Red Hat Enterprise Linux 9

네트워킹 인프라 서비스 관리

Red Hat Enterprise Linux 9에서 네트워킹 인프라 서비스 관리 가이드

Last Updated: 2024-08-06

Red Hat Enterprise Linux 9 네트워킹 인프라 서비스 관리

Red Hat Enterprise Linux 9에서 네트워킹 인프라 서비스 관리 가이드

법적 공지

초록

이 문서에서는 Red Hat Enterprise Linux 9에서 DNS 및 DHCP와 같은 네트워킹 핵심 인프라 서비스를 설정하고 관리하는 방법을 설명합니다.

차례

RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 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. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

1장. BIND DNS 서버 설정 및 구성

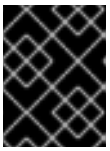
BIND는 IETF(Internet Engineering Task Force) DNS 표준 및 초안 표준을 완벽하게 준수하는 풍부한 기능의 DNS 서버입니다. 예를 들어 관리자는 BIND를 다음과 같이 자주 사용합니다.

- 로컬 네트워크에서 DNS 서버 캐싱
- 영역을 위한 권한 있는 DNS 서버
- 영역에 고가용성을 제공하는 보조 서버

1.1. SELINUX를 사용하여 BIND를 보호하거나 변경 루트 환경에서 실행하는 방법에 대한 고려 사항

BIND 설치를 보호하려면 다음을 수행할 수 있습니다.

- 변경 루트 환경 없이 **named** 서비스를 실행합니다. 이 경우 **강제** 모드에서 SELinux는 알려진 BIND 보안 취약점을 악용하지 않습니다. 기본적으로 Red Hat Enterprise Linux는 SELinux를 **강제** 모드로 사용합니다.



중요

강제 모드에서 SELinux를 사용하여 RHEL에서 BIND를 실행하는 것이 변경 루트 환경에서 BIND를 실행하는 것보다 더 안전합니다.

- change-root 환경에서 **named-chroot** 서비스를 실행합니다.
관리자는 change-root 기능을 사용하여 프로세스의 루트 디렉터리가 / 디렉터리와 다른지 정의할 수 있습니다. **named-chroot** 서비스를 시작하면 BIND에서 루트 디렉토리를 **/var/named/chroot/** 로 전환합니다. 결과적으로 서비스는 **mount --bind** 명령을 사용하여 **/var/named/chroot/** 에서 사용 가능한 **/etc/named-chroot.files** 에 나열된 파일 및 디렉터리를 만들고 프로세스는 **/var/named/chroot/** 외부의 파일에 액세스할 수 없습니다.

BIND를 사용하려면 다음을 수행합니다.

- 일반 모드에서 **named** 서비스를 사용합니다.
- change-root 환경에서는 **named-chroot** 서비스를 사용합니다. 이를 위해서는 **named-chroot** 패키지를 추가로 설치해야 합니다.

1.2. BIND를 캐싱 DNS 서버로 구성

기본적으로 BIND DNS 서버는 성공 및 실패한 조회를 해석하고 캐시합니다. 그러면 서비스는 캐시의 동일한 레코드에 대한 요청에 응답합니다. 이렇게 하면 DNS 조회 속도가 크게 향상됩니다.

사전 요구 사항

- 서버의 IP 주소는 고정됩니다.

절차

1. **bind** 및 **bind-utils** 패키지를 설치합니다.

```
# dnf install bind bind-utils
```

2. 변경 루트 환경에서 BIND를 실행하려면 **bind-chroot** 패키지를 설치합니다.

```
# dnf install bind-chroot
```

SELinux가 **강제** 모드인 호스트에서 BIND를 실행하는 것은 기본적으로 더 안전합니다.

3. **/etc/named.conf** 파일을 편집하고 **options** 문을 다음과 같이 변경합니다.

a. **listen-on** 및 **listen-on-v6** 문을 업데이트하여 BIND에서 수신 대기해야 하는 IPv4 및 IPv6 인터페이스에 지정합니다.

```
listen-on port 53 { 127.0.0.1; 192.0.2.1; };
listen-on-v6 port 53 { ::1; 2001:db8:1::1; };
```

b. **allow-query** 문을 업데이트하여 이 DNS 서버를 쿼리할 수 있는 IP 주소 및 범위에서 구성합니다.

```
allow-query { localhost; 192.0.2.0/24; 2001:db8:1::/64; };
```

c. **allow-recursion** 문을 추가하여 BIND에서 재귀 쿼리를 허용하는 IP 주소 및 범위를 정의합니다.

```
allow-recursion { localhost; 192.0.2.0/24; 2001:db8:1::/64; };
```

주의

서버의 공용 IP 주소에 대한 재귀를 허용하지 마십시오. 그렇지 않으면 서버가 대규모 DNS 수정 공격의 일부가 될 수 있습니다.

d. 기본적으로 BIND에서는 루트 서버에서 권한 있는 DNS 서버로 반복적으로 쿼리하여 쿼리를 확인합니다. 또는 공급자 중 하나와 같은 다른 DNS 서버로 쿼리를 전달하도록 BIND를 구성할 수도 있습니다. 이 경우 BIND에서 쿼리를 전달해야 하는 DNS 서버의 IP 주소 목록과 함께 **forwarders** 문을 추가합니다.

```
forwarders { 198.51.100.1; 203.0.113.5; };
```

대체 동작으로 인해 전달자 서버가 응답하지 않는 경우 BIND에서 쿼리를 재귀적으로 확인합니다. 이 동작을 비활성화하려면 **forward only;** 문을 추가합니다.

4. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

명령이 출력을 표시하지 않으면 구문이 올바른지.

5. 들어오는 DNS 트래픽을 허용하도록 **firewalld** 규칙을 업데이트합니다.

```
# firewall-cmd --permanent --add-service=dns
# firewall-cmd --reload
```

6. BIND를 시작하고 활성화합니다.

```
# systemctl enable --now named
```

변경 루트 환경에서 BIND를 실행하려면 **systemctl enable --now named-chroot** 명령을 사용하여 서비스를 활성화하고 시작합니다.

검증

1. 새로 설정한 DNS 서버를 사용하여 도메인을 확인합니다.

```
# dig @localhost www.example.org
...
www.example.org.    86400    IN    A    198.51.100.34

;; Query time: 917 msec
...
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

처음으로 레코드를 쿼리하면 BIND에서 해당 항목을 해당 캐시에 추가합니다.

2. 이전 쿼리를 반복합니다.

```
# dig @localhost www.example.org
...
www.example.org.    85332    IN    A    198.51.100.34

;; Query time: 1 msec
...
```

캐시된 항목으로 인해 항목이 만료될 때까지 동일한 레코드에 대한 추가 요청이 훨씬 더 빨라집니다.

다음 단계

- 이 DNS 서버를 사용하도록 네트워크에서 클라이언트를 구성합니다. DHCP 서버에서 DNS 서버 설정을 클라이언트에 제공하는 경우 이에 따라 DHCP 서버의 구성을 업데이트합니다.

추가 리소스

- SELinux를 사용하여 BIND를 보호하거나 변경 루트 환경에서 실행하는 방법에 대한 고려 사항
- **named.conf(5)** man page
- **/usr/share/doc/bind/sample/etc/named.conf**

1.3. BIND DNS 서버에 로깅 구성

bind 패키지에서 제공하는 대로 기본 **/etc/named.conf** 파일의 구성은 **default_debug** 채널을 사용하고 **/var/named/data/named.run** 파일에 메시지를 기록합니다. **default_debug** 채널은 서버의 디버그 수준이 0이 아닌 경우에만 로그 항목입니다.

다양한 채널 및 카테고리를 사용하여 파일을 분리하기 위해 정의된 심각도가 있는 다양한 이벤트를 작성하도록 BIND를 구성할 수 있습니다.

사전 요구 사항

- BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. **/etc/named.conf** 파일을 편집하고 **logging** 문에 **카테고리** 및 **채널** 구문을 추가합니다. 예를 들면 다음과 같습니다.

```
logging {
    ...

    category notify { zone_transfer_log; };
    category xfer-in { zone_transfer_log; };
    category xfer-out { zone_transfer_log; };
    channel zone_transfer_log {
        file "/var/named/log/transfer.log" versions 10 size 50m;
        print-time yes;
        print-category yes;
        print-severity yes;
        severity info;
    };

    ...
};
```

   이 예제 설정을 사용하면 BIND에서 영역 전송과 관련된 메시지를 **/var/named/log/transfer.log** 로 기록합니다. BIND에서 최대 **10** 개의 버전의 로그 파일을 생성하고 최대 **50** MB에 도달하면 순환합니다.

   카테고리 문구는 BIND에서 범주의 메시지를 보내는 채널을 정의합니다.

   채널 구는 버전 수, 최대 파일 크기, 심각도 수준 BIND가 채널에 기록되어야 하는 로그 메시지의 대상을 정의합니다. 이벤트의 타임스탬프, 카테고리 및 심각도 로깅과 같은 추가 설정은 선택 사항이지만 디버깅에 유용합니다.

2. 로그 디렉터리가 없는 경우 생성하고 이 디렉터리의 **named** 사용자에게 쓰기 권한을 부여합니다.

```
# mkdir /var/named/log/
# chown named:named /var/named/log/
# chmod 700 /var/named/log/
```

3. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

   명령이 출력을 표시하지 않으면 구문이 올바른지.

4. BIND를 다시 시작합니다.

```
# systemctl restart named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl restart named-chroot** 명령을 사용하여 서비스를 다시 시작합니다.

검증

- 로그 파일의 내용을 표시합니다.

```
# cat /var/named/log/transfer.log
...
06-Jul-2022 15:08:51.261 xfer-out: info: client @0x7fecbc0b0700 192.0.2.2#36121/key
example-transfer-key (example.com): transfer of 'example.com/IN': AXFR started: TSIG
example-transfer-key (serial 2022070603)
06-Jul-2022 15:08:51.261 xfer-out: info: client @0x7fecbc0b0700 192.0.2.2#36121/key
example-transfer-key (example.com): transfer of 'example.com/IN': AXFR ended
```

추가 리소스

- **named.conf(5)** man page

1.4. BIND ACL 작성

BIND의 특정 기능에 대한 액세스를 제어하면 서비스 거부(DoS)와 같은 무단 액세스 및 공격을 방지할 수 있습니다. BIND 액세스 제어 목록(**acl**) 문은 IP 주소 및 범위의 목록입니다. 각 ACL에는 **allow-query** 와 같은 여러 문에서 사용할 수 있는 닉네임이 있어 지정된 IP 주소 및 범위를 참조합니다.

주의

BIND에서는 ACL의 첫 번째 일치 항목만 사용합니다. 예를 들어 ACL **{ 192.0.2/24; !192.0.2.1; !192.0.2.1; }** 및 IP 주소가 **192.0.2.1** 인 호스트를 정의하는 경우 두 번째 항목이 이 주소를 제외하는 경우에도 액세스 권한이 부여됩니다.

BIND에는 다음과 같은 기본 제공 ACL이 있습니다.

- **none**: 호스트를 찾을 수 없습니다.
- **any**: 모든 호스트와 일치합니다.
- **localhost**: 루프백 주소 **127.0.0.1** 및 **::1** 와 일치하며 BIND를 실행하는 서버에서 모든 인터페이스의 IP 주소와 일치시킵니다.
- **localnets**: 루프백 주소 **127.0.0.1** 및 **::1** 와 일치하며 BIND를 실행하는 서버는 직접 연결됩니다.

사전 요구 사항

- BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. **/etc/named.conf** 파일을 편집하고 다음과 같이 변경합니다.

   a. 파일에 **acl** 문을 추가합니다. 예를 들어 **127.0.0.1**,**192.0.2.0/24** 및 **2001:db8:1::/64** 용으로 **internal-networks** 라는 ACL을 생성하려면 다음을 입력합니다.

   ```
   acl internal-networks { 127.0.0.1; 192.0.2.0/24; 2001:db8:1::/64; };
   acl dmz-networks { 198.51.100.0/24; 2001:db8:2::/64; };
   ```

   b. 이를 지원하는 문에서 ACL의 nickname을 사용하십시오. 예를 들면 다음과 같습니다.

   ```
   allow-query { internal-networks; dmz-networks; };
   allow-recursion { internal-networks; };
   ```

2. **/etc/named.conf** 파일의 구문을 확인합니다.

   ```
   # named-checkconf
   ```

   명령이 출력을 표시하지 않으면 구문이 올바른지.

3. BIND를 다시 로드합니다.

   ```
   # systemctl reload named
   ```

   변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

검증

- 구성된 ACL을 사용하는 기능을 트리거하는 작업을 실행합니다. 예를 들어 이 절차의 ACL은 정의된 IP 주소에서 재귀적 쿼리만 허용합니다. 이 경우 ACL 정의 내에 없는 호스트에서 다음 명령을 입력하여 외부 도메인 확인을 시도합니다.

  ```
  # dig +short @192.0.2.1 www.example.com
  ```

  명령에서 출력을 반환하지 않으면 BIND에서 거부된 액세스와 ACL이 작동합니다. 클라이언트에서 자세한 출력 내용을 보려면 **+short** 옵션 없이 명령을 사용합니다.

  ```
  # dig @192.0.2.1 www.example.com
  ...
  ;; WARNING: recursion requested but not available
  ...
  ```

1.5. BIND DNS 서버의 영역 구성

DNS 영역은 도메인 공간의 특정 하위 트리에 대한 리소스 레코드가 있는 데이터베이스입니다. 예를 들어 **example.com** 도메인을 담당하는 경우 BIND에서 해당 도메인에 대한 영역을 설정할 수 있습니다. 결과적으로 클라이언트는 이 영역에 구성된 IP 주소로 **www.example.com** 를 확인할 수 있습니다.

1.5.1. 영역 파일의 SOA 레코드

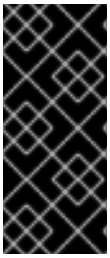
SOA(권한 시작) 레코드는 DNS 영역에 필수 레코드입니다. 예를 들어 여러 DNS 서버가 영역에 대해 권한이 있지만 DNS 확인자에게도 이 레코드가 중요합니다.

BIND의 SOA 레코드에는 다음 구문이 있습니다.

> *name class type mname rname serial refresh retry expire minimum*

읽기 쉽도록 관리자는 일반적으로 영역 파일의 레코드를 세미콜론(**;**)으로 시작하는 주석을 사용하여 여러 행으로 분할합니다. SOA 레코드를 분할하면 괄호는 레코드를 함께 유지합니다.

```
@ IN SOA ns1.example.com. hostmaster.example.com. (
                2022070601 ; serial number
                1d         ; refresh period
                3h         ; retry period
                3d         ; expire time
                3h )       ; minimum TTL
```



중요

정규화된 도메인 이름(FQDN) 끝에 있는 후행 점을 확인합니다. FQDN은 점으로 구분된 여러 도메인 레이블로 구성됩니다. DNS 루트의 레이블은 비어 있으므로 FQDN은 점으로 끝납니다. 따라서 BIND는 후행 점이 없는 이름에 영역 이름을 추가합니다. 후행 점이 없는 호스트 이름(예: **ns1.example.com**)은 **ns1.example.com으로 확장됩니다. .** 이는 기본 이름 서버의 올바른 주소가 아닙니다.

다음은 SOA 레코드의 필드입니다.

- **Name**: 영역의 이름, 소위 **origin**. 이 필드를 **@** 로 설정하면 BIND에서 **/etc/named.conf** 에 정의된 영역 이름으로 확장합니다.
- **클래스**: SOA 레코드에서는 이 필드를 항상 인터넷(**IN**)으로 설정해야 합니다.
- **type**: SOA 레코드에서 이 필드를 항상 **SOA** 로 설정해야 합니다.
- **mname** (마스터 이름): 이 영역의 기본 이름 서버의 호스트 이름입니다.
- R **NAME** (responsible name): 이 영역을 담당하는 사람의 이메일 주소입니다. 형식은 다릅니다. at 기호(**@**)를 점(.)으로 교체해야 합니다**.**
- **serial**: 이 영역 파일의 버전 번호입니다. 보조 이름 서버는 기본 서버의 일련 번호가 더 높은 경우에만 영역의 복사본을 업데이트합니다.
  형식은 모든 숫자 값일 수 있습니다. 일반적으로 사용되는 형식은 < **year><month><two-digit-number>입니다**. 이 형식을 사용하면 이론적으로 영역 파일을 하루에 백 번까지 변경할 수 있습니다.
- **새로 고침**: 영역이 업데이트된 경우 기본 서버를 확인하기 전에 보조 서버가 대기하는 시간입니다.
- **재시도**: 보조 서버에서 실패한 시도 후 기본 서버를 쿼리하도록 다시 시도한 후의 시간입니다.
- **만료** 됨: 이전 시도가 모두 실패한 경우 보조 서버가 주 서버 쿼리를 중지한 후의 시간입니다.
- **minimum**: RFC 2308은 이 필드의 의미를 부정 캐싱 시간으로 변경했습니다. 호환 확인자는 이를 사용하여 **NXDOMAIN** 이름 오류를 캐시하는 기간을 결정합니다.

참고

새로 고침 의 숫자 값,**재시도**,**만료** 및 **최소** 필드는 시간(초)을 정의합니다. 그러나 가독성을 높이기 위해 **m** for minute, **h** for hours, **d** for days과 같은 시간 접미사를 사용합니다. 예를 들어, **3h** 는 3 시간 동안 서 있습니다.

추가 리소스

- RFC 1035: 도메인 이름 - 구현 및 사양
- RFC 1034: 도메인 이름 - 개념 및 기능
- RFC 2308: DNS 쿼리 (DNS 캐시)의 중첩

1.5.2. BIND 기본 서버에서 전달 영역 설정

영역의 이름을 IP 주소 및 기타 정보로 전달합니다. 예를 들어 도메인 **example.com** 을 담당하는 경우 BIND에 전달 영역을 설정하여 **www.example.com** 과 같은 이름을 확인할 수 있습니다.

사전 요구 사항

- BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. **/etc/named.conf** 파일에 영역 정의를 추가합니다.

```
zone "example.com" {
    type master;
    file "example.com.zone";
    allow-query { any; };
    allow-transfer { none; };
};
```

이러한 설정은 다음을 정의합니다.

- 이 서버는 **example.com** 영역의 기본 서버(**마스터 유형**)로 구성됩니다.
- **/var/named/example.com.zone** 파일은 영역 파일입니다. 이 예에서와 같이 상대 경로를 설정하면 이 경로는 **options** 문의 디렉터리에 설정한 **디렉터리를** 기준으로 합니다.
- 모든 호스트는 이 영역을 쿼리할 수 있습니다. 또는 IP 범위 또는 BIND ACL(액세스 제어 목록) nickname을 지정하여 액세스를 제한합니다.
- 호스트를 통해 영역을 전송할 수 없습니다. 보조 서버를 설정하고 보조 서버의 IP 주소에 대해서만 영역 전송을 허용합니다.

2. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

명령이 출력을 표시하지 않으면 구문이 올바른지.

3. **/var/named/example.com.zone** 파일을 생성합니다(예: 다음 콘텐츠를 포함).

```
$TTL 8h
@ IN SOA ns1.example.com. hostmaster.example.com. (
                    2022070601 ; serial number
                    1d         ; refresh period
                    3h         ; retry period
                    3d         ; expire time
                    3h )       ; minimum TTL

               IN NS   ns1.example.com.
               IN MX   10 mail.example.com.

www            IN A    192.0.2.30
www            IN AAAA 2001:db8:1::30
ns1            IN A    192.0.2.1
ns1            IN AAAA 2001:db8:1::1
mail           IN A    192.0.2.20
mail           IN AAAA 2001:db8:1::20
```

이 영역 파일:

- 리소스 레코드의 기본 TTL(Time-to-live) 값을 8시간으로 설정합니다. **h** for hour과 같은 시간 접미사가 없으면 BIND에서 값을 초로 해석합니다.
- 영역에 대한 세부 정보와 필수 SOA 리소스 레코드가 포함되어 있습니다.
- **ns1.example.com** 을 이 영역에 대한 권한 있는 DNS 서버로 설정합니다. 작동하려면 영역에 하나 이상의 이름 서버(**NS**) 레코드가 필요합니다. 그러나 RFC 1912를 준수하려면 두 개 이상의 이름 서버가 필요합니다.
- **mail.example.com** 을 **example.com** 도메인의 메일 교환기(**MX**)로 설정합니다. 호스트 이름 앞에 있는 숫자 값은 레코드의 우선 순위입니다. 더 낮은 값이 있는 항목은 우선 순위가 높습니다.
- **www.example.com**,**mail.example.com**, **ns1.example.com** 의 IPv4 및 IPv6 주소를 설정합니다.

4. **명명** 된 그룹만 읽을 수 있도록 영역 파일에서 보안 권한을 설정합니다.

```
# chown root:named /var/named/example.com.zone
# chmod 640 /var/named/example.com.zone
```

5. **/var/named/example.com.zone** 파일의 구문을 확인합니다.

```
# named-checkzone example.com /var/named/example.com.zone
zone example.com/IN: loaded serial 2022070601
OK
```

6. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

검증

- **example.com** 영역의 다른 레코드를 쿼리하고 출력이 영역 파일에 구성한 레코드와 일치하는지 확인합니다.

```
# dig +short @localhost AAAA www.example.com
2001:db8:1::30

# dig +short @localhost NS example.com
ns1.example.com.

# dig +short @localhost A ns1.example.com
192.0.2.1
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

추가 리소스

- 영역 파일의 SOA 레코드
- BIND ACL 작성
- RFC 1912 - 일반적인 DNS 작동 및 구성 오류

1.5.3. BIND 기본 서버에서 역방향 영역 설정

역방향 영역은 IP 주소를 이름에 매핑합니다. 예를 들어 IP 범위 **192.0.2.0/24** 를 담당하는 경우 BIND에서 역방향 영역을 설정하여 이 범위의 IP 주소를 호스트 이름으로 확인할 수 있습니다.

참고

전체 분류 네트워크의 역방향 영역을 생성하는 경우 해당 영역의 이름을 적절하게 지정합니다. 예를 들어 C 클래스의 경우 **192.0.2.0/24** 클래스의 경우 영역의 이름은 **2.0.192.in-addr.arpa** 입니다. 다른 네트워크 크기에 대한 역방향 영역을 생성하려면(예: **192.0.2.0/28**) 영역 이름은 **28-2.0.192.in-addr.arpa** 입니다.

사전 요구 사항

- BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. **/etc/named.conf** 파일에 영역 정의를 추가합니다.

```
zone "2.0.192.in-addr.arpa" {
    type master;
    file "2.0.192.in-addr.arpa.zone";
    allow-query { any; };
    allow-transfer { none; };
};
```

이러한 설정은 다음을 정의합니다.

- 이 서버는 **2.0.192.in-addr.arpa** 역방향 영역의 기본 서버(**마스터**)로 구성됩니다.
- **/var/named/2.0.192.in-addr.arpa.zone** 파일은 영역 파일입니다. 이 예에서와 같이 상대 경로를 설정하면 이 경로는 **options** 문의 디렉터리에 설정한 **디렉터리를** 기준으로 합니다.
- 모든 호스트는 이 영역을 쿼리할 수 있습니다. 또는 IP 범위 또는 BIND ACL(액세스 제어 목록) nickname을 지정하여 액세스를 제한합니다.
- 호스트를 통해 영역을 전송할 수 없습니다. 보조 서버를 설정하고 보조 서버의 IP 주소에 대해서만 영역 전송을 허용합니다.

2. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

명령이 출력을 표시하지 않으면 구문이 올바른지.

3. 예를 들어 다음 콘텐츠를 사용하여 **/var/named/2.0.192.in-addr.arpa.zone** 파일을 생성합니다.

```
$TTL 8h
@ IN SOA ns1.example.com. hostmaster.example.com. (
                    2022070601 ; serial number
                    1d         ; refresh period
                    3h         ; retry period
                    3d         ; expire time
                    3h )       ; minimum TTL

               IN NS   ns1.example.com.

1              IN PTR  ns1.example.com.
30             IN PTR  www.example.com.
```

이 영역 파일:

- 리소스 레코드의 기본 TTL(Time-to-live) 값을 8시간으로 설정합니다. **h** for hour과 같은 시간 접미사가 없으면 BIND에서 값을 초로 해석합니다.
- 영역에 대한 세부 정보와 필수 SOA 리소스 레코드가 포함되어 있습니다.
- **ns1.example.com** 을 이 역방향 영역에 대한 권한 있는 DNS 서버로 설정합니다. 작동하려면 영역에 하나 이상의 이름 서버(**NS**) 레코드가 필요합니다. 그러나 RFC 1912를 준수하려면 두 개 이상의 이름 서버가 필요합니다.
- 192.0. **2.1 및 19 2.0.2.30** 주소에 대한 포인터(**PTR**) 레코드를 설정합니다.

4. **명명** 된 그룹만 읽을 수 있도록 영역 파일에서 보안 권한을 설정합니다.

```
# chown root:named /var/named/2.0.192.in-addr.arpa.zone
# chmod 640 /var/named/2.0.192.in-addr.arpa.zone
```

5. **/var/named/2.0.192.in-addr.arpa.zone** 파일의 구문을 확인합니다.

```
# named-checkzone 2.0.192.in-addr.arpa /var/named/2.0.192.in-addr.arpa.zone
zone 2.0.192.in-addr.arpa/IN: loaded serial 2022070601
OK
```

6. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

검증

- 역방향 영역에서 다른 레코드를 쿼리하고 출력이 영역 파일에 구성한 레코드와 일치하는지 확인합니다.

```
# dig +short @localhost -x 192.0.2.1
ns1.example.com.

# dig +short @localhost -x 192.0.2.30
www.example.com.
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

추가 리소스

- 영역 파일의 SOA 레코드
- BIND ACL 작성
- RFC 1912 - 일반적인 DNS 작동 및 구성 오류

1.5.4. BIND 영역 파일 업데이트

예를 들어 서버의 IP 주소가 변경되면 영역 파일을 업데이트해야 합니다. 여러 DNS 서버가 영역을 담당하는 경우 기본 서버에서만 이 절차를 수행하십시오. 영역 사본을 저장하는 다른 DNS 서버는 영역 전송을 통해 업데이트를 받습니다.

사전 요구 사항

- 영역이 구성되어 있습니다.
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. 선택 사항: **/etc/named.conf** 파일에서 영역 파일의 경로를 식별합니다.

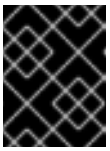
```
options {
    ...
    directory       "/var/named";
}

zone "example.com" {
    ...
    file "example.com.zone";
};
```

영역 정의의 file 문에서 영역 **파일** 의 경로를 찾습니다. 상대 경로는 **options** 문의 디렉터리에 설정된 **디렉터리를** 기준으로 합니다.

2. 영역 파일을 편집합니다.

   a. 필요한 변경을 수행합니다.

   b. 권한 시작(SOA) 레코드의 일련 번호를 늘립니다.



중요

일련 번호가 이전 값과 같거나 낮은 경우 보조 서버는 영역의 사본을 업데이트하지 않습니다.

3. 영역 파일의 구문을 확인합니다.

```
# named-checkzone example.com /var/named/example.com.zone
zone example.com/IN: loaded serial 2022062802
OK
```

4. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

검증

- 추가, 수정 또는 제거한 레코드를 쿼리합니다. 예를 들면 다음과 같습니다.

```
# dig +short @localhost A ns2.example.com
192.0.2.2
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

추가 리소스

- 영역 파일의 SOA 레코드
- BIND 기본 서버에서 전달 영역 설정
- BIND 기본 서버에서 역방향 영역 설정

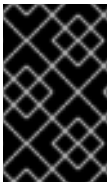
1.5.5. DNSSEC 영역 서명은 자동화된 키 생성 및 영역 유지 관리 기능을 사용한

도메인 이름 시스템 보안 확장 (DNSSEC)을 사용하여 영역에 서명하여 인증 및 데이터 무결성을 보장할 수 있습니다. 이러한 영역에는 추가 리소스 레코드가 포함되어 있습니다. 클라이언트는 이를 사용하여 영역 정보의 진위 여부를 확인할 수 있습니다.

영역에 대한 DNSSEC 정책 기능을 활성화하면 BIND에서 다음 작업을 자동으로 수행합니다.

- 키를 만듭니다.

- 영역 서명
- 키를 다시 서명하고 주기적으로 교체하는 등 영역을 유지 관리합니다.



중요

외부 DNS 서버를 활성화하여 영역의 진위 여부를 확인하려면 영역의 공개 키를 상위 영역에 추가해야 합니다. 이를 수행하는 방법에 대한 자세한 내용은 도메인 공급자 또는 레지스트리에 문의하십시오.

이 절차에서는 BIND에서 **기본** DNSSEC 정책을 사용합니다. 이 정책은 단일 **ECDSAP256SHA** 키 서명을 사용합니다. 또는 사용자 지정 키, 알고리즘 및 타이밍을 사용하는 고유한 정책을 만듭니다.

사전 요구 사항

- DNSSEC를 활성화할 영역입니다.
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.
- 서버는 시간 서버와 시간을 동기화합니다. DNSSEC 검증을 위해 정확한 시스템 시간이 중요합니다.

절차

1. **/etc/named.conf** 파일을 편집하고 DNSSEC를 활성화할 영역에 **dnssec-policy default;** 를 추가합니다.

```
zone "example.com" {
    ...
    dnssec-policy default;
};
```

2. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

3. BIND에서는 공개 키를 **/var/named/K** ***<zone_name>*** **.+** ***<algorithm>*** **+** ***<key_ID>*** **.key** 파일에 저장합니다. 이 파일을 사용하여 영역의 공개 키를 상위 영역에 필요한 형식으로 표시합니다.

   - DS 레코드 형식:

```
# dnssec-dsfromkey /var/named/Kexample.com.+013+61141.key
example.com. IN DS 61141 13 2
3E184188CF6D2521EDFDC3F07CFEE8D0195AACBD85E68BAE0620F638B4B1B027
```

   - DNSKEY 형식:

```
# grep DNSKEY /var/named/Kexample.com.+013+61141.key
example.com. 3600 IN DNSKEY 257 3 13
sjzT3jNEp120aSO4mPEHHSkReHUf7AABNnT8hNRTzD5cKMQSjDJin2l3
5CaKVcWO1pm+HltxUEt+X9dfp8OZkg==
```

4. 영역의 공개 키를 상위 영역에 추가하도록 요청합니다. 이를 수행하는 방법에 대한 자세한 내용은 도메인 공급자 또는 레지스트리에 문의하십시오.

검증

1. DNSSEC 서명을 활성화한 영역에서 레코드에 대한 자체 DNS 서버를 쿼리합니다.

```
# dig +dnssec +short @localhost A www.example.com
192.0.2.30
A 13 3 28800 20220718081258 20220705120353 61141 example.com.
e7Cfh6GuOBMAWsgsHSVTPh+JJSOI/Y6zctzIuqIU1JqEgOOAfL/Qz474
M0sgi54m1Kmnr2ANBKJN9uvOs5eXYw==
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

2. 공개 키가 상위 영역에 추가되고 다른 서버로 전파된 후 서버가 서명된 영역에 대한 쿼리에 인증된 데이터(**ad**) 플래그를 설정하는지 확인합니다.

```
#  dig @localhost example.com +dnssec
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
...
```

추가 리소스

- BIND 기본 서버에서 전달 영역 설정
- BIND 기본 서버에서 역방향 영역 설정

1.6. BIND DNS 서버 간 영역 전송 구성

영역 전송을 통해 영역 사본이 있는 모든 DNS 서버가 최신 데이터를 사용하도록 합니다.

사전 요구 사항

- 향후 기본 서버에서 영역 전송을 설정할 영역이 이미 구성되어 있습니다.
- 향후 보조 서버에서 BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- 두 서버 모두에서 **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. 기존 기본 서버에서 다음을 수행합니다.

   a. 공유 키를 생성하고 **/etc/named.conf** 파일에 추가합니다.

```
# tsig-keygen example-transfer-key | tee -a /etc/named.conf
key "example-transfer-key" {
        algorithm hmac-sha256;
        secret "q7ANbnyliDMuvWgnKOxMLi313JGcTZB5ydMW5CyUGXQ=";
};
```

이 명령은 **tsig-keygen** 명령의 출력을 표시하고 **/etc/named.conf** 에 자동으로 추가합니다.

나중에 보조 서버에서도 명령의 출력이 필요합니다.

b. **/etc/named.conf** 파일에서 영역 정의를 편집합니다.

i. **allow-transfer** 문에서 서버에서 영역을 전송하기 위해 **example-transfer-key** 문에 지정된 키를 제공해야 함을 정의합니다.

```
zone "example.com" {
    ...
    allow-transfer { key example-transfer-key; };
};
```

또는 **allow-transfer** 문에서 BIND ACL(액세스 제어 목록) nicknames를 사용합니다.

ii. 기본적으로 영역을 업데이트한 후 BIND는 이 영역에 이름 서버(**NS**) 레코드가 있는 모든 이름 서버에 알립니다. 보조 서버의 **NS** 레코드를 영역에 추가할 계획이 없는 경우 BIND에서 이 서버에 알리는 것을 구성할 수 있습니다. 이를 위해 이 보조 서버의 IP 주소에 **also-notify** 문을 영역에 추가합니다.

```
zone "example.com" {
    ...
    also-notify { 192.0.2.2; 2001:db8:1::2; };
};
```

c. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

명령이 출력을 표시하지 않으면 구문이 올바른지.

d. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

2. 향후 보조 서버에서 다음을 수행합니다.

a. **/etc/named.conf** 파일을 다음과 같이 편집합니다.

i. 기본 서버에서와 동일한 키 정의를 추가합니다.

```
key "example-transfer-key" {
        algorithm hmac-sha256;
        secret "q7ANbnyliDMuvWgnKOxMLi313JGcTZB5ydMW5CyUGXQ=";
};
```

ii. **/etc/named.conf** 파일에 영역 정의를 추가합니다.

```
zone "example.com" {
    type slave;
    file "slaves/example.com.zone";
```

```
    allow-query { any; };
    allow-transfer { none; };
    masters {
      192.0.2.1 key example-transfer-key;
      2001:db8:1::1 key example-transfer-key;
    };
};
```

이러한 설정 상태:

- 이 서버는 **example.com** 영역의 보조 서버(**slave**)입니다.
- **/var/named/slaves/example.com.zone** 파일은 영역 파일입니다. 이 예에서와 같이 상대 경로를 설정하면 이 경로는 **options** 문의 디렉터리에 설정한 **디렉터리를** 기준으로 합니다. 이 서버가 기본 서버와 보조인 영역 파일을 분리하려면 예를 들어 **/var/named/slaves/** 디렉터리에 저장할 수 있습니다.
- 모든 호스트는 이 영역을 쿼리할 수 있습니다. 또는 IP 범위 또는 ACL 닉네임을 지정하여 액세스를 제한합니다.
- 호스트는 이 서버에서 영역을 전송할 수 없습니다.
- 이 영역의 기본 서버 IP 주소는 **192.0.2.1** 및 **2001:db8:1::2** 입니다. 또는 ACL 닉네임을 지정할 수 있습니다. 이 보조 서버는 **example-transfer-key** 라는 키를 사용하여 주 서버에 인증합니다.

b. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

c. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

3. 선택 사항: 기본 서버에서 영역 파일을 수정하고 새 보조 서버의 **NS** 레코드를 추가합니다.

검증

보조 서버에서 다음을 수행합니다.

1. **named** 서비스의 **systemd** 저널 항목을 표시합니다.

```
# journalctl -u named
...
Jul 06 15:08:51 ns2.example.com named[2024]: zone example.com/IN: Transfer started.
Jul 06 15:08:51 ns2.example.com named[2024]: transfer of 'example.com/IN' from
192.0.2.1#53: connected using 192.0.2.2#45803
Jul 06 15:08:51 ns2.example.com named[2024]: zone example.com/IN: transferred serial
2022070101
Jul 06 15:08:51 ns2.example.com named[2024]: transfer of 'example.com/IN' from
192.0.2.1#53: Transfer status: success
```

```
Jul 06 15:08:51 ns2.example.com named[2024]: transfer of 'example.com/IN' from
192.0.2.1#53: Transfer completed: 1 messages, 29 records, 2002 bytes, 0.003 secs (667333
bytes/sec)
```

변경 루트 환경에서 BIND를 실행하는 경우 **journalctl -u named-chroot** 명령을 사용하여 저널 항목을 표시합니다.

2. BIND에서 영역 파일을 생성했는지 확인합니다.

```
# ls -l /var/named/slaves/
total 4
-rw-r--r--. 1 named named 2736 Jul  6 15:08 example.com.zone
```

기본적으로 보조 서버는 영역 파일을 바이너리 원시 형식으로 저장합니다.

3. 보조 서버에서 전송된 영역의 레코드를 쿼리합니다.

```
# dig +short @192.0.2.2 AAAA www.example.com
2001:db8:1::30
```

이 예제에서는 이 프로세스에서 설정한 보조 서버가 IP 주소 **192.0.2.2** 에서 수신 대기한다고 가정합니다.

추가 리소스

- BIND 기본 서버에서 전달 영역 설정
- BIND 기본 서버에서 역방향 영역 설정
- BIND ACL 작성
- BIND 영역 파일 업데이트

1.7. DNS 레코드를 재정의하도록 BIND에서 응답 정책 영역 구성

DNS 차단 및 필터링을 사용하여 관리자는 DNS 응답을 다시 작성하여 특정 도메인 또는 호스트에 대한 액세스를 차단할 수 있습니다. BIND에서 응답 정책 영역(RPZ)은 이 기능을 제공합니다. **NXDOMAIN** 오류를 반환하거나 쿼리에 응답하지 않는 등 차단된 항목에 대해 다른 작업을 구성할 수 있습니다.

환경에 여러 DNS 서버가 있는 경우 이 절차를 사용하여 기본 서버에서 RPZ를 구성하고 보조 서버에서 RPZ를 사용할 수 있도록 영역 전송을 구성합니다.

사전 요구 사항

- BIND가 이미 구성되어 있습니다(예: 캐싱 이름 서버).
- **named** 또는 **named-chroot** 서비스가 실행 중입니다.

절차

1. **/etc/named.conf** 파일을 편집하고 다음과 같이 변경합니다.
   a. **options** 문에 **response-policy** 정의를 추가합니다.

```
options {
```

```
    ...

    response-policy {
       zone "rpz.local";
    };

    ...
}
```

response-policy 에서 **zone** 문에서 RPZ의 사용자 정의 이름을 설정할 수 있습니다. 그러나 다음 단계에서 영역 정의에서 동일한 이름을 사용해야 합니다.

b. 이전 단계에서 설정한 RPZ의 **영역** 정의를 추가합니다.

```
zone "rpz.local" {
   type master;
   file "rpz.local";
   allow-query { localhost; 192.0.2.0/24; 2001:db8:1::/64; };
   allow-transfer { none; };
};
```

이러한 설정 상태:

- 이 서버는 **rpz.local** 이라는 RPZ의 기본 서버(**마스터 유형**)입니다.
- **/var/named/rpz.local** 파일은 영역 파일입니다. 이 예에서와 같이 상대 경로를 설정하면 이 경로는 **options** 문의 디렉터리에 설정한 **디렉터리를** 기준으로 합니다.
- **allow-query** 에 정의된 모든 호스트는 이 RPZ를 쿼리할 수 있습니다. 또는 IP 범위 또는 BIND ACL(액세스 제어 목록) nickname을 지정하여 액세스를 제한합니다.
- 호스트를 통해 영역을 전송할 수 없습니다. 보조 서버를 설정하고 보조 서버의 IP 주소에 대해서만 영역 전송을 허용합니다.

2. **/etc/named.conf** 파일의 구문을 확인합니다.

```
# named-checkconf
```

명령이 출력을 표시하지 않으면 구문이 올바른지.

3. **/var/named/rpz.local** 파일을 생성합니다(예: 다음 콘텐츠를 포함).

```
$TTL 10m
@ IN SOA ns1.example.com. hostmaster.example.com. (
                  2022070601 ; serial number
                  1h         ; refresh period
                  1m         ; retry period
                  3d         ; expire time
                  1m )       ; minimum TTL

           IN NS    ns1.example.com.

example.org      IN CNAME .
*.example.org    IN CNAME .
example.net      IN CNAME rpz-drop.
*.example.net    IN CNAME rpz-drop.
```

이 영역 파일:

- 리소스 레코드의 기본 TTL(Time-to-live) 값을 10분으로 설정합니다. **h** for hour과 같은 시간 접미사가 없으면 BIND에서 값을 초로 해석합니다.
- 영역에 대한 세부 정보를 포함하는 필수 권한 시작(SOA) 리소스 레코드가 포함되어 있습니다.
- **ns1.example.com** 을 이 영역에 대한 권한 있는 DNS 서버로 설정합니다. 작동하려면 영역에 하나 이상의 이름 서버(**NS**) 레코드가 필요합니다. 그러나 RFC 1912를 준수하려면 두 개 이상의 이름 서버가 필요합니다.
- 이 도메인의 **example.org** 및 호스트에 대한 쿼리의 **NXDOMAIN** 오류를 반환합니다.
- 이 도메인에서 **example.net** 및 호스트에 대한 쿼리를 삭제합니다.

전체 작업 및 예제 목록은 IETF 초안: DNS Response Policy Zones (RPZ) 를 참조하십시오.

4. **/var/named/rpz.local** 파일의 구문을 확인합니다.

```
# named-checkzone rpz.local /var/named/rpz.local
zone rpz.local/IN: loaded serial 2022070601
OK
```

5. BIND를 다시 로드합니다.

```
# systemctl reload named
```

변경 루트 환경에서 BIND를 실행하는 경우 **systemctl reload named-chroot** 명령을 사용하여 서비스를 다시 로드합니다.

검증

1. NXDOMAIN 오류를 반환하도록 RPZ에 구성된 **example.org** 에서 호스트를 해결 **하려고** 합니다.

```
# dig @localhost www.example.org
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 30286
...
```

이 예에서는 BIND가 동일한 호스트에서 실행되고 **localhost** 인터페이스의 쿼리에 응답하는 것으로 가정합니다.

2. 쿼리를 삭제하기 위해 RPZ에 구성된 **example.net** 도메인에서 호스트를 해결하려고 합니다.

```
# dig @localhost www.example.net
...
;; connection timed out; no servers could be reached
...
```

추가 리소스

- IETF 초안: DNS Response Policy Zones (RPZ)

1.8. DNSTAP을 사용하여 DNS 쿼리 기록

네트워크 관리자는 DNS(Domain Name System) 세부 정보를 기록하여 DNS 트래픽 패턴을 분석하고 DNS 서버 성능을 모니터링하고 DNS 문제를 해결할 수 있습니다. 고급 방법으로 들어오는 이름 쿼리의 세부 정보를 모니터링하고 기록하려면 **named** 서비스에서 보낸 메시지를 기록하는 **dnstap** 인터페이스를 사용합니다. DNS 쿼리를 캡처하고 기록하여 웹사이트 또는 IP 주소에 대한 정보를 수집할 수 있습니다.

사전 요구 사항

- **bind-9.16.15-3** 패키지 또는 이후 버전이 설치됩니다.

주의

BIND 버전이 이미 설치되어 실행 중인 경우 새 버전의 **BIND** 를 추가하면 기존 버전이 덮어씁니다.

절차

1. **options** 블록에서 **/etc/named.conf** 파일을 편집하여 **dnstap** 및 대상 파일을 활성화합니다.

```
options
{
# ...
dnstap { all; }; # Configure filter
dnstap-output file "/var/named/data/dnstap.bin"; versions 2;
# ...
};
# end of options
```

2. 로깅할 DNS 트래픽 유형을 지정하려면 **/etc/named.conf** 파일의 **dnstap** 블록에 **dnstap** 필터를 추가합니다. 다음 필터를 사용할 수 있습니다.
   - **auth** - 권한 있는 영역 응답 또는 응답.
   - **클라이언트** - 내부 클라이언트 쿼리 또는 응답.
   - **Forwarder** - 쿼리 또는 응답에서 전달합니다.
   - **해결자** - 반복적 해결 쿼리 또는 응답.
   - **업데이트** - 동적 영역 업데이트 요청.
   - All - 위 옵션 중 **모두** 입니다.
   - **쿼리** 또는 **응답** - **쿼리** 또는 **응답** 키워드를 지정하지 않으면 **dnstap** 이 둘 다 기록합니다.

참고

dnstap 필터에는 dnstap {} 블록의 **dnstap {}** 블록의 **dnstap {(모든 | auth | client | forwardr | update) [(쿼리 | 응답)] ; ... }; dnstap {} 블록으로 구분되는 정의가 포함되어 있습니다.**

3. 기록된 패킷에서 **dnstap** 유틸리티의 동작을 사용자 지정하려면 다음과 같이 추가 매개변수를 제공하여 **dnstap-output** 옵션을 수정합니다.

   - **size** (제한되지 않음 | <size>) - 크기가 지정된 제한에 도달하면 **dnstap** 파일을 자동으로 롤백할 수 있습니다.
   - **버전** (제한되지 않음 | <integer>) - 유지할 자동 롤오버된 파일 수를 지정합니다.
   - **접미사** (increment | timestamp) - 롤아웃된 파일의 이름 지정 규칙을 선택합니다. 기본적으로 증가는 **.0** 으로 시작합니다. 또는 timestamp 값을 설정하여 UNIX **타임스탬프** 를 사용할 수 있습니다.
     다음 예제에서는 자동 **응답** 만 요청, **클라이언트** 쿼리, 동적 **업데이트** 의 쿼리 및 응답 모두:

     Example:

     dnstap {auth response; client query; update;};

4. 변경 사항을 적용하려면 **named** 서비스를 다시 시작하십시오.

   # systemctl restart named.service

5. 활성 로그에 대한 주기적인 롤아웃 구성
   다음 예에서 **cron** 스케줄러는 사용자가 편집한 스크립트의 콘텐츠를 하루에 한 번 실행합니다. 값이 **3** 인 **roll** 옵션은 **dnstap** 에서 최대 3개의 백업 로그 파일을 생성할 수 있음을 지정합니다. value **3** 은 **dnstap-output** 변수의 **version** 매개 변수를 재정의하고 백업 로그 파일 수를 3으로 제한합니다. 또한 바이너리 로그 파일이 다른 디렉터리로 이동하여 이름이 변경되고 3개의 백업 로그 파일이 이미 존재하는 경우에도 **.2** 접미사에 도달하지 않습니다. 크기 제한에 따라 바이너리 로그를 자동으로 롤링하는 경우 이 단계를 건너뛸 수 있습니다.

   Example:

   sudoedit /etc/cron.daily/dnstap

   ```
   #!/bin/sh
   rndc dnstap -roll 3
   mv /var/named/data/dnstap.bin.1 /var/log/named/dnstap/dnstap-$(date -I).bin

   # use dnstap-read to analyze saved logs
   sudo chmod a+x /etc/cron.daily/dnstap
   ```

6. **dnstap-read** 유틸리티를 사용하여 사람이 읽을 수 있는 형식으로 로그를 처리하고 분석합니다.
   다음 예에서 **dnstap-read** 유틸리티는 **YAML** 파일 형식으로 출력을 출력합니다.

   Example:

   dnstap-read -y *[file-name]*

2장. 바인딩되지 않은 DNS 서버 설정

바인딩되지 않은 DNS 서버는 검증, 재귀 및 캐싱 DNS 확인자입니다. **또한 unbound 는 보안에 중점을 두고 있으며 여기에는 기본적으로 DNSSEC(Domain Name System Security Extensions)가 활성화되어 있습니다.**

2.1. UNBOUND를 캐싱 DNS 서버로 구성

기본적으로 바인딩되지 않은 DNS 서비스는 확인 및 캐시 성공 및 실패한 조회입니다. 그러면 서비스는 캐시의 동일한 레코드에 대한 요청에 응답합니다.

절차

1. **unbound 패키지를 설치합니다.**

```
# dnf install unbound
```

2. **/etc/cabundle/cabundle.conf 파일을 편집하고 server 절에서 다음과 같이 변경합니다.**

   a. **인터페이스 매개변수를 추가하여 바인딩되지 않은 서비스가 쿼리를 수신 대기하는 IP 주소를 구성합니다. 예를 들면 다음과 같습니다.**

```
interface: 127.0.0.1
interface: 192.0.2.1
interface: 2001:db8:1::1
```

   이러한 설정을 사용하면 unbound 가 지정된 IPv4 및 IPv6 주소에서만 수신 대기합니다.

   인터페이스를 필수로 제한하면 인터넷과 같은 인증되지 않은 네트워크의 클라이언트가 이 DNS 서버로 쿼리를 보낼 수 없습니다.

   b. **access-control 매개변수를 추가하여 클라이언트가 DNS 서비스를 쿼리할 수 있는 서브넷에서 구성합니다. 예를 들면 다음과 같습니다.**

```
access-control: 127.0.0.0/8 allow
access-control: 192.0.2.0/24 allow
access-control: 2001:db8:1::/64 allow
```

3. **unbound** 서비스를 원격으로 관리하기 위한 개인 키 및 인증서를 생성합니다.

```
# systemctl restart unbound-keygen
```

이 단계를 건너뛰면 다음 단계의 구성을 확인하면 누락된 파일이 보고됩니다. 그러나 바인딩되지 않은 서비스는 누락된 경우 파일을 자동으로 생성합니다.

4. 구성 파일을 확인합니다.

```
# unbound-checkconf
unbound-checkconf: no errors in /etc/unbound/unbound.conf
```

5. 들어오는 **DNS** 트래픽을 허용하도록 **firewalld** 규칙을 업데이트합니다.

```
# firewall-cmd --permanent --add-service=dns
# firewall-cmd --reload
```

6. **unbound** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now unbound
```

검증

1. **localhost** 인터페이스에서 수신 대기하는 바인딩되지 않은 **DNS** 서버를 쿼리하여 도메인을 확인합니다.

```
# dig @localhost www.example.com
...
www.example.com.    86400    IN    A    198.51.100.34

;; Query time: 330 msec
...
```

처음으로 레코드를 쿼리한 후 **unbound** 는 해당 캐시에 항목을 추가합니다.

2. 이전 쿼리를 반복합니다.

```
# dig @localhost www.example.com
```

```
...
www.example.com.    85332    IN    A    198.51.100.34

;; Query time: 1 msec
...
```

캐시된 항목으로 인해 항목이 만료될 때까지 동일한 레코드에 대한 추가 요청이 훨씬 더 빨라집니다.

다음 단계

- 이 DNS 서버를 사용하도록 네트워크에서 클라이언트를 구성합니다. 예를 들어 nmcli 유틸리티를 사용하여 NetworkManager 연결 프로필에서 DNS 서버의 IP를 설정합니다.

```
# nmcli connection modify Example_Connection ipv4.dns 192.0.2.1
# nmcli connection modify Example_Connection ipv6.dns 2001:db8:1::1
```

추가 리소스

- unbound.conf(5) man page

3장. DHCP 서비스 제공

DHCP(Dynamic host Configuration Protocol)는 클라이언트에 IP 정보를 자동으로 할당하는 네트워크 프로토콜입니다. dhcpd 서비스를 설정하여 네트워크에서 DHCP 서버 및 DHCP 중계를 제공할 수 있습니다.

3.1. 고정 IP 주소와 동적 IP 주소 지정의 차이점

고정 IP 주소 지정

장치에 고정 IP 주소를 할당하면 수동으로 변경하지 않는 한 이 주소는 시간이 지남에 따라 변경되지 않습니다. 원하는 경우 고정 IP 주소 지정을 사용합니다.

- DNS 및 인증 서버와 같은 서버에 대한 네트워크 주소 일관성을 보장하기 위해
- 다른 네트워크 인프라와 독립적으로 작동하는 대역 외 관리 장치를 사용하려면 다음을 수행합니다.

동적 IP 주소 지정

동적 IP 주소를 사용하도록 장치를 구성하면 시간이 지남에 따라 주소가 변경될 수 있습니다. 이러한 이유로 동적 주소는 호스트를 재부팅한 후 IP 주소가 다를 수 있으므로 네트워크에 연결되는 장치에 일반적으로 사용됩니다.

동적 IP 주소는 보다 유연하고 설정하기 쉽고 관리할 수 있습니다. DHCP(Dynamic Host Control Protocol)는 호스트에 네트워크 구성을 동적으로 할당하는 기존 방법입니다.

참고

정적 또는 동적 IP 주소를 사용할 시기를 정의하는 엄격한 규칙은 없습니다. 사용자의 요구, 기본 설정 및 네트워크 환경에 따라 달라집니다.

3.2. DHCP 트랜잭션 단계

DHCP는 Discovery, Offer, Request, Acknowledgement, DORA 프로세스라고도 하는 네 단계로 작동합니다. DHCP는 이 프로세스를 사용하여 클라이언트에 IP 주소를 제공합니다.

검색

DHCP 클라이언트는 메시지를 전송하여 네트워크에서 **DHCP** 서버를 검색합니다. 이 메시지는 네트워크 및 데이터 링크 계층에서 브로드캐스트됩니다.

offer

DHCP 서버는 클라이언트에서 메시지를 수신하고 **DHCP** 클라이언트에 **IP** 주소를 제공합니다. 이 메시지는 데이터 링크 계층에서 유니캐스트이지만 네트워크 계층에서 브로드캐스트됩니다.

요청

DHCP 클라이언트는 제공된 **IP** 주소에 대한 **DHCP** 서버를 요청합니다. 이 메시지는 데이터 링크 계층에서 유니캐스트이지만 네트워크 계층에서 브로드캐스트됩니다.

감사 인사

DHCP 서버에서 **DHCP** 클라이언트에 확인을 보냅니다. 이 메시지는 데이터 링크 계층에서 유니캐스트이지만 네트워크 계층에서 브로드캐스트됩니다. **DHCP DORA** 프로세스의 최종 메시지입니다.

3.3. DHCPV4 및 DHCPV6에 DHCPD를 사용할 때의 차이점

dhcpd 서비스는 한 서버에서 **DHCPv4** 및 **DHCPv6**를 모두 제공할 수 있도록 지원합니다. 그러나 각 프로토콜에 **DHCP**를 제공하려면 별도의 구성 파일이 있는 **dhcpd** 의 별도의 인스턴스가 필요합니다.

DHCPv4

- 구성 파일: **/etc/dhcp/dhcpd.conf**
- **systemd** 서비스 이름: **dhcpd**

DHCPv6

- 구성 파일: **/etc/dhcp/dhcpd6.conf**
- **systemd** 서비스 이름: **dhcpd6**

3.4. DHCPD 서비스의 리스 데이터베이스

DHCP 리스는 **dhcpd** 서비스에서 네트워크 주소를 클라이언트에 할당하는 기간입니다. **dhcpd** 서비스는 **DHCP** 리스를 다음 데이터베이스에 저장합니다.

- DHCPv4의 경우: /var/lib/dhcpd/dhcpd.leases
- DHCPv6의 경우: /var/lib/dhcpd/dhcpd6.leases

주의

데이터베이스 파일을 수동으로 업데이트하면 데이터베이스가 손상될 수 있습니다.

리스 데이터베이스에는 MAC(Media Access Control) 주소에 할당된 IP 주소 또는 리스가 만료되는 타임스탬프와 같이 할당된 리스에 대한 정보가 포함되어 있습니다. 리스 데이터베이스의 모든 타임스탬프는 UTC(Universal Time)에 있습니다.

dhcpd 서비스는 정기적으로 데이터베이스를 다시 생성합니다.

1. 서비스는 기존 파일의 이름을 변경합니다.
   - /var/lib/dhcpd/dhcpd.leases to /var/lib/dhcpd/dhcpd.leases~
   - /var/lib/dhcpd/dhcpd6.leases to /var/lib/dhcpd/dhcpd6.leases~
2. 이 서비스는 알려진 모든 리스를 새로 생성된 /var/lib/dhcpd/dhcpd.leases 및 /var/lib/dhcpd/dhcpd6.leases 파일에 씁니다.

추가 리소스

- dhcpd.leases(5) man page
- 손상된 리스 데이터베이스 복원

3.5. DHCPV6과 RADVD 비교

IPv6 네트워크에서는 라우터 알림 메시지만 **IPv6** 기본 게이트웨이에 대한 정보를 제공합니다. 결과적으로 기본 게이트웨이 설정이 필요한 서브넷에서 **DHCPv6**을 사용하려면 라우터 알림 데몬(**radvd**)과 같은 라우터 알림 서비스를 추가로 구성해야 합니다.

radvd 서비스는 라우터 알림 패킷의 플래그를 사용하여 **DHCPv6** 서버의 가용성을 발표합니다.

다음 표에서는 **DHCPv6** 및 **radvd** 의 기능을 비교합니다.

	DHCPv6	radvd
기본 게이트웨이에 대한 정보를 제공합니다.	제공되지 않음	제공됨
개인 정보 보호를 위해 임의의 주소 보장	제공됨	제공되지 않음
추가 네트워크 구성 옵션 전송	제공됨	제공되지 않음
미디어 액세스 제어(MAC) 주소를 IPv6 주소에 매핑	제공됨	제공되지 않음

3.6. IPV6 라우터를 위한 RADVD 서비스 구성

라우터 광고 데몬(**radvd**)은 **IPv6** 상태 비저장 자동 구성에 필요한 라우터 알림 메시지를 전송합니다. 이를 통해 사용자는 주소, 설정, 경로를 자동으로 구성하고 이러한 알림을 기반으로 기본 라우터를 선택할 수 있습니다.

참고

radvd 서비스에서만 **/64** 접두사를 설정할 수 있습니다. 다른 접두사를 사용하려면 **DHCPv6**을 사용합니다.

사전 요구 사항

- **root** 사용자로 로그인합니다.

절차

1. **radvd** 패키지를 설치합니다.

```
# dnf install radvd
```

2. /etc/radvd.conf 파일을 편집하고 다음 구성을 추가합니다.

```
interface enp1s0
{
 AdvSendAdvert on;
 AdvManagedFlag on;
 AdvOtherConfigFlag on;

 prefix 2001:db8:0:1::/64 {
 };
};
```

이러한 설정은 2001:db8:0:1::/64 서브넷의 enp1s0 장치에 라우터 알림 메시지를 전송하도록 radvd 를 구성합니다. AdvManaged flags on 설정은 클라이언트가 DHCP 서버에서 IP 주소를 수신해야 함을 정의하고, Adv OtherConfigflag 매개 변수는 클라이언트가 DHCP 서버에서 주소 이외의 정보를 수신해야 함을 정의합니다.

3. 선택적으로 시스템이 부팅될 때 radvd 가 자동으로 시작되도록 구성합니다.

```
# systemctl enable radvd
```

4. radvd 서비스를 시작합니다.

```
# systemctl start radvd
```

5. 선택적으로, 라우터 광고 패키지의 콘텐츠와 구성된 값 radvd send을 표시합니다.

```
# radvdump
```

추가 리소스

- radvd.conf(5) man page
- /usr/share/doc/radvd/radvd.conf.example 파일
- IPv6 라우터 알림에서 64비트 이외의 접두사 길이를 사용할 수 있습니까?

3.7. DHCP 서버의 네트워크 인터페이스 설정

기본적으로 dhcpd 서비스는 서비스의 구성 파일에 정의된 서브넷에 IP 주소가 있는 네트워크 인터페이스에서만 요청합니다.

예를 들어 다음 시나리오에서는 dhcpd 가 enp0s1 네트워크 인터페이스에서만 수신 대기합니다.

- /etc/dhcp/dhcpd.conf 파일에 192.0.2.0/24 네트워크의 서브넷 정의만 있습니다.
- enp0s1 네트워크 인터페이스는 192.0.2.0/24 서브넷에 연결되어 있습니다.
- enp7s0 인터페이스는 다른 서브넷에 연결되어 있습니다.

DHCP 서버에 동일한 네트워크에 연결된 여러 네트워크 인터페이스가 포함되어 있는 경우에만 이 절차를 수행하지만 서비스는 특정 인터페이스에서만 수신 대기해야 합니다.

IPv4, IPv6 또는 두 프로토콜 모두에 DHCP를 제공할지 여부에 따라 다음 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- root 사용자로 로그인합니다.
- dhcp-server 패키지가 설치되어 있어야 합니다.

절차

- IPv4 네트워크의 경우:

1. **/usr/lib/systemd/system/dhcpd.service** 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
```

/usr/lib/systemd/system/dhcpd.service 파일을 편집하지 마십시오. **dhcp-server** 패키지의 향후 업데이트에서는 변경 사항을 재정의할 수 있습니다.

2. **/etc/systemd/system/dhcpd.service** 파일을 편집하고 인터페이스의 이름을 추가합니다. **dhcpd** 는 **ExecStart** 매개변수의 명령에 수신해야 합니다.

```
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --
no-pid $DHCPDARGS enp0s1 enp7s0
```

이 예제에서는 **dhcpd** 가 ***enp0s1*** 및 **enp7s0** 인터페이스에서만 수신 대기하도록 구성합니다.

3. **systemd** 관리자 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

4. **dhcpd** 서비스를 다시 시작합니다.

```
# systemctl restart dhcpd.service
```

- **IPv6** 네트워크의 경우:

1. **/usr/lib/systemd/system/dhcpd6.service** 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp /usr/lib/systemd/system/dhcpd6.service /etc/systemd/system/
```

/usr/lib/systemd/system/dhcpd6.service 파일을 편집하지 마십시오. **dhcp-server** 패키지의 향후 업데이트에서는 변경 사항을 재정의할 수 있습니다.

2. **/etc/systemd/system/dhcpd6.service** 파일을 편집하고 **ExecStart** 매개 변수의 명령에

dhcpd 가 수신해야 하는 인터페이스의 이름을 추가합니다.

```
ExecStart=/usr/sbin/dhcpd -f -6 -cf /etc/dhcp/dhcpd6.conf -user dhcpd -group
dhcpd --no-pid $DHCPDARGS enp0s1 enp7s0
```

이 예제에서는 dhcpd 가 *enp0s1* 및 enp7s0 인터페이스에서만 수신 대기하도록 구성합니다.

3. systemd 관리자 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

4. dhcpd6 서비스를 다시 시작합니다.

```
# systemctl restart dhcpd6.service
```

3.8. DHCP 서버에 직접 연결된 서브넷의 DHCP 서비스 설정

DHCP 서버가 DHCP 요청에 응답해야 하는 서브넷에 직접 연결된 경우 다음 절차를 사용하십시오. 서버의 네트워크 인터페이스에 이 서브넷의 IP 주소가 할당된 경우입니다.

IPv4, IPv6 또는 두 프로토콜 모두에 DHCP를 제공할지 여부에 따라 다음 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- root 사용자로 로그인합니다.
- dhcp-server 패키지가 설치되어 있어야 합니다.

절차

- **IPv4** 네트워크의 경우:

1. **/etc/dhcp/dhcpd.conf** 파일을 편집합니다.

a. 선택적으로 다른 지시문에 이러한 설정이 포함되어 있지 않은 경우 **dhcpd** 에서 기본값으로 사용하는 글로벌 매개 변수를 추가합니다.

```
option domain-name "example.com";
default-lease-time 86400;
```

이 예에서는 **example.com** 으로 연결의 기본 도메인 이름을 설정하고 기본 리스 시간을 **86400** 초(**1**일)로 설정합니다.

b. 새 줄에 **authoritative** 문을 추가합니다.

```
authoritative;
```

중요

authoritative 문이 없으면 **dhcpd** 서비스는 클라이언트에서 풀 외부에 있는 주소를 요청하는 경우 **DHCPNAK** 를 사용하여 **DHCPREQUEST** 메시지를 응답하지 않습니다.

c. 서버 인터페이스에 직접 연결된 각 **IPv4** 서브넷에 대해 서브넷 선언을 추가합니다.

```
subnet 192.0.2.0 netmask 255.255.255.0 {
  range 192.0.2.20 192.0.2.100;
  option domain-name-servers 192.0.2.1;
  option routers 192.0.2.1;
  option broadcast-address 192.0.2.255;
  max-lease-time 172800;
}
```

이 예제에서는 **192.0.2.0/24** 네트워크에 대한 서브넷 선언을 추가합니다. 이 구성을 사용하면 **DHCP** 서버에서 이 서브넷에서 **DHCP** 요청을 전송하는 클라이언트에 다음 설정을 할당합니다.

- range 매개변수에 정의된 범위에서 사용 가능한 IPv4 주소
- 이 서브넷에 대한 DNS 서버의 IP: 192.0.2.1
- 이 서브넷의 기본 게이트웨이: 192.0.2.1
- 이 서브넷의 브로드캐스트 주소: 192.0.2.255
- 이 서브넷의 클라이언트는 최대 임대 시간입니다. 이 서브넷의 클라이언트는 IP를 해제하고 서버에 새 요청을 보냅니다. 172800 초 (2일)

2. 선택적으로 시스템이 부팅될 때 dhcpd 가 자동으로 시작되도록 구성합니다.

```
# systemctl enable dhcpd
```

3. dhcpd 서비스를 시작합니다.

```
# systemctl start dhcpd
```

- IPv6 네트워크의 경우:

1. /etc/dhcp/dhcpd6.conf 파일을 편집합니다.

a. 선택적으로 다른 지시문에 이러한 설정이 포함되어 있지 않은 경우 dhcpd 에서 기본값으로 사용하는 글로벌 매개 변수를 추가합니다.

```
option dhcp6.domain-search "example.com";
default-lease-time 86400;
```

이 예에서는 example.com 으로 연결의 기본 도메인 이름을 설정하고 기본 리스 시간을 86400 초(1일)로 설정합니다.

b.

새 줄에 **authoritative** 문을 추가합니다.

```
authoritative;
```

중요

authoritative 문이 없으면 **dhcpd** 서비스는 클라이언트에서 풀 외부에 있는 주소를 요청하는 경우 **DHCPNAK** 를 사용하여 **DHCPREQUEST** 메시지를 응답하지 않습니다.

c.

서버 인터페이스에 직접 연결된 각 **IPv6** 서브넷에 대해 서브넷 선언을 추가합니다.

```
subnet6 2001:db8:0:1::/64 {
  range6 2001:db8:0:1::20 2001:db8:0:1::100;
  option dhcp6.name-servers 2001:db8:0:1::1;
  max-lease-time 172800;
}
```

이 예제에서는 **2001:db8:0:1::/64** 네트워크에 대한 서브넷 선언을 추가합니다. 이 구성을 사용하면 **DHCP** 서버에서 이 서브넷에서 **DHCP** 요청을 전송하는 클라이언트에 다음 설정을 할당합니다.

- **range6** 매개변수에 정의된 범위에서 사용 가능한 **IPv6** 주소입니다.
- 이 서브넷의 **DNS** 서버의 **IP**는 **2001:db8:0:1::1** 입니다.
- 이 서브넷의 클라이언트가 **IP**를 해제한 후 서버에 새 요청을 전송하는 최대 리스 시간은 **172800** 초(**2**일)입니다.

IPv6에서는 라우터 알림 메시지를 사용하여 기본 게이트웨이를 식별해야 합니다.

2.

선택적으로 시스템이 부팅될 때 **dhcpd6** 이 자동으로 시작되도록 구성합니다.

```
# systemctl enable dhcpd6
```

3. dhcpd6 서비스를 시작합니다.

```
# systemctl start dhcpd6
```

추가 리소스

- dhcp-options(5) man page
- dhcpd.conf(5) 매뉴얼 페이지
- /usr/share/doc/dhcp-server/dhcpd.conf.example 파일
- /usr/share/doc/dhcp-server/dhcpd6.conf.example file

3.9. DHCP 서버에 직접 연결되지 않은 서브넷의 DHCP 서비스 설정

DHCP 서버가 DHCP 요청에 응답해야 하는 서브넷에 직접 연결되지 않은 경우 다음 절차를 사용하십시오. 이는 서버가 제공해야 하는 서브넷에 직접 연결된 DHCP 서버의 인터페이스가 없기 때문에 DHCP 릴레이 에이전트가 DHCP 서버로 요청을 전달하는 경우입니다.

IPv4, IPv6 또는 두 프로토콜 모두에 DHCP를 제공할지 여부에 따라 다음 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- root 사용자로 로그인합니다.
- dhcp-server 패키지가 설치되어 있어야 합니다.

절차

- **IPv4** 네트워크의 경우:

1. **/etc/dhcp/dhcpd.conf** 파일을 편집합니다.

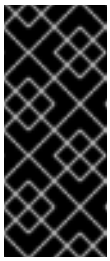
a. 선택적으로 다른 지시문에 이러한 설정이 포함되어 있지 않은 경우 **dhcpd** 에서 기본값으로 사용하는 글로벌 매개 변수를 추가합니다.

```
option domain-name "example.com";
default-lease-time 86400;
```

이 예에서는 **example.com** 으로 연결의 기본 도메인 이름을 설정하고 기본 리스 시간을 **86400** 초(**1**일)로 설정합니다.

b. 새 줄에 **authoritative** 문을 추가합니다.

```
authoritative;
```



중요

authoritative 문이 없으면 **dhcpd** 서비스는 클라이언트에서 풀 외부에 있는 주소를 요청하는 경우 **DHCPNAK** 를 사용하여 **DHCPREQUEST** 메시지를 응답하지 않습니다.

c. 서버 인터페이스에 직접 연결되지 않은 **IPv4** 서브넷에 대해 다음과 같은 공유 네트워크 선언을 추가합니다.

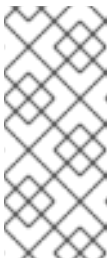
```
shared-network example {
  option domain-name-servers 192.0.2.1;
  ...

  subnet 192.0.2.0 netmask 255.255.255.0 {
    range 192.0.2.20 192.0.2.100;
    option routers 192.0.2.1;
  }

  subnet 198.51.100.0 netmask 255.255.255.0 {
    range 198.51.100.20 198.51.100.100;
    option routers 198.51.100.1;
  }
  ...
}
```

이 예에서는 **192.0.2.0/24** 및 **198.51.100.0/24** 네트워크에 대한 서브넷 선언이 포함된 공유 네트워크 선언을 추가합니다. 이 구성을 사용하면 **DHCP** 서버에서 다음 서브넷 중 하나에서 **DHCP** 요청을 전송하는 클라이언트에 다음 설정을 할당합니다.

- 두 서브넷 모두에서 클라이언트의 **DNS** 서버의 **IP**는 **192.0.2.1** 입니다.
- 클라이언트가 요청을 보낸 서브넷에 따라 **range** 매개변수에 정의된 범위에서 사용 가능한 **IPv4** 주소입니다.
- 기본 게이트웨이는 클라이언트가 요청을 보낸 서브넷에 따라 **192.0.2.1** 또는 **198.51.100.1** 입니다.

d. 서버가 직접 연결되고 위의 **shared-network** 에 지정된 원격 서브넷에 도달하는 데 사용되는 서브넷에 서브넷 선언을 추가합니다.

```
subnet 203.0.113.0 netmask 255.255.255.0 {
}
```



참고

서버가 이 서브넷에 **DHCP** 서비스를 제공하지 않으면 예제에 표시된 대로 서브넷 선언이 비어 있어야 합니다. 직접 연결된 서브넷에 대한 선언이 없으면 **dhcpd** 가 시작되지 않습니다.

2. 선택적으로 시스템이 부팅될 때 **dhcpd** 가 자동으로 시작되도록 구성합니다.

```
# systemctl enable dhcpd
```

3. **dhcpd** 서비스를 시작합니다.

```
# systemctl start dhcpd
```

- **IPv6** 네트워크의 경우:

1. **/etc/dhcp/dhcpd6.conf** 파일을 편집합니다.

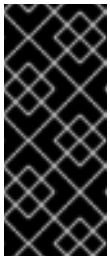
a. 선택적으로 다른 지시문에 이러한 설정이 포함되어 있지 않은 경우 **dhcpd** 에서 기본값으로 사용하는 글로벌 매개 변수를 추가합니다.

```
option dhcp6.domain-search "example.com";
default-lease-time 86400;
```

이 예에서는 **example.com** 으로 연결의 기본 도메인 이름을 설정하고 기본 리스 시간을 **86400** 초(**1**일)로 설정합니다.

b. 새 줄에 **authoritative** 문을 추가합니다.

```
authoritative;
```



중요

authoritative 문이 없으면 **dhcpd** 서비스는 클라이언트에서 풀 외부에 있는 주소를 요청하는 경우 **DHCPNAK** 를 사용하여 **DHCPREQUEST** 메시지를 응답하지 않습니다.

c. 서버 인터페이스에 직접 연결되지 않은 **IPv6** 서브넷에 대해 다음과 같은 공유 네트워크 선언을 추가합니다.

```
shared-network example {
  option domain-name-servers 2001:db8:0:1::1:1
  ...

  subnet6 2001:db8:0:1::1:0/120 {
    range6 2001:db8:0:1::1:20 2001:db8:0:1::1:100
  }

  subnet6 2001:db8:0:1::2:0/120 {
    range6 2001:db8:0:1::2:20 2001:db8:0:1::2:100
  }
  ...
}
```

이 예제에서는 **2001:db8:0:1::1:0/120** 및 **2001:db8:0:1::2:0/120** 네트워크 둘 다에 **subnet6** 선언이 포함된 공유 네트워크 선언을 추가합니다. 이 구성을 사용하면 **DHCP** 서

버에서 다음 서브넷 중 하나에서 **DHCP** 요청을 전송하는 클라이언트에 다음 설정을 할당합니다.

- 두 서브넷 모두에서 클라이언트의 **DNS** 서버의 **IP**는 **2001:db8:0:1::1:1** 입니다.
- 클라이언트가 요청을 보낸 서브넷에 따라 **range6** 매개변수에 정의된 범위에서 사용 가능한 **IPv6** 주소입니다.

  IPv6에서는 라우터 알림 메시지를 사용하여 기본 게이트웨이를 식별해야 합니다.

d. 서버가 직접 연결되고 위의 **shared-network** 에 지정된 원격 서브넷에 도달하는 데 사용되는 서브넷에 **subnet6** 선언을 추가합니다.

```
subnet6 2001:db8:0:1::50:0/120 {
}
```

참고

서버가 이 서브넷에 **DHCP** 서비스를 제공하지 않으면 예제에 표시된 대로 **subnet6** 선언이 비어 있어야 합니다. 직접 연결된 서브넷에 대한 선언이 없으면 **dhcpd** 가 시작되지 않습니다.

2. 선택적으로 시스템이 부팅될 때 **dhcpd6** 이 자동으로 시작되도록 구성합니다.

```
# systemctl enable dhcpd6
```

3. **dhcpd6** 서비스를 시작합니다.

```
# systemctl start dhcpd6
```

추가 리소스

- **dhcp-options(5) man page**

- dhcpd.conf(5) 매뉴얼 페이지
- /usr/share/doc/dhcp-server/dhcpd.conf.example 파일
- /usr/share/doc/dhcp-server/dhcpd6.conf.example file
- DHCP 릴레이 에이전트 설정

3.10. DHCP를 사용하여 호스트에 고정 주소 할당

호스트 선언을 사용하여 호스트의 MAC(Media Access Control) 주소에 고정 IP 주소를 할당하도록 DHCP 서버를 구성할 수 있습니다. 예를 들어 이 방법을 사용하여 항상 동일한 IP 주소를 서버 또는 네트워크 장치에 할당합니다.

IPv4, IPv6 또는 두 프로토콜 모두에 대해 고정 주소를 구성할지 여부에 따라 다음 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- dhcpd 서비스가 구성되어 실행 중입니다.
- root 사용자로 로그인합니다.

절차

- IPv4 네트워크의 경우:

1. /etc/dhcp/dhcpd.conf 파일을 편집합니다.

a. 호스트 선언을 추가합니다.

```
host server.example.com {
 hardware ethernet 52:54:00:72:2f:6e;
 fixed-address 192.0.2.130;
}
```

이 예제에서는 52:54:00:72:2f:6e MAC 주소가 있는 호스트에 192.0.2.130 IP 주소를 항상 할당하도록 DHCP 서버를 구성합니다.

dhcpd 서비스는 호스트 선언의 이름이 아닌 고정 주소 매개변수에 지정된 MAC 주소로 시스템을 식별합니다. 결과적으로 이 이름을 다른 호스트 선언과 일치하지 않는 문자열로 설정할 수 있습니다. 여러 네트워크에 대해 동일한 시스템을 구성하려면 다른 이름을 사용합니다. 그러지 않으면 dhcpd 가 시작되지 않습니다.

b. 선택적으로 이 호스트에 특정된 호스트 선언에 추가 설정을 추가합니다.

2. dhcpd 서비스를 다시 시작합니다.

```
# systemctl start dhcpd
```

- IPv6 네트워크의 경우:

1. /etc/dhcp/dhcpd6.conf 파일을 편집합니다.

a. 호스트 선언을 추가합니다.

```
host server.example.com {
 hardware ethernet 52:54:00:72:2f:6e;
 fixed-address6 2001:db8:0:1::200;
}
```

이 예에서는 DHCP 서버가 항상 2001:db8:0:1::20 IP 주소를 52:54:00:72:2f:6e MAC 주소를 갖는 호스트에 할당하도록 구성합니다.

dhcpd 서비스는 호스트 선언의 이름이 아닌 fixed-address6 매개변수에 지정된 MAC 주소로 시스템을 식별합니다. 결과적으로 다른 호스트 선언에 고유한 경우 이 이름

을 임의의 문자열로 설정할 수 있습니다. 여러 네트워크에 대해 동일한 시스템을 구성하려면 다른 이름을 사용합니다. 그러지 않으면 **dhcpd** 가 시작되지 않습니다.

b. 선택적으로 이 호스트에 특정된 호스트 선언에 추가 설정을 추가합니다.

2. **dhcpd6** 서비스를 다시 시작합니다.

```
# systemctl start dhcpd6
```

추가 리소스

- **dhcp-options(5) man page**
- **/usr/share/doc/dhcp-server/dhcpd.conf.example** 파일
- **/usr/share/doc/dhcp-server/dhcpd6.conf.example file**

3.11. 그룹 선언을 사용하여 여러 호스트, 서브넷 및 공유 네트워크에 동시에 매개 변수 적용

그룹 선언을 사용하면 동일한 매개 변수를 여러 호스트, 서브넷 및 공유 네트워크에 적용할 수 있습니다.

절차는 호스트에 그룹 선언을 사용하는 방법을 설명하지만 단계는 서브넷 및 공유 네트워크에 대해 동일합니다.

IPv4, IPv6 또는 두 프로토콜 모두에 대해 그룹을 구성할지 여부에 따라 다음에 대한 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- **dhcpd** 서비스가 구성되어 실행 중입니다.
- **root** 사용자로 로그인합니다.

절차

- **IPv4** 네트워크의 경우:

1. **/etc/dhcp/dhcpd.conf** 파일을 편집합니다.

a. 그룹 선언을 추가합니다.

```
group {
  option domain-name-servers 192.0.2.1;

  host server1.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address 192.0.2.130;
  }

  host server2.example.com {
    hardware ethernet 52:54:00:1b:f3:cf;
    fixed-address 192.0.2.140;
  }
}
```

이 그룹 정의에서는 두 개의 호스트 항목을 그룹화합니다. **dhcpd** 서비스는 옵션 **domain-name-servers** 매개 변수에 설정된 값을 그룹의 두 호스트에 적용합니다.

b. 선택적으로 이러한 호스트에 고유한 그룹 선언에 추가 설정을 추가합니다.

2. **dhcpd** 서비스를 다시 시작합니다.

```
# systemctl start dhcpd
```

- **IPv6** 네트워크의 경우:

1. /etc/dhcp/dhcpd6.conf 파일을 편집합니다.

   a. 그룹 선언을 추가합니다.

```
group {
  option dhcp6.domain-search "example.com";

  host server1.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address 2001:db8:0:1::200;
  }

  host server2.example.com {
    hardware ethernet 52:54:00:1b:f3:cf;
    fixed-address 2001:db8:0:1::ba3;
  }
}
```

   이 그룹 정의에서는 두 개의 호스트 항목을 그룹화합니다. dhcpd 서비스는 옵션 dhcp6.domain-search 매개 변수에 설정된 값을 그룹의 두 호스트에 적용합니다.

   b. 선택적으로 이러한 호스트에 고유한 그룹 선언에 추가 설정을 추가합니다.

2. dhcpd6 서비스를 다시 시작합니다.

```
# systemctl start dhcpd6
```

추가 리소스

- dhcp-options(5) man page
- /usr/share/doc/dhcp-server/dhcpd.conf.example 파일
- /usr/share/doc/dhcp-server/dhcpd6.conf.example file

3.12. 손상된 리스 데이터베이스 복원

DHCP 서버가 Corrupt lease file - 가능한 데이터 손실 과 같은 리스 데이터베이스와 관련된 오류를 기

록하는 경우, 생성된 **dhcpd** 서비스 사본에서 리스 데이터베이스를 복원할 수 있습니다. 이 복사본은 데이터베이스의 최신 상태를 반영하지 않을 수 있습니다.

주의

백업으로 교체하지 않고 리스 데이터베이스를 제거하면 현재 할당된 리스에 대한 모든 정보가 손실됩니다. 결과적으로 **DHCP** 서버는 이전에 다른 호스트에 할당되었으며 아직 만료되지 않은 클라이언트에 리스를 할당할 수 있었습니다. 이로 인해 **IP** 충돌이 발생합니다.

DHCPv4, DHCPv6 또는 두 데이터베이스를 모두 복원할지 여부에 따라 다음 절차를 참조하십시오.

- **DHCPv4** 리스 데이터베이스 복원
- **DHCPv6** 리스 데이터베이스 복원

사전 요구 사항

- **root** 사용자로 로그인합니다.
- 리스 데이터베이스가 손상되었습니다.

절차

- **DHCPv4** 리스 데이터베이스 복원:

1. **dhcpd** 서비스를 중지합니다.

```
# systemctl stop dhcpd
```

2. 손상된 리스 데이터베이스의 이름을 변경합니다.

-

```
# mv /var/lib/dhcpd/dhcpd.leases /var/lib/dhcpd/dhcpd.leases.corrupt
```

3. **lease** 데이터베이스를 새로 고칠 때 **dhcp** 서비스에서 생성한 리스 데이터베이스의 복사본을 복원합니다.

```
# cp -p /var/lib/dhcpd/dhcpd.leases~ /var/lib/dhcpd/dhcpd.leases
```

중요

최근 리스 데이터베이스 백업이 있는 경우 이 백업을 대신 복원합니다.

4. **dhcpd** 서비스를 시작합니다.

```
# systemctl start dhcpd
```

- **DHCPv6** 리스 데이터베이스 복원:

1. **dhcpd6** 서비스를 중지합니다.

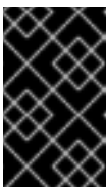
```
# systemctl stop dhcpd6
```

2. 손상된 리스 데이터베이스의 이름을 변경합니다.

```
# mv /var/lib/dhcpd/dhcpd6.leases /var/lib/dhcpd/dhcpd6.leases.corrupt
```

3. **lease** 데이터베이스를 새로 고칠 때 **dhcp** 서비스에서 생성한 리스 데이터베이스의 복사본을 복원합니다.

```
# cp -p /var/lib/dhcpd/dhcpd6.leases~ /var/lib/dhcpd/dhcpd6.leases
```



중요

최근 리스 데이터베이스 백업이 있는 경우 이 백업을 대신 복원합니다.

4. dhcpd6 서비스를 시작합니다.

```
# systemctl start dhcpd6
```

추가 리소스

- dhcpd 서비스의 lease 데이터베이스

3.13. DHCP 릴레이 에이전트 설정

DHCP Relay Agent(dhcrelay)를 사용하면 DHCP 서버가 없는 서브넷에서 다른 서브넷의 DHCP 서버로 DHCP 및 BOOTP 요청을 릴레이할 수 있습니다. DHCP 클라이언트에서 정보를 요청하면 DHCP 릴레이 에이전트는 지정된 DHCP 서버 목록으로 요청을 전달합니다. DHCP 서버에서 응답을 반환하면 DHCP 릴레이 에이전트는 이 요청을 클라이언트에 전달합니다.

IPv4, IPv6 또는 두 프로토콜 모두에 대한 DHCP 릴레이를 설정할지 여부에 따라 다음 절차를 참조하십시오.

- IPv4 네트워크
- IPv6 네트워크

사전 요구 사항

- root 사용자로 로그인합니다.

절차

- IPv4 네트워크의 경우:

1. dhcp-relay 패키지를 설치합니다.

```
# dnf install dhcp-relay
```

2. /lib/systemd/system/dhcrelay.service 파일을 /etc/systemd/system/ 디렉터리에 복

사합니다.

```
# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/
```

/usr/lib/systemd/system/dhcrelay.service 파일을 편집하지 마십시오. **dhcp-relay** 패키지의 향후 업데이트로 인해 변경 사항을 재정의할 수 있습니다.

3. **/etc/systemd/system/dhcrelay.service** 파일을 편집하고 서브넷을 담당하는 **DHCPv4** 서버의 **IP** 주소 목록과 함께 **-i** *인터페이스* 매개 변수를 추가합니다.

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid -i enp1s0 192.0.2.1
```

이러한 추가 매개 변수를 사용하여 **dhcrelay** 는 **enp1s0** 인터페이스에서 **DHCPv4** 요청을 수신 대기하고 **IP 192.0.2.1** 을 사용하여 **DHCP** 서버로 전달합니다.

4. **systemd** 관리자 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

5. 선택적으로 시스템이 부팅될 때 **dhcrelay** 서비스가 시작되도록 구성합니다.

```
# systemctl enable dhcrelay.service
```

6. **dhcrelay** 서비스를 시작합니다.

```
# systemctl start dhcrelay.service
```

- **IPv6** 네트워크의 경우:

1. **dhcp-relay** 패키지를 설치합니다.

```
# dnf install dhcp-relay
```

2. **/lib/systemd/system/dhcrelay.service** 파일을 **/etc/systemd/system/** 디렉터리에 복사하고 파일 이름을 **dhcrelay6.service** 로 복사합니다.

```
# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/dhcrelay6.service
```

/usr/lib/systemd/system/dhcrelay.service 파일을 편집하지 마십시오. **dhcp-relay** 패키지의 향후 업데이트로 인해 변경 사항을 재정의할 수 있습니다.

3. **/etc/systemd/system/dhcrelay6.service** 파일을 편집하고 **-l *receiving_interface*** 및 **-u *outgoing_interface*** 매개변수를 추가합니다.

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid -l enp1s0 -u enp7s0
```

이러한 추가 매개 변수를 사용하여 **dhcrelay** 는 **enp1s0** 인터페이스에서 **DHCPv6** 요청을 수신 대기하고 **enp7s0** 인터페이스에 연결된 네트워크로 전달합니다.

4. **systemd** 관리자 구성을 다시 로드합니다.

```
# systemctl daemon-reload
```

5. 선택적으로 시스템이 부팅될 때 **dhcrelay6** 서비스가 시작되도록 구성합니다.

```
# systemctl enable dhcrelay6.service
```

6. **dhcrelay6** 서비스를 시작합니다.

```
# systemctl start dhcrelay6.service
```

추가 리소스

- **dhcrelay(8)** 도움말 페이지