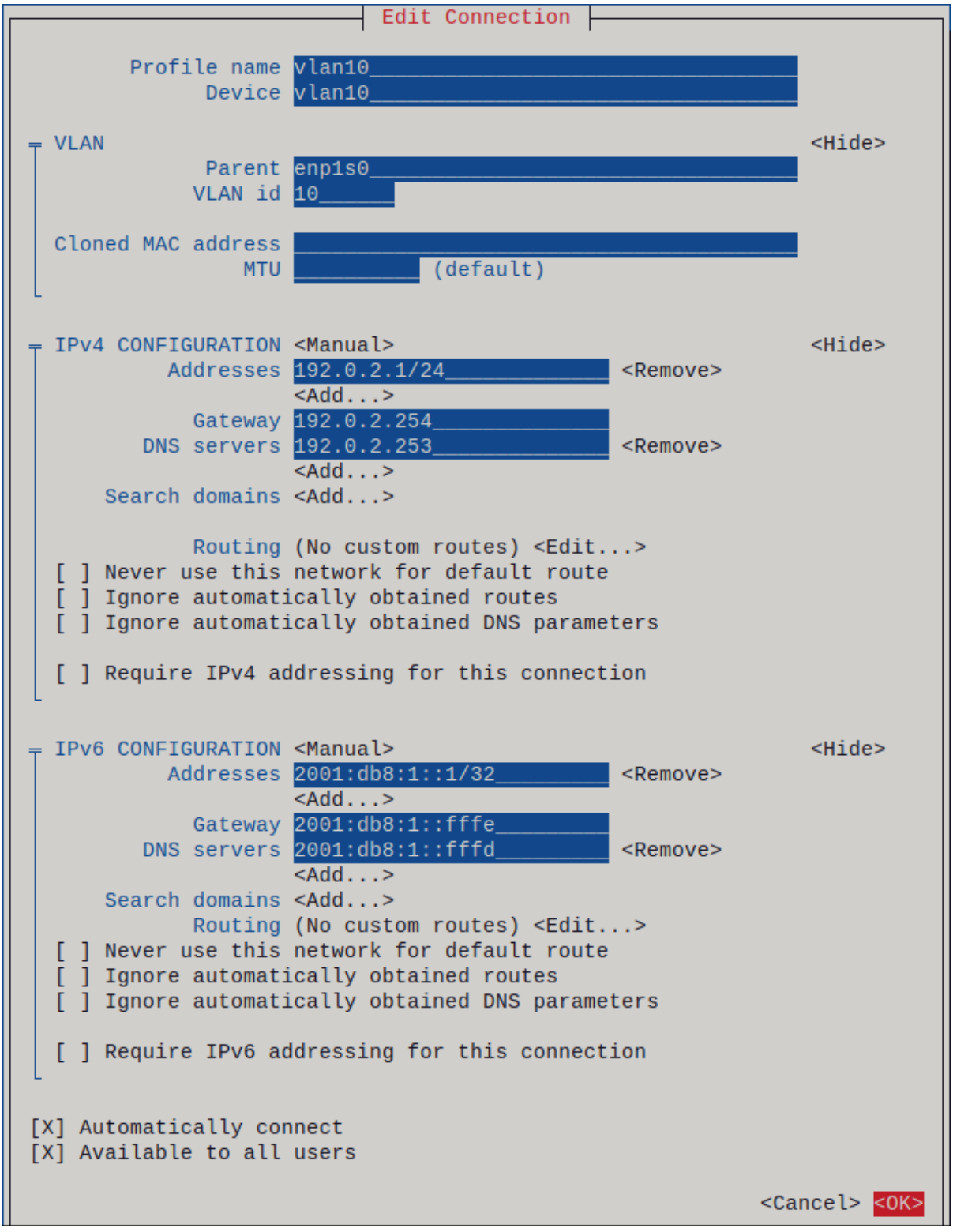
  서브넷 마스크를 지정하지 않으면 **NetworkManager**는 **IPv4** 주소에 대해 **/32** 서브넷 마스크와 **IPv6** 주소에 대해 **/64** 를 설정합니다.

  iii. 기본 게이트웨이의 주소를 입력합니다.

  iv. **DNS** 서버 옆에 있는 추가 를 클릭하고 **DNS** 서버 주소를 입력합니다.

  v. 검색 도메인 옆에 있는 추가 를 클릭하고 **DNS** 검색 도메인을 입력합니다.

그림 5.1. 고정 IP 주소 설정을 사용한 VLAN 연결 예



11. **OK** 를 눌러 새 연결을 만들고 자동으로 활성화합니다.

12. 다시 키를 눌러 기본 메뉴로 돌아갑니다.

13. **Quit** 를 선택하고 **Enter** 를 눌러 **nmtui** 애플리케이션을 종료합니다.

검증

- 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
```

5.4. NM-CONNECTION-EDITOR를 사용하여 VLAN 태그 지정 설정

nm-connection-editor 애플리케이션을 사용하여 그래픽 인터페이스에서 VLAN(Virtual Local Area Network) 태그를 구성할 수 있습니다.

사전 요구 사항

- 가상 VLAN 인터페이스의 상위로 사용하려는 인터페이스는 VLAN 태그를 지원합니다.
- 본딩 인터페이스 상단에 VLAN을 구성하는 경우:
  - 본딩의 포트가 설정되어 있습니다.
  - 본딩은 fail_over_mac=follow 옵션으로 구성되지 않습니다. VLAN 가상 장치는 MAC 주소를 상위의 새 MAC 주소와 일치하도록 변경할 수 없습니다. 이러한 경우 트래픽이 잘못된 소스 MAC 주소로 전송됩니다.
- 호스트가 연결되어 VLAN 태그를 지원하도록 구성된 스위치입니다. 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.

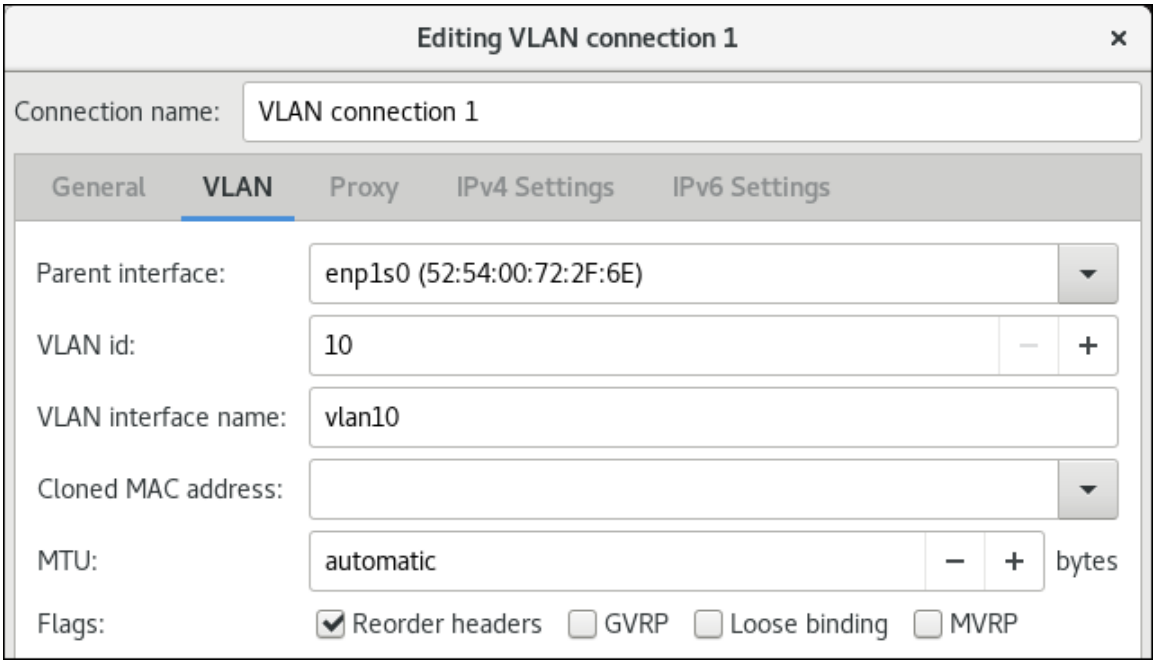
4. **VLAN** 탭에서 다음을 수행합니다.

   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.

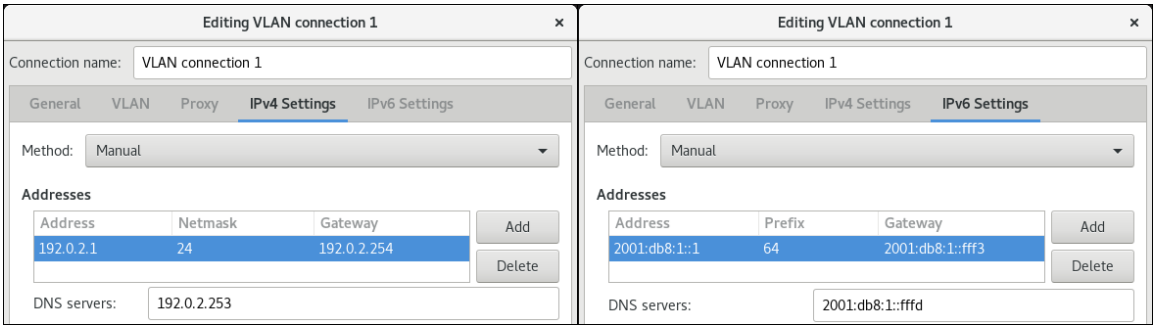
   d. 선택적으로 **VLAN** 인터페이스의 이름과 추가 **VLAN** 관련 옵션을 설정합니다.



5.

IPv4 설정 및 **IPv6 설정** 탭에서 **IP** 주소 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 **Method** 필드를 **Disabled** 로 설정합니다.
- **DHCP**를 사용하려면 **Method** 필드를 기본값인 **Automatic(DHCP)** 으로 둡니다.
- 고정 **IP** 설정을 사용하려면 **Method** 필드를 **Manual** 로 설정하고 그에 따라 필드를 작성합니다.



6.

저장을 클릭합니다.

7.

nm-connection-editor 를 종료합니다.

검증

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 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성

5.5. NMSTATECTL을 사용하여 VLAN 태그 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 Virtual Local Area Network VLAN을 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 nmstatectl 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

환경에 따라 YAML 파일을 적절하게 조정합니다. 예를 들어 VLAN에서 이더넷 어댑터와 다른 장치를 사용하려면 VLAN에서 사용하는 포트의 base-iface 특성 및 유형 속성을 조정합니다.

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 만듭니다.

```
---
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
```

이러한 설정은 **enp1s0** 장치를 사용하는 **ID 10**이 있는 **VLAN**을 정의합니다. 하위 장치로 **VLAN** 연결에는 다음과 같은 설정이 있습니다.

- /24 서브넷 마스크가 있는 정적 **IPv4** 주소 **- 192.0.2.1**
- 정적 **IPv6** 주소 **- 2001:db8:1::1 (/64** 서브넷 마스크 포함**)**
- **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

2. 시스템에 설정을 적용합니다.

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

검증

1. 장치 및 연결 상태를 표시합니다.

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

2. 연결 프로필의 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) man page
- /usr/share/doc/nmstate/examples/ 디렉터리

5.6. 네트워크 RHEL 시스템 역할을 사용하여 VLAN 태그 구성

네트워크 RHEL 시스템 역할을 사용하여 VLAN 태그를 구성할 수 있습니다. 이 예제에서는 이더넷 연결과 이 이더넷 연결 위에 ID 10 이 있는 VLAN을 추가합니다. 하위 장치로 VLAN 연결에는 IP, 기본 게이

트웨이 및 DNS 구성이 포함됩니다.

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

- 본딩과 같은 다른 연결의 포트로 VLAN을 사용하려면 ip 특성을 생략하고 하위 구성의 IP 구성을 설정합니다.
- VLAN에서 팀, 브리지 또는 본딩 장치를 사용하려면 VLAN에서 사용하는 포트의 interface_name 및 type 속성을 조정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a VLAN that uses an Ethernet connection
      ansible.builtin.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
```

이러한 설정은 **enp1s0** 장치 상단에서 작동할 **VLAN**을 정의합니다. **VLAN** 인터페이스에는 다음과 같은 설정이 있습니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
- VLAN ID - 10

VLAN 프로필의 상위 속성은 **VLAN**이 **enp1s0** 장치 상단에서 작동하도록 구성합니다. 하위 장치로 **VLAN** 연결에는 IP, 기본 게이트웨이 및 DNS 구성이 포함됩니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

6장. 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-Fi에서 3 주소 프레임 사용을 지정하는 IEEE 802.11 표준으로 인해 Ad-Hoc 또는 Infrastructure 모드에서 작동하는 Wi-Fi 네트워크를 통한 브리지를 구성할 수 없습니다.

6.1. NMCLI를 사용하여 네트워크 브리지 구성

명령줄에서 네트워크 브리지를 구성하려면 nmcli 유틸리티를 사용합니다.

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 에 설명된 대로 미리 생성할 수 있습니다.
  - nmcli를 사용하여 네트워크 팀 구성
  - nmcli를 사용하여 네트워크 본딩 구성
  - nmcli를 사용하여 VLAN 태그 지정 설정

절차

1. 브리지 인터페이스를 만듭니다.

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

이 명령은 bridge0 이라는 브릿지를 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
```

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

b. 기존 연결 프로필을 브리지에 할당하려면 다음을 수행합니다.

i. 이러한 연결의 **master** 매개변수를 **bridge0** 으로 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.

ii. 연결을 다시 활성화합니다.

```
# nmcli connection up bond0
# nmcli connection up bond1
```

4. **IPv4** 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify bridge0 ipv4.method disabled
```

- **DHCP**를 사용하려면 작업이 필요하지 않습니다.
-

정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **bridge0** 연결로 설정하려면 다음을 입력합니다.

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

5. **IPv6** 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 다음을 입력합니다.

```
# nmcli connection modify bridge0 ipv6.method disabled
```

- **SLAAC(stateless address autoconfiguration)**를 사용하려면 작업이 필요하지 않습니다.

- 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 **bridge0** 연결로 설정하려면 다음을 입력합니다.

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

6. 선택 사항: 브리지의 추가 속성을 구성합니다. 예를 들어 **bridge0** 의 **Spanning Tree Protocol(STP)** 우선순위를 **16384** 로 설정하려면 다음을 입력합니다.

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

기본적으로 **STP**가 활성화됩니다.

7. 연결을 활성화합니다.

```
# nmcli connection up bridge0
```

8. 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**는 브리지를 활성화하지만 다른 포트는 활성화하지 않습니다. 브리지가 활성화되면 **Red Hat Enterprise Linux**가 모든 포트를 자동으로 사용하도록 설정할 수 있습니다.

a. **bridge** 연결의 **connection.autoconnect-slaves** 매개변수를 활성화합니다.

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

b. 브리지를 다시 활성화합니다.

```
# nmcli connection up bridge0
```

검증

- **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
```

- **bridge** 유틸리티를 사용하여 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>*** **명**

령을 사용합니다.

추가 리소스

- **nm-settings(5)** 도움말 페이지
- **bridge(8)** 도움말 페이지
- NetworkManager 서비스를 다시 시작한 후 NetworkManager에서 연결을 중복합니다.
- VLAN 정보로 브리지를 구성하는 방법은 무엇입니까?

6.2. RHEL 웹 콘솔을 사용하여 네트워크 브리지 구성

웹 브라우저 기반 인터페이스를 사용하여 네트워크 설정을 관리하려면 **RHEL** 웹 콘솔을 사용하여 네트워크 브리지를 구성합니다.

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 에 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 설정

절차

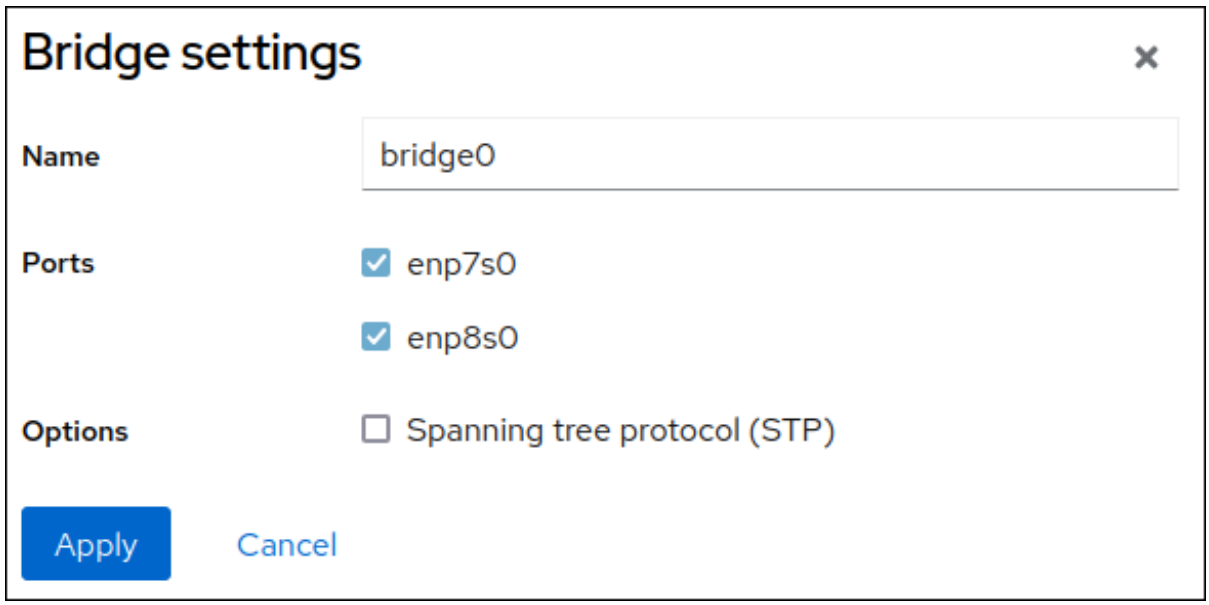
1. 화면 왼쪽의 탐색에서 네트워킹 탭을 선택합니다.

2. 인터페이스 섹션에서 **Add bridge** 를 클릭합니다.

3. 생성할 브리지 장치의 이름을 입력합니다.

4. 브리지의 포트여야 하는 인터페이스를 선택합니다.

5. 선택 사항: **STP(Sspanning tree Protocol)** 기능을 활성화하여 브리지 루프 및 브로드캐스트 복사를 방지할 수 있습니다.



6. **Apply**(적용)를 클릭합니다.

7. 기본적으로 브릿지는 동적 **IP** 주소를 사용합니다. 고정 **IP** 주소를 설정하려면 다음을 수행합니다.

   a. **Interfaces** 섹션에서 브리지 이름을 클릭합니다.

   b. 구성할 프로토콜 옆에 있는 편집을 클릭합니다.

c.

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

d.

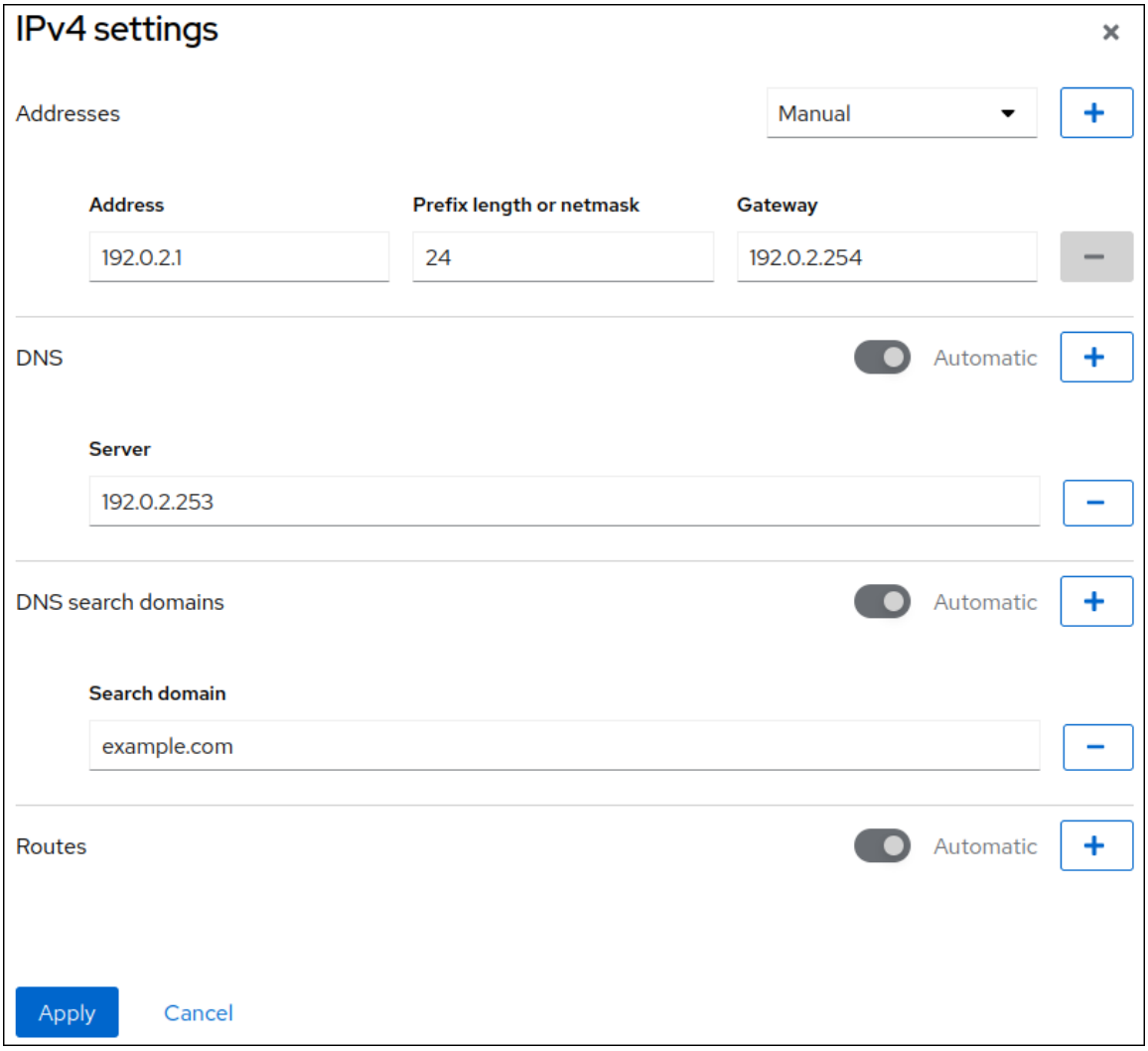
DNS 섹션에서 **+** 버튼을 클릭하고 **DNS** 서버의 **IP** 주소를 입력합니다. 이 단계를 반복하여 여러 **DNS** 서버를 설정합니다.

e.

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

f.

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



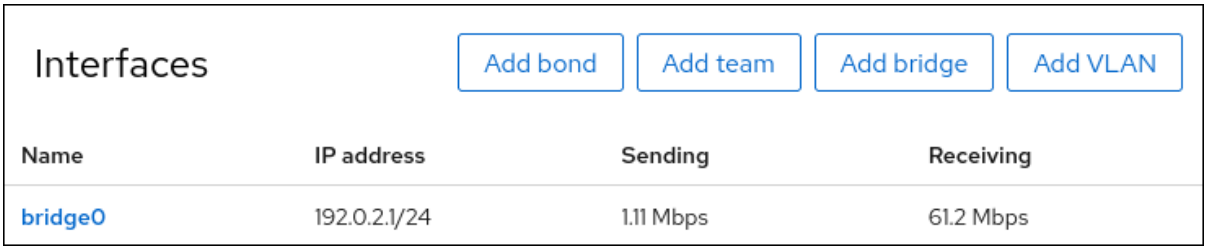
g.

적용을클릭합니다.

검증

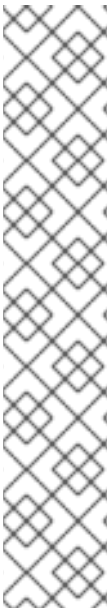
1.

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



6.3. NMTUI를 사용하여 네트워크 브리지 구성

nmtui 애플리케이션은 **NetworkManager**에 대한 텍스트 기반 사용자 인터페이스를 제공합니다. **nmtui** 를 사용하여 그래픽 인터페이스 없이 호스트에서 네트워크 브릿지를 구성할 수 있습니다.



참고

nmtui 에서 :

- 커서 키를 사용하여 이동합니다.
- 버튼을 선택하고 **Enter** 를 누릅니다.
- **Space** 를 사용하여 확인란을 선택하고 지웁니다.

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.

절차

1. 네트워크 브리지를 구성할 네트워크 장치 이름을 모르는 경우 사용 가능한 장치를 표시합니다.

```
# nmcli device status
DEVICE    TYPE      STATE              CONNECTION
enp7s0    ethernet  unavailable         --
```

```
enp8s0    ethernet  unavailable          --
...
```

2. start nmtui:

```
# nmtui
```

3. Edit a connection 을 선택하고 Enter 를 누릅니다.

4. 추가를 누릅니다.

5. 네트워크 유형 목록에서 Bridge 를 선택하고 Enter 를 누릅니다.

6. 선택 사항: 생성할 NetworkManager 프로필의 이름을 입력합니다.

   프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

7. 생성할 브리지 장치 이름을 장치 필드에 입력합니다.

8. 생성할 브리지에 포트를 추가합니다.

   a. Slaves 목록 옆에 있는 Add 를 누릅니다.

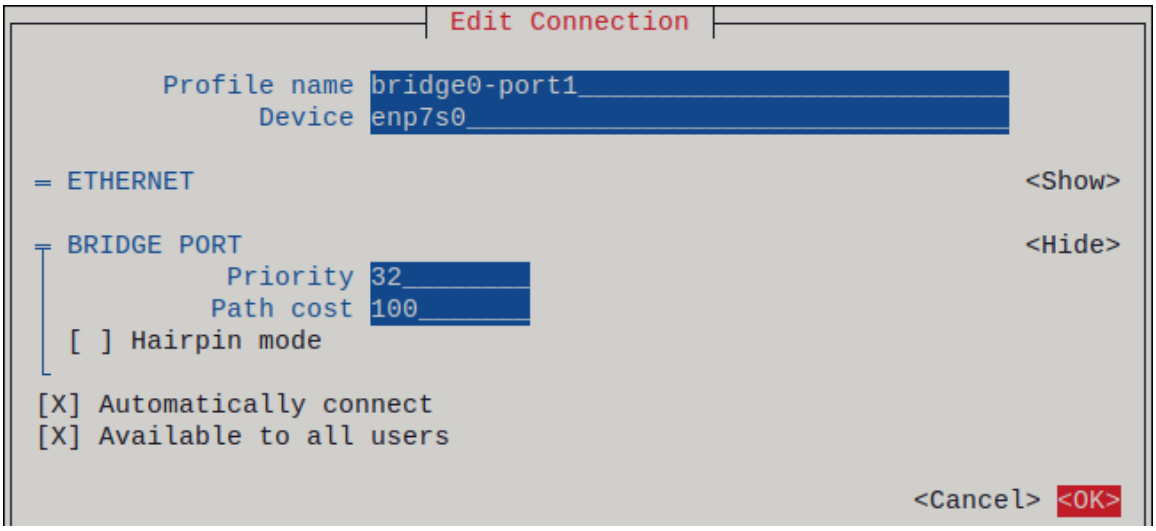
   b. 브리지에 포트로 추가할 인터페이스 유형(예: 이더넷)을 선택합니다.

   c. 선택 사항: 이 브리지 포트에 대해 생성할 NetworkManager 프로필의 이름을 입력합니다.

   d. 포트의 장치 이름을 장치 필드에 입력합니다.

e. **OK** 를 눌러 브리지 설정을 사용하여 창으로 돌아갑니다.

그림 6.1. 이더넷 장치를 브리지에 포트로 추가



f. 이 단계를 반복하여 브리지에 더 많은 포트를 추가합니다.

9. 환경에 따라 **IPv4** 구성 및 **IPv6** 구성 영역에서 **IP** 주소 설정을 적절하게 구성합니다. 이를 위해 다음 영역 옆에 있는 버튼을 누른 후 다음을 선택합니다.

- 브리지에 **IP** 주소가 필요하지 않은 경우 비활성화됨
- **DHCP** 서버 또는 **SLAAC**(상태 비저장 주소 자동 구성)가 브리지에 **IP** 주소를 동적으로 할당하는 경우 자동 입니다.
- 네트워크에 고정 **IP** 주소 설정이 필요한 경우 수동. 이 경우 추가 필드를 작성해야 합니다.

i. 추가 필드를 표시하도록 구성할 프로토콜 옆에 **Show** 를 누릅니다.

ii. 주소 옆에 있는 추가 를 클릭하고 **CIDR(Classless Inter-Domain Routing)** 형식으로 **IP** 주소와 서브넷 마스크를 입력합니다.

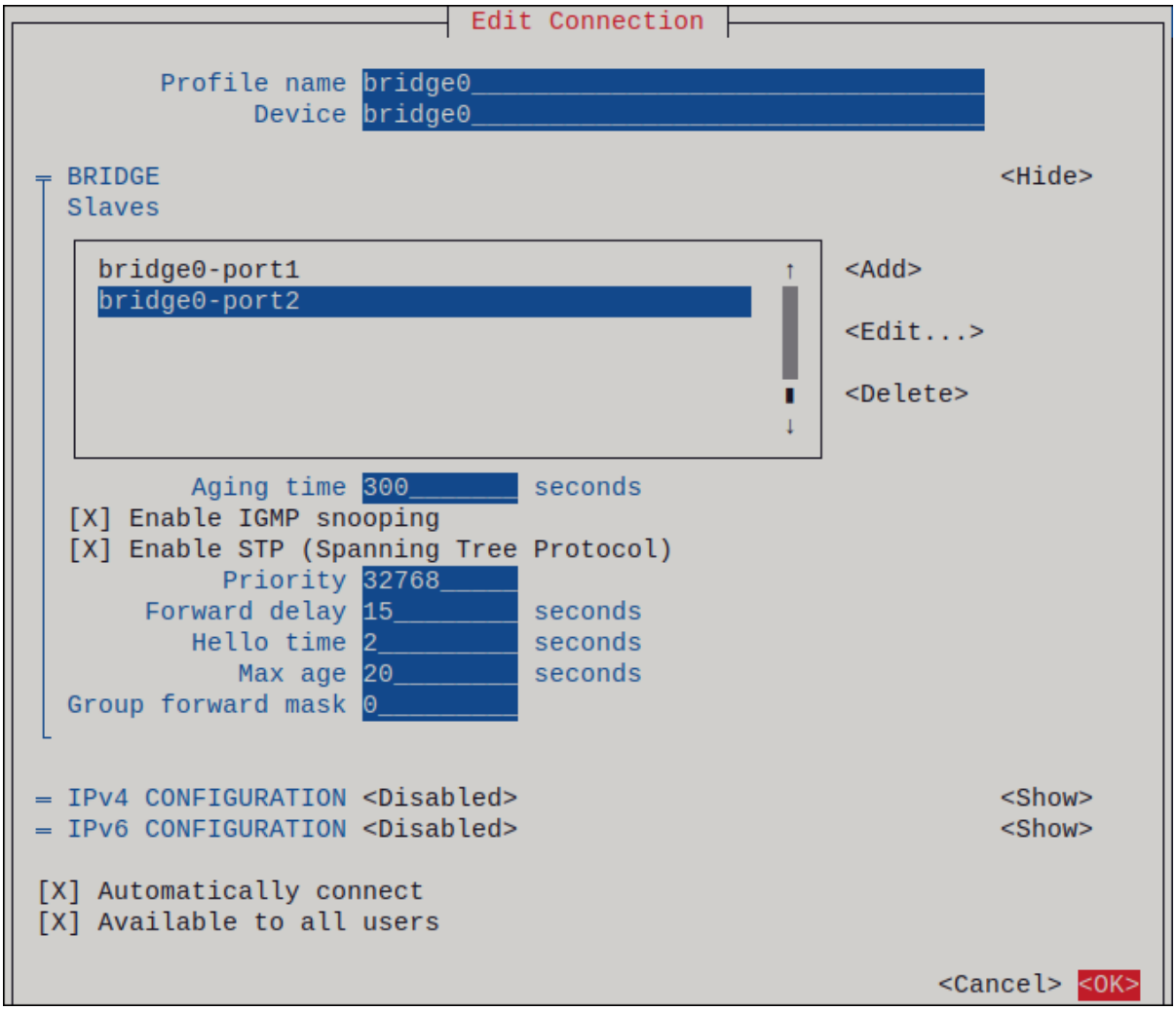
서브넷 마스크를 지정하지 않으면 **NetworkManager**는 **IPv4** 주소에 대해 **/32** 서브넷 마스크와 **IPv6** 주소에 대해 **/64** 를 설정합니다.

iii. 기본 게이트웨이의 주소를 입력합니다.

iv. **DNS** 서버 옆에 있는 추가 를 클릭하고 **DNS** 서버 주소를 입력합니다.

v. 검색 도메인 옆에 있는 추가 를 클릭하고 **DNS** 검색 도메인을 입력합니다.

그림 6.2. IP 주소 설정이 없는 브리지 연결 예



10. **OK** 를 눌러 새 연결을 만들고 자동으로 활성화합니다.

11. 다시 키를 눌러 기본 메뉴로 돌아갑니다.

12. **Quit** 을 선택하고 **Enter** 를 눌러 **nmtui** 애플리케이션을 종료합니다.

검증

1. 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
```

2. bridge 유틸리티를 사용하여 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
...
```

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

6.4. NM-CONNECTION-EDITOR를 사용하여 네트워크 브리지 구성

그래픽 인터페이스와 함께 **Red Hat Enterprise Linux**를 사용하는 경우 **nm-connection-editor** 애플리케이션을 사용하여 네트워크 브리지를 구성할 수 있습니다.

nm-connection-editor 는 새 포트만 브리지에 추가할 수 있습니다. 기존 연결 프로필을 포트로 사용하려면 **nmcli** 를 사용하여 네트워크 브리지 구성에 설명된 대로 **nmcli** 유틸리티를 사용하여 브리지 를 생성합니다.

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 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 포트로 추가합니다.**

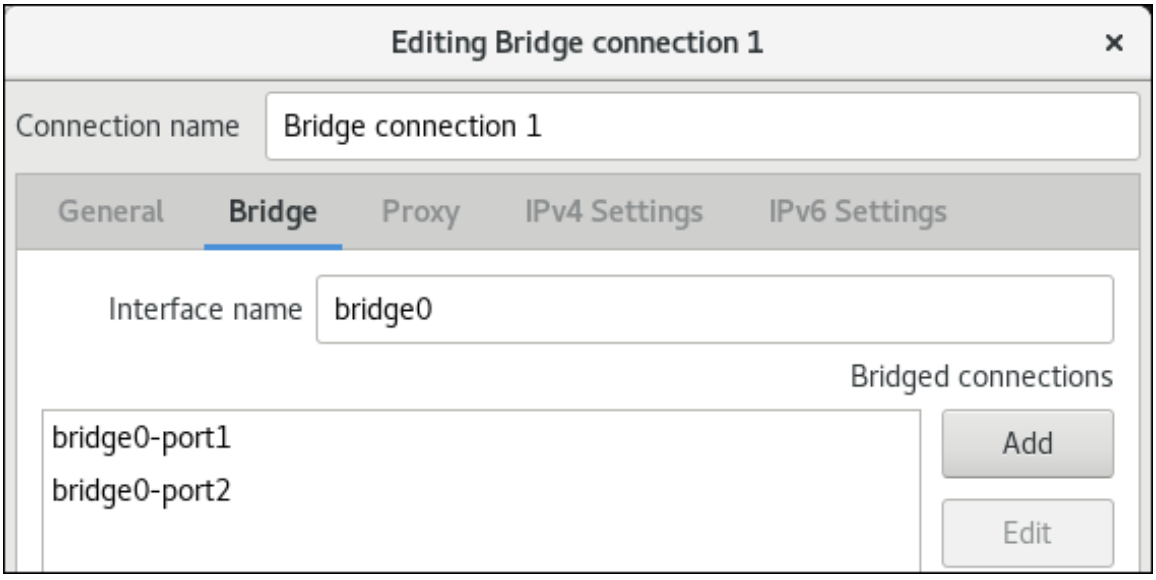
      i. **인터페이스의 연결 유형을 선택합니다. 예를 들어 유선 연결로 이더넷 을 선택합니다.**

      ii. **선택적으로 포트 장치의 연결 이름을 설정합니다.**

      iii. **이더넷 장치에 대한 연결 프로필을 생성하는 경우 이더넷 탭을 열고 장치 필드에서 브릿지에 포트로 추가할 네트워크 인터페이스를 선택합니다. 다른 장치 유형을 선택한 경우 적절하게 구성합니다.**

      iv. **저장을 클릭합니다.**

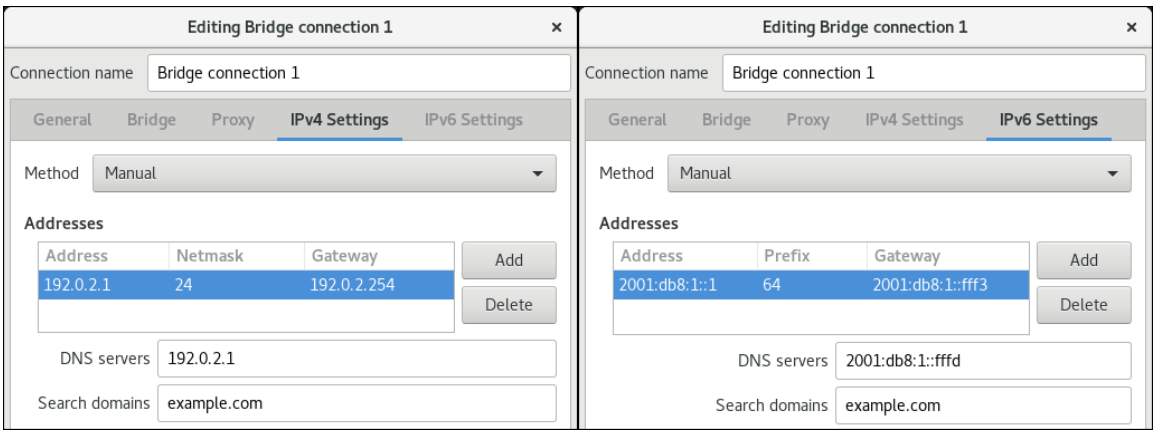
   c. **브리지에 추가할 각 인터페이스에 대해 이전 단계를 반복합니다.**



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

6. **IPv4** 설정 및 **IPv6** 설정 탭에서 **IP** 주소 설정을 구성합니다.

- 이 브리지 장치를 다른 장치의 포트로 사용하려면 **Method** 필드를 **Disabled** 로 설정합니다.
- **DHCP**를 사용하려면 **Method** 필드를 기본값인 **Automatic(DHCP)** 으로 둡니다.
- 고정 **IP** 설정을 사용하려면 **Method** 필드를 **Manual** 로 설정하고 그에 따라 필드를 작성합니다.



7. 저장을 클릭합니다.

8. nm-connection-editor 를 종료합니다.

검증

- 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
```

- bridge 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 지정 설정

- 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NetworkManager 구성
- VLAN 정보로 브리지를 구성하는 방법은 무엇입니까?

6.5. NMSTATECTL을 사용하여 네트워크 브리지 구성

nmstatectl 유틸리티를 사용하여 **Nmstate API**를 통해 네트워크 브리지를 구성합니다. **Nmstate API**는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 **nmstatectl** 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

환경에 따라 **YAML** 파일을 적절하게 조정합니다. 예를 들어 브리지에서 이더넷 어댑터와 다른 장치를 사용하려면 브릿지에서 사용하는 포트의 **base-iface** 특성 및 유형 속성을 조정합니다.

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
```

이러한 설정은 다음 설정을 사용하여 네트워크 브리지를 정의합니다.

- 브리지의 네트워크 인터페이스 **enp1s0** 및 **enp7s0**
- **STP(Spanning Tree Protocol)**: 활성화됨
- 정적 **IPv4** 주소: **192.0.2.1** 및 **/24** 서브넷 마스크

- 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

2. 시스템에 설정을 적용합니다.

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

검증

1. 장치 및 연결 상태를 표시합니다.

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

2. 연결 프로필의 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) man page
- /usr/share/doc/nmstate/examples/ 디렉터리
- VLAN 정보로 브리지를 구성하는 방법은 무엇입니까?

6.6. 네트워크 RHEL 시스템 역할을 사용하여 네트워크 브릿지 구성

네트워크 RHEL 시스템 역할을 사용하여 네트워크 브릿지를 원격으로 구성할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.
- 서버에 두 개 이상의 실제 또는 가상 네트워크 장치가 설치되어 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a network bridge that uses two Ethernet ports
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
```

```
network_connections:
 # Define the bridge profile
 - name: bridge0
   type: bridge
   interface_name: bridge0
   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

 # Add an Ethernet profile to the bridge
 - name: bridge0-port1
   interface_name: enp7s0
   type: ethernet
   controller: bridge0
   port_type: bridge
   state: up

 # Add a second Ethernet profile to the bridge
 - name: bridge0-port2
   interface_name: enp8s0
   type: ethernet
   controller: bridge0
   port_type: bridge
   state: up
```

이러한 설정은 다음 설정을 사용하여 네트워크 브리지를 정의합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
- 브리지 포트 - enp7s0 및 enp8s0

참고

Linux 브리지의 포트가 아닌 브리지에서 IP 구성을 설정합니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

7장. IPSEC VPN 설정

VPN(가상 사설 네트워크)은 인터넷을 통해 로컬 네트워크에 연결하는 방법입니다. Libreswan 에서 제공하는 IPsec 은 VPN을 생성하는 기본 방법입니다. Libreswan 은 VPN을 위한 사용자 공간 IPsec 구현입니다. VPN은 인터넷과 같은 중간 네트워크에서 터널을 설정하여 LAN과 다른 LAN 간의 통신을 활성화합니다. 보안상의 이유로 VPN 터널은 항상 인증 및 암호화를 사용합니다. 암호화 작업의 경우 Libreswan 은 NSS 라이브러리를 사용합니다.

7.1. CONTROL-CENTER를 사용한 VPN 연결 구성

그래픽 인터페이스와 함께 Red Hat Enterprise Linux를 사용하는 경우 GNOME 제어 센터에서 VPN 연결을 구성할 수 있습니다.

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. 왼쪽의 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.

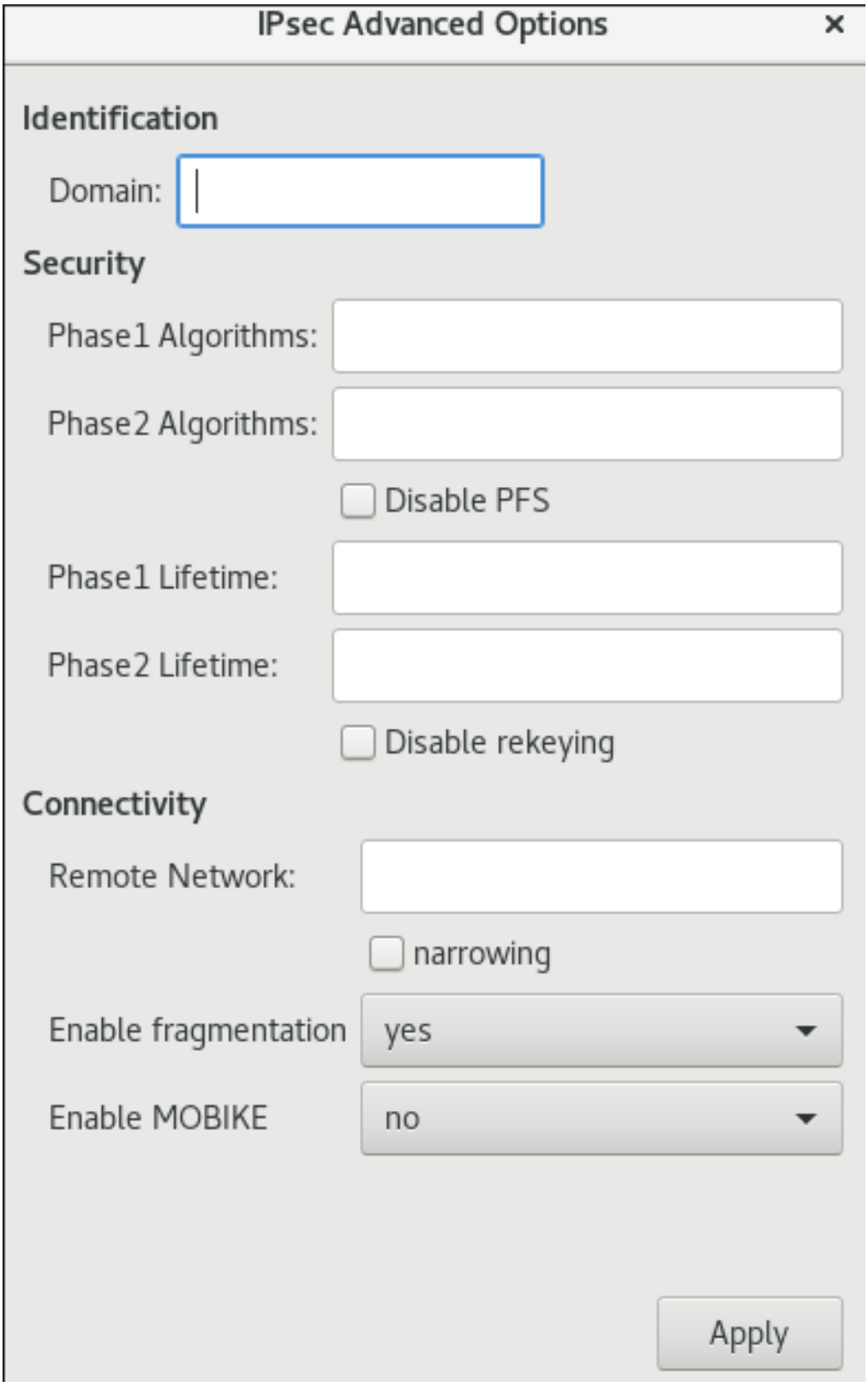
인증

유형

- **IKEv2(인증서)- 클라이언트는 인증서를 통해 인증됩니다. 더 안전합니다(기본값).**
- **IKEv1(XAUTH) - 클라이언트는 사용자 이름과 암호 또는 사전 공유 키()로 인증됩니다.**

고급 섹션에서 다음 설정 설정을 사용할 수 있습니다.

그림 7.1. VPN 연결의 고급 옵션



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.

- 단계**2** 알고리즘 **- esp Libreswan** 매개변수에 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** 매개변수에 해당합니다. **-** 트래픽을 암호화하는 데 사용되는 키가 유효한지 확인합니다.

- **2** 단계 라이프 사이클 **- salifetime Libreswan** 매개변수에 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** 매개변수에 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) - 연결 중인 네트워크에서 DHCP 서버를 사용하여 동적 IP 주소를 할당하는 경우 이 옵션을 사용합니다.**

- **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 IP를 쉼표로 구분하는 데 사용할 DNS 서버의 IP 주소를 입력합니다.

라우트

Routes(경로) 섹션에서 Automatic(자동) 이 ON 이면 DHCP의 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 알림(**RA**)을 기반으로 자동 상태 비저장 구성을 생성하도록 **IPv6** 상태 비저장 주소 자동 구성(**SLAAC**)을 사용합니다.
   - **Automatic, DHCP only -** 이 옵션을 선택하여 **RA**를 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,경로,네트워크 리소스에만 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)**

7.2. NM-CONNECTION-EDITOR를 사용하여 VPN 연결 구성

그래픽 인터페이스와 함께 **Red Hat Enterprise Linux**를 사용하는 경우 **nm-connection-editor** 애플리케이션에서 **VPN** 연결을 구성할 수 있습니다.

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.

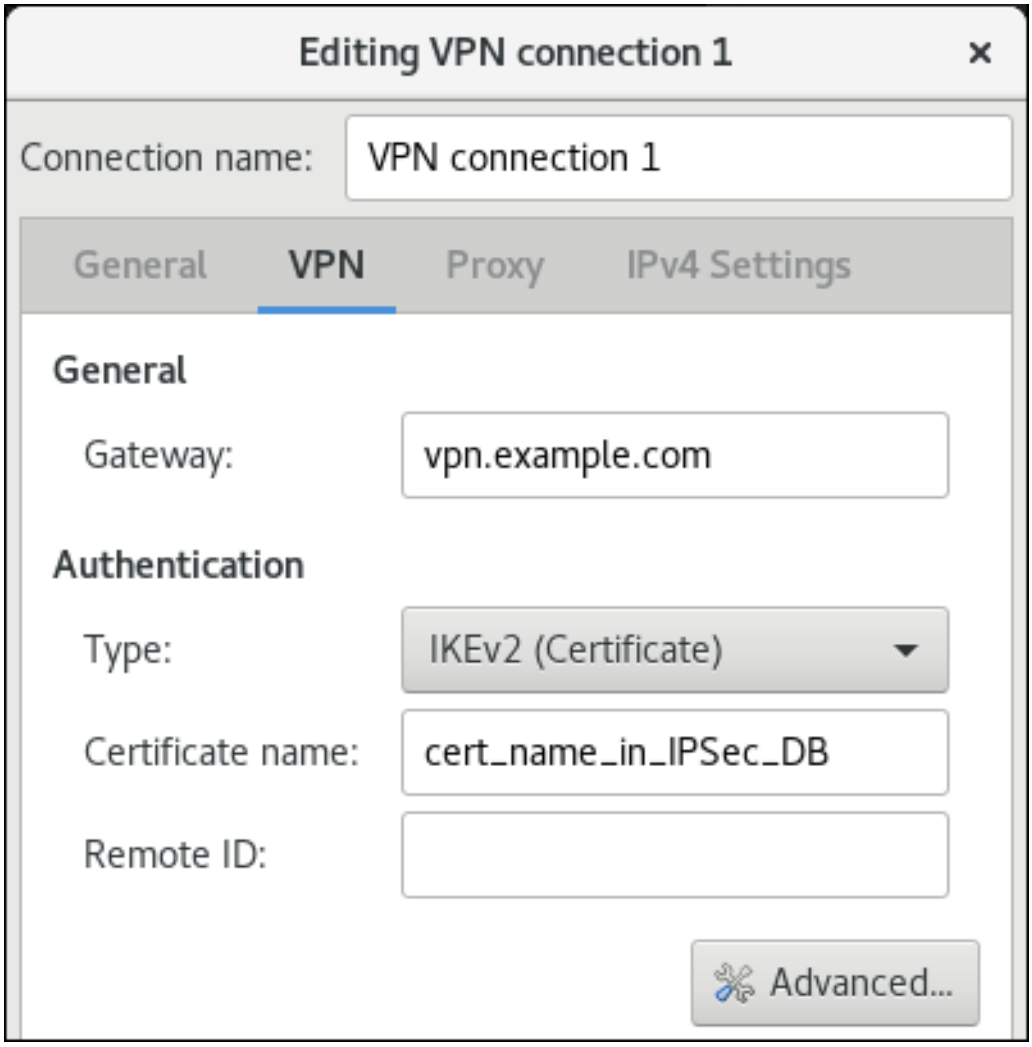
3. **IPsec** 기반 **VPN** 연결 유형을 선택하고 **Create** 를 클릭합니다.

4. **VPN** 탭에서 다음을 수행합니다.

   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 인증서를 사용하여 클라이언트를 인증하며 더 안전합니다. 이 설정을 사용하려면 **IPsec NSS** 데이터베이스의 인증서가 필요합니다.
   - **IKEv1(XAUTH)** 은 사용자 이름과 암호(이전 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** 매개변수에 설정됩니다.



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 사이클은 ikelifetime Libreswan** 매개 변수에 해당합니다. 이 매개 변수는 트래픽을 암호화하는 데 사용되는 키가 유효한 기간을 정의합니다.

- **2 단계 라이프 타임은 salifetime Libreswan** 매개변수에 해당합니다. 이 매개 변수는 보안 연결이 유효한 기간을 정의합니다.

- 연결

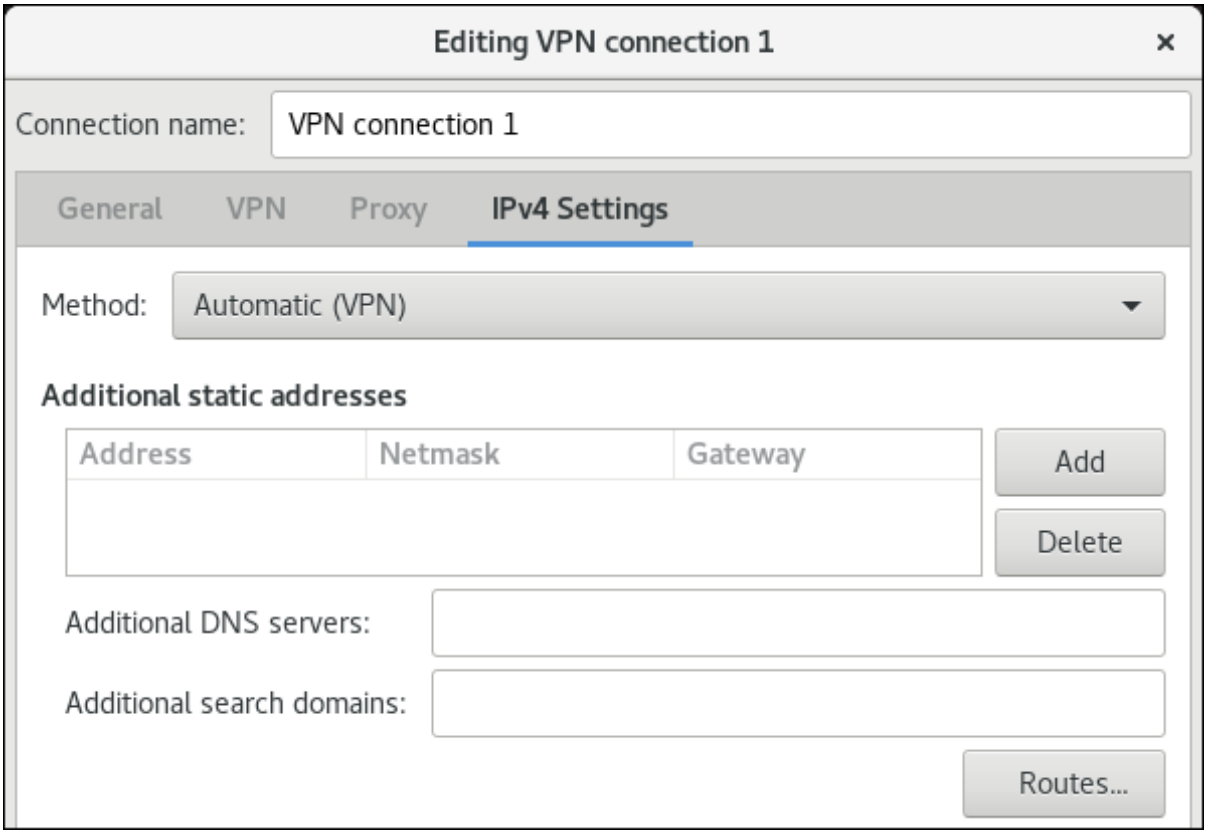
  - 원격 네트워크는 **rightsubnet Libreswan** 매개 변수에 해당하며 **VPN**을 통해 연결할 대상 사설 원격 네트워크를 정의합니다.

    좁히기를 활성화하려면 좁히는 필드를 확인합니다. **IKEv2** 협상에서만 유효합니다.

  - 조각화 활성화 는 **fragmentation Libreswan** 매개변수에 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 고정 주소, **DNS** 서버, 검색 도메인 및 경로를 설정합니다.



6. 연결을 저장합니다.

7. **nm-connection-editor** 를 종료합니다.

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

7.3. IPSEC 연결 가속화를 위해 ESP 하드웨어 오프로드의 자동 감지 및 사용 구성

하드웨어로 **ESP(Encap isolateing Security Payload)**를 하드웨어로 오프로드하면 이더넷을 통한 **IPsec** 연결이 가속화됩니다. 기본적으로 **Libreswan**은 하드웨어에서 이 기능을 지원하는지 여부를 감지

하고 결과적으로 **ESP** 하드웨어 오프로드를 활성화합니다. 기능을 비활성화하거나 명시적으로 활성화한 경우 자동 탐지로 다시 전환할 수 있습니다.

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 검색을 사용해야 하는 연결의 **/etc/ipsec.d/** 디렉터리에서 **Libreswan** 구성 파일을 편집합니다.
2. **nic-offload** 매개변수가 연결 설정에 설정되지 않았는지 확인합니다.
3. **nic-offload** 를 제거한 경우 **ipsec** 서비스를 다시 시작합니다.

```
# systemctl restart ipsec
```

검증

네트워크 카드가 **ESP** 하드웨어 오프로드 지원을 지원하는 경우 결과를 확인하려면 다음 단계를 따르십시오.

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

- **IPsec**을 사용하여 **VPN** 구성

7.4. 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/** 디렉터리에서 **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
- 보안 네트워크 문서에서 **IPsec** 섹션을 사용하여 **VPN** 구성

8장. IP 터널 구성

VPN과 유사하게 IP 터널은 인터넷과 같은 세 번째 네트워크를 통해 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.

NetworkManager는 다음 IP 터널을 지원합니다.

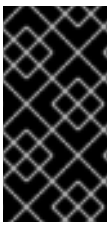
- GRE(Generic Routing Encapsulation)
- 일반 라우팅 Encapsulation over IPv6 (IP6GRE)
- GRETAP(Generic Routing Encapsulation Terminal Access Point)
- 일반 라우팅 캡슐화 터미널 액세스 지점 over IPv6 (IP6GRETAP)
- IPv4 over IPv4 (IPIP)
- IPv6를 통한 IPv4 (IPIP6)
- IPv6 over IPv6 (IP6IP6)

- SIT(Simple Internet transition)

유형에 따라 이러한 터널은 OSI(Open Systems Interconnection) 모델의 계층 2 또는 3에서 작동합니다.

8.1. IPV4 패킷에서 IPV4 트래픽을 캡슐화하기 위해 NMCLI 를 사용하여 IPIP 터널 구성

IPIP(IP IP) 터널은 OSI 계층 3에서 작동하며 RFC 2003 에 설명된 대로 IPv4 패킷의 IPv4 트래픽을 캡슐화합니다.

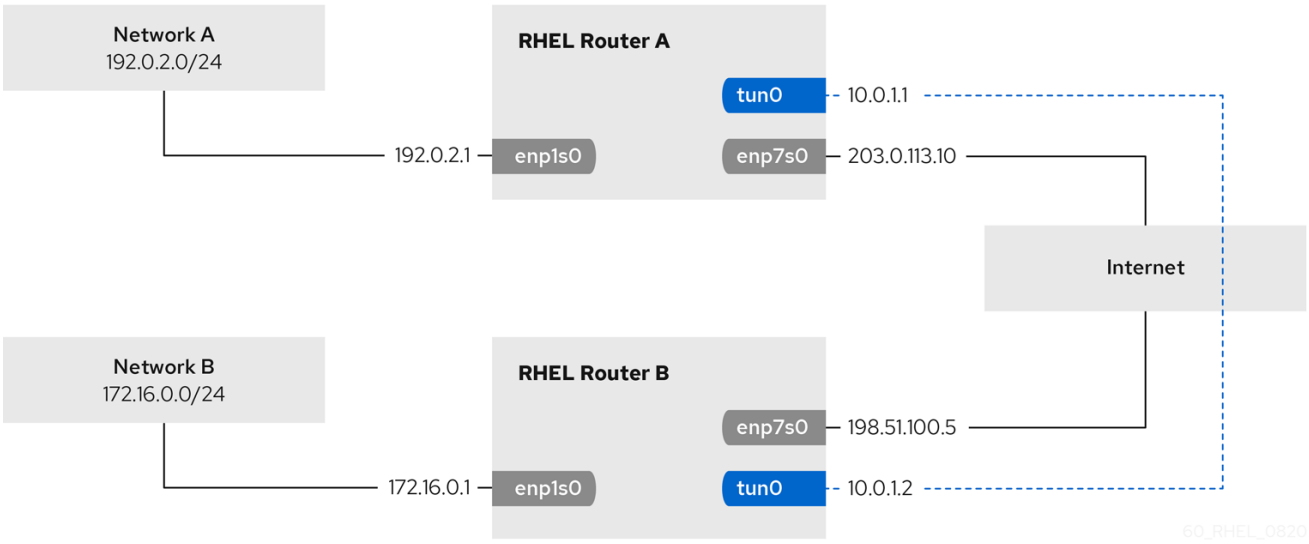


중요

IPIP 터널을 통해 전송된 데이터는 암호화되지 않습니다. 보안상의 이유로 터널은 이미 암호화된 데이터에만 사용합니다(예: HTTPS와 같은 다른 프로토콜).

IPIP 터널은 유니캐스트 패킷만 지원합니다. 멀티 캐스트를 지원하는 IPv4 터널이 필요한 경우 IPv4 패킷에서 계층-3 트래픽을 캡슐화하도록 nmcli를 사용하여 GRE 터널 구성을 참조하십시오.

예를 들어 다음 다이어그램에 표시된 대로 두 RHEL 라우터 간에 IPIP 터널을 생성하여 인터넷을 통해 두 개의 내부 서브넷을 연결할 수 있습니다.



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 A의 RHEL 라우터에서:

a. 이름이 tun0 인 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
```

remote 및 local 매개변수는 원격의 공용 IP 주소와 로컬 라우터를 설정합니다.

b. IPv4 주소를 tun0 장치로 설정합니다.

```
# nmcli connection modify tun0 ipv4.addresses '10.0.1.1/30'
```

두 개의 IP 주소가 있는 /30 서브넷이면 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 정적 경로를 라우터 B의 터널 IP로 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. 이름이 tun0 인 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
```

remote 및 local 매개변수는 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. 트래픽을 192.0.2.0/24 네트워크로 라우팅하는 정적 경로를 라우터 A의 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

- 각 RHEL 라우터에서 다른 라우터의 내부 인터페이스의 IP 주소를 ping합니다.

a. 라우터 A에서 172.16.0.1 을 ping합니다.

```
# ping 172.16.0.1
```

b. 라우터 B에서 192.0.2.1 ping :

```
# ping 192.0.2.1
```

추가 리소스

- nmcli(1) 도움말 페이지
- nm-settings(5) 도움말 페이지

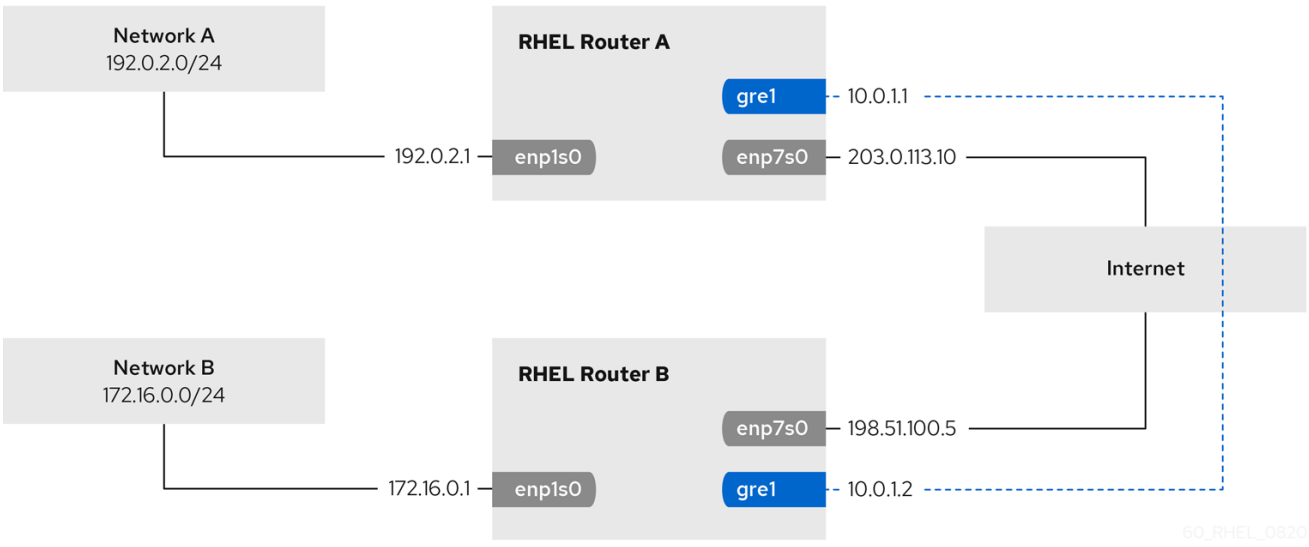
8.2. NMCLI 를 사용하여 IPV4 패킷의 계층-3 트래픽을 캡슐화하여 GRE 터널 구성

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 사용합니다(예: HTTPS와 같은 다른 프로토콜).

예를 들어 다음 다이어그램에 표시된 대로 두 RHEL 라우터 간에 GRE 터널을 생성하여 인터넷을 통해 두 개의 내부 서브넷을 연결할 수 있습니다.



참고

gre0 장치 이름이 예약되어 있습니다. **gre1** 또는 장치에 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
   ```

   remote 및 **local** 매개변수는 원격의 공용 **IP** 주소와 로컬 라우터를 설정합니다.

   b. **IPv4** 주소를 **gre1** 장치로 설정합니다.

```
# nmcli connection modify gre1 ipv4.addresses '10.0.1.1/30'
```

두 개의 IP 주소가 있는 /30 서브넷이면 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 정적 경로를 라우터 B의 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
```

remote 및 local 매개변수는 원격의 공용 IP 주소와 로컬 라우터를 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. 트래픽을 192.0.2.0/24 네트워크로 라우팅하는 정적 경로를 라우터 A의 터널 IP로 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

1. 각 RHEL 라우터에서 다른 라우터의 내부 인터페이스의 IP 주소를 ping합니다.

a. 라우터 A에서 172.16.0.1 을 ping합니다.

```
# ping 172.16.0.1
```

b. 라우터 B에서 192.0.2.1 ping :

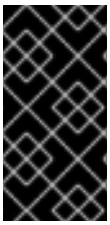
```
# ping 192.0.2.1
```

추가 리소스

- nmcli(1) 도움말 페이지
- nm-settings(5) 도움말 페이지

8.3. IPV4를 통해 이더넷 프레임을 전송하도록 GRETAP 터널 구성

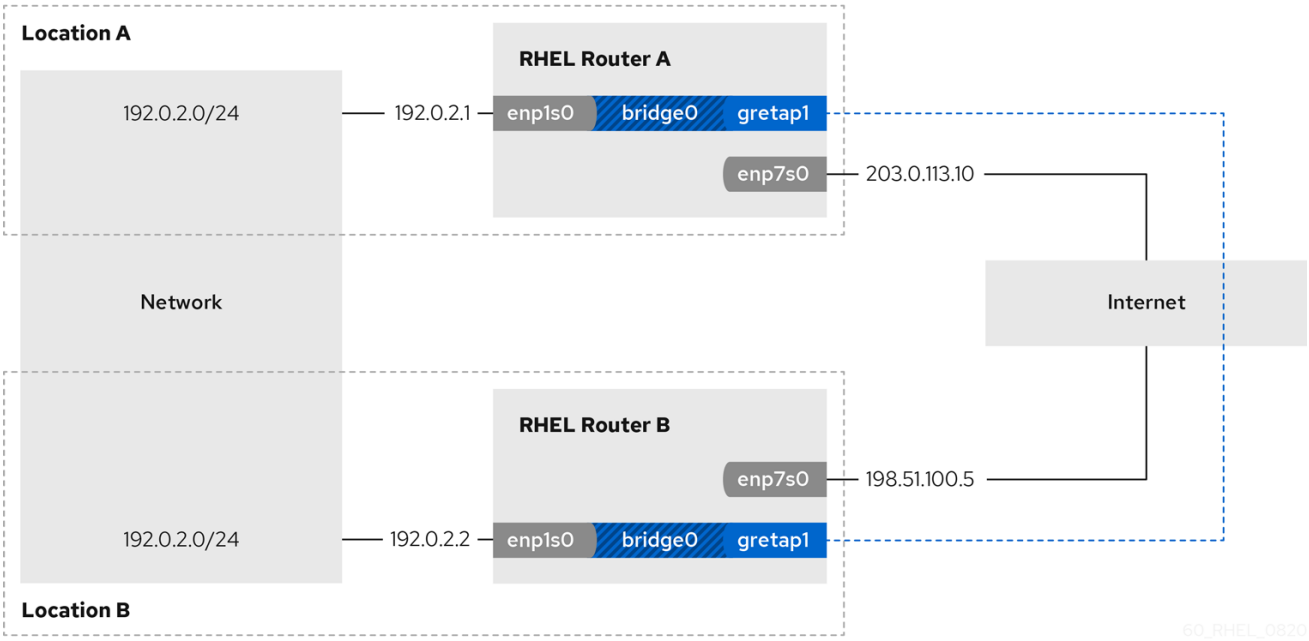
GRETAP(Generic Routing Encapsulation Terminal Access Point) 터널은 OSI 수준 2에서 작동하고 RFC 2784 에 설명된 대로 IPv4 패킷의 이더넷 트래픽을 캡슐화합니다.



중요

GRETAP 터널을 통해 전송되는 데이터는 암호화되지 않습니다. 보안상의 이유로 VPN 또는 다른 암호화된 연결을 통해 터널을 설정하십시오.

예를 들어 다음 다이어그램에 표시된 대로 두 RHEL 라우터 간에 GRETAP 터널을 생성하여 브리지를 사용하여 두 개의 네트워크를 연결할 수 있습니다.



참고

gretap0 장치 이름이 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 A의 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
   ```

   remote 및 local 매개변수는 원격의 공용 IP 주소와 로컬 라우터를 설정합니다.

   e. 선택 사항: 필요하지 않은 경우 스패닝 트리 프로토콜 (STP)을 비활성화합니다.

   ```
   # nmcli connection modify bridge0 bridge.stp no
   ```

   기본적으로 STP가 활성화되어 연결을 사용할 수 있기 전에 지연이 발생합니다.

f. **bridge0** 연결을 활성화하면 브리지의 포트를 자동으로 활성화합니다.

```
# 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
```

remote 및 **local** 매개변수는 원격의 공용 **IP** 주소와 로컬 라우터를 설정합니다.

e. 선택 사항: 필요하지 않은 경우 스패닝 트리 프로토콜 **(STP)**을 비활성화합니다.

```
# nmcli connection modify bridge0 bridge.stp no
```

f. **bridge0** 연결을 활성화하면 브리지의 포트를 자동으로 활성화합니다.

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

g. **bridge0** 연결을 활성화합니다.

```
# nmcli connection up bridge0
```

검증

1. 두 라우터 모두에서 **enp1s0** 및 **gretap1** 연결이 연결되고 **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. 라우터 **A**에서 **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

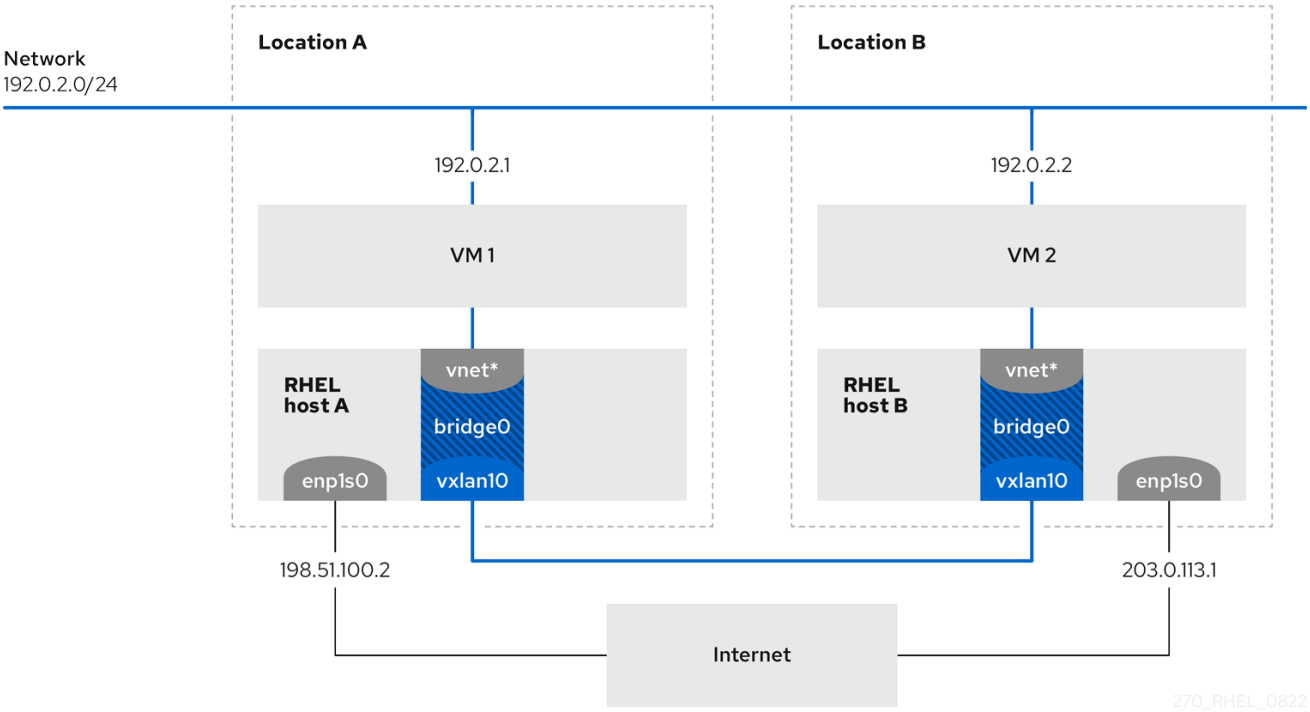
- **nmcli(1)** 도움말 페이지
- **nm-settings(5)** 도움말 페이지

8.4. 추가 리소스

- **ip-link(8)** 도움말 페이지

9장. VXLAN을 사용하여 VM의 가상 계층 2 도메인 생성

VXLAN(가상 확장 가능한 LAN)은 UDP 프로토콜을 사용하여 IP 네트워크를 통해 계층 2 트래픽을 터널링하는 네트워킹 프로토콜입니다. 예를 들어 다른 호스트에서 실행되는 특정 VM(가상 머신)은 VXLAN 터널을 통해 통신할 수 있습니다. 호스트는 다른 서브넷에 있거나 전 세계 다른 데이터 센터에 있을 수 있습니다. VM의 관점에서 동일한 VXLAN의 다른 VM은 동일한 계층-2 도메인 내에 있습니다.



이 예에서 RHEL-host-A 및 RHEL-host-B는 브리지 br0 을 사용하여 각 호스트의 가상 네트워크를 vxlan10 이라는 VXLAN으로 연결합니다. 이 구성으로 인해 VXLAN은 VM에 표시되지 않으며 VM에는 특별한 구성이 필요하지 않습니다. 나중에 더 많은 VM을 동일한 가상 네트워크에 연결하면 VM이 동일한 가상 계층-2 도메인의 멤버가 자동으로 됩니다.

중요

일반 계층 2 트래픽과 마찬가지로 VXLAN의 데이터는 암호화되지 않습니다. 보안상의 이유로 VPN 또는 기타 유형의 암호화된 연결을 통해 VXLAN을 사용하십시오.

9.1. VXLAN의 이점

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

- VXLAN에서는 24비트 ID를 사용합니다. 따라서 최대 16,777,216개의 격리된 네트워크를 생성할 수 있습니다. 예를 들어 VLAN(가상 LAN)은 4,096 격리된 네트워크만 지원합니다.

- VXLAN은 IP 프로토콜을 사용합니다. 이를 통해 동일한 계층-2 도메인 내의 서로 다른 네트워크 및 위치에서 트래픽을 라우팅하고 가상으로 시스템을 실행할 수 있습니다.

- 대부분의 터널 프로토콜과 달리 VXLAN은 지점 간 네트워크뿐만 아니라 VXLAN은 다른 엔드포인트의 IP 주소를 동적으로 학습하거나 정적으로 구성된 전달 항목을 사용할 수 있습니다.

- 특정 네트워크 카드는 UDP 터널 관련 오프로드 기능을 지원합니다.

추가 리소스

- /usr/share/doc/kernel-doc- <*kernel_version*> /Documentation/networking/vxlan.rst 가 kernel-doc 패키지에서 제공하는

9.2. 호스트에서 이더넷 인터페이스 구성

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

두 RHEL 호스트 모두에서 이 절차를 실행하고 그에 따라 IP 주소 구성을 조정합니다.

사전 요구 사항

- 호스트는 이더넷에 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 건너뜁니다.

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 ping하여 IP 설정을 확인합니다.

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

해당 호스트에서 네트워크도 구성하기 전에 다른 **VM** 호스트를 **ping**할 수 없습니다.

추가 리소스

- **nm-settings(5)** 도움말 페이지

9.3. VXLAN이 연결된 네트워크 브리지 만들기

VXLAN(가상 확장 가능한 **LAN**)을 **VM**(가상 머신)에 표시되지 않게 하려면 호스트에 브리지를 만들고 **VXLAN**을 브리지에 연결합니다. **NetworkManager**를 사용하여 브리지와 **VXLAN**을 모두 만듭니다. **VM**의 트래픽 액세스 포인트(**TAP**) 장치는 일반적으로 호스트의 **vnet*** 라는 장치를 브리지에 추가하지 않습니다. **libvirtd** 서비스는 **VM**이 시작될 때 동적으로 추가합니다.

두 **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 **2**에서 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.
- 로컬 **198.51.100.2:** 발신 패킷의 소스 **IP** 주소를 설정합니다.
- 원격 **203.0.113.1: VXLAN** 장치 전달 데이터베이스에서 대상 링크 계층 주소를 알 수 없는 경우 발신 패킷에서 사용할 유니캐스트 또는 멀티 캐스트 **IP** 주소를 설정합니다.
- 마스터 **br0: br0** 연결에서 이 **VXLAN** 연결을 포트로 생성하도록 설정합니다.
- **ipv4.method**가 비활성화 되고 **ipv6.method**가 비활성화되었습니다. 브리지에서 **IPv4** 및 **IPv6**를 비활성화합니다.

기본적으로 **NetworkManager**는 **8472** 를 대상 포트로 사용합니다. 대상 포트가 다르면 대상 포트 **<** ***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)** 도움말 페이지

9.4. 기존 브리지를 사용하여 LIBVIRT에 가상 네트워크 생성

VM(가상 머신**)**이 연결된 **VXLAN(**가상 확장 가능 **LAN)**과 함께 **br0** 브리지를 사용하도록 하려면 먼저 이 브리지를 사용하는 **libvirtd** 서비스에 가상 네트워크를 추가합니다.

사전 요구 사항

- **libvirt** 패키지가 설치되어 있어야 합니다.
- **libvirtd** 서비스를 시작하고 활성화했습니다.
- **RHEL**에서 **VXLAN**을 사용하여 **br0** 장치를 구성하셨습니다.

절차

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

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

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

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

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

3. ~/vxlan10-bridge.xml 파일을 제거합니다.

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

4. vxlan10-bridge 가상 네트워크를 시작합니다.

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

5. libvirtd 서비스가 시작될 때 vxlan10-bridge 가상 네트워크가 자동으로 시작되도록 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) man page

9.5. VXLAN을 사용하도록 가상 머신 구성

호스트에서 연결된 VXLAN(가상 확장 가능 LAN)이 있는 브리지 장치를 사용하도록 VM을 구성하려면 vxlan10-bridge 가상 네트워크를 사용하는 새 VM을 생성하거나 이 네트워크를 사용하도록 기존 VM의

설정을 업데이트합니다.

RHEL 호스트에서 다음 절차를 수행합니다.

사전 요구 사항

- **libvirtd** 에서 **vxlan10-bridge** 가상 네트워크를 구성하셨습니다.

절차

- 새 **VM**을 생성하고 **vxlan10-bridge** 네트워크를 사용하도록 구성하려면 **VM**을 생성할 때 **--network network:*vxlan10-bridge*** 옵션을 **virt-install** 명령에 전달합니다.

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

- 기존 **VM**의 네트워크 설정을 변경하려면 다음을 수행합니다.

a. **VM**의 네트워크 인터페이스를 **vxlan10-bridge** 가상 네트워크에 연결합니다.

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

b. **VM**을 종료하고 다시 시작합니다.

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

검증

1. 호스트에서 **VM**의 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. **vxlan10-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
```

libvirtd 서비스는 브리지 구성을 동적으로 업데이트합니다. **vxlan10-bridge** 네트워크를 사용하는 **VM**을 시작하면 호스트의 해당 **vnet*** 장치가 브리지 포트로 표시됩니다.

3. **ARP(Address Resolution Protocol)** 요청을 사용하여 **VM**이 동일한 **VXLAN**에 있는지 확인합니다.

   a. 동일한 **VXLAN**에서 두 개 이상의 **VM**을 시작합니다.

   b. 한 **VM**에서 다른 **VM**으로 **ARP** 요청을 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))
```

명령에서 응답을 표시하는 경우 **VM**은 동일한 계층**-2** 도메인에, 이 경우 동일한 **VXLAN**에 있습니다.

iputils 패키지를 설치하여 **arping** 유틸리티를 사용합니다.

추가 리소스

- **virt-install(1)** 매뉴얼 페이지
- **virt-xml(1)** man page
- **virsh(1)** man page

- **arping(8)** 매뉴얼 페이지

10장. ECDHE 연결 관리

RHEL은 FlexVolume 네트워크를 구성하고 연결하는 여러 유틸리티 및 애플리케이션을 제공합니다. 예를 들면 다음과 같습니다.

- nmcli 유틸리티를 사용하여 명령줄을 사용하여 연결을 구성합니다.
- nmtui 애플리케이션을 사용하여 텍스트 기반 사용자 인터페이스에서 연결을 구성합니다.
- GNOME 시스템 메뉴를 사용하여 구성이 필요하지 않은 10.0.0.1 네트워크에 신속하게 연결합니다.
- GNOME 애플리케이션을 사용하여 연결을 구성하려면 GNOME Settings 애플리케이션을 사용합니다.
- nm-connection-editor 애플리케이션을 사용하여 그래픽 사용자 인터페이스에서 연결을 구성합니다.
- 네트워크 RHEL 시스템 역할을 사용하여 하나 이상의 호스트에서 연결 구성을 자동화합니다.

10.1. 지원되는ECDHE 보안 유형

ECDHE 네트워크 지원 보안 유형에 따라 데이터를 더 많이 또는 덜 안전하게 전송할 수 있습니다.

주의

암호화를 사용하지 않거나 안전하지 않은 WEP 또는ECDHE 표준만 지원하는 ECDHE 네트워크에 연결하지 마십시오.

RHEL 8에서는 다음 hieradata 보안 유형을 지원합니다.

- 제공되지 않음: 암호화가 비활성화되어 있으며 네트워크를 통해 데이터가 일반 텍스트로 전송됩니다.

- 향상된 Open: OWE(opportunistic 무선 암호화)를 사용하면 장치는 고유한 쌍의 PMK(마스터 키)를 협상하여 인증 없이 무선 네트워크에서 연결을 암호화합니다.

- WEP 40/128 비트 키 (Hex 또는 ASCII): 이 모드의 throughred Equivalent 개인 정보 보호 (WEP) 프로토콜은 16 진수 또는 ASCII 형식에서만 사전 공유 키를 사용합니다. WEP는 더 이상 사용되지 않으며 RHEL 9.1에서 제거될 예정입니다.

- WEP 128비트 Passphrase. 이 모드의 WEP 프로토콜은 암호의 MD5 해시를 사용하여 WEP 키를 도출합니다. WEP는 더 이상 사용되지 않으며 RHEL 9.1에서 제거될 예정입니다.

- 동적 WEP (802.1x): WEP 프로토콜을 동적 키로 사용하는 802.1X 및 EAP의 조합입니다. WEP는 더 이상 사용되지 않으며 RHEL 9.1에서 제거될 예정입니다.

- LEAP: Cisco에서 개발한 Lightweight Extensible Authentication Protocol은 확장 가능한 인증 프로토콜(EAP)의 독점 버전입니다.

- WPA & WPA2 Personal: 개인 모드에서WPA(Wi-Fi Protected Access) 및-8-Fi 보호 액세스 2(WPA2) 인증 방법은 사전 공유 키를 사용합니다.

- ECDHE &ECDHE2 Enterprise: 엔터프라이즈 모드에서ECDHE 및ECDHE2는 EAP 프레임워크를 사용하고 사용자를 RDIUS(원격 인증 서비스) 서버로 인증합니다.

- WPA3 Personal: ovs-Fi Protected Access 3 (WPA3) 개인은 사전 공격을 방지하기 위해 PSK(Pre-shared keys) 대신 SAE(동시 인증)를 사용합니다. ECDHE3은 완전한 전달 기밀성(PFS)을 사용합니다.

10.2. NMCLI를 사용하여 INTERNET NETWORK에 연결

nmcli 유틸리티를 사용하여ECDHE 네트워크에 연결할 수 있습니다. 처음으로 네트워크에 연결하려고 하면 유틸리티에서 NetworkManager 연결 프로필을 자동으로 생성합니다. 네트워크에 고정 IP 주소와 같은 추가 설정이 필요한 경우 자동으로 생성된 후 프로필을 수정할 수 있습니다.

사전 요구 사항

- host에ECDHE 장치가 설치되어 있어야 합니다.
- 하드웨어 스위치가 있는 경우ECDHE 장치가 활성화됩니다.

절차

1. NetworkManager에서 CoreDNS 무선이 비활성화된 경우 다음 기능을 활성화합니다.

```
# nmcli radio wifi on
```

2. 선택 사항: 사용 가능한ECDHE 네트워크를 표시합니다.

```
# nmcli device wifi list
IN-USE  BSSID              SSID        MODE   CHAN  RATE       SIGNAL  BARS  SECURITY
        00:53:00:2F:3B:08  Office      Infra  44    270 Mbit/s  57      ▂▄▆_  WPA2 WPA3
        00:53:00:15:03:BF  --          Infra  1     130 Mbit/s  48      ▂▄__  WPA2 WPA3
```

SSID(서비스 세트 식별자) 열에는 네트워크 이름이 포함되어 있습니다. 열이 표시되는 경우 이 네트워크의 액세스 지점이 SSID를 브로드캐스트하지 않습니다.

3. ECDHE 네트워크에 연결합니다.

```
# nmcli device wifi connect Office --ask
Password: wifi-password
```

명령에 대화식으로 입력하는 대신 암호를 설정하려면 --ask 대신 *암호ECDHE-password* 옵션을 사용합니다.

```
# nmcli device wifi connect Office wifi-password
```

네트워크에 고정 IP 주소가 필요한 경우 NetworkManager는 이 시점에서 연결을 활성화하지 못합니다. 이후 단계에서 IP 주소를 구성할 수 있습니다.

4. 네트워크에 고정 IP 주소가 필요한 경우:

a. IPv4 주소 설정을 구성합니다. 예를 들면 다음과 같습니다.

```
# nmcli connection modify Office ipv4.method manual ipv4.addresses 192.0.2.1/24
ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com
```

b. IPv6 주소 설정을 구성합니다. 예를 들면 다음과 같습니다.

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

5. 연결을 다시 활성화합니다.

```
# nmcli connection up Office
```

검증

1. 활성 연결을 표시합니다.

```
# nmcli connection show --active
NAME    ID                                    TYPE  DEVICE
Office  2501eb7e-7b16-4dc6-97ef-7cc460139a58  wifi  wlp0s20f3
```

출력에 사용자가 생성한ECDHE 연결이 나열되면 연결이 활성화됩니다.

2. 호스트 이름 또는 IP 주소를 ping합니다.

```
# ping -c 3 example.com
```

추가 리소스

- nm-settings-nmcli(5) man page

10.3. GNOME 시스템 메뉴를 사용하여 IPXE 네트워크에 연결

GNOME 시스템 메뉴를 사용하여ECDHE 네트워크에 연결할 수 있습니다. 네트워크에 처음 연결하면 GNOME에서 NetworkManager 연결 프로필을 만듭니다. 연결 프로필을 자동으로 연결하지 않도록 구성하는 경우 GNOME 시스템 메뉴를 사용하여 기존 NetworkManager 연결 프로필을 사용하여 BOOM 네트워크에 수동으로 연결할 수도 있습니다.

참고

GNOME 시스템 메뉴를 사용하여 처음으로ECDHE 네트워크에 대한 연결을 설정하는 데에는 몇 가지 제한 사항이 있습니다. 예를 들어 IP 주소 설정을 구성할 수 없습니다. 이 경우 먼저 연결을 구성합니다.

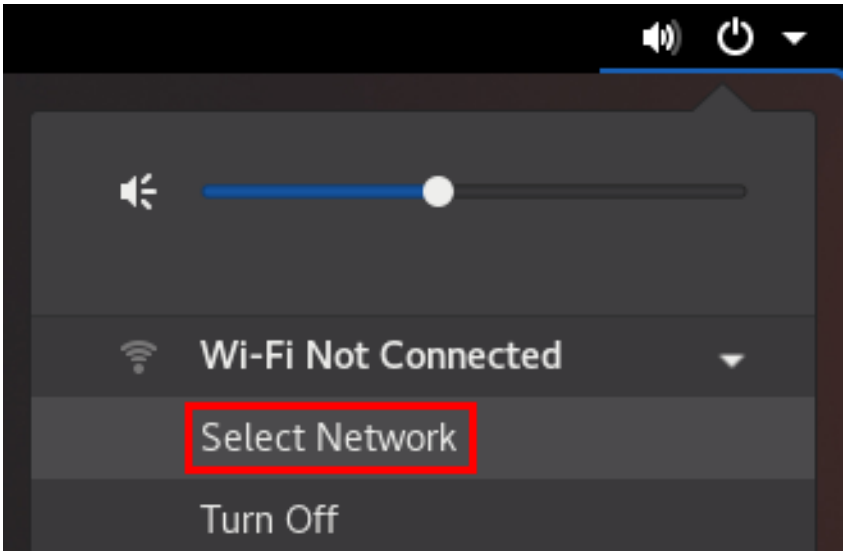
- GNOME 설정 애플리케이션에서
- nm-connection-editor 애플리케이션에서
- nmcli 명령 사용

사전 요구 사항

- host에ECDHE 장치가 설치되어 있어야 합니다.
- 하드웨어 스위치가 있는 경우ECDHE 장치가 활성화됩니다.

절차

1. 상단 표시줄 오른쪽에 있는 시스템 메뉴를 엽니다.
2. hub -Fi Not Connected 항목을 확장합니다.
3. 네트워크 선택을 클릭합니다.



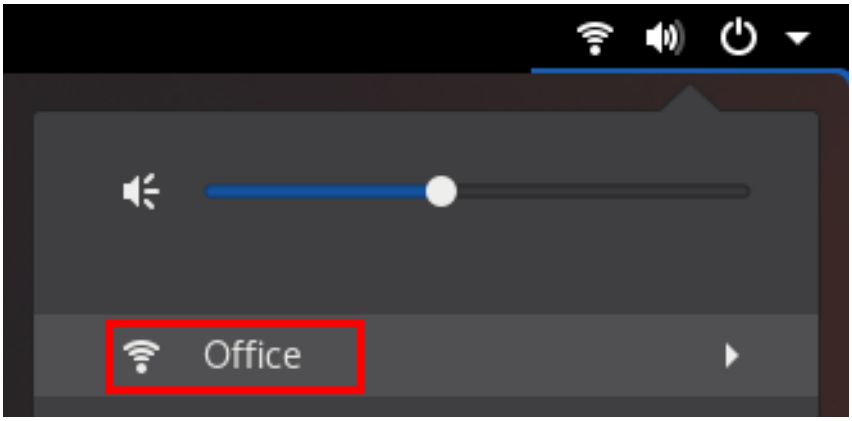
4. 연결하려는 **networks**를 선택합니다.

5. 연결을 클릭합니다.

6. 이 네트워크에 처음 연결하는 경우 네트워크의 암호를 입력하고 연결을 클릭합니다.

검증

1. 상단 표시줄 오른쪽에 있는 시스템 메뉴를 열고**ECDHE** 네트워크가 연결되어 있는지 확인합니다.



네트워크가 목록에 표시되면 연결된 것입니다.

2. 호스트 이름 또는 **IP** 주소를 **ping**합니다.

```
# ping -c 3 example.com
```

10.4. GNOME 설정 애플리케이션을 사용하여 10.0.0.1 네트워크에 연결

gnome-control-center 라고도 하는 GNOME 설정 애플리케이션을 사용하여ECDHE 네트워크에 연결하고 연결을 구성할 수 있습니다. 네트워크에 처음 연결하면 GNOME에서 NetworkManager 연결 프로필을 만듭니다.

GNOME 설정에서 RHEL에서 지원하는 모든ECDHE 네트워크 보안 유형에 대해ECDHE 연결을 구성할 수 있습니다.

사전 요구 사항

- host에ECDHE 장치가 설치되어 있어야 합니다.
- 하드웨어 스위치가 있는 경우ECDHE 장치가 활성화됩니다.

절차

1. Super 키를 누르고-8- Fi 를 입력한 후 Enter 를 누릅니다.
2. 연결하려는ECDHE 네트워크의 이름을 클릭합니다.
3. 네트워크의 암호를 입력하고 연결을 클릭합니다.
4. 네트워크에 고정 IP 주소 또는ECDHE2 개인 이외의 보안 유형과 같은 추가 설정이 필요한 경우:
   a. 네트워크 이름 옆에 있는 톱니바퀴 아이콘을 클릭합니다.
   b. 선택 사항: 세부 정보 탭에서 네트워크 프로필을 자동으로 연결하지 않도록 구성합니다.

   이 기능을 비활성화하는 경우 GNOME 설정 또는 GNOME 시스템 메뉴를 사용하여 항상 네트워크에 수동으로 연결해야 합니다.

c.

IPv4 탭에서 IPv4 설정을 구성하고 IPv6 탭의 IPv6 설정을 구성합니다.

d.

보안 탭에서 네트워크 인증을 선택합니다(예:ECDHE 3 개인)에서 암호를 입력합니다.

선택한 보안에 따라 애플리케이션에 추가 필드가 표시됩니다. 이에 따라 채우십시오. 자세한 내용은 administrator of theECDHE network를 참조하십시오.

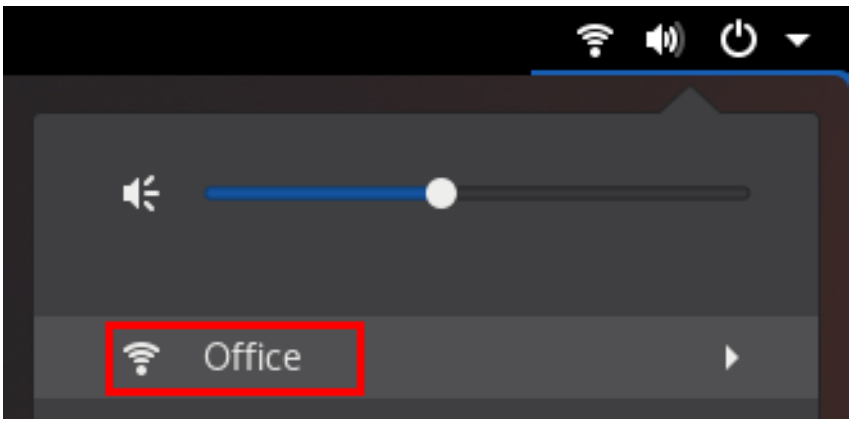
e.

적용을 클릭합니다.

검증

1.

상단 표시줄 오른쪽에 있는 시스템 메뉴를 열고ECDHE 네트워크가 연결되어 있는지 확인합니다.



네트워크가 목록에 표시되면 연결된 것입니다.

2.

호스트 이름 또는 IP 주소를 ping합니다.

```
# ping -c 3 example.com
```

10.5. NMTUI를 사용하여 CRYOSTAT 연결 구성

nmtui 애플리케이션은 NetworkManager에 대한 텍스트 기반 사용자 인터페이스를 제공합니다. nmtui 를 사용하여ECDHE 네트워크에 연결할 수 있습니다.

참고

nmtui 에서 :

- 커서 키를 사용하여 이동합니다.
- 버튼을 선택하고 Enter 를 누릅니다.
- Space 를 사용하여 확인란을 선택하고 지웁니다.

절차

1. 연결에 사용할 네트워크 장치 이름을 모르는 경우 사용 가능한 장치를 표시합니다.

```
# nmcli device status
DEVICE    TYPE      STATE              CONNECTION
wlp2s0    wifi      unavailable        --
...
```

2. start nmtui:

```
# nmtui
```

3. Edit a connection 을 선택하고 Enter 를 누릅니다.

4. 추가 버튼을 누릅니다.

5. 네트워크 유형 목록에서 Wi-Fi 를 선택하고 Enter 를 누릅니다.

6. 선택 사항: 생성할 NetworkManager 프로필의 이름을 입력합니다.

프로필이 여러 개인 호스트에서 의미 있는 이름을 사용하면 프로필의 용도를 쉽게 식별할 수 있습니다.

7. 네트워크 장치 이름을 장치 필드에 입력합니다.

8. swi-Fi 네트워크의 이름, Service Set Identifier(SSID)를 SSID 필드에 입력합니다.

9. Mode 필드를 기본값인 Client 로 설정된 상태로 둡니다.

10. 보안 필드를 선택하고 Enter 를 누른 다음 목록에서 네트워크의 인증 유형을 설정합니다.

    선택한 인증 유형에 따라 nmtui 는 다른 필드를 표시합니다.

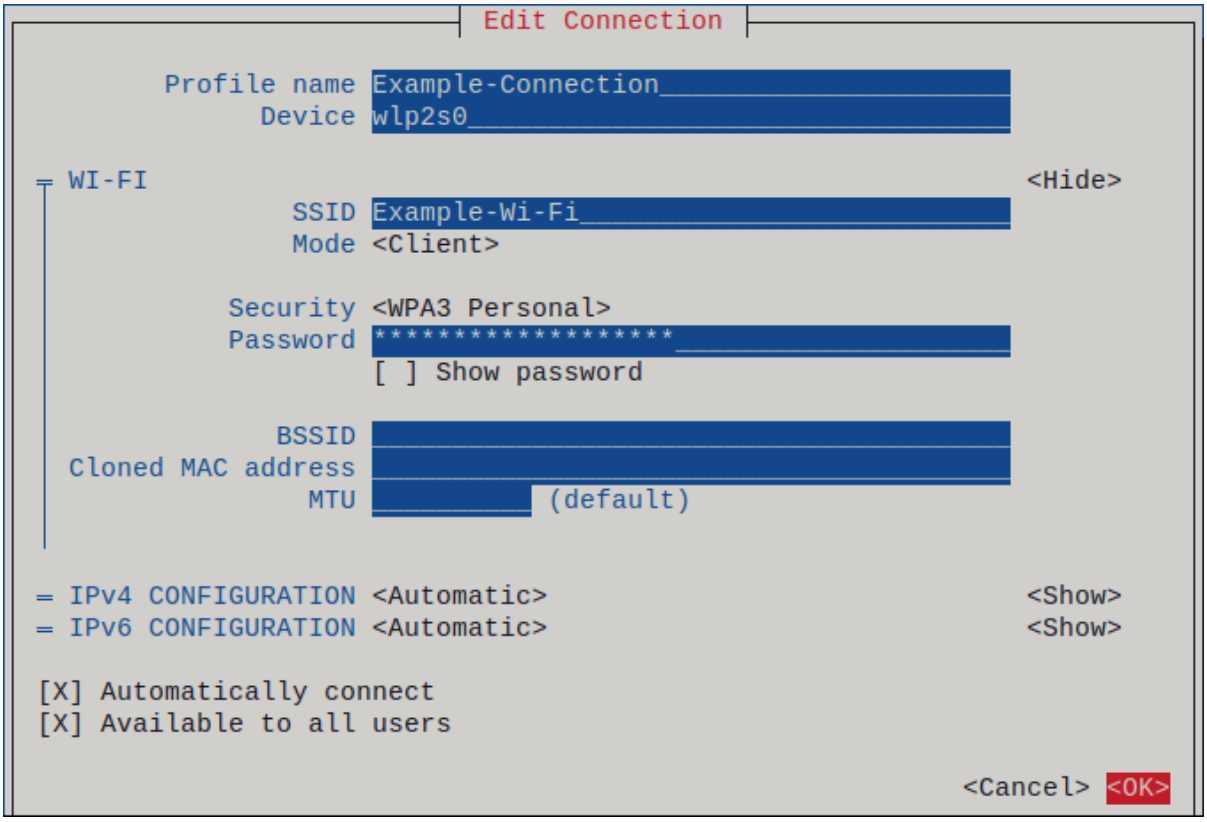
11. 인증 유형 관련 필드를 작성합니다.

12. hub-Fi 네트워크에 고정 IP 주소가 필요한 경우:

    a. 프로토콜 옆에 있는 자동 버튼을 눌러 표시된 목록에서 Manual 을 선택합니다.

    b. 추가 필드를 표시하고 작성하도록 구성할 프로토콜 옆에 있는 표시 버튼을 누릅니다.

13. 확인 버튼을 눌러 새 연결을 생성하고 자동으로 활성화합니다.



14. **Back** 버튼을 눌러 주 메뉴로 돌아갑니다.

15. **Quit** 을 선택하고 **Enter** 를 눌러 **nmtui** 애플리케이션을 종료합니다.

검증

1. 활성 연결을 표시합니다.

```
# nmcli connection show --active
NAME    ID                                    TYPE  DEVICE
Office  2501eb7e-7b16-4dc6-97ef-7cc460139a58  wifi  wlp0s20f3
```

출력에 사용자가 생성한**ECDHE** 연결이 나열되면 연결이 활성화됩니다.

2. 호스트 이름 또는 **IP** 주소를 **ping**합니다.

```
# ping -c 3 example.com
```

10.6. NM-CONNECTION-EDITOR를 사용하여 10.0.0.1 연결 구성

nm-connection-editor 애플리케이션을 사용하여 무선 네트워크에 대한 연결 프로필을 생성할 수 있습니다. 이 애플리케이션에서는 **RHEL**에서 지원하는 모든**ECDHE** 네트워크 인증 유형을 구성할 수 있습니다.

기본적으로 **NetworkManager**는 연결 프로필에 자동 연결 기능을 활성화하고 사용 가능한 경우 저장된 네트워크에 자동으로 연결됩니다.

사전 요구 사항

- **host**에**ECDHE** 장치가 설치되어 있어야 합니다.
- 하드웨어 스위치가 있는 경우**ECDHE** 장치가 활성화됩니다.

절차

1. 터미널을 열고 다음을 입력합니다.

```
# nm-connection-editor
```

2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. **hub -Fi** 연결 유형을 선택하고 **Create** 를 클릭합니다.
4. 선택 사항: 연결 프로필의 이름을 설정합니다.
5. 선택 사항: 일반 탭에서 자동으로 연결되지 않도록 네트워크 프로필을 구성합니다.

   이 기능을 비활성화하는 경우 **GNOME** 설정 또는 **GNOME** 시스템 메뉴를 사용하여 항상 네트워크에 수동으로 연결해야 합니다.
6. **hub -Fi** 탭에서 **SSID**(서비스 세트 식별자)를 **SSID** 필드에 입력합니다.
7. **ovs -Fi Security** 탭에서 **network**의 인증 유형 (예:**ECDHE 3 Personal**)을 선택하고 암호를 입력합니다.

선택한 보안에 따라 애플리케이션에 추가 필드가 표시됩니다. 이에 따라 채우십시오. 자세한 내용은 **administrator of theECDHE network**를 참조하십시오.

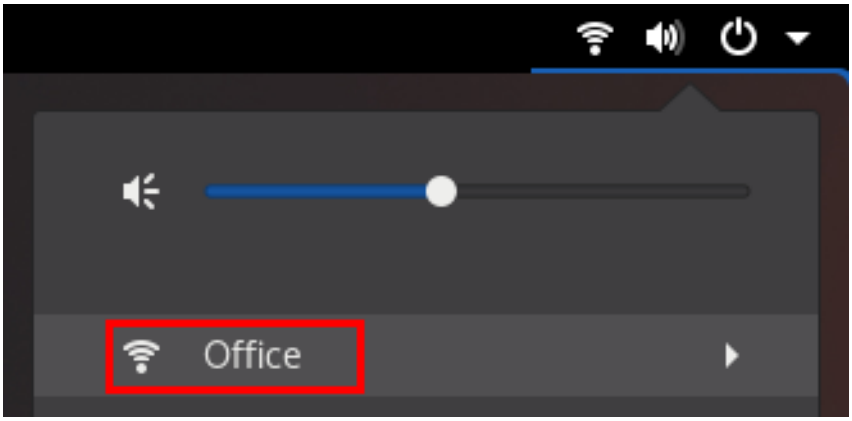
8. **IPv4** 탭에서 **IPv4** 설정을 구성하고 **IPv6** 탭의 **IPv6** 설정을 구성합니다.

9. 저장을 클릭합니다.

10. 네트워크 연결 창을 닫습니다.

검증

1. 상단 표시줄 오른쪽에 있는 시스템 메뉴를 열고**ECDHE** 네트워크가 연결되어 있는지 확인합니다.



네트워크가 목록에 표시되면 연결된 것입니다.

2. 호스트 이름 또는 **IP** 주소를 **ping**합니다.

```
# ping -c 3 example.com
```

10.7. 네트워크 RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 CRYO STAT 연결 구성

RHEL 시스템 역할을 사용하여 **Cryostat** 연결 생성을 자동화할 수 있습니다. 예를 들어 **Ansible** 플레이북을 사용하여 **wlp1s0** 인터페이스에 대한 무선 연결 프로필을 원격으로 추가할 수 있습니다. 생성된 프로필은 **802.1X** 표준을 사용하여 **client**를**ECDHE** 네트워크로 인증합니다. 플레이북은 **DHCP**를 사용하도록 연결 프로필을 구성합니다. 고정 **IP** 설정을 구성하려면 그에 따라 **ip** 사전의 매개변수를 조정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 관리 노드에 **wpa_supplicant** 패키지를 설치했습니다.
- **DHCP**는 관리 노드의 네트워크에서 사용할 수 있습니다.
- **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** 인증서는 **/srv/data/ca.crt** 파일에 저장됩니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure a wifi connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      ansible.builtin.copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0400
```

```
- name: Copy client certificate for 802.1X authentication
  ansible.builtin.copy:
    src: "/srv/data/client.crt"
    dest: "/etc/pki/tls/certs/client.crt"

- name: Copy CA certificate for 802.1X authentication
  ansible.builtin.copy:
    src: "/srv/data/ca.crt"
    dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

- block:
    - ansible.builtin.import_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: Configure the Example-wifi profile
            interface_name: wlp1s0
            state: up
            type: wireless
            autoconnect: yes
            ip:
              dhcp4: true
              auto6: true
            wireless:
              ssid: "Example-wifi"
              key_mgmt: "wpa-eap"
            ieee802_1x:
              identity: "user_name"
              eap: tls
              private_key: "/etc/pki/tls/client.key"
              private_key_password: "password"
              private_key_password_flags: none
              client_cert: "/etc/pki/tls/client.pem"
              ca_cert: "/etc/pki/tls/cacert.pem"
              domain_suffix_match: "example.com"
```

이러한 설정은 **wlp1s0** 인터페이스에 대한 **Cryostat** 연결 프로필을 정의합니다. 프로필은 **802.1X** 표준을 사용하여 **internet network**에 대한 클라이언트를 인증합니다. 연결은 **DHCP** 서버와 **IPv6** 상태 비저장 주소 자동 구성(**SLAAC**)에서 **IPv4** 주소, **IPv6** 주소, 기본 게이트웨이, 경로, **DNS** 서버 및 검색 도메인을 검색합니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

10.8. NMCLI를 사용하여 기존 프로필에서 802.1X 네트워크 인증으로 CRYOSTAT 연결 구성

nmcli 유틸리티를 사용하여 네트워크에 자신을 인증하도록 클라이언트를 구성할 수 있습니다. 예를 들어, wlp1s0 이라는 기존 NetworkManagerECDHE 연결 프로필에서 Microsoft Challenge-Handshake Authentication Protocol 버전 2(MSCHAPv2)를 사용하여 PEAP(Protected Extensible Authentication Protocol) 인증을 구성할 수 있습니다.

사전 요구 사항

- 네트워크에는 802.1X 네트워크 인증이 있어야 합니다.
- NetworkManager에 FlexVolume 연결 프로필이 존재하며 유효한 IP 구성이 있습니다.
- 클라이언트가 인증 기관의 인증서를 확인해야 하는 경우 CA(인증 기관) 인증서는 /etc/pki/ca-trust/source/anchors/ 디렉터리에 저장해야 합니다.
- wpa_supplicant 패키지가 설치되어 있어야 합니다.

절차

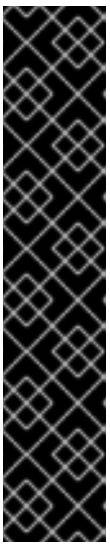
1. 10.0.0.1 보안 모드를 wpa-eap 으로 설정하고, EAP(Extensible Authentication Protocol)를 peap 으로, 내부 인증 프로토콜을 mschapv2 로, 사용자 이름을 설정합니다.

```
# 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 명령으로 **wireless-security.key-mgmt,802-1x.eap,802-1x.phase2-auth, 802-1x.identity** 매개변수를 설정해야 합니다.

2. 선택적으로 구성에 암호를 저장합니다.

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



중요

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

보안을 늘리려면 **802-1x.password-flags** 매개변수를 **0x1** 로 설정합니다. 이 설정을 사용하여 **GNOME** 데스크탑 환경 또는 **nm-ECDHEt** 가 실행 중인 서버에서 **NetworkManager**는 이러한 서비스에서 암호를 검색합니다. 다른 경우에는 **NetworkManager**에서 암호를 묻는 메시지를 표시합니다.

3. 클라이언트가 **authenticator**의 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 클라이언트는 인증자의 인증을 확인해야 합니다.

4. 연결 프로필을 활성화합니다.

```
# nmcli connection up wlp1s0
```

검증

- 네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

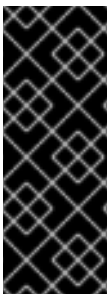
추가 리소스

- ECDHE 연결 관리
- **nm-settings(5)** 도움말 페이지
- **nmcli(1)** 도움말 페이지

10.9. 무선 규제 도메인 수동 설정

RHEL에서 **udev** 규칙은 **setregdomain** 유틸리티를 실행하여 무선 규제 도메인을 설정합니다. 그런 다음 유틸리티에서 이 정보를 커널에 제공합니다.

기본적으로 **setregdomain** 은 국가 코드를 자동으로 결정하려고 합니다. 이 경우 무선 규제 도메인 설정이 잘못될 수 있습니다. 이 문제를 해결하려면 국가 코드를 수동으로 설정할 수 있습니다.



중요

규제 도메인을 수동으로 설정하면 자동 탐지가 비활성화됩니다. 따라서 나중에 다른 국가에서 컴퓨터를 사용하는 경우 이전에 구성된 설정이 더 이상 올바르지 않을 수 있습니다. 이 경우 **/etc/sysconfig/regdomain** 파일을 제거하여 자동 탐지로 다시 전환하거나 이 절차를 사용하여 규제 도메인 설정을 수동으로 업데이트합니다.

절차

1. 선택 사항: 현재 규제 도메인 설정을 표시합니다.

```
# iw reg get
global
country US: DFS-FCC
...
```

2. 다음 콘텐츠를 사용하여 **/etc/sysconfig/regdomain** 파일을 만듭니다.

```
COUNTRY=<country_code>
```

COUNTRY 변수를 미국 용 **DE** 또는 미국과 같은 **ISO 3166-1 alpha2** 국가 코드로 설정합니다.

3. 규제 도메인 설정:

```
# setregdomain
```

검증

- 규제 도메인 설정을 표시합니다.

```
# iw reg get
global
country DE: DFS-ETSI
...
```

추가 리소스

- **setregdomain(1)** 도움말 페이지
- **IW(8)** 도움말 페이지
- **regulatory.bin(5)** 도움말 페이지
- ISO 3166 Code

11장. RHEL을ECDHE2 또는ECDHE3 개인 액세스 포인트로 구성

ECDHE 장치가 있는 호스트에서 NetworkManager를 사용하여 이 호스트를 액세스 포인트로 구성할 수 있습니다. Wi-FiProtected Access 2 (WPA2) 및 Wi-FiProtected Access 3 (WPA3) 개인 인증 방법을 제공하며 무선 클라이언트는 사전 공유 키 (PSK)를 사용하여 액세스 포인트에 연결하고 RHEL 호스트 및 네트워크에서 서비스를 사용할 수 있습니다.

액세스 포인트를 구성하면 NetworkManager가 자동으로 다음을 수행합니다.

- 클라이언트에 DHCP 및 DNS 서비스를 제공하도록 dnsmasq 서비스를 구성합니다.
- IP 전달 활성화
- nftables 방화벽 규칙을ECDHE 장치의 트래픽을 마스커레이드에 추가하고 IP 전달을 구성합니다.

사전 요구 사항

- ECDHE 장치는 액세스 포인트 모드에서 실행을 지원합니다.
- BOOM 장치는 사용 중이 아닙니다.
- 호스트는 인터넷에 액세스할 수 있습니다.

절차

1. CHAP 장치를 나열하여 액세스 포인트를 제공해야 하는 장치를 식별합니다.

```
# nmcli device status | grep wifi
wlp0s20f3    wifi   disconnected    --
```

2. 장치가 액세스 포인트 모드를 지원하는지 확인합니다.

```
# nmcli -f WIFI-PROPERTIES.AP device show wlp0s20f3
WIFI-PROPERTIES.AP:     yes
```

CHAP 장치를 액세스 포인트로 사용하려면 장치가 이 기능을 지원해야 합니다.

3. **dnsmasq** 및 **NetworkManager-wifi** 패키지를 설치합니다.

```
# yum install dnsmasq NetworkManager-wifi
```

NetworkManager는 **dnsmasq** 서비스를 사용하여 액세스 포인트의 클라이언트에 **DHCP** 및 **DNS** 서비스를 제공합니다.

4. 초기 액세스 포인트 구성을 생성합니다.

```
# nmcli device wifi hotspot ifname wlp0s20f3 con-name Example-Hotspot ssid
Example-Hotspot password "password"
```

이 명령을 수행하면 **wlp0s20f3** 장치에서 액세스 포인트에 대한 연결 프로파일이 생성되어 **ECDHE2** 및**ECDHE3** 개인 인증을 제공합니다. 무선 네트워크의 이름인 **SSID(Service Set Identifier)**는 **Example-Hotspot** 이며 사전 공유 키 암호 를 사용합니다.

5. 선택 사항: **improve3**만 지원하도록 액세스 포인트를 구성합니다.

```
# nmcli connection modify Example-Hotspot 802-11-wireless-security.key-mgmt sae
```

6. 기본적으로 **NetworkManager**는**ECDHE** 장치에 **IP** 주소 **10.42.0.1** 을 사용하고 나머지 **10.42.0.0/24** 서브넷의 **IP** 주소를 클라이언트에 할당합니다. 다른 서브넷과 **IP** 주소를 구성하려면 다음을 입력합니다.

```
# nmcli connection modify Example-Hotspot ipv4.addresses 192.0.2.254/24
```

설정한 **IP** 주소(이 경우 **192.0.2.254**) 는 **NetworkManager**가**ECDHE** 장치에 할당하는 주소입니다. 클라이언트는 이 **IP** 주소를 기본 게이트웨이 및 **DNS** 서버로 사용합니다.

7. 연결 프로필을 활성화합니다.

```
# nmcli connection up Example-Hotspot
```

검증

1. 서버에서 다음을 수행합니다.

   a. NetworkManager가 dnsmasq 서비스를 시작하고 서비스가 포트 67(DHCP) 및 53(DNS)에서 수신 대기하는지 확인합니다.

```
# ss -tulpn | egrep ":53|:67"
udp   UNCONN 0  0   10.42.0.1:53    0.0.0.0:*    users:(("dnsmasq",pid=55905,fd=6))
udp   UNCONN 0  0     0.0.0.0:67    0.0.0.0:*    users:(("dnsmasq",pid=55905,fd=4))
tcp   LISTEN 0  32  10.42.0.1:53    0.0.0.0:*    users:(("dnsmasq",pid=55905,fd=7))
```

   b. nftables 규칙 세트를 표시하여 NetworkManager가 10.42.0.0/24 서브넷의 트래픽에 대해 전달 및 마스커레이딩을 활성화했는지 확인합니다.

```
# nft list ruleset
table ip nm-shared-wlp0s20f3 {
   chain nat_postrouting {
      type nat hook postrouting priority srcnat; policy accept;
      ip saddr 10.42.0.0/24 ip daddr != 10.42.0.0/24 masquerade
   }

   chain filter_forward {
      type filter hook forward priority filter; policy accept;
      ip daddr 10.42.0.0/24 oifname "wlp0s20f3" ct state { established, related }
accept
      ip saddr 10.42.0.0/24 iifname "wlp0s20f3" accept
      iifname "wlp0s20f3" oifname "wlp0s20f3" accept
      iifname "wlp0s20f3" reject
      oifname "wlp0s20f3" reject
   }
}
```

2. ECDHE 어댑터가 있는 클라이언트에서 다음을 수행합니다.

   a. 사용 가능한 네트워크 목록을 표시합니다.

```
# nmcli device wifi
IN-USE  BSSID              SSID            MODE   CHAN  RATE     SIGNAL  BARS
SECURITY
      00:53:00:88:29:04  Example-Hotspot  Infra  11    130 Mbit/s  62      ▂▄▆_  WPA3
...
```

   b. Example-Hotspot 무선 네트워크에 연결합니다. ManagingSync-Fi 연결 관리 단원을 참조하십시오.

c. 원격 네트워크 또는 인터넷에서 호스트를 **ping**하여 연결이 작동하는지 확인합니다.

```
# ping -c 3 www.redhat.com
```

추가 리소스

- **nm-settings(5)** 도움말 페이지

12장. MACSEC을 사용하여 동일한 물리적 네트워크에서 계층 2 트래픽 암호화

MACsec을 사용하여 두 장치 간 통신을 보호할 수 있습니다(포인트 간 통신). 예를 들어, 브랜치 사무실은 중앙 사무실과 Metro-Ethernet 연결을 통해 연결되어 있으며, 사무실을 연결하는 두 호스트에서 MACsec을 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 UDP와 같은 IP를 통한 트래픽

MACsec은 기본적으로 GCM-AES-128 알고리즘을 사용하여 LAN의 모든 트래픽을 암호화하고 인증하며, 사전 공유 키를 사용하여 참가자 호스트 간 연결을 설정합니다. 사전 공유 키를 변경하려면 MACsec을 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.

MACsec에는 특별한 하드웨어가 필요하지 않습니다. 예를 들어 호스트와 스위치 간의 트래픽만 암호화하려는 경우를 제외하고 모든 스위치를 사용할 수 있습니다. 이 시나리오에서는 스위치에서 MACsec도 지원해야 합니다.

즉, MACsec을 구성하는 일반적인 두 가지 방법이 있습니다.

- 호스트 대 호스트 및

- 다른 호스트로 전환한 후 다른 호스트로 전환하는 호스트

중요

MACsec은 동일한 (물리적 또는 가상) **LAN**에 있는 호스트 간에만 사용할 수 있습니다.

12.1. NMCLI를 사용하여 MACSEC 연결 구성

nmcli 유틸리티를 사용하여 **MACsec**을 사용하도록 이더넷 인터페이스를 구성할 수 있습니다. 예를 들어 이더넷을 통해 연결된 두 호스트 간에 **MACsec** 연결을 생성할 수 있습니다.

절차

1. **MACsec**을 구성하는 첫 번째 호스트에서 다음을 수행합니다.

   - 사전 공유 키에 대한 연결 키(**CAK**) 및 연결 연결 키 이름(**CKN**)을 만듭니다.

     a. **16**바이트 **16**바이트 **16**진수 **CAK**를 생성합니다.

     ```
     # dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
     50b71a8ef0bd5751ea76de6d6c98c03a
     ```

     b. **32**바이트 **16**진수 **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
   ```

macsec.mka-cak 및 **macsec.mka-ckn** 매개변수의 이전 단계에서 생성된 **CAK** 및 **CKN**을 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

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 보호 유형에 대한 개별 카운터 표시: 무결성 전용(암호화 전용) 및 암호화(암호화)

```
# ip -s macsec show
```

12.2. 추가 리소스

- MACsec: 네트워크 트래픽 블로그 암호화에 다른 솔루션.

13장. IPVLAN 시작하기

IPVLAN은 컨테이너 환경에서 호스트 네트워크에 액세스하는 데 사용할 수 있는 가상 네트워크 장치의 드라이버입니다. IPVLAN은 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.

13.1. IPVLAN 모드

IPVLAN에는 다음 모드를 사용할 수 있습니다.

- L2 모드

  IPVLAN L2 모드에서 가상 장치는 ARP(주소 확인 프로토콜) 요청을 수신하고 응답합니다. netfilter 프레임워크는 가상 장치를 소유한 컨테이너 내에서만 실행됩니다. 컨테이너화된 트래픽의 default 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. 수신 트래픽이 L2 모드 와 동일한 방식으로 스레드되는 반면 관련 컨테이너의 송신 트래픽은 기본 네임스페이스의 netfilter POSTROUTING 및 OUTPUT 체인에 배치됩니다. L3 모드를 사용하면 좋은 제어가 가능하지만 네트워크 트래픽 성능이 저하됩니다.

- L3S 모드

  L3S 모드에서 가상 장치는 관련 컨테이너의 송신 및 수신 트래픽이 기본 네임스페이스의 netfilter 체인에 배치된다는 점을 제외하고 L3 모드에서 와 동일한 방식으로 처리합니다. L3S 모드는 L3 모드 와 유사한 방식으로 작동하지만 네트워크 제어 기능이 향상됩니다.

참고

L3 및 L3 S 모드의 경우 IPVLAN 가상 장치는 브로드캐스트 및 멀티캐스트 트래픽을 수신하지 않습니다.

13.2. IPVLAN 및 MACVLAN 비교

다음 표에서는 **MACVLAN**과 **IPVLAN**의 주요 차이점을 보여줍니다.

MACVLAN	IPVLAN
각 MACVLAN 장치에 대해 MAC 주소를 사용합니다. 스위치가 MAC 테이블에 저장할 수 있는 최대 MAC 주소 수에 도달하면 연결이 끊어질 수 있습니다.	IPVLAN 장치 수를 제한하지 않는 단일 MAC 주소를 사용합니다.
글로벌 네임스페이스의 Netfilter 규칙은 하위 네임스페이스의 MACVLAN 장치 또는 MACVLAN 장치에 대한 트래픽에 영향을 미칠 수 없습니다.	**L3 모드 및 L3S 모드에서** IPVLAN 장치에 대한 트래픽 또는 트래픽을 제어할 수 있습니다.

IPVLAN 및 **MACVLAN** 모두 수준의 캡슐화가 필요하지 않습니다.

13.3. 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.

예 13.1. 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 **NIC**의 **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.

14장. 특정 장치를 무시하도록 NETWORKMANAGER 구성

기본적으로 NetworkManager는 루프백(lo) 장치를 제외한 모든 장치를 관리합니다. 그러나 NetworkManager를 관리되지 않아 특정 장치를 무시할 수 있습니다. 이 설정을 사용하면 스크립트를 사용하여 이러한 장치를 수동으로 관리할 수 있습니다.

14.1. NETWORKMANAGER에서 UNMANAGED로 영구적으로 장치 구성

인터페이스 이름, MAC 주소 또는 장치 유형과 같은 여러 기준에 따라 장치를 관리되지 않음으로 영구적으로 구성할 수 있습니다.

네트워크 장치를 비관리 형으로 일시적으로 구성하려면 NetworkManager에서 장치를 Unmanaged로 구성하는 것을 참조하십시오.

절차

1. 선택 사항: Unmanaged 로 설정하려는 장치 또는 MAC 주소를 식별하는 장치 목록을 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 52:54:00:74:79:56 brd ff:ff:ff:ff:ff:ff
...
```

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

- 특정 인터페이스를 Unmanaged로 구성하려면 다음을 추가합니다.

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

- 특정 MAC 주소를 Unmanaged로 사용하여 장치를 구성하려면 다음을 추가합니다.

```
[keyfile]
unmanaged-devices=mac:52:54:00:74:79:56
```

- 특정 유형의 모든 장치를 관리되지 않음으로 구성하려면 다음을 추가합니다.

```
[keyfile]
unmanaged-devices=type:ethernet
```

- 여러 장치를 **Unmanaged**로 설정하려면 **unmanaged-devices** 매개변수의 항목을 분리합니다. 예를 들면 다음과 같습니다.

```
[keyfile]
unmanaged-devices=interface-name:enp1s0;interface-name:enp7s0
```

3. **NetworkManager** 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

검증

- 장치 목록을 표시합니다.

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

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

문제 해결

- 장치가 관리되지 않음으로 표시되지 않으면 **NetworkManager** 구성을 표시합니다.

```
# NetworkManager --print-config
...
[keyfile]
unmanaged-devices=interface-name:enp1s0
...
```

출력이 구성한 설정과 일치하지 않는 경우 우선 순위가 높은 구성 파일이 설정을 재정의하지 않아야 합니다. **NetworkManager**가 여러 구성 파일을 병합하는 방법에 대한 자세한 내용은 **NetworkManager.conf(5)** 도움말 페이지를 참조하십시오.

14.2. NETWORKMANAGER에서 일시적으로 관리되지 않는 장치로 구성

장치를 임시로 비관리 형으로 구성할 수 있습니다.

예를 들어 테스트용으로 이 방법을 사용합니다. 네트워크 장치를 관리되지 않은 상태로 영구적으로 구성하려면 NetworkManager에서 장치를 Unmanaged로 구성하는 것을 참조하십시오.

절차

1. 선택 사항: Unmanaged 로 설정할 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

- 장치 목록을 표시합니다.

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

enp1s0 장치 옆에 있는 관리되지 않는 상태는 NetworkManager가 이 장치를 관리하지 않음을 나타냅니다.

추가 리소스

- NetworkManager.conf(5) 도움말 페이지

15장. 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. NetworkManager에서 관리하는 루프백과 같은 장치를 추가로 생성할 수 있으며 비활성 SLIP(Serial Line Internet Protocol) 주소가 로컬 프로그램의 실제 주소처럼 표시되도록 할 수 있습니다.

15.1. NMCLI를 사용하여 IPV4 및 IPV6 주소를 모두 사용하여 더미 인터페이스 만들기

IPv4 및 IPv6 주소와 같은 다양한 설정을 사용하여 더미 인터페이스를 만들 수 있습니다. 인터페이스를 생성한 후 NetworkManager는 기본 public firewalld 영역에 자동으로 할당합니다.

절차

- 정적 IPv4 및 IPv6 주소를 사용하여 dummy0 이라는 더미 인터페이스를 만듭니다.

```
# 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
```

참고

IPv4 및 IPv6 주소 없이 더미 인터페이스를 구성하려면 ipv4.method 및 ipv6.method 매개 변수를 모두 disabled 로 설정합니다. 그렇지 않으면 IP 자동 구성이 실패하고 NetworkManager는 연결을 비활성화하고 장치를 제거합니다.

검증

- 연결 프로필을 나열합니다.

```
# nmcli connection show
NAME          UUID                                  TYPE     DEVICE
dummy-dummy0  aaf6eb56-73e5-4746-9037-eed42caa8a65  dummy    dummy0
```

추가 리소스

- nm-settings(5) 도움말 페이지

16장. 특정 연결에 대해 NETWORKMANAGER를 사용하여 IPV6 비활성화

NetworkManager를 사용하여 네트워크 인터페이스를 관리하는 시스템에서는 네트워크에서 IPv4만 사용하는 경우 IPv6 프로토콜을 비활성화할 수 있습니다. IPv6 를 비활성화하면 NetworkManager는 커널에서 해당 sysctl 값을 자동으로 설정합니다.

참고

커널 튜닝 가능 항목 또는 커널 부팅 매개 변수를 사용하여 IPv6를 비활성화하는 경우 시스템 구성에 추가 고려 사항을 지정해야 합니다. 자세한 내용은 RHEL에서 IPv6 프로토콜을 비활성화하거나 활성화하는 방법 문서를 참조하십시오.

16.1. NMCLI를 사용하여 연결에서 IPV6 비활성화

nmcli 유틸리티를 사용하여 명령줄에서 IPv6 프로토콜을 비활성화할 수 있습니다.

사전 요구 사항

- 시스템은 NetworkManager를 사용하여 네트워크 인터페이스를 관리합니다.

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

1. 장치의 IP 설정을 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 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 값 1 이 포함되어 있는지 확인합니다.

```
# cat /proc/sys/net/ipv6/conf/enp1s0/disable_ipv6
1
```

값 1 은 장치에 대해 IPv6 가 비활성화되어 있음을 나타냅니다.

17장. 호스트 이름 변경

시스템의 호스트 이름은 시스템 자체의 이름입니다. RHEL을 설치할 때 이름을 설정할 수 있으며 나중에 변경할 수 있습니다.

17.1. NMCLI를 사용하여 호스트 이름 변경

nmcli 유틸리티를 사용하여 시스템 호스트 이름을 업데이트할 수 있습니다. 다른 유틸리티에서는 정적 또는 영구 호스트 이름과 같은 다른 용어를 사용할 수 있습니다.

절차

1. 선택 사항: 현재 호스트 이름 설정을 표시합니다.

```
# nmcli general hostname
old-hostname.example.com
```

2. 새 호스트 이름을 설정합니다.

```
# nmcli general hostname new-hostname.example.com
```

3. NetworkManager는 systemd-hostnamed 를 자동으로 다시 시작하여 새 이름을 활성화합니다. 변경 사항을 적용하려면 호스트를 재부팅합니다.

```
# reboot
```

또는 호스트 이름을 사용하는 서비스를 알고 있는 경우 다음을 수행합니다.

a. 서비스가 시작될 때만 호스트 이름을 읽는 모든 서비스를 다시 시작합니다.

```
# systemctl restart <service_name>
```

b. 변경 사항을 적용하려면 활성 쉘 사용자가 다시 로그인해야 합니다.

검증

- **hostname**을 표시합니다.

```
# nmcli general hostname
new-hostname.example.com
```

17.2. HOSTNAMECTL을 사용하여 호스트 이름 변경

hostnamectl 유틸리티를 사용하여 호스트 이름을 업데이트할 수 있습니다. 기본적으로 이 유틸리티는 다음과 같은 호스트 이름 유형을 설정합니다.

- 정적 호스트 이름: **/etc/hostname** 파일에 저장 일반적으로 서비스는 이 이름을 호스트 이름으로 사용합니다.
- 호스트 이름: 데이터 센터 **A**의 **Proxy** 서버와 같은 설명적 이름입니다.
- 일시적인 호스트 이름: 일반적으로 네트워크 구성에서 수신되는 대체 값입니다.

절차

1. 선택 사항: 현재 호스트 이름 설정을 표시합니다.

```
# hostnamectl status --static
old-hostname.example.com
```

2. 새 호스트 이름을 설정합니다.

```
# hostnamectl set-hostname new-hostname.example.com
```

이 명령은 정적, 매우, 일시적인 호스트 이름을 새 값으로 설정합니다. 특정 유형만 설정하려면 **--static,--pretty** 또는 **--transient** 옵션을 명령에 전달합니다.

3. **hostnamectl** 유틸리티에서 **systemd-hostnamed** 를 자동으로 다시 시작하여 새 이름을 활성화합니다. 변경 사항을 적용하려면 호스트를 재부팅합니다.

```
# reboot
```

또는 호스트 이름을 사용하는 서비스를 알고 있는 경우 다음을 수행합니다.

a. 서비스가 시작될 때만 호스트 이름을 읽는 모든 서비스를 다시 시작합니다.

```
# systemctl restart <service_name>
```

b. 변경 사항을 적용하려면 활성 쉘 사용자가 다시 로그인해야 합니다.

검증

- hostname을 표시합니다.

```
# hostnamectl status --static
new-hostname.example.com
```

추가 리소스

- hostnamectl(1)
- systemd-hostnamed.service(8)

18장. NETWORKMANAGER DHCP 설정 구성

NetworkManager는 DHCP와 관련된 다양한 구성 옵션을 제공합니다. 예를 들어 build-in DHCP 클라이언트(기본값) 또는 외부 클라이언트를 사용하도록 NetworkManager를 구성하고 개별 프로필의 DHCP 설정에 영향을 줄 수 있습니다.

18.1. NETWORKMANAGER의 DHCP 클라이언트 변경

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

RHEL은 dhcpcd 를 제공하지 않으므로 NetworkManager는 이 클라이언트를 사용할 수 없습니다.

절차

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

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

dhcp 매개변수를 internal (기본값) 또는 dhclient 로 설정할 수 있습니다.

2. dhcp 매개변수를 dhclient 로 설정하면 dhcp-client 패키지를 설치합니다.

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

3. NetworkManager를 다시 시작하십시오.

```
# systemctl restart NetworkManager
```

재시작 시 모든 네트워크 연결이 일시적으로 중단됩니다.

검증

- /var/log/ log 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)** 도움말 페이지

18.2. 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 **45**초 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 **30**초로 설정하려면 다음을 입력합니다.

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

또는 매개 변수를 무한대 로 설정하여 **NetworkManager**가 성공할 때까지 **IP** 주소 요청 및 업데이트 시도를 중지하지 않도록 구성합니다.

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:

- 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 **autoconnect** 속성이 활성화된 경우 **NetworkManager**는 **autoconnect-retries** 속성에 설정된 횟수만큼 연결을 다시 시도합니다. 기본값은 **4** 입니다.
  - 연결이 계속 **DHCP** 주소를 가져올 수 없는 경우 자동 활성화에 실패합니다. **5**분 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

19장. NETWORKMANAGER를 사용하여 DHCLIENT 종료 후크 실행

NetworkManager 디스패치기 스크립트를 사용하여 dhclient 종료 후크를 실행할 수 있습니다.

19.1. NETWORKMANAGER 디스패치 도구 스크립트의 개념

NetworkManager-dispatcher 서비스는 네트워크 이벤트가 발생할 때 알파벳순으로 사용자 제공 스크립트를 실행합니다. 이러한 스크립트는 일반적으로 쉘 스크립트이지만 실행 가능한 스크립트 또는 애플리케이션일 수 있습니다. 예를 들어 디스패치 스크립트 를 사용하여 NetworkManager로 관리할 수 없는 네트워크 관련 설정을 조정할 수 있습니다.

다음 디렉터리에 디스패치 스크립트를 저장할 수 있습니다.

- /etc/NetworkManager/dispatcher.d/: root 사용자가 편집할 수 있는 디스패치 스크립트의 일반 위치입니다.
- /usr/lib/NetworkManager/dispatcher.d/: 사전 배포된 변경 불가능한 디스패처 스크립트의 경우.

보안상의 이유로 NetworkManager-dispatcher 서비스는 다음 조건이 충족되는 경우에만 스크립트를 실행합니다.

- 이 스크립트는 root 사용자가 소유합니다.
- 이 스크립트는 루트 에서만 읽고 쓸 수 있습니다.
- setuid 비트는 스크립트에 설정되지 않습니다.

NetworkManager-dispatcher 서비스는 두 개의 인수를 사용하여 각 스크립트를 실행합니다.

1. 작업이 수행한 장치의 인터페이스 이름입니다.

2. 인터페이스가 활성화된 경우 **up** 과 같은 작업입니다.

NetworkManager(8) 도움말 페이지의 **Dispatcher scripts** 섹션은 스크립트에서 사용할 수 있는 작업 및 환경 변수에 대한 개요를 제공합니다.

NetworkManager-dispatcher 서비스는 한 번에 하나의 스크립트를 실행하지만 기본 **NetworkManager** 프로세스에서 비동기식으로 실행합니다. 스크립트가 대기열에 저장되면 이후 이벤트가 더 이상 사용되지 않는 경우에도 서비스는 항상 해당 스크립트가 실행됩니다. 그러나 **NetworkManager-dispatcher** 서비스는 이전 스크립트가 종료될 때까지 기다리지 않고 즉시 **/etc/NetworkManager/dispatcher.d/no-wait.d/** 의 파일을 참조하는 심볼릭 링크인 스크립트를 실행합니다.

추가 리소스

- **NetworkManager(8)** 매뉴얼 페이지

19.2. DHCLIENT 종료 후크를 실행하는 NETWORKMANAGER 디스패치 스크립트 생성

DHCP 서버가 **IPv4** 주소를 할당하거나 업데이트하면 **NetworkManager**는 **/etc/dhcp/dhclient-exit-hooks.d/** 디렉터리에 저장된 디스패치 스크립트를 실행할 수 있습니다. 그러면 이 디스패치 스크립트는 예를 들어 **dhclient** 종료 후크를 실행할 수 있습니다.

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 만듭니다.

```
#!/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
```

```
    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) 매뉴얼 페이지

20장. /ETC/RESOLV.CONF 파일 수동 구성

기본적으로 NetworkManager는 활성 NetworkManager 연결 프로필의 DNS 설정으로 /etc/resolv.conf 파일을 동적으로 업데이트합니다. 그러나 이 동작을 비활성화하고 /etc/resolv.conf 에서 DNS 설정을 수동으로 구성할 수 있습니다.

참고

또는 /etc/resolv.conf 에 특정 DNS 서버 순서가 필요한 경우 DNS 서버 순서를 참조하십시오.

20.1. NETWORKMANAGER 구성에서 DNS 처리 비활성화

기본적으로 NetworkManager는 /etc/resolv.conf 파일에서 DNS 설정을 관리하고 DNS 서버 순서를 구성할 수 있습니다. 또는 /etc/resolv.conf 에서 DNS 설정을 수동으로 구성하려면 NetworkManager에서 DNS 처리를 비활성화할 수 있습니다.

절차

1. 텍스트 편집기를 사용하여 다음 콘텐츠를 사용하여 root 사용자로 /etc/NetworkManager/conf.d/90-dns-none.conf 파일을 만듭니다.

```
[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 /etc/resolv.conf 에서 Generated by NetworkManager 주석을 제거하여 혼동을 방지합니다.

검증

1. /etc/resolv.conf 파일을 편집하고 구성을 수동으로 업데이트합니다.

2. NetworkManager 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

3. /etc/resolv.conf 파일을 표시합니다.

```
# cat /etc/resolv.conf
```

DNS 처리를 비활성화한 경우 NetworkManager에서 수동으로 구성된 설정을 재정의하지 않았습니다.

문제 해결

- 우선 순위가 높은 다른 구성 파일이 설정을 덮어쓰지 않도록 NetworkManager 구성을 표시합니다.

```
# NetworkManager --print-config
...
dns=none
...
```

추가 리소스

- NetworkManager.conf(5) 도움말 페이지
- NetworkManager를 사용하여 DNS 서버 순서 구성

20.2. DNS 설정을 수동으로 구성하려면 /ETC/RESOLV.CONF를 심볼릭 링크로 교체

기본적으로 NetworkManager는 /etc/resolv.conf 파일에서 DNS 설정을 관리하고 DNS 서버 순서를 구성할 수 있습니다. 또는 /etc/resolv.conf 에서 DNS 설정을 수동으로 구성하려면 NetworkManager에서 DNS 처리를 비활성화할 수 있습니다. 예를 들어, NetworkManager는 /etc/resolv.conf 가 심볼릭 링크인 경우 DNS 구성을 자동으로 업데이트하지 않습니다.

사전 요구 사항

- NetworkManager rc-manager 구성 옵션은 파일로 설정되지 않습니다. 확인하려면 NetworkManager --print-config 명령을 사용합니다.

절차

1. /etc/resolv.conf.ECDHEly-configured 와 같은 파일을 만들고 해당 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.ECDHEly -configured :를 참조하는 심볼릭 링크를 /etc/resolv.conf라는 심볼릭 링크를 만듭니다.

```
# ln -s /etc/resolv.conf.manually-configured /etc/resolv.conf
```

추가 리소스

- resolv.conf(5) man page

- NetworkManager.conf(5) 도움말 페이지

- NetworkManager를 사용하여 DNS 서버 순서 구성

21장. DNS 서버 순서 구성

대부분의 애플리케이션에서는 glibc 라이브러리의 getaddrinfo() 함수를 사용하여 DNS 요청을 확인합니다. 기본적으로 glibc 는 모든 DNS 요청을 /etc/resolv.conf 파일에 지정된 첫 번째 DNS 서버로 보냅니다. 이 서버가 응답하지 않으면 RHEL은 이 파일에서 다음 서버를 사용합니다. NetworkManager를 사용하면 etc/resolv.conf 에서 DNS 서버 순서에 영향을 미칠 수 있습니다.

21.1. NETWORKMANAGER가 /ETC/RESOLV.CONF에서 DNS 서버를 주문하는 방법

NetworkManager는 다음 규칙에 따라 /etc/resolv.conf 파일에서 DNS 서버를 주문합니다.

- 하나의 연결 프로필만 있는 경우 NetworkManager는 해당 연결에 지정된 IPv4 및 IPv6 DNS 서버의 순서를 사용합니다.

- 여러 연결 프로필이 활성화되면 NetworkManager는 DNS 우선 순위 값을 기반으로 DNS 서버를 주문합니다. DNS 우선순위를 설정하면 NetworkManager 동작이 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:

    NetworkManager는 각 연결에서 ipv4.dns-priority 및 ipv6.dns-priority 매개변수를 기반으로 다양한 연결에서 DNS 서버를 주문합니다.

    값을 설정하지 않거나 ipv4.dns-priority 및 ipv6.dns-priority 를 0 으로 설정한 경우 NetworkManager는 글로벌 기본값을 사용합니다. DNS 우선 순위 매개 변수의 기본값 을 참조하십시오.

  - dns=dnsmasq or dns=systemd-resolved:

    이러한 설정 중 하나를 사용하는 경우 NetworkManager는 dnsmasq 에 대한 127.0.0.1 또는 127.0.0.53 을 /etc/resolv.conf 파일의 nameserver 항목으로 설정합니다.

    dnsmasq 및 systemd 확인 서비스 둘 다 NetworkManager 연결에 있는 검색 도메인 세트의 쿼리를 해당 연결에 지정된 DNS 서버로 전달하고 다른 도메인으로 쿼리를 기본 경로와 연결합니다. 여러 연결에 동일한 검색 도메인이 설정된 경우 dnsmasq 및 systemd-resolved forward 쿼리를 우선순위가 가장 낮은 연결에 설정된 DNS 서버로 설정합니다.

DNS 우선순위 매개변수의 기본값

NetworkManager는 연결에 다음과 같은 기본값을 사용합니다.

- **VPN** 연결의 경우 **50**
- 다른 연결의 경우 **100**

유효한 DNS 우선순위 값:

글로벌 기본값 및 연결별 **ipv4.dns-priority** 및 **ipv6.dns-priority** 매개변수를 **-2147483647** 및 **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 페이지
- 다른 도메인에 다른 **DNS** 서버 사용

21.2. NETWORKMANAGER 전체 기본 DNS 서버 우선순위 값 설정

NetworkManager는 연결에 다음과 같은 DNS 우선 순위 기본값을 사용합니다.

- VPN 연결의 경우 50
- 다른 연결의 경우 100

이러한 시스템 전체 기본값을 IPv4 및 IPv6 연결의 사용자 지정 기본값으로 덮어쓸 수 있습니다.

절차

1. /etc/NetworkManager/NetworkManager.conf 파일을 편집합니다.

   a. [connection] 섹션이 없는 경우 다음을 추가합니다.

   ```
   [connection]
   ```

   b. 사용자 지정 기본값을 [connection] 섹션에 추가합니다. 예를 들어 IPv4 및 IPv6의 새 기본값을 200 으로 설정하려면 다음을 추가합니다.

   ```
   ipv4.dns-priority=200
   ipv6.dns-priority=200
   ```

   매개변수를 -2147483647 과 2147483647 사이의 값으로 설정할 수 있습니다. 매개 변수를 0 으로 설정하면 기본 제공 기본값(VPN 연결의 경우50 개, 기타 연결의 경우 100)이 활성화됩니다.

2. NetworkManager 서비스를 다시 로드합니다.

   ```
   # systemctl reload NetworkManager
   ```

추가 리소스

- NetworkManager.conf(5) 도움말 페이지

21.3. NETWORKMANAGER 연결의 DNS 우선 순위 설정

특정 DNS 서버 순서가 필요한 경우 연결 프로필에서 우선순위 값을 설정할 수 있습니다. NetworkManager는 서비스가 /etc/resolv.conf 파일을 생성하거나 업데이트할 때 이러한 값을 사용하여 서버를 정렬합니다.

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.
- /etc/NetworkManager/NetworkManager.conf 파일에 dns 매개 변수가 설정되지 않았거나 매개 변수가 기본값으로 설정되어 있습니다.

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 매개변수를 설정합니다. 예를 들어 두 매개변수를 모두 10 으로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify <connection_name> ipv4.dns-priority 10 ipv6.dns-priority
10
```

3. 선택적으로 다른 연결에 대해 이전 단계를 반복합니다.

4. 업데이트한 연결을 다시 활성화합니다.

```
# nmcli connection up <connection_name>
```

검증

- **/etc/resolv.conf** 파일의 내용을 표시하여 **DNS** 서버 순서가 올바른지 확인합니다.

```
# cat /etc/resolv.conf
```

22장. 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 RHEL은 이 파일에서 다음 서버를 사용합니다. 하나의 DNS 서버가 모든 도메인을 확인할 수 없는 환경에서 관리자는 특정 도메인의 DNS 요청을 선택한 DNS 서버로 보내도록 RHEL을 구성할 수 있습니다.

예를 들어 서버를 VPN(Virtual Private Network)에 연결하고 VPN의 호스트는 example.com 도메인을 사용합니다. 이 경우 다음과 같은 방식으로 DNS 쿼리를 처리하도록 RHEL을 구성할 수 있습니다.

- example.com 에 대한 DNS 요청만 VPN 네트워크의 DNS 서버로 보냅니다.
- 다른 모든 요청을 기본 게이트웨이를 사용하여 연결 프로필에 구성된 DNS 서버로 보냅니다.

22.1. NETWORKMANAGER에서 DNSMASQ를 사용하여 특정 도메인에 대한 DNS 요청을 선택한 DNS 서버로 전송

dnsmasq 인스턴스를 시작하도록 NetworkManager를 구성할 수 있습니다. 그런 다음 이 DNS 캐싱 서버는 루프백 장치의 포트 53 에서 수신 대기합니다. 결과적으로 이 서비스는 네트워크가 아닌 로컬 시스템에서만 연결할 수 있습니다.

이 구성을 통해 NetworkManager는 nameserver 127.0.0.1 항목을 /etc/resolv.conf 파일에 추가하고 dnsmasq 는 DNS 요청을 NetworkManager 연결 프로필에 지정된 해당 DNS 서버로 동적으로 라우팅합니다.

사전 요구 사항

- 시스템에는 여러 개의 NetworkManager 연결이 구성되어 있습니다.
- DNS 서버 및 검색 도메인은 특정 도메인을 확인하는 NetworkManager 연결 프로필에 구성됩니다.

  예를 들어 VPN 연결에 지정된 DNS 서버가 example.com 도메인에 대한 쿼리를 확인하도록 하려면 VPN 연결 프로필에 다음 설정이 포함되어야 합니다.

- example.com을 확인할 수 있는 DNS 서버
- ipv4.dns-search 및 ipv6.dns-search 매개변수에서 example.com 으로 설정된 검색 도메인

- dnsmasq 서비스는 다른 인터페이스에서 수신 대기하도록 구성되지 않은 다음 localhost 에서 수신 대기하도록 구성되어 있지 않습니다.

절차

1. dnsmasq 패키지를 설치합니다.

```
# yum install dnsmasq
```

2. /etc/NetworkManager/NetworkManager.conf 파일을 편집하고 [main] 섹션에 다음 항목을 설정합니다.

```
dns=dnsmasq
```

3. NetworkManager 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

검증

1. 서비스에서 다른 DNS 서버를 사용하는 도메인의 NetworkManager 장치의 systemd 저널을 검색합니다.

```
# journalctl -xeu NetworkManager
...
Jun 02 13:30:17 <client_hostname>_ dnsmasq[5298]: using nameserver
198.51.100.7#53 for domain example.com
...
```

2. tcpdump 패킷 스니퍼를 사용하여 DNS 요청의 올바른 경로를 확인합니다.

a.

tcpdump 패키지를 설치합니다.

```
# yum install tcpdump
```

b.

한 터미널에서 **tcpdump** 를 시작하여 모든 인터페이스에서 **DNS** 트래픽을 캡처합니다.

```
# tcpdump -i any port 53
```

c.

다른 터미널에서 예외와 다른 도메인이 존재하는 도메인의 호스트 이름을 확인합니다. 예를 들면 다음과 같습니다.

```
# host -t A www.example.com
# host -t A www.redhat.com
```

d.

tcpdump 출력에서 **Red Hat Enterprise Linux**가 **example.com** 도메인에 대한 **DNS** 쿼리만 지정된 **DNS** 서버와 해당 인터페이스를 통해 전송하는지 확인합니다.

```
...
13:52:42.234533 IP server.43534 > 198.51.100.7.domain: 50121+ [1au] A?
www.example.com. (33)
...
13:52:57.753235 IP server.40864 > 192.0.2.1.domain: 6906+ A? www.redhat.com.
(33)
...
```

Red Hat Enterprise Linux는 **www.example.com** 의 **DNS** 쿼리를 **198.51.100.7** 의 **DNS** 서버에 전송하고 **www.redhat.com** 에 대한 쿼리를 **192.0.2.1** 로 보냅니다.

문제 해결

1.

/etc/resolv.conf 파일의 **nameserver** 항목이 **127.0.0.1:**을 참조하는지 확인합니다.

```
# cat /etc/resolv.conf
nameserver 127.0.0.1
```

항목이 없는 경우 **/etc/NetworkManager/NetworkManager.conf** 파일에서 **dns** 매개변수를 확인합니다.

2. **dnsmasq** 서비스가 루프백 장치의 포트 **53** 에서 수신 대기하는지 확인합니다.

```
# ss -tulpn | grep "127.0.0.1:53"
udp  UNCONN 0  0    127.0.0.1:53   0.0.0.0:*    users:(("dnsmasq",pid=7340,fd=18))
tcp  LISTEN 0  32   127.0.0.1:53   0.0.0.0:*    users:(("dnsmasq",pid=7340,fd=19))
```

서비스가 **127.0.0.1:53** 에서 수신 대기하지 않으면 **NetworkManager** 장치의 저널 항목을 확인합니다.

```
# journalctl -u NetworkManager
```

22.2. NETWORKMANAGER에서 SYSTEMD-RESOLVED를 사용하여 특정 도메인에 대한 DNS 요청을 선택한 DNS 서버로 전송

systemd-resolved 인스턴스를 시작하도록 **NetworkManager**를 구성할 수 있습니다. 그런 다음 이 **DNS** 스텁 확인기에서 **IP** 주소 **127.0.0. 53** 의 포트 **53**에서 수신 대기합니다. 결과적으로 이 스텁 확인자는 네트워크가 아닌 로컬 시스템에서만 연결할 수 있습니다.

이 구성을 통해 **NetworkManager**는 이름 서버 **127.0.0.53** 항목을 **/etc/resolv.conf** 파일에 추가하고 **systemd-resolved** 는 **DNS** 요청을 **NetworkManager** 연결 프로필에 지정된 해당 **DNS** 서버로 동적으로 라우팅합니다.

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.

지원되는 솔루션은 **NetworkManager**에서 **dnsmasq**를 사용하여 특정 도메인에 대한 **DNS** 요청을 선택한 **DNS** 서버로 보내는 것을 참조하십시오.

사전 요구 사항

- 시스템에는 여러 개의 **NetworkManager** 연결이 구성되어 있습니다.

- **DNS** 서버 및 검색 도메인은 특정 도메인을 확인하는 **NetworkManager** 연결 프로필에 구성됩니다.

  예를 들어 **VPN** 연결에 지정된 **DNS** 서버가 **example.com** 도메인에 대한 쿼리를 확인하도록 하려면 **VPN** 연결 프로필에 다음 설정이 포함되어야 합니다.

  - **example.com**을 확인할 수 있는 **DNS** 서버
  - **ipv4.dns-search** 및 **ipv6.dns-search** 매개변수에서 **example.com** 으로 설정된 검색 도메인

절차

1. **systemd-resolved** 서비스를 활성화하고 시작합니다.

```
# systemctl --now enable systemd-resolved
```

2. **/etc/NetworkManager/NetworkManager.conf** 파일을 편집하고 **[main]** 섹션에 다음 항목을 설정합니다.

```
dns=systemd-resolved
```

3. **NetworkManager** 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

검증

1. 서비스에서 다른 **DNS** 서버를 사용하는 도메인과 **systemd-resolved** 의 사용을 표시합니다.

```
# resolvectl
...
Link 2 (enp1s0)
    Current Scopes: DNS
         Protocols: +DefaultRoute ...
Current DNS Server: 192.0.2.1
```

```
      DNS Servers: 192.0.2.1

Link 3 (tun0)
   Current Scopes: DNS
        Protocols: -DefaultRoute ...
Current DNS Server: 198.51.100.7
       DNS Servers: 198.51.100.7 203.0.113.19
        DNS Domain: example.com
```

systemd-resolved 가 **example.com** 도메인에 다른 **DNS** 서버를 사용하는 것을 출력에서 확인합니다.

2. **tcpdump** 패킷 스니퍼를 사용하여 **DNS** 요청의 올바른 경로를 확인합니다.

   a. **tcpdump** 패키지를 설치합니다.

   ```
   # yum install tcpdump
   ```

   b. 한 터미널에서 **tcpdump** 를 시작하여 모든 인터페이스에서 **DNS** 트래픽을 캡처합니다.

   ```
   # tcpdump -i any port 53
   ```

   c. 다른 터미널에서 예외와 다른 도메인이 존재하는 도메인의 호스트 이름을 확인합니다. 예를 들면 다음과 같습니다.

   ```
   # host -t A www.example.com
   # host -t A www.redhat.com
   ```

   d. **tcpdump** 출력에서 **Red Hat Enterprise Linux**가 **example.com** 도메인에 대한 **DNS** 쿼리만 지정된 **DNS** 서버와 해당 인터페이스를 통해 전송하는지 확인합니다.

   ```
   ...
   13:52:42.234533 IP server.43534 > 198.51.100.7.domain: 50121+ [1au] A?
   www.example.com. (33)
   ...
   13:52:57.753235 IP server.40864 > 192.0.2.1.domain: 6906+ A? www.redhat.com.
   (33)
   ...
   ```

   Red Hat Enterprise Linux는 **www.example.com** 의 **DNS** 쿼리를 **198.51.100.7** 의 **DNS** 서버에 전송하고 **www.redhat.com** 에 대한 쿼리를 **192.0.2.1** 로 보냅니다.

문제 해결

1. **/etc/resolv.conf** 파일의 **nameserver** 항목이 **127.0.0.53** 을 참조하는지 확인합니다.

```
# cat /etc/resolv.conf
nameserver 127.0.0.53
```

항목이 없는 경우 **/etc/NetworkManager/NetworkManager.conf** 파일에서 **dns** 매개변수를 확인합니다.

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))
```

서비스가 **127.0.0.53:53** 에서 수신 대기하지 않으면 **systemd-resolved** 서비스가 실행 중인지 확인합니다.

23장. 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.

23.1. NMCLI를 사용하여 기존 연결에 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 **nmcli** 유틸리티를 사용하여 이전에 생성된 연결에서 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

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.

   IPv4 기본 게이트웨이를 설정하려면 다음을 입력합니다.

   ```
   # nmcli connection modify <connection_name> ipv4.gateway
   "<IPv4_gateway_address>"
   ```

   IPv6 기본 게이트웨이를 설정하려면 다음을 입력합니다.

   ```
   # nmcli connection modify <connection_name> ipv6.gateway
   "<IPv6_gateway_address>"
   ```

2. 변경 사항을 적용하려면 네트워크 연결을 다시 시작하십시오.

   ```
   # nmcli connection up <connection_name>
   ```

주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

검증

- 경로가 활성 상태인지 확인합니다.

a. IPv4 기본 게이트웨이를 표시하려면 다음을 입력합니다.

```
# ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

b. IPv6 기본 게이트웨이를 표시하려면 다음을 입력합니다.

```
# ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

23.2. NMCLI 대화형 모드를 사용하여 기존 연결에 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 nmcli 유틸리티의 대화형 모드를 사용하여 이전에 생성된 연결에 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

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 대해 **nmcli** 대화형 모드를 엽니다.

```
# nmcli connection edit <connection_name>
```

2. 기본 게이트웨이 설정

   IPv4 기본 게이트웨이를 설정하려면 다음을 입력합니다.

```
nmcli> set ipv4.gateway "<IPv4_gateway_address>"
```

   IPv6 기본 게이트웨이를 설정하려면 다음을 입력합니다.

```
nmcli> set ipv6.gateway "<IPv6_gateway_address>"
```

3. 선택적으로 기본 게이트웨이가 올바르게 설정되었는지 확인합니다.

```
nmcli> print
...
ipv4.gateway:          <IPv4_gateway_address>
...
ipv6.gateway:          <IPv6_gateway_address>
...
```

4. 설정을 저장합니다.

```
nmcli> save persistent
```

5. 변경 사항을 적용하려면 네트워크 연결을 다시 시작하십시오.

```
nmcli> activate <connection_name>
```

주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

6. **nmcli** 대화형 모드를 종료합니다.

```
nmcli> quit
```

검증

- 경로가 활성 상태인지 확인합니다.

a. **IPv4** 기본 게이트웨이를 표시하려면 다음을 입력합니다.

```
# ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

b. **IPv6** 기본 게이트웨이를 표시하려면 다음을 입력합니다.

```
# ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

23.3. NM-CONNECTION-EDITOR를 사용하여 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 **nm-connection-editor** 애플리케이션을 사용하여 이전에 생성한 연결에서 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

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 **192.0.2.1** 연결에서 기본 게이트웨이의 **IPv4** 주소를 설정하려면 다음을 수행합니다.

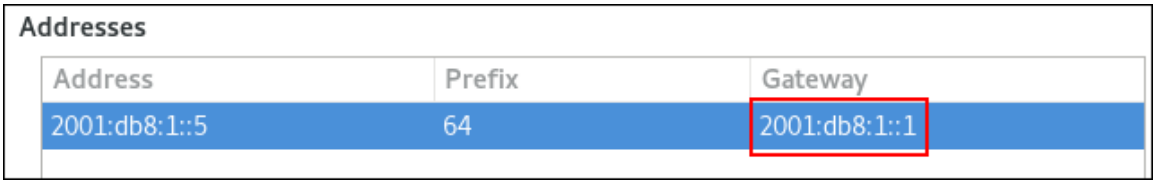
   a. **IPv4 Settings** 탭을 엽니다.

   b. 게이트웨이 주소가 속하는 **IP** 범위 옆에 있는 **gateway** 필드에 주소를 입력합니다.

4. **IPv6** 기본 게이트웨이를 설정합니다. 예를 들어 연결에서 기본 게이트웨이의 **IPv6** 주소를 **2001:db8:1::1** 로 설정하려면 다음을 실행합니다.

   a. **IPv6** 탭을 엽니다.

   b. 게이트웨이 주소가 속하는 **IP** 범위 옆에 있는 **gateway** 필드에 주소를 입력합니다.



-

5. OK(확인)를 클릭합니다.

6. 저장을 클릭합니다.

7. 변경 사항을 적용하려면 네트워크 연결을 다시 시작합니다. 예를 들어 명령줄을 사용하여 *예제* 연결을 다시 시작하려면 다음을 수행합니다.

```
# nmcli connection up example
```

주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

8. 선택적으로 경로가 활성 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

23.4. CONTROL-CENTER를 사용하여 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 control-center 애플리

케이션을 사용하여 이전에 만든 연결에서 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

사전 요구 사항

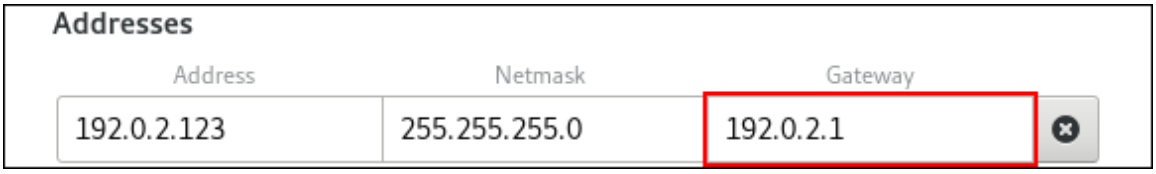
- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.
- 연결의 네트워크 구성은 control-center 애플리케이션에서 열려 있습니다.

절차

1. IPv4 기본 게이트웨이를 설정합니다. 예를 들어 192.0.2.1 연결에서 기본 게이트웨이의 IPv4 주소를 설정하려면 다음을 수행합니다.

   a. IPv4 탭을 엽니다.

   b. 게이트웨이 주소가 속하는 IP 범위 옆에 있는 gateway 필드에 주소를 입력합니다.



2. IPv6 기본 게이트웨이를 설정합니다. 예를 들어 연결에서 기본 게이트웨이의 IPv6 주소를 2001:db8:1::1 로 설정하려면 다음을 실행합니다.

   a. IPv6 탭을 엽니다.

   b. 게이트웨이 주소가 속하는 IP 범위 옆에 있는 gateway 필드에 주소를 입력합니다.

3. **Apply**(적용)를 클릭합니다.

4. 네트워크 창으로 돌아가 연결의 버튼을 전환하여 연결을 비활성화 및 다시 활성화한 후 변경 사항을 적용하려면 연결을 다시 시작합니다.

주의

현재 이 네트워크 연결을 사용하는 모든 연결은 다시 시작하는 동안 일시적으로 중단됩니다.

5. 선택적으로 경로가 활성 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-center를 사용하여 이더넷 연결 구성

23.5. NMSTATECTL을 사용하여 기존 연결에 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 **nmstatectl** 유틸리티를 사용하여 이전에 생성된 연결에서 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 기본 게이트웨이를 설정합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 nmstatectl 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

사전 요구 사항

- 기본 게이트웨이를 설정할 연결에는 하나 이상의 고정 IP 주소를 구성해야 합니다.
- enp1s0 인터페이스가 구성되고 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
```

이러한 설정은 192.0.2.1 을 기본 게이트웨이로 정의하고 기본 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.

2. 시스템에 설정을 적용합니다.

```
# nmstatectl apply ~/set-default-gateway.yml
```

추가 리소스

- nmstatectl(8) man page
- /usr/share/doc/nmstate/examples/ 디렉터리

23.6. 네트워크 RHEL 시스템 역할을 사용하여 기존 연결에 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 네트워크 RHEL 시스템 역할을 사용하여 기본 게이트웨이를 설정하여 이전에 생성된 연결에 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

중요

네트워크 RHEL 시스템 역할을 사용하는 플레이를 실행하고 설정 값이 플레이에 지정된 값과 일치하지 않으면 역할은 동일한 이름의 기존 연결 프로필을 재정의합니다. 이러한 값을 기본값으로 재설정하지 않으려면 구성이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다.

이미 존재하는지 여부에 따라 절차는 다음 설정으로 enp1s0 연결 프로필을 생성하거나 업데이트합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 198.51.100.20
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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

- 제어 노드와 관리형 노드가 준비되어 있습니다.

- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP and default gateway
      ansible.builtin.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
```

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

23.7. 레거시 네트워크 스크립트를 사용할 때 기존 연결에서 기본 게이트웨이 설정

대부분의 경우 관리자는 연결을 만들 때 기본 게이트웨이를 설정합니다. 그러나 레거시 네트워크 스크립트를 사용할 때 이전에 생성된 연결에서 기본 게이트웨이 설정을 설정하거나 업데이트할 수도 있습니다.

사전 요구 사항

- NetworkManager 패키지가 설치되지 않았거나 NetworkManager 서비스가 비활성화되어 있습니다.
- network-scripts 패키지가 설치되어 있습니다.

절차

1. /etc/sysconfig/network-scripts/ifcfg-enp1s0 파일의 GATEWAY 매개변수를 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.

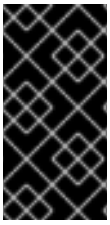
```
# systemctl restart network
```

23.8. 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 두 개의 이더넷 연결이 있는 경우 **NetworkManager**는 경로에 있는 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
이더넷	100
MACsec	125
InfiniBand	150
본딩	300
팀	350
VLAN	400
브릿지	425

연결 유형	기본 메트릭 값
TUN	450
Wi-Fi	600
IP 터널	675

추가 리소스

- 대체 경로를 정의하도록 정책 기반 라우팅 구성

23.9. 기본 게이트웨이를 제공하기 위해 특정 프로필을 사용하지 않도록 NETWORKMANAGER 구성

NetworkManager가 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 프로필에서 해당 프로토콜에 대한 기본 게이트웨이의 **IP** 주소가 자동으로 제거됩니다.

2. 연결을 활성화합니다.

```
# nmcli connection up <connection_name>
```

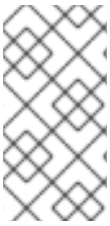
검증

-

ip -4 route 및 ip -6 route 명령을 사용하여 RHEL이 IPv4 및 IPv6 프로토콜의 기본 경로에 네트워크 인터페이스를 사용하지 않는지 확인합니다.

23.10. 여러 기본 게이트웨이로 인해 예기치 않은 라우팅 동작 수정

Multipath TCP를 사용하는 경우와 같이 호스트에 여러 기본 게이트웨이가 필요한 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 NetworkManager를 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
...
```

default 로 시작하는 항목은 기본 경로를 나타냅니다. **dev** 옆에 표시되는 이러한 항목의 인터페이스 이름을 기록해 둡니다.

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:            192.0.2.1
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 프로필에는 기본 게이트웨이가 설정됩니다. 로컬 네트워크에서 기본 게이트웨이는 일반적으로 인터넷에 더 가까운 호스트이므로 이 절차의 나머지 부분에서는 **Corporate-LAN** 의 기본 게이트웨이가 잘못되었다고 가정합니다.

3. **NetworkManager**가 **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 프로필에서 해당 프로토콜에 대한 기본 게이트웨이의 **IP** 주소가 자동으로 제거됩니다.

4. **Corporate-LAN** 연결을 활성화합니다.

```
# nmcli connection up Corporate-LAN
```

검증

- **IPv4** 및 **IPv6** 라우팅 테이블을 표시하고 각 프로토콜에 하나의 기본 게이트웨이만 사용할 수 있는지 확인합니다.

- IPv4의 경우 다음을 입력합니다.

```
# ip -4 route
default via 192.0.2.1 dev enp1s0 proto static metric 101
...
```

- IPv6의 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

24장. 정적 경로 구성

라우팅을 사용하면 상호 연결된 네트워크 간에 트래픽을 보내고 수신할 수 있습니다. 대규모 환경에서는 일반적으로 관리자가 라우터가 다른 라우터를 동적으로 확인할 수 있도록 서비스를 구성합니다. 소규모 환경에서는 관리자가 트래픽이 하나의 네트워크에서 다음 네트워크로 도달할 수 있도록 정적 경로를 구성하는 경우가 많습니다.

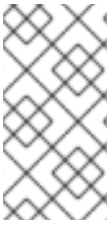
이러한 조건이 모두 적용되는 경우 여러 네트워크 간에 작동하는 통신을 달성하기 위해 정적 경로가 필요합니다.

- 트래픽은 여러 네트워크를 전달해야 합니다.
- 기본 게이트웨이를 통한 전용 트래픽 흐름은 충분하지 않습니다.

정적 경로 섹션이 필요한 네트워크의 예는 시나리오와 정적 경로를 구성하지 않을 때 다른 네트워크 간에 트래픽이 이동하는 방법을 설명합니다.

24.1. 정적 경로가 필요한 네트워크의 예

일부 IP 네트워크가 하나의 라우터를 통해 직접 연결되어 있지 않기 때문에 이 예제의 정적 경로가 필요합니다. 정적 경로가 없으면 일부 네트워크가 서로 통신할 수 없습니다. 또한 일부 네트워크의 트래픽은 한 방향으로만 이동합니다.



참고

이 예제의 네트워크 토폴로지는 인위적이며 정적 라우팅 개념을 설명하는 데만 사용됩니다. 프로덕션 환경에서는 권장 토폴로지가 아닙니다.

이 예에서 모든 네트워크 간에 작동하는 통신의 경우 다음 홉 라우터 2 (203.0.113.10)를 사용하여 Raleigh(198.51.100.0/24)로 정적 경로를 구성하십시오. 다음 홉의 IP 주소는 데이터 센터 네트워크의 라우터 2 (203.0.113.0/24) 중 하나입니다.

다음과 같이 정적 경로를 구성할 수 있습니다.

- 간소화된 구성의 경우 이 정적 경로를 라우터 1에서만 설정합니다. 그러나 데이터 센터

(203.0.113.0/24)의 호스트는 트래픽을 라우터 1을 통해 항상 Raleigh(198.51.100.0/24)로 전송하므로 라우터 1의 트래픽이 증가합니다.

- 보다 복잡한 구성의 경우 데이터 센터의 모든 호스트에서 이 정적 경로를 구성합니다 (203.0.113.0/24). 그런 다음 이 서브넷의 모든 호스트는 Raleigh(198.51.100.0/24)에 더 가까운 라우터 2(203.0.113.10)로 직접 트래픽을 보냅니다.

트래픽이 이동하는 네트워크 간 자세한 내용은 다이어그램 아래의 설명을 참조하십시오.

필요한 정적 경로가 구성되지 않은 경우 다음은 통신이 작동하는 상황 및 그렇지 않은 경우입니다.

- 수행 네트워크 호스트 (192.0.2.0/24)
  - 동일한 서브넷의 다른 호스트와 직접 연결되므로 통신할 수 있습니다.

- 라우터 1이 192.0.2. 네트워크(192.0.2.0/24)에 있고 인터넷으로 이어지는 기본 게이트웨이가 있기 때문에 인터넷과 통신할 수 있습니다.

- 라우터 1은ECDHE(192.0.2.0/24) 및 데이터 센터(203.0.113.0/24) 네트워크 모두에 인터페이스가 있기 때문에 데이터 센터 네트워크(203.0.113.0/24)와 통신할 수 있습니다.

- 라우터 1에는 이 네트워크에 인터페이스가 없기 때문에 Raleigh 네트워크(198.51.100.0/24)와 통신할 수 없습니다. 따라서 라우터 1은 트래픽을 자체 기본 게이트웨이(인터넷)로 보냅니다.

- 데이터 센터 네트워크의 호스트 (203.0.113.0/24)

  - 동일한 서브넷의 다른 호스트와 직접 연결되므로 통신할 수 있습니다.

  - 기본 게이트웨이가 Router 1로 설정되어 있으므로 인터넷과 통신할 수 있으며 Router 1에는 네트워크, 데이터 센터(203.0.113.0/24) 및 인터넷 모두에 인터페이스가 있기 때문입니다.

  - 기본 게이트웨이가 라우터 1로 설정되어 있고 라우터 1은 데이터 센터(203.0.113.0/24) 및 SRV(192.0.2.0/24) 네트워크 둘 다에 인터페이스가 있기 때문에 수행 네트워크(192.0.2.0/24)와 통신할 수 있습니다.

  - 데이터 센터 네트워크에 이 네트워크에 인터페이스가 없기 때문에 Raleigh 네트워크(198.51.100.0/24)와 통신할 수 없습니다. 따라서 데이터 센터(203.0.113.0/24)의 호스트는 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(internet)로 보냅니다.

- Raleigh 네트워크의 호스트(198.51.100.0/24)

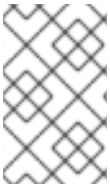
  - 동일한 서브넷의 다른 호스트와 직접 연결되므로 통신할 수 있습니다.

  - 인터넷상의 호스트와 통신할 수 없습니다. 라우터 2는 기본 게이트웨이 설정으로 인해 트래픽을 라우터 1로 보냅니다. 라우터 1의 실제 동작은 역방향 경로 필터(rp_filter) 시스템 제어(sysctl) 설정에 따라 달라집니다. 기본적으로 RHEL에서 라우터 1은 인터넷으로 라우팅

하는 대신 발신 트래픽을 삭제합니다. 그러나 구성된 동작에 관계없이 정적 경로 없이는 통신이 불가능합니다.

- 데이터 센터 네트워크 (203.0.113.0/24)와 통신할 수 없습니다. 기본 게이트웨이 설정으로 인해 발신 트래픽이 라우터 2를 통해 대상에 도달합니다. 그러나 데이터 센터 네트워크 (203.0.113.0/24)의 호스트는 기본 게이트웨이(Router 1)에 응답을 전송하므로 패킷에 응답하지 않습니다. 그런 다음 라우터 1은 트래픽을 인터넷으로 전송합니다.

- official network (192.0.2.0/24)와 통신할 수 없습니다. 라우터 2는 기본 게이트웨이 설정으로 인해 트래픽을 라우터 1로 보냅니다. 라우터 1의 실제 동작은 rp_filter sysctl 설정에 따라 달라집니다. 기본적으로 RHEL에서 라우터 1은 이 트래픽을란드 네트워크 (192.0.2.0/24)로 보내는 대신 나가는 트래픽을 삭제합니다. 그러나 구성된 동작에 관계없이 정적 경로 없이는 통신이 불가능합니다.



참고

정적 경로를 구성하는 것 외에도 두 라우터 모두에서 IP 전달을 활성화해야 합니다.

추가 리소스

- net.ipv4.conf.all.rp_filter가 서버에 설정된 경우 서버를 ping할 수 없는 이유는 무엇입니까?
- IP 전달 활성화

24.2. NMCLI 유틸리티를 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 수로 정의된 CWND(congestion 창) 크기를 설정합니다.
- lock-cwnd=true|false: 커널이 CWND 값을 업데이트할 수 있는지 여부를 정의합니다.

- **lock-mtu=true|false:** 커널이 **MTU** 검색 경로로 **MTU**를 업데이트할 수 있는지 여부를 정의합니다.

- **lock-window=true|false:** 커널이 **TCP** 패킷의 최대 창 크기를 업데이트할 수 있는지 여부를 정의합니다.

- **mtu=*<mtu_value>*:** 대상 경로와 함께 사용할 최대 전송 단위**(MTU)**를 설정합니다.

- **onlink=true|false:** 인터페이스 접두사와 일치하지 않는 경우에도 다음 홉을 이 링크에 직접 연결할지 여부를 정의합니다.

- **scope=*<scope>*:** **IPv4** 경로의 경우 이 속성은 경로 접두사에서 다루는 대상의 범위를 설정합니다. 값을 정수**(0-255)**로 설정합니다.

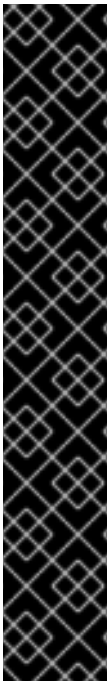
- **src=*<source_address>*:** 경로 접두사가 다루는 대상으로 트래픽을 보낼 때 선호하는 소스 주소를 설정합니다.

- **table=*<table_id>*:** 경로를 추가해야 하는 테이블의 **ID**를 설정합니다. 이 매개변수를 생략하면 **NetworkManager**는 기본 테이블을 사용합니다.

- **tos=*<type_of_service_key>*:** 서비스 유형**(TOS)** 키를 설정합니다. 값을 정수**(0-255)**로 설정합니다.

- **type=*<route_type>*:** 경로 유형을 설정합니다. **NetworkManager**는 유니캐스트,로컬,검정신체,연결할 수 없음,금지 경로 유형을 지원합니다. 기본값은 **unicast** 입니다.

- **window=*<window_size>*:** 바이트로 측정되는 이러한 대상에 대해 **TCP**의 최대 창 크기를 설정합니다.



중요

이전 + 기호 없이 **ipv4.routes** 옵션을 사용하는 경우 **nmcli** 는 이 매개변수의 모든 현재 설정을 덮어씁니다.

- 추가 경로를 생성하려면 다음을 입력합니다.

```
$ nmcli connection modify connection_name +ipv4.routes "<route>"
```

- 특정 경로를 제거하려면 다음을 입력합니다.

```
$ nmcli connection modify connection_name -ipv4.routes "<route>"
```

24.3. NMCLI를 사용하여 정적 경로 구성

nmcli connection modify 명령을 사용하여 기존 **NetworkManager** 연결 프로필에 정적 경로를 추가할 수 있습니다.

아래 절차에서는 다음 경로를 구성합니다.

- 원격 **198.51.100.0/24** 네트워크로의 **IPv4** 경로입니다. **IP** 주소 **192.0.2.10** 이 있는 해당 게이트웨이는 **LAN** 연결 프로필을 통해 연결할 수 있습니다.
- 원격 **2001:db8:2::/64** 네트워크에 대한 **IPv6** 경로입니다. **IP** 주소 **2001:db8:1::10** 이 있는 해당 게이트웨이는 **LAN** 연결 프로필을 통해 연결할 수 있습니다.

사전 요구 사항

- **LAN** 연결 프로필이 존재하고 게이트웨이와 동일한 **IP** 서브넷에 이 호스트를 구성합니다.

절차

1. **LAN** 연결 프로필에 정적 **IPv4** 경로를 추가합니다.

```
# nmcli connection modify LAN +ipv4.routes "198.51.100.0/24 192.0.2.10"
```

한 단계로 여러 경로를 설정하려면 쉼표로 구분된 개별 경로를 명령에 전달합니다.

```
# nmcli connection modify <connection_profile> +ipv4.routes
"<remote_network_1>/<subnet_mask_1> <gateway_1>,
<remote_network_n>/<subnet_mask_n> <gateway_n>, ..."
```

2. LAN 연결 프로필에 정적 IPv6 경로를 추가합니다.

```
# nmcli connection modify LAN +ipv6.routes "2001:db8:2::/64 2001:db8:1::10"
```

3. 연결을 다시 활성화합니다.

```
# nmcli connection up LAN
```

검증

1. IPv4 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. IPv6 경로를 표시합니다.

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

24.4. NMTUI를 사용하여 정적 경로 구성

nmtui 애플리케이션은 NetworkManager에 대한 텍스트 기반 사용자 인터페이스를 제공합니다. nmtui 를 사용하여 그래픽 인터페이스 없이 호스트에서 정적 경로를 구성할 수 있습니다.

예를 들어 아래 절차는 기존 연결 프로필을 통해 연결할 수 있는 198.51.100.1 에서 실행되는 게이트웨이를 사용하는 192.0.2.0/24 네트워크에 경로를 추가합니다.

참고

nmtui 에서 :

- 커서 키를 사용하여 이동합니다.
- 버튼을 선택하고 Enter 를 누릅니다.
- Space 를 사용하여 확인란을 선택하고 지웁니다.

사전 요구 사항

- 네트워크가 구성되어 있습니다.
- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- 사용자가 물리적 콘솔에 로그인된 경우 사용자 권한만으로 충분합니다. 그러지 않으면 명령에 root 권한이 필요합니다.

절차

1. start nmtui:

```
# nmtui
```

2. Edit a connection 을 선택하고 Enter 를 누릅니다.
3. 대상 네트워크에 다음 홉에 도달할 수 있는 연결 프로필을 선택하고 Enter 키를 누릅니다.
4. IPv4 또는 IPv6 경로인지 여부에 따라 프로토콜의 구성 영역 옆에 있는 Show 버튼을 누릅니다.

5. 라우팅 옆에 있는 편집 버튼을 누릅니다. 그러면 정적 경로를 구성하는 새 창이 열립니다.

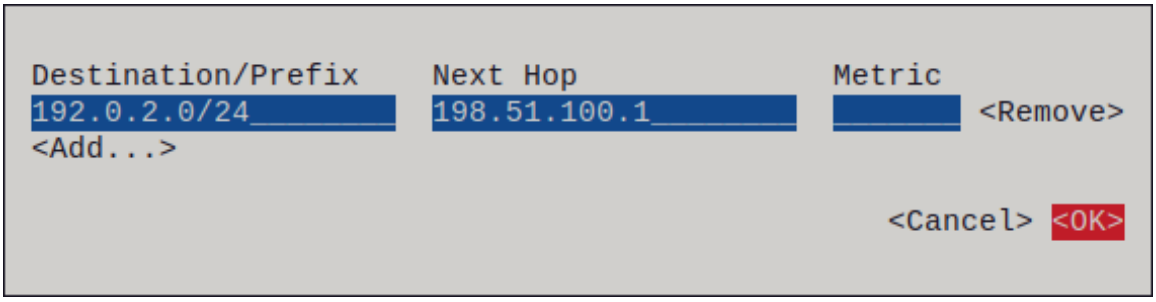
   a. 추가 버튼을 누른 후 다음을 작성합니다.

   - CIDR(Classless Inter-Domain Routing) 형식의 접두사를 포함한 대상 네트워크
   - 다음 홉의 IP 주소
   - 동일한 네트워크에 여러 경로를 추가하고 경로의 우선 순위를 높인 경우 메트릭 값

   b. 추가할 모든 경로에 대해 이전 단계를 반복하고 이 연결 프로필을 통해 연결할 수 있습니다.

   c. 확인 버튼을 눌러 연결 설정으로 창으로 돌아갑니다.

   그림 24.1. 메트릭이 없는 정적 경로의 예



6. OK 버튼을 눌러 nmtui 메인 메뉴로 돌아갑니다.

7. 연결 활성화를 선택하고 Enter 를 누릅니다.

8. 편집한 연결 프로필을 선택하고 Enter 를 두 번 눌러 비활성화한 후 다시 활성화합니다.

중요

다시 활성화하려는 연결 프로필을 사용하는 SSH와 같은 원격 연결을 통해 nmtui 를 실행하는 경우 이 단계를 건너뜁니다. 이 경우 nmtui 에서 비활성화하면 연결이 종료되므로 다시 활성화할 수 없습니다. 이 문제를 방지하려면 nmcli connection < *connection_profile* > up 명령을 사용하여 언급된 시나리오에서 연결을 다시 활성화합니다.

9. 뒤로 버튼을 눌러 메인 메뉴로 돌아갑니다.

10. Quit 를 선택하고 Enter 를 눌러 nmtui 애플리케이션을 종료합니다.

검증

- 경로가 활성 상태인지 확인합니다.

```
$ ip route
...
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

24.5. CONTROL-CENTER를 사용하여 정적 경로 구성

GNOME에서 control-center 를 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

아래 절차에서는 다음 경로를 구성합니다.

- 원격 198.51.100.0/24 네트워크로의 IPv4 경로입니다. 해당 게이트웨이의 IP 주소는 192.0.2.10 입니다.
- 원격 2001:db8:2::/64 네트워크에 대한 IPv6 경로입니다. 해당 게이트웨이의 IP 주소는 2001:db8:1::10 입니다.

사전 요구 사항

- 네트워크가 구성되어 있습니다.

- 이 호스트는 게이트웨이와 동일한 IP 서브넷에 있습니다.

- 연결의 네트워크 구성이 control-center 애플리케이션에서 열립니다. nm-connection-editor를 사용하여 이더넷 연결 구성을 참조하십시오.

절차

1. IPv4 탭에서 다음을 수행합니다.

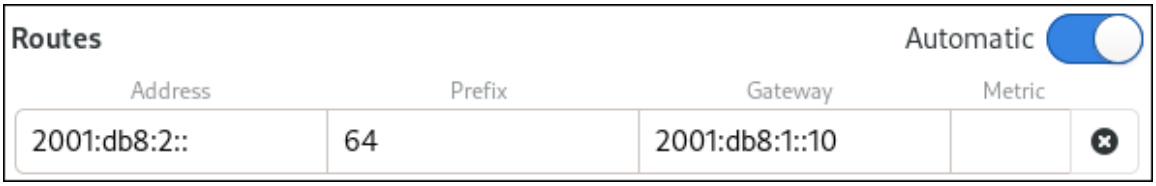
   a. 선택 사항: 정적 경로만 사용하도록 IPv4 탭의 경로 섹션에 있는 On 버튼을 클릭하여 자동 경로를 비활성화합니다. 자동 경로가 활성화되면 Red Hat Enterprise Linux는 DHCP 서버에서 수신한 정적 경로와 경로를 사용합니다.

   b. address, 넷마스크, 게이트웨이 및 선택적으로 IPv4 경로의 메트릭 값을 입력합니다.

2. IPv6 탭에서 다음을 수행합니다.

   a. 선택 사항: 정적 경로만 사용하도록 IPv4 탭의 경로 i 버튼 i를 클릭하여 자동 경로를 비활성화합니다.

   b. address, 넷마스크, 게이트웨이 및 선택적으로 IPv6 경로의 메트릭 값을 입력합니다.



3. **Apply(적용)**를 클릭합니다.

4. 네트워크 창으로 돌아가 연결의 버튼을 전환하여 연결을 비활성화 및 다시 활성화한 후 변경 사항을 적용하려면 연결을 다시 시작합니다.

주의

연결을 다시 시작하면 해당 인터페이스에서 연결이 잠시 중단됩니다.

검증

1. **IPv4** 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. **IPv6** 경로를 표시합니다.

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

24.6. NM-CONNECTION-EDITOR를 사용하여 정적 경로 구성

nm-connection-editor 애플리케이션을 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

아래 절차에서는 다음 경로를 구성합니다.

- 원격 **198.51.100.0/24** 네트워크로의 **IPv4** 경로입니다. **IP** 주소 **192.0.2.10** 을 사용하는 해당 게이트웨이는 예제 연결을 통해 연결할 수 있습니다.

-

원격 **2001:db8:2::/64** 네트워크에 대한 **IPv6** 경로입니다. **IP** 주소가 **2001:db8:1::10** 인 해당 게이트웨이는 예제 연결을 통해 연결할 수 있습니다.

사전 요구 사항

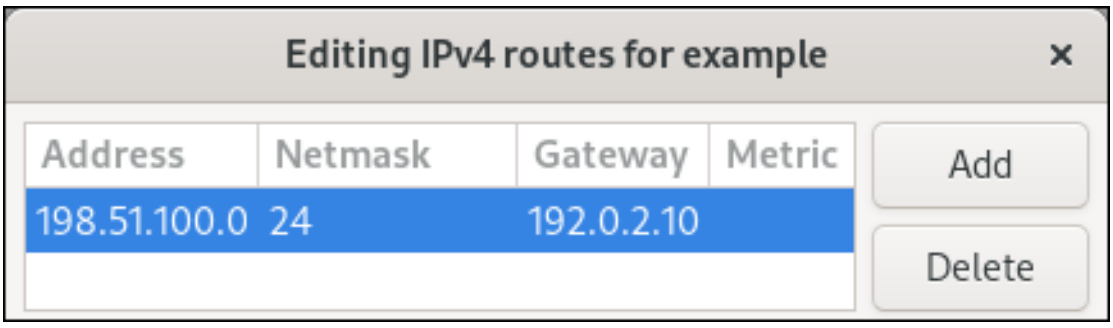
- 네트워크가 구성되어 있습니다.
- 이 호스트는 게이트웨이와 동일한 **IP** 서브넷에 있습니다.

절차

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

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

2. 연결 프로파일 예제 를 선택하고, 기존 연결을 편집하기 위해 톱니바퀴 아이콘을 클릭합니다.
3. **IPv4** 설정 탭에서 다음을 수행합니다.
   a. **Routes**(경로) 버튼을 클릭합니다.
   b. **Add**(추가) 버튼을 클릭하고 주소, 넷마스크, 게이트웨이 및 지표 값을 선택적으로 입력합니다.

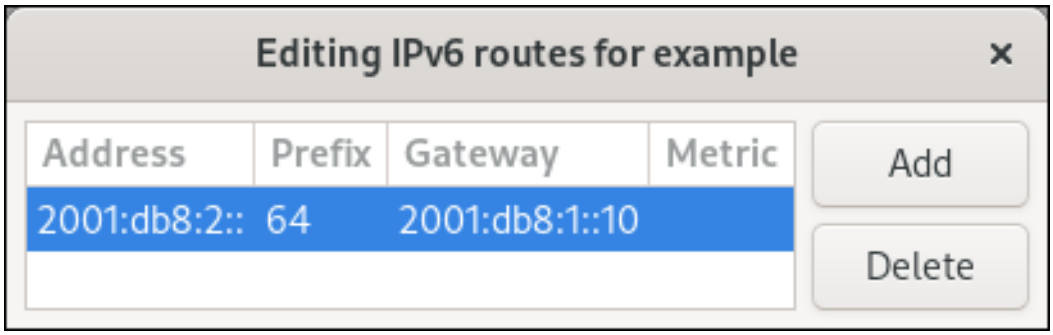


   c. **OK**(확인)를 클릭합니다.

4. **IPv6** 설정 탭에서 다음을 수행합니다.

   a. **Routes**(경로) 버튼을 클릭합니다.

   b. **Add**(추가) 버튼을 클릭하고 주소, 넷마스크, 게이트웨이 및 지표 값을 선택적으로 입력합니다.



   c. **OK**(확인)를 클릭합니다.

5. 저장을 클릭합니다.

6. 변경 사항을 적용하려면 네트워크 연결을 다시 시작합니다. 예를 들어 명령줄을 사용하여 예제 연결을 다시 시작하려면 다음을 수행합니다.

```
# nmcli connection up example
```

검증

1. **IPv4** 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. **IPv6** 경로를 표시합니다.

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

24.7. NMCLI 대화형 모드를 사용하여 정적 경로 구성

nmcli 유틸리티의 대화형 모드를 사용하여 네트워크 연결 구성에 정적 경로를 추가할 수 있습니다.

아래 절차에서는 다음 경로를 구성합니다.

- 원격 198.51.100.0/24 네트워크로의 IPv4 경로입니다. IP 주소 192.0.2.10 을 사용하는 해당 게이트웨이는 예제 연결을 통해 연결할 수 있습니다.
- 원격 2001:db8:2::/64 네트워크에 대한 IPv6 경로입니다. IP 주소가 2001:db8:1::10 인 해당 게이트웨이는 예제 연결을 통해 연결할 수 있습니다.

사전 요구 사항

- 예제 연결 프로파일이 존재하고 이 호스트가 게이트웨이와 동일한 IP 서브넷에 있도록 구성합니다.

절차

1. 예제 연결에 사용할 nmcli 대화형 모드를 엽니다.

```
# nmcli connection edit example
```

2. 정적 IPv4 경로를 추가합니다.

```
nmcli> set ipv4.routes 198.51.100.0/24 192.0.2.10
```

3. 정적 IPv6 경로를 추가합니다.

```
nmcli> set ipv6.routes 2001:db8:2::/64 2001:db8:1::10
```

4. 선택적으로 경로가 구성에 올바르게 추가되었는지 확인합니다.

```
nmcli> print
...
ipv4.routes:    { ip = 198.51.100.0/24, nh = 192.0.2.10 }
...
ipv6.routes:    { ip = 2001:db8:2::/64, nh = 2001:db8:1::10 }
...
```

ip 속성은 라우팅할 네트워크와 게이트웨이(다음 홉)를 표시합니다.

5. 설정을 저장합니다.

```
nmcli> save persistent
```

6. 네트워크 연결을 다시 시작합니다.

```
nmcli> activate example
```

7. nmcli 대화형 모드를 종료합니다.

```
nmcli> quit
```

검증

1. IPv4 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. IPv6 경로를 표시합니다.

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

추가 리소스

- nmcli(1) 도움말 페이지
- nm-settings-nmcli(5) man page

24.8. NMSTATECTL을 사용하여 정적 경로 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 정적 경로를 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 nmstatectl 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

사전 요구 사항

- enp1s0 네트워크 인터페이스가 구성되어 게이트웨이와 동일한 IP 서브넷에 있습니다.
- nmstate 패키지가 설치되어 있습니다.

절차

1. 다음 콘텐츠를 사용하여 YAML 파일(예: ~/add-static-route-to-enp1s0.yml)을 만듭니다.

```
---
routes:
  config:
  - destination: 198.51.100.0/24
    next-hop-address: 192.0.2.10
    next-hop-interface: enp1s0
  - destination: 2001:db8:2::/64
    next-hop-address: 2001:db8:1::10
    next-hop-interface: enp1s0
```

이러한 설정은 다음 정적 경로를 정의합니다.

- 원격 198.51.100.0/24 네트워크로의 IPv4 경로입니다. IP 주소 192.0.2.10 을 사용하는 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.
- 원격 2001:db8:2::/64 네트워크에 대한 IPv6 경로입니다. IP 주소가 2001:db8:1::10 인 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.

2. 시스템에 설정을 적용합니다.

```
# nmstatectl apply ~/add-static-route-to-enp1s0.yml
```

검증

1. IPv4 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. IPv6 경로를 표시합니다.

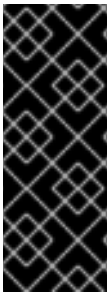
```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

추가 리소스

- nmstatectl(8) man page
- /usr/share/doc/nmstate/examples/ 디렉터리

24.9. 네트워크 RHEL 시스템 역할을 사용하여 정적 경로 구성

네트워크 RHEL 시스템 역할을 사용하여 정적 경로를 구성할 수 있습니다.



중요

네트워크 RHEL 시스템 역할을 사용하는 플레이를 실행하고 설정 값이 플레이에 지정된 값과 일치하지 않으면 역할은 동일한 이름의 기존 연결 프로필을 재정의합니다. 이러한 값을 기본값으로 재설정하지 않으려면 구성이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP and additional routes
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - 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
              route:
                - network: 198.51.100.0
                  prefix: 24
                  gateway: 192.0.2.10
                - network: 2001:db8:2::
                  prefix: 64
                  gateway: 2001:db8:1::10
            state: up
```

이미 존재하는지 여부에 따라 절차는 다음 설정으로 **enp7s0** 연결 프로필을 생성하거나 업데이트합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
- 정적 경로:
  - 192.0.10 게이트웨이가 있는 198.51.100.0/24
  - 2001:db8:2::/64 게이트웨이 2001:db8:1::10

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

1. 관리형 노드에서 다음을 수행합니다.

   a. IPv4 경로를 표시합니다.

   ```
   # ip -4 route
   ...
   198.51.100.0/24 via 192.0.2.10 dev enp7s0
   ```

   b. IPv6 경로를 표시합니다.

   ```
   # ip -6 route
   ...
   2001:db8:2::/64 via 2001:db8:1::10 dev enp7s0 metric 1024 pref medium
   ```

추가 리소스

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

24.10. 레거시 네트워크 스크립트를 사용하는 경우 키-값 형식으로 정적 경로 구성 파일 생성

레거시 네트워크 스크립트는 키-값 형식으로 statics 경로를 설정할 수 있도록 지원합니다.

아래 절차에서는 원격 198.51.100.0/24 네트워크에 대한 IPv4 경로를 구성합니다. IP 주소 192.0.2.10 을 사용하는 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.

참고

레거시 네트워크 스크립트는 정적 IPv4 경로에 대해서만 키-값 형식을 지원합니다. IPv6 경로의 경우 ip-command 형식을 사용합니다. 레거시 네트워크 스크립트를 사용하는 경우 ip-command 형식으로 정적 경로 구성 파일 생성을 참조하십시오.

사전 요구 사항

- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- **NetworkManager** 패키지가 설치되지 않았거나 **NetworkManager** 서비스가 비활성화되어 있습니다.
- **network-scripts** 패키지가 설치되어 있습니다.
- 네트워크 서비스가 활성화되어 있습니다.

절차

1. 정적 **IPv4** 경로를 **/etc/sysconfig/network-scripts/route-enp0s1** 파일에 추가합니다.

```
ADDRESS0=198.51.100.0
NETMASK0=255.255.255.0
GATEWAY0=192.0.2.10
```

- **ADDRESS0** 변수는 첫 번째 라우팅 항목의 네트워크를 정의합니다.
- **NETECDHEK0** 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
```

검증

- IPv4 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

문제 해결

- 네트워크 단위의 저널 항목을 표시합니다.

```
# journalctl -u network
```

다음은 가능한 오류 메시지와 그 원인입니다.

  - 오류: NextECDHE에는 잘못된 게이트웨이 가 있습니다. 이 라우터와 동일한 서브넷에 없는 route-*enp1s0* 파일에 IPv4 게이트웨이 주소를 지정하셨습니다.
  - RTNETLINK 응답: 호스트 경로 없음: 이 라우터와 동일한 서브넷에 없는 route6-*enp1s0* 파일에 IPv6 게이트웨이 주소를 지정하셨습니다.
  - 오류: 지정된 접두사 길이 의 접두사가 유효하지 않습니다. 네트워크 주소가 아닌 원격 네트워크 내의 IP 주소를 사용하여 route-*enp1s0* 파일에 원격 네트워크를 지정했습니다.

추가 리소스

- /usr/share/doc/network-scripts/sysconfig.txt file

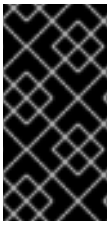
24.11. 레거시 네트워크 스크립트를 사용하는 경우 IP-COMMAND 형식으로 정적 경로 구성 파일 생성

레거시 네트워크 스크립트는 정적 경로 설정을 지원합니다.

아래 절차에서는 다음 경로를 구성합니다.

- 원격 198.51.100.0/24 네트워크로의 IPv4 경로입니다. IP 주소 192.0.2.10 을 사용하는 해당 게이트웨이는 enp1s0 인터페이스를 통해 연결할 수 있습니다.

- 원격 **2001:db8:2::/64** 네트워크에 대한 **IPv6** 경로입니다. **IP** 주소가 **2001:db8:1::10** 인 해당 게이트웨이는 **enp1s0** 인터페이스를 통해 연결할 수 있습니다.



중요

게이트웨이의 **IP** 주소(다음 홉)는 고정 경로를 구성하는 호스트와 동일한 **IP** 서브넷에 있어야 합니다.

이 절차의 예제에서는 **ip-command** 형식의 구성 항목을 사용합니다.

사전 요구 사항

- 정적 경로의 게이트웨이는 인터페이스에서 직접 연결할 수 있어야 합니다.
- **NetworkManager** 패키지가 설치되지 않았거나 **NetworkManager** 서비스가 비활성화되어 있습니다.
- **network-scripts** 패키지가 설치되어 있습니다.
- 네트워크 서비스가 활성화되어 있습니다.

절차

1. 정적 **IPv4** 경로를 **/etc/sysconfig/network-scripts/route-enp1s0** 파일에 추가합니다.

```
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

항상 원격 네트워크의 네트워크 주소를 지정합니다(예: **198.51.100.0**). **198.51.100.1** 과 같이 원격 네트워크 내에 **IP** 주소를 설정하면 네트워크 스크립트가 이 경로를 추가하지 못합니다.

2. 정적 **IPv6** 경로를 **/etc/sysconfig/network-scripts/route6-enp1s0** 파일에 추가합니다.

```
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0
```

3. 네트워크 서비스를 다시 시작합니다.

```
# systemctl restart network
```

검증

1. IPv4 경로를 표시합니다.

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. IPv6 경로를 표시합니다.

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

문제 해결

- 네트워크 단위의 저널 항목을 표시합니다.

```
# journalctl -u network
```

다음은 가능한 오류 메시지와 그 원인입니다.

  - 오류: NextECDHE에는 잘못된 게이트웨이 가 있습니다. 이 라우터와 동일한 서브넷에 없는 route-*enp1s0* 파일에 IPv4 게이트웨이 주소를 지정하셨습니다.
  - RTNETLINK 응답: 호스트 경로 없음: 이 라우터와 동일한 서브넷에 없는 route6-*enp1s0* 파일에 IPv6 게이트웨이 주소를 지정하셨습니다.
  - 오류: 지정된 접두사 길이 의 접두사가 유효하지 않습니다. 네트워크 주소가 아닌 원격 네트워크 내의 IP 주소를 사용하여 route-*enp1s0* 파일에 원격 네트워크를 지정했습니다.

추가 리소스

- **/usr/share/doc/network-scripts/sysconfig.txt file**

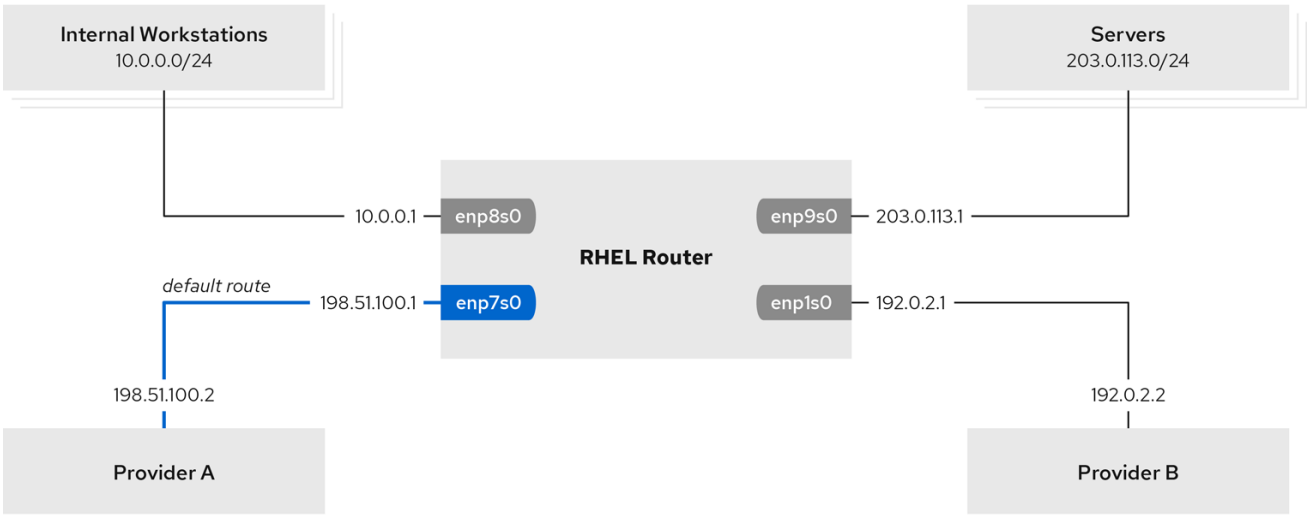
25장. 대체 경로를 정의하도록 정책 기반 라우팅 구성

기본적으로 RHEL의 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.

25.1. NMCLI를 사용하여 특정 서브넷에서 다른 기본 게이트웨이로 트래픽 라우팅

정책 기반 라우팅을 사용하여 특정 서브넷의 트래픽에 대해 다른 기본 게이트웨이를 구성할 수 있습니다. 예를 들어 기본적으로 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 A로 라우팅하는 라우터로 RHEL을 구성할 수 있습니다. 그러나 내부 워크스테이션 서브넷에서 수신되는 트래픽은 공급자 B로 라우팅됩니다.

이 절차에서는 다음 네트워크 토폴로지를 가정합니다.



사전 요구 사항

- 시스템은 NetworkManager 를 사용하여 네트워크를 구성합니다(기본값).
- 프로시저에서 설정할 RHEL 라우터에는 네 개의 네트워크 인터페이스가 있습니다.
  - enp7s0 인터페이스는 공급자 A의 네트워크에 연결되어 있습니다. 공급자 네트워크의

게이트웨이 IP는 198.51.100.2 이며 네트워크는 /30 네트워크 마스크를 사용합니다.

- enp1s0 인터페이스는 공급자 B의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 IP는 192.0.2.2 이며 네트워크는 /30 네트워크 마스크를 사용합니다.
- enp8s0 인터페이스는 내부 워크스테이션이 있는 10.0.0.0/24 서브넷에 연결되어 있습니다.
- 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 이 IP 주소를 라우터의 enp9s0 네트워크 인터페이스에 할당합니다.
- firewalld 서비스가 활성화되어 있고 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. 명령은 다음 옵션을 사용합니다.

- 유형 이더넷: 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>*: 네트워크 인터페이스를 정의된 firewalld 영역에 할당합니다. firewalld 는 외부 영역에 할당된 인터페이스에 대해 자동으로 마스커레이딩을 활성화합니다.**

2. **공급자 B에 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 할당하는 데 필요합니다. 연결이 활성화되면 NetworkManager에서 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 서브넷에 대한 이 고정 경로는 로컬 네트워크 인터페이스의 IP를 공급자 B(192.0.2.1)에 다음 홉으로 사용합니다.

또한 명령에서는 **ipv4.routing-rules** 매개변수를 사용하여 **10.0.0.0/24** 서브넷의 트래픽을 테이블 **5000** 으로 라우팅하는 우선 순위 **5** 인 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

1. 내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

   a. **traceroute** 패키지를 설치합니다.

   ```
   # yum install traceroute
   ```

   b. 인터넷의 호스트에 대한 경로를 표시하려면 **traceroute** 유틸리티를 사용합니다.

   ```
   # 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
    ...
   ```

   명령의 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.

인터넷의 호스트에 대한 경로를 표시하려면 **traceroute** 유틸리티를 사용합니다.

```
# 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
 ...
```

명령의 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 및 기본 테이블에 대한 규칙이 포함되어 있습니다.

2.

표 **5000** 에 경로를 표시 :

```
# 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 **masquerading**이 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

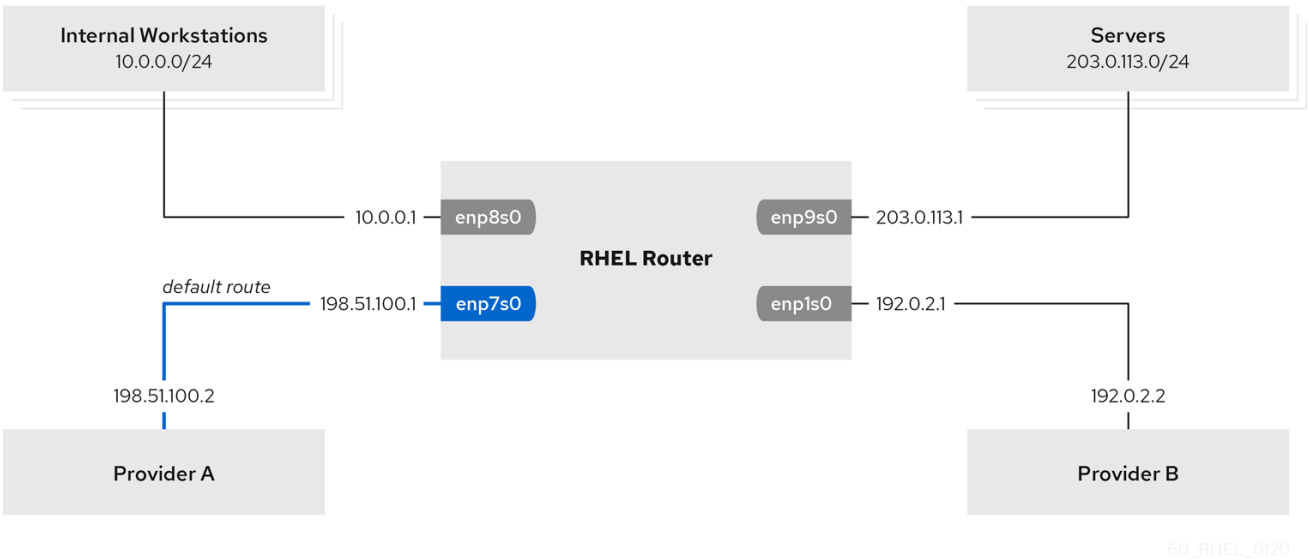
- **nm-settings(5)** 도움말 페이지
- **nmcli(1)** 도움말 페이지

25.2. 네트워크 RHEL 시스템 역할을 사용하여 특정 서브넷에서 다른 기본 게이트웨이로 트래픽 라우팅

정책 기반 라우팅을 사용하여 특정 서브넷의 트래픽에 대해 다른 기본 게이트웨이를 구성할 수 있습니다. 예를 들어 기본적으로 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 **A**로 라우팅하는 라우터로 **RHEL**을 구성할 수 있습니다. 그러나 내부 워크스테이션 서브넷에서 수신되는 트래픽은 공급자 **B**로 라우팅됩니다.

정책 기반 라우팅을 원격으로 여러 노드에서 구성하려면 네트워크 **RHEL** 시스템 역할을 사용할 수 있습니다.

이 절차에서는 다음과 같은 네트워크 토폴로지를 가정합니다.



사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 관리형 노드는 **NetworkManager** 및 **firewalld** 서비스를 사용합니다.
- 구성하려는 관리형 노드에는 네 개의 네트워크 인터페이스가 있습니다.
  - **enp7s0** 인터페이스는 공급자 **A**의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 IP는 **198.51.100.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp1s0** 인터페이스는 공급자 **B**의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 IP는 **192.0.2.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp8s0** 인터페이스는 내부 워크스테이션이 있는 **10.0.0.0/24** 서브넷에 연결되어 있습니다.

- 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 이 IP 주소를 라우터의 enp9s0 네트워크 인터페이스에 할당합니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configuring policy-based routing
  hosts: managed-node-01.example.com
  tasks:
    - name: Routing traffic from a specific subnet to a different default gateway
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: Provider-A
            interface_name: enp7s0
            type: ethernet
            autoconnect: True
            ip:
              address:
                - 198.51.100.1/30
              gateway4: 198.51.100.2
              dns:
                - 198.51.100.200
            state: up
            zone: external

          - name: Provider-B
            interface_name: enp1s0
            type: ethernet
            autoconnect: True
            ip:
              address:
                - 192.0.2.1/30
              route:
                - network: 0.0.0.0
                  prefix: 0
                  gateway: 192.0.2.2
                  table: 5000
            state: up
            zone: external
```

```
    - name: Internal-Workstations
      interface_name: enp8s0
      type: ethernet
      autoconnect: True
      ip:
        address:
          - 10.0.0.1/24
        route:
          - network: 10.0.0.0
            prefix: 24
            table: 5000
        routing_rule:
          - priority: 5
            from: 10.0.0.0/24
            table: 5000
      state: up
      zone: trusted

    - name: Servers
      interface_name: enp9s0
      type: ethernet
      autoconnect: True
      ip:
        address:
          - 203.0.113.1/24
      state: up
      zone: trusted
```

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

1. 내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

   a. **traceroute** 패키지를 설치합니다.

```
# yum install traceroute
```

b. 인터넷의 호스트에 대한 경로를 표시하려면 **traceroute** 유틸리티를 사용합니다.

```
# 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
 ...
```

명령의 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. 인터넷의 호스트에 대한 경로를 표시하려면 **traceroute** 유틸리티를 사용합니다.

```
# 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
 ...
```

명령의 출력은 라우터가 공급자 **A**의 네트워크인 **198.51.100.2** 를 통해 패킷을 전송함을 표시합니다.

3. **RHEL** 시스템 역할을 사용하여 구성한 **RHEL** 라우터에서 다음을 수행합니다.

a. 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 RHEL에는 로컬, 기본 및 기본 테이블에 대한 규칙이 포함되어 있습니다.

b. 표 5000 에 경로를 표시 :

```
# 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
```

c. 인터페이스 및 방화벽 영역을 표시합니다.

```
# firewall-cmd --get-active-zones
external
  interfaces: enp1s0 enp7s0
trusted
  interfaces: enp8s0 enp9s0
```

d. 외부 영역에 masquerading이 활성화되어 있는지 확인합니다.

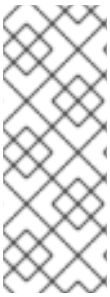
```
# firewall-cmd --info-zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0 enp7s0
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  ...
```

추가 리소스

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

25.3. 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 네트워크를 구성하는 것은 **RHEL 8**에서 더 이상 사용되지 않습니다. **NetworkManager**를 사용하여 정책 기반 라우팅을 구성합니다. 예를 들어 **nmcli** 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)** 매뉴얼 페이지

- **ip-rule(8)** 매뉴얼 페이지

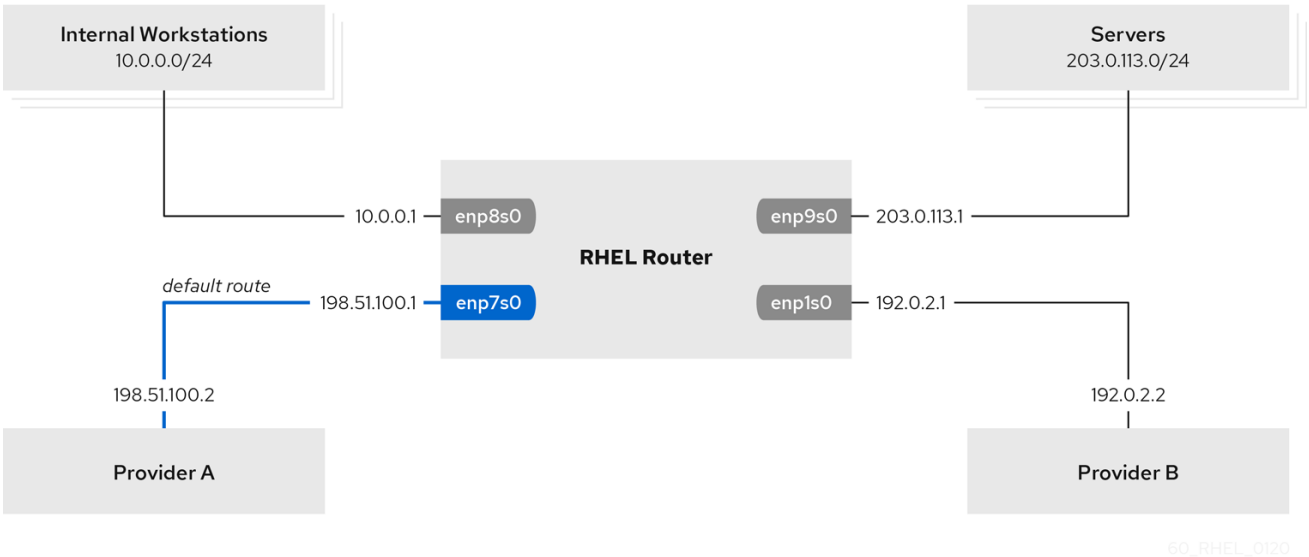
25.4. 레거시 네트워크 스크립트를 사용하여 특정 서브넷에서 다른 기본 게이트웨이로 트래픽을 라우팅

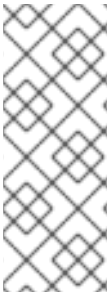
정책 기반 라우팅을 사용하여 특정 서브넷의 트래픽에 대해 다른 기본 게이트웨이를 구성할 수 있습니다. 예를 들어 기본적으로 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 **A**로 라우팅하는 라우터로 **RHEL**을 구성할 수 있습니다. 그러나 내부 워크스테이션 서브넷에서 수신되는 트래픽은 공급자 **B**로 라우팅됩니다.

중요

network-scripts 패키지에서 제공하는 레거시 네트워크 스크립트를 사용하여 네트워크를 구성하는 것은 **RHEL 8**에서 더 이상 사용되지 않습니다. 호스트에서 **NetworkManager** 대신 레거시 네트워크 스크립트를 사용하는 경우에만 절차를 따릅니다. **NetworkManager**를 사용하여 네트워크 설정을 관리하는 경우 **nmcli**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 수행하기 위해 이 절차에서는 **ifcfg-***, **route-***, **rules-*** 파일의 번호를 사용합니다.

사전 요구 사항

- **NetworkManager** 패키지가 설치되지 않았거나 **NetworkManager** 서비스가 비활성화되어 있습니다.
- **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 **IP**는 **198.51.100.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp1s0** 인터페이스는 공급자 **B**의 네트워크에 연결되어 있습니다. 공급자 네트워크의 게이트웨이 **IP**는 **192.0.2.2** 이며 네트워크는 **/30** 네트워크 마스크를 사용합니다.
  - **enp8s0** 인터페이스는 내부 워크스테이션이 있는 **10.0.0.0/24** 서브넷에 연결되어 있습니다.
  - **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 이 **IP** 주소를 라우터의 **enp9s0** 네트워크 인터페이스에 할당합니다.
- **firewalld** 서비스가 활성화되어 있고 활성화됩니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/ifcfg-1_Provider-A** 파일을 생성하여 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
```

구성 파일은 다음 매개변수를 사용합니다.

- **TYPE=Ethernet:** 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: RHEL**이 시스템이 부팅될 때 이 연결을 시작하도록 정의합니다.

- **ZONE=*firewalld_zone*: 네트워크 인터페이스를 정의된 firewalld 영역에 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 라우팅되는 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 우선순위 **5** 인 라우팅 규칙을 정의합니다. 낮은 값은 우선 순위가 높습니다.

c. **ID 5000** 이 있는 라우팅 테이블에 정적 경로를 추가하려면 다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/route-3_Internal-Workstations** 파일을 만듭니다.

```
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 전송하도록 정의합니다. 이 인터페이스는 테이블 **5000** 을 라우팅하고 다음 홉으로 사용됩니다.

4. 다음 콘텐츠를 사용하여 **/etc/sysconfig/network-scripts/ifcfg-4_Servers** 파일을 생성하여 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

1. 내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

a.
traceroute 패키지를 설치합니다.

```
# yum install traceroute
```

b.
인터넷의 호스트에 대한 경로를 표시하려면 traceroute 유틸리티를 사용합니다.

```
# 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
 ...
```

명령의 출력은 라우터가 공급자 B의 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.
인터넷의 호스트에 대한 경로를 표시하려면 traceroute 유틸리티를 사용합니다.

```
# 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
 ...
```

명령의 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 RHEL에는 로컬, 기본 및 기본 테이블에 대한 규칙이 포함되어 있습니다.

2. 표 5000 에 경로를 표시 :

```
# 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 masquerading이 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) 매뉴얼 페이지

- **ip-rule(8) 매뉴얼 페이지**

- **/usr/share/doc/network-scripts/sysconfig.txt file**

26장. 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.

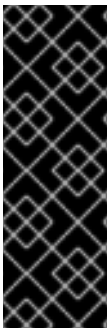
계층 2에서 VRF를 파티셔닝하는 한 가지 이점은 라우팅이 관련된 피어 수를 고려하는 것이 더 효율적이라는 것입니다.

Red Hat Enterprise Linux는 각 VRF 도메인에 가상 vrt 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.

26.1. 다른 인터페이스에서 동일한 IP 주소 영구적으로 재사용

가상 라우팅 및 전달(VRF) 기능을 사용하여 한 서버의 서로 다른 인터페이스에서 동일한 IP 주소를 영구적으로 사용할 수 있습니다.



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 로그인합니다.
- 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 VRF에 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 VRF에 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
```

26.2. 다른 인터페이스에서 동일한 IP 주소 일시적으로 재사용

가상 라우팅 및 전달(VRF) 기능을 사용하여 한 서버의 서로 다른 인터페이스에서 동일한 IP 주소를 일시적으로 사용할 수 있습니다. 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 로그인합니다.
- 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. Blue 장치를 활성화합니다.

```
# ip link set dev blue up
```

c. 네트워크 장치를 VRF 장치에 할당합니다. 예를 들어 enp1s0 이더넷 장치를 Blue VRF 장치에 추가하려면 다음을 수행합니다.

```
# ip link set dev enp1s0 master blue
```

d. enp1s0 장치를 활성화합니다.

```
# ip link set dev enp1s0 up
```

e. IP 주소와 서브넷 마스크를 enp1s0 장치에 할당합니다. 예를 들어 192.0.2.1/24 로 설정하려면 다음을 실행합니다.

```
# 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 이라는 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 enp7s0 이더넷 장치를 빨간색 VRF 장치에 추가하려면 다음을 수행합니다.

```
# ip link set dev enp7s0 master red
```

d. enp7s0 장치를 활성화합니다.

```
# ip link set dev enp7s0 up
```

e.

Blue VRF 도메인의 **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.

26.3. 추가 리소스

- **kernel- doc** 패키지의 **/usr/share/doc/kernel-doc-<*kernel_version*>/Documentation/networking/vrf.txt**

27장. 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.

27.1. 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. 이름이 동일한 가상 장치로 vrf0 연결을 생성하고 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 포트로 생성합니다. 이 구성으로 인해 vrf0 장치와 연결된 라우팅 테이블 1000 에 라우팅 정보가 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"
```

그러면 192.0.2.2.2를 라우터로 사용하는 198.51.100.0/24 네트워크에 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    192.0.2.1/24
```

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 203.0.113.0/24 dev enp7s0 proto static metric 100
```

기본 라우팅 테이블은 장치 enp1s0 장치 또는 192.0.2.1/24 서브넷과 연결된 경로를 언급하지 않습니다.

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 **192.0.2.254** 를 기본 게이트웨이로 사용함을 나타냅니다.

5. **vrf0** 과 연결된 네트워크에서 **traceroute** 유틸리티를 실행하여 유틸리티가 표 **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 시스템 기본 라우팅 테이블의 기본 게이트웨이가 아닌 라우팅 테이블 **1000** 에 할당된 기본 게이트웨이입니다.

추가 리소스

- **ip-vrf(8) man page**

27.2. 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 구성하셨습니다.
-

vrf0 장치와 연결된 인터페이스에 할당된 **IP** 주소에서만 수신 대기하도록 **Apache HTTP** 서버를 구성했습니다.

절차

1. **httpd systemd** 서비스의 콘텐츠를 표시합니다.

```
# systemctl cat httpd
...
[Service]
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
...
```

이후 단계에서 **VRF** 네트워크 내에서 동일한 명령을 실행하려면 **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 생성합니다.

```
[Service]
ExecStart=
ExecStart=/usr/sbin/ip vrf exec vrf0 /usr/sbin/httpd $OPTIONS -DFOREGROUND
```

ExecStart 매개변수를 재정의하려면 먼저 설정을 설정 해제한 다음 다음과 같이 새 값으로 설정해야 합니다.

4. **systemd**를 다시 로드합니다.

```
# systemctl daemon-reload
```

5. **httpd** 서비스를 다시 시작합니다.

```
# systemctl restart httpd
```

검증

1. **httpd 프로세스의 PID(프로세스 ID)를 표시합니다.**

```
# pidof -c httpd
1904 ...
```

2. **PID에 대한 VRF 연결을 표시합니다. 예를 들면 다음과 같습니다.**

```
# ip vrf identify 1904
vrf0
```

3. **vrf0 장치와 연결된 모든 PID를 표시합니다.**

```
# ip vrf pids vrf0
1904  httpd
...
```

추가 리소스

- **ip-vrf(8) man page**

28장. NETWORKMANAGER 연결 프로필에서 ETHTOOL 설정 구성

NetworkManager는 특정 네트워크 드라이버 및 하드웨어 설정을 영구적으로 구성할 수 있습니다. **ethtool** 유틸리티를 사용하여 이러한 설정을 관리하는 것과 비교하여 재부팅 후 설정을 손실하지 않는 이점이 있습니다.

NetworkManager 연결 프로필에서 다음 **ethtool** 설정을 설정할 수 있습니다.

오프로드 기능

네트워크 인터페이스 컨트롤러는 **TOE(TCP** 오프로드 엔진)를 사용하여 특정 작업을 네트워크 컨트롤러에 오프로드할 수 있습니다. 이렇게 하면 네트워크 처리량이 향상됩니다.

병합 설정 중단

인터럽트 병합을 사용하면 시스템은 네트워크 패킷을 수집하고 여러 패킷에 대한 단일 인터럽트를 생성합니다. 이렇게 하면 하드웨어 인터럽트가 **1**개인 커널로 전송되는 데이터의 양이 증가하여 인터럽트 부하가 감소하고 처리량이 극대화됩니다.

링 버퍼

이러한 버퍼는 수신 및 발신 네트워크 패킷을 저장합니다. 링 버퍼 크기를 늘리면 패킷 드롭 속도를 줄일 수 있습니다.

28.1. NMCLI를 사용하여 ETHTOOL 오프로드 기능 구성

NetworkManager를 사용하여 연결 프로필에서 **ethtool** 오프로드 기능을 활성화 및 비활성화할 수 있습니다.

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 활성화 또는 비활성화한 오프로드 기능의 설정을 제거하려면 기능의 매개변수를 **null** 값으로 설정합니다. 예를 들어 **TX** 오프로드의 구성을 제거하려면 다음을 입력합니다.

```
# nmcli con modify enp1s0 ethtool.feature-tx ""
```

3. 네트워크 프로필을 다시 활성화합니다.

```
# nmcli connection up enp1s0
```

검증

- ethtool -k 명령을 사용하여 네트워크 장치의 현재 오프로드 기능을 표시합니다.

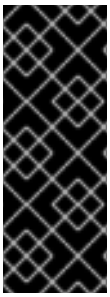
```
# ethtool -k network_device
```

추가 리소스

- nm-settings-nmcli(5) man page

28.2. 네트워크 RHEL 시스템 역할을 사용하여 ETHTOOL 오프로드 기능 구성

네트워크 RHEL 시스템 역할을 사용하여 NetworkManager 연결의 ethtool 기능을 구성할 수 있습니다.



중요

네트워크 RHEL 시스템 역할을 사용하는 플레이를 실행하고 설정 값이 플레이에 지정된 값과 일치하지 않으면 역할은 동일한 이름의 기존 연결 프로필을 재정의합니다. 이러한 값을 기본값으로 재설정하지 않으려면 구성이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with ethtool features
      ansible.builtin.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
```

이 플레이북은 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 프로필이 이미 있는 경우 업데이트합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 198.51.100.20
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

28.3. NMCLI를 사용하여 ETHTOOL 병합 설정 구성

NetworkManager를 사용하여 연결 프로필에서 **ethtool** 병합 설정을 설정할 수 있습니다.

절차

1. 예를 들어 **enp1s0** 연결 프로파일에서 지연되도록 수신되는 패킷의 최대 수를 **128** 개로 설정하려면 다음을 입력합니다.

```
# nmcli connection modify enp1s0 ethtool.coalesce-rx-frames 128
```

2. 병합 설정을 제거하려면 이를 **null** 값으로 설정합니다. 예를 들어 **ethtool.coalesce-rx-frames** 설정을 제거하려면 다음을 입력합니다.

```
# nmcli connection modify enp1s0 ethtool.coalesce-rx-frames ""
```

3. 네트워크 프로필을 다시 활성화하려면 다음을 수행합니다.

```
# nmcli connection up enp1s0
```

검증

1. **ethtool -c** 명령을 사용하여 네트워크 장치의 현재 오프로드 기능을 표시합니다.

```
# ethtool -c network_device
```

추가 리소스

- **nm-settings-nmcli(5) man page**

28.4. 네트워크 RHEL 시스템 역할을 사용하여 ETHTOOL 병합 설정 구성

네트워크 **RHEL** 시스템 역할을 사용하여 **NetworkManager** 연결의 **ethtool** 병합 설정을 구성할 수 있습니다.

중요

네트워크 **RHEL** 시스템 역할을 사용하는 플레이를 실행하고 설정 값이 플레이에 지정된 값과 일치하지 않으면 역할은 동일한 이름의 기존 연결 프로필을 재정의합니다. 이러한 값을 기본값으로 재설정하지 않으려면 구성이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/**playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with ethtool coalesce settings
      ansible.builtin.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
```

이 플레이북은 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 프로필이 이미 있는 경우 업데이트합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 198.51.100.20
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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 병합 설정:
  - RX 프레임: 128
  - TX 프레임: 128

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

28.5. NMCLI를 사용하여 높은 패킷 드롭 속도를 줄이기 위해 링 버퍼 크기를 늘리십시오.

패킷 드롭 비율로 인해 애플리케이션이 데이터, 시간 초과 또는 기타 문제가 발생하는 경우 이더넷 장치의 링 버퍼 크기를 늘립니다.

수신 링 버퍼는 장치 드라이버와 NIC(네트워크 인터페이스 컨트롤러) 간에 공유됩니다. 카드는 전송(TX)을 할당하고 (RX) 링 버퍼를 수신합니다. 이름에서 알 수 있듯이 링 버퍼는 오버플로가 기존 데이터를 덮어쓰는 순환 버퍼입니다. NIC에서 커널로 데이터를 이동하는 방법에는 두 가지가 있습니다. 하드웨어 인터럽트 및 소프트웨어 인터럽트라고도 합니다.

커널은 장치 드라이버가 이를 처리할 수 있을 때까지 RX 링 버퍼를 사용하여 들어오는 패킷을 저장합니다. 장치 드라이버는 일반적으로 SoftIRQs를 사용하여 RX 링을 드레이닝하여 들어오는 패킷을 sk_buff 또는 skb 라는 커널 데이터 구조에 배치하여 커널과 관련 소켓을 소유하는 애플리케이션까지 이동합니다.

커널은 TX 링 버퍼를 사용하여 네트워크로 전송해야 하는 발신 패킷을 보관합니다. 이러한 링 버퍼는 스택의 하단에 있으며 패킷 드롭이 발생할 수 있는 중요한 지점이며, 이로 인해 네트워크 성능에 부정적인 영향을 미칩니다.

절차

1. 인터페이스의 패킷 삭제 통계를 표시합니다.

```
# ethtool -S enp1s0
  ...
  rx_queue_0_drops: 97326
```

```
    rx_queue_1_drops: 63783
    ...
```

명령 출력은 네트워크 카드 및 드라이버에 따라 다릅니다.

삭제 또는 드롭 카운터의 높은 값은 사용 가능한 버퍼가 패킷을 처리할 수 있는 속도보다 빠르게 채워지는 것을 나타냅니다. 링 버퍼를 늘리면 이러한 손실을 피할 수 있습니다.

2. 최대 링 버퍼 크기를 표시합니다.

```
# ethtool -g enp1s0
Ring parameters for enp1s0:
Pre-set maximums:
RX:             4096
RX Mini:        0
RX Jumbo:       16320
TX:             4096
Current hardware settings:
RX:             255
RX Mini:        0
RX Jumbo:       0
TX:             255
```

Pre-set maximums 섹션의 값이 현재 하드웨어 설정 섹션에서보다 높은 경우 다음 단계에서 설정을 변경할 수 있습니다.

3. 인터페이스를 사용하는 **NetworkManager** 연결 프로필을 식별합니다.

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Example-Connection  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

4. 연결 프로필을 업데이트하고 링 버퍼를 늘립니다.

- **RX** 링 버퍼를 늘리려면 다음을 입력합니다.

```
# nmcli connection modify Example-Connection ethtool.ring-rx 4096
```

- **TX** 링 버퍼를 늘리려면 다음을 입력합니다.

-

```
# nmcli connection modify Example-Connection ethtool.ring-tx 4096
```

5. NetworkManager 연결을 다시 로드합니다.

```
# nmcli connection up Example-Connection
```

중요

NIC가 사용하는 드라이버에 따라 링 버퍼를 변경하면 네트워크 연결이 곧 중단될 수 있습니다.

추가 리소스

- ifconfig 및 ip 명령 보고서 패킷 삭제
- 0.05%의 패킷 감소율에 대해 우려해야 합니까?
- ethtool(8) 도움말 페이지

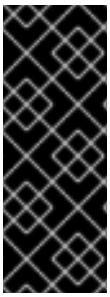
28.6. 네트워크 RHEL 시스템 역할을 사용하여 높은 패킷 드롭 속도를 줄이기 위해 링 버퍼 크기 증가

패킷 드롭 비율로 인해 애플리케이션이 데이터, 시간 초과 또는 기타 문제가 발생하는 경우 이더넷 장치의 링 버퍼 크기를 늘립니다.

링 버퍼는 오버플로가 기존 데이터를 덮어쓰는 순환 버퍼입니다. 네트워크 카드는 전송(TX)을 할당하고 (RX) 링 버퍼를 수신합니다. 수신 링 버퍼는 장치 드라이버와 NIC(네트워크 인터페이스 컨트롤러) 간에 공유됩니다. 데이터는 SoftIRQs라고도 하는 하드웨어 인터럽트 또는 소프트웨어 인터럽트를 통해 NIC에서 커널로 이동할 수 있습니다.

커널은 장치 드라이버가 이를 처리할 수 있을 때까지 RX 링 버퍼를 사용하여 들어오는 패킷을 저장합니다. 장치 드라이버는 일반적으로 SoftIRQs를 사용하여 RX 링을 드레이닝하여 들어오는 패킷을 sk_buff 또는 skb 라는 커널 데이터 구조에 배치하여 커널과 관련 소켓을 소유하는 애플리케이션까지 이동합니다.

커널은 TX 링 버퍼를 사용하여 네트워크로 전송해야 하는 발신 패킷을 보관합니다. 이러한 링 버퍼는 스택의 하단에 있으며 패킷 드롭이 발생할 수 있는 중요한 지점이며, 이로 인해 네트워크 성능에 부정적인 영향을 미칩니다.



중요

네트워크 **RHEL** 시스템 역할을 사용하는 플레이를 실행하고 설정 값이 플레이에 지정된 값과 일치하지 않으면 역할은 동일한 이름의 기존 연결 프로필을 재정의합니다. 이러한 값을 기본값으로 재설정하지 않으려면 구성이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 장치가 지원하는 최대 링 버퍼 크기를 알고 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with increased ring buffer sizes
      ansible.builtin.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:
      ring:
        rx: 4096
        tx: 4096
    state: up
```

이 플레이북은 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 프로필이 이미 있는 경우 업데이트합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 198.51.100.20
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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
- 최대 링 버퍼 항목 수:
  - 수신(RX): 4096
  - 전송(TX): 4096

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

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

29장. NETWORKMANAGER 디버깅 소개

모든 또는 특정 도메인에 대한 로그 수준을 늘리면 **NetworkManager**가 수행하는 작업에 대한 세부 정보를 기록하는 데 도움이 됩니다. 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.

29.1. NETWORKMANAGER 다시 적용 방법 소개

NetworkManager 서비스는 프로필을 사용하여 장치의 연결 설정을 관리합니다. 데스크탑 버스(**D-Bus**) **API**는 이러한 연결 설정을 생성, 수정 및 삭제할 수 있습니다. 프로필 변경의 경우 **D-Bus API**는 기존 설정을 연결의 수정된 설정에 복제합니다. 복제에도 불구하고 수정된 설정은 변경 사항이 적용되지 않습니다. 이를 적용하려면 연결의 기존 설정을 다시 활성화하거나 **reapply()** 메서드를 사용합니다.

reapply() 메서드에는 다음과 같은 기능이 있습니다.

1. 네트워크 인터페이스를 비활성화하거나 다시 시작하지 않고 수정된 연결 설정을 업데이트합니다.
2. 수정된 연결 설정에서 보류 중인 변경 사항을 제거합니다. **NetworkManager** 는 수동 변경 사항을 되돌리지 않으므로 장치를 재구성하고 외부 또는 수동 매개변수를 되돌릴 수 있습니다.
3. 기존 연결 설정과 다른 수정된 연결 설정을 생성합니다.

또한 **reapply()** 메서드는 다음과 같은 속성을 지원합니다.

- **bridge.ageing-time**
- **bridge.forward-delay**
- **bridge.group-address**

- **bridge.group-forward-mask**
- **bridge.hello-time**
- **bridge.max-age**
- **bridge.multicast-hash-max**
- **bridge.multicast-last-member-count**
- **bridge.multicast-last-member-interval**
- **bridge.multicast-membership-interval**
- **bridge.multicast-querier**
- **bridge.multicast-querier-interval**
- **bridge.multicast-query-interval**
- **bridge.multicast-query-response-interval**
- **bridge.multicast-query-use-ifaddr**
- **bridge.multicast-router**
- **bridge.multicast-snooping**

- **bridge.multicast-startup-query-count**
- **bridge.multicast-startup-query-interval**
- **bridge.priority**
- **bridge.stp**
- **bridge.VLAN-filtering**
- **bridge.VLAN-protocol**
- **bridge.VLANs**
- **802-3-ethernet.accept-all-mac-addresses**
- **802-3-ethernet.cloned-mac-address**
- **IPv4.addresses**
- **IPv4.dhcp-client-id**
- **IPv4.dhcp-iaid**
- **IPv4.dhcp-timeout**
- **IPv4.DNS**

- **IPv4.DNS-priority**
- **IPv4.DNS-search**
- **IPv4.gateway**
- **IPv4.ignore-auto-DNS**
- **IPv4.ignore-auto-routes**
- **IPv4.may-fail**
- **IPv4.method**
- **IPv4.never-default**
- **IPv4.route-table**
- **IPv4.routes**
- **IPv4.routing-rules**
- **IPv6.addr-gen-mode**
- **IPv6.addresses**
- **IPv6.dhcp-duid**

- **IPv6.dhcp-iaid**
- **IPv6.dhcp-timeout**
- **IPv6.DNS**
- **IPv6.DNS-priority**
- **IPv6.DNS-search**
- **IPv6.gateway**
- **IPv6.ignore-auto-DNS**
- **IPv6.may-fail**
- **IPv6.method**
- **IPv6.never-default**
- **IPv6.ra-timeout**
- **IPv6.route-metric**
- **IPv6.route-table**
- **IPv6.routes**

- **IPv6.routing-rules**

추가 리소스

- **nm-settings-nmcli(5) man page**

29.2. NETWORKMANAGER 로그 수준 설정

기본적으로 모든 로그 도메인은 **INFO** 로그 수준을 기록하도록 설정됩니다. 디버그 로그를 수집하기 전에 속도 제한을 비활성화합니다. 속도 제한으로 **systemd-journald** 는 짧은 시간에 너무 많은 메시지가 있는 경우 메시지를 삭제합니다. 이는 로그 수준이**ACE**인 경우 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 해당 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 만듭니다.

```
[logging]
domains=ALL:TRACE
```

domains 매개변수는 쉼표로 구분된 **domain:level** 쌍을 여러 개 포함할 수 있습니다.

4. **NetworkManager** 서비스를 다시 시작합니다.

```
# systemctl restart NetworkManager
```

검증

- **systemd** 저널을 쿼리하여 **NetworkManager** 단위의 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
...
```

29.3. NMCLI를 사용하여 런타임 시 로그 수준 임시 설정

nmcli 를 사용하여 런타임 시 로그 수준을 변경할 수 있습니다.

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** 으로 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
```

29.4. 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(1) 매뉴얼 페이지

29.5. 디버깅 수준 및 도메인

수준 및 도메인 매개변수를 사용하여 NetworkManager의 디버깅을 관리할 수 있습니다. 수준은 상세 수준을 정의하는 반면 도메인은 지정된 심각도(레벨)로 로그를 기록하는 메시지의 카테고리를 정의합니다.

로그 수준	설명
OFF	NetworkManager에 대한 메시지를 기록하지 않음
ERR	중요한 오류만 기록합니다.

로그 수준	설명
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 활성화

후속 수준은 이전 수준의 모든 메시지를 기록합니다. 예를 들어 로그 수준을 INFO 로 설정하면 ERR 및 WARN 로그 수준에 포함된 메시지가 기록됩니다.

추가 리소스

- **NetworkManager.conf(5) 도움말 페이지**

30장. LLDP를 사용하여 네트워크 구성 문제 디버그

LLDP(Link Layer Discovery Protocol)를 사용하여 토폴로지의 네트워크 구성 문제를 디버그할 수 있습니다. 즉 LLDP는 다른 호스트 또는 라우터 및 스위치와의 구성 불일치를 보고할 수 있습니다.

30.1. LLDP 정보를 사용하여 잘못된 VLAN 구성 디버깅

특정 VLAN을 사용하도록 스위치 포트를 구성하고 호스트에서 이러한 VLAN 패킷을 수신하지 않는 경우LLDP(Link Layer Discovery Protocol)를 사용하여 문제를 디버깅할 수 있습니다. 패킷을 수신하지 않는 호스트에서 다음 절차를 수행합니다.

사전 요구 사항

- nmstate 패키지가 설치되어 있습니다.
- 스위치는 LLDP를 지원합니다.
- LLDP는 근접한 장치에서 사용할 수 있습니다.

절차

1. 다음 콘텐츠를 사용하여 ~/enable-LLDP-enp1s0.yml 파일을 생성합니다.

```
interfaces:
 - name: enp1s0
   type: ethernet
   lldp:
     enabled: true
```

2. ~/enable-LLDP-enp1s0.yml 파일을 사용하여 enp1s0 인터페이스에서 LLDP를 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. 출력이 설정이 예상 구성과 일치하는지 확인합니다. 예를 들어 스위치에 연결된 인터페이스의 **LLDP** 정보는 이 호스트가 연결된 스위치 포트가 **VLAN ID 448** 을 사용하는 것으로 표시됩니다.

```
- type: 127
      ieee-802-1-vlans:
      - name: v2-0488-03-0505
        vid: 488
```

enp1s0 인터페이스의 네트워크 구성에서 다른 **VLAN ID**를 사용하는 경우 그에 따라 변경합니다.

추가 리소스

- VLAN 태그 구성

31장. LINUX 트래픽 제어

Linux는 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 TC는 필터와 작업을 사용하여 분류 중에 패킷 콘텐츠를 무시합니다. TC 하위 시스템은 TC 아키텍처의 기본 요소인 대기열 교육(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 NetworkManager를 사용하여 영구적으로 수행할 수 있습니다.

Red Hat Enterprise Linux에서는 네트워크 인터페이스의 트래픽을 관리하는 다양한 방법으로 기본 대기열을 구성할 수 있습니다.

31.1. 대기열 분야의 개요

페인팅 비지니스 (qdiscs)는 네트워크 인터페이스에 의한 트래픽 전송 예약을 대기열에 달하는 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 handle 이라고 하는 16비트 16진수(예: 1: 또는 abcd:)가 있습니다. 이 숫자를 qdisc major number라고 합니다. qdisc 에 클래스가 있는 경우 식별자는 마이너 전 <major>:<minor>(예 : abcd:1) 이전의 두 숫자의 쌍으로 구성됩니다. 마이너 번호의 번호 체계는 qdisc 유형에 따라 다릅니다. 종종 번호 매각은 1 클래스의 ID < major>:1, 두 번째 < major>:2 등이 있는 경우도 있습니다. 일부 qdiscs 를 사용하면 클래스를 만들 때 임의로 클래스 마이너 번호를 설정할 수 있습니다.

classful qdiscs

다양한 유형의 qdiscs 가 존재하고 네트워킹 인터페이스로 패킷을 전송하는 데 도움이 됩니다. root, 상위 또는 하위 클래스를 사용하여 qdiscs 를 구성할 수 있습니다. 하위 항목을 연결할 수 있는 지점을 클래스라고 합니다. qdisc 의 클래스는 유연하며 항상 여러 자식 클래스 또는 단일 자식 클래스를 포함할 수 있습니다. 준비된 qdisc 자체를 포함하는 클래스에 대해 금지되지 않으므로 복잡한 트래픽 제어 시나리오가 용이해집니다.

클래스의 qdiscs 는 패킷을 자체적으로 저장하지 않습니다. 대신 qdisc 와 관련된 기준에 따라 하위 항목 중 하나에 대한 요청을 큐에 추가 및 해제합니다. 결국 이 재귀 패킷 통과는 패킷이 저장되는

위치(또는 분리 시 선택)가 종료됩니다.

classless qdiscs

일부 **qdiscs** 에는 하위 클래스가 없으며 클래스가 없는 **qdiscs** 라고 합니다. **classless qdiscs** 는 **classful qdiscs** 에 비해 사용자 지정이 줄어듭니다. 일반적으로 인터페이스를 인터페이스에 연결하는 것만으로도 충분합니다.

추가 리소스

- **TC(8)** 매뉴얼 페이지
- **TC-actions(8)** 매뉴얼 페이지

31.2. TC 유틸리티를 사용하여 네트워크 인터페이스의 QDISC 검사

기본적으로 **Red Hat Enterprise Linux** 시스템은 **fq_codel qdisc** 를 사용합니다. **tc** 유틸리티를 사용하여 **qdisc** 카운터를 검사할 수 있습니다.

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 횟수입니다.
- **Overlimits -** 구성된 링크 용량이 채워지는 횟수

- sent - dequeues 수

31.3. 기본 QDISC 업데이트

현재 qdisc 로 네트워킹 패킷 손실을 관찰하면 네트워크 요구 사항에 따라 qdisc 를 변경할 수 있습니다.

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.

```
# modprobe -r NETWORKDRIVERNAME
# modprobe NETWORKDRIVERNAME
```

5. 네트워크 인터페이스를 시작합니다.

```
# ip link set enp0s1 up
```

검증

- 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

31.4. TC 유틸리티를 사용하여 네트워크 인터페이스의 현재 QDISC를 일시적으로 설정

기본 **qdisc**를 변경하지 않고 현재 **qdisc** 를 업데이트할 수 있습니다.

절차

1. 선택 사항: 현재 **qdisc** 보기:

```
# tc -s qdisc show dev enp0s1
```

2. 현재 **qdisc** 를 업데이트합니다.

```
# tc qdisc replace dev enp0s1 root htb
```

검증

- 업데이트 된 **qdisc**:

```
# 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
```

31.5. NETWORKMANAGER를 사용하여 네트워크 인터페이스의 현재 QDISC를 영구적으로 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

- 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

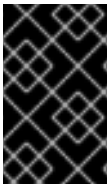
31.6. RHEL에서 사용 가능한 QDISC

각 qdisc 는 고유한 네트워킹 관련 문제를 해결합니다. 다음은 RHEL에서 사용할 수 있는 qdiscs 목록입니다. 다음 qdisc 중 하나를 사용하여 네트워킹 요구 사항에 따라 네트워크 트래픽을 형성할 수 있습니다.

표 31.1. RHEL에서 사용 가능한 스케줄러

qdisc 이름	에 포함	오프로드 지원
비동기 전송 모드(ATM)	**kernel-modules-extra**	
클래스 기반 큐잉	**kernel-modules-extra**	
크레딧 기반 셰퍼	**kernel-modules-extra**	있음
응답하지 않는 흐름(CHOKE)을 위해 을 선택하고 응답하지 않는 흐름에 대해 계속합니다.	**kernel-modules-extra**	
제어된 지연 (CoDel)	**kernel-core**	
deficit Round Robin (DRR)	**kernel-modules-extra**	
차별화된 서비스 마커 (DSANG)	**kernel-modules-extra**	
향상된 전송 선택 (ETS)	**kernel-modules-extra**	있음
공정 대기열 (FQ)	**kernel-core**	
공정 쿼리 제어 지연 (FQ_CODel)	**kernel-core**	
GRED(Commonized Random Early Detection)	**kernel-modules-extra**	
HSFC(Herarchical 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**	

qdisc 이름	에 포함	오프로드 지원
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 NIC에서 하드웨어 및 드라이버 지원이 필요합니다.

추가 리소스

- **TC(8) 매뉴얼 페이지**

32장. 파일 시스템에 저장된 인증서와 함께 802.1X 표준을 사용하여 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. 클라이언트가 이러한 네트워크에 연결할 수 있도록 하려면 이 클라이언트에서 802.1X 인증을 구성해야 합니다.

32.1. NMCLI를 사용하여 기존 이더넷 연결에서 802.1X 네트워크 인증 구성

nmcli 유틸리티를 사용하여 명령줄에서 802.1X 네트워크 인증을 사용하여 이더넷 연결을 구성할 수 있습니다.

사전 요구 사항

- 네트워크는 802.1X 네트워크 인증을 지원합니다.
- 이더넷 연결 프로필은 NetworkManager에 있으며 유효한 IP 구성이 있습니다.
- 클라이언트에는 TLS 인증에 필요한 다음 파일이 있습니다.
  - 저장된 클라이언트 키는 /etc/pki/tls/private/client.key 파일에 있으며 파일은 root 사용자만 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 있어야 합니다.

절차

1. EAP(Extensible Authentication Protocol)를 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 명령에 **802-1x.eap,802-1x.client-cert, 802-1x.private-key** 매개변수를 설정해야 합니다.

2. **CA** 인증서의 경로를 설정합니다.

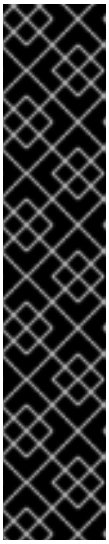
```
# nmcli connection modify enp1s0 802-1x.ca-cert /etc/pki/tls/certs/ca.crt
```

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 늘리려면 **802-1x.password-flags** 매개변수를 **0x1** 로 설정합니다. 이 설정을 사용하여 **GNOME** 데스크탑 환경 또는 **nm-ECDHEt** 가 실행 중인 서버에서 **NetworkManager**는 이러한 서비스에서 암호를 검색합니다. 다른 경우에는 **NetworkManager**에서 암호를 묻는 메시지를 표시합니다.

5. 연결 프로필을 활성화합니다.

```
# nmcli connection up enp1s0
```

검증

- 네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

추가 리소스

- 이더넷 연결 구성
- **nm-settings(5)** 도움말 페이지
- **nmcli(1)** 도움말 페이지

32.2. NMSTATECTL을 사용하여 802.1X 네트워크 인증을 사용하여 정적 이더넷 연결 구성

nmstatectl 유틸리티를 사용하여 **Nmstate API**를 통해 **802.1X** 네트워크 인증으로 이더넷 연결을 구성합니다. **Nmstate API**는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 **nmstatectl** 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

참고

nmstate 라이브러리는 **TLS EAP(Extensible Authentication Protocol)** 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 파일은 **root** 사용자만 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
```

이러한 설정은 다음 설정을 사용하여 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다.

- **/24** 서브넷 마스크가 있는 정적 **IPv4** 주소 **- 192.0.2.1**

- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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 프로토콜을 사용한 802.1x 네트워크 인증

2. 시스템에 설정을 적용합니다.

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

검증

- 네트워크 인증이 필요한 네트워크에서 리소스에 액세스합니다.

32.3. 네트워크 RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성

네트워크 RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증을 사용하여 이더넷 연결을 원격으로 구성할 수 있습니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.

- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 관리형 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. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure an Ethernet connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      ansible.builtin.copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0600

    - name: Copy client certificate for 802.1X authentication
      ansible.builtin.copy:
        src: "/srv/data/client.crt"
        dest: "/etc/pki/tls/certs/client.crt"

    - name: Copy CA certificate for 802.1X authentication
      ansible.builtin.copy:
        src: "/srv/data/ca.crt"
        dest: "/etc/pki/ca-trust/source/anchors/ca.crt"
```

```
- name: Configure connection
  ansible.builtin.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
```

이러한 설정은 다음 설정을 사용하여 **enp1s0** 장치에 대한 이더넷 연결 프로필을 정의합니다.

- /24 서브넷 마스크가 있는 정적 IPv4 주소 - 192.0.2.1
- 정적 IPv6 주소 - 2001:db8:1::1 (/64 서브넷 마스크 포함)
- 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

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

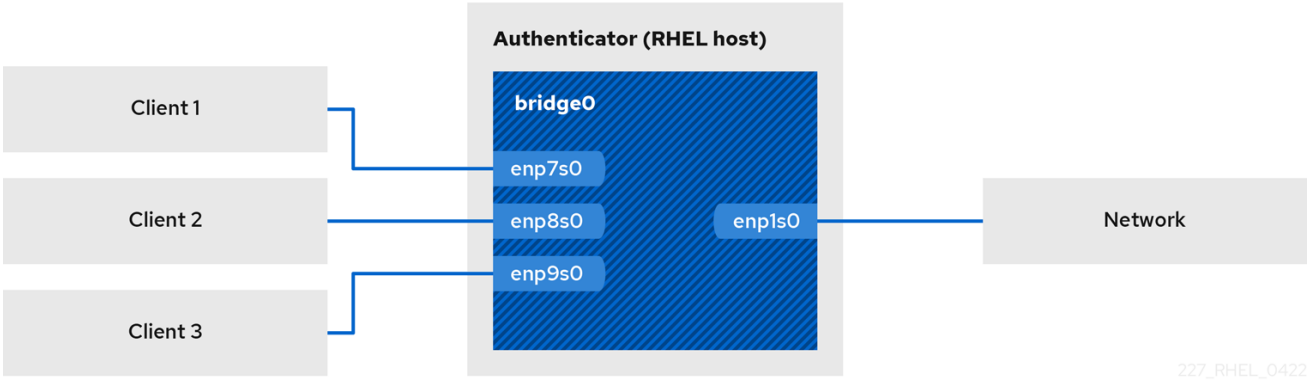
추가 리소스

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

33장. FREERADIUS 백엔드와 함께 HOSTAPD 를 사용하여 LAN 클라이언트에 대한 802.1X 네트워크 인증 서비스 설정

IEEE 802.1X 표준은 인증되지 않은 클라이언트로부터 네트워크를 보호하기 위한 보안 인증 및 권한 부여 방법을 정의합니다. hostapd 서비스 및 FreeRADIUS를 사용하면 네트워크에서 NAC(Network Access Control)를 제공할 수 있습니다.

이 문서에서 RHEL 호스트는 다른 클라이언트를 기존 네트워크에 연결하는 브릿지 역할을 합니다. 그러나 RHEL 호스트는 인증된 클라이언트만 네트워크에 대한 액세스 권한을 부여합니다.



33.1. 사전 요구 사항

- freeradius 패키지를 새로 설치합니다.

  패키지가 이미 설치된 경우 /etc/raddb/ 디렉터리를 제거한 다음 패키지를 다시 설치합니다. /etc/raddb/ 디렉터리의 권한 및 심볼릭 링크가 다르므로 yum reinstall 명령을 사용하여 패키지를 다시 설치하지 마십시오.

33.2. 인증자에서 브릿지 설정

네트워크 브릿지는 MAC 주소 테이블을 기반으로 호스트와 네트워크 간에 트래픽을 전달하는 링크 계층 장치입니다. RHEL을 802.1X 인증기로 설정한 경우 인증을 수행할 인터페이스와 LAN 인터페이스를 브리지에 추가합니다.

사전 요구 사항

- 서버에는 여러 개의 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 전달할 수 있습니다.

```
# 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
```

명령에서 **0x8** 을 반환하는 경우 전달이 활성화됩니다.

추가 리소스

- **nm-settings(5)** 도움말 페이지

33.3. FREERADIUS의 인증서 요구 사항

안전한 **FreeRADIUS** 서비스의 경우 다음과 같이 다양한 목적으로 **TLS** 인증서가 필요합니다.

- 서버에 대한 암호화된 연결에 필요한 **TLS** 서버 인증서입니다. 신뢰할 수 있는 **CA**(인증 기관)를 사용하여 인증서를 발급합니다.

  서버 인증서를 사용하려면 확장 키 사용량(**EKU**) 필드를 **TLS** 웹 서버 인증 으로 설정해야 합니다.

- **EAP-TLS**(**Extended Authentication Protocol Transport Layer Security**)를 위해 동일한 **CA**에서 발급한 클라이언트 인증서입니다. **EAP-TLS**는 인증서 기반 인증을 제공하며 기본적으로 활성화됩니다.

  클라이언트 인증서에는 해당 **EKU** 필드가 **TLS** 웹 클라이언트 인증 으로 설정되어야 합니다.

주의

연결을 보호하려면 회사의 **CA**를 사용하거나 **FreeRADIUS**의 인증서를 발급하기 위해 자체 **CA**를 생성합니다. 공용 **CA**를 사용하는 경우 사용자를 인증하고 **EAP-TLS**에 대한 클라이언트 인증서를 발행하도록 허용합니다.

33.4. 테스트를 위해 FREERADIUS 서버에서 인증서 세트 생성

테스트를 위해 **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 스크립트를 통해 생성된 인증서는 **60**일 후에 만료되고 비보안 암호를 사용합니다("모든"). 그러나 **CA**, 서버 및 클라이언트 구성을 사용자 지정할 수 있습니다.

절차를 수행한 후 이 문서의 뒷부분에서 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: /etc/raddb/certs/ca.cnf 파일에서 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: /etc/raddb/certs/server.cnf 파일에서 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: /etc/raddb/certs/client.cnf 파일에서 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 " d"로 변경합니다.

```
# chgrp radiusd /etc/raddb/certs/server.pem
```

추가 리소스

- /etc/raddb/certs/README.md

33.5. EAP를 사용하여 네트워크 클라이언트를 안전하게 인증하도록 FREERADIUS 구성

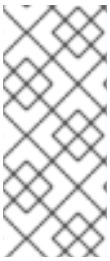
Freeradius는 다양한 EAP(Extensible Authentication Protocol) 방법을 지원합니다. 그러나 보안 네트워크의 경우 다음과 같은 보안 EAP 인증 방법만 지원하도록 FreeRADIUS를 구성합니다.

- EAP-TLS(전송 계층 보안)는 인증서를 사용하여 클라이언트를 인증하는 보안 TLS 연결을 사용합니다. EAP-TLS를 사용하려면 각 네트워크 클라이언트 및 서버의 서버 인증서에 대한 TLS 클라이언트 인증서가 필요합니다. 동일한 CA(인증 기관)가 인증서를 발급해야 합니다. 사용하는 CA에서 발급한 모든 클라이언트 인증서는 FreeRADIUS 서버에 인증할 수 있으므로 항상 자체 CA를 사용하여 인증서를 생성합니다.

- EAP-TTLS(터널링된 전송 계층 보안)는 보안 TLS 연결을 외부 인증 프로토콜로 사용하여 터널을 설정합니다. 그런 다음 내부 인증은 예를 들어 암호 인증 프로토콜(PAP) 또는 챌린지 핸드셰이크 인증 프로토콜(CHAP)을 사용합니다. EAP-TTLS를 사용하려면 TLS 서버 인증서가 필요합니다.

- EAP-PEAP(보안 확장 가능한 인증 프로토콜)는 보안 TLS 연결을 외부 인증 프로토콜로 사용하여 터널을 설정합니다. 인증자는 RADIUS 서버의 인증서를 인증합니다. 이후 supplicant는

Microsoft 챌린지 핸드셰이크 인증 프로토콜 버전 2(MS-CHAPv2) 또는 기타 방법을 사용하여 암호화된 터널을 통해 인증합니다.



참고

기본 FreeRADIUS 구성 파일은 문서 역할을 하며 모든 매개 변수 및 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/ 디렉토리에 있는 구성 파일은 변경되지 않고 freeradius 패키지에서 제공하는 대로입니다.
- 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 파일에 private _key_file ,certificate_file 매개변수를 설정합니다.

절차

1. Diffie-Hellman(Diffie-Hellman) 매개변수가 있는 /etc/raddb/certs/dh 가 없는 경우 새로 생성합니다. 예를 들어 2048비트로 DH 파일을 만들려면 다음을 입력합니다.

```
# openssl dhparam -out /etc/raddb/certs/dh 2048
```

보안상의 이유로 2048비트 미만의 DH 파일을 사용하지 마십시오. 비트 수에 따라 파일 생성에 몇 분이 걸릴 수 있습니다.

2.

DH 매개변수를 사용하여 TLS 개인 키, 서버 인증서, CA 인증서 및 파일에 대한 보안 권한을 설정합니다.

```
# 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 매개변수에서 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 = ttls
    ...
}
```

보안 환경의 경우 ttls,tls 또는 peap 만 사용하십시오.

c.

md5 지시문을 주석 처리하여 비보안 EAP-MD5 인증 방법을 비활성화합니다.

```
eap {
    ...
    # md5 {
    # }
    ...
}
```

기본 구성 파일에서 다른 비보안 **EAP** 인증 방법은 기본적으로 주석 처리됩니다.

4. **/etc/raddb/sites-available/default** 파일을 편집하고 **eap** 이외의 모든 인증 방법을 주석으로 처리하십시오.

```
authenticate {
    ...
    # Auth-Type PAP {
    #     pap
    # }

    # Auth-Type CHAP {
    #     chap
    # }

    # Auth-Type MS-CHAP {
    #     mschap
    # }

    # mschap

    # digest
    ...
}
```

이렇게 하면 외부 인증에 대해 **EAP**만 활성화되고 일반 텍스트 인증 방법을 비활성화합니다.

5. **/etc/raddb/clients.conf** 파일을 편집합니다.

a. **localhost** 및 **localhost_ipv6** 클라이언트 지시문에 보안 암호를 설정합니다.

```
client localhost {
    ipaddr = 127.0.0.1
    ...
    secret = localhost_client_password
    ...
}

client localhost_ipv6 {
    ipv6addr = ::1
    secret = localhost_client_password
}
```

b. 네트워크 인증자의 클라이언트 지시문을 추가합니다.

```
client hostapd.example.org {
    ipaddr = 192.0.2.2/32
    secret = hostapd_client_password
}
```

c.

선택 사항: 다른 호스트도 FreeRADIUS 서비스에 액세스할 수 있어야 하는 경우 다음과 같이 클라이언트 지시문을 추가합니다.

```
client <hostname_or_description> {
    ipaddr = <IP_address_or_range>
    secret = <client_password>
}
```

ipaddr 매개변수는 IPv4 및 IPv6 주소를 허용하고, 선택적 CIDR(Classless inter-domain routing) 표기법을 사용하여 범위를 지정할 수 있습니다. 그러나 이 매개변수에는 하나의 값만 설정할 수 있습니다. 예를 들어 IPv4 및 IPv6 주소에 대한 액세스 권한을 부여하려면 두 개의 클라이언트 지시문을 추가해야 합니다.

client 지시문에 대한 설명적 이름(예: 호스트 이름 또는 IP 범위가 사용되는 위치를 설명하는 단어)을 사용합니다.

6.

EAP-TTLS 또는 EAP-PEAP를 사용하려면 사용자를 /etc/raddb/users 파일에 추가합니다.

```
example_user       Cleartext-Password := "user_password"
```

EAP-TLS(인증서 기반 인증)를 사용해야 하는 사용자의 경우 항목을 추가하지 마십시오.

7.

구성 파일을 확인합니다.

```
# radiusd -XC
...
Configuration appears to be OK
```

8.

firewalld 서비스에서 RADIUS 포트를 엽니다.

```
# firewall-cmd --permanent --add-service=radius
# firewall-cmd --reload
```

9. 이름이 지정된 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now radiusd
```

검증

- FreeRADIUS 서버 또는 인증기에 대한 EAP-TTLS 인증 테스트
- FreeRADIUS 서버 또는 인증자에 대한 EAP-TLS 인증 테스트

문제 해결

1. 이름이 지정된 서비스를 중지합니다.

```
# systemctl stop radiusd
```

2. 디버그 모드에서 서비스를 시작합니다.

```
# radiusd -X
...
Ready to process requests
```

3. 확인 섹션에 언급된 대로 FreeRADIUS 호스트에서 인증 테스트를 수행합니다.

다음 단계

- 더 이상 필요하지 않은 인증 방법 및 기타 기능을 비활성화합니다.

33.6. 유선 네트워크에서 HOSTAPD 를 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.

호스트 서비스는 통합된 RADIUS 서버를 제공합니다. 그러나 테스트 용도로만 통합 RADIUS 서버를 사용하십시오. 프로덕션 환경의 경우 다양한 인증 방법 및 액세스 제어와 같은 추가 기능을 지원하는

FreeRADIUS 서버를 사용합니다.

중요

hostapd 서비스는 트래픽 플레인과 상호 작용하지 않습니다. 이 서비스는 인증자 역할을 합니다. 예를 들어 **hostapd** 제어 인터페이스를 사용하는 스크립트 또는 서비스를 사용하여 인증 이벤트 결과에 따라 트래픽을 허용하거나 거부합니다.

사전 요구 사항

- **hostapd** 패키지가 설치되어 있어야 합니다.
- **FreeRADIUS** 서버가 구성되어 클라이언트를 인증할 준비가 되어 있습니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/hostapd/hostapd.conf** 파일을 만듭니다.

```
# 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=hostapd_client_password

# RADIUS accounting server
acct_server_addr=192.0.2.1
acct_server_port=1813
acct_server_shared_secret=hostapd_client_password
```

이 구성에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/doc/hostapd/hostapd.conf** 예제 구성 파일의 설명을 참조하십시오.

2. **hostapd** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now hostapd
```

검증

- 다음 내용을 참조하십시오.
  - **FreeRADIUS** 서버 또는 인증기에 대한 **EAP-TTLS** 인증 테스트
  - **FreeRADIUS** 서버 또는 인증자에 대한 **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 섹션에 언급된 대로 **FreeRADIUS** 호스트에서 인증 테스트를 수행합니다.

추가 리소스

- **hostapd.conf(5) man page**
- **/usr/share/doc/hostapd/hostapd.conf file**

33.7. FREERADIUS 서버 또는 인증기에 대한 EAP-TTLS 인증 테스트

터널링된 전송 계층 보안(**EAP-TTLS**)에서 **EAP**(확장된 인증 프로토콜)를 사용하여 인증이 작동하는지 테스트하려면 다음 절차를 실행하십시오.

- **FreeRADIUS** 서버를 설정한 후
- **hostapd** 서비스를 **802.1X** 네트워크 인증의 인증자로 설정한 후

이 프로세스에 사용된 테스트 유틸리티의 출력에서는 **EAP** 통신에 대한 추가 정보를 제공하고 문제를 디버깅하는 데 도움이 됩니다.

사전 요구 사항

- 인증하려는 경우 다음을 수행합니다.
  - **FreeRADIUS** 서버:
    - **hostapd** 패키지에서 제공하는 **eapol_test** 유틸리티가 설치됩니다.

- 이 절차를 실행하는 클라이언트는 **FreeRADIUS** 서버의 클라이언트 데이터베이스에서 승인되었습니다.

- 동일한 이름의 패키지에서 제공되는 **wpa_supplicant** 유틸리티인 **Authenticator**가 설치됩니다.

- **CA(**인증 기관**)** 인증서를 **/etc/pki/tls/certs/ca.pem** 파일에 저장했습니다.

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.

- 인증자는 다음을 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 확장 인증 프로토콜을 전송하는 네트워크 인터페이스 이름을 지정합니다.

자세한 디버깅 정보는 **-d** 옵션을 명령에 전달합니다.

추가 리소스

- **/usr/share/doc/wpa_supplicant/wpa_supplicant.conf file**

33.8. FREERADIUS 서버 또는 인증자에 대한 EAP-TLS 인증 테스트

EAP(확장된 인증 프로토콜) 전송 계층 보안(EAP-TLS)을 사용하여 인증이 작동하는지 테스트하려면 다음 절차를 실행합니다.

- **FreeRADIUS** 서버를 설정한 후
- **hostapd** 서비스를 **802.1X** 네트워크 인증의 인증자로 설정한 후

이 프로세스에 사용된 테스트 유틸리티의 출력에서는 **EAP** 통신에 대한 추가 정보를 제공하고 문제를 디버깅하는 데 도움이 됩니다.

사전 요구 사항

- 인증하려는 경우 다음을 수행합니다.
  - **FreeRADIUS** 서버:
    - **hostapd** 패키지에서 제공하는 **eapol_test** 유틸리티가 설치됩니다.
    - 이 절차를 실행하는 클라이언트는 **FreeRADIUS** 서버의 클라이언트 데이터베이스에서 승인되었습니다.
  - 동일한 이름의 패키지에서 제공되는 **wpa_supplicant** 유틸리티인 **Authenticator**가 설치됩니다.
- **CA**(인증 기관) 인증서를 **/etc/pki/tls/certs/ca.pem** 파일에 저장했습니다.
- 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 저장하셨습니다.

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.

- 인증자는 다음을 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 확장 인증 프로토콜을 전송하는 네트워크 인터페이스 이름을 지정합니다.

자세한 디버깅 정보는 -d 옵션을 명령에 전달합니다.

추가 리소스

- /usr/share/doc/wpa_supplicant/wpa_supplicant.conf file

33.9. 호스트된 인증 이벤트를 기반으로 트래픽 차단 및 허용

hostapd 서비스는 트래픽 플레인과 상호 작용하지 않습니다. 이 서비스는 인증자 역할을 합니다. 그러나 인증 이벤트 결과에 따라 트래픽을 허용 및 거부하는 스크립트를 작성할 수 있습니다.

중요

이 절차는 지원되지 않으며 엔터프라이즈 대응 솔루션이 아닙니다. **hostapd_cli** 에서 검색된 이벤트를 평가하여 트래픽을 차단하거나 허용하는 방법만 보여줍니다.

802-1x-tr-mgmt systemd 서비스가 시작되면 **RHEL**은 **LAN (EAPOL)** 패킷을 통한 확장 가능한 인증 프로토콜을 제외하고 **hostapd** 포트의 모든 트래픽을 차단하고 **hostapd_cli** 유틸리티를 사용하여 호스트 설정 제어 인터페이스에 연결합니다. **/usr/local/bin/802-1x-tr-mgmt** 스크립트는 이벤트를 평가합니다. **hostapd_cli** 에서 수신한 다양한 이벤트에 따라 스크립트는 **MAC** 주소에 대한 트래픽을 허용하거나 차단합니다. **802-1x-tr-mgmt** 서비스가 중지되면 모든 트래픽이 다시 자동으로 허용됩니다.

hostapd 서버에서 다음 절차를 수행합니다.

사전 요구 사항

- **hostapd** 서비스가 구성되었으며 서비스는 클라이언트를 인증할 준비가 되었습니다.

절차

1. 다음 콘텐츠를 사용하여 **/usr/local/bin/802-1x-tr-mgmt** 파일을 생성합니다.

```
#!/bin/sh

if [ "x$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 [ "x$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-mgmt@.service systemd** 서비스 파일을 생성합니다.

```
[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 중인 **802-1x-tr-mgmt** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now 802-1x-tr-mgmt@br0.service
```

검증

- 클라이언트에 네트워크를 인증합니다. 다음 내용을 참조하십시오.
  - FreeRADIUS 서버 또는 인증기에 대한 EAP-TTLS 인증 테스트
  - FreeRADIUS 서버 또는 인증자에 대한 EAP-TLS 인증 테스트

추가 리소스

- **systemd.service(5) man page**

34장. MULTIPATH TCP 시작하기

TCP(Transmission Control Protocol)는 인터넷을 통해 데이터를 안정적으로 제공하고 네트워크 로드에 대한 응답으로 대역폭을 자동으로 조정합니다. 다중 경로 TCP(MPTCP)는 원래 TCP 프로토콜(single-path)의 확장입니다. MPTCP를 사용하면 전송 연결이 여러 경로에서 동시에 작동할 수 있으며 사용자 끝점 장치에 대한 네트워크 연결 중복성이 제공됩니다.

34.1. MPTCP 이해

Multipath TCP(MPTCP) 프로토콜을 사용하면 연결 끝점 간에 여러 경로를 동시에 사용할 수 있습니다. 프로토콜 설계는 연결 안정성을 향상시키고 단일 경로 TCP에 비해 다른 이점을 제공합니다.

참고

MPTCP 용어에서 링크는 경로로 간주됩니다.

다음은 MPTCP를 사용할 때의 몇 가지 이점입니다.

- 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.

MPTCP에 대한 자세한 내용은 *추가 리소스*를 참조하십시오.

추가 리소스

- 다중 경로 TCP 이해: 끝점 및 향후 네트워킹의 고가용성
- RFC8684: 여러 주소로 다중 경로 작업용 TCP 확장

- Red Hat Enterprise Linux 8.3의 다중 경로 TCP: 0에서 1 하위 흐름까지

34.2. MPTCP 지원을 사용하도록 RHEL 준비

RHEL에서 MPTCP 지원은 기본적으로 비활성화되어 있습니다. 이 기능을 지원하는 애플리케이션에서 사용할 수 있도록 MPTCP를 활성화합니다. 또한 해당 애플리케이션에 기본적으로 TCP 소켓이 있는 경우 MPTCP 소켓을 강제로 사용하도록 사용자 공간 애플리케이션을 구성해야 합니다.

sysctl 유틸리티를 사용하여 MPTCP 지원을 활성화하고 SystemTap 스크립트를 사용하여 전체 애플리케이션에 대해 MPTCP를 활성화하기 위해 RHEL을 준비할 수 있습니다.

사전 요구 사항

다음 패키지가 설치되어 있습니다.

- systemtap
- 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 MPTCP가 활성화되어 있는지 확인합니다.

```
# sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 1
```

3. 다음 콘텐츠를 사용하여 ECDHEtcp-app.stap 파일을 만듭니다.

```
#!/usr/bin/env stap

%{
#include <linux/in.h>
```

```
#include <linux/ip.h>
%}

/* RSI contains 'type' and RDX contains 'protocol'.
 */

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 MPTCP가 특정 애플리케이션에만 사용하도록 허용하려면 다음 내용으로ECDHE 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. **ECDHE tcp-app.stap** 스크립트가 커널 프로브를 설치한 후 커널 **dmesg** 출력에 다음 경고가 표시됩니다.

```
# 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: ffffffffc0550000, 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 흐름 관련 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
advmss: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. **MPTCP** 카운터를 확인합니다.

```
# nstat MPTcp*

#kernel
MPTcpExtMPCapableSYNRX          2                  0.0
MPTcpExtMPCapableSYNTX          2                  0.0
MPTcpExtMPCapableSYNACKRX       2                  0.0
MPTcpExtMPCapableACKRX          2                  0.0
```

추가 리소스

- RHEL 시스템용 debuginfo 패키지를 다운로드하거나 설치하려면 어떻게 해야 합니까?
- **TCPECDHE** 도움말 페이지
- **mptcpize(8)** man page

34.3. IPROUTE2를 사용하여 MPTCP 애플리케이션의 여러 경로를 일시적으로 구성하고 활성화

각 **MPTCP** 연결은 일반 **TCP**와 유사한 단일 하위 흐름을 사용합니다. **MPTCP** 이점을 얻으려면 각 **MPTCP** 연결에 대한 최대 하위 흐름 수에 대해 더 높은 제한을 지정합니다. 그런 다음 해당 하위 흐름을 생성하도록 추가 끝점을 구성합니다.

중요

이 절차의 구성은 시스템을 재부팅한 후 유지되지 않습니다.

MPTCP는 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 제공하는 대로 최대 **1**개의 추가 원격 주소를 허용하도록 클라이언트를 구성합니다.

```
# ip mptcp limits set add_addr_accepted 1
```

2. **IP** 주소 **198.51.100.1** 을 서버의 새 **MPTCP** 엔드포인트로 추가합니다.

```
# ip mptcp endpoint add 198.51.100.1 dev enp1s0 signal
```

signal 옵션을 사용하면 3방향 핸드셰이크 후에 **ADD_ADDR** 패킷이 전송됩니다.

3. **iperf3** 서버를 시작합니다.

```
# iperf3 -s

Server listening on 5201
```

4. 클라이언트를 서버에 연결합니다.

```
# iperf3 -c 192.0.2.1 -t 3
```

검증

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) man page
- mptcpize(8) man page

34.4. MPTCP 애플리케이션의 여러 경로 영구적으로 구성

nmcli 명령을 사용하여 소스와 대상 시스템 간에 여러 하위 흐름을 영구적으로 설정하는 MultiPath TCP(MPTCP)를 구성할 수 있습니다. 하위 흐름은 다른 리소스, 대상에 대한 다양한 경로, 심지어 다른 네트워크를 사용할 수 있습니다. 이더넷(Ethernet), 가전성(Ethernet),ECDHE(Edistentity), so on. 결과적으로 네트워크 탄력성과 처리량이 증가하도록 결합된 연결을 수행할 수 있습니다.

서버는 다음 예제에서 네트워크 인터페이스를 사용합니다.

- enp4s0: 192.0.2.1/24
- enp1s0: 198.51.100.1/24
- enp7s0: 192.0.2.3/24

클라이언트는 예제에서는 다음 네트워크 인터페이스를 사용합니다.

- enp4s0f0: 192.0.2.2/24
- enp4s0f1: 198.51.100.2/24
- enp6s0: 192.0.2.5/24

사전 요구 사항

- 관련 인터페이스에서 기본 게이트웨이를 구성하셨습니다.

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. 선택 사항: 하위 흐름 제한의 **RHEL** 커널 기본값은 **2**입니다. 더 필요한 경우:

a. 다음 콘텐츠를 사용하여 **/etc/systemd/system/set_mptcp_limit.service** 파일을 생성합니다.

```
[Unit]
Description=Set MPTCP subflow limit to 3
After=network.target

[Service]
ExecStart=ip mptcp limits set subflows 3
Type=oneshot

[Install]
WantedBy=multi-user.target
```

oneshot 장치는 모든 부팅 과정에서 네트워크 **(network.target)**가 작동 한 후 **ip CHAPtcp** 제한을 설정된 하위 흐름 **3** 명령을 실행합니다.

ip CHAPtcp 제한 **set subflows 3** 명령은 각 연결에 대한 최대 *추가* 하위 흐름 수를 설정하므로 총 **4**개입니다. 최대 **3**개의 추가 하위 흐름을 추가할 수 있습니다.

b. **set_mptcp_limit** 서비스를 활성화합니다.

```
# systemctl enable --now set_mptcp_limit
```

3. 연결 집계에 사용할 모든 연결 프로필에서 **MPTCP**를 활성화합니다.

```
# nmcli connection modify <profile_name> connection.mptcp-flags
signal,subflow,also-without-default-route
```

connection.mptcp-flags 매개변수는 **MPTCP** 끝점 및 **IP** 주소 플래그를 구성합니다. **NetworkManager** 연결 프로필에서 **MPTCP**가 활성화된 경우 설정은 관련 네트워크 인터페이스

의 IP 주소를 MPTCP 엔드포인트로 구성합니다.

기본적으로 NetworkManager는 기본 게이트웨이가 없는 경우 IP 주소에 MPTCP 플래그를 추가하지 않습니다. 해당 검사를 바이패스하려면 also- default-route 플래그를 사용해야 합니다.

검증

1. MPTCP 커널 매개변수를 활성화했는지 확인합니다.

```
# sysctl net.mptcp.enabled
net.mptcp.enabled = 1
```

2. 기본값이 충분하지 않은 경우 하위 흐름 제한을 올바르게 설정했는지 확인합니다.

```
# ip mptcp limit show
add_addr_accepted 2 subflows 3
```

3. 주소별 MPTCP 설정을 올바르게 구성되었는지 확인합니다.

```
# ip mptcp endpoint show
192.0.2.1 id 1 subflow dev enp4s0
198.51.100.1 id 2 subflow dev enp1s0
192.0.2.3 id 3 subflow dev enp7s0
192.0.2.4 id 4 subflow dev enp3s0
...
```

추가 리소스

- nm-settings-nmcli(5)
- ip-mptcp(8)
- 34.1절. “MPTCP 이해”
- 다중 경로 TCP 이해: 끝점 및 향후 네트워킹의 고가용성

- RFC8684: 여러 주소로 다중 경로 작업용 TCP 확장
- Multipath TCP를 사용하여 중단을 보다 효과적으로 유지하고 대역폭을 늘리십시오.

34.5. MPTCP 하위 흐름 모니터링

다중 경로 TCP(MPTCP) 소켓의 라이프사이클은 복잡할 수 있습니다. 기본 MPTCP 소켓이 생성되어 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.

이 절차에서는 MPTCP 이벤트를 모니터링하는 방법을 설명합니다. 이를 위해 MPTCP 서버 애플리케이션을 시뮬레이션하고 클라이언트가 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 요구 사항이 아닙니다. 그러나 라우팅이 올바르게 구성되어 있고 클라이언트가 두 인터페이스를 통해 서버에 연결할 수 있는 것이 중요합니다.

사전 요구 사항

- 두 개의 네트워크 인터페이스가 있는 RHEL 클라이언트(예: 이더넷 및ECDHE 포함)
- 클라이언트는 두 인터페이스를 통해 서버에 연결할 수 있습니다.

- RHEL 서버

- 클라이언트와 서버 둘 다 RHEL 8.6 이상을 실행합니다.

절차

1. 클라이언트와 서버 모두에서 연결별 하위 흐름 제한을 1 로 설정합니다.

```
# ip mptcp limits set add_addr_accepted 0 subflows 1
```

2. 서버에서 MPTCP 서버 애플리케이션을 시뮬레이션하려면 TCP 소켓 대신 적용된 MPTCP 소켓을 사용하여 netcat (nc)을 수신 모드에서 시작합니다.

```
# nc -l -k -p 12345
```

-k 옵션을 사용하면 처음 허용된 연결 후 nc 가 리스너를 닫히지 않습니다. 이는 하위 흐름의 모니터링을 보여주는 데 필요합니다.

3. 클라이언트에서 다음을 수행합니다.

   a. 가장 낮은 메트릭으로 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 RHEL은 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
```

RHEL은 enp1s0 인터페이스와 관련 IP 주소를 이 연결의 소스로 사용합니다.

모니터링 터미널에서 ip CloudEventtcp 모니터 명령이 로그됩니다.

```
[      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 ID로 식별하고 나중에 동일한 소켓에서 MPTCP 이벤트를 연관시킬 수 있습니다.

d. 서버에 대한 nc 연결이 실행 중인 터미널에서 Enter 를 누릅니다. 이 첫 번째 데이터 패킷은 연결을 완전히 설정합니다. 데이터가 전송되지 않은 한 연결이 설정되지 않습니다.

모니터링 터미널에서 ipECDHEtcp는 이제 로그를 모니터링합니다.

```
[   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 4.6.145 에 대한 연결을 표시합니다.

```
# ss -taunp | grep ":12345"
tcp ESTAB  0  0        192.0.2.2:36444 192.0.2.1:12345
```

이 시점에는 서버에 대한 하나의 연결만 설정되었습니다.

f. 세 번째 터미널에서 다른 끝점을 생성합니다.

```
# ip mptcp endpoint add dev wlp1s0 192.0.2.3 subflow
```

이 명령은 이 명령에서 클라이언트의 IRQ 인터페이스의 이름 및 IP 주소를 설정합니다.

모니터링 터미널에서 ipECDHEtcp는 이제 로그를 모니터링합니다.

```
[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 필드에는 새 하위 흐름의 로컬 주소 **ID**가 표시되고 연결이 **NAT(**네트워크 주소 변환**)**를 사용하는 경우에도 이 하위 흐름을 식별합니다**. saddr4** 필드는 **ipECDHEtcp** 엔드포인트 **add** 명령의 끝점의 **IP** 주소와 일치합니다**.**

g. 선택 사항**:** 서버의 포트 **4.6.145** 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 명령은 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 흐름에서의 연결입니다**.**

h. 세 번째 터미널에서 끝점을 삭제합니다**.**

```
# ip mptcp endpoint delete id 2
```

ipECDHE tcp 모니터 출력의 **locid** 필드에서 **ID**를 사용하거나 **ipECDHEtcp** 엔드포인트 **show** 명령을 사용하여 끝점 **ID**를 검색합니다**.**

모니터링 터미널에서 **ipECDHEtcp**는 이제 로그를 모니터링합니다**.**

```
[    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** 눌러 세션을 종료합니다**.**

모니터링 터미널에서 **ipECDHEtcp**는 이제 로그를 모니터링합니다**.**

```
[       CLOSED] token=63c070d2
```

추가 리소스

- ip-mptcp(1) man page
- NetworkManager에서 여러 기본 게이트웨이를 관리하는 방법

34.6. 커널에서 MULTIPATH TCP 비활성화

커널에서 MPTCP 옵션을 명시적으로 비활성화할 수 있습니다.

절차

- ECDHE 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
```

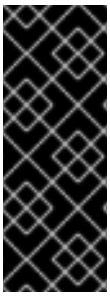
검증

- 커널에서ECDHE tcp.enabled 가 비활성화되어 있는지 확인합니다.

```
# sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 0
```

35장. 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.

35.1. 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 쉘 스크립트에 연결됩니다.

절차

- network-scripts 패키지를 설치합니다.

```
# yum install network-scripts
```

36장. IFCFG 파일로 IP 네트워킹 구성

인터페이스 구성(**ifcfg**) 파일은 개별 네트워크 장치의 소프트웨어 인터페이스를 제어합니다. 시스템이 부팅되면 이러한 파일을 사용하여 가져올 인터페이스와 구성 방법을 결정합니다. 이러한 파일의 이름은 **ifcfg-** ***name_pass*** 입니다. 여기서 접미사 이름은 구성 파일에서 제어하는 장치의 이름을 나타냅니다. 규칙에 따라 **ifcfg** 파일의 접미사는 구성 파일 자체에서 **DEVICE** 지시문으로 지정된 문자열과 동일합니다.

중요

NetworkManager는 키 파일 형식으로 저장된 프로필을 지원합니다. 그러나 기본적으로 **NetworkManager**는 **NetworkManager API**를 사용하여 프로필을 생성하거나 업데이트할 때 **ifcfg** 형식을 사용합니다.

향후 주요 **RHEL** 릴리스에서는 키 파일 형식이 기본값이 됩니다. 구성 파일을 수동으로 생성하고 관리하려면 키 파일 형식을 사용하는 것이 좋습니다. 자세한 내용은 키 파일 형식의 **NetworkManager** 연결 프로필을 참조하십시오.

36.1. IFCFG 파일을 사용하여 정적 네트워크 설정으로 인터페이스 구성

NetworkManager 유틸리티 및 애플리케이션을 사용하지 않는 경우 **ifcfg** 파일을 생성하여 네트워크 인터페이스를 수동으로 구성할 수 있습니다.

절차

- 이름이 **enp1s0** 인 인터페이스의 경우 **ifcfg** 파일을 사용하여 정적 네트워크 설정으로 인터페이스를 구성하려면 **/etc/sysconfig/network-scripts/** 디렉터리에 이름이 **ifcfg-enp1s0** 인 파일을 만듭니다.
  - **IPv4** 구성의 경우:

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=192.0.2.1
GATEWAY=192.0.2.254
```

  - **IPv6** 구성의 경우:

```
DEVICE=enp1s0
```

```
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=2001:db8:1::2/64
```

추가 리소스

- **nm-settings-ifcfg-rh(5) man page**

36.2. IFCFG 파일을 사용하여 동적 네트워크 설정으로 인터페이스 구성

NetworkManager 유틸리티 및 애플리케이션을 사용하지 않는 경우 **ifcfg** 파일을 생성하여 네트워크 인터페이스를 수동으로 구성할 수 있습니다.

절차

1. **ifcfg** 파일을 사용하여 동적 네트워크 설정으로 ***em1*** 이라는 인터페이스를 구성하려면 다음을 포함하는 **/etc/sysconfig/network-scripts/** 디렉터리에 이름이 **ifcfg-em1** 인 파일을 만듭니다.

```
DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes
```

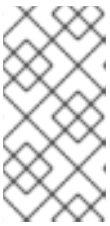
2. 전송할 인터페이스를 구성하려면 다음을 수행합니다.

- **DHCP** 서버에 다른 호스트 이름을 추가하고 다음 행을 **ifcfg** 파일에 추가합니다.

```
DHCP_HOSTNAME=hostname
```

- **DHCP** 서버에 다른 정규화된 도메인 이름(**FQDN**)을 사용하여 다음 행을 **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 서비스에서 지정된 **DNS** 서버로 **/etc/resolv.conf** 를 업데이트합니다. 하나의 **DNS** 서버 주소만 필요하며 다른 주소는 선택 사항입니다.

36.3. IFCFG 파일을 사용하여 시스템 전체 및 개인 연결 프로필 관리

기본적으로 호스트의 모든 사용자는 **ifcfg** 파일에 정의된 연결을 사용할 수 있습니다. **USERS** 매개변수를 **ifcfg** 파일에 추가하여 특정 사용자로 이 동작을 제한할 수 있습니다.

사전 요구 사항

- **ifcfg** 파일이 이미 있습니다.

절차

1. 특정 사용자로 제한하려는 **/etc/sysconfig/network-scripts/** 디렉터리에서 **ifcfg** 파일을 편집하고 다음을 추가합니다.

```
USERS="username1 username2 ..."
```

2. 연결을 다시 활성화합니다.

```
# nmcli connection up connection_name
```

37장. 키 파일 형식의 NETWORKMANAGER 연결 프로필

기본적으로 **NetworkManager**는 연결 프로필을 **ifcfg** 형식으로 저장하지만 키 파일 형식으로 프로필을 사용할 수도 있습니다. 더 이상 사용되지 않는 **ifcfg** 형식과 달리 키 파일 형식은 **NetworkManager**가 제공하는 모든 연결 설정을 지원합니다.

Red Hat Enterprise Linux 9에서는 키 파일 형식이 기본값입니다.

37.1. NETWORKMANAGER 프로필의 키 파일 형식

keyfile 형식은 **INI** 형식과 유사합니다. 예를 들어 다음은 키 파일 형식의 이더넷 연결 프로필입니다.

```
[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
```

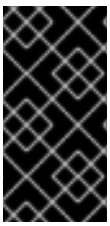
주의

매개 변수의 오타 또는 잘못된 배치로 인해 예기치 않은 동작이 발생할 수 있습니다. 따라서 **NetworkManager** 프로필을 수동으로 편집하거나 생성하지 마십시오.

nmcli 유틸리티, 네트워크 **RHEL** 시스템 역할 또는 **nmstate API**를 사용하여 **NetworkManager** 연결을 관리합니다. 예를 들어 **nmcli** 유틸리티를 오프라인 모드에서 사용하여 연결 프로필을 만들 수 있습니다.

각 섹션은 **nm-settings(5)** 및 **nm-settings-keyfile(5)** 매뉴얼 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 변수와 동일한 형식으로 키 파일에 저장됩니다. 그러나 주로 키 파일 구문을 더 쉽게 읽을 수 있도록 하는 예외가 있습니다. 이러한 예외 목록은 nm-settings-keyfile(5) 매뉴얼 페이지를 참조하십시오.



중요

보안상의 이유로 연결 프로필에 개인 키 및 암호와 같은 중요한 정보를 포함할 수 있으므로 NetworkManager는 root 사용자만 읽고 쓸 수 있는 구성 파일만 사용합니다.

연결 프로필의 목적에 따라 다음 디렉토리 중 하나에 저장합니다.

- /etc/NetworkManager/system-connections/: 영구 프로필의 위치. NetworkManager API를 사용하여 영구 프로필을 수정하는 경우 NetworkManager는 이 디렉터리의 파일을 쓰고 덮어씁니다.
- /run/NetworkManager/system-connections/: 시스템을 재부팅할 때 자동으로 제거되는 임시 프로필의 경우.
- /usr/lib/NetworkManager/system-connections/: 변경 불가능한 프로필의 경우. NetworkManager API를 사용하여 이러한 프로필을 편집하는 경우 NetworkManager는 이 프로필을 영구 스토리지 또는 임시 스토리지에 복사합니다.

NetworkManager는 디스크에서 프로파일을 자동으로 다시 로드하지 않습니다. 키 파일 형식으로 연결 프로필을 생성하거나 업데이트할 때 nmcli connection reload 명령을 사용하여 NetworkManager에 변경 사항을 알립니다.

37.2. NMCLI 를 사용하여 오프라인 모드에서 키 파일 연결 프로필 생성

nmcli, network RHEL 시스템 역할 또는 nmstate API와 같은 NetworkManager 유틸리티를 사용하여 NetworkManager 연결을 관리하고 구성 파일을 만들고 업데이트합니다. 그러나 nmcli --offline connection add 명령을 사용하여 오프라인 모드에서 키file 형식으로 다양한 연결 프로필을 생성할 수도 있습니다.

오프라인 모드를 사용하면 nmcli 가 NetworkManager 서비스 없이 작동하여 표준 출력을 통해 키 파일 연결 프로필을 생성할 수 있습니다. 이 기능은 다음과 같은 경우 유용할 수 있습니다.

- 다른 위치에서 사전 배포해야 하는 연결 프로필을 생성하려고 합니다. 예를 들어 컨테이너 이미지의 경우 또는 **RPM** 패키지로 사용할 수 있습니다.

- **NetworkManager** 서비스를 사용할 수 없는 환경에서 연결 프로필을 생성하려고 합니다. 예를 들어 **chroot** 유틸리티를 사용하려는 경우입니다. 또는 **Kickstart %post** 스크립트를 통해 설치할 **RHEL** 시스템의 네트워크 구성을 생성하거나 수정하려면 다음을 수행합니다.

다음 연결 프로필 유형을 생성할 수 있습니다.

- 고정 이더넷 연결
- 동적 이더넷 연결
- 네트워크 본딩
- 네트워크 브리지
- **VLAN** 또는 지원되는 연결의 종류

절차

1. 키 파일 형식으로 새 연결 프로필을 생성합니다. 예를 들어 **DHCP**를 사용하지 않는 이더넷 장치의 연결 프로필의 경우 유사한 **nmcli** 명령을 실행합니다.

```
# nmcli --offline connection add type ethernet con-name Example-Connection
ipv4.addresses 192.0.2.1/24 ipv4.dns 192.0.2.200 ipv4.method manual >
/etc/NetworkManager/system-connections/output.nmconnection
```

참고

con-name 키로 지정한 연결 이름은 생성된 프로필의 id 변수에 저장됩니다. nmcli 명령을 사용하여 나중에 이 연결을 관리하는 경우 다음과 같이 연결을 지정합니다.

- id 변수를 생략하지 않으면 연결 이름(예: Example-Connection)을 사용합니다.
- id 변수가 생략되면 .nmconnection 접미사 없이 파일 이름을 사용합니다(예: output).

2. root 사용자만 읽고 업데이트할 수 있도록 권한을 구성 파일에 설정합니다.

```
# chmod 600 /etc/NetworkManager/system-connections/output.nmconnection
# chown root:root /etc/NetworkManager/system-connections/output.nmconnection
```

3. NetworkManager 서비스를 시작합니다.

```
# systemctl start NetworkManager.service
```

4. 프로필의 autoconnect 변수를 false 로 설정하면 연결을 활성화합니다.

```
# nmcli connection up Example-Connection
```

검증

1. NetworkManager 서비스가 실행 중인지 확인합니다.

```
# systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
  Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor
preset: enabled)
  Active: active (running) since Wed 2022-08-03 13:08:32 CEST; 1min 40s ago
...
```

2. NetworkManager가 구성 파일에서 프로필을 읽을 수 있는지 확인합니다.

```
# nmcli -f TYPE,FILENAME,NAME connection
TYPE      FILENAME                                        NAME
ethernet /etc/NetworkManager/system-connections/output.nmconnection Example-
Connection
ethernet  /etc/sysconfig/network-scripts/ifcfg-enp1s0              enp1s0
...
```

출력에 새로 생성된 연결이 표시되지 않으면 **keyfile** 권한과 사용한 구문이 올바른지 확인합니다.

3. 연결 프로필을 표시합니다.

```
# nmcli connection show Example-Connection
connection.id:                          Example-Connection
connection.uuid:                        232290ce-5225-422a-9228-cb83b22056b4
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              --
connection.autoconnect:                 yes
...
```

추가 리소스

- **nmcli(1)**
- **nm-settings-keyfile(5)**
- **NetworkManager** 프로필의 키 파일 형식
- **nmcli**를 사용하여 이더넷 연결 구성
- **nmcli**를 사용하여 **VLAN** 태그 지정 설정
- **nmcli**를 사용하여 네트워크 브리지 구성
- **nmcli**를 사용하여 네트워크 본딩 구성

37.3. 키 파일 형식으로 NETWORKMANAGER 프로필을 수동으로 생성

키 파일 형식으로 NetworkManager 연결 프로필을 수동으로 생성할 수 있습니다.

참고

구성 파일을 수동으로 생성하거나 업데이트하면 예기치 않거나 작동하지 않는 네트워크 구성이 발생할 수 있습니다. 또는 offline 모드에서 nmcli 를 사용할 수 있습니다. nmcli 를 사용하여 오프라인 모드에서 키 파일 연결 프로필 만들기를참조하십시오.

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 DHCP를 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

.nmconnection 접미사로 모든 파일 이름을 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 나타냅니다.

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. NetworkManager가 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. 프로필의 autoconnect 변수를 false 로 설정하면 연결을 활성화합니다.

```
# nmcli connection up example_connection
```

검증

1. 연결 프로필을 표시합니다.

```
# nmcli connection show example_connection
```

추가 리소스

- nm-settings-keyfile (5)

37.4. IFCFG 및 키 파일 형식의 프로필과 인터페이스 이름 변경의 차이점

provider 또는 lan 과 같은 사용자 지정 네트워크 인터페이스 이름을 정의하여 인터페이스 이름을 보다 설명적으로 만들 수 있습니다. 이 경우 udev 서비스는 인터페이스의 이름을 바꿉니다. 이름 변경 프로세

스는 **ifcfg** 또는 키 파일 형식으로 연결 프로필을 사용하는지 여부에 따라 다르게 작동합니다.

ifcfg 형식으로 프로필을 사용할 때 인터페이스 이름 변경 프로세스

1. **/usr/lib/udev/rules.d/60-net.rules** 규칙에서는 **/lib/udev/rename_device** 도우미 유틸리티를 호출합니다.
2. 도우미 유틸리티는 **/etc/sysconfig/network-scripts/ifcfg-*** 파일에서 **HWADDR** 매개변수를 검색합니다.
3. 변수에 설정된 값이 인터페이스의 **MAC** 주소와 일치하는 경우 도우미 유틸리티는 인터페이스의 이름을 파일의 **DEVICE** 매개변수에 설정된 이름으로 변경합니다.

키 파일 형식으로 프로필을 사용할 때 인터페이스 이름 변경 프로세스

1. 인터페이스 이름을 변경할 **systemd** 링크 파일 또는 **udev** 규칙을 생성합니다.
2. **NetworkManager** 연결 프로필의 **interface-name** 속성에 사용자 지정 인터페이스 이름을 사용합니다.

추가 리소스

- **udev** 장치 관리자의 네트워크 인터페이스 이름 변경 방법
- **udev** 규칙을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성
- **systemd** 링크 파일을 사용하여 사용자 정의 네트워크 인터페이스 이름 구성

37.5. NETWORKMANAGER 프로필에서 IFCFG에서 키 파일 형식으로 마이그레이션

ifcfg 형식으로 연결 프로필을 사용하는 경우 기본 형식과 한 위치에 모든 프로필을 갖도록 키 파일 형식으로 변환할 수 있습니다.

참고

ifcfg 파일에 NM_CONTROLLED=no 설정이 포함된 경우 NetworkManager는 이 프로필을 제어하지 않으므로 마이그레이션 프로세스에서 해당 프로필을 무시합니다.

사전 요구 사항

- /etc/sysconfig/network-scripts/ 디렉터리에 ifcfg 형식으로 연결 프로필이 있습니다.
- 연결 프로필에 provider 또는 lan 과 같은 사용자 지정 장치 이름으로 설정된 DEVICE 변수가 포함된 경우 각 사용자 지정 장치 이름에 대한 systemd 링크 파일 또는 udev 규칙을 생성했습니다.

절차

- 연결 프로필을 마이그레이션합니다.

```
# nmcli connection migrate
Connection 'enp1s0' (43ed18ab-f0c4-4934-af3d-2b3333948e45) successfully migrated.
Connection 'enp2s0' (883333e8-1b87-4947-8ceb-1f8812a80a9b) successfully migrated.
...
```

검증

- 필요한 경우 모든 연결 프로필을 성공적으로 마이그레이션했는지 확인할 수 있습니다.

```
# nmcli -f TYPE,FILENAME,NAME connection
TYPE      FILENAME                                                  NAME
ethernet  /etc/NetworkManager/system-connections/enp1s0.nmconnection      enp1s0
ethernet  /etc/NetworkManager/system-connections/enp2s0.nmconnection      enp2s0
...
```

추가 리소스

- nm-settings-keyfile(5)
- nm-settings-ifcfg-rh(5)
- udev 장치 관리자의 네트워크 인터페이스 이름 변경 방법

38장. SYSTEMD 네트워크 대상 및 서비스

NetworkManager는 시스템 부팅 프로세스 중에 네트워크를 구성합니다. 그러나 루트 디렉터리가 iSCSI 장치에 저장된 경우와 같이 원격 루트(/)로 부팅하는 경우 RHEL을 시작하기 전에 네트워크 설정이 초기 RAM 디스크(initrd)에 적용됩니다. 예를 들어 rd.neednet=1 을 사용하여 커널 명령줄에 네트워크 구성이 지정되거나 원격 파일 시스템을 마운트하도록 구성이 지정된 경우 initrd 에 네트워크 설정이 적용됩니다.

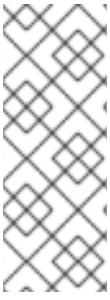
RHEL은 네트워크 설정을 적용하여 network 및 network-online 대상 및 NetworkManager-wait-online 서비스를 사용합니다. 또한 이러한 서비스를 동적으로 다시 로드할 수 없는 경우 네트워크를 완전히 사용할 수 있는 후 systemd 서비스를 시작할 수 있습니다.

38.1. 네트워크 및 네트워크 온라인 SYSTEMD 대상의 차이점

systemd는 네트워크 및 네트워크 온라인 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 단위는 network-online.target 으로 시작하고 일정 형태의 네트워크 연결이 설정될 때까지 대상이 지연됩니다. 네트워크가 연결될 때까지 네트워크 온라인 대상을 지연시킵니다.

네트워크 온라인 대상은 서비스를 시작하여 추가 실행에 지연을 추가합니다. systemd는 자동으로 Wants 에 종속 항목을 추가하고 이 대상 단위의 매개 변수를 $network 기능을 참조하는 Linux Standard Base(LSB) 헤더가 있는 모든 SysV(System V) init 스크립트 서비스 장치에 추가합니다. LSB 헤더는 init 스크립트의 메타데이터입니다. 이를 사용하여 종속성을 지정할 수 있습니다. 이는 systemd 대상과 유사합니다.

네트워크 대상은 부팅 프로세스 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 후에 순서가 지정된 서비스가 있는 경우 종료 중에 이 종속성이 취소됩니다. 네트워크가 서비스가 중지될 때까지 연결이 끊어지지 않습니다. 원격 네트워크 파일 시스템의 모든 마운트 단위는 네트워크 온라인 대상 장치를 자동으로 시작하고 그 후 자체적으로 정렬합니다.



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.

38.2. 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 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 -** 장치를 감지하기 위해 드라이버의 타임아웃을 설정합니다.
- **ipv4.may-fail** 및 **ipv6.may-fail -** 하나의 **IP** 주소 제품군이 준비되었는지 또는 특정 주소 제품군의 구성이 완료되었는지 여부를 설정하여 활성화를 설정합니다.
- **ipv4.gateway-ping-timeout - delays activation.**

추가 리소스

- **nm-settings(5)** 도움말 페이지

38.3. 네트워크를 시작한 후 시작하도록 SYSTEMD 서비스 구성

Red Hat Enterprise Linux는 **/usr/lib/ systemd /system/** 디렉터리에 **systemd** 서비스 파일을 설치합니다. 이 절차에서는 네트워크가 온라인 상태가 된 후 특정 서비스를 시작하기 위해 **/usr/lib/systemd/system/** 의 서비스 파일과 함께 사용되는 **/etc/systemd/system/*service_name*.*service*.d /**에 대한 드롭인 스니펫을 생성합니다. 드롭인 조각의 설정이 **/usr/lib/systemd/system/** 의 서비스 파일의 설정과 겹치는 경우 우선 순위가 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
```

39장. NMSTATE 소개

NMState는 선언적 네트워크 관리자 API입니다. nmstate 패키지는 RHEL에서 NetworkManager를 관리하기 위해 libnmstate Python 라이브러리와 명령줄 유틸리티 nmstatectl 을 제공합니다. Nmstate를 사용하는 경우 YAML 또는 JSON 형식의 지침을 사용하여 예상 네트워킹 상태를 설명합니다.

NMState는 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

39.1. PYTHON 애플리케이션에서 LIBNMSTATE 라이브러리 사용

libnmstate Python 라이브러리를 사용하면 개발자가 자체 애플리케이션에서 Nmstate를 사용할 수 있습니다.

라이브러리를 사용하려면 소스 코드로 가져옵니다.

```
import libnmstate
```

이 라이브러리를 사용하려면 nmstate 패키지를 설치해야 합니다.

예 39.1. libnmstate 라이브러리를 사용하여 네트워크 상태 쿼리

다음 Python 코드는 libnmstate 라이브러리를 가져오고 사용 가능한 네트워크 인터페이스와 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])
```

39.2. NMSTATECTL을 사용하여 현재 네트워크 구성 업데이트

nmstatectl 유틸리티를 사용하여 하나 또는 모든 인터페이스의 현재 네트워크 구성을 파일에 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.

예를 들어 **enp1s0** 인터페이스의 설정을 파일로 내보내고 구성을 수정한 다음 호스트에 설정을 적용할 수 있습니다.

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.

39.3. 네트워크 RHEL 시스템 역할의 네트워크 상태

네트워크 RHEL 시스템 역할은 플레이북의 상태 구성을 지원하여 장치를 구성합니다. 이를 위해 network_state 변수 다음에 상태 구성을 사용합니다.

플레이북에서 network_state 변수를 사용할 때의 이점:

- 선언적 메서드를 상태 구성과 함께 사용하면 인터페이스를 구성할 수 있으며 NetworkManager는 이러한 인터페이스에 대한 프로필을 백그라운드에서 만듭니다.
- network_state 변수를 사용하면 변경해야 하는 옵션을 지정할 수 있으며 다른 모든 옵션은 그대로 유지됩니다. 그러나 network_connections 변수를 사용하면 네트워크 연결 프로필을 변경하려면 모든 설정을 지정해야 합니다.

예를 들어 동적 IP 주소 설정을 사용하여 이더넷 연결을 생성하려면 플레이북에서 다음 vars 블록을 사용합니다.

상태 구성이 있는 플레이북	일반 플레이북

```
vars:
  network_state:
    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
```

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

예를 들어 위와 같이 생성한 동적 IP 주소 설정의 연결 상태만 변경하려면 플레이북에서 다음 vars 블록을 사용합니다.

상태 구성이 있는 플레이북

```
vars:
  network_state:
    interfaces:
    - name: enp7s0
      type: ethernet
      state: down
```

일반 플레이북

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

추가 리소스

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

39.4. 추가 리소스

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

40장. FIREWALLD 사용 및 구성

방화벽 은 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 을 통해 허용하는 데 사용됩니다.

firewalld 는 **D-Bus** 인터페이스를 사용하여 동적 사용자 지정 가능한 호스트 기반 방화벽을 제공하는 방화벽 서비스 데몬입니다. 동적이므로 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.

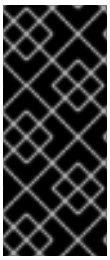
nftables 백엔드가 있는 **firewalld** 는 **--direct** 옵션을 사용하여 사용자 지정 **nftables** 규칙을 **firewalld** 에 전달하는 것을 지원하지 않습니다.

40.1. FIREWALLD, NFTABLES 또는 IPTABLES를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld:** 간단한 방화벽 활용 사례에 **firewalld** 유틸리티를 사용합니다. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.
- **nftables: nftables** 유틸리티를 사용하여 전체 네트워크에 대해 과 같이 복잡하고 성능이 중요한 방화벽을 설정합니다.
- **iptables: Red Hat Enterprise Linux**의 **iptables** 유틸리티는 기존 백엔드 대신 **nf_tables** 커널 **API** 를 사용합니다. **nf_tables API**는 **iptables** 명령을 사용하는 스크립트가 **Red Hat**

Enterprise Linux에서 계속 작동하도록 이전 버전과의 호환성을 제공합니다. 새 방화벽 스크립트의 경우 **Red Hat**은 **nftables** 를 사용하는 것이 좋습니다.



중요

다른 방화벽 관련 서비스(**firewalld**,**nftables** 또는 **iptables**)가 서로 영향을 미치지 않도록 하려면 **RHEL** 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

40.2. 방화벽 영역

firewalld 유틸리티를 사용하여 해당 네트워크 내의 인터페이스 및 트래픽과 함께 있는 신뢰 수준에 따라 네트워크를 다른 영역으로 분리할 수 있습니다. 연결은 하나의 영역의 일부일 수 있지만 많은 네트워크 연결에 해당 영역을 사용할 수 있습니다.

firewalld 는 영역과 관련하여 엄격한 원칙을 따릅니다.

1. 트래픽 수신은 하나의 영역만 포함됩니다.
2. 트래픽은 하나의 영역만 송신합니다.
3. 영역은 신뢰 수준을 정의합니다.
4. 기본적으로 **Intrazone** 트래픽(동일한 영역 내)이 허용됩니다.
5. 영역 간 트래픽은 기본적으로 거부됩니다.

규칙 **4**와 **5**는 원칙 **3**의 결과입니다.

원칙 **4**는 영역 옵션 **--remove-forward** 를 통해 구성할 수 있습니다. 원칙 **5**는 새로운 정책을 추가하여 구성할 수 있습니다.

NetworkManager 는 인터페이스 영역의 **firewalld** 에 알립니다. 다음 유틸리티를 사용하여 인터페이스에 영역을 할당할 수 있습니다.

- NetworkManager
- firewall-config 유틸리티
- firewall-cmd 유틸리티
- RHEL 웹 콘솔

RHEL 웹 콘솔, firewall-config 및 firewall-cmd 는 적절한 NetworkManager 구성 파일만 편집할 수 있습니다. 웹 콘솔, firewall-cmd 또는 firewall-config 를 사용하여 인터페이스 영역을 변경하면 요청이 NetworkManager 로 전달되고firewalld 에서 처리되지 않습니다.

/usr/lib/firewalld/zones/ 디렉터리는 사전 정의된 영역을 저장하고 사용 가능한 네트워크 인터페이스에 즉시 적용할 수 있습니다. 이러한 파일은 수정된 후에만 /etc/firewalld/zones/ 디렉토리에 복사됩니다. 사전 정의된 영역의 기본 설정은 다음과 같습니다.

블록

- 적합한 대상: 들어오는 네트워크 연결은 IPv4 및 IPv6-adm-prohibited에 대한 icmp-host-prohibited 메시지와 함께 거부됩니다.
- 허용: 시스템 내에서 시작된 네트워크 연결만 수행합니다.

dmz

- 적합한 대상: DMZ의 컴퓨터는 내부 네트워크에 대한 액세스 제한으로 공개적으로 액세스할 수 있습니다.
- 허용: 선택한 연결만 제공됩니다.

drop

적합한 대상: 들어오는 네트워크 패킷은 알림 없이 삭제됩니다.

- 허용: 나가는 네트워크 연결만 가능합니다.

external

- 적합한 대상: 특히 라우터에 대해 마스커레이딩이 활성화된 외부 네트워크입니다. 네트워크에서 다른 컴퓨터를 신뢰하지 않는 경우입니다.

- 허용: 선택한 연결만 제공됩니다.

홈

- 적합한 대상: 네트워크상의 다른 컴퓨터를 주로 신뢰하는 홈 환경.

- 허용: 선택한 연결만 제공됩니다.

internal

- 적합한 대상: 네트워크에 있는 다른 컴퓨터를 주로 신뢰하는 내부 네트워크입니다.

- 허용: 선택한 연결만 제공됩니다.

public

- 적합한 대상: 네트워크에서 다른 컴퓨터를 신뢰하지 않는 공용 영역입니다.

- 허용: 선택한 연결만 제공됩니다.

신뢰할 수 있는

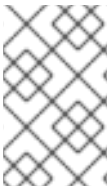
- 허용: 모든 네트워크 연결

work

적합한 대상: 네트워크에 있는 다른 컴퓨터를 주로 신뢰하는 작업 환경.

- 허용: 선택한 연결만 제공됩니다.

이러한 영역 중 하나는 *기본* 영역으로 설정됩니다. **NetworkManager** 에 인터페이스 연결이 추가되면 기본 영역에 할당됩니다. 설치 시 **firewalld** 의 기본 영역은 퍼블릭 영역입니다. 기본 영역을 변경할 수 있습니다.



참고

네트워크 영역 이름을 자체 설명하여 사용자가 신속하게 이해할 수 있도록 합니다.

보안 문제를 방지하려면 기본 영역 구성을 검토하고 요구 사항 및 위험 평가에 따라 불필요한 서비스를 비활성화합니다.

추가 리소스

- **firewalld.zone(5)** 도움말 페이지

40.3. 방화벽 정책

방화벽 정책은 원하는 네트워크 보안 상태를 지정합니다. 다양한 유형의 트래픽에 대해 수행할 규칙과 작업을 간략하게 설명합니다. 일반적으로 정책에는 다음 유형의 트래픽에 대한 규칙이 포함됩니다.

- 들어오는 트래픽
- 나가는 트래픽
- 전송 트래픽
- 특정 서비스 및 애플리케이션
- NAT(네트워크 주소 변환)

방화벽 정책은 방화벽 영역의 개념을 사용합니다. 각 영역은 허용되는 트래픽을 결정하는 특정 방화벽 규칙 세트와 연결됩니다. 정책은 상태 저장되지 않은 방식으로 방화벽 규칙을 적용합니다. 즉, 트래픽의 한 방향만 고려합니다. **firewalld** 의 상태 저장 필터링으로 인해 트래픽 반환 경로는 암시적으로 허용됩니다.

정책은 **Ingress** 영역 및 송신 영역과 연결됩니다. **Ingress** 영역은 트래픽이 시작된 위치(**received**)입니다. 송신 영역은 트래픽이 떠나는 위치입니다(**sent**).

정책에 정의된 방화벽 규칙은 방화벽 영역을 참조하여 여러 네트워크 인터페이스에 일관된 구성을 적용할 수 있습니다.

40.4. 방화벽 규칙

방화벽 규칙을 사용하여 네트워크 트래픽을 허용하거나 차단하는 특정 구성을 구현할 수 있습니다. 따라서 네트워크 트래픽 흐름을 제어하여 시스템을 보안 위협으로부터 보호할 수 있습니다.

방화벽 규칙은 일반적으로 다양한 속성을 기반으로 특정 기준을 정의합니다. 속성은 다음과 같습니다.

- 소스 **IP** 주소
- 대상 **IP** 주소
- 전송 프로토콜 (**TCP, UDP, ...**)
- 포트
- 네트워크 인터페이스

firewalld 유틸리티는 방화벽 규칙을 영역(예: 공용,내부 및 기타) 및 정책으로 구성합니다. 각 영역에는 특정 영역과 연결된 네트워크 인터페이스에 대한 트래픽 자유 수준을 결정하는 자체 규칙 세트가 있습니다.

40.5. 영역 구성 파일

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 *zone-name*.xml 이어야 합니다. 여기서 *zone-name* 의 길이는 현재 17ECDHE로 제한됩니다. 영역 구성 파일은 /usr/lib/firewalld/zones/ 및 /etc/firewalld/zones/ 디렉터리에 있습니다.

다음 예제에서는 TCP 및 UDP 프로토콜 모두에 대해 하나의 서비스(SSH)와 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>
```

추가 리소스

- firewalld.zone 매뉴얼 페이지

40.6. 사전 정의된 FIREWALLD 서비스

firewalld 서비스는 특정 애플리케이션 또는 네트워크 서비스에 대한 액세스를 정의하는 사전 정의된 방화벽 규칙 세트입니다. 각 서비스는 다음 요소의 조합을 나타냅니다.

- 로컬 포트
- 네트워크 프로토콜
- 연결된 방화벽 규칙
- 소스 포트 및 대상
- 서비스가 활성화된 경우 자동으로 로드되는 방화벽 도우미 모듈

서비스는 패킷 필터링을 단순화하고 한 번에 여러 작업을 수행하기 때문에 시간을 절약합니다. 예를 들어 **firewalld** 는 다음 작업을 한 번에 수행할 수 있습니다.

- 포트 열기
- 네트워크 프로토콜 정의
- 패킷 전달 활성화

서비스 구성 옵션 및 일반 파일 정보는 **firewalld.service(5)** 매뉴얼 페이지에 설명되어 있습니다. 서비스는 다음과 같은 형식으로 이름이 지정된 개별 **XML** 구성 파일인 ***service-name*.xml** 을 통해 지정됩니다. 프로토콜 이름은 **firewalld** 에서 서비스 또는 애플리케이션 이름보다 우선합니다.

다음과 같은 방법으로 **firewalld** 를 구성할 수 있습니다.

- 유틸리티 사용:
  - **firewall-config -** 그래픽 유틸리티
  - **firewall-cmd -** 명령줄 유틸리티
  - **firewall-offline-cmd -** 명령줄 유틸리티
- **/etc/firewalld/services/** 디렉토리에서 **XML** 파일을 편집합니다.

  서비스를 추가하거나 변경하지 않으면 **/etc/firewalld/services/** 에 해당 **XML** 파일이 존재하지 않습니다. **/usr/lib/firewalld/services/** 의 파일을 템플릿으로 사용할 수 있습니다.

추가 리소스

- **firewalld.service(5)** 매뉴얼 페이지

40.7. 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.

40.7.1. 보안을 강화하기 위해 특정 영역에 대한 방화벽 설정 사용자 정의

방화벽 설정을 수정하고 특정 네트워크 인터페이스 또는 특정 방화벽 영역과 연결하여 네트워크 보안을 강화할 수 있습니다. 영역에 대한 세분화된 규칙 및 제한을 정의하면 원하는 보안 수준에 따라 인바운드 및 아웃바운드 트래픽을 제어할 수 있습니다.

예를 들어 다음과 같은 이점을 얻을 수 있습니다.

- 민감한 데이터 보호
- 무단 액세스 방지
- 잠재적인 네트워크 위협 완화

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 사용 가능한 방화벽 영역을 나열합니다.

```
# firewall-cmd --get-zones
```

firewall-cmd --get-zones 명령은 시스템에서 사용할 수 있는 모든 영역을 표시하지만 특정 영역에 대한 세부 정보는 표시하지 않습니다. 모든 영역에 대한 자세한 정보를 보려면 **firewall-cmd --list-all-zones** 명령을 사용합니다.

2. 이 구성에 사용할 영역을 선택합니다.

3. 선택한 영역에 대한 방화벽 설정을 수정합니다. 예를 들어 **SSH** 서비스를 허용하고 **ftp** 서비스를 제거하려면 다음을 수행합니다.

```
# firewall-cmd --add-service=ssh --zone=<your_chosen_zone>
# firewall-cmd --remove-service=ftp --zone=<same_chosen_zone>
```

4. 방화벽 영역에 네트워크 인터페이스를 할당합니다.

a. 사용 가능한 네트워크 인터페이스를 나열합니다.

```
# firewall-cmd --get-active-zones
```

영역의 작업은 해당 구성과 일치하는 네트워크 인터페이스 또는 소스 주소 범위가 있는지에 따라 결정됩니다. 기본 영역은 분류되지 않은 트래픽에 대해 활성 상태이지만 트래픽이 규칙과 일치하지 않는 경우 항상 활성 상태인 것은 아닙니다.

b. 선택한 영역에 네트워크 인터페이스를 할당합니다.

```
# firewall-cmd --zone=<your_chosen_zone> --change-interface=<interface_name> --permanent
```

영역에 네트워크 인터페이스를 할당하는 것은 특정 인터페이스(물리적 또는 가상)의 모든 트래픽에 일관된 방화벽 설정을 적용하는 데 더 적합합니다.

firewall-cmd 명령을 **--permanent** 옵션과 함께 사용하는 경우 종종 **NetworkManager** 연결 프로필을 업데이트하여 방화벽 구성을 영구적으로 변경해야 합니다. **firewalld** 와 **NetworkManager** 간의 통합은 일관된 네트워크 및 방화벽 설정을 보장합니다.

검증

1. 선택한 영역에 대한 업데이트된 설정을 표시합니다.

```
# firewall-cmd --zone=<your_chosen_zone> --list-all
```

명령 출력은 할당된 서비스, 네트워크 인터페이스 및 네트워크 연결(소스)을 포함한 모든 영역 설정을 표시합니다.

40.7.2. 기본 영역 변경

시스템 관리자는 구성 파일의 네트워킹 인터페이스에 영역을 할당합니다. 인터페이스가 특정 영역에 할당되지 않은 경우 기본 영역에 할당됩니다. **firewalld** 서비스를 다시 시작한 후 **firewalld** 는 기본 영역에 대한 설정을 로드하여 활성화합니다. 다른 모든 영역에 대한 설정은 유지되며 사용할 준비가 되어 있습니다.

일반적으로 영역은 **NetworkManager** 연결 프로필의 **connection.zone** 설정에 따라 **NetworkManager**에 의해 인터페이스에 할당됩니다. 또한 재부팅 후 **NetworkManager**는 해당 영역의 "활성화" 할당을 관리합니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

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** 옵션 없이 영구적인 설정입니다.

40.7.3. 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 구분하도록 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
```

40.7.4. nmcli를 사용하여 연결에 영역 할당

nmcli 유틸리티를 사용하여 **NetworkManager** 연결에 **firewalld** 영역을 추가할 수 있습니다.

절차

1. **NetworkManager** 연결 프로필에 영역을 할당합니다.

```
# nmcli connection modify profile connection.zone zone_name
```

2. 연결을 활성화합니다.

```
# nmcli connection up profile
```

40.7.5. 연결 프로필 파일에서 네트워크 연결에 수동으로 영역 할당

nmcli 유틸리티를 사용하여 연결 프로필을 수정할 수 없는 경우 프로필의 해당 파일을 수동으로 편집하여 **firewalld** 영역을 할당할 수 있습니다.

참고

firewalld 영역을 할당하도록 **nmcli** 유틸리티를 사용하여 연결 프로필을 수정하는 것이 더 효율적입니다. 자세한 내용은 영역에 네트워크 인터페이스 할당을 참조하십시오.

절차

1. 연결 프로필의 경로와 해당 형식을 결정합니다.

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

NetworkManager는 서로 다른 연결 프로필 형식에 대해 별도의 디렉터리 및 파일 이름을 사용합니다.

- /etc/NetworkManager/system-connections/ <*connection_name* > .nmconnection 파일의 프로필은 keyfile 형식을 사용합니다.
- /etc/sysconfig/network-scripts/ifcfg- <*interface_name* > 파일의 프로필은 ifcfg 형식을 사용합니다.

2. 형식에 따라 해당 파일을 업데이트합니다.

- 파일이 키 파일 형식을 사용하는 경우 zone= <*name* >을 /etc/NetworkManager/system-connections/ <connection_*name*> *.nmconnection* 파일의 [connection] 섹션에 추가합니다.

```
[connection]
...
zone=internal
```

- 파일에서 ifcfg 형식을 사용하는 경우 ZONE= <*name* >을 /etc/sysconfig/network-scripts/ifcfg- <*interface_name* > 파일에 추가합니다.

```
ZONE=internal
```

3. 연결 프로필을 다시 로드합니다.

```
# nmcli connection reload
```

4. 연결 프로필 다시 활성화

```
# nmcli connection up <profile_name>
```

검증

- 인터페이스 영역을 표시합니다. 예를 들면 다음과 같습니다.

```
# firewall-cmd --get-zone-of-interface enp1s0
internal
```

40.7.6. ifcfg 파일에서 네트워크 연결에 수동으로 영역을 할당

NetworkManager 에서 연결을 관리하는 경우 해당 연결을 사용하는 영역을 알고 있어야 합니다. 모든 네트워크 연결 프로필의 경우 이동식 장치가 있는 컴퓨터의 위치에 따라 다양한 방화벽 설정의 유연성을 제공하는 영역을 지정할 수 있습니다. 따라서 회사 또는 집과 같은 다른 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
```

40.7.7. 새 영역 생성

사용자 지정 영역을 사용하려면 새 영역을 생성하고 사전 정의된 영역처럼 사용합니다. 새 영역에는 **--permanent** 옵션이 필요합니다. 그러지 않으면 명령이 작동하지 않습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 새 영역을 생성합니다.

```
# firewall-cmd --permanent --new-zone=zone-name
```

2. 새 영역을 사용할 수 있도록 설정합니다.

```
# firewall-cmd --reload
```

명령은 이미 실행 중인 네트워크 서비스를 중단하지 않고 최근 방화벽 구성에 변경 사항을 적용합니다.

검증

- 새 영역이 영구 설정에 추가되었는지 확인합니다.

```
# firewall-cmd --get-zones --permanent
```

40.7.8. 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. 4가지 옵션이 있습니다.

- **ACCEPT:** 특정 규칙에 의해 허용된 항목을 제외하고 들어오는 모든 패킷을 수락합니다.
- **REJECT:** 특정 규칙에서 허용하는 경우를 제외하고 들어오는 모든 패킷을 거부합니다. **firewalld** 에서 패킷을 거부하면 소스 시스템에 거부에 대한 정보가 표시됩니다.
- **DROP:** 특정 규칙에서 허용하는 항목을 제외하고 들어오는 모든 패킷을 삭제합니다. **firewalld** 가 패킷을 삭제할 때 소스 시스템에 패킷 드롭에 대한 정보가 표시되지 않습니다.
- 기본값: **REJECT** 와 유사하지만 특정 시나리오에서 특별한 의미가 있습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

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

40.8. FIREWALLD를 사용하여 네트워크 트래픽 제어

firewalld 패키지는 많은 수의 사전 정의된 서비스 파일을 설치하고 더 많은 서비스를 추가하거나 사용자 지정할 수 있습니다. 그런 다음 이러한 서비스 정의를 사용하여 사용하는 프로토콜과 포트 번호를 몰라도 서비스에 대한 포트를 열거나 닫을 수 있습니다.

40.8.1. 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.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. **firewalld** 의 서비스가 아직 허용되지 않았는지 확인합니다.

```
# firewall-cmd --list-services
ssh dhcpv6-client
```

명령은 기본 영역에서 활성화된 서비스를 나열합니다.

2. **firewalld** 에서 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 ...
```

명령은 기본 영역에 사용 가능한 서비스 목록을 표시합니다.

3. **firewalld** 에서 허용하는 서비스 목록에 서비스를 추가합니다.

```
# firewall-cmd --add-service=<service_name>
```

명령은 지정된 서비스를 기본 영역에 추가합니다.

4. 새 설정을 영구적으로 설정합니다.

```
# firewall-cmd --runtime-to-permanent
```

명령은 이러한 런타임 변경 사항을 방화벽의 영구 구성에 적용합니다. 기본적으로 이러한 변경 사항은 기본 영역의 구성에 적용됩니다.

검증

1. 모든 영구 방화벽 규칙을 나열합니다.

```
# firewall-cmd --list-all --permanent
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

명령은 기본 방화벽 영역(공용)의 영구 방화벽 규칙을 사용하여 전체 구성을 표시합니다.

2. **firewalld** 서비스의 영구 구성의 유효성을 확인합니다.

```
# firewall-cmd --check-config
success
```

영구 구성이 유효하지 않으면 명령에서 추가 세부 정보와 함께 오류를 반환합니다.

```
# firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
```

영구 구성 파일을 수동으로 검사하여 설정을 확인할 수도 있습니다. 기본 설정 파일은 **/etc/firewalld/firewalld.conf** 입니다. 영역별 구성 파일은 **/etc/firewalld/zones/** 디렉터리에 있으며 정책은 **/etc/firewalld/policies/** 디렉터리에 있습니다.

40.8.2. GUI를 사용하여 사전 정의된 서비스로 트래픽 제어

그래픽 사용자 인터페이스를 사용하여 사전 정의된 서비스로 네트워크 트래픽을 제어할 수 있습니다. 방화벽 구성 애플리케이션은 명령줄 유틸리티에 대한 액세스 가능하고 사용자에게 친숙한 대안을 제공합니다.

사전 요구 사항

- **firewall-config** 패키지가 설치되어 있어야 합니다.
- **firewalld** 서비스가 실행 중입니다.

절차

1. 사전 정의된 또는 사용자 지정 서비스를 활성화하거나 비활성화하려면 다음을 수행합니다.
   a. **firewall-config** 유틸리티를 시작하고 서비스를 구성할 네트워크 영역을 선택합니다.
   b. **Zones** 탭을 선택한 다음 아래의 **Services** 탭을 선택합니다.

c.

신뢰할 각 서비스 유형에 대한 확인란을 선택하거나 선택한 영역에서 서비스를 차단하는 확인란을 지웁니다.

2.

서비스를 편집하려면 다음을 수행합니다.

a.

firewall-config 유틸리티를 시작합니다.

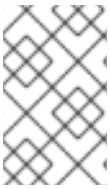
b.

Configuration (구성)이라는 메뉴에서 **Permanent** 를 선택합니다. **Services** (서비스) 창의 아래쪽에 추가 아이콘 및 메뉴 버튼이 표시됩니다.

c.

구성할 서비스를 선택합니다.

Ports,Protocols, Source Port 탭을 사용하면 선택한 서비스의 포트, 프로토콜, 소스 포트를 추가, 변경, 제거할 수 있습니다. 모듈 탭은**ECDHE** 도우미 모듈을 구성하는 데 사용됩니다. 대상 탭에서는 특정 대상 주소 및 인터넷 프로토콜(**IPv4** 또는 **IPv6**)으로의 트래픽을 제한할 수 있습니다.

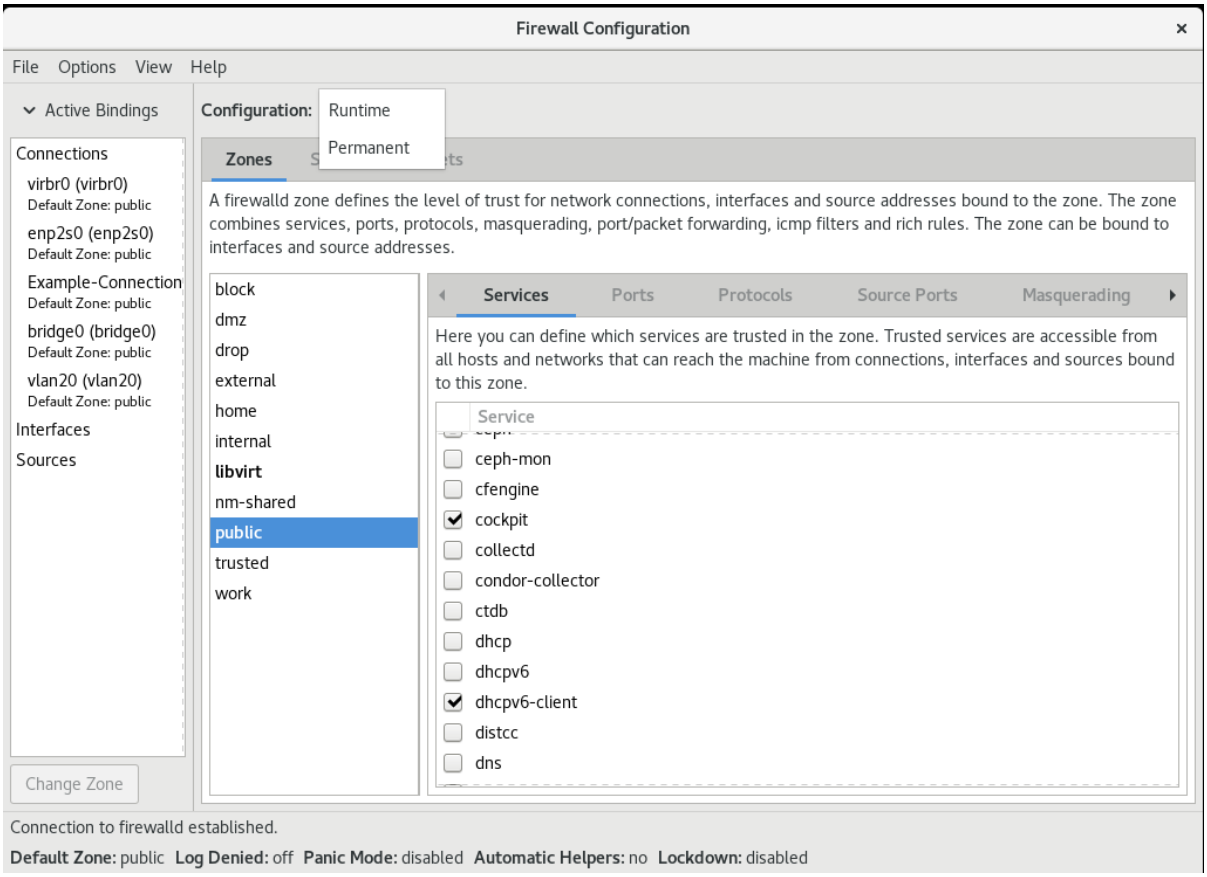


참고

런타임 모드에서는 서비스 설정을 변경할 수 없습니다.

검증

- **Super** 키를 눌러 활동 개요를 입력합니다.
- **Firewall Configuration** 유틸리티를 선택합니다.
  - **firewall-config** 명령을 입력하여 명령줄을 사용하여 그래픽 방화벽 구성 유틸리티를 시작할 수도 있습니다.
- 방화벽 구성 목록을 확인합니다.



방화벽 구성 창이 열립니다. 이 명령은 일반 사용자로 실행할 수 있지만, 관리자 암호를 묻는 메시지가 표시되는 경우가 있습니다.

40.8.3. 보안 웹 서버 호스팅을 허용하도록 firewalld 구성

포트는 운영 체제가 네트워크 트래픽을 수신 및 구분하고 시스템 서비스로 전달할 수 있는 논리 서비스입니다. 시스템 서비스는 포트에서 수신 대기하고 이 포트로 들어오는 모든 트래픽을 대기하는 데몬으로 표시됩니다.

일반적으로 시스템 서비스는 예약된 표준 포트에서 수신 대기합니다. 예를 들어 **httpd** 데몬은 포트 **80**에서 수신 대기합니다. 그러나 시스템 관리자는 서비스 이름 대신 포트 번호를 직접 지정할 수 있습니다.

firewalld 서비스를 사용하여 데이터를 호스팅하기 위해 보안 웹 서버에 대한 액세스를 구성할 수 있습니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 현재 활성화된 방화벽 영역을 확인합니다.

```
# firewall-cmd --get-active-zones
```

2. 적절한 영역에 **HTTPS** 서비스를 추가합니다.

```
# firewall-cmd --zone=<zone_name> --add-service=https --permanent
```

3. 방화벽 구성을 다시 로드합니다.

```
# firewall-cmd --reload
```

검증

1. **firewalld** 에서 포트가 열려 있는지 확인합니다.

- 포트 번호를 지정하여 포트를 연 경우 다음을 입력합니다.

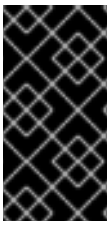
```
# firewall-cmd --zone=<zone_name> --list-all
```

- 서비스 정의를 지정하여 포트를 연 경우 다음을 입력합니다.

```
# firewall-cmd --zone=<zone_name> --list-services
```

40.8.4. 네트워크 보안을 강화하기 위해 사용되지 않거나 불필요한 포트 종료

열려 있는 포트가 더 이상 필요하지 않으면 **firewalld** 유틸리티를 사용하여 이를 종료할 수 있습니다.



중요

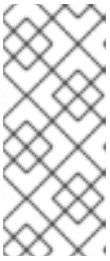
불필요한 모든 포트를 닫고 잠재적인 공격 면적을 줄이고 무단 액세스 또는 취약점 악용 위험을 최소화합니다.

절차

1. 허용되는 모든 포트를 나열합니다.

```
# firewall-cmd --list-ports
```

기본적으로 이 명령은 기본 영역에서 활성화된 포트를 나열합니다.



참고

이 명령은 포트로 열려 있는 포트 목록만 제공합니다. 서비스로 열려 있는 열려 있는 포트는 볼 수 없습니다. 이 경우 **--list-ports** 대신 **--list-all** 옵션을 사용하는 것이 좋습니다.

2. 허용되는 포트 목록에서 포트를 제거하여 들어오는 트래픽에 대해 종료합니다.

```
# firewall-cmd --remove-port=port-number/port-type
```

이 명령은 영역에서 포트를 제거합니다. 영역을 지정하지 않으면 기본 영역에서 포트가 제거됩니다.

3. 새 설정을 영구적으로 설정합니다.

```
# firewall-cmd --runtime-to-permanent
```

영역을 지정하지 않으면 이 명령은 런타임 변경 사항을 기본 영역의 영구 구성에 적용합니다.

검증

1. 활성 영역을 나열하고 검사할 영역을 선택합니다.

```
# firewall-cmd --get-active-zones
```

2. 선택한 영역에서 현재 열려 있는 포트를 나열하여 사용되지 않거나 불필요한 포트가 닫혀 있는지 확인합니다.

```
# firewall-cmd --zone=<zone_to_inspect> --list-ports
```

40.8.5. CLI를 통한 트래픽 제어

firewall-cmd 명령을 사용하여 다음을 수행할 수 있습니다.

- 네트워킹 트래픽 비활성화
- 네트워킹 트래픽 활성화

예를 들어 시스템 방어 기능을 강화하거나 데이터 개인 정보를 보장하거나 네트워크 리소스를 최적화할 수 있습니다.

중요

패닉 모드를 활성화하면 모든 네트워킹 트래픽이 중지됩니다. 따라서 시스템에 대한 물리적 액세스 권한이 있거나 직렬 콘솔을 사용하여 로그인하는 경우에만 사용해야 합니다.

절차

1. 네트워킹 트래픽을 즉시 비활성화하려면 에서 패닉 모드를 전환합니다.

```
# firewall-cmd --panic-on
```

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
```

40.8.6. GUI를 사용하여 프로토콜로 트래픽 제어

특정 프로토콜을 사용하여 방화벽을 통한 트래픽을 허용하려면 GUI를 사용할 수 있습니다.

사전 요구 사항

- firewall-config 패키지가 설치되어 있어야 합니다.

절차

1. firewall-config 도구를 시작하고 설정을 변경할 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.

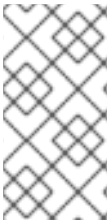
40.9. 소스에 따라 영역을 사용하여 들어오는 트래픽 관리

영역을 사용하여 소스에 따라 들어오는 트래픽을 관리할 수 있습니다. 이 컨텍스트에서 들어오는 트래픽은 시스템을 대상으로 하거나 firewalld 를 실행하는 호스트를 통과하는 모든 데이터입니다. 소스는 일반적으로 트래픽이 시작된 IP 주소 또는 네트워크 범위를 나타냅니다. 결과적으로 들어오는 트래픽을 정렬하고 다른 영역에 할당하여 해당 트래픽으로 연결할 수 있는 서비스를 허용하거나 허용하지 않을 수 있습니다.

소스 주소의 일치는 인터페이스 이름별 일치보다 우선합니다. 영역에 소스를 추가하면 방화벽은 인터페이스 기반 규칙을 통한 들어오는 트래픽에 대한 소스 기반 규칙의 우선 순위를 지정합니다. 즉, 들어오는 트래픽이 특정 영역에 지정된 소스 주소와 일치하는 경우 해당 소스 주소와 연결된 영역에 도달하는 인터페이스에 관계없이 트래픽이 처리되는 방법이 결정됩니다. 반면 인터페이스 기반 규칙은 일반적으로 특정 소스 기반 규칙과 일치하지 않는 트래픽에 대한 폴백입니다. 이러한 규칙은 소스가 영역과 명시적으로 연결되어 있지 않은 트래픽에 적용됩니다. 이를 통해 특정 소스 정의 영역이 없는 트래픽에 대한 기본 동작을 정의할 수 있습니다.

40.9.1. 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
```

40.9.2. 소스 제거

영역에서 소스를 제거하면 소스에서 시작된 트래픽은 더 이상 해당 소스에 지정된 규칙을 통해 전달되지 않습니다. 대신 트래픽이 시작된 인터페이스와 연결된 영역의 규칙 및 설정으로 대체되거나 기본 영역으로 이동합니다.

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
```

40.9.3. 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>
```

40.9.4. 특정 도메인에만 서비스를 허용하려면 영역 및 소스를 사용합니다.

특정 네트워크의 트래픽이 시스템에서 서비스를 사용하도록 허용하려면 영역 및 소스를 사용합니다. 다음 절차에서는 다른 모든 트래픽이 차단되는 동안 192.0.2.0/24 네트워크의 HTTP 트래픽만 허용합니다.

주의

이 시나리오를 구성할 때 기본 대상이 있는 영역을 사용합니다. 대상이 ACCEPT 로 설정된 영역을 사용하면 192.0.2.0/24 의 트래픽의 경우 모든 네트워크 연결이 허용되기 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. 내부 영역에 IP 범위를 추가하여 소스에서 영역을 통해 발생하는 트래픽을 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

40.10. 영역 간 전달된 트래픽 필터링

firewalld 를 사용하면 서로 다른 firewalld 영역 간의 네트워크 데이터 흐름을 제어할 수 있습니다. 규칙과 정책을 정의하면 이러한 영역 간에 이동할 때 트래픽이 허용되거나 차단되는 방법을 관리할 수 있습

니다.

정책 오브젝트 기능은 **firewalld** 에서 전달 및 출력 필터링 기능을 제공합니다. **firewalld** 를 사용하여 다른 영역 간 트래픽을 필터링하여 로컬 호스트 **VM**에 대한 액세스를 통해 호스트를 연결할 수 있습니다.

40.10.1. 정책 오브젝트와 영역 간의 관계

정책 오브젝트를 사용하면 사용자가 서비스, 포트 및 리치 규칙과 같은 **firewalld** 기본 기능을 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 심볼 릭 영역은 모든 현재 및 향후 영역에 정책을 적용합니다. 모든 심볼릭 영역은 모든 영역에 대해 와일드카드 역할을 합니다.

40.10.2. 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.

낮은 숫자 우선순위 값은 우선순위가 높고 먼저 적용됩니다.

40.10.3. 정책 오브젝트를 사용하여 로컬 호스트 컨테이너와 호스트에 물리적으로 연결된 네트워크 간의 트래픽을 필터링

정책 오브젝트 기능을 사용하면 사용자가 **Podman**과 **firewalld** 영역 간의 트래픽을 필터링할 수 있습니다.

참고

Red Hat은 기본적으로 모든 트래픽을 차단하고 **Podman** 유틸리티에 필요한 선택적 서비스를 여는 것이 좋습니다.

절차

1. 새 방화벽 정책을 생성합니다.

```
# firewall-cmd --permanent --new-policy podmanToAny
```

2. **Podman**에서 다른 영역으로의 모든 트래픽을 차단하고 **Podman**에서 필요한 서비스만 허용합니다.

```
# firewall-cmd --permanent --policy podmanToAny --set-target REJECT
# firewall-cmd --permanent --policy podmanToAny --add-service dhcp
# firewall-cmd --permanent --policy podmanToAny --add-service dns
# firewall-cmd --permanent --policy podmanToAny --add-service https
```

3. 새 **Podman** 영역을 생성합니다.

```
# firewall-cmd --permanent --new-zone=podman
```

4. 정책의 수신 영역을 정의합니다.

```
# firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

5. 다른 모든 영역에 대한 송신 영역을 정의합니다.

```
# firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

송신 영역을 **ANY**로 설정하면 **Podman**에서 다른 영역으로 필터링합니다. 호스트에 필터링하려면 송신 영역을 **HOST**로 설정합니다.

6. **firewalld** 서비스를 다시 시작합니다.

```
# systemctl restart firewalld
```

검증

- **Podman** 방화벽 정책을 다른 영역에 확인합니다.

```
# firewall-cmd --info-policy podmanToAny
podmanToAny (active)
  ...
  target: REJECT
  ingress-zones: podman
  egress-zones: ANY
  services: dhcp dns https
  ...
```

40.10.4. 정책 오브젝트의 기본 대상 설정

정책에 대해 **--set-target** 옵션을 지정할 수 있습니다. 다음 대상을 사용할 수 있습니다.

- **ACCEPT -** 패킷을 수락
- **DROP -** 원하지 않는 패킷을 삭제합니다.
- **REJECT - ICMP** 응답을 사용하여 원하지 않는 패킷을 거부
- **CONTINUE** (기본값) **-** 패킷에는 다음 정책 및 영역의 규칙이 적용됩니다.

```
# firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

검증

- 정책에 대한 정보 확인

```
# firewall-cmd --info-policy mypolicy
```

40.10.5. DNAT를 사용하여 HTTPS 트래픽을 다른 호스트로 전달

웹 서버가 개인 IP 주소가 있는 DMZ에서 실행되는 경우 인터넷의 클라이언트가 이 웹 서버에 연결할 수 있도록 대상 네트워크 주소 변환(DNAT)을 구성할 수 있습니다. 이 경우 웹 서버의 호스트 이름이 라우터의 공용 IP 주소로 확인됩니다. 클라이언트에서 라우터의 정의된 포트에 대한 연결을 설정하면 라우터가 패킷을 내부 웹 서버로 전달합니다.

사전 요구 사항

- DNS 서버는 웹 서버의 호스트 이름을 라우터의 IP 주소로 확인합니다.
- 다음 설정을 알고 있습니다.
  - 전달할 개인 IP 주소 및 포트 번호입니다.
  - 사용할 IP 프로토콜
  - 패킷을 리디렉션할 웹 서버의 대상 IP 주소 및 포트

절차

1. 방화벽 정책을 생성합니다.

```
# firewall-cmd --permanent --new-policy <example_policy>
```

영역과 달리 정책은 입력, 출력 및 전달된 트래픽에 대한 패킷 필터링을 허용합니다. 로컬에서 웹 서버, 컨테이너 또는 가상 머신에서 끝점으로 트래픽을 전달하려면 이러한 기능이 필요하므로 중요합니다.

2. 라우터 자체에서 로컬 IP 주소에 연결하고 이 트래픽을 전달할 수 있도록 수신 및 송신 트래픽에 대한 심볼릭 영역을 구성합니다.

```
# firewall-cmd --permanent --policy=<example_policy> --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=<example_policy> --add-egress-zone=ANY
```

--add-ingress-zone=HOST 옵션은 로컬에서 생성되어 로컬 호스트에서 전송된 패킷을 나타냅니다. **--add-egress-zone=ANY** 옵션은 모든 영역으로 이동하는 트래픽을 나타냅니다.

3. 트래픽을 웹 서버로 전달하는 리치 규칙을 추가합니다.

```
# firewall-cmd --permanent --policy=<example_policy> --add-rich-rule='rule
family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp"
to-port="443" to-addr="192.51.100.20"'
```

리치 규칙은 라우터의 **IP** 주소에 있는 포트 **443**에서 웹 서버(**192.51.100.20**)의 **IP** 주소의 포트 **443**으로 **TCP** 트래픽을 전달합니다.

4. 방화벽 구성 파일을 다시 로드합니다.

```
# firewall-cmd --reload
success
```

5. 커널에서 **127.0.0.0/8**의 라우팅을 활성화합니다.

- 영구 변경 사항의 경우 다음을 실행합니다.

```
# echo "net.ipv4.conf.all.route_localnet=1" > /etc/sysctl.d/90-enable-route-
localnet.conf
```

이 명령은 **route_localnet** 커널 매개 변수를 영구적으로 구성하고 시스템이 재부팅된 후 설정이 보존되도록 합니다.

- 시스템 재부팅없이 설정을 즉시 적용하려면 다음을 실행합니다.

```
# sysctl -p /etc/sysctl.d/90-enable-route-localnet.conf
```

sysctl 명령은 변경 사항을 적용하는 데 유용하지만 시스템 재부팅 시 구성은 유지되지 않습니다.

검증

1. 라우터의 IP 주소 및 웹 서버로 전달된 포트에 연결합니다.

```
# curl https://192.0.2.1:443
```

2. 선택 사항: net.ipv4.conf.all.route_localnet 커널 매개변수가 활성화되어 있는지 확인합니다.

```
# sysctl net.ipv4.conf.all.route_localnet
net.ipv4.conf.all.route_localnet = 1
```

3. 가 <example_policy> 활성화되어 있고 필요한 설정, 특히 소스 IP 주소와 포트, 사용할 프로토콜, 대상 IP 주소와 포트가 포함되어 있는지 확인합니다.

```
# firewall-cmd --info-policy=<example_policy>
example_policy (active)
  priority: -1
  target: CONTINUE
  ingress-zones: HOST
  egress-zones: ANY
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
 rule family="ipv4" destination address="192.0.2.1" forward-port port="443"
protocol="tcp" to-port="443" to-addr="192.51.100.20"
```

추가 리소스

- firewall-cmd(1), firewalld.policies(5), firewalld.richECDHE(5), sysctl(8), sysctl.conf(5) 매뉴얼 페이지
- /etc/sysctl.d/의 설정 파일을 사용하여 커널 매개변수 조정

40.11. FIREWALLD를 사용하여 NAT 구성

firewalld 를 사용하면 다음 NAT(네트워크 주소 변환) 유형을 구성할 수 있습니다.

- 마스커레이딩
- 대상 NAT(DNAT)
- 리디렉션

40.11.1. 네트워크 주소 변환 유형

다음은 다양한 NAT(네트워크 주소 변환) 유형입니다.

마스커레이딩

이러한 NAT 유형 중 하나를 사용하여 패킷의 소스 IP 주소를 변경합니다. 예를 들어, 인터넷 서비스 공급자(ISP)는 10.0.0.0/8 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하고 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 이러한 범위의 패킷의 소스 IP 주소를 공용 IP 주소에 매핑합니다.

마스커레이딩은 나가는 인터페이스의 IP 주소를 자동으로 사용합니다. 따라서 나가는 인터페이스에서 동적 IP 주소를 사용하는 경우 마스커레이딩을 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 수신 트래픽을 이 서버로 리디렉션할 수 있습니다.

리디렉션

이 유형은 패킷을 로컬 시스템의 다른 포트로 리디렉션하는 특수한 DNAT의 경우입니다. 예를 들어 서비스가 표준 포트와 다른 포트에서 실행되는 경우 표준 포트에서 들어오는 트래픽을 이 특정 포트로 리디렉션할 수 있습니다.

40.11.2. IP 주소 마스커레이딩 구성

시스템에서 IP 마스커레이딩을 활성화할 수 있습니다. IP 마스커레이딩은 인터넷에 액세스할 때 게이트웨이 뒤에 있는 개별 머신을 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. 영역을 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 설정하려면 명령에 --permanent 옵션을 전달합니다.

40.11.3. DNAT를 사용하여 들어오는 HTTP 트래픽 전달

대상 네트워크 주소 변환(DNAT)을 사용하여 들어오는 트래픽을 하나의 대상 주소 및 포트에서 다른 대상 주소로 보낼 수 있습니다. 일반적으로 외부 네트워크 인터페이스에서 특정 내부 서버 또는 서비스로 들어오는 요청을 리디렉션하는 데 유용합니다.

사전 요구 사항

- firewalld 서비스가 실행 중입니다.

절차

1. 다음 콘텐츠를 사용하여 /etc/sysctl.d/90-enable-IP-forwarding.conf 파일을 생성합니다.

```
net.ipv4.ip_forward=1
```

이 설정은 커널에서 IP 전달을 활성화합니다. 내부 RHEL 서버가 라우터 역할을 하고 네트워크에서 네트워크로 패킷을 전달합니다.

2. /etc/sysctl.d/90-enable-IP-forwarding.conf 파일에서 설정을 로드합니다.

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. 들어오는 HTTP 트래픽을 전달합니다.

```
# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toaddr=198.51.100.10:toport=8080 --permanent
```

이전 명령은 다음 설정으로 DNAT 규칙을 정의합니다.

- --zone=public - DNAT 규칙을 구성하는 방화벽 영역입니다. 필요한 모든 영역에 맞게 조정할 수 있습니다.
- --add-forward-port - 포트 전달 규칙을 추가 중임을 나타내는 옵션입니다.
- port=80 - 외부 대상 포트입니다.
- proto=tcp - TCP 트래픽을 전달함을 나타내는 프로토콜입니다.
- toaddr=198.51.100.10 - 대상 IP 주소입니다.
- toport=8080 - 내부 서버의 대상 포트입니다.
- --permanent - 재부팅 시 DNAT 규칙을 유지할 수 있는 옵션입니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

```
# firewall-cmd --reload
```

검증

- 사용한 방화벽 영역에 대한 **DNAT** 규칙을 확인합니다.

```
# firewall-cmd --list-forward-ports --zone=public
port=80:proto=tcp:toport=8080:toaddr=198.51.100.10
```

또는 해당 **XML** 구성 파일을 확인합니다.

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on
networks to not harm your computer. Only selected incoming connections are
accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="80" protocol="tcp" to-port="8080" to-addr="198.51.100.10"/>
  <forward/>
</zone>
```

추가 리소스

- 런타임에 커널 매개변수 구성
- **firewall-cmd(1)** 매뉴얼 페이지

40.11.4. 비표준 포트에서 트래픽을 리디렉션하여 표준 포트에서 웹 서비스에 액세스하도록 설정

리디렉션 메커니즘을 사용하여 사용자가 **URL**에 포트를 지정할 필요 없이 내부적으로 비표준 포트에서 실행되는 웹 서비스를 만들 수 있습니다. 결과적으로 **URL**은 더 간단하며 더 나은 검색 환경을 제공하는 반면 비표준 포트는 여전히 내부적으로 또는 특정 요구 사항에 사용됩니다.

사전 요구 사항

- **firewalld** 서비스가 실행 중입니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/sysctl.d/90-enable-IP-forwarding.conf** 파일을 생성합니다.

```
net.ipv4.ip_forward=1
```

이 설정은 커널에서 **IP** 전달을 활성화합니다.

2. **/etc/sysctl.d/90-enable-IP-forwarding.conf** 파일에서 설정을 로드합니다.

```
# sysctl -p /etc/sysctl.d/90-enable-IP-forwarding.conf
```

3. **NAT** 리디렉션 규칙을 생성합니다.

```
# firewall-cmd --zone=public --add-forward-
port=port=<standard_port>:proto=tcp:toport=<non_standard_port> --permanent
```

이전 명령은 다음 설정으로 **NAT** 리디렉션 규칙을 정의합니다.

- **--zone=public -** 규칙을 구성하는 방화벽 영역입니다. 필요한 모든 영역에 맞게 조정할 수 있습니다.
- **--add-forward-port=port= <*non_standard_port* > -** 들어오는 트래픽을 처음 수신하는 소스 포트를 사용하여 포트 전달(리렉션) 규칙을 추가 중임을 나타내는 옵션입니다.
- **proto=tcp - TCP** 트래픽을 리디렉션함을 나타내는 프로토콜입니다.
- **toport=<*standard_port* > -** 소스 포트에서 수신한 후 들어오는 트래픽을 리디렉션해야 하는 대상 포트입니다.
- **--permanent -** 다시 부팅 시 규칙을 유지할 수 있는 옵션입니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

```
# firewall-cmd --reload
```

검증

- 다음을 사용한 방화벽 영역의 리디렉션 규칙을 확인합니다.

```
# firewall-cmd --list-forward-ports
port=8080:proto=tcp:toport=80:toaddr=
```

또는 해당 **XML** 구성 파일을 확인합니다.

```
# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <forward-port port="8080" protocol="tcp" to-port="80"/>
  <forward/>
</zone>
```

추가 리소스

- 런타임에 커널 매개변수 구성
- **firewall-cmd(1)** 매뉴얼 페이지

40.12. ICMP 요청 관리

ICMP(Internet Control Message Protocol)는 다양한 네트워크 장치에서 테스트, 문제 해결 및 진단을 위해 사용하는 지원 프로토콜입니다. **ICMP** 는 시스템 간에 데이터를 교환하는 데 사용되지 않으므로 **TCP** 및 **UDP**와 같은 전송 프로토콜과 다릅니다.

ICMP 메시지, 특히 **echo-request** 및 **echo-reply** 를 사용하여 네트워크에 대한 정보를 공개하고 다양한 종류의 사기 활동에 대해 이러한 정보를 오용할 수 있습니다. 따라서 **firewalld** 는 네트워크 정보를 보호하기 위해 **ICMP** 요청을 제어할 수 있습니다.

40.12.1. ICMP 필터링 구성

ICMP 필터링을 사용하여 방화벽에서 시스템에 도달할 수 있도록 허용하거나 거부할 ICMP 유형 및 코드를 정의할 수 있습니다. ICMP 유형 및 코드는 ICMP 메시지의 특정 카테고리 및 하위 범주입니다.

예를 들어 ICMP 필터링은 다음 영역에서 도움이 됩니다.

- 보안 개선 - 잠재적으로 유해한 ICMP 유형 및 코드를 차단하여 공격 면적을 줄입니다.
- 네트워크 성능 - 네트워크 성능을 최적화하고 과도한 ICMP 트래픽으로 인한 잠재적인 네트워크 혼잡을 방지하는 데 필요한 ICMP 유형만 허용합니다.
- 문제 해결 제어 - 잠재적인 보안 위험을 나타내는 네트워크 문제 해결 및 차단 ICMP 유형에 대한 필수 ICMP 기능을 유지 관리합니다.

사전 요구 사항

- firewalld 서비스가 실행 중입니다.

절차

1. 사용 가능한 ICMP 유형 및 코드를 나열합니다.

```
# firewall-cmd --get-icmptypes
address-unreachable bad-header beyond-scope communication-prohibited
destination-unreachable echo-reply echo-request failed-policy fragmentation-needed
host-precedence-violation host-prohibited host-redirect host-unknown host-
unreachable
...
```

사전 정의된 목록에서 허용 또는 차단할 ICMP 유형 및 코드를 선택합니다.

2. 특정 ICMP 유형을 다음과 같이 필터링합니다.

- ICMP 유형 허용:

```
# firewall-cmd --zone=<target-zone> --remove-icmp-block=echo-request --
permanent
```

명령은 **echo requests ICMP** 유형의 기존 차단 규칙을 제거합니다.

- **ICMP** 유형 차단:

```
# firewall-cmd --zone=<target-zone> --add-icmp-block=redirect --permanent
```

이 명령을 사용하면 리디렉션 메시지 **ICMP** 유형이 방화벽에 의해 차단됩니다.

3. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

```
# firewall-cmd --reload
```

검증

- 필터링 규칙이 적용되었는지 확인합니다.

```
# firewall-cmd --list-icmp-blocks
redirect
```

명령 출력에는 허용 또는 차단한 **ICMP** 유형 및 코드가 표시됩니다.

추가 리소스

- **firewall-cmd(1)** 매뉴얼 페이지

40.13. FIREWALLD를 사용하여 IP 세트 설정 및 제어

IP 세트는 보다 유연하고 효율적인 방화벽 규칙 관리를 위해 **IP** 주소 및 네트워크를 세트로 그룹화하는 **RHEL** 기능입니다.

예를 들어 필요한 경우 **IP** 세트는 시나리오에서 중요합니다.

- 대규모 **IP** 주소 목록 처리

- 이러한 대규모 IP 주소 목록에 대한 동적 업데이트 구현
- 사용자 지정 IP 기반 정책을 생성하여 네트워크 보안 및 제어 강화

주의

Red Hat은 firewall-cmd 명령을 사용하여 IP 세트를 생성하고 관리하는 것이 좋습니다.

40.13.1. IP 세트를 사용하여 허용 목록에 대한 동적 업데이트 구성

예측할 수 없는 조건에서도 IP 세트의 특정 IP 주소 또는 범위를 유연하게 허용하도록 거의 실시간 업데이트를 수행할 수 있습니다. 이러한 업데이트는 보안 위협 탐지 또는 네트워크 동작 변경과 같은 다양한 이벤트에 의해 트리거될 수 있습니다. 일반적으로 이러한 솔루션은 자동화를 활용하여 수동 작업을 줄이고 상황에 신속하게 대응하여 보안을 개선합니다.

사전 요구 사항

- firewalld 서비스가 실행 중입니다.

절차

1. 의미 있는 이름으로 IP 세트를 생성합니다.

```
# firewall-cmd --permanent --new-ipset=allowlist --type=hash:ip
```

allowlist 라는 새 IP 세트에는 방화벽에서 허용할 IP 주소가 포함되어 있습니다.

2. IP 세트에 동적 업데이트를 추가합니다.

```
# firewall-cmd --permanent --ipset=allowlist --add-entry=198.51.100.10
```

이 구성은 방화벽에서 네트워크 트래픽을 전달할 수 있는 새로 추가된 IP 주소로 허용 목록 IP

세트를 업데이트합니다.

3. 이전에 생성한 IP 세트를 참조하는 방화벽 규칙을 생성합니다.

```
# firewall-cmd --permanent --zone=public --add-source=ipset:allowlist
```

이 규칙이 없으면 IP 세트가 네트워크 트래픽에 영향을 미치지 않습니다. 기본 방화벽 정책이 우선합니다.

4. 방화벽 구성을 다시 로드하여 변경 사항을 적용합니다.

```
# firewall-cmd --reload
```

검증

1. 모든 IP 세트를 나열합니다.

```
# firewall-cmd --get-ipsets
allowlist
```

2. 활성 규칙을 나열합니다.

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s1
  sources: ipset:allowlist
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  ...
```

명령줄 출력의 소스 섹션에서는 특정 방화벽 영역에 대한 액세스 허용 또는 거부되는 트래픽(호스트, 인터페이스, IP 세트, 서브넷 등)에 대한 인사이트를 제공합니다. 이 경우 허용 목록 IP 세트에 포함된 IP 주소는 공용 영역의 방화벽을 통해 트래픽을 전달할 수 있습니다.

3. IP 세트의 내용을 살펴봅니다.

```
# cat /etc/firewalld/ipsets/allowlist.xml
<?xml version="1.0" encoding="utf-8"?>
<ipset type="hash:ip">
  <entry>198.51.100.10</entry>
</ipset>
```

다음 단계

- 스크립트 또는 보안 유틸리티를 사용하여 위협 정보 피드를 가져오고 이에 따라 허용 목록을 자동화된 방식으로 업데이트합니다.

추가 리소스

- firewall-cmd(1) 매뉴얼 페이지

40.14. 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. **priority** 매개변수를 사용하는 경우 규칙은 우선 순위 값으로 오름차순으로 정렬됩니다. 더 많은 규칙에 동일한 우선 순위가 있는 경우 규칙 작업에 따라 순서가 결정되며, 작업이 동일한 경우 순서가 정의되지 않을 수 있습니다.

40.14.1. 우선순위 매개변수가 규칙을 다른 체인으로 구성하는 방법

리치 규칙의 **priority** 매개변수를 **-32768** 과 **32767** 사이의 임의의 숫자로 설정할 수 있으며 더 낮은 숫자 값은 우선 순위가 높습니다.

firewalld 서비스는 우선순위 값에 따라 규칙을 다른 체인으로 구성합니다.

- **0**보다 낮은 우선 순위: 규칙이 **_pre** 접미사가 있는 체인으로 리디렉션됩니다.
- **0**보다 높은 우선 순위: 규칙이 **_post** 접미사가 있는 체인으로 리디렉션됩니다.
- **priority equals 0: action**에 따라 규칙이 **_log,_deny,** 또는 **_allow the action**이 있는 체인으로 리디렉션됩니다.

이러한 하위 체인 내에서 **firewalld** 는 우선 순위 값에 따라 규칙을 정렬합니다.

40.14.2. 리치 규칙의 우선 순위 설정

다음은 다른 규칙에서 허용되거나 거부되지 않는 모든 트래픽을 기록하기 위해 **priority** 매개변수를 사용하는 리치 규칙을 생성하는 예입니다. 이 규칙을 사용하여 예기치 않은 트래픽에 플래그를 지정할 수 있습니다.

절차

- 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"'
```

이 명령은 또한 로그 항목 수를 분당 **5** 개로 제한합니다.

검증

- 이전 단계에서 생성된 명령을 사용하여 **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
  }
}
```

40.15. 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.

40.15.1. CLI를 사용하여 잠금 구성

명령줄을 사용하여 잠금 기능을 활성화하거나 비활성화할 수 있습니다.

절차

1. 잠금이 활성화되었는지 여부를 쿼리하려면 다음을 수행합니다.

```
# firewall-cmd --query-lockdown
```

2. 다음 중 하나를 통해 잠금 구성을 관리합니다.

- 잠금 활성화:

```
# firewall-cmd --lockdown-on
```

- 잠금 비활성화:

```
# firewall-cmd --lockdown-off
```

40.15.2. 잠금 허용 목록 구성 파일 개요

기본 허용 목록 구성 파일에는 **NetworkManager** 컨텍스트와 **libvirt** 의 기본 컨텍스트가 포함되어 있습니다. 사용자 **ID 0**도 목록에 있습니다.

허용 목록 구성 파일은 **/etc/firewalld/** 디렉터리에 저장됩니다.

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
  <command name="/usr/bin/python3 -s /usr/bin/firewall-config"/>
  <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 **ID**가 **815** 인 사용자의 경우 **firewall-cmd** 유틸리티에 대한 모든 명령을 활성화하는 **allowlist** 설정 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 이름 을 모두 보여주지만 하나의 옵션만 필요합니다. **Python**은 인터프리터이며 명령줄 앞에 추가됩니다.

Red Hat Enterprise Linux에서 모든 유틸리티는 **/usr/bin/** 디렉토리에 배치되고 **/bin/** 디렉토리는 **/usr/bin/** 디렉토리에 대칭입니다. 즉, **root** 로 입력했을 때 **firewall-cmd** 의 경로는 **/bin/firewall-cmd** 로 확인될 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 일반적으로 루트가 아닌 사용자에게만 사용되는 **/usr/bin/firewall-cmd** 경로 외에도 **allowlist**에 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.

40.16. 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.

40.16.1. 기본 타겟이 ACCEPT로 설정된 인트라 영역 전달과 영역의 차이점

영역 내 전달이 활성화된 경우 단일 **firewalld** 영역 내의 트래픽이 하나의 인터페이스 또는 소스에서 다른 인터페이스 또는 소스로 전달될 수 있습니다. 영역은 인터페이스 및 소스의 신뢰 수준을 지정합니다. 신뢰 수준이 동일한 경우 트래픽은 동일한 영역 내에 유지됩니다.

참고

firewalld 의 기본 영역에서 영역 내 전달을 활성화하면 현재 기본 영역에 추가된 인터페이스와 소스에만 적용됩니다.

firewalld 는 다른 영역을 사용하여 들어오고 나가는 트래픽을 관리합니다. 각 영역에는 고유한 규칙과 동작 세트가 있습니다. 예를 들어 신뢰할 수 있는 영역은 기본적으로 전달된 모든 트래픽을 허용합니다.

다른 영역에는 기본 동작이 다를 수 있습니다. 표준 영역에서 영역의 대상이 기본값으로 설정된 경우

전달된 트래픽은 일반적으로 기본적으로 삭제됩니다.

영역 내의 다양한 인터페이스 또는 소스 간에 트래픽이 전달되는 방법을 제어하려면 해당 영역의 대상을 적절하게 이해하고 구성해야 합니다.

40.16.2. 영역 내 전달을 사용하여 이더넷 및 Wi-Fi 네트워크 간의 트래픽 전달

intra-zone 전달을 사용하여 동일한 firewalld 영역 내의 인터페이스와 소스 간에 트래픽을 전달할 수 있습니다. 이 기능은 다음과 같은 이점을 제공합니다.

- *유선 및 무선 장치 간의 원활한 연결(W lp 0s 20에 연결된 이더넷 네트워크와 Wi-Fi 네트워크 간에 트래픽을 전달할 수 있음)*
- *유연한 작업 환경 지원*
- *프린터, 데이터베이스, 네트워크 연결 스토리지 등 여러 장치 또는 네트워크에서 액세스하고 사용하는 공유 리소스*
- *효율적인 내부 네트워킹(예: 원활한 통신, 대기 시간 감소, 리소스 접근성 등)*

개별 firewalld 영역에 대해 이 기능을 활성화할 수 있습니다.

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. *영역 내 전달을 활성화할 인터페이스가 내부 영역에만 할당되도록 합니다.*

```
# firewall-cmd --get-active-zones
```

3. *인터페이스가 현재 내부 이외의 영역에 할당되면 이를 다시 할당합니다.*

```
# firewall-cmd --zone=internal --change-interface=interface_name --permanent
```

4. *enp1s0 및 wlp0s20 인터페이스를 내부 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. *어떤 것을 입력하고 Enter 키를 누릅니다. 텍스트가 다시 전송되었는지 확인합니다.*

추가 리소스

- *firewalld.zones(5) 도움말 페이지*

40.17. RHEL 시스템 역할을 사용하여 FIREWALLD 구성

방화벽 RHEL 시스템 역할을 사용하여 한 번에 여러 클라이언트에서 firewalld 서비스 설정을 구성할 수 있습니다. 이 해결책:

- *효율적인 입력 설정을 제공하는 인터페이스를 제공합니다.*
- *원하는 모든 firewalld 매개변수를 한 곳에 보관합니다.*

제어 노드에서 방화벽 역할을 실행한 후 RHEL 시스템 역할은 firewalld 매개변수를 관리 노드에 즉시 적용하여 재부팅 시에도 지속됩니다.

40.17.1. 방화벽 RHEL 시스템 역할 소개

RHEL 시스템 역할은 Ansible 자동화 유틸리티의 콘텐츠 집합입니다. Ansible 자동화 유틸리티와 함께 이 콘텐츠는 여러 시스템을 원격으로 관리할 수 있는 일관된 구성 인터페이스를 제공합니다.

RHEL 시스템 역할의 rhel-system-roles.firewall 역할은 firewalld 서비스의 자동화된 구성을 위해 도입되었습니다. rhel-system-roles 패키지에는 이 RHEL 시스템 역할과 참조 문서가 포함되어 있습니다.

하나 이상의 시스템에 firewalld 매개변수를 자동화된 방식으로 적용하려면 플레이북에서 방화벽 RHEL 시스템 역할 변수를 사용합니다. 플레이북은 텍스트 기반 YAML 형식으로 작성된 하나 이상의 플레이 목록입니다.

인벤토리 파일을 사용하여 Ansible에서 구성할 시스템 세트를 정의할 수 있습니다.

firewall 역할을 사용하면 다음과 같이 다양한 firewalld 매개변수를 구성할 수 있습니다.

- *영역.*
- *패킷을 허용해야 하는 서비스입니다.*
- *포트에 대한 트래픽 액세스 권한 부여, 거부 또는 삭제.*

- 영역의 포트 또는 포트 범위 전달.

추가 리소스

- /usr/share/ansible/roles/rhel-system-roles.firewall/README.md 파일
- /usr/share/doc/rhel-system-roles/firewall/ 디렉터리
- 플레이북 작업
- 인벤토리 빌드 방법

40.17.2. 방화벽 RHEL 시스템 역할을 사용하여 firewalld 설정 재설정

방화벽 RHEL 시스템 역할을 사용하면 firewalld 설정을 기본 상태로 재설정할 수 있습니다. 이전:replaced 매개변수를 변수 목록에 추가하면 RHEL 시스템 역할은 기존 사용자 정의 설정을 모두 제거하고 firewalld 를 기본값으로 재설정합니다. 이전:replaced 매개변수를 다른 설정과 결합하면 방화벽 역할은 새 설정을 적용하기 전에 기존 설정을 모두 제거합니다.

Ansible 제어 노드에서 다음 절차를 수행합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
    - name: Reset firewalld
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - previous: replaced
```

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

- 이 명령을 관리 노드에서 **root** 로 실행하여 모든 영역을 확인합니다.

```
# firewall-cmd --list-all-zones
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일
- **/usr/share/doc/rhel-system-roles/firewall/** 디렉터리

40.17.3. 방화벽 RHEL 시스템 역할을 사용하여 하나의 로컬 포트에서 다른 로컬 포트로 firewalld 에서 들어오는 트래픽 전달

firewall 역할을 사용하면 여러 관리 호스트에 지속적인 영향을 적용하여 **firewalld** 매개변수를 원격으로 구성할 수 있습니다.

Ansible 제어 노드에서 다음 절차를 수행합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Forward incoming traffic on port 8080 to 443
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

- 관리 호스트에서 **firewalld** 설정을 표시합니다.

```
# firewall-cmd --list-forward-ports
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일
- **/usr/share/doc/rhel-system-roles/firewall/** 디렉터리

40.17.4. 방화벽 RHEL 시스템 역할을 사용하여 firewalld 에서 포트 관리

방화벽 **RHEL** 시스템 역할을 사용하여 수신 트래픽을 위해 로컬 방화벽의 포트를 열거나 종료하고 재부팅 시 새 구성을 유지할 수 있습니다. 예를 들어 **HTTPS** 서비스에 대한 들어오는 트래픽을 허용하도록 기본 영역을 구성할 수 있습니다.

Ansible 제어 노드에서 다음 절차를 수행합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Allow incoming HTTPS traffic to the local host
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
```

```
    vars:
      firewall:
        - port: 443/tcp
          service: http
          state: enabled
          runtime: true
          permanent: true
```

permanent: true 옵션을 사용하면 재부팅해도 새 설정이 유지됩니다.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

- 관리형 노드에서 **HTTPS** 서비스와 연결된 **443/tcp** 포트가 열려 있는지 확인합니다.

```
# firewall-cmd --list-ports
443/tcp
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 파일
- **/usr/share/doc/rhel-system-roles/firewall/** 디렉터리

40.17.5. 방화벽 RHEL 시스템 역할을 사용하여 firewalld DMZ 영역 구성

시스템 관리자는 방화벽 **RHEL** 시스템 역할을 사용하여 **enp1s0** 인터페이스에서 **dmz** 영역을 구성하여 **HTTPS** 트래픽을 영역에 허용할 수 있습니다. 이렇게 하면 외부 사용자가 웹 서버에 액세스할 수 있습니다.

Ansible 제어 노드에서 다음 절차를 수행합니다.

사전 요구 사항

- 제어 노드와 관리형 노드가 준비되어 있습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리형 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

절차

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
    - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in
DMZ
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - zone: dmz
            interface: enp1s0
            service: https
            state: enabled
            runtime: true
            permanent: true
```

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. 플레이북을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

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:
```

추가 리소스

- */usr/share/ansible/roles/rhel-system-roles.firewall/README.md* 파일
- */usr/share/doc/rhel-system-roles/firewall/* 디렉터리

41장. NFTABLES 시작하기

nftables 프레임워크는 패킷을 분류하고 iptables,ip6tables,arptables,ebtables, ipset 유틸리티의 후속 조치입니다. 이전의 패킷 필터링 툴에 비해 편의성, 기능 및 성능이 크게 향상되었으며 주요 개선 사항은 다음과 같습니다.

- 선형 처리 대신 기본 제공 조회 테이블
- IPv4 및 IPv6 프로토콜 모두를 위한 단일 프레임워크
- 모든 규칙은 전체 규칙 세트를 가져오기, 업데이트 및 저장하는 대신 원자적으로 적용됩니다.
- 규칙 세트(nftrace) 및 모니터링 추적 이벤트(nft 툴 내)에서 디버깅 및 추적 지원
- 프로토콜별 확장 없이 보다 일관되고 컴팩트한 구문
- 타사 애플리케이션을 위한 Netlink API

nftables 프레임워크는 테이블을 사용하여 체인을 저장합니다. 체인에는 작업을 수행하기 위한 개별 규칙이 포함되어 있습니다. nft 유틸리티는 이전 패킷 필터링 프레임워크의 모든 도구를 대체합니다. libnftnl 라이브러리를 사용하여 libmnl 라이브러리를 통해 nftables Netlink API와 하위 수준의 상호 작용을 수행할 수 있습니다.

규칙 세트 변경의 효과를 표시하려면 nft list ruleset 명령을 사용합니다. 이러한 유틸리티는 테이블, 체인, 규칙, 세트 및 기타 오브젝트를 nftables 규칙 세트에 추가하므로 nft flush ruleset 명령과 같은 nftables 규칙 세트 작업이 iptables 명령을 사용하여 설치된 규칙 세트에 영향을 미칠 수 있습니다.

41.1. IPTABLES에서 NFTABLES로 마이그레이션

방화벽 구성에서 여전히 iptables 규칙을 사용하는 경우 iptables 규칙을 nftables 로 마이그레이션할 수 있습니다.

41.1.1. firewalld, nftables 또는 iptables를 사용하는 경우

다음은 다음 유틸리티 중 하나를 사용해야 하는 시나리오에 대한 간략한 개요입니다.

- **firewalld:** 간단한 방화벽 활용 사례에 **firewalld** 유틸리티를 사용합니다. 유틸리티는 사용하기 쉽고 이러한 시나리오의 일반적인 사용 사례를 다룹니다.

- **nftables:** **nftables** 유틸리티를 사용하여 전체 네트워크에 대해 과 같이 복잡하고 성능이 중요한 방화벽을 설정합니다.

- **iptables:** **Red Hat Enterprise Linux**의 **iptables** 유틸리티는 기존 백엔드 대신 **nf_tables** 커널 **API** 를 사용합니다. **nf_tables API**는 **iptables** 명령을 사용하는 스크립트가 **Red Hat Enterprise Linux**에서 계속 작동하도록 이전 버전과의 호환성을 제공합니다. 새 방화벽 스크립트의 경우 **Red Hat**은 **nftables** 를 사용하는 것이 좋습니다.

중요

다른 방화벽 관련 서비스(**firewalld**,**nftables** 또는 **iptables**)가 서로 영향을 미치지 않도록 하려면 **RHEL** 호스트에서 해당 서비스 중 하나만 실행하고 다른 서비스를 비활성화합니다.

41.1.2. 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 설치되어 있습니다.

- 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. 생성된 **nftables** 규칙을 검토하고 필요한 경우 수동으로 업데이트합니다.

4. **nftables** 서비스가 생성된 파일을 로드하도록 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

41.1.3. 단일 iptables 및 ip6tables 규칙을 nftables로 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 있습니다.**

절차

- **iptables 또는 ip6tables 대신 iptables-translate 또는 ip6tables-translate 유틸리티를 사용하여 해당 nftables 규칙을 표시합니다.**

```
# 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 붙은 번역되지 않은 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**

41.1.4. 일반적인 iptables 및 nftables 명령 비교

다음은 공통 **iptables** 및 **nftables** 명령을 비교한 것입니다.

- 모든 규칙 나열:

iptables	nftables
iptables-save	**nft list ruleset**

- 특정 테이블 및 체인 나열:

iptables	nftables
iptables -L	**nft list table ip filter**
iptables -L INPUT	**nft list chain ip filter INPUT**
iptables -t nat -L PREROUTING	**nft list chain ip nat PREROUTING**

nft 명령은 테이블 및 체인을 미리 생성하지 않습니다. 이는 사용자가 수동으로 생성한 경우에만 존재합니다.

firewalld를 통해 생성된 규칙 나열:

```
# nft list table inet firewalld
# nft list table ip firewalld
# nft list table ip6 firewalld
```

41.2. NFTABLES 스크립트 작성 및 실행

nftables 프레임워크를 사용할 때의 주요 이점은 스크립트 실행이 atomic이라는 것입니다. 즉, 시스템이 전체 스크립트를 적용하거나 오류가 발생할 경우 실행을 방지합니다. 이렇게 하면 방화벽이 항상 일관된 상태에 있습니다.

또한 nftables 스크립트 환경에서 다음을 수행할 수 있습니다.

- *댓글 추가*
- *변수 정의*
- *기타 규칙 세트 파일 포함*

*nftables 패키지를 설치할 때 Red Hat Enterprise Linux는 /etc/nftables/ 디렉토리에 *.nft 스크립트를 자동으로 생성합니다. 이러한 스크립트에는 다양한 용도로 테이블과 빈 체인을 생성하는 명령이 포함되어 있습니다.*

41.2.1. 지원되는 nftables 스크립트 형식

다음 형식으로 **nftables** 스크립팅 환경에서 스크립트를 작성할 수 있습니다.

- **nft list ruleset** 명령과 동일한 형식으로 규칙 세트를 표시합니다.

```
#!/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** 명령의 구문은 다음과 같습니다.

```
#!/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
```

41.2.2. nftables 스크립트 실행 중

nftables 스크립트를 **nft** 유틸리티에 전달하거나 직접 스크립트를 실행하여 실행할 수 있습니다.

절차

- **nftables** 스크립트를 **nft** 유틸리티에 전달하여 실행하려면 다음을 입력합니다.

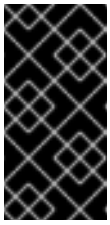
```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

- **nftables** 스크립트를 직접 실행하려면 다음을 수행합니다.

a. 이 작업을 수행하는 단일 시간 동안 다음을 수행합니다.

i. 스크립트가 다음 **shebang** 시퀀스로 시작되는지 확인합니다.

```
#!/usr/sbin/nft -f
```



중요

-f 매개변수를 생략하면 **nft** 유틸리티에서 스크립트를 읽지 않고 표시됩니다. 오류: 구문 오류, 예기치 않은 **newline**, 예상 문자열.

ii. 선택 사항: 스크립트 소유자를 **root** 로 설정합니다.

```
# chown root /etc/nftables/<example_firewall_script>.nft
```

iii. 소유자에 대해 스크립트를 실행 파일로 만듭니다.

```
# chmod u+x /etc/nftables/<example_firewall_script>.nft
```

b. 스크립트를 실행합니다.

```
# /etc/nftables/<example_firewall_script>.nft
```

출력이 표시되지 않으면 시스템에서 스크립트를 성공적으로 실행했습니다.

중요

nft 가 스크립트를 성공적으로 실행하고, 잘못 배치된 규칙, 누락된 매개 변수 또는 스크립트의 기타 문제로 인해 방화벽이 예상대로 작동하지 않을 수 있습니다.

추가 리소스

- chown(1) 매뉴얼 페이지
- ECDHE(1) 매뉴얼 페이지
- 시스템 부팅 시 nftables 규칙 자동 로딩

41.2.3. nftables 스크립트에서 주석 사용

nftables 스크립팅 환경은 모든 것을 # 문자 오른쪽에 주석으로 해석합니다.

주석은 행 시작 시 시작되거나 명령 옆에 있을 수 있습니다.

```
...
# Flush the rule set
flush ruleset

add table inet example_table  # Create a table
...
```

41.2.4. nftables 스크립트의 변수 사용

nftables 스크립트에 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 enp1s0 값을 사용하여 INET_DEV 라는 변수를 정의합니다.

```
define INET_DEV = enp1s0
```

$ 기호 다음에 변수 이름을 입력하여 스크립트에서 변수를 사용할 수 있습니다.

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

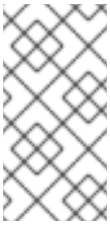
익명 세트를 포함하는 변수

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

중괄호에는 변수가 집합을 나타내는 것을 나타내기 때문에 규칙에서 사용할 때 특수한 의미가 있습니다.

추가 리소스

- nftables 명령에서 세트 사용
- nftables 명령에서 verdict map 사용

41.2.5. nftables 스크립트에 파일 포함

nftables 스크립팅 환경에서는 include 문을 사용하여 다른 스크립트를 포함할 수 있습니다.

절대 경로 또는 상대 경로가 없는 파일 이름만 지정하면 nftables 에 기본 검색 경로의 파일이 포함되어 있으며 이는 Red Hat Enterprise Linux에서 /etc 로 설정됩니다.

예 41.1. 기본 검색 디렉토리의 파일 포함

기본 검색 디렉터리에서 파일을 포함하려면 다음을 수행합니다.

```
include "example.nft"
```

예 41.2. 디렉토리의 *.nft 파일 모두 포함

/etc/nftables/rulesets/ 디렉토리에 저장된 *.nft 로 끝나는 모든 파일을 포함하려면 다음을 수행하십시오.

```
include "/etc/nftables/rulesets/*.nft"
```

include 구문은 점으로 시작하는 파일과 일치하지 않습니다.

추가 리소스

- nft(8) 매뉴얼 페이지에 파일 포함 섹션

41.2.6. 시스템이 부팅될 때 nftables 규칙을 자동으로 로드

nftables systemd 서비스는 /etc/sysconfig/nftables.conf 파일에 포함된 방화벽 스크립트를 로드합니다.

사전 요구 사항

- nftables 스크립트는 /etc/nftables/ 디렉터리에 저장됩니다.

절차

1. /etc/sysconfig/nftables.conf 파일을 편집합니다.
   - nftables 패키지 설치로 /etc/nftables/ 에 생성된 *.nft 스크립트를 수정한 경우 해당 스크립트에 대한 include 문의 주석 처리를 해제합니다.
   - 새 스크립트를 작성한 경우 include 문을 추가하여 이러한 스크립트를 포함합니다. 예를 들어 nftables 서비스가 시작될 때 /etc/nftables/예제.nft 스크립트를 로드하려면 다음을 추가합니다.

     ```
     include "/etc/nftables/_example_.nft"
     ```

2. 선택 사항: 시스템을 재부팅하지 않고 nftables 서비스를 시작하여 방화벽 규칙을 로드합니다.

```
# systemctl start nftables
```

3. *nftables* 서비스를 활성화합니다.

```
# systemctl enable nftables
```

추가 리소스

- 지원되는 *nftables* 스크립트 형식

41.3. NFTABLES 테이블, 체인 및 규칙 생성 및 관리

nftables 규칙 세트를 표시하고 관리할 수 있습니다.

41.3.1. nftables 테이블 기본

nftables 의 테이블은 체인, 규칙, 세트 및 기타 오브젝트 컬렉션을 포함하는 네임스페이스입니다.

각 테이블에는 주소 제품군이 할당되어 있어야 합니다. 주소 *family*는 이 테이블이 처리하는 패킷 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.
- 브릿지: 브리지 장치를 통과하는 패킷과 일치합니다.
- *netdev:* 수신에서 패킷 일치.

테이블을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- 기본 구문 스크립트의 경우 다음을 사용합니다.

```
table <table_address_family> <table_name> {
}
```

- 쉘 스크립트에서는 다음을 사용합니다.

```
nft add table <table_address_family> <table_name>
```

41.3.2. nftables 체인의 기본 사항

테이블은 체인으로 구성되며, 이 체인은 규칙용 컨테이너입니다. 다음 두 가지 규칙 유형이 있습니다.

- 기본 체인: 기본 체인을 네트워킹 스택의 패킷 진입점으로 사용할 수 있습니다.
- 일반 체인: 규칙을 더 잘 구성하기 위해 일반 체인을 이동 대상으로 사용할 수 있습니다.

테이블에 기본 체인을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- 기본 구문 스크립트의 경우 다음을 사용합니다.

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority>
    policy <policy> ;
  }
}
```

- 쉘 스크립트에서는 다음을 사용합니다.

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook
<hook> priority <priority> \; policy <policy> \; }
```

쉘이 명령 끝으로 해석되지 않도록 하려면 \ 이스케이프 문자 앞에 \ 이스케이프 문자를 배치합니다.

두 예에서는 기본 체인을 생성합니다. 일반 체인 을 생성하려면 중괄호에서 매개 변수를 설정하지 마십시오.

체인 유형

다음은 체인 유형 및 제품군과 후크를 사용할 수 있는 개요입니다.

유형	주소 제품군	후크	설명
filter	모두	모두	표준 체인 유형
nat	**ip**, **ip6**, **inet**	**PREROUTING, 입력,출력,후드**	이 유형의 체인은 연결 추적 항목을 기반으로 기본 주소 변환을 수행합니다. 첫 번째 패킷만 이 체인 유형을 통과합니다.
Route	**ip**, **ip6**	**output**	이 체인 유형을 트래버스하는 수락된 패킷은 IP 헤더의 관련 부분이 변경된 경우 새로운 경로 조회를 유발합니다.

체인 우선순위

***priority** 매개변수는 패킷이 동일한 후크 값을 사용하는 체인을 트래버스하는 순서를 지정합니다. 이 매개변수를 정수 값으로 설정하거나 표준 우선순위 이름을 사용할 수 있습니다.*

다음 매트릭스는 표준 우선 순위 이름과 해당 숫자 값에 대한 개요와 함께 사용할 수 있는 제품군과 후크를 처리합니다.

텍스트 값	숫자 값	주소 제품군	후크
원시	**-300**	**ip**, **ip6**, **inet**	모두
mangle	**-150**	**ip**, **ip6**, **inet**	모두
dstnat	**-100**	**ip**, **ip6**, **inet**	**PREROUTING**
	-300	**Bridge**	**PREROUTING**
filter	**0**	IP , **ip 6**,**inet**,**arp**,**netdev**	모두
	-200	**Bridge**	모두

텍스트 값	숫자 값	주소 제품군	후크
보안	**50**	**ip**, **ip6**, **inet**	모두
srcnat	**100**	**ip**, **ip6**, **inet**	**POSTROUTING**
	300	**Bridge**	**POSTROUTING**
out	**100**	**Bridge**	**output**

체인 정책

*체인 정책은 이 체인의 규칙이 작업을 지정하지 않는 경우 **nftables** 가 패킷을 수락하거나 삭제해야 하는지 여부를 정의합니다. 체인에서 다음 정책 중 하나를 설정할 수 있습니다.*

- *수락 (기본값)*
- ***drop***

41.3.3. nftables 규칙의 기본 사항

*규칙은 이 규칙을 포함하는 체인을 전달하는 패킷에서 수행할 작업을 정의합니다. 규칙에 일치하는 표현식도 포함된 경우 **nftables** 는 모든 이전 표현식이 적용되는 경우에만 작업을 수행합니다.*

체인에 규칙을 추가하려면 사용할 형식은 방화벽 스크립트에 따라 다릅니다.

- *기본 구문 스크립트의 경우 다음을 사용합니다.*

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority> ; policy <policy> ;
      <rule>
  }
}
```

- *쉘 스크립트에서는 다음을 사용합니다.*

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

*이 쉘 명령은 체인 끝에 새 규칙을 추가합니다. 체인 시작 부분에 규칙을 추가하려면 **nft add** 대신 **nft insert** 명령을 사용하십시오.*

41.3.4. nft 명령을 사용하여 테이블, 체인 및 규칙 관리

*명령줄 또는 쉘 스크립트에서 **nftables** 방화벽을 관리하려면 **nft** 유틸리티를 사용합니다.*

중요

*이 절차의 명령은 일반적인 워크플로우를 나타내지 않으며 최적화되지 않습니다. 이 절차에서는 일반적으로 **nft** 명령을 사용하여 테이블, 체인 및 규칙을 관리하는 방법을 보여줍니다.*

절차

1. *테이블이 **IPv4** 및 **IPv6** 패킷을 모두 처리할 수 있도록 **inet** 주소 **Family**를 사용하여 **nftables_svc** 라는 테이블을 만듭니다.*

```
# nft add table inet nftables_svc
```

2. *들어오는 네트워크 트래픽을 처리하는 **INPUT** 라는 기본 체인을 **inet nftables_svc** 테이블에 추가합니다.*

```
# nft add chain inet nftables_svc INPUT { type filter hook input priority filter \; policy accept \; }
```

*쉘이 명령 끝으로 해석되는 것을 방지하기 위해 \ 문자를 사용하여 이름이 **0**으로 이스케이프합니다.*

3. ***INPUT** 체인에 규칙을 추가합니다. 예를 들어 포트 **22** 및 **443**에서 들어오는 **TCP** 트래픽을 허용하고 **INPUT** 체인의 마지막 규칙으로 **IMP(Internet Control Message Protocol)** 포트 연결할 수 없는 메시지가 있는 다른 들어오는 트래픽을 거부합니다.*

```
# nft add rule inet nftables_svc INPUT tcp dport 22 accept
# nft add rule inet nftables_svc INPUT tcp dport 443 accept
# nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

*다음과 같이 **nft add rule** 명령을 입력하면 **nft** 는 명령을 실행할 때와 동일한 순서로 규칙을 체인에 추가합니다.*

4. 프로세스를 포함한 현재 규칙 세트를 표시합니다.

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

5. handle 3이 있는 기존 규칙 앞에 규칙을 삽입합니다. 예를 들어 포트 636에서 TCP 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.

```
# nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6. handle 3이 있는 기존 규칙 뒤에 규칙을 추가합니다. 예를 들어 포트 80에서 TCP 트래픽을 허용하는 규칙을 삽입하려면 다음을 입력합니다.

```
# nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7. handles를 사용하여 규칙 세트를 다시 표시합니다. 나중에 추가된 규칙이 지정된 위치에 추가되었는지 확인합니다.

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # 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
    reject # handle 4
  }
}
```

8. handle 6을 사용하여 규칙을 제거합니다.

```
# nft delete rule inet nftables_svc INPUT handle 6
```

규칙을 제거하려면 **handle**을 지정해야 합니다.

9. 규칙 세트를 표시하고 제거된 규칙이 더 이상 존재하지 않는지 확인합니다.

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # 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
    reject # handle 4
  }
}
```

10. **INPUT** 체인에서 나머지 모든 규칙을 제거합니다.

```
# nft flush chain inet nftables_svc INPUT
```

11. 규칙 세트를 표시하고 **INPUT** 체인이 비어 있는지 확인합니다.

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
    type filter hook input priority filter; policy accept
  }
}
```

12. **INPUT** 체인을 삭제합니다.

```
# nft delete chain inet nftables_svc INPUT
```

이 명령을 사용하여 여전히 규칙이 포함된 체인을 삭제할 수도 있습니다.

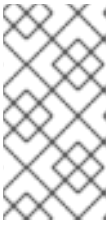
13. 규칙 세트를 표시하고 **INPUT** 체인이 삭제되었는지 확인합니다.

```
# nft list table inet nftables_svc
table inet nftables_svc {
}
```

14. *nftables_svc 테이블을 삭제합니다.*

```
# nft delete table inet nftables_svc
```

이 명령을 사용하여 체인이 여전히 포함된 테이블을 삭제할 수도 있습니다.



참고

*전체 규칙 세트를 삭제하려면 별도의 명령으로 모든 규칙, 체인 및 테이블을 수동으로 삭제하는 대신 **nft flush ruleset** 명령을 사용합니다.*

추가 리소스

- ***nft(8)** 매뉴얼 페이지*

41.4. 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** 매개변수에서는 실제 인터페이스 이름만 사용할 수 있으며 대체 이름(**ltname**)은 지원되지 않습니다.*

41.4.1. NAT 유형

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 공급자(ISP)는 10.0.0.0/8 과 같은 개인 IP 범위를 라우팅하지 않습니다. 네트워크에서 개인 IP 범위를 사용하고 사용자가 인터넷의 서버에 연결할 수 있어야 하는 경우 이러한 범위의 패킷의 소스 IP 주소를 공용 IP 주소에 매핑합니다.

마스커레이딩과 SNAT는 서로 매우 비슷합니다. 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 수신 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.

41.4.2. nftables를 사용하여 마스커레이딩 구성

마스커레이딩을 사용하면 라우터가 인터페이스를 통해 전송된 패킷의 소스 IP를 인터페이스의 IP 주소로 동적으로 변경할 수 있습니다. 즉, 인터페이스에서 새 IP가 할당되면 nftables 가 소스 IP를 교체할 때 새 IP를 자동으로 사용합니다.

ens3 인터페이스를 통해 호스트를 떠나는 패킷의 소스 IP를 ens3 의 IP 세트로 교체합니다.

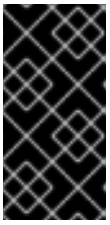
절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. prerouting 및 postrouting 체인을 테이블에 추가합니다.

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



중요

prerouting 체인에 규칙을 추가하지 않더라도 nftables 프레임워크를 사용하려면 들어오는 패킷 응답과 일치해야 합니다.

셸에서 음수 우선 순위 값을 nft 명령의 옵션으로 해석하지 못하도록 -- 옵션을 nft 명령에 전달해야 합니다.

3. ens3 인터페이스의 발신 패킷과 일치하는 후 체인에 규칙을 추가합니다.

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

41.4.3. nftables를 사용하여 소스 NAT 구성

라우터에서 SNAT(Source NAT)를 사용하면 인터페이스를 통해 전송된 패킷의 IP를 특정 IP 주소로 변경할 수 있습니다. 그런 다음 라우터는 발신 패킷의 소스 IP를 대체합니다.

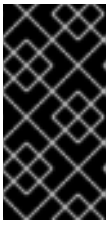
절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. prerouting 및 postrouting 체인을 테이블에 추가합니다.

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



중요

postrouting 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크를 사용하려면 이 체인이 발신 패킷 응답과 일치해야 합니다.

쉘에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. **ens3** 을 통해 나가는 패킷의 소스 **IP**를 **192.0.2.1** 로 대체하는 지연 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

41.4.4. nftables를 사용하여 대상 NAT 구성

대상 **NAT(DNAT)**를 사용하면 라우터의 트래픽을 인터넷에서 직접 액세스할 수 없는 호스트로 리디렉션할 수 있습니다.

예를 들어, **DNAT**를 사용하면 라우터가 포트 **80** 및 **443** 으로 전송된 들어오는 트래픽을 **IP** 주소 **192.0.2.1** 이 있는 웹 서버로 리디렉션합니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. **prerouting** 및 **postrouting** 체인을 테이블에 추가합니다.

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

중요

postrouting 체인에 규칙을 추가하지 않더라도 **nftables** 프레임워크를 사용하려면 이 체인이 발신 패킷 응답과 일치해야 합니다.

셸에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록 **--** 옵션을 **nft** 명령에 전달해야 합니다.

3. 라우터의 **ens3** 인터페이스에서 **IP** 주소 **192.0.2.1** 을 사용하여 웹 서버로 들어오는 트래픽을 포트 **80** 및 **443** 으로 리디렉션하는 이전 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 웹 서버에서 보낸 사람에게 반환되는 패킷의 소스 주소를 변경합니다.

   a. **ens3** 인터페이스에서 동적 **IP** 주소를 사용하는 경우 **masquerading** 규칙을 추가합니다.

   ```
   # nft add rule nat postrouting oifname "ens3" masquerade
   ```

   b. **ens3** 인터페이스에서 고정 **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
   ```

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
```

추가 리소스

- NAT 유형

41.4.5. nftables를 사용하여 리디렉션 구성

리디렉션 기능은 체인 후크에 따라 패킷을 로컬 시스템으로 리디렉션하는 특수 대상 네트워크 주소 변환(DNAT)입니다.

예를 들어 로컬 호스트의 포트 22 로 전송된 수신 및 전달 트래픽을 포트 2222 로 리디렉션할 수 있습니다.

절차

1. 테이블을 생성합니다.

```
# nft add table nat
```

2. prerouting chain을 테이블에 추가합니다.

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

셸에서 음수 우선 순위 값을 nft 명령의 옵션으로 해석하지 못하도록 -- 옵션을 nft 명령에 전달해야 합니다.

3. 포트 22에서 들어오는 트래픽을 포트 22 로 리디렉션하는 준비 체인에 규칙을 추가합니다.

```
# nft add rule nat prerouting tcp dport 22 redirect to 2222
```

추가 리소스

- NAT 유형

41.4.6. nftables를 사용하여 flowtable 구성

nftables 유틸리티는 netfilter 프레임워크를 사용하여 네트워크 트래픽에 대해 NAT(네트워크 주소 변환)를 제공하고 패킷 전달을 가속화하기 위한 fastpath 기능 기반 흐름 가능 메커니즘을 제공합니다.

흐름 메커니즘에는 다음과 같은 기능이 있습니다.

- 연결 추적을 사용하여 클래식 패킷 전달 경로를 바이패스합니다.
- 클래식 패킷 처리를 우회하여 라우팅 테이블을 다시 방문하지 않도록 합니다.
- TCP 및 UDP 프로토콜에서만 작동합니다.
- 하드웨어 독립 소프트웨어 빠른 경로.

절차

1. inet family의 예제 테이블 추가:

```
# nft add table inet <example-table>
```

2. Ingress 후크를 사용하여 example-flowtable flowtable을 추가하고 우선순위 유형으로 filter 를 추가합니다.

```
# nft add flowtable inet <example-table> <example-flowtable> { hook ingress priority filter \; devices = { enp1s0, enp7s0 } \; }
```

3. 패킷 처리 테이블의 flowtable에 example-forwardchain flow를 추가합니다.

```
# nft add chain inet <example-table> <example-forwardchain> { type filter hook forward priority filter \; }
```

이 명령은 전달 후크 및 필터 우선 순위로 필터 유형의 흐름을 추가합니다.

4. 설정된 연결 추적 상태가 포함된 규칙을 추가하여 example-flowtable 흐름을 오프로드합니다.

```
# nft add rule inet <example-table> <example-forwardchain> ct state established flow add @<example-flowtable>
```

검증

- example-table 의 속성을 확인합니다.

```
# nft list table inet <example-table>
table inet example-table {
    flowtable example-flowtable {
      hook ingress priority filter
          devices = { enp1s0, enp7s0 }
    }

    chain example-forwardchain {
type filter hook forward priority filter; policy accept;
ct state established flow add @example-flowtable
   }
}
```

추가 리소스

- nft(8) 매뉴얼 페이지

41.5. NFTABLES 명령의 세트 사용

nftables 프레임워크는 세트를 기본적으로 지원합니다. 예를 들어 규칙이 여러 IP 주소, 포트 번호, 인터페이스 또는 기타 일치 기준에 일치해야 하는 경우 세트를 사용할 수 있습니다.

41.5.1. nftables에서 익명 세트 사용

익명 세트에는 규칙에서 직접 사용하는 { 22, 80, 443 } 과 같이 중괄호로 묶은 쉼표로 구분된 값이 포함되어 있습니다. IP 주소 및 기타 일치 기준에도 익명 세트를 사용할 수 있습니다.

익명 세트의 단점은 세트를 변경하려는 경우 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 세트를 사용하도록 에 설명된 대로 named sets을 사용합니다.

사전 요구 사항

- example_chain 체인과 inet family의 example_table 테이블이 있습니다.

절차

1. 예를 들어 포트 22,80, 443 으로 들어오는 트래픽을 허용하는 example_table 의 example_chain 에 규칙을 추가하려면 다음을 실행합니다.

```
# nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2. 선택 사항: **example_table** 로 모든 체인과 해당 규칙을 표시하십시오.

```
# 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
  }
}
```

41.5.2. nftables에서 명명된 세트 사용

nftables 프레임워크는 변경 가능한 **named sets**를 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.

- **192.0.2.1** 또는 **192.0.2.0/24** 와 같은 **IPv4** 주소 또는 범위가 포함된 세트의 **ipv4_addr.**
- **2001:db8:1::1** 또는 **2001:db8:1::1 /64** 와 같은 **IPv6** 주소 또는 범위가 포함된 세트의 **ipv6_addr.**
- **52:54:00:6b:66:42** 와 같은 **MAC(Media Access Control)** 주소 목록이 포함된 세트의 **ether_addr.**
- **inet_proto** 는 **tcp** 와 같은 인터넷 프로토콜 유형 목록이 포함된 세트의 경우입니다.
- **ssh** 와 같은 인터넷 서비스 목록이 포함된 세트의 **inet_service.**
- 패킷 표시 목록이 포함된 집합에 대해 표시합니다. 패킷 표시는 모든 양의 **32**비트 정수 값(**0** 에서 **214748364 7)**일 수 있습니다.

사전 요구 사항

- *example_chain 체인과 example_table 테이블이 있습니다.*

절차

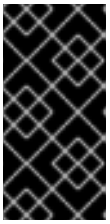
1. *빈 세트를 만듭니다. 다음 예제에서는 IPv4 주소 세트를 생성합니다.*

   - *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

   쉘이 명령 끝부분을 해석하지 못하도록 하려면 백슬래시를 사용하여 host를 이스케이프해야 합니다.

2. *선택 사항: 세트를 사용하는 규칙을 만듭니다. 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 는 여전히 비어 있으므로 규칙은 현재 적용되지 않습니다.

3. *example_set 에 IPv4 주소를 추가합니다.*

   - *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 수 있습니다.

41.5.3. 추가 리소스

- nft(8) 매뉴얼 페이지의 Sets 섹션

41.6. NFTABLES 명령에서 확인 맵 사용

사전이라고도 하는 정점 맵을 사용하면 nft 가 일치 기준을 작업에 매핑하여 패킷 정보를 기반으로 작업을 수행할 수 있습니다.

41.6.1. nftables에서 익명 맵 사용

익명 맵은 규칙에서 직접 사용하는 { match_ crite lack : action } 문입니다. 문에는 쉼표로 구분된 여러 매핑이 포함될 수 있습니다.

익명 맵의 단점은 맵을 변경하려면 규칙을 교체해야 한다는 것입니다. 동적 솔루션의 경우 nftables에서 명명된 맵을 사용하도록 설명된 대로 이름이 지정된 맵을 사용합니다.

예를 들어 익명 맵을 사용하여 IPv4 및 IPv6 프로토콜의 TCP 및 UDP 패킷을 서로 다른 체인으로 라우팅하여 들어오는 TCP 및 UDP 패킷을 별도로 계산할 수 있습니다.

절차

1. 새 테이블을 만듭니다.

```
# nft add table inet example_table
```

2. example_table 에서 tcp_packets 체인을 만듭니다.

```
# 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** 체인 생성

```
# 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 들어오는 트래픽을 필터링하는 **example_table** 에서 **incoming_traffic** 라는 체인을 생성하려면 다음을 수행합니다.

```
# nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7. **anonymous map**이 있는 규칙을 **incoming_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 패킷 수와 바이트 수를 모두 표시합니다.

41.6.2. nftables에서 명명된 맵 사용

nftables 프레임워크는 이름이 지정된 map을 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 부분이 있는 맵의 ipv4_addr 에는 192.0.2.1 과 같은 IPv4 주소가 포함되어 있습니다.
- 2001:db8:1::1 과 같은 IPv6 주소를 포함하는 맵의 ipv6_addr.
- 52:54:00:6b:66:42 와 같은 MAC(Media Access Control) 주소가 일치하는 맵의 ether_addr.
- inet_proto 일치하는 부분이 있는 맵의 경우 tcp 와 같은 인터넷 프로토콜 유형이 포함되어 있습니다.
- inet_service 일치하는 맵의 경우 ssh 또는 22 와 같은 인터넷 서비스 이름 포트 번호가 포함되어 있습니다.
- 일치하는 부분에 패킷 표시가 포함된 맵의 경우 표시됩니다. 패킷 마크는 모든 양의 32 비트 정수 값 (0 에서 2147483647)일 수 있습니다.
- 카운터 값이 일치 하는 맵에 대 한 카운터입니다.A counter for a map whose match part contains a counter value. 카운터 값은 임의의 64비트 정수 값일 수 있습니다.
-

일치 부분에 할당량 값이 포함된 맵의 할당량입니다. 할당량 값은 임의의 **64**비트 정수 값일 수 있습니다.

예를 들어 소스 **IP** 주소에 따라 들어오는 패킷을 허용하거나 삭제할 수 있습니다. 이름 지정된 맵을 사용하면 **IP** 주소 및 작업이 맵에 동적으로 저장되는 동안 이 시나리오를 구성하는 데 단일 규칙만 필요합니다.

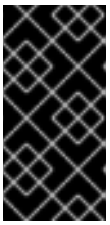
절차

1. 테이블 만들기. 예를 들어 **IPv4** 패킷을 처리하는 **example_table** 라는 테이블을 만들려면 다음을 실행합니다.

```
# nft add table ip example_table
```

2. 체인을 만듭니다. 예를 들어 **example_table** 에서 **example_chain** 이라는 체인을 생성하려면 다음을 수행합니다.

```
# nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```



중요

쉘이 명령 끝부분을 해석하지 못하도록 하려면 백슬래시를 사용하여 **host**를 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_map** 에 정의된 **IPv4** 주소에 작업을 적용하는 **example_table** 의 **example_chain** 에 규칙을 추가합니다.

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. **IPv4** 주소 및 해당 작업을 **example_map** 에 추가합니다.

```
# 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 생성된 규칙과 함께 방화벽은 **192.0.2.1** 의 패킷을 수락하고 **192.0.2.2** 에서 패킷을 삭제합니다.

6. 선택 사항: 다른 **IP** 주소 및 **action** 문을 추가하여 맵을 개선합니다.

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. 선택 사항: 맵에서 항목을 제거합니다.

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. 선택 사항: 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
  }
}
```

41.6.3. 추가 리소스

- **nft(8)** 매뉴얼 페이지의 **Maps** 섹션

41.7. 예제: NFTABLES 스크립트를 사용하여 LAN 및 DMZ 보호

RHEL 라우터의 **nftables** 프레임워크를 사용하여 내부 **LAN**의 네트워크 클라이언트와 **DMZ**의 웹 서버를 인터넷 및 기타 네트워크에서 무단 액세스로부터 보호하는 방화벽 스크립트를 작성하고 설치합니다.

중요

이 예는 예시 목적으로만 사용되며 특정 요구 사항이 있는 시나리오를 설명합니다.

방화벽 스크립트는 네트워크 인프라 및 보안 요구 사항에 따라 크게 달라집니다. 사용자 환경에 대한 스크립트를 작성할 때 **nftables** 방화벽의 개념을 알아보려면 이 예제를 사용합니다.

41.7.1. 네트워크 조건

이 예제의 네트워크에는 다음 조건이 있습니다.

- 라우터는 다음 네트워크에 연결되어 있습니다.
  - 인터페이스 **enp1s0**을 통한 인터넷
  - 내부 **LAN through** 인터페이스 **enp7s0**
  - **enp8s0**을 통한 **DMZ**
- 라우터의 인터넷 인터페이스에는 정적 **IPv4** 주소(**203.0.113.1**)와 **IPv6** 주소(**2001:db8:a::1**)가 할당되어 있습니다.
- 내부 **LAN**의 클라이언트는 **10.0.0.0/24** 범위의 개인 **IPv4** 주소만 사용합니다. 결과적으로 **LAN**에서 인터넷으로 전송되는 경우 소스 네트워크 주소 변환(**SNAT**)이 필요합니다.
- 내부 **LAN**의 관리자는 **IP** 주소 **10.0.0.100** 및 **10.0.0.200** 을 사용합니다.
- **DMZ**는 **198.51.100.0/24** 및 **2001:db8:b::/56** 범위의 공용 **IP** 주소를 사용합니다.
- **DMZ**의 웹 서버는 **198.51.100.5** 및 **2001:db8:b::5 IP** 주소를 사용합니다.

- 라우터는 **LAN** 및 **DMZ**에 있는 호스트에 대한 캐싱 **DNS** 서버 역할을 합니다.

41.7.2. 방화벽 스크립트에 대한 보안 요구 사항

다음은 예제 네트워크의 **nftables** 방화벽에 대한 요구 사항입니다.

- 라우터는 다음을 수행할 수 있어야 합니다.
  - **DNS** 쿼리를 반복적으로 확인합니다.
  - 루프백 인터페이스에서 모든 연결을 수행합니다.
- 내부 **LAN**의 클라이언트는 다음을 수행할 수 있어야 합니다.
  - 라우터에서 실행 중인 캐싱 **DNS** 서버를 쿼리합니다.
  - **DMZ**의 **HTTPS** 서버에 액세스합니다.
  - 인터넷의 모든 **HTTPS** 서버에 액세스합니다.
- 관리자는 **SSH**를 사용하여 라우터 및 **DMZ**의 모든 서버에 액세스할 수 있어야 합니다.
- **DMZ**의 웹 서버는 다음을 수행할 수 있어야 합니다.
  - 라우터에서 실행 중인 캐싱 **DNS** 서버를 쿼리합니다.
  - 인터넷의 **HTTPS** 서버에 액세스하여 업데이트를 다운로드합니다.
- 인터넷의 호스트는 다음을 수행할 수 있어야 합니다.

- *DMZ의 HTTPS 서버에 액세스합니다.*

- *또한 다음과 같은 보안 요구 사항이 있습니다.*
  - *명시적으로 허용되지 않은 연결 시도는 삭제해야 합니다.*
  - *삭제된 패킷이 기록되어야 합니다.*

41.7.3. 삭제된 패킷의 로깅 구성

*기본적으로 **systemd** 는 삭제된 패킷과 같은 커널 메시지를 저널에 기록합니다. 또한 이러한 항목을 별도의 파일에 기록하도록 **rsyslog** 서비스를 구성할 수 있습니다. 로그 파일이 무한대로 확장되지 않도록 하려면 순환 정책을 구성합니다.*

사전 요구 사항

- ***rsyslog** 패키지가 설치되어 있어야 합니다.*
- ***rsyslog** 서비스가 실행 중입니다.*

절차

1. *다음 콘텐츠를 사용하여 **/etc/ECDHE.d/nftables.conf** 파일을 만듭니다.*

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

*이 구성을 사용하여 **rsyslog** 서비스는 **/var/log/ECDHE** 대신 **/var/log/nftables.log** 파일에 패킷을 로그했습니다.*

2. ***rsyslog** 서비스를 다시 시작하십시오.*

```
# systemctl restart rsyslog
```

3.

크기가 10MB를 초과하는 경우 /etc/logrotate.d/nftables.log를 교체하여 /var/log/nftables.log 를 순환하도록 /etc/logrotate.d/nftables 파일을 만듭니다.

```
/var/log/nftables.log {
  size +10M
  maxage 30
  sharedscripts
  postrotate
    /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
  endscript
}
```

maxage 30 설정은 다음 순환 작업 중에 30일이 지난 순환 로그를 제거하도록 정의합니다.

추가 리소스

- rsyslog.conf(5) man page
- ECDHE(8) 매뉴얼 페이지

41.7.4. nftables 스크립트 작성 및 활성화

이 예는 RHEL 라우터에서 실행되며 내부 LAN 및 DMZ의 웹 서버에서 클라이언트를 보호하는 nftables 방화벽 스크립트입니다. 예제에 사용된 방화벽의 네트워크 및 요구 사항에 대한 자세한 내용은 방화벽 스크립트에 대한 네트워크 조건 및 보안 요구 사항을 참조하십시오.

주의

이 nftables 방화벽 스크립트는 데모 목적으로만 사용됩니다. 환경 및 보안 요구 사항에 맞게 조정하지 않고 사용하지 마십시오.

사전 요구 사항

- 네트워크는 네트워크 조건에 설명된 대로 구성됩니다.

절차

1. 다음 콘텐츠를 사용하여 **/etc/nftables/firewall.nft** 스크립트를 만듭니다.

```
# Remove all rules
flush ruleset


# Table for both IPv4 and IPv6 rules
table inet nftables_svc {

  # Define variables for the interface name
  define INET_DEV = enp1s0
  define LAN_DEV  = enp7s0
  define DMZ_DEV  = enp8s0


  # Set with the IPv4 addresses of admin PCs
  set admin_pc_ipv4 {
    type ipv4_addr
    elements = { 10.0.0.100, 10.0.0.200 }
  }


  # Chain for incoming trafic. Default policy: drop
  chain INPUT {
    type filter hook input priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # Accept incoming traffic on loopback interface
    iifname lo accept

    # Allow request from LAN and DMZ to local DNS server
    iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

    # Allow admins PCs to access the router using SSH
    iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

    # Last action: Log blocked packets
    # (packets that were not accepted in previous rules in this chain)
    log prefix "nft drop IN : "
  }


  # Chain for outgoing traffic. Default policy: drop
  chain OUTPUT {
    type filter hook output priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # Accept outgoing traffic on loopback interface
```

```
    oifname lo accept

    # Allow local DNS server to recursively resolve queries
    oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

    # Last action: Log blocked packets
    log prefix "nft drop OUT: "
  }


  # Chain for forwarding traffic. Default policy: drop
  chain FORWARD {
    type filter hook forward priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # IPv4 access from LAN and internet to the HTTPS server in the DMZ
    iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp
dport 443 accept

    # IPv6 access from internet to the HTTPS server in the DMZ
    iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443
accept

    # Access from LAN and DMZ to HTTPS servers on the internet
    iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

    # Last action: Log blocked packets
    log prefix "nft drop FWD: "
  }


  # Postrouting chain to handle SNAT
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;

    # SNAT for IPv4 traffic from LAN to internet
    iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
  }
}
```

2. */etc/nftables/firewall.nft* 스크립트를 */etc/sysconfig/nftables.conf* 파일에 포함합니다.

```
include "/etc/nftables/firewall.nft"
```

3. *IPv4* 전달을 활성화합니다.

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4. *nftables 서비스를 활성화하고 시작합니다.*

```
# systemctl enable --now nftables
```

검증

1. *선택 사항: nftables 규칙 세트를 확인합니다.*

```
# nft list ruleset
...
```

2. *방화벽에서 방지하는 액세스를 시도합니다. 예를 들어 DMZ에서 SSH를 사용하여 라우터에 액세스하십시오.*

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. *로깅 설정에 따라 검색합니다.*

- *차단된 패킷의 systemd 저널:*

```
# journalctl -k -g "nft drop"
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

- *차단된 패킷의 /var/log/nftables.log 파일:*

```
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

41.8. NFTABLES를 사용하여 포트 전달 구성

관리자는 포트 전달을 통해 특정 대상 포트로 전송된 패킷을 다른 로컬 또는 원격 포트로 전달할 수 있습니다.

예를 들어 웹 서버에 공용 IP 주소가 없는 경우 방화벽의 포트 80 및 443 에서 들어오는 패킷을 웹 서버로 전달하는 방화벽에서 포트 전달 규칙을 설정할 수 있습니다. 이 방화벽 규칙을 사용하면 인터넷의 사용

자가 방화벽의 IP 또는 호스트 이름을 사용하여 웹 서버에 액세스할 수 있습니다.

41.8.1. 들어오는 패킷이 다른 로컬 포트로 전달

nftables 를 사용하여 패킷을 전달할 수 있습니다. 예를 들어 8022 포트의 수신 IPv4 패킷을 로컬 시스템의 포트 22 로 전달할 수 있습니다.

절차

1. ip address family를 사용하여 nat 라는 테이블을 만듭니다.

```
# nft add table ip nat
```

2. prerouting 및 postrouting 체인을 테이블에 추가합니다.

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
```

참고

nft 명령에 -- 옵션을 전달하여 쉘에서 음수 우선 순위 값을 nft 명령의 옵션으로 해석하지 못하도록 합니다.

3. 포트 8022 에서 들어오는 패킷을 로컬 포트 22 로 리디렉션하는 준비 체인에 규칙을 추가합니다.

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

41.8.2. 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 인터넷의 사용자는 개인 IP 주소가 있는 호스트에서 실행되는 서비스에 액세스할 수 있습니다.

예를 들어 로컬 포트 443 에서 들어오는 IPv4 패킷을 192.0.2.1 IP 주소를 사용하여 원격 시스템의 동일한 포트 번호로 전달할 수 있습니다.

사전 요구 사항

- 패킷을 전달해야 하는 시스템에서 **root** 사용자로 로그인했습니다.

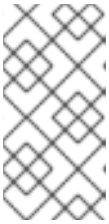
절차

1. **ip address family**를 사용하여 **nat** 라는 테이블을 만듭니다.

```
# nft add table ip nat
```

2. **prerouting** 및 **postrouting** 체인을 테이블에 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

nft 명령에 **--** 옵션을 전달하여 쉘에서 음수 우선 순위 값을 **nft** 명령의 옵션으로 해석하지 못하도록 합니다.

3. 포트 **443** 에서 들어오는 패킷을 **192.0.2.1** 의 동일한 포트로 리디렉션하는 사전 지정 체인에 규칙을 추가합니다.

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 후 행 체인에 규칙을 추가하여 발신 트래픽을 마스커레이드합니다.

```
# 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
```

41.9. NFTABLES를 사용하여 연결 양 제한

nftables 를 사용하여 연결 수를 제한하거나 지정된 양의 연결을 설정하려는 **IP** 주소를 차단하여 너무 많은 시스템 리소스를 사용하지 않도록 할 수 있습니다.

41.9.1. nftables를 사용하여 연결 수 제한

nft 유틸리티의 ct count 매개변수를 사용하면 관리자가 연결 수를 제한할 수 있습니다.

사전 요구 사항

- example_table 의 기본 example_chain 이 존재합니다.

절차

1. IPv4 주소에 대한 동적 세트를 만듭니다.

```
# 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 IP의 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. 선택 사항: 이전 단계에서 만든 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 요소는 SSH 포트에 활성 연결이 있는 IP 주소를 나열합니다. 출력에는 활성 연결 또는 연결이 거부된 경우 출력에 표시되지 않습니다.

41.9.2. 1분 내에 새로 들어오는 TCP 연결을 10개 이상 시도하는 IP 주소 차단

1분 이내에 10개 이상의 IPv4 TCP 연결을 설정하는 호스트를 일시적으로 차단할 수 있습니다.

절차

1. ip address family를 사용하여 필터 테이블을 생성합니다.

```
# nft add table ip filter
```

2. 필터 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
```

시간 초과 5m 매개변수는 계측이 오래된 항목으로 채워지지 않도록 nftables 가 5분 후에 자동으로 항목을 제거하도록 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 }
  }
}
```

41.10. NFTABLES 규칙 디버깅

nftables 프레임워크는 관리자가 규칙을 디버그하고 패킷이 일치하는 경우 다양한 옵션을 제공합니다.

41.10.1. 카운터를 사용하여 규칙 생성

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다.

- 기존 규칙에 카운터를 추가하는 프로시저에 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 **22**에서 **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
  }
}
```

41.10.2. 기존 규칙에 카운터 추가

규칙이 일치하는지 확인하려면 카운터를 사용할 수 있습니다.

- 카운터를 사용하여 새 규칙을 추가하는 프로시저에 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 대체하지만 **counter** 매개 변수로 대체하여 카운터 를 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
  }
}
```

41.10.3. 기존 규칙과 일치하는 패킷 모니터링

nft monitor 명령과 함께 **nftables** 의 추적 기능을 사용하면 관리자가 규칙과 일치하는 패킷을 표시할 수 있습니다. 이 규칙과 일치하는 패킷을 모니터링하는 데 사용하는 규칙에 대한 추적을 활성화할 수 있습니다.

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 교체하지만 메타 **nftrace** 세트 **1** 매개변수로 추적 기능을 추가합니다. 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 nftrace set 1 accept (verdict accept)
...
```

주의

추적이 활성화된 규칙 수와 일치하는 트래픽 양에 따라 **nft monitor** 명령이 많은 출력을 표시할 수 있습니다. **grep** 또는 기타 유틸리티를 사용하여 출력을 필터링합니다.

41.11. NFTABLES 규칙 세트 백업 및 복원

nftables 규칙을 파일에 백업하고 나중에 복원할 수 있습니다. 또한 관리자는 규칙과 함께 파일을 사용하여 다른 서버로 규칙을 전송할 수도 있습니다.

41.11.1. nftables 규칙 세트를 파일에 백업

nft 유틸리티를 사용하여 **nftables** 규칙 세트를 파일로 백업할 수 있습니다.

절차

- nftables 규칙을 백업하려면 다음을 수행합니다.
  - nft list ruleset 형식으로 생성된 형식으로 되어 있습니다.

```
# nft list ruleset > file.nft
```

  - JSON 형식으로 다음을 수행합니다.

```
# nft -j list ruleset > file.json
```

41.11.2. 파일에서 nftables 규칙 세트 복원

파일에서 nftables 규칙 세트를 복원할 수 있습니다.

절차

- nftables 규칙을 복원하려면 다음을 수행합니다.
  - 복원할 파일이 nft list ruleset 에 의해 생성된 형식이거나 nft 명령을 직접 포함하는 경우:

```
# nft -f file.nft
```

  - 복원할 파일이 JSON 형식인 경우:

```
# nft -j -f file.json
```

41.12. 추가 리소스

- Red Hat Enterprise Linux 8에서 nftables 사용
- iptables 뒤에는 무엇이 있습니까? 후자의 경우, 물론 nftables

- *firewalld: 미래는 nftables입니다.*

42장. 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 XDP는 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 Hat은 단일 코어에서 초당 2천 6백만 네트워크 패킷을 삭제했으며 이는 동일한 하드웨어에서 nftable의 드롭 속도보다 훨씬 높습니다.

xdp-filter 유틸리티는 XDP를 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. XDP를 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.

42.1. XDP-FILTER 규칙과 일치하는 네트워크 패킷 삭제

xdp-filter 를 사용하여 네트워크 패킷을 삭제할 수 있습니다.

- *특정 대상 포트로*
- *특정 **IP** 주소에서*
- *특정 **MAC** 주소*

***xdp-filter** 의 **allow** 정책은 모든 트래픽이 허용되는 것을 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 설치되어 있습니다.*
- ***XDP** 프로그램을 지원하는 네트워크 드라이버입니다.*

절차

1. ***Load xdp-filter to process incoming packets on a certain interface, such as enp1s0:***

```
# xdp-filter load enp1s0
```

*기본적으로 **xdp-filter** 는 허용 정책을 사용하며 유틸리티는 규칙과 일치하는 트래픽만 삭제합니다.*

*필요한 경우 **-f** 기능 옵션을 사용하여 **tcp,ipv4** 또는 **ethernet** 과 같은 특정 기능만 활성화합니다. 필요한 기능만 로드하면 패킷 처리 속도가 향상됩니다. 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 프로토콜과만 일치시키려면 **-p protocol** 옵션을 사용합니다.

- **192.0.2.1** 으로부터 수신되는 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 수신되는 패킷을 삭제하려면 다음을 입력합니다.

```
# xdp-filter ether 00:53:00:AA:07:BE -m src
```

검증

- 삭제 및 허용된 패킷에 대한 통계를 표시하려면 다음 명령을 사용합니다.

```
# xdp-filter status
```

추가 리소스

- **XDP-filter(8)** 매뉴얼 페이지

- 개발자가 **xdp-filter** 코드에 관심이 있다면 **Red Hat** 고객 포털에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치하십시오.

42.2. XDP-FILTER 규칙과 일치하는 네트워크 패킷을 제외하고 모든 네트워크 패킷 삭제

xdp-filter 를 사용하여 네트워크 패킷만 허용할 수 있습니다.

- *특정 대상 포트로 이동*
- *특정 **IP** 주소에서 특정 **IP** 주소로*
- *특정 **MAC** 주소에서 특정 **MAC** 주소로*

*이렇게 하려면 필터가 특정 규칙과 일치하는 것을 제외한 모든 네트워크 패킷을 삭제하는 것을 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 커널은 특정 트래픽을 허용하는 규칙을 만들 때까지 이 인터페이스의 모든 패킷을 즉시 삭제합니다. 시스템에서 잠김을 방지하려면 로컬로 명령을 입력하거나 다른 네트워크 인터페이스를 통해 호스트에 연결합니다.*

사전 요구 사항

- ***xdp-tools** 패키지가 설치되어 있습니다.*
- *로컬로 로그인하거나 트래픽을 필터링하지 않으려는 네트워크 인터페이스를 사용합니다.*
- ***XDP** 프로그램을 지원하는 네트워크 드라이버입니다.*

절차

1. ***Load xdp-filter to process packets on a certain interface, such as enp1s0:***

```
# xdp-filter load enp1s0 -p deny
```

*필요한 경우 **-f** 기능 옵션을 사용하여 **tcp,ipv4** 또는 **ethernet** 과 같은 특정 기능만 활성화합니*

다. 필요한 기능만 로드하면 패킷 처리 속도가 향상됩니다. 여러 기능을 사용하려면 쉼표로 구분합니다.

*명령이 오류와 함께 실패하면 네트워크 드라이버에서 **XDP** 프로그램을 지원하지 않습니다.*

2. *패킷을 허용하는 규칙을 추가합니다. 예를 들면 다음과 같습니다.*

- *__22__ 포트로 패킷을 허용하려면 다음을 입력합니다.*

```
# xdp-filter port 22
```

*이 명령은 **TCP** 및 **UDP** 트래픽과 일치하는 규칙을 추가합니다. 특정 프로토콜만 일치시키려면 **-p** 프로토콜 옵션을 명령에 전달합니다.*

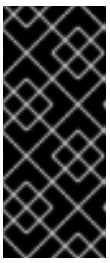
- *__192.0.2.1__ 패킷을 허용하려면 다음을 입력합니다.*

```
# xdp-filter ip 192.0.2.1
```

*__xdp-filter__ 는 **IP** 범위를 지원하지 않습니다.*

- *__MAC__ 주소 __00:53:00:AA:07:BE__ 로 패킷을 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** 모드 옵션을 사용하여 모드를 설정하지 않거나 시스템이 발신 트래픽에 응답할 때 수신되는 트래픽을 허용하는 명시적 규칙을 추가해야 합니다.*

검증

- *삭제 및 허용된 패킷에 대한 통계를 표시하려면 다음 명령을 사용합니다.*

```
# xdp-filter status
```

추가 리소스

- ***XDP-filter(8)** 매뉴얼 페이지.*

- *개발자이고 **xdp-filter** 코드에 관심이 있는 경우 **Red Hat** 고객 포털에서 해당 소스 **RPM(SRPM)**을 다운로드하여 설치합니다.*

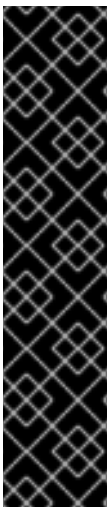
43장. 네트워크 패킷 캡처

네트워크 문제 및 통신을 디버깅하기 위해 네트워크 패킷을 캡처할 수 있습니다. 다음 섹션에서는 네트워크 패킷 캡처에 대한 지침과 추가 정보를 제공합니다.

43.1. XDPDUMP를 사용하여 XDP 프로그램에서 삭제한 패킷을 포함하여 네트워크 패킷 캡처

xdpdump 유틸리티는 네트워크 패킷을 캡처합니다. **tcpdump** 유틸리티와 달리 **xdpdump** 는 이 작업에 대해 **eBPF(extended Berkeley Packet Filter)** 프로그램을 사용합니다. 이를 통해 **xdpdump** 는 **XDP(Express Data Path)** 프로그램에 의해 삭제된 패킷도 캡처할 수 있습니다. **tcpdump** 와 같은 사용자 공간 유틸리티는 이러한 삭제된 패키지와 **XDP** 프로그램에서 수정한 원본 패킷을 캡처할 수 없습니다.

xdpdump 를 사용하여 이미 인터페이스에 연결된 **XDP** 프로그램을 디버그할 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 항목에서 들어오는 패킷을 캡처합니다.



중요

AMD 및 **Intel 64**비트 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 **tcpdump** 와 함께 패킷 디코딩을 위해 사용할 수 있습니다.

사전 요구 사항

- **XDP** 프로그램을 지원하는 네트워크 드라이버입니다.
- **XDP** 프로그램은 **enp1s0** 인터페이스에 로드됩니다. 프로그램이 로드되지 않은 경우 **xdpdump** 는 이전 버전과의 호환성을 위해 **tcpdump** 와 유사한 방식으로 패킷을 캡처합니다.

절차

1. enp1s0 인터페이스의 패킷을 캡처하여 /root/capECDHE.pcap 파일에 작성하려면 다음을 입력합니다.

```
# xdpdump -i enp1s0 -w /root/capture.pcap
```

2. 패킷 캡처를 중지하려면 CtrlC를 누릅니다.

추가 리소스

- xdpdump(8) man page

- 개발자이고 xdpdump 의 소스 코드에 관심이 있는 경우 Red Hat 고객 포털에서 해당 소스 RPM(SRPM)을 다운로드하여 설치합니다.

43.2. 추가 리소스

- tcpdump로 네트워크 패킷을 캡처하는 방법은 무엇입니까?

44장. RHEL 8의 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 eBPF를 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

44.1. RHEL 8의 네트워킹 EBPF 기능 개요

확장 eBPF(extended Berkeley Packet Filter) 네트워킹 프로그램을 RHEL의 다음 후크에 연결할 수 있습니다.

- XDP(Express Express Data Path): 커널 네트워킹 스택이 처리하기 전에 수신된 패킷에 대한 초기 액세스를 제공합니다.
- TC eBPF (직접 동작 플래그 포함): 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 소켓의 프로그래밍 가능 선택을 제공합니다.

- 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 **RHEL**에서 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.

XDP를 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.

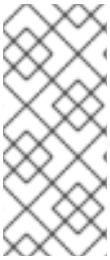
사용 가능한 **XDP** 모드는 다음과 같습니다.

- 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 하는 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.*

RHEL에서 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 8.7에서 Red Hat은 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 Hat은 지원되지 않는 기술 프리뷰로 XDP 기능을 다음과 같이 사용합니다.

- *AMD 및 Intel 64비트가 아닌 아키텍처에 XDP 프로그램 로드. libxdp 라이브러리는 AMD 및 Intel 64비트 이외의 아키텍처에서는 사용할 수 없습니다.*
- *XDP 하드웨어 오프로딩.*

AF_XDP

패킷을 필터링하고 지정된 AF_XDP 소켓으로 리디렉션하는 XDP 프로그램을 사용하면 AF_XDP 프로토콜 제품군에서 하나 이상의 소켓을 사용하여 커널에서 사용자 공간으로 패킷을 빠르게 복사할 수 있습니다.

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 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 eBPF에서 사용자 정의 tc 및 tc 작업을 작성할 수 있습니다. tc 에코시스템의 부분과 함께 이것은 강력한 패킷 처리 기능을 제공하며 여러 컨테이너 네트워킹 오케스트레이션 솔루션의 핵심 요소입니다.

대부분의 경우 classifier만 direct-action 플래그와 마찬가지로 eBPF 분류기에서 동일한 eBPF 프로그램에서 직접 작업을 실행할 수 있습니다. clsact Queueing Discipline (qdisc)은 Ingress 측에서 이를 사용하도록 설계되었습니다.

flow dissector eBPF 프로그램을 사용하면 다른 일부 qdiscs 및 tc ECDHE의 작동에 영향을 미칠 수 있습니다.

eBPF for tc 기능은 RHEL 8.2 이상에서 완전하게 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 tcpdump 는 cBPF 코드로 변환됩니다.

cBPF의 대안으로, 커널은 동일한 목적으로 BPF_PROG_TYPE_SOCKET_FILTER 유형의 eBPF 프로그램을 허용합니다.

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.

컨트롤 그룹

RHEL에서는 cgroup에 연결할 수 있는 여러 유형의 eBPF 프로그램을 사용할 수 있습니다. 지정된

cgroup의 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 connect,bind to ,sendto,recvmsg,getpeername, getsockname 작업 중에 이 프로그램을 호출합니다. 이 프로그램을 통해 IP 주소 및 포트를 변경할 수 있습니다. 이는 eBPF에서 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_INET6_GETSOCKNAME: : 이러한 프로그램을 사용하면 getsockname 및 getpeername 시스템 호출 결과를 덮어쓸 수 있습니다. 이는 eBPF에서 소켓 기반 NAT(네트워크 주소 변환)를 구현할 때 유용합니다.*

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.

스트림 구문

스트림 구문 분석기는 특수 eBPF 맵에 추가된 소켓 그룹에서 작동합니다. 그런 다음 eBPF 프로그램은

커널이 해당 소켓을 수신하거나 보내는 패킷을 처리합니다.

RHEL에서는 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 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.

SO_REUSEPORT 소켓 선택

이 소켓 옵션을 사용하여 여러 소켓을 동일한 IP 주소 및 포트에 바인딩할 수 있습니다. eBPF가 없으면 커널은 연결 해시를 기반으로 수신 소켓을 선택합니다. BPF_PROG_TYPE_SK_REUSEPORT 프로그램을 사용하면 수신 소켓을 완전히 프로그래밍할 수 있습니다.

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.

흐름 분산 섹터

커널이 전체 프로토콜 디코딩을 거치지 않고 패킷 헤더를 처리해야 하는 경우, 분산 됩니다. 예를 들어 tc 하위 시스템, 다중 경로 라우팅, 본딩의 또는 패킷 해시 계산 시 이러한 상황이 발생합니다. 이 경우 커널은 패킷 헤더를 구문 분석하고 패킷 헤더의 정보로 내부 구조를 채웁니다. 이 내부 구문 분석은 BPF_PROG_TYPE_FLOW_DISSECTOR 프로그램을 사용하여 교체할 수 있습니다. RHEL의 eBPF에서 IPv4 및 IPv6을 통해서만 TCP 및 UDP를 차단할 수 있습니다.

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.

TCP 혼잡 제어

struct tcp_congestion_oops 콜백을 구현하는 BPF_PROG_TYPE_STRUCT_OPS 프로그램 그룹을 사용하여 사용자 지정 TCP 정체 제어 알고리즘을 작성할 수 있습니다. 이러한 방식으로 구현된 알고리즘을 시스템에서 기본 제공 커널 알고리즘과 함께 사용할 수 있습니다.

RHEL 8.7에서 Red Hat은 지원되지 않는 기술 프리뷰로 이 기능을 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 8.7**에서 **Red Hat**은 지원되지 않는 기술 프리뷰로 이 기능을 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 8.7**에서 **Red Hat**은 지원되지 않는 기술 프리뷰로 이 기능을 제공합니다.*

*44.2. 네트워크 카드별 **RHEL 8**의 **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	**ena**	제공됨	제공됨	제공됨 [a]	아니요	아니요
Broadcom NetXtreme-C/E 10/25/40/50 기가비트 이더넷	**bnxt_en**	제공됨	제공됨	제공됨 [a]	아니요	아니요

네트워크 카드	드라이버	Basic	리디렉션	대상	HW 오프로드	0-copy
Cavium Thunder Virtual 함수	**nicvf**	제공됨	아니요	아니요	아니요	아니요
Google 가상 NIC(gVNIC) 지원	**GvE**	제공됨	제공됨	제공됨	아니요	제공됨
Intel® 10GbE PCI Express 가상 기능 이더넷	**ixgbevf**	제공됨	아니요	아니요	아니요	아니요
Intel® 10GbE PCI Express 어댑터	**ixgbe**	제공됨	제공됨	제공됨 [a]	아니요	제공됨
Intel® Ethernet Connection E800 시리즈	**ice**	제공됨	제공됨	제공됨 [a]	아니요	제공됨
Intel® 이더넷 컨트롤러 I225-LM/I225-V 제품군	**igc**	제공됨	제공됨	제공됨	아니요	제공됨
Intel® 이더넷 컨트롤러710 제품군	**i40e**	제공됨	제공됨	제공됨 [a] [b]	아니요	제공됨
Intel® PCI Express 기가비트 어댑터	**igb**	제공됨	제공됨	제공됨 [a]	아니요	아니요
Mellanox 5세대 네트워크 어댑터 (ConnectX 시리즈)	**mlx5_core**	제공됨	제공됨	제공됨 [b]	아니요	제공됨
Mellanox Technologies 1/10/40Gbit 이더넷	**mlx4_en**	제공됨	제공됨	아니요	아니요	아니요
Microsoft Azure Network Adapter	**mana**	제공됨	제공됨	제공됨	아니요	아니요
Microsoft Hyper-V 가상 네트워크	**hv_netvsc**	제공됨	제공됨	제공됨	아니요	아니요
Netronome® NFP4000/NFP6000 NIC	**nfp**	제공됨	아니요	아니요	제공됨	아니요
QEMU Virtio 네트워크	**virtio_net**	제공됨	제공됨	제공됨 [a]	아니요	아니요
Qlogic QED 25/40/100Gb 이더넷 NIC	**qede**	제공됨	제공됨	제공됨	아니요	아니요

네트워크 카드	드라이버	Basic	리디렉션	대상	HW 오프로드	0-copy
Solarflare SFC9000/SFC9100/EF100-family	**sfc**	제공됨	제공됨	제공됨 [b]	아니요	아니요
범용 TUN/TAP 장치	**tun**	제공됨	제공됨	제공됨	아니요	아니요
가상 이더넷 쌍 장치	**veth**	제공됨	제공됨	제공됨	아니요	아니요

[a] XDP 프로그램이 인터페이스에 로드된 경우에만 해당합니다.

[b] 가장 큰 CPU 인덱스와 같은 또는 가장 큰 CPU 인덱스와 같은 여러 XDP TX 대기열이 할당되어야 합니다.

범례:

- ***기본: 기본 반환 코드 지원: DROP,ECDHE,ABORTED 및 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 대해 제로 복사 모드를 지원합니다.***

45장. BPF 컴파일러 컬렉션을 사용한 네트워크 추적

BCC(BPF Compiler Collection)는 eBPF(extended Berkeley Packet Filter) 프로그램을 쉽게 생성할 수 있는 라이브러리입니다. eBPF 프로그램의 주요 유틸리티는 오버헤드 또는 보안 문제가 발생하지 않고 운영 체제 성능 및 네트워크 성능을 분석하는 것입니다.

BCC는 사용자가 eBPF의 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.

45.1. BCC-TOOLS 패키지 설치

bcc-tools 패키지를 설치합니다. 이 패키지는 BPF Compiler Collection(BCC) 라이브러리를 종속성으로 설치합니다.

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.

45.2. 커널의 허용 대기열에 추가된 TCP 연결 표시

커널이 **TCP 3**방향 핸드셰이크에서 **ACK** 패킷을 수신한 후 커널은 연결 상태가 **IRQ BLISHED** 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 캡처하고 필터링하는 대신 커널의 **accept()** 함수를 추적하므로 경량입니다. 예를 들어 일반적인 문제 해결에는 **tcpaccept** 를 사용하여 서버가 수락한 새 연결을 표시합니다.

절차

1. 다음 명령을 입력하여 커널 허용 대기열 추적을 시작합니다.

```
# /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** 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- **tcpaccept(8)** 도움말 페이지
- **/usr/share/bcc/tools/doc/tcpaccept_example.txt file**

45.3. 나가는 TCP 연결 시도 추적

tcpconnect 유틸리티는 eBPF 기능을 사용하여 발신 TCP 연결 시도를 추적합니다. 유틸리티 출력에는 실패한 연결도 포함되어 있습니다.

tcpconnect 유틸리티는 예를 들어 패킷을 캡처하고 필터링하는 대신 커널의 connect() 함수를 추적하므로 가벼워집니다.

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 에 연결 세부 정보가 표시됩니다.

2. *Ctrl +C 를눌러 추적 프로세스를 중지합니다.*

추가 리소스

- *TCPConnect(8) 매뉴얼 페이지*
- */usr/share/bcc/tools/doc/tcpconnect_example.txt file*

45.4. 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.

tcpconnlat 유틸리티는 eBPF 기능을 사용하여 전송된 SYN 패킷과 수신된 응답 패킷 간의 시간을 측정합니다.

절차

1. 나가는 연결의 대기 시간 측정을 시작합니다.

```
# /usr/share/bcc/tools/tcpconnlat
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 **tcpconnlat** 는 커널이 응답 패킷을 수신한 후 연결 세부 정보를 표시합니다.

2. **Ctrl +C** 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- **tcpconnlat(8)** 매뉴얼 페이지
- **/usr/share/bcc/tools/doc/tcpconnlat_example.txt file**

45.5. 커널에서 삭제한 TCP 패킷 및 세그먼트에 대한 세부 정보 표시

tcpdrop 유틸리티를 사용하면 관리자가 커널에 의해 삭제된 **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.

리소스가 많이 사용되는 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** 에 삭제된 패키지로 이어지는 커널 스택 추적을 포함하여 연결 세부 정보가 표시됩니다.

2. **Ctrl +C** 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- **tcpdrop(8) man page**
- **/usr/share/bcc/tools/doc/tcpdrop_example.txt file**

45.6. TCP 세션 추적

tcplife 유틸리티는 **eBPF**를 사용하여 열고 닫는 **TCP** 세션을 추적하고 출력 행을 출력하여 각 세션을 요약합니다. 관리자는 **tcplife** 를 사용하여 전송된 트래픽 양을 식별할 수 있습니다.

예를 들어 포트 **22 (SSH)**에 대한 연결을 표시하여 다음 정보를 검색할 수 있습니다.

- 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 에는 연결 세부 정보가 표시됩니다.

2. Ctrl +C 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- tcplife(8) 매뉴얼 페이지
- /usr/share/bcc/tools/doc/tcplife_example.txt file

45.7. 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 재전송 기능을 호출할 때마다 tcpretrans 는 연결 세부 정보를 표시합니다.

2. Ctrl +C 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- tcpretrans(8) man page
- /usr/share/bcc/tools/doc/tcpretrans_example.txt file

45.8. 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
...
```

연결이 해당 상태를 변경할 때마다 tcpstates 는 업데이트된 연결 세부 정보가 포함된 새 행을 표시합니다.

여러 연결이 동시에 상태를 변경하는 경우 첫 번째 열의 소켓 주소를 사용하여 동일한 연결에 속하는 항목을 확인합니다.

2. *Ctrl +C 를눌러 추적 프로세스를 중지합니다.*

추가 리소스

- *tcpstates(8) 매뉴얼 페이지*
- */usr/share/bcc/tools/doc/tcpstates_example.txt file*

45.9. 특정 서브넷에 전송된 TCP 트래픽 요약 및 집계

tcpsubnet 유틸리티는 로컬 호스트에서 서브넷으로 보내는 IPv4 TCP 트래픽을 요약하고 집계하고 고정된 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 서브넷에 대한 모든 트래픽을 계산합니다.

절차에 따라 192.0.2.0/24 및 198.51.100.0/24 서브넷의 트래픽을 계산합니다. 다른 서브넷에 대한 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 양 모니터링을 시작합니다.*

```
# /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 를눌러 추적 프로세스를 중지합니다.*

추가 리소스

- *tcpsubnet(8) man page*
- */usr/share/bcc/tools/doc/tcpsubnet.txt file*

45.10. IP 주소 및 포트를 통한 네트워크 처리량 표시

tcptop 유틸리티는 호스트가 전송 및 수신하는 TCP 트래픽을 킬로바이트로 표시합니다. 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** 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- **tcptop(8)** 도움말 페이지
- **/usr/share/bcc/tools/doc/tcptop.txt file**

45.11. 설정된 TCP 연결 추적

tcptracer 유틸리티는 **TCP** 연결을 연결, 수락 및 종료하는 커널 함수를 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 종료할 때마다 **tcptracer** 에 연결 세부 정보가 표시됩니다.

2. **Ctrl +C** 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- tcptracer(8) 매뉴얼 페이지
- /usr/share/bcc/tools/doc/tcptracer_example.txt file

45.12. IPV4 및 IPV6 추적 시도를 수신 대기

solisten 유틸리티는 모든 IPv4 및 IPv6 수신 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 를눌러 추적 프로세스를 중지합니다.

추가 리소스

- solisten(9) 매뉴얼 페이지
- /usr/share/bcc/tools/doc/solisten_example.txt file

45.13. 소프트 인터럽트의 서비스 시간 요약

softirqs 유틸리티는 소프트 인터럽트 (soft IRQ)를 서비스하는 데 소요되는 시간을 요약하고 이번에 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 를눌러 추적 프로세스를 중지합니다.*

추가 리소스

- *softirqs(8) 매뉴얼 페이지*
- */usr/share/bcc/tools/doc/softirqs_example.txt file*
- *mpstat(1) man page*

45.14. 네트워크 인터페이스의 패킷 크기 및 개수 요약

netqtop 유틸리티는 특정 네트워크 인터페이스의 각 네트워크 큐에 수신 및 전송(TX) 패킷의 속성에 대한 통계를 표시합니다. 통계는 다음과 같습니다.

- *초당 바이트 수(BPS)*
- *초당 패킷 (PPS)*
- *평균 패킷 크기*

- 총 패킷 수

이러한 통계를 생성하기 위해 **netqtop** 는 전송된 **net_dev_start_xmit** 패킷의 이벤트를 수행하고 **netif_receive_skb** 패킷을 수신한 커널 기능을 추적합니다.

절차

1. **2** 초의 시간 간격의 바이트 크기 범위 내에 패킷 수를 표시합니다.

```
# /usr/share/bcc/tools/netqtop -n enp1s0 -i 2

Fri Jan 31 18:08:55 2023
TX
 QueueID avg_size   [0, 64) [64, 512)  [512, 2K)  [2K, 16K)  [16K, 64K)
 0       0       0       0       0       0       0
 Total   0       0       0       0       0       0

RX
 QueueID avg_size   [0, 64) [64, 512)  [512, 2K)  [2K, 16K)  [16K, 64K)
 0       38.0    1       0       0       0       0
 Total   38.0    1       0       0       0       0
--------------------------------------------------------------------------------
Fri Jan 31 18:08:57 2023
TX
 QueueID avg_size   [0, 64) [64, 512)  [512, 2K)  [2K, 16K)  [16K, 64K)
 0       0       0       0       0       0       0
 Total   0       0       0       0       0       0

RX
 QueueID avg_size   [0, 64) [64, 512)  [512, 2K)  [2K, 16K)  [16K, 64K)
 0       38.0    1       0       0       0       0
 Total   38.0    1       0       0       0       0
--------------------------------------------------------------------------------
```

2. **Ctrl+C** 눌러 **netqtop** 를 중지합니다.

추가 리소스

- **netqtop(8)** 매뉴얼 페이지
- **/usr/share/bcc/tools/doc/netqtop_example.txt**

45.15. 추가 리소스

- */usr/share/doc/bcc/README.md*

46장. 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 활동 모니터링
- 네트워크에서 개인 데이터 전송 또는 침입 가로채기

InfiniBand 를 제외한 모든 종류의 네트워크 장치에서 이 모드를 활성화할 수 있습니다.

46.1. 모든 트래픽을 허용하도록 장치 임시 구성

ip 유틸리티를 사용하여 MAC 주소에 관계없이 모든 트래픽을 수락하도록 네트워크 장치를 임시로 구성할 수 있습니다.

절차

1. 선택 사항: 모든 트래픽을 수신하려는 네트워크 인터페이스를 표시합니다.

```
# ip address show
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. 이 속성을 활성화하거나 비활성화하려면 장치를 수정합니다.

- enp1s0 에 대해 accept-all-mac-addresses 모드를 활성화하려면 :

```
# ip link set enp1s0 promisc on
```

- *enp1s0 에 대해 accept-all-mac-addresses 모드를 비활성화하려면 다음을 수행합니다.*

```
# ip link set enp1s0 promisc off
```

검증

- *accept-all-mac-addresses 모드가 활성화되어 있는지 확인합니다.*

```
# 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 활성화되어 있음을 나타냅니다.

46.2. NMCLI를 사용하여 모든 트래픽을 수락하도록 네트워크 장치를 영구적으로 구성

nmcli 유틸리티를 사용하여 MAC 주소에 관계없이 모든 트래픽을 수락하도록 네트워크 장치를 영구적으로 구성할 수 있습니다.

절차

1. *선택 사항: 모든 트래픽을 수신하려는 네트워크 인터페이스를 표시합니다.*

```
# ip address show
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
...
```

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 :*

```
# 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 적용하고 연결을 다시 활성화합니다.

```
# nmcli connection up enp1s0
```

검증

- ethernet.accept-all-mac-addresses 모드가 활성화되어 있는지 확인합니다.

```
# 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.

46.3. NMSTATECTL을 사용하여 모든 트래픽을 수락하도록 네트워크 장치를 영구적으로 구성

nmstatectl 유틸리티를 사용하여 Nmstate API를 통해 MAC 주소와 관계없이 모든 트래픽을 수락하도록 장치를 구성합니다. Nmstate API는 구성을 설정한 후 결과가 구성 파일과 일치하는지 확인합니다. 아무것도 실패하면 nmstatectl 에서 시스템을 잘못된 상태로 두지 않도록 변경 사항을 자동으로 롤백합니다.

사전 요구 사항

- nmstate 패키지가 설치되어 있습니다.
- 장치를 구성하는 데 사용한 enp1s0.yml 파일을 사용할 수 있습니다.

절차

1. enp1s0 연결에 대한 기존 enp1s0.yml 파일을 편집하고 다음 내용을 추가합니다.

```
---
interfaces:
  - name: enp1s0
    type: ethernet
    state: up
    accept -all-mac-address: true
```

*이러한 설정은 모든 트래픽을 수락하도록 **enp1s0** 장치를 구성합니다.*

2. *네트워크 설정을 적용합니다.*

```
# nmstatectl apply ~/enp1s0.yml
```

검증

- ***802-3-ethernet.accept-all-mac-addresses** 모드가 활성화되어 있는지 확인합니다.*

```
# 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(8) man page***
- ***/usr/share/doc/nmstate/examples/** 디렉터리*

47장. NMCLI를 사용하여 네트워크 인터페이스 미러링

네트워크 관리자는 포트 미러링을 사용하여 한 네트워크 장치에서 다른 네트워크 장치로 통신 중인 인바운드 및 아웃바운드 네트워크 트래픽을 복제할 수 있습니다. 인터페이스의 트래픽 미러링은 다음과 같은 상황에서 유용할 수 있습니다.

- 네트워킹 문제를 디버그하고 네트워크 흐름을 조정하려면 다음을 수행합니다.
- 네트워크 트래픽 검사 및 분석
- 침입 감지

사전 요구 사항

- 네트워크 트래픽을 미러링할 네트워크 인터페이스입니다.

절차

1. 다음에서 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:"
```

하위 없이 연결된 **prio qdisc** 는 필터를 연결할 수 있습니다.

3. **ffff: handle**을 사용하여 인그레스 트래픽에 **qdisc** 를 추가합니다.

```
# nmcli connection modify enp1s0 +tc.qdisc "ingress handle ffff:"
```

4. 수신 및 송신 **qdiscs** 의 패킷과 일치하도록 다음 필터를 추가하고 **enp7s0** 에 미러링합니다.

```
# 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

1. **tcpdump** 유틸리티를 설치합니다.

```
# yum install tcpdump
```

2. 대상 장치에 미러링된 트래픽을 표시합니다(**enp7s0**).

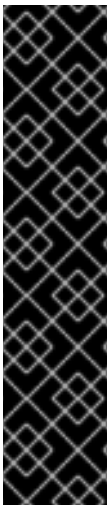
```
# tcpdump -i enp7s0
```

추가 리소스

- **tcpdump**를 사용하여 네트워크 패킷을 캡처하는 방법

48장. NMSTATE-AUTOCONF를 사용하여 LLDP를 사용하여 네트워크 상태 자동 구성

네트워크 장치는 LLDP(링크 계층 검색 프로토콜)를 사용하여 LAN에서 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.

48.1. NMSTATE-AUTOCONF를 사용하여 네트워크 인터페이스 자동 구성

nmstate-autoconf 유틸리티는 LLDP를 사용하여 스위치에 연결된 인터페이스의 VLAN 설정을 식별하여 로컬 장치를 구성합니다.

이 절차에서는 다음 시나리오와 LLDP를 사용하여 VLAN 설정을 브로드캐스트하는 것으로 가정합니다.

- RHEL 서버의 enp1s0 및 enp2s0 인터페이스는 VLAN ID 100 및 VLAN 이름 prod-net 로 구성된 스위치 포트에 연결됩니다.
- RHEL 서버의 enp3s0 인터페이스는 VLAN ID 200 및 VLAN 이름 mgmt-net 로 구성된 스위치 포트에 연결됩니다.

그러면 nmstate-autoconf 유틸리티에서 이 정보를 사용하여 서버에 다음 인터페이스를 생성합니다.

- bond100 - enp1s0 및 enp2s0 포트가 있는 본딩 인터페이스입니다.

- *prod-net - VLAN ID 100 이 포함된 bond100 상단에 있는 VLAN 인터페이스입니다.*
- *Mgmt-net - VLAN ID가 200인 enp3s0 상단에 있는 VLAN 인터페이스*

LLDP가 동일한 VLAN ID를 브로드캐스트하는 다른 스위치 포트에 여러 네트워크 인터페이스를 연결하는 경우 nmstate-autoconf 는 이러한 인터페이스와의 본딩을 생성하고 그 위에 공통 VLAN ID를 구성합니다.

사전 요구 사항

- *nmstate 패키지가 설치되어 있습니다.*
- *LLDP는 네트워크 스위치에서 활성화됩니다.*
- *이더넷 인터페이스가 작동 중입니다.*

절차

1. *이더넷 인터페이스에서 LLDP를 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 display를 표시하고 nmstate-autoconf 에서 생성하는 YAML 구성을 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 LLDP에서 수신된 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.*

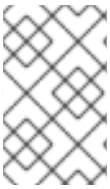
```
# nmstatectl show <interface_name>
```

추가 리소스

- ***nmstate-autoconf(8) man page***

49장. 802.3 링크 설정 구성

자동 협상은 **IEEE 802.3u Fast Ethernet** 프로토콜의 기능입니다. 이는 장치 포트를 대상으로 최적의 속도 성능, 듀플렉스 모드, 링크를 통한 정보 교환을 위한 흐름 제어를 제공합니다. 자동 협상 프로토콜을 사용하면 이더넷을 통한 데이터 전송 성능을 최적으로 수행할 수 있습니다.



참고

자동 협상의 최대 성능을 활용하려면 링크 양쪽에서 동일한 구성을 사용합니다.

49.1. 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. 속도 및 듀플 링크 설정을 설정합니다.

```
# nmcli connection modify Example-connection 802-3-ethernet.auto-negotiate yes 802-3-ethernet.speed 10000 802-3-ethernet.duplex full
```

이 명령은 자동 협상을 활성화하고 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:
...
Speed: 10000 Mb/s
Duplex: Full
Auto-negotiation: on
...
Link detected: yes
```

추가 리소스

- **nm-settings(5)** 도움말 페이지

50장. DPDK 시작하기

DPDK(데이터 플레인 개발 키트)는 사용자 공간에서 패킷 처리를 가속화하기 위한 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 DPDK를 사용합니다.

참고

Red Hat은 실험적 DPDK API를 지원하지 않습니다.

50.1. DPDK 패키지 설치

DPDK를 사용하려면 dpdk 패키지를 설치합니다.

절차

- yum 유틸리티를 사용하여 dpdk 패키지를 설치합니다.

```
# yum install dpdk
```

50.2. 추가 리소스

- 네트워크 어댑터 Fast Datapath 기능 지원 매트릭스

51장. 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. TIPC는 애플리케이션 개발자가 이러한 상황을 처리하기 위한 노력을 최소화하고 이를 정확하고 최적으로 처리할 수 있는 가능성을 극대화합니다. 또한 TIPC는 TCP와 같은 일반 프로토콜보다 더 효율적이고 내결함성 통신을 제공합니다.

51.1. TIPC의 아키텍처

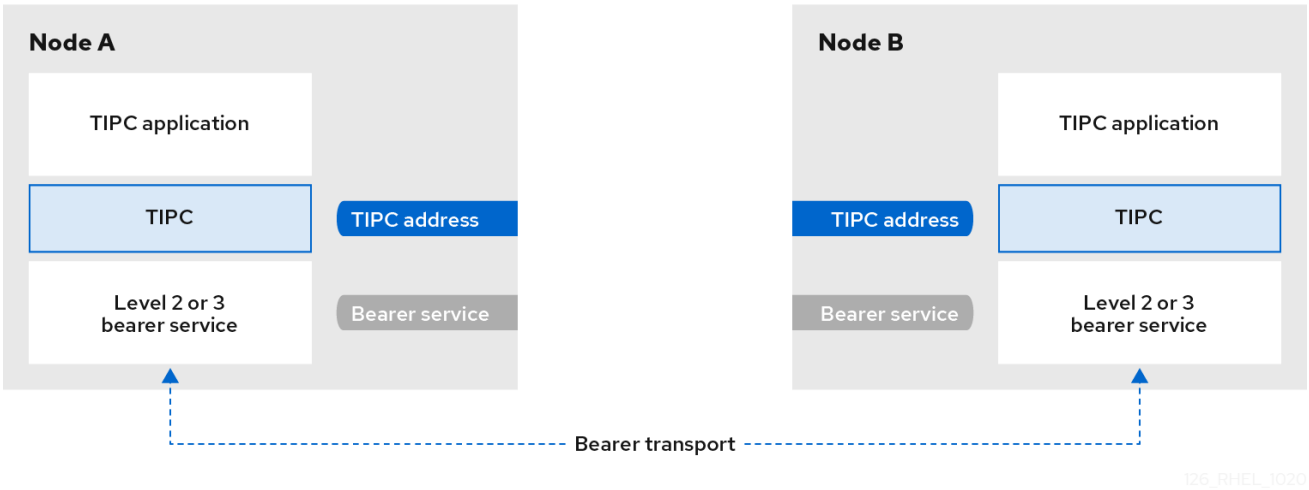
TIPC는 TIPC와 패킷 전송 서비스(Bearer)를 사용하는 애플리케이션 간의 계층이며 전송, 네트워크 및 신호 링크 계층의 수준에 걸쳐 있습니다. 그러나 TIPC는 다른 전송 프로토콜을 전달자로 사용할 수 있으므로, 예를 들어 TCP 연결이 TIPC 신호 링크의 전달자 역할을 할 수 있습니다.

TIPC는 다음 전달자를 지원합니다.

- 이더넷
- InfiniBand
- UDP 프로토콜

TIPC는 모든 TIPC 통신의 엔드포인트인 TIPC 포트 간에 안정적으로 메시지를 전송합니다.

다음은 TIPC 아키텍처 다이어그램입니다.



51.2. 시스템이 부팅될 때 TIPSC 모듈 로드

***TIPC** 프로토콜을 사용하려면 먼저 **tips c** 커널 모듈을 로드해야 합니다. 시스템이 부팅될 때 이 커널 모듈을 자동으로 로드하도록 **Red Hat Enterprise Linux**를 구성할 수 있습니다.*

절차

1. *다음 콘텐츠를 사용하여 **/etc/modules-load.d/tipc.conf** 파일을 만듭니다.*

```
tipc
```

2. *시스템을 재부팅하지 않고 **systemd-modules-load** 서비스를 다시 시작하여 모듈을 로드합니다.*

```
# systemctl start systemd-modules-load
```

검증

1. *다음 명령을 사용하여 **RHEL**에서**Provide c** 모듈을 로드 했는지 확인합니다.*

```
# lsmod | grep tipc
tipc    311296  0
```

*명령에서**Provide c** 모듈에 대한 항목이 표시되지 않으면 **RHEL**이 이를 로드하지 못했습니다.*

추가 리소스

- *modules-load.d(5) man page*

51.3. TIPC 네트워크 생성

TIPC 네트워크를 생성하려면 TIPC 네트워크에 참여해야 하는 각 호스트에서 이 절차를 수행합니다.

중요

명령은 TIPC 네트워크를 임시로만 구성합니다. 노드에서 TIPC를 영구적으로 구성하려면 스크립트에서 이 절차의 명령을 사용하고 시스템이 부팅될 때 해당 스크립트를 실행하도록 RHEL을 구성합니다.

사전 요구 사항

- *tips c 모듈이 로드되었습니다. 자세한 내용은 시스템 부팅 시 tipsc 모듈 로드를참조하십시오.*

절차

1. *선택 사항: UUID 또는 노드의 호스트 이름과 같은 고유한 노드 ID를 설정합니다.*

```
# tipc node set identity host_name
```

ID는 최대 16자 및 숫자로 구성된 고유한 문자열입니다.

이 단계 후에는 ID를 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 3개의 전달자를 구성할 수 있지만 2개를 초과할 수는 없습니다.*

4. TIPC 네트워크에 참여해야 하는 각 노드에서 이전 단계를 모두 반복합니다.

검증

1. 클러스터 구성원의 링크 상태를 표시합니다.

```
# tipc link list
broadcast-link: up
5254006b74be:enp1s0-525400df55d1:enp1s0: up
```

이 출력은 노드 5254006b74be 의 전달자 enp1s0 과 노드 525400df55d1 의 전달자 enp1s0 간의 링크가 임을 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
```

- service 유형 0 이 있는 두 항목은 두 개의 노드가 이 클러스터의 멤버임을 나타냅니다.
- 서비스 유형 1 이 있는 항목은 기본 제공 토폴로지 서비스 추적 서비스를 나타냅니다.
- 서비스 유형 2 가 있는 항목에는 실행 중인 노드와 같은 링크가 표시됩니다. 범위 제한 3741353223 은 10진수 형식으로 피어 끝점의 주소(노드 ID를 기반으로 하는 고유한 32비트 해시 값)를 나타냅니다.

추가 리소스

- hearc-bearer(8) 매뉴얼 페이지
- hearc-namespace(8) 매뉴얼 페이지

51.4. 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

- TIPC를 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 Hat에서 제공되지 않습니다.

- /usr/share/doc/kernel-doc- <kernel_version> /Documentation/output/networking/tipc.html 은 kernel-doc 패키지에서 제공합니다.

52장. NM-CLOUD-SETUP을 사용하여 퍼블릭 클라우드에서 네트워크 인터페이스 자동 구성

일반적으로 VM(가상 머신)에는 DHCP를 통해 구성할 수 있는 인터페이스가 하나만 있습니다. 그러나 DHCP는 인터페이스, IP 서브넷 및 IP 주소와 같은 여러 네트워크 엔티티를 사용하여 VM을 구성할 수 없습니다. 또한 VM 인스턴스가 실행 중인 경우 설정을 적용할 수 없습니다. 이 런타임 구성 문제를 해결하기 위해 nm-cloud-setup 유틸리티는 클라우드 서비스 공급자의 메타데이터 서버에서 구성 정보를 자동으로 검색하고 호스트의 네트워크 구성을 업데이트합니다. 유틸리티는 하나의 인터페이스에서 여러 네트워크 인터페이스, 여러 IP 주소 또는 IP 서브넷을 자동으로 선택하고 실행 중인 VM 인스턴스의 네트워크를 재구성하는 데 도움이 됩니다.

52.1. NM-CLOUD-SETUP 구성 및 사전 배포

퍼블릭 클라우드에서 네트워크 인터페이스를 활성화하고 구성하려면 nm-cloud-setup 을 타이머 및 서비스로 실행합니다.

참고

Red Hat Enterprise Linux 온 디맨드 및 AWS golden 이미지에서 nm-cloud-setup 이 이미 활성화되어 있으므로 작업이 필요하지 않습니다.

사전 요구 사항

- 네트워크 연결이 있습니다.
- 연결에서는 DHCP를 사용합니다.

  기본적으로 NetworkManager는 DHCP를 사용하는 연결 프로필을 생성합니다. /etc/NetworkManager/NetworkManager.conf 에서 no-auto-default 매개변수를 설정하기 때문에 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 서비스의 snap-in 파일을 생성하고 실행합니다.

a. 다음 명령을 사용하여 **snap-in** 파일 편집을 시작합니다.

```
# systemctl edit nm-cloud-setup.service
```

서비스를 명시적으로 시작하거나 시스템을 재부팅하여 구성 설정을 적용하는 것이 중요합니다.

b. **systemd snap-in** 파일을 사용하여 **nm-cloud-setup** 으로 클라우드 공급자를 구성합니다. 예를 들어 **Amazon EC2**를 사용하려면 다음을 입력합니다.

```
[Service]
Environment=NM_CLOUD_SETUP_EC2=yes
```

다음 환경 변수를 설정하여 클라우드가 다음을 사용하도록 설정할 수 있습니다.

- **Microsoft Azure**를 위한 **NM_CLOUD_SETUP_AZURE**
- **NM_CLOUD_SETUP_EC2 for Amazon EC2 (AWS)**
- **NM_CLOUD_SETUP_GCP for Google Cloud Platform(GCP)**
- **NM_CLOUD_SETUP_ALIYUN forECDHE Cloud (Aliyun)**

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) man page*
- *이더넷 연결 구성*

52.2. RHEL EC2 인스턴스에서 IMDSV2 및 NM-CLOUD-SETUP의 역할 이해

Amazon EC2의 인스턴스 메타데이터 서비스(IMDS)를 사용하면 실행 중인 RHEL(Red Hat Enterprise Linux) EC2 인스턴스의 인스턴스 메타데이터에 액세스할 수 있는 권한을 관리할 수 있습니다. RHEL EC2 인스턴스는 세션 지향 방법인 IMDS 버전 2(IMDSv2)를 사용합니다. nm-cloud-setup 유틸리티를 사용하여 관리자는 네트워크를 재구성하고 RHEL EC2 인스턴스 실행의 구성을 자동으로 업데이트할 수 있습니다. nm-cloud-setup 유틸리티는 사용자 개입 없이 IMDSv2 토큰을 사용하여 IMDSv2 API 호출을 처리합니다.

- *IMDS는 RHEL EC2 인스턴스의 네이티브 애플리케이션에 대한 액세스를 제공하기 위해 링크-로컬 주소 169.254.169.254 에서 실행됩니다.*
- *애플리케이션 및 사용자에 대한 각 RHEL EC2 인스턴스에 대해 IMDSv2를 지정 및 구성한 후에는 IMDSv1에 더 이상 액세스할 수 없습니다.*
- *IMDSv2를 사용하면 RHEL EC2 인스턴스는 IAM 역할을 사용하지 않고 IAM 역할을 통해 액세스할 수 있는 메타데이터를 유지 관리합니다.*
- *RHEL EC2 인스턴스가 부팅되면 nm-cloud-setup 유틸리티가 자동으로 실행되어 RHEL EC2 인스턴스 API를 사용하기 위한 EC2 인스턴스 API 액세스 토큰을 가져옵니다.*

참고

IMDSv2 토큰을 HTTP 헤더로 사용하여 EC2 환경의 세부 정보를 확인합니다.

추가 리소스

- ***nm-cloud-setup(8) man page***