Red Hat Enterprise Linux 8

基本的なシステム設定

システムに必要な機能の設定とシステム環境のカスタマイズ

Last Updated: 2024-06-26

Red Hat Enterprise Linux 8 基本的なシステム設定

システムに必要な機能の設定とシステム環境のカスタマイズ

法律上の通知

概要

基本的なシステム管理タスク、環境設定、システムの登録、ネットワークアクセスとシステムセキュリティーの設定を行います。ユーザー、グループ、ファイルのパーミッションを管理します。複数の RHEL システム上のシステム設定インターフェイスを管理するには、システムロールを使用します。効率的なサービス管理には systemd を使用します。chrony を使用して Network Time Protocol (NTP) を設定します。ReaR を使用してシステムをバックアップおよび復元します。Python 3 や PHP などの動的プログラミング言語をインストールして使用します。

目次

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 基本的なネットワークアクセスの設定と管理

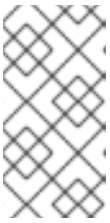
このセクションでは、Red Hat Enterprise Linux でネットワーク設定を設定する方法に関する基本的なオプションについてのみ説明します。

1.1. グラフィカルインストールモードでのネットワークおよびホスト名の設定

以下の手順に従って、ネットワークとホスト名を設定します。

手順

1. **インストール概要** 画面から、**ネットワークとホスト名** をクリックします。
2. 左側のペインのリストから、インターフェイスを選択します。詳細が右側のペインに表示されます。



注記

em1 や **wl3sp0** といった一貫性のある名前をネットワークデバイスの特定に使用するネットワークデバイス命名の標準仕様には、いくつかのタイプがあります。このような標準仕様の詳細は **ネットワークの設定および管理** を参照してください。

3. 選択したインタフェースを有効または無効にするには、**ON/OFF** スイッチを切り替えます。

注記

インストールプログラムは、ローカルでアクセス可能なインターフェイスを自動的に検出し、手動で追加または削除できません。

4. **+** をクリックして、仮想ネットワークインターフェイスを追加します。仮想ネットワークインターフェイスは、チーム、ボンド、ブリッジ、または VLAN のいずれかです。
5. **-** を選択して、仮想インターフェイスを削除します。
6. **設定** をクリックして、既存のインターフェイスの IP アドレス、DNS サーバー、またはルーティング設定 (仮想と物理の両方) などの設定を変更します。
7. **ホスト名** フィールドに、システムのホスト名を入力します。

注記

- ホスト名は、**hostname.domainname** 形式の完全修飾ドメイン名 (FQDN)、またはドメインなしの短縮ホスト名のいずれかにします。多くのネットワークには、自動的に接続したシステムにドメイン名を提供する DHCP (Dynamic Host Configuration Protocol) サービスがあります。DHCP サービスがこのシステムにドメイン名を割り当てるようにするには、短縮ホスト名のみを指定します。

- 静的 IP およびホスト名の設定を使用する場合、短縮名または FQDN を使用するかどうかは、計画したシステムのユースケースによって異なります。Red Hat Identity Management はプロビジョニング時に FQDN を設定しますが、サードパーティーのソフトウェア製品によっては短縮名が必要になる場合があります。いずれの場合も、すべての状況で両方のフォームの可用性を確保するには、**IP FQDN short-alias** の形式で **/etc/hosts** にホストのエントリーを追加します。

- **localhost** の値は、ターゲットシステムの静的ホスト名が指定されておらず、(たとえば、DHCP または DNS を使用する NetworkManager による) ネットワーク設定時に、インストールされるシステムの実際のホスト名が設定されることを示しています。

- ホスト名に使用できるのは、英数字と **-** または **.** のみです。ホスト名は 64 文字以下である必要があります。ホスト名は、**-** および **.** で開始したり終了したりできません。DNS に準拠するには、FQDN の各部分は 63 文字以下で、ドットを含む FQDN の合計の長さは 255 文字を超えることができません。

8. **Apply** をクリックして、ホスト名をインストーラー環境に適用します。

9. また、**ネットワークおよびホスト名** 画面では、ワイヤレスオプションを選択できます。右側のペインで **ネットワークの選択** をクリックして Wifi 接続を選択します。必要に応じてパスワードを入力し、**完了** をクリックします。

関連情報

- 高度な RHEL 8 インストールの実行

1.2. NMCLI を使用したイーサネット接続の設定

イーサネット経由でホストをネットワークに接続する場合は、**nmcli** ユーティリティーを使用してコマンドラインで接続の設定を管理できます。

前提条件

- 物理または仮想イーサネットネットワークインターフェイスコントローラー (NIC) がサーバーに設定されている。

手順

1. NetworkManager 接続プロファイルをリストします。

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

デフォルトでは、NetworkManager はホスト内の各 NIC のプロファイルを作成します。この NIC を特定のネットワークにのみ接続する予定がある場合は、自動作成されたプロファイルを調整してください。この NIC をさまざまな設定のネットワークに接続する予定がある場合は、ネットワークごとに個別のプロファイルを作成してください。

2. 追加の接続プロファイルを作成する場合は、次のように入力します。

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

既存のプロファイルを変更するには、この手順をスキップしてください。

3. オプション: 接続プロファイルの名前を変更します。

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

ホストに複数のプロファイルがある場合は、わかりやすい名前を付けると、プロファイルの目的を識別しやすくなります。

4. 接続プロファイルの現在の設定を表示します。

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

5. IPv4 を設定します。

- DHCP を使用するには、次のように入力します。

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

ipv4.method がすでに **auto** (デフォルト) に設定されている場合は、この手順をスキップしてください。

- 静的 IPv4 アドレス、ネットワークマスク、デフォルトゲートウェイ、DNS サーバー、および検索ドメインを設定するには、次のように入力します。

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

6. IPv6 設定を行います。

- ステートレスアドレス自動設定 (SLAAC) を使用するには、次のように入力します。

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

ipv6.method がすでに **auto** (デフォルト) に設定されている場合は、この手順をスキップしてください。

- 静的 IPv6 アドレス、ネットワークマスク、デフォルトゲートウェイ、DNS サーバー、および検索ドメインを設定するには、次のように入力します。

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

7. プロファイルの他の設定をカスタマイズするには、次のコマンドを使用します。

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

値はスペースまたはセミコロンで引用符で囲みます。

8. プロファイルをアクティブ化します。

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

検証

1. NIC の IP 設定を表示します。

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
```

2. IPv4 デフォルトゲートウェイを表示します。

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

3. IPv6 デフォルトゲートウェイを表示します。

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

4. DNS 設定を表示します。

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

複数の接続プロファイルが同時にアクティブな場合、**nameserver** エントリーの順序は、これらのプロファイルの DNS 優先度の値と接続タイプによって異なります。

5. **ping** ユーティリティーを使用して、このホストがパケットを他のホストに送信できることを確認します。

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

トラブルシューティング

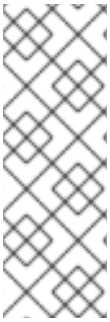
- ネットワークケーブルがホストとスイッチに差し込まれていることを確認します。
- リンク障害がこのホストだけに存在するか、同じスイッチに接続された他のホストにも存在するかを確認します。
- ネットワークケーブルとネットワークインターフェイスが予想どおりに機能していることを確認します。ハードウェア診断手順を実施して、不具合ケーブルとネットワークインターフェイスカードを置き換えます。
- ディスクの設定がデバイスの設定と一致しない場合は、NetworkManager を起動するか再起動して、インメモリー接続を作成することで、デバイスの設定を反映します。この問題を回避する方法および詳細は、NetworkManager サービスの再起動後に、NetworkManager が接続を複製する ソリューションを参照してください。

関連情報

- **nm-settings(5)** man ページ

1.3. NMTUI を使用したイーサネット接続の設定

イーサネット経由でホストをネットワークに接続する場合は、**nmtui** アプリケーションを使用して、テキストベースのユーザーインターフェイスで接続の設定を管理できます。**nmtui** では、グラフィカルインターフェイスを使用せずに、新しいプロファイルの作成や、ホスト上の既存のプロファイルの更新を行います。



注記

nmtui で 以下を行います。

- カーソルキーを使用してナビゲートします。
- ボタンを選択して **Enter** を押します。
- **Space** を使用してチェックボックスをオンまたはオフにします。

前提条件

- 物理または仮想イーサネットネットワークインターフェイスコントローラー (NIC) がサーバーに設定されている。

手順

1. 接続に使用するネットワークデバイス名がわからない場合は、使用可能なデバイスを表示します。

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

2. **nmtui** を開始します。

```
# nmtui
```

3. **Edit a connection** 選択し、**Enter** を押します。

4. 新しい接続プロファイルを追加するか、既存の接続プロファイルを変更するかを選択します。

   - 新しいプロファイルを作成するには、以下を実行します。

     i. **Add** を押します。

     ii. ネットワークタイプのリストから **Ethernet** を選択し、**Enter** を押します。

   - 既存のプロファイルを変更するには、リストからプロファイルを選択し、**Enter** を押します。

5. オプション: 接続プロファイルの名前を更新します。
   ホストに複数のプロファイルがある場合は、わかりやすい名前を付けると、プロファイルの目的を識別しやすくなります。

6. 新しい接続プロファイルを作成する場合は、ネットワークデバイス名を **connection** フィールドに入力します。

7. 環境に応じて、**IPv4 configuration** および **IPv6 configuration** 領域に IP アドレス設定を設定します。これを行うには、これらの領域の横にあるボタンを押して、次を選択します。

   - この接続に IP アドレスが必要ない場合は、**Disabled** にします。

   - DHCP サーバーが IP アドレスをこの NIC に動的に割り当てる場合は、**Automatic** にします。

   - ネットワークで静的 IP アドレス設定が必要な場合は、**Manual** にします。この場合、さらにフィールドに入力する必要があります。

     i. 設定するプロトコルの横にある **Show** を押して、追加のフィールドを表示します。

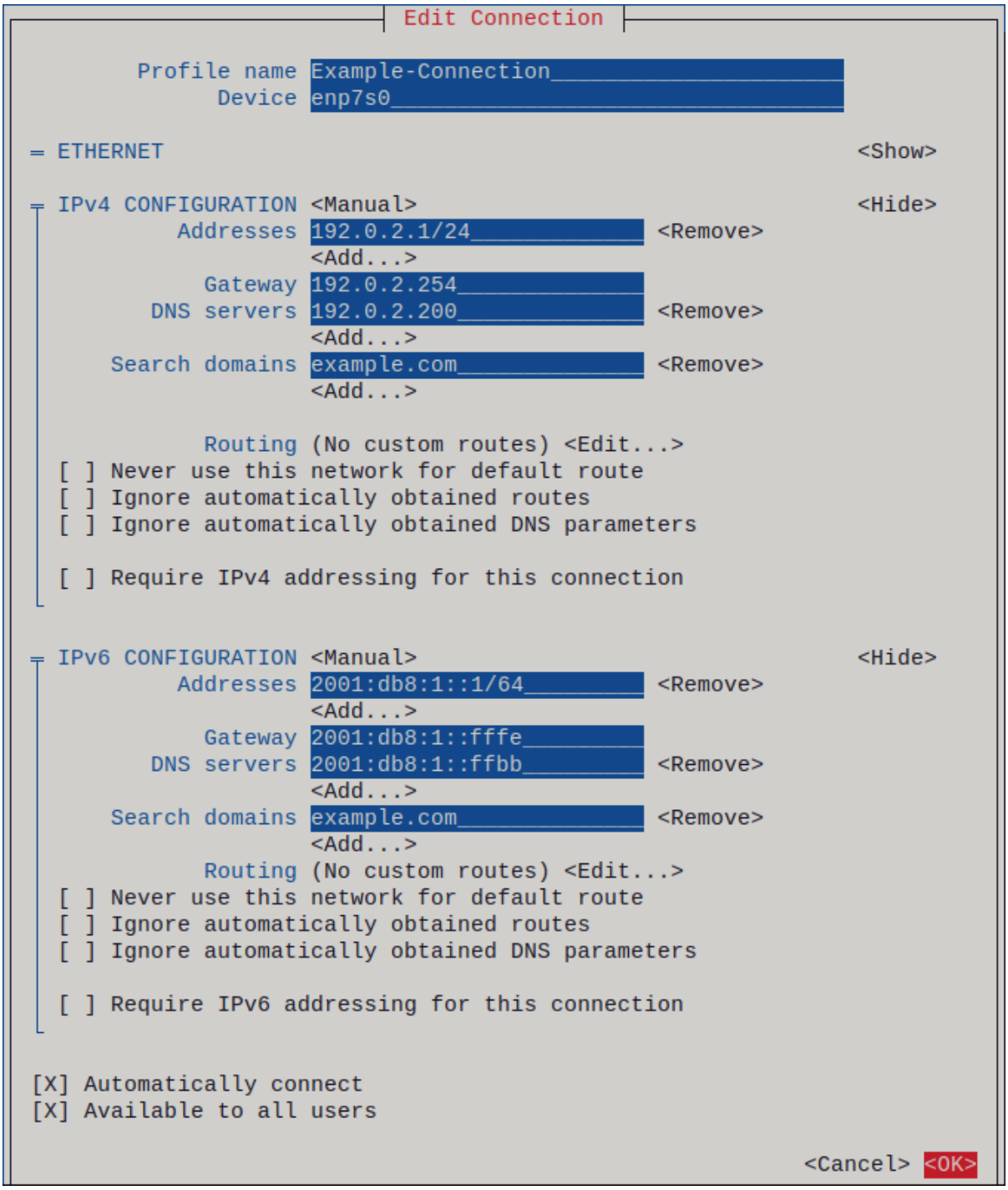
     ii. **Addresses** の横にある **Add** を押して、IP アドレスとサブネットマスクを Classless Inter-Domain Routing (CIDR) 形式で入力します。
     サブネットマスクを指定しない場合、NetworkManager は IPv4 アドレスに **/32** サブネットマスクを設定し、IPv6 アドレスに **/64** サブネットマスクを設定します。

     iii. デフォルトゲートウェイのアドレスを入力します。

     iv. **DNS servers** の横にある **Add** を押して、DNS サーバーのアドレスを入力します。

     v. **Search domains** の横にある **Add** を押して、DNS 検索ドメインを入力します。

図1.1 静的 IP アドレス設定によるイーサネット接続の例



8. **OK** を押すと、新しい接続が作成され、自動的にアクティブ化されます。
9. **Back** を押してメインメニューに戻ります。
10. **Quit** を選択し、**Enter** キーを押して **nmtui** アプリケーションを閉じます。

検証

1. NIC の IP 設定を表示します。

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
```

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

2. IPv4 デフォルトゲートウェイを表示します。

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

3. IPv6 デフォルトゲートウェイを表示します。

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

4. DNS 設定を表示します。

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

   複数の接続プロファイルが同時にアクティブな場合、**nameserver** エントリーの順序は、これらのプロファイルの DNS 優先度の値と接続タイプによって異なります。

5. **ping** ユーティリティーを使用して、このホストがパケットを他のホストに送信できることを確認します。

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

トラブルシューティング

- ネットワークケーブルがホストとスイッチに差し込まれていることを確認します。
- リンク障害がこのホストだけに存在するか、同じスイッチに接続された他のホストにも存在するかを確認します。
- ネットワークケーブルとネットワークインターフェイスが予想どおりに機能していることを確認します。ハードウェア診断手順を実施して、不具合ケーブルとネットワークインターフェイスカードを置き換えます。
- ディスクの設定がデバイスの設定と一致しない場合は、NetworkManager を起動するか再起動して、インメモリー接続を作成することで、デバイスの設定を反映します。この問題を回避する方法および詳細は、NetworkManager サービスの再起動後に、NetworkManager が接続を複製する ソリューションを参照してください。

関連情報

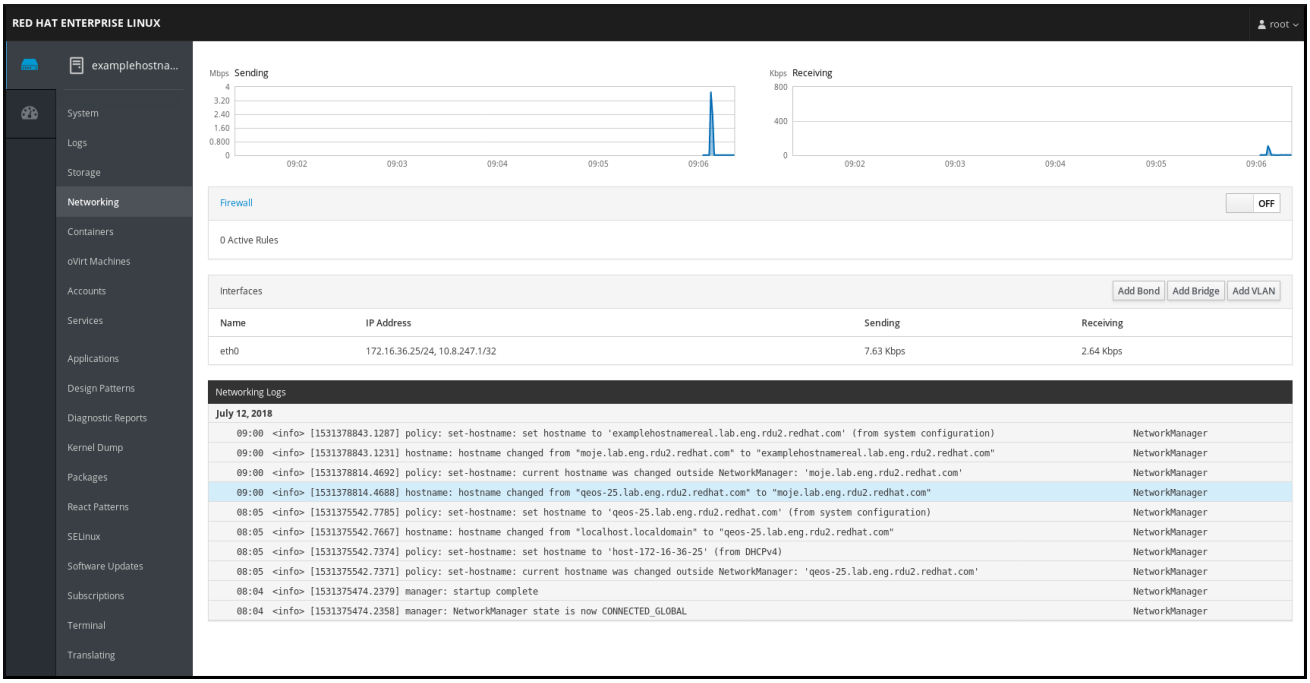
- 特定のプロファイルでのデフォルトゲートウェイの指定を防ぐための NetworkManager の設定
- DNS サーバーの順序の設定

1.4. RHEL WEB コンソールにおけるネットワークの管理

Web コンソールの **Networking** メニューでは、以下が可能です。

- 最近送受信したパケットの表示
- 利用可能なネットワークインターフェイスの最も重要な特徴の表示
- ネットワーキングログのコンテンツの表示
- ネットワークインターフェイスの様々なタイプ (ボンディング、チーム、ブリッジ、VLAN) の追加

図1.2 RHEL Web コンソールにおけるネットワークの管理



1.5. RHEL システムロールを使用したネットワークの管理

network ロールを使用して、複数のターゲットマシンにネットワーク接続を設定できます。

network ロールでは、以下のタイプのインターフェイスを設定できます。

- イーサネット
- ブリッジ
- ボンディング
- VLAN
- MacVLAN
- Infiniband

各ホストに必要なネットワーク接続は、**network_connections** 変数内にリストとして提供されます。

警告

network ロールは、**network_connections** 変数で指定されているとおりに、ターゲットシステムにあるすべての接続プロファイルを更新または作成します。したがって、そのオプションがそのシステムにのみ存在し、**network_connections** 変数にはない場合、**network** ロールは指定されたプロファイルからオプションを削除します。

以下の例は、必要なパラメーターを持つイーサネット接続が確実に設定されるように、**network** ロールを適用する方法を示しています。

必要なパラメーターでイーサネット接続を設定する network ロールを適用する Playbook の例

```
# SPDX-License-Identifier: BSD-3-Clause
---
- hosts: managed-node-01.example.com
  vars:
    network_connections:

      # Create one Ethernet profile and activate it.
      # The profile uses automatic IP addressing
      # and is tied to the interface by MAC address.
      - name: prod1
        state: up
        type: ethernet
        autoconnect: yes
        mac: "00:00:5e:00:53:00"
        mtu: 1450

  roles:
    - rhel-system-roles.network
```

関連情報

- RHEL システムロールを使用するためのコントロールノードと管理対象ノードの準備

1.6. 関連情報

- ネットワークの設定および管理

第2章 システム登録およびサブスクリプション管理

Red Hat Enterprise Linux オペレーティングシステムと、そこにインストールされている製品は、サブスクリプションの対象となります。

Red Hat コンテンツ配信ネットワーク (CDN) サブスクリプションを使用して、以下を追跡します。

- 登録したシステム
- システムにインストールされている製品
- インストール済みの製品に割り当てられているサブスクリプション

2.1. インストール後のシステムの登録

インストールプロセス中にシステムを登録していない場合は、以下の手順に従って登録します。

前提条件

- Red Hat カスタマーポータルに有効なユーザーアカウントがある。
- Red Hat アカウントの作成 ページを参照してください。
- RHEL システムに使用するアクティブなサブスクリプションがある。
- インストールプロセスの詳細は 標準的な RHEL 8 インストールの実行 を参照してください。

手順

1. ワンステップでシステムを登録し、自動的にサブスクライブします。

```
# subscription-manager register --username <username> --password <password> --auto-attach
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: 37to907c-ece6-49ea-9174-20b87ajk9ee7
The registered system name is: client1.idm.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status:       Subscribed
```

コマンドを実行すると、Red Hat カスタマーポータルのユーザー名とパスワードの入力を求めるプロンプトが表示されます。

登録プロセスに失敗した場合は、システムを特定のプールに登録できます。これを実行する方法については、以下の手順に従います。

a. 必要なサブスクリプションのプール ID を確認します。

```
# subscription-manager list --available
```

このコマンドは、使用している Red Hat アカウントで利用可能なサブスクリプションをすべて表示します。サブスクリプションごとに、プール ID を含むさまざまな情報が表示されます。

b. **pool_id** を、確認したプール ID に置き換えて、適切なサブスクリプションをシステムに割り当てます。

```
# subscription-manager attach --pool=pool_id
```

注記

Red Hat Insights にシステムを登録するには、**rhc connect** ユーティリティーを使用できます。リモートホスト設定のセットアップ を参照してください。

関連情報

- カスタマーポータル: 自動アタッチシステム
- カスタマーポータル: 登録および手動サブスクリプション

2.2. WEB コンソールで認証情報を使用してサブスクリプションを登録

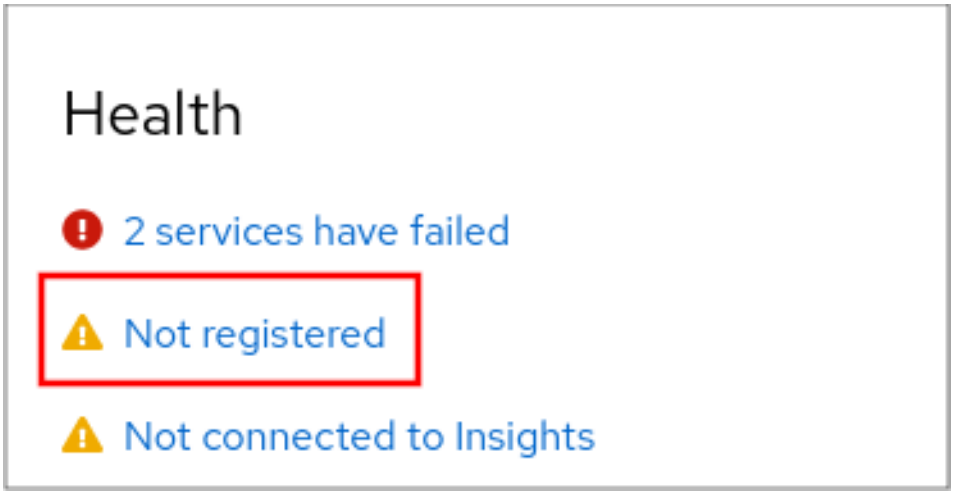
RHEL Web コンソールを使用して、新しくインストールされた Red Hat Enterprise Linux をアカウント認証情報で登録するには、次の手順を使用します。

前提条件

- Red Hat カスタマーポータルに有効なユーザーアカウントがある。
  Red Hat アカウントの作成 ページを参照してください。
- RHEL システムに使用するアクティブなサブスクリプションがある。

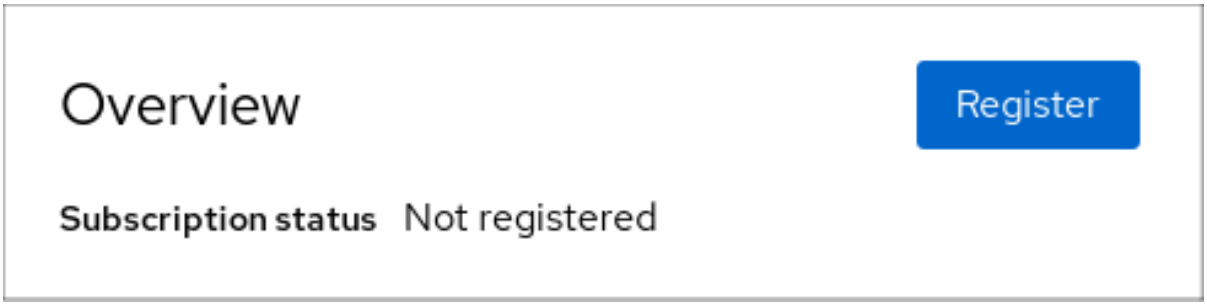
手順

1. RHEL Web コンソールにログインします。詳細は、Web コンソールへのログイン を参照してください。
2. **概要** ページの **ヘルス** ファイル内の **未登録** の警告をクリックするか、メインメニューの **サブスクリプション** をクリックして、サブスクリプション情報のあるページに移動します。

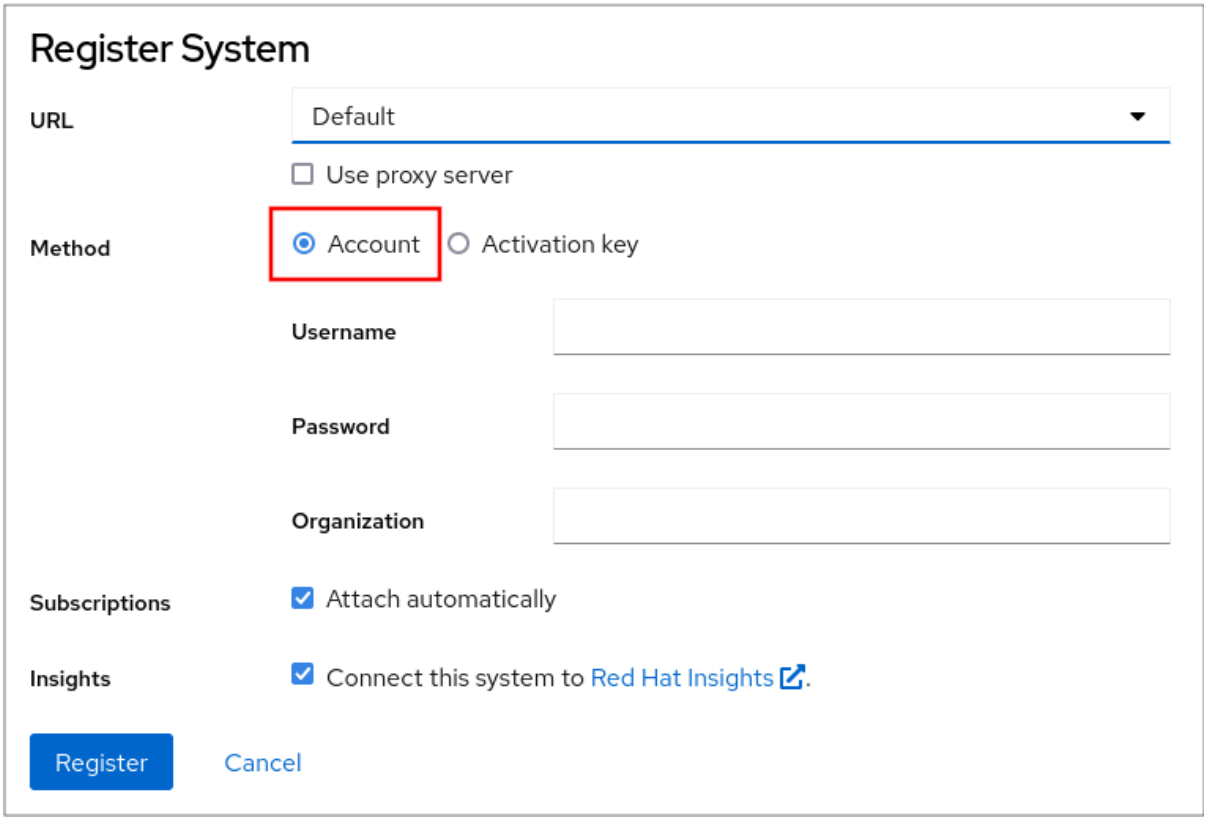


をクリックします。

3. **Overview** フィールドの **Register** をクリックします。



4. **システムの登録** ダイアログボックスで、アカウント情報での登録を選択します。



5. ユーザー名を入力します。
6. パスワードを入力します。
7. オプションで、組織名または ID を入力します。
   アカウントが Red Hat カスタマーポータルで複数の組織に所属している場合には、組織名または組織 ID を追加する必要があります。組織 ID は、Red Hat の連絡先に問い合わせてください。

   - Red Hat Insights にシステムを接続しない場合は、**Insights** チェックボックスのチェックを外してください。

8. **登録** ボタンをクリックします。

この時点で、Red Hat Enterprise Linux システムが正常に登録されました。

2.3. GNOME での RED HAT アカウントを使用したシステム登録

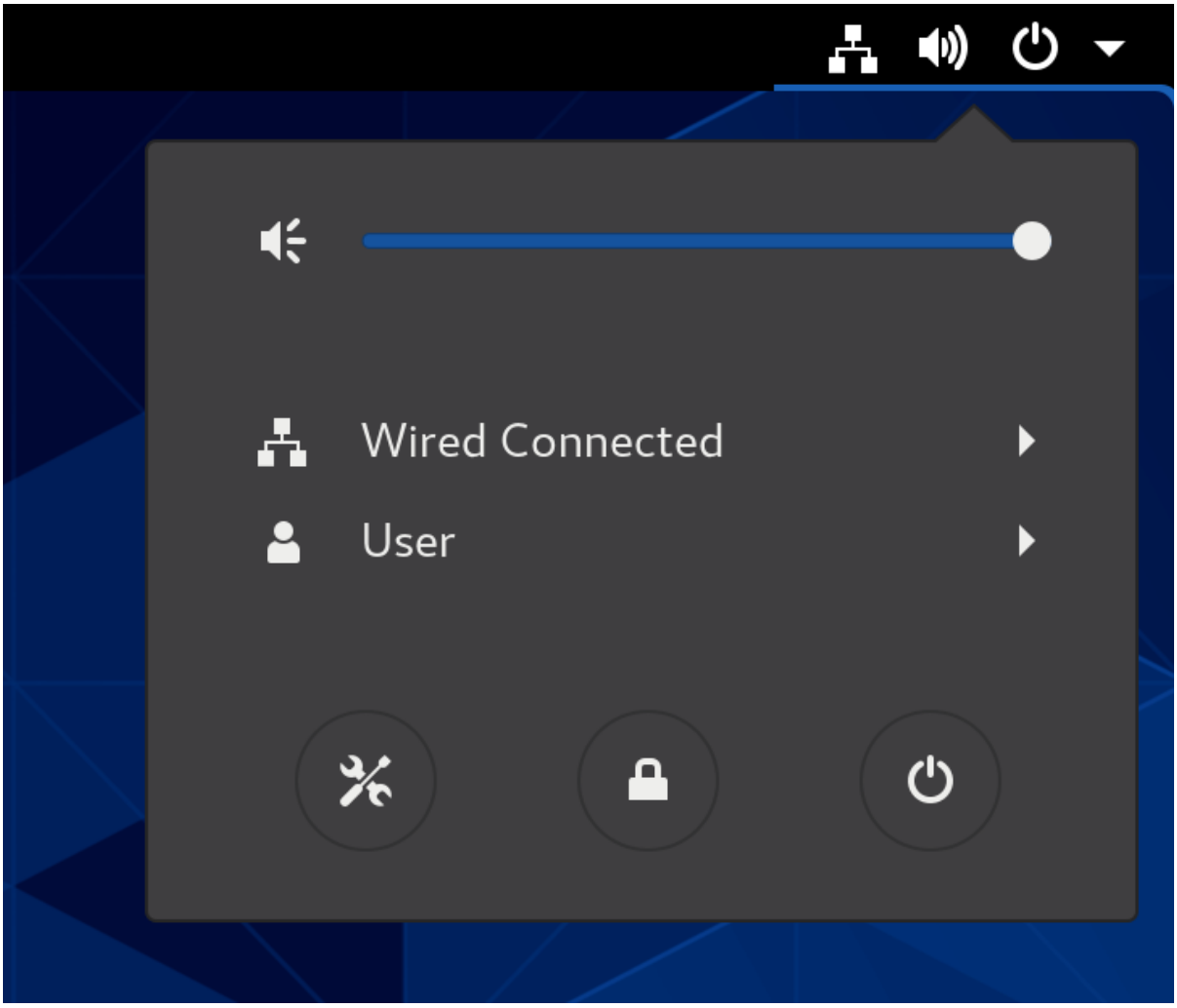
以下の手順に従って、システムを Red Hat アカウントに登録します。

前提条件

- Red Hat カスタマーポータルで有効なアカウント
  新規ユーザー登録は、Red Hat アカウントの作成 ページを参照してください。

手順

1. 画面の右上隅からアクセスできる **システムメニュー** を開き、**設定** アイコンをクリックします。



2. **Details → About** セクションで、**Register** をクリックします。
3. **Registration Server** を選択します。
4. Red Hat サーバーを使用しない場合は、**URL** フィールドにサーバーアドレスを入力します。
5. **Registration Type** メニューで、**Red Hat Account** を選択します。
6. **Registration Details** で以下を行います。
   - **Login** フィールドに、Red Hat アカウントのユーザー名を入力します。
   - **Password** フィールドに、Red Hat アカウントのパスワードを入力します。
   - **Organizaiton** フィールドに組織の名前を入力します。
7. **Register** をクリックします。

2.4. GNOME でのアクティベーションキーを使用したシステム登録

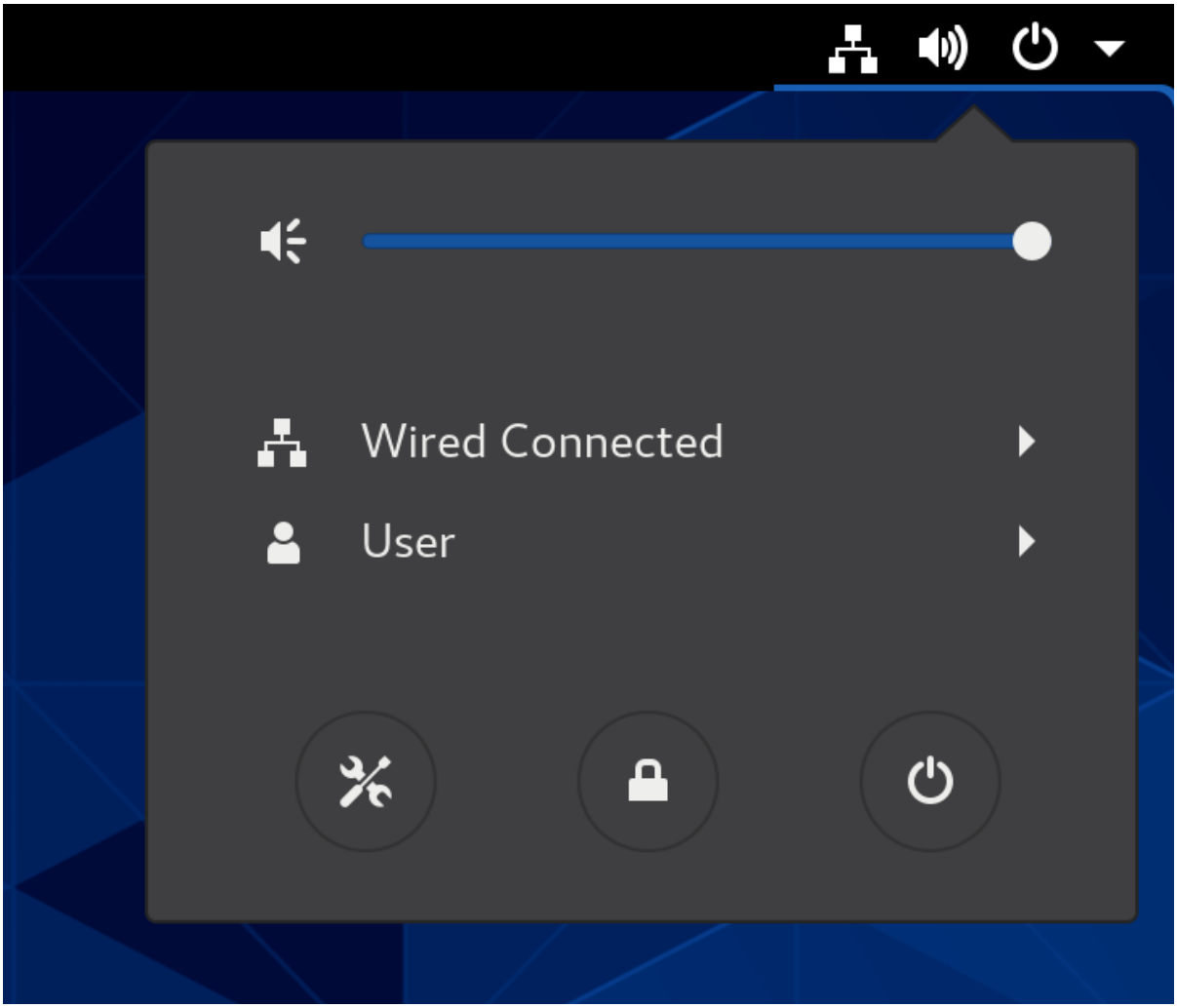
以下の手順に従って、システムをアクティベーションキーに登録します。組織の管理者からアクティベーションキーを取得できます。

前提条件

- アクティベーションキーまたはキー。
  新しい アクティベーションキー を作成するには、アクティベーションキーページを参照してください。

手順

1. 画面の右上隅からアクセスできる **システムメニュー** を開き、**設定** アイコンをクリックします。



2. **Details → About** セクションで、**Register** をクリックします。
3. **Registration Server** を選択します。
4. Red Hat サーバーを使用しない場合は、**URL** フィールドにサーバーアドレスを入力します。
5. **Registration Type** メニューで、**Activation keys** を選択します。
6. **Registration Details** で以下を行います。
   - **アクティベーションキーフィールドにアクティベーション** キーを入力します。
     キーをコンマ (,) で区切ります。

- **組織** フィールドに組織の名前または ID を入力します。

7. **Register** をクリックします。

2.5. インストーラー GUI を使用した RHEL 8 の登録

RHEL インストーラー GUI を使用して Red Hat Enterprise Linux 8 を登録するには、次の手順を実行します。

前提条件

- Red Hat カスタマーポータルに有効なユーザーアカウントがある。Create a Red Hat Login ページを参照してください。
- 有効なアクティベーションキーと組織 ID を持っている。

手順

1. **Installation Summary** 画面の **Software** で、**Connect to Red Hat** をクリックします。
2. **Account** または **Activation Key** オプションを使用して、Red Hat アカウントを認証します。
3. オプション: **Set System Purpose** フィールドで、設定する **Role**、**SLA**、および **Usage** 属性をドロップダウンメニューから選択します。
   この時点で、Red Hat Enterprise Linux 8 システムが正常に登録されました。

第3章 RED HAT サポートへのアクセス

本セクションでは、Red Hat サポートおよび **sosreport** を使用して問題を効果的にトラブルシューティングする方法を説明します。

Red Hat サポートを利用する場合は、Red Hat カスタマーポータル にアクセスしてください。カスタマーポータルでは、サブスクリプションで利用可能なものをすべて提供します。

3.1. RED HAT カスタマーポータルで利用できる RED HAT サポート

以下のセクションでは、Red Hat カスタマーポータルを使用してサポートを受ける方法を説明します。

前提条件

- Red Hat カスタマーポータルに有効なユーザーアカウントがある。Red Hat ログインの作成 を参照してください。
- RHEL システムに使用するアクティブなサブスクリプションがある。

手順

1. Red Hat サポート にアクセスします。
   a. サポートケースを作成する
   b. Red Hat 専門スタッフとのライブチャットを開始する
   c. 電話または電子メールで Red Hat 専門スタッフに問い合わせる

3.2. SOSREPORT を使用した問題のトラブルシューティング

sosreport コマンドは設定の詳細、システム情報、および診断情報を Red Hat Enterprise Linux システムから収集します。

次のセクションでは、**sosreport** コマンドを使用して、サポートケースのレポートを作成する方法を説明します。

前提条件

- Red Hat カスタマーポータルに有効なユーザーアカウントがある。Red Hat ログインの作成 を参照してください。
- RHEL システムに使用するアクティブなサブスクリプションがある。
- サポートケース番号。

手順

1. **sos** パッケージをインストールするには、以下のコマンドを実行します。

```
# yum install sos
```

注記

Red Hat Enterprise Linux のデフォルトの最小インストールには、**sosreport** コマンドを提供する **sos** パッケージは含まれません。

2. レポートを生成します。

```
# sosreport
```

3. サポートケースにレポートを添付します。
   How can I attach a file to a Red Hat support case? を参照してください。詳細は、Red Hat ナレッジベースの記事を参照してください。

   レポートを添付すると、該当のサポートケース番号の入力が求められます。

関連情報

- What is an sosreport and how to create one in Red Hat Enterprise Linux?

第4章 基本的な環境設定の変更

基本的な環境設定は、インストールプロセスの一部です。以下のセクションでは、後での変更する時に説明します。環境の基本設定には、以下が含まれます。

- 日付と時刻
- システムロケール
- キーボードのレイアウト
- 言語

4.1. 日付および時刻の設定

正確な時間管理は、いくつかの理由で重要です。Red Hat Enterprise Linux では、**NTP** プロトコルにより、時間管理が保証されます。これは、デーモンにより、ユーザー領域に実装されます。ユーザー領域のデーモンは、カーネルで実行しているシステムクロックを更新します。システムクロックは、さまざまなクロックソースを使用して時間を維持します。

Red Hat Enterprise Linux 8 は、**chronyd** デーモンを使用して **NTP** を実装します。**chronyd** は **chrony** パッケージから利用できます。詳細は、Chrony スイートを使用した NTP の設定 を参照してください。

4.1.1. システムの現在日時の表示

現在の日時を表示するには、以下のいずれかの手順を行います。

手順

1. **date** コマンドを実行します。

```
$ date
Mon Mar 30 16:02:59 CEST 2020
```

2. 詳細は、**timedatectl** コマンドを使用して確認します。

```
$ timedatectl
Local time: Mon 2020-03-30 16:04:42 CEST
Universal time: Mon 2020-03-30 14:04:42 UTC
  RTC time: Mon 2020-03-30 14:04:41
 Time zone: Europe/Prague (CEST, +0200)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

関連情報

- Web コンソールでの時間設定の設定
- **man date(1)** および **man timedatectl(1)**

4.2. システムロケールの設定

システム全体にわたるロケール設定は **/etc/locale.conf** ファイルに保存され、システム起動の初期段階で **systemd** デーモンにより読み込まれます。**/etc/locale.conf** に設定したロケール設定は、個別のプログラムやユーザーが上書きしない限り、すべてのサービスやユーザーに継承されます。

手順

- 利用可能なシステムロケール設定をリスト表示するには、次のコマンドを実行します。

```
$ localectl list-locales
C.utf8
aa_DJ
aa_DJ.iso88591
aa_DJ.utf8
...
```

- システムロケール設定の現在のステータスを表示するには、次のコマンドを実行します。

```
$ localectl status
```

- デフォルトのシステムロケールオプションを設定または変更するには、**root** ユーザーで **localectl set-locale** サブコマンドを使用します。以下に例を示します。

```
# localectl set-locale LANG=en_US
```

関連情報

- **man localectl(1)**、**man locale(7)**、および **man locale.conf(5)**

4.3. キーボードレイアウトの設定

キーボードレイアウト設定では、テキストコンソールとグラフィカルユーザーインターフェイスで使用するレイアウトを管理します。

手順

- 利用可能なキーマップをリスト表示するには、以下を実行します。

```
$ localectl list-keymaps
ANSI-dvorak
al
al-plisi
amiga-de
amiga-us
...
```

- キーマップ設定の現在のステータスを表示するには、次のコマンドを実行します。

```
$ localectl status
...
VC Keymap: us
...
```

- デフォルトのシステムキーマップを設定または変更します。以下に例を示します。

```
# localectl set-keymap us
```

関連情報

- **man localectl(1)**、**man locale(7)**、および **man locale.conf(5)**

4.4. テキストコンソールモードでフォントサイズの変更

setfont コマンドを使用して、仮想コンソールのフォントサイズを変更できます。

- フォント名を指定した **setfont** コマンドを実行します。以下に例を示します。

```
# setfont /usr/lib/kbd/consolefonts/LatArCyrHeb-19.psfu.gz
```

注記

setfont コマンドは、デフォルトで複数のハードコードされたパスを検索します。したがって、**setfont** では、フォントの完全な名前とパスは必要ありません。

- フォントのサイズを水平方向と垂直方向に 2 倍にするには、**-d** パラメーターを指定して **setfont** コマンドを入力します。

```
# setfont -d LatArCyrHeb-16
```

注記

2 倍にできる最大フォントサイズは 16 x 16 ピクセルです。

- システムの再起動中に選択したフォントを保持するには、**/etc/vconsole.conf** ファイルの **FONT** 変数を使用します。次に例を示します。

```
# cat /etc/vconsole.conf
KEYMAP="us"
FONT="eurlatgr"
```

- `kbd` パッケージとともにインストールされる **kbd-misc** パッケージには、さまざまなフォントが含まれています。たとえば、フォント **LatArCyrHeb** には多くのバリアントがあります。

```
# rpm -ql kbd-misc | grep LatAr

/usr/lib/kbd/consolefonts/LatArCyrHeb-08.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-14.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-16+.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-16.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-19.psfu.gz
```

注記

仮想コンソールでサポートされる最大フォントサイズは 32 ピクセルです。コンソールの解像度を小さくすることで、フォントの読みやすさの問題を軽減できます。

4.5. 関連情報

- 標準の RHEL 8 インストールの実行

第5章 2 台のシステム間で OPENSSH を使用した安全な通信の使用

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間で安全な通信を提供し、ユーザーがリモートでサーバーホストシステムにログインできるようにするプロトコルです。FTP、Telnet などの他のリモート通信プロトコルとは異なり、SSH はログインセッションを暗号化するため、侵入者が接続して暗号化されていないパスワードを入手するのが困難になります。

Red Hat Enterprise Linux には、基本的な **OpenSSH** パッケージ (一般的な **openssh** パッケージ、**openssh-server** パッケージ、および **openssh-clients** パッケージ) が含まれます。**OpenSSH** パッケージには、**OpenSSL** パッケージ (**openssl-libs**) が必要です。このパッケージは、重要な暗号化ライブラリーをいくつかインストールして、暗号化通信を提供する **OpenSSH** を有効にします。

5.1. SSH と OPENSSH

SSH (Secure Shell) は、リモートマシンにログインしてそのマシンでコマンドを実行するプログラムです。SSH プロトコルは、安全でないネットワーク上で、信頼されていないホスト間で安全な通信を提供します。また、X11 接続と任意の TCP/IP ポートを安全なチャンネルで転送することもできます。

SSH プロトコルは、リモートシェルのログインやファイルコピー用に使用する場合に、システム間の通信の傍受や特定ホストの偽装など、セキュリティーの脅威を軽減します。これは、SSH クライアントとサーバーがデジタル署名を使用してそれぞれの ID を確認するためです。さらに、クライアントシステムとサーバーシステムとの間の通信はすべて暗号化されます。

ホストキーは、SSH プロトコルのホストを認証します。ホスト鍵は、OpenSSH の初回インストール時、またはホストの初回起動時に自動的に生成される暗号鍵です。

OpenSSH は、Linux、UNIX、および同様のオペレーティングシステムでサポートされている SSH プロトコルの実装です。OpenSSH クライアントとサーバー両方に必要なコアファイルが含まれます。OpenSSH スイートは、以下のユーザー空間ツールで構成されます。

- **SSH** は、リモートログインプログラム (SSH クライアント) です。
- **sshd** は、OpenSSH SSH デーモンです。
- **scp** は、安全なリモートファイルコピープログラムです。
- **sftp** は、安全なファイル転送プログラムです。
- **ssh-agent** は、秘密鍵をキャッシュする認証エージェントです。
- **ssh-add** は、秘密鍵の ID を **ssh-agent** に追加します。
- **ssh-keygen** が、**ssh** の認証キーを生成、管理、および変換します。
- **ssh-copy-id** は、ローカルの公開鍵をリモート SSH サーバーの **authorized_keys** ファイルに追加するスクリプトです。
- **ssh-keyscan** - SSH パブリックホストキーを収集します。

現在、SSH のバージョンには、バージョン 1 と新しいバージョン 2 の 2 つがあります。RHEL の OpenSSH スイートは、SSH バージョン 2 のみをサポートします。このスイートは、バージョン 1 で知られているエクスプロイトに対して脆弱ではない拡張キー交換アルゴリズムを備えています。

RHEL コア暗号化サブシステムの 1 つである OpenSSH は、システム全体の暗号化ポリシーを使用します。これにより、弱い暗号スイートおよび暗号化アルゴリズムがデフォルト設定で無効になります。ポ

リシーを変更するには、管理者が **update-crypto-policies** コマンドを使用して設定を調節するか、システム全体の暗号化ポリシーを手動でオプトアウトする必要があります。

OpenSSH スイートは、2 セットの設定ファイルを使用します。1 つはクライアントプログラム (つまり、**ssh**、**scp**、および **sftp**) 用で、もう 1 つはサーバー (**sshd** デーモン) 用です。

システム全体の SSH 設定情報が **/etc/ssh/** ディレクトリーに保存されます。ユーザー固有の SSH 設定情報は、ユーザーのホームディレクトリーの **~/.ssh/** に保存されます。OpenSSH 設定ファイルの詳細なリストは、**sshd (8)** の man ページの **FILES** セクションを参照してください。

関連情報

- **man -k ssh** コマンドを使用してリスト表示される man ページ
- システム全体の暗号化ポリシーの使用

5.2. OPENSSH サーバーの設定および起動

お使いの環境と OpenSSH サーバーの起動に必要となる基本設定には、以下の手順を使用します。デフォルトの RHEL インストールを行うと、**sshd** デーモンがすでに起動し、サーバーのホスト鍵が自動的に作成されることに注意してください。

前提条件

- **openssh-server** パッケージがインストールされている。

手順

1. 現行セッションで **sshd** デーモンを開始し、ブート時に自動的に起動するように設定します。

```
# systemctl start sshd
# systemctl enable sshd
```

2. **/etc/ssh/sshd_config** 設定ファイルの **ListenAddress** ディレクティブに、デフォルトの **0.0.0.0** (IPv4) または **::** (IPv6) とは異なるアドレスを指定し、より低速な動的ネットワーク設定を使用するには、**network-online.target** ターゲットユニットの依存関係を **sshd.service** ユニットファイルに追加します。これを行うには、以下の内容で **/etc/systemd/system/sshd.service.d/local.conf** ファイルを作成します。

```
[Unit]
Wants=network-online.target
After=network-online.target
```

3. **/etc/ssh/sshd_config** 設定ファイルの OpenSSH サーバーの設定がシナリオの要件を満たしているかどうかを確認します。

4. 必要に応じて、**/etc/issue** ファイルを編集して、クライアント認証を行う前に OpenSSH サーバーに表示される welcome メッセージを変更します。以下に例を示します。

```
Welcome to ssh-server.example.com
Warning: By accessing this server, you agree to the referenced terms and conditions.
```

Banner オプションが **/etc/ssh/sshd_config** でコメントアウトされておらず、その値に **/etc/issue** が含まれていることを確認します。

```
# less /etc/ssh/sshd_config | grep Banner
Banner /etc/issue
```

ログインに成功すると表示されるメッセージを変更するには、サーバーの **/etc/motd** ファイルを編集する必要があります。詳細は、**pam_motd** の man ページを参照してください。

5. **systemd** 設定を再読み込みし、**sshd** を再起動して変更を適用します。

```
# systemctl daemon-reload
# systemctl restart sshd
```

検証

1. **sshd** デーモンが実行していることを確認します。

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-11-18 14:59:58 CET; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1149 (sshd)
    Tasks: 1 (limit: 11491)
   Memory: 1.9M
   CGroup: /system.slice/sshd.service
           └─1149 /usr/sbin/sshd -D -oCiphers=aes128-ctr,aes256-ctr,aes128-cbc,aes256-cbc -
oMACs=hmac-sha2-256,>

Nov 18 14:59:58 ssh-server-example.com systemd[1]: Starting OpenSSH server daemon...
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on 0.0.0.0 port 22.
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on :: port 22.
Nov 18 14:59:58 ssh-server-example.com systemd[1]: Started OpenSSH server daemon.
```

2. SSH クライアントを使用して SSH サーバーに接続します。

```
# ssh user@ssh-server-example.com
ECDSA key fingerprint is SHA256:dXbaS0RG/UzlTTku8GtXSz0S1++lPegSy31v3L/FAEc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ssh-server-example.com' (ECDSA) to the list of known hosts.

user@ssh-server-example.com's password:
```

関連情報

- **sshd(8)** および **sshd_config(5)** の man ページ。

5.3. 鍵ベースの認証用の OPENSSH サーバーの設定

システムのセキュリティーを強化するには、OpenSSH サーバーでパスワード認証を無効にして鍵ベースの認証を有効にします。

前提条件

- **openssh-server** パッケージがインストールされている。
- サーバーで **sshd** デーモンが実行している。

手順

1. テキストエディターで **/etc/ssh/sshd_config** 設定を開きます。以下に例を示します。

```
# vi /etc/ssh/sshd_config
```

2. **PasswordAuthentication** オプションを **no** に変更します。

```
PasswordAuthentication no
```

新しいデフォルトインストール以外のシステムで **PubkeyAuthentication no** が設定されていないことと、**ChallengeResponseAuthentication** ディレクティブが **no** に設定されていることを確認します。リモートで接続している場合は、コンソールもしくは帯域外アクセスを使用せず、パスワード認証を無効にする前に、鍵ベースのログインプロセスをテストします。

3. NFS がマウントされたホームディレクトリーで鍵ベースの認証を使用するには、SELinux ブール値 **use_nfs_home_dirs** を有効にします。

```
# setsebool -P use_nfs_home_dirs 1
```

4. **sshd** デーモンを再読み込みし、変更を適用します。

```
# systemctl reload sshd
```

関連情報

- **sshd(8)**、**sshd_config(5)**、および **setsebool(8)** の man ページ。

5.4. SSH 鍵ペアの生成

以下の手順を使用して、ローカルシステムに SSH 鍵ペアを生成し、生成された公開鍵を OpenSSH サーバーにコピーします。サーバーが正しく設定されている場合は、パスワードなしで OpenSSH サーバーにログインできます。

重要

root で次の手順を完了すると、鍵を使用できるのは **root** だけとなります。

手順

1. SSH プロトコルのバージョン 2 用の ECDSA 鍵ペアを生成するには、次のコマンドを実行します。

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/joesec/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joesec/.ssh/id_ecdsa.
```

```
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

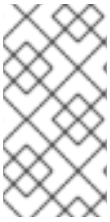
ssh-keygen コマンドまたは Ed25519 鍵ペアに **-t rsa** オプションを指定して RSA 鍵ペアを生成するには、**ssh-keygen -t ed25519** コマンドを実行します。

2. 公開鍵をリモートマシンにコピーするには、次のコマンドを実行します。

```
$ ssh-copy-id joesec@ssh-server-example.com
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
joesec@ssh-server-example.com's password:
...
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joesec@ssh-server-example.com'" and check to make sure that only the key(s) you wanted were added.
```

セッションで **ssh-agent** プログラムを使用しない場合は、上記のコマンドで、最後に変更した **~/.ssh/id*.pub** 公開鍵をコピーします (インストールされていない場合)。別の公開キーファイルを指定したり、**ssh-agent** により、メモリーにキャッシュされた鍵よりもファイル内の鍵の方が優先順位を高くするには、**-i** オプションを指定して **ssh-copy-id** コマンドを使用します。



注記

システムを再インストールする際に、生成しておいた鍵ペアを引き続き使用する場合は、**~/.ssh/** ディレクトリーのバックアップを作成します。再インストール後に、このディレクトリーをホームディレクトリーにコピーします。これは、(**root** を含む) システムの全ユーザーで実行できます。

検証

1. パスワードなしで OpenSSH サーバーにログインします。

```
$ ssh joesec@ssh-server-example.com
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

関連情報

- **ssh-keygen (1)** および **ssh-copy-id (1)** の man ページ

5.5. スマートカードに保存された SSH 鍵の使用

Red Hat Enterprise Linux では、OpenSSH クライアントでスマートカードに保存されている RSA 鍵および ECDSA 鍵を使用できるようになりました。この手順に従って、パスワードの代わりにスマートカードを使用した認証を有効にします。

前提条件

- クライアントで、**opensc** パッケージをインストールして、**pcscd** サービスを実行している。

手順

1. PKCS #11 の URI を含む OpenSC PKCS #11 モジュールが提供する鍵のリストを表示し、その出力を **keys.pub** ファイルに保存します。

```
$ ssh-keygen -D pkcs11: > keys.pub
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-
path=/usr/lib64/pkcs11/opensc-pkcs11.so
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. リモートサーバー (**example.com**) でスマートカードを使用した認証を有効にするには、公開鍵をリモートサーバーに転送します。前の手順で作成された **keys.pub** で **ssh-copy-id** コマンドを使用します。

```
$ ssh-copy-id -f -i keys.pub username@example.com
```

3. 手順 1 の **ssh-keygen -D** コマンドの出力にある ECDSA 鍵を使用して **example.com** に接続するには、鍵を一意に参照する URI のサブセットのみを使用できます。以下に例を示します。

```
$ ssh -i "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so" example.com
Enter PIN for 'SSH key':
[example.com] $
```

4. ~/**.ssh**/**config** ファイルで同じ URI 文字列を使用して、設定を永続化できます。

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so"
$ ssh example.com
Enter PIN for 'SSH key':
[example.com] $
```

OpenSSH は **p11-kit-proxy** ラッパーを使用し、OpenSC PKCS #11 モジュールが PKCS#11 キットに登録されているため、以前のコマンドを簡素化できます。

```
$ ssh -i "pkcs11:id=%01" example.com
Enter PIN for 'SSH key':
[example.com] $
```

PKCS #11 の URI の **id=** の部分を飛ばすと、OpenSSH が、プロキシーモジュールで利用可能な鍵をすべて読み込みます。これにより、必要な入力の量を減らすことができます。

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
[example.com] $
```

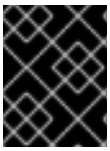
関連情報

- Fedora 28: Better smart card support in OpenSSH
- **p11-kit(8)**、**opensc.conf(5)**、**pcscd(8)**、**ssh(1)**、および **ssh-keygen(1)** の man ページ

5.6. OPENSSH のセキュリティーの強化

以下のヒントは、OpenSSH を使用する際にセキュリティーを高めるのに役に立ちます。OpenSSH 設定ファイル **/etc/ssh/sshd_config** を変更するには、**sshd** デーモンを再読み込みして有効にする必要があることに注意してください。

```
# systemctl reload sshd
```



重要

ほとんどのセキュリティー強化の設定変更により、最新のアルゴリズムまたは暗号スイートに対応していないクライアントとの互換性が低下します。

安全ではない接続プロトコルの無効化

- SSH を本当の意味で有効なものにするため、OpenSSH スイートに置き換えられる安全ではない接続プロトコルを使用しないようにします。このような接続プロトコルを使用すると、ユーザーのパスワード自体は SSH を使用した 1 回のセッションで保護されても、その後に Telnet を使用してログインした時に傍受されてしまうためです。このため、telnet、rsh、rlogin、ftp などの安全ではないプロトコルを無効にすることを検討してください。

鍵ベースの認証の有効化およびパスワードベースの認証の無効化

- 認証用パスワードを無効にして鍵のペアのみを許可すると、攻撃対象領域が減ってユーザーの時間を節約できる可能性があります。クライアントにおいて、**ssh-keygen** ツールを使用して鍵のペアを生成し、**ssh-copy-id** ユーティリティーを使用して OpenSSH サーバーのクライアントから公開鍵をコピーします。OpenSSH サーバーでパスワードベースの認証を無効にするには、**/etc/ssh/sshd_config** の **PasswordAuthentication** オプションを **no** に変更します。

  ```
  PasswordAuthentication no
  ```

鍵のタイプ

- **ssh-keygen** コマンドは、デフォルトで RSA 鍵のペアを生成しますが、**-t** オプションを使用して ECDSA 鍵または Ed25519 鍵を生成するように指定できます。ECDSA (Elliptic Curve Digital Signature Algorithm) は、同等の対称鍵強度で RSA よりも優れたパフォーマンスを提供します。また、短いキーも生成します。Ed25519 公開鍵アルゴリズムは、RSA、DSA、および ECDSA より安全で高速な歪曲エドワーズ曲線の実装です。
  サーバーホストの鍵の RSA、ECDSA、および Ed25519 がない場合は、OpenSSH が自動的に作成します。RHEL でホストの鍵の作成を設定するには、インスタンス化したサービス **sshd-**

keygen@.service を使用します。たとえば、RSA 鍵タイプの自動作成を無効にするには、次のコマンドを実行します。

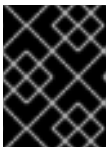
```
# systemctl mask sshd-keygen@rsa.service
```

注記

cloud-init が有効になっているイメージでは、**ssh-keygen** ユニットが自動的に無効になります。これは、**ssh-keygen template** サービスが **cloud-init** ツールに干渉し、ホストキーの生成で問題が発生する可能性があるためです。これらの問題を回避するには、**cloud-init** が実行している場合に、**etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** ドロップイン設定ファイルにより **ssh-keygen** ユニットが無効になります。

- SSH 接続の特定の鍵タイプを除外するには、**/etc/ssh/sshd_config** で該当行をコメントアウトして **sshd** サービスを再読み込みします。たとえば、Ed25519 ホストキーだけを許可するには、次のコマンドを実行します。

```
# HostKey /etc/ssh/ssh_host_rsa_key
# HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```



重要

Ed25519 アルゴリズムは FIPS-140 に準拠していないため、OpenSSH は FIPS モードの Ed25519 キーでは機能しません。

デフォルト以外のポート

- デフォルトでは、**sshd** デーモンは TCP ポート 22 をリッスンします。ポートを変更すると、自動ネットワークスキャンに基づく攻撃にシステムがさらされる可能性が減るため、あいまいさによりセキュリティーが向上します。ポートは、**/etc/ssh/sshd_config** 設定ファイルの **Port** ディレクティブを使用して指定できます。
  また、デフォルト以外のポートを使用できるように、デフォルトの SELinux ポリシーも更新する必要があります。そのためには、**policycoreutils-python-utils** パッケージの **semanage** ツールを使用します。

```
# semanage port -a -t ssh_port_t -p tcp <port_number>
```

さらに、**firewalld** 設定を更新します。

```
# firewall-cmd --add-port <port_number>/tcp
# firewall-cmd --remove-port=22/tcp
# firewall-cmd --runtime-to-permanent
```

前のコマンドの **<port_number>** は、**Port** ディレクティブを使用して指定した新しいポート番号に置き換えます。

root ログインなし

- 特定の使用例で root ユーザーとしてログインする必要がない場合は、**/etc/ssh/sshd_config**

ファイルで **PermitRootLogin** 設定ディレクティブを **no** に設定できます。root ユーザーとしてログインする可能性を無効にすることにより、管理者は、通常のユーザーとしてログインし、root 権限を取得した後に、どのユーザーがどの特権コマンドを実行するかを監査できます。
または、**PermitRootLogin** を **prohibit-password** に設定します。

```
PermitRootLogin prohibit-password
```

これにより、root としてログインしてパスワードを使用する代わりに鍵ベースの認証が使用され、ブルートフォース攻撃を防ぐことでリスクが軽減します。

X セキュリティー拡張機能の使用

- Red Hat Enterprise Linux クライアントの X サーバーは、X セキュリティー拡張を提供しません。そのため、クライアントは X11 転送を使用して信頼できない SSH サーバーに接続するときに別のセキュリティー層を要求できません。ほとんどのアプリケーションは、この拡張機能を有効にしても実行できません。
デフォルトでは、**/etc/ssh/ssh_config.d/05-redhat.conf** ファイルの **ForwardX11Trusted** オプションが **yes** に設定され、**ssh -X remote_machine** コマンド (信頼できないホスト) と **ssh -Y remote_machine** コマンド (信頼できるホスト) には違いがありません。

  シナリオで X11 転送機能を必要としない場合は、**/etc/ssh/sshd_config** 設定ファイルの **X11Forwarding** ディレクティブを **no** に設定します。

特定のユーザー、グループ、またはドメインへのアクセス制限

- **/etc/ssh/sshd_config** 設定ファイルの **AllowUsers** ディレクティブおよび **AllowGroups** ディレクティブを使用すると、特定のユーザー、ドメイン、またはグループのみが OpenSSH サーバーに接続することを許可できます。**AllowUsers** および **AllowGroups** を組み合わせて、アクセスをより正確に制限できます。以下に例を示します。

  ```
  AllowUsers *@192.168.1.* *@10.0.0.* !*@192.168.1.2
  AllowGroups example-group
  ```

  この設定行は、192.168.1.* サブネットおよび 10.0.0.* のサブネットのシステムの全ユーザーからの接続を許可します (192.168.1.2 アドレスのシステムを除く)。すべてのユーザーは、**example-group** グループに属している必要があります。OpenSSH サーバーは、その他のすべての接続を拒否します。

  OpenSSH サーバーは、**/etc/ssh/sshd_config** 内のすべての Allow および Deny ディレクティブを渡す接続のみを許可します。たとえば、**AllowUsers** ディレクティブに、**AllowGroups** ディレクティブにリストされているグループの一部ではないユーザーがリストされている場合、そのユーザーはログインできません。

  許可リストは、許可されていない新しいユーザーまたはグループもブロックするため、許可リスト (Allow で始まるディレクティブ) の使用は、拒否リスト (Deny で始まるオプション) を使用するよりも安全です。

システム全体の暗号化ポリシーの変更

- OpenSSH は、RHEL のシステム全体の暗号化ポリシーを使用し、デフォルトのシステム全体の暗号化ポリシーレベルは、現在の脅威モデルに安全な設定を提供します。暗号化の設定をより厳格にするには、現在のポリシーレベルを変更します。

  ```
  # update-crypto-policies --set FUTURE
  Setting system policy to FUTURE
  ```

-

警告

システムがインターネット上で通信する場合、**FUTURE** ポリシーの厳密な設定により、相互運用性の問題が発生する可能性があります。

システム全体の暗号化ポリシーにより、SSH プロトコルの特定の暗号のみを無効にすることもできます。詳細は、セキュリティーの強化 ドキュメントの サブポリシーを使用したシステム全体の暗号化ポリシーのカスタマイズ セクションを参照してください。

OpenSSH サーバーに対するシステム全体の暗号化ポリシーを除外するには、**/etc/sysconfig/sshd** ファイルの **CRYPTO_POLICY=** 変数行のコメントを除外します。この変更後、**/etc/ssh/sshd_config** ファイルの **Ciphers** セクション、**MACs** セクション、**KexAlgoritms** セクション、および **GSSAPIKexAlgorithms** セクションで指定した値は上書きされません。

詳細は、**sshd_config(5)** の man ページを参照してください。

OpenSSH クライアントのシステム全体の暗号化ポリシーをオプトアウトするには、次のいずれかのタスクを実行します。

- 指定のユーザーの場合は、**~/.ssh/config** ファイルのユーザー固有の設定でグローバルの **ssh_config** を上書きします。
- システム全体の場合は、**/etc/ssh/ssh_config.d/** ディレクトリーにあるドロップイン設定ファイルに暗号化ポリシーを指定します。このとき、辞書式順序で **05-redhat.conf** ファイルよりも前に来るように、5 未満の 2 桁の接頭辞と、**.conf** という接尾辞を付けます (例: **04-crypto-policy-override.conf**)。

関連情報

- **sshd_config(5)**、**ssh-keygen(1)**、**crypto-policies(7)**、および **update-crypto-policies(8)** の man ページ
- セキュリティー強化 ドキュメントの システム全体の暗号化ポリシーの使用
- ssh サービスのみに対して特定のアルゴリズムと暗号を無効化する方法 の記事

5.7. SSH ジャンプホストを使用してリモートサーバーに接続

この手順に従って、ジャンプホストとも呼ばれる中間サーバーを介してローカルシステムをリモートサーバーに接続します。

前提条件

- ジャンプホストでローカルシステムからの SSH 接続に対応している。
- リモートサーバーが、ジャンプホストからのみ SSH 接続を受け入れる。

手順

1. ローカルシステムの ~/**.ssh/config** ファイルを編集してジャンプホストを定義します。以下に例を示します。

```
Host jump-server1
  HostName jump1.example.com
```

   - **Host** パラメーターは、**ssh** コマンドで使用できるホストの名前またはエイリアスを定義します。値は実際のホスト名と一致可能ですが、任意の文字列にすることもできます。
   - **HostName** パラメーターは、ジャンプホストの実際のホスト名または IP アドレスを設定します。

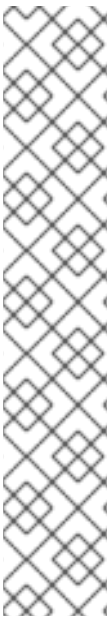
2. **ProxyJump** ディレクティブを使用してリモートサーバーのジャンプ設定を、ローカルシステムの ~/**.ssh/config** ファイルに追加します。以下に例を示します。

```
Host remote-server
  HostName remote1.example.com
  ProxyJump jump-server1
```

3. ローカルシステムを使用して、ジャンプサーバー経由でリモートサーバーに接続します。

```
$ ssh remote-server
```

   このコマンドは、設定手順 1 および 2 を省略したときの **ssh -J jump-server1 remote-server** コマンドと同じです。



注記

ジャンプサーバーをさらに指定することもできます。また、完全なホスト名を指定する場合は、設定ファイルへのホスト定義の追加を飛ばすこともできます。以下に例を示します。

```
$ ssh -J jump1.example.com,jump2.example.com,jump3.example.com
remote1.example.com
```

ジャンプサーバーのユーザー名または SSH ポートが、リモートサーバーの名前およびポートと異なる場合は、上記のコマンドのホスト名のみの表記を変更します。以下に例を示します。

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.example.com:75 joesec@remote1.example.com:220
```

関連情報

- **ssh_config(5)** および **ssh(1)** の man ページ

5.8. SSH-AGENT を使用して SSH キーでリモートマシンに接続する手順

パスフレーズを SSH 接続を開始するたびに入力しなくて済むようにするには、**ssh-agent** ユーティリティーを使用して SSH 秘密鍵をキャッシュします。秘密鍵とパスフレーズのセキュリティーが確保されます。

前提条件

前提条件

- SSH デーモンが実行中で、ネットワーク経由で到達可能なリモートホストがある。
- リモートホストにログインするための IP アドレスまたはホスト名および認証情報を把握している。
- パスフレーズで SSH キーペアを生成し、公開鍵をリモートマシンに転送している。

詳細は、SSH 鍵ペアの生成 を参照してください。

手順

1. 必要に応じて、キーを使用してリモートホストに対して認証できることを確認します。
   a. SSH を使用してリモートホストに接続します。

   ```
   $ ssh example.user1@198.51.100.1 hostname
   ```

   b. 秘密鍵へのアクセス権を付与する鍵の作成時に指定したパスフレーズを入力します。

   ```
   $ ssh example.user1@198.51.100.1 hostname
    host.example.com
   ```

2. **ssh-agent** を起動します。

   ```
   $ eval $(ssh-agent)
   Agent pid 20062
   ```

3. **ssh-agent** にキーを追加します。

   ```
   $ ssh-add ~/.ssh/id_rsa
   Enter passphrase for ~/.ssh/id_rsa:
   Identity added: ~/.ssh/id_rsa (example.user0@198.51.100.12)
   ```

検証

- オプション: SSH を使用してホストマシンにログインします。

  ```
  $ ssh example.user1@198.51.100.1

  Last login: Mon Sep 14 12:56:37 2020
  ```

  パスフレーズを入力する必要がないことに注意してください。

5.9. 関連情報

- **sshd(8)**、**ssh(1)**、**scp(1)**、**sftp(1)**、**ssh-keygen(1)**、**ssh-copy-id(1)**、**ssh_config(5)**、**sshd_config(5)**、**update-crypto-policies(8)**、および **crypto-policies(7)** の man ページ
- OpenSSH のホームページ
- 非標準設定でのアプリケーションとサービスの SELinux 設定

- Controlling network traffic using firewalld

第6章 基本的なシステムセキュリティーの設定

コンピューターセキュリティーとは、盗難、損傷、破壊、および誤りからコンピューターシステムやハードウェア、ソフトウェア、情報、およびサービスを保護することです。機密データを処理してビジネス取引を扱う企業では特に、コンピューターセキュリティーの確保は必須タスクです。

本セクションでは、オペレーティングシステムのインストール後に設定できる基本的なセキュリティー機能のみを説明します。

6.1. ファイアウォールサービスの有効化

ファイアウォールは、デフォルトのセキュリティールールに基づいてネットワークトラフィックの送受信の監視および制御を行うネットワークセキュリティーシステムです。ファイアウォールは、通常、信頼できる安全な内部ネットワークと、その他の外部ネットワークとの間に壁を作ります。

Red Hat Enterprise Linux でファイアウォールを提供する **firewalld** サービスは、インストール時に自動的に有効になります。

firewalld サービスを有効にするには、以下の手順に従います。

手順

- **firewalld** の現在の状況の表示

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset:
enabled)
   Active: inactive (dead)
...
```

- **firewalld** が有効になっていない場合は、**root** ユーザーに切り替えて、**firewalld** サービスを起動し、システムの再起動後に自動的に起動できるようにします。

```
# systemctl enable --now firewalld
```

検証手順

- **firewalld** が実行中で、有効になっていることを確認します。

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset:
enabled)
   Active: active (running)
...
```

関連情報

- firewalld の使用および設定
- **man firewalld(1)**

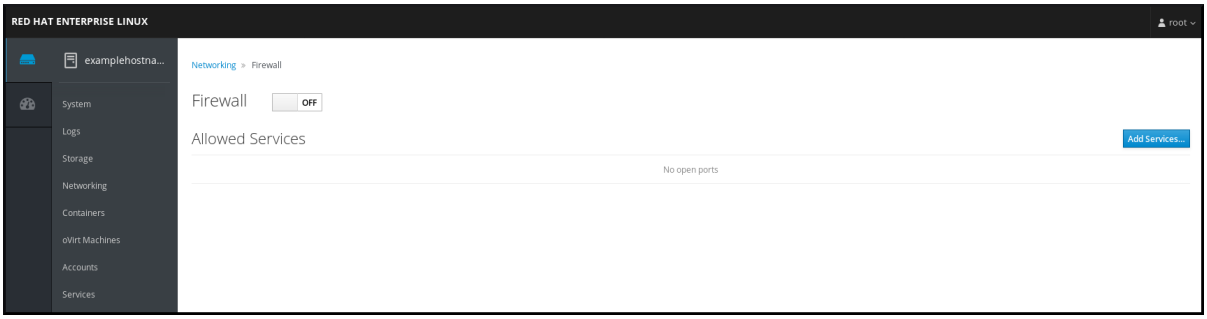
6.2. RHEL 8 WEB コンソールでファイアウォールの管理

Web コンソールで **firewalld** サービスを設定するには、**ネットワーク → ファイアウォール** に移動します。

デフォルトでは、**firewalld** サービスは有効になっています。

手順

1. Web コンソールで **firewalld** を有効または無効にするには、**ファイアウォール** のトグルボタンを切り替えます。



注記

さらに、**Add services...** ボタンを使用して、ファイアウォール経由でのサービスへのアクセスをより詳細にわたり定義できます。

6.3. 基本的な SELINUX 設定の管理

Security Enhanced Linux (SELinux) は、どのプロセスがどのファイル、ディレクトリー、およびポートにアクセスできるのかを指定するシステムセキュリティーの追加レイヤーです。これらのパーミッションは、SELinux ポリシーで定義します。ポリシーは、SELinux セキュリティーエンジンをガイドする一連のルールです。

SELinux のステータスには、以下の 2 つがあります。

- 無効
- 有効

SELinux が有効な場合は、以下のいずれのモードで実行できます。

- 有効
  - Enforcing
  - Permissive

Enforcing モード では、SELinux は読み込まれたポリシーを強制的に実行します。SELinux は、SELinux ポリシールールに基づいてアクセスを拒否し、明示的に許可された対話だけを有効にします。Enforcing モードは最も安全な SELinux モードであり、インストール後のデフォルトモードです。

Permissive モード では、SELinux は読み込まれたポリシーを強制に実行しません。SELinux はアクセスを拒否しませんが、ルールに違反するアクションを **/var/log/audit/audit.log** ログで報告します。Permissive モードは、インストール時のデフォルトのモードです。Permissive モードは、問題のトラブルシューティングなど、特定のケースで役に立ちます。

関連情報

- SELinux の使用

6.4. SELINUX で必要なステータスの確認

デフォルトでは、SELinux は Enforcing モードで動作します。ただし、特定のシナリオでは、SELinux を Permissive モードに設定したり、無効にしたりすることもできます。

重要

Red Hat は、Enforcing モードでシステムを使用することを推奨します。デバッグの目的で、SELinux を Permissive モードに設定します。

以下の手順に従って、システムの SELinux の状態とモードを変更します。

手順

1. 現在有効な SELinux モードを表示します。

```
$ getenforce
```

2. SELinux を一時的に設定します。

   a. Enforcing モードに設定する場合:

```
# setenforce Enforcing
```

   b. Permissive モードに設定する場合:

```
# setenforce Permissive
```

注記

再起動後に、SELinux モードは **/etc/selinux/config** 設定ファイルで指定された値に設定されます。

3. 再起動後も SELinux モードを保持するように設定するには、**/etc/selinux/config** 設定ファイルの **SELINUX** 変数を変更します。
   たとえば、SELinux を Enforcing モードに切り替えるには、以下のように設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
...
```

警告

SELinux を無効にすると、システムセキュリティーが低下します。無効にすると、メモリーリークや競合状態によりカーネルパニックが発生する可能性があるため、**/etc/selinux/config** ファイルの **SELINUX=disabled** オプションを使用して SELinux を無効にしないでください。代わりに、**selinux=0** パラメーターをカーネルコマンドラインに追加して、SELinux を無効にします。詳細は、Changing SELinux modes at boot time を参照してください。

関連情報

- SELinux のステータスおよびモードの変更

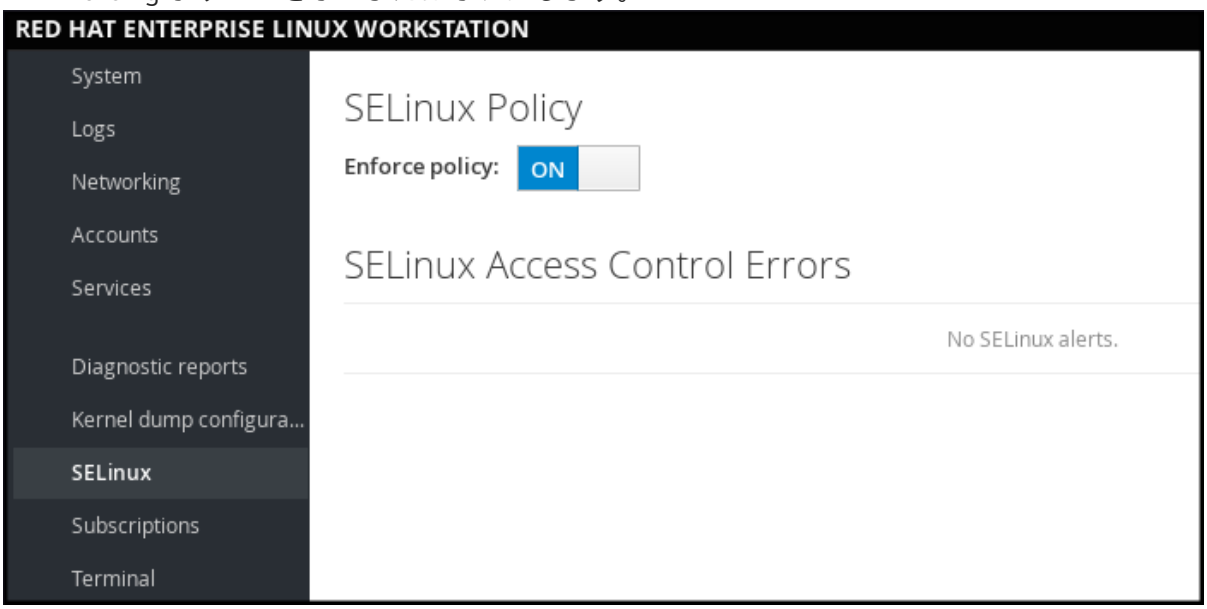
6.5. RHEL 8 WEB コンソールで SELINUX モードの切り替え

SELinux メニュー項目の RHEL 8 Web コンソールで SELinux モードを設定できます。

デフォルトでは、Web コンソールの SELinux の Enforcing ポリシーが有効になっており、SELinux が Enforcing モードで動作します。このモードを無効にして、SELinux を Permissive モードに切り替えます。モードの選択は、次回の起動時に **/etc/sysconfig/selinux** ファイルで定義されている設定に自動的に元に戻されます。

手順

1. Web コンソールで、SELinux メニュー項目の **Enforce policy** トグルボタンを使用して SELinux の Enforcing ポリシーをオンまたはオフにします。



6.6. 関連情報

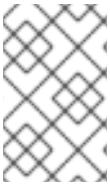
- SSH 鍵ペアの生成
- 鍵ベースの認証用の OpenSSH サーバーの設定

- セキュリティーの強化
- SELinux の使用
- ネットワークのセキュリティー保護
- 複数のシステムへの同じ SELinux 設定のデプロイメント

第7章 ソフトウェアパッケージの管理

7.1. RHEL 8 のソフトウェア管理ツール

RHEL 8 では、**DNF** テクノロジーをベースにする新しいバージョンの **YUM** ツール (**YUM v4**) によりソフトウェアインストールが有効になります。



注記

アップストリームのドキュメントでは、このテクノロジーを **DNF** と識別しているので、このツールはアップストリームの **DNF** と呼ばれます。これにより、RHEL 8 の新しい **YUM** ツールが返した出力の一部で、**DNF** と表示されます。

RHEL 8 で使用される **YUM v4** は **DNF** に基づいていますが、RHEL 7 で使用される **YUM v3** と互換性があります。ソフトウェアのインストールでは、**yum** コマンドとそのオプションのほとんどが、RHEL 7 で実行したように RHEL 8 で機能します。

選択した **yum** プラグインおよびユーティリティーは、新しい DNF バックエンドに移植されており、RHEL 7 と同じ名前でインストールできます。パッケージは互換性を持ったシンボリックリンクを提供するため、バイナリー、設定ファイル、ディレクトリーは通常の場所で確認できます。

YUM v3 が提供する以前の Python API は利用できなくなりました。**YUM v4** (DNF Python API) が提供する安定し、完全に対応する新しい API に、使用しているプラグインおよびスクリプトを移行できます。詳細は、DNF API Reference を参照してください。

7.2. アプリケーションストリーム

RHEL 8 では、Application Streams という概念が導入されています。ユーザー空間コンポーネントのバージョンが複数配信され、オペレーティングシステムのコアパッケージよりも頻繁に更新されるようになりました。これにより、プラットフォームや特定デプロイメントの基本的な安定性に影響を及ぼすことなく、Red Hat Enterprise Linux をカスタマイズできる柔軟性が向上しました。

Application Streams として使用できるコンポーネントは、モジュールまたは RPM パッケージとしてパッケージ化され、RHEL 8 の AppStream リポジトリーを介して配信されます。各アプリケーションストリームには、特定のアプリケーションにより適した、RHEL 8 と同じか、より短いライフサイクルが指定されています。ライフサイクルが短いアプリケーションストリームは、Red Hat Enterprise Linux 8 Application Streams ライフサイクル ページに記載されています。

モジュールは、論理ユニット (アプリケーション、言語スタック、データベース、またはツールセット) を表すパッケージの集まりです。これらのパッケージはまとめてビルドされ、テストされ、そしてリリースされます。

モジュールストリームは、アプリケーションストリームコンポーネントのバージョンを表します。たとえば、postgresql モジュールでは 2 つのストリーム (バージョン) の PostgreSQL データベースサーバー、つまり PostgreSQL 10 (デフォルトストリーム) および PostgreSQL 9.6 が利用できます。システムにインストールできるモジュールストリームは 1 つだけです。複数のコンテナーで異なるバージョンを使用できます。

詳細なモジュールコマンドは ユーザー空間コンポーネントのインストール、管理、および削除 を参照してください。AppStream で利用可能なモジュールのリストは、Package manifest を参照してください。

7.3. ソフトウェアパッケージの検索

yum を使用すると、ソフトウェアパッケージをすべて操作できます。

以下のセクションでは、**yum** を使用して以下を行う方法を説明します。

- パッケージを検索する
- パッケージをリスト表示する
- リポジトリーをリスト表示する
- パッケージに関する情報を表示する
- パッケージグループをリスト表示する
- yum input での glob 表現を指定する

7.3.1. YUM を使用したパッケージの検索

特定のアプリケーションやその他のコンテンツを提供するパッケージを探すには、以下の手順で行います。

手順

- パッケージを検索するには、以下を使用します。

```
# yum search term
```

term は、パッケージ関連の用語に置き換えます。

yum search コマンドでは、パッケージの名前と概要に含まれる用語で一致したものが返されることに注意してください。これにより、検索時間が短縮され、名前が分からないものの、関連用語が分かっているパッケージの検索が可能になります。

- パッケージの説明に一致する用語を含めるには、以下を使用します。

```
# yum search --all term
```

term は、パッケージ名、概要、または説明で検索する用語に置き換えます。

yum search --all はより完全な検索を可能にしますが、速度が遅くなることに注意してください。

7.3.2. YUM を使用したパッケージのリスト表示

以下の手順を使用して、インストール済みおよび使用可能なパッケージをリスト表示します。

手順

- インストール済みおよび利用可能なすべてのパッケージに関する情報を一覧表示するには、次を使用します。

```
# yum list --all
```

- システムにインストールされているパッケージのリストを表示するには、以下のコマンドを使用します。

```
# yum list --installed
```

- 有効なすべてのリポジトリーで、インストール可能な全パッケージを表示するには、以下を使用します。

```
# yum list --available
```

glob 表現を引数として追加して結果をフィルターできることに注意してください。詳細は、yum 入力でのグローバル表現の指定

を参照してください。

7.3.3. YUM を使用したリポジトリーのリスト表示

有効なリポジトリーと無効なリポジトリーをリスト表示するには、以下の手順を使用します。

手順

- システムで有効なリポジトリーをすべてリスト表示するには、以下を使用します。

```
# yum repolist
```

- システムで無効になっているリポジトリーをすべてリストを表示するには、以下を使用します。

```
# yum repolist --disabled
```

- 有効および無効なリポジトリーの両方をリスト表示するには、以下を使用します。

```
# yum repolist --all
```

- リポジトリーに関する追加情報をリスト表示するには、以下を使用します。

```
# yum repoinfo
```

リポジトリーの ID または名前を引数として渡すか、glob 表現を追加して結果をフィルタリングできることに注意してください。詳細は、yum 入力でのグローバル表現の指定

を参照してください。

7.3.4. YUM を使用したパッケージ情報の表示

YUM を使用して、パッケージに関する様々な種類の情報、例えばバージョン、リリース、サイズ、ロードされたプラグインなどを表示することができます。

手順

- パッケージに関する情報を表示するには、以下を使用します。

```
# yum info package-name
```

package-name を、パッケージ名に置き換えます。

glob 表現を引数として追加して結果をフィルターできることに注意してください。詳細は、yum 入力でのグローバル表現の指定

を参照してください。

7.3.5. YUM を使用したパッケージグループのリスト表示

インストールされたパッケージグループを表示し、リスト表示の結果をフィルタリングするには、**yum** を使用します。

手順

- インストール済みおよび利用可能なグループの数を表示するには、以下を使用します。

```
# yum group summary
```

- インストール済みおよび利用可能なグループをすべてリスト表示するには、以下のコマンドを使用します。

```
# yum group list
```

yum group list コマンドのコマンドラインオプション (**--hidden**、**--available**) を追加して結果をフィルタリングできます。利用可能なオプションの詳細は、man ページを参照してください。

- 特定のグループに含まれている必須および任意のパッケージをリスト表示するには、次のコマンドを実行します。

```
# yum group info group-name
```

group-name は、グループ名に置き換えます。

glob 表現を引数として追加して結果をフィルターできることに注意してください。詳細は、yum 入力でのグローバル表現の指定

を参照してください。

7.3.6. YUM 入力でのグローバル表現の指定

yum コマンドでは、1 つ以上の **glob 表現** を引数として追加することで、結果をフィルタリングできます。**yum** コマンドの引数として渡す場合は、グローバル表現をエスケープする必要があります。

手順

To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- glob 表現全体を二重引用符または単一引用符でくくる

```
# yum provides "*/file-name"
```

file-name は、ファイルの名前に置き換えます。

- ワイルドカード文字の前にバックスラッシュ記号 (\) を入力して、ワイルドカード文字をエスケープする

```
# yum provides \*/file-name
```

file-name は、ファイルの名前に置き換えます。

7.4. ソフトウェアパッケージのインストール

以下のセクションでは、**yum** を使用して以下を行う方法を説明します。

- パッケージをインストールする
- パッケージグループをインストールする
- yum input にパッケージ名を指定する

7.4.1. YUM を使用したパッケージのインストール

- パッケージとすべてのパッケージ依存関係をインストールするには、以下を使用します。

```
# yum install package-name
```

package-name を、パッケージ名に置き換えます。

- 複数のパッケージとその依存関係を同時にインストールするには、以下を使用します。

```
# yum install package-name-1 package-name-2
```

package-name-1 および **package-name-2** は、パッケージ名に置き換えます。

- **multilib** システムにパッケージをインストールする場合に (AMD64、Intel 64 マシン)、パッケージのアーキテクチャーをパッケージ名に追加して指定できます。

```
# yum install package-name.arch
```

package-name.arch は、パッケージの名前およびアーキテクチャーに置き換えます。

- インストールするバイナリー名は分かっているが、パッケージ名が分からない場合は、引数としてバイナリーへのパスを使用できます。

```
# yum install /usr/sbin/binary-file
```

/usr/sbin/binary-file は、バイナリーファイルへのパスに置き換えます。

yum はパッケージリストで検索を行い、**/usr/sbin/binary-file** を提供するパッケージを探します。パッケージが見つかると、そのパッケージをインストールするかどうかを尋ねられます。

- ローカルディレクトリーからダウンロード済みのパッケージをインストールするには、以下を使用します。

```
# yum install /path/
```

/path/ は、パッケージへのパスに置き換えます。

引数の解析方法を明示的に定義することで、パッケージ検索を最適化できます。詳細は、「YUM 入力でのパッケージ名の指定」 を参照してください。

7.4.2. YUM を使用したパッケージグループのインストール

以下の手順では、**yum** を使用して、グループ名または groupID でパッケージグループをインストールする方法を説明します。

手順

- パッケージグループをグループ名でインストールするには、以下を使用します。

  ```
  # yum group install group-name
  ```

  または

  ```
  # yum install @group-name
  ```

  group-name は、グループまたは環境グループのフルネームに置き換えます。

- groupID でパッケージグループをインストールするには、以下を使用します。

  ```
  # yum group install groupID
  ```

  groupID は、グループの ID に置き換えます。

7.4.3. YUM 入力でのパッケージ名の指定

To optimize the installation and removal process, you can append **-n**, **-na**, or **-nevra** suffixes to **yum install** and **yum remove** commands to explicitly define how to parse an argument:

- 正確な名前を使用してパッケージをインストールするには、以下を使用します。

  ```
  # yum install-n name
  ```

  name は、パッケージの正確な名前に置き換えます。

- 正確な名前およびアーキテクチャーを使用してパッケージをインストールするには、以下を使用します。

  ```
  # yum install-na name.architecture
  ```

  name および **architecture** は、パッケージの正確な名前およびアーキテクチャーに置き換えます。

- 正確な名前、エポック、バージョン、リリース、およびアーキテクチャーを使用してパッケージをインストールするには、以下を使用します。

  ```
  # yum install-nevra name-epoch:version-release.architecture
  ```

  name、**epoch**、**version**、**release**、および **architecture** は、パッケージの正確な名前、エポック、バージョン、リリース、およびアーキテクチャーに置き換えます。

7.5. ソフトウェアパッケージの更新

yum を使用すると、システムに保留中の更新があるかどうかを確認できます。更新が必要なパッケージをリスト表示して、1つのパッケージ、複数のパッケージ、またはすべてのパッケージを一度に更新できます。更新パッケージに依存関係がある場合は、合わせて更新されます。

以下のセクションでは、yum を使用して以下を行う方法を説明します。

- 更新の有無を確認する
- 1つのパッケージを更新する
- パッケージグループを更新する
- すべてのパッケージとその依存関係を更新する
- セキュリティー更新を適用する
- ソフトウェアの更新を自動化する

7.5.1. YUM による更新の確認

以下の手順では、**yum** を使用して、システムにインストールされているパッケージで利用可能な更新を確認する方法を説明します。

手順

- システムにインストールされているパッケージで更新を利用できるのはどれかを確認するには、以下のコマンドを実行します。

```
# yum check-update
```

このコマンドは、更新が利用可能なパッケージおよびその依存関係のリストを表示します。

7.5.2. YUM を使用した単一パッケージの更新

以下の手順を使用し、**yum** を使用して単一のパッケージとその依存関係を更新します。

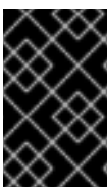
重要

カーネルの更新を適用する場合、**yum** は、**yum update** コマンドまたは **yum install** コマンドを使用しているかどうかに関わらず、新しいカーネルを常に **インストール** します。

- パッケージを更新するには、以下を使用します。

```
# yum update package-name
```

package-name を、パッケージ名に置き換えます。



重要

BIOS または IBM Power システムで GRUB ブートローダーパッケージをアップグレードした場合は、GRUB を再インストールします。GRUB の再インストール を参照してください。

7.5.3. YUM を使用したパッケージグループの更新

以下の手順を使用し、**yum** を使用してパッケージのグループとその依存関係を更新します。

手順

- パッケージグループを更新するには、以下を使用します。

```
# yum group update group-name
```

group-name は、パッケージグループの名前に置き換えます。

重要

BIOS または IBM Power システムで GRUB ブートローダーパッケージをアップグレードした場合は、GRUB を再インストールします。GRUB の再インストール を参照してください。

7.5.4. YUM を使用したすべてのパッケージとその依存関係の更新

以下の手順を使用し、**yum** を使用してすべてのパッケージとその依存関係を更新します。

手順

- すべてのパッケージとその依存関係を更新するには、次のコマンドを実行します。

```
# yum update
```

重要

BIOS または IBM Power システムで GRUB ブートローダーパッケージをアップグレードした場合は、GRUB を再インストールします。GRUB の再インストール を参照してください。

7.5.5. YUM を使用したセキュリティー関連パッケージの更新

以下の手順を使用し、**yum** を使用してセキュリティーエラータを持つ利用可能なパッケージを更新します。

手順

- セキュリティーエラータが含まれる、利用可能な最新パッケージにアップグレードするには以下を使用します。

```
# yum update --security
```

- 最新のセキュリティーエラータパッケージにアップグレードするには、以下を使用します。

```
# yum update-minimal --security
```

重要

BIOS または IBM Power システムで GRUB ブートローダーパッケージをアップグレードした場合は、GRUB を再インストールします。GRUB の再インストール を参照してください。

7.5.6. ソフトウェア更新の自動化

パッケージの更新の確認とダウンロードを定期的に行うには、**dnf-automatic** パッケージの DNF Automatic ツールを使用できます。

DNF Automatic は、systemd タイマー、cron ジョブ、その他のツールを使用した自動実行および定期実行に適した **yum** の代替コマンドラインインターフェイスです。

DNF Automatic は、必要に応じてパッケージのメタデータを同期し、利用できる更新を確認します。その後、このツールは、設定方法に応じて以下のアクションのいずれかを実行できます。

- 終了
- 更新済みパッケージのダウンロード
- 更新のダウンロードおよび適用

その後、標準出力やメールなど、選択したメカニズムによって操作の結果が報告されます。

7.5.6.1. DNF Automatic のインストール

以下の手順では、DNF Automatic ツールをインストールする方法を説明します。

手順

- **dnf-automatic** パッケージをインストールするには、次のコマンドを実行します。

```
# yum install dnf-automatic
```

検証手順

- インストールの成功を確認するには、以下のコマンドを実行して **dnf-automatic** パッケージが存在することを確認します。

```
# rpm -qi dnf-automatic
```

7.5.6.2. DNF Automatic 設定ファイル

デフォルトでは、DNF Automatic は、動作を定義するための設定ファイルとして **/etc/dnf/automatic.conf** を使用します。

設定ファイルは、以下のトピックセクションに分かれています。

- **[commands]** セクション
  DNF Automatic の操作モードを設定します。
- **[emitters]** セクション
  DNF Automatic の結果が報告される方法を定義します。

- **[command_email]** セクション
  電子メールの送信に使用する外部コマンドのメールエミッター設定を提供します。
- **[email]** セクション
  電子メールエミッターの設定を提供します。
- **[base]** セクション
  yum の主な設定ファイルのオプションを上書きします。

/etc/dnf/automatic.conf のデフォルト設定では、**DNF Automatic** は、利用可能な更新を自動的にチェックし、ダウンロードして、結果を標準出力として報告します。

警告

[commands] セクションの操作モードの設定は、**dnf-automatic.timer** 以外のすべてのタイマーユニットに対して、systemd タイマーユニットで使用される設定によって上書きされます。

関連情報

- 特定のセクションの詳細は、DNF Automatic ドキュメント を参照してください。
- systemd タイマーユニットの詳細は、**man dnf-automatic** で man ページを参照してください。
- **dnf-automatic パッケージ** に含まれる systemd タイマーユニットの概要については、Overview of the systemd timer units included in the dnf-automatic package Overview of the systemd timer units included in the dnf-automatic package のセクションを参照してください。

7.5.6.3. DNF Automatic の有効化

DNF Automatic を実行するには、常に特定の systemd タイマーユニットを有効にして起動する必要があります。**dnf-automatic** パッケージで提供されるタイマーユニットのいずれかを使用するか、必要に応じて独自のタイマーユニットを作成することができます。

以下のセクションでは、**DNF Automatic** を有効にする方法を説明します。

前提条件

- **/etc/dnf/automatic.conf** 設定ファイルを変更して、DNF Automatic の動作を指定している。

DNF Automatic 設定ファイルの詳細は、セクション 2.5.6.2 DNF Automatic 設定ファイルを参照してください。

手順

- ニーズに合った systemd タイマーユニットを選択し、有効にして開始します。

```
# systemctl enable --now <unit>
```

ここで、**<unit>** は以下のタイマーのいずれかです。

- **dnf-automatic-download.timer**
- **dnf-automatic-install.timer**
- **dnf-automatic-notifyonly.timer**
- **dnf-automatic.timer**
  - 利用可能な更新 **をダウンロードする** には、次を使用します。

```
# systemctl enable dnf-automatic-download.timer
# systemctl start dnf-automatic-download.timer
```

  - 利用可能な更新を **ダウンロードしてインストールする** には、次を使用します。

```
# systemctl enable dnf-automatic-install.timer
# systemctl start dnf-automatic-install.timer
```

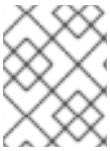
  - 利用可能な更新について **レポートする** には、次を使用します。

```
# systemctl enable dnf-automatic-notifyonly.timer
# systemctl start dnf-automatic-notifyonly.timer
```

  - 必要に応じて、以下を使用できます。

```
# systemctl enable dnf-automatic.timer
# systemctl start dnf-automatic.timer
```

更新のダウンロードおよび適用では、このタイマーユニットは **/etc/dnf/automatic.conf** 設定ファイルの設定に応じて動作します。デフォルトの動作は **dnf-automatic-download.timer** に似ています。更新されたパッケージをダウンロードしますが、インストールはしません。



注記

別の方法として、**/usr/bin/dnf-automatic** ファイルをコマンドラインまたはカスタムスクリプトから直接実行して、DNF Automatic も実行できます。

検証手順

- タイマーが有効になっていることを確認するには、次のコマンドを実行します。

```
# systemctl status <systemd timer unit>
```

関連情報

- dnf-automatic タイマーの詳細は、**man dnf-automatic** の man ページを参照してください。
- **dnf-automatic** パッケージに含まれる systemd タイマーユニットの概要は、dnf-automatic パッケージに含まれる systemd タイマーユニットの概要 のセクションを参照してください。

7.5.6.4. dnf-automatic パッケージに含まれる systemd タイマーユニットの概要

systemd タイマーユニットが優先され、更新のダウンロードと適用に関する **/etc/dnf/automatic.conf** 設定ファイルのオプションを上書きします。

たとえば、**/etc/dnf/automatic.conf** 設定ファイルで以下のオプションを設定していても、**dnf-automatic-notifyonly.timer** ユニットをアクティブにした場合は、パッケージはダウンロードされません。

```
download_updates = yes
```

dnf-automatic パッケージには、次の systemd タイマーユニットが含まれます。

タイマーユニット	機能	/etc/dnf/automatic.conf ファイルの設定を上書きしますか？
dnf-automatic-download.timer	キャッシュにパッケージをダウンロードし、更新を利用できるようにします。 注記: このタイマーユニットでは更新パッケージはインストールされません。インストールを実行するには、**dnf update** コマンドを実行する必要があります。	はい
dnf-automatic-install.timer	更新したパッケージをダウンロードしてインストールします。	はい
dnf-automatic-notifyonly.timer	リポジトリーデータのみをダウンロードして、リポジトリーキャッシュを最新の状態に維持し、利用可能な更新について通知します。 注記: このタイマーユニットでは、更新されたパッケージはダウンロードまたはインストールされません。	はい
dnf-automatic.timer	更新のダウンロードと適用に関するこのタイマーの動作は、**/etc/dnf/automatic.conf** 設定ファイルの設定で指定されます。 デフォルトの動作は、**dnf-automatic-download.timer** ユニットと同じで、パッケージのみをダウンロードし、インストールは行いません。	いいえ

関連情報

- **dnf-automatic** タイマーの詳細は、**man dnf-automatic** の man ページを参照してください。

- **/etc/dnf/automatic.conf** 設定ファイルの詳細については、セクション DNF Automatic 設定ファイル を参照してください。

7.6. ソフトウェアパッケージのアンインストール

以下のセクションでは、**yum** を使用して以下を行う方法を説明します。

- パッケージを削除する
- パッケージグループを削除する
- yum input にパッケージ名を指定する

7.6.1. YUM を使用したパッケージの削除

以下の手順を使用して、グループ名または groupID のいずれかでパッケージを削除します。

手順

- 特定のパッケージおよび依存パッケージをすべて削除するには、以下を使用します。

```
# yum remove package-name
```

package-name を、パッケージ名に置き換えます。

- 複数のパッケージとその依存関係を同時に削除するには、以下を使用します。

```
# yum remove package-name-1 package-name-2
```

package-name-1 および package-name-2 は、パッケージ名に置き換えます。

注記

yum では、依存パッケージを削除しなければパッケージを削除できません。

引数の解析方法を明示的に定義することで、パッケージ検索を最適化できます。詳細は、YUM 入力でのパッケージ名の指定 を参照してください。

7.6.2. YUM を使用したパッケージグループの削除

以下の手順を使用して、グループ名または groupID のいずれかでパッケージを削除します。

手順

- グループ名を使用してパッケージグループを削除するには、以下を使用します。

```
# yum group remove group-name
```

または

```
# yum remove @group-name
```

group-name は、グループの完全な名前に置き換えます。

- groupID でパッケージグループを削除するには、以下を使用します。

```
# yum group remove groupID
```

groupID は、グループの ID に置き換えます。

7.6.3. YUM 入力でのパッケージ名の指定

To optimize the installation and removal process, you can append **-n**, **-na**, or **-nevra** suffixes to **yum install** and **yum remove** commands to explicitly define how to parse an argument:

- 正確な名前を使用してパッケージをインストールするには、以下を使用します。

```
# yum install-n name
```

name は、パッケージの正確な名前に置き換えます。

- 正確な名前およびアーキテクチャーを使用してパッケージをインストールするには、以下を使用します。

```
# yum install-na name.architecture
```

name および architecture は、パッケージの正確な名前およびアーキテクチャーに置き換えます。

- 正確な名前、エポック、バージョン、リリース、およびアーキテクチャーを使用してパッケージをインストールするには、以下を使用します。

```
# yum install-nevra name-epoch:version-release.architecture
```

name、epoch、version、release、および architecture は、パッケージの正確な名前、エポック、バージョン、リリース、およびアーキテクチャーに置き換えます。

7.7. ソフトウェアパッケージグループの管理

パッケージグループは、共通の目的 (**システムツール**、**サウンドとビデオ**) でサービスを行うパッケージの集合です。パッケージグループをインストールすると、依存パッケージも取得するため、時間が大幅に短縮できます。

以下のセクションでは、**yum** を使用して以下を行う方法を説明します。

- パッケージグループをリスト表示する
- パッケージグループをインストールする
- パッケージグループを削除する
- yum input での glob 表現を指定する

7.7.1. YUM を使用したパッケージグループのリスト表示

インストールされたパッケージグループを表示し、リスト表示の結果をフィルタリングするには、**yum** を使用します。

手順

- インストール済みおよび利用可能なグループの数を表示するには、以下を使用します。

```
# yum group summary
```

- インストール済みおよび利用可能なグループをすべてリスト表示するには、以下のコマンドを使用します。

```
# yum group list
```

yum group list コマンドのコマンドラインオプション (**--hidden**、**--available**) を追加して結果をフィルタリングできます。利用可能なオプションの詳細は、man ページを参照してください。

- 特定のグループに含まれている必須および任意のパッケージをリスト表示するには、次のコマンドを実行します。

```
# yum group info group-name
```

group-name は、グループ名に置き換えます。

glob 表現を引数として追加して結果をフィルターできることに注意してください。詳細は、yum 入力でのグローバル表現の指定

を参照してください。

7.7.2. YUM を使用したパッケージグループのインストール

以下の手順では、**yum** を使用して、グループ名または groupID でパッケージグループをインストールする方法を説明します。

手順

- パッケージグループをグループ名でインストールするには、以下を使用します。

```
# yum group install group-name
```

または

```
# yum install @group-name
```

group-name は、グループまたは環境グループのフルネームに置き換えます。

- groupID でパッケージグループをインストールするには、以下を使用します。

```
# yum group install groupID
```

groupID は、グループの ID に置き換えます。

7.7.3. YUM を使用したパッケージグループの削除

以下の手順を使用して、グループ名または groupID のいずれかでパッケージを削除します。

手順

- グループ名を使用してパッケージグループを削除するには、以下を使用します。

```
# yum group remove group-name
```

または

```
# yum remove @group-name
```

group-name は、グループの完全な名前に置き換えます。

- groupID でパッケージグループを削除するには、以下を使用します。

```
# yum group remove groupID
```

groupID は、グループの ID に置き換えます。

7.7.4. YUM 入力でのグローバル表現の指定

yum コマンドでは、1つ以上の **glob 表現** を引数として追加することで、結果をフィルタリングできます。**yum** コマンドの引数として渡す場合は、グローバル表現をエスケープする必要があります。

手順

To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- glob 表現全体を二重引用符または単一引用符でくくる

```
# yum provides "*/file-name"
```

file-name は、ファイルの名前に置き換えます。

- ワイルドカード文字の前にバックスラッシュ記号 (\) を入力して、ワイルドカード文字をエスケープする

```
# yum provides \*/file-name
```

file-name は、ファイルの名前に置き換えます。

7.8. パッケージ管理履歴の処理

yum history コマンドを使用すると、**yum** のトランザクションのタイムライン、トランザクションの発生日時、影響を受けたパッケージ数、トランザクション成功の有無、RPM データベースがトランザクション間で変更されたかどうかといった情報を確認できます。**yum history** コマンドを使用して、トランザクションを元に戻すまたはやり直すこともできます。

以下のセクションでは、**yum** を使用して以下を行う方法を説明します。

- トランザクションをリスト表示する
- トランザクションを元に戻す
- トランザクションを繰り返す

- yum input での glob 表現を指定する

7.8.1. YUM を使用したトランザクションのリスト表示

以下の手順を使用して、最新のトランザクション、選択したパッケージの最新の操作、および特定のトランザクションの詳細をリスト表示します。

手順

- 最新の yum トランザクションのリストを表示するには、以下を使用します。

```
# yum history
```

- 選択したパッケージの最新操作のリストを表示するには、以下を使用します。

```
# yum history list package-name
```

package-name を、パッケージ名に置き換えます。glob 表現を追加して、コマンドの出力をフィルタリングできます。詳細は、YUM 入力でのグローバル表現の指定 を参照してください。

- 特定のトランザクションを調べるには、以下を使用します。

```
# yum history info transactionID
```

transactionID は、トランザクションの ID に置き換えます。

7.8.2. YUM を使用してトランザクションを元に戻す方法

以下の手順では、**yum** を使用して、選択したトランザクションまたは最後のトランザクションを元に戻す方法を説明します。

手順

- 特定のトランザクションを元に戻すには、以下を使用します。

```
# yum history undo transactionID
```

transactionID は、トランザクションの ID に置き換えます。

- 最後のトランザクションを元に戻すには、以下を使用します。

```
# yum history undo last
```

yum history undo コマンドは、トランザクション中に実行されたステップのみを元に戻すことに注意してください。トランザクションが新しいパッケージをインストールすると、**yum history undo** コマンドはこれをアンインストールします。トランザクションがパッケージをアンインストールすると、**yum history undo** コマンドはこれを再インストールします。**yum history undo** は、(古いパッケージが引き続き利用可能な場合に) 更新済みパッケージをすべて以前のバージョンにダウングレードする試みも行います。

7.8.3. yum を使用したトランザクションの繰り返し

以下の手順を使用して、**yum** を使用して、選択したトランザクションまたは最後のトランザクションを繰り返します。

手順

- 特定のトランザクションを繰り返すには、以下を使用します。

```
# yum history redo transactionID
```

  transactionID は、トランザクションの ID に置き換えます。

- 最後のトランザクションを繰り返すには、以下を使用します。

```
# yum history redo last
```

yum history redo コマンドは、トランザクション中に実行されたステップのみを繰り返すことに注意してください。

7.8.4. YUM 入力でのグローバル表現の指定

yum コマンドでは、1 つ以上の **glob 表現** を引数として追加することで、結果をフィルタリングできます。**yum** コマンドの引数として渡す場合は、グローバル表現をエスケープする必要があります。

手順

To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- glob 表現全体を二重引用符または単一引用符でくくる

```
# yum provides "*/file-name"
```

  file-name は、ファイルの名前に置き換えます。

- ワイルドカード文字の前にバックスラッシュ記号 (\) を入力して、ワイルドカード文字をエスケープする

```
# yum provides \*/file-name
```

  file-name は、ファイルの名前に置き換えます。

7.9. ソフトウェアリポジトリーの管理

yum および関連ユーティリティーの設定情報は **/etc/yum.conf** ファイルに保存されます。このファイルには **[repository]** セクションが含まれており、リポジトリー固有のオプションを設定できます。

/etc/yum.repos.d/ ディレクトリーにある、新規または既存の **.repo** ファイルに個々のリポジトリーを定義することが推奨されます。

/etc/yum.conf ファイルの各 **[repository]** セクションで定義した値は、**[main]** セクションに設定した値をオーバーライドします。

次のセクションでは、以下を行う方法を説明します。

- **[repository]** オプションを設定する

- yum リポジトリーを追加する
- yum リポジトリーを有効にする
- yum リポジトリーを無効にする

7.9.1. YUM リポジトリーオプションの設定

/etc/yum.conf 設定ファイルには **[repository]** のセクションが含まれ、repository は一意のリポジトリー ID に置き換えます。**[repository]** セクションでは、各 yum リポジトリーを定義できます。

注記

競合を回避するために、カスタムリポジトリーには Red Hat リポジトリーで使用される名前を指定しないでください。

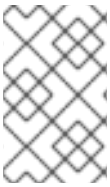
利用可能な **[repository]** オプションの完全なリストは、**yum.conf**(5) man ページの **[repository] OPTIONS** セクションを参照してください。

7.9.2. YUM リポジトリーの追加

手順

新規リポジトリーを定義するには、以下を行います。

- **[repository]** セクションを **/etc/yum.conf** ファイルに追加します。
- **/etc/yum.repos.d/** ディレクトリーの **.repo** ファイルに **[repository]** セクションを追加します。yum リポジトリーは、一般的に **.repo** ファイルを提供します。



注記

このディレクトリーにある、**.repo** ファイル拡張子が付いたすべてのファイルを yum が読み取るため、リポジトリーは、**/etc/yum.conf** ではなく、**.repo** ファイルに定義することが推奨されます。

- システムにリポジトリーを追加して有効にするには、以下を使用します。

```
# yum-config-manager --add-repo repository_URL
```

repository_url は、リポジトリーを参照する URL に置き換えます。

警告

ソフトウェアパッケージを、Red Hat の認証ベース **Content Delivery Network (CDN)** 以外の未検証または信頼できないソースから取得してインストールする場合には、セキュリティー上のリスクが伴います。セキュリティー、安定性、互換性、保全性に関する問題につながる恐れがあります。

7.9.3. YUM リポジトリーの有効化

システムに **yum** レポジトリーを追加したら、これを有効にし、確実にインストールと更新を実行できるようにしてください。

手順

- リポジトリーを有効にするには、以下を使用します。

```
# yum-config-manager --enable repositoryID
```

repositoryID は、一意のリポジトリー ID に置き換えます。

利用可能なリポジトリー ID をリスト表示するには、YUM を使用したパッケージのリスト表示 を参照してください。

7.9.4. YUM リポジトリーの無効化

特定の YUM リポジトリーを無効にして、特定のパッケージがインストールまたは更新されないようにします。

手順

- yum リポジトリーを無効にするには、以下のコマンドを使用します。

```
# yum-config-manager --disable repositoryID
```

repositoryID は、一意のリポジトリー ID に置き換えます。

利用可能なリポジトリー ID をリスト表示するには、YUM を使用したパッケージのリスト表示 を参照してください。

7.10. YUM の設定

yum および関連ユーティリティーの設定情報は **/etc/yum.conf** ファイルに保存されます。このファイルには、必須の **[main]** セクションが含まれており、このセクションを使用することで **yum** オプションを設定してグローバルに設定を適用できます。

次のセクションでは、以下を行う方法を説明します。

- 現在の **yum** 設定を表示します。
- **yum** [main] オプションを設定します。
- **yum** プラグインを使用します。

7.10.1. 現在の YUM の設定を表示する

以下の手順を使用して、現在の **yum** 設定を表示します。

手順

- **/etc/yum.conf** ファイルの **[main]** セクションで指定されるグローバル yum オプションの現在の値を表示するには、以下を使用します。

```
# yum config-manager --dump
```

7.10.2. YUM のメインオプションの設定

/etc/yum.conf 設定ファイルには **[main]** セクションが 1 つ含まれています。以下にリストされているキーと値のペアは、**yum** によるリポジトリーの操作方法および処理方法に影響を及ぼします。

/etc/yum.conf の **[main]** セクションの下に、オプションを追加できます。

利用可能な **[main]** オプションの詳細なリストは、**yum.conf**(5) man ページの **[main] OPTIONS** セクションを参照してください。

7.10.3. YUM プラグインの使用

yum は、その操作を拡張し、強化するプラグインを提供します。特定のプラグインが、デフォルトでインストールされています。

以下のセクションでは、**yum** プラグインを有効、設定、および無効にする方法を説明します。

7.10.3.1. YUM プラグインの管理

手順

プラグイン設定ファイルには常に **[main]** セクションが含まれます。このセクションでは、**enabled=** オプションで、**yum** コマンドを実行する際にプラグインを有効にするかどうかを制御します。このオプションがファイルに含まれていない場合は手動で追加できます。

/etc/dnf/plugins/ ディレクトリーには、インストールしているすべてのプラグインに対する設定ファイルがあります。これらのファイルでは、プラグイン固有のオプションを有効または無効にできます。

7.10.3.2. YUM プラグインの有効化

以下の手順では、すべての YUM プラグインを無効または有効にする方法、特定のコマンドのすべてのプラグインを無効にする方法、または単一のコマンドの特定の YUM プラグインを無効にする方法について説明します。

手順

- すべての yum プラグインを有効にするには、以下を実行します。

  1. **plugins=** で始まる行が **/etc/yum.conf** ファイルの **[main]** セクションにあることを確認します。
  2. **plugins=** の値を **1** に設定します。

     ```
     plugins=1
     ```

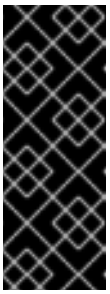
7.10.3.3. YUM プラグインの無効化

- yum プラグインをすべて無効にするには、以下を実行します。

  1. **plugins=** で始まる行が **/etc/yum.conf** ファイルの **[main]** セクションにあることを確認します。

2. **plugins=** の値を **0** に設定します。

```
plugins=0
```



重要

プラグインをすべて無効にすることは推奨 **していません**。プラグインによっては、重要な yum サービスを提供します。その中でも **product-id** プラグインおよび **subscription-manager** プラグインは、証明書ベースの **Content Delivery Network** (**CDN**) への対応に必要です。システム全体でプラグインを簡単に無効にできますが、通常は **yum** での潜在的な問題を診断する時に限り使用することを推奨します。

- 特定のコマンドで yum プラグインをすべて無効にするには、**--noplugins** オプションをコマンドに追加します。

```
# yum --noplugins update
```

- 1 つのコマンドで特定の yum プラグインを無効にするには、**--disableplugin=plugin-name** オプションをコマンドに追加します。

```
# yum update --disableplugin=plugin-name
```

plugin-name をプラグインの名前に置き換えます。

第8章 RHEL システムロールの概要

RHEL システムロールを使用すると、複数の RHEL メジャーバージョンにわたる複数の RHEL システムのシステム設定をリモートで管理できます。

重要な用語と概念

以下では、Ansible 環境における重要な用語と概念を説明します。

コントロールノード

コントロールノードは、Ansible コマンドと Playbook を実行するシステムです。コントロールノードには、Ansible Automation Platform、Red Hat Satellite、または RHEL 9、8、または 7 ホストを使用できます。詳細は、RHEL 8 でのコントロールノードの準備 を参照してください。

管理対象ノード

管理対象ノードは、Ansible で管理するサーバーとネットワークデバイスです。管理対象ノードは、ホストと呼ばれることもあります。管理対象ノードに Ansible をインストールする必要はありません。詳細は、管理対象ノードの準備 を参照してください。

Ansible Playbook

Playbook では、管理対象ノード上で実現したい設定、または管理対象ノード上のシステムが実行する一連の手順を定義します。Playbook は、Ansible の設定、デプロイメント、およびオーケストレーションの言語です。

Inventory

インベントリーファイルでは、管理対象ノードをリストし、各管理対象ノードの IP アドレスなどの情報を指定します。インベントリーでは、管理対象ノードを整理し、グループを作成およびネストして、スケーリングを容易にすることもできます。インベントリーファイルは、ホストファイルと呼ばれることもあります。

Red Hat Enterprise Linux 9 コントロールノードで利用可能なロール

Red Hat Enterprise Linux 9 コントロールノードでは、**rhel-system-roles** パッケージが次のロールを提供します。

ロール名	ロールの説明	章のタイトル
certificate	証明書の発行および更新	RHEL システムロールを使用した証明書の要求
cockpit	Web コンソール	Cockpit RHEL システムロールを使用した Web コンソールのインストールと設定
crypto_policies	システム全体の暗号化ポリシー	システム間でのカスタム暗号化ポリシーの設定
ファイアウォール (firewall)	Firewalld	システムロールを使用した firewalld の設定
ha_cluster	HA クラスター	システムロールを使用した高可用性クラスターの設定
kdump	カーネルダンプ	RHEL システムロールを使用した kdump の設定

ロール名	ロールの説明	章のタイトル
kernel_settings	カーネル設定	Ansible ロールを使用したカーネルパラメーターの永続的な設定
logging	Logging	ロギングシステムロールの使用
metrics	メトリック (PCP)	RHEL システムロールを使用したパフォーマンスの監視
microsoft.sql.server	Microsoft SQL Server	Ansible ロール microsoft.sql.server を使用した Microsoft SQL Server の設定
network	ネットワーク	network RHEL システムロールを使用した InfiniBand 接続の管理
nbde_client	ネットワークバインドディスク暗号化クライアント	nbde_client および nbde_server システムロールの使用
nbde_server	ネットワークバインドディスク暗号化サーバー	nbde_client および nbde_server システムロールの使用
postfix	postfix	システムロールの postfix ロールの変数
postgresql	PostgreSQL	postgresql RHEL システムロールを使用した PostgreSQL のインストールと設定
selinux	SELinux	システムロールを使用した SELinux の設定
ssh	SSH クライアント	ssh システムロールを使用したセキュアな通信の設定
sshd	SSH サーバー	ssh システムロールを使用したセキュアな通信の設定
ストレージ	Storage	RHEL システムロールを使用したローカルストレージの管理
tlog	ターミナルセッションの記録	tlog RHEL システムロールを使用したセッション記録用システムの設定
timesync	時刻同期	RHEL システムロールを使用した時刻同期の設定
vpn	VPN	vpn RHEL システムロールを使用した IPsec による VPN 接続の設定

関連情報

- RHEL システムロールを使用したシステム管理の自動化
- Red Hat Enterprise Linux (RHEL) system roles
- **/usr/share/ansible/roles/rhel-system-roles.<role_name>/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/<role_name>/** ディレクトリー

第9章 ロギングの設定

Red Hat Enterprise Linux のサービスの大半は、ステータスメッセージ、警告、エラーをログに記録します。**rsyslogd** サービスを使用して、これらのエントリーをローカルファイルまたはリモートロギングサーバーに記録できます。

9.1. リモートロギングソリューションの設定

環境内の各種マシンからのログをロギングサーバーに集中的に記録するために、クライアントシステムからサーバーに特定の基準に合致するログを記録するように Rsyslog アプリケーションを設定できます。

9.1.1. Rsyslog ロギングサービス

Rsyslog アプリケーションは、**systemd-journald** サービスと組み合わせて、Red Hat Enterprise Linux でローカルおよびリモートのロギングサポートを提供します。**rsyslogd** デーモンは、ジャーナルから **systemd-journald** サービスが受信した **syslog** メッセージを継続的に読み取ります。**rsyslogd** は、**syslog** イベントをフィルタリングして処理し、**rsyslog** ログファイルに記録するか、設定に応じて他のサービスに転送します。

rsyslogd デーモンは、拡張されたフィルタリング、暗号化で保護されたメッセージのリレー、入出力モジュール、TCP プロトコルおよび UDP プロトコルを使用した転送のサポートも提供します。

rsyslog の主な設定ファイルである **/etc/rsyslog.conf** では、どの **rsyslogd** がメッセージを処理するかに応じてルールを指定できます。通常は、ソースおよびトピック (ファシリティー) 別および緊急度 (優先度) 別にメッセージを分類し、メッセージがその基準に合致したときに実行するアクションを割り当てることができます。

/etc/rsyslog.conf では、**rsyslogd** が維持するログファイルのリストも確認できます。ほとんどのログファイルは **/var/log/** ディレクトリーにあります。**httpd**、**samba** などの一部のアプリケーションは、ログファイルを **/var/log/** 内のサブディレクトリーに保存します。

関連情報

- man ページの **rsyslogd(8)** および **rsyslog.conf(5)**
- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.2. Rsyslog ドキュメントのインストール

Rsyslog アプリケーションには、https://www.rsyslog.com/doc/ で利用可能な詳細なオンラインドキュメントがありますが、**rsyslog-doc** ドキュメントパッケージをローカルにインストールすることもできます。

前提条件

- システムで **AppStream** リポジトリーをアクティベートしている。
- **sudo** を使用して新規パッケージをインストールする権限がある。

手順

- **rsyslog-doc** パッケージをインストールします。

```
# yum install rsyslog-doc
```

検証

- 任意のブラウザーで **/usr/share/doc/rsyslog/html/index.html** ファイルを開きます。次に例を示します。

```
$ firefox /usr/share/doc/rsyslog/html/index.html &
```

9.1.3. TCP でのリモートロギング用のサーバーの設定

Rsyslog アプリケーションを使用すると、ロギングサーバーを実行し、個別のシステムがログファイルをロギングサーバーに送信するように設定できます。TCP 経由でリモートロギングを使用するには、サーバーとクライアントの両方を設定します。サーバーは、クライアントシステムにより送信されたログを収集し、分析します。

Rsyslog アプリケーションを使用すると、ログメッセージがネットワークを介してサーバーに転送される中央ロギングシステムを維持できます。サーバーが利用できない場合にメッセージが失われないようにするには、転送アクションのアクションキューを設定します。これにより、送信に失敗したメッセージは、サーバーが再度到達可能になるまでローカルに保存されます。このようなキューは、UDP プロトコルを使用する接続用に設定できないことに注意してください。

omfwd プラグインは、UDP または TCP による転送を提供します。デフォルトのプロトコルは UDP です。このプラグインは組み込まれているため、読み込む必要はありません。

デフォルトでは、**rsyslog** はポート **514** で TCP を使用します。

前提条件

- rsyslog がサーバーシステムにインストールされている。
- サーバーに **root** としてログインしている。
- **policycoreutils-python-utils** パッケージは、**semanage** コマンドを使用して任意の手順でインストールします。
- **firewalld** サービスが実行している。

手順

1. 必要に応じて、**rsyslog** トラフィックに別のポートを使用するには、SELinux タイプ **syslogd_port_t** をポートに追加します。たとえば、ポート **30514** を有効にします。

```
# semanage port -a -t syslogd_port_t -p tcp 30514
```

2. 必要に応じて、**rsyslog** トラフィックに別のポートを使用するには、**firewalld** がそのポートでの着信 **rsyslog** トラフィックを許可するように設定します。たとえば、ポート**30514** で TCP トラフィックを許可します。

```
# firewall-cmd --zone=<zone-name> --permanent --add-port=30514/tcp
success
# firewall-cmd --reload
```

3. **/etc/rsyslog.d/** ディレクトリーに新規ファイル (例: **remotelog.conf**) を作成し、以下のコンテンツを挿入します。

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides TCP syslog reception
module(load="imtcp")

# Adding this ruleset to process remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="30514" ruleset="remote1")
```

4. **/etc/rsyslog.d/remotelog.conf** ファイルへの変更を保存します。

5. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

6. **Rsyslog** サービスがロギングサーバーで実行中で、有効になっていることを確認します。

```
# systemctl status rsyslog
```

7. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

8. 必要に応じて、**rsyslog** が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動するようにします。

```
# systemctl enable rsyslog
```

環境内の他のシステムからログファイルを受け取り、保存するように、ログサーバーが設定されています。

関連情報

- **rsyslogd(8)**、**rsyslog.conf(5)**、**semanage(8)**、および **firewall-cmd(1)** man ページ。
- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.4. TCP 経由のサーバーへのリモートロギングの設定

以下の手順に従って、TCP プロトコルを介してサーバーにログメッセージを転送するようにシステムを設定します。**omfwd** プラグインは、UDP または TCP による転送を提供します。デフォルトのプロトコルは UDP です。プラグインは組み込まれているのでロードする必要はありません。

前提条件

- **rsyslog** パッケージが、サーバーに報告する必要のあるクライアントシステムにインストールされている。
- リモートロギング用にサーバーを設定している。
- 指定したポートが SELinux で許可され、ファイアウォールで開いている。
- システムには、**policycoreutils-python-utils** パッケージが含まれています。このパッケージは、標準以外のポートを SELinux 設定に追加するための **semanage** コマンドを提供します。

手順

1. **/etc/rsyslog.d/** ディレクトリーに新規ファイル (例: **10-remotelog.conf**) を作成し、以下のコンテンツを挿入します。

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="30514" protocol="tcp"
    )
```

   ここでは、以下のようになります。

   - **queue.type="linkedlist"** 設定は、LinkedList インメモリーキューを有効にします。
   - **queue.filename** 設定は、ディスクストレージを定義します。バックアップファイルは、前のグローバルの **workDirectory** ディレクティブで指定された作業ディレクトリーに **example_fwd** 接頭辞を付けて作成されます。
   - **action.resumeRetryCount -1** 設定は、サーバーが応答しない場合に接続を再試行するときに **rsyslog** がメッセージを破棄しないようにします。
   - **queue.saveOnShutdown="on"** 設定は、**rsyslog** がシャットダウンした場合にインメモリーデータを保存します。

- 最後の行は、受信したすべてのメッセージをロギングサーバーに転送します。ポートの指定は任意です。
  この設定では、**rsyslog** はメッセージをサーバーに送信しますが、リモートサーバーに到達できない場合には、メッセージをメモリーに保持します。ディスク上にあるファイルは、設定されたメモリーキュー領域が **rsyslog** で不足するか、シャットダウンする必要がある場合にのみ作成されます。これにより、システムパフォーマンスが向上します。

注記

Rsyslog は設定ファイル **/etc/rsyslog.d/** を字句順に処理します。

2. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

検証

クライアントシステムがサーバーにメッセージを送信することを確認するには、以下の手順に従います。

1. クライアントシステムで、テストメッセージを送信します。

```
# logger test
```

2. サーバーシステムで、**/var/log/messages** ログを表示します。以下に例を示します。

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

hostname はクライアントシステムのホスト名です。ログには、**logger** コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **rsyslogd(8)** および **rsyslog.conf(5)** man ページ。
- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.5. TLS 暗号化リモートロギングの設定

デフォルトでは、Rsyslog はプレーンテキスト形式でリモートロギング通信を送信します。シナリオでこの通信チャネルのセキュリティーを確保する必要がある場合は、TLS を使用して暗号化できます。

TLS を介した暗号化されたトランスポートを使用するには、サーバーとクライアントの両方を設定します。サーバーは、クライアントシステムにより送信されたログを収集し、分析します。

ossl ネットワークストリームドライバー (OpenSSL) または **gtls** ストリームドライバー (GnuTLS) のいずれかを使用できます。

注記

ネットワークに接続されていない、厳格な認可を受けているなど、セキュリティーが強化された別のシステムがある場合は、その別のシステムを認証局 (CA) として使用します。

サーバー側では **global**、**module**、**input** レベルで、クライアント側では **global** および **action** レベルで、ストリームドライバーを使用して接続設定をカスタマイズできます。より具体的な設定は、より一般的な設定よりも優先されます。そのため、たとえば、ほとんどの接続のグローバル設定で **ossl** を使用し、特定の接続の入力とアクション設定で **gtls** を使用することができます。

前提条件

- クライアントシステムとサーバーシステムの両方に **root** にアクセスできる。
- サーバーおよびクライアントシステムに次のパッケージがインストールされている。
  - **rsyslog** パッケージ
  - **ossl** ネットワークストリームドライバー用の **rsyslog-openssl** パッケージ
  - **gtls** ネットワークストリームドライバー用の **rsyslog-gnutls** パッケージ
  - **certtool** コマンドを使用して証明書を生成するための **gnutls-utils** パッケージ
- ログサーバーの **/etc/pki/ca-trust/source/anchors/** ディレクトリーには、次の証明書があり、**update-ca-trust** コマンドを使用してシステム設定を更新します。
  - **ca-cert.pem** - ログサーバーとクライアントで鍵と証明書を検証できる CA 証明書。
  - **server-cert.pem** - ロギングサーバーの公開鍵。
  - **server-key.pem** - ロギングサーバーの秘密鍵。
- ログクライアントでは、次の証明書が **/etc/pki/ca-trust/source/anchors/** ディレクトリーにあり、**update-ca-trust** を使用してシステム設定を更新します。
  - **ca-cert.pem** - ログサーバーとクライアントで鍵と証明書を検証できる CA 証明書。
  - **client-cert.pem** - クライアントの公開鍵。
  - **client-key.pem** - クライアントの秘密鍵。

手順

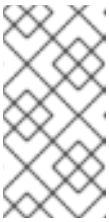
1. クライアントシステムから暗号化したログを受信するようにサーバーを設定します。
   a. **/etc/rsyslog.d/**ディレクトリーに、新規ファイル (**securelogser.conf** など) を作成します。
   b. 通信を暗号化するには、設定ファイルに、サーバーの証明書ファイルへのパス、選択した認証方法、および TLS 暗号化に対応するストリームドライバーが含まれている必要があります。**/etc/rsyslog.d/securelogser.conf** に以下の行を追加します。

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
```

```
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/server-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/server-key.pem"
)

# TCP listener
module(
  load="imtcp"
  PermittedPeer=["client1.example.com", "client2.example.com"]
  StreamDriver.AuthMode="x509/name"
  StreamDriver.Mode="1"
  StreamDriver.Name="ossl"
)

# Start up listener at port 514
input(
  type="imtcp"
  port="514"
)
```



注記

GnuTLS ドライバーが必要な場合は、**StreamDriver.Name="gtls"** 設定オプションを使用します。**x509/name** よりも厳密ではない認証モードの詳細は、**rsyslog-doc** にインストールされているドキュメントを参照してください。

c. 変更を **/etc/rsyslog.d/securelogser.conf** ファイルに保存します。

d. **/etc/rsyslog.conf** ファイルの構文と **/etc/rsyslog.d/** ディレクトリー内のすべてのファイルを確認します。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

e. **Rsyslog** サービスがロギングサーバーで実行中で、有効になっていることを確認します。

```
# systemctl status rsyslog
```

f. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

g. オプション: Rsyslog が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動されることを確認してください。

```
# systemctl enable rsyslog
```

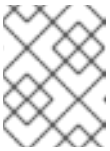
2. 暗号化したログをサーバーに送信するようにクライアントを設定するには、以下のコマンドを実行します。

a. クライアントシステムで、**/etc/rsyslog.d/**ディレクトリーに、新規ファイル (**securelogcli.conf** など) を作成します。

b. **/etc/rsyslog.d/securelogcli.conf** に以下の行を追加します。

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/client-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/client-key.pem"
)


# Set up the action for all messages
*.* action(
  type="omfwd"
  StreamDriver="ossl"
  StreamDriverMode="1"
  StreamDriverPermittedPeers="server.example.com"
  StreamDriverAuthMode="x509/name"
  target="server.example.com" port="514" protocol="tcp"
)
```



注記

GnuTLS ドライバーが必要な場合は、**StreamDriver.Name="gtls"** 設定オプションを使用します。

c. 変更を **/etc/rsyslog.d/securelogcli.conf** ファイルに保存します。

d. **/etc/rsyslog.conf** ファイルの構文と **/etc/rsyslog.d/** ディレクトリー内のその他のファイルを確認します。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

e. **Rsyslog** サービスがロギングサーバーで実行中で、有効になっていることを確認します。

```
# systemctl status rsyslog
```

f. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

g. オプション: Rsyslog が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動されることを確認してください。

```
# systemctl enable rsyslog
```

検証

クライアントシステムがサーバーにメッセージを送信することを確認するには、以下の手順に従います。

1. クライアントシステムで、テストメッセージを送信します。

■

```
# logger test
```

2. サーバーシステムで、**/var/log/messages** ログを表示します。以下に例を示します。

```
# cat /var/log/remote/msg/<hostname>/root.log
Feb 25 03:53:17 <hostname> root[6064]: test
```

<hostname> はクライアントシステムのホスト名です。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **certtool(1)**、**openssl(1)**、**update-ca-trust(8)**、**rsyslogd(8)**、および **rsyslog.conf(5)** man ページ
- **/usr/share/doc/rsyslog/html/index.html** に **rsyslog-doc** パッケージでインストールされたドキュメント。
- TLS での logging システムロールの使用

9.1.6. UDP でリモートロギング情報を受信するためのサーバー設定

Rsyslog アプリケーションを使用すると、リモートシステムからロギング情報を受信するようにシステムを設定できます。UDP 経由でリモートロギングを使用するには、サーバーとクライアントの両方を設定します。受信サーバーは、クライアントシステムが送信したログの収集および分析を行います。デフォルトでは、**rsyslog** はポート **514** で UDP を使用して、リモートシステムからログ情報を受信します。

以下の手順に従って、UDP プロトコルでクライアントシステムが送信したログの収集および分析を行うサーバーを設定します。

前提条件

- rsyslog がサーバーシステムにインストールされている。
- サーバーに **root** としてログインしている。
- **policycoreutils-python-utils** パッケージは、**semanage** コマンドを使用して任意の手順でインストールします。
- **firewalld** サービスが実行している。

手順

1. 必要に応じて、デフォルトのポート **514** 以外の **rsyslog** トラフィックに別のポートを使用するには、次のコマンドを実行します。

   a. SELinux ポリシー設定に **syslogd_port_t** SELinux タイプを追加し、**portno** は **rsyslog** で使用するポート番号に置き換えます。

   ```
   # semanage port -a -t syslogd_port_t -p udp portno
   ```

   b. **rsyslog** の受信トラフィックを許可するように **firewalld** を設定します。**portno** はポート番号に、**zone** は **rsyslog** が使用するゾーンに置き換えます。

```
# firewall-cmd --zone=zone --permanent --add-port=portno/udp
success
# firewall-cmd --reload
```

c. ファイアウォールルールを再読み込みします。

```
# firewall-cmd --reload
```

2. **/etc/rsyslog.d/** ディレクトリーに **.conf** の新規ファイル (例: **remotelogserv.conf**) を作成し、以下のコンテンツを挿入します。

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides UDP syslog reception
module(load="imudp")

# This ruleset processes remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imudp" port="514" ruleset="remote1")
```

514 は、**rsyslog** がデフォルトで使用するポート番号です。代わりに別のポートを指定できます。

3. **/etc/rsyslog.conf** ファイルの構文と **/etc/rsyslog.d/** ディレクトリー内の全 **.conf** ファイルを確認します。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
```

4. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

5. 必要に応じて、**rsyslog** が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動するようにします。

```
# systemctl enable rsyslog
```

関連情報

- **rsyslogd(8)**、**rsyslog.conf(5)**、**semanage(8)**、および **firewall-cmd(1)** man ページ。
- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.7. UDP 経由のサーバーへのリモートロギングの設定

以下の手順に従って、UDP プロトコルを介してサーバーにログメッセージを転送するようにシステムを設定します。**omfwd** プラグインは、UDP または TCP による転送を提供します。デフォルトのプロトコルは UDP です。プラグインは組み込まれているのでロードする必要はありません。

前提条件

- **rsyslog** パッケージが、サーバーに報告する必要のあるクライアントシステムにインストールされている。
- UDP でリモートロギング情報を受信するためのサーバー設定 で説明されているように、リモートロギング用にサーバーを設定している。

手順

1. **/etc/rsyslog.d/** ディレクトリーに **.conf** の新規ファイル (例: **10-remotelogcli.conf**) を作成し、以下のコンテンツを挿入します。

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="portno" protocol="udp"
    )
```

ここでは、以下のようになります。

- **queue.type="linkedlist"** 設定は、LinkedList インメモリーキューを有効にします。
- **queue.filename** 設定は、ディスクストレージを定義します。バックアップファイルは、前のグローバルの **workDirectory** ディレクティブで指定された作業ディレクトリーに **example_fwd** 接頭辞を付けて作成されます。
- **action.resumeRetryCount -1** 設定は、サーバーが応答しない場合に接続を再試行するときに **rsyslog** がメッセージを破棄しないようにします。
- **queue.saveOnShutdown="on"** 設定を有効にすると、**rsyslog** がシャットダウンした場合にインメモリーデータが保存されます。
- **portno** 値は、**rsyslog** で使用するポート番号です。デフォルト値は **514** です。

- 最後の行は受信メッセージをすべてロギングサーバーに転送します。ポートの指定は任意です。
  この設定では、**rsyslog** はメッセージをサーバーに送信しますが、リモートサーバーに到達できない場合には、メッセージをメモリーに保持します。ディスク上にあるファイルは、設定されたメモリーキュー領域が **rsyslog** で不足するか、シャットダウンする必要がある場合にのみ作成されます。これにより、システムパフォーマンスが向上します。

注記

Rsyslog は設定ファイル **/etc/rsyslog.d/** を字句順に処理します。

2. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

3. 必要に応じて、**rsyslog** が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動するようにします。

```
# systemctl enable rsyslog
```

検証

クライアントシステムがサーバーにメッセージを送信することを確認するには、以下の手順に従います。

1. クライアントシステムで、テストメッセージを送信します。

```
# logger test
```

2. サーバーシステムで、**/var/log/remote/msg/hostname/root.log** ログを表示します。以下に例を示します。

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

hostname はクライアントシステムのホスト名です。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **rsyslogd(8)** および **rsyslog.conf(5)** man ページ。
- **/usr/share/doc/rsyslog/html/index.html** に **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.8. rsyslog の負荷分散ヘルパー

RebindInterval 設定では、現行接続を切断して再確立する間隔を指定します。この設定は、TCP、UDP、および RELP のトラフィックに適用されます。ロードバランサーはこれを新しい接続と認識し、メッセージを別の物理ターゲットシステムに転送します。

RebindInterval 設定は、ターゲットシステムの IP アドレスが変更した場合にシナリオで役に立ちます。Rsyslog アプリケーションは、接続の確立時に IP アドレスをキャッシュするため、メッセージは同じサーバーに送信されます。IP アドレスが変更すると、Rsyslog サービスが再起動するまで UDP パ

ケットが失われます。接続を再確立すると、IP が DNS により再度解決されます。

```
action(type=”omfwd” protocol=”tcp” RebindInterval=”250” target=”example.com” port=”514” …)

action(type=”omfwd” protocol=”udp” RebindInterval=”250” target=”example.com” port=”514” …)

action(type=”omrelp” RebindInterval=”250” target=”example.com” port=”6514” …)
```

9.1.9. 信頼できるリモートロギングの設定

Reliable Event Logging Protocol (RELP) を使用すると、メッセージ損失のリスクを大幅に軽減して TCP で **syslog** メッセージを送受信できます。RELP は、信頼できるイベントメッセージを配信するので、メッセージ損失が許されない環境で有用です。RELP を使用するには、**imrelp** の入力モジュール (サーバー上での実行とログの受信) と **omrelp** 出力モジュール (クライアント上での実行とロギングサーバーへのログの送信) を設定します。

前提条件

- **rsyslog** パッケージ、**librelp** パッケージ、および **rsyslog-relp** パッケージをサーバーおよびクライアントシステムにインストールしている。
- 指定したポートが SELinux で許可され、ファイアウォール設定で開放されている。

手順

1. 信頼できるリモートロギング用にクライアントシステムを設定します。
   a. クライアントシステムの **/etc/rsyslog.d/** ディレクトリーに、**relpclient.conf** などと名前を指定して新しい **.conf** ファイルを作成し、以下のコンテンツを挿入します。

      ```
      module(load="omrelp")
      *.* action(type="omrelp" target="_target_IP_" port="_target_port_")
      ```

      詳細は以下のようになります。

      - **target_IP** は、ロギングサーバーの IP アドレスに置き換えます。
      - **target_port** はロギングサーバーのポートに置き換えます。

   b. 変更を **/etc/rsyslog.d/relpclient.conf** ファイルに保存します。
   c. **rsyslog** サービスを再起動します。

      ```
      # systemctl restart rsyslog
      ```

   d. 必要に応じて、**rsyslog** が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動するようにします。

      ```
      # systemctl enable rsyslog
      ```

2. 信頼できるリモートロギング用にサーバーシステムを設定します。
   a. サーバーシステムの **/etc/rsyslog.d/** ディレクトリーに、**relpserv.conf** などと名前を指定して新しい **.conf** ファイルを作成し、以下のコンテンツを挿入します。

```
ruleset(name="relp"){
*.* action(type="omfile" file="_log_path_")
}


module(load="imrelp")
input(type="imrelp" port="_target_port_" ruleset="relp")
```

詳細は以下のようになります。

- **log_path** は、メッセージを保存するパスを指定します。
- **target_port** はロギングサーバーのポートに置き換えます。クライアント設定ファイルと同じ値を使用します。

b. **/etc/rsyslog.d/relpserv.conf** ファイルへの変更を保存します。

c. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

d. 必要に応じて、**rsyslog** が有効になっていない場合は、再起動後に **rsyslog** サービスが自動的に起動するようにします。

```
# systemctl enable rsyslog
```

検証

クライアントシステムがサーバーにメッセージを送信することを確認するには、以下の手順に従います。

1. クライアントシステムで、テストメッセージを送信します。

```
# logger test
```

2. サーバーシステムで、指定された **log_path** でログを表示します。以下に例を示します。

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

hostname はクライアントシステムのホスト名です。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **rsyslogd(8)** および **rsyslog.conf(5)** man ページ。
- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

9.1.10. サポート対象の Rsyslog モジュール

Rsyslog アプリケーションの機能を拡張するために、特定のモジュールを使用できます。モジュールは、追加の入力 (入力モジュール)、出力 (出力モジュール)、およびその他の機能を提供します。モ

ジュールは、モジュールの読み込み後に利用可能な設定ディレクティブを追加で提供することも可能です。

以下のコマンドを使用して、システムにインストールされている入出力モジュールをリスト表示できます。

```
# ls /usr/lib64/rsyslog/{i,o}m*
```

rsyslog-doc パッケージをインストールした後、**/usr/share/doc/rsyslog/html/configuration/modules/idx_output.html** ファイルで使用可能なすべての **rsyslog** モジュールのリストを表示できます。

9.1.11. カーネルメッセージをリモートホストに記録するように netconsole サービスを設定

ディスクへのログインやシリアルコンソールの使用ができない場合は、**netconsole** カーネルモジュールおよび同じ名前のサービスを使用して、ネットワーク経由でカーネルメッセージをリモートの **rsyslog** サービスに記録できます。

前提条件

- **rsyslog** などのシステムログサービスがリモートホストにインストールされている。
- リモートシステムログサービスは、このホストから受信ログエントリーを受け取るように設定されています。

手順

1. **netconsole-service** パッケージをインストールします。

   ```
   # yum install netconsole-service
   ```

2. **/etc/sysconfig/netconsole** ファイルを編集し、**SYSLOGADDR** パラメーターをリモートホストの IP アドレスに設定します。

   ```
   # SYSLOGADDR=192.0.2.1
   ```

3. **netconsole** サービスを有効にして起動します。

   ```
   # systemctl enable --now netconsole
   ```

検証手順

- リモートシステムログサーバーの **/var/log/messages** ファイルを表示します。

関連情報

リモートロギングソリューションの設定

9.1.12. 関連情報

- **/usr/share/doc/rsyslog/html/index.html** ファイルに **rsyslog-doc** パッケージでインストールされたドキュメント。

- **rsyslog.conf(5)** および **rsyslogd(8)** man ページ
- ナレッジベースの記事 Configuring system logging without journald
- ナレッジベースの記事 Negative effects of the RHEL default logging setup on performance and their mitigations
- logging システムロールの使用 の章

9.2. LOGGING システムロールの使用

システム管理者は、**logging** システムロールを使用して、Red Hat Enterprise Linux ホストをロギングサーバーとして設定し、多数のクライアントシステムからログを収集できます。

9.2.1. logging RHEL システムロール

logging RHEL システムロールを使用すると、ローカルホストとリモートホストにロギング設定をデプロイできます。

ロギングソリューションは、ログと複数のロギング出力を読み取る複数の方法を提供します。

たとえば、ロギングシステムは以下の入力を受け取ることができます。

- ローカルファイル
- **systemd/journal**
- ネットワーク上の別のロギングシステム

さらに、ロギングシステムでは以下を出力できます。

- **/var/log** ディレクトリーのローカルファイルに保存されているログ
- Elasticsearch に送信されたログ
- 別のロギングシステムに転送されたログ

logging RHEL システムロールを使用すると、状況に合わせて入力と出力を組み合わせることができます。たとえば、**journal** からの入力をローカルのファイルに保存しつつも、複数のファイルから読み込んだ入力を別のロギングシステムに転送してそのローカルのログファイルに保存するようにロギングソリューションを設定できます。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー
- RHEL システムロール

9.2.2. ローカルの logging RHEL システムロールの適用

Ansible Playbook を準備して適用し、別のマシンにロギングソリューションを設定します。各マシンはログをローカルに記録します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

注記

rsyslog パッケージをインストールする必要はありません。RHEL システムロールがデプロイ時に **rsyslog** をインストールするためです。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Deploying basics input and implicit files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: system_input
        type: basics
    logging_outputs:
      - name: files_output
        type: files
    logging_flows:
      - name: flow1
        inputs: [system_input]
        outputs: [files_output]
```

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

検証

1. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. システムがログにメッセージを送信していることを確認します。

   a. テストメッセージを送信します。

   ```
   # logger test
   ```

   b. **/var/log/messages ログ** を表示します。以下に例を示します。

   ```
   # cat /var/log/messages
   Aug  5 13:48:31 <hostname> root[6778]: test
   ```

   <hostname> はクライアントシステムのホスト名に置き換えます。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー

9.2.3. ローカルの logging RHEL システムロールでのログのフィルタリング

rsyslog プロパティーベースのフィルターをもとにログをフィルターするロギングソリューションをデプロイできます。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

注記

rsyslog パッケージをインストールする必要はありません。RHEL システムロールがデプロイ時に **rsyslog** をインストールするためです。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Deploying files input and configured files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
    logging_outputs:
```

```
    - name: files_output0
      type: files
      property: msg
      property_op: contains
      property_value: error
      path: /var/log/errors.log
    - name: files_output1
      type: files
      property: msg
      property_op: "!contains"
      property_value: error
      path: /var/log/others.log
  logging_flows:
    - name: flow0
      inputs: [files_input]
      outputs: [files_output0, files_output1]
```

この設定を使用すると、**error** 文字列を含むメッセージはすべて **/var/log/errors.log** に記録され、その他のメッセージはすべて **/var/log/others.log** に記録されます。

error プロパティーの値はフィルタリングする文字列に置き換えることができます。

設定に合わせて変数を変更できます。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

検証

1. **/etc/rsyslog.conf** ファイルの構文をテストします。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. システムが **error** 文字列を含むメッセージをログに送信していることを確認します。

   a. テストメッセージを送信します。

   ```
   # logger error
   ```

   b. 以下のように **/var/log/errors.log** ログを表示します。

   ```
   # cat /var/log/errors.log
   Aug  5 13:48:31 hostname root[6778]: error
   ```

hostname はクライアントシステムのホスト名に置き換えます。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー

9.2.4. logging RHEL システムロールを使用したリモートロギングソリューションの適用

以下の手順に従って、Red Hat Ansible Core Playbook を準備および適用し、リモートロギングソリューションを設定します。この Playbook では、1 つ以上のクライアントが **systemd-journal** からログを取得し、リモートサーバーに転送します。サーバーは、**remote_rsyslog** および **remote_files** からリモート入力を受信し、リモートホスト名によって名付けられたディレクトリーのローカルファイルにログを出力します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

注記

rsyslog パッケージをインストールする必要はありません。RHEL システムロールがデプロイ時に **rsyslog** をインストールするためです。

手順

1. 次の内容を含む Playbook ファイル (例: **~/playbook.yml**) を作成します。

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
```

```
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: managed-node-02.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: <host1.example.com>
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: <host1.example.com>
        tcp_port: 601
    logging_flows:
      - name: flows0
        inputs: [basic_input]
        outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

<host1.example.com> はロギングサーバーに置き換えます。

注記

必要に応じて、Playbook のパラメーターを変更することができます。

警告

ロギングソリューションは、サーバーまたはクライアントシステムの SELinux ポリシーで定義され、ファイアウォールで開放されたポートでしか機能しません。デフォルトの SELinux ポリシーには、ポート 601、514、6514、10514、および 20514 が含まれます。別のポートを使用するには、クライアントシステムおよびサーバーシステムで SELinux ポリシーを変更 します。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

検証

1. クライアントとサーバーシステムの両方で、**/etc/rsyslog.conf** ファイルの構文をテストします。

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. クライアントシステムがサーバーにメッセージを送信することを確認します。

   a. クライアントシステムで、テストメッセージを送信します。

   ```
   # logger test
   ```

   b. サーバーシステムで、**/var/log/<host2.example.com>/messages** ログを表示します。次に例を示します。

   ```
   # cat /var/log/<host2.example.com>/messages
   Aug  5 13:48:31 <host2.example.com> root[6778]: test
   ```

   <host2.example.com> は、クライアントシステムのホスト名に置き換えます。ログには、logger コマンドを入力したユーザーのユーザー名 (この場合は **root**) が含まれていることに注意してください。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー

9.2.5. TLS を使用した logging RHEL システムロールの使用

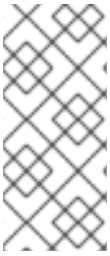
Transport Layer Security (TLS) は、コンピューターネットワーク上でセキュアな通信を可能にするために設計された暗号化プロトコルです。

管理者は、**logging** RHEL システムロールを使用し、Red Hat Ansible Automation Platform を使用したセキュアなログ転送を設定できます。

9.2.5.1. TLS を使用したクライアントロギングの設定

logging RHEL システムロールを持つ Ansible Playbook を使用して、RHEL クライアントでのロギングを設定し、TLS 暗号化を使用してログをリモートロギングシステムに転送できます。

この手順では、秘密鍵と証明書を作成し、Ansible インベントリーのクライアントグループ内のすべてのホストに TLS を設定します。TLS プロトコルは、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。



注記

証明書を作成するために、Playbook で **certificate** RHEL システムロールを呼び出す必要はありません。**logging** RHEL システムロールがこのロールを自動的に呼び出します。

CA が作成された証明書に署名できるようにするには、管理対象ノードが IdM ドメインに登録されている必要があります。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。
- 管理対象ノードが IdM ドメインに登録されている。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Deploying files input and forwards output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.pem
    logging_inputs:
      - name: input_name
        type: files
        input_log_path: /var/log/containers/*.log
    logging_outputs:
      - name: output_name
        type: forwards
        target: your_target_host
        tcp_port: 514
        tls: true
        pki_authmode: x509/name
        permitted_server: 'server.example.com'
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

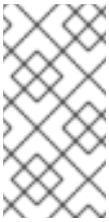
Playbook は以下のパラメーターを使用します。

logging_certificates

このパラメーターの値は、**certificate** RHEL システムロールの **certificate_requests** に渡され、秘密鍵と証明書の作成に使用されます。

logging_pki_files

このパラメーターを使用すると、TLS に使用する CA、証明書、および鍵ファイルを検索するためにロギングで使用するパスとその他の設定 (サブパラメーター **ca_cert**、**ca_cert_src**、**cert**、**cert_src**、**private_key**、**private_key_src**、および **tls** で指定) を設定できます。



注記

logging_certificates を使用してターゲットノードにファイルを作成する場合は、**ca_cert_src**、**cert_src**、および **private_key_src** を使用しないでください。これらは、**logging_certificates** によって作成されていないファイルのコピーに使用されます。

ca_cert

ターゲットノード上の CA 証明書ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

ターゲットノード上の証明書ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

ターゲットノード上の秘密鍵ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ターゲットホストの **ca_cert** で指定された場所にコピーされる、コントロールノード上の CA 証明書ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

cert_src

ターゲットホストの **cert** で指定された場所にコピーされる、コントロールノード上の証明書ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

private_key_src

ターゲットホストの **private_key** で指定された場所にコピーされる、コントロールノード上の秘密鍵ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

tls

このパラメーターを **true** に設定すると、ネットワーク上でログがセキュアに転送されます。セキュアなラッパーが必要ない場合は、**tls: false** に設定します。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

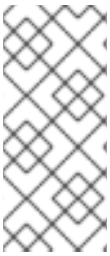
関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー
- RHEL システムロールを使用した証明書の要求

9.2.5.2. TLS を使用したサーバーロギングの設定

logging RHEL システムロールを持つ Ansible Playbook を使用して、RHEL サーバーでのロギングを設定し、TLS 暗号化を使用してリモートロギングシステムからログを受信するように設定できます。

この手順では、秘密鍵と証明書を作成し、Ansible インベントリーのサーバーグループ内のすべてのホストに TLS を設定します。



注記

証明書を作成するために、Playbook で **certificate** RHEL システムロールを呼び出す必要はありません。**logging** RHEL システムロールがこのロールを自動的に呼び出します。

CA が作成された証明書に署名できるようにするには、管理対象ノードが IdM ドメインに登録されている必要があります。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。
- 管理対象ノードが IdM ドメインに登録されている。

手順

1. 次の内容を含む Playbook ファイル (例: **~/playbook.yml**) を作成します。

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.pem
```

```
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

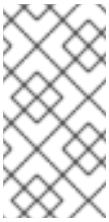
Playbook は以下のパラメーターを使用します。

logging_certificates

このパラメーターの値は、**certificate** RHEL システムロールの **certificate_requests** に渡され、秘密鍵と証明書の作成に使用されます。

logging_pki_files

このパラメーターを使用すると、TLS に使用する CA、証明書、および鍵ファイルを検索するためにロギングで使用するパスとその他の設定 (サブパラメーター **ca_cert**、**ca_cert_src**、**cert**、**cert_src**、**private_key**、**private_key_src**、および **tls** で指定) を設定できます。



注記

logging_certificates を使用してターゲットノードにファイルを作成する場合は、**ca_cert_src**、**cert_src**、および **private_key_src** を使用しないでください。これらは、**logging_certificates** によって作成されていないファイルのコピーに使用されます。

ca_cert

ターゲットノード上の CA 証明書ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

ターゲットノード上の証明書ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

ターゲットノード上の秘密鍵ファイルへのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ターゲットホストの **ca_cert** で指定された場所にコピーされる、コントロールノード上の CA 証明書ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

cert_src

ターゲットホストの **cert** で指定された場所にコピーされる、コントロールノード上の証明書ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

private_key_src

ターゲットホストの **private_key** で指定された場所にコピーされる、コントロールノード上の秘密鍵ファイルへのパスを表します。**logging_certificates** を使用する場合は、これを使用しないでください。

tls

このパラメーターを **true** に設定すると、ネットワーク上でログがセキュアに転送されます。セキュアなラッパーが必要ない場合は、**tls: false** に設定します。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー
- RHEL システムロールを使用した証明書の要求

9.2.6. RELP で logging RHEL システムロールの使用

Reliable Event Logging Protocol (RELP) とは、TCP ネットワークを使用する、データとメッセージロギング用のネットワーキングプロトコルのことです。イベントメッセージを確実に配信するので、メッセージの損失が許されない環境で使用できます。

RELP の送信側はコマンド形式でログエントリーを転送し、受信側は処理後に確認応答します。RELP は、一貫性を保つために、転送されたコマンドごとにトランザクション番号を保存し、各種メッセージの復旧します。

RELP Client と RELP Server の間に、リモートロギングシステムを検討することができます。RELP Client はリモートロギングシステムにログを転送し、RELP Server はリモートロギングシステムから送信されたすべてのログを受け取ります。

管理者は、**logging** RHEL システムロールを使用して、ログエントリーが確実に送受信されるようにロギングシステムを設定できます。

9.2.6.1. RELP を使用したクライアントロギングの設定

logging RHEL システムロールを使用すると、ローカルマシンにログが記録されている RHEL システムのロギングを設定し、Ansible Playbook を実行して RELP を使用してリモートロギングシステムにログを転送できます。

この手順では、Ansible インベントリーの **client** グループ内の全ホストに RELP を設定します。RELP 設定は Transport Layer Security (TLS) を使用して、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Deploying basic input and relp output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: logging.server.com
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/client-cert.pem
        private_key: /etc/pki/tls/private/client-key.pem
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: example_flow
        inputs: [basic_input]
        outputs: [relp_client]
```

Playbook では次の設定を使用します。

target
: リモートロギングシステムが稼働しているホスト名を指定する必須パラメーターです。

port
: リモートロギングシステムがリッスンしているポート番号です。

tls
: ネットワーク上でログをセキュアに転送します。セキュアなラッパーが必要ない場合は、**tls** 変数を **false** に設定します。デフォルトでは **tls** パラメーターは true に設定されますが、RELP を使用する場合には鍵/証明書およびトリプレット {**ca_cert**、**cert**、**private_key**} や {**ca_cert_src**、**cert_src**、**private_key_src**} が必要です。

- {**ca_cert_src**、**cert_src**、**private_key_src**} のトリプレットを設定すると、デフォルトの場所 (**/etc/pki/tls/certs** と **/etc/pki/tls/private**) が、コントロールノードからファイルを転送するため、管理対象ノードの宛先として使用されます。この場合、ファイル名はトリプレットの元の名前と同じです。

- {**ca_cert**、**cert**、**private_key**} トリプレットが設定されている場合は、ファイルはロギング設定の前にデフォルトのパスに配置されている必要があります。

- トリプレットの両方が設定されている場合には、ファイルはコントロールノードのローカルのパスから管理対象ノードの特定のパスへ転送されます。

ca_cert

CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。**ca_cert** を指定している場合は、その場所にコピーされます。

cert_src

ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。**cert** を指定している場合には、その証明書が場所にコピーされます。

private_key_src

ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。**private_key** を指定している場合は、その場所にコピーされます。

pki_authmode

name または **fingerprint** の認証モードを使用できます。

permitted_servers

ロギングクライアントが、TLS 経由での接続およびログ送信を許可するサーバーのリスト。

inputs

ロギング入力ディクショナリーのリスト。

outputs

ロギング出力ディクショナリーのリスト。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー

9.2.6.2. RELP を使用したサーバーログの設定

logging RHEL システムロールを使用して、RHEL システムでのロギングをサーバーとして設定し、Ansible Playbook を実行して RELP を使用してリモートロギングシステムからログを受信できます。

以下の手順では、Ansible インベントリーの **server** グループ内の全ホストに RELP を設定します。RELP 設定は TLS を使用して、メッセージ送信を暗号化し、ネットワーク経由でログを安全に転送します。

前提条件

- 制御ノードと管理ノードを準備している
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/server-cert.pem
        private_key: /etc/pki/tls/private/server-key.pem
        pki_authmode: name
        permitted_clients:
          - '*example.client.com'
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: example_flow
        inputs: relp_server
        outputs: remote_files_output
```

Playbook は以下の設定を使用します。

port

リモートロギングシステムがリッスンしているポート番号です。

tls

ネットワーク上でログをセキュアに転送します。セキュアなラッパーが必要ない場合は、**tls** 変数を **false** に設定します。デフォルトでは **tls** パラメーターは true に設定されますが、RELP を使用する場合には鍵/証明書およびトリプレット {**ca_cert**、**cert**、**private_key**} や {**ca_cert_src**、**cert_src**、**private_key_src**} が必要です。

- {**ca_cert_src**、**cert_src**、**private_key_src**} のトリプレットを設定すると、デフォルトの場所 (**/etc/pki/tls/certs** と **/etc/pki/tls/private**) が、コントロールノードからファイルを転送するため、管理対象ノードの宛先として使用されます。この場合、ファイル名はトリプレットの元の名前と同じです。

- {**ca_cert**、**cert**、**private_key**} トリプレットが設定されている場合は、ファイルはロギング設定の前にデフォルトのパスに配置されている必要があります。

- トリプレットの両方が設定されている場合には、ファイルはコントロールノードのローカルのパスから管理対象ノードの特定のパスへ転送されます。

ca_cert

CA 証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/ca.pem** で、ファイル名はユーザーが設定します。

cert

証明書へのパスを表します。デフォルトのパスは **/etc/pki/tls/certs/server-cert.pem** で、ファイル名はユーザーが設定します。

private_key

秘密鍵へのパスを表します。デフォルトのパスは **/etc/pki/tls/private/server-key.pem** で、ファイル名はユーザーが設定します。

ca_cert_src

ローカルの CA 証明書ファイルパスを表します。これはターゲットホストにコピーされます。**ca_cert** を指定している場合は、その場所にコピーされます。

cert_src

ローカルの証明書ファイルパスを表します。これはターゲットホストにコピーされます。**cert** を指定している場合には、その証明書が場所にコピーされます。

private_key_src

ローカルキーファイルのパスを表します。これはターゲットホストにコピーされます。**private_key** を指定している場合は、その場所にコピーされます。

pki_authmode

name または **fingerprint** の認証モードを使用できます。

permitted_clients

ロギングサーバーが TLS 経由での接続およびログ送信を許可するクライアントのリスト。

inputs

ロギング入力ディクショナリーのリスト。

outputs

ロギング出力ディクショナリーのリスト。

2. Playbook の構文を検証します。

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

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

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

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/logging/** ディレクトリー

9.2.7. 関連情報

- RHEL システムロールを使用するためのコントロールノードと管理対象ノードの準備
- **rhel-system-roles** パッケージでインストールされたドキュメントは、**/usr/share/ansible/roles/rhel-system-roles.logging/README.html** にあります。
- RHEL システムロール
- **ansible-playbook(1)** の man ページ。

第10章 ログファイルを使用した問題のトラブルシューティング

ログファイルは、システム (カーネル、サービス、および実行中のアプリケーションなど) に関するメッセージが含まれるファイルです。ログファイルには、問題のトラブルシューティングやシステム機能の監視に役立つ情報が含まれています。Red Hat Enterprise Linux におけるロギングシステムは、組み込みの **syslog** プロトコルに基づいています。特定のプログラムがこのシステムを使用してイベントを記録し、ログファイルに分類します。これは、オペレーティングシステムの監査およびさまざまな問題のトラブルシューティングに役立ちます。

10.1. SYSLOG メッセージを処理するサービス

以下の 2 つのサービスは、**syslog** メッセージを処理します。

- **systemd-journald** デーモン
- **Rsyslog** サービス

systemd-journald デーモンは、さまざまなソースからメッセージを収集し、収集したメッセージを処理するために **Rsyslog** に転送します。**systemd-journald** デーモンは、以下のソースからメッセージを収集します。

- カーネル
- ブートプロセスの初期段階
- 起動時および実行時のデーモンの標準出力とエラー
- **Syslog**

Rsyslog サービスは、タイプおよび優先順で **syslog** のメッセージを分類し、**/var/log** ディレクトリー内のファイルに書き込みます。**/var/log** ディレクトリーは、ログメッセージを永続的に保存します。

10.2. SYSLOG メッセージを保存するサブディレクトリー

/var/log ディレクトリー内の以下のサブディレクトリーでは、**syslog** メッセージを保存します。

- **/var/log/messages** - 以下を除くすべての **syslog** メッセージ
- **/var/log/secure** - セキュリティーおよび認証に関連するメッセージおよびエラー
- **/var/log/maillog** - メールサーバーに関連するメッセージおよびエラー
- **/var/log/cron** - 定期的に実行されるタスクに関連するログファイル
- **/var/log/boot.log** - システムの起動に関連するログファイル

10.3. WEB コンソールでログファイルの検査

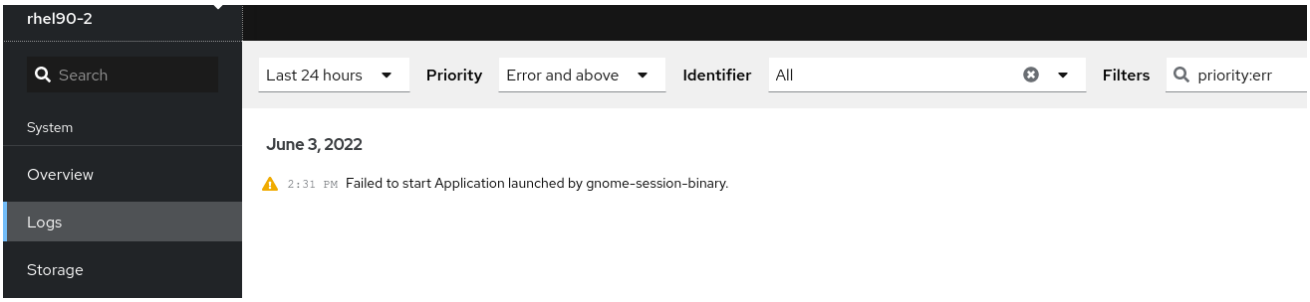
以下の手順に従って、RHEL Web コンソールを使用してログファイルを検証します。

手順

1. RHEL Web コンソールにログインします。詳細はWeb コンソールへのログイン を参照してください。

2. **Logs** をクリックします。

図10.1 RHEL 8 Web コンソールでログファイルの検査



10.4. コマンドラインでのログの表示

Journal は、ログファイルの表示および管理に役立つ systemd のコンポーネントです。従来のロギングに関連する問題に対応し、残りのシステムと密接に統合され、ログファイルのさまざまなロギングテクノロジーおよびアクセス管理をサポートします。

journalctl コマンドを使用すると、以下のようにコマンドラインを使用してシステムジャーナルのメッセージを表示できます。

```
$ journalctl -b | grep kvm
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: Using msrs 4b564d01 and 4b564d00
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: cpu 0, msr 76401001, primary cpu clock
...
```

表10.1 システム情報の表示

コマンド	説明
journalctl	収集されたジャーナルエントリーをすべて表示します。
journalctl FILEPATH	特定のファイルに関連するログを表示します。たとえば、**journalctl /dev/sda** コマンドは、**/dev/sda** ファイルシステムに関連するログを表示します。
journalctl -b	現在のブートのログを表示します。
journalctl -k -b -1	現在のブートのカーネルログを表示します。

表10.2 特定のサービスに関する情報の表示

コマンド	説明
journalctl -b _SYSTEMD_UNIT=<name.service>	ログをフィルターして、**systemd** サービスに一致するエントリーを表示します。

コマンド	説明
journalctl -b _SYSTEMD_UNIT=<name.service> _PID=<number>	一致を組み合わせます。たとえば、このコマンドは、**<name.service>** と PID**<number>** に一致する **systemd-units** のログを表示します。
journalctl -b _SYSTEMD_UNIT=<name.service> _PID=<number> + _SYSTEMD_UNIT=<name2.service>	プラス記号 (+) セパレーターは、論理 OR で 2 つの式を組み合わせます。たとえば、このコマンドは **<name.service>** サービスプロセスからのすべてのメッセージを **PID** で表示し、(プロセスのいずれかの) **<name2.service>** サービスからのすべてのメッセージも表示します。
journalctl -b _SYSTEMD_UNIT=<name.service> _SYSTEMD_UNIT=<name2.service>	このコマンドは、同じフィールドを参照し、いずれかの式に一致するエントリーをすべて表示します。このコマンドは、systemd-unit **<name.service>** または systemd-unit **<name2.service>** に一致するログを表示します。

表10.3 特定のブートに関連するログの表示

コマンド	説明
journalctl --list-boots	ブート番号、その ID、およびブートに関する最初のメッセージと最後のメッセージのタイムスタンプの表形式リストが表示されます。以下のコマンドの ID を使用して詳細情報を表示できます。
journalctl --boot=ID _SYSTEMD_UNIT=<name.service>	指定したブート ID に関する情報を表示します。

10.5. 関連情報

- **man journalctl(1)**
- リモートロギングソリューションの設定

第11章 ユーザーおよびグループの管理

ファイルやプロセスへの不正アクセスを防止するには、ユーザーとグループを正確に管理する必要があります。アカウントを一元管理していない場合、または特定のシステム上でのみユーザーアカウントやグループが必要な場合は、そのホスト上でローカルにユーザーアカウントやグループを作成できます。

11.1. ユーザーアカウントおよびグループアカウントの管理の概要

ユーザーとグループの制御は、Red Hat Enterprise Linux (RHEL) システム管理の中核となる要素です。各 RHEL ユーザーには各種ログイン認証情報があり、さまざまなグループに割り当ててシステム権限をカスタマイズすることができます。

11.1.1. ユーザーとグループの概要

ファイルを作成するユーザーは、そのファイルの所有者 **および** グループ所有者です。ファイルには、所有者、グループ、およびそのグループ外のユーザーに対して読み取り、書き込み、実行のパーミッションが別々に割り当てられます。ファイルの所有者は、**root** ユーザーのみが変更できます。ファイルへのアクセス権限を変更できるのは、**root** ユーザー、ファイル所有者の両方です。通常ユーザーは、所有するファイルのグループ所有権を、所属するグループに変更できます。

各ユーザーは、**ユーザー ID** (**UID**) と呼ばれる一意の数値 ID に関連付けられています。各グループは **グループ ID** (**GID**) に関連付けられています。グループ内のユーザーは、そのグループが所有するファイルの読み取り、書き込み、実行を行う権限を共有します。

11.1.2. 予約ユーザーおよびグループ ID の設定

RHEL は、999 以下のユーザー ID とグループ ID をシステムユーザーとグループ用に予約しています。予約ユーザー ID とグループ ID は、**setup** パッケージで確認できます。予約ユーザー ID とグループ ID を表示するには、以下を使用します。

```
cat /usr/share/doc/setup*/uidgid
```

予約範囲は今後増える可能性があるため、新規ユーザーおよびグループには、5000 以降の ID を割り当てることを推奨します。

デフォルトで新規ユーザーに割り当てる ID を 5000 以降に指定するには、**/etc/login.defs** ファイルの **UID_MIN** と **GID_MIN** パラメーターを変更します。

手順

デフォルトで新規ユーザーに割り当てる ID を 5000 以降にするには、以下のコマンドを実行します。

1. 任意のエディターで **/etc/login.defs** ファイルを開きます。
2. UID の自動選択の最小値を定義する行を見つけます。

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
```

3. **UID_MIN** の値を 5000 から開始するように変更します。

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  5000
```

4. GID の自動選択の最小値を定義する行を見つけます。

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  1000
```

5. **GID_MIN** の値を 5000 から開始するように変更します。

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  5000
```

通常のユーザーに動的に割り当てられる UID と GID は、5000 から始まります。

注記

UID_MIN および GID_MIN の値を変更する前に作成された UID および GID のユーザーおよびグループは変更されません。

これにより、新規ユーザーのグループに UID および GID と同じ 5000+ ID を持たせることができます。

警告

上限が 1000 のシステムとの競合を回避するため、**SYS_UID_MAX** を変更して、システムが予約している ID を 1000 以上にしないようにしてください。

11.1.3. ユーザープライベートグループ

RHEL は、**ユーザープライベートグループ** (**UPG**) システム設定を使用するため、UNIX グループの管理が容易になります。ユーザープライベートグループは、新規ユーザーがシステムに追加されるたびに作成されます。ユーザープライベートグループは作成したユーザーと同じ名前となり、そのユーザーがそのユーザープライベートグループの唯一のメンバーになります。

UPG は、複数ユーザー間のプロジェクトの連携を簡素化します。さらに、UPG のシステム設定では、ユーザーおよびこのユーザーが所属するグループ両方がファイルまたはディレクトリーを変更できるので、新規作成されたファイルまたはディレクトリーのデフォルトの権限を安全に設定できます。

グループのリストは、**/etc/group** 設定ファイルに保存されます。

11.2. ユーザーアカウントの管理

Red Hat Enterprise Linux は、マルチユーザー向けのオペレーティングシステムです。つまり、1 台のマ

シンにインストールされた 1 つのシステムに、複数のユーザーが別々のコンピューターからアクセスできます。各ユーザーは自身のアカウントで操作します。このような方法でユーザーアカウントを管理することは、Red Hat Enterprise Linux のシステム管理の中心的要素になります。

以下に各種ユーザーアカウントを紹介します。

- **通常のユーザーアカウント**
  通常のアカウントは特定システムのユーザー用に作成されます。このようなアカウントは、通常のシステム管理中に追加、削除、および修正できます。

- **システムユーザーアカウント:**
  システムユーザーアカウントは、システムで特定のアプリケーション識別子を表します。このようなアカウントは通常、ソフトウェアのインストール時にのみ追加または操作され、後で変更することはありません。

警告

システムアカウントは、システムでローカルに利用できると想定されています。アカウントがリモートで設定され、提供されている (LDAP の設定など) と、システムが破損したり、サービスが開始できない場合があります。

  システムアカウント用に、1000 番未満のユーザー ID が予約されています。通常のアカウントには、1000 から始まる ID を使用できます。ただし、5000 以降の ID を割り当てることが推奨されます。ID の割り当ては、**/etc/login.defs** ファイルを参照してください。

- **グループ:**
  グループとは、複数のユーザーアカウントを共通目的 (特定のファイルにアクセス権を与えるなど) で統合するエンティティーです。

11.2.1. コマンドラインツールを使用したアカウントとグループの管理

ユーザーアカウントとグループを管理するには、次の基本的なコマンドラインツールを使用します。

- ユーザーおよびグループ ID を表示します。

```
$ id
uid=1000(example.user) gid=1000(example.user) groups=1000(example.user),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 新規ユーザーアカウントを作成します。

```
# useradd example.user
```

- example.user に属するユーザーアカウントに新しいパスワードを割り当てます。

```
# passwd example.user
```

- ユーザーをグループに追加します。

```
# usermod -a -G example.group example.user
```

関連情報

- **man useradd(8)**、**man passwd(1)**、および **man usermod(8)**

11.2.2. Web コンソールで管理されるシステムユーザーアカウント

RHEL Web コンソールに表示されているユーザーアカウントでは、以下が可能になります。

- システムにアクセスする際にユーザーを認証する
- システムへのアクセス権を設定する

RHEL Web コンソールは、システムに存在するすべてのユーザーアカウントを表示します。そのため、最初に Web コンソールにログインした直後は、ユーザーアカウントが少なくとも 1 つ表示されます。

RHEL Web コンソールにログインしたら、以下の操作を実行できます。

- 新規ユーザーアカウントの作成
- パラメーターの変更
- アカウントのロック
- ユーザーセッションの終了

11.2.3. Web コンソールで新規アカウントの追加

RHEL Web コンソールを使用して、システムにユーザーアカウントを追加し、アカウントに管理権限を設定できます。

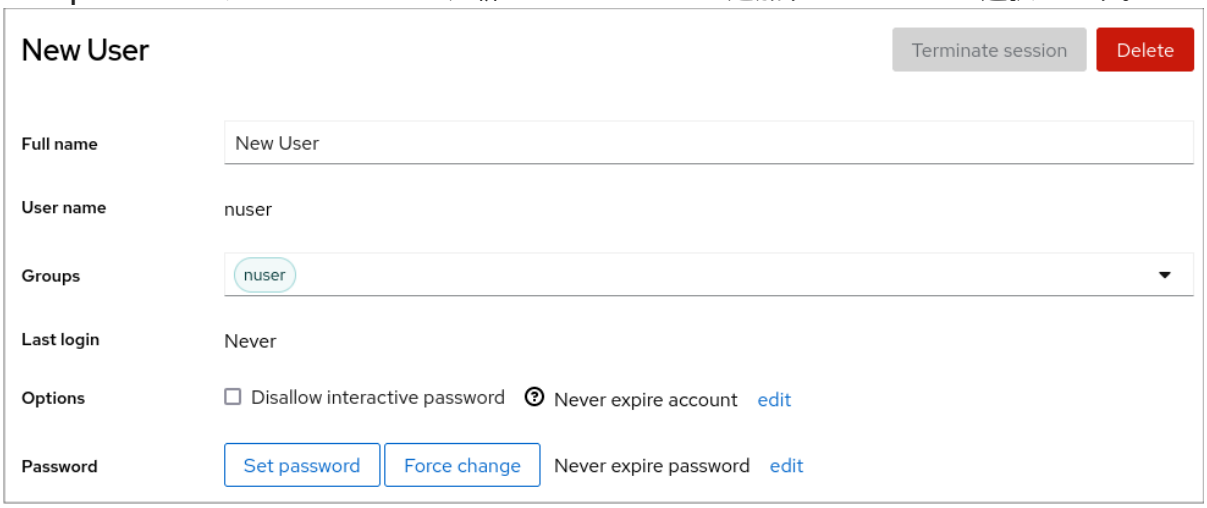
前提条件

- RHEL Web コンソールがインストールされており、アクセス可能である。詳細は、Web コンソールのインストール を参照してください。

手順

1. RHEL Web コンソールにログインします。
2. **Accounts** をクリックします。
3. **Create New Account** をクリックします。
4. **フルネーム** フィールドにユーザーの氏名を入力します。
   RHEL Web コンソールは、入力した氏名からユーザー名が自動的に作成され、**ユーザー名** フィールドに入力されます。名前の頭文字と、苗字で設定される命名規則を使用しない場合は、入力されたユーザー名を変更します。
5. **パスワード/確認** フィールドにパスワードを入力し、再度パスワードを入力します。
   フィールドの下にあるカラーバーは、入力したパスワードの強度を表し、弱いパスワードは使用できないようにします。
6. **作成** をクリックして設定を保存し、ダイアログボックスを閉じます。

7. 新規作成したアカウントを選択します。
8. **Groups** ドロップダウンメニューで、新しいアカウントに追加するグループを選択します。



これで **アカウント** 設定に新規アカウントが表示され、認証情報を使用してシステムに接続できるようになりました。

11.3. コマンドラインからのユーザーの管理

コマンドラインインターフェイス (CLI) を使用してユーザーおよびグループを管理できます。これにより、Red Hat Enterprise Linux 環境でユーザーおよびユーザーグループを追加、削除、および変更できます。

11.3.1. コマンドラインでの新規ユーザーの追加

useradd ユーティリティーを使用して、新規ユーザーを追加できます。

前提条件

- **Root** アクセス

手順

- 新規ユーザーを追加するには、以下を使用します。

```
# useradd options username
```

options は **useradd** コマンドのコマンドラインオプションに、username はユーザー名に置き換えます。

例11.1 新規ユーザーの追加

ユーザー ID が **5000** のユーザー **sarah** を追加するには、以下を使用します。

```
# useradd -u 5000 sarah
```

検証手順

- 新規ユーザーが追加されたことを確認するには、**id** ユーティリティーを使用します。

```
# id sarah
```

返される出力は以下のとおりです。

```
uid=5000(sarah) gid=5000(sarah) groups=5000(sarah)
```

関連情報

- **useradd** の man ページ

11.3.2. コマンドラインでの新規グループの追加

groupadd ユーティリティーを使用して、新規グループを追加できます。

前提条件

- **Root** アクセス

手順

- 新規グループを追加するには、以下を使用します。

```
# groupadd options group-name
```

options は **groupadd** コマンドのコマンドラインオプションに、group-name はグループ名に置き換えます。

例11.2 新規グループの追加

グループ ID が **5000** のグループ **sysadmins** を追加するには、以下を使用します。

```
# groupadd -g 5000 sysadmins
```

検証手順

- 新規グループが追加されていることを確認するには、**tail** ユーティリティーを使用します。

```
# tail /etc/group
```

返される出力は以下のとおりです。

```
sysadmins:x:5000:
```

関連情報

- **groupadd** の man ページ

11.3.3. コマンドラインから補助グループにユーザーを追加

補助グループにユーザーを追加して、権限を管理したり、特定のファイルまたはデバイスへのアクセスを有効にしたりできます。

前提条件

- **root** アクセス

手順

- ユーザーの補助グループにグループを追加するには、以下を使用します。

```
# usermod --append -G group-name username
```

group-name はグループ名に、username はユーザー名に置き換えます。

例11.3 補助グループへのユーザーの追加

ユーザーの **sysadmin** をグループ **system-administrators** に追加するには、以下を使用します。

```
# usermod --append -G system-administrators sysadmin
```

検証手順

- ユーザー **sysadmin** の補助グループに新規グループが追加されていることを確認するには、以下を使用します。

```
# groups sysadmin
```

この出力では、以下が表示されます。

```
sysadmin : sysadmin system-administrators
```

11.3.4. グループディレクトリーの作成

UPG システム設定では、**グループ ID 権限の設定** (setgid) をディレクトリーに適用できます。**setgid** を使用して、ディレクトリーを共有するグループプロジェクトの管理が簡単になります。**setgid** をディレクトリーに適用すると、ディレクトリー内に作成されたファイルは、そのディレクトリーを所有するグループに自動的に割り当てられます。このグループ内の書き込みおよび実行権限があるユーザーは、対象のディレクトリーにファイルを作成、変更、および削除できるようになりました。

次のセクションでは、グループディレクトリーを作成する方法を説明します。

前提条件

- **Root** アクセス

手順

1. ディレクトリーを作成します。

```
# mkdir directory-name
```

directory-name は、ディレクトリー名に置き換えます。

2. グループを作成します。

```
# groupadd group-name
```

group-name は、グループ名に置き換えます。

3. ユーザーをグループに追加します。

```
# usermod --append -G group-name username
```

group-name はグループ名に、username はユーザー名に置き換えます。

4. ディレクトリーのユーザーとグループの所有権は、group-name グループに関連付けます。

```
# chgrp group-name directory-name
```

group-name はグループ名に、directory-name はディレクトリー名に置き換えます。

5. ファイルおよびディレクトリーを作成および修正し、**setgid** を設定してこの権限を directory-name ディレクトリー内で適用できるようにします。

```
# chmod g+rwxs directory-name
```

directory-name は、ディレクトリー名に置き換えます。

group-name グループのすべてのメンバーが、**directory-name** ディレクトリーにファイルを作成し、編集できるようになりました。新規に作成されたファイルは、**group-name** グループの所有権を保持します。

検証手順

- パーミッション設定の正確性を検証するには、以下を使用します。

```
# ls -ld directory-name
```

directory-name は、ディレクトリー名に置き換えます。

返される出力は以下のとおりです。

```
drwx**rws**r-x. 2 root **group-name** 6 Nov 25 08:45 **directory-name**
```

11.3.5. コマンドラインでのユーザーの削除

コマンドラインを使用してユーザーアカウントを削除できます。ユーザーアカウントの削除に加えて、必要に応じてユーザーデータとメタデータ (ホームディレクトリーや設定ファイルなど) を削除できます。

前提条件

- **root** アクセスがある。
- ユーザーが存在している。
- ユーザーがログアウトしていることを確認します。

```
# loginctl terminate-user user-name
```

手順

- ユーザーデータではなく、ユーザーアカウントのみを削除するには、以下を実行します。

```
# userdel user-name
```

- ユーザー、データ、およびメタデータを削除するには、以下を実行します。

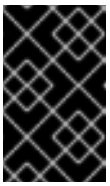
a. ユーザー、そのホームディレクトリー、メールスプール、および SELinux ユーザーマッピングを削除します。

```
# userdel --remove --selinux-user user-name
```

b. 追加のユーザーメタデータを削除します。

```
# rm -rf /var/lib/AccountsService/users/user-name
```

このディレクトリーには、ホームディレクトリーが使用可能になる前にシステムが必要とする、ユーザーに関する情報が保存されます。システムの設定によっては、ユーザーがログイン画面で認証するまで、ホームディレクトリーが利用できない可能性があります。



重要

このディレクトリーを削除せずに、後で同じユーザーを再作成した場合、再作成されたユーザーは、削除されたユーザーから継承した特定の設定を引き続き使用します。

関連情報

- **userdel(8)** man ページ

11.4. WEB コンソールでユーザーアカウントの管理

RHEL Web コンソールは、直接ターミナルにアクセスせずに、幅広い管理タスクを実行できるグラフィカルインターフェイスを提供します。たとえば、システムユーザーアカウントの追加、編集、または削除が可能です。

本セクションの内容を読むと、以下を理解できます。

- 既存のアカウントが存在する場所
- 新規アカウントの追加方法
- パスワードの有効期限の設定方法
- ユーザーセッションを終了する方法および時期

前提条件

- RHEL Web コンソールを設定しておく。詳細はRHEL Web コンソールの使用ガイド を参照してください。
- 管理者権限が割り当てられたアカウントで RHEL Web コンソールにログインしている。詳細は RHEL Web コンソールへのログイン を参照してください。

11.4.1. Web コンソールで管理されるシステムユーザーアカウント

RHEL Web コンソールに表示されているユーザーアカウントでは、以下が可能になります。

- システムにアクセスする際にユーザーを認証する
- システムへのアクセス権を設定する

RHEL Web コンソールは、システムに存在するすべてのユーザーアカウントを表示します。そのため、最初に Web コンソールにログインした直後は、ユーザーアカウントが少なくとも 1 つ表示されます。

RHEL Web コンソールにログインしたら、以下の操作を実行できます。

- 新規ユーザーアカウントの作成
- パラメーターの変更
- アカウントのロック
- ユーザーセッションの終了

11.4.2. Web コンソールで新規アカウントの追加

RHEL Web コンソールを使用して、システムにユーザーアカウントを追加し、アカウントに管理権限を設定できます。

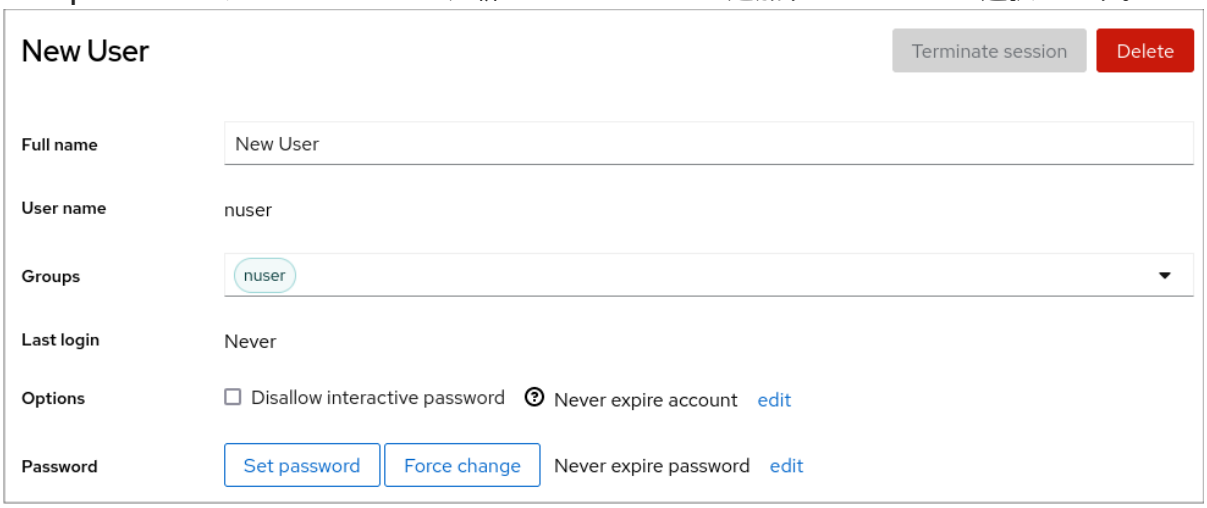
前提条件

- RHEL Web コンソールがインストールされており、アクセス可能である。詳細は、Web コンソールのインストール を参照してください。

手順

1. RHEL Web コンソールにログインします。
2. **Accounts** をクリックします。
3. **Create New Account** をクリックします。
4. **フルネーム** フィールドにユーザーの氏名を入力します。
   RHEL Web コンソールは、入力した氏名からユーザー名が自動的に作成され、**ユーザー名** フィールドに入力されます。名前の頭文字と、苗字で設定される命名規則を使用しない場合は、入力されたユーザー名を変更します。
5. **パスワード/確認** フィールドにパスワードを入力し、再度パスワードを入力します。
   フィールドの下にあるカラーバーは、入力したパスワードの強度を表し、弱いパスワードは使用できないようにします。

6. **作成** をクリックして設定を保存し、ダイアログボックスを閉じます。
7. 新規作成したアカウントを選択します。
8. **Groups** ドロップダウンメニューで、新しいアカウントに追加するグループを選択します。



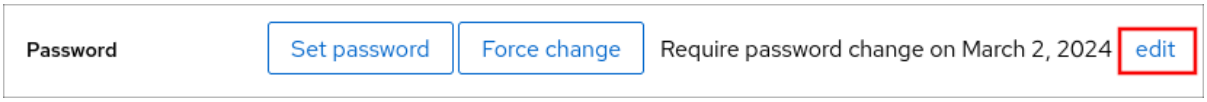
これで **アカウント** 設定に新規アカウントが表示され、認証情報を使用してシステムに接続できるようになりました。

11.4.3. Web コンソールでパスワード有効期限の強制

デフォルトでは、ユーザーアカウントのパスワードに期限はありません。定義した日数が経過したら、システムパスワードが期限切れになるように設定できます。パスワードが期限切れになると、次回のログイン時にパスワードの変更が要求されます。

手順

1. RHEL 8 Web コンソールにログインします。
2. **Accounts** をクリックします。
3. パスワードの有効期限を設定するユーザーアカウントを選択します。
4. **Password** 行の **edit** をクリックします。



5. **Password expiration** ダイアログボックスで、**Require password change every ... days**を選択し、パスワードの期限が切れるまでの日数 (正の整数) を入力します。
6. **Change** をクリックします。
Web コンソールの **Password** 行に、将来のパスワード変更リクエストの日付がすぐに表示されます。

11.4.4. Web コンソールでユーザーセッションの終了

ユーザーがシステムにログインすると、ユーザーセッションが作成されます。ユーザーセッションを終了すると、ユーザーはシステムからログアウトされます。これは、システムのアップグレードなどの、設定変更の影響を受ける管理タスクを実行する必要がある場合に便利です。

RHEL 8 Web コンソールの各ユーザーアカウントで、現在使用している Web コンソールセッション以外のセッションすべてを終了できます。これにより、システムへの不正アクセスを阻止できます。

手順

1. RHEL 8 Web コンソールにログインします。
2. **Accounts** をクリックします。
3. セッションを終了するユーザーアカウントをクリックします。
4. **Terminate Session** をクリックします。
   Terminate Session ボタンが無効になっている場合は、ユーザーがシステムにログインしていません。

   RHEL Web コンソールはセッションを終了します。

11.5. コマンドラインを使用したユーザーグループの編集

ユーザーは、ファイルおよびフォルダーに同様のアクセスを持つユーザーの論理的な集合を許可する、特定のグループセットに属します。コマンドラインから、プライマリーユーザーグループおよび補助ユーザーグループを編集して、ユーザーの権限を変更できます。

11.5.1. プライマリーユーザーグループおよび補助ユーザーグループ

グループとは、複数のユーザーアカウントを共通目的 (特定のファイルにアクセス権を与えるなど) で統合するエンティティーです。

Linux では、ユーザーグループはプライマリーまたは補助として機能できます。プライマリーグループおよび補助グループには、以下のプロパティーがあります。

プライマリーグループ

- すべてのユーザーに、常に 1 つのプライマリーグループのみが存在します。
- ユーザーのプライマリーグループは変更できます。

補助グループ

- 既存の補助グループに既存のユーザーを追加して、グループ内で同じセキュリティーおよびアクセス権限を持つユーザーを管理できます。
- ユーザーは、ゼロまたは複数の補助グループのメンバーになります。

11.5.2. ユーザーのプライマリーグループおよび補助グループのリスト表示

ユーザーのグループをリスト表示して、どのプライマリーグループおよび補助グループに属しているかを確認できます。

手順

- ユーザーのプライマリーおよび補助グループの名前を表示します。

```
$ groups user-name
```

user-name は、ユーザー名に置き換えます。ユーザー名を指定しないと、コマンドは現在のユーザーのグループメンバーシップを表示します。最初のグループはプライマリーグループで、その後に任意の補助グループが続きます。

例11.4 ユーザー sarah のグループのリスト表示

```
$ groups sarah
```

この出力では、以下が表示されます。

```
sarah : sarah wheel developer
```

ユーザー **sarah** にはプライマリーグループ **sarah** があり、補助グループ **wheel** および **developer** のメンバーになります。

例11.5 ユーザー marc のグループのリスト表示

```
$ groups marc
```

この出力では、以下が表示されます。

```
marc : marc
```

ユーザー **marc** には、プライマリーグループ **marc** のみがあり、補助グループはありません。

11.5.3. ユーザーのプライマリーグループの変更

既存ユーザーのプライマリーグループを、新しいグループに変更できます。

前提条件:

1. **root** アクセス
2. 新しいグループが存在する必要があります。

手順

- ユーザーのプライマリーグループを変更します。

```
# usermod -g group-name user-name
```

group-name を、新しいプライマリーグループの名前に置き換え、**user-name** を、ユーザーの名前に置き換えます。

注記

ユーザーのプライマリーグループを変更すると、コマンドは、ユーザーのホームディレクトリーにあるすべてのファイルのグループ所有権も、自動的に新しいプライマリーグループに変更します。ユーザーのホームディレクトリー外のファイルのグループ所有権を手動で修正する必要があります。

例11.6 ユーザーのプライマリーグループを変更する例:

ユーザー **sarah** がプライマリーグループ **sarah1** に所属しており、ユーザーのプライマリーグループを **sarah2** に変更する場合は、以下を使用します。

```
# usermod -g sarah2 sarah
```

検証手順

- ユーザーのプライマリーグループを変更したことを確認します。

```
$ groups sarah
```

この出力では、以下が表示されます。

```
sarah : sarah2
```

11.5.4. コマンドラインから補助グループにユーザーを追加

補助グループにユーザーを追加して、権限を管理したり、特定のファイルまたはデバイスへのアクセスを有効にしたりできます。

前提条件

- **root** アクセス

手順

- ユーザーの補助グループにグループを追加するには、以下を使用します。

```
# usermod --append -G group-name username
```

group-name はグループ名に、username はユーザー名に置き換えます。

例11.7 補助グループへのユーザーの追加

ユーザーの **sysadmin** をグループ **system-administrators** に追加するには、以下を使用します。

```
# usermod --append -G system-administrators sysadmin
```

検証手順

- ユーザー **sysadmin** の補助グループに新規グループが追加されていることを確認するには、以下を使用します。

```
# groups sysadmin
```

この出力では、以下が表示されます。

```
sysadmin : sysadmin system-administrators
```

11.5.5. 補助グループからユーザーの削除

補助グループから既存のユーザーを削除して、権限や、ファイルやデバイスへのアクセスを制限できます。

前提条件

- **root** アクセス

手順

- 補助グループからユーザーを削除します。

```
# gpasswd -d user-name group-name
```

user-name をユーザー名に置き換え、group-name を、補助グループの名前に置き換えます。

例11.8 補助グループからユーザーの削除

ユーザー sarah にプライマリーグループ **sarah2** があり、セカンダリーグループ **wheel** および **developers** に属し、そのユーザーをグループ **developers** から削除する場合は、次のコマンドを実行します。

```
# gpasswd -d sarah developers
```

検証手順

- セカンダリーグループの開発者からユーザー sarah を削除したことを確認します。

```
$ groups sarah
```

この出力では、以下が表示されます。

```
sarah : sarah2 wheel
```

11.5.6. ユーザーの補助グループのすべての変更

ユーザーをメンバーとして残す補助グループのリストを上書きできます。

前提条件

- **root** アクセス
- 補助グループが存在している

手順

- ユーザーの補助グループのリストを上書きします。

```
# usermod -G group-names username
```

group-names を、1 つ以上の補助グループの名前に置き換えます。ユーザーを複数の補助グループに一度に追加するには、グループ名をコンマで区切り、スペースを使用しないでください。たとえば、**wheel,developer** です。

user-name は、ユーザー名に置き換えます。

重要

ユーザーが、指定しないグループのメンバーである場合は、グループからそのユーザーが削除されます。

例11.9 ユーザーの補助グループのリストの変更

ユーザー **sarah** にプライマリーグループ **sarah2** があり、補助グループ **wheel** に属し、さらに 3 つの補助グループ **developer**、**sysadmin**、および **security** に属するユーザーにする場合は、次のコマンドを実行します。

```
# usermod -G wheel,developer,sysadmin,security sarah
```

検証手順

- 補助グループのリストが正しく設定されていることを確認します。

```
# groups sarah
```

この出力では、以下が表示されます。

```
sarah : sarah2 wheel developer sysadmin security
```

11.6. ROOT パスワードの変更およびリセット

既存の root パスワードが要件を満たさないか、パスワードを忘れた場合には、**root** ユーザーおよび root 以外のユーザーの両方として、パスワードを変更またはリセットできます。

11.6.1. root ユーザーとしての root パスワードの変更

passwd コマンドを使用して、**root** ユーザーとして **root** パスワードを変更できます。

前提条件

- **Root** アクセス

手順

- **root** パスワードを変更する場合は、次のコマンドを実行します。

```
# passwd
```

変更する前に、現在のパスワードを入力するように求められます。

11.6.2. root 以外のユーザーが root パスワードを変更またはリセット

passwd コマンドを使用して、root 以外のユーザーとして **root** パスワードを変更またはリセットできます。

前提条件

- root 以外のユーザーとしてログインできる。
- **wheel** 管理グループのメンバーである。

手順

- **wheel** グループに属する root ユーザー以外のユーザーとして **root** パスワードを変更またはリセットするには、以下を使用します。

```
$ sudo passwd root
```

root パスワードを変更する前に、root 以外のユーザーのパスワードを入力するように求められます。

11.6.3. 起動時の root パスワードのリセット

root 以外のユーザーとしてログインできない場合や、**wheel** 管理グループに所属しない場合は、特別な **chroot jail** 環境に切り替えることで起動時に root パスワードをリセットできます。

手順

1. システムを再起動して、GRUB 2 ブート画面で **e** キーを押して、起動プロセスを中断します。カーネルブートパラメーターが表示されます。

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
initrd ($root)/initramfs-4.18.0-80.e18.x86_64.img $tuned_initrd
```

2. **linux** で始まる行の最後に移動します。

```
linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
```

Ctrl+e を押して、行の最後に移動します。

3. **rd.break** を **linux** で始まる行の最後に追加します。

```
linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet rd.break
```

4. **Ctrl+x** を押して、変更したパラメーターでシステムを起動します。
switch_root プロンプトが表示されます。

5. ファイルシステムを書き込み可能で再マウントします。

```
mount -o remount,rw /sysroot
```

ファイルシステムは、**/sysroot** ディレクトリーに読み取り専用としてマウントされます。書き込み可能なファイルシステムとして再マウントすると、パスワードを変更できます。

6. **chroot** 環境に入ります。

```
chroot /sysroot
```

sh-4.4# プロンプトが表示されます。

7. **root** パスワードをリセットします。

```
passwd
```

コマンドラインに表示される指示に従って、**root** パスワードの変更を完了します。

8. 次回のシステム起動時に SELinux の再ラベルプロセスを有効にします。

```
touch /.autorelabel
```

9. **chroot** 環境を終了します。

```
exit
```

10. **switch_root** プロンプトを終了します。

```
exit
```

11. SELinux の再ラベルプロセスが終了するまで待機します。大規模なディスクの再ラベルには時間がかかる可能性があることに注意してください。プロセスが完了すると、システムが自動的に再起動します。

検証手順

1. **root** パスワードが正常に変更されたことを確認するには、通常のユーザーとしてログインし、ターミナルを開きます。

2. root としてインタラクティブシェルを実行します。

```
$ su
```

3. 新しい **root** パスワードを入力します。

4. 現在の実効ユーザー ID に関連付けられたユーザー名を出力します。

```
# whoami
```

返される出力は以下のとおりです。

```
root
```

第12章 SUDO アクセスの管理

システム管理者は、root 以外のユーザーに、通常 root ユーザー用に予約されている管理コマンドを実行できるようにする **sudo** アクセスを付与できます。これにより、root 以外のユーザーは、root ユーザーアカウントにログインせずに、このようなコマンドを実行できます。

12.1. SUDOERS のユーザー認可

/etc/sudoers ファイルは、どのユーザーが **sudo** コマンドを使用して他のコマンドを実行できるかを指定します。ルールは、個別のユーザーおよびユーザーグループに適用できます。エイリアスを使用して、ホスト、コマンド、さらにはユーザーのグループに対するルールをより簡単に定義することもできます。デフォルトのエイリアスは、**/etc/sudoers** ファイルの最初の部分で定義されます。

認可されていないユーザーが **sudo** を使用してコマンドを入力すると、システムは文字列 **<username> : user NOT in sudoers** を含むメッセージをジャーナルログに記録します。

デフォルトの **/etc/sudoers** ファイルは、認可の情報と例を提供します。対応する行をコメント解除すると、特定のルール例をアクティブにできます。ユーザー認可に関するセクションには、以下の概要が示されます。

```
## Next comes the main part: which users can run what software on
## which machines  (the sudoers file can be shared between multiple
## systems).
```

次の形式を使用して、新しい **sudoers** 認可を作成したり、既存の認可を変更したりできます。

```
<username> <hostname.example.com>=(<run_as_user>:<run_as_group>) <path/to/command>
```

ここでは、以下のようになります。

- **<username>** はコマンドを入力するユーザーです (例: **user1**)。値が **%** で始まる場合はグループを定義します (例: **%group1**)。
- **<hostname.example.com>** は、ルールが適用されるホストの名前です。
- セクション **(<run_as_user>:<run_as_group>)** は、コマンドを実行するユーザーまたはグループを定義します。このセクションを省略すると、**<username>** は root としてコマンドを実行できます。
- **<path/to/command>** は、コマンドへの完全な絶対パスです。また、コマンドパスの後にオプションを追加することにより、特定のオプションおよび引数を指定したコマンドのみを実行するようにユーザーを制限することもできます。オプションを指定しないと、すべてのオプションが有効な状態でコマンドを使用できます。

これらの変数のいずれかを **ALL** に置き換えることで、ルールをすべてのユーザー、ホスト、またはコマンドに適用できます。

警告

ALL ALL=(ALL) ALL などの過度に寛容なルールを使用すると、すべてのユーザーがすべてのコマンドを、いずれのホストのいずれのユーザーとしても使用できます。これは重大なセキュリティーリスクをもたらします。

! 演算子を使用して引数を負の値で指定できます。たとえば、**!root** は root 以外のすべてのユーザーを指定します。特定のユーザー、グループ、およびコマンドを許可する方が、特定のユーザー、グループ、およびコマンドを拒否するよりも安全です。これは、許可ルールにより、認可されていない新規ユーザーまたはグループもブロックされるためです。

警告

alias コマンドでコマンドの名前を変更すると、このような規則が上書きされるため、コマンドに否定的な規則を使用しないでください。

システムは、**/etc/sudoers** ファイルを最初から最後まで読み取ります。したがって、ファイルにユーザーの複数のエントリーが含まれている場合、エントリーは順番に適用されます。値が競合する場合は、最も具体的な一致ではない場合でも、システムは最後の一致を使用します。

システム更新中にルールを保持し、エラーを簡単に修正するには、ルールを **/etc/sudoers** ファイルに直接入力するのではなく、**/etc/sudoers.d/** ディレクトリーに新しいファイルを作成して、新しいルールを入力します。システムは、**/etc/sudoers** ファイル内で以下の行に到達する際に、**/etc/sudoers.d** ディレクトリー内のファイルを読み取ります。

```
#includedir /etc/sudoers.d
```

この行の頭にある番号記号 (**#**) は構文の一部であり、行がコメントであることを意味するものではありません。そのディレクトリー内のファイル名にはピリオドを使用することができません。また、末尾にチルダ (~) を使用することもできません。

関連情報

- **sudo(8)** および **sudoers(5)** の man ページ

12.2. ユーザーへの SUDO アクセス権限の付与

システム管理者は、root 以外のユーザーに **sudo** アクセスを付与することで、管理コマンドの実行を許可できます。**sudo** コマンドは、root ユーザーのパスワードを使用せずに、管理アクセスを持つユーザーを提供する方法です。

ユーザーが管理コマンドを実行する必要がある場合には、コマンドの前に **sudo** を指定してください。ユーザーがコマンドに対する認可を持っている場合、コマンドは root であるかのように実行されます。

以下の制限事項に注意してください。

- **/etc/sudoers** 設定ファイルにリスト表示されているユーザーのみが **sudo** コマンドを使用できます。
- コマンドは、root シェルではなく、ユーザーのシェルで実行されます。

前提条件

- システムへの root アクセス権があります。

手順

1. root で **/etc/sudoers** ファイルを開きます。

```
# visudo
```

   /etc/sudoers ファイルは、**sudo** コマンドで適用されるポリシーを定義します。

2. **/etc/sudoers** ファイルで、**wheel** 管理グループのユーザーに **sudo** アクセスを付与する行を見つけます。

```
## Allows people in group wheel to run all commands
%wheel        ALL=(ALL)       ALL
```

3. **%wheel** で始まる行が番号記号 (**#**) でコメントアウトされていないことを確認してください。
4. 変更を保存し、エディターを終了します。
5. **sudo** アクセスを付与するユーザーを、**wheel** 管理グループに追加します。

```
 # usermod --append -G wheel <username>
```

   <username> は、ユーザー名に置き換えます。

検証手順

- ユーザーが **wheel** 管理グループに含まれていることを確認します。

```
# id <username>
uid=5000(<username>) gid=5000(<username>) groups=5000(<username>),10(wheel)
```

関連情報

- **sudo(8)**、**visudo(8)**、および **sudoers(5)** の man ページ

12.3. 非特権ユーザーが特定のコマンドを実行できるようにする

管理者は、**/etc/sudoers.d/** ディレクトリーにポリシーを設定することで、非特権ユーザーが特定のワークステーションに特定のコマンドを入力できるようにすることができます。

前提条件

- システムへの root アクセス権があります。

手順

1. root で、**/etc/** の下に新しい **sudoers.d** ディレクトリーを作成します。

```
# mkdir -p /etc/sudoers.d/
```

2. **/etc/sudoers.d** ディレクトリーに新規ファイルを作成します。

```
# visudo -f /etc/sudoers.d/<filename>
```

ファイルが自動的に開きます。

3. 次の行を **/etc/sudoers.d/<filename>** ファイルに追加します。

```
<username> <hostname.example.com> = (<run_as_user>:<run_as_group>)
<path/to/command>
```

- **<username>** は、ユーザー名に置き換えます。
- **<hostname.example.com>** は、ホストの URL に置き換えます。
- **(<run_as_user>:<run_as_group>)** は、コマンドを実行できるユーザーまたはグループに置き換えます。このセクションを省略すると、**<username>** は root としてコマンドを実行できます。
- **<path/to/command>**は、コマンドへの完全な絶対パスに置き換えます。また、コマンドパスの後にオプションを追加することにより、特定のオプションおよび引数を指定したコマンドのみを実行するようにユーザーを制限することもできます。オプションを指定しないと、すべてのオプションが有効な状態でコマンドを使用できます。
- 同じホストで 2 つ以上のコマンドを 1 行で許可するには、コンマで区切り、コンマの後にスペースを付けることができます。
  たとえば、**user1** が **host1.example.com** で **dnf** および **reboot** コマンドを実行できるようにするには、**user1 host1.example.com = /bin/dnf, /sbin/reboot** と入力します。

4. ユーザーが **sudo** 特権の使用を試みるたびにメール通知を受け取るには、以下の行をファイルに追加します。

```
Defaults    mail_always
Defaults    mailto="<email@example.com>"
```

5. 変更を保存し、エディターを終了します。

検証

1. ユーザーが **sudo** 特権でコマンドを実行できるかどうかを確認するには、アカウントを切り替えます。

```
# su <username> -
```

2. ユーザーとして、**sudo** コマンドを使用してコマンドを入力します。

```
$ sudo <command>
[sudo] password for <username>:
```

ユーザーの **sudo** パスワードを入力します。

3. 特権が正しく設定されている場合、システムはコマンドとオプションのリストを表示します。たとえば、**dnf** コマンドを使用すると、次の出力が表示されます。

```
...
usage: dnf [options] COMMAND
...
```

システムが **<username> is not in the sudoers file. This incident will be reported** というエラーメッセージを返した場合、**/etc/sudoers.d/** に **<username>** のファイルが存在しません。

システムが **<username> is not allowed to run sudo on <host.example.com>** というエラーメッセージを返した場合、設定が正しく完了していません。root としてログインしていること、および設定が正しく実行されていることを確認してください。

システムが **Sorry, user <username> is not allowed to execute '<path/to/command>' as root on <host.example.com>.** というエラーメッセージを返した場合、コマンドがユーザーのルールで正しく定義されていません。

関連情報

- **sudo(8)**、**visudo(8)**、および **sudoers(5)** の man ページ

第13章 ファイルシステムの権限の管理

ファイルシステムの権限は、ユーザーおよびグループアカウントがファイルの内容を読み取り、変更し、実行する権限、およびディレクトリーに入る権限を制御します。権限を慎重に設定して、不正アクセスからデータを保護します。

13.1. ファイル権限の管理

すべてのファイルまたはディレクトリーには、以下のレベルの所有権があります。

- ユーザー所有者 (**u**)。
- グループの所有者 (**g**)。
- その他 (**o**)。

各所有権レベルには、以下のパーミッションを割り当てることができます。

- 読み取り (**r**)。
- 書き込み (**w**)。
- 実行 (**x**)。

ファイルの execute 権限があると、そのファイルを実行するできることに注意してください。ディレクトリーの実行権限では、ディレクトリーのコンテンツにアクセスできますが、実行はできません。

新規ファイルまたはディレクトリーが作成されると、デフォルトのパーミッションセットが自動的に割り当てられます。ファイルまたはディレクトリーのデフォルトパーミッションは、以下の 2 つの要素に基づいています。

- 基本パーミッション。
- **ユーザーのファイル作成モードマスク** (umask)

13.1.1. ベースファイルのパーミッション

新規ファイルまたはディレクトリーが作成されるたびに、基本パーミッションが自動的に割り当てられます。ファイルまたはディレクトリーの基本パーミッションは、**シンボリック** または **8 進数** の値で表すことができます。

パーミッション	**シンボリック値**	**8 進数値**
パーミッションなし	---	0
実行	--x	1
書き込み	-w-	2
書き込みおよび実行	-wx	3
読み取り	r--	4

読み取りおよび実行	r-x	5
読み取りおよび書き込み	rw-	6
読み取り、書き込み、実行	rwx	7

ディレクトリーの基本パーミッションは **777** (**drwxrwxrwx**) で、すべてのユーザーに読み取り、書き込み、実行権限を付与します。つまり，ディレクトリーの所有者、グループ、その他のユーザーはディレクトリーのコンテンツ表示、そのディレクトリーでの項目の作成、削除、編集、サブディレクトリーへの移動が可能です。

ディレクトリー内の個別ファイルには、ディレクトリーに無制限にアクセスできるにも拘らず、編集ができない独自のパーミッションが割り当てられている可能性があります。

ファイルの基本パーミッションは **666** (**-rw-rw-rw-**) で、すべてのユーザーに読み取りおよび書き込み権限を付与します。ファイルの所有者、グループ、その他のユーザーは、ファイルの読み取りと編集が可能です。

例13.1 ファイルのパーミッション

ファイルに以下のパーミッションがある場合:

```
$ ls -l
-rwxrw----. 1 sysadmins sysadmins 2 Mar 2 08:43 file
```

- **-** ファイルであることを示します。
- **rwx** は、ファイルの所有者にファイルの読み取り、書き込み、実行のパーミッションがあることを示します。
- **rw-** は、グループに読み取りと書き込みのパーミッションがあるが、ファイルの実行はできません。
- **---** は、他のユーザーにファイルの読み取り、書き込み、実行のパーミッションがないことを示します。
- **.** は、SELinux セキュリティーコンテキストがファイルに設定されていることを示しています。

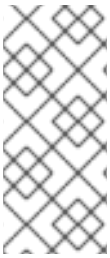
例13.2 ディレクトリーのパーミッション

ディレクトリーに以下のパーミッションがある場合:

```
$ ls -dl directory
drwxr-----. 1 sysadmins sysadmins 2 Mar 2 08:43 directory
```

- **d** は、ディレクトリーであることを示します。
- **rwx** は、ディレクトリーの所有者にディレクトリーの内容を読み取り、書き込み、およびアクセスするパーミッションがあることを示します。
  ディレクトリーの所有者は、ディレクトリー内のアイテム (ファイル、サブディレクトリー) を表示して、アイテムのコンテンツへのアクセスや変更が可能です。

- **r-x** は、そのグループがディレクトリーの内容を読み取るパーミッションを持っているが、書き込み (新しいエントリーの作成やファイルの削除) はできないことを示します。**x** パーミッションは、**cd** コマンドを使用してディレクトリーにアクセスできることを意味します。
- **---** は、他のユーザーがディレクトリーの内容の読み取り、書き込み、またはアクセスパーミッションがないことを示します。
  ディレクトリーのユーザー所有者またはグループ所有者ではない場合に、そのディレクトリーのアイテムを表示したり、アイテムの情報にアクセスしたり、変更したりできません。
- **.** は、SELinux セキュリティーコンテキストがディレクトリーに設定されていることを示しています。



注記

ファイルまたはディレクトリーに自動的に割り当てられる基本パーミッションは、最終的にファイルまたはディレクトリーに指定されるデフォルトのパーミッション **ではありません**。ファイルまたはディレクトリーを作成すると、**umask** により基本パーミッションが変更されます。基本パーミッションと **umask** の組み合わせにより、ファイルおよびディレクトリーのデフォルトパーミッションが作成されます。

13.1.2. ユーザーのファイル作成モードマスク

ユーザーファイル作成モードマスク (**umask**) は、新規作成ファイルおよびディレクトリーにファイル権限を設定する方法を制御する変数です。**umask** は、基本パーミッション値からパーミッションを自動的に削除して、Linux システムの全体的なセキュリティーを強化します。**umask** は、**シンボリック** 値または **8 進数** の値で表すことができます。

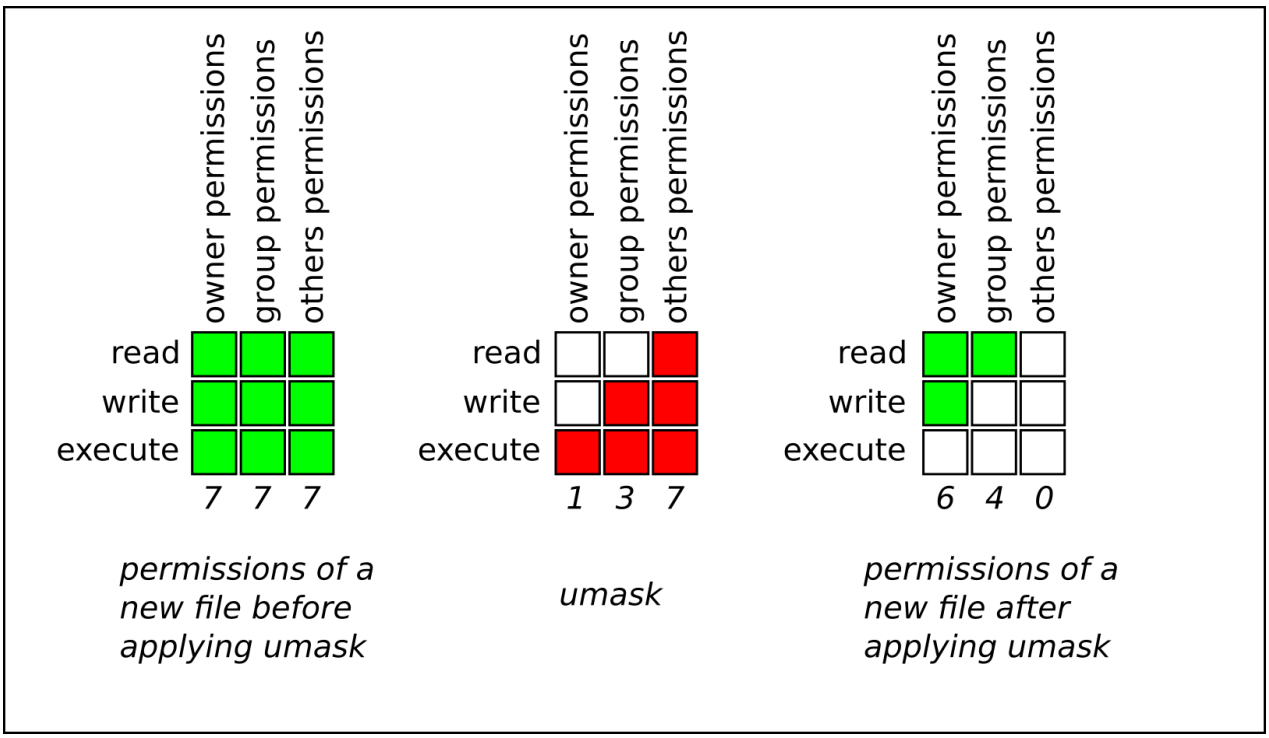
パーミッション	シンボリック値	8 進数値
読み取り、書き込み、実行	rwx	0
読み取りおよび書き込み	rw-	1
読み取りおよび実行	r-x	2
読み取り	r--	3
書き込みおよび実行	-wx	4
書き込み	-w-	5
実行	--x	6
権限なし	---	7

標準ユーザーのデフォルトの **umask** は **0002** です。**root** ユーザーのデフォルトの **umask** は **0022** です。

umask の最初の数字は、特別なパーミッション (スティッキービット) を表します。umask の最後の 3 桁はユーザーの所有者 (u)、グループの所有者 (g) およびその他 (o) からそれぞれ削除したパーミッションを表します。

例13.3 ファイルの作成時に umask を適用

以下の例では、umask の 8 進数値 **0137** が基本パーミッション **777** に適用され、デフォルトパーミッション **640** のファイルが作成されます。



13.1.3. デフォルトのファイル権限

デフォルトのパーミッションは、新規作成ファイルおよびディレクトリーに対して自動的に設定されます。デフォルトのパーミッションの値は、umask を基本パーミッションに適用して決定します。

例13.4 標準ユーザーが作成したディレクトリーのデフォルト権限

標準ユーザー が新しい **ディレクトリー** を作成すると、umask は **002** (**rwxrwxr-x**) に、ディレクトリーの基本パーミッションは **777** (**rwxrwxrwx**) に設定されます。これにより、デフォルトのパーミッションが **775** (**drwxrwxr-x**) になります。

	シンボリック値	**8 進数値**
基本パーミッション	rwxrwxrwx	777
Umask	rwxrwxr-x	002
デフォルトパーミッション	rwxrwxr-x	775

このパーミッションでは、ディレクトリーの所有者とグループはディレクトリーのコンテンツの表示、ディレクトリーでのアイテムの作成、削除、編集、サブディレクトリーへの移動が可能です。他のユーザーはディレクトリーの内容を表示して、サブディレクトリーに移動することしかできません。

例13.5 標準ユーザーが作成したファイルのデフォルト権限

標準ユーザー が新しい **ファイル** を作成すると、umask は **002** (**rwxrwxr-x**) に、ファイルの基本パーミッションは **666** (**rw-rw-rw-**) に設定されます。これにより、デフォルトのパーミッション **664** (**-rw-rw-r--**) になります。

	シンボリック値	**8 進数値**
基本パーミッション	rw-rw-rw-	666
Umask	rwxrwxr-x	002
デフォルトパーミッション	rw-rw-r--	664

このパーミッションでは、ファイルの所有者とグループはファイルの読み取りと編集が可能ですが、他のユーザーはこのファイルの読み取りしかできません。

例13.6 root ユーザーが作成したディレクトリーのデフォルト権限

root ユーザー が新規 **ディレクトリー** を作成すると、umask は **022** (**rwxr-xr-x**) に、ディレクトリーの基本パーミッションは **777** (**rwxrwxrwx**) に設定されます。これにより、デフォルトのパーミッションが **755** (**rwxr-xr-x**) になります。

	シンボリック値	**8 進数値**
基本パーミッション	rwxrwxrwx	777
Umask	rwxr-xr-x	022
デフォルトパーミッション	rwxr-xr-x	755

このパーミッションでは、ディレクトリーの所有者はディレクトリーの内容の表示、ディレクトリー内のアイテムの作成、削除、編集、サブディレクトリーへの移動が可能です。グループとその他のユーザーは、ディレクトリーの内容の表示とサブディレクトリーの移動のみが可能です。

例13.7 root ユーザーが作成したファイルのデフォルト権限

root ユーザー が新規 **ファイル** を作成すると、umask は **022** (**rwxr-xr-x**) に、ファイルの基本パーミッションは **666** (**rw-rw-rw-**) に設定されます。これにより、デフォルトのパーミッションは **644** (**-rw-r—r--**) になりました。

	シンボリック値	**8 進数値**

基本パーミッション	rw-rw-rw-	666
Umask	rwxr-xr-x	022
デフォルトパーミッション	rw-r--r--	644

このパーミッションでは、ファイルの所有者はファイルを読み取りと編集が可能ですが、グループや他のユーザーはこのファイルの読み取りのみが可能です。

注記

セキュリティー上の理由から、通常のファイルに umask が **000** (**rwxrwxrwx**) に設定されていても、デフォルトで実行権限がありません。ただし、ディレクトリーは実行権限を持つ状態で作成できます。

13.1.4. シンボリック値を使用したファイル権限の変更

chmod ユーティリティーにシンボリック値 (組み合わせ文字および記号) を付けて、ファイルまたはディレクトリーのファイルのパーミッションを変更できます。

以下の **パーミッション** を割り当てることができます。

- 読み取り (**r**)
- 書き込み (**w**)
- 実行 (**x**)

パーミッションは、以下の **レベルの所有権** に割り当てることができます。

- ユーザー所有者 (**u**)
- グループ所有者 (**g**)
- その他 (**o**)
- すべて (**a**)

パーミッションを追加または削除するには、以下の **記号** を使用できます。

- **+**: 既存のパーミッションの上にパーミッションを追加します。
- **-:** 既存のパーミッションからパーミッションを削除します。
- **=**: 既存のパーミッションを削除し、新しいパーミッションを明示的に定義します。

手順

- ファイルまたはディレクトリーのパーミッションを変更するには、以下を使用します。

```
$ chmod <level><operation><permission> file-name
```

<level> は、パーミッションを設定する 所有権のレベル に置き換えます。**<operation>** は、署名 の 1 つに置き換えます。**<permission>** は、割り当てる パーミッション に置き換えます。**file-name** は、ファイルまたはディレクトリーの名前に置き換えます。たとえば、すべてのユーザーに読み取り、書き込み、実行 (**rwx**) **my-script.sh** のパーミッションを付与するには、**chmod a=rwx my-script.sh** コマンドを使用します。

詳細は ベースファイルのパーミッション を参照してください。

検証手順

- 特定のファイルのパーミッションを表示するには、以下を使用します。

```
$ ls -l file-name
```

file-name は、ファイルの名前に置き換えます。

- 特定のディレクトリーのパーミッションを表示するには、以下を使用します。

```
$ ls -dl directory-name
```

directory-name は、ディレクトリー名に置き換えます。

- 特定のディレクトリー内の全ファイルのパーミッションを表示するには、以下を使用します。

```
$ ls -l directory-name
```

directory-name は、ディレクトリー名に置き換えます。

例13.8 ファイルおよびディレクトリーの権限の変更

- **my-file.txt** のパーミッションを **-rw-rw-r--** から **-rw------** に変更するには以下を使用します。

1. **my-file.txt** の現在のパーミッションを表示します。

```
$ ls -l my-file.txt
-rw-rw-r--. 1 username username 0 Feb 24 17:56 my-file.txt
```

2. グループ所有者 (g) およびその他のファイル (**o**) からファイルを読み取り、書き込み、**実行** (**rwx**) するパーミッションを削除します。

```
$ chmod go= my-file.txt
```

パーミッションを等号 (**=**) の後ろに指定していない場合には自動的に無視される点に注意してください。

3. **my-file.txt** のパーミッションが正しく設定されていることを確認します。

```
$ ls -l my-file.txt
-rw-------. 1 username username 0 Feb 24 17:56 my-file.txt
```

- **my-directory** のパーミッションを **drwxrwx---** から **drwxrwxr-x** に変更するには、以下を使用します。

1. **my-directory** の現在のパーミッションを表示します。

```
$ ls -dl my-directory
drwxrwx---. 2 username username 4096 Feb 24 18:12 my-directory
```

2. すべてのユーザー (**a**) に対する読み取りおよび実行 (**r-x**) アクセスを追加します。

```
$ chmod o+rx my-directory
```

3. **my-directory** とそのコンテンツが正しく設定されていることを確認します。

```
$ ls -dl my-directory
drwxrwxr-x. 2 username username 4096 Feb 24 18:12 my-directory
```

13.1.5. 8 進数値を使用したファイル権限の変更

chmod ユーティリティーに 8 進数値 (数値) を指定して chmod ユーティリティーを使用し、ファイルまたはディレクトリーのファイルパーミッションを変更できます。

手順

- 既存のファイルまたはディレクトリーのファイル権限を変更するには、以下を使用します。

```
$ chmod octal_value file-name
```

file-name は、ファイルまたはディレクトリーの名前に置き換えます。**octal_value** は 8 進数値に置き換えます。詳細は ベースファイルのパーミッション を参照してください。

13.2. アクセス制御リストの管理

各ファイルおよびディレクトリーには、ユーザー所有者とグループ所有者を一度に指定できます。他のファイルやディレクトリーを非公開のままにし、別のユーザーまたはグループが所有する特定のファイルまたはディレクトリーにアクセスできるようなユーザーのパーミッションを付与する場合には、Linux アクセス制御リスト (ACL) を使用できます。

13.2.1. 現在のアクセス制御リストの表示

getfacl ユーティリティーを使用して、現在の ACL を表示できます。

手順

- 特定のファイルまたはディレクトリーの現在の ACL を表示するには、以下を使用します。

```
$ getfacl file-name
```

file-name は、ファイルまたはディレクトリーの名前に置き換えます。

13.2.2. アクセス制御リストの設定

setfacl ユーティリティーを使用して、ファイルまたはディレクトリーに ACL を設定できます。

前提条件

- **root** アクセス

手順

- ファイルまたはディレクトリーに ACL を設定するには、以下を使用します。

```
# setfacl -m u:username:symbolic_value file-name
```

username はユーザー名に、symbolic_value はシンボリック値に、file-name はファイルまたはディレクトリーの名前に置き換えます。詳細は、 **setfacl** の man ページを参照してください。

例13.9 グループプロジェクトのパーミッションの変更

以下の例では、**root** グループに所属する **root** ユーザーが所有する **group-project** ファイルのパーミッションを修正する方法を説明します。このファイルは以下のように設定します。

- 誰にも実行権限がない。
- ユーザー **andrew** のパーミッションは **rw-** である。
- **susan** ユーザーのパーミッションは **---** である。
- 他のユーザーのパーミッションは **r--** である。

手順

```
# setfacl -m u:andrew:rw- group-project
# setfacl -m u:susan:--- group-project
```

検証手順

- ユーザー **andrew** に **rw-** パーミッションがあり、ユーザー **susan** には **---** パーミッションがあり、その他のユーザーに **r--** パーミッションがあることを確認するには、以下を実行します。

```
$ getfacl group-project
```

返される出力は以下のとおりです。

```
# file: group-project
# owner: root
# group: root
user:andrew:rw-
user:susan:---
group::r--
mask::rw-
other::r--
```

13.3. UMASK の管理

umask ユーティリティーを使用して、umask の現在の値またはデフォルト値を表示、設定、または変更できます。

13.3.1. umask の現在の値の表示

umask ユーティリティーを使用して、**umask** の現在の値をシンボリックモードまたは 8 進数モードで表示できます。

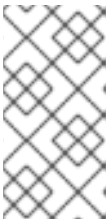
手順

- **umask** の現在の値をシンボリックモードで表示するには、以下のコマンドを使用します。

```
$ umask -S
```

- **umask** の現在の値を 8 進法で表示するには、以下のコマンドを使用します。

```
$ umask
```



注記

umask を 8 進法で表示するには、4 桁の数字 (**0002** または **0022**) で表示される場合があります。**umask** の最初の数字は、特殊ビット (スティッキービット、SGID ビット、または SUID ビット) を表します。最初の数字を **0** に設定すると、特別なビットは設定されません。

13.3.2. デフォルトの bash umask の表示

bash、**ksh**、**zsh**、**tcsh** などの多くのシェルを使用できます。これらのシェルはログインまたは nologin シェルとして動作します。ネイティブまたは GUI 端末を開いて、ログインシェルを呼び出すことができます。

ログインシェルまたは nologin シェルのどちらでコマンドを実行しているかを確認するには、**echo $0** コマンドを使用します。

例13.10 ログインまたは nologin bash シェルで作業しているかどうかの確認

- **echo $0** コマンドの出力が **bash** を返す場合、nologin シェルでコマンドを実行します。

```
$ echo $0
bash
```

  nologin シェルのデフォルトの **umask** は、**/etc/bashrc** 設定ファイルで設定します。

- **echo $0** コマンドの出力が **-bash** を返す場合は、ログインシェルでコマンドを実行します。

```
# echo $0
-bash
```

  ログインシェルのデフォルトの **umask** は **/etc/profile** 設定ファイルで設定します。

手順

- nologin シェルのデフォルトの **bash umask** を表示するには、以下のコマンドを使用します。

```
$ grep umask /etc/bashrc
```

返される出力は以下のとおりです。

```
# By default, we want umask to get set. This sets it for non-login shell.
      umask 002
      umask 022
```

- ログインシェルのデフォルトの **bash umask** を表示するには、以下のコマンドを使用します。

```
$ grep umask /etc/profile
```

返される出力は以下のとおりです。

```
# By default, we want umask to get set. This sets it for login shell
      umask 002
      umask 022
```

13.3.3. シンボリック値を使用した umask の設定

シンボリック値 (組み合わせ文字および記号) を指定して **umask** ユーティリティーを使用し、現在のシェルセッションの **umask** を設定できます。

以下の **パーミッション** を割り当てることができます。

- 読み取り (**r**)
- 書き込み (**w**)
- 実行 (**x**)

パーミッションは、以下の **レベルの所有権** に割り当てることができます。

- ユーザー所有者 (**u**)
- グループ所有者 (**g**)
- その他 (**o**)
- すべて (**a**)

パーミッションを追加または削除するには、以下の **記号** を使用できます。

- **+**: 既存のパーミッションの上にパーミッションを追加します。
- **-:** 既存のパーミッションからパーミッションを削除します。
- **=**: 既存のパーミッションを削除し、新しいパーミッションを明示的に定義します。

注記

パーミッションを等号 (**=**) の後ろに指定していない場合には自動的に無視されます。

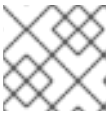
手順

- 現在のシェルセッションの **umask** を設定するには、以下のコマンドを使用します。

```
$ umask -S <level><operation><permission>
```

<level> は、umask を設定する 所有権のレベル に置き換えます。**<operation>** は、署名 の 1 つに置き換えます。**<permission>** は、割り当てる パーミッション に置き換えます。たとえば、**umask** を **u=rwx,g=rwx,o=rwx** に設定するには **umask -S a=rwx** を使用します。

詳細は、ユーザーファイル作成モードを参照してください。



注記

umask は、現在のシェルセッション限定で有効になります。

13.3.4. 8 進数値を使用した umask の設定

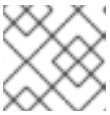
8 進数値 (数字) を指定して **umask** ユーティリティーを使用し、現在のシェルセッションの **umask** を設定できます。

手順

- 現在のシェルセッションの **umask** を設定するには、以下のコマンドを使用します。

```
$ umask octal_value
```

octal_value は 8 進数値に置き換えます。詳細は、 ユーザーファイル作成モードマスクを参照してください。



注記

umask は、現在のシェルセッション限定で有効になります。

13.3.5. nologin シェルのデフォルト umask の変更

/etc/bashrc ファイルを変更して、標準ユーザーのデフォルトの **bash** umask を変更できます。

前提条件

- **root** アクセス

手順

1. **root** として、エディターで **/etc/bashrc** ファイルを開きます。
2. 以下のセクションを変更して、新しいデフォルトの **bash** umask を設定します。

```
if [ $UID -gt 199 ] && [ “id -gn” = “id -un” ]; then
   umask 002
else
   umask 022
fi
```

umask (**002**) のデフォルト値を別の進数値に置き換えます。詳細は、ユーザーファイル作成モードマスクを参照してください。

3. 変更を保存し、エディターを終了します。

13.3.6. ログインシェルのデフォルト umask の変更

/etc/profile ファイルを変更して、**root** ユーザーのデフォルトの **bash** umask を変更できます。

前提条件

- **root** アクセス

手順

1. **root** として、エディターで **/etc/profile** ファイルを開きます。
2. 以下のセクションを変更して、新しいデフォルトの **bash** umask を設定します。

```
if [ $UID -gt 199 ] && [ “/usr/bin/id -gn” = “/usr/bin/id -un” ]; then
    umask 002
else
    umask 022
fi
```

umask (**022**) のデフォルト値を別の 8 進数値に置き換えます。詳細は、ユーザーファイル作成モードマスクを参照してください。

3. 変更を保存し、エディターを終了します。

13.3.7. 特定ユーザーのデフォルトの umask の変更

特定ユーザーのデフォルトの umask を変更するには、そのユーザーの **.bashrc** を変更します。

手順

- umask の 8 進数値を指定する行を、特定ユーザーの **.bashrc** ファイルに追加します。

```
$ echo 'umask octal_value' >> /home/username/.bashrc
```

octal_value は 8 進数値に、 username はユーザー名に置き換えます。詳細は、ユーザーファイル作成モードマスクを参照してください。

13.3.8. 新しく作成されたホームディレクトリーのデフォルト権限設定

新しく作成されたユーザーのホームディレクトリーのパーミッションモードは、**/etc/login.defs** ファイルを修正して変更できます。

手順

1. **root** として、エディターで **/etc/login.defs** ファイルを開きます。
2. 以下のセクションを変更して、HOME_MODE のデフォルトを新規設定します。

```
# HOME_MODE is used by useradd(8) and newusers(8) to set the mode for new
# home directories.
# If HOME_MODE is not set, the value of UMASK is used to create the mode.
HOME_MODE       0700
```

デフォルトの 8 進数値 (**0700**) を別の 8 進数値に置き換えます。選択したモードは、ホームディレクトリーのパーミッションの作成に使用されます。

3. **HOME_MODE** が設定されている場合は、変更を保存してエディターを終了します。
4. **HOME_MODE** が設定されていない場合は、**UMASK** を変更して、新しく作成されたホームディレクトリーにモードを設定します。

```
# Default initial "umask" value used by login(1) on non-PAM enabled systems.
# Default "umask" value for pam_umask(8) on PAM enabled systems.
# UMASK is also used by useradd(8) and newusers(8) to set the mode for new
# home directories if HOME_MODE is not set.
# 022 is the default value, but 027, or even 077, could be considered
# for increased privacy. There is no One True Answer here: each sysadmin
# must make up their mind.

UMASK           022
```

デフォルトの 8 進数値 (**022**) を別の 8 進数値に置き換えます。詳細は、ユーザーファイル作成モードマスクを参照してください。

5. 変更を保存し、エディターを終了します。

第14章 SYSTEMD の管理

システム管理者は、**systemd** を使用してシステムの重要な側面を管理できます。Linux オペレーティングシステムのシステムおよびサービスマネージャーとして機能する **systemd** ソフトウェアスイートは、制御、レポート、およびシステム初期化のためのツールとサービスを提供します。**systemd** の主な機能は次のとおりです。

- ブート時のシステムサービスの並行起動
- デーモンのオンデマンドアクティベーション
- 依存関係ベースのサービス制御ロジック

systemd が管理する基本オブジェクトは、システムのリソースとサービスを表す **systemd ユニット** です。**systemd** ユニットは、名前、タイプ、および特定のタスクを定義および管理する設定ファイルで構成されます。ユニットファイルを使用すると、システムの動作を設定できます。さまざまな systemd ユニットタイプの例を以下に示します。

サービス
個々のシステムサービスを制御および管理します。

ターゲット
システム状態を定義するユニットのグループを表します。

デバイス
ハードウェアデバイスとその可用性を管理します。

マウント
ファイルシステムのマウントを処理します。

Timer
タスクを特定の間隔で実行するようにスケジュールします。

注記

利用可能なすべてのユニットタイプを表示するには、以下を実行します。

```
# systemctl -t help
```

14.1. SYSTEMD のユニットファイルの場所

ユニット設定ファイルは、次のいずれかのディレクトリーにあります。

表14.1 systemd のユニットファイルの場所

ディレクトリー	説明
/usr/lib/systemd/system/	インストール済みの RPM パッケージで配布された **systemd** のユニットファイル。
/run/systemd/system/	ランタイム時に作成された **systemd** ユニットファイル。このディレクトリーは、インストール済みのサービスのユニットファイルのディレクトリーよりも優先されます。

ディレクトリー	説明
/etc/systemd/system/	**systemctl enable** コマンドを使用して作成された **systemd** ユニットファイルと、サービスを拡張するために追加されたユニットファイル。このディレクトリーは、runtime のユニットファイルのディレクトリーよりも優先されます。

systemd のデフォルト設定はコンパイル中に定義され、**/etc/systemd/system.conf** ファイルで確認できます。このファイルを編集すると、**systemd** ユニットの値をシステム全体でオーバーライドしてデフォルト設定を変更できます。

たとえば、タイムアウト制限のデフォルト値 (90 秒) を上書きする場合は、**DefaultTimeoutStartSec** パラメーターを使用して、上書きする値を秒単位で入力します。

```
DefaultTimeoutStartSec=required value
```

14.2. SYSTEMCTL によるシステムサービス管理

システム管理者は、**systemctl** ユーティリティーを使用してシステムサービスを管理できます。実行中のサービスの起動、停止、再起動、ブート時に起動するサービスの有効化と無効化、利用可能なサービスのリスト表示、システムサービスのステータスの表示など、さまざまなタスクを実行できます。

14.2.1. システムサービスのリスト表示

現在ロードされているすべてのサービスユニットをリストし、使用可能なすべてのサービスユニットのステータスを表示できます。

手順

systemctl コマンドを使用して、次のタスクのいずれかを実行します。

- 現在ロードされているすべてのサービスユニットをリストします。

```
$ systemctl list-units --type service
UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
abrt-ccpp.service       loaded active exited  Install ABRT coredump hook
abrt-oops.service       loaded active running ABRT kernel log watcher
abrtd.service           loaded active running ABRT Automated Bug Reporting Tool
...
systemd-vconsole-setup.service loaded active exited  Setup Virtual Console
tog-pegasus.service     loaded active running OpenPegasus CIM Server

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, or a generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

46 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'
```

デフォルトでは、**systemctl list-units** コマンドは、アクティブなユニットのみを表示します。このコマンドは、サービスユニットファイルごとに、次のパラメーターの概要を提供します。

UNIT

サービスユニットのフルネーム

LOAD

設定ファイルのロード状態

ACTIVE または SUB

現在の高レベルおよび低レベルのユニットファイルのアクティベーション状態

DESCRIPTION

ユニットの目的と機能の簡単な説明

- **--all** または **-a** コマンドラインオプションを指定して次のコマンドを使用し、**ロードされたすべてのユニットを状態に関係なく** をリスト表示します。

```
$ systemctl list-units --type service --all
```

- 利用可能なすべてのサービスユニットのステータス (**enabled** または **disabled**) をリスト表示します。

```
$ systemctl list-unit-files --type service
UNIT FILE                              STATE
abrt-ccpp.service                        enabled
abrt-oops.service                        enabled
abrtd.service                          enabled
...
wpa_supplicant.service                       disabled
ypbind.service                         disabled

208 unit files listed.
```

このコマンドでは、サービスユニットごとに以下を表示します。

UNIT FILE

サービスユニットのフルネーム

STATE

サービスユニットがブート時に自動的に起動するかどうかの情報

関連情報

- システムサービスステータスの表示

14.2.2. システムサービスステータスの表示

サービスユニットを検査して詳細情報を取得し、サービスの状態 (ブート時の起動が有効かどうか、現在実行中かどうか) を確認できます。特定のサービスユニットの前または後に起動するように指定されたサービスを表示することもできます。

手順

systemctl コマンドを使用して、次のタスクのいずれかを実行します。

- システムサービスに対応するサービスユニットに関する詳細情報を表示します。

```
$ systemctl status <name>.service
```

<name> は、確認するサービスユニットの名前 (**gdm** など) に置き換えます。

このコマンドでは、以下の情報が表示されます。

- 選択したサービスユニットの名前とその後に続く簡単な説明
- 利用可能なサービスユニットの情報 で説明されている 1 つ以上のフィールド
- サービスユニットの実行: ユニットが **root** ユーザーによって実行される場合
- 最新のログエントリー

表14.2 利用可能なサービスユニットの情報

フィールド	説明
Loaded	サービスユニットがロードされているかどうかの説明、ユニットファイルへの絶対パス、およびブート時のユニット起動が有効かどうかの注記。
Active	サービスユニットが実行中かどうかの説明と、タイムスタンプ
Main PID	プロセス ID と、対応するシステムサービスの名前。
ステータス	対応するシステムサービスに関する追加情報
Process	関連プロセスに関する追加情報
CGroup	関連するコントロールグループ (**cgroups**) に関する追加情報。

例14.1 サービスステータスの表示

GNOME Display Manager のサービスユニット名は **gdm.service** になります。このサービスユニットの現在のステータスを確認するには、シェルプロンプトで次のコマンドを実行します。

```
# systemctl status gdm.service
gdm.service - GNOME Display Manager
   Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled)
   Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min ago
 Main PID: 1029 (gdm)
   CGroup: /system.slice/gdm.service
           ├─1029 /usr/sbin/gdm
           └─1047 /usr/bin/Xorg :0 -background none -verbose -auth /r...

Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.
```

- 特定のサービスユニットが実行中であることを確認します。

```
$ systemctl is-active <name>.service
```

- 特定のサービスユニットのブート時起動が有効かどうかを確認します。

```
$ systemctl is-enabled <name>.service
```

注記

systemctl is-active および **systemctl is-enabled** コマンドは、指定したサービスユニットが実行中または有効な場合に、終了ステータス **0** を返します。

- 指定したサービスユニットの前に **systemd** がどのサービスの起動を指示するかを確認します。

```
# systemctl list-dependencies --after <name>.service
```

たとえば、**gdm** の前に起動するサービスのリストを表示するには、次のように入力します。

```
# systemctl list-dependencies --after gdm.service
gdm.service
├─dbus.socket
├─getty@tty1.service
├─livesys.service
├─plymouth-quit.service
├─system.slice
├─systemd-journald.socket
├─systemd-user-sessions.service
└─basic.target
[output truncated]
```

- 指定したサービスユニットの後に **systemd** がどのサービスの起動を指示するかを確認します。

```
# systemctl list-dependencies --before <name>.service
```

たとえば、**gdm** の後に起動するように **systemd** が指示するサービスのリストを表示するには、次のように入力します。

```
# systemctl list-dependencies --before gdm.service
gdm.service
├─dracut-shutdown.service
├─graphical.target
│ ├─systemd-readahead-done.service
│ ├─systemd-readahead-done.timer
│ └─systemd-update-utmp-runlevel.service
└─shutdown.target
  ├─systemd-reboot.service
  └─final.target
    └─systemd-reboot.service
```

関連情報

- システムサービスのリスト表示

14.2.3. システムサービスの起動

start コマンドを使用すると、現在のセッションでシステムサービスを起動できます。

前提条件

- Root アクセス

手順

- 現在のセッションでシステムサービスを起動します。

```
# systemctl start <name>.service
```

<name> は、起動するサービスユニットの名前 (**httpd.service** など) に置き換えます。

注記

systemd には、サービス間で正と負の依存関係が存在します。特定のサービスを起動するとき、別のサービスを 1 つまたは複数開始 (**正の依存関係**)、あるいはサービスを 1 つまたは複数停止 (**負の依存関係**) することが必要となる場合があります。

新しいサービスの起動を試みると、ユーザーに明示的な通知なしに、**systemd** がすべての依存関係を自動的に解決します。つまり、サービスを実行していて、負の依存関係にある別のサービスを起動しようとすると、最初のサービスが自動的に停止します。

たとえば、**postfix** サービスを実行している時に **sendmail** サービスを起動すると、**systemd** は、自動的に **postfix** を停止します。この 2 つのサービスは競合するため、同じポートでは実行できません。

関連情報

- **systemctl(1)** man ページ
- ブート時のシステムサービス起動の有効化
- システムサービスステータスの表示

14.2.4. システムサービスの停止

現行セッションでシステムサービスを停止するには、**stop** コマンドを使用します。

前提条件

- Root アクセス

手順

- システムサービスを停止します。

```
# systemctl stop <name>.service
```

<name> は、停止するサービスユニットの名前 (**bluetooth** など) に置き換えます。

関連情報

- **systemctl(1)** man ページ
- ブート時のシステムサービス起動の無効化
- システムサービスステータスの表示

14.2.5. システムサービスの再起動

restart コマンドを使用して次のアクションを実行すると、現在のセッションでシステムサービスを再起動できます。

- 現在のセッションで選択したサービスユニットを停止し、すぐに再起動する。
- 対応するサービスがすでに実行中の場合にのみ、サービスユニットを再起動する。
- システムサービスの実行を中断せずに、システムサービスの設定を再ロードする。

前提条件

- Root アクセス

手順

- システムサービスを再起動します。

```
# systemctl restart <name>.service
```

<name> は、再起動するサービスユニットの名前 (**httpd** など) に置き換えます。

注記

選択したサービスユニットが実行中でない場合には、このコマンドでこのサービスユニットが起動します。

- オプション: 対応するサービスがすでに実行中の場合にのみ、サービスユニットを再起動します。

```
# systemctl try-restart <name>.service
```

- オプション: サービスの実行を中断せずに設定を再ロードします。

```
# systemctl reload <name>.service
```

注記

システムサービスがこの機能をサポートしない場合は、このコマンドは無視されることに注意してください。このようなサービスを再起動するには、代わりに **reload-or-restart** コマンドおよび **reload-or-try-restart** コマンドを使用します。

関連情報

- **systemctl** の man ページ
- システムサービスステータスの表示

14.2.6. ブート時のシステムサービス起動の有効化

ブート時のサービスの自動起動を有効にすることができます。この変更は次回のリブート時に適用されます。

前提条件

- Root アクセス
- 有効にするサービスがマスクされていない。マスクされたサービスがある場合は、まずマスクを解除します。

```
# systemctl unmask <name>.service
```

手順

- システムの起動時にサービスが起動するようにします。

```
# systemctl enable <name>.service
```

<name> は、有効にするサービスユニット名 (**httpd** など) に置き換えます。

- オプション: 1 つのコマンドでサービスを有効にして起動することもできます。

```
# systemctl enable --now <name>.service
```

関連情報

- **systemctl (1)** man ページ
- システムサービスステータスの表示
- システムサービスの起動

14.2.7. ブート時のシステムサービス起動の無効化

システムの起動時にサービスユニットが自動的に起動しないようにすることができます。サービスを無効にすると、ブート時に起動されませんが、手動で起動できます。手動で開始できないようにサービスをマスクすることもできます。マスキングは、サービスが再度マスク解除されるまでサービスが永続的に使用できなくなるようにするサービスを無効にする方法です。

前提条件

- Root アクセス

手順

- サービスがブート時に起動するのを無効にします。

```
# systemctl disable <name>.service
```

<name> は、無効にするサービスユニットの名前 (**bluetooth** など) に置き換えます。

- オプション: サービスを永久に使用不可にする場合は、サービスをマスクします。

```
# systemctl mask <name>.service
```

このコマンドにより、**/etc/systemd/system/name.service** ファイルを、**/dev/null** へのシンボリックリンクに置き換え、実際のユニットファイルが **systemd** ファイルにアクセスできないようにします。

関連情報

- **systemctl (1)** man ページ
- システムサービスステータスの表示
- システムサービスの停止

14.3. ターゲットシステム状態でのブート

システム管理者は、システムのブートプロセスを制御し、システムがブートする状態を定義できます。これは **systemd** ターゲットと呼ばれ、特定のレベルの機能を達成するためにシステムが起動する **systemd** ユニットのセットです。systemd ターゲットの操作時には、デフォルトのターゲットの表示、実行時のターゲットの選択、デフォルトのブートターゲットの変更、緊急ターゲットまたはレスキューターゲットでのブートを行うことができます。

14.3.1. ターゲットユニットファイル

systemd ターゲットは、システムの起動時に同期ポイントとして機能する関連ユニットのグループです。**.target** ファイル拡張子で終わるターゲットユニットファイルは、**systemd** ターゲットを表します。ターゲットユニットの目的は、依存関係のチェーンでさまざまな **systemd** ユニットをグループ化することです。

以下の例を参照してください。

- グラフィカルセッションを開始するための **graphical.target unit** は、GNOME Display Manager (**gdm.service**)または Accounts Service (**accounts-daemon.service**)などのシステムサービスを開始し、**multi-user.target unit** もアクティブにします。
- 同様に、**multi-user.target** ユニットは、NetworkManager (**NetworkManager.service**)、D-Bus (**dbus.service**) といった、その他の必須システムサービスを開始し、**basic.target** という別のターゲットユニットをアクティブにします。

次の **systemd** ターゲットをデフォルトまたは現在のターゲットとして設定できます。

表14.3 一般的なsystemd ターゲット

rescue	ベースシステムにプルしてレスキューシェルを生成するユニットターゲット
multi-user	マルチユーザーシステムを設定するためのユニットターゲット
graphical	グラフィカルログイン画面を設定するためのユニットターゲット
emergency	メインコンソールで緊急シェルを起動するユニットターゲット

関連情報

- **systemd.special(7)** man ページ
- **systemd.target(5)** man ページ

14.3.2. ブート先のデフォルトターゲットの変更

システムが起動すると、**systemd** は、実際のターゲットユニットを指す **default.target** シンボリックリンクをアクティブにします。現在選択されているデフォルトのターゲットユニットは、**/etc/systemd/system/default.target** ファイルで確認できます。各ターゲットは特定のレベルの機能を表し、他のユニットをグループ化するために使用されます。さらに、ターゲットユニットはブート時に同期ポイントとして機能します。システムがブートするデフォルトターゲットを変更できます。デフォルトのターゲットユニットを設定すると、次回の再起動まで現在のターゲットは変更されません。

前提条件

- Root アクセス

手順

1. **systemd** がシステムを起動するために使用する現在のデフォルトのターゲットユニットを確認します。

```
# systemctl get-default
graphical.target
```

2. 現在ロードされているターゲットをリストします。

```
# systemctl list-units --type target
```

3. 別のターゲットユニットをデフォルトで使用するようにシステムを設定します。

```
# systemctl set-default <name>.target
```

<name> は、デフォルトで使用するターゲットユニットの名前に置き換えます。

```
Example:
# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target
```

–

4. デフォルトのターゲットユニットを確認します。

```
# systemctl get-default
multi-user.target
```

5. リブートして変更を適用します。

```
# reboot
```

関連情報

- **systemctl(1)** man ページ
- **systemd.special(7)** man ページ
- **bootup(7)** man ページ

14.3.3. 現在のターゲットの変更

実行中のシステムでは、リブートせずに現在のブートでターゲットユニットを変更できます。別のターゲットに切り替えると、**systemd** は、このターゲットが必要とするすべてのサービスとその依存関係を起動し、新しいターゲットで有効になっていないすべてのサービスを停止します。別のターゲットの分離は、現在のブートにのみ影響します。

手順

1. オプション: 現在のターゲットを決定します。

```
# systemctl get-default
graphical.target
```

2. オプション: 選択できるターゲットのリストを表示します。

```
# systemctl list-units --type target
```

注記

ユニットファイルに **AllowIsolate=yes** オプションが設定されているターゲットのみを分離できます。

3. 現在のブートで別のターゲットユニットに変更します。

```
# systemctl isolate <name>.target
```

<name> は、現在のブートで使用するターゲットユニットの名前に置き換えます。

```
Example:
# systemctl isolate multi-user.target
```

このコマンドは、**multi-user** という名前のターゲットユニットとすべての従属ユニットを起動し、他のすべてのユニットをただちに停止します。

関連情報

- **systemctl(1)** man ページ

14.3.4. レスキューモードでの起動

システムが後のターゲットにアクセスできず、通常のブートプロセスが失敗した場合に、トラブルシューティングまたは修復のためのシングルユーザー環境を提供する **レスキューモード** で起動できます。レスキューモードでは、システムはすべてのローカルファイルシステムをマウントし、特定の重要なシステムサービスを起動しようとしますが、ネットワークインターフェイスはアクティブになりません。

前提条件

- Root アクセス

手順

- レスキューモードに入るには、現行セッションで現在のターゲットを変更します。

```
# systemctl rescue

Broadcast message from root@localhost on pts/0 (Fri 2023-03-24 18:23:15 CEST):

The system is going down to rescue mode NOW!
```

注記

このコマンドは **systemctl isolate rescue.target** と似ていますが、システムに現在ログイン中の全ユーザーに情報メッセージを送信します。

systemd がメッセージを送信しないようにするには、**--no-wall** コマンドラインオプションを指定して次のコマンドを入力します。

```
# systemctl --no-wall rescue
```

トラブルシューティングの手順

システムがレスキューモードに移行できない場合は、可能な限り最小限の環境を提供する **緊急モード** でブートできます。緊急モードでは、システムは root ファイルシステムを読み込み専用でマウントし、他のローカルファイルシステムのマウントは試みません。また、ネットワークインターフェイスのアクティブ化も行わず、限定的な必須サービスのみを起動します。

14.3.5. ブートプロセスのトラブルシューティング

システム管理者は、ブート時にデフォルト以外のターゲットを選択して、ブートプロセスのトラブルシューティングを行うことができます。ブート時のターゲットの変更は、1回のブートにしか影響しません。可能な限り最小限の環境を提供する **緊急モード** で起動できます。

手順

1. システムを再起動し、通常のブートを開始する Enter キー以外の任意のキーを押してブートローダーメニューのカウントダウンを中断します。

2. 開始するカーネルエントリーにカーソルを移動します。
3. E キーを押して、現在のエントリーを編集します。
4. **linux** で始まる行の末尾に移動し、Ctrl+E を押して行の末尾にジャンプします。

```
linux ($root)/vmlinuz-5.14.0-70.22.1.e19_0.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
```

5. 別のブーターゲットを選択するには、**linux** で始まる行の末尾に **systemd.unit=** パラメーターを追加します。

```
linux ($root)/vmlinuz-5.14.0-70.22.1.e19_0.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
systemd.unit=<name>.target
```

   <name> は、使用するターゲットユニットの名前に置き換えます。たとえば、**systemd.unit=emergency.target** です。

6. Ctrl+X を押して、これらの設定で起動します。

14.4. システムのシャットダウン、サスペンド、およびハイバネート

システム管理者は、さまざまな電源管理オプションを使用して電力消費を管理したり、適切なシャットダウンを実行してすべてのデータを確実に保存したり、システムを再起動して変更や更新を適用したりできます。

14.4.1. システムのシャットダウン

システムをシャットダウンする場合は、**systemctl** ユーティリティーを使用するか、**shutdown** コマンドを使用してこのユーティリティーを呼び出します。

shutdown を使用すると、次の利点があります。

- **time** 引数を使用してシャットダウンをスケジュールできます。また、この引数を使用すると、システムのシャットダウンがスケジュールされていることがユーザーに警告されます。
- シャットダウンをキャンセルできます。

関連情報

- systemctl を使用した電源管理コマンドの概要

14.4.2. システムのシャットダウンのスケジュール設定

システム管理者は、ユーザーが作業内容を保存してシステムからログオフする時間を与えるために、遅延シャットダウンをスケジュールできます。**shutdown** コマンドを使用して、次の操作を実行します。

- 特定の時刻にシステムをシャットダウンし、マシンの電源をオフにする
- マシンの電源を切らずにシステムをシャットダウンして停止する
- 保留中のシャットダウンをキャンセルする

前提条件

- Root アクセス

手順

shutdown コマンドを使用して、次のタスクのいずれかを実行します。

- システムをシャットダウンしてマシンの電源をオフにする時刻を指定します。

```
# shutdown --poweroff hh:mm
```

hh:mm は 24 時間表記の時刻です。新しいログインを防ぐために、システムがシャットダウンする 5 分前に **/run/nologin** ファイルが作成されます。

time 引数を使用すると、オプションの **ウォールメッセージ** を指定して、システムにログインしているユーザーにシャットダウンの予定を通知できます。たとえば、**shutdown --poweroff 13:59 "Attention.The system will shut down at 13:59"** などのメッセージです。

- マシンの電源を切らずに、時間をおいてからシステムをシャットダウンして停止します。

```
# shutdown --halt +m
```

+m は遅らせる時間 (分) です。 **now** キーワードを **+0** のエイリアスとして使用できます。

- 保留中のシャットダウンをキャンセルします。

```
# shutdown -c
```

関連情報

- **shutdown(8)** の man ページ
- systemctl コマンドを使用したシステムのシャットダウン

14.4.3. systemctl コマンドを使用したシステムのシャットダウン

システム管理者は、**systemctl** コマンドを使用して、システムをシャットダウンしてマシンの電源をオフにしたり、マシンの電源をオフにせずにシステムをシャットダウンして停止したりできます。

前提条件

- Root アクセス

手順

systemctl コマンドを使用して、次のタスクのいずれかを実行します。

- システムをシャットダウンし、マシンの電源をオフにします。

```
# systemctl poweroff
```

- マシンの電源を切らずにシステムをシャットダウンして停止します。

```
# systemctl halt
```

注記

デフォルトでは、これらのコマンドのいずれかを実行すると、**systemd** が現在システムにログインしているすべてのユーザーに情報メッセージを送信します。**systemd** がメッセージを送信しないようにするには、コマンドラインオプション **--no-wall** を付けてコマンドを実行します。

14.4.4. システムの再起動

システムを再起動すると、**systemd** は実行中のすべてのプログラムとサービスを停止します。システムはシャットダウンすると、すぐに再起動します。システムの再起動は、次の場合に役立ちます。

- 新しいソフトウェアまたは更新をインストールした後
- システム設定を変更した後
- システムの問題のトラブルシューティングを行う場合

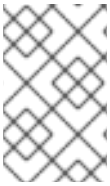
前提条件

- Root アクセス

手順

- システムを再起動します。

```
# systemctl reboot
```



注記

デフォルトでは、このコマンドを使用すると、**systemd** が現在システムにログインしているすべてのユーザーに情報メッセージを送信します。**systemd** がこのメッセージを送信しないようにするには、**--no-wall** オプションを指定してこのコマンドを実行します。

14.4.5. システムのサスペンドおよびハイバネートによる電力消費の最適化

システム管理者は、電力消費を管理し、システムのエネルギーを節約し、システムの現在の状態を保存できます。これを行うには、次のモードのいずれかを適用します。

サスペンド

サスペンドは、システムの状態を RAM に保存し、マシンにある、RAM モジュール以外のほとんどのデバイスの電源を切ります。マシンの電源を戻すと、システムは再起動せずに RAM からその状態を復元します。システムの状態がハードディスクではなくメモリーに保存されるため、システムは、ハイバネートよりも、サスペンドモードからのほうがはるかに早く復元できます。ただし、サスペンドされたシステム状態は停電に対して脆弱です。

ハイバネート

ハイバネートは、システムの状態をハードディスクドライブに保存し、マシンの電源を切ります。マシンの電源を戻すと、システムは再起動せずに、保存されたデータからその状態を復元します。システムの状態がメモリーではなくハードディスクに保存されるため、マシンでメモリーモジュールへの電源供給を維持する必要はありません。ただし、システムは、サスペンドモードより、ハイバネーションから復元する方がはるかに遅くなります。

ハイブリッドスリープ

これは、ハイバネートとサスペンドの両方の要素を組み合わせたものです。システムはまず現在の

状態をハードディスクドライブに保存し、サスペンドと同様の低電力状態に入ります。これにより、システムはより迅速に再開できるようになります。ハイブリッドスリープの利点は、スリープ状態中にシステムの電源が失われた場合でも、ハイバネーションと同様に、ハードディスクに保存されたイメージから以前の状態を回復できることです。

サスペンドしてからハイバネート

このモードでは、まずシステムがサスペンドされます。これにより、現在のシステムの状態が RAM に保存され、システムが低電力モードになります。一定期間 (**HibernateDelaySec** パラメーターで定義可能) サスペンド状態が続くと、システムはハイバネートします。ハイバネーションは、システムの状態をハードディスクドライブに保存し、システムを完全にシャットダウンします。サスペンドしてからハイバネートするモードには、バッテリー電力を節約しながら、作業をすぐに再開できるという利点があります。さらに、このモードでは、停電に備えてデータが確実に保存されます。

前提条件

- Root アクセス

手順

適切な省電力方法を選択します。

- システムをサスペンドします。

```
# systemctl suspend
```

- システムをハイバネートします。

```
# systemctl hibernate
```

- システムをハイバネートおよびサスペンドします。

```
# systemctl hybrid-sleep
```

- システムをサスペンドしてからハイバネートします。

```
# systemctl suspend-then-hibernate
```

14.4.6. systemctl を使用した電源管理コマンドの概要

以下の **systemctl** コマンドを使用して、システムの電源管理を制御できます。

表14.4 systemctl 電源管理コマンドの概要

systemctl コマンド	説明
systemctl halt	システムを停止します。
systemctl poweroff	システムの電源を切ります。
systemctl reboot	システムを再起動します。
systemctl suspend	システムをサスペンドします。

systemctl コマンド	説明
systemctl hibernate	システムをハイバネートします。
systemctl hybrid-sleep	システムをハイバネートおよびサスペンドします。

14.4.7. 電源ボタンの動作を変更する

コンピューターの電源ボタンを押すと、デフォルトではシステムがサスペンドまたはシャットダウンされます。この動作は好みに応じてカスタマイズできます。

14.4.7.1. systemd で電源ボタンの動作を変更する

非グラフィカル **systemd** ターゲットで電源ボタンを押すと、デフォルトでシステムがシャットダウンします。この動作は好みに応じてカスタマイズできます。

前提条件

- 管理アクセスがある。

手順

1. **/etc/systemd/logind.conf** 設定ファイルを開きます。
2. **HandlePowerKey=poweroff** という行を探します。
3. 行が **#** 記号で始まる場合は、この記号を削除して設定を有効にします。
4. **poweroff** を次のオプションのいずれかに置き換えます。

   poweroff
   コンピューターをシャットダウンします。

   reboot
   システムを再起動します。

   halt
   システムの停止を開始します。

   kexec
   kexec の再起動を開始します。

   suspend
   システムを一時停止します。

   hibernate
   システムの休止状態を開始します。

   ignore
   なにもしない

   たとえば、電源ボタンを押したときにシステムを再起動するには、次の設定を使用します。

   ```
   HandlePowerKey=reboot
   ```

5. 変更を保存してエディターを閉じます。

次のステップ

- グラフィカルセッションを使用する場合は、GNOME でも電源ボタンを設定します。「GNOME で電源ボタンの動作を変更する」を参照してください。

14.4.7.2. GNOME で電源ボタンの動作を変更する

グラフィカルログイン画面またはグラフィカルユーザーセッションでは、電源ボタンを押すと、デフォルトでマシンがサスペンドされます。これはユーザーが物理的に電源ボタンを押した場合と、リモートコンソールから仮想の電源ボタンを押した場合の両方で起きます。電源ボタンの動作は、別のものを選択することもできます。

前提条件

- **systemd** で電源ボタンの動作を設定しました。「systemd で電源ボタンの動作を変更する」を参照してください。

手順

1. **/etc/dconf/db/local.d/01-power** ファイルにシステム全体の設定用のローカルデータベースを作成します。次の内容を入力してください:

```
[org/gnome/settings-daemon/plugins/power]
power-button-action='suspend'
```

suspend を次の電源ボタンのアクションのいずれかに置き換えます。

nothing
: 何も実行しません。

suspend
: システムをサスペンドします。

hibernate
: システムをハイバネートします。

interactive
: 何を実行するかをユーザーに質問するポップアップクエリーを表示します。
インタラクティブモードでは、電源ボタンを押すと 60 秒後にシステムの電源が自動的にオフになります。ただし、ポップアップクエリーとは異なる動作を選択することもできます。

2. オプション: ユーザーの設定をオーバーライドし、ユーザーが変更できないようにします。**/etc/dconf/db/local.d/locks/01-power** ファイルに次の設定を入力します。

```
/org/gnome/settings-daemon/plugins/power/power-button-action
```

3. システムデータベースを更新します。

```
# dconf update
```

4. システム全体の設定を有効にするには、ログアウトして再度ログインしてください。

第15章 時刻同期の設定

IT 環境では、正確な時間管理が重要です。すべてのネットワークデバイスで時刻が一貫していれば、ログファイルのトレーサビリティーが向上します。また、特定のプロトコルは同期されたクロックを使用します。たとえば、Kerberos はタイムスタンプを使用してリプレイ攻撃を防ぎます。

15.1. CHRONY スイートを使用した NTP の設定

IT では、複数の理由から、正確な時間管理が重要です。たとえばネットワーキングでは、パケットとログのタイムスタンプが正確であることが必要になります。Linux システムでは、**NTP** プロトコルがユーザー領域で実行しているデーモンにより実装されます。

ユーザー領域のデーモンは、カーネルで実行しているシステムクロックを更新します。システムクロックは、さまざまなクロックソースを使用して時間を維持します。一般的に使用されるのは **Time Stamp Counter** (**TSC**) です。TSC は、最後にリセットされた時点からのサイクル数を計測する CPU レジスターです。非常に高速でハイレゾリューションであり、中断も発生しません。

Red Hat Enterprise Linux 8 以降、**NTP** プロトコルは **chronyd** デーモンにより実装されます。このデーモンは、**chrony** パッケージのリポジトリーから利用できます。

次のセクションでは、**chrony** スイートを使用して NTP を設定する方法を説明します。

15.1.1. chrony スイートの概要

chrony は **Network Time Protocol (NTP)** の実装です。**chrony** を使用すると、以下のことができます。

- システムクロックを、**NTP** サーバーと同期する
- システムクロックを、GPS レシーバーなどの基準クロックと同期する
- システムクロックを、手動で入力した時間と同期する
- ネットワーク内の他のコンピューターにタイムサービスを提供する **NTPv4(RFC 5905)** サーバーまたはピアとして

chrony は、ネットワーク接続が頻繁に切断される、ネットワークの混雑が長時間続く、温度が変わる (一般的なコンピューターのクロックは温度に敏感) といったさまざまな状況や、継続して実行されない、または仮想マシンで実行されているといったシステムにおいても、良好に動作します。

インターネット上で同期している 2 つのマシン間の一般的精度は数ミリ秒以内、LAN 上のマシン間では数十マイクロ秒以内です。ハードウェアのタイムスタンプまたはハードウェア基準クロックは、同期している 2 つのマシン間の精度をサブマイクロ秒レベルにまで高めることができます。

chrony は、ユーザー空間で実行する **chronyd** と、**chronyd** のパフォーマンスを監視し、実行時にさまざまなオペレーティングパラメーターを変更するのに使用できるコマンドラインプログラムである **chronyc** で構成されます。

chrony デーモンである **chronyd** は、コマンドラインユーティリティーの **chronyc** を使用して監視と管理を行います。このユーティリティーは、**chronyd** の現在の状態に対してクエリーを実行し、その設定を変更する多数のコマンド入力を可能にするコマンドプロンプトを提供します。デフォルトでは、**chronyd** は **chronyc** のローカルインスタンスのコマンドのみを受け付けますが、リモートホストから監視コマンドを受け付けるように設定することも可能です。リモートアクセスは制限する必要があります。

15.1.2. chronyc を使用した chronyd の制御

chronyc コマンドラインユーティリティーを使用して **chronyd** を制御できます。

手順

1. 対話モードでコマンドラインユーティリティー **chronyc** を使用して、**chronyd** のローカルインスタンスを変更するには、**root** で以下のコマンドを実行します。

```
# chronyc
```

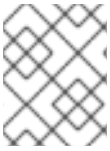
制限されているコマンドを使用する場合は、**root** で **chronyc** を実行する必要があります。

以下のように、**chronyc** コマンドプロンプトが表示されます。

```
chronyc>
```

2. コマンドのリストを表示するには、**help** と入力します。
3. 以下のように、コマンドと合わせて呼び出した場合には、非対話的なコマンドモードでユーティリティーを呼び出すこともできます。

```
chronyc command
```



注記

chronyc を使用して変更した内容は永続的ではなく、**chronyd** を再起動すると元に戻ります。永続的に変更する場合は、**/etc/chrony.conf** を変更してください。

15.1.3. chrony への移行

Red Hat Enterprise Linux 7 では、正確な時間管理を行う方法として、**ntp** または **chrony** を選択できます。**ntp** と **chrony** の相違点、**ntpd** と **chronyd** の相違点は、ntpd と chronyd の違い を参照してください。

Red Hat Enterprise Linux 8 以降、**ntp** はサポートされなくなりました。**chrony** は、デフォルトで有効になっています。このため、**ntp** から **chrony** への移行が必要になる場合があります。

ntp から **chrony** への移行は、ほとんどの場合簡単です。プログラムの、設定ファイル、およびサービスに対応する名前は、次のとおりです。

表15.1 ntp から chrony へ移行する際に、プログラム、設定ファイル、サービスに対応する名前

ntp の名前	chrony の名前
/etc/ntp.conf	/etc/chrony.conf
/etc/ntp/keys	/etc/chrony.keys
ntpd	chronyd
ntpq	chronyc
ntpd.service	chronyd.service

ntp の名前	chrony の名前
ntp-wait.service	chrony-wait.service

ntpdate ユーティリティーおよび **sntp** ユーティリティーは **ntp** ディストリビューションに含まれていますが、**-q** オプションまたは **-t** オプションを使用して **chronyd** に置き換えることができます。この設定は、**/etc/chrony.conf** を読み込まないようにコマンドラインで指定できます。たとえば、**ntpdate ntp.example.com** を実行する代わりに、以下のように **chronyd** を起動できます。

```
# chronyd -q 'server ntp.example.com iburst'
2018-05-18T12:37:43Z chronyd version 3.3 starting (+CMDMON +NTP +REFCLOCK +RTC
+PRIVDROP +SCFILTER +SIGND +ASYNCDNS +SECHASH +IPV6 +DEBUG)
2018-05-18T12:37:43Z Initial frequency -2.630 ppm
2018-05-18T12:37:48Z System clock wrong by 0.003159 seconds (step)
2018-05-18T12:37:48Z chronyd exiting
```

ntpstat ユーティリティーは、**ntp** パッケージに含まれていましたが、**ntpd** だけをサポートしていました。現在は、**ntpd** と **chronyd** の両方をサポートします。これは、**ntpstat** パッケージで入手できます。

15.1.3.1. 移行スクリプト

chrony パッケージのドキュメント (**/usr/share/doc/chrony**) には、**ntp2chrony.py** という名前の Python スクリプトがあります。このスクリプトは、既存の **ntp** 設定を **chrony** に自動的に変換します。**ntp.conf** ファイルで最も一般的なディレクティブおよびオプションがサポートされます。変換で無視されている行はすべて、確認のために、生成された **chrony.conf** ファイルにコメントとして追加されます。**ntp** 鍵ファイルに指定し、**ntp.conf** で信頼される鍵としてマークされていない鍵は、生成された **chrony.keys** ファイルにコメントとして追加されます。

デフォルトでは、スクリプトはいずれのファイルも上書きしません。**/etc/chrony.conf** または **/etc/chrony.keys** が存在する場合は、**-b** オプションを使用してファイルの名前を変更すれば、バックアップとして使用できます。このスクリプトはその他のオプションもサポートします。**--help** オプションを使用すると、サポートされるすべてのオプションが表示されます。

ntp パッケージで提供されるデフォルトの **ntp.conf** を使用したスクリプトの呼び出し例は以下のようになります。

```
# python3 /usr/share/doc/chrony/ntp2chrony.py -b -v
Reading /etc/ntp.conf
Reading /etc/ntp/crypto/pw
Reading /etc/ntp/keys
Writing /etc/chrony.conf
Writing /etc/chrony.keys
```

この場合に無視される唯一のディレクティブは **disable monitor** です。これは、**noclientlog** ディレクティブで chrony に相当するものがありますが、アンプ攻撃を軽減するためだけに、デフォルトの **ntp.conf** に含まれていました。

生成した **chrony.conf** ファイルには、通常、**ntp.conf** の制御行に対応する **allow** ディレクティブが多数含まれています。**chronyd** を **NTP** サーバーとして実行しない場合は、**allow** ディレクティブをすべて **chrony.conf** から削除します。

15.2. CHRONY の使用

次のセクションでは、**chronyd** のインストール、起動、停止の方法や、**chrony** が同期しているかどうかを確認する方法を説明します。また、システムクロックを手動で調整する方法も説明されています。

15.2.1. chrony の管理

以下の手順では、**chronyd** のインストール、起動、停止、およびステータスの確認方法を説明します。

手順

1. Red Hat Enterprise Linux では、**chrony** スイートがデフォルトでインストールされます。インストールされていることを確認するには、**root** で以下のコマンドを実行します。

```
# yum install chrony
```

chrony デーモンのデフォルトの場所は、**/usr/sbin/chronyd** です。このコマンドラインユーティリティーは **/usr/bin/chronyc** にインストールされます。

2. **chronyd** のステータスを確認するには、以下のコマンドを実行します。

```
$ systemctl status chronyd
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
   Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

3. **chronyd** を開始するには、**root** で以下のコマンドを実行します。

```
# systemctl start chronyd
```

システムの起動時に **chronyd** を自動的に起動するように設定するには、**root** で以下のコマンドを実行します。

```
# systemctl enable chronyd
```

4. **chronyd** を停止するには、**root** で以下のコマンドを実行します。

```
# systemctl stop chronyd
```

システムの起動時に **chronyd** を自動的に起動しないように設定するには、**root** で以下のコマンドを実行します。

```
# systemctl disable chronyd
```

15.2.2. chrony の同期確認

以下の手順では、**tracking** コマンド、**sources** コマンド、および **sourcestats** コマンドを使用して、chrony が同期されているかどうかを確認する方法を説明します。

手順

1. chrony の追跡を確認するには、以下のコマンドを実行します。

```
$ chronyc tracking
Reference ID    : CB00710F (ntp-server.example.net)
Stratum         : 3
Ref time (UTC)  : Fri Jan 27 09:49:17 2017
System time     :  0.000006523 seconds slow of NTP time
Last offset     : -0.000006747 seconds
RMS offset      : 0.000035822 seconds
Frequency       : 3.225 ppm slow
Residual freq   : 0.000 ppm
Skew            : 0.129 ppm
Root delay      : 0.013639022 seconds
Root dispersion : 0.001100737 seconds
Update interval : 64.2 seconds
Leap status     : Normal
```

2. sources コマンドは、**chronyd** がアクセスしている現在の時間ソースの情報を表示します。**chrony** ソースを確認するには、以下のコマンドを実行します。

```
$ chronyc sources
 210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
====
#* GPS0                          0   4   377    11   -479ns[ -621ns] /-  134ns
^? a.b.c                        2   6   377    23   -923us[ -924us] +/-   43ms
^ d.e.f                         1   6   377    21  -2629us[-2619us] +/-   86ms
```

オプションの **-v** 引数を指定すると、より詳細な情報を出力できます。この例では、余分なキャプション行は、コラムの意味を説明するものとして表示されます。

3. **sourcestats** コマンドは、**chronyd** が現在調べている各ソースに関するドリフト量とオフセット推定プロセスの情報を表示します。**chrony** ソースの統計情報を確認するには、以下のコマンドを実行します。

```
$ chronyc sourcestats
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
===============================================================================
====
abc.def.ghi                11   5   46m     -0.001      0.045      1us    25us
```

任意の引数 **-v** (verbose (詳細) の意) を指定できます。この例では、余分なキャプション行は、コラムの意味を説明するものとして表示されます。

関連情報

- **chronyc(1)** の man ページ

15.2.3. システムクロックの手動調整

以下の手順では、システムクロックを手動で調整する方法を説明します。

手順

1. システムクロックを徐々に調整していく (slew) のを止め、一度に修正 (step) するには、**root** で以下のコマンドを実行します。

```
# chronyc makestep
```

rtcfile ディレクティブを使用している場合は、リアルタイムクロックを手動で調整しないでください。ランダムな調整を行うと、リアルタイムクロックがずれる変化量を測定する必要がある **chrony** に影響を与えます。

15.2.4. chrony ディスパッチャースクリプトの無効化

chrony ディスパッチャースクリプトは、NTP サーバーのオンラインとオフラインの状態を管理します。システム管理者は、ディスパッチャースクリプトを無効にして、**chronyd** がサーバーを常にポーリングし続けるようにすることができます。

システムで NetworkManager を有効にしてネットワーク設定を管理する場合、NetworkManager はインターフェイスの再設定中、操作の停止または開始中に **chrony** ディスパッチャースクリプトを実行します。ただし、NetworkManager の外部で特定のインターフェイスまたはルートを設定すると、次の状況が発生する可能性があります。

1. NTP サーバーへのルートが存在しない場合にディスパッチャースクリプトが実行され、NTP サーバーがオフライン状態に切り替わる可能性があります。

2. 後でルートを確立すると、デフォルトではスクリプトは再実行されず、NTP サーバーはオフライン状態のままになります。

chronyd が、個別に管理されたインターフェイスを持つ NTP サーバーと確実に同期できるようにするには、ディスパッチャースクリプトを無効にします。

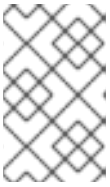
前提条件

- システムに NetworkManager をインストールして有効にしました。
- Root アクセス

手順

1. **chrony** ディスパッチャースクリプトを無効にするには、**/etc/NetworkManager/dispatcher.d/20-chrony-onoffline** ファイルを次のように編集します。

```
#!/bin/sh
exit 0
```



注記

chrony パッケージをアップグレードまたは再インストールすると、変更したディスパッチャースクリプトがパッケージ化されたバージョンのディスパッチャースクリプトに置き換えられます。

15.2.5. 孤立したネットワークでのシステムにおける chrony の設定

インターネットに接続されていないネットワークの場合、1 台のコンピューターがプライマリータイムサーバーとして選択されます。他のコンピューターは、サーバーの直接のクライアント、またはクライ

アントのクライアントです。サーバーでは、ドリフトファイルは、システムクロックのドリフトの平均率を使用して手動で設定します。サーバーを再起動すると、周囲のシステムから時間を取得し、システムクロックを設定するために平均値を計算します。その後、drift ファイルに基づいて調整の適用を再開します。drift ファイルは、**settime** コマンドが使用されたときに自動的に更新されます。

以下の手順では、分離ネットワークで asystem に **chrony** を設定する方法を説明します。

手順

1. マスターに選ばれたシステムで、**root** でテキストエディターを実行し、以下のように **/etc/chrony.conf** を実行します。

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0/24
```

   ここで、**192.0.2.0/24** は、クライアントが接続を許可されるネットワークまたはサブネットアドレスです。詳細は **chrony.conf (7) の** man ページを参照してください。

2. サーバーのダイレクトクライアントに選ばれたシステムで、**root** でテキストエディターを実行し、以下のように **/etc/chrony.conf** を実行します。

```
server ntp1.example.net
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 ntp1.example.net
allow 192.0.2.123
```

   192.0.2.123 はサーバーのアドレスで、**ntp1.example.net** はサーバーのホスト名です。この設定のクライアントは、再起動するとサーバーと再同期します。

サーバーの直接のクライアントにはならないクライアントシステムの **/etc/chrony.conf** ファイルでは、**local** ディレクティブおよび **allow** ディレクティブが省略される以外は、同じになるべきです。

孤立したネットワークでは、ローカルの参照モードを有効にする **local** ディレクティブも使用できます。これにより、**NTP** サーバーとして動作している **chronyd** は、サーバーが一度も同期されていなかったり、クロックの最終更新から長い時間が経過している場合でも、リアルタイムに同期してるように見えます。

複数のサーバーをポーリングしているクライアントを混同することなく、ネットワーク上の複数のサーバーに同じローカル設定を使用し、互いを同期させるには、Orphan モードを有効にする **local** ディレクティブの **orphan** オプションを使用します。各サーバーは、他のすべてのサーバーを **local** でポーリングするように設定する必要があります。これにより、最小の参照 ID を持つサーバーでのみローカル参照が有効になり、他のサーバーはそれに同期します。サーバーが失敗すると別のサーバーが引き継ぎます。

15.2.6. リモート監視アクセスの設定

chronyc は、以下の 2 つの方法で **chronyd** にアクセスします。

- インターネットプロトコル (IPv4 または IPv6)
- Unix ドメインソケット (ユーザー **root** または **chrony** がローカルにアクセス可能)

デフォルトでは、**chronyc** は、Unix ドメインソケットに接続します。デフォルトのパスは **/var/run/chrony/chronyd.sock** です。この接続に失敗すると (たとえば非特権ユーザーで **chronyc** を実行していると失敗する可能性があります)、**chronyc** は 127.0.0.1 への接続を試み、その後 ::1 への接続を試みます。

chronyd の動作に影響しない次の監視コマンドのみが、ネットワークに許可されています。

- activity
- manual list
- rtcdata
- smoothing
- sources
- sourcestats
- tracking
- waitsync

chronyd がこのコマンドを受け取るホスト郡は、**chronyd** の設定ファイルにある **cmdallow** ディレクティブ、または **chronyc** の **cmdallow** コマンドで設定できます。デフォルトでは、このコマンドが許可されるのは、ローカルホスト (127.0.0.1 または ::1) のものだけになります。

その他のコマンドはすべて、Unix ドメインソケットのみを介して許可されます。ネットワーク上で送信されると、たとえローカルホストであっても、**chronyd** は **Not authorised** エラーを返します。

以下の手順では、**chronyc** を使用して chronyd にリモートでアクセスする方法を説明します。

手順

1. 以下を **/etc/chrony.conf** ファイルに追加すると、IPv4 と IPv6 の両方のアドレスからアクセスが可能になります。

   ```
   bindcmdaddress 0.0.0.0
   ```

   または

   ```
   bindcmdaddress ::
   ```

2. **cmdallow** ディレクティブを使用すると、リモート IP アドレス、ネットワーク、またはサブネットからのコマンドが許可されます。
   /etc/chrony.conf ファイルに以下の内容を追加します。

   ```
   cmdallow 192.168.1.0/24
   ```

3. ファイアウォールでポート 323 を開き、リモートシステムから接続します。

```
# firewall-cmd --zone=public --add-port=323/udp
```

必要に応じて、**--permanent** オプションを使用してポート 323 を永続的に開くことができます。

```
# firewall-cmd --permanent --zone=public --add-port=323/udp
```

4. ポート 323 を永続的に開く場合は、ファイアウォール設定を再読み込みします。

```
# firewall-cmd --reload
```

関連情報

- **chrony.conf(5)** の man ページ

15.2.7. RHEL システムロールを使用した時刻同期の管理

timesync ロールを使用して、複数のターゲットマシンで時刻同期を管理できます。**timesync** ロールは、NTP または PTP 実装をインストールして、NTP または PTP クライアントとして動作してシステムクロックと同期するように設定します。

timesync ロールを使用すると、システムが **ntp** または **chrony** を使用して NTP プロトコルを実装するかどうかにかかわらず、RHEL 6 以降のすべてのバージョンの Red Hat Enterprise Linux で同じ Playbook を使用できるため、chrony への移行 も容易になる点に留意してください。

警告

timesync ロールは、マネージドホストで指定または検出されたプロバイダーサービスの設定を置き換えます。以前の設定は、ロール変数で指定されていなくても失われます。**timesync_ntp_provider** 変数が定義されていない場合は、プロバイダーの唯一の設定が適用されます。

以下の例は、サーバーにプールが 1 つしかない場合に、**timesync** ロールを適用する方法を示しています。

例15.1 サーバーの 1 つのプールに、timesync ロールを適用する Playbook の例

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
        pool: yes
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

timesync ロール変数の詳細は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/timesync** ディレクトリーの **README.md** または **README.html** ファイルを参照してください。

関連情報

- RHEL システムロールを使用するためのコントロールノードと管理対象ノードの準備

15.2.8. 関連情報

- **chronyc(1)** の man ページ
- **chronyd(8)** の man ページ
- よくある質問

15.3. ハードウェアのタイムスタンプを使用した CHRONY

ハードウェアのタイムスタンプは、一部の Network Interface Controller (NIC) でサポートされている機能です。着信パケットおよび送信パケットのタイムスタンプを正確に提供します。**NTP** タイムスタンプは通常、カーネルにより作成され、システムクロックを使用して **chronyd** が作成されます。ただし、ハードウェアのタイムスタンプが有効な場合、NIC は独自のクロックを使用して、パケットがリンク層または物理層に出入りするときにタイムスタンプを生成します。ハードウェアスタンプで **NTP** を使用すると、同期の精度を大幅に向上できます。最高精度を実現するには、**NTP** サーバーおよび **NTP** クライアントの両方が、ハードウェアのタイムスタンプを使用している必要があります。理想的な条件下では、サブマイクロ秒単位の精度を実現できるかもしれません。

ハードウェアのタイムスタンプを使用する時間同期の別のプロトコルには、**PTP** があります。

NTP とは異なり、**PTP** は、ネットワークスイッチおよびルーターの補助に依存しています。同期の精度を最高の状態にしたい場合は、**PTP** をサポートしているスイッチやルーターがあるネットワークで **PTP** を使用し、そのようなスイッチおよびルーターがないネットワークでは、**NTP** を使用することが推奨されます。

以下のセクションでは、次の方法を説明します。

- ハードウェアタイムスタンプのサポートの確認
- ハードウェアのタイムスタンプの有効化
- クライアントポーリング間隔の設定
- インターリーブモードの有効化
- 多数のクライアント向けのサーバー設定
- ハードウェアのタイムスタンプの確認
- PTP-NTP ブリッジの設定

15.3.1. ハードウェアタイムスタンプのサポートの確認

NTP を使用したハードウェアのタイムスタンプがインターフェイスでサポートされていることを確認するには、**ethtool -T** コマンドを実行します。**ethtool** が、**SOF_TIMESTAMPING_TX_HARDWARE** 機

能、**SOF_TIMESTAMPING_TX_SOFTWARE** 機能、および **HWTSTAMP_FILTER_ALL** フィルターモードをリスト表示する場合は、**NTP** を使用して、ハードウェアのタイムスタンプにインターフェイスを使用できます。

例15.2 特定のインターフェイスにおけるハードウェアのタイムスタンプのサポートの確認

```
# ethtool -T eth0
```

出力:

```
Timestamping parameters for eth0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
        ptpv1-l4-sync         (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
        ptpv1-l4-delay-req    (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
        ptpv2-l4-sync         (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
        ptpv2-l4-delay-req    (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
        ptpv2-l2-sync         (HWTSTAMP_FILTER_PTP_V2_L2_SYNC)
        ptpv2-l2-delay-req    (HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ)
        ptpv2-event           (HWTSTAMP_FILTER_PTP_V2_EVENT)
        ptpv2-sync            (HWTSTAMP_FILTER_PTP_V2_SYNC)
        ptpv2-delay-req       (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
```

15.3.2. ハードウェアのタイムスタンプの有効化

ハードウェアのタイムスタンプを有効にするには、**/etc/chrony.conf** ファイルの **hwtimestamp** ディレクティブを使用します。ディレクティブは、個別のインターフェイスを指定できますが、ワイルドカード文字を使用して、ハードウェアのタイムスタンプをサポートするすべてのインターフェイスでハードウェアのタイムスタンプを有効にすることもできます。**linuxptp** パッケージの **ptp4l** などのアプリケーションが、ハードウェアのタイムスタンプを使用していない場合は、ワイルドカード仕様を使用してください。chrony 設定ファイルで、複数の **hwtimestamp** ディレクティブが使用されます。

例15.3 hwtimestamp のディレクティブを使用したハードウェアのタイムスタンプの有効化

```
hwtimestamp eth0
hwtimestamp eth1
hwtimestamp *
```

15.3.3. クライアントポーリング間隔の設定

インターネット上のサーバーのポーリング間隔は、デフォルトの範囲である 64 秒から 1024 秒が推奨されています。ローカルサーバーおよびハードウェアのタイムスタンプでは、システムクロックのオフセットを最小限にとどめるため、ポーリング間隔は短く設定する必要があります。

/etc/chrony.conf における以下のディレクティブは、1 秒のポーリング間隔を使用してローカルの **NTP** サーバーを指定します。

```
server ntp.local minpoll 0 maxpoll 0
```

15.3.4. インターリーブモードの有効化

ハードウェアの **NTP** アプライアンスではなく、**chrony** など、ソフトウェアの **NTP** 実装を実行する汎用コンピューターの **NTP** サーバーは、パケット送信後にのみハードウェア送信タイムスタンプを取得します。この動作により、サーバーは、対応するパケットのタイムスタンプを保存できません。**NTP** クライアントが、送信後に生成された送信タイムスタンプを受け取るようにするには、**/etc/chrony.conf** のサーバーディレクティブに **xleave** オプションを追加し、クライアントが **NTP** インターリーブモードを使用するように設定します。

```
server ntp.local minpoll 0 maxpoll 0 xleave
```

15.3.5. 多数のクライアント向けのサーバーの設定

デフォルトのサーバー設定では、最多で数千のクライアントが同時にインターリーブモードを使用できます。さらに多くのクライアント向けにサーバーを設定するには、**/etc/chrony.conf** の **clientloglimit** ディレクティブを増やします。このディレクティブは、サーバーでクライアントのアクセスログに割り当てられるメモリーの最大サイズを指定します。

```
clientloglimit 100000000
```

15.3.6. ハードウェアのタイムスタンプの確認

インターフェイスがハードウェアのタイムスタンプを有効にできたことを確認するには、システムログを確認してください。ログには、**chronyd** からの各インターフェイス向けメッセージに、有効にしたハードウェアのタイムスタンプが追記されているはずです。

例15.4 ハードウェアのタイムスタンプが有効になったインターフェイスのログメッセージ

```
chronyd[4081]: Enabled HW timestamping on eth0
chronyd[4081]: Enabled HW timestamping on eth1
```

chronyd が、**NTP** クライアントまたはピアとして設定されている場合は、**chronyc ntpdata** コマンドにより、送信先と受信先のタイムスタンプおよびインターリーブモードを、各 **NTP** ソースに報告できます。

例15.5 各 NTP ソースの送信先および受信先のタイムスタンプおよびインターリーブモードの報告

```
# chronyc ntpdata
```

出力:

```
Remote address  : 203.0.113.15 (CB00710F)
Remote port     : 123
Local address   : 203.0.113.74 (CB00714A)
Leap status     : Normal
Version         : 4
Mode            : Server
Stratum         : 1
Poll interval   : 0 (1 seconds)
Precision       : -24 (0.000000060 seconds)
Root delay      : 0.000015 seconds
Root dispersion : 0.000015 seconds
Reference ID    : 47505300 (GPS)
Reference time  : Wed May 03 13:47:45 2017
Offset          : -0.000000134 seconds
Peer delay      : 0.000005396 seconds
Peer dispersion : 0.000002329 seconds
Response time   : 0.000152073 seconds
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
Interleaved     : Yes
Authenticated   : No
TX timestamping : Hardware
RX timestamping : Hardware
Total TX        : 27
Total RX        : 27
Total valid RX  : 27
```

例15.6 NTP 測定の安定性の報告

```
# chronyc sourcestats
```

ハードウェアのタイムスタンプを有効にすると、**NTP** 測定の安定性は、通常のロードにおいて数十ナノ秒または数百ナノ秒となります。この安定性は、**chronyc sourcestats** コマンドの出力の **Std Dev** 列に報告されます。

出力:

```
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
ntp.local                  12   7    11     +0.000      0.019     +0ns    49ns
```

15.3.7. PTP-NTP ブリッジの設定

非常に精度が高い **PTP** (Precision Time Protocol) のプライマリータイムセーバーが、**PTP** サポートのあるスイッチまたはルーターを持たないネットワークで利用可能な場合、コンピューターは、**PTP** スレーブおよび stratum-1 の **NTP** サーバーとしての操作に専念する可能性があります。このようなコンピューターには、2 つ以上のネットワークインターフェイスが必要であり、プライマリータイムサーバーの近くに配置するか、プライマリータイムサーバーに直接接続する必要があります。これにより、ネットワークで非常に精度の高い同期が確実に実行されます。

1 つのインターフェイスを使用して、**PTP** でシステムクロックを同期するように、**linuxptp** パッケージの **ptp4l** プログラムおよび **phc2sys** プログラムを設定します。

chronyd を設定して、その他のインターフェイスを使用してシステム時間を提供するには、以下を行います。

例15.7 その他のインターフェイスを使用してシステム時間を提供するように chronyd を設定

```
bindaddress 203.0.113.74
hwtimestamp eth1
local stratum 1
```

15.4. 以前サポートされていた設定を CHRONY で実現する手順

ntp がサポートする Red Hat Enterprise Linux の前のメジャーバージョンにあった一部の設定が、**chrony** ではサポートされません。以下のセクションでは、このような設定のリストを表示し、**chrony** を使用して以前サポートされていた設定をシステムで実現する方法を説明します。

15.4.1. ntpq および ntpdc による監視

chrony は、**NTP** モードの 6 および 7 に対応していないため、 **chronyd** は、ntp ディストリビューションの **ntpq** ユーティリティーおよび **ntpdc** ユーティリティーからは監視できません。これにより異なるプロトコルに対応します。**chronyc** はクライアントの実装になります。詳細は **chronyc(1)** の man ページを参照してください。

chronyd で同期しているシステムクロックの状態を監視するには、次のことができます。

- 追跡コマンドの使用
- **ntpstat** ユーティリティーを使用します。 これは、 **chrony** をサポートし、**ntpd** で使用したときと同様の出力を提供します。

例15.8 追跡コマンドの使用

```
$ chronyc -n tracking
Reference ID    : 0A051B0A (10.5.27.10)
Stratum         : 2
Ref time (UTC)  : Thu Mar 08 15:46:20 2018
System time     : 0.000000338 seconds slow of NTP time
Last offset     : +0.000339408 seconds
RMS offset      : 0.000339408 seconds
Frequency       : 2.968 ppm slow
Residual freq   : +0.001 ppm
Skew            : 3.336 ppm
Root delay      : 0.157559142 seconds
Root dispersion : 0.001339232 seconds
Update interval : 64.5 seconds
Leap status     : Normal
```

例15.9 ntpstat ユーティリティーの使用

```
$ ntpstat
synchronised to NTP server (10.5.27.10) at stratum 2
   time correct to within 80 ms
   polling server every 64 s
```

15.4.2. 公開鍵暗号に基づく認証メカニズムの使用

Red Hat Enterprise Linux 7 では、**ntp**は、公開鍵暗号に基づく認証メカニズムである **Autokey** をサポートしていました。

Red Hat Enterprise Linux 8 では、**chronyd** は、**Autokey** の代わりに、最新のセキュアな認証メカニズムである Network Time Security (NTS) をサポートしています。詳細は、chrony における Network Time Security (NTS) の概要を参照してください。

15.4.3. 一時的な対称関係の使用

Red Hat Enterprise Linux 7 では、**ntpd** は、**ntp.conf** 設定ファイルで指定されていないピアからのパケットにより収集できる一時的な対称関係をサポートしていました。Red Hat Enterprise Linux 8 では、**chronyd** は、**chrony.conf** ですべてのピアを指定する必要があります。一時的な対称関係はサポートされません。

peer ディレクティブで有効にした対象モードと比べると、**server** ディレクティブまたは **pool** ディレクティブで有効になっているクライアント/サーバーモードを使用したほうが安全です。

15.4.4. マルチキャストまたはブロードキャストのクライアント

Red Hat Enterprise Linux 7 では、クライアントの設定を簡素化するブロードキャストまたはマルチキャストの **NTP** モードをサポートしていました。このモードでは、個々のユーザーの特定名またはアドレスに対してリッスンする代わりに、マルチキャストまたはブロードキャストのアドレスに送信されたパケットのみをリッスンするようにクライアントを設定できます。 これは、時間の経過とともに変化する場合があります。

Red Hat Enterprise Linux 8 では、**chronyd** は、ブロードキャストモードまたはマルチキャストモードをサポートしていません。主な理由は、通常のクライアント/サーバーおよび対象モードに比べると正確性に欠け、セキュリティーも保護されていないからです。

NTP のブロードキャストまたはマルチキャストの設定からの移行には、オプションがいくつかあります。

- 1 つの名前 (ntp.example.com など) を、異なるサーバーの複数のアドレスに変換するように DNS を設定します。
  クライアントには、複数サーバーと同期するために、プールディレクティブを 1 つだけ使用した静的な設定を指定できます。サーバーがプールから到達できない場合、または同期に適切ではない場合は、クライアントが自動的にそのサーバーを、プールの別サーバーに置き換えます。

- DHCP における **NTP** サーバーリストを配布します。
  NetworkManager が、DHCP サーバーから **NTP** サーバーのリストを取得すると、それを使用するように **chronyd** が自動的に設定されます。この機能は、**PEERNTP=no** を **/etc/sysconfig/network** ファイルに追加すると無効にできます。

- **Precision Time Protocol (PTP)** を使用します。

このオプションは、サーバーが頻繁に変更する環境、または、大規模なクライアントグループが、宛先サーバーを持たずに、相互に同期できるようにする必要がある場合に主に適しています。

PTP は、マルチキャストメッセージング用に設計されており、**NTP** ブロードキャストモードと同じように動作します。**PTP** 実装は、**linuxptp** パッケージから入手できます。

通常、**PTP** が適切に動作するには、ハードウェアのタイムスタンプとネットワークスイッチのサポートが必要になります。ただし、ブロードキャストモードでは、ソフトウェアタイムスタンプを使用し、ネットワークスイッチのサポートがない場合でも、**PTP** の方が **NTP** よりも適しています。

1 つの通信経路に非常に多くの **PTP** スレーブがあるネットワークでは、そのクライアントが生成したネットワークトラフィックの量を減らすために、**hybrid_e2e** オプションで **PTP** クライアントを設定することが推奨されます。**NTP** クライアント (場合により **NTP** サーバー) として **chronyd** を実行するコンピューターを設定し、マルチキャストメッセージングを使用して、同期した時間を多数のコンピューターに配信する **PTP** タイムセーバーとして動作させることができます。

15.5. CHRONY における NETWORK TIME SECURITY (NTS) の概要

Network Time Security (NTS) は、大規模なクライアントを拡張するように設計された Network Time Protocol (NTP) の認証メカニズムです。これは、クライアントマシンへの移動時に、サーバーマシンから受信したパケットが変更されていないことを確認します。NTS (Network Time Security) には、サーバーとそのクライアント間で使用される暗号鍵を自動的に作成する NTS-KE (Key Establishment) プロトコルが含まれます。

警告

NTS は、FIPS および OSPP プロファイルと互換性がありません。FIPS および OSPP プロファイルを有効にすると、NTS で設定された **chronyd** が致命的なメッセージを表示して中断する可能性があります。**GNUTLS_FORCE_FIPS_MODE=0** を **/etc/sysconfig/chronyd** ファイルに追加することで、**chronyd** サービスの OSPP プロファイルと FIPS モードを無効にできます。

15.5.1. クライアント設定ファイルでの Network Time Security (NTS) の有効化

デフォルトでは、Network Time Security (NTS) は有効になっていません。**/etc/chrony.conf** では、NTS を有効にできます。これを行うには、以下の手順を実行します。

前提条件

- NTS に対応するサーバー

手順

クライアント設定ファイル

1. 推奨される **iburst** オプションのほかに、**nts** オプションを使用してサーバーを指定します。

   For example:

```
server time.example.com iburst nts
server nts.netnod.se iburst nts
server ptbtime1.ptb.de iburst nts
```

2. システムの起動時に Network Time Security-Key Establishment (NTS-KE) セッションが繰り返されないようにするには、次の行 (がない場合) を **chrony.conf** に追加します。

```
ntsdumpdir /var/lib/chrony
```

3. 以下の行を **/etc/sysconfig/network** に追加し、**DHCP** が提供する NTP (Network Time Protocol) サーバーとの同期を無効にします。

```
PEERNTP=no
```

4. 変更を保存します。

5. **chronyd** を再起動します。

```
systemctl restart chronyd
```

検証

- **NTS** キーが正常に確立されたかどうかを確認します。

```
# chronyc -N authdata

Name/IP address  Mode KeyID Type KLen Last Atmp  NAK Cook CLen
==============================================================
time.example.com  NTS     1   15  256  33m    0    0    8  100
nts.sth1.ntp.se   NTS     1   15  256  33m    0    0    8  100
nts.sth2.ntp.se   NTS     1   15  256  33m    0    0    8  100
```

KeyID、**Type**、および **KLen** には、ゼロ以外の値を指定する必要があります。この値が 0 になっていない場合は、システムログで **chronyd** からのエラーメッセージを確認します。

- クライアントが NTP 測定を行っていることを確認します。

```
# chronyc -N sources

MS Name/IP address Stratum Poll Reach LastRx Last sample
===============================================================
time.example.com  3       6   377    45   +355us[ +375us] +/-   11ms
nts.sth1.ntp.se    1       6   377    44   +237us[ +237us] +/-   23ms
nts.sth2.ntp.se    1       6   377    44   -170us[ -170us] +/-   22ms
```

Reach 列の値はゼロ以外にする必要があります。理想的には 377 です。この値が 377 になることがめったにないか、377 に到達しない場合は、NTP の要求または応答がネットワークで失われていることを示しています。

関連情報

- **chrony.conf(5)** の man ページ

15.5.2. サーバーで NTS (Network Time Security) の有効化

独自の Network Time Protocol (NTP) サーバーを実行している場合は、サーバーの Network Time Security (NTS) サポートを有効にして、クライアントの同期を容易にし、安全に行うことができます。

NTP サーバーがその他のサーバーのクライアントである (Stratum 1 サーバーではない) 場合は、同期に NTS または対称鍵を使用する必要があります。

前提条件

- **PEM** 形式のサーバー秘密鍵
- **PEM** 形式で必要な中間証明書を持つサーバー証明書

手順

1. **chrony.conf** で秘密鍵と証明書ファイルを指定します。以下に例を示します。

```
ntsserverkey /etc/pki/tls/private/<ntp-server.example.net>.key
ntsservercert /etc/pki/tls/certs/<ntp-server.example.net>.crt
```

2. グループの所有権を設定し、鍵と証明書ファイルの両方が chrony システムユーザーにより読み取り可能であることを確認します。以下に例を示します。

```
# chown :chrony /etc/pki/tls//<ntp-server.example.net>.
```

3. **ntsdumpdir /var/lib/chrony** ディレクティブが **chrony.conf** に存在することを確認します。
4. **chronyd** を再起動します。

```
# systemctl restart chronyd
```

重要

サーバーにファイアウォールがある場合は、NTP 用の **UDP 123** ポートと **TCP 4460** ポート、および NTS-KE (Network Time Security-Key Establishment) の両方を許可する必要があります。

検証

- 次のコマンドを使用して、クライアントマシンからクイックテストを実行します。

```
$ chronyd -Q -t 3 'server

ntp-server.example.net iburst nts maxsamples 1'
2021-09-15T13:45:26Z chronyd version 4.1 starting (+CMDMON +NTP +REFCLOCK +RTC
+PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH +IPV6 +DEBUG)
2021-09-15T13:45:26Z Disabled control of system clock
2021-09-15T13:45:28Z System clock wrong by 0.002205 seconds (ignored)
2021-09-15T13:45:28Z chronyd exiting
```

System clock wrong メッセージは、NTP サーバーが NTS-KE 接続を受け入れ、NTS で保護されている NTP メッセージで応答していることを示しています。

- サーバーで監視されている NTS-KE 接続と認証された NTP パケットを確認します。

```
# chronyc serverstats

NTP packets received       : 7
NTP packets dropped        : 0
Command packets received   : 22
Command packets dropped    : 0
Client log records dropped : 0
NTS-KE connections accepted: 1
NTS-KE connections dropped : 0
Authenticated NTP packets: 7
```

NTS-KE connections accepted および **Authenticated NTP packets** の値がゼロ以外の値の場合は、少なくとも 1 台のクライアントが NTS-KE ポートに接続し、認証された NTP リクエストを送信できたことを意味します。

第16章 言語パックの使用

Langpacks は、システムにインストールされているすべてのパッケージに対する翻訳、ディクショナリー、およびロケールを含む追加のアドオンパッケージをインストールするメタパッケージです。

Red Hat Enterprise Linux 8 システムでは、**langpacks** インストールは **langpacks-<langcode>** 言語メタパッケージおよび RPM の弱い依存関係 (補完タグ) に基づいています。

選択した言語に **langpacks** を使用する条件が 2 つあります。この前提条件が満たされている場合は、言語のメタパッケージが選択した言語の言語パックをトランザクションセットに自動的に取得します。

前提条件

- 選択した言語の **langpacks-<langcode>** 言語メタパッケージがインストールされている。Red Hat Enterprise Linux 8 では、言語パックのメタパッケージが Application Stream リポジトリーで利用できるため、Anaconda インストーラーを使用して、オペレーティングシステムの初期インストールで自動的にインストールされます。

  詳細は、言語パックを提供する言語の確認 を参照してください。

- ベースパッケージ (locale パッケージを検索) は、システムにすでにインストールされています。

16.1. 言語パックを提供する言語の確認

この手順では、言語パックを提供する言語を確認します。

手順

- 以下のコマンドを実行します。

```
# yum list langpacks-*
```

16.2. RPM の弱い依存関係ベースの言語パックでの作業

本セクションでは、RPM の弱い依存関係ベースの言語パックのクエリー、言語サポートのインストールまたは削除の時に実行できる複数アクションを説明します。

16.2.1. インストールされている言語サポートのリスト表示

インストールされている言語サポートのリストを表示するには、この手順を使用します。

手順

- 以下のコマンドを実行します。

```
# yum list installed langpacks*
```

16.2.2. 言語サポートの可用性の確認

任意の言語で言語サポートが利用できるかどうかを確認するには、以下の手順に従います。

手順

- 以下のコマンドを実行します。

```
# yum list available langpacks*
```

16.2.3. 言語にインストールしたパッケージのリスト表示

任意の言語にインストールしたパッケージのリストを取得するには、次のコマンドを実行します。

手順

- 以下のコマンドを実行します。

```
# yum repoquery --whatsupplements langpacks-<locale_code>
```

16.2.4. 言語サポートのインストール

新しい言語サポートを追加するには、以下の手順に従います。

手順

- 以下のコマンドを実行します。

```
# yum install langpacks-<locale_code>
```

16.2.5. 言語サポートの削除

インストールされている言語サポートを削除するには、以下の手順に従います。

手順

- 以下のコマンドを実行します。

```
# yum remove langpacks-<locale_code>
```

16.3. GLIBC-LANGPACK-<LOCALE_CODE> でディスク領域の節約

現在、すべてのロケールは **/usr/lib/locale/locale-archive** ファイルに格納されますが、多くのディスク領域が必要になります。

コンテナーやクラウドなど、ディスク容量が重要なシステム、または少数のロケールが必要なシステムでは、glibc ロケール言語パックパッケージ (**glibc-langpack-<locale_code>**) を使用できます。

ロケールを個別にインストールするには、パッケージインストールのフットプリントが小さいため、以下の手順を使用します。

手順

- 以下のコマンドを実行します。

```
# yum install glibc-langpack-<locale_code>
```

Anaconda でオペレーティングシステムをインストールする場合は、インストール時に使用する言語と、追加言語として選択した言語の **glibc-langpack-<locale_code>** がインストールされます。すべてのロケールが含まれている **glibc-all-langpacks** はデフォルトでインストールされており、一部のロケールだけをインストールすることは非推奨であることに注意してください。1 つ以上の選択した言語の **glibc-langpack-<locale_code>** をインストールする場合は、インストール後に **glibc-all-langpacks** を削除して、ディスク領域を空けることができます。

glibc-all-langpacks の代わりに、選択した **glibc-langpack-<locale_code>** パッケージだけをインストールすると、ランタイムパフォーマンスに影響を及ぼします。

注記

ディスク領域が問題ではない場合は、**glibc-all-langpacks** パッケージを使用してすべてのロケールをインストールします。

第17章 後で分析するためにクラッシュしたカーネルのダンプ

kdump サービスを使用して後で分析できるようにシステムのメモリー内容を保存することで、システムがクラッシュした理由を分析できます。本セクションでは、**kdump** の概要と、RHEL Web コンソールまたは対応する RHEL システムロールを使用して **kdump** を設定する方法を説明します。

17.1. KDUMP とは

kdump は、クラッシュダンプメカニズムを提供し、クラッシュダンプまたは **vmcore** ファイルとして知られるダンプファイルを生成するサービスです。**vmcore** ファイルには、分析とトラブルシューティングに役立つシステムメモリーのコンテンツが含まれています。**kdump** は **kexec** システムコールを使用して、再起動せずに **キャプチャーカーネル** である 2 番目のカーネルで起動し、クラッシュしたカーネルメモリーの内容をキャプチャしてファイルに保存します。この別のカーネルは、システムメモリーの予約部分で使用できます。

重要

カーネルクラッシュダンプは、システム障害時に利用できる唯一の情報になります。したがって、ミッションクリティカルな環境では、**kdump** を稼働させることが重要です。Red Hat は、通常のカーネル更新サイクルで **kexec-tools** を定期的に更新してテストすることを推奨します。これは、新しいカーネル機能をインストールする場合に特に重要です。

kdump は、マシンにインストールされているすべてのカーネルに対して、または指定したカーネルに対してのみ有効にできます。これは、マシンで複数のカーネルが使用されており、その一部が安定しており、クラッシュの心配がない場合に役立ちます。**kdump** をインストールすると、デフォルトの **/etc/kdump.conf** ファイルが作成されます。**/etc/kdump.conf** ファイルにはデフォルトの最小 **kdump** 設定が含まれており、これを編集して **kdump** 設定をカスタマイズできます。

17.2. WEB コンソールで KDUMP メモリーの使用量およびターゲットの場所を設定

RHEL Web コンソールインターフェイスを使用して、**kdump** カーネルのメモリー予約を設定し、**vmcore** ダンプファイルをキャプチャーするターゲットの場所を指定することもできます。

前提条件

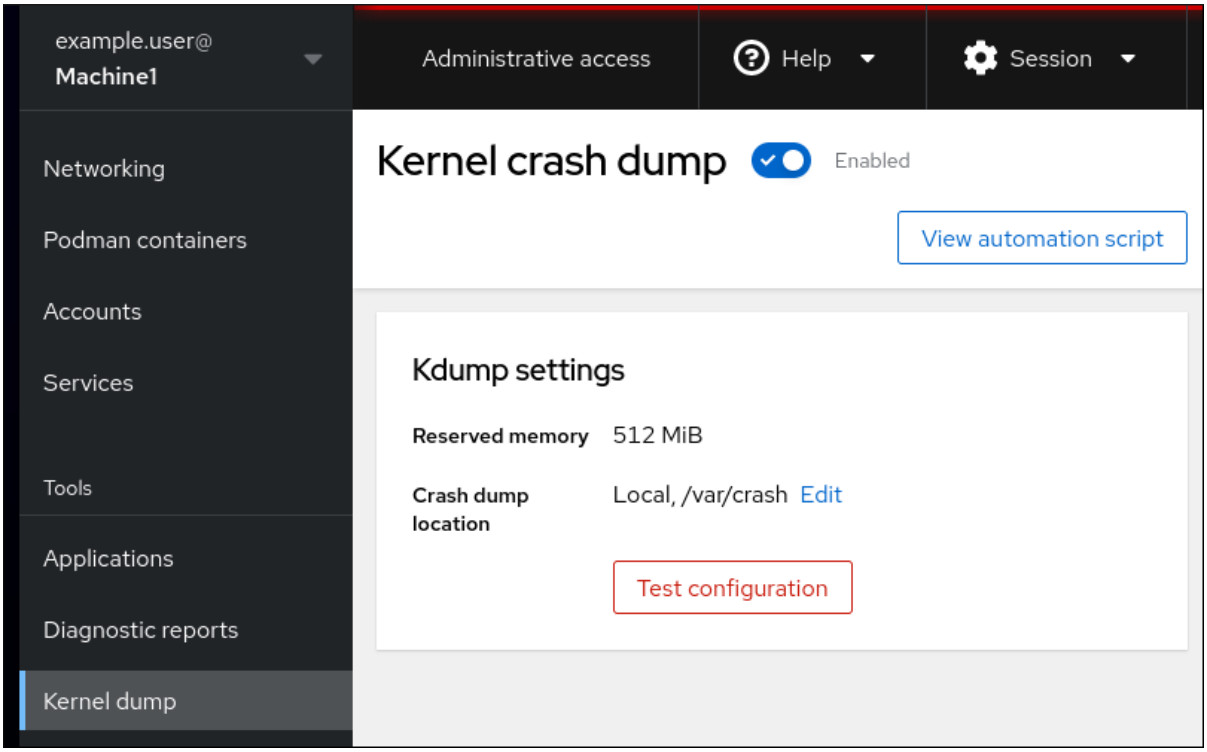
- Web コンソールがインストールされており、アクセス可能である。
  詳細は、Web コンソールのインストール を参照してください。

手順

1. Web コンソールで、**Kernel dump** タブを開き、**Kernel crash dump** スイッチをオンに設定して **kdump** サービスを起動します。
2. ターミナルで **kdump** のメモリー使用量を設定します。以下に例を示します。

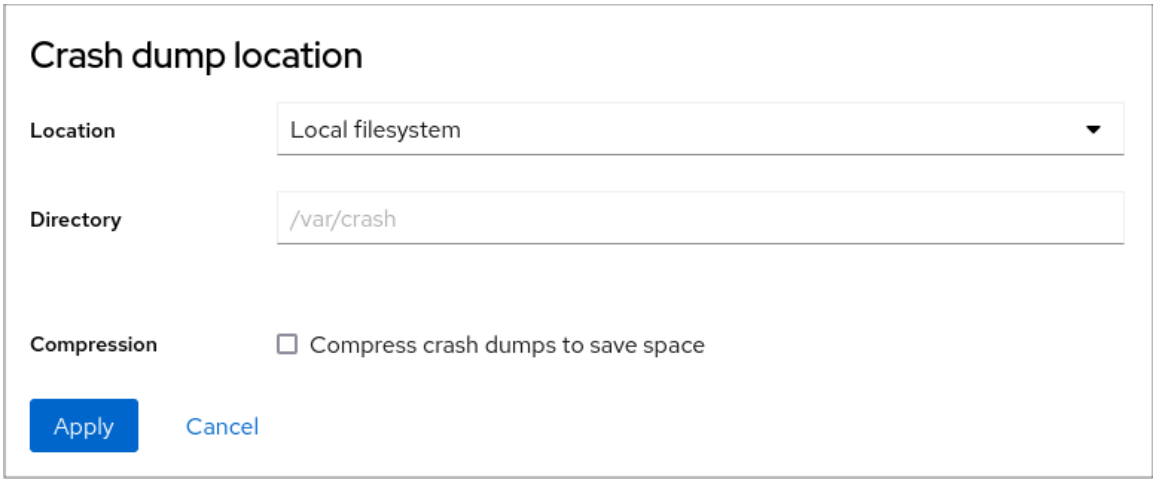
   ```
   $ sudo grubby --update-kernel ALL --args crashkernel=512M
   ```

   変更を適用するにはシステムを再起動します。
3. **Kernel dump** タブで、**Crash dump location** フィールドの末尾にある **Edit** をクリックします。



4. **vmcore** ダンプファイルを保存するターゲットディレクトリーを指定します。

- ローカルファイルシステムの場合は、ドロップダウンメニューから **Local Filesystem** を選択します。



- SSH プロトコルを使用したリモートシステムの場合は、ドロップダウンメニューから **Remote over SSH** を選択し、次のフィールドを指定します。
  - **Server** フィールドに、リモートサーバーのアドレスを入力します。
  - **SSH key** フィールドに、SSH キーの場所を入力します。
  - **Directory** フィールドに、ターゲットディレクトリーを入力します。
- NFS プロトコルを使用したリモートシステムの場合は、ドロップダウンメニューから **Remote over NFS** を選択し、次のフィールドを指定します。
  - **Server** フィールドに、リモートサーバーのアドレスを入力します。
  - **Export** フィールドに、NFS サーバーの共有フォルダーの場所を入力します。
  - **Directory** フィールドに、ターゲットディレクトリーを入力します。

注記

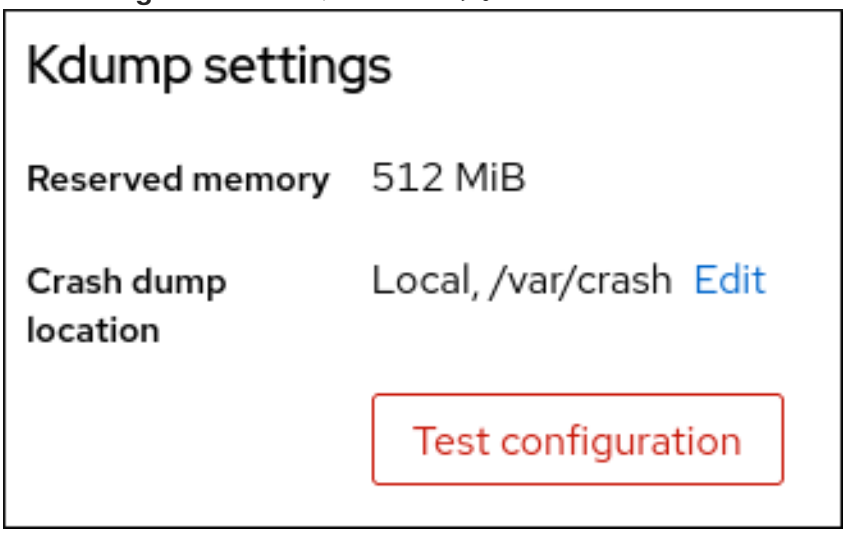
Compression チェックボックスをオンにすると、**vmcore** ファイルのサイズを削減できます。

5. **View automation script** をクリックして自動化スクリプトを表示します。
生成されたスクリプトを含むウィンドウが開きます。シェルスクリプトと Ansible Playbook の生成オプションタブ間を移動できます。

6. オプション: **[クリップボードにコピー]** をクリックしてスクリプトをコピーします。
このスクリプトを使用すると、複数のマシンに同じ設定を適用できます。

検証

1. **Test configuration** をクリックします。



2. Test kdump settings の下にある **Crash system** をクリックします。

警告

システムクラッシュを開始すると、カーネルの動作が停止し、システムがクラッシュしてデータが失われます。

関連情報

- サポートしている kdump のダンプ出力先

17.3. RHEL システムロールを使用した KDUMP

RHEL システムロールは、複数の RHEL システムをリモートで管理するための一貫した設定インターフェイスを提供する Ansible ロールおよびモジュールのコレクションです。**kdump** ロールを使用すると、複数のシステムに基本的なカーネルダンプパラメーターを設定できます。

警告

kdump ロールは、**/etc/kdump.conf** ファイルを置き換えて、マネージドホストの **kdump** 設定をすべて置き換えます。また、**kdump** ロールが適用されると、**/etc/sysconfig/kdump** ファイルを置き換えて、ロール変数で指定されていない場合でも、以前の **kdump** の設定もすべて置き換えられます。

以下の例は、**kdump** システムロールを適用してクラッシュダンプファイルの場所を設定する方法を示しています。

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

kdump ロール変数の詳細は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/kdump** ディレクトリーの **README.md** または **README.html** ファイルを参照してください。

関連情報

- RHEL システムロールの概要

17.4. 関連情報

- kdump のインストール
- コマンドラインで kdump の設定
- Web コンソールで kdump の設定

第18章 システムの復旧および復元

Red Hat Enterprise Linux では、既存のバックアップを使用してシステムを復旧および復元するために、ReaR (Relax-and-Recover) ユーティリティーが同梱されています。

このユーティリティーは、障害復旧ソリューションとして、またシステム移行にも使用できます。

このユーティリティーを使用すると、以下のタスクを実行できます。

- イメージを使用して、起動可能なイメージを作成し、既存のバックアップからシステムを復元する
- オリジナルのストレージレイアウトを複製する
- ユーザーおよびシステムファイルを復元する
- システムを別のハードウェアに復元する

また、障害復旧の場合は、特定のバックアップソフトウェアを ReaR に統合することもできます。

ReaR 設定の概要手順は以下のとおりです。

1. ReaR をインストールします。
2. ReaR 設定ファイルを変更して、バックアップ手法の詳細を追加します。
3. レスキューシステムを作成します。
4. バックアップファイルを生成します。

18.1. REAR の設定

以下の手順を使用して、Relax-and-Recover (ReaR) ユーティリティーを使用するパッケージのインストール、レスキューシステムの作成、バックアップの設定および生成を行います。

前提条件

- バックアップ復元計画をもとに、必要な設定ができている。
  NETFS バックアップメソッド (ReaR に完全に統合され、組み込まれたメソッド) を使用できることに注意してください。

手順

1. 次のコマンドを実行して ReaR ユーティリティーをインストールします。

```
# yum install rear
```

2. 以下の例のように、任意のエディターで ReaR 設定ファイルを変更します。

```
# vi /etc/rear/local.conf
```

3. バックアップ設定の詳細を **/etc/rear/local.conf** に追加します。たとえば、**NETFS** バックアップメソッドの場合は、以下の行を追加します。

```
BACKUP=NETFS
BACKUP_URL=backup.location
```

backup.location は、バックアップ先の URL に置き換えます。

4. 新規バックアップの作成時に以前のバックアップアーカイブを維持するように RaaR 設定を行うには、以下の行を設定ファイルに追加します。

```
NETFS_KEEP_OLD_BACKUP_COPY=y
```

5. 増分バックアップ (実行するたびに変更されたファイルのみがバックアップされる) を設定する場合は、以下の行を追加します。

```
BACKUP_TYPE=incremental
```

6. レスキューシステムを作成します。

```
# rear mkrescue
```

7. 復元計画に従ってバックアップを作成します。たとえば、**NETFS** バックアップメソッドの場合は、以下のコマンドを実行します。

```
# rear mkbackuponly
```

または、以下のコマンドを実行すると、1 つの手順でレスキューシステムとバックアップを作成できます。

```
# rear mkbackup
```

このコマンドは、**rear mkrescue** コマンドと **rear mkbackuponly** コマンドの機能を組み合わせたものです。

18.2. 64 ビット IBM Z アーキテクチャーで REAR レスキューイメージの使用

Basic Relax and Recover (ReaR) 機能が、64 ビットの IBM Z アーキテクチャーでテクノロジープレビューとして利用できるようになりました。IBM Z では、z/VM 環境でのみ ReaR レスキューイメージを作成できます。論理パーティション (LPAR) のバックアップおよび復元はテストされていません。

重要

64 ビット IBM Z アーキテクチャーでの ReaR は、**rear** パッケージのバージョン 2.6-9.el8 以降でのみサポートされます。以前のバージョンは、テクノロジープレビュー機能としてのみ利用できます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

現在利用できる出力方法は、Initial Program Load (IPL) のみです。IPL は、**zIPL** ブートローダーで使用できるカーネルと初期 RAM ディスク (initrd) を生成します。

前提条件

- ReaR がインストールされている。

- ReaR をインストールするには、**yum install rear** コマンドを実行します。

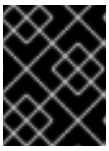
手順

以下の変数を**/etc/rear/local.conf**に追加し、64 ビット IBM Z アーキテクチャーでレスキューイメージを生成するように ReaR を設定します。

1. **IPL** アウトプットメソッドを設定するには、**OUTPUT=IPL** を追加します。
2. バックアップメソッドとバックアップ先を設定するには、**BACKUP** 変数および **BACKUP_URL** 変数を追加します。以下に例を示します。

```
BACKUP=NETFS

BACKUP_URL=nfs://<nfsserver name>/<share path>
```



重要

ローカルバックアップストレージは、現在、64 ビットの IBM Z アーキテクチャーではサポートされていません。

3. 必要に応じて、**OUTPUT_URL** を設定して、カーネルファイルおよび **initrd** ファイルを保存することもできます。初期設定では、**OUTPUT_URL** は **BACKUP_URL** に合わせて配置されています。
4. バックアップとレスキューのイメージの作成を実行するには、次のコマンドを実行します。

```
# rear mkbackup
```

5. これにより、**BACKUP_URL** 変数または **OUTPUT_URL** (設定されている場合) 変数で指定された場所にカーネルファイルと initrd ファイルが作成され、指定されたバックアップメソッドを使用してバックアップが作成されます。
6. システムを復元するには、手順 3 で作成した ReaR カーネルファイルおよび initrd ファイルを使用し、**zipl** ブートローダー、カーネル、および **initrd** で準備した DASD (Direct Attached Storage Device) または FCP (Fibre Channel Protocol) 接続の SCSI デバイスから起動します。詳細は準備した DASD の使用を参照してください。
7. レスキューカーネルと **initrd** が起動すると、ReaR レスキュー環境が起動します。システムの復元を続行します。

警告

現在、レスキュープロセスは、システムに接続したすべての DASD (Direct Attached Storage Devices) を再フォーマットします。システムのストレージデバイスに貴重なデータが存在する場合は、システムの復旧を行わないでください。これには、レスキュー環境で起動するのに使用された zipl ブートローダー、ReaR カーネル、および initrd で準備されたデバイスも含まれます。必ずコピーを保管してください。

関連情報

- z/VM へのインストール
- 設定済み DASD の使用

第19章 動的プログラミング言語のインストールおよび使用

Red Hat では、Python、PHP、Tcl/TK などのさまざまなプログラミング言語が利用可能です。それらを使用して独自のアプリケーションやサービスを開発します。

19.1. PYTHON の概要

Python は、オブジェクト指向、命令、機能、手順などの複数のプログラミングパラダイムをサポートする、高レベルのプログラミング言語です。Python は動的なセマンティクスを持ち、汎用プログラミングに使用できます。

Red Hat Enterprise Linux では、システムツール、データ分析用のツール、または Web アプリケーションを提供するパッケージなど、システムにインストールされている多くのパッケージが Python で記述されています。このパッケージを使用するには、**python*** パッケージがインストールされている必要があります。

19.1.1. Python のバージョン

Python で互換性のない 2 つのバージョン (Python 2.x および Python 3.x) が広く使用されています。RHEL 8 は、以下のバージョンの Python を提供します。

表19.1 RHEL 8 の Python バージョン

バージョン	インストールするパッケージ	コマンドの例	利用可能となったバージョン	ライフサイクル
Python 3.6	**python3**、**python36**	**python3**、**python3.6**、**pip3**、**pip3.6**	RHEL 8.0	完全な RHEL 8
Python 2.7	**python2**	**python2**、**pip2**	RHEL 8.0	より短い
Python 3.8	**python38**	**python3.8**、**pip3.8**	RHEL 8.2	より短い
Python 3.9	**python39**	**python3.9**、**pip3.9**	RHEL 8.4	より短い
Python 3.11	**python3.11**	**python3.11**、**pip3.11**	RHEL 8.8	より短い
Python 3.12	**python3.12**	**python3.12**, **pip3.12**	RHEL 8.10	より短い

サポート期間の詳細は、Red Hat Enterprise Linux のライフサイクル および Red Hat Enterprise Linux Application Streams ライフサイクル を参照してください。

3.9 までの Python バージョンはそれぞれ別のモジュールで配布されます。Python 3.11 および Python 3.12 は、**python3.11** および **python3.12** パッケージを含む非モジュール RPM パッケージのスイートとして配布されます。

同じ RHEL 8 システムに複数の Python バージョンを並行してインストールできます。

重要

Python のインストール時、起動時、対話時にはバージョンを常に指定します。たとえば、パッケージ名またはコマンド名で、**python** の代わりに **python3** を使用します。すべての Python 関連コマンドには、pip3、pip2、**pip3.8**、**pip3.9**、または **pip3.11** などのバージョンも含める必要があります。

RHEL 8 では、バージョンを指定しない **python** コマンド (**/usr/bin/python**) は、デフォルトで利用できません。**alternatives** コマンドを使用すれば設定できます。手順については、バージョンを指定しない Python の設定 を参照してください。

alternatives コマンドを使用して加えられた変更を除き、**/usr/bin/python** への手動の変更は、更新時に上書きされる可能性があります。

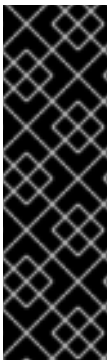
システム管理者は、以下の理由で Python 3 を使用します。

- Python 3 は、Python プロジェクトの主な開発方向を表します。
- アップストリームコミュニティーでの Python 2 のサポートは 2020 年に終了しました。
- 人気のある Python ライブラリーは、アップストリームでの Python 2 サポートを終了します。
- お客様の **Python 3** への移行を容易にするために、Red Hat Enterprise Linux 8 の Python 2 のライフサイクルが短くなっています。

開発者の場合、Python 2 と比較して Python 3 には以下の利点があります。

- Python 3 を使用すると、表現力があり、メンテナンスが可能な正しいコードをより簡単に記述できます。
- Python 3 で書かれたコード寿命は長くなります。
- Python 3 には、**asyncio**、f-strings、高度なアンパッキング、キーワードのみの引数、例外チェーンなどの新機能があります。

ただし、レガシーソフトウェアでは、**/usr/bin/python** を Python 2 に設定する必要がある場合があります。この理由により、デフォルトの python パッケージは Red Hat Enterprise Linux 8 で配布されず、**バージョンを指定しない Python の設定** で説明されているように、**/usr/bin/python** として使用する Python のバージョンを 2 または 3 から選択できます。



重要

Red Hat Enterprise Linux 8 のシステムツールは、内部の **platform-python** パッケージで提供される Python バージョン 3.6 を使用します。このパッケージをお客様が直接使用することは意図されていません。Python 3.6 用の **python36** パッケージの **python3** または **python3.6** コマンドを使用するか、それ以降の Python バージョンを使用することを推奨します。

platform-python パッケージは他のパッケージで必要となるため、RHEL 8 から削除しないでください。

19.1.2. Python のバージョン間の主な違い

RHEL 8 に含まれる Python のバージョンは、さまざまな点で異なります。

Python バインディング

python38 モジュールと **python39** モジュール、および **python3.11** パッケージスイートには、**python36** モジュールに提供されるシステムツール (RPM、DNF、SELinux など) と同じバインディングが含まれていません。したがって、基本オペレーティングシステムとの最大の互換性またはバイナリー互換性が必要な場合は、**python36** を使用してください。さまざまな Python モジュールの新しいバージョンと共にシステムバインディングが必要な特殊な例では、**pip** を介して、Python の **venv** または **virtualenv** 環境にインストールされたサードパーティーのアップストリーム Python モジュールと組み合わせて、**python36** モジュールを使用します。

Python 3.11 仮想環境は、 virtualenv ではなく venv を使用して作成する必要があります

python3-virtualenv パッケージによって提供される RHEL 8 の **virtualenv** ユーティリティーは、Python 3.11 と互換性がありません。**virtualenv** を使用して仮想環境を作成しようとすると、次のようなエラーメッセージが表示されて失敗します。

```
$ virtualenv -p python3.11 venv3.11
Running virtualenv with interpreter /usr/bin/python3.11
ERROR: Virtual environments created by virtualenv < 20 are not compatible with Python 3.11.
ERROR: Use python3.11 -m venv instead.
```

Python 3.11 または Python 3.12 仮想環境を作成するには、代わりに **python3.11 -m venv** または **python3.12 -m venv** コマンドを使用します。これらのコマンドは、標準ライブラリーの **venv** モジュールを使用します。

19.2. PYTHON のインストールおよび使用

Red Hat Enterprise Linux 8 では、Python 3 はバージョン 3.6、3.8、および 3.9 で配布され、AppStream リポジトリーの **python36**、**python38**、および **python39** モジュールと **python3.11** パッケージスイートによって提供されます。

警告

バージョンを指定しない **python** コマンドを使用して Python をインストールまたは実行すると、曖昧なためデフォルトでは動作しません。Python のバージョンを常に指定するか、**alternatives** コマンドを使用してシステムのデフォルトバージョンを設定します。

19.2.1. Python 3 のインストール

設計上、**python27**、**python36**、**python38**、**python39** モジュールと **python3.11** パッケージスイートを含む RHEL 8 モジュールを並行してインストールできます。

mod_wsgi モジュールを除き、Python 3.8、Python 3.9、および Python 3.11 (各バージョン用にビルドされたパッケージを含む) を、Python 3.6 と並行して同じシステムにインストールできます。Apache HTTP サーバーの制限により、python3-mod_wsgi、**python38-mod_wsgi**、**python39-mod_wsgi**、または **python3.11-mod_wsgi** パッケージのいずれか 1 つだけをシステムにインストールできます。

手順

- **python36** モジュールから Python 3.6 をインストールするには、以下を使用します。

```
# yum install python3
```

python36:3.6 モジュールストリームは、自動的に有効になります。

- **python38** モジュールから Python 3.8 をインストールするには、以下を使用します。

```
# yum install python38
```

python38:3.8 モジュールストリームは、自動的に有効になります。

- **python39** モジュールから Python 3.9 をインストールするには、以下を使用します。

```
# yum install python39
```

python39:3.9 モジュールストリームは、自動的に有効になります。

- Python **3.11** RPM パッケージから Python 3.11 をインストールするには、次を使用します。

```
# yum install python3.11
```

- Python **3.11** RPM パッケージから Python 3.11 をインストールするには、次を使用します。

```
# yum install python3.12
```

検証手順

- お使いのシステムにインストールされている Python のバージョンを確認するには、必要なバージョンの Python 固有の **python** コマンドで **--version** オプションを使用します。
  - Python 3.6 の場合

```
$ python3 --version
```

  - Python 3.8 の場合

```
$ python3.8 --version
```

  - Python 3.9 の場合

```
$ python3.9 --version
```

  - Python 3.11 の場合

```
$ python3.11 --version
```

  - Python 3.12 の場合:

```
$ python3.12 --version
```

関連情報

- ユーザー空間コンポーネントのインストール、管理、および削除

19.2.2. Python 3 追加パッケージのインストール

Python 3.6 のアドオンモジュールを含むパッケージでは通常、**python3-** 接頭辞が使用され、Python 3.8 のパッケージには **python38-** 接頭辞が含まれ、Python 3.9 のパッケージには **python39-** 接頭辞が含まれ、Python 3.11 のパッケージには **python3.11-** 接頭辞が含まれ、Python 3.12 のパッケージには **python3.12-** 接頭辞が含まれます。以下の例にあるように、追加の Python パッケージのインストール時には常に接頭辞を含めます。

手順

- Python 3.6 の **Requests** モジュールをインストールするには、以下を使用します。

```
# yum install python3-requests
```

- **Cython** 拡張を Python 3.8 にインストールするには、以下を使用します。

```
# yum install python38-Cython
```

- Python 3.9 から **pip** パッケージインストーラーをインストールするには、以下を使用します。

```
# yum install python39-pip
```

- Python 3.11 から **pip** パッケージインストーラーをインストールするには、以下を使用します。

```
# yum install python3.11-pip
```

- Python 3.12 から **pip** パッケージインストーラーをインストールするには、以下を使用します。

```
# yum install python3.12-pip
```

関連情報

- Python アドオンモジュールに関するアップストリームドキュメント

19.2.3. 開発者用の Python 3 追加ツールのインストール

開発者向けの追加の Python ツールは、主に、それぞれの **python38-devel モジュール** または **python39-devel** モジュール内の CodeReady Linux Builder (CRB) リポジトリー、または **python3.11-*** パッケージを通じて配布されます。

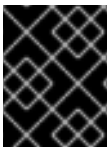
python3-pytest パッケージ (Python 3.6 用) とその依存関係は、AppStream リポジトリーで入手できます。

CRB リポジトリーは以下を提供します。

- **python38-devel** モジュール。**python38-pytest** パッケージとその依存関係が含まれます。
- **python39-devel** モジュール。これには、**python39-pytest** パッケージとその依存関係、**python39-debug** パッケージと **python39-Cython** パッケージが含まれます。
- **python3.11-*** パッケージには次のものが含まれます。
  - **python3.11-pytest** とその依存関係

- **python3.11-idle**
- **python3.11-debug**
- **python3.11-Cython**

- **python3.12-*** パッケージには、**python3.11-*** と同様のパッケージセットが含まれています。



重要

CodeReady Linux Builder リポジトリーの内容は、Red Hat ではサポートされていません。

注記

アップストリームの Python 関連のパッケージがすべて RHEL で利用できるわけではありません。

python3*-pytest パッケージをインストールするには、次の手順を使用します。

手順

1. Python 3.8 以降の場合は、CodeReady Linux Builder リポジトリーを有効にします。

```
# subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

2. Python 3.8 または 3.9 の場合は、次のように、それぞれの **python3*-devel** モジュールを有効にします。

```
# yum module enable python39-devel
```

3. **python3*-pytest** パッケージをインストールします。
   - Python 3.6 の場合

```
# yum install python3-pytest
```

   - Python 3.8 の場合

```
# yum install python38-pytest
```

   - Python 3.9 の場合

```
# yum install python39-pytest
```

   - Python 3.11 の場合

```
# yum install python3.11-pytest
```

   - Python 3.12 の場合:

```
# yum install python3.12-pytest
```

関連情報

- CodeReady Linux Builder 内のコンテンツを有効にして使用する方法
- パッケージマニフェスト

19.2.4. Python 2 のインストール

一部のアプリケーションやスクリプトが Python 3 に完全に移植されておらず、Python 2 を実行する必要があります。Red Hat Enterprise Linux 8 では、Python 3 と Python 2 を同時にインストールできます。Python 2 機能が必要な場合は **python27** モジュールをインストールしてください。これは AppStream リポジトリーから入手できます。

警告

Python 3 は、Python プロジェクトの主な開発方針です。Python 2 のサポートは段階的に廃止されています。**python27** モジュールのサポート期間は、Red Hat Enterprise Linux 8 よりも短くなっています。

手順

- **python27** モジュールから Python 2.7 をインストールするには、以下を使用します。

```
# yum install python2
```

python27:2.7 モジュールストリームは、自動的に有効になります。

Python 2 用のアドオンモジュールのパッケージは、通常、接頭辞 **python2-** を使用します。以下の例にあるように、追加の Python パッケージのインストール時には常に接頭辞を含めます。

- Python 2 の **Requests** モジュールをインストールするには、以下を使用します。

```
# yum install python2-requests
```

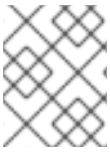
- Python 2 に **Cython** 拡張をインストールするには、以下を使用します。

```
# yum install python2-Cython
```

検証手順

- システムに Python バージョンがインストールされていることを確認するには、以下を使用します。

```
$ python2 --version
```



注記

設計上、**python27**、**python36**、**python38**、および **python39** モジュールを含む RHEL 8 モジュールを同時にインストールできます。

関連情報

- RHEL 8 でのユーザー空間コンポーネントのインストール、管理、および削除

19.2.5. Python 2 から Python 3 への移行

開発者は、Python 2 で記述したコードを Python 3 に移行できます。

大規模なコードベースを Python 3 に移行する方法は The Conservative Python 3 Porting Guide を参照してください。

この移行が終了すると、元の Python 2 コードは Python 3 インタープリターにより解釈できるようになり、同様に Python 2 インタープリターは解釈できるままとなることに注意してください。

19.2.6. Python の使用

Python インタープリターまたは Python 関連のコマンドを実行する場合は、常にバージョンを指定します。

前提条件

- 必要なバージョンの Python がインストールされていることを確認する。
- **Python 3.11** または **Python 3.12** 用のサードパーティーアプリケーションをダウンロードしてインストールする場合は、python3.11-pip または python3.12-pip パッケージをインストールします。

手順

- Python 3.6 インタープリターまたは関連コマンドを実行するには、以下を使用します。

```
$ python3
$ python3 -m venv --help
$ python3 -m pip install package
$ pip3 install package
```

- Python 3.8 インタープリターまたは関連コマンドを実行するには、以下を使用します。

```
$ python3.8
$ python3.8 -m venv --help
$ python3.8 -m pip install package
$ pip3.8 install package
```

- Python 3.9 インタープリターまたは関連コマンドを実行するには、以下を使用します。

```
$ python3.9
$ python3.9 -m venv --help
$ python3.9 -m pip install package
$ pip3.9 install package
```

- Python 3.11 インタープリターまたは関連コマンドを実行するには、たとえば、以下を使用します。

```
$ python3.11
```

```
$ python3.11 -m venv --help
$ python3.11 -m pip install package
$ pip3.11 install package
```

- Python 3.12 インタープリターまたは関連コマンドを実行するには、たとえば、以下を使用します。

```
$ python3.12
$ python3.12 -m venv --help
$ python3.12 -m pip install package
$ pip3.12 install package
```

- Python 2 インタープリターまたは関連コマンドを実行するには、以下を使用します。

```
$ python2
$ python2 -m pip install package
$ pip2 install package
```

19.3. バージョンを指定しない PYTHON の設定

システム管理者は、**alternatives** コマンドを使用して、**/usr/bin/python** に、バージョンを管理しない **python** コマンドを設定できます。必要なパッケージ (python3、python38、python39、**python3.11**、または **python2**) は、バージョンを指定しないコマンドをそれぞれのバージョンに設定する前にインストールする必要があります。

重要

/usr/bin/python 実行ファイルは **代替** システムによって制御されます。更新時に手動の変更が上書きされる可能性があります。

その他の Python 関連のコマンド (**pip3** など) には、バージョンを指定しないで設定できるバリアントがあります。

19.3.1. バージョンを指定しない python コマンドを直接設定

バージョンを指定しない **python** コマンドを、選択した Python バージョンに直接設定できます。

前提条件

- 必要なバージョンの Python がインストールされていることを確認する。

手順

- バージョンを指定しない **python** コマンドを Python 3.6 に設定するには、以下を使用します。

```
# alternatives --set python /usr/bin/python3
```

- バージョンを指定しない **python** コマンドを Python 3.8 に設定するには、以下を使用します。

```
# alternatives --set python /usr/bin/python3.8
```

- バージョンを指定しない **python** コマンドを Python 3.9 に設定するには、以下を使用します。

```
# alternatives --set python /usr/bin/python3.9
```

- バージョンを指定しない **python** コマンドを Python 3.11 に設定するには、以下を使用します。

```
# alternatives --set python /usr/bin/python3.11
```

- バージョンを指定しない **python** コマンドを Python 3.11 に設定するには、以下を使用します。

```
# alternatives --set python /usr/bin/python3.12
```

- バージョンを指定しない **python** コマンドを Python 2 に設定するには、以下のコマンドを実行します。

```
# alternatives --set python /usr/bin/python2
```

19.3.2. バージョンを指定しない python コマンドを、必要な Python バージョンに対話的に設定する

バージョンを指定しない **python** コマンドを、必要な Python バージョンに対話的に設定できます。

前提条件

- 必要なバージョンの Python がインストールされていることを確認する。

手順

1. バージョンを指定しない **python** コマンドを対話的に設定するには、次のコマンドを実行します。

```
# alternatives --config python
```

2. 表示されたリストから必要なバージョンを選択します。
3. この設定をリセットし、バージョンを指定しない **python** コマンドを削除するには、次のコマンドを実行します。

```
# alternatives --auto python
```

19.3.3. 関連情報

- man ページの **alternatives(8)** および **unversioned-python(1)**

19.4. PYTHON 3 RPM のパッケージ化

ほとんどの Python プロジェクトは、パッケージ化に Setuptools を使用して、**setup.py** ファイルにパッケージ情報を定義します。Setuptools パッケージ化の詳細は Setuptools ドキュメント を参照してください。

Python プロジェクトを RPM パッケージにパッケージ化することもできます。これには、Setuptools パッケージ化と比較して以下の利点があります。

- その他の RPM のパッケージの依存関係の指定 (Python 以外も含む)

- 電子署名
  電子署名を使用すると、RPM パッケージの内容は、オペレーティングシステムのその他の部分とともに検証、統合、およびテストできます。

19.4.1. Python パッケージ用の SPEC ファイルの説明

SPEC ファイルには、RPM のビルドに **rpmbuild** ユーティリティーを使用する命令が含まれています。命令は、一連のセクションに含まれています。SPEC ファイルには、セクションが定義されている 2 つの主要部分があります。

- プリアンブル (ボディーに使用されている一連のメタデータ項目が含まれています)
- ボディー (命令の主要部分が含まれています)

Python プロジェクトの RPM SPEC ファイルには、非 Python RPM SPEC ファイルと比較していくつかの詳細があります。特に注目すべきは、Python ライブラリーの RPM パッケージ名に、Python 3.6 の場合は **python3**、Python 3.8 の場合は **python38**、Python 3.9 の場合は **python39**、Python 3.11 の場合は **python3.11** など、バージョンを判別する接頭辞を常に含める必要があります。

その他の詳細は、次の SPEC ファイルの **python3-detox パッケージの例** に記載されています。その詳細の説明は、例の下に記載されている注意事項を参照してください。

```
%global modname detox                                                    1

Name:           python3-detox                                            2
Version:        0.12
Release:        4%{?dist}
Summary:        Distributing activities of the tox tool
License:        MIT
URL:            https://pypi.io/project/detox
Source0:        https://pypi.io/packages/source/d/%{modname}/%{modname}-%{version}.tar.gz

BuildArch:      noarch

BuildRequires:  python36-devel                                           3
BuildRequires:  python3-setuptools
BuildRequires:  python36-rpm-macros
BuildRequires:  python3-six
BuildRequires:  python3-tox
BuildRequires:  python3-py
BuildRequires:  python3-eventlet

%?python_enable_dependency_generator                                     4

%description

Detox is the distributed version of the tox python testing tool. It makes efficient use of multiple CPUs
by running all possible activities in parallel.
Detox has the same options and configuration that tox has, so after installation you can run it in the
same way and with the same options that you use for tox.

    $ detox

%prep
%autosetup -n %{modname}-%{version}
```

```
%build
%py3_build                                    5

%install
%py3_install

%check
%{__python3} setup.py test                    6

%files -n python3-%{modname}
%doc CHANGELOG
%license LICENSE
%{_bindir}/detox
%{python3_sitelib}/%{modname}/
%{python3_sitelib}/%{modname}-%{version}*

%changelog
...
```

1. **modname** マクロには、Python プロジェクトの名前が含まれます。この例では **detox** となります。

2. Python プロジェクトを RPM にパッケージ化する場合は、常にプロジェクトの元の名前に接頭辞 **python3** を追加する必要があります。ここでの元の名前は **detox** で、**RPM の名前** は **python3-detox** です。

3. **BuildRequires** は、このパッケージのビルドおよびテストに必要なパッケージを指定します。BuildRequires では、Python パッケージをビルドするのに必要なツールを提供する項目 (**python36-devel** および **python3-setuptools**) が常に含まれます。**/usr/bin/python3** インタープリターディレクティブを持つファイルが自動的に **/usr/bin/python3.6** に変更されるように、**python36-rpm-macros** パッケージが必要です。

4. すべての Python パッケージが正しく動作するためには、その他のパッケージがいくつか必要です。このようなパッケージも、SPEC ファイルで指定する必要があります。**依存関係**を指定するには、**%python_enable_dependency_generator** マクロを使用して、**setup.py** ファイルに定義した依存関係を自動的に使用できます。パッケージに、Setuptools で指定していない依存関係がある場合は、追加の **Requires** ディレクティブ内に指定します。

5. **%py3_build** マクロおよび **%py3_install** マクロは、**setup.py build** コマンドおよび **setup.py install** コマンドを実行します。それぞれには、インストール場所、使用するインタープリター、その他の詳細を指定する引数を用います。

6. **check** セクションは、Python の正しいバージョンを実行するマクロを提供します。**%{__python3}** マクロには、Python 3 インタープリターのパス (**/usr/bin/python3** など) が含まれます。リテラルパスではなく、マクロを使用することが常に推奨されます。

19.4.2. Python 3 RPM の一般的なマクロ

SPEC ファイルでは、値をハードコーディングするのではなく、以下の **Python 3 RPM のマクロ** の表で説明されているマクロを常に使用します。

マクロ名では、バージョンを指定しない **python** ではなく、**python3** または **python2** を使用してください。SPEC ファイルの BuildRequires セクションで特定の Python 3 バージョンを python36-rpm-macros、python38-rpm-macros、**python39-rpm-macros**、または **python3.11-rpm-macros** に設定し

ます。

表19.2 Python 3 RPM 用のマクロ

マクロ	一般的な定義	説明
%{__python3}	/usr/bin/python3	Python 3 のインタープリター
%{python3_version}	3.6	Python 3 インタープリターのフルバージョン
%{python3_sitelib}	/usr/lib/python3.6/site-packages	pure-Python モジュールのインストール先
%{python3_sitearch}	/usr/lib64/python3.6/site-packages	アーキテクチャー固有の拡張を含むモジュールがインストールされている場合
%py3_build		システムパッケージに適した引数で **setup.py build** コマンドを実行します。
%py3_install		システムパッケージに適した引数で **setup.py install** コマンドを実行します。

19.4.3. Python RPM の自動 Provides

Python プロジェクトをパッケージ化する際、以下のディレクトリーが存在する場合は、作成される RPM に以下のディレクトリーが含まれていることを確認してください。

- **.dist-info**
- **.egg-info**
- **.egg-link**

このディレクトリーから、RPM ビルドプロセスは自動的に仮想 **pythonX.Ydist** Provides (**python3.6dist(detox)** など) を生成します。この仮想 Provides は、**%python_enable_dependency_generator** マクロにより指定されるパッケージにより提供されます。

19.5. PYTHON スクリプトでのインタープリターディレクティブの処理

Red Hat Enterprise Linux 8 では、実行可能な Python スクリプトは、少なくとも主要な Python バージョンを明示的に指定するインタープリターディレクティブ (別名 hashbangs または shebangs) を使用することが想定されます。以下に例を示します。

```
#!/usr/bin/python3
#!/usr/bin/python3.6
#!/usr/bin/python3.8
#!/usr/bin/python3.9
```

```
#!/usr/bin/python3.11
#!/usr/bin/python3.12
#!/usr/bin/python2
```

/usr/lib/rpm/redhat/brp-mangle-shebangs BRP (buildroot policy) スクリプトは、RPM パッケージをビルドする際に自動的に実行され、実行可能なすべてのファイルでインタープリターディレクティブを修正しようとします。

BRP スクリプトは、以下のようにあいまいなインタープリターディレクティブを含む Python スクリプトを検出すると、エラーを生成します。

```
#!/usr/bin/python
```

または

```
#!/usr/bin/env python
```

19.5.1. Python スクリプトでインタープリターディレクティブの変更

RPM ビルド時にビルドエラーが発生する Python スクリプト内のインタープリターディレクティブを変更します。

前提条件

- Python スクリプトのインタープリターディレクティブの一部でビルドエラーが発生する。

手順

インタープリターディレクティブを変更するには、以下のタスクのいずれかを実行します。

- **platform-python-devel** パッケージから **pathfix.py** スクリプトを適用します。

  ```
  # pathfix.py -pn -i %{__python3} PATH ...
  ```

  複数の **PATH** を指定できます。**PATH** がディレクトリーの場合、**pathfix.py** はあいまいなインタープリターディレクティブを持つスクリプトだけでなく、**^[a-zA-Z0-9_]+\.py$** のパターンに一致する Python スクリプトを再帰的にスキャンします。このコマンドを **%prep** セクション、または **%install** セクションに追加します。

- パッケージ化した Python スクリプトを、想定される形式に準拠するように変更します。この目的のために、**pathfix.py** は、RPM ビルドプロセス以外でも使用できます。**pathfix.py** を RPM ビルド以外で実行する場合は、上記の例の **%{__python3}** を、**/usr/bin/python3** などのインタープリターディレクティブのパスに置き換えます。

パッケージ化された Python スクリプトに Python 3.6 以外のバージョンが必要な場合は、上記のコマンドを調整して必要なバージョンを含めます。

19.5.2. カスタムパッケージの /usr/bin/python3 インタープリターディレクティブの変更

デフォルトでは、**/usr/bin/python3** の形式でのインタープリターディレクティブは、Red Hat Enterprise Linux のシステムツールに使用される **platform-python** パッケージから Python を参照するインタープリターディレクティブに置き換えられます。カスタムパッケージの **/usr/bin/python3** イン

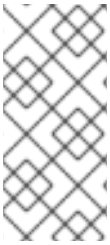
タープリターディレクティブを変更して、AppStream リポジトリーからインストールした特定バージョンの Python を参照できます。

手順

- Python の特定バージョンのパッケージを構築するには、対応する **python** パッケージの **python*-rpm-macros** サブパッケージを SPEC ファイルの **BuildRequires** セクションに追加します。たとえば、Python 3.6 の場合は、以下の行を追加します。

```
BuildRequires:  python36-rpm-macros
```

これにより、カスタムパッケージの **/usr/bin/python3** インタープリターディレクティブは、自動的に **/usr/bin/python3.6** に変換されます。



注記

BRP スクリプトがインタープリターディレクティブを確認したり、変更したりしないようにするには、以下の RPM ディレクティブを使用します。

```
%undefine __brp_mangle_shebangs
```

19.6. PHP スクリプト言語の使用

Hypertext Preprocessor (PHP) はサーバー側のスクリプトに主に使用する汎用スクリプト言語で、Web サーバーを使用して PHP コードを実行できるようにします。

RHEL 8 では、PHP スクリプト言語は **php** モジュールにより提供されます。これは、複数のストリーム (バージョン) で利用できます。

ユースケースによっては、選択したモジュールストリームの特定のプロファイルをインストールできます。

- **common**: Web サーバーを使用したサーバー側のスクリプトのデフォルトプロファイル。これには、広範に使用される拡張機能が複数含まれています。
- **minimal**: このプロファイルは、Web サーバーを使用せずに PHP でのスクリプト用のコマンドラインインターフェイスのみをインストールします。
- **devel**: このプロファイルには、**共通** プロファイルのパッケージと開発用の追加パッケージが含まれます。

19.6.1. PHP スクリプト言語のインストール

選択したバージョンの **php** モジュールをインストールできます。

手順

- デフォルトのプロファイルで **php** モジュールストリームをインストールするには、以下を使用します。

```
# yum module install php:stream
```

stream は、インストールする PHP のバージョンに置き換えます。

たとえば、PHP 8.0 をインストールするには、以下を実行します。

```
# yum module install php:8.0
```

デフォルトの **共通** プロファイルは **php-fpm** パッケージもインストールし、**Apache HTTP Server** または **nginx** で使用する PHP を事前設定します。

- **php** モジュールストリームの特定のプロファイルをインストールするには、以下を使用します。

```
# yum module install php:stream/profile
```

stream は、インストールする **プロファイル** の名前に置き換えます。

たとえば、Web サーバーを使用しない PHP 8.0 をインストールするには、以下を実行します。

```
# yum module install php:8.0/minimal
```

関連情報

- RHEL 8 で利用可能な以前のバージョンの PHP からアップグレードする場合は、後続のストリームへの切り替え を参照してください。
- RHEL 8 モジュールおよびストリームの詳細は、ユーザー空間コンポーネントのインストール、管理、および削除 を参照してください。

19.6.2. Web サーバーでの PHP スクリプト言語の使用

19.6.2.1. Apache HTTP Server での PHP の使用

Red Hat Enterprise Linux 8 では、**Apache HTTP Server** で PHP を FastCGI プロセスサーバーとして実行できます。FastCGI Process Manager (FPM) は、Web サイトで高負荷を管理できるようにする代替の PHP FastCGI デーモンです。RHEL 8 では、PHP はデフォルトで FastCGI Process Manager を使用します。

FastCGI プロセスサーバーを使用して PHP コードを実行できます。

前提条件

- PHP スクリプト言語がシステムにインストールされている。
  PHP スクリプト言語のインストール を参照してください。

手順

1. **httpd** モジュールをインストールします。

```
# yum module install httpd:2.4
```

2. **Apache HTTP Server** を起動します。

```
# systemctl start httpd
```

または、**Apache HTTP Server** をシステムで実行している場合は、PHP のインストール後に **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

3. **php-fpm** サービスを起動します。

```
# systemctl start php-fpm
```

4. 必要に応じて、両方のサービスが起動時に開始できるようにします。

```
# systemctl enable php-fpm httpd
```

5. PHP の設定に関する情報を取得するには、以下の内容を含む **index.php** ファイルを **/var/www/html/** ディレクトリーに作成します。

```
# echo '<?php phpinfo(); ?>' > /var/www/html/index.php
```

6. **index.php** ファイルを実行するには、ブラウザーで以下を指定します。

```
http://<hostname>/
```

7. オプション: 特定の要件がある場合は、設定を調整します。

- **/etc/httpd/conf/httpd.conf** - 一般的な **httpd** 設定
- **/etc/httpd/conf.d/php.conf** - **httpd** の PHP 固有の設定
- **/usr/lib/systemd/system/httpd.service.d/php-fpm.conf** - デフォルトでは、**php-fpm** サービスは **httpd** と一緒に起動します。
- **/etc/php-fpm.conf** - FPM の主な設定
- **/etc/php-fpm.d/www.conf** - デフォルトの **www** プール設定

例19.1 "Hello, World!" の実行Apache HTTP Server を使用した PHP スクリプト

1. **/var/www/html/** ディレクトリーにプロジェクト用の **hello** ディレクトリーを作成します。

```
# mkdir hello
```

2. 以下の内容を含む **/var/www/html/hello/** ディレクトリーに **hello.php** ファイルを作成します。

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
    echo 'Hello, World!';
```

```
?>
</body>
</html>
```

3. **Apache HTTP Server** を起動します。

```
# systemctl start httpd
```

4. **hello.php** ファイルを実行するには、ブラウザーに以下を指定します。

```
http://<hostname>/hello/hello.php
```

結果として、"Hello, World!" というテキストを含む Web ページが表示されます。

関連情報

- Apache HTTP Web サーバーの設定

19.6.2.2. nginx Web サーバーでの PHP の使用

nginx Web サーバーを介して PHP コードを実行できます。

前提条件

- PHP スクリプト言語がシステムにインストールされている。
  PHP スクリプト言語のインストール を参照してください。

手順

1. **nginx** モジュールストリームをインストールします。

```
# yum module install nginx:stream
```

stream は、インストールする **nginx** のバージョンに置き換えます。

たとえば、**nginx** バージョン 1.18 をインストールするには、以下を実行します。

```
# yum module install nginx:1.18
```

2. **nginx** サーバーを起動します。

```
# systemctl start nginx
```

または、使用中のシステムで **nginx** サーバーを実行している場合は、PHP のインストール後に **nginx** サービスを再起動します。

```
# systemctl restart nginx
```

3. **php-fpm** サービスを起動します。

```
# systemctl start php-fpm
```

4. 必要に応じて、両方のサービスが起動時に開始できるようにします。

```
# systemctl enable php-fpm nginx
```

5. PHP の設定に関する情報を取得するには、以下の内容を含む **index.php** ファイルを **/usr/share/nginx/html/** ディレクトリーに作成します。

```
# echo '<?php phpinfo(); ?>' > /usr/share/nginx/html/index.php
```

6. **index.php** ファイルを実行するには、ブラウザーで以下を指定します。

```
http://<hostname>/
```

7. オプション: 特定の要件がある場合は、設定を調整します。
   - **/etc/nginx/nginx.conf** - **nginx** main configuration
   - **/etc/nginx/conf.d/php-fpm.conf** - **nginx** の FPM 設定
   - **/etc/php-fpm.conf** - FPM の主な設定
   - **/etc/php-fpm.d/www.conf** - デフォルトの **www** プール設定

例19.2 "Hello, World!" の実行nginx サーバーを使用した PHP スクリプト

1. プロジェクトの **hello** ディレクトリーを **/usr/share/nginx/html/** ディレクトリーに作成します。

```
# mkdir hello
```

2. 以下の内容で **/usr/share/nginx/html/hello/** ディレクトリーに **hello.php** ファイルを作成します。

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
    echo 'Hello, World!';
?>
</body>
</html>
```

3. **nginx** サーバーを起動します。

```
# systemctl start nginx
```

4. **hello.php** ファイルを実行するには、ブラウザーに以下を指定します。

```
http://<hostname>/hello/hello.php
```

   結果として、"Hello, World!" というテキストを含む Web ページが表示されます。

関連情報

- NGINX の設定および設定

19.6.3. コマンドラインインターフェイスを使用した PHP スクリプトの実行

通常、PHP スクリプトは Web サーバーを使用して実行されますが、コマンドラインインターフェイスを使用して実行することも可能です。

コマンドラインのみを使用して **php** スクリプトを実行する場合は、**php** モジュールストリームの **最低限** のプロファイルをインストールします。

PHP スクリプト言語のインストール を参照してください。

前提条件

- PHP スクリプト言語がシステムにインストールされている。
  PHP スクリプト言語のインストール を参照してください。

手順

1. テキストエディターで **filename.php** ファイルを作成します。
   filename は、使用するファイル名に置き換えます。
2. コマンドラインから、作成した **filename.php** ファイルを実行します。

```
# php filename.php
```

例19.3 "Hello, World!" の実行コマンドラインインターフェイスを使用した PHP スクリプト

1. テキストエディターを使用して、以下の内容で **hello.php** ファイルを作成します。

```
<?php
    echo 'Hello, World!';
?>
```

2. コマンドラインで **hello.php** ファイルを実行します。

```
# php hello.php
```

結果として、"Hello, World!" が出力されます。

19.6.4. 関連情報

- **httpd(8)** - **httpd** サービスの man ページ。コマンドラインオプションの全リストが記載されています。
- **httpd.conf (5)** - **httpd** 設定の man ページ。**httpd** 設定ファイルの構造と場所が説明されています。

- **nginx(8)** - **nginx** Web サーバーの man ページ。コマンドラインオプションの全リストとシグナルのリストが記載されています。
- **php-fpm(8)** - PHP FPM の man ページ。コマンドラインオプションおよび設定ファイルの全リストが記載されています。

19.7. TCL/TK の使用

19.7.1. Tcl/Tk の概要

Tool command language (Tcl) は、動的なプログラミング言語です。この言語のインタープリターと C ライブラリーは、**tcl** パッケージにより提供されます。

Tk とともに **Tcl** を使用すると (**Tcl/Tk**)、プラットフォーム間共通の GUI アプリケーションを作成できます。**Tk** は、**tk** パッケージから入手できます。

Tk は次のいずれかを参照できることに注意してください。

- 複数言語のプログラミングツールキット
- Tk C ライブラリーバインディングは、複数の言語 (C、Ruby、Perl、Python など) で利用できます。
- Tk コンソールのインスタンスを作成する wish インタープリター
- 特定の Tcl インタープリターに新しいコマンドを多数追加する Tk の拡張

Tcl/Tk の詳細は Tcl/Tk マニュアル または Tcl/Tk ドキュメントの Web ページ を参照してください。

19.7.2. Tcl/Tk 8.6 に関する注目すべき変更点

Red Hat Enterprise Linux 7 では **Tcl/Tk 8.5** が使用されていました。Red Hat Enterprise Linux 8 の **Tcl/Tk バージョン 8.6** は、Base OS リポジトリーで提供されます。

Tcl/Tk 8.5 と比較した **Tcl/Tk 8.6** における主な変更点は以下の通りです。

- オブジェクト指向のプログラミングサポート
- スタックレス評価の実装
- 強化された例外処理
- Tcl で構築およびインストールしたサードパーティーパッケージのコレクション
- 有効なマルチスレッド操作
- SQL データベースを提供するスクリプトサポート
- IPv6 ネットワーキングサポート
- ビルドインの zlib 圧縮
- リスト処理
  新しい 2 つのコマンド **lmap** および **dict map** が利用できます。これにより、**Tcl** コンテナーにおける変換の表現が可能になります。

- スクリプトにより積み上げられたチャンネル
  新しい 2 つのコマンド **chan push** および **chan pop** が利用できるため、I/O チャンネルへ、または I/O チャンネルからの変換を追加または削除できます。

Tk の主な変更は、以下のようになります。

- ビルドイン PNG イメージサポート
- ビジーウィンドウ
  新しいコマンド **tk busy** が利用できます。これは、ウィンドウまたはウィジェットのユーザーとの対話を無効にし、ビジーカーソルが表示されます。
- 新しいフォント選択ダイアログインターフェイス
- 角度のついたテキストサポート
- キャンバスサポートでの移動

Tcl 8.5 から **Tcl 8.6** への変更点の詳細は、Changes in Tcl/Tk 8.6 を参照してください。

19.7.3. Tcl/Tk 8.6 への移行

Red Hat Enterprise Linux 7 では **Tcl/Tk 8.5** が使用されていました。Red Hat Enterprise Linux 8 の **Tcl/Tk バージョン 8.6** は、Base OS リポジトリーで提供されます。

本セクションでは、以下を対象に、**Tcl/Tk 8.6** への移行パスを説明します。

- **Tcl** 拡張を記述する、または **Tcl** インタープリターをアプリケーションに組み込む開発者
- **Tcl/Tk** を使用してタスクのスクリプトを作成するユーザー

19.7.3.1. Tcl 拡張機能の開発者のための移行パス

コードを **Tcl 8.6** と互換性を持たせるには、以下の手順に従います。

手順

1. コードを書き直し、**interp** 構造を使用するようにします。たとえば、コードが **interp→errorLine** を読み込む場合は、これを書き直して以下の関数を使用します。

   ```
   Tcl_GetErrorLine(interp)
   ```

   Tcl 8.6 は、**interp** 構造のメンバーへの直接アクセスを制限するため、これが必要です。

2. コードを、**Tcl 8.5** および **Tcl 8.6** の両方と互換性を持たせるには、C または C++ アプリケーションのヘッダーファイル、または **Tcl** ライブラリーを含む拡張に、以下のコードスニペットを使用してください。

   ```
   # include <tcl.h>
   # if !defined(Tcl_GetErrorLine)
   # define Tcl_GetErrorLine(interp) (interp→errorLine)
   # endif
   ```

19.7.3.2. Tcl/Tk を使用してタスクのスクリプトを作成したユーザーのパスの移行

Tcl 8.6 では、ほとんどのスクリプトが、以前のバージョン Tcl と同じように動作します。

コードを Tcl 8.6 に移行するには、以下の手順を使用します。

手順

- ポータブルなコードを記載する場合は、Tk 8.6 でサポートされないコマンドを使用しないようにする必要があります。

```
tkIconList_Arrange
tkIconList_AutoScan
tkIconList_Btn1
tkIconList_Config
tkIconList_Create
tkIconList_CtrlBtn1
tkIconList_Curselection
tkIconList_DeleteAll
tkIconList_Double1
tkIconList_DrawSelection
tkIconList_FocusIn
tkIconList_FocusOut
tkIconList_Get
tkIconList_Goto
tkIconList_Index
tkIconList_Invoke
tkIconList_KeyPress
tkIconList_Leave1
tkIconList_LeftRight
tkIconList_Motion1
tkIconList_Reset
tkIconList_ReturnKey
tkIconList_See
tkIconList_Select
tkIconList_Selection
tkIconList_ShiftBtn1
tkIconList_UpDown
```

サポートされていないコマンドのリストは、**/usr/share/tk8.6/unsupported.tcl** ファイルで確認できます。