Red Hat Enterprise Linux 9

ネットワークインフラストラクチャーサービスの管理

Red Hat Enterprise Linux 9 でネットワークインフラストラクチャーサービスを管理するためのガイド

Last Updated: 2024-05-02

Red Hat Enterprise Linux 9 ネットワークインフラストラクチャーサービスの管理

Red Hat Enterprise Linux 9 でネットワークインフラストラクチャーサービスを管理するためのガイド

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

このドキュメントでは、Red Hat Enterprise Linux 9 で DNS や DHCP などのネットワークコアインフラストラクチャーサービスを設定および管理する方法について説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 BIND DNS サーバーのセットアップおよび設定

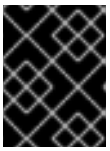
BIND は、Internet Engineering Task Force (IETF) の DNS 標準およびドラフト標準に完全に準拠した機能豊富な DNS サーバーです。たとえば、管理者は BIND を次のように頻繁に使用します。

- ローカルネットワークでの DNS サーバーのキャッシング
- ゾーンの権威 DNS サーバー
- ゾーンに高可用性を提供するセカンダリーサーバー

1.1. SELINUX を使用した BIND の保護または CHANGE-ROOT 環境での BIND の実行に関する考慮事項

BIND インストールを保護するには、次のことができます。

- change-root 環境なしで **named** サービスを実行します。この場合、**enforcing** モードの SELinux は、既知の BIND セキュリティー脆弱性の悪用を防ぎます。デフォルトでは、Red Hat Enterprise Linux は SELinux を **enforcing** モードで使用します。

  重要

  RHEL で SELinux を **enforcing** モードで使用して BIND を実行すると、change-root 環境で BIND を実行するよりも安全です。

- **name-chroot** サービスを change-root 環境で実行します。
  管理者は、change-root 機能を使用して、プロセスとそのサブプロセスのルートディレクトリーが / ディレクトリーとは異なるものであることを定義できます。**named-chroot** サービスを開始すると、BIND はそのルートディレクトリーを **/var/named/chroot/** に切り替えます。その結果、サービスは **mount --bind** コマンドを使用して、**/etc/named-chroot.files** にリストされているファイルおよびディレクトリーを **/var/named/chroot/** で使用できるようにし、プロセスは **/var/named/chroot/** 以外のファイルにアクセスできません。

BIND を使用する場合:

- 通常モードでは、**named** サービスを使用します。
- change-root 環境では、**named-chroot** サービスを使用します。これには、**named-chroot** パッケージを追加でインストールする必要があります。

1.2. BIND をキャッシュ DNS サーバーとして設定する

デフォルトでは、BIND DNS サーバーは、成功したルックアップと失敗したルックアップを解決してキャッシュします。その後、サービスはキャッシュから同じレコードへの要求に応答します。これにより、DNS ルックアップの速度が大幅に向上します。

前提条件

- サーバーの IP アドレスは静的です。

手順

1. **bind** パッケージおよび **bind-utils** パッケージをインストールします。
   ■

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

2. BIND を change-root 環境で実行する場合は、**bind-chroot** パッケージをインストールします。

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

デフォルトである **enforcing** モードで SELinux を使用するホストで BIND を実行すると、より安全になることに注意してください。

3. **/etc/named.conf** ファイルを編集し、**options** ステートメントに次の変更を加えます。

a. **listen-on** および **listen-on-v6** ステートメントを更新して、BIND がリッスンする IPv4 インターフェイスおよび IPv6 インターフェイスを指定します。

```
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** ステートメントを更新して、クライアントがこの DNS サーバーにクエリーを実行できる IP アドレスおよび範囲を設定します。

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

c. **allow-recursion** ステートメントを追加して、BIND が再帰クエリーを受け入れる IP アドレスおよび範囲を定義します。

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

警告

サーバーのパブリック IP アドレスで再帰を許可しないでください。そうしないと、サーバーが大規模な DNS 増幅攻撃の一部になる可能性があります。

d. デフォルトでは、BIND は、ルートサーバーから権限のある DNS サーバーに再帰的にクエリーを実行することにより、クエリーを解決します。または、プロバイダーのサーバーなど、他の DNS サーバーにクエリーを転送するように BIND を設定することもできます。この場合、BIND がクエリーを転送する DNS サーバーの IP アドレスのリストを含む **forwarders** ステートメントを追加します。

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

フォールバック動作として、フォワーダーサーバーが応答しないと、BIND はクエリーを再帰的に解決します。この動作を無効にするには、**forward only;** ステートメントを追加します。

4. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

5. 着信 DNS トラフィックを許可するように **firewalld** ルールを更新します。

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

6. BIND を開始して有効にします。

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

change-root 環境で BIND を実行する場合は、**systemctl enable --now named-chroot** コマンドを使用して、サービスを有効にして開始します。

検証

1. 新しく設定した DNS サーバーを使用してドメインを解決します。

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

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

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

初めてレコードをクエリーした後、BIND はエントリーをそのキャッシュに追加します。

2. 前のクエリーを繰り返します。

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

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

エントリーがキャッシュされるため、エントリーの有効期限が切れるまで、同じレコードに対するそれ以降のリクエストは大幅に高速化されます。

次のステップ

- この DNS サーバーを使用するようにネットワーク内のクライアントを設定します。DHCP サーバーが DNS サーバー設定をクライアントに提供する場合は、それに応じて DHCP サーバーの設定を更新します。

関連情報

- SELinux を使用した BIND の保護または change-root 環境での BIND の実行に関する考慮事項
- **named.conf(5)** man ページ
- **/usr/share/doc/bind/sample/etc/named.conf**

1.3. BIND DNS サーバーでのログの設定

bind パッケージによって提供されるデフォルトの **/etc/named.conf** ファイル内の設定は、**default_debug** チャネルを使用し、メッセージのログを **/var/named/data/named.run** ファイルに記録します。**default_debug** チャネルは、サーバーのデバッグレベルがゼロ以外の場合にのみエントリーをログに記録します。

さまざまなチャネルおよびカテゴリーを使用して、BIND を設定して、定義された重大度でさまざまなイベントを個別のファイルに書き込むことができます。

前提条件

- BIND は、たとえばキャッシングネームサーバーとしてすでに設定されています。
- **named** または **named-chroot** サービスが実行しています。

手順

1. **/etc/named.conf** ファイルを編集し、**category** および **channel** フレーズを **logging** ステートメントに追加します。次に例を示します。

```
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;
    };

    ...
};
```

この設定例では、BIND はゾーン転送に関連するメッセージのログを **/var/named/log/transfer.log** に記録します。BIND は最大 **10** バージョンのログファイルを作成し、最大サイズが **50** MB に達するとローテーションします。

category 句は、BIND がカテゴリーのメッセージを送信するチャネルを定義します。

channel 句は、バージョン数、最大ファイルサイズ、および BIND がチャネルにログ記録する必要がある重大度レベルを含むログメッセージの宛先を定義します。イベントのタイムスタンプ、カテゴリー、および重大度のログ記録を有効にするなどの追加設定はオプションですが、デバッグ目的で役立ちます。

2. ログディレクトリーが存在しない場合は作成し、このディレクトリーの **named** ユーザーに書き込み権限を付与します。

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

3. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

4. BIND を再起動します。

```
# systemctl restart named
```

change-root 環境で BIND を実行する場合は、**systemctl restart named-chroot** コマンドを使用してサービスを再起動します。

検証

- ログファイルの内容を表示します。

```
# 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
```

関連情報

- **named.conf(5)** man ページ

1.4. BIND ACL の作成

BIND の特定の機能へのアクセスを制御することで、サービス拒否 (DoS) などの不正アクセスや攻撃を防ぐことができます。BIND アクセス制御リスト (**acl**) ステートメントは、IP アドレスと範囲のリストです。各 ACL には、指定された IP アドレスと範囲を参照するために **allow-query** などのいくつかのステートメントで使用できるニックネームがあります。

警告

BIND は、ACL で最初に一致したエントリーのみを使用します。たとえば、ACL **{ 192.0.2/24; !192.0.2.1; }** を定義し、IP アドレス **192.0.2.1** でホストが接続すると、2 番目のエントリーでこのアドレスが除外されていてもアクセスが許可されます。

BIND には次の組み込み ACL があります。

- **none**: どのホストとも一致しません。
- **any**: すべてのホストに一致します。

- **localhost**: ループバックアドレス **127.0.0.1** と **::1**、および BIND を実行するサーバー上のすべてのインターフェイスの IP アドレスに一致します。
- **localnets** : ループバックアドレス **127.0.0.1** と **::1**、および BIND を実行するサーバーが直接接続しているすべてのサブネットに一致します。

前提条件

- BIND は、たとえばキャッシングネームサーバーとしてすでに設定されています。
- **named** または **named-chroot** サービスが実行しています。

手順

1. **/etc/named.conf** ファイルを編集して、次の変更を行います。
   a. **acl** ステートメントをファイルに追加します。たとえば、**127.0.0.1**、**192.0.2.0/24**、および **2001:db8:1::/64** に対して **internal-networks** という名前の ACL を作成するには、次のように入力します。

   ```
   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. ACL のニックネームをサポートするステートメントで使用します。たとえば、次のようになります。

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

2. **/etc/named.conf** ファイルの構文を確認します。

   ```
   # named-checkconf
   ```

   コマンドが出力を表示しない場合は、構文に間違いがありません。

3. BIND をリロードします。

   ```
   # systemctl reload named
   ```

   change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

検証

- 設定された ACL を使用する機能をトリガーするアクションを実行します。たとえば、この手順の ACL は、定義された IP アドレスからの再帰クエリーのみを許可します。この場合は、ACL の定義に含まれていないホストで次のコマンドを入力して、外部ドメインの解決を試みます。

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

  コマンドが出力を返さないと、BIND はアクセスを拒否し、ACL は機能しています。クライアントで詳細な出力を得るには、**+short** オプションを指定せずにコマンドを使用します。

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

```
...
;; WARNING: recursion requested but not available
...
```

1.5. BIND DNS サーバーでのゾーンの設定

DNS ゾーンは、ドメインスペース内の特定のサブツリーのリソースレコードを含むデータベースです。たとえば、**example.com** ドメインを担当している場合は、BIND でそのゾーンを設定できます。その結果、クライアントは **www.example.com** をこのゾーンで設定された IP アドレスに解決できます。

1.5.1. ゾーンファイルの SOA レコード

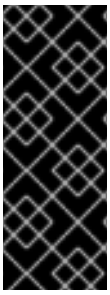
SOA (Start of Authority) レコードは、DNS ゾーンで必要なレコードです。このレコードは、たとえば、複数の DNS サーバーがゾーンだけでなく DNS リゾルバーに対しても権限を持っている場合に重要です。

BIND の SOA レコードの構文は次のとおりです。

```
name class type mname rname serial refresh retry expire minimum
```

読みやすくするために、管理者は通常、ゾーンファイル内のレコードを、セミコロン (**;**) で始まるコメントを使用して複数の行に分割します。SOA レコードを分割する場合は、括弧でレコードをまとめることに注意してください。

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



重要

完全修飾ドメイン名 (FQDN) の末尾にあるドットに注意してください。FQDN は、ドットで区切られた複数のドメインラベルで設定されます。DNS ルートには空のラベルがあるため、FQDN はドットで終わります。したがって、BIND はゾーン名を末尾のドットなしで名前に追加します。末尾にドットがないホスト名 (例: **ns1.example.com**) は、**ns1.example.com.example.com.** に展開されます。これは、プライマリーネームサーバーの正しいアドレスではありません。

SOA レコードのフィールドは次のとおりです。

- **name**: ゾーンの名前 (つまり **origin**)。このフィールドを **@** に設定すると、BIND はそれを **/etc/named.conf** で定義されたゾーン名に展開します。
- **class**: SOA レコードでは、このフィールドを常に Internet (**IN**) に設定する必要があります。
- **type**: SOA レコードでは、このフィールドを常に **SOA** に設定する必要があります。
- **mname** (マスター名): このゾーンのプライマリーネームサーバーのホスト名。
- **rname** (責任者名): このゾーンの責任者の電子メールアドレス。形式が異なりますのでご注意ください。アットマーク (**@**) をドット (**.**) に置き換える必要があります。

- **serial**: このゾーンファイルのバージョン番号。セカンダリーネームサーバーは、プライマリーサーバーのシリアル番号の方が大きい場合にのみ、ゾーンのコピーを更新します。
  形式は任意の数値にすることができます。一般的に使用される形式は **<year><month><day><two-digit-number>** です。この形式を使用すると、理論的には、ゾーンファイルを 1 日に 100 回まで変更できます。

- **refresh**: ゾーンが更新された場合は、プライマリーサーバーをチェックする前にセカンダリーサーバーが待機する時間。

- **retry**: 試行が失敗した後、セカンダリーサーバーがプライマリーサーバーへのクエリーを再試行するまでの時間。

- **expire**: 以前の試行がすべて失敗した場合に、セカンダリーサーバーがプライマリーサーバーへのクエリーを停止した後の時間。

- **minimum**: RFC 2308 は、このフィールドの意味を負のキャッシュ時間に変更しました。準拠リゾルバーは、**NXDOMAIN** 名エラーをキャッシュする期間を決定するために使用します。

注記

refresh、**retry**、**expire**、および **maximum** フィールドの数値は、時間を秒単位で定義します。ただし、読みやすくするために、時間の接尾辞 (**m** は分、**h** は時間、**d** は日など) を使用してください。たとえば、**3h** は 3 時間を表します。

関連情報

- RFC 1035: ドメイン名 - 実装および仕様
- RFC 1034: ドメイン名 - 概念および機能
- RFC 2308: DNS クエリーのネガティブキャッシュ (DNS キャッシュ)

1.5.2. BIND プライマリーサーバーでの正引きゾーンの設定

正引きゾーンは、名前を IP アドレスやその他の情報にマップします。たとえば、ドメイン **example.com** を担当している場合は、BIND で転送ゾーンを設定して、**www.example.com** などの名前を解決できます。

前提条件

- BIND は、たとえばキャッシングネームサーバーとしてすでに設定されています。
- **named** または **named-chroot** サービスが実行しています。

手順

1. **/etc/named.conf** ファイルにゾーン定義を追加します。

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

これらの設定により、次が定義されます。

- このサーバーは、**example.com** ゾーンのプライマリーサーバー (**type master**) です。
- **/var/named/example.com.zone** ファイルはゾーンファイルです。この例のように相対パスを設定すると、このパスは、**options** ステートメントの **directory** に設定したディレクトリーからの相対パスになります。
- どのホストもこのゾーンにクエリーを実行できます。または、IP 範囲または BIND アクセス制御リスト (ACL) のニックネームを指定して、アクセスを制限します。
- どのホストもゾーンを転送できません。ゾーン転送を許可するのは、セカンダリーサーバーをセットアップする際に限られ、セカンダリーサーバーの IP アドレスに対してのみ許可します。

2. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

3. たとえば、次の内容で **/var/named/example.com.zone** ファイルを作成します。

```
$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
```

このゾーンファイル:

- リソースレコードのデフォルトの有効期限 (TTL) 値を 8 時間に設定します。時間の **h** などの時間接尾辞がない場合、BIND は値を秒として解釈します。
- ゾーンに関する詳細を含む、必要な SOA リソースレコードが含まれています。
- このゾーンの権威 DNS サーバーとして **ns1.example.com** を設定します。ゾーンを機能させるには、少なくとも 1 つのネームサーバー (**NS**) レコードが必要です。ただし、RFC 1912 に準拠するには、少なくとも 2 つのネームサーバーが必要です。
- **example.com** ドメインのメールエクスチェンジャー (**MX**) として **mail.example.com** を設定します。ホスト名の前の数値は、レコードの優先度です。値が小さいエントリーほど優先度が高くなります。

- **www.example.com**、**mail.example.com**、および **ns1.example.com** の IPv4 アドレスおよび IPv6 アドレスを設定します。

4. **named** グループだけがそれを読み取ることができるように、ゾーンファイルに安全なアクセス許可を設定します。

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

5. **/var/named/example.com.zone** ファイルの構文を確認します。

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

6. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

検証

- **example.com** ゾーンからさまざまなレコードをクエリーし、出力がゾーンファイルで設定したレコードと一致することを確認します。

```
# 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
```

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

関連情報

- ゾーンファイルの SOA レコード
- BIND ACL の作成
- RFC 1912 - 一般的な DNS 操作および設定エラー

1.5.3. BIND プライマリーサーバーでの逆引きゾーンの設定

逆ゾーンは、IP アドレスを名前にマップします。たとえば、IP 範囲 **192.0.2.0/24** を担当している場合は、BIND で逆引きゾーンを設定して、この範囲の IP アドレスをホスト名に解決できます。

注記

クラスフルネットワーク全体の逆引きゾーンを作成する場合は、それに応じてゾーンに名前を付けます。たとえば、クラス C ネットワーク **192.0.2.0/24** の場合は、ゾーンの名前が **2.0.192.in-addr.arpa** です。**192.0.2.0/28** など、異なるネットワークサイズの逆引きゾーンを作成する場合は、ゾーンの名前が **28-2.0.192.in-addr.arpa** です。

前提条件

- BIND は、たとえばキャッシングネームサーバーとしてすでに設定されています。
- **named** または **named-chroot** サービスが実行しています。

手順

1. **/etc/named.conf** ファイルにゾーン定義を追加します。

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

これらの設定により、次が定義されます。

- **2.0.192.in-addr.arpa** 逆引きゾーンのプライマリーサーバー (**type master**) としてのこのサーバー。
- **/var/named/2.0.192.in-addr.arpa.zone** ファイルはゾーンファイルです。この例のように相対パスを設定すると、このパスは、**options** ステートメントの **directory** に設定したディレクトリーからの相対パスになります。
- どのホストもこのゾーンにクエリーを実行できます。または、IP 範囲または BIND アクセス制御リスト (ACL) のニックネームを指定して、アクセスを制限します。
- どのホストもゾーンを転送できません。ゾーン転送を許可するのは、セカンダリーサーバーをセットアップする際に限られ、セカンダリーサーバーの IP アドレスに対してのみ許可します。

2. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

3. たとえば、次の内容で **/var/named/2.0.192.in-addr.arpa.zone** ファイルを作成します。

```
$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.
```

このゾーンファイル:

- リソースレコードのデフォルトの有効期限 (TTL) 値を 8 時間に設定します。時間の **h** などの時間接尾辞がない場合、BIND は値を秒として解釈します。
- ゾーンに関する詳細を含む、必要な SOA リソースレコードが含まれています。
- **ns1.example.com** をこの逆引きゾーンの権威 DNS サーバーとして設定します。ゾーンを機能させるには、少なくとも 1 つのネームサーバー (**NS**) レコードが必要です。ただし、RFC 1912 に準拠するには、少なくとも 2 つのネームサーバーが必要です。
- **192.0.2.1** および **192.0.2.30** アドレスのポインター (**PTR**) レコードを設定します。

4. **named** グループのみがそれを読み取ることができるように、ゾーンファイルに安全なアクセス許可を設定します。

```
# 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** ファイルの構文を確認します。

```
# 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. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

検証

- 逆引きゾーンからさまざまなレコードをクエリーし、出力がゾーンファイルで設定したレコードと一致することを確認します。

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

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

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

関連情報

- ゾーンファイルの SOA レコード

- BIND ACL の作成
- RFC 1912 - 一般的な DNS 操作および設定エラー

1.5.4. BIND ゾーンファイルの更新

サーバーの IP アドレスが変更された場合など、特定の状況では、ゾーンファイルを更新する必要があります。複数の DNS サーバーが 1 つのゾーンを担ってる場合は、この手順をプライマリーサーバーでのみ実行してください。ゾーンのコピーを保存する他の DNS サーバーは、ゾーン転送を通じて更新を受け取ります。

前提条件

- ゾーンが設定されました。
- **named** または **named-chroot** サービスが実行しています。

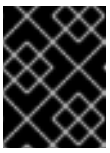
手順

1. オプション: **/etc/named.conf** ファイル内のゾーンファイルへのパスを特定します。

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

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

ゾーンの定義の **file** ステートメントで、ゾーンファイルへのパスを見つけます。相対パスは、**options** ステートメントの **directory** に設定されたディレクトリーからの相対パスです。

2. ゾーンファイルを編集します。
   a. 必要な変更を行います。
   b. SOA (Start of Authority) レコードのシリアル番号を増やします。



重要

シリアル番号が以前の値以下の場合、セカンダリーサーバーはゾーンのコピーを更新しません。

3. ゾーンファイルの構文を確認します。

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

4. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

検証

- 追加、変更、または削除したレコードを照会します。たとえば、次のようになります。

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

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

関連情報

- ゾーンファイルの SOA レコード
- BIND プライマリーサーバーでの正引きゾーンの設定
- BIND プライマリーサーバーでの逆引きゾーンの設定

1.5.5. 自動鍵生成およびゾーン保守機能を使用した DNSSEC ゾーン署名

DNSSEC (Domain Name System Security Extensions) でゾーンに署名して、認証およびデータの整合性を確保できます。このようなゾーンには、追加のリソースレコードが含まれます。クライアントはそれらを使用して、ゾーン情報の信頼性を検証できます。

ゾーンの DNSSEC ポリシー機能を有効にすると、BIND は次のアクションを自動的に実行します。

- キーを作成します。
- ゾーンに署名します
- 鍵の再署名や定期的な交換など、ゾーンを維持します。

重要

外部 DNS サーバーがゾーンの信頼性を検証できるようにするには、ゾーンの公開キーを親ゾーンに追加する必要があります。これを行う方法の詳細については、ドメインプロバイダーまたはレジストリーにお問い合わせください。

この手順では、BIND に組み込まれている **default** の DNSSEC ポリシーを使用します。このポリシーは、単一の **ECDSAP256SHA** 鍵署名を使用します。または、独自のポリシーを作成して、カスタムキー、アルゴリズム、およびタイミングを使用します。

前提条件

- DNSSEC を有効にするゾーンが設定されている。
- **named** または **named-chroot** サービスが実行しています。
- サーバーは時刻をタイムサーバーと同期します。DNSSEC 検証では、正確なシステム時刻が重要です。

手順

1. **/etc/named.conf** ファイルを編集し、DNSSEC を有効にするゾーンに **dnssec-policy default;** を追加します。

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

2. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

3. BIND は公開鍵を **/var/named/K<zone_name>.+<algorithm>+<key_ID>.key** ファイルに保存します。このファイルを使用して、親ゾーンが必要とする形式でゾーンの公開鍵を表示します。

   - DS レコード形式:

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

   - DNSKEY 形式:

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

4. ゾーンの公開鍵を親ゾーンに追加するように要求します。これを行う方法の詳細については、ドメインプロバイダーまたはレジストリーにお問い合わせください。

検証

1. DNSSEC 署名を有効にしたゾーンのレコードについて、独自の DNS サーバーにクエリーを実行します。

```
# dig +dnssec +short @localhost A www.example.com
192.0.2.30
A 13 3 28800 20220718081258 20220705120353 61141 example.com.
e7Cfh6GuOBMAWsgsHSVTPh+JJSOI/Y6zctzluqIU1JqEgOOAfL/Qz474
M0sgi54m1Kmnr2ANBKJN9uvOs5eXYw==
```

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

2. 公開鍵が親ゾーンに追加され、他のサーバーに伝播されたら、サーバーが署名付きゾーンへのクエリーで認証済みデータ (**ad**) フラグを設定することを確認します。

```
#  dig @localhost example.com +dnssec
```

```
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
...
```

関連情報

- BIND プライマリーサーバーでの正引きゾーンの設定
- BIND プライマリーサーバーでの逆引きゾーンの設定

1.6. BIND DNS サーバー間のゾーン転送の設定

ゾーン転送により、ゾーンのコピーを持つすべての DNS サーバーが最新のデータを使用することが保証されます。

前提条件

- 将来のプライマリーサーバーでは、ゾーン転送を設定するゾーンが設定されている。
- 将来のセカンダリーサーバーでは、BIND はキャッシュネームサーバーなどとして設定されている。
- 両方のサーバーで、**named** サービスまたは **named-chroot** サービスが実行している。

手順

1. 既存のプライマリーサーバーで、以下を行います。
   a. 共有キーを作成し、**/etc/named.conf** ファイルに追加します。

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

   このコマンドは、**tsig-keygen** コマンドの出力を表示し、自動的に **/etc/named.conf** に追加します。

   後でセカンダリーサーバーでもコマンドの出力が必要になります。

   b. **/etc/named.conf** ファイルのゾーン定義を編集します。
      i. **allow-transfer** ステートメントで、サーバーがゾーンを転送するために **example-transfer-key** ステートメントで指定されたキーを提供する必要があることを定義します。

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

      または、**allow-transfer** ステートメントで BIND アクセス制御リスト (ACL) ニックネームを使用します。

ii. デフォルトでは、ゾーンが更新された後、BIND は、このゾーンにネームサーバー (**NS**) レコードを持つすべてのネームサーバーに通知します。セカンダリーサーバーの **NS** レコードをゾーンに追加する予定がない場合は、BIND がこのサーバーに通知するように設定できます。そのために、このセカンダリーサーバーの IP アドレスを含む **also-notify** ステートメントをゾーンに追加します。

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

c. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

d. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

2. 将来のセカンダリーサーバーで、以下を行います。

a. **/etc/named.conf** ファイルを次のように編集します。

i. プライマリーサーバーと同じキー定義を追加します。

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

ii. **/etc/named.conf** ファイルにゾーン定義を追加します。

```
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;
    };
};
```

これらの設定状態:

- このサーバーは、**example.com** ゾーンのセカンダリーサーバー (**type slave**) です。
- **/var/named/slaves/example.com.zone** ファイルはゾーンファイルです。この例のように相対パスを設定すると、このパスは、**options** ステートメントの **directory**

に設定したディレクトリーからの相対パスになります。このサーバーがセカンダリーであるゾーンファイルをプライマリーサーバーから分離するには、それを **/var/named/slaves/** ディレクトリーなどに保存します。

- どのホストもこのゾーンにクエリーを実行できます。または、IP 範囲または ACL ニックネームを指定して、アクセスを制限します。
- このサーバーからゾーンを転送できるホストはありません。
- このゾーンのプライマリーサーバーの IP アドレスは **192.0.2.1** および **2001:db8:1::2** です。または、ACL ニックネームを指定できます。このセカンダリーサーバーは、**example-transfer-key** という名前のキーを使用して、プライマリーサーバーに対する認証を行います。

b. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

c. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

3. オプション: プライマリーサーバーのゾーンファイルを変更し、新しいセカンダリーサーバーの **NS** レコードを追加します。

検証

セカンダリーサーバーで次の手順を実行します。

1. **named** サービスの **systemd** ジャーナルエントリーを表示します。

```
# 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)
```

change-root 環境で BIND を実行する場合は、**journalctl -u named-chroot** コマンドを使用してジャーナルエントリーを表示します。

2. BIND がゾーンファイルを作成したことを確認します。

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

デフォルトでは、セカンダリーサーバーはゾーンファイルを未修正のバイナリー形式で保存することに注意してください。

3. セカンダリーサーバーから転送されたゾーンのレコードをクエリーします。

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

この例では、この手順で設定したセカンダリーサーバーが IP アドレス **192.0.2.2** でリッスンしていると想定しています。

関連情報

- BIND プライマリーサーバーでの正引きゾーンの設定
- BIND プライマリーサーバーでの逆引きゾーンの設定
- BIND ACL の作成
- BIND ゾーンファイルの更新

1.7. DNS レコードを上書きするように BIND で応答ポリシーゾーンを設定する

管理者は、DNS のブロックとフィルタリングを使用して、DNS 応答を書き換えて、特定のドメインまたはホストへのアクセスをブロックできます。BIND では、応答ポリシーゾーン (RPZ) がこの機能を提供します。**NXDOMAIN** エラーを返す、クエリーに応答しないなど、ブロックされたエントリーに対してさまざまなアクションを設定できます。

環境内に複数の DNS サーバーがある場合は、この手順を使用してプライマリーサーバーで RPZ を設定し、後でゾーン転送を設定して、セカンダリーサーバーで RPZ を使用できるようにします。

前提条件

- BIND は、たとえばキャッシングネームサーバーとしてすでに設定されています。
- **named** または **named-chroot** サービスが実行しています。

手順

1. **/etc/named.conf** ファイルを編集し、次の変更を行います。
   a. **options** ステートメントに **response-policy** 定義を追加します。

```
options {
    ...

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

    ...
}
```

response-policy の **zone** ステートメントで RPZ のカスタム名を設定できます。ただし、次のステップのゾーン定義で同じ名前を使用する必要があります。

b. 前の手順で設定した RPZ の **zone** 定義を追加します。

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

これらの設定状態:

- このサーバーは、**rpz.local** という名前の RPZ のプライマリーサーバー (**type master**) です。
- **/var/named/rpz.local** ファイルはゾーンファイルです。この例のように相対パスを設定すると、このパスは、**options** ステートメントの **directory** に設定したディレクトリーからの相対パスになります。
- **allow-query** で定義されたホストは、この RPZ をクエリーできます。または、IP 範囲または BIND アクセス制御リスト (ACL) のニックネームを指定して、アクセスを制限します。
- どのホストもゾーンを転送できません。ゾーン転送を許可するのは、セカンダリーサーバーをセットアップする際に限られ、セカンダリーサーバーの IP アドレスに対してのみ許可します。

2. **/etc/named.conf** ファイルの構文を確認します。

```
# named-checkconf
```

コマンドが出力を表示しない場合は、構文に間違いがありません。

3. たとえば、次の内容で **/var/named/rpz.local** ファイルを作成します。

```
$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.
```

このゾーンファイル:

- リソースレコードのデフォルトの有効期限 (TTL) 値を 10 分に設定します。時間の **h** などの時間接尾辞がない場合、BIND は値を秒として解釈します。

- ゾーンに関する詳細を含む、必要な SOA (Start of Authority) リソースレコードが含まれます。
- このゾーンの権威 DNS サーバーとして **ns1.example.com** を設定します。ゾーンを機能させるには、少なくとも 1 つのネームサーバー (**NS**) レコードが必要です。ただし、RFC 1912 に準拠するには、少なくとも 2 つのネームサーバーが必要です。
- このドメイン内の **example.org** およびホストへのクエリーに対して **NXDOMAIN** エラーを返します。
- このドメイン内の **example.net** およびホストにクエリーを破棄します。

アクションおよび例の完全なリストは、IETF draft: DNS Response Policy Zones (RPZ) を参照してください。

4. **/var/named/rpz.local** ファイルの構文を確認します。

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

5. BIND をリロードします。

```
# systemctl reload named
```

change-root 環境で BIND を実行する場合は、**systemctl reload named-chroot** コマンドを使用してサービスをリロードします。

検証

1. **NXDOMAIN** エラーを返すように RPZ で設定されている **example.org** のホストを解決しようとします。

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

この例では、BIND が同じホストで実行し、**localhost** インターフェイスでクエリーに応答することを前提としています。

2. RPZ でクエリーを破棄するように設定されている **example.net** ドメイン内のホストの解決を試みます。

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

関連情報

- IETF draft: DNS Response Policy Zones (RPZ)

1.8. DNSTAP を使用して DNS クエリーを記録する

ネットワーク管理者は、ドメインネームシステム (DNS) の詳細を記録して、DNS トラフィックパターンの分析、DNS サーバーのパフォーマンスの監視、DNS 問題のトラブルシューティングを行うことができます。受信する名前クエリーの詳細を監視してログに記録する高度な方法が必要な場合は、**named** サービスから送信されたメッセージを記録する **dnstap** インターフェイスを使用します。DNS クエリーをキャプチャーおよび記録して、Web サイトまたは IP アドレスに関する情報を収集できます。

前提条件

- **bind-9.16.15-3** パッケージ以降のバージョンがインストールされている。

警告

BIND バージョンがすでにインストールされ、実行されている場合、新しいバージョンの **BIND** を追加すると、既存のバージョンが上書きされます。

手順

1. **/etc/named.conf** ファイルの **options** ブロックを編集して、**dnstap** とターゲットファイルを有効にします。

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

2. ログに記録する DNS トラフィックのタイプを指定するには、**/etc/named.conf** ファイルの **dnstap** ブロックに **dnstap** フィルターを追加します。次のフィルターを使用できます。
   - **auth**: 権威ゾーンの応答または回答。
   - **client**: 内部クライアントクエリーまたは回答。
   - **forwarder**: 転送クエリーまたは応答。
   - **resolver**: 反復的解決クエリーまたは応答。
   - **update**: 動的ゾーン更新要求。
   - **all**: 上記のオプションのいずれか。
   - **query** または **response**: **query** または **response** キーワードを指定しない場合、**dnstap** は両方を記録します。

注記

dnstap フィルターでは、**dnstap {}** ブロック内に **;** で区切った複数の定義を含めます。次の構文を使用してください。**dnstap { (all | auth | client | forwarder | resolver | update) [(query | response)]; ... };**

3. 記録されたパケットに対する **dnstap** ユーティリティーの動作をカスタマイズするには、次のように追加パラメーターを指定して **dnstap-output** オプションを変更します。

   - **size** (unlimited | <size>) - サイズが指定した制限に達したときに **dnstap** ファイルの自動ロールオーバーを有効にします。
   - **versions** (unlimited | <integer>) - 保持する自動ロールオーバーファイルの数を指定します。
   - **suffix** (increment | timestamp) - ロールアウトされるファイルの命名規則を選択します。デフォルトでは、増分は **.0** から始まります。あるいは、**timestamp** 値を設定して UNIX タイムスタンプを使用することもできます。
     以下の例では、**auth** 応答のみ、**client** クエリー、および動的 **updates** のクエリーと応答の両方を要求します。

     Example:

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

4. 変更を適用するために、**named** サービスを再起動します。

   ```
   # systemctl restart named.service
   ```

5. アクティブなログの定期的なロールアウトを設定します。
   次の例では、**cron** スケジューラーは、ユーザーが編集したスクリプトの内容を 1 日に 1 回実行します。**roll** オプションに値 **3** 指定し、**dnstap** が最大 3 つのバックアップログファイルを作成できるようにしています。この値 **3** は、**dnstap-output** 変数の **version** パラメーターをオーバーライドし、バックアップログファイルの数を 3 に制限します。また、バイナリーログファイルは別のディレクトリーに移動されて名前が変更されます。3 つのバックアップログファイルがすでに存在する場合でも、ファイルの接尾辞が **.2** に達することはありません。サイズ制限に基づくバイナリーログの自動ローリングで十分な場合は、このステップを省略できます。

   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** ユーティリティーを使用して、人間が判読できる形式でログを処理および分析します。
   次の例では、**dnstap-read** ユーティリティーは出力を **YAML** ファイル形式で出力します。

Example:

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

第2章 アンバウンド DNS サーバーのセットアップ

unbound DNS サーバーは、検証、再帰、およびキャッシング DNS リゾルバーです。さらに、**unbound** はセキュリティーに重点を置いており、たとえば、デフォルトで Domain Name System Security Extensions (DNSSEC) が有効になっています。

2.1. UNBOUND をキャッシング DNS サーバーとして設定する

デフォルトでは、**unbound** されている DNS サービスは、成功したルックアップと失敗したルックアップを解決してキャッシュします。その後、サービスはキャッシュから同じレコードへの要求に応答します。

手順

1. **unbound** されているパッケージをインストールします。

```
# dnf install unbound
```

2. **/etc/unbound/unbound.conf** ファイルを編集し、**server** 句で次の変更を行います。

   a. **interface** パラメーターを追加して、**unbound** されているサービスがクエリーをリッスンする IP アドレスを設定します。次に例を示します。

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

   これらの設定では、**unbound** は指定された IPv4 および IPv6 アドレスでのみリッスンします。

   インターフェイスを必要なものに制限することで、インターネットなどの承認されていないネットワークからのクライアントがこの DNS サーバーにクエリーを送信するのを防ぎます。

   b. **access-control** パラメーターを追加して、クライアントが DNS サービスを照会できるサブネットを設定します。次に例を示します。

```
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** されているサービスをリモートで管理するための秘密鍵と証明書を作成します。

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

   この手順を省略した場合、次の手順で設定を確認すると、不足しているファイルが報告されます。ただし、**unbound** されているサービスは、ファイルが見つからない場合に自動的に作成します。

4. 設定ファイルを確認します。

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

5. 着信 DNS トラフィックを許可するように firewalld ルールを更新します。

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

6. **unbound** されているサービスを有効にして開始します。

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

検証

1. **localhost** インターフェイスでリッスンしている **unbound** されている DNS サーバーにクエリーを実行して、ドメインを解決します。

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

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

初めてレコードをクエリーした後、**unbound** はエントリーをそのキャッシュに追加します。

2. 前のクエリーを繰り返します。

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

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

エントリーがキャッシュされるため、エントリーの有効期限が切れるまで、同じレコードに対するそれ以降のリクエストは大幅に高速化されます。

次のステップ

- この DNS サーバーを使用するようにネットワーク内のクライアントを設定します。たとえば、**nmcli** ユーティリティーを使用して、DNS サーバーの IP を NetworkManager 接続プロファイルに設定します。

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

関連情報

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

第3章 DHCP サービスの提供

DHCP (Dynamic Host Configuration Protocol) は、クライアントに IP 情報を自動的に割り当てるネットワークプロトコルです。**dhcpd** サービスを設定して、ネットワークに DHCP サーバーおよび DHCP リレーを提供できます。

3.1. 静的 IP アドレスと動的 IP アドレス設定の違い

静的な IP アドレス指定

静的 IP アドレスをデバイスに割り当てると、そのアドレスは手動で変更しない限り、時間が経過しても変わることはありません。必要に応じて静的 IP アドレスを使用します。

- DNS などのサーバーや認証サーバーのネットワークアドレスの整合性を確保する。
- 他のネットワークインフラストラクチャーから独立して動作する、帯域外管理デバイスを使用する。

動的な IP アドレス指定

動的 IP アドレスを使用するようにデバイスを設定すると、アドレスは時間の経過とともに変わる可能性があります。このため、ホストの再起動後に IP アドレスが異なる可能性があるため、通常は動的アドレスがネットワークに接続されるデバイスに使用されます。
動的 IP アドレスは、より柔軟で、設定と管理が簡単です。Dynamic Host Control Protocol (DHCP) は、ネットワーク設定をホストに動的に割り当てる従来の方法です。

注記

静的 IP アドレスまたは動的 IP アドレスをどのような場合に使用するかを定義する厳密な規則はありません。ユーザーのニーズ、設定、およびネットワーク環境によって異なります。

3.2. DHCP トランザクションフェーズ

DHCP は、Discovery、Offer、Request、Acknowledgement の 4 つのフェーズで機能します。DHCP はこのプロセスを使用してクライアントに IP アドレスを提供します。

Discovery

DHCP クライアントがメッセージを送信して、ネットワーク上にある DHCP サーバーを検出します。このメッセージは、ネットワークおよびデータリンク層でブロードキャストされます。

Offer

DHCP サーバーはクライアントからメッセージを受け取り、DHCP クライアントに IP アドレスを提供します。このメッセージはデータリンク層でのユニキャストですが、ネットワーク層でブロードキャストします。

Request

DHCP クライアントは、提供される IP アドレスを DHCP サーバーに要求します。このメッセージはデータリンク層でのユニキャストですが、ネットワーク層でブロードキャストします。

Acknowledgment

DHCP サーバーは、DHCP クライアントに確認応答を送信します。このメッセージはデータリンク層でのユニキャストですが、ネットワーク層でブロードキャストします。これは、DHCP DORA プロセスの最後のメッセージです。

3.3. DHCPV4 および DHCPV6 で DHCPD を使用する場合の相違点

dhcpd サービスは、1台のサーバーで DHCPv4 と DHCPv6 の両方を提供できます。ただし、各プロトコルに DHCP を提供するには、別に設定ファイルを使用する **dhcpd** のインスタンスがそれぞれ必要です。

DHCPv4

- 設定ファイル -**/etc/dhcp/dhcpd.conf**
- systemd サービス名 - **dhcpd**

DHCPv6

- 設定ファイル - **/etc/dhcp/dhcpd6.conf**
- systemd サービス名 - **dhcpd6**

3.4. DHCPD サービスのリースデータベース

DHCP リースは、**dhcpd** サービスがネットワークアドレスをクライアントに割り当てる期間です。**dhcpd** サービスは、DHCP リースを以下のデータベースに保存します。

- DHCPv4 の場合 - **/var/lib/dhcpd/dhcpd.leases**
- DHCPv6 の場合 - **/var/lib/dhcpd/dhcpd6.leases**

警告

データベースファイルを手動で更新すると、データベースが破損する可能性があります。

リースデータベースには、メディアアクセス制御 (MAC) アドレスに割り当てられた IP アドレス、リースが期限切れになる際のタイムスタンプなど、割り当てられたリースに関する情報が含まれます。リースデータベースのタイムスタンプはすべて、協定世界時 (UTC) であることに注意してください。

dhcpd サービスは、定期的にデータベースを再作成します。

1. サービスは、既存のファイルの名前を変更します。
   - **/var/lib/dhcpd/dhcpd.leases** から **/var/lib/dhcpd/dhcpd.leases~**
   - **/var/lib/dhcpd/dhcpd6.leases** から **/var/lib/dhcpd/dhcpd6.leases~**
2. このサービスは、新たに作成された **/var/lib/dhcpd/dhcpd.leases** ファイルおよび **/var/lib/dhcpd/dhcpd6.leases** ファイルに、既知のリースをすべて書き込みます。

関連情報

- **dhcpd.leases(5)** の man ページ

- 破損したリースデータベースの復元

3.5. DHCPV6 と RADVD の比較

IPv6 ネットワークでは、ルーター広告メッセージのみが IPv6 デフォルトゲートウェイに関する情報を提供します。これにより、デフォルトのゲートウェイ設定を必要とするサブネットで DHCPv6 を使用する場合は、ルーター通知デーモン (**radvd**) などのルーター広告サービスを追加で設定する必要があります。

radvd サービスは、ルーター通知パケットのフラグを使用して、DHCPv6 サーバーの可用性をアナウンスします。

次の表では、DHCPv6 と **radvd** の機能を比較しています。

	DHCPv6	radvd
デフォルトゲートウェイに関する情報を提供する。	いいえ	はい
プライバシーを保護するために、ランダムなアドレスを保証する。	はい	いいえ
その他のネットワーク設定オプションを送信する。	はい	いいえ
メディアアクセス制御 (MAC) アドレスを IPv6 アドレスにマッピングする。	はい	いいえ

3.6. IPV6 ルーター用に RADVD サービスの設定

ルーター広告デーモン (**radvd**) は、IPv6 のステートレス自動設定に必要なルーター広告メッセージを送信します。これにより、ユーザーがアドレス、設定、ルートを自動的に設定し、そこから提供された情報に基づいてデフォルトのルーターを選択できます。

注記

/64 接頭辞は、**radvd** でのみ設定できます。その他の接頭辞を使用するには、DHCPv6 を使用します。

前提条件

- **root** ユーザーとしてログインしている。

手順

1. **radvd** パッケージをインストールします。

```
# dnf install radvd
```

2. **/etc/radvd.conf** ファイルを編集し、以下の設定を追加します。

```
interface enp1s0
{
```

```
    AdvSendAdvert on;
    AdvManagedFlag on;
    AdvOtherConfigFlag on;

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

この設定により、**2001:db8:0:1::/64** サブネット用の **enp1s0** デバイスにルーター広告メッセージを送信するように **radvd** を設定します。**AdvManagedFlag on** 設定は、クライアントが、DHCP サーバーから IP アドレスを受け取る必要があることを定義し、**on** に設定した **AdvOtherConfigFlag** パラメーターは、DHCP サーバーからもアドレス以外の情報を取得する必要があることを定義します。

3. 必要に応じて、システムの起動時に **radvd** が自動的に起動するように設定します。

```
# systemctl enable radvd
```

4. **radvd** サービスを起動します。

```
# systemctl start radvd
```

5. 必要に応じて、ルーター広告パッケージのコンテンツと、**radvd** が送信する設定値を表示します。

```
# radvdump
```

関連情報

- **radvd.conf(5)** man ページ
- **/usr/share/doc/radvd/radvd.conf.example** ファイル
- Can I use a prefix length other than 64 bits in IPv6 Router Advertisements?

3.7. DHCP サーバーのネットワークインターフェイスの設定

デフォルトでは、**dhcpd** サービスプロセスは、サービスの設定ファイルで定義されているサブネットに IP アドレスのあるネットワークインターフェイスでのみ要求します。

たとえば、以下のシナリオでは、**dhcpd** は、**enp0s1** ネットワークインターフェイスでのみリッスンします。

- **/etc/dhcp/dhcpd.conf** ファイルには、192.0.2.0/24 ネットワークの **subnet** 定義しかない。
- **enp0s1** ネットワークインターフェイスが 192.0.2.0/24 サブネットに接続されている。
- **enp7s0** インターフェイスが別のサブネットに接続されます。

DHCP サーバーに同じネットワークに接続された複数のネットワークインターフェイスが含まれているにもかかわらず、サービスが特定のインターフェイスでのみリッスンする必要がある場合にのみ、この手順に従ってください。

IPv4、IPv6、またはその両方のプロトコルに DHCP を提供するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **root** ユーザーとしてログインしている。
- **dhcp-server** パッケージがインストールされている。

手順

- IPv4 ネットワークの場合:

1. **/usr/lib/systemd/system/dhcpd.service** ファイルを **/etc/systemd/system/** ディレクトリーにコピーします。

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

/usr/lib/systemd/system/dhcpd.service ファイルは編集しないでください。**dhcp-server** パッケージの今後の更新により、変更が上書きされます。

2. **/etc/systemd/system/dhcpd.service** ファイルを編集し、**dhcpd** が、**ExecStart** パラメーターのコマンドでリッスンする必要があるインターフェイスの名前を追加します。

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

この例では、**dhcpd** が **enp0s1** インターフェイスおよび **enp7s0** インターフェイスでのみリッスンするように設定します。

3. **systemd** マネージャー設定を再読み込みします。

```
# systemctl daemon-reload
```

4. **dhcpd** サービスを再起動します。

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

- IPv6 ネットワークの場合:

1. **/usr/lib/systemd/system/dhcpd6.service** ファイルを **/etc/systemd/system/** ディレクトリーにコピーします。

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

/usr/lib/systemd/system/dhcpd6.service ファイルは編集しないでください。**dhcp-server** パッケージの今後の更新により、変更が上書きされます。

2. **/etc/systemd/system/dhcpd6.service** ファイルを編集し、**dhcpd** は、**ExecStart** パラメーターのコマンドでリッスンする必要があるインターフェイスの名前を追加します。

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

この例では、**dhcpd** が **enp0s1** インターフェイスおよび **enp7s0** インターフェイスでのみリッスンするように設定します。

3. **systemd** マネージャー設定を再読み込みします。

```
# systemctl daemon-reload
```

4. **dhcpd6** サービスを再起動します。

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

3.8. DHCP サーバーに直接接続されたサブネット用の DHCP サービスの設定

DHCP サーバーが、DHCP 要求に応答するサブネットに直接接続されている場合は、以下の手順を使用します。サーバーのネットワークインターフェイスに、このサブネットの IP アドレスが割り当てられている場合は、以下のようになります。

IPv4、IPv6、またはその両方のプロトコルに DHCP を提供するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **root** ユーザーとしてログインしている。
- **dhcp-server** パッケージがインストールされている。

手順

- IPv4 ネットワークの場合:

1. **/etc/dhcp/dhcpd.conf** ファイルを編集します。

a. その他のディレクティブにこの設定がない場合は、**dhcpd** が使用するグローバルパラメーターをデフォルトとして追加します。

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

この例では、接続のデフォルトのドメイン名を **example.com** に設定し、デフォルトのリース時間を **86400** 秒 (1 日) に設定します。

b. **authoritative** ステートメントを新しい行に追加します。

```
authoritative;
```

重要

authoritative ステートメントがない場合、**dhcpd** サービスは、クライアントがプール外にあるアドレスを要求する場合に、**DHCPNAK** で **DHCPREQUEST** メッセージに応答することはありません。

c. サーバーのインターフェイスに直接接続された各 IPv4 サブネットに、**subnet** 宣言を追加します。

```
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;
}
```

この例では、192.0.2.0/24 ネットワークにサブネットの宣言を追加します。この設定では、DHCP サーバーは以下の設定を、このサブネットから DHCP 要求を送信するクライアントに割り当てます。

- **range** パラメーターで定義されている範囲からの空き IPv4 アドレス
- このサブネットの DNS サーバーの IP - **192.0.2.1**
- このサブネットのデフォルトゲートウェイ - **192.0.2.1**
- このサブネットのブロードキャストアドレス - **192.0.2.255**
- このサブネットのクライアントが IP を解放し、サーバーに新しい要求を送信する最大リース時間 - **172800** 秒 (2 日)

2. 必要に応じて、システムの起動時に **dhcpd** が自動的に起動するように設定します。

```
# systemctl enable dhcpd
```

3. **dhcpd** サービスを開始します。

```
# systemctl start dhcpd
```

- IPv6 ネットワークの場合:

1. **/etc/dhcp/dhcpd6.conf** ファイルを編集します。

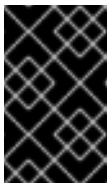
a. その他のディレクティブにこの設定がない場合は、**dhcpd** が使用するグローバルパラメーターをデフォルトとして追加します。

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

この例では、接続のデフォルトのドメイン名を **example.com** に設定し、デフォルトのリース時間を **86400** 秒 (1 日) に設定します。

b. **authoritative** ステートメントを新しい行に追加します。

```
authoritative;
```



重要

authoritative ステートメントがない場合、**dhcpd** サービスは、クライアントがプール外にあるアドレスを要求する場合に、**DHCPNAK** で **DHCPREQUEST** メッセージに応答することはありません。

c. サーバーのインターフェイスに直接接続された各 IPv6 サブネットに、**subnet** 宣言を追加します。

```
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;
}
```

以下の例では、 2001:db8:0:1::/64 ネットワークにサブネット宣言を追加します。この設定では、DHCP サーバーは以下の設定を、このサブネットから DHCP 要求を送信するクライアントに割り当てます。

- **range6** パラメーターで定義されている範囲外の空き IPv6 アドレスです。
- このサブネットの DNS サーバーの IP は **2001:db8:0:1::1** です。
- このサブネットのクライアントが IP を解放し、サーバーに新しい要求を送信する最大リース時間は、**172800** 秒 (2 日) です。
  IPv6 では、デフォルトのゲートウェイを特定するために、ルーター広告メッセージを使用する必要があることに注意してください。

2. 必要に応じて、システムの起動時に **dhcpd6** が自動的に開始するように設定します。

```
# systemctl enable dhcpd6
```

3. **dhcpd6** サービスを起動します。

```
# systemctl start dhcpd6
```

関連情報

- **dhcp-options(5)** の man ページ
- **dhcpd.conf(5)** の man ページ
- **/usr/share/doc/dhcp-server/dhcpd.conf.example** ファイル
- **/usr/share/doc/dhcp-server/dhcpd6.conf.example** ファイル

3.9. DHCP サーバーに直接接続していないサブネット用の DHCP サービスの設定

DHCP サーバーが、DHCP 要求に応答するサブネットに直接接続していない場合は、以下の手順に従います。これは、DHCP リレーエージェントが DHCP サーバーに要求を転送する場合を指します。DHCP サーバーのインターフェイスが、サーバーが提供するサブネットに直接接続していないためです。

IPv4、IPv6、またはその両方のプロトコルに DHCP を提供するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **root** ユーザーとしてログインしている。
- **dhcp-server** パッケージがインストールされている。

手順

- IPv4 ネットワークの場合:

1. **/etc/dhcp/dhcpd.conf** ファイルを編集します。

a. その他のディレクティブにこの設定がない場合は、**dhcpd** が使用するグローバルパラメーターをデフォルトとして追加します。

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

この例では、接続のデフォルトのドメイン名を **example.com** に設定し、デフォルトのリース時間を **86400** 秒 (1 日) に設定します。

b. **authoritative** ステートメントを新しい行に追加します。

```
authoritative;
```

重要

authoritative ステートメントがない場合、**dhcpd** サービスは、クライアントがプール外にあるアドレスを要求する場合に、**DHCPNAK** で **DHCPREQUEST** メッセージに応答することはありません。

c. 以下のように、サーバーのインターフェイスに直接接続していない IPv4 サブネットに、**shared-network** 宣言を追加します。

```
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 {
```

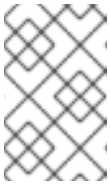
```
    range 198.51.100.20 198.51.100.100;
    option routers 198.51.100.1;
  }
  ...
}
```

この例では、192.0.2.0/24 と 198.51.100.0/24 の両方のネットワークの **subnet** 宣言を含む、共有ネットワーク宣言を追加します。この設定では、DHCP サーバーは以下の設定を、このサブネットのいずれかから DHCP 要求を送信するクライアントに割り当てます。

- 両方のサブネットにおけるクライアントの DNS サーバーの IP は **192.0.2.1** です。
- クライアントがどのサブネットから要求を送信したかに応じて、**range** パラメーターで定義された範囲の 空き IPv4 アドレスです。
- デフォルトゲートウェイは、クライアントがどのサブネットから要求を送信したかに応じて、**192.0.2.1** または **198.51.100.1** のいずれかになります。

d. サーバーが直接接続し、上記の **shared-network** で指定したリモートのサブネットに到達するのに使用されるサブネットの **subnet** 宣言を追加します。

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



注記

サーバーがこのサブネットに DHCP サービスを提供しない場合は、以下の例のように、**subnet** 宣言を空にする必要があります。直接接続したサブネットの宣言がないと、**dhcpd** は起動しません。

2. 必要に応じて、システムの起動時に **dhcpd** が自動的に起動するように設定します。

```
# systemctl enable dhcpd
```

3. **dhcpd** サービスを開始します。

```
# systemctl start dhcpd
```

- IPv6 ネットワークの場合:

1. **/etc/dhcp/dhcpd6.conf** ファイルを編集します。

a. その他のディレクティブにこの設定がない場合は、**dhcpd** が使用するグローバルパラメーターをデフォルトとして追加します。

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

この例では、接続のデフォルトのドメイン名を **example.com** に設定し、デフォルトのリース時間を **86400** 秒 (1 日) に設定します。

b. **authoritative** ステートメントを新しい行に追加します。

```
authoritative;
```

重要

authoritative ステートメントがない場合、**dhcpd** サービスは、クライアントがプール外にあるアドレスを要求する場合に、**DHCPNAK** で **DHCPREQUEST** メッセージに応答することはありません。

c. 以下のように、サーバーのインターフェイスに直接接続していない IPv6 サブネットに、**shared-network** 宣言を追加します。

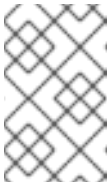
```
shared-network example {
  option domain-name-servers 2001:db8:0:1::1:1
  ...

  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
  }
  ...
}
```

この例では、2001:db8:0:1::1:0/120 と 2001:db8:0:1::2:0/120 の両方のネットワークの **subnet6** 宣言を含む共有ネットワーク宣言を追加します。この設定では、DHCP サーバーは以下の設定を、このサブネットのいずれかから DHCP 要求を送信するクライアントに割り当てます。

- 両方のサブネットからのクライアントに対する DNS サーバーの IP は **2001:db8:0:1::1:1** です。
- クライアントがどのサブネットから要求を送信したかに応じて、**range6** パラメーターで定義された範囲の 空き IPv6 アドレスです。
  IPv6 では、デフォルトのゲートウェイを特定するために、ルーター広告メッセージを使用する必要があることに注意してください。

d. サーバーが直接接続され、上記の **shared-network** で指定したリモートサブネットに到達するのに使用されるサブネットの **subnet6** 宣言を追加します。

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



注記

サーバーがこのサブネットに DHCP サービスを提供しない場合、**subnet6** 宣言は以下の例のように空にする必要があります。直接接続したサブネットの宣言がないと、**dhcpd** は起動しません。

2. 必要に応じて、システムの起動時に **dhcpd6** が自動的に開始するように設定します。

```
# systemctl enable dhcpd6
```

3. **dhcpd6** サービスを起動します。

```
# systemctl start dhcpd6
```

関連情報

- **dhcp-options(5)** の man ページ
- **dhcpd.conf(5)** の man ページ
- **/usr/share/doc/dhcp-server/dhcpd.conf.example** ファイル
- **/usr/share/doc/dhcp-server/dhcpd6.conf.example** ファイル
- DHCP リレーエージェントの設定

3.10. DHCP を使用してホストに静的アドレスの割り当て

host 宣言を使用して、DHCP サーバーを設定して、ホストのメディアアクセス制御 (MAC) アドレスに固定 IP アドレスを割り当てることができます。たとえば、この方法を使用して、常に同じ IP アドレスをサーバーまたはネットワークデバイスに割り当てます。

IPv4、IPv6、またはその両方のプロトコルに固定アドレスを設定するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **dhcpd** サービスを設定し、実行している。
- **root** ユーザーとしてログインしている。

手順

- IPv4 ネットワークの場合:

1. **/etc/dhcp/dhcpd.conf** ファイルを編集します。

a. **host** 宣言を追加します。

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

以下の例では、DHCP サーバーが、MAC アドレス **52:54:00:72:2f:6e** を使用して、常に IP アドレス **192.0.2.130** をホストに割り当てるように設定します。

dhcpd サービスは、**fixed-address** パラメーターで指定された MAC アドレスでシステムを識別しますが、**host** 宣言の名前とは異なります。これにより、この名前を、他の **host** 宣言に一致しない任意の文字列に設定できます。複数のネットワークに同じシス

テムを設定するには、別の名前を使用します。同じ名前を使用すると、**dhcpd** が起動に失敗します。

b. 必要に応じて、このホストに固有の **host** 宣言にその他の設定を追加します。

2. **dhcpd** サービスを再起動します。

```
# systemctl start dhcpd
```

- IPv6 ネットワークの場合:

1. **/etc/dhcp/dhcpd6.conf** ファイルを編集します。

a. **host** 宣言を追加します。

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

この例では、DHCP サーバーを設定して、IP アドレス **2001:db8:0:1::20** を常に MAC アドレス **52:54:00:72:2f:6e** を持つホストに割り当てます。

dhcpd サービスは、**host** 宣言の名前ではなく、**fixed-address6** パラメーターで指定された MAC アドレスでシステムを識別します。これにより、他の **host** 宣言に固有のものであれば、この名前を任意の文字列に設定できます。複数のネットワークに同じシステムを設定する場合は、同じ名前を使用すると **dhcpd** が起動に失敗するため、別の名前を使用します。

b. 必要に応じて、このホストに固有の **host** 宣言にその他の設定を追加します。

2. **dhcpd6** サービスを再起動します。

```
# systemctl start dhcpd6
```

関連情報

- **dhcp-options(5)** の man ページ
- **/usr/share/doc/dhcp-server/dhcpd.conf.example** ファイル
- **/usr/share/doc/dhcp-server/dhcpd6.conf.example** ファイル

3.11. GROUP 宣言を使用して、パラメーターを複数のホスト、サブネット、および共有ネットワークを同時に適用

group 宣言を使用すると、同じパラメーターを複数のホスト、サブネット、および共有ネットワークに適用できます。

この手順では、ホストの **group** 宣言を使用する方法を説明しますが、手順はサブネットと共有ネットワークの場合と同じです。

IPv4、IPv6、またはその両方プロトコルにグループを設定するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **dhcpd** サービスを設定し、実行している。
- **root** ユーザーとしてログインしている。

手順

- IPv4 ネットワークの場合:
  1. **/etc/dhcp/dhcpd.conf** ファイルを編集します。
     a. **group** 宣言を追加します。

```
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;
  }
}
```

     この **group** は、2 つの **host** エントリーを 1 つのグループにまとめます。**dhcpd** サービスは、**option domain-name-servers** パラメーターに設定した値を、グループ内の両方のホストに適用します。

     b. 必要に応じて、このホストに固有の **group** 宣言にその他の設定を追加します。

  2. **dhcpd** サービスを再起動します。

```
# systemctl start dhcpd
```

- IPv6 ネットワークの場合:
  1. **/etc/dhcp/dhcpd6.conf** ファイルを編集します。
     a. **group** 宣言を追加します。

```
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;
    }
}
```

この **group** は、2 つの **host** エントリーを 1 つのグループにまとめます。**dhcpd** サービスは、**option dhcp6.domain-search** パラメーターに設定された値をグループ内の両方のホストに適用します。

b. 必要に応じて、このホストに固有の **group** 宣言にその他の設定を追加します。

2. **dhcpd6** サービスを再起動します。

```
# systemctl start dhcpd6
```

関連情報

- **dhcp-options(5)** の man ページ
- **/usr/share/doc/dhcp-server/dhcpd.conf.example** ファイル
- **/usr/share/doc/dhcp-server/dhcpd6.conf.example** ファイル

3.12. 破損したリースデータベースの復元

DHCP サーバーが、リースデータベースに関連するエラー (**Corrupt lease file - possible data loss!** など) をログに記録すると、**dhcpd** サービスが作成するコピーからリースデータベースを復元できます。このコピーには、データベースの最新のステータスが反映されない場合があることに注意してください。

警告

リースデータベースをバックアップに置き換えるのではなく削除すると、現在割り当てられているリースに関する情報がすべて失われます。その結果、DHCP サーバーは、他のホストに割り当てられていて、まだ期限が切れていないクライアントにリースを割り当てることができます。これにより IP の競合が発生します。

DHCPv4、DHCPv6、またはその両方のデータベースを復元するかどうかに応じて、以下の手順を参照してください。

- DHCPv4 リースデータベースの復元
- DHCPv6 リースデータベースの復元

前提条件

- **root** ユーザーとしてログインしている。
- リースデータベースが破損している。

手順

- DHCPv4 リースデータベースの復元:

  1. **dhcpd** サービスを停止します。

     ```
     # systemctl stop dhcpd
     ```

  2. 破損したリースデータベースの名前を変更します。

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

  3. リースデータベースを更新する際に作成される **dhcp** サービスのリースデータベースのコピーを復元します。

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

     重要

     リースデータベースの最新のバックアップがある場合には、代わりにこのバックアップを復元します。

  4. **dhcpd** サービスを開始します。

     ```
     # systemctl start dhcpd
     ```

- DHCPv6 リースデータベースの復元:

  1. **dhcpd6** サービスを停止します。

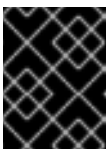
     ```
     # systemctl stop dhcpd6
     ```

  2. 破損したリースデータベースの名前を変更します。

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

  3. リースデータベースを更新する際に作成される **dhcp** サービスのリースデータベースのコピーを復元します。

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

     重要

     リースデータベースの最新のバックアップがある場合には、代わりにこのバックアップを復元します。

  4. **dhcpd6** サービスを起動します。

     ```
     # systemctl start dhcpd6
     ```

関連情報

- dhcpd サービスのリースデータベース

3.13. DHCP リレーエージェントの設定

DHCP リレーエージェント (**dhcrelay**) を使うと、DHCP サーバーがないサブネットから他のサブネットにある DHCP サーバーに DHCP および BOOTP リクエストのリレーができるようになります。DHCP クライアントが情報を要求すると、DHCP リレーエージェントは、指定した DHCP サーバーのリストに要求を転送します。DHCP サーバーが応答を返すと、DHCP リレーエージェントはこの要求をクライアントに転送します。

IPv4、IPv6、またはその両方のプロトコルに DHCP リレーを設定するかどうかに応じて、以下の手順を参照してください。

- IPv4 ネットワーク
- IPv6 ネットワーク

前提条件

- **root** ユーザーとしてログインしている。

手順

- IPv4 ネットワークの場合:

1. **dhcp-relay** パッケージをインストールします。

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

2. **/lib/systemd/system/dhcrelay.service** ファイルを **/etc/systemd/system/** ディレクトリーにコピーします。

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

/usr/lib/systemd/system/dhcrelay.service ファイルは編集しないでください。**dhcp-relay** パッケージの今後の更新により、変更がオーバーライドされます。

3. **/etc/systemd/system/dhcrelay.service** ファイルを編集し、**-i interface** パラメーターと、サブネットに対応する DHCPv4 サーバーの IP アドレスリストを追加します。

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

dhcrelay は、この追加パラメーターを使用して、**enp1s0** インターフェイスの DHCPv4 要求をリッスンし、IP **192.0.2.1** で DHCP サーバーに転送します。

4. **systemd** マネージャー設定を再読み込みします。

```
# systemctl daemon-reload
```

5. 必要に応じて、システムの起動時に **dhcrelay** サービスが開始するように設定します。

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

6. **dhcrelay** サービスを開始します。

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

- IPv6 ネットワークの場合:

1. **dhcp-relay** パッケージをインストールします。

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

2. **/lib/systemd/system/dhcrelay.service** ファイルを **/etc/systemd/system/** ディレクトリーにコピーして、**dhcrelay6.service** ファイルに名前を付けます。

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

/usr/lib/systemd/system/dhcrelay.service ファイルは編集しないでください。**dhcp-relay** パッケージの今後の更新により、変更がオーバーライドされます。

3. **/etc/systemd/system/dhcrelay6.service** ファイルを編集し、**-l receiving_interface** パラメーターおよび **-u outgoing_interface** パラメーターを追加します。

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

これらの追加パラメーターを使用すると、**dhcrelay** は **enp1s0** インターフェイスの DHCPv6 要求をリッスンし、**enp7s0** インターフェイスに接続されたネットワークに転送します。

4. **systemd** マネージャー設定を再読み込みします。

```
# systemctl daemon-reload
```

5. 必要に応じて、システムの起動時に **dhcrelay6** サービスが開始するように設定します。

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

6. **dhcrelay6** サービスを開始します。

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

関連情報

- **dhcrelay(8)** の man ページ