Red Hat Enterprise Linux 8

システムデザインガイド

RHEL 8 システムの設計

Last Updated: 2023-10-03

Red Hat Enterprise Linux 8 システムデザインガイド

RHEL 8 システムの設計

法律上の通知

概要

このコンテンツでは、Red Hat Enterprise Linux 8 の使用を開始する方法について説明します。Red Hat Enterprise Linux テクノロジーの機能と制限について
は、https://access.redhat.com/articles/rhel-limits を参照してください。

目次

多様性を受け入れるオープンソースの強化

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

RED HAT ドキュメントへのフィードバック (英語のみ)

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

特定の文章に関するコメントの送信

1. **Multi-page HTML** 形式でドキュメントを表示し、ページが完全にロードされてから右上隅に **Feedback** ボタンが表示されていることを確認します。
2. カーソルを使用して、コメントを追加するテキスト部分を強調表示します。
3. 強調表示されたテキストの近くに表示される **Add Feedback** ボタンをクリックします。
4. フィードバックを追加し、**Submit** をクリックします。

Bugzilla からのフィードバック送信 (アカウントが必要)

1. Bugzilla の Web サイトにログインします。
2. **Version** メニューから正しいバージョンを選択します。
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. **Submit Bug** をクリックします。

パート I. インストールの設計

第1章 サポート対象の RHEL アーキテクチャーおよびシステム要件

Red Hat Enterprise Linux 8 は、ワークロードの提供にかかる時間や労力の軽減に必要なツールを使用することで、ハイブリッドクラウドデプロイメント全体に安定性、安全性、一貫性のある基盤を提供します。RHEL は、対応しているハイパーバイザー環境やクラウドプロバイダー環境にゲストとしてデプロイすることも、物理インフラストラクチャーにデプロイすることもできるため、アプリケーションは、主要なハードウェアアーキテクチャープラットフォームの革新的な機能を利用できます。

1.1. サポート対象のアーキテクチャー

Red Hat Enterprise Linux では、次のアーキテクチャーに対応します。

- AMD、Intel、および ARM 64 ビットアーキテクチャー
- IBM Power Systems (リトルエンディアン)
  - IBM Power System LC サーバー
  - IBM Power System AC サーバー
  - IBM Power System L サーバー
- 64 ビット IBM Z

注記

IBM Power Server へのインストール手順は、**IBM installation documentation** を参照してください。システムで RHEL のインストールがサポートされていることを確認するには、https://catalog.redhat.com および https://access.redhat.com/articles/rhel-limits を参照してください。

1.2. システム要件

Red Hat Enterprise Linux を初めてインストールする場合は、インストールの前にシステム、ハードウェア、セキュリティー、メモリー、および RAID に関するガイドラインを確認することが推奨されます。詳細は、システム要件の参照 を参照してください。

システムを仮想ホストとして使用する場合は、仮想化に必要なハードウェア要件 を確認してください。

関連情報

- セキュリティーの強化
- RHEL システムイメージのカスタマイズ

第2章 インストールの準備

Red Hat Enterprise Linux をインストールする前に、以下のセクションを参照してインストールのセットアップを準備します。

2.1. 推奨される手順

RHEL インストールの準備は、以下の手順で設定されます。

手順

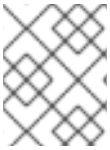
1. インストール方法を確認し、決定します。
2. システム要件の確認
3. インストール起動用メディアのオプションを確認します。
4. 必要なインストール ISO イメージをダウンロードします。
5. 起動可能なインストールメディアを作成します。
6. インストールソース* を準備します。

*コンテンツ配信ネットワーク (CDN) を使用して必要なソフトウェアパッケージをダウンロードしていない場合は、Boot ISO (最小インストール) イメージにのみ必要です。

2.2. RHEL のインストール方法

Red Hat Enterprise Linux は、以下のいずれかの方法でインストールできます。

- GUI ベースのインストール
- システムまたはクラウドイメージベースのインストール
- 高度なインストール



注記

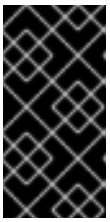
本ガイドでは、ユーザーインターフェイス (GUI) を使用した RHEL のインストール方法を説明します。

GUI ベースのインストール

以下の GUI ベースのインストール方法から選択できます。

- **カスタマーポータルから ISO イメージを使用した RHEL のインストール**:カスタマーポータルから **DVD ISO** イメージファイルをダウンロードして Red Hat Enterprise Linux をインストールします。登録は、GUI インストールの完了後に行われます。このインストール方法は、キックスタートでも対応しています。
- **コンテンツ配信ネットワークからの RHEL の登録およびインストール**コンテンツ配信ネットワーク (CDN) から、システムを登録し、サブスクリプションを割り当て、Red Hat Enterprise Linux をインストールします。このインストール方法は、**Boot ISO** イメージファイルおよび **DVD ISO** イメージファイルに対応します。ただし、Boot ISO イメージファイルのインストー

ルソースのデフォルトは CDN であるため、**Boot ISO** イメージファイルが推奨されます。システムの登録後、インストーラーは CDN からパッケージをダウンロードしてインストールします。このインストール方法は、キックスタートでも対応しています。



重要

GUI で、特定の要件に合わせて RHEL インストールをカスタマイズできます。特定の環境要件 (Red Hat への接続、ソフトウェア選択、パーティション設定、セキュリティーなど) の追加オプションを選択できます。詳細は、インストールのカスタマイズ を参照してください。

システムまたはクラウドイメージベースのインストール

システムまたはクラウドイメージベースのインストール方法は、仮想環境およびクラウド環境でのみ使用できます。

システムまたはクラウドイメージベースのインストールを実行するには、Red Hat Image Builder を使用します。Image Builder は、クラウドデプロイメントのシステムイメージを含む、Red Hat Enterprise Linux のカスタマイズされたシステムイメージを作成します。

Image Builder を使用して RHEL をインストールする方法の詳細は、RHEL システムイメージのカスタマイズ を参照してください。

高度なインストール

以下の高度なインストール方法から選択できます。

- **キックスタートを使用した RHEL の自動インストールを実行します。**キックスタートは、ファイルの要件と設定をすべて指定して、オペレーティングシステムのインストールに役立つ自動化されたプロセスです。キックスタートファイルには、RHEL インストールオプション (タイムゾーン、ドライブパーティション、インストールするパッケージなど) が含まれます。事前に準備したキックスタートファイルを使用すると、ユーザーによる操作を必要とせずにインストールが完了します。これは、一度に多数のシステムに Red Hat Enterprise Linux をデプロイする場合に便利です。

- **VNC を使用したリモート RHEL インストールの実行**RHEL インストールプログラムは、2 つの Virtual Network Computing (VNC) インストールモードを提供します。Direct および Connect です。接続が確立されると、2 つのモードに違いはありません。選択するモードは、環境によって異なります。

- **PXE を使用して、ネットワークから RHEL をインストール**PXE (Preboot eXecution Environment) を使用するネットワークインストールでは、インストールサーバーへのアクセスがあるシステムに、Red Hat Enterprise Linux をインストールできます。ネットワークインストールには、少なくとも 2 つのシステムが必要です。

関連情報

- 高度なインストール方法の詳細は、**高度な RHEL 8 インストールの実行** を参照してください。

2.3. システム要件

Red Hat Enterprise Linux を初めてインストールする場合は、インストールの前にシステム、ハードウェア、セキュリティー、メモリー、および RAID に関するガイドラインを確認することが推奨されます。詳細は、システム要件の参照 を参照してください。

システムを仮想ホストとして使用する場合は、仮想化に必要なハードウェア要件 を確認してください。

関連情報

- セキュリティーの強化
- RHEL システムイメージのカスタマイズ

2.4. インストール起動用メディアオプション

Red Hat Enterprise Linux インストールプログラムを起動する方法はいくつかあります。

フルインストール用 DVD または USB フラッシュドライブ

DVD ISO イメージを使用して、フルインストールの DVD または USB フラッシュドライブを作成します。ソフトウェアパッケージをインストールする場合は、DVD または USB フラッシュドライブを、ブートデバイスおよびインストールソースとして使用できます。

最小インストール用の DVD、CD、または USB フラッシュドライブ

最小インストール用 CD、DVD、または USB フラッシュドライブは、**Boot ISO** イメージを使用して作成されます。これには、システムを起動し、インストールプログラムを開始するのに最低限必要なファイルのみが含まれます。

重要

コンテンツ配信ネットワーク (CDN) を使用して必要なソフトウェアパッケージをダウンロードする場合は、**Boot ISO** イメージに、必要なソフトウェアパッケージを含むインストールソースが必要です。

PXE サーバー

PXE (preboot execution environment) サーバーを使用すると、ネットワーク経由でインストールプログラムを起動できます。システムが起動したら、ローカルのハードドライブやネットワーク経由など、別のインストールソースからインストールを完了します。

Image Builder

Image Builder を使用すると、システムおよびクラウドイメージをカスタマイズして、仮想環境およびクラウド環境に Red Hat Enterprise Linux をインストールできます。

関連情報

- 高度な RHEL 8 インストールの実行
- RHEL システムイメージのカスタマイズ

2.5. インストール ISO イメージの種類

Red Hat カスタマーポータルでは、2 種類の Red Hat Enterprise Linux 8 インストール ISO イメージが利用できます。

DVD ISO イメージファイル

これは、BaseOS リポジトリーおよび AppStream リポジトリーを含む完全なインストールプログラムです。DVD ISO ファイルを使用すると、追加のリポジトリーにアクセスせずにインストールを完了できます。

重要

Binary DVD は、64 ビットの IBM Z でもご利用になれます。SCSI DVD ドライブを使用してインストールプログラムを起動したり、インストールソースとしても使用できます。

Boot ISO イメージファイル

Boot ISO イメージは、以下のような方法で RHEL をインストールするのに使用できる最小限のインストールです。

a. コンテンツ配信ネットワーク (CDN) から RHEL を登録してインストールする場合。

b. ソフトウェアパッケージをインストールするのに、BaseOS リポジトリーおよび AppStream リポジトリーにアクセスする必要がある最小限のイメージとして。リポジトリーは、Red Hat カスタマーポータル からダウンロードできる DVD ISO イメージに含まれます。DVD ISO イメージをダウンロードしてデプロイメントし、リポジトリーにアクセスします。

次の表に、サポートされているアーキテクチャーで利用可能なイメージに関する情報を示します。

表2.1 起動用およびインストール用のイメージ

アーキテクチャー	インストール DVD	ブート DVD
AMD64 および Intel 64	x86_64 DVD ISO イメージファイル	x86_64 Boot ISO イメージファイル
ARM 64	AArch64 DVD ISO イメージファイル	AArch64 Boot ISO イメージファイル
IBM POWER	ppc64le DVD ISO イメージファイル	ppc64le Boot ISO イメージファイル
64 ビット IBM Z	s390x DVD ISO イメージファイル	s390x Boot ISO イメージファイル

2.6. RHEL インストール ISO イメージのダウンロード

Red Hat Enterprise Linux は、Red Hat カスタマーポータル にアクセスするか、**curl** コマンドを使用してダウンロードできます。

2.6.1. インストール ISO イメージの種類

Red Hat カスタマーポータルでは、2 種類の Red Hat Enterprise Linux 8 インストール ISO イメージが利用できます。

DVD ISO イメージファイル

これは、BaseOS リポジトリーおよび AppStream リポジトリーを含む完全なインストールプログラムです。DVD ISO ファイルを使用すると、追加のリポジトリーにアクセスせずにインストールを完了できます。

重要

Binary DVD は、64 ビットの IBM Z でもご利用になれます。SCSI DVD ドライブを使用してインストールプログラムを起動したり、インストールソースとしても使用できます。

Boot ISO イメージファイル

Boot ISO イメージは、以下のような方法で RHEL をインストールするのに使用できる最小限のインストールです。

a. コンテンツ配信ネットワーク (CDN) から RHEL を登録してインストールする場合。

b. ソフトウェアパッケージをインストールするのに、BaseOS リポジトリーおよび AppStream リポジトリーにアクセスする必要がある最小限のイメージとして。リポジトリーは、Red Hat カスタマーポータル からダウンロードできる DVD ISO イメージに含まれます。DVD ISO イメージをダウンロードしてデプロイメントし、リポジトリーにアクセスします。

次の表に、サポートされているアーキテクチャーで利用可能なイメージに関する情報を示します。

表2.2 起動用およびインストール用のイメージ

アーキテクチャー	インストール DVD	ブート DVD
AMD64 および Intel 64	x86_64 DVD ISO イメージファイル	x86_64 Boot ISO イメージファイル
ARM 64	AArch64 DVD ISO イメージファイル	AArch64 Boot ISO イメージファイル
IBM POWER	ppc64le DVD ISO イメージファイル	ppc64le Boot ISO イメージファイル
64 ビット IBM Z	s390x DVD ISO イメージファイル	s390x Boot ISO イメージファイル

2.6.2. カスタマーポータルから ISO イメージのダウンロード

Boot ISO イメージは、システムの登録、サブスクリプションの割り当て、およびコンテンツ配布ネットワーク (CDN) からの RHEL のインストールに対応する最小限のイメージファイルです。DVD ISO イメージファイルには、リポジトリーとソフトウェアパッケージがすべて含まれ、追加設定は必要ありません。

前提条件

- アクティブな Red Hat サブスクリプションがある。
- Product Downloads の Red Hat カスタマーポータルの **Product Downloads** セクションにログインしている。

手順

1. ブラウザーを開いて https://access.redhat.com/downloads/content/rhel にアクセスします。このページには、Red Hat Enterprise Linux の人気のあるダウンロードがリストされています。
2. 必要な ISO イメージの横にある **今すぐダウンロードする** をクリックします。
3. 目的のバージョンの RHEL がリストにない場合は、**All Red Hat Enterprise Linux Downloads** をクリックします。
   a. **Product Variant** ドロップダウンメニューから、必要なバリアントとアーキテクチャーを選択します。
      - オプション: **パッケージ** タブを選択して、選択したバリアントに含まれるパッケージを表示します。Red Hat Enterprise Linux 8 で利用可能なパッケージについては、パッケージマニフェスト ドキュメントを参照してください。
   b. **Version** ドロップダウンメニューから、ダウンロードする RHEL バージョンを選択します。デフォルトでは、選択したバリアントとアーキテクチャーの最新バージョンが選択されています。
   Product Software タブには以下のようなイメージファイルがあります。
      - **Red Hat Enterprise Linux Binary DVD**イメージ
      - **Red Hat Enterprise Linux Boot ISO**イメージ

      他のイメージ (たとえば、事前設定されている仮想マシンイメージ) も利用できます。
   c. 必要な ISO イメージの横にある **今すぐダウンロードする** をクリックします。

2.6.3. curl で ISO イメージのダウンロード

curl ツールを使用すると、コマンドラインを使用して Web から必要なファイルを取得し、ローカルに保存するか、必要に応じて別のプログラムにパイプできます。本セクションでは、**curl** コマンドを使用してインストールイメージをダウンロードする方法を説明します。

前提条件

- **curl** パッケージおよび **jq** パッケージがインストールされている。
  Linux ディストリビューションで **yum** または **apt** を使用していない場合、もしくは Linux を使用していない場合は、curl の Web サイト から、最適なソフトウェアパッケージをダウンロードします。
- Red Hat API トークン から生成したオフライントークンがある。
- 製品のダウンロード からダウンロードするファイルのチェックサムがある。

手順

1. 以下の内容で bash ファイルを作成します。

```
#!/bin/bash
# set the offline token and checksum parameters
offline_token="<offline_token>"
checksum=<checksum>

# get an access token
access_token=$(curl https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-
```

```
connect/token -d grant_type=refresh_token -d client_id=rhsm-api -d
refresh_token=$offline_token | jq -r '.access_token')

# get the filename and download url
image=$(curl -H "Authorization: Bearer $access_token"
"https://api.access.redhat.com/management/v1/images/$checksum/download")
filename=$(echo $image | jq -r .body.filename)
url=$(echo $image | jq -r .body.href)

# download the file
curl $url -o $filename
```

上記のテキストで、**<offline_token>** を Red Hat API ポータルから収集したトークンに置き換え、**<checksum>** を **製品ダウンロード** ページから取得したチェックサム値に置き換えます。

2. このファイルを実行可能な状態にします。

```
$ chmod u+x FILEPATH/FILENAME.sh
```

3. ターミナルウィンドウを開き、bash ファイルを実行します。

```
$ ./FILEPATH/FILENAME.sh
```

警告

ネットワークのベストプラクティスと一貫性のあるパスワード管理を使用します。

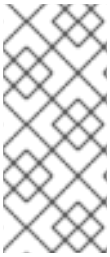
- パスワードや認証情報をプレーンテキストに保存しないでください。
- トークンを不正使用から安全に保護してください。

関連情報

- Getting started with Red Hat APIs

2.7. 起動可能な RHEL 用インストールメディアの作成

本セクションでは、ダウンロードした ISO イメージファイルを使用して、USB、DVD、CD などの起動可能な物理インストールメディアを作成する方法を説明します。ISO イメージのダウンロードの詳細は、インストール用 ISO イメージのダウンロード を参照してください。



注記

デフォルトでは、インストールメデイアで **inst.stage2=** 起動オプションが使用され、特定のラベル (たとえば **inst.stage2=hd:LABEL=RHEL8\x86_64**) に設定されます。ランタイムイメージが含まれるファイルシステムのデフォルトのラベルを変更します。インストールシステムの起動手順をカスタマイズする場合は、このラベルが正しい値に設定されていることを確認します。

2.7.1. インストール起動用メディアオプション

Red Hat Enterprise Linux インストールプログラムを起動する方法はいくつかあります。

フルインストール用 DVD または USB フラッシュドライブ

DVD ISO イメージを使用して、フルインストールの DVD または USB フラッシュドライブを作成します。ソフトウェアパッケージをインストールする場合は、DVD または USB フラッシュドライブを、ブートデバイスおよびインストールソースとして使用できます。

最小インストール用の DVD、CD、または USB フラッシュドライブ

最小インストール用 CD、DVD、または USB フラッシュドライブは、**Boot ISO** イメージを使用して作成されます。これには、システムを起動し、インストールプログラムを開始するのに最低限必要なファイルのみが含まれます。

重要

コンテンツ配信ネットワーク (CDN) を使用して必要なソフトウェアパッケージをダウンロードする場合は、**Boot ISO** イメージに、必要なソフトウェアパッケージを含むインストールソースが必要です。

PXE サーバー

PXE (preboot execution environment) サーバーを使用すると、ネットワーク経由でインストールプログラムを起動できます。システムが起動したら、ローカルのハードドライブやネットワーク経由など、別のインストールソースからインストールを完了します。

Image Builder

Image Builder を使用すると、システムおよびクラウドイメージをカスタマイズして、仮想環境およびクラウド環境に Red Hat Enterprise Linux をインストールできます。

関連情報

- 高度な RHEL 8 インストールの実行
- RHEL システムイメージのカスタマイズ

2.7.2. 起動可能な DVD または CD の作成

起動可能なインストール DVD または CD は、ディスク書き込みソフトウェアや、CD/DVD バーナーを使用して作成できます。ISO イメージファイルから DVD または CD を作成する手順は、オペレーティングシステムや、インストールされているディスク書き込みソフトウェアにより大きく異なります。CD または DVD への ISO イメージファイルの書き込み方法は、お使いの書き込みソフトウェアのドキュメントを参照してください。

警告

DVD ISO イメージ (フルインストール) または Boot ISO イメージ (最小インストール) のいずれかを使用して、起動可能な DVD または CD を作成できます。ただし、DVD ISO イメージが 4.7 GB より大きくなり、1 層または 2 層 DVD に収まらない場合があります。作業を続行する前に、DVD ISO イメージファイルのサイズを確認してください。DVD ISO イメージを使用して起動可能なインストールメディアを作成する場合は、USB フラッシュドライブが推奨されます。

2.7.3. Linux で起動可能な USB デバイスの作成

起動可能な USB デバイスを作成し、それを使用して他のマシンに Red Hat Enterprise Linux をインストールできます。

重要

この手順を実行すると、USB ドライブに保存しておいたデータはすべて警告なしに上書きされます。データをバックアップするか、空のフラッシュドライブを使用してください。起動可能な USB ドライブは、データの保存には使用できません。

前提条件

- インストール用 ISO イメージのダウンロード の説明に従って、インストール用 ISO イメージをダウンロードしている。
- ISO イメージに十分な容量の USB フラッシュドライブがある。必要なサイズはさまざまですが、推奨される USB サイズは 8 GB です。

手順

1. USB フラッシュドライブをシステムに接続します。
2. ターミナルウィンドウを開き、最近のイベントのログを表示します。

```
$ dmesg|tail
```

このログの下部に、接続している USB フラッシュドライブから出力されたメッセージが表示されます。接続したデバイスの名前を記録してください。

3. root ユーザーとしてログインします。

```
$ su -
```

プロンプトに従い root パスワードを入力します。

4. ドライブに割り当てられているデバイスノードを見つけます。この例で使用されているドライブの名前は **sdd** です。

```
# dmesg|tail
[288954.686557] usb 2-1.8: New USB device strings: Mfr=0, Product=1, SerialNumber=2
```

```
[288954.686559] usb 2-1.8: Product: USB Storage
[288954.686562] usb 2-1.8: SerialNumber: 000000009225
[288954.712590] usb-storage 2-1.8:1.0: USB Mass Storage device detected
[288954.712687] scsi host6: usb-storage 2-1.8:1.0
[288954.712809] usbcore: registered new interface driver usb-storage
[288954.716682] usbcore: registered new interface driver uas
[288955.717140] scsi 6:0:0:0: Direct-Access     Generic  STORAGE DEVICE   9228 PQ: 0
ANSI: 0
[288955.717745] sd 6:0:0:0: Attached scsi generic sg4 type 0
[288961.876382] sd 6:0:0:0: sdd Attached SCSI removable disk
```

5. ISO イメージを USB デバイスに直接書き込みます。

```
# dd if=/image_directory/image.iso of=/dev/device
```

- **/image_directory/image.iso** を、ダウンロードした ISO イメージファイルへのフルパスに置き換えます。
- **device** を、**dmesg** コマンドで取得したデバイス名に置き換えます。
  この例では、ISO イメージのフルパスが **/home/testuser/Downloads/rhel-8-x86_64-boot.iso** で、検出されたデバイス名が **sdd** です。

```
# dd if=/home/testuser/Downloads/rhel-8-x86_64-boot.iso of=/dev/sdd
```

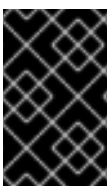
注記

デバイス上のパーティション名ではなく、正しいデバイス名を使用していることを確認してください。パーティション名は、通常、数字の接尾辞が付いたデバイス名です。たとえば、**sdd** がデバイス名の場合、デバイス **sdd** 上のパーティションの名前は、**sdd1** になります。

6. **dd** コマンドがデバイスへのイメージの書き込みを終了するのを待ちます。データ転送が完了すると、**#** プロンプトが表示されます。プロンプトが表示されたら、root アカウントからログアウトして、USB ドライブを取り外します。これで USB ドライブを起動デバイスとして使用する準備が整いました。

2.7.4. Windows で起動可能な USB デバイスの作成

さまざまなツールを使用して、Windows システムに起動可能な USB デバイスを作成できます。Red Hat は、https://github.com/FedoraQt/MediaWriter/releases からダウンロードできる Fedora Media Writer の使用を推奨します。Fedora Media Writer はコミュニティー製品であり、Red Hat のサポート対象外になる点に注意してください。このツールの問題は、https://github.com/FedoraQt/MediaWriter/issues から報告できます。



重要

この手順を実行すると、USB ドライブに保存しておいたデータはすべて警告なしに上書きされます。データをバックアップするか、空のフラッシュドライブを使用してください。起動可能な USB ドライブは、データの保存には使用できません。

前提条件

- インストール用 ISO イメージのダウンロード の説明に従って、インストール用 ISO イメージをダウンロードしている。
- ISO イメージに十分な容量の USB フラッシュドライブがある。必要なサイズはさまざまですが、推奨される USB サイズは 8 GB です。

手順

1. https://github.com/FedoraQt/MediaWriter/releases から Fedora Media Writer をダウンロードしてインストールします。
2. USB フラッシュドライブをシステムに接続します。
3. Fedora Media Writer を開きます。
4. メイン画面で **Custom Image** をクリックして、ダウンロードしておいた Red Hat Enterprise Linux ISO イメージを選択します。
5. **Write Custom Image** 画面で、使用するドライブを選択します。
6. **Write to disk** をクリックします。起動用メディアの作成プロセスが開始します。プロセスが完了するまでドライブを抜かないでください。ISO イメージのサイズや、USB ドライブの書き込み速度により、この操作には数分かかる場合があります。
7. 操作が完了したら、USB ドライブをアンマウントします。これで USB ドライブを起動デバイスとして使用する準備が整いました。

2.7.5. Mac OS X で起動可能な USB デバイスの作成

起動可能な USB デバイスを作成し、それを使用して他のマシンに Red Hat Enterprise Linux をインストールできます。

重要

この手順を実行すると、USB ドライブに保存しておいたデータはすべて警告なしに上書きされます。データをバックアップするか、空のフラッシュドライブを使用してください。起動可能な USB ドライブは、データの保存には使用できません。

前提条件

- インストール用 ISO イメージのダウンロード の説明に従って、インストール用 ISO イメージをダウンロードしている。
- ISO イメージに十分な容量の USB フラッシュドライブがある。必要なサイズはさまざまですが、推奨される USB サイズは 8 GB です。

手順

1. USB フラッシュドライブをシステムに接続します。
2. **diskutil list** コマンドでデバイスパスを特定します。デバイスパスの形式は **/dev/disknumber** です。**number** はディスクの数になります。ディスク番号は、0 から始まります。通常、**disk0** は OS X リカバリーディスク、**disk1** はメインの OS X インストールになります。以下の例では、**disk2** が USB デバイスです。

```
$ diskutil list
/dev/disk0
#:                       TYPE NAME                    SIZE       IDENTIFIER
0:      GUID_partition_scheme                        *500.3 GB   disk0
1:                        EFI EFI                     209.7 MB   disk0s1
2:          Apple_CoreStorage                         400.0 GB   disk0s2
3:                Apple_Boot Recovery HD              650.0 MB   disk0s3
4:          Apple_CoreStorage                         98.8 GB    disk0s4
5:                Apple_Boot Recovery HD              650.0 MB   disk0s5
/dev/disk1
#:                       TYPE NAME                    SIZE       IDENTIFIER
0:                 Apple_HFS YosemiteHD               *399.6 GB   disk1
Logical Volume on disk0s1
8A142795-8036-48DF-9FC5-84506DFBB7B2
Unlocked Encrypted
/dev/disk2
#:                       TYPE NAME                    SIZE       IDENTIFIER
0:     FDisk_partition_scheme                        *8.1 GB     disk2
1:               Windows_NTFS SanDisk USB            8.1 GB     disk2s1
```

3. NAME、TYPE、および SIZE の列をフラッシュドライブと比較し、USB フラッシュドライブを特定します。たとえば、NAME は、**Finder** ツールのフラッシュドライブアイコンのタイトルになります。この値は、フラッシュドライブの情報パネルの値と比較することもできます。

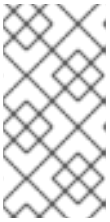
4. フラッシュドライブのファイルシステムボリュームをアンマウントします。

```
$ diskutil unmountDisk /dev/disknumber
    Unmount of all volumes on disknumber was successful
```

コマンドが完了すると、デスクトップからフラッシュドライブのアイコンが消えます。アイコンが消えない場合は、誤ったディスクを選択した可能性があります。誤ってシステムディスクのマウントを解除しようとすると、**failed to unmount** エラーが返されます。

5. ISO イメージをフラッシュドライブに書き込みます。

```
# sudo dd if=/path/to/image.iso of=/dev/rdisknumber
```



注記

Mac OS X では、ブロック (**/dev/disk***) とキャラクターデバイス (**/dev/rdisk***) の両方のファイルが各ストレージデバイスに提供されます。**/dev/rdisknumber** キャラクターデバイスにイメージを書き込む方が、**/dev/disknumber** ブロックデバイスに書き込むよりも高速です。

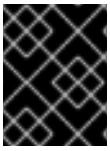
たとえば、**/Users/user_name/Downloads/rhel-8-x86_64-boot.iso** ファイルを **/dev/rdisk2** デバイスに書き込むには、以下のコマンドを実行します。

```
# sudo dd if=/Users/user_name/Downloads/rhel-8-x86_64-boot.iso of=/dev/rdisk2
```

6. **dd** コマンドがデバイスへのイメージの書き込みを終了するのを待ちます。データ転送が完了すると、**#** プロンプトが表示されます。プロンプトが表示されたら、root アカウントからログアウトして、USB ドライブを取り外します。これで USB ドライブを起動デバイスとして使用する準備が整いました。

2.8. インストールソースの準備

Boot ISO イメージファイルには、リポジトリーやソフトウェアパッケージが含まれておらず、システムを起動し、インストールを開始するのに必要なインストールプログラムとツールのみが含まれます。本セクションでは、必要なリポジトリーおよびソフトウェアパッケージを含む DVD ISO イメージを使用して、Boot ISO イメージのインストールソースを作成する方法を説明します。



重要

コンテンツ配信ネットワーク (CDN) から RHEL を登録してインストールしない場合に限り、Boot ISO イメージファイルにインストールソースが必要になります。

2.8.1. インストールソースの種類

最小限のブートイメージには、以下のいずれかのインストールソースを使用できます。

- **DVD:**DVD に DVD ISO イメージを書き込みます。DVD はインストールソース (ソフトウェアパッケージソース) として自動的に使用されます。
- **ハードドライブまたは USB ドライブ:**DVD ISO イメージをドライブにコピーして、ドライブからソフトウェアパッケージをインストールするように、インストールプログラムを設定します。USB ドライブを使用する場合は、インストールを開始する前に、USB ドライブがシステムに接続されていることを確認してください。インストールプログラムは、インストールの開始後にメディアを検出することができません。
  - **ハードドライブの制限**: ハードドライブの DVD ISO イメージは、インストールプログラムがマウントできるファイルシステムを使用しているパーティションに置く必要があります。対応するファイルシステムは、**xfs**、**ext2**、**ext3**、**ext4**、および **vfat (FAT32)** となります。

警告

Microsoft Windows システムで、ハードドライブをフォーマットする際に使用されるデフォルトのファイルシステムは NTFS です。exFAT ファイルシステムも利用できます。ただし、このファイルシステムは、いずれもインストール時に変更することができません。Microsoft Windows のインストールソースとして、ハードドライブまたは USB ドライブを作成する場合は、ドライブを FAT32 としてフォーマットするようにしてください。FAT32 ファイルシステムは、4 GiB を超えるファイルを保存できません。

Red Hat Enterprise Linux 8 では、ローカルのハードドライブのディレクトリーからインストールできます。これを行うには、DVD ISO イメージの内容をハードドライブのディレクトリーにコピーし、ISO イメージの代わりに、そのディレクトリーをインストールソースとして指定します。たとえば、**inst.repo=hd:<device>:<path to the directory>** です。

- **ネットワークの場所:**DVD ISO イメージまたはインストールツリー (DVD ISO イメージから抽出したコンテンツ) をネットワーク上の場所にコピーし、次のプロトコルを使用して、ネットワーク経由でインストールを実行します。
  - **NFS:**DVD ISO イメージは、ネットワークファイルシステム (NFS) 共有にあります。

- **HTTPS、HTTP、または FTP の場合:**インストールツリーは、HTTP、HTTPS、または FTP 経由でアクセス可能なネットワーク上にあります。

2.8.2. インストールソースの指定

インストールソースは、次のいずれかの方法で指定します。

- **ユーザーインターフェイス:**グラフィカルインストールの **インストールソース** 画面で、インストールソースを選択します。詳細は、インストールソースの設定 を参照してください。
- **起動オプション**インストールソースを指定するカスタム起動オプションを設定します。詳細は、ブートオプションの設定 を参照してください。
- **キックスタートファイル:**キックスタートファイルでインストールコマンドを使用して、インストールファイルソースを指定します。詳細は、高度な RHEL 8 インストールの実行 を参照してください。

2.8.3. ネットワークインストール用のポート

次の表は、ネットワークベースの各種インストールにファイルを提供するためにサーバーで開く必要があるポートの一覧です。

表2.3 ネットワークインストール用のポート

使用プロトコル	開くべきポート
HTTP	80
HTTPS	443
FTP	21
NFS	2049、111、20048
TFTP	69

関連情報

- ネットワークのセキュリティー保護

2.8.4. NFS サーバーへのインストールソースの作成

この方法を使用して、物理メディアに接続しなくても、1 つのソースから複数のシステムをインストールできます。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。

- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。
- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。

手順

1. **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

2. DVD ISO イメージを、NFS サーバーのディレクトリーにコピーします。
3. テキストエディターで **/etc/exports** ファイルを開き、以下の構文の行を追加します。

```
/exported_directory/ clients
```

   - **/exported_directory/** を、ISO イメージが含まれるディレクトリーのフルパスに置き換えます。
   - **clients** を次のいずれかに置き換えます。
     - ターゲットシステムのホスト名または IP アドレス
     - すべてのターゲットシステムが ISO イメージへのアクセスに使用できるサブネットワーク
     - NFS サーバーへのネットワークアクセスを持つすべてのシステムが ISO イメージを使用できるようにするためのアスタリスク記号 (*)

   このフィールドの形式に関する詳細は、**exports(5)** の man ページを参照してください。

   たとえば、**/rhel8-install/** ディレクトリーを、すべてのクライアントに対する読み取り専用として使用できるようにする基本設定は次のようになります。

```
/rhel8-install *
```

4. **/etc/exports** ファイルを保存して、テキストエディターを終了します。
5. nfs サービスを起動します。

```
# systemctl start nfs-server.service
```

   /etc/exports ファイルを変更する前に サービスが稼働していた場合は、NFS サーバーの設定をリロードします。

```
# systemctl reload nfs-server.service
```

   ISO イメージは、NFS 経由でアクセス可能になり、インストールソースとして使用できるようになりました。

注記

インストールソースを設定するには、プロトコルに **nfs:** を使用し、サーバーのホスト名または IP アドレス、コロン記号 **(:)**、および ISO イメージを保存しているディレクトリーを指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、ISO イメージを **/rhel8-install/** に保存した場合、指定するインストールソースは **nfs:myserver.example.com:/rhel8-install/** となります。

2.8.5. HTTP または HTTPS を使用するインストールソースの作成

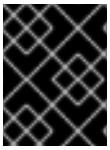
インストールツリー (DVD ISO イメージから抽出したコンテンツと、有効な **.treeinfo** ファイル含むディレクトリー) を使用したネットワークベースのインストール用のインストールソースを作成できます。インストールソースには、HTTP、または HTTPS でアクセスします。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。
- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。
- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。
- **httpd** パッケージがインストールされている。
- **https** インストールソースを使用すると、**mod_ssl** パッケージがインストールされます。

警告

Apache Web サーバー設定で SSL セキュリティーが有効になっている場合は、TLSv1.3 プロトコルを有効にすることが推奨されます。デフォルトでは、TLSv1.2 が有効になっており、TLSv1 (LEGACY) プロトコルを使用できます。



重要

自己署名証明書付きの HTTPS サーバーを使用する場合は、**noverifyssl** オプションを指定してインストールプログラムを起動する必要があります。

手順

1. HTTP(S) サーバーに DVD ISO イメージをコピーします。
2. DVD ISO イメージをマウントするのに適したディレクトリーを作成します。以下はその例です。

```
# mkdir /mnt/rhel8-install/
```

3. DVD ISO イメージをディレクトリーにマウントします。

```
# mount -o loop,ro -t iso9660 /image_directory/image.iso /mnt/rhel8-install/
```

/image_directory/image.iso を DVD ISO イメージへのパスに置き換えます。

4. マウントされたイメージから、HTTP(S) サーバーの root にファイルをコピーします。

```
# cp -r /mnt/rhel8-install/ /var/www/html/
```

このコマンドにより、イメージに含まれるファイルが保存される **/var/www/html/rhel8-install/** ディレクトリーを作成します。他の一部のコピー方法は、有効なインストールソースに必要な **.treeinfo** ファイルを省略する可能性があることに注意してください。この手順で示されているように、ディレクトリー全体に対して **cp** コマンドを入力すると、**.treeinfo** が正しくコピーされます。

5. **httpd** サービスを起動します。

```
# systemctl start httpd.service
```

これにより、インストールツリーにアクセスできるようになり、インストールソースとして使用できるようになります。

注記

インストールソースを設定するには、プロトコルに **http://** または **https://** を使用して、サーバーのホスト名または IP アドレス、および ISO イメージのファイルを保存するディレクトリー (HTTP サーバーの root への相対パス) を指定します。たとえば、HTTP を使用し、サーバーのホスト名が **myserver.example.com** で、イメージのファイルが **/var/www/html/rhel8-install/** にコピーされた場合、指定するインストールソースは **http://myserver.example.com/rhel8-install/** となります。

関連情報

- さまざまな種類のサーバーのデプロイメント

2.8.6. FTP を使用するインストールソースの作成

インストールツリー (DVD ISO イメージから抽出したコンテンツと、有効な **.treeinfo** ファイル含むディレクトリー) を使用したネットワークベースのインストール用のインストールソースを作成できます。インストールソースには、FTP を使用してアクセスします。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。

- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。
- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。
- **vsftpd** パッケージがインストールされている。

手順

1. 必要に応じて、**/etc/vsftpd/vsftpd.conf** 設定ファイルをテキストエディターで開いて編集します。
   a. **anonymous_enable=NO** の行を **anonymous_enable=YES** に変更します。
   b. **write_enable=YES** の行を **write_enable=NO** に変更します。
   c. **pasv_min_port=<min_port>** および **pasv_max_port=<max_port>** の行を追加します。<min_port> と <max_port> を、FTP サーバーがパッシブモードで使用するポート番号の範囲 (**10021** と **10031** など) に置き換えます。
   この手順は、各種のファイアウォール/NAT 設定を採用するネットワーク環境で必要になる可能性があります。
   d. オプション: カスタム変更を設定に追加します。利用可能なオプションは、**vsftpd.conf(5)** の man ページを参照してください。この手順では、デフォルトのオプションが使用されていることを前提としています。

警告

vsftpd.conf ファイルで SSL/TLS セキュリティーを設定している場合は、TLSv1 プロトコルのみを有効にし、SSLv2 と SSLv3 は無効にしてください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は、https://access.redhat.com/solutions/1234773 を参照してください。

2. サーバーのファイアウォールを設定します。
   a. ファイアウォールを有効にします。

```
# systemctl enable firewalld
```

   b. ファイアウォールを起動します。

```
# systemctl start firewalld
```

   c. 前の手順で設定した FTP ポートとポート範囲を許可するようにファイアウォールを設定します。

```
# firewall-cmd --add-port min_port-max_port/tcp --permanent
# firewall-cmd --add-service ftp --permanent
```

<min_port> と <max_port> を **/etc/vsftpd/vsftpd.conf** 設定ファイルに入力したポート番号に置き換えます。

d. ファイアウォールをリロードして、新しいルールを適用します。

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

3. DVD ISO イメージを FTP サーバーにコピーします。

4. DVD ISO イメージをマウントするのに適したディレクトリーを作成します。以下はその例です。

```
# mkdir /mnt/rhel8-install
```

5. DVD ISO イメージをディレクトリーにマウントします。

```
# mount -o loop,ro -t iso9660 /image-directory/image.iso /mnt/rhel8-install
```

/image-directory/image.iso を DVD ISO イメージへのパスに置き換えます。

6. マウントされたイメージから、FTP サーバーのルートにファイルをコピーします。

```
# mkdir /var/ftp/rhel8-install
# cp -r /mnt/rhel8-install/ /var/ftp/
```

このコマンドは、イメージに含まれるファイルが保存される **/var/ftp/rhel8-install/** ディレクトリーを作成します。一部のコピー方法は、有効なインストールソースに必要な **.treeinfo** ファイルを省略できることに注意してください。この手順で示されているように、ディレクトリー全体に対して **cp** コマンドを入力しても、**.treeinfo** が正しくコピーされます。

7. 正しい SELinux コンテキストとアクセスモードが、コピーされたコンテンツに設定されていることを確認します。

```
# restorecon -r /var/ftp/rhel8-install
# find /var/ftp/rhel8-install -type f -exec chmod 444 {} \;
# find /var/ftp/rhel8-install -type d -exec chmod 755 {} \;
```

8. **vsftpd** サービスを開始します。

```
# systemctl start vsftpd.service
```

/etc/vsftpd/vsftpd.conf ファイルを変更する前から、このサービスがすでに実行されていた場合は、サービスを再起動して必ず編集後のファイルを読み込ませてください。

```
# systemctl restart vsftpd.service
```

vsftpd サービスを有効にして、システムの起動プロセス時に開始するようにします。

```
# systemctl enable vsftpd
```

これにより、インストールツリーにアクセスできるようになり、インストールソースとして使用できるようになります。

注記

インストールソースを設定するには、プロトコルに **ftp://** を使用して、サーバーのホスト名または IP アドレス、および ISO イメージのファイルを保存するディレクトリー (FTP サーバーの root への相対パス) を指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、イメージからコピーしたファイルを **/var/ftp/rhel8-install/** に置いた場合、指定するインストールソースは **ftp://myserver.example.com/rhel8-install/** となります。

2.8.7. インストールソースとしてのハードドライブの準備

このモジュールでは、**ext2**、**ext3**、**ext4**、または **XFS** ファイルシステムのハードドライブをインストールソースとして使用して RHEL をインストールする方法を説明します。この方法は、ネットワークアクセスや光学ドライブがないシステムに使用できます。ハードドライブのインストールではインストール DVD の ISO イメージを使用します。ISO イメージは、DVD のコンテンツの完全なコピーが含まれるファイルです。ハードドライブにこのファイルが存在すると、インストールプログラムの起動時に、インストールソースとしてハードドライブを選択できます。

- Windows オペレーティングシステムでハードドライブパーティションのファイルシステムを確認するには、**Disk Management** ツールを使用します。
- Linux オペレーティングシステムでハードドライブパーティションのファイルシステムを確認するには、**parted** ツールを使用します。

注記

LVM(論理ボリューム管理) パーティションでは、ISO ファイルは使用できません。

手順

1. Red Hat Enterprise Linux インストール DVD の ISO イメージをダウンロードします。あるいは、物理メディアに DVD がある場合は、Linux システムで以下のコマンドを使用して ISO のイメージを作成できます。

```
dd if=/dev/dvd of=/path_to_image/name_of_image.iso
```

ここで、dvd は DVD ドライブのデバイス名、**name_of_image** は作成する ISO イメージファイルに指定する名前、**path_to_image** はイメージを格納するシステム上の場所へのパスになります。

2. ISO イメージをシステムのハードドライブまたは USB ドライブにコピーアンドペーストします。

3. **SHA256** チェックサムプログラムを使用して、コピーした ISO イメージが健全であることを確認します。さまざまなオペレーティングシステム用に、多くの SHA256 チェックサムプログラムが利用できます。Linux システムで、以下を実行します。

```
$ sha256sum /path_to_image/name_of_image.iso
```

ここで、**name_of_image** は ISO イメージファイルの名前です。**SHA256** チェックサムプログラムは、**ハッシュ** と呼ばれる 64 文字の文字列を表示します。このハッシュを、Red Hat カスタマーポータルの **ダウンロード** ページにあるこの特定のイメージに表示されるハッシュと比較

します。2 つのハッシュは同一でなければなりません。

4. インストールを開始する前に、カーネルコマンドラインで HDD インストールソースを指定します。

```
inst.repo=hd:<device>:/path_to_image/name_of_image.iso
```

関連情報

- インストールソースの指定
- インストールソースの起動オプション

第3章 スタートガイド

インストールを開始するには、まず起動メニューと利用可能な起動オプションを確認します。次に、選択した内容に応じて、インストールを起動します。

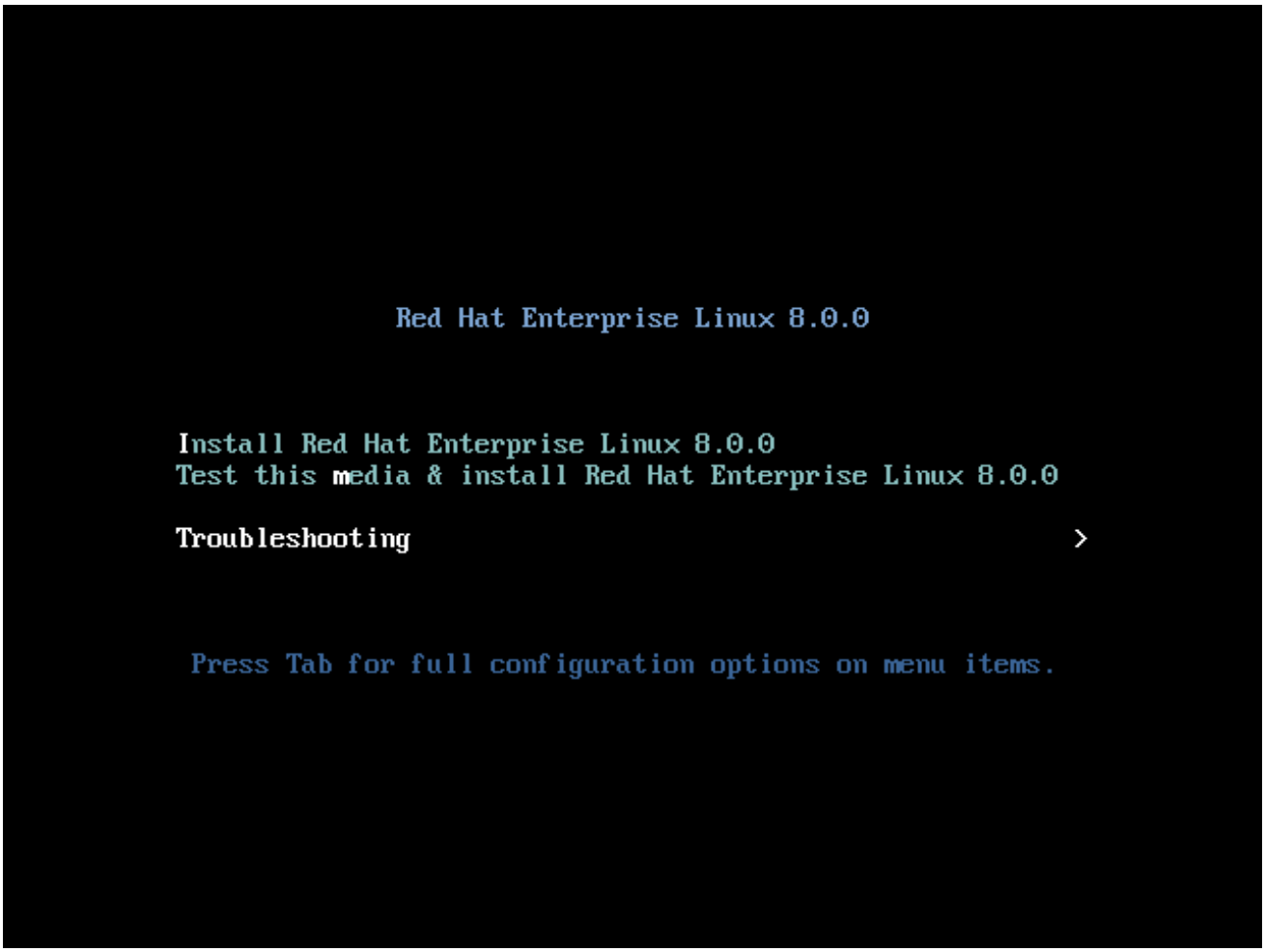
3.1. インストールの起動

起動可能なメディアを作成したら、Red Hat Enterprise Linux インストールを起動する準備ができました。

3.1.1. 起動メニュー

Red Hat Enterprise Linux の起動メニューは、システムが起動メディアの読み込みを完了すると、**GRand Unified Bootloader version 2**(GRUB2) を使用して表示されます。

図3.1 Red Hat Enterprise Linux 起動メニュー



起動メニューには、インストールプログラムを起動する以外に、複数のオプションがあります。60 秒以内に選択しないと、デフォルトの起動オプション (白で強調表示されているもの) が実行します。別のオプションを選択する場合は、キーボードの矢印キーで選択し、**Enter** を押します。

特定のメニューエントリーの起動オプションをカスタマイズできます。

- **BIOS ベースのシステムの場合:Tab** キーを押して、コマンドラインにカスタム起動オプションを追加します。**Esc** キーを押して **boot:** プロンプトにアクセスすることもできますが、必要な起動オプションは事前設定されていません。このシナリオでは、その他の起動オプションを使用する前に、Linux オプションを常に指定する必要があります。

- **UEFI ベースのシステムの場合:e** キーを押して、コマンドラインにカスタム起動オプションを追加します。準備ができたら **Ctrl+X** を押して、修正したオプションを起動します。

表3.1 起動メニューオプション

起動メニューオプション	説明
Red Hat Enterprise Linux 8 のインストール	このオプションは、グラフィカルなインストールプログラムを使用して Red Hat Enterprise Linux をインストールする場合に使用します。詳細は、カスタマーポータルから ISO イメージを使用した RHEL のインストール を参照してください。
このメディアをテストし、Red Hat Enterprise Linux 8 をインストールします。	このオプションは、インストールメディアの整合性を確認する場合に使用します。詳細は、ブートメディアの検証 を参照してください。
Troubleshooting >	このオプションは、インストールに関するさまざまな問題を解決する場合に使用します。**Enter** を押して、そのコンテンツを表示します。

表3.2 トラブルシューティングのオプション

トラブルシューティングのオプション	説明
Troubleshooting > Install Red Hat Enterprise Linux 8 in basic graphics mode	このオプションを使用すると、インストールプログラムがビデオカード用に適切なドライバーを読み込むことができない場合でも、グラフィカルモードで Red Hat Enterprise Linux をインストールします。Install Red Hat Enterprise Linux 8 オプションの使用時に画面が歪んでいる場合は、システムを再起動してこのオプションを使用します。詳細は、グラフィカルインストールにブートできない を参照してください。
Troubleshooting > Rescue a Red Hat Enterprise Linux system	このオプションは、起動を妨げる問題を修復する場合に使用します。詳細は、レスキューモードの使用 を参照してください。
Troubleshooting > Run a memory test	このオプションは、システムでメモリーテストを実行する場合に使用します。**Enter** を押して、そのコンテンツを表示します。詳細は、memtest86 を参照してください。
Troubleshooting > Boot from local drive	このオプションは、最初にインストールしたディスクからシステムを起動する場合に使用します。誤ってこのディスクを起動した場合は、このオプションを使用して、インストールプログラムを起動せずにすぐにハードディスクから起動します。

3.1.2. 起動オプションの入力

起動オプションには、等号 (=) が付いているものと、付けていないものがあります。ブートオプションはブートコマンドラインに追加され、スペースで区切って複数のオプションを追加できます。インストールプログラムに固有の起動オプションは、常に **inst** から始まります。

等号 (=) 記号を使用するオプション

起動オプションに、**=** 記号を使用する値を指定する必要があります。たとえば、**inst.vncpassword=** オプションには値 (この場合はパスワード) を指定する必要があります。この例の正しい構文は **inst.vncpassword=password** です。

等号 (=) 記号を使用しないオプション

この起動オプションでは、値またはパラメーターを使用できません。たとえば、**rd.live.check** オプションでは、インストール開始前にインストールメディアの検証が強制されます。インストールプログラムは、このブートオプションが存在すると検証を実行します。ブートオプションが存在しないと、検証はスキップされます。

3.1.3. BIOS で boot: プロンプトの編集

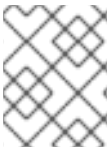
boot: プロンプトを使用すると、最初のオプションは、読み込むインストールプログラムのイメージファイルを常に指定する必要があります。ほとんどの場合、このイメージはキーワードを使用して指定できます。要件に応じて、追加オプションを指定できます。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューが開いたら、キーボードの **Esc** キーを押します。
2. **boot:** プロンプトにアクセスできるようになります。
3. キーボードの **Tab** キーを押して、ヘルプコマンドを表示します。
4. キーボードの **Enter** キーを押して、オプションでインストールを開始します。**boot:** プロンプトから起動メニュー画面に戻るには、システムを再起動して、インストールメディアから再度起動します。



注記

boot: プロンプトでは、**dracut** カーネルオプションも使用できます。利用可能なオプションの一覧は、**dracut.cmdline(7)** の man ページを参照してください。

3.1.4. > プロンプトを使用して事前定義されたブートオプションの編集

BIOS ベースの AMD64 および Intel64 システムでは、**>** プロンプトを使用して、事前定義されたブートオプションを編集できます。オプションの完全なセットを表示するには、ブートメニューから **Test this media and install RHEL 8** を選択します。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューでオプションを選択し、キーボードの **Tab** キーを押します。**>** プロンプトにアクセスし、利用可能なオプションを表示します。
2. **>** プロンプトに必要なオプションを追加します。
3. **Enter** を押してインストールを開始します。
4. **Esc** キーを押して編集をキャンセルし、ブートメニューに戻ります。

3.1.5. UEFI ベースのシステムの GRUB2 メニューの編集

GRUB2 メニューは、UEFI ベースの AMD64、Intel 64、および 64 ビット ARM システムで利用できます。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューウィンドウから必要なオプションを選択し、**e** を押します。
2. UEFI システムでは、カーネルコマンドラインは **linuxefi** で始まります。カーソルを **linuxefi** カーネルコマンドラインの最後に移動します。
3. 必要に応じてパラメーターを編集します。たとえば、1 つ以上のネットワークインターフェイスを設定するには、**linuxefi** カーネルコマンドラインの最後に **ip=** パラメーターを追加し、その後に必要な値を追加します。
4. 編集が終了したら、**Ctrl** + **X** を押して、指定したオプションを使用してインストールを開始します。

3.1.6. USB、CD、または DVD からのインストールの起動

以下の手順に従って、USB、CD、または DVD を使用して Red Hat Enterprise Linux のインストールを起動します。次の手順は一般的なものです。具体的な手順は、ハードウェアの製造元のドキュメントを参照してください。

前提条件

起動可能なインストールメディア (USB、CD、または DVD) を作成している。詳細は、起動可能な DVD または CD の作成 を参照してください。

手順

1. Red Hat Enterprise Linux をインストールするシステムの電源を切ります。
2. システムからドライブを切断します。
3. システムの電源を入れます。
4. 起動可能なインストールメディア (USB、DVD、または CD) を挿入します。

5. システムの電源は切りますが、ブートメディアは取り出さないでください。
6. システムの電源を入れます。

注記

メディアから起動するため特定のキーやキーの組み合わせを押さなければならない場合や、メディアから起動するようにシステムの BIOS (Basic Input/Output System) を設定しなければならない場合があります。詳細は、システムに同梱されているドキュメントをご覧ください。

7. **Red Hat Enterprise Linux ブート** 画面が起動し、さまざまな起動オプションが表示されます。
8. キーボードの矢印キーを使用して起動オプションを選択し、**Enter** を押して、ブートオプションを選択します。**Red Hat Enterprise Linux へようこそ**画面が開き、グラフィカルユーザーインターフェイスを使用して Red Hat Enterprise Linux をインストールできます。

注記

起動画面で、60 秒以内に何も行わないと、インストールプログラムが自動的に開始します。

9. 必要に応じて、利用可能な起動オプションを編集します。
    a. **UEFI ベースのシステム:E** を押して、編集モードにします。事前定義済みのコマンドラインを変更して、起動オプションを追加または削除します。**Enter** キーを押して、選択を確認します。
    b. **BIOS ベースのシステム:**キーボードの **Tab** キーを押して編集モードに入ります。事前定義済みのコマンドラインを変更して、起動オプションを追加または削除します。**Enter** キーを押して、選択を確認します。

関連情報

- グラフィカルインストール
- Boot オプションの参照

3.1.7. PXE を使用してネットワークからインストールを起動

同時に多数のシステムに Red Hat Enterprise Linux をインストールする場合の最善のアプローチは、PXE サーバーから起動し、共有ネットワークにあるソースからインストールすることです。以下の手順に従って、PXE を使用してネットワークから Red Hat Enterprise Linux のインストールを起動します。

注記

PXE を使用してネットワークからインストールプロセスを起動するには、イーサネットなどの物理ネットワーク接続を使用する必要があります。ワイヤレス接続でインストールプロセスを起動することはできません。

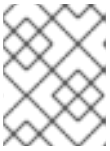
前提条件

- TFTP サーバーを設定しており、PXE に対応するシステムにネットワークインターフェイスがある。詳細は、**関連情報** を参照してください。

- ネットワークインタフェースから起動するように、システムを設定している。このオプションは BIOS にあり、**Network Boot** または **Boot Services** のラベルが付けられる場合があります。
- 指定されたネットワークインターフェイスから BIOS が起動するように設定されており、PXE 標準をサポートしていることを確認した。詳細は、ハードウェアのドキュメントを参照してください。

手順

1. ネットワークケーブルが接続されていることを確認します。コンピューターの電源スイッチが入っていない状態であっても、ネットワークソケットのリンク表示ライトは点灯しているはずです。
2. システムを切り替えます。
   ハードウェアによっては、システムが PXE サーバーに接続する前に、ネットワーク設定と診断情報が表示されることがあります。接続すると、PXE サーバーの設定に応じたメニューが表示されます。
3. 目的のオプションに対応する数字キーを押します。



注記

場合によっては、起動オプションが表示されない場合があります。この場合は、キーボードの **Enter** キーを押します。起動画面が開くまで待ちます。

Red Hat Enterprise Linux ブート 画面が起動し、さまざまな起動オプションが表示されます。

4. キーボードの矢印キーを使用して起動オプションを選択し、**Enter** を押して、ブートオプションを選択します。**Red Hat Enterprise Linux へようこそ**画面が開き、グラフィカルユーザーインターフェイスを使用して Red Hat Enterprise Linux をインストールできます。

注記

起動画面で、60 秒以内に何も行わないと、インストールプログラムが自動的に開始します。

5. 必要に応じて、利用可能な起動オプションを編集します。
   a. **UEFI ベースのシステム:E** を押して、編集モードにします。事前定義済みのコマンドラインを変更して、起動オプションを追加または削除します。**Enter** キーを押して、選択を確認します。
   b. **BIOS ベースのシステム:**キーボードの **Tab** キーを押して編集モードに入ります。事前定義済みのコマンドラインを変更して、起動オプションを追加または削除します。**Enter** キーを押して、選択を確認します。

関連情報

- 高度な RHEL 8 インストールの実行

3.2. カスタマーポータルから ISO イメージを使用した RHEL のインストール

以下の手順に従って、カスタマーポータルからダウンロードした DVD ISO イメージを使用して RHEL をインストールします。この手順では、RHEL インストールプログラムを実行する方法を説明します。

警告

DVD ISO イメージファイルを使用して GUI インストールを実行する場合は、Red Hat 機能への接続機能を使用してシステムを登録するまで、インストーラーの競合状態によりインストールが続行できなくなることがあります。詳細は、**RHEL リリースノート** の既知の問題に記載されている BZ#1823578 を参照してください。

前提条件

- カスタマーポータルから DVD ISO イメージファイルをダウンロードしている。詳細は、ベータインストールイメージのダウンロード を参照してください。
- 起動可能なインストールメディアを作成している。詳細は、起動可能な DVD または CD の作成 を参照してください。
- インストールプログラムを起動し、起動メニューが表示されている。詳細は、インストーラーの起動 を参照してください。

手順

1. 起動メニューで **Install Red Hat Enterprise Linux 8**を選択し、キーボードの **Enter** を押します。
2. **Welcome to Red Hat Enterprise Linux 8**画面で、言語およびロケーションを選択し、**Continue** をクリックします。**Installation Summary** 画面が開き、各設定のデフォルト値が表示されます。
3. **System** > **Installation Destination** を選択し、**Local Standard Disks** ペーンでターゲットのディスクを選択してから **Done** をクリックします。ストレージ設定では、デフォルト設定が選択されます。
4. **システム** > **ネットワークとホスト名** を選択します。**Network and Hostname** 画面が開きます。
5. **Network and Hostname** 画面で、 **Ethernet** を **ON** に切り替えて、**Done** をクリックします。インストーラーは利用可能なネットワークに接続し、そのネットワークで利用可能なデバイスを設定します。必要に応じて、利用可能なネットワークリストから、任意のネットワークを選択して、そのネットワークで利用可能なデバイスを設定できます。
6. **ユーザー設定** > **root パスワード** を選択します。**root パスワード** 画面が開きます。
7. **Root Password** 画面で、root アカウントに設定するパスワードを入力し、**Done** をクリックします。インストールプロセスを完了し、システム管理者ユーザーアカウントにログインするには、root パスワードが必要です。
8. オプション:**User Settings** > **User Creation** を選択して、ユーザーアカウントを作成してインストールプロセスを完了します。root アカウントの代わりに、このユーザーアカウントを使用して、システム管理タスクを実行できます。
9. **Create User** 画面で以下を実行し、**Done** をクリックします。

a. 作成するアカウントの名前とユーザー名を入力します。

b. **Make this user administrator** と **Require a password to use this account**を選択します。インストールプログラムは、このユーザーを wheel グループに追加し、デフォルト設定でパスワード保護されたユーザーアカウントを作成します。パスワードで保護された管理ユーザーアカウントを作成することを推奨します。

10. **Begin Installation** をクリックしてインストールを開始し、インストールが完了するまで待ちます。これには数分かかる場合があります。

11. インストールプロセスが完了したら、**再起動** をクリックして、システムを再起動します。

12. 起動時にインストールメディアが自動的に取り出せない場合は、忘れずに取り出してください。
Red Hat Enterprise Linux 8 は、通常のシステム起動シーケンスが完了すると起動します。X Window System でワークステーションにシステムをインストールしている場合は、システムを設定するアプリケーションが起動します。このアプリケーションを使用すると初期設定が可能になり、システムの時刻と日付の設定、Red Hat へのマシンの登録などが行えます。X Window System がインストールされていない場合は、**login:** プロンプトが表示されます。

注記

UEFI セキュアブートが有効になっているシステムに、Red Hat Enterprise Linux ベータ版リリースをインストールした場合は、システムの Machine Owner Key (MOK) リストにベータ版の公開鍵を追加します。

13. **初期セットアップ** 画面で、ライセンスアグリーメントに同意して、システムを登録します。

関連情報

- **標準的な RHEL 8 インストールの実行**
- インストール起動用メディアオプション

3.3. GUI で CDN から RHEL の登録およびインストール

このセクションでは、GUI を使用して、システムを登録し、RHEL サブスクリプションを割り当て、Red Hat コンテンツ配信ネットワーク (CDN) から RHEL をインストールする方法を説明します。

3.3.1. コンテンツ配信ネットワークとは

cdn.redhat.com で利用できる Red Hat コンテンツ配信ネットワーク (CDN) は、地理的に分散している一連の静的な Web サーバーです。これには、システムが使用するコンテンツとエラータが含まれます。コンテンツは、Red Hat Subscription Management に登録されたシステムを使用するなどして、直接使用できます。CDN は x.509 証明書認証で保護され、有効なユーザーのみがアクセスできるようにします。システムが Red Hat Subscription Management に登録されると、割り当てたサブスクリプションにより、システムがアクセスできる CDN のサブセットが管理されます。

CDN から RHEL を登録してインストールすると、以下の利点があります。

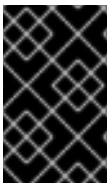
- CDN のインストール方法は、Boot ISO および DVD ISO のイメージファイルに対応します。ただし、大きな DVD ISO イメージファイルよりも領域が少ないため、小さい Boot ISO イメージファイルを使用することが推奨されます。

- CDN は最新のパッケージを使用するため、インストール直後は完全に最新のシステムになります。DVD ISO イメージファイルを使用する場合によくあるように、インストール直後にすぐにパッケージの更新をインストールする必要はありません。
- Red Hat Insights への接続、およびシステムの目的の有効化に対するサポートが統合されました。

CDN から RHEL を登録してインストールする方法は、GUI およびキックスタートで対応しています。GUI を使用して RHEL を登録してインストールする方法は、**標準的な RHEL 8 インストールの実行** を参照してください。キックスタートを使用して RHEL を登録してインストールする方法は、**高度な RHEL 8 インストールの実行** を参照してください。

3.3.2. CDN から RHEL の登録およびインストール

この手順に従って、GUI で、システムを登録し、RHEL サブスクリプションを割り当て、Red Hat コンテンツ配信ネットワーク (CDN) から RHEL をインストールします。



重要

CDN 機能は、**Boot ISO** および **DVD ISO** のイメージファイルでサポートされています。ただし、**Boot ISO** イメージファイルのインストールソースのデフォルトは CDN であるため、Boot ISO イメージファイルを使用することが推奨されます。

前提条件

- CDN にアクセスできるネットワークに接続されている。
- カスタマーポータルから **Boot ISO** イメージファイルをダウンロードしている。
- 起動可能なインストールメディアを作成している。
- インストールプログラムを起動し、起動メニューが表示されている。システム登録後に使用されるインストールリポジトリーは、システムの起動方法により異なる点に注意してください。

手順

1. 起動メニューで **Install Red Hat Enterprise Linux 8**を選択し、キーボードの **Enter** を押します。
2. **Welcome to Red Hat Enterprise Linux 8**画面で、言語およびロケーションを選択し、**Continue** をクリックします。**Installation Summary** 画面が開き、各設定のデフォルト値が表示されます。
3. **System > Installation Destination** を選択し、**Local Standard Disks** ペーンでターゲットのディスクを選択してから **Done** をクリックします。ストレージ設定では、デフォルト設定が選択されます。ストレージ設定のカスタマイズの詳細は、ソフトウェア設定の定義、ストレージデバイス、手動パーティション設定 を参照してください。
4. **システム > ネットワークとホスト名** を選択します。**Network and Hostname** 画面が開きます。
5. **Network and Hostname** 画面で、 **Ethernet** を **ON** に切り替えて、**Done** をクリックします。インストーラーは利用可能なネットワークに接続し、そのネットワークで利用可能なデバイスを設定します。必要に応じて、利用可能なネットワークリストから、任意のネットワークを選択して、そのネットワークで利用可能なデバイスを設定できます。ネットワークまたはネットワークデバイスの設定に関する詳細は、ネットワークホスト名 を参照してください。

6. **Software > Connect to Red Hat**を選択します。**Connect to Red Hat**ウィンドウが開きます。

7. **Connect to Red Hat**ウィンドウで以下の手順を実行します。

    a. **Authentication** の方法を選択し、選択した方法をもとに詳細を指定します。
    アカウント 認証方式の場合: Red Hat カスタマーポータルのユーザー名およびパスワードの詳細を入力します。

    アクティベーションキー 認証方式の場合: 組織 ID およびアクティベーションキーを入力します。サブスクリプションにアクティベーションキーが登録されている限り、複数のアクティベーションキーをコンマで区切って入力できます。

    b. **Set System Purpose** チェックボックスを選択し、該当するドロップダウンリストから必要な **Role**、**SLA**、**Usage** を選択します。
    システムの目的を使用して、Red Hat Enterprise Linux 8 システムの使用目的を記録し、エンタイトルメントサーバーがシステムに最も適したサブスクリプションを自動的に割り当てていることを確認します。

    c. **Red Hat Insights への接続**チェックボックスはデフォルトで有効になっています。Red Hat Insights に接続する必要がない場合には、チェックボックスの選択を解除します。
    Red Hat Insights は SaaS (Software-as-a-Service) 製品で、継続的に、登録済みの Red Hat ベースのシステムに詳細な分析を提供し、物理環境、仮想環境、クラウド環境、およびコンテナーデプロイメントでセキュリティー、パフォーマンス、および安定性に関する脅威をプロアクティブに特定します。

    d. 必要に応じて、**オプション** をデプロイメントし、ネットワーク通信タイプを選択します。

        - ネットワーク環境で、外部のインターネットアクセスのみ、または HTTP プロキシーを介したコンテンツサーバーへのアクセスが許可されている場合は、**HTTP プロキシーの使用** チェックボックスを選択します。

a. **Register** をクリックします。システムが正常に登録され、サブスクリプションが割り当てられると、**Red Hat への接続** ウィンドウに、割り当てられているサブスクリプションの詳細が表示されます。
サブスクリプションのサイズによっては、登録および割り当てのプロセスが完了するのに最大 1 分かかることがあります。

b. **完了** をクリックします。
Red Hat への接続 の下に **登録** メッセージが表示されます。

    1. **ユーザー設定 > root パスワード** を選択します。**root パスワード** 画面が開きます。

    2. **Root Password** 画面で、root アカウントに設定するパスワードを入力し、**Done** をクリックします。インストールプロセスを完了し、システム管理者ユーザーアカウントにログインするには、root パスワードが必要です。
    パスワード作成の要件および推奨事項の詳細は、root パスワードの設定 を参照してください。

    3. オプション:**User Settings > User Creation**を選択して、ユーザーアカウントを作成してインストールプロセスを完了します。root アカウントの代わりに、このユーザーアカウントを使用して、システム管理タスクを実行できます。

    4. **Create User** 画面で以下を実行し、**Done** をクリックします。

c. 作成するアカウントの名前とユーザー名を入力します。

d. **Make this user administrator**と **Require a password to use this account**を選択します。イン

ストールプログラムは、このユーザーを wheel グループに追加し、デフォルト設定でパスワード保護されたユーザーアカウントを作成します。パスワードで保護された管理ユーザーアカウントを作成することを推奨します。
ユーザーアカウントのデフォルト設定を編集する方法は、ユーザーアカウントの作成 を参照してください。

1. **Begin Installation** をクリックしてインストールを開始し、インストールが完了するまで待ちます。これには数分かかる場合があります。
2. インストールプロセスが完了したら、**再起動** をクリックして、システムを再起動します。
3. 起動時にインストールメディアが自動的に取り出せない場合は、忘れずに取り出してください。
   Red Hat Enterprise Linux 8 は、通常のシステム起動シーケンスが完了すると起動します。X Window System でワークステーションにシステムをインストールしている場合は、システムを設定するアプリケーションが起動します。このアプリケーションを使用すると初期設定が可能になり、システムの時刻と日付の設定、Red Hat へのマシンの登録などが行えます。X Window System がインストールされていない場合は、**login:** プロンプトが表示されます。

   注記

   UEFI セキュアブートが有効になっているシステムに、Red Hat Enterprise Linux ベータ版リリースをインストールした場合は、システムの Machine Owner Key (MOK) リストにベータ版の公開鍵を追加します。
4. **初期セットアップ** 画面で、ライセンスアグリーメントに同意して、システムを登録します。

関連情報

- ネットワークのカスタマイズ方法、Red Hat への接続、システムの目的、インストール先、KDUMP、およびセキュリティーポリシーをカスタマイズする方法
- Red Hat Insights product documentation
- Understanding Activation Keys
- Subscription Manager の HTTP プロキシーの設定は、**subscription-manager** man ページの **PROXY CONFIGURATION** セクションを参照してください。

3.3.2.1. システム登録後のインストールソースリポジトリー

システム登録後に使用されるインストールソースリポジトリーは、システムの起動方法により異なります。

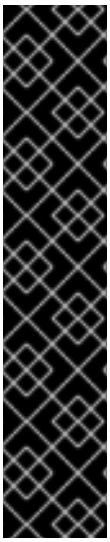
Boot ISO または DVD ISO のイメージファイルから起動するシステム

Boot ISO または **DVD ISO** のいずれかのイメージファイルを使用して、デフォルトの起動パラメーターを使用して RHEL インストールを起動した場合、インストールプログラムは、登録後にインストールソースリポジトリーを CDN に自動的に切り替えます。

inst.repo=<URL> ブートパラメーターで起動したシステム

起動パラメーター **inst.repo=<URL>** を使用して RHEL インストールを起動すると、インストールプログラムは、登録後に自動的にインストールソースリポジトリーを CDN に切り替えません。CDN を使用して RHEL をインストールする場合は、グラフィカルインストールの **インストールソース** 画

面で **Red Hat CDN** オプションを選択し、インストールソースリポジトリーを CDN に手動で切り替える必要があります。CDN に手動で切り替えないと、インストールプログラムは、カーネルコマンドラインで指定されたリポジトリーからパッケージをインストールします。



重要

- キックスタートコマンドの **rhsm** を使用してインストールソースリポジトリーを CDN に切り替えることができるのは、カーネルコマンドラインの **inst.repo=** またはキックスタートファイルの **url** コマンドを使用してインストールソースを指定しない場合に限定されます。インストールイメージを取得するには、カーネルコマンドラインで **inst.stage2=<URL>** を使用する必要がありますが、インストールソースは指定しないでください。
- 起動オプションを使用して指定したインストールソース URL、またはキックスタートファイルに含まれるインストールソース URL は、キックスタートファイルに有効な認証情報を持つ **rhsm** コマンドが含まれている場合でも CDN よりも優先されます。システムが登録されていますが、URL インストールソースからインストールされています。これにより、以前のインストールプロセスが通常通りに動作するようになります。

3.3.3. CDN からシステム登録の確認

以下の手順に従って、GUI で、システムが CDN に登録されていることを確認します。

警告

インストール概要 画面から **インストールの開始** ボタンを **クリックしていない** 場合に限り、CDN から登録を確認できます。**インストールの開始** ボタンをクリックしたら、インストール概要画面に戻って登録を確認することができなくなります。

前提条件

- GUI を使用した CDN からの登録およびインストール に従って登録プロセスを完了し、**インストール概要** 画面の **Red Hat への接続** の下に **登録済** と表示されている。

手順

1. **インストール概要** 画面で、**Red Hat への接続** を選択します。
2. ウィンドウが開き、登録の概要が表示されます。

   方法
   登録済みアカウント名またはアクティベーションキーが表示されます。

   システムの目的
   設定されていると、ロール、SLA、使用方法の詳細が表示されます。

   Insights
   有効にすると、Insights の詳細が表示されます。

   サブスクリプションの数

割り当てたサブスクリプションの数が表示されます。注記:シンプルコンテンツアクセスモードでは、サブスクリプションがリスト表示されないのは有効な動作です。

3. 登録概要が、入力した詳細と一致していることを確認します。

関連情報

- Simple Content Access

3.3.4. CDN からシステムの登録解除

以下の手順に従って、GUI で CDN からシステムの登録を解除します。

警告

- **インストール概要** 画面から **インストールの開始** ボタンを **クリックしていない** 場合は、CDN から登録を解除できます。**インストールの開始** ボタンをクリックしたら、インストール概要画面に戻って登録を解除することができなくなります。
- 登録を解除すると、インストールプログラムは、利用可能な最初のリポジトリーに以下の順序で切り替えます。
  a. カーネルコマンドラインの inst.repo=<URL> 起動パラメーターで使用される URL
  b. インストールメディア (USB または DVD) で自動的に検出されるリポジトリー

前提条件

- CDN から RHEL の登録およびインストール に従って登録プロセスを完了し、**インストール概要** 画面の **Red Hat への接続** の下に **登録済** と表示されている。

手順

1. **インストール概要** 画面で、**Red Hat への接続** を選択します。
2. **Red Hat への接続** 画面が開き、登録の概要が表示されます。

   方法
   使用される登録アカウント名またはアクティベーションキーが表示されます。

   システムの目的
   設定されていると、ロール、SLA、使用方法の詳細が表示されます。

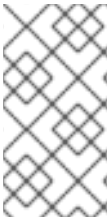
   Insights
   有効にすると、Insights の詳細が表示されます。

   サブスクリプションの数
   割り当てたサブスクリプションの数が表示されます。注記:シンプルコンテンツアクセスモードでは、サブスクリプションがリスト表示されないのは有効な動作です。

3. **登録解除** をクリックして、CDN から登録を削除します。元の登録情報が表示され、画面の中央下部に **未登録** メッセージが表示されます。

4. **完了** をクリックして、**インストール概要** 画面に戻ります。

5. **Red Hat への接続** に **未登録** メッセージが表示され、**ソフトウェアの選択** には **Red Hat CDN では登録が必要です** メッセージが表示されます。



注記

システムの登録を解除したら、システムを再登録できます。**Red Hat への接続** をクリックします。以前入力した詳細が入力されます。元の詳細情報を編集するか、アカウント、目的、および接続に基づいてフィールドを更新します。**登録** をクリックして終了します。

3.4. インストールの完了

インストールが完了するまで待ちます。これには数分の時間がかかる場合があります。

インストールが完了したら、再起動時にインストールメディアが自動的に取り出されない場合は削除します。

Red Hat Enterprise Linux 8 は、通常のシステム起動シーケンスが完了すると起動します。X Window System でワークステーションにシステムをインストールしている場合は、システムを設定するアプリケーションが起動します。このアプリケーションを使用すると初期設定が可能になり、システムの時刻と日付の設定、Red Hat へのマシンの登録などが行えます。X Window System がインストールされていない場合は、**login:** プロンプトが表示されます。

システムの初期セットアップ、登録、および保護を完了する方法については、**標準の RHEL 8 インストールの実行** ドキュメントの インストール後のタスクの完了 セクションを参照してください。

第4章 インストールのカスタマイズ

Red Hat Enterprise Linux をインストールする場合は、**インストール概要** 画面を使用して、場所、ソフトウェア、およびシステム設定およびパラメーターをカスタマイズできます。

インストール概要 画面には、以下のカテゴリーが含まれます。

多言語化

キーボード、言語サポート、および時間と日付を設定できます。

ソフトウェア

Red Hat への接続、インストールソース、およびソフトウェアの選択を設定できます。

システム

インストール先、KDUMP、ネットワークおよびホスト名、セキュリティーポリシーを設定できます。

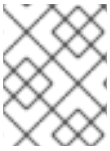
ユーザー設定

システム管理タスクに使用する管理者アカウントにログインし、システムにログインするユーザーアカウントを作成するように root パスワードを設定できます。

カテゴリーは、インストールプログラムのどこにあるかによって、ステータスが異なります。

表4.1 カテゴリーのステータス

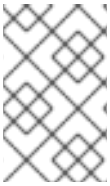
状態	説明
感嘆符と赤いテキストが付いた黄色の三角形	インストールする前に注意が必要です。たとえば、コンテンツ配信ネットワーク (CDN) から登録してダウンロードする前に、ネットワークおよびホスト名を確認する必要があります。
灰色で警告マークが付いたもの (感嘆符付きの黄色の三角形)	インストールプログラムがカテゴリーを設定しているため、カテゴリーが終了しないとその画面にアクセスできません。



注記

インストール概要 画面の下部には警告メッセージが表示され、**インストールの開始** ボタンは、必要なカテゴリーがすべて設定されるまで無効になっています。

このセクションは、グラフィカルユーザーインターフェイス (GUI) を使用した Red Hat Enterprise Linux インストールのカスタマイズを説明します。GUI は、CD、DVD、または USB フラッシュドライブから、もしくは PXE を使用してネットワークからシステムを起動する場合に、Red Hat Enterprise Linux をインストールするのに推奨される方法です。



注記

オンラインヘルプと、カスタマーポータルで公開している内容に矛盾がある可能性もあります。最新の更新は、カスタマーポータルのインストールコンテンツを参照してください。

4.1. 言語およびロケーションの設定

インストールプログラムは、インストール時に選択した言語を使用します。

前提条件

1. インストールメディアを作成している。詳細は、起動可能な DVD または CD の作成 を参照してください。
2. Boot ISO イメージファイルを使用してインストールソースを指定している。詳細は、インストールソースの準備 を参照してください。
3. インストールを起動している。詳細は、インストーラーの起動 を参照してください。

手順

1. **Welcome to Red Hat Enterprise Linux**画面の左側のペインで、言語を選択します。または、**検索** フィールドに、希望の言語を入力します。

注記

言語は、デフォルトで設定されています。ネットワークアクセスが設定されている、つまりローカルメディアではなくネットワークサーバーからシステムを起動した場合、事前選択の言語は、**GeoIP** モジュールの位置自動検出機能により決定します。起動コマンドライン、または PXE サーバー設定で **inst.lang=** オプションを使用した場合は、起動オプションで定義した言語が選択されます。

2. **Red Hat Enterprise Linux へようこそ**画面の右側のペインから、お住まいの地域に合ったロケーションを選択してください。
3. **Continue** をクリックして、グラフィカルインストール 画面に進みます。
4. Red Hat Enterprise Linux のプレリリース版をインストールしようとしている場合は、インストールメディアのプレリリースステータスに関する警告メッセージが表示されます。
   a. インストールを続行するには、**I want to proceed** をクリックします。あるいは、
   b. インストールを終了してシステムを再起動するには、**I want to exit** をクリックします。

関連情報

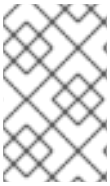
- ローカライゼーション設定の定義

4.2. ローカライゼーションオプションの設定

このセクションでは、キーボード、言語サポート、および日時設定を行う方法を説明します。

重要

ロシア語 のようにラテン文字を受け付けないレイアウトを使用する場合は、一緒に **英語 (US)** レイアウトも追加して、2 つのレイアウトを切り替えられるようにキーボードを設定します。ラテン文字を含まないレイアウトを選択すると、この後のインストールプロセスで有効な **root** パスワードおよびユーザー認証情報を入力できない場合があります。これにより、インストールを完了できない可能性があります。



キーボード、言語、および日時の設定

キーボード、言語、および日時の設定は、デフォルトで Anaconda を使用した RHEL のインストール で行います。設定を変更する場合は次の手順を実行します。変更しない場合は ソフトウェア設定の定義 に進みます。

手順

1. キーボード設定を定義します。
   a. **インストール概要** 画面で **キーボード** をクリックします。デフォルトのレイアウトは、Anaconda を使用した RHEL のインストール で選択したオプションによって異なります。
   b. **+** をクリックして **キーボードレイアウトを追加** 画面を開き、別のレイアウトに変更します。
   c. リストを参照してレイアウトを選択するか、**検索** フィールドを使用します。
   d. 必要なレイアウトを選択して、**追加** をクリックします。デフォルトレイアウトの下に新しいレイアウトが表示されます。
   e. 必要に応じて **オプション** をクリックして、使用可能なレイアウトを切り替えるキーボードスイッチを設定します。**レイアウト切り替えのオプション** 画面が開きます。
   f. 切り替え用のキーの組み合わせを設定するには、1 つ以上のキーの組み合わせを選択し、**OK** をクリックして選択を確定します。

注記

レイアウトを選択して **キーボード** ボタンをクリックすると、選択したレイアウトの視覚的表現を表示する新しいダイアログボックスが開きます。

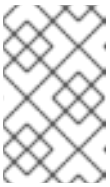
   g. **Done** をクリックして設定を適用し、グラフィカルインストール に戻ります。
2. 言語設定を定義します。
   a. **インストール概要** 画面で **言語サポート** をクリックします。**言語サポート** 画面が開きます。左側のペインには、利用可能な言語グループのリストが表示されます。グループの中から 1 つ以上の言語を設定すると、チェックマークが表示され、対応する言語が強調表示されます。
   b. 左側のペインからグループをクリックして追加の言語を選択し、右側のペインから地域のオプションを選択します。必要なすべての言語に対してこの手順を繰り返します。
   c. **Done** をクリックして変更を適用し、グラフィカルインストール に戻ります。
3. 日時設定を定義します。
   a. **インストール概要** 画面から、**日付と時刻** をクリックします。**日付と時刻** 画面が開きます。

注記

日付と時刻 で選択した設定に基づいて、Anaconda を使用した RHEL のインストール の設定がデフォルトで設定されます。

表示される都市や地域のリストは、タイムゾーンデータベース (**tzdata**) のパブリックドメインのものが使用されています。このドメインは IANA (Internet Assigned Numbers Authority) で管理されています。Red Hat がこのデータベースに都市や地域を追加することはできません。詳細は、IANA 公式の Web サイト をご覧ください。

b. **地域** ドロップダウンメニューから、地域を選択します。



注記

ロケーションを特定の地域に設定せずに、グリニッジ標準時 (GMT) を基準にしたタイムゾーンを設定する場合は、お住まいの地域に **Etc** を選択できます。

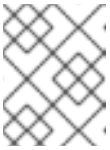
c. **都市** ドロップダウンメニューから都市、もしくは同じタイムゾーン内でお住まいの場所に最も近い都市を選択します。

d. **ネットワーク時刻** スイッチを切り替え、ネットワークタイムプロトコル (NTP) を使用して、ネットワーク時刻同期を有効または無効にします。

注記

ネットワークスイッチを有効にし、システムにインターネットへのアクセスがあれば、システムの時刻が正確に保たれます。デフォルトでは、NTP プールが 1 つ設定されています。新しいオプションを追加するか、**ネットワーク時刻** スイッチの横にある **歯車のボタン** をクリックして、デフォルトのオプションを無効にするか削除します。

e. **Done** をクリックして変更を適用し、グラフィカルインストール に戻ります。



注記

ネットワークの時刻同期を無効にすると、画面下部のコントロールがアクティブになり、手動で時刻と日付を設定できます。

4.3. システムオプションの設定

この接続は、インストール先、KDUMP、ネットワークおよびホスト名、ならびにセキュリティーポリシーを設定する方法を説明します。

4.3.1. インストール先の設定

インストール先 画面では、Red Hat Enterprise Linux のインストール先として使用するディスクなどのストレージオプションを設定します。ディスクは、1 つ以上選択する必要があります。

警告

今後、データが含まれているディスクを使用する予定がある場合は、データをバックアップします。たとえば、既存の Microsoft Windows パーティションを縮小し、Red Hat Enterprise Linux を 2 つ目のシステムとしてインストールする場合、または以前のリリースの Red Hat Enterprise Linux をアップグレードする場合です。パーティションの操作は常にリスクが伴います。たとえば、何らかの理由でプロセスが中断または失敗した場合は、ディスクのデータが失われる可能性があります。

重要

特殊なケース

- BIOS によっては、RAID カードからの起動に対応していないため注意が必要です。このとき、別のハードドライブなど、RAID アレイ以外のパーティションに **/boot** パーティションを作成する必要があります。そのような RAID カードへのパーティション作成には、内蔵ハードドライブを使用する必要があります。また、**/boot** パーティションは、ソフトウェア RAID の設定にも必要です。システムのパーティション設定を自動で選択した場合は、**/boot** パーティションを手動で修正する必要があります。
- Red Hat Enterprise Linux ブートローダーが、別のブートローダーから **チェーンロード** するように設定するには、**インストール先** 画面で **完全なディスク要約とブートローダー** をクリックして、手動でブートドライブを指定する必要があります。
- マルチパスのストレージデバイスと、非マルチパスのストレージデバイスの両方が使用されているシステムに Red Hat Enterprise Linux をインストールすると、インストールプログラムによる自動パーティション設定のレイアウトに、マルチパスのデバイスと非マルチパスのデバイスが混在したボリュームグループが作成されます。これはマルチパスストレージの目的に反することになります。**インストール先** 画面では、マルチパスのみ、または非マルチパスのみのいずれかを選択することが推奨されます。もしくは、手動のパーティション設定を実行してください。

前提条件

インストール概要 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開きます。

    a. **ローカルの標準ディスク** セクションから、必要なストレージデバイスを選択します。選択したストレージデバイスには白いチェックマークが表示されます。白いチェックマークが付いていないディスクはインストール時には使用されません。自動パーティショニングを選択した場合は無視され、手動パーティショニングでは使用できません。

注記

ローカルで利用可能なすべてのストレージデバイス (SATA、IDE、SCSI ハードドライブ、USB フラッシュ、および外部ディスク) は、**ローカルの標準ディスク** に表示されます。インストールプログラムの起動後に接続したストレージデバイスは検出されません。リムーバブルドライブを使用して Red Hat Enterprise Linux をインストールする場合は、デバイスを削除するとシステムが使用できなくなります。

b. オプション:画面右下の **更新** リンクをクリックして、新しいハードドライブに接続するローカルストレージデバイスを設定します。**ディスクの再スキャン** ダイアログボックスが開きます。

注記

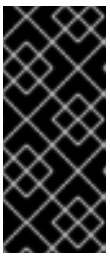
インストール時に行ったストレージへの変更は、**ディスクの再スキャン** をクリックするとすべて失われます。

i. **ディスクの再スキャン** をクリックし、スキャン処理が完了するまで待ちます。

ii. **OK** をクリックして、**インストール先** 画面に戻ります。検出したディスク (新しいディスクを含む) はすべて、**ローカルの標準ディスク** セクションに表示されます。

2. オプション:専用のストレージデバイスを追加するには、**ディスクの追加...** **ストレージデバイスの選択** 画面が開き、インストールプログラムがアクセスするストレージデバイスのリストを表示します。

3. オプション:**ストレージの設定** から **自動** ラジオボタンを選択します。



重要

自動パーティション分割は、ストレージをパーティション分割するのに **推奨される** 方法です。

パーティション設定はカスタマイズできます。詳細は、手動パーティションの設定 を参照してください。

4. オプション:既存のパーティションレイアウトから領域を確保するには、**利用可能な領域を追加する** チェックボックスを選択します。たとえば、使用するディスクにオペレーティングシステムが含まれ、このシステムのパーティションを小さくして、Red Hat Enterprise Linux 用の領域を広くした場合などです。

5. オプション:**データの暗号化** を選択し、**Linux Unified Key Setup** (LUKS) を使用して、(**/boot** などの) システムを起動する必要があるパーティションを除いた、すべてのパーティションを暗号化します。ハードドライブの暗号化が推奨されます。

a. **完了** をクリックします。**ディスク暗号化パスフレーズ** ダイアログボックスが開きます。

i. **パスフレーズ** フィールドおよび **確認** フィールドに、パスフレーズを入力します。

ii. **パスフレーズの保存** をクリックして、ディスクの暗号化を完了します。

警告

LUKS パスフレーズが分からなくなると、暗号化されたパーティションと、その上にあるデータには完全にアクセスできなくなります。分からなくなったパスフレーズを復元する方法はありません。ただし、キックスタートインストールを実行した場合は、インストール中に暗号パスフレーズを保存し、バックアップ用に暗号化パスフレーズを作成できます。詳細は、**高度な RHEL 8 インストールの実行** を参照してください。

6. オプション: 画面左下の **完全なディスク要約とブートローダー** をクリックして、ブートローダーを追加するストレージデバイスを選択します。
詳細は ブートローダーのインストール を参照してください。

注記

大概は、ブートローダーをデフォルトの場所に置いておくだけで十分です。たとえば、他のブートローダーからのチェーンロードを必要とするシステムなど、一部の設定ではブートドライブを手動で指定する必要があります。

7. **完了** をクリックします。

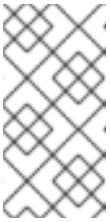
   a. **自動パーティショニング** と **利用可能な領域を追加する** を選択した場合、または、Red Hat Enterprise Linux のインストールに選択したハードドライブの空き領域が十分ではない場合は、**ディスク領域の再利用** ダイアログボックスを開いて **完了** をクリックすると、そのデバイスに設定したディスクデバイスとパーティションのリストが表示されます。ダイアログボックスは、システムで最小インストールに必要な領域に関する情報と、確保した領域のサイズに関する情報が表示されます。

警告

パーティションを **削除** すると、そのパーティションのデータはすべて失われます。データを保存したい場合は、**削除** オプションではなく、**縮小** オプションを使用してください。

   b. 表示された、利用可能なストレージデバイスのリストを確認します。**再利用可能な領域** 列には、各エントリーから再利用できる領域のサイズが表示されます。

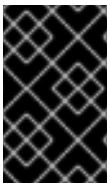
   c. 領域を確保し、ディスクまたはパーティションを選択してから **削除** ボタンをクリックしてそのパーティションを削除するか、選択したディスクにあるすべてのパーティションを削除します。もしくは **縮小** ボタンを押して、既存データを維持しながらパーティションの空き領域を使用します。



注記

または、**すべて削除** をクリックすると、すべてのディスクに存在するすべてのパーティションが削除されるため、Red Hat Enterprise Linux 8 でこの領域を利用できるようになります。すべてのディスクにあるデータはすべて失われます。

d. **Reclaim space** をクリックして変更を適用し、 グラフィカルインストール に戻ります。



重要

インストール概要 画面で **インストールの開始** をクリックするまで、ディスクへの変更は行われません。**再利用** ダイアログボックスは、パーティションをサイズ変更や削除の対象としてマークするだけで、そのアクションはすぐには実行されません。

関連情報

- IBM Z、LinuxONE、および PAES 暗号で dm-crypt を使用する方法

4.3.2. ブートローダーの設定

Red Hat Enterprise Linux は、GRand Unified Bootloader バージョン 2 (**GRUB2**) を、AMD64、Intel 64、IBM Power Systems、および ARM として使用します。64 ビットの IBM Z では、**zipl** ブートローダーが使用されます。

ブートローダーは、システムの起動時に実行し、制御をオペレーティングシステムに読み込み、転送する最初のプログラムです。**GRUB2** は、互換性のあるオペレーティングシステム (Microsoft Windows を含む) であれば起動可能で、チェーンロードを使用すれば、未対応のオペレーティングシステムのブートローダーにも読み込んだ指示を渡すことができます。

警告

GRUB2 をインストールすると、既存のブートローダーを上書きできます。

オペレーティングシステムがすでにインストールされていると、Red Hat Enterprise Linux インストールプログラムはそのブートローダーを自動的に検出して、別のオペレーティングシステムを起動するように設定します。そのブートローダーが正しく検出されない場合は、インストールの完了後に、追加のオペレーティングシステムを手動で設定できます。

複数のディスクを搭載した Red Hat Enterprise Linux システムをインストールする場合は、ブートローダーをインストールするディスクを手動で指定することを推奨します。

手順

1. **インストール先** 画面で **完全なディスク要約とブートローダー** をクリックします。**選択したディスク** ダイアログボックスが開きます。
   ブートローダーは、選択したデバイス、または UEFI システムにインストールされます。ガイド付きパーティションの作成時に、そのデバイスに **EFI システムパーティション** が作成されます。

2. 起動デバイスを変更するには、リストからデバイスを選択して **ブートデバイスとして設定** をクリックします。起動デバイスとして設定できるデバイスは 1 つだけです。

3. 新しいブートローダーのインストールを無効にする場合は、現在起動用として設定されているデバイスを選択し、**ブートローダーをインストールしない** をクリックします。これにより、いずれのデバイスにも **GRUB2** がインストールされないようになります。

警告

ブートローダーをインストールしないを選択した場合は、システムを直接起動できなくなるため、別の起動方法 (市販のスタンドアロンのブートローダーアプリケーションなど) を使用しなければならなくなります。ブートローダーをインストールしないは、システムを起動させる方法が別に確保されている場合に限定してください。

ブートローダーは、システムが BIOS または UEFI のファームウェアを使用しているか、ブートドライブに **GUID Partition Table** (GPT) または **Master Boot Record** (MBR) (**msdos** としても知られている) があるかどうかによって、特別なパーティションを作成する必要があります。自動パーティション作成を使用していると、インストールプログラムがパーティションを作成します。

4.3.3. Kdump の設定

Kdump は、カーネルのクラッシュダンプメカニズムです。システムがクラッシュすると、**Kdump** が、障害発生時のシステムメモリーの内容をキャプチャーします。キャプチャーしたメモリーを解析すると、クラッシュの原因を見つけることができます。**Kdump** が有効になっている場合は、システムメモリー (RAM) のごく一部をそれ自身に予約する必要があります。予約したメモリーは、メインのカーネルにアクセスできません。

手順

1. **インストール概要** 画面から、**Kdump** をクリックします。**Kdump** 画面が開きます。

2. **kdump を有効にする** チェックボックスを選択します。

3. メモリー予約設定を、**自動** または **手動** のいずれかから選択します。

   a. **手動** を選択し、+ ボタンおよび - ボタンを使用して、**予約されるメモリー** フィールドに、予約するメモリー量 (メガバイト) を入力します。予約入力フィールドの下にある **使用可能なシステムメモリー** には、選択したサイズの RAM を予約してから、メインシステムにアクセスできるメモリーの量が示されます。

4. **Done** をクリックして設定を適用し、グラフィカルインストール に戻ります。

注記

予約するメモリーの量は、システムのアーキテクチャー (AMD64 と Intel 64 の要件は IBM Power とは異なります) と、システムメモリーの総量により決まります。ほとんどの場合は、自動予約で十分です。

重要

カーネルクラッシュダンプの保存場所などの追加設定は、インストール後に **system-config-kdump** グラフィカルインターフェイスで設定するか、**/etc/kdump.conf** 設定ファイルに手動で設定できます。

4.3.4. ネットワークおよびホスト名のオプションの設定

ネットワークとホスト名 画面は、ネットワークインターフェイスを設定するために使用されます。ここで選択したオプションは、インストール済みシステムだけでなく、インストール時にリモートからパッケージをダウンロードするなどのタスクを行う際にも利用できます。

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。

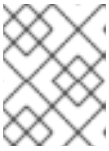
手順

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 Configuring and managing networking を参照してください。

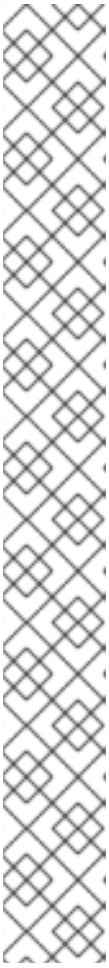
3. 選択したインタフェースを有効または無効にするには、**ON/OFF** スイッチを切り替えます。



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。

4.3.4.1. 仮想ネットワークインターフェイスの追加

この手順では、仮想ネットワークインターフェイスを追加する方法を説明します。

手順

1. **ネットワークとホスト名** 画面で、**+** ボタンをクリックして、仮想ネットワークインターフェイスを追加します。**デバイスの追加** ダイアログが開きます。

2. 使用可能な 4 つのタイプの仮想インターフェイスから 1 つ選択してください。

   - **Bond**: NIC (**ネットワークインターフェイスコントローラー**) のボンドです。複数の物理ネットワークインターフェイスを 1 つのボンドチャネルに結合する方法です。

   - **Bridge**: NIC ブリッジングです。複数のネットワークを 1 つの集積ネットワークに接続します。

   - **Team**: NIC のチーミングです。複数のリンクを集約する新しい実装方法です。小型のカーネルドライバーを提供することでパケットフローを高速で処理し、各種アプリケーションがその他のすべてのタスクをユーザー領域で行うように設計されています。

- **Vlan (仮想 LAN)**: それぞれ独立している複数のブロードキャストドメインを作成する方法です。

3. インターフェイスの種類を選択し、**追加** をクリックします。インターフェイスの編集ダイアログボックスが開き、選択したインターフェイスタイプに使用できる設定を編集できます。
詳細は、ネットワークインタフェース設定の変更 を参照してください。

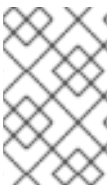
4. **保存** をクリックして仮想インターフェイス設定を確認し、**ネットワークおよびホスト名** 画面に戻ります。

注記

仮想インターフェイスの設定を変更する必要がある場合は、インターフェイスを選択し、**設定** をクリックします。

4.3.4.2. ネットワークインタフェース設定の変更

このセクションは、インストール時に使用される一般的な有線接続に最も重要な設定を説明します。その他の種類のネットワークの設定方法は、一部の設定パラメーターが異なる場合がありますが、ここで説明する内容とあまり変わりません。

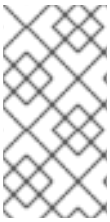


注記

64 ビットの IBM Z では、ネットワークサブチャンネルをあらかじめグループ化してオンラインに設定する必要があるため、新しい接続を追加することはできません。これは現在、起動段階でのみ行われます。

手順

1. 手動でネットワーク接続を設定するには、**ネットワークおよびホスト名** 画面からインターフェイスを選択し、**設定** をクリックします。
選択したインターフェイスに固有の編集ダイアログが開きます。



注記

表示されるオプションは接続の種類によって異なります。使用可能なオプションは、接続の種類が物理インターフェイス (有線または無線のネットワークインターフェイスコントローラー) か、仮想インターフェイスの追加 で設定した仮想インターフェイス (ボンド、ブリッジ、チーム、または Vlan) かによって若干異なります。

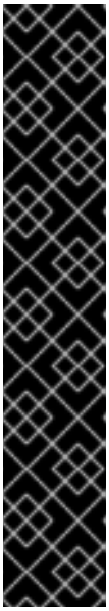
4.3.4.3. インターフェイス接続の有効化または無効化

以下の手順に従って、インターフェイス接続を有効または無効にします。

手順

1. **全般** タブをクリックします。

2. **優先的に自動的に接続** チェックボックスを選択して、デフォルトで接続を有効にします。デフォルトの優先度設定は **0** のままにします。



重要

- 有線接続で有効にすると、システムは起動時または再起動時に自動的に接続されます。無線接続では、インターフェイスにより、範囲内の既知の無線ネットワークへの接続が試されます。**nm-connection-editor** ツールを含む NetworkManager の詳細は、Configuring and managing networking のドキュメントを参照してください。
- **全ユーザーがこのネットワークに接続可能とする** オプションを使用して、このシステムの全ユーザーがこのネットワークに接続するのを有効または無効にできます。このオプションを無効にすると、**root** だけがこのネットワークに接続できます。
- インストール中のこの時点ではその他のユーザーが作成されないため、**root** 以外の特定のユーザーだけがこのインターフェイスを使用するように許可することはできません。別のユーザーが使用する接続が必要な場合は、インストール後に設定する必要があります。

3. **保存** をクリックして変更を適用し、**ネットワークおよびホスト名** 画面に戻ります。

4.3.4.4. 静的な IPv4 または IPv6 の設定

デフォルトでは、現在のネットワーク設定に応じて、IPv4 と IPv6 の両方が自動設定に指定されています。つまり、ローカルの IP アドレス、DNS アドレスなどのアドレスは、インターフェイスがネットワークに接続すると自動的に検出されます。多くの場合はこれで十分ですが、**IPv4 Settings** タブと **IPv6 Settings** タブで静的な設定を行うこともできます。IPv4 設定または IPv6 設定を設定するには、以下の手順を実行します。

手順

1. 静的ネットワーク設定を行うには、IPv 設定タブのいずれかに移動し、**方式** ドロップダウンメニューから、**自動** 以外の方法 (**手動** など) を選択します。**アドレス** ペインが有効になります。

注記

IPv6 設定 タブでは、メソッドを **無視する** に設定して、このインターフェイスの IPv6 を無効にできます。

2. **追加** をクリックして、アドレス設定を入力します。
3. **追加の DNS サーバー** フィールドに IP アドレスを入力します。DNS サーバーの IP アドレス (**10.0.0.1,10.0.0.8** など) を 1 つ以上設定できます。
4. **この接続には IPvX アドレス設定が必要になります** を選択します。

注記

IPv4 または IPv6 が成功した場合にのみこの接続を許可するには、**IPv4 設定** タブまたは **IPv6 設定** タブでこのオプションを選択します。IPv4 および IPv6 の両方でこのオプションを無効にしたままにしておくと、いずれかの IP プロトコル設定に成功した場合にインターフェイスが接続できるようになります。

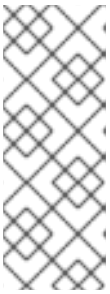
5. **保存** をクリックして変更を適用し、**ネットワークおよびホスト名** 画面に戻ります。

4.3.4.5. ルートの設定

ルートを設定するには、以下の手順を実行します。

手順

1. **IPv4 設定** タブおよび **IPv6 設定** タブで、**ルート** をクリックして特定の IP プロトコルのルーティング設定を行います。そのインターフェイス用のルート編集ダイアログが開きます。
2. **追加** をクリックして、ルートを追加します。
3. 1 つ以上の静的ルートを設定し、設定していないすべてのルートを無効にするには、**自動的に得られたルートを無視する** チェックボックスを選択します。
4. **この接続はネットワーク上のリソースにのみ使用** チェックボックスを選択して、デフォルトルートにはならないようにします。



注記

このオプションは、静的ルートを設定していなくても選択できます。このルートは、ローカルまたは VPN 接続を必要とするイントラネットページなど、特定のリソースにアクセスするためにのみ使用されます。公開されているリソースには別の (デフォルトの) ルートが使用されます。追加ルートが設定されているのとは異なり、この設定はインストール済みシステムに転送されます。このオプションは、複数のインターフェイスを設定する場合に限り役に立ちます。

5. **OK** をクリックして設定を保存し、インターフェイス固有のルートの編集ダイアログボックスに戻ります。
6. **保存** をクリックして設定を適用し、**ネットワークおよびホスト名** 画面に戻ります。

4.3.4.6. 関連情報

- Configuring and managing networking

4.3.5. Red Hat への接続の設定

cdn.redhat.com で利用できる Red Hat コンテンツ配信ネットワーク (CDN) は、地理的に分散している一連の静的な Web サーバーです。これには、システムが使用するコンテンツとエラータが含まれます。コンテンツは、Red Hat Subscription Management に登録されたシステムを使用するなどして、直接使用できます。CDN は x.509 証明書認証で保護され、有効なユーザーのみがアクセスできるようにします。システムが Red Hat Subscription Management に登録されると、割り当てたサブスクリプションにより、システムがアクセスできる CDN のサブセットが管理されます。

CDN から RHEL を登録してインストールすると、以下の利点があります。

- CDN のインストール方法は、Boot ISO および DVD ISO のイメージファイルに対応します。ただし、大きな DVD ISO イメージファイルよりも領域が少ないため、小さい Boot ISO イメージファイルを使用することが推奨されます。
- CDN は最新のパッケージを使用するため、インストール直後は完全に最新のシステムになります。DVD ISO イメージファイルを使用する場合によくあるように、インストール直後にすぐにパッケージの更新をインストールする必要はありません。

- Red Hat Insights への接続、およびシステムの目的の有効化に対するサポートが統合されました。

4.3.5.1. システムの目的の概要

システムの目的は任意ですが、Red Hat Enterprise Linux インストールで推奨される機能です。システムの目的を使用して、Red Hat Enterprise Linux 8 システムの使用目的を記録し、エンタイトルメントサーバーがシステムに最も適したサブスクリプションを自動的に割り当てていることを確認します。

次の利点があります。

- システム管理および事業運営に関する詳細なシステムレベルの情報
- システムを調達した理由とその目的を判断する際のオーバーヘッドを削減
- Subscription Manager の自動割り当てと、システムの使用状況の自動検出および調整のカスタマーエクスペリエンスの向上

以下のいずれかの方法でシステムの目的のデータを入力できます。

- イメージの作成時
- **Connect to Red Hat**画面を使用してシステムを登録し、Red Hat サブスクリプションを割り当てる際の GUI インストール時
- キックスタート自動化スクリプトを使用したキックスタートインストール時
- **subscription-manager syspurpose** コマンドライン (CLI) ツールを使用したインストール後

システムの目的を記録するために、システムの目的の以下のコンポーネントを設定できます。選択された値は、登録時にエンタイトルメントサーバーが、システムに最適なサブスクリプションを割り当てるのに使用されます。

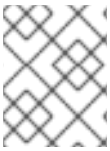
- **ロール**
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Workstation
  - Red Hat Enterprise Linux Compute Node
- **サービスレベルアグリーメント**
  - Premium
  - Standard
  - Self-Support
- **使用率**
  - Production
  - Development/Test
  - Disaster Recovery

関連情報

- RHEL システムイメージのカスタマイズ
- 高度な RHEL 8 インストールの実行
- Red Hat Subscription Manager の使用および設定

4.3.5.2. Red Hat への接続オプションの設定

以下の手順に従って、GUI で Red Hat への接続オプションを設定します。



注記

Red Hat アカウントまたはアクティベーションキーの詳細を使用して CDN に登録できます。

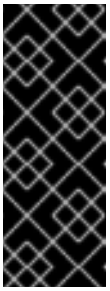
手順

1. **アカウント** をクリックします。
   a. Red Hat カスタマーポータルのユーザー名およびパスワードの詳細を入力します。
2. オプション:**アクティベーションキー** をクリックします。
   a. 組織 ID およびアクティベーションキーを入力します。サブスクリプションにアクティベーションキーが登録されている限り、複数のアクティベーションキーをコンマで区切って入力できます。
3. **システムの目的の設定** チェックボックスを選択します。システムの目的を使用して、エンタイトルメントサーバーが Red Hat Enterprise Linux 8 システムの使用目的を満たすために、最適なサブスクリプションを自動的に判断して割り当てることができます。
   a. ドロップダウンリストから必要な **ロール**、**SLA**、および **使用方法** を選択します。
4. **Red Hat Insights への接続** チェックボックスはデフォルトで有効になっています。Red Hat Insights に接続する必要がない場合には、チェックボックスの選択を解除します。

注記

Red Hat Insights は SaaS (Software-as-a-Service) 製品で、継続的に、登録済みの Red Hat ベースのシステムに詳細な分析を提供し、物理環境、仮想環境、クラウド環境、およびコンテナーデプロイメントでセキュリティー、パフォーマンス、および安定性に関する脅威をプロアクティブに特定します。

5. オプション:**オプション** をデプロイメントします。
   a. ネットワーク環境で、外部のインターネットアクセスまたは HTTP プロキシーを介したコンテンツサーバーへのアクセスのみが許可されている場合は、**HTTP プロキシーの使用** チェックボックスを選択します。HTTP プロキシーを使用していない場合は、**HTTP プロキシーの使用** チェックボックスの選択を解除します。
   b. Satellite Server を実行しているか、内部テストを実行している場合は、**カスタムサーバーの URL** チェックボックスと **カスタムベース URL** チェックボックスを選択して、必要な情報を入力します。



重要

- **カスタムサーバーの URL** フィールドには HTTP プロトコル (**nameofhost.com** など) が必要ありません。ただし、**Custom base URL** フィールドには HTTP プロトコルが必要です。
- 登録後に **カスタムベース URL** を変更するには、登録を解除し、新しい詳細を指定してから再登録する必要があります。

6. **登録** をクリックしてシステムを登録します。システムが正常に登録され、サブスクリプションが割り当てられると、**Red Hat への接続** ウィンドウに、割り当てられているサブスクリプションの詳細が表示されます。

注記

サブスクリプションのサイズによっては、登録および割り当てのプロセスが完了するのに最大 1 分かかることがあります。

7. **完了** をクリックして、**インストール概要** 画面に戻ります。
   a. **Red Hat への接続** の下に **登録** メッセージが表示されます。

4.3.5.3. システム登録後のインストールソースリポジトリー

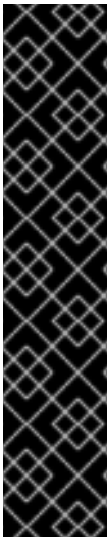
システム登録後に使用されるインストールソースリポジトリーは、システムの起動方法により異なります。

Boot ISO または DVD ISO のイメージファイルから起動するシステム

Boot ISO または **DVD ISO** のいずれかのイメージファイルを使用して、デフォルトの起動パラメーターを使用して RHEL インストールを起動した場合、インストールプログラムは、登録後にインストールソースリポジトリーを CDN に自動的に切り替えます。

inst.repo=<URL> ブートパラメーターで起動したシステム

起動パラメーター **inst.repo=<URL>** を使用して RHEL インストールを起動すると、インストールプログラムは、登録後に自動的にインストールソースリポジトリーを CDN に切り替えません。CDN を使用して RHEL をインストールする場合は、グラフィカルインストールの **インストールソース** 画面で **Red Hat CDN** オプションを選択し、インストールソースリポジトリーを CDN に手動で切り替える必要があります。CDN に手動で切り替えないと、インストールプログラムは、カーネルコマンドラインで指定されたリポジトリーからパッケージをインストールします。



重要

- キックスタートコマンドの **rhsm** を使用してインストールソースリポジトリーを CDN に切り替えることができるのは、カーネルコマンドラインの **inst.repo=** またはキックスタートファイルの **url** コマンドを使用してインストールソースを指定しない場合に限定されます。インストールイメージを取得するには、カーネルコマンドラインで **inst.stage2=<URL>** を使用する必要がありますが、インストールソースは指定しないでください。
- 起動オプションを使用して指定したインストールソース URL、またはキックスタートファイルに含まれるインストールソース URL は、キックスタートファイルに有効な認証情報を持つ **rhsm** コマンドが含まれている場合でも CDN よりも優先されます。システムが登録されていますが、URL インストールソースからインストールされています。これにより、以前のインストールプロセスが通常通りに動作するようになります。

4.3.5.4. CDN からシステム登録の確認

以下の手順に従って、GUI で、システムが CDN に登録されていることを確認します。

警告

インストール概要 画面から **インストールの開始** ボタンを **クリックしていない** 場合に限り、CDN から登録を確認できます。**インストールの開始** ボタンをクリックしたら、インストール概要画面に戻って登録を確認することができなくなります。

前提条件

- GUI を使用した CDN からの登録およびインストール に従って登録プロセスを完了し、**インストール概要** 画面の **Red Hat への接続** の下に **登録済** と表示されている。

手順

1. **インストール概要** 画面で、**Red Hat への接続** を選択します。
2. ウィンドウが開き、登録の概要が表示されます。

   方法
   登録済みアカウント名またはアクティベーションキーが表示されます。

   システムの目的
   設定されていると、ロール、SLA、使用方法の詳細が表示されます。

   Insights
   有効にすると、Insights の詳細が表示されます。

   サブスクリプションの数
   割り当てたサブスクリプションの数が表示されます。注記:シンプルコンテンツアクセスモードでは、サブスクリプションがリスト表示されないのは有効な動作です。
3. 登録概要が、入力した詳細と一致していることを確認します。

関連情報

- Simple Content Access

4.3.5.5. CDN からシステムの登録解除

以下の手順に従って、GUI で CDN からシステムの登録を解除します。

警告

- **インストール概要** 画面から **インストールの開始** ボタンを **クリックしていない** 場合は、CDN から登録を解除できます。**インストールの開始** ボタンをクリックしたら、インストール概要画面に戻って登録を解除することができなくなります。
- 登録を解除すると、インストールプログラムは、利用可能な最初のリポジトリーに以下の順序で切り替えます。
  a. カーネルコマンドラインの inst.repo=<URL> 起動パラメーターで使用される URL
  b. インストールメディア (USB または DVD) で自動的に検出されるリポジトリー

前提条件

- CDN から RHEL の登録およびインストール に従って登録プロセスを完了し、**インストール概要** 画面の **Red Hat への接続** の下に **登録済** と表示されている。

手順

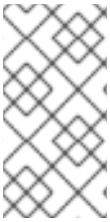
1. **インストール概要** 画面で、**Red Hat への接続** を選択します。
2. **Red Hat への接続** 画面が開き、登録の概要が表示されます。

   方法
   使用される登録アカウント名またはアクティベーションキーが表示されます。

   システムの目的
   設定されていると、ロール、SLA、使用方法の詳細が表示されます。

   Insights
   有効にすると、Insights の詳細が表示されます。

   サブスクリプションの数
   割り当てたサブスクリプションの数が表示されます。注記:シンプルコンテンツアクセスモードでは、サブスクリプションがリスト表示されないのは有効な動作です。
3. **登録解除** をクリックして、CDN から登録を削除します。元の登録情報が表示され、画面の中央下部に **未登録** メッセージが表示されます。
4. **完了** をクリックして、**インストール概要** 画面に戻ります。
5. **Red Hat への接続** に **未登録** メッセージが表示され、**ソフトウェアの選択** には **Red Hat CDN では登録が必要です** メッセージが表示されます。



注記

システムの登録を解除したら、システムを再登録できます。**Red Hat への接続** をクリックします。以前入力した詳細が入力されます。元の詳細情報を編集するか、アカウント、目的、および接続に基づいてフィールドを更新します。**登録** をクリックして終了します。

4.3.5.6. 関連情報

- Red Hat Insights の詳細は、**Red Hat Insights の製品ドキュメント** を参照してください。
- アクティベーションキーの詳細は、**Red Hat サブスクリプション管理の使用** ドキュメントの **アクティベーションキーについて** の章を参照してください。
- Subscription Manager の HTTP プロキシーの設定方法は、**subscription-manager** man ページの **PROXY CONFIGURATION** セクションを参照してください。

4.3.6. セキュリティーポリシーに沿ったシステムのインストール

本セクションは、インストール時に Red Hat Enterprise Linux 8 セキュリティーポリシーを適用する方法と、初めて起動する前にシステムで使用するように設定する方法を説明します。

4.3.6.1. セキュリティーポリシーの概要

Red Hat Enterprise Linux には、特定のセキュリティーポリシーに合わせてシステムの自動設定を有効にする OpenSCAP スイートが同梱されています。このポリシーは、SCAP (Security Content Automation Protocol) 標準を使用して実装されます。パッケージは、AppStream リポジトリーで利用できます。ただし、デフォルトでは、インストールおよびインストール後のプロセスではポリシーが強制されないため、特に設定しない限りチェックは行われません。

インストールプログラムでは、セキュリティーポリシーを適用することは必須ではありません。セキュリティーポリシーを適用する場合は、選択したプロファイルに定義した制限および推奨事項を使用してインストールされます。**openscap-scanner** パッケージおよび **scap-security-guide** パッケージがパッケージ選択に追加され、コンプライアンスおよび脆弱性スキャンのプリインストールツールが利用できるようになります。

セキュリティーポリシーを選択すると、Anaconda GUI インストーラーでは、ポリシーの要件に準拠する設定が必要になります。パッケージの選択が競合したり、別のパーティションが定義されている場合があります。要件がすべて満たされた場合に限り、インストールを開始できます。

インストールプロセスの終了時に、選択した OPEnSCAP セキュリティーポリシーにより、システムが自動的に強化され、スキャンされてコンプライアンスが確認され、インストール済みシステムの **/root/openscap_data** ディレクトリーにスキャン結果が保存されます。

注記

デフォルトでは、インストーラーは、インストールイメージにバンドルされている **scap-security-guide** パッケージの内容を使用します。外部コンテンツは、HTTP サーバー、HTTPS サーバー、または FTP サーバーから読み込むこともできます。

4.3.6.2. セキュリティーポリシーの設定

セキュリティーポリシーを設定するには、以下の手順を実行します。

前提条件

インストール概要 画面が開いている。

手順

1. **インストール概要** 画面から、**セキュリティーポリシー** をクリックします。**セキュリティーポリシー** 画面が開きます。
2. システムでセキュリティーポリシーを有効にするには、**セキュリティーポリシーの適用** を **ON** に切り替えます。
3. 上部ペインに表示されているプロファイルから 1 つ選択します。
4. **プロファイルを選択** をクリックします。
インストール前に適用が必要なプロファイルの変更が、下部ペインに表示されます。
5. カスタムプロファイルを使用するには、**コンテンツの変更** をクリックします。別の画面が開いて、有効なセキュリティーコンテンツの URL を入力できます。
   a. **取得** をクリックして URL を取得します。
   b. **SCAP セキュリティーガイドを使用する** をクリックして、**セキュリティーポリシー** 画面に戻ります。

注記

HTTP サーバー、**HTTPS** サーバー、または **FTP** サーバーから、カスタムプロファイルを読み込むこともできます。コンテンツのフルアドレス (**http://** などのプロトコルを含む) を使用してください。カスタムプロファイルを読み込む前に、ネットワーク接続がアクティブになっている必要があります。インストールプログラムは、コンテンツの種類を自動的に検出します。

6. **完了** をクリックして設定を適用し、**インストール概要** 画面に戻ります。

4.3.6.3. 関連情報

- **scap-security-guide(8)** - **scap-security-guide** プロジェクトの man ページには、SCAP セキュリティープロファイルに関する情報と、OpenSCAP ユーティリティーを使用して提供されているベンチマークの使用例が記載されています。
- Red Hat Enterprise Linux のセキュリティーのコンプライアンス情報は、**セキュリティーの強化** を参照してください。

4.4. ソフトウェア設定の設定

このセクションは、インストールソースおよびソフトウェア選択設定を設定し、リポジトリーをアクティベートする方法を説明します。

4.4.1. インストールソースの設定

以下の手順を完了して、自動検出したインストールメディア、Red Hat CDN、またはネットワークからインストールソースを設定します。

注記

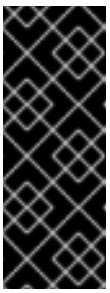
インストール概要 画面を最初に開いた時に、インストールプログラムが、システムの起動に使用されたメディアの種類に基づいて、インストールソースを設定しようとします。完全な Red Hat Enterprise Linux Server DVD は、ソースをローカルメディアとして設定します。

前提条件

- 完全なインストールイメージをダウンロードしている。詳細は、RHEL インストール ISO イメージのダウンロード を参照してください。
- 起動可能な物理メディアを作成している。詳細は、起動可能な CD または DVD の作成 を参照してください。
- **インストール概要** 画面が開いている。

手順

1. **インストール概要** 画面から、**インストールソース** をクリックします。**インストールソース** 画面が開きます。
   a. **自動検出したインストールメディア** セクションを見直して、詳細を確認します。インストールソースを含むメディア (DVD) からインストールプログラムを起動した場合は、このオプションがデフォルトで選択されます。
   b. **検証** をクリックして、メディアの整合性を確認します。
   c. **追加のリポジトリー** セクションを確認してください。デフォルトでは AppStream チェックボックスが選択されています。



重要

- **BaseOS リポジトリーと AppStream リポジトリーはフルインストールイメージでインストールされるため、追加の設定は必要ありません。**
- **Red Hat Enterprise Linux 8 のフルインストールを行う場合は、AppStream リポジトリーのチェックボックスを無効にしないでください。**

2. オプション: **Red Hat CDN**オプションを選択して、システムを登録し、RHEL サブスクリプションを割り当てて、Red Hat コンテンツ配信ネットワーク (CDN) から RHEL をインストールします。詳細は**CDN から RHEL の登録およびインストール**を参照してください。
3. オプション:**ネットワーク上** オプションを選択して、ローカルメディアの代わりに、ネットワーク上からパッケージをダウンロードしてインストールします。

注記

- ネットワーク経由でその他のリポジトリーをダウンロードしてインストールしない場合は ソフトウェア選択の設定 に進みます。
- このオプションは、ネットワーク接続がアクティブな場合にのみ利用できます。GUI でネットワーク接続を設定する方法は ネットワークおよびホスト名のオプションの設定 を参照してください。

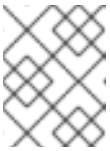
a. **ネットワーク上** ドロップダウンメニューを選択し、パッケージのダウンロードに使用するプロトコルを指定します。この設定は、使用するサーバーによって異なります。

b. アドレスフィールドに、(プロトコルなしで) サーバーアドレスを入力します。NFS を選択すると、入力フィールドが開き、カスタムの **NFS マウントオプション** を指定できます。このフィールドでは、**nfs(5)** の man ページに含まれるオプションを使用できます。

重要

NFS のインストールソースを選択する際には、アドレスを指定する必要があります。ホスト名とパスはコロン (**:**) で区切ります。以下に例を示します。

```
server.example.com:/path/to/directory
```



注記

以下の手順は任意で、ネットワークアクセスにプロキシーが使用されているかどうかのみが必要となります。

c. **プロキシーの設定...** をクリックして、HTTP または HTTPS のソースにプロキシーを設定します。

d. **HTTP プロキシーの有効化** チェックボックスを選択し、**プロキシーホスト** フィールドに URL を入力します。

e. プロキシーサーバーで認証が必要な場合は、**認証を使用する** チェックボックスを選択します。

f. ユーザー名とパスワードを入力します。

g. **OK** をクリックして設定を終了し、**プロキシーの設定...** ダイアログボックスを終了します。

注記

HTTP または HTTPS の URL が、リポジトリーミラーを参照する場合は、**URL type** ドロップダウンリストから必要なオプションを選択します。ソースの設定が終わると、選択に対して環境と追加のソフトウェアパッケージがすべて利用できます。

4. **+** をクリックして、リポジトリーを追加します。

5. **-** をクリックして、リポジトリーを削除します。

6. **矢印** アイコンをクリックして、現在のエントリーを、 **インストールソース** 画面を開いたときに表示されていた設定に戻します。

7. リポジトリーを有効または無効にするには、リストの各エントリーで **有効** 列のチェックボックスをクリックします。

注記

ネットワークにプライマリーリポジトリーを設定するときと同じように、追加リポジトリーに名前を付けて設定できます。

8. **完了** をクリックして設定を適用し、**インストール概要** 画面に戻ります。

4.4.2. ソフトウェア選択の設定

必要なソフトウェアパッケージを選択するには、**ソフトウェアの選択** 画面を使用します。パッケージはベース環境と追加ソフトウェアにより設定されています。

- **ベース環境** には、事前に定義されたパッケージが含まれます。たとえば、Server with GUI (デフォルト)、Server、Minimal Install、Workstation、Custom Operating System、Custom Operating System など、基本環境を 1 つだけ選択できます。可用性は、インストールソースとして使用されているインストール ISO イメージにより異なります。
- **選択した環境の追加ソフトウェア** には、ベース環境用の追加のソフトウェアパッケージが含まれています。複数のソフトウェアパッケージを選択できます。

事前に定義された環境と追加のソフトウェアを使用して、システムをカスタマイズします。ただし、標準的なインストールでは、インストールする個々のパッケージを選択することはできません。特定の環境に含まれるパッケージを表示するには、インストールソースメディア (DVD、CD、USB) にある **repository/repodata/*-comps-repository.architecture.xml** ファイルを参照してください。XML ファイルには、ベース環境としてインストールされたパッケージの詳細が記載されています。利用可能な環境には **<environment>** タグ、そして追加のソフトウェアパッケージには **<group>** タグが付いています。

Red Hat は、インストールするパッケージが分からない場合は、**最小インストール** のベース環境を選択することを推奨します。最小インストールでは、基本バージョンの Red Hat Enterprise Linux と、最低限の追加ソフトウェアがインストールされます。システムのインストールが終了して初めてログインしたら、**YUM パッケージマネージャー** を使用して、必要なソフトウェアをインストールできます。Yum パッケージマネージャーの詳細は、基本的なシステム設定の設定 を参照してください。

注記

- **yum group list** コマンドを実行すると、yum リポジトリーのパッケージグループリストが表示されます。詳細は 基本的なシステム設定の設定 を参照してください。
- インストールするパッケージを制御する必要がある場合は、キックスタートファイルの **%packages** セクションにパッケージを定義します。キックスタートを使用して Red Hat Enterprise Linux をインストールする方法は、高度な RHEL 8 インストールの実行 を参照してください。

前提条件

- インストールソースを設定している。
- インストールプログラムが、パッケージのメタデータをダウンロードしている。
- **インストール概要** 画面が開いている。

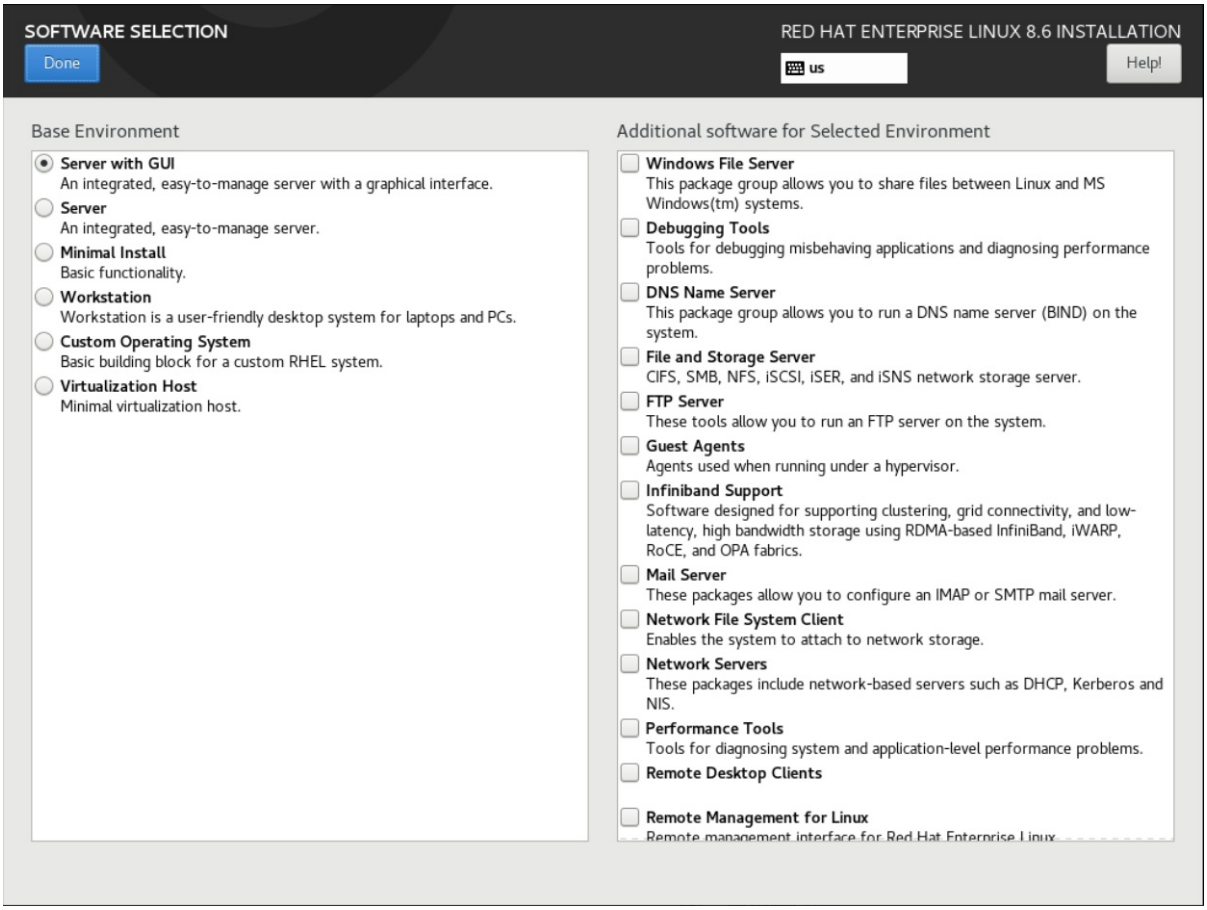
手順

1. **インストール概要** 画面で、**ソフトウェアの選択** をクリックします。**ソフトウェアの選択** 画面が開きます。
2. **ベース環境** ペインで、ベース環境を選択します。たとえば、Server with GUI (デフォルト)、Server、Minimal Install、Workstation、Custom Operating System、Custom Operating System など、基本環境を 1 つだけ選択できます。

注記

サーバー (GUI 使用) ベース環境はデフォルトのベース環境で、インストールを完了してシステムを再起動すると、**初期セットアップ** アプリケーションが起動します。

図4.1 Red Hat Enterprise Linux ソフトウェアの選択



3. **選択した環境の追加ソフトウェア** ペインから、1つ以上のオプションを選択します。
4. **Done** をクリックして設定を適用し、グラフィカルインストール に戻ります。

4.5. ストレージデバイスの設定

さまざまなストレージデバイスに Red Hat Enterprise Linux をインストールできます。**インストール先** 画面で、ローカルでアクセス可能な、基本的なストレージデバイスを設定できます。ハードディスクドライブやソリッドステートドライブなどのローカルシステムに直接接続する基本的なストレージデバイスは、その画面の **ローカルの標準ディスク** セクションに表示されます。64 ビットの IBM Z の場合は、このセクションに、アクティベートした DASD (Direct Access Storage Devices) が含まれます。

警告

既知の問題により、HyperPAV エイリアスとして設定した DASD を、インストールの完了後に自動的にシステムに割り当てることができません。このようなストレージデバイスはインストール時に利用できますが、インストールが完了して再起動しても、すぐにはアクセスできません。HyperPAV エイリアスデバイスを接続するには、システムの **/etc/dasd.conf** 設定ファイルに手動で追加します。

4.5.1. ストレージデバイスの選択

ストレージデバイス選択画面には、インストールプログラムがアクセスできるストレージデバイスがリスト表示されます。システムや利用可能なハードウェアによっては、一部のタブが表示されない場合があります。デバイスは、次のタブに分類されます。

マルチパスデバイス

同じシステムにある、複数の SCSI コントローラーやファイバーチャネルポートなどの複数のパスからアクセスできるストレージデバイスです。

重要

インストールプログラムで検出できるのは、16 文字または 32 文字の長さのシリアル番号を持つマルチパスストレージデバイスのみです。

その他の SAN デバイス

SAN (Storage Area Network) 上にあるデバイスです。

ファームウェア RAID

ファームウェア RAID コントローラーに接続されているストレージデバイスです。

NVDIMM デバイス

特定の状況下では、Red Hat Enterprise Linux 8 は、Intel 64 アーキテクチャーおよび AMD64 アーキテクチャー上で、(NVDIMM) デバイスからセクターモードで起動および実行できます。

System z デバイス

zSeries Linux FCP (ファイバーチャネルプロトコル) ドライバーで接続されたストレージデバイスもしくは LUN (論理ユニット) です。

4.5.2. ストレージデバイスのフィルタリング

ストレージデバイス選択画面では、WWID (World Wide Identifier)、ポート、ターゲット、または論理ユニット番号 (LUN) のいずれかを使用して、ストレージデバイスをフィルタリングできます。

前提条件

インストール概要 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。

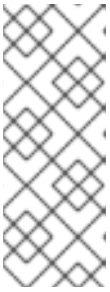
2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加...**をクリックします。ストレージデバイスの選択画面が表示されます。

3. ポート、ターゲット、LUN、または WWID で検索するには、**検索項目** タブをクリックします。WWID または LUN で検索するには、対応する入力テキストフィールドに値を入力する必要があります。

4. **検索** ドロップダウンメニューから、必要なオプションを選択します。

5. **検索** をクリックして検索を開始します。各デバイスと、対応するチェックボックスが、別の行に表示されます。

6. インストールプロセス時に必要なデバイスが利用できるようにするには、チェックボックスを選択します。
後続のインストールプロセスで、選択したデバイスの中から、Red Hat Enterprise Linux をインストールするデバイスを選択できます。その他のデバイスの中から、インストール済みシステムに自動的にマウントするものを選択できます。



注記

- 選択したデバイスがインストールプロセスにより自動的に消去されることはなく、デバイスを選択しても、デバイスに保存されているデータが危険にさらされることはありません。
- インストール後に **/etc/fstab** ファイルを変更することで、システムにデバイスを追加できます。

7. **完了** をクリックして、**インストール先** 画面に戻ります。

重要

ここで選択しないストレージデバイスはすべて、インストールプログラムでは表示されなくなります。別のブートローダーからこのブートローダーをチェーンロードする場合は、ここに表示されているすべてのデバイスを選択します。

4.5.3. 高度なストレージオプションの使用

高度なストレージデバイスを使用するには、iSCSI (SCSI over TCP/IP) ターゲットまたは FCoE (Fibre Channel over Ethernet) の SAN (Storage Area Network) を設定できます。

インストールに iSCSI ストレージデバイスを使用する場合は、インストールプログラム側で iSCSI ストレージデバイスを iSCSI ターゲットとして検出し、そのターゲットにアクセスするための iSCSI セッションを作成できるようにする必要があります。各手順で、CHAP (Challenge Handshake Authentication Protocol) 認証用のユーザー名とパスワードが必要になる場合があります。さらに、検出、またはセッション作成のいずれの場合も、iSCSI ターゲット側でターゲットの接続先となるシステムの iSCSI イニシエーターを認証する (リバース CHAP) ように設定することもできます。CHAP とリバース CHAP を併用する場合は、相互 CHAP または双方向 CHAP と呼ばれます。相互 CHAP を使用すると、特に CHAP 認証とリバース CHAP 認証でユーザー名やパスワードが異なる場合などに、iSCSI 接続に対する最大限の安全レベルを確保できます。

注記

iSCSI 検出と iSCSI ログインの手順を繰り返して、必要な iSCSI ストレージをすべて追加します。初回の検出試行後は、iSCSI イニシエーターの名前を変更できません。iSCSI イニシエーターの名前を変更する場合は、インストールを最初からやり直す必要があります。

4.5.3.1. iSCSI セッションの検出および開始

次の手順を完了して、iSCSI セッションを検出して開始する方法を説明します。

前提条件

- **インストール概要** 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。
2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加..**) クリックします。ストレージデバイスの選択画面が表示されます。
3. **iSCSI ターゲットを追加...**) クリックします。**iSCSI ストレージターゲットの追加** 画面が開きます。

重要

この方法を使用して手動で追加した iSCSI ターゲットには **/boot** パーティションを置くことができません。**/boot** パーティションを含む iSCSI ターゲットを iBFT で使用するように設定する必要があります。ただし、インストールされたシステムが、たとえば iPXE を使用して、ファームウェアの iBFT 以外の方法で提供された iBFT 設定で iSCSI から起動する場合は、**inst.nonibftiscsiboot** インストーラー起動オプションを使用して **/boot** パーティション制限を削除できます。

4. **ターゲットの IP アドレス** フィールドに、iSCSI ターゲットの IP アドレスを入力します。
5. **iSCSI イニシエーター名** フィールドに、iSCSI 修飾名 (IQN) の形式で iSCSI イニシエーターの名前を入力します。IQN エントリーには次を含めてください。
   - **iqn.** の文字列 (ピリオドが必要)。
   - 日付コード (企業や組織のインターネットドメイン名またはサブドメイン名が登録された年と月。記述の順序は年を表す 4 桁の数字、ハイフン、月を表す 2 桁の数字、ピリオドの順で設定されます)。たとえば、2010 年 9 月の場合は **2010-09.** のようになります。
   - 企業や組織のインターネットのドメイン名またはサブドメイン名 (トップレベルのドメインを先頭にして逆順で表します)。たとえば、**storage.example.com** のサブドメインは、**com.example.storage** のようになります。
   - コロン (:) と、ドメインまたはサブドメイン内でその iSCSI イニシエーターを固有に識別する文字列。たとえば、**:diskarrays-sn-a8675309** のようになります。
     完全な IQN は **iqn.2010-09.storage.example.com:diskarrays-sn-a8675309** のようになります。インストールプログラムでは、IQN を設定しやすいように、この形式による任意の名前がすでに **iSCSI Initiator Name** フィールドに自動入力されています。IQN の詳細は、

tools.ietf.org の **RFC 3720 - Internet Small Computer Systems Interface (iSCSI)**に記載されている **3.2.6. iSCSI Names** と、tools.ietf.org の **RFC 3721 - Internet Small Computer Systems Interface (iSCSI) Naming and Discovery** に記載されている **1. iSCSI Names and Addresses** を参照してください。

6. **認証のタイプの探索** ドロップダウンメニューを使用して、iSCSI 検出に使用する認証タイプを指定します。以下のタイプが使用できます。
   - 証明書なし
   - CHAP 秘密鍵
   - CHAP 秘密鍵と逆順鍵
7. a. 認証タイプに **CHAP ペア** を選択した場合は、**CHAP ユーザー名** と **CHAP パスワード** の各フィールドに、iSCSI ターゲットのユーザー名とパスワードを入力します。

   b. 認証タイプに **CHAP 秘密鍵と逆順鍵** を選択した場合は、**CHAP ユーザー名** と **CHAP パスワード** の各フィールドに、iSCSI ターゲットのユーザー名とパスワードを入力します。また、**リバース CHAP ユーザー名** と **CHAP パスワード** の各フィールドに、iSCSI イニシエーターのユーザー名とパスワードを入力します。
8. 必要に応じて、**ターゲットをネットワークインターフェイスへバインドする** チェックボックスをオンにします。
9. **探索を開始** をクリックします。
   入力した情報に基づいて、インストールプログラムが iSCSI ターゲットを調べます。検出に成功すると、**iSCSI ターゲットを追加** 画面には、ターゲットで検出された iSCSI ノードのリストが表示されます。
10. インストールに使用するノードのチェックボックスを選択します。

注記

ノードのログイン認証のタイプ メニューには、**認証のタイプの探索** メニューと同じオプションがあります。ただし、ディスカバリー認証に証明書が必要な場合は、見つかったノードに同じ証明書を使用してログインします。

11. **探索に証明書を使用** ドロップダウンメニューをクリックします。適切な認証情報を指定すると、**ログイン** ボタンが利用可能になります。
12. **ログイン** をクリックして、iSCSI セッションを開始します。

4.5.3.2. FCoE パラメーターの設定

FCoE パラメーターを設定するには、次の手順を実行します。

前提条件

インストール概要 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。

2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加...**をクリックします。ストレージデバイスの選択画面が表示されます。

3. **FCoE SAN を追加...**) クリックします。FCoE ストレージデバイスを検出するようにネットワークインターフェイスを設定するダイアログボックスが開きます。

4. **NIC** ドロップダウンメニューで、FCoE スイッチに接続するネットワークインターフェイスを選択します。

5. **FCoE ディスクの追加** をクリックして、SAN デバイスのネットワークをスキャンします。

6. 必要なチェックボックスを選択します。

   - **Use DCB: Data Center Bridging** (DCB) は、ストレージネットワークやクラスターでイーサネット接続の効率性を向上させる目的で設計されたイーサネットプロトコルに対する拡張セットです。このチェックボックスを選択して、インストールプログラムによる DCB 認識を有効または無効にします。このオプションは、ネットワークインターフェイスでホストベースの DCBX クライアントを必要とする場合にのみ有効にします。ハードウェアの DCBX クライアントを使用するインターフェイスで設定する場合は、このチェックボックスを無効にします。

   - **Use auto vlan: 自動 VLAN** はデフォルトで有効になり、VLAN 検出を行うかどうかを指定します。このチェックボックスを選択すると、リンク設定が検証された後、イーサネットインターフェイスで FIP (FCoE Initiation Protocol) VLAN 検出プロトコルが実行します。設定が行われていない場合は、検出されたすべての FCoE VLAN に対してネットワークインターフェイスが自動的に作成され、VLAN インターフェイスに FCoE のインスタンスが作成されます。

7. 検出された FCoE デバイスが、**インストール先** 画面の **他の SAN デバイス** タブに表示されます。

4.5.3.3. DASD ストレージデバイスの設定

DASD ストレージデバイスを設定するには、以下の手順を実行してください。

前提条件

インストール概要 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。

2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加...**をクリックします。ストレージデバイスの選択画面が表示されます。

3. **DASD の追加** をクリックします。**DASD ストレージターゲットの追加** ダイアログボックスが開いて、0.0.0204 などのデバイス番号を指定し、インストールの開始時に検出されなかった DASD を登録するように求められます。

4. **デバイス番号** フィールドに、接続する DASD のデバイス番号を入力します。

5. **探索を開始** をクリックします。

注記

- 指定したデバイス番号を持つ DASD が検出され、その DASD が接続されていない場合は、ダイアログボックスが閉じ、新たに検出されたドライブが、ドライブのリストに表示されます。次に、必要なデバイスのチェックボックスを選択して、**完了** をクリックします。**インストール先** 画面の **ローカルの標準ディスク** セクションで、新しい DASD が選択できるようになります (**DASD device 0.0.xxxx** と表示されます)。
- 無効なデバイス番号を入力した場合、または指定したデバイス番号の DASD がすでにシステムに割り当てられている場合は、ダイアログボックスにエラーメッセージとその理由が表示され、別のデバイス番号で再試行するように求められます。

4.5.3.4. FCP デバイスの設定

FCP デバイスは、64 ビットの IBM Z が DASD デバイスの代わりに、または DASD デバイスに加えて、SCSI デバイスを使用できるようにするものです。FCP デバイスは交換ファブリックスイッチを提供し、これにより 64 ビットの IBM Z システムが SCSI LUN を従来の DASD デバイスとして用いる使い方に加えて、ディスクデバイスとして使えるようにします。

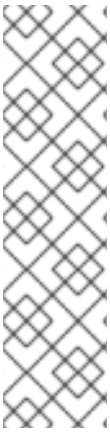
前提条件

- **インストール概要** 画面が開いている。
- FCP のみのインストールで、DASD がないことを示すために、CMS 設定ファイルから **DASD=** オプションを削除するか、パラメーターファイルから **rd.dasd=** オプションを削除した。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。
2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加...**をクリックします。ストレージデバイスの選択画面が表示されます。
3. **zFCP LUN を追加** をクリックします。**zFCP ターゲットの追加** ダイアログボックスが開いて、FCP (ファイバーチャネルプロトコル) ストレージデバイスを追加できます。
   64 ビットの IBM Z では、インストールプログラムが FCP LUN をアクティベートするために、FCP デバイスを手動で入力する必要があります。これは、グラフィカルインストールで指定するか、パラメーターもしくは CMS 設定ファイル内で一意のパラメーターエントリーとして指定することで可能になります。設定する各サイトに固有の値を入力する必要があります。
4. 4 桁の 16 進数のデバイス番号を、**デバイス番号** フィールドに入力します。
5. RHEL-8.6 以前のリリースをインストールする場合、**zFCP** デバイスが NPIV モードで設定されていない場合や、**zfcp.allow_lun_scan=0** カーネルモジュールパラメーターで **auto LUN** スキャンが無効になっている場合は、以下の値を指定します。
   a. 16 桁の 16 進数の WWPN (World Wide Port Number) を、**WWPN** フィールドに入力します。
   b. 16 桁の 16 進数の FCP LUN 識別子を、**LUN** フィールドに入力します。
6. **探索を開始** をクリックして、FCP デバイスに接続します。

新たに追加されたデバイスは、**インストール先** 画面の **System z デバイス** のタブに表示されます。



注記

- FCP デバイスの対話形式の作成は、グラフィカルモードでのみ可能です。テキストモードのインストールでは、FCP デバイスを対話形式で設定することはできません。
- 16 進法で小文字のみを使用してください。間違った値を入力して **探索を開始** をクリックすると、インストールプログラムにより警告が表示されます。設定情報の編集と、探索の再試行が可能です。
- 値の詳細は、ハードウェアに添付のドキュメントを参照し、システム管理者に確認してください。

4.5.4. NVDIMM デバイスへのインストール

不揮発性デュアルインラインメモリーモジュール (NVDIMM) デバイスは、電源が供給されていない時に、RAM のパフォーマンスと、ディスクのようなデータの持続性を兼ね備えています。特定の状況下では、NVDIMM デバイスから Red Hat Enterprise Linux 8 を起動して実行できます。

4.5.4.1. NVDIMM デバイスをインストール先として使用するための基準

Red Hat Enterprise Linux 8 は、**nd_pmem** ドライバーがサポートする Intel 64 アーキテクチャーおよび AMD64 アーキテクチャーにある、セクターモードの不揮発性デュアルインラインメモリーモジュール (NVDIMM) デバイスにインストールできます。

NVDIMM デバイスをストレージとして使用するための条件

NVDIMM デバイスをストレージとして使用するには、次の条件を満たす必要があります。

- システムのアーキテクチャーが Intel 64 または AMD64 である。
- NVDIMM デバイスがセクターモードに設定されている。インストールプログラムにより NVDIMM デバイスをこのモードに再設定できます。
- NVDIMM デバイスが、**nd_pmem** ドライバーで対応している。

NVDIMM デバイスからの起動の条件

以下の条件が満たされる場合には、NVDIMM デバイスからの起動が可能です。

- NVDIMM デバイスを使用するための条件がすべて満たされている。
- システムが UEFI を使用している。
- システムで使用可能なファームウェアまたは UEFI ドライバーが NVDIMM デバイスをサポートしている。UEFI ドライバーは、デバイス自体のオプション ROM から読み込むことができます。
- NVDIMM デバイスが名前空間で利用可能である。

システムの起動中に高性能な NVDIMM デバイスを利用するには、**/boot** ディレクトリーおよび **/boot/efi** ディレクトリーをデバイスに置きます。NVDIMM デバイスの XIP (Execute-in-place) 機能は、起動時にはサポートされません。カーネルは従来どおりメモリーに読み込まれます。

4.5.4.2. グラフィカルインストールモードを使用した NVDIMM デバイスの設定

不揮発性デュアルインラインメモリーモジュール (NVDIMM) デバイスは、Red Hat Enterprise Linux 8 で使用するために、グラフィカルインストールを使用して正しく設定する必要があります。

警告

NVDIMM デバイスを再設定するプロセスにより、デバイスに格納されていたデータがすべて失われます。

前提条件

- NVDIMM デバイスがシステムに存在し、その他の、インストールターゲットとして使用するための条件を満たしている。
- インストールが起動し、**インストール概要** 画面が開いている。

手順

1. **インストール概要** 画面から、**インストール先** をクリックします。**インストール先** 画面が開き、利用可能なドライブのリストが表示されます。
2. **特殊なディスクおよびネットワークディスク** セクションで **ディスクの追加..**) クリックします。ストレージデバイスの選択画面が表示されます。
3. **NVDIMM デバイス** タブをクリックします。
4. デバイスを再設定する場合は、リストから選択します。
   デバイスがリストにない場合は、セクターモードになっていません。
5. **NVDIMM の再設定...**) クリックします。再設定ダイアログが開きます。
6. 必要なセクターサイズを入力し、**再設定の開始** をクリックします。
   サポートされるセクターサイズは 512 バイトおよび 4096 バイトです。
7. 再設定が終了したら、**OK** をクリックします。
8. デバイスのチェックボックスを選択します。
9. **完了** をクリックして、**インストール先** 画面に戻ります。
   再設定した NVDIMM は、**特殊なディスクおよびネットワークディスク** セクションに表示されます。
10. **完了** をクリックして、**インストール概要** 画面に戻ります。

NVDIMM デバイスがインストール先として選択できるようになります。デバイスが起動の要件を満たしている場合は、そのように設定できます。

4.6. 手動パーティションの設定

手動パーティション設定を使用して、ディスクパーティションおよびマウントポイントを設定し、Red Hat Enterprise Linux がインストールされているファイルシステムを定義できます。

注記

インストールの前に、ディスクデバイスにパーティションを設定するかどうかを検討する必要があります。直接または LVM を使用して、LUN でパーティショニングを使用することの利点と欠点の詳細については、https://access.redhat.com/solutions/163853 の記事を参照してください。

Red Hat Enterprise Linux のインストールで最低限必要なパーティションは 1 つですが、Red Hat は、少なくとも /、**/home**、**/boot**、**swap** のパーティションまたはボリュームを使用することを推奨します。必要に応じて、その他のパーティションやボリュームを作成することもできます。

警告

データを失わないように、先に進める前に、データのバックアップを作成しておくことが推奨されます。デュアルブートシステムをアップグレードまたは作成する場合は、保存しておくストレージデバイスの全データのバックアップを作成してください。

4.6.1. 手動パーティションの設定

前提条件

- **Installation Summary** 画面が開いている。
- インストールプログラムで、すべてのディスクが利用可能である。

手順

1. インストールに使用するディスクを選択します。
   a. **インストール先** をクリックして、**インストール先** 画面を開きます。
   b. 対応するアイコンをクリックして、インストールに必要なディスクを選択します。選択したディスクにはチェックマークが表示されています。
   c. **ストレージの設定** で、**カスタム** ラジオボタンを選択します。
   d. オプション:LUKS によるストレージの暗号化を有効にする場合は、**データを暗号化する** チェックボックスを選択します。
   e. **完了** をクリックします。
2. ストレージの暗号化を選択した場合は、ディスク暗号化パスフレーズを入力するダイアログボックスが開きます。LUKS パスフレーズを入力します。
   a. 2 つのテキストフィールドにパスフレーズを入力してください。キーボードレイアウトを切り替えるには、キーボードアイコンを使用します。

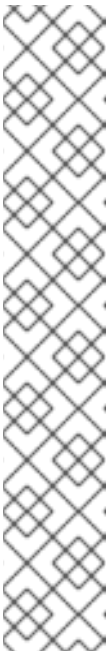
警告

パスフレーズを入力するダイアログボックスでは、キーボードレイアウトを変更できません。インストールプログラムでパスフレーズを入力するには、英語のキーボードレイアウトを選択します。

b. **パスフレーズの保存** をクリックします。**手動パーティション設定** 画面が開きます。

3. 削除したマウントポイントが、左側のペインにリスト表示されます。マウントポイントは、検出されたオペレーティングシステムのインストールごとにまとめられています。したがって、複数のインストールでパーティションを共有していると、ファイルシステムによっては複数回表示されることがあります。

a. 左側のペインでマウントポイントを選択します。カスタマイズ可能なオプションが右側のペインに表示されます。



注記

- システムに既存のファイルシステムがある場合には、インストールに十分な領域があることを確認してください。パーティションを削除するには、リストから選択して、**-** ボタンをクリックします。
ダイアログには、削除されたパーティションが属するシステムが使用しているその他のパーティションをすべて削除するチェックボックスがあります。

- 既存のパーティションがなく、出発点として推奨されるパーティションセットを作成する場合は、左側のペインから、使用するパーティションスキーム (Red Hat Enterprise Linux のデフォルトは LVM) を選択し、**ここをクリックすると自動的に作成します** リンクをクリックします。
利用可能なストレージのサイズに比例して、**/boot** パーティション、/ (root) ボリューム、および **swap** ボリュームが作成され、左側のペインに表示されます。これは、一般的なインストールに推奨されるファイルシステムですが、ファイルシステムやマウントポイントを追加することもできます。

b. **完了** をクリックして変更を適用し、**インストール概要** 画面に戻ります。

4.6.2. マウントポイントのファイルシステム追加

マウントポイントのファイルシステムは複数追加できます。

前提条件

- パーティションの計画が完了している。

重要

領域の割り当てに関する問題を回避するには、**/boot** などの既知の固定サイズの小型パーティションを作成してから残りのパーティションを作成して、インストールプログラムが残りの領域をそのパーティションに割り当てられるようにします。複数のディスクにシステムをインストールする場合、またはこれらのディスクのサイズが異なり、BIOS に検出される最初のディスクに特定のパーティションを作成する必要がある場合は、そのパーティションを最初に作成するようにしてください。

手順

1. **+** をクリックして、マウントポイントのファイルシステムを作成します。**マウントポイントを追加します** ダイアログが表示されます。
2. **マウントポイント** ドロップダウンメニューから、事前に設定したパスの中から 1 つ選択するか、別のパスを入力します。たとえば、root パーティションの場合は / を選択し、ブートパーティションの場合は **/boot** を選択します。
3. ファイルシステムのサイズを **要求される容量** フィールドに入力します。たとえば **2GiB** です。

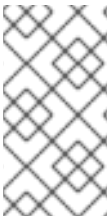
警告

要求される容量フィールドを空のままにするか、利用可能な領域より大きい値を指定すると、残りの空き容量がすべて使用されます。

4. **マウントポイントの追加** をクリックしてパーティションを作成し、**手動パーティション設定** 画面に戻ります。

4.6.3. マウントポイントのファイルシステム用ストレージの設定

この手順は、手動で作成した各マウントポイントにパーティショニング設定を設定する方法を説明します。利用可能なオプションは、**Standard Partition**、**LVM**、および **LVM Thin Provisioning** です。



注記

- Red Hat Enterprise Linux 8 では、Btrf のサポートが非推奨になりました。
- **/boot** パーティションは、選択した値に関係なく、常に標準パーティションに置かれます。

手順

1. 非 LVM マウントポイントを 1 つ配置するデバイスを変更するには、左側のペインから必要なマウントポイントを選択します。
2. **デバイス** の下にある **修正...**) クリックします。**マウントポイントの設定** ダイアログが開きます。

3. 1つ以上のデバイスを選択し、**選択** をクリックして選択を確認し、**手動パーティション設定** 画面に戻ります。

4. **設定を更新** をクリックして、変更を適用します。

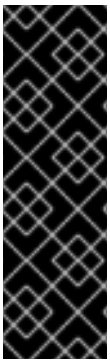
5. **手動パーティション設定** 画面左下で **ストレージデバイスが選択されています** リンクをクリックして、**選択したディスク** ダイアログを開いて、ディスク情報を確認します。

注記

ローカルディスクとパーティションをすべてリフレッシュするには、**再スキャン** ボタン (円形の矢印ボタン) をクリックします。この作業が必要になるのは、インストールプログラム以外で高度なパーティション設定を行った場合のみです。**ディスクの再スキャン** ボタンをクリックすると、インストールプログラムに行った設定変更がすべてリセットされます。

4.6.4. マウントポイントのファイルシステムのカスタマイズ

特定の設定を行う場合は、パーティションまたはボリュームをカスタマイズできます。



重要

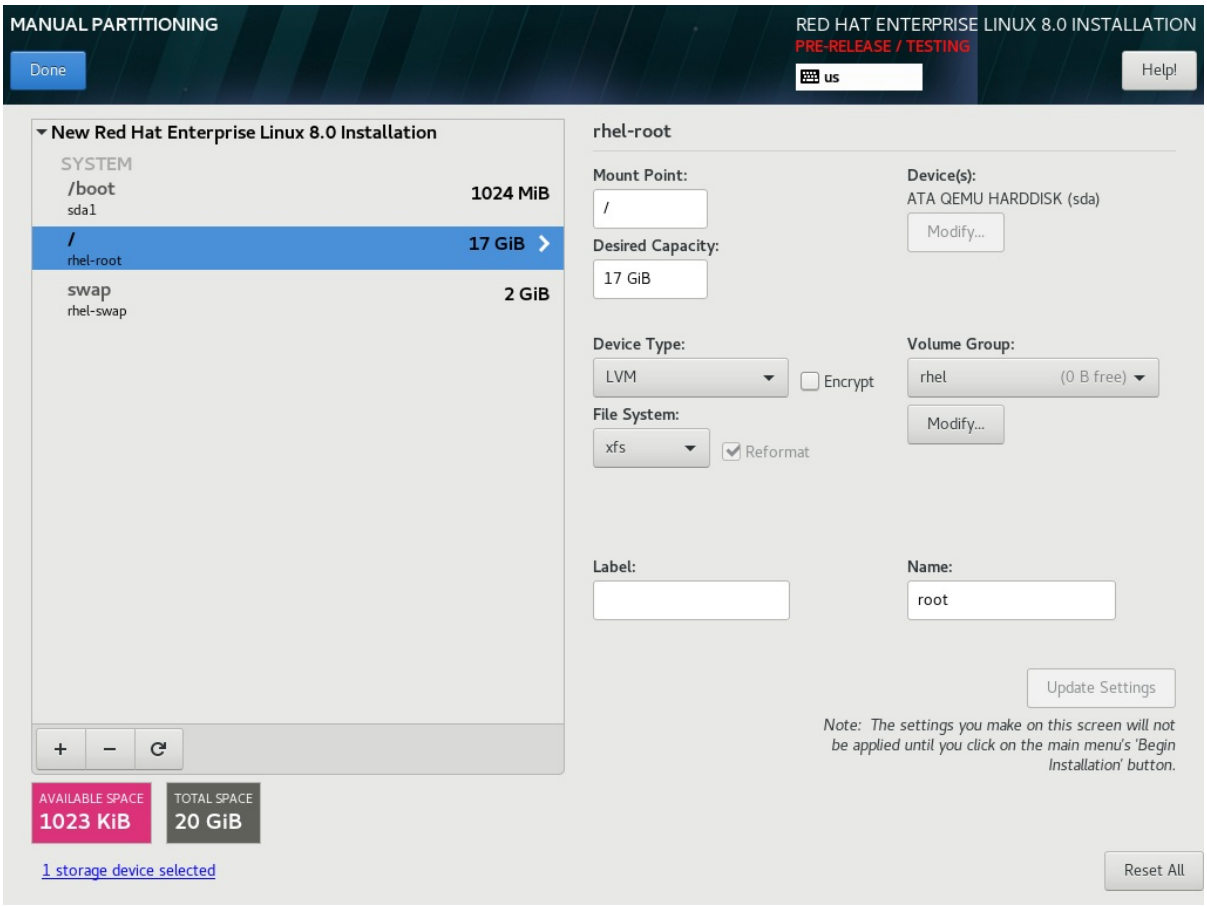
/usr または **/var** には重要なコンポーネントが含まれているため、このディレクトリーのパーティションをルートボリュームとは別の場所に設定すると、起動プロセスが非常に複雑になります。iSCSI ドライブや FCoE などの場所に配置してしまった場合には、システムが起動できなくなったり、電源オフや再起動の際に **Device is busy** のエラーでハングしたりする可能性があります。

これらの制限は **/usr** と **/var** にのみ適用され、その下のディレクトリーには適用されません。たとえば、**/var/www** 向けの個別パーティションは問題なく機能します。

手順

1. 左側のペインから、マウントポイントを選択します。

図4.2 パーティションのカスタマイズ



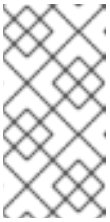
2. 右側のペインで、次のオプションをカスタマイズできます。

   a. **マウントポイント** フィールドに、ファイルシステムのマウントポイントを入力します。たとえば、ファイルシステムが root ファイルシステムの場合は / を入力します。**/boot** ファイルシステムの場合は **/boot** を入力します。swap ファイルシステムの場合は、ファイルシステムタイプを **swap** に設定すれば十分であるため、マウントポイントを設定しないでください。

   b. **割り当てる容量** フィールドに、ファイルシステムのサイズを入力します。単位には KiB や GiB が使用できます。単位を指定しない場合は、MiB がデフォルトになります。

   c. ドロップダウンの **デバイスタイプ** から、必要なデバイスタイプ **標準パーティション**、**LVM**、または**LVM シンプロビジョニング** を選択します。

> **警告**
>
> インストールプログラムは、オーバープロビジョニングの LVM シンプールをサポートしていません。



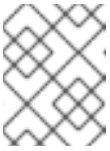
注記

RAID は、パーティションの作成に 2 つ以上のディスクが選択されている場合にのみ使用できます。**RAID** を選択した場合は、**RAID レベル** も設定できます。同様に、**LVM** を選択した場合は、**ボリュームグループ** を選択できます。

d. パーティションまたはボリュームを暗号化する場合は、**暗号化** チェックボックスを選択します。後続のインストールプログラムで、パスワードを設定する必要があります。**LUKS バージョン** ドロップダウンメニューが表示されます。

e. ドロップダウンメニューから、LUKS バージョンを選択します。

f. **ファイルシステム** ドロップダウンメニューから、このパーティションまたはボリュームに適したファイルシステムタイプを選択します。



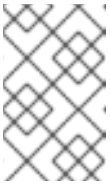
注記

Linux システムパーティションでは、**VFAT** ファイルシステムのサポートは利用できません。たとえば、/、**/var**、**/usr** などです。

g. 既存のパーティションをフォーマットする場合は **再フォーマット** チェックボックスを選択します。データを保持するには、**再フォーマット** チェックボックスの選択を解除します。新たに作成したパーティションとボリュームは再フォーマットする必要があるため、チェックボックスの選択を解除することはできません。

h. **ラベル** フィールドのパーティションにラベルを割り当てます。ラベルを使用すると、個別のパーティションの認識とアドレス指定が容易になります。

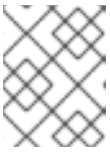
i. **名前** フィールドに名前を入力します。



注記

標準パーティションの場合は作成時に自動的に名前が付けられるため、名前の変更はできません。たとえば、**/boot** の名前 **sda1** を編集することはできません。

3. **設定を更新** をクリックして変更を適用し、必要に応じてカスタマイズする別のパーティションを選択します。**インストール概要** 画面で **インストールの開始** をクリックするまで、変更は適用されません。



注記

パーティションの変更を破棄して、最初からやり直すには、**すべてリセット** をクリックします。

4. ファイルシステムとマウントポイントをすべて作成してカスタマイズしたら、**完了** をクリックします。ファイルシステムの暗号化を選択すると、パスフレーズを作成するように求められます。
変更の概要 ダイアログボックスが開き、インストールプログラムの全ストレージアクションの概要が表示されます。

5. **変更を許可する** をクリックして変更を適用し、**インストール概要** 画面に戻ります。

4.6.5. /home ディレクトリーの維持

Red Hat Enterprise Linux 8 グラフィカルインストールでは、RHEL 7 システムで使用されていた **/home** ディレクトリーを保存できます。

警告

RHEL 7 システムの別の **/home** パーティションに、**/home** ディレクトリーが存在する場合に限り、**/home** を予約できます。

さまざまな設定を含む **/home** ディレクトリーを保持すると、新しい Red Hat Enterprise Linux 8 システムで新しい RHEL 7 システムでの GNOME Shell 環境を、RHEL 8 システムと同じように設定できるようになります。これは、以前の RHEL 7 システムと同様、同じユーザー名と ID を持つ Red Hat Enterprise Linux 8 のユーザーに対してのみ適用されることに注意してください。

この手順は、RHEL 7 システムから **/home** ディレクトリーを保存します。

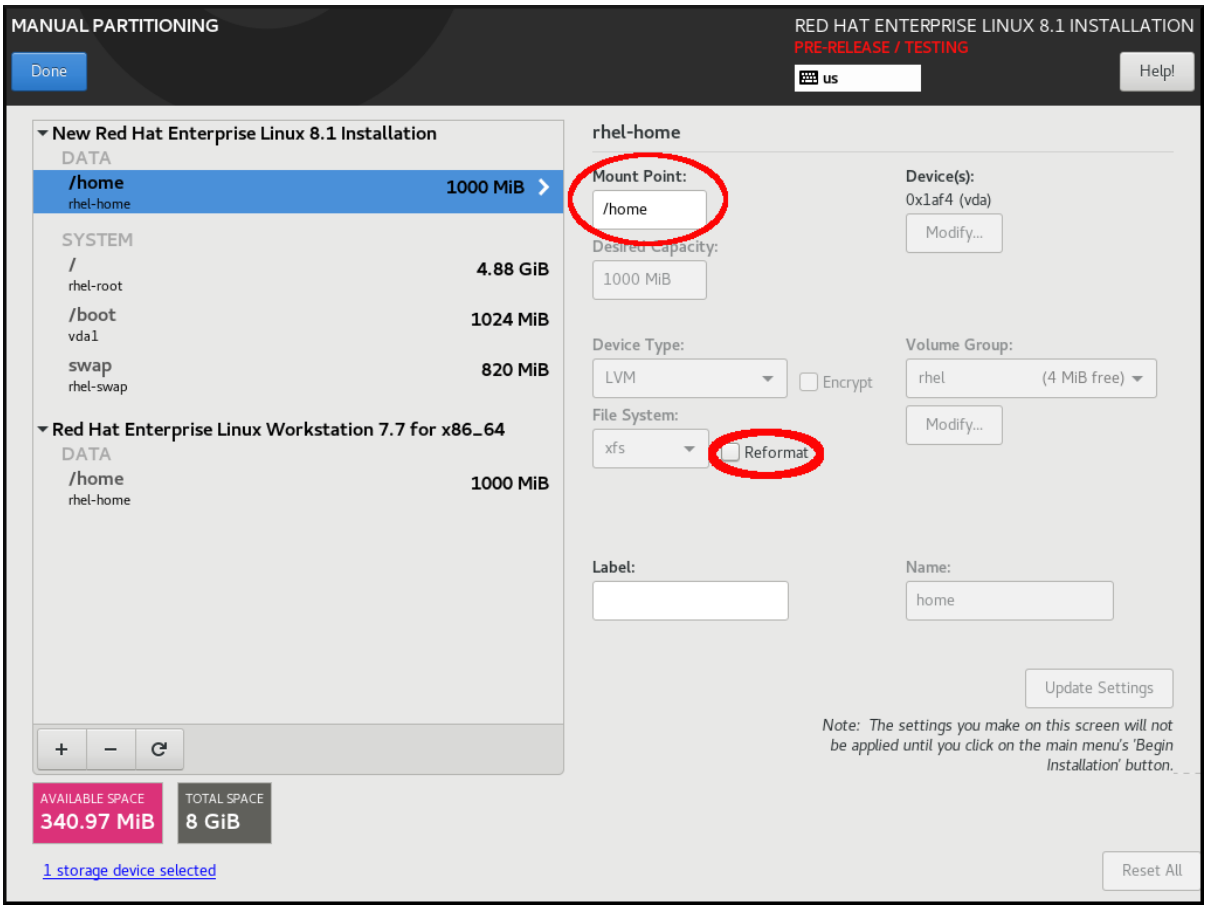
前提条件

- コンピューターに RHEL 7 がインストールされている。
- **/home** ディレクトリーが RHEL 7 システムの別の **/home** パーティションにある。
- Red Hat Enterprise Linux 8 の **Installation Summary** ウィンドウが開いている。

手順

1. **インストール先** をクリックして、**インストール先** 画面を開きます。
2. **ストレージの設定** で、**カスタム** ラジオボタンを選択します。**完了** をクリックします。
3. **終了** をクリックすると、**手動パーティション設定** 画面が開きます。
4. **/home** パーティションを選択し、**Mount Point:** 下に **/home** を入力し、**Reformat** チェックボックスの選択を解除します。

図4.3 /home がフォーマットされていないことを確認



5. オプション: マウントポイントのファイルシステムのカスタマイズの説明に従って、Red Hat Enterprise Linux 8 システムに必要な **/home** パーティションのさまざまなアスペクトをカスタマイズすることができます。ただし、RHEL 7 システムから **/home** を保持するには、**Reformat** チェックボックスの選択を解除する必要があります。

6. 要件に従ってすべてのパーティションをカスタマイズしたら、**完了** をクリックします。**変更の概要** ダイアログボックスが開きます。

7. **変更の概要** ダイアログボックスに **/home** の変更 が表示されていないことを確認します。つまり、**/home** パーティションは保持されます。

8. **変更を許可する** をクリックして変更を適用し、**インストール概要** 画面に戻ります。

4.6.6. インストール中のソフトウェア RAID の作成

Redundant Arrays of Independent Disks (RAID) デバイスは、パフォーマンスを向上させ、一部の設定ではより優れたフォールトトレランスを提供するように配置された複数のストレージデバイスから構築されます。

RAID デバイスの作成は 1 つのステップで終わり、必要に応じてディスクを追加または削除できます。システムでは、1 つの物理ディスクに 1 つの RAID パーティションが作成できるため、インストールプログラムで使用できるディスク数により、利用できる RAID デバイスのレベルが決定します。たとえば、システムにハードドライブが 2 つある場合は、RAID 10 デバイスを作成することはできません。少なくともディスクが 3 つ必要になるためです。

注記

64 ビットの IBM Z では、ストレージサブシステムが RAID を透過的に使用します。ソフトウェア RAID を手動で設定する必要はありません。

前提条件

- RAID 設定オプションは、インストール用に複数のディスクを選択している場合にのみ表示される。作成する RAID タイプに応じて、少なくとも 2 つのディスクが必要です。
- マウントポイントを作成している。マウントポイントを設定して、RAID デバイスを設定します。
- **インストール先** 画面で **カスタム** ラジオボタンを選択している。

手順

1. **手動パーティション設定** 画面の左側のペインで、必要なパーティションを選択します。
2. **デバイス** セクションの下にある **修正** をクリックします。**マウントポイントの設定** ダイアログボックスが開きます。
3. RAID デバイスに追加するディスクを選択して、**選択** をクリックします。
4. **デバイスタイプ** ドロップダウンメニューをクリックして、**RAID** を選択します。
5. **ファイルシステム** のドロップダウンメニューをクリックして、目的のファイルシステムタイプを選択します。
6. **RAID レベル** ドロップダウンメニューをクリックして、目的の RAID レベルを選択します。
7. **設定を更新** をクリックして、変更を保存します。
8. **完了** をクリックして設定を適用し、**インストールの概要** ウィンドウに戻ります。

関連情報

- DM 整合性での RAID LV の作成

4.6.7. LVM 論理ボリュームの作成

論理ボリューム管理 (LVM) では、ハードドライブや LUN などの基本的な物理ストレージ領域を、論理的な観点から表示します。物理ストレージ上のパーティションは物理ボリュームとして表示され、ボリュームグループにグループ化できます。各ボリュームグループは複数の論理ボリュームに分割できます。各論理ボリュームは標準のディスクパーティションによく似ています。したがって、LVM 論理ボリュームは、複数の物理ディスクにまたがることが可能なパーティションとして機能します。

注記

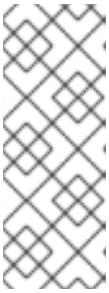
LVM 設定は、グラフィカルインストールプログラムでのみ利用できます。

重要

テキストモードによるインストールの場合は、LVM を設定できません。LVM 設定を作成するには、**Ctrl**+**Alt**+**F2** を押し、別の仮想コンソールのシェルプロンプトを使用します。このシェルで **vgcreate** および **lvm** コマンドを実行できます。テキストモードのインストールに戻るには **Ctrl**+**Alt**+**F1** を押します。

手順

1. **手動パーティション設定** 画面の左側のペインから、マウントポイントを選択します。
2. **デバイスタイプ** ドロップダウンメニューをクリックして、**LVM** を選択します。**ボリュームグループ** ドロップダウンメニューが表示され、新たに作成したボリュームグループ名が表示されます。



注記

設定ダイアログではボリュームグループの物理エクステントのサイズは指定できません。このサイズは、常にデフォルト値の 4 MiB に設定されます。別の物理エクステントのボリュームグループを作成する場合は、対話シェルに切り替えて、**vgcreate** コマンドで手動で作成するか、キックスタートファイルで **volgroup --pesize=size** コマンドを使用して作成します。詳細は **高度な RHEL 8 インストールの実行** を参照してください。

関連情報

- 論理ボリュームの設定および管理

4.6.8. LVM 論理ボリュームの設定

以下の手順に従って、新たに作成された LVM 論理ボリュームを設定します。

警告

/boot パーティションを LVM ボリュームに配置することには対応していません。

手順

1. **手動パーティション設定** 画面の左側のペインから、マウントポイントを選択します。
2. **デバイスタイプ** ドロップダウンメニューをクリックして、**LVM** を選択します。**ボリュームグループ** ドロップダウンメニューが表示され、新たに作成したボリュームグループ名が表示されます。
3. **修正** をクリックして、新たに作成したボリュームグループを設定します。
ボリュームグループの設定 ダイアログボックスが開きます。

注記

設定ダイアログではボリュームグループの物理エクステントのサイズは指定できません。このサイズは、常にデフォルト値の 4 MiB に設定されます。別の物理エクステントのボリュームグループを作成する場合は、対話シェルに切り替えて、**vgcreate** コマンドで手動で作成するか、キックスタートファイルで **volgroup --pesize=size** コマンドを使用して作成します。詳細は **高度な RHEL 8 インストールの実行** を参照してください。

4. **RAID レベル** ドロップダウンメニューから、必要な RAID レベルを選択します。
利用可能な RAID レベルは、実際の RAID デバイスと同じです。

5. ボリュームグループに暗号化のマークを付けるには、**暗号化** チェックボックスを選択します。

6. **サイズポリシー** ドロップダウンメニューから、ボリュームグループのサイズポリシーを選択します。
利用可能なポリシーオプションは以下のようになります。

   - **自動**:ボリュームグループのサイズは自動で設定されるため、設定した論理ボリュームを格納するのに適切なサイズになります。ボリュームグループに空の領域が必要ない場合に最適です。

   - **可能な限り大きく**: 設定した論理ボリュームのサイズに関係なく、最大サイズのボリュームグループが作成されます。これは、ほとんどのデータを LVM に保存する場合、または後で既存の論理ボリュームのサイズを拡大する可能性がある場合、もしくはこのグループに別の論理ボリュームを作成する必要がある場合などに最適です。

   - **固定**:このオプションではボリュームグループのサイズを正確に設定できます。設定している論理ボリュームが格納できるサイズにする必要があります。ボリュームグループに設定する容量が正確に分かっている場合に便利です。

7. **保存** をクリックして設定を適用し、**手動パーティション設定** 画面に戻ります。

8. **設定を更新** をクリックして、変更を保存します。

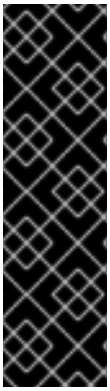
9. **完了** をクリックして、**インストール概要** 画面に戻ります。

関連情報

- IBM Z、LinuxONE、および PAES 暗号で dm-crypt を使用する方法

4.7. ROOT パスワードの設定

インストールプロセスを完了し、管理者 (スーパーユーザーまたは root としても知られている) アカウントでログインするには、**root** パスワードを設定する必要があります。これらのタスクには、ソフトウェアパッケージのインストールおよび更新と、ネットワーク、ファイアウォール設定、ストレージオプションなどのシステム全体の設定の変更と、ユーザー、グループ、およびファイルのパーミッションの追加または修正が含まれます。



重要

- インストール済みシステムに root 権限を取得するには、以下のいずれか、両方の方法を行います。
  - root アカウントの使用
  - 管理者権限を持つユーザーアカウント (wheel グループのメンバー) の作成。**root** アカウントは、インストール中に作成されます。管理者アクセスが必要なタスクを実行する必要がある場合に限り、管理者アカウントに切り替えてください。

警告

root アカウントは、システムを完全に制御できます。このアカウントへのアクセスを不正に入手すると、ユーザーの個人ファイルへのアクセスや削除が可能になります。

手順

1. **インストール概要** 画面から、**ユーザー設定 > root パスワード** を選択します。**root パスワード** 画面が開きます。
2. **root パスワード** フィールドにパスワードを入力します。
   強固な root パスワードを作成する際の必須要件と推奨事項を以下に示します。
   - 最低でも 8 文字の長さが **必要**
   - 数字、文字 (大文字と小文字)、記号を含めることができる
   - 大文字と小文字が区別される
3. **確認** フィールドにも同じパスワードを入力します。
4. **完了** をクリックして root パスワードを確定し、**インストール概要** 画面に戻ります。

注記

弱いパスワードを使用した場合は、**完了** を 2 回クリックする必要があります。

4.8. ユーザーアカウントの作成

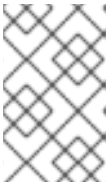
ユーザーアカウントを作成してインストールを完了することが推奨されます。ユーザーアカウントを作成しない場合は、 **root** ユーザーとしてシステムに直接ログインする必要がありますが、この方法は**推奨されていません**。

手順

1. **インストール概要画面** で、**ユーザー設定 > ユーザーの作成** を選択します。**ユーザーの作成** 画面が開きます。

2. **フルネーム** フィールドに、ユーザーアカウント名を入力します。John Smith.

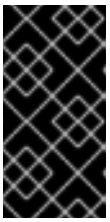
3. **ユーザー名** フィールドに、ユーザー名 (jsmith など) を入力します。



注記

コマンドラインからログインするには、**ユーザー名** を使用します。 グラフィカル環境をインストールする場合、グラフィカルログインマネージャーは、**フルネーム** を使用します。

4. ユーザーに管理者権限が必要な場合は、**このユーザーを管理者にする** チェックボックスを選択します (インストールプログラムにより、このユーザーが **wheel** グループに追加されます)。



重要

管理者ユーザーは、**sudo** コマンドを実行し、**root** パスワードの代わりにユーザーパスワードを使用して、**root** のみが実行できるタスクを実行できます。こちらを使用した方が便利な場合もありますが、セキュリティーリスクを引き起こす可能性があります。

5. Require a password to use this accountチェックボックスを選択します。

警告

ユーザーに管理者権限を付与する場合は、そのアカウントがパスワードで保護されていることを確認してください。アカウントにパスワードを割り当てない場合は、ユーザーに管理者特権を与えないでください。

6. Password フィールドにパスワードを入力します。

7. Confirm password フィールドに同じパスワードを入力します。

8. **Done** をクリックして変更を適用し、Installation Summary 画面に戻ります。

4.9. ユーザーの詳細設定の編集

以下の手順では、Advanced User Configuration ダイアログボックスでユーザーアカウントのデフォルト設定を編集する方法を説明します。

手順

1. Create User 画面で、**Advanced** をクリックします。

2. 必要に応じて、Home directory フィールドの詳細を変更します。このフィールドには、デフォルトで **/home/username** が表示されます。

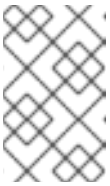
3. User and Groups IDsセクションでは、次のことができます。

a. **Specify a user ID manually**チェックボックスを選択し、**+** または **-** を使用して、必要な値を入力します。

注記

デフォルト値は 1000 です。ユーザー ID (UID) の 0 ~ 999 はシステムが予約しているため、ユーザーに割り当てることができません。

b. **Specify a group ID manually**チェックボックスを選択し、**+** または **-** を使用して、必要な値を入力します。



注記

デフォルトのグループ名はユーザー名と同じで、デフォルトのグループ ID (GID) は 1000 です。GID の 0 ~ 999 はシステムが予約しているため、ユーザーグループに割り当てることができません。

4. **Group Membership** フィールドに、コンマ区切りの追加グループリストを指定します。グループが存在しない場合は作成されます。追加されるグループにカスタムの GID を指定する場合は、カスタムの GID を括弧に入れて指定します。新しいグループにカスタムの GID を指定しない場合は、GID が自動的に割り当てられます。

注記

作成されたユーザーアカウントには、デフォルトグループメンバーシップが常に 1 つあります (**Specify a group ID manually** フィールドに設定した ID を持つユーザーのデフォルトグループ)。

5. **変更の保存** をクリックして更新を適用し、**ユーザーの作成** 画面に戻ります。

第5章 インストール後の作業の完了

このセクションは、インストール後のタスクを完了する方法を説明します。

- 初期セットアップの完了
- システムの登録

注記

要件によって、システムを登録する方法は複数あります。これらのメソッドのほとんどは、インストール後作業の一部として完了します。ただし、Red Hat コンテンツ配信ネットワーク (CDN) は、インストールプロセスを開始する **前** に、システムを登録して、RHEL サブスクリプションを割り当てます。

詳細は、CDN から RHEL の登録およびインストール を参照してください。

- システムの保護

5.1. 初期セットアップの完了

このセクションは、Red Hat Enterprise Linux 8 システムの初期セットアップを完了する方法を説明します。

重要

- インストール時に **Server with GUI** ベース環境を選択した場合は、インストールプロセスが完了し、システムを最初に再起動する際に、**初期セットアップ** 画面が開きます。
- CDN から RHEL を登録してインストールした場合は、Subscription Manager オプションに、インストールされているすべての製品に有効なエンタイトルメントが割り当てられているというメッセージが表示されます。

初期セットアップ 画面に表示される情報は、インストール時に設定した内容により異なる場合があります。ただし、**ライセンス** オプションおよび **Subscription Manager** オプションは必ず表示されます。

前提条件

- カスタマーポータルから ISO イメージを使用した RHEL のインストール に記載されている推奨ワークフローに従って、グラフィカルインストールを完了している。
- アクティブで、評価版以外の Red Hat Enterprise Linux サブスクリプションがある。

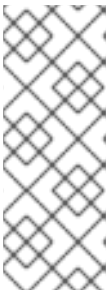
手順

1. **初期セットアップ** 画面で、**ライセンス情報** を選択します。
   ライセンス契約 画面が開き、Red Hat Enterprise Linux のライセンス条項が表示されます。
2. 使用許諾契約書を確認して、**ライセンス契約に同意します** チェックボックスを選択します。

注記

ライセンス契約への同意が必要です。この手順を完了せずに **初期セットアップ** を終了すると、システムが再起動します。再起動プロセスが完了すると、ライセンス契約に同意するように求められます。

3. **完了** をクリックして設定を適用し、**初期セットアップ** 画面に戻ります。



注記

ネットワーク設定を設定していない場合は、システムをすぐに登録できません。この場合は **設定完了** をクリックします。Red Hat Enterprise Linux 8 が起動したらログインして、ネットワークアクセスを有効にし、システムを登録します。詳細は、サブスクリプションマネージャーのインストール後 を参照してください。ネットワークのホスト名 で説明されているように、ネットワーク設定を設定している場合は、以下に示すようにシステムをすぐに登録できます。

4. **インストール概要** 画面で、**サブスクリプションマネージャー** を選択します。

重要

CDN から RHEL を登録してインストールした場合は、Subscription Manager オプションに、インストールされているすべての製品に有効なエンタイトルメントが割り当てられているというメッセージが表示されます。

5. **サブスクリプションマネージャー** グラフィカルインターフェイスを開き、登録しようとしているオプション (**subscription.rhsm.redhat.com**) を表示します。
6. **次へ** をクリックします。
7. **ログイン** および **パスワード** を入力し、**登録** ボタンをクリックします。
8. サブスクリプションの詳細を確認し、**割り当て** をクリックします。次の確認メッセージを受け取る必要があります。**Red Hat Subscription Management への登録が完了しました。**
9. **完了** をクリックします。**初期セットアップ** 画面が開きます。
10. **設定完了** をクリックしてください。ログイン画面が表示されます。
11. システムを設定します。詳細は **基本的なシステム設定の設定** を参照してください。

関連情報

要件によって、システムを登録する方法は 5 つあります。

- Red Hat コンテンツ配信ネットワーク (CDN) を使用してシステムを登録し、RHEL サブスクリプションを割り当て、Red Hat Enterprise Linux をインストール。詳細は、GUI を使用した CDN からの登録およびインストール を参照してください。
- **初期セットアップ** を使用してインストール中に。
- インストール後にコマンドラインを使用して登録
- インストール後に Subscription Manager ユーザーインターフェイスを使用して登録詳細は、サブスクリプションマネージャーのインストール後の UI を参照してください。

- インストール後に Registration Assistant で登録。Registration Assistant は、お使いの Red Hat Enterprise Linux 環境に最適な登録オプションの選択をサポートします。詳細は Registration Assistant を参照してください。

5.2. コマンドラインでシステムの登録

本セクションは、コマンドラインで Red Hat Enterprise Linux 8 サブスクリプションを登録する方法を説明します。

注記

- システムを自動登録すると、サブスクリプションサービスは、システムが物理システムまたは仮想システムであるかと、システムにあるソケット数を確認します。通常、物理システムはエンタイトルメントを 2 つを使用し、仮想システムは 1 つ使用します。システムのソケット 2 個に対して、エンタイトルメントが 1 つ必要です。
- ホストを Red Hat に登録するエクスペリエンスを改善および簡素化するには、リモートホスト設定 (RHC) を使用します。RHC クライアントはシステムを Red Hat Insights および Red Hat Subscription Manager に登録し、システムを Insights データ収集の準備を整え、Insights for Red Hat Enterprise Linux から直接問題を修正できるようにします。詳細は、RHC の登録 と Insights を使用した修復 を参照してください。

前提条件

- アクティブで、評価版ではない Red Hat Enterprise Linux サブスクリプションを持っている。
- Red Hat のサブスクリプションステータスを確認している。
- Red Hat Enterprise Linux 8 サブスクリプションを受け取ったことがない。
- カスタマーポータルからエンタイトルメントをダウンロードする前に、サブスクリプションをアクティベートしている。使用が予定されているインスタンスごとに、エンタイトルメントが 1 つ必要です。サブスクリプションのアクティベートに関するご質問は、Red Hat カスタマーサービスにお問い合わせください。
- Red Hat Enterprise Linux 8 システムを正常にインストールし、root としてログインしている。

手順

1. 端末ウィンドウを開き、Red Hat カスタマーポータルのユーザー名とパスワードを使用して Red Hat Enterprise Linux システムを登録します。

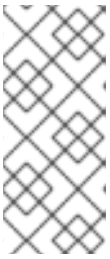
```
# subscription-manager register --username [username] --password [password]
```

2. システムが正常に登録されると、次の例のような出力が表示されます。

```
# The system has been registered with ID: 123456abcdef
# The registered system name is: localhost.localdomain
```

3. システムのロールを設定します。次に例を示します。

```
# subscription-manager role --set="Red Hat Enterprise Linux Server"
```



注記

利用可能なロールは、組織が購入したサブスクリプションと、Red Hat Enterprise Linux 8 システムのアーキテクチャーによって異なります。次のロールのいずれかを設定できます。**Red Hat Enterprise Linux Server**、**Red Hat Enterprise Linux Workstation**、または **Red Hat Enterprise Linux Compute Node** が含まれます。

4. システムのサービスレベルを設定します。次に例を示します。

```
# subscription-manager service-level --set="Premium"
```

5. たとえば、システムの使用目的を設定します。

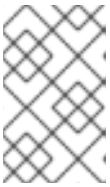
```
# subscription-manager usage --set="Production"
```

6. ホストシステムのアーキテクチャーに一致するエンタイトルメントに、システムを登録します。

```
# subscription-manager attach --auto
```

7. サブスクリプションが正常に登録されると、次のような出力が表示されます。

```
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```



注記

Red Hat Enterprise Linux 8 システムを登録する別の方法は、**root** ユーザーとしてシステムログインし、Subscription Manager グラフィカルユーザーインターフェイスを使用することです。

5.3. SUBSCRIPTION MANAGER ユーザーインターフェイスを使用したシステム登録

このセクションでは、Subscription Manager ユーザーインターフェイスを使用して Red Hat Enterprise Linux 8 システムを登録し、更新を受け取って、パッケージリポジトリーにアクセスする方法を説明します。

前提条件

- カスタマーポータルから ISO イメージを使用した RHEL のインストール に記載されている、推奨されるワークフローに従って、グラフィカルインストールを完了している。
- アクティブで、評価版以外の Red Hat Enterprise Linux サブスクリプションがある。
- Red Hat のサブスクリプションステータスを確認している。

手順

手順

1. システムにログインします。
2. 画面左上で、**アクティビティー** をクリックします。
3. メニューオプションから、**アプリケーションを表示する** アイコンをクリックします。
4. **Red Hat Subscription Manager** アイコンをクリックするか、検索に **Red Hat Subscription Manager** と入力します。
5. **認証が必要です** ダイアログボックスで管理者パスワードを入力します。

注記

システムで特権タスクを実行するには、認証が必要です。

6. **サブスクリプション** 画面が開き、サブスクリプションの現在のステータス、システムの目的、インストール済み製品が表示されます。未登録の製品には、赤い X 印が表示されます。
7. **登録** ボタンをクリックします。
8. **システムの登録** ダイアログボックスが開きます。**カスタマーポータル** の認証情報を入力して、**登録** ボタンをクリックします。

サブスクリプション 画面の **登録** ボタンが **登録解除** に変更し、インストール済み製品は緑の X が表示されます。登録に失敗した場合は、**subscription-manager status** コマンドを実行してターミナルウィンドウからトラブルシューティングを行うことができます。

関連情報

- Using and Configuring Red Hat Subscription Manager
- Configuring Virtual Machine Subscriptions in Red Hat Subscription Management

5.4. インストーラー GUI を使用した RHEL 8 の登録

次の手順を使用し、RHEL インストーラー GUI を使用して新しくインストールされた Red Hat Enterprise Linux 8 を登録します。

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。Create a Red Hat Login ページを参照してください。
  ユーザーアカウントに適切なエンタイトルメントがある場合 (またはアカウントが Simple Content Access モードで動作する場合)、アクティベーションキーを提示せずに、ユーザー名とパスワードのみで登録することが可能です。
- 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. **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。
2. **Set System Purpose** フィールドを選択し、ドロップダウンメニューから、**Role**、**SLA**、および **Usage for the RHEL 8 installation**を選択します。

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。

5.5. REGISTRATION ASSISTANT

Registration Assistant は、お使いの Red Hat Enterprise Linux 環境に最適な登録オプションの選択をサポートします。詳細は Registration Assistant を参照してください。

5.6. SUBSCRIPTION-MANAGER コマンドラインツールを使用したシステムの目的の設定

システムの目的は任意ですが、Red Hat Enterprise Linux インストールで推奨される機能です。システムの目的を使用して、Red Hat Enterprise Linux 8 システムの使用目的を記録し、エンタイトルメントサーバーがシステムに最も適したサブスクリプションを自動的に割り当てていることを確認することができます。インストールプロセスでシステムの目的を設定しなかった場合は、インストール後に **subscription-manager syspurpose** コマンドラインツールを使用して必要な属性を設定できます。

前提条件

- Red Hat Enterprise Linux 8 システムをインストールして登録しているが、システムの目的が設定されていない。
- **root** ユーザーとしてログインしている。

注記

システムが登録されているものの、必要な目的を満たさないサブスクリプションをお持ちの場合は、**subscription-manager remove --all** コマンドを実行して、割り当てたサブスクリプションを削除できます。次に、コマンドラインの subscription-manager syspurpose {ロール、使用条件、サービスレベル} ツールを使用して必要な目的属性を設定し、最後に subscription-manager attach --auto を実行して、更新した属性を考慮してシステムを再登録できます。

手順

以下の手順を完了して、インストール後に、**subscription-manager syspurpose** コマンドラインツールでシステムの目的を設定します。選択した値は、エンタイトルメントサーバーが最適なサブスクリプションをシステムに割り当てるために使用されます。

1. 端末で、次のコマンドを実行して、システムの目的のロールを設定します。

```
# subscription-manager syspurpose role --set "VALUE"
```

VALUE を、割り当てるロールに置き換えます。

- **Red Hat Enterprise Linux Server**
- **Red Hat Enterprise Linux Workstation**
- **Red Hat Enterprise Linux Compute Node**

以下に例を示します。

```
# subscription-manager syspurpose role --set "Red Hat Enterprise Linux Server"
```

a. オプション:値を設定する前に、組織のサブスクリプションがサポートする利用可能なロールを確認します。

```
# subscription-manager syspurpose role --list
```

b. オプション:次のコマンドを実行してロールの設定を解除します。

```
# subscription-manager syspurpose role --unset
```

2. 次のコマンドを実行して、希望するシステムのサービスレベルアグリーメント (SLA) を設定します。

```
# subscription-manager syspurpose service-level --set "VALUE"
```

VALUE を、割り当てる SLA に置き換えます。

- **Premium**
- **Standard**
- **Self-Support**

以下に例を示します。

```
# subscription-manager syspurpose service-level --set "Standard"
```

a. オプション:値を設定する前に、組織のサブスクリプションがサポートする利用可能なサービスレベルを確認します。

```
# subscription-manager syspurpose service-level --list
```

b. オプション:次のコマンドを実行して SLA の設定を解除します。

```
# subscription-manager syspurpose service-level --unset
```

3. 次のコマンドを実行して、希望する使用方法をシステムに設定します。

```
# subscription-manager syspurpose usage --set "VALUE"
```

VALUE を、割り当てる使用方法に置き換えます。

- **Production**
- **Disaster Recovery**
- **Development/Test**

以下に例を示します。

```
# subscription-manager syspurpose usage --set "Production"
```

a. オプション:値を設定する前に、組織のサブスクリプションがサポートする利用可能な使用条件を確認します。

```
# subscription-manager syspurpose usage --list
```

b. オプション:次のコマンドを実行して使用率の設定を解除します。

```
# subscription-manager syspurpose usage --unset
```

4. 次のコマンドを実行して、現在のシステム目的のプロパティーを表示します。

```
# subscription-manager syspurpose --show
```

a. オプション:詳細な構文情報については、以下のコマンドを実行して **subscription-manager** の man ページにアクセスし、SYSPURPOSE OPTIONS を参照します。

```
# man subscription-manager
```

検証手順

- システムのサブスクリプションのステータスを確認するには、以下を実行します。

```
# subscription-manager status
+-------------------------------------------+
   System Status Details
+-------------------------------------------+
Overall Status: Current

System Purpose Status: Matched
```

- 全体的なステータス **Current** とは、インストールされている製品がすべて割り当てられたサブスクリプションの対象となり、コンテンツセットリポジトリーにアクセスするためのエンタイトルメントが付与されています。
- システム目的のステータス **Matched** とは、システムに設定したすべてのシステム目的の属性 (ロール、使用条件、サービスレベル) が、割り当てられたサブスクリプションによって満たされることを意味します。
- ステータス情報が理想的ではない場合、システム管理者がインストール済みの製品と目的のシステムの目的に対応するために、アタッチされているサブスクリプションに加える修正を決定するのに役立つ追加情報が表示されます。

5.7. システムの保護

Red Hat Enterprise Linux をインストールしたらすぐに、次のセキュリティー関連の手順を完了してください。

前提条件

- グラフィカルインストールを完了している。

手順

1. root で以下のコマンドを実行して、システムを更新します。

```
# yum update
```

2. ファイアウォールサービスの **firewalld** は、Red Hat Enterprise Linux のインストールで自動的に有効になっていますが、キックスタート設定などで明示的に無効となっている場合もあります。このような場合は、ファイアウォールを再度有効にすることが推奨されます。
firewalld を開始するには、root で次のコマンドを実行します。

```
# systemctl start firewalld
# systemctl enable firewalld
```

3. セキュリティーを強化するために、不要なサービスは無効にしてください。たとえば、コンピューターにプリンターがインストールされていなければ、次のコマンドを実行して cups サービスを無効にします。

```
# systemctl mask cups
```

アクティブなサービスを確認するには、次のコマンドを実行します。

```
$ systemctl list-units | grep service
```

5.8. インストール直後にセキュリティープロファイルに準拠するシステムのデプロイメント

OpenSCAP スイートを使用して、インストールプロセスの直後に、OSPP や PCI-DSS、HIPAA プロファイルなどのセキュリティープロファイルに準拠する RHEL システムをデプロイできます。このデプロイメント方法を使用すると、修正スクリプトを使用して後で適用できない特定のルール (パスワードの強度とパーティション化のルールなど) を適用できます。

5.8.1. GUI を備えたサーバーと互換性のないプロファイル

SCAP セキュリティーガイド の一部として提供される一部のセキュリティープロファイルは、**Server with GUI** ベースの環境の拡張パッケージセットと互換性がない場合があります。したがって、次のプロファイルのいずれかに準拠するシステムをインストールする場合は、**Server with GUI**を選択しないでください。

表5.1 GUI を備えたサーバーと互換性のないプロファイル

プロファイル名	プロファイル ID	理由	備考
CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server	**xccdf_org.ssgproject.content_profile_cis**	パッケージ **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils**、および **xorg-x11-server-Xwayland** は、Server with GUIパッケージセットの一部ですが、ポリシーではそれらを削除する必要があります。	

プロファイル名	プロファイル ID	理由	備考
CIS Red Hat Enterprise Linux 8 Benchmark for Level 1 - Server	**xccdf_org.ssgproject.content_profile_cis_server_l1**	パッケージ **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils**、および **xorg-x11-server-Xwayland** は、Server with GUIパッケージセットの一部ですが、ポリシーではそれらを削除する必要があります。	
Unclassified Information in Non-federal Information Systems and Organizations (NIST 800-171)	**xccdf_org.ssgproject.content_profile_cui**	**nfs-utils** パッケージはServer with GUIパッケージセットの一部ですが、ポリシーではその削除が必要です。	
Protection Profile for General Purpose Operating Systems	**xccdf_org.ssgproject.content_profile_ospp**	**nfs-utils** パッケージはServer with GUIパッケージセットの一部ですが、ポリシーではその削除が必要です。	
DISA STIG for Red Hat Enterprise Linux 8	**xccdf_org.ssgproject.content_profile_stig**	パッケージ **xorg-x11-server-Xorg**、**xorg-x11-server-common**、**xorg-x11-server-utils**、および **xorg-x11-server-Xwayland** は、Server with GUIパッケージセットの一部ですが、ポリシーではそれらを削除する必要があります。	RHEL バージョン 8.4 以降で、RHEL システムを DISA STIG に準拠した Server with GUI としてインストールするには、**DISA STIG with GUI** プロファイルを使用できます。

5.8.2. グラフィカルインストールを使用したベースライン準拠の RHEL システムのデプロイメント

この手順を使用して、特定のベースラインに合わせた RHEL システムをデプロイします。この例では、OSPP (Protection Profile for General Purpose Operating System) を使用します。

警告

SCAP セキュリティーガイド の一部として提供される一部のセキュリティープロファイルは、**Server with GUI** ベースの環境の拡張パッケージセットと互換性がない場合があります。詳細については、GUI サーバーと互換性のないプロファイル を参照してください。

前提条件

- **グラフィカル** インストールプログラムでシステムを起動している。**OSCAP Anaconda アドオン** はインタラクティブなテキストのみのインストールをサポートしていないことに注意してください。
- **インストール概要** 画面を開いている。

手順

1. **インストール概要** 画面で、**ソフトウェアの選択** をクリックします。**ソフトウェアの選択** 画面が開きます。
2. **ベース環境** ペインで、**サーバー** 環境を選択します。ベース環境は、1 つだけ選択できます。
3. **完了** をクリックして設定を適用し、**インストール概要** 画面に戻ります。
4. **セキュリティーポリシー** をクリックします。**セキュリティーポリシー** 画面が開きます。
5. システムでセキュリティーポリシーを有効にするには、**セキュリティーポリシーの適用** を **ON** に切り替えます。
6. プロファイルペインで **Protection Profile for General Purpose Operating Systems** プロファイルを選択します。
7. **プロファイルの選択** をクリックして選択を確定します。
8. 画面下部に表示される **Protection Profile for General Purpose Operating Systems** の変更を確定します。残りの手動変更を完了します。
9. OSPP には、準拠する必要がある厳密なパーティション分割要件があるため、**/boot**、**/home**、**/var**、**/var/log**、**/var/tmp**、および **/var/log/audit** にそれぞれパーティションを作成します。
10. グラフィカルインストールプロセスを完了します。

注記

グラフィカルインストールプログラムは、インストールに成功すると、対応するキックスタートファイルを自動的に作成します。**/root/anaconda-ks.cfg** ファイルを使用して、OSPP 準拠のシステムを自動的にインストールできます。

検証

- インストール完了後にシステムの現在のステータスを確認するには、システムを再起動して新しいスキャンを開始します。

```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

関連情報

- 手動パーティションの設定

5.8.3. キックスタートを使用したベースライン準拠の RHEL システムのデプロイメント

この手順を使用して、特定のベースラインに合わせた RHEL システムをデプロイします。この例では、OSPP (Protection Profile for General Purpose Operating System) を使用します。

前提条件

- RHEL 8 システムに、**scap-security-guide** パッケージがインストールされている。

手順

1. キックスタートファイル **/usr/share/scap-security-guide/kickstart/ssg-rhel8-ospp-ks.cfg** を、選択したエディターで開きます。
2. 設定要件を満たすように、パーティション設定スキームを更新します。OSPP コンプライアンスでは、**/boot**、**/home**、**/var**、**/var/log**、**/var/tmp**、および **/var/log/audit** にそれぞれ設定したパーティションを維持して、パーティションのサイズのみを変更できます。
3. キックスタートインストールを開始する方法は、キックスタートインストールの開始を参照してください。

重要

キックスタートファイルのパスワードでは、OSPP の要件が確認されていません。

検証

1. インストール完了後にシステムの現在のステータスを確認するには、システムを再起動して新しいスキャンを開始します。

```
# oscap xccdf eval --profile ospp --report eval_postinstall_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

関連情報

- OSCAP Anaconda Add-on

5.9. 次のステップ

インストール後の必要なステップを完了したら、基本的なシステム設定を設定できます。yum を使用したソフトウェアのインストール、systemd を使用したサービスの管理、ユーザー、グループ、およびファイルパーミッションの管理、chrony を使用した NTP の設定、および Python 3 の使用などの作業は、**基本的なシステム設定の設定** を参照してください。

付録A トラブルシューティング

以下のセクションでは、インストールプロセスの各段階で問題を診断するのに役に立つさまざまなトラブルシューティング情報を説明します。

付録B トラブルシューティングおよびバグ報告のためのツールおよびヒント

以下のセクションのトラブルシューティング情報は、インストールプロセスの開始時に問題を診断する際に役に立つ場合があります。以下のセクションは、サポートしているすべてのアーキテクチャーに対応します。ただし、問題が特定のアーキテクチャーに関する場合は、セクションの冒頭にその旨が記載されます。

B.1. Dracut

Dracut は、Linux オペレーティングシステムの起動プロセス時に **initramfs** イメージを管理するツールです。**dracut** の緊急シェルは、**initramfs** イメージが読み込まれる際に開始できるインタラクティブモードです。**dracut** の緊急シェルから基本的なトラブルシューティングコマンドを実行できます。詳細は、**dracut** の man ページの **Troubleshooting** セクションを参照してください。

B.2. インストールログファイルの使用

デバッグの目的で、インストールプログラムは、**/tmp** ディレクトリーにあるファイルに、インストールアクションのログを記録します。以下の表は、ログファイルのリストです。

表B.1 インストール時に生成されるログファイル

ログファイル	内容
/tmp/anaconda.log	一般メッセージ
/tmp/program.log	インストール時に実行したすべての外部プログラム
/tmp/storage.log	ストレージモジュールの詳細情報
/tmp/packaging.log	yum パッケージおよび rpm パッケージのインストールメッセージ
/tmp/dbus.log	インストールプログラムモジュールに使用される **dbus** セッションに関する情報
/tmp/sensitive-info.log	他のログに含まれず、インストール後のシステムにコピーされない設定情報
/tmp/syslog	ハードウェア関連のシステムメッセージこのファイルには、他の Anaconda ファイルからのメッセージが含まれます。

インストールが失敗すると、メッセージは **/tmp/anaconda-tb-identifier** で一元管理されます。identifier はランダムな文字列になります。インストールに成功すると、このファイルは **/var/log/anaconda/** ディレクトリー下のインストール済みシステムにコピーされます。ただし、インストールが失敗した場合、またはインストールシステムの起動時に **inst.nosave=all** オプションまたは **inst.nosave=logs** オプションを使用すると、ログはインストールプログラムの RAM ディスクにのみ存在します。これは、ログが永続的に保存されず、システムの電源が切れると失われることを意味します。永続的に保存するには、ファイルをネットワーク上の別のシステムにコピーするか、マウントしたストレージデバイス (USB フラッシュドライブなど) にコピーします。

B.2.1. インストール前のログファイルの作成

この手順に従って、インストールプロセスを開始する前にログファイルを作成する **inst.debug** オプションを設定します。このログファイルには、たとえば現在のストレージ設定が含まれます。

前提条件

- Red Hat Enterprise Linux ブートメニューが開いている。

手順

1. 起動メニューから **Install Red Hat Enterprise Linux**オプションを選択します。
2. BIOS ベースのシステムで **Tab** キーを押します。または UEFI ベースのシステムで **e** キーを押して、選択した起動オプションを編集します。
3. オプションに **inst.debug** を追加します。以下に例を示します。

```
vmlinuz ... inst.debug
```

4. キーボードの **Enter** キーを押します。システムが、インストール前のログファイルを **/tmp/pre-anaconda-logs/** ディレクトリーに保存し、インストールプログラムが開始します。
5. ログファイルにアクセスするには、コンソールに切り替えます。
6. **/tmp/pre-anaconda-logs/** ディレクトリーに移動します。

```
# cd /tmp/pre-anaconda-logs/
```

関連情報

- Boot オプションの参照
- インストール時のコンソールロギング

B.2.2. インストールログファイルを USB ドライブへ転送

以下の手順に従って、インストールログファイルを USB ドライブに転送します。

前提条件

- USB ドライブからデータをバックアップした。
- root アカウントにログインし、インストールプログラムの一時ファイルシステムにアクセスできるようにする。

手順

1. **Ctrl** + **Alt** + **F2** を押して、インストールするシステムのシェルプロンプトにアクセスします。
2. USB フラッシュドライブをシステムに接続し、**dmesg** コマンドを実行します。

```
# dmesg
```

最近の全イベントの詳細を記録したログが表示されます。このログの最後に、一連のメッセージが表示されます。以下に例を示します。

```
[ 170.171135] sd 5:0:0:0: [sdb] Attached SCSI removable disk
```

3. 接続したデバイスの名前を書き留めます。上記の例では **sdb** です。
4. **/mnt** ディレクトリーに移動し、USB ドライブのマウントターゲットとして機能する新規ディレクトリーを作成します。この例では **usb** という名前を使用します。

```
# mkdir usb
```

5. USB フラッシュドライブを、新たに作成したディレクトリーにマウントします。ほとんどの場合、ドライブ全体ではなく、ドライブのパーティションをマウントする必要があります。**sdb** の名前は使用せず、ログファイルを書き込むパーティションの名前を使用してください。この例では、**sdb1** という名前を使用します。

```
# mount /dev/sdb1 /mnt/usb
```

6. デバイスにアクセスし、そのコンテンツをリスト表示して、正しいデバイスをマウントしたことを確認します。

```
# cd /mnt/usb
```

```
# ls
```

7. ログファイルを、マウントしたデバイスにコピーします。

```
# cp /tmp/*log /mnt/usb
```

8. USB フラッシュドライブのマウントを解除します。ターゲットがビジーであるというエラーメッセージが表示された場合は、作業ディレクトリーをマウント外 (たとえば /) に変更します。

```
# umount /mnt/usb
```

B.2.3. ネットワーク経由でインストールログファイルの転送

以下の手順に従って、インストールログファイルをネットワーク経由で転送します。

前提条件

- root アカウントにログインし、インストールプログラムの一時ファイルシステムにアクセスできるようにする。

手順

1. **Ctrl** + **Alt** + **F2** を押して、インストールするシステムのシェルプロンプトにアクセスします。
2. ログファイルが格納されている **/tmp** ディレクトリーに移動します。

```
# cd /tmp
```

3. **scp** コマンドを使用して、ネットワーク経由でログファイルを別のシステムにコピーします。

```
# scp *log user@address:path
```

a. **user** には、ターゲットシステムの有効なユーザー名を入力します。**address** には、ターゲットシステムのアドレスまたはホスト名を入力します。**path** には、ログファイルを保存するディレクトリーへのパスを入力します。たとえば、IP アドレスが 192.168.0.122 のシステムに **john** としてログインし、ログファイルをそのシステムの **/home/john/logs/** ディレクトリーに置く場合のコマンドは次のようになります。

```
# scp *log john@192.168.0.122:/home/john/logs/
```

初めてターゲットシステムに接続する際に、SSH クライアントにより、リモートシステムのフィンガープリントが正しいことと、継続するかを尋ねられます。

```
The authenticity of host '192.168.0.122 (192.168.0.122)' can't be established.
ECDSA key fingerprint is a4:60:76:eb:b2:d0:aa:23:af:3d:59:5c:de:bb:c4:42.
Are you sure you want to continue connecting (yes/no)?
```

b. **yes** と入力し、**Enter** を押して続行します。プロンプトが表示されたら、有効なパスワードを入力します。ファイルは、ターゲットシステムの指定されたディレクトリーに転送されます。

B.3. Memtest86 アプリケーションの使用によるメモリー障害の検出

メモリー (RAM) モジュールの障害により、システムで予期しないエラーが生じる可能性があります。特定の状況では、メモリー障害は、ソフトウェアの特定の組み合わせでのみエラーが発生する可能性があります。このため、Red Hat Enterprise Linux をインストールする前に、システムのメモリーをテストする必要があります。

注記

Red Hat Enterprise Linux には、BIOS システム用の **Memtest86+** メモリーテストアプリケーションのみが含まれます。UEFI システムのサポートは現在利用できません。

B.3.1. Memtest86 の実行

Red Hat Enterprise Linux をインストールする前に、この手順で **Memtest86** アプリケーションを実行し、システムでメモリー障害をテストします。

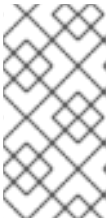
前提条件

- Red Hat Enterprise Linux 起動メニューにアクセスできる。

手順

1. Red Hat Enterprise Linux 起動メニューから **Troubleshooting > Run a memory test**を選択します。**Memtest86** アプリケーション画面が表示され、テストがすぐに開始します。デフォルトでは、**Memtest86** はすべてのパスで 10 個のテストを実行します。最初のパスが完了すると、画面の下部に、現在のステータスを知らせるメッセージが表示されます。その他のパスは自動的に開始します。
Memtest86+ がエラーを検出すると、画面の中央ペインにエラーが表示され、赤で強調表示されます。メッセージには、問題を検出したテスト、障害が発生しているメモリーの場所などの

詳細情報が含まれます。ほとんどの場合、10 個 すべてのテストに一度成功すれば、RAM が良好な状態であることを確認できます。ただし、まれに、最初のパスで検出されなかったエラーが、後続のパスに検出される場合があります。重要なシステムで徹底的なテストを実行するには、そのテストを一晩または数日間実行して、複数のパスを完了します。



注記

Memtest86+ の 1 回の完全パスを完了するのにかかる時間は、システムの設定、特に RAM のサイズと速度により異なります。たとえば、667 MHz で 2 GiB の DDR2 メモリーを搭載したシステムでは、1 回のパスが完了するまでに 20 分かかります。

2. オプション:画面上の指示に従って **設定** 画面にアクセスし、別の設定を指定します。
3. テストを中止してコンピューターを再起動する場合は、いつでも **Esc** キーを押すことができます。

関連情報

- How to use **Memtest86**

B.4. 起動用メディアの検証

ISO イメージの検証は、インストール時にしばしば発生する問題を回避するのに役立ちます。このソースには、ハードドライブまたは NFS サーバーに保存されている DVD イメージと ISO イメージが含まれます。以下の手順に従って、Red Hat Enterprise Linux のインストールに使用する前に、ISO ベースのインストールソースの整合性をテストします。

前提条件

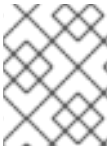
- Red Hat Enterprise Linux 起動メニューにアクセスできる。

手順

1. 起動メニューから **Test this media & install Red Hat Enterprise Linux 8.1**を選択して、起動メディアをテストします。
2. この起動プロセスは、メディアをテストして問題を強調表示します。
3. オプション:起動コマンドラインに **rd.live.check** を追加して、検証プロセスを開始できます。

B.5. インストール中のコンソールとロギング

Red Hat Enterprise Linux インストーラーは、**tmux** 端末マルチプレクサーを使用して、メインのインターフェイスのほかに複数の画面を表示し、制御します。この画面は、それぞれ目的が異なり、インストールプロセス中に発生した問題をトラブルシューティングするのに使用できるさまざまなログを表示します。画面の 1 つでは、起動オプションまたはキックスタートコマンドを使用して明示的に無効にしない限り、**root** 権限で使用できる対話式シェルプロンプトを使用できます。



注記

一般的に、インストール関連の問題を診断する必要がなければ、デフォルトのグラフィカルインストール環境から、他の環境に移動する必要はありません。

端末マルチプレクサーは、仮想コンソール 1 で実行しています。インストール環境を、**tmux** に変更する場合は、**Ctrl**+**Alt**+**F1** を押します。仮想コンソール 6 で実行されているメインのインストールインターフェイスに戻るには、**Ctrl**+**Alt**+**F6** を押します。

注記

テキストモードのインストールを選択するには、仮想コンソール 1 (**tmux**) を開始し、その後にコンソール 6 に切り替えると、グラフィカルインターフェイスではなくシェルプロンプトが開きます。

tmux を実行しているコンソールには、利用可能な画面が 5 つあります。その内容と、キーボードショートカットは、以下の表で説明します。キーボードショートカットは 2 段階となっており、最初に **Ctrl**+**b** を押し、両方のキーを離してから、使用する画面で数字キーを押す必要があります。

また、**Ctrl**+**b n**、**Alt+ Tab**、および **Ctrl**+**b p** を使用して、次または前の **tmux** 画面に切り替えることもできます。

表B.2 利用可能な tmux 画面

ショートカット	内容
Ctrl+**b 1**	メインのインストールプログラム画面。テキストベースのプロンプト (テキストモードのインストール中もしくは VNC Direct モードを使用の場合) とデバッグ情報があります。
Ctrl+**b 2**	**root** 権限のある対話式シェルプロンプト。
Ctrl+**b 3**	インストールログ: **/tmp/anaconda.log** に保存されているメッセージを表示します。
Ctrl+**b 4**	ストレージログ - **/tmp/storage.log** に保存されているストレージデバイスおよび設定に関連するメッセージを表示します。
Ctrl+**b 5**	プログラムログ - **/tmp/program.log** に保存されている、インストールプロセス時に実行するユーティリティーのメッセージを表示します。

B.6. スクリーンショットの保存

グラフィカルインストール中に **Shift**+**Print Screen** を押すと、いつでも画面をキャプチャーできます。このスクリーンショートカットは、**/tmp/anaconda-screenshots** に保存されます。

B.7. 設定およびデバイスドライバーの表示

ビデオカードの中には、Red Hat Enterprise Linux グラフィカルインストールプログラムでの起動に問題があるものがあります。インストールプログラムがデフォルト設定を使用して実行しない場合は、それより低い解像度モードでの実行を試みます。これに失敗すると、インストールプログラムはテキストモードでの実行を試みます。ディスプレイの問題を解決するソリューションは複数あります。そのほとんどは、カスタムの起動オプションを指定する必要があります。

詳細は、コンソール起動オプション を参照してください。

表B.3 ソリューション

ソリューション	説明
基本的なグラフィックモードを使用する	基本的なグラフィックスドライバーを使用して、インストールの実行を試みることができます。これを行うには、起動メニューから **Troubleshooting > Install Red Hat Enterprise Linux in basic graphics mode** を選択するか、インストールプログラムの起動オプションを編集して、コマンドラインの末尾に **inst.xdriver=vesa** を追加します。
ディスプレイの解像度を手動で指定する	インストールプログラムが画面の解像度の検出に失敗した場合は、自動検出を無効にして手動で指定できます。これには、起動メニューに **inst.resolution=x** オプションを追加します。x はディスプレイの解像度 (1024x768 など) になります。
代替のビデオドライバーを使用する	インストールプログラムの自動検出を無効にして、カスタムビデオドライバーを指定できます。ドライバーを指定するには、**inst.xdriver=x** オプションを使用します。x は使用するデバイスドライバー (nouveau など)* です。
VNC を使用したインストールを行う	上記のオプションが失敗した場合は、仮想ネットワークコンピューティング (VNC) プロトコルを使用して、別のシステムでネットワーク経由でグラフィカルインストールにアクセスできます。VNC を使用したインストールの詳細は、**高度な RHEL インストールの実行**の **VNC を使用したリモート RHEL 8 インストールの実行** を参照してください。

* カスタムのビデオドライバーを指定すると問題が解決する場合は、https://bugzilla.redhat.com で **anaconda** コンポーネントに対するバグとしてこの問題を報告してください。インストールプログラムは、ハードウェアを自動的に検出し、適切なドライバーを介入なしで使用できるようにする必要があります。

B.8. Red Hat カスタマーポータルへエラーメッセージの報告

グラフィカルインストールでエラーが発生すると、**unknown error** ダイアログボックスが表示されます。エラーに関する情報を Red Hat カスタマーサポートに送信できます。レポートを送信するには、カスタマーポータルの認証情報を入力する必要があります。カスタマーポータルアカウントをお持ちでない場合は、https://www.redhat.com/wapps/ugc/register.html で登録できます。自動エラー報告にはネットワーク接続が必要です。

前提条件

グラフィカルインストールプログラムでエラーが発生し、**unknown error** ダイアログボックスが表示されます。

手順

1. unknown error ダイアログボックスから **Report Bug** をクリックして問題を報告するか、**Quit** をクリックしてインストールを終了します。

    a. 必要に応じて、**More Info...** をクリックして、エラーの原因を特定するのに役立つ詳細な出力を表示します。デバッグに精通している場合は、**Debug** をクリックします。これにより、追加の情報を要求できる仮想ターミナル **tty1** が表示されます。**tty1** からグラフィカルインターフェイスに戻るには、**continue** コマンドを使用します。

2. **Report a bug to Red Hat Customer Support** をクリックします。

3. Red Hat Customer Support - Reporting Configurationダイアログボックスが表示されます。Basic タブで、カスタマーポータルのユーザー名およびパスワードを入力します。ネットワーク設定で HTTP プロキシーまたは HTTPS プロキシーを使用する必要がある場合は、Advanced タブを選択し、プロキシーサーバーのアドレスを入力して設定できます。

4. すべてのフィールドを完了し、**OK** をクリックします。

5. テキストボックスが表示されます。unknown error ダイアログボックスが表示される前に行った各手順を説明します。

6. How reproducible is this problem ドロップダウンメニューからオプションを選択し、テキストボックスに追加情報を入力します。

7. **Forward** をクリックします。

8. 入力したすべての情報が Comment タブにあることを確認します。他のタブには、システムのホスト名やインストール環境に関する詳細などの情報が含まれます。Red Hat に送信したくない情報は削除できますが、詳細情報が少なくなると、問題の調査に影響を及ぼす可能性があることに注意してください。

9. すべてのタブの確認が終了したら Forward をクリックします。

10. ダイアログボックスは、Red Hat に送信されるすべてのファイルを表示します。Red Hat に送信しないファイルの横にあるチェックボックスの選択を解除します。ファイルを追加するには、Attach a file をクリックします。

11. I have reviewed the data and agree with submitting it.チェックボックスを選択します。

12. **Forward** をクリックして、レポートと添付ファイルを Red Hat に送信します。

13. **Show log** をクリックしてレポートプロセスの詳細を表示します。**Close** をクリックすると、unknown error ダイアログボックスに戻ります。

14. **Quit** をクリックして、インストールを終了します。

A.1. インストール時のトラブルシューティング

以下のセクションのトラブルシューティング情報は、インストールプロセス時に問題を診断する際に役に立つ場合があります。以下のセクションは、サポートしているすべてのアーキテクチャーに対応します。ただし、問題が特定のアーキテクチャーに関する場合は、セクションの冒頭にその旨が記載されます。

A.1.1. ディスクが検出されない

インストールプログラムがインストール先となる書き込み可能なストレージデバイスを検出できない場合、**インストール先** 画面に次のエラーメッセージが返されます。**ディスクが検出されない。**Please

shut down the computer, connect at least one disk, and restart to complete installation が返ります。

以下の項目を確認します。

- システムにストレージデバイスが少なくとも 1 つ割り当てられている。
- ご使用のシステムがハードウェア RAID コントローラーを使用している場合は、そのコントローラーが正しく設定され、期待通りに機能している。手順は、コントローラーのドキュメントを参照してください。
- 1 つ以上の iSCSI デバイスにインストールし、そのシステムにローカルストレージがない場合は、必要なすべての LUN が適切なホストバスアダプター (HBA) に表示されている。

システムを再起動してインストールプロセスを開始した後もエラーメッセージが表示される場合は、インストールプログラムがストレージの検出に失敗しています。このエラーメッセージは、多くの場合、インストールプログラムで認識されない iSCSI デバイスにインストールしようとした場合に表示されます。

このシナリオでは、インストールを開始する前に、ドライバー更新を実行する必要があります。ハードウェアベンダーの Web サイトで、ドライバーの更新が利用可能かどうかを確認します。ドライバー更新に関する一般的な情報は、**高度な RHEL 8 インストールの実行**の **インストール時のドライバーの更新** を参照してください。

また、Red Hat Hardware Compatibility List (https://access.redhat.com/ecosystem/search/#/category/Server) を確認してください。

A.1.2. Red Hat カスタマーポータルへエラーメッセージの報告

グラフィカルインストールでエラーが発生すると、**unknown error** ダイアログボックスが表示されます。エラーに関する情報を Red Hat カスタマーサポートに送信できます。レポートを送信するには、カスタマーポータルの認証情報を入力する必要があります。カスタマーポータルアカウントをお持ちでない場合は、https://www.redhat.com/wapps/ugc/register.html で登録できます。自動エラー報告にはネットワーク接続が必要です。

前提条件

グラフィカルインストールプログラムでエラーが発生し、**unknown error** ダイアログボックスが表示されます。

手順

1. **unknown error** ダイアログボックスから **Report Bug** をクリックして問題を報告するか、**Quit** をクリックしてインストールを終了します。
   a. 必要に応じて、**More Info...** をクリックして、エラーの原因を特定するのに役立つ詳細な出力を表示します。デバッグに精通している場合は、**Debug** をクリックします。これにより、追加の情報を要求できる仮想ターミナル **tty1** が表示されます。**tty1** からグラフィカルインターフェイスに戻るには、**continue** コマンドを使用します。
2. **Report a bug to Red Hat Customer Support** をクリックします。
3. **Red Hat Customer Support - Reporting Configuration**ダイアログボックスが表示されます。**Basic** タブで、カスタマーポータルのユーザー名およびパスワードを入力します。ネットワーク設定で HTTP プロキシーまたは HTTPS プロキシーを使用する必要がある場合は、**Advanced** タブを選択し、プロキシーサーバーのアドレスを入力して設定できます。

4. すべてのフィールドを完了し、**OK** をクリックします。
5. テキストボックスが表示されます。**unknown error** ダイアログボックスが表示される前に行った各手順を説明します。
6. **How reproducible is this problem** ドロップダウンメニューからオプションを選択し、テキストボックスに追加情報を入力します。
7. **Forward** をクリックします。
8. 入力したすべての情報が **Comment** タブにあることを確認します。他のタブには、システムのホスト名やインストール環境に関する詳細などの情報が含まれます。Red Hat に送信したくない情報は削除できますが、詳細情報が少なくなると、問題の調査に影響を及ぼす可能性があることに注意してください。
9. すべてのタブの確認が終了したら **Forward** をクリックします。
10. ダイアログボックスは、Red Hat に送信されるすべてのファイルを表示します。Red Hat に送信しないファイルの横にあるチェックボックスの選択を解除します。ファイルを追加するには、**Attach a file** をクリックします。
11. **I have reviewed the data and agree with submitting it.**チェックボックスを選択します。
12. **Forward** をクリックして、レポートと添付ファイルを Red Hat に送信します。
13. **Show log** をクリックしてレポートプロセスの詳細を表示します。**Close** をクリックすると、**unknown error** ダイアログボックスに戻ります。
14. **Quit** をクリックして、インストールを終了します。

A.1.3. IBM Power Systems のパーティション作成の問題

注記

この問題は、IBM Power Systems の問題です。

手動でパーティションを作成したにもかかわらずインストールプロセスを進めることができない場合は、インストールを進めるために必要なパーティションがすべて作成されていない可能性があります。少なくとも、以下のパーティションが必要です。

- / **(root)** パーティション
- **PReP** 起動パーティション
- **/boot** パーティション (root パーティションが LVM 論理ボリュームの場合のみ)

関連情報

- 推奨されるパーティション設定スキーム を参照してください。

付録C トラブルシューティング

以下のセクションのトラブルシューティング情報は、インストールプロセス後に問題を診断する際に役に立つ場合があります。以下のセクションは、サポートしているすべてのアーキテクチャーに対応します。ただし、問題が特定のアーキテクチャーに関する場合は、セクションの冒頭にその旨が記載されます。

C.1. 中断されたダウンロードの再開

curl コマンドを使用して、中断したダウンロードを再開します。

前提条件

- Red Hat カスタマーポータルの **製品ダウンロード** セクション (https://access.redhat.com/downloads) に移動し、必要なバリアント、バージョン、およびアーキテクチャーを選択している。
- 必要な ISO ファイルを右クリックし、**リンク先をコピー** を選択して、ISO イメージファイルの URL をクリップボードにコピーしている。

手順

1. 新しいリンクから ISO イメージをダウンロードしてください。ダウンロードを自動的に再開するには、**--continue-at -** オプションを追加します。

```
$ curl --output directory-path/filename.iso 'new_copied_link_location' --continue-at -
```

2. ダウンロードが完了した後、イメージファイルの整合性を確認するには、**sha256sum** などのチェックサムユーティリティーを使用します。

```
$ sha256sum rhel-x.x-x86_64-dvd.iso
  `85a...46c rhel-x.x-x86_64-dvd.iso`
```

その出力を、Red Hat Enterprise Linux の Web ページ **製品ダウンロード** にある参照チェックサムと比較します。

例C.1 中断されたダウンロードの再開

以下は、部分的にダウンロードした ISO イメージに対する **curl** コマンドの例です。

```
$ curl --output _rhel-x.x-x86_64-dvd.iso
'https://access.cdn.redhat.com//content/origin/files/sha256/85/85a...46c/rhel-x.x-x86_64-dvd.iso?_auth=141...963' --continue-at -
```

C.2. ディスクが検出されない

インストールプログラムがインストール先となる書き込み可能なストレージデバイスを検出できない場合、**インストール先** 画面に次のエラーメッセージが返されます。**ディスクが検出されない。Please shut down the computer, connect at least one disk, and restart to complete installation が返ります。**

以下の項目を確認します。

- システムにストレージデバイスが少なくとも 1 つ割り当てられている。
- ご使用のシステムがハードウェア RAID コントローラーを使用している場合は、そのコントローラーが正しく設定され、期待通りに機能している。手順は、コントローラーのドキュメントを参照してください。
- 1 つ以上の iSCSI デバイスにインストールし、そのシステムにローカルストレージがない場合は、必要なすべての LUN が適切なホストバスアダプター (HBA) に表示されている。

システムを再起動してインストールプロセスを開始した後もエラーメッセージが表示される場合は、インストールプログラムがストレージの検出に失敗しています。このエラーメッセージは、多くの場合、インストールプログラムで認識されない iSCSI デバイスにインストールしようとした場合に表示されます。

このシナリオでは、インストールを開始する前に、ドライバー更新を実行する必要があります。ハードウェアベンダーの Web サイトで、ドライバーの更新が利用可能かどうかを確認します。ドライバー更新に関する一般的な情報は、**高度な RHEL 8 インストールの実行**の **インストール時のドライバーの更新** を参照してください。

また、Red Hat Hardware Compatibility List (https://access.redhat.com/ecosystem/search/#/category/Server) を確認してください。

C.3. RAID カードで起動できない

インストール後にシステムを起動できない場合は、システムのストレージのパーティション設定と再インストールが必要になる場合があります。BIOS によっては、RAID カードからの起動に対応していないため注意が必要です。インストールが完了し、初めてシステムを再起動すると、テキストベースの画面にブートローダーのプロンプト (**grub>** など) が表示され、カーソルのフラッシュが表示されます。この場合は、システムのパーティションを再設定し、**/boot** パーティションと、RAID アレイの外にあるブートローダーを移動する必要があります。**/boot** パーティションとブートローダーは、同じドライブに置く必要があります。このような変更が行われたら、インストールを完了し、システムを適切に起動できるはずです。

C.4. グラフィカルな起動シーケンスが応答しない

インストール後に初めてシステムを再起動すると、グラフィカルな起動シーケンス時にシステムが応答しなくなることがあります。この場合は、リセットが必要です。このシナリオでは、ブートローダーメニューは正常に表示されますが、エントリーを選択してシステムを起動しようとすると停止します。これは通常、グラフィカルな起動シーケンスに問題があることを示しています。この問題を解決するには、永続的に変更する前に、システムの起動時に設定を一時的に変更することで、グラフィカルブートを無効にする必要があります。

手順: グラフィカルブートを一時的に無効にする

1. システムを起動し、ブートローダーメニューが表示されるまで待ちます。起動のタイムアウト期間を **0** に設定し、**Ecc** キーを押してアクセスします。
2. ブートローダーメニューからカーソルキーを使用して、起動するエントリーを強調表示します。**Tab** キー (システムが BIOS ベースの場合) または **e** キー (UEFI ベースの場合) を押して、選択したエントリーオプションを編集します。
3. オプションリストでカーネル行を探します。カーネル行は **linux** というキーワードで始まります。この行で、**rhgb** を探して、削除します。
4. **F10** または **Ctrl+X** を押して、編集したオプションでシステムを起動します。

システムが正常に起動した場合は、通常通りにログインできます。ただし、グラフィカルブートを永続的に無効にしない場合は、システムが起動するたびにこの手順を実行する必要があります。

手順: グラフィカルブートを永続的に無効にする

1. システムの root アカウントにログインします。
2. grubby ツールを使用して、デフォルトの GRUB2 カーネルを検索します。

```
# grubby --default-kernel
/boot/vmlinuz-4.18.0-94.el8.x86_64
```

3. grubby ツールを使用して、GRUB2 設定のデフォルトカーネルから **rhgb** 起動オプションを削除します。以下に例を示します。

```
# grubby --remove-args="rhgb" --update-kernel /boot/vmlinuz-4.18.0-94.el8.x86_64
```

4. システムを再起動します。グラフィカル起動シーケンスが使用されなくなりました。グラフィカルな起動シーケンスを有効にする場合は、同じ手順に従って、**--remove-args="rhgb"** パラメーターを **--args="rhgb"** パラメーターに置き換えます。これにより、GRUB2 設定のデフォルトカーネルに **rhgb** ブートオプションが復元されます。

C.5. ログイン後に X サーバーが失敗する

X サーバーは、ローカルマシン、つまりユーザーが直接使用するコンピューターで実行する X Window System のプログラムです。X サーバーは、グラフィックカード、ディスプレイ画面、入力デバイス (通常はこれらのコンピューターのキーボードとマウス) へのすべてのアクセスを処理します。X Window System (しばしば X と呼ばれます) は、1 台コンピューターおよびコンピューターのネットワークで GUI を管理するための、完全なクロスプラットフォームの無料クライアントサーバーシステムです。クライアントサーバーモデルは、クライアントとサーバーと呼ばれる、リンクされている 2 つのアプリケーション間で作業を分割するアーキテクチャーです。*

ログイン後に X サーバーがクラッシュした場合は、1 つ以上のファイルシステムが満杯になっている可能性があります。問題をトラブルシューティングするには、次のコマンドを実行します。

```
$ df -h
```

この出力は、どのパーティションが満杯かを検証します。ほとんどの場合、問題は **/home** パーティションにあります。以下は、**df** コマンドの出力例です。

```
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  396M     0  396M   0%  /dev
tmpfs                     411M     0  411M   0%  /dev/shm
tmpfs                     411M  6.7M  405M   2%  /run
tmpfs                     411M     0  411M   0%  /sys/fs/cgroup
/dev/mapper/rhel-root          17G    4.1G  13G   25% /
/dev/sda1                 1014M  173M 842M  17% /boot
tmpfs                     83M    20K   83M   1%  /run/user/42
tmpfs                     83M    84K  83M    1%  /run/user/1000
/dev/dm-4                   90G    90G    0  100% /home
```

この例では、**/home** パーティションが満杯になっていることが失敗の原因になっていることがわかります。不要なファイルを削除します。ディスク領域の一部を解放したら、**startx** コマンドを使用して X を起動します。**df** に関する詳細情報と、使用できるオプション (この例で使用する **-h** オプションなど) の

詳細は、**df(1)** の man ページを参照してください。

*ソース - http://www.linfo.org/x_server.html

C.6. RAM が認識されない

シナリオによっては、カーネルがすべてのメモリー (RAM) を認識しないため、システムが使用するメモリーが、インストールされているメモリーより少なくなる場合があります。システムが報告するメモリーの合計サイズが期待値と一致しない場合は、少なくとも 1 つのメモリーモジュールに問題がある可能性があります。BIOS ベースのシステムでは、**Memtest86+** ユーティリティーを使用して、システムのメモリーをテストできます。

ハードウェアの設定によっては、システムの RAM の一部が予約されているため、システムが使用できなくなります。統合グラフィックスカードが搭載されている一部のラップトップコンピューターは、GPU 用のメモリーの一部を予約します。たとえば、4 GiB の RAM および統合 Intel グラフィックスカードを搭載したラップトップでは、約 3.7 GiB の使用可能なメモリーが表示されます。さらに、多くの Red Hat Enterprise Linux システムでデフォルトで有効になっている **kdump** クラッシュカーネルダンプメカニズムは、プライマリーカーネルに障害が発生した場合に使用されるセカンダリーカーネル用にメモリーの一部を予約します。この予約メモリーは、利用可能としては表示されません。

以下の手順を使用して、メモリー量を手動で設定します。

手順

1. システムが現在報告しているメモリー容量を MiB 単位で確認します。

```
$ free -m
```

2. システムを起動し、ブートローダーメニューが表示されるまで待ちます。
起動タイムアウト期間が **0** に設定されている場合は、**Esc** キーを押してメニューにアクセスします。

3. ブートローダーメニューからカーソルキーを使用して、起動するエントリーを強調表示し、**Tab** キー (BIOS ベースのシステムの場合)、または **e** キー (UEFI ベースのシステムの場合) を押して、選択したエントリーオプションを編集します。

4. オプション一覧でカーネル行を探します。カーネル行は **linux** というキーワードで始まります。以下のオプションをこの行の最後に追加します。

```
mem=xxM
```

5. **xx** の部分は実際の容量を MiB 単位で入力してください。

6. **F10** キーまたは **Ctrl**+**X** の組み合わせを押して、編集を行ったオプションでシステムを起動します。

7. システムが起動するのを待ってログインし、コマンドラインを開きます。

8. システムが報告するメモリー量を MiB 単位で確認します。

```
$ free -m
```

9. コマンドで表示される RAM の合計サイズが期待値と一致する場合は、変更を永続化します。

```
# grubby --update-kernel=ALL --args="mem=xxM"
```

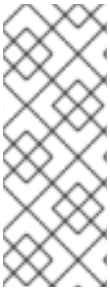
C.7. シグナル 11 エラーが表示される

一般的にセグメンテーションフォールトとして知られるシグナル 11 エラーは、割り当てられていないメモリー位置にプログラムがアクセスしたことを意味します。シグナル 11 エラーは、インストールされているソフトウェアプログラムのバグ、または障害のあるハードウェアが原因で発生する可能性があります。インストールプロセスでシグナル 11 エラーが表示された場合は、最新のインストールイメージを使用していることを確認し、インストールプログラムで、イメージが破損していないことを確認するように求めます。

詳細は、起動用メディアの検証 を参照してください。

インストールメディアの不良 (書き込みが正しく行われていなかったり、光ディスクに傷がついているなど) は、シグナル 11 エラーの一般的な原因です。インストールを行う前には、必ずインストールメディアの整合性を確認することが推奨されます。最新のインストールメディアを取得する方法は、インストール用 ISO イメージのダウンロード を参照してください。

インストールを開始する前にメディアチェックを実行するには、起動メニューに **rd.live.check** 起動オプションを追加します。エラーなしでメディアチェックを実行しても、セグメンテーションフォールトで問題が引き続き発生する場合は、通常、システムでハードウェアエラーが発生したことを示しています。このシナリオでは、問題はおそらくシステムのメモリー (RAM) にあります。これは、以前に同じコンピューターで別のオペレーティングシステムをエラーなしで使用した場合でも、問題になる可能性があります。



注記

AMD、Intel 64 ビット、および 64 ビット ARM アーキテクチャーの場合: BIOS ベースのシステムであれば、インストールメディアに含まれている **Memtest86+** メモリーテストモジュールを使用してシステムメモリー全体のテストを行うことができます。

詳細は、Memtest86 アプリケーションの使用によるメモリー障害の検出 を参照してください。

これ以外に考えられる原因は、本書では扱いません。ハードウェアの製造元のドキュメントと、Red Hat Hardware Compatibility List (https://access.redhat.com/ecosystem/search/#/category/Server) を確認してください。

C.8. ネットワークストレージ領域から IPL できない

注記

- この問題は、IBM Power Systems の問題です。
- PowerNV システムでは、**PReP** Boot パーティションは必要ありません。

ネットワークストレージ領域 (*NWSSTG) から IPL を実行する際に問題が発生する場合は、おそらく PReP パーティションがないことが原因です。このシナリオでは、システムを再インストールし、パーティション作成フェーズまたはキックスタートファイルでこのパーティションを作成する必要があります。

C.9. XDMCP の使用

X Window System をインストールし、グラフィカルログインマネージャーを使用して Red Hat

Enterprise Linux システムにログインするシナリオがあります。以下の手順に従って、X Display Manager Control Protocol (XDMCP) を有効にし、ネットワークに接続したワークステーションや X11 端末など、X 互換のクライアントからデスクトップ環境にリモートでログインします。

注記

XDMCP は、Wayland プロトコルでは対応していません。

手順

1. **vi** や **nano** などの平文エディターで **/etc/gdm/custom.conf** 設定ファイルを開きます。
2. **custom.conf** ファイルで、**[xdmcp]** で始まるセクションを見つけます。このセクションに、以下の行を追加します。

```
Enable=true
```

3. XDMCP を使用している場合は、**WaylandEnable=false** が **/etc/gdm/custom.conf** ファイルに存在することを確認してください。
4. ファイルを保存し、テキストエディターを編集します。
5. X Window System を再起動します。これには、システムを再起動するか、root で次のコマンドを実行して GNOME Display Manager を再起動します。

```
# systemctl restart gdm.service
```

6. ログインプロンプトを待ち、ユーザー名とパスワードを使用してログインします。X Window System が XDMCP 用に設定されました。クライアントワークステーションで X コマンドを使用して、リモート X セッションを開始し、別のワークステーション (クライアント) から接続できます。以下に例を示します。

```
$ X :1 -query address
```

7. **address** を、リモート X11 サーバーのホスト名に置き換えます。このコマンドは、XDMCP を使用してリモートの X11 サーバーに接続し、X11 サーバーシステムのディスプレイ :1 でリモートグラフィカルログイン画面を表示します (通常は **Ctrl-Alt-F8** を押してアクセスできます)。nested X11 サーバーを使用してリモートデスクトップセッションにアクセスすることもできます。これにより、リモートデスクトップが現在の X11 セッションの画面として開きます。Xnest を使用して、ローカルの X11 セッションでネストされたリモートデスクトップを開くことができます。たとえば、以下のコマンドを使用して Xnest を実行します。address を、リモート X11 サーバーのホスト名に置き換えます。

```
$ Xnest :1 -query address
```

関連情報

- X Window System のドキュメント

C.10. レスキューモードの使用

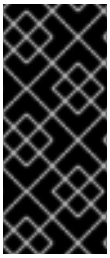
インストールプログラムのレスキューモードは、Red Hat Enterprise Linux DVD またはその他の起動メディアから起動できる最小限の Linux 環境です。さまざまな問題を修復するコマンドラインユーティリ

ティーが含まれています。レスキューモードは、起動メニューの Troubleshooting メニューからアクセスできます。このモードでは、ファイルシステムを読み取り専用としてマウントしたり、拒否リストに登録したり、ドライバーディスクで提供されるドライバーを追加したり、システムパッケージをインストールまたはアップグレードしたり、パーティションを管理したりできます。

注記

インストールプログラムのレスキューモードは、**systemd** システムおよびサービスマネージャーの一部として提供されるレスキューモード (シングルユーザーモードに相当) および緊急モードとは異なります。

レスキューモードで起動するには、最小起動ディスク、USB ドライブ、フルインストール DVD など、Red Hat Enterprise Linux の起動用メディアを使用してシステムを起動できる必要があります。



重要

iSCSI デバイスや zFCP デバイスなどの高度なストレージは、**rd.zfcp=**、**root=iscsi: オプション** などの **dracut** 起動オプションを使用するか、64 ビットの IBM Z 上の CMS 設定ファイルで設定する必要があります。レスキューモードで起動した後は、ストレージデバイスを対話的に設定できません。**dracut** 起動オプションの詳細は、**dracut.cmdline(7)** の man ページを参照してください。

C.10.1. レスキューモードでシステムの起動

この手順では、レスキューモードで起動する方法を説明します。

手順

1. 最小限の起動用メディア、フルインストールの DVD、または USB ドライブからシステムを起動し、起動メニューが表示されるまで待ちます。
2. 起動メニューから **Troubleshooting** > **Rescue a Red Hat Enterprise Linux system**オプションを選択するか、**inst.rescue** オプションを起動コマンドラインに追加します。起動コマンドラインに入るには、**Tab** キー (BIOS ベースのシステムの場合) を押すか、**e** キー (UEFI ベースのシステムの場合) を押します。
3. オプション:起動するドライバーディスクで提供されるサードパーティーのドライバーが必要な場合は、**inst.dd=driver_name** を起動コマンドラインに追加します。

```
inst.rescue inst.dd=driver_name
```

4. オプション:Red Hat Enterprise Linux ディストリビューションに含まれるドライバーが原因でシステムが起動しない場合は、**modprobe.blacklist=** オプションを起動コマンドラインに追加します。

```
inst.rescue modprobe.blacklist=driver_name
```

5. **Enter** (BIOS ベースのシステムの場合) または **Ctrl**+**X** (UEFI ベースのシステムの場合) を押して、変更したオプションを起動します。次のメッセージが表示されるまで待ちます。

```
The rescue environment will now attempt to find your Linux installation and mount it under the directory: /mnt/sysroot/. You can then make any changes required to your system. Choose 1 to proceed with this step. You can choose to mount your file systems read-only instead of read-write by choosing 2. If for some reason this process does not work choose 3 to skip
```

```
directly to a shell.

1) Continue
2) Read-only mount
3) Skip to shell
4) Quit (Reboot)
```

1 を選択すると、インストールプログラムは **/mnt/sysroot/** ディレクトリーにファイルシステムをマウントしようとします。パーティションのマウントに失敗すると通知されます。**2** を選択すると、ファイルシステムを **/mnt/sysroot/** ディレクトリーにマウントしようとしますが、読み取り専用モードになります。**3** を選択すると、ファイルシステムはマウントされません。

システムルートの場合には、インストーラーは **/mnt/sysimage** と **/mnt/sysroot** の 2 つのマウントポイントをサポートします。**/mnt/sysroot** パスは、ターゲットシステムの / をマウントするために使用されます。通常、物理ルートとシステムの root は同じであるため、**/mnt/sysroot** は **/mnt/sysimage** と同じファイルシステムに割り当てられます。唯一の例外は、デプロイメントに基づいてシステムの root が変更する rpm-ostree システムのみです。次に、**/mnt/sysroot** は、**/mnt/sysimage** のサブディレクトリーに割り当てられます。chroot には **/mnt/sysroot** を使用することを推奨します。

6. 続行するには **1** を選択します。システムがレスキューモードになると、VC (仮想コンソール) 1 および VC 2 にプロンプトが表示されます。**Ctrl+Alt+F1** キーの組み合わせで VC 1 にアクセスし、**Ctrl+Alt+F2** で VC 2 にアクセスします。

```
sh-4.2#
```

7. ファイルシステムがマウントされていても、レスキューモードではデフォルトの root パーティションは一時的な root パーティションであり、通常のユーザーモード (**multi-user.target** または **graphical.target**) で使用するファイルシステムの root パーティションではありません。ファイルシステムのマウントを選択し、正常にマウントされた場合は、次のコマンドを実行してレスキューモード環境の root パーティションを、ファイルシステムの root パーティションに変更できます。

```
sh-4.2# chroot /mnt/sysroot
```

これは、root パーティションが / としてマウントされることが求められる **rpm** などのコマンドを実行する必要がある場合に便利です。chroot 環境を終了するには、**exit** と入力してプロンプトに戻ります。

8. **3** を選択した場合でも、**/directory/** などのディレクトリーを作成し、次のコマンドを入力すると、レスキューモード内でパーティションまたは LVM2 論理ボリュームを手動でマウントできます。

```
sh-4.2# mount -t xfs /dev/mapper/VolGroup00-LogVol02 /directory
```

上記のコマンドでは、**/directory/** は作成したディレクトリーで、**/dev/mapper/VolGroup00-LogVol02** はマウントする LVM2 論理ボリュームになります。パーティションのタイプが XFS 以外の場合は、文字列 xfs を正しい種類 (ext4 など) に置き換えます。

9. すべての物理パーティションの名前が不明な場合は、次のコマンドを実行するとリストが表示されます。

```
sh-4.2# fdisk -l
```

LVM2 物理ボリューム、ボリュームグループ、または論理ボリュームの名前がすべて不明な場合は、**pvdisplay** コマンド、**vgdisplay** コマンド、または **lvdisplay** コマンドを使用します。

C.10.2. レスキューモードでの SOS レポートの使用

sosreport コマンドラインユーティリティーは、実行中のカーネルバージョン、読み込み済みモジュール、システムおよびサービスの設定ファイルなどの設定および診断情報をシステムから収集します。このユーティリティーの出力は、**/var/tmp/** ディレクトリーの tar アーカイブに保存されます。 **sosreport** ユーティリティーは、システムエラーの分析とトラブルシューティングに役立ちます。この手順に従って、レスキューモードで **sosreport** 出力を取得します。

前提条件

- レスキューモードでシステムを起動している。
- インストール済みのシステムの / **(root)** パーティションを読み書きモードでマウントしている。
- この問題を Red Hat サポートに連絡し、ケース番号を受け取っている。

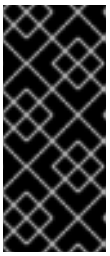
手順

1. root ディレクトリーを **/mnt/sysroot/** ディレクトリーに変更します。

```
sh-4.2# chroot /mnt/sysroot/
```

2. **sosreport** を実行して、システム設定と診断情報を含むアーカイブを生成します。

```
sh-4.2# sosreport
```



重要

sosreport は、Red Hat サポートから受け取った名前とケース番号の入力を求めるプロンプトが表示されます。次の文字またはスペースを追加するとレポートが使用できなくなる可能性があるため、英数字のみを使用してください。

 # % & { } \ < > > * ? / $ ~ ' " : @ + ` | =

3. オプション:ネットワークを使用して、生成されたアーカイブを新しい場所に転送する場合は、ネットワークインターフェイスを設定する必要があります。このシナリオでは、他の手順は必要ないため、動的 IP アドレス指定を使用します。ただし、静的アドレスを使用する場合は、次のコマンドを実行して、ネットワークインターフェイス (dev eth0 など) に IP アドレス (10.13.153.64/23 など) を割り当てます。

```
bash-4.2# ip addr add 10.13.153.64/23 dev eth0
```

4. chroot 環境を終了します。

```
sh-4.2# exit
```

5. 生成されたアーカイブを新しい場所に保存し、その場所からアーカイブへのアクセスを容易にします。

```
sh-4.2# cp /mnt/sysroot/var/tmp/sosreport new_location
```

6. ネットワークを介したアーカイブの転送は、**scp** ユーティリティーを使用します。

```
sh-4.2# scp /mnt/sysroot/var/tmp/sosreport username@hostname:sosreport
```

関連情報

- Red Hat Enterprise Linux 上での sosreport の役割と生成方法
- How to generate sosreport from the rescue environment
- How do I make sosreport write to an alternative location?
- Sosreport fails.What data should I provide in its place?

C.10.3. GRUB2 ブートローダーの再インストール

シナリオによっては、GRUB2 ブートローダーが誤って削除または破損されたり、他のオペレーティングシステムによって置き換えられることがあります。この手順に従って、BIOS を使用する AMD64 システムおよび Intel 64 システムのマスターブートレコード (MBR) に GRUB2 を再インストールします。または、Open Firmware を使用する IBM Power Systems のリトルエンディアンバリアントに GRUB2 を再インストールします。

前提条件

- レスキューモードでシステムを起動している。
- インストール済みのシステムの / **(root)** パーティションを読み書きモードでマウントしている。
- **/boot** マウントポイントを読み取り/書き込みモードでマウントしている。

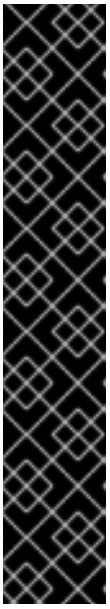
手順

1. root パーティションを変更します。

```
sh-4.2# chroot /mnt/sysroot/
```

2. **install_device** ブロックデバイスがインストールされている GRUB2 ブートローダーを再インストールします。

```
sh-4.2# /sbin/grub2-install install_device
```



重要

grub2-install コマンドを実行すると、以下の条件がすべて適用されると、マシンが起動できなくなる可能性があります。

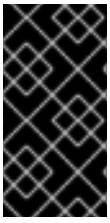
- システムは、EFI (Extensible Firmware Interface) を使用する AMD64 または Intel 64 です。
- Secure Boot が有効になります。

grub2-install コマンドを実行すると、EFI (Extendsible Firmware Interface) および Secure Boot が有効な AMD64 システムまたは Intel 64 システムを起動することはできません。この問題は、**grub2-install** コマンドが shim アプリケーションを使用する代わりに直接起動する署名されていない GRUB2 イメージをインストールするために発生します。システムが起動すると shim アプリケーションはイメージの署名を検証します。見つからない場合は、システムの起動に失敗します。

3. システムを再起動します。

C.10.4. RPM を使用してドライバーを追加または削除

ドライバーが見つからないか、誤作動すると、システムの起動時に問題が発生します。レスキューモードは、システムが起動に失敗してもドライバーを追加または削除できる環境を提供します。可能な場合は、RPM パッケージマネージャーを使用して、誤作動するドライバーを削除するか、更新されたドライバーや不足しているドライバーを追加することが推奨されます。以下の手順に従って、ドライバーを追加または削除します。



重要

ドライバーディスクからドライバーをインストールすると、ドライバーディスクは、このドライバーを使用するシステムにある **initramfs** イメージをすべて更新します。ドライバーが原因でシステムが起動できない場合は、別の **initramfs** イメージからシステムを起動する方法は使用できません。

C.10.4.1. RPM を使用したドライバーの追加

以下の手順に従ってドライバーを追加します。

前提条件

- レスキューモードでシステムを起動している。
- インストール済みのシステムを読み書きモードでマウントしている。

手順

1. そのドライバーを含む RPM パッケージを利用できるようにします。たとえば、CD または USB フラッシュドライブをマウントして、RPM パッケージを **/mnt/sysroot/** 配下の任意の場所 (例: **/mnt/sysroot/root/drivers/**) にコピーします。
2. root ディレクトリーを **/mnt/sysroot/** に変更します。

```
sh-4.2# chroot /mnt/sysroot/
```

3. **rpm -ivh** コマンドを使用して、ドライバーパッケージをインストールします。たとえば、以下のコマンドを実行して、**xorg-x11-drv-wacom** ドライバーパッケージを **/root/drivers/** からインストールします。

```
sh-4.2# rpm -ivh /root/drivers/xorg-x11-drv-wacom-0.23.0-6.el7.x86_64.rpm
```

注記

この chroot 環境の **/root/drivers/** ディレクトリーは、元のレスキュー環境の **/mnt/sysroot/root/drivers/** ディレクトリーです。

4. chroot 環境を終了します。

```
sh-4.2# exit
```

C.10.4.2. RPM を使用したドライバーの削除

以下の手順に従ってドライバーを削除します。

前提条件

- レスキューモードでシステムを起動している。
- インストール済みのシステムを読み書きモードでマウントしている。

手順

1. root ディレクトリーを **/mnt/sysroot/** ディレクトリーに変更します。

```
sh-4.2# chroot /mnt/sysroot/
```

2. **rpm -e** コマンドを使用して、ドライバーパッケージを削除します。たとえば、**xorg-x11-drv-wacom** ドライバーパッケージを削除するには、次のコマンドを実行します。

```
sh-4.2# rpm -e xorg-x11-drv-wacom
```

3. chroot 環境を終了します。

```
sh-4.2# exit
```

誤動作のあるドライバーを何らかの理由で削除できない場合は、代わりにドライバーを拒否リストに登録することで、起動時に読み込まれないようにすることができます。

4. ドライバーの追加および削除が終了したら、システムを再起動します。

C.11. ip= 起動オプションがエラーを返す

ip= 起動オプションの形式 **ip=[ip address]** (**ip=192.168.1.1** など) を使用すると、エラーメッセージ **Fatal for argument 'ip=[insert ip here]'\n sorry, unknown value [ip address] refusing to continue** が返ります。

Red Hat Enterprise Linux の以前のリリースにおける起動オプションの形式は次のようになります。

-

```
ip=192.168.1.15 netmask=255.255.255.0 gateway=192.168.1.254 nameserver=192.168.1.250
hostname=myhost1
```

ただし、Red Hat Enterprise Linux 8 では、起動オプションの形式は次のようになります。

```
ip=192.168.1.15::192.168.1.254:255.255.255.0:myhost1::none: nameserver=192.168.1.250
```

この問題を解決するには、**ip=ip::gateway:netmask:hostname:interface:none** の形式を使用します。ここでは、以下のようになります。

- **ip** はクライアントの IP アドレスを指定します。IPv6 アドレスは角括弧で囲んで指定できます (**[2001:DB8::1]** など)。
- **gateway** はデフォルトのゲートウェイです。IPv6 アドレスも使用できます。
- **netmask** は使用するネットマスクです。完全ネットマスク (255.255.255.0 など) または接頭辞 (**64** など) を使用できます。
- **hostname** はクライアントシステムのホスト名です。このパラメーターは任意です。

関連情報

- ネットワーク起動オプション

C.12. iLO デバイスまたは iDRAC デバイスにおいてグラフィカルインストールで起動できない

iLO デバイスまたは iDRAC デバイスでのリモート ISO インストールのグラフィカルインストーラーは、インターネット接続が遅いために利用できないことがあります。この場合、インストールを続行するには、以下のいずれかの方法を選択できます。

1. タイムアウトを避ける。そのためには、以下を実施します。
   a. インストールメディアから起動する際に、BIOS を使用している場合は **Tab** キーを、UEFI を使用している場合は **e** キーを押します。これにより、カーネルコマンドライン引数を変更できます。
   b. インストールを続行するには、**rd.live.ram=1** を追加し、BIOS を使用している場合は **Enter** を、UEFI を使用している場合は **Ctrl+x** を押します。
   インストールプログラムを読み込むのに時間がかかる場合があります。
2. グラフィカルインストーラーのロード時間を延長する別のオプションは、**inst.xtimeout** カーネル引数を秒単位で設定することです。

   ```
   inst.xtimeout=N
   ```

3. システムをテキストモードでインストールできます。詳細は Installing RHEL8 in text mode を参照してください。
4. ローカルメディアソースではなく、iLO や iDRAC などのリモート管理コンソールで、Red Hat カスタマーポータルの Download center にあるインストール ISO ファイルへの直接 URL を使用します。このセクションにアクセスするには、ログインしている必要があります。

C.13. Rootfs イメージは initramfs ではありません

インストーラーの起動中にコンソールに次のメッセージが表示される場合は、インストーラーの **initrd.img** の転送でエラーが発生した可能性があります。

```
[ ...] rootfs image is not initramfs
```

この問題を解決するには、**initrd** を再度ダウンロードするか、**initrd.img** を使用して **sha256sum** を実行し、インストールメディアの **.treeinfo** ファイルに保存されているチェックサムと比較します。

```
$ sha256sum dvd/images/pxeboot/initrd.img
fdb1a70321c06e25a1ed6bf3d8779371b768d5972078eb72b2c78c925067b5d8
dvd/images/pxeboot/initrd.img
```

.treeinfo でチェックサムを表示するには、以下を行います。

```
$ grep sha256 dvd/.treeinfo
images/efiboot.img =
sha256:d357d5063b96226d643c41c9025529554a422acb43a4394e4ebcaa779cc7a917
images/install.img =
sha256:8c0323572f7fc04e34dd81c97d008a2ddfc2cfc525aef8c31459e21bf3397514
images/pxeboot/initrd.img =
sha256:fdb1a70321c06e25a1ed6bf3d8779371b768d5972078eb72b2c78c925067b5d8
images/pxeboot/vmlinuz =
sha256:b9510ea4212220e85351cbb7f2ebc2b1b0804a6d40ccb93307c165e16d1095db
```

正しい **initrd.img** があるにもかかわらず、インストーラーの起動中に次のカーネルメッセージが表示される場合は、多くの場合、ブートパラメーターが欠落しているか、スペルが間違っており、インストーラーは、通常インメモリー root ファイルシステムの完全なインストーラーの初期ラムディスクを提供する **inst.repo=** パラメーターによって参照される **stage2** をロードできませんでした。

```
[ ...] No filesystem could mount root, tried:
[ ...] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
[ ...] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.14.0-55.el9.s390x #1
[ ...] ...
[ ...] Call Trace:
[ ...] ([<...>] show_trace+0x.../0x...)
[ ...]  [<...>] show_stack+0x.../0x...
[ ...]  [<...>] panic+0x.../0x...
[ ...]  [<...>] mount_block_root+0x.../0x...
[ ...]  [<...>] prepare_namespace+0x.../0x...
[ ...]  [<...>] kernel_init_freeable+0x.../0x...
[ ...]  [<...>] kernel_init+0x.../0x...
[ ...]  [<...>] kernel_thread_starter+0x.../0x...
[ ...]  [<...>] kernel_thread_starter+0x.../0x…
```

この問題を解決するには、次を確認してください

- 指定されたインストールソースがカーネルコマンドライン (**inst.repo=**) またはキックスタートファイルで正しい場合
- ネットワーク設定はカーネルコマンドラインで指定される (インストールソースがネットワークとして指定されている場合)
- ネットワークインストールソースが別のシステムからアクセス可能

付録D システム要件の参照

このセクションでは、Red Hat Enterprise Linux をインストールする際のハードウェア、インストール先、システム、メモリー、RAID に関する情報とガイドラインを提供します。

D.1. ハードウェアの互換性

Red Hat は、対応しているハードウェアについて、ハードウェアベンダーと密接に連携しています。

- ハードウェアがサポートされていることを確認する場合は、Red Hat Hardware Compatibility List(https://access.redhat.com/ecosystem/search/#/category/Server) を参照してください。
- サポートされているメモリーサイズまたは CPU 数を確認する場合は、Red Hat Enterprise Linux テクノロジーの機能と制限 で詳細を参照してください。

D.2. インストール先として対応しているターゲット

インストールターゲットは、Red Hat Enterprise Linux を格納し、システムを起動するストレージデバイスです。Red Hat Enterprise Linux は、AMD64、Intel 64、および 64 ビット ARM のシステム向けに、以下のインストールターゲットに対応しています。

- SCSI、SATA、SAS などの標準的な内部インターフェイスで接続するストレージ
- BIOS/ファームウェアの RAID デバイス
- nd_pmem ドライバーが対応する、セクターモードに設定された Intel 64 および AMD64 アーキテクチャーの NVDIMM デバイス
- ファイバーチャネルのホストバスアダプターおよびマルチパスのデバイス。製造元が提供しているドライバーが必要な場合があります。
- Xen 仮想マシンの Intel のプロセッサーの Xen ブロックデバイス
- KVM 仮想マシンの Intel のプロセッサーの VirtIO ブロックデバイス

Red Hat では、USB ドライブや SD メモリーカードへのインストールはサポートしていません。サードパーティーによる仮想化技術のサポートは、Red Hat Hardware Compatibility List を参照してください。

D.3. システムの仕様

Red Hat Enterprise Linux インストールプログラムはシステムのハードウェアを自動的に検出してインストールするため、特定のシステム情報を提供する必要はありません。ただし、特定の Red Hat Enterprise Linux インストールシナリオでは、将来の参照用にシステム仕様を記録しておくことを推奨します。次のようなシナリオになります。

カスタマイズしたパーティションレイアウトで RHEL をインストール

レコード:システムに接続されているハードドライブのモデル番号、サイズ、種類、およびインタフェース。たとえば、SATA0 上には Seagate 製 ST3320613AS (320 GB)、SATA1 上には Western Digital WD7500AAKS (750 GB) です。

既存のシステムに、追加のオペレーティングシステムとして RHEL をインストール

レコード:システムで使用するパーティション。この情報には、ファイルシステムの種類、デバイスノード名、ファイルシステムのラベル、およびサイズを記載でき、パーティションを作成する際に特定の

パーティションを識別できます。オペレーティングシステムの 1 つが Unix オペレーティングシステムの場合、Red Hat Enterprise Linux はデバイス名を異なる方法で報告することがあります。追加の情報は、**mount** コマンド、**blkid** コマンドを実行して表示するか、**/etc/fstab** ファイルを参照してください。

複数のオペレーティングシステムがインストールされている場合、Red Hat Enterprise Linux インストールプログラムはそのオペレーティングシステムを自動的に検出して、それを起動するようにブートローダーを設定しようとします。追加のオペレーティングシステムが自動的に検出されない場合は、手動で設定できます。

詳細については、ソフトウェア設定の設定 の **ブートローダーの設定** を参照してください。

ローカルのハードドライブにあるイメージからの RHEL インストール

レコード:イメージを保存するハードドライブおよびディレクトリー

ネットワーク経由で RHEL のインストール

ネットワークを手動で設定する必要がある場合、つまり DHCP を使用しない場合です。

レコード:

- IP アドレス
- ネットマスク
- ゲートウェイの IP アドレス
- (必要に応じて) サーバーの IP アドレス

ネットワーク要件が不明な場合は、ネットワーク管理者に連絡してください。

iSCSI ターゲットへの RHEL のインストール

レコード:iSCSI ターゲットの場所ネットワークに応じて、CHAP ユーザー名とパスワードと、リバースの CHAP ユーザー名とパスワードが必要になる場合があります。

ドメインに含まれるシステムへの RHEL のインストール

ドメイン名が DHCP サーバーにより提供されることを確認してください。提供されない場合は、インストール中にドメイン名を入力する必要があります。

D.4. ディスクおよびメモリーの要件

複数のオペレーティングシステムがインストールされている場合は、割り当てられたディスク領域が Red Hat Enterprise Linux で必要なディスク領域とは異なることを確認することが重要です。

注記

- AMD64、Intel 64、64 ビット ARM の場合は、少なくとも 2 つのパーティション (/ および **swap**) を Red Hat Enterprise Linux 専用とする必要があります。
- IBM Power Systems サーバーでは、少なくとも 3 つのパーティション (/、**swap**、**PReP** 起動パーティション) を Red Hat Enterprise Linux 専用にする必要があります。
- PowerNV システムでは、**Prep** Boot パーティションは必要ありません。

最低 10 GiB の空きディスク容量が必要です。Red Hat Enterprise Linux をインストールするには、パーティションが分割されていないディスク領域か、削除できるパーティション内に、最低 10 GiB の容量が必要です。

パーティション設定の参照 を参照してください。

表D.1 最小 RAM 要件

インストールタイプ	推奨される最小 RAM
ローカルメディアによるインストール (USB、DVD)	• aarch64、s390x、および x86_64 アーキテクチャー: 1.5 GiB • ppc64le アーキテクチャー: 3 GiB
NFS ネットワークインストール	• aarch64、s390x、および x86_64 アーキテクチャー: 1.5 GiB • ppc64le アーキテクチャー: 3 GiB
HTTP、HTTPS、または FTP ネットワークインストール	• s390x および x86_64 アーキテクチャー: 3 GiB • aarch64 および ppc64le アーキテクチャー: 4 GiB

注記

推奨される最小要件より少ないメモリーでインストールを完了できます。正確な要件は、環境とインストールパスにより異なります。ご使用の環境に必要な最小 RAM を決定するために、さまざまな設定をテストすることが推奨されます。キックスタートファイルを使用して Red Hat Enterprise Linux をインストールするには、標準的なインストールと同じように推奨される最小 RAM 要件があります。ただし、キックスタートファイルに追加のメモリーを必要とするコマンド、または RAM ディスクにデータを書き込むコマンドが含まれている場合は、追加の RAM が必要になることがあります。詳細は、**高度な RHEL 8 インストールの実行** を参照してください。

D.5. UEFI セキュアブートおよびベータ版の要件

UEFI セキュアブートが有効になっているシステムに Red Hat Enterprise Linux のベータ版リリースをインストールする予定がある場合は、UEFI セキュアブートオプションを無効にしてから、インストールを開始します。

UEFI セキュアブートでは、オペレーティングシステムのカーネルが、対応する公開鍵を使用してシステムのファームウェアが検証できる、認識済みの秘密鍵で署名されている必要があります。Red Hat Enterprise Linux ベータ版リリースの場合には、カーネルは Red Hat ベータ版固有の公開鍵で署名されていますが、この鍵はデフォルトではシステムで認識できません。その結果、インストールメディアの起動にも失敗します。

付録E パーティション設定の参照

E.1. 対応デバイスの種類

標準パーティション

標準パーティションには、ファイルシステムまたは swap 領域を使用できます。標準パーティションは、**/boot**、**BIOS Boot**、および **EFI System パーティション** で最も一般的に使用されます。その他のほとんどの用途には、LVM 論理ボリュームが推奨されます。

LVM

デバイスタイプに **LVM** (または論理ボリューム管理) を選択すると、LVM 論理ボリュームが作成されます。LVM は、物理ディスクを使用する際にパフォーマンスを向上できます。また、パフォーマンスや信頼性、またはその両方を向上させるために、高度な設定 (1 つのマウントポイントに複数の物理ディスクの使用、ソフトウェア RAID の設定など) が可能になります。

LVM シンプロビジョニング

シンプロビジョニングを使用すると、シンプールと呼ばれる、空き領域のストレージプールを管理でき、アプリケーションで必要になった時に任意の数のデバイスに割り当てることができます。ストレージ領域の割り当ての費用対効果を高くする必要がある場合は、プールを動的に拡張できます。

警告

インストールプログラムは、オーバープロビジョニングの LVM シンプールをサポートしていません。

E.2. 対応ファイルシステム

このセクションでは、Red Hat Enterprise Linux で利用可能なファイルシステムを説明します。

xfs

XFS は、最大 16 エクサバイト (約 1600 万テラバイト) のファイルシステム、最大 8 エクサバイト (約 800 万テラバイト) のファイル、および数千万のエントリーを含むディレクトリー構造に対応する、スケーラビリティーが高く高性能なファイルシステムです。**XFS** は、メタデータジャーナリングもサポートしているため、クラッシュに対するより迅速な復元が容易になります。1 つの XFS ファイルシステムで対応している最大サイズは 500 TB です。**XFS** は、Red Hat Enterprise Linux でデフォルトの、推奨されるファイルシステムです。XFS ファイルシステムは縮小して空き領域を確保することはできません。

ext4

ext4 ファイルシステムは、**ext3** ファイルシステムをベースとし、改善が加えられています。より大きなファイルシステム、そしてより大きなファイルに対応するようになり、ディスク領域の割り当てに要する時間が短縮され効率化されています。また、ディレクトリー内のサブディレクトリーの数に制限がなく、ファイルシステムチェックが速くなり、ジャーナリングがより強力になりました。1 つの **ext4** ファイルシステムで対応している最大サイズは 50 TB です。

ext3

ext3 ファイルシステムは **ext2** ファイルシステムをベースとし、ジャーナリング機能という大きな利点を備えています。ジャーナリングファイルシステムを使用すると、クラッシュが発生するたびに fsck ユーティリティーを実行してメタデータの整合性をチェックする必要がないため、突然終了し

たあとに、ファイルシステムの復元に要する時間を短縮できます。

ext2

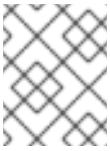
ext2 ファイルシステムは標準の Unix ファイルタイプに対応しています (通常のファイル、ディレクトリー、シンボリックリンクなど)。最大 255 文字までの長いファイル名を割り当てることができます。

swap

swap パーティションは、仮想メモリーに対応するために使用されます。つまり、システムが処理しているデータを格納する RAM が不足すると、そのデータが swap パーティションに書き込まれます。

vfat

VFAT ファイルシステムは Linux ファイルシステムです。FAT ファイルシステムにある Microsoft Windows の長いファイル名と互換性があります。



注記

Linux システムパーティションでは、**VFAT** ファイルシステムのサポートは利用できません。たとえば、/、**/var**、**/usr** などです。

BIOS ブート

BIOS 互換モードで、BIOS システムおよび UEFI システムの GUID パーティションテーブル (GPT) を使用するデバイスから起動するのに必要な、非常に小さいパーティションです。

EFI システムパーティション

UEFI システムの GUID パーティションテーブル (GPT) でデバイスを起動する場合に必要な、小さいパーティションです。

PReP

この小さなブートパーティションは、ハードドライブの最初のパーティションにあります。**PReP** 起動パーティションには GRUB2 ブートローダーが含まれ、その他の IBM Power Systems サーバーが Red Hat Enterprise Linux を起動できるようにします。

注記

- PowerNV システムでは、**PReP** Boot パーティションは必要ありません。

E.3. 対応する RAID のタイプ

RAID は Redundant Array of Independent Disks の略で、複数の物理ディスクを論理ユニットにまとめることを可能にする技術です。設定によっては、信頼性を犠牲にしてパフォーマンスを向上させるように設計されていますが、一方で、利用可能な領域のサイズが同じでも、より多くのディスクを使用することで、信頼性が向上します。

このセクションは、LVM および LVM シンプロビジョニングを使用して、インストール済みシステムのストレージを設定できるソフトウェアの RAID タイプを説明します。

RAID 0

パフォーマンス:データを複数のディスクに分散させます。RAID 0 は、標準パーティションのパフォーマンスを向上させ、1 つの大規模仮想デバイスに複数のディスクのストレージをプールします。RAID 0 には冗長性がなく、アレイ内の 1 つのディスクに障害が発生すると、アレイ全体のデータが壊れる点に注意してください。RAID 0 には少なくとも 2 つのディスクが必要です。

RAID 1

冗長性:1 つのパーティションにあるデータをすべて別のディスク (複数可) にミラーリングします。アレイ内のデバイスを増やすことで冗長レベルを強化します。RAID 1 には少なくとも 2 つのディスクが必要です。

RAID 4

エラーの確認:データを複数のディスクに分散し、アレイ内の 1 つのディスクにパリティー情報を格納しているため、アレイ内のいずれかのディスクに障害が発生した場合にアレイを保護します。すべてのパリティー情報が 1 つのディスクに保存されているため、このディスクにアクセスすると、アレイのパフォーマンスにボトルネックが生じます。RAID 4 には少なくとも 3 つのディスクが必要です。

RAID 5

分散エラーの確認:データおよびパリティー情報を複数のディスクに分散させます。そのため、RAID 5 は複数ディスクにデータを分散させるためパフォーマンスが向上する一方、パリティー情報もアレイ全体に分散されるため、RAID 4 のようにパフォーマンスのボトルネックは共有されません。RAID 5 には少なくとも 3 つのディスクが必要です。

RAID 6

冗長なエラーの確認:RAID 6 は RAID 5 と似ていますが、格納するパリティーデータのセットは 2 つになります。RAID 6 には少なくとも 4 つのディスクが必要です。

RAID 10

パフォーマンスと冗長性:RAID 10 は、ネストまたはハイブリッド RAID です。ミラーリングしているディスクセットにデータを分散させることで構築します。たとえば、4 つの RAID パーティションで構築した RAID 10 のアレイは、ストライプ化されたパーティションをミラーリングする 2 組のペアで設定されます。RAID 10 には少なくとも 4 つのディスクが必要です。

E.4. 推奨されるパーティション設定スキーム

Red Hat は、以下のマウントポイントで、異なるファイルシステムを作成することを推奨します。ただし、必要に応じて **/usr**、**/var** および **/tmp** のマウントポイントでファイルシステムを作成することもできます。

- **/boot**
- / (ルート)
- **/home**
- **swap**
- **/boot/efi**
- **PReP**

このパーティションスキームは、ベアメタルのデプロイメントに推奨されますが、仮想およびクラウドのデプロイメントには適用されません。

/boot パーティション: 最小限 1 GiB のサイズを推奨しています

/boot にマウントするパーティションには、オペレーティングシステムのカーネルが含まれます。これにより、起動プロセス中に使用されるファイルと共に Red Hat Enterprise Linux 8 が起動します。大概のファームウェアには制限があるため、そのファームウェアを格納する小さいパーティションを作成することが推奨されます。ほとんどの場合は、1 GiB の boot パーティションで十分です。他のマウントポイントとは異なり、LVM ボリュームを **/boot** に使用することはできません。**/boot** は別個のディスクパーティションにある必要があります。

警告

通常、**/boot** パーティションはインストールプログラムで自動的に作成されます。ただし、/ (ルート) パーティションのサイズが 2 TiB を超え、また起動に (U)EFI を使用する場合は、マシンを正常に起動させるため 2 TiB 未満の **/boot** パーティションを別途に作成する必要があります。

注記

RAID カードを実装している場合、BIOS タイプは、RAID カードからの起動に対応していない場合がある点に注意してください。これに該当する場合は、**/boot** パーティションを別のハードドライブなどの RAID アレイ以外のパーティションに作成する必要があります。

/: 10 GiB のサイズを推奨しています

これは、/ (ルート) ディレクトリーを置く場所です。ルートディレクトリーは、ディレクトリー構造のトップレベルです。デフォルトでは、書き込み先のパスに別のファイルシステムがマウントされていない限り (**/boot**、**/home** など)、すべてのファイルがこのファイルシステムに書き込まれます。root ファイルシステムが 5 GiB の場合は最小インストールが可能ですが、パッケージグループをいくつでもインストールできるように、少なくとも 10 GiB を割り当てておくことが推奨されます。

重要

/ ディレクトリーと **/root** ディレクトリーを混同しないよう注意してください。**/root** ディレクトリーは root ユーザーのホームディレクトリーになります。**/root** ディレクトリーは、root ディレクトリーと区別するため、**スラッシュルート** ディレクトリーと呼ばれることがあります。

/home: 最小限 1 GiB のサイズを推奨しています

システムデータとユーザーデータを別々に格納する場合は、**/home** ディレクトリー用の専用ファイルシステムを作成します。ファイルシステムのサイズは、ローカルで保存するデータ量やユーザー数などを基に決定してください。こうすることで、ユーザーデータのファイルを消去せずに Red Hat Enterprise Linux 8 をアップグレードしたり、再インストールできるようになります。自動パーティション設定を選択する場合は、インストールに少なくとも 55GiB のディスク領域を確保して、**/home** ファイルシステムが作成されるようにすることが推奨されます。

swap パーティション: 最小限 1 GB のサイズを推奨しています

仮想メモリーは、swap ファイルシステムによりサポートされています。つまり、システムが処理しているデータを格納する RAM が不足すると、そのデータは swap ファイルシステムに書き込まれます。swap サイズはシステムメモリーのワークロードに依存するため、システムメモリーの合計ではありません。したがって、システムメモリーサイズの合計とは等しくなりません。システムメモリーの作業負荷を判断するためには、システムで実行するアプリケーションの種類、およびそのアプリケーションにより生じる負荷を分析することが重要になります。アプリケーションにより生じる負荷に関するガイダンスは、アプリケーション提供元または開発側より提供されます。
システムで swap 領域が不足すると、システムの RAM メモリーがすべて使用されるため、カーネルがプロセスを終了します。swap 領域が大き過ぎても、割り当てられているストレージデバイスがアイドル状態となり、リソース運用面では効率が悪くなります。また、swap 領域が大き過ぎるとメモリーリークに気付きにくくなる可能性があります。swap パーティションの最大サイズおよび詳細は、**mkswap(8)** の man ページを参照してください。

システムの RAM の容量別に推奨される swap サイズと、ハイバネートするのに十分なサイズを以下の表に示します。インストールプログラムでシステムのパーティション設定を自動的に設定すると、swap パーティションのサイズはこのガイドラインに沿って決められます。自動パーティション設定では、ハイバネートは使用しないことを前提としています。このため、swap パーティションの上限が、ハードドライブの合計サイズの最大 10% に制限され、インストールプログラムでは、1 TiB 以上のサイズの swap パーティションが作成されません。ハイバネートを行うために十分な swap 領域を設定したい場合、もしくはシステムのストレージ領域の 10% 以上を swap パーティションに設定したい場合、または 1 TiB を超えるサイズにしたい場合は、パーティション設定のレイアウトを手動で編集する必要があります。

表E.1 システムの推奨スワップ領域

システム内の RAM の容量	推奨されるスワップ領域	ハイバネートを許可する場合に推奨される swap 領域
2 GB 未満。	RAM 容量の 2 倍	RAM 容量の 3 倍
2 GiB - 8 GiB	RAM 容量と同じ	RAM 容量の 2 倍
8 GiB - 64 GiB	4 GiB から RAM 容量の半分まで	RAM 容量の 1.5 倍
64 GB 以上	ワークロードによる (最小 4GB)	ハイバネートは推奨されない

/boot/efi パーティション (サイズは 200 MiB を推奨)

UEFI ベースの AMD64、Intel 64、および 64 ビットの ARM は、200 MiB の EFI システムパーティションが必要です。推奨される最小サイズは 200 MiB で、デフォルトサイズは 600 MiB で、最大サイズは 600 MiB です。BIOS システムは、EFI システムパーティションを必要としません。

値が、範囲の境界線上にある場合 (システムの RAM が 2 GiB、8 GiB、または 64 GiB などの場合)、swap 領域の決定やハイバネートへのサポートは適宜判断してください。システムリソースに余裕がある場合は、swap 領域を増やすとパフォーマンスが向上することがあります。

swap 領域を複数のストレージデバイスに分散させても、swap 領域のパフォーマンスが向上します (高速ドライブやコントローラー、インターフェイスなどを備えたシステムで特に効果的)。

多くのシステムでは、パーティションおよびボリュームの数は上述の最小数より多くなります。パーティション設定は、システム固有のニーズに応じて決定してください。

注記

- すぐに必要となるパーティションにのみストレージ容量を割り当ててください。必要に応じて空き容量をいつでも割り当てることができます。
- パーティションを設定する方法が分からない場合は、インストールプログラムで提供されているデフォルトの自動パーティションのレイアウトをご利用ください。

PReP 起動パーティション (4 - 8 MiB のサイズを推奨)

IBM Power System サーバーに Red Hat Enterprise Linux をインストールする場合は、ハードドライブの最初のパーティションに **PReP** 起動パーティションが含まれている必要があります。これには、他の IBM Power Systems サーバーで Red Hat Enterprise Linux を起動できるようにする GRUB2

ブートローダーが含まれます。

注記

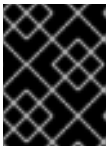
- PowerNV システムでは、**PReP** Boot パーティションは必要ありません。

E.5. パーティション設定に関するアドバイス

すべてのシステムに最善となる分割方法はありません。インストール済みシステムをどのように使用するかによって異なります。ただし、次のヒントは、ニーズに最適なレイアウトを見つけるのに役立つかもしれません。

- たとえば、特定のパーティションを特定のディスクに配置する必要がある場合など、特定の要件を満たすパーティションを最初に作成します。
- 機密データを格納する可能性があるパーティションやボリュームには、暗号化を検討してください。暗号化を行うと、権限を持たない人が物理ストレージデバイスにアクセスできても、暗号化したパーティションにあるデータにアクセスできなくなります。ほとんどの場合は、少なくともユーザーデータが含まれる **/home** パーティションを暗号化してください。
- 場合によっては、/、**/boot**、および **/home** 以外のディレクトリーに個別のマウントポイントを作成すると役に立つかもしれません。たとえば、**MySQL** データベースを実行するサーバーで、**/var/lib/mysql** 用のマウントポイントを別に持つことで、後でバックアップからデータベースを復元しなくても、再インストール中にデータベースを保存できます。ただし、不要なマウントポイントがあると、ストレージ管理がより困難になります。
- 特定のディレクトリーには、どのレイアウトに配置できるかについて、特別な制限がいくつか適用されます。特に、**/boot** ディレクトリーは常に、(LVM ボリュームではなく) 物理パーティションに存在する必要があります。
- Linux を初めて使用する場合は、さまざまなシステムディレクトリーとそのコンテンツの詳細を、 Linux ファイルシステム階層標準 を確認してください。
- 各カーネルには、おおよそ次のものが必要です。60MiB (initrd 34MiB、11MiB vmlinuz、および 5MiB System.map)
- レスキューモードの場合:100MiB (initrd 76MiB、11MiB vmlinuz、および 5MiB システムマップ)
- システムで **kdump** を有効にすると、さらに約 40 MiB(33 MiB の別の initrd) が必要になります。
  最も一般的なユースケースでは、**/boot** にはデフォルトの 1 GiB のパーティションサイズが必要です。ただし、複数のカーネルリリースまたはエラータカーネルを保持する予定がある場合は、このパーティションのサイズを増大させることが推奨されます。
- **/var** ディレクトリーには、Apache Web サーバーなど、多数のアプリケーションのコンテンツが格納されていて、YUM パッケージマネージャーが、ダウンロードしたパッケージの更新を一時的に保管するのに使用します。**/var** を含むパーティションまたはボリュームは、最低 5 GB となることを確認してください。
- **/usr** ディレクトリーには、一般的な Red Hat Enterprise Linux インストールの大抵のソフトウェアが格納されています。このディレクトリーを含むパーティションまたはボリュームは、最小インストールの場合は最低 5 GiB、グラフィカル環境のインストールの場合は最低 10 GiB 必要です。

- **/usr** または **/var** のパーティションをルートボリュームとは別の場所に設定すると、これらのディレクトリーには起動に欠かせないコンポーネントが含まれているため、起動プロセスが非常に複雑になります。iSCSI ドライブや FCoE などの場所に配置してしまった場合には、システムが起動できなくなったり、電源オフや再起動の際に **Device is busy** のエラーでハングしたりする可能性があります。
  これらの制限は **/usr** と **/var** にのみ適用され、その下のディレクトリーには適用されません。たとえば、**/var/www** 向けの個別パーティションは、問題なく機能します。



重要

一部のセキュリティーポリシーでは、管理がより複雑になりますが、**/usr** と **/var** の分離が必要になります。

- LVM ボリュームグループ内の一部領域を未割り当てのまま残しておくことを検討してください。このように未割り当ての領域を残すことで、領域の要件が変化した際に、その他のボリュームからデータを削除したくない場合に、柔軟性が得られます。また、パーティションに **LVM シンプロビジョニング** デバイスタイプを選択し、ボリュームに未使用の領域を自動的に処理させることもできます。

- XFS ファイルシステムのサイズを縮小することはできません。このファイルシステムのパーティションまたはボリュームを小さくする必要がある場合は、データのバックアップを作成し、ファイルシステムを破棄して、代わりに小規模なファイルシステムを新たに作成する必要があります。したがって、後でパーティションレイアウトを変更する予定の場合には、代わりに ext4 ファイルシステムを使用してください。

- インストール後に、ハードドライブの追加、または仮想マシンのハードドライブの拡張によりストレージを拡張することを予定している場合は、論理ボリューム管理 (LVM) を使用してください。LVM を使用すると、新しいドライブに物理ボリュームを作成し、必要に応じてそのボリュームをボリュームグループおよび論理ボリュームに割り当てることができます。たとえば、システムの **/home** (または論理ボリュームに存在するその他のディレクトリー) は簡単に拡張できます。

- システムのファームウェア、起動ドライブのサイズ、および起動ドライブのディスクラベルによっては、BIOS の起動パーティションまたは EFI システムパーティションの作成が必要になる場合があります。システムで BIOS ブートまたは EFI システムパーティションが **必要ない** 場合は、グラフィカルインストールで BIOS ブートまたは EFI システムパーティションを作成することはできません。この場合は、メニューに表示されなくなります。

- インストール後にストレージ設定に変更を加える必要がある場合は、Red Hat Enterprise Linux リポジトリーで役に立つツールがいくつか提供されています。コマンドラインツールを使用する場合は、**system-storage-manager** を試してみてください。

関連情報

- IBM Z、LinuxONE、および PAES 暗号で dm-crypt を使用する方法

E.6. サポート対象のハードウェアストレージ

Red Hat Enterprise Linux のメジャーバージョン間でストレージ技術がどのように設定され、そのサポートがどのように変更したかを理解することが重要になります。

ハードウェア RAID

インストールプロセスを開始する前に、コンピューターのマザーボードが提供する RAID 機能、またはコントローラーカードが接続する RAID 機能を設定する必要があります。アクティブな RAID アレイは、それぞれ Red Hat Enterprise Linux 内で 1 つのドライブとして表示されます。

ソフトウェア RAID

システムに複数のハードドライブが搭載されている場合は、Red Hat Enterprise Linux インストールプログラムを使用して、複数のドライブを 1 つの Linux ソフトウェア RAID アレイとして動作させることができます。ソフトウェア RAID アレイを使用すると、RAID 機能は専用のハードウェアではなく、オペレーティングシステムにより制御されることになります。

注記

以前から存在している RAID アレイの全メンバーデバイスが、パーティションが設定されていないディスクまたはドライブの場合、インストールプログラムは、アレイをディスクとして扱い、アレイを削除する方法はありません。

USB ディスク

インストール後に外付け USB ストレージを接続して設定できます。ほとんどのデバイスはカーネルにより認識されますが、認識されないデバイスもあります。インストール中にこれらのディスクを設定する必要がない場合は切断して、潜在的な問題を回避してください。

NVDIMM デバイス

不揮発性デュアルインラインメモリーモジュール (NVDIMM) デバイスをストレージとして使用するには、次の条件を満たす必要があります。

- Red Hat Enterprise Linux のバージョンが、7.6 以降である。
- システムのアーキテクチャーが Intel 64 または AMD64 である。
- デバイスが、セクターモードに設定されている。Anaconda で、NVDIMM デバイスをこのモードに再設定できます。
- nd_pmem ドライバーがそのデバイスに対応している。

さらに以下の条件が満たされる場合には、NVDIMM デバイスからの起動が可能です。

- システムが UEFI を使用している。
- システムで使用可能なファームウェアまたは UEFI ドライバーがデバイスをサポートしている。UEFI ドライバーは、デバイス自体のオプション ROM から読み込むことができます。
- デバイスが名前空間で利用可能である。

システムの起動中に高性能な NVDIMM デバイスを利用するには、デバイスに **/boot** ディレクトリーおよび **/boot/efi** ディレクトリーを置きます。

注記

NVDIMM デバイスの XIP (Execute-in-place) 機能は、起動時にはサポートされません。カーネルは従来どおりメモリーに読み込まれます。

Intel の BIOS RAID に関する注意点

Red Hat Enterprise Linux は、Intel BIOS RAID セットへのインストールに、**mdraid** を使用します。こ

のセットは起動プロセスで自動検出されるため、起動するたびにデバイスノードパスが変わる可能性があります。デバイスノードのパス (**/dev/sda** など) を、ファイルシステムのラベルまたはデバイスの UUID に置き換えることが推奨されます。ファイルシステムのラベルおよびデバイスの UUID は、**blkid** コマンドを使用すると確認できます。

付録F BOOT オプションの参照

このセクションは、インストールプログラムのデフォルトの挙動を変更するのに使用できる起動オプションの一部を説明します。キックスタートと高度な起動オプションについては、RHEL インストーラーの起動オプション を参照してください。

F.1. インストールソースの起動オプション

このセクションでは、さまざまなインストールソースのブートオプションについて説明します。

inst.repo=

inst.repo= 起動オプションはインストールソースを指定します。つまり、パッケージリポジトリーと、そのリポジトリーを記述する有効な **.treeinfo** ファイルを提供する場所にあたります。たとえば、**inst.repo=cdrom** になります。**inst.repo=** オプションの対象は、以下のいずれかのインストールメディアになります。

- インストール可能なツリー (インストールプログラムのイメージ、パッケージ群、リポジトリーデータおよび有効な **.treeinfo** ファイルを含むディレクトリー設定)
- DVD (システムの DVD ドライブにある物理ディスク)
- Red Hat Enterprise Linux のフルインストール用 DVD の ISO イメージ (ハードドライブ、またはシステムにアクセスできるネットワーク上の場所)
  inst.repo= 起動オプションでは、さまざまなインストール方法を設定します。以下の表は、**inst.repo=** 起動オプションの詳細な構文を記載します。

表F.1 inst.repo= ブートオプションおよびインストールソースのタイプおよびフォーマット

ソースタイプ	起動オプションの形式	ソースの形式
CD/DVD ドライブ	**inst.repo=cdrom:<device>**	物理ディスクとしてのインストール DVD。[a]
マウント可能なデバイス (HDD および USB スティック)	**inst.repo=hd:<device>:/<path>**	インストール DVD のイメージファイル
NFS サーバー	**inst.repo=nfs:[options:]<server>:/<path>**	インストール DVD のイメージファイル、またはインストールツリー (インストール DVD にあるディレクトリーおよびファイルの完全なコピー)。[b]
HTTP サーバー	**inst.repo=http://<host>/<path>**	インストールツリー (インストール DVD 上にあるディレクトリーおよびファイルの完全なコピー)。
HTTPS サーバー	**inst.repo=https://<host>/<path>**	

ソースタイプ	起動オプションの形式	ソースの形式
FTP サーバー	**inst.repo=ftp://<username>:<password>@<host>/<path>**	
HMC	**inst.repo=hmc**	

[a] **device** が省略された場合、インストールプログラムはインストール DVD を含むドライブを自動的に検索します。

[b] NFS サーバーのオプションでは、デフォルトで NFS プロトコルのバージョン 3 が使用されます。別のバージョンを使用するには、**nfsvers=X** を **オプション** に追加し、**X** を、使用するバージョン番号に置き換えます。

ディスクデバイス名は、次の形式で設定します。

- カーネルデバイス名 (例: **/dev/sda1** または **sdb2**)
- ファイルシステムのラベル (例: **LABEL=Flash** または **LABEL=RHEL8**)
- ファイルシステムの UUID (例: **UUID=8176c7bf-04ff-403a-a832-9557f94e61db**)

英数字以外は **\xNN** で表す必要があります。**NN** は文字の 16 進数表示になります。たとえば、**\x20** なら空白 **(" ")** になります。

inst.addrepo=

inst.addrepo= 起動オプションを使用して、別のインストールソースとして、メインリポジトリー (**inst.repo=**) とともに追加のリポジトリーを追加します。起動時に、**inst.addrepo=** 起動オプションを複数回使用できます。以下の表では、**inst.addrepo=** 起動オプションの構文の詳細を記載します。

注記

REPO_NAME はリポジトリーの名前であり、インストールプロセスでは必須です。これらのリポジトリーは、インストールプロセス時にのみ使用され、インストールしたシステムにはインストールされません。

統一された ISO に関する詳細は、Unified ISO を参照してください。

表F.2 インストールソースおよびブートオプションの形式

インストールソース	起動オプションの形式	関連情報
URL にあるインストール可能なツリー	**inst.addrepo=REPO_NAME, [http,https,ftp]://<host>/<path>**	指定の URL にあるインストール可能なツリーを探します。

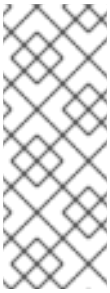
インストールソース	起動オプションの形式	関連情報
NFS パスにあるインストール可能なツリー	**inst.addrepo=REPO_NAME,nfs://<server>:/<path>**	指定した NFS パスのインストール可能なツリーを探します。コロンは、ホストの後に必要です。インストールプログラムは、RFC 2224 に従って URL の解析を行うのではなく、**nfs://** ディレクトリーの後のすべてを mount コマンドに渡します。
インストール環境でインストール可能なツリー	**inst.addrepo=REPO_NAME,file://<path>**	インストール環境の指定した場所にあるインストール可能なツリーを探します。このオプションを使用するには、インストールプログラムが利用可能なソフトウェアグループのロードを試行する前に、リポジトリーがマウントされる必要があります。このオプションの利点は、起動可能な ISO に複数のリポジトリーを利用でき、ISO からメインリポジトリーと追加のリポジトリーの両方をインストールできることです。追加のリポジトリーへのパスは **/run/install/source/REPO_ISO_PATH** です。また、キックスタートファイルの **%pre** セクションにリポジトリーディレクトリーをマウントできます。パスは、**inst.addrepo=REPO_NAME,file:///<path>** など、/ で始まる必要があります。
ハードドライブ	**inst.addrepo=REPO_NAME,hd:<device>:<path>**	指定した <device> パーティションをマウントして、<path> で指定した ISO からインストールします。<path> を指定しないと、インストールプログラムは <device> 上の有効なインストール ISO を探します。このインストール方法には、有効なインストールツリーを持つ ISO が必要です。

inst.stage2=

inst.stage2= 起動オプションは、インストールプログラムのランタイムイメージの場所を指定します。このオプションは、有効な **.treeinfo** ファイルが含まれるディレクトリーへのパスを想定し、**.treeinfo** ファイルからランタイムイメージの場所を読み取ります。**.treeinfo** ファイルが利用できないと、インストールプログラムは、**images/install.img** からイメージを読み込もうとします。**inst.stage2** オプションを指定しない場合、インストールプログラムは **inst.repo** オプションで指定された場所を使用しようとします。

このオプションは、後でインストールプログラム内でインストールソースを手動で指定する場合に使用します。たとえば、インストールソースとしてコンテンツ配信ネットワーク (CDN) を選択する場合などに使用します。インストール DVD および Boot ISO には、それぞれの ISO からインストールプログラムを起動するための適切な **inst.stage2** オプションがすでに含まれています。

インストールソースを指定する場合は、代わりに **inst.repo=** オプションを使用します。



注記

デフォルトでは、インストールメデイアで **inst.stage2=** 起動オプションが使用され、これは特定のラベル (たとえば **inst.stage2=hd:LABEL=RHEL-x-0-0-BaseOS-x86_64**) に設定されています。ランタイムイメージが含まれるファイルシステムのデフォルトラベルを修正する場合、またはカスタマイズされた手順を使用してインストールシステムを起動する場合は、**inst.stage2=** 起動オプションに正しい値が設定されていることを確認してください。

inst.noverifyssl

inst.noverifyssl 起動オプションを使用して、追加のキックスタートリポジトリーを除き、すべての HTTPS 接続の SSL 証明書が検証されないようにします。ただし、**--noverifyssl** はリポジトリーごとに設定できます。
たとえば、リモートのインストールソースが自己署名 SSL 証明書を使用している場合には、**inst.noverifyssl** 起動オプションは、SSL 証明書を検証せずにインストーラーがインストールを完了できるようにします。

inst.stage2= を使用してソースを指定する場合の例

```
inst.stage2=https://hostname/path_to_install_image/ inst.noverifyssl
```

inst.repo= を使用してソースを指定する場合の例

```
inst.repo=https://hostname/path_to_install_repository/ inst.noverifyssl
```

inst.stage2.all

inst.stage2.all 起動オプションを使用して、複数の HTTP、HTTPS、または FTP ソースを指定します。**inst.stage2=** 起動オプションは、**inst.stage2.all** オプションとともに複数回使用して、成功するまで、イメージを順番にフェッチできます。以下に例を示します。

```
inst.stage2.all
inst.stage2=http://hostname1/path_to_install_tree/
inst.stage2=http://hostname2/path_to_install_tree/
inst.stage2=http://hostname3/path_to_install_tree/
```

inst.dd=

インストール時にドライバーの更新を実行する場合は、**inst.dd=** 起動オプションを使用します。インストール時にドライバーを更新する方法の詳細は、高度な RHEL 8 インストールの実行 を参照してください。

inst.repo=hmc

このオプションにより、外部ネットワーク設定の必要がなくなるため、インストールのオプションが増えます。Binary DVD から起動すると、インストーラープログラムにより、追加のカーネルパラメーターを入力するように求められます。DVD をインストールソースとして設定するには、**inst.repo=hmc** オプションをカーネルパラメーターに追加します。インストールプログラム

は、サポート要素 (SE) およびハードウェア管理コンソール (HMC) のファイルアクセスを有効にし、DVD から stage2 のイメージをフェッチし、ソフトウェア選択のために DVD のパッケージへのアクセスを提供します。

inst.proxy=

HTTP、HTTPS、および FTP プロトコルからインストールを実行する場合には、**inst.proxy=** 起動オプションが使用されます。以下に例を示します。

```
[PROTOCOL://][USERNAME[:PASSWORD]@]HOST[:PORT]
```

inst.nosave=

inst.nosave= 起動オプションを指定して、インストールログや関連ファイルがインストール済みのシステムに保存されないように制御します (例: **input_ks**、**output_ks**、**all_ks**、**logs**、**all**)。複数の値をコンマで区切って組み合わせることができます。以下に例を示します。

```
inst.nosave=Input_ks,logs
```

注記

inst.nosave 起動オプションは、インストール済みのシステムから、キックスタートのログや入力/出力などの Kickstart %post スクリプトで削除できないファイルの除外に使用されます。

input_ks

キックスタートによる入力を保存する機能を無効にします。

output_ks

インストールプログラムで生成されたキックスタートによる出力を保存する機能を無効にします。

all_ks

キックスタートによる入出力を保存する機能を無効にします。

logs

すべてのインストールログを保存する機能を無効にします。

all

すべてのキックスタート結果とすべてのログを保存する機能を無効にします。

inst.multilib

inst.multilib 起動オプションを使用して、DNF の **multilib_policy** を、**best** ではなく **all** に設定します。

inst.memcheck

inst.memcheck 起動オプションは、インストールを完了するのにシステムに十分な RAM があることを確認するためのチェックを実行します。RAM が十分でない場合は、インストールプロセスが停止します。システムのチェックはおおよそのもので、インストールの際のメモリー使用率は、パッケージ選択やユーザーインターフェイス (グラフィカル、テキスト)、その他のパラメーターにより異なります。

inst.nomemcheck

inst.nomemcheck 起動オプションは、インストールを完了するのに十分な RAM があるかどうかの確認を実行しません。推奨よりも低いメモリー量でのインストールはサポートされていないため、インストールプロセスが失敗する場合があります。

F.2. ネットワーク起動オプション

シナリオでローカルイメージから起動するのではなく、ネットワーク経由でイメージから起動する必要がある場合は、次のオプションを使用してネットワーク起動をカスタマイズできます。

注記

dracut ツールを使用してネットワークを初期化します。**dracut** オプションの完全なリストについては、**dracut.cmdline(7)** の man ページを参照してください。

ip=

ip= 起動オプションは、1 つ以上のネットワークインターフェイスを設定します。複数のインターフェイスを設定するには、次のいずれかの方法を使用します。

- インターフェイスごとに 1 回ずつ、**ip** オプションを複数回使用します。これを行うには、**rd.neednet=1** オプションを使用し、**bootdev** オプションを使用してプライマリーブートインターフェイスを指定します。
- **ip** オプションを 1 回使用してから、Kickstart を使用してさらにインターフェイスを設定します。このオプションでは、複数の形式が使用できます。以下の表は、最も一般的なオプションの情報が含まれます。

以下の表では、下記の点を前提としています。

- **ip** パラメーターはクライアントの IP アドレスを指定し、**IPv6** には角括弧が必要です (例: 192.0.2.1 または [2001:db8::99])。
- **gateway** パラメーターはデフォルトのゲートウェイになります。**IPv6** には角括弧必要です。
- **netmask** パラメーターは使用するネットマスクです。完全ネットマスク (255.255.255.0 など) または接頭辞 (64 など) を使用できます。
- **hostname** パラメーターはクライアントシステムのホスト名です。このパラメーターは任意です。

表F.3 ネットワークインターフェイスを設定するためのブートオプション形式

起動オプションの形式	設定方法
ip=method	全インターフェイスの自動設定
ip=interface:method	特定インターフェイスの自動設定
ip=ip::gateway:netmask:hostname:interface:none	静的設定 (例: IPv4 **ip=192.0.2.1::192.0.2.254:255.255.255.0:server.example.com:enp1s0:none**) IPv6: **ip=[2001:db8::1]::[2001:db8::fffe]:64:server.example.com:enp1s0:none**
ip=ip::gateway:netmask:hostname:interface:method:mtu	オーバーライドを使用した特定インターフェイスの自動設定

自動インターフェイスの設定方法

オーバーライドを使用した特定インターフェイスの自動設定 では、**dhcp** など、指定した自動設定方法を使用してインターフェイスを起動しますが、自動取得した IP アドレス、ゲートウェイ、ネットマスク、ホスト名、他のパラメーターなどで指定したものは無効にします。パラメーターはすべて任意となるため、無効にするパラメーターだけを指定します。

method パラメーターには、以下のいずれかを使用します。

DHCP

dhcp

IPv6 DHCP

dhcp6

IPv6 自動設定

auto6

iBFT (iSCSI Boot Firmware Table)

ibft

注記

- **ip** オプションを指定せずに、**inst.ks=http://host/path** などのネットワークアクセスを必要とするブートオプションを使用する場合、**ip** オプションのデフォルト値は **ip=dhcp** です。
- iSCSI ターゲットに自動的に接続するには、**ip=ibft** ブートオプションを使用して、ターゲットにアクセスするネットワークデバイスをアクティブ化します。

nameserver=

nameserver= オプションは、ネームサーバーのアドレスを指定します。このオプションは複数回使用できます。

注記

ip= パラメーターには角括弧が必要です。ただし、IPv6 アドレスには角括弧が使用できません。IPv6 アドレスに使用する正しい構文は **nameserver=2001:db8::1** のようになります。

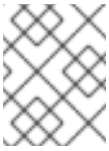
bootdev=

bootdev= オプションは、起動インターフェイスを指定します。このオプションは、**ip** オプションを複数回使用する場合に必要になります。

ifname=

ifname= オプションは、特定の MAC アドレスを持つネットワークデバイスにインターフェイス名を割り当てます。このオプションは複数回使用できます。構文は、**ifname=interface:MAC** です。以下に例を示します。

```
ifname=eth0:01:23:45:67:89:ab
```



注記

ifname= オプションは、インストール中にカスタムのネットワークインターフェイス名を設定する際にサポートされる唯一の方法となります。

inst.dhcpclass=

inst.dhcpclass= オプションは、DHCP のベンダークラス識別子を指定します。**dhcpd** サービスではこの値を **vendor-class-identifier** として認識します。デフォルト値は **anaconda-$(uname -srm)** です。

inst.waitfornet=

inst.waitfornet=SECONDS 起動オプションを使用すると、インストールシステムは、ネットワーク接続を待ってからインストールします。**SECONDS** 引数で指定する値は、ネットワーク接続がない場合でもすぐにはタイムアウトにせず、ネットワーク接続を待ち続け、インストールプロセスを継続する最大秒数を表します。

vlan=

vlan= オプションを使用して、仮想 LAN (VLAN) デバイスに特定の名前を付け、指定インターフェイスにそのデバイスを設定します。構文は **vlan=name:interface** です。以下に例を示します。

```
vlan=vlan5:enp0s1
```

これにより、**enp0s1** インターフェイスに **vlan5** という名前の VLAN デバイス が設定されます。name は以下のような形式をとります。

- VLAN_PLUS_VID: **vlan0005**
- VLAN_PLUS_VID_NO_PAD: **vlan5**
- DEV_PLUS_VID: **enp0s1.0005**
- DEV_PLUS_VID_NO_PAD: **enp0s1.5**

bond=

bond= オプションを使用して、**bond=name[:interfaces][:options]** 構文でボンディングデバイスを設定します。**name** はボンディングデバイス名に置き換え、**interfaces** は物理 (イーサネット) インターフェイスのコンマ区切りリストに置き換え、**options** はボンディングオプションのコンマ区切りリストに置き換えます。以下に例を示します。

```
bond=bond0:enp0s1,enp0s2:mode=active-backup,tx_queues=32,downdelay=5000
```

利用可能なオプションのリストは、ボンディングコマンド **modinfo** を実行します。

team=

team= オプションを使用して、**team=name:interfaces** 構文でチームデバイスを設定します。チームデバイスの基礎となるインターフェイスとして使用されるように、**name** はチームデバイスの望ましい名前に、**interfaces** は物理 (イーサネット) デバイスのコンマ区切りリストに置き換えます。以下に例を示します。

```
team=team0:enp0s1,enp0s2
```

bridge=

bridge= オプションを使用して、**bridge=name:interfaces** 構文でブリッジデバイスを設定します。ブリッジデバイスの基礎となるインターフェイスとして使用されるように、**name** はブリッジデバイスの望ましい名前に、**interfaces** は物理 (イーサネット) デバイスのコンマ区切りリストに置き換えます。以下に例を示します。

```
bridge=bridge0:enp0s1,enp0s2
```

関連情報

- **ネットワークの設定および管理**

F.3. コンソール起動オプション

このセクションでは、コンソール、モニターディスプレイ、およびキーボードの起動オプションを設定する方法を説明します。

console=

console= オプションを使用して、プライマリーコンソールとして使用するデバイスを指定します。たとえば、最初のシリアルポートでコンソールを使用するには、**console=ttyS0** を使用します。**console=** 引数を使用する場合、インストールはテキスト UI から始まります。 **console=** オプションを複数回使用する必要がある場合は、指定したすべてのコンソールにブートメッセージが表示されます。ただし、インストールプログラムは、最後に指定されたコンソールのみを使用します。たとえば、**console=ttyS0 console=ttyS1** と指定すると、インストールプログラムでは **ttyS1** が使用されます。

inst.lang=

inst.lang= オプションを使用して、インストール時に使用する言語を設定します。ロケールのリストを表示するには、コマンド **locale -a | grep _** または **localectl list-locales | grep _** コマンドを実行します。

inst.singlelang

inst.singlelang を指定して単一の言語モードでインストールを行うと、そのインストール言語と言語サポート設定に対する対話オプションを利用できません。**inst.lang** 起動オプションまたは **lang** キックスタートコマンドを使用して言語を指定すると、オプションが指定されます。言語を指定しないと、インストールプログラムのロケールはデフォルトで **en_US.UTF-8** となります。

inst.geoloc=

インストールプログラムで、地理位置情報の使用方法を設定するには、**inst.geoloc=** オプションを使用します。地理位置情報は、言語およびタイムゾーンの事前設定に使用され、**inst.geoloc=value** 構文を使用します。**value** には、以下のいずれかのパラメーターを使用します。

- 地理位置情報の無効化: **inst.geoloc=0**
- Fedora GeoIP API (**inst.geoloc=provider_fedora_geoip**) を使用します。
- Hostip.info GeoIP API (**inst.geoloc=provider_hostip**) を使用します。

inst.geoloc= オプションを指定しない場合、デフォルトのオプションは **provider_fedora_geoip** です。

inst.keymap=

inst.keymap= オプションを使用して、インストールに使用するキーボードレイアウトを指定します。

inst.cmdline

inst.cmdline オプションを使用して、インストールプログラムをコマンドラインモードで強制的に実行します。このモードでは対話が使用できないため、キックスタートファイルまたはコマンドラインですべてのオプションを指定する必要があります。

inst.graphical

インストールプログラムをグラフィカルモードで強制的に実行するには、**inst.graphical** オプションを使用します。グラフィカルモードがデフォルトです。

inst.text

inst.text オプションを使用して、グラフィカルモードではなく、テキストモードでインストールプログラムを強制的に実行します。

inst.noninteractive

inst.noninteractive 起動オプションを使用して、非対話モードでインストールプログラムを実行します。非対話型モード (および **inst.noninteractive**) では、ユーザーとの対話は許可されていません。グラフィカルまたはテキストインストールで **inst.nointeractive** オプションを使用できます。**inst.noninteractive** オプションをテキストモードで使用すると、**inst.cmdline** オプションと同じように動作します。

inst.resolution=

inst.resolution= オプションを使用して、グラフィカルモードで、画面の解像度を指定します。形式は **NxM** です。**N** は画面の幅で、**M** は画面の高さ (ピクセル単位) です。サポートされる最小解像度は 1024x768 です。

inst.vnc

inst.vnc オプションを使用して、Virtual Network Computing (VNC) を使用したグラフィカルインストールを実行します。インストールプログラムと対話するには VNC クライアントアプリケーションを使用する必要があります。VNC 共有を有効にすると、複数のクライアントに接続できます。VNC を使用してインストールしたシステムは、テキストモードで起動します。

inst.vncpassword=

inst.vncpassword= オプションを使用して、インストールプログラムが使用する VNC サーバーにパスワードを設定します。

inst.vncconnect=

inst.vncconnect= オプションを使用して、指定されたホストの場所にあるリスニング VNC クライアントに接続します (例: **inst.vncconnect=<host>[:<port>]**)。デフォルトのポートは 5900 です。このオプションを使用するには、コマンド **vncviewer -listen** を入力します。

inst.xdriver=

inst.xdriver= オプションを使用して、インストール時およびインストール済みシステムで使用される X ドライバーの名前を指定します。

inst.usefbx

inst.usefbx オプションを使用して、ハードウェア固有のドライバーではなく、フレームバッファー X ドライバーを使用するようにインストールプログラムに要求します。このオプションは、**inst.xdriver=fbdev** オプションと同等です。

modprobe.blacklist=

modprobe.blacklist= オプションを使用して、1 つ以上のドライバーを拒否リストに追加するか、完全に無効にします。このオプションを使用して無効にしたドライバー (mods) は、インストールの開始時にロードできません。インストールが完了すると、インストールされたシステムはこれらの設定を保持します。拒否リストに指定したドライバーのリストは、**/etc/modprobe.d/** ディレクトリーにあります。複数のドライバーを無効にするには、コンマ区切りリストを使用します。以下に例を示します。

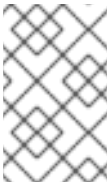
```
modprobe.blacklist=ahci,firewire_ohci
```

inst.xtimeout=

inst.xtimeout= オプションを使用して、X サーバーの起動のタイムアウトを秒単位で指定します。

inst.sshd

インストール時に、SSH を使用してシステムに接続し、インストールの進捗を監視できるように、**inst.sshd** オプションを使用して、**sshd** サービスを開始します。SSH の詳細は、man ページの **ssh(1)** を参照してください。デフォルトでは、**sshd** オプションは、64 ビットの IBM Z アーキテクチャーでのみ自動的に起動します。その他のアーキテクチャーでは、**sshd** は、**inst.sshd** オプションを使用しない限り起動しません。



注記

インストール中に、root アカウントにはデフォルトでパスワードが設定されていません。キックスタートコマンド **sshpw** を使用して、インストール時に root パスワードを設定できます。

inst.kdump_addon=

インストールプログラムで Kdump 設定画面 (アドオン) を有効または無効にするには、**inst.kdump_addon=** オプションを使用します。この画面はデフォルトで有効になっているため、無効にする場合は **inst.kdump_addon=off** を使用します。アドオンを無効にすると、グラフィカルおよびテキストベースのインターフェイスと、キックスタートコマンド **%addon com_redhat_kdump** の両方で Kdump 画面が無効になります。

F.4. 起動オプションのデバッグ

このセクションでは、問題をデバッグするときに使用できるオプションを説明します。

inst.rescue

inst.rescue オプションを使用して、システムの診断と修正のためのレスキュー環境を実行します。たとえば、レスキューモードでファイルシステムを修復 できます。

inst.updates=

inst.updates= オプションを使用して、インストール時に適用する **updates.img** ファイルの場所を指定します。**updated.img** ファイルは、いくつかのソースの 1 つから取得できます。

表F.4 updates.img ファイルソース

ソース	説明	例
ネットワークからの更新	**updates.img** のネットワーク上の場所を指定します。インストールツリーを変更する必要はありません。この方法を使用するには、カーネルコマンドラインを編集して **inst.updates** を追加します。	**inst.updates=http://website.com/path/to/updates.img**.

ソース	説明	例
ディスクイメージからの更新	フロッピードライブまたは USB キーに **updates.img** を保存できます。これは、ファイルシステムタイプが **ext2** の **updates.img** でのみ可能です。イメージの内容をフロッピードライブに保存するには、フロッピーディスクを挿入し、次のコマンドを実行します。	**dd if=updates.img of=/dev/fd0 bs=72k count=20**USB キーまたはフラッシュメディアを使用するには、**/dev/fd0** を、USB キーのデバイス名に置き換えます。
インストールツリーからの更新	CD、ハードドライブ、HTTP、または FTP のインストールを使用する場合は、すべてのインストールツリーが **.img** ファイルを検出できるように、インストールツリーに **updates.img** を保存できます。このファイル名は、**updates.img** にする必要があります。	NFS インストールの場合は、ファイルを **images/** ディレクトリーまたは **RHupdates/** ディレクトリーに保存します。

inst.loglevel=

inst.loglevel= オプションを使用して、端末に記録するログメッセージの最小レベルを指定します。このオプションは、ターミナルログにのみ適用されます。ログファイルには、常にすべてのレベルのメッセージが含まれます。このオプションで可能な値は、最低レベルから最高レベルまで次のとおりです。

- **debug**
- **info**
- **warning**
- **error**
- **critical**

デフォルト値は **info** となるため、デフォルトでは、**info** から **critical** までのメッセージがログの端末に表示されます。

inst.syslog=

インストールの開始時に、指定されたホスト上の **syslog** プロセスにログメッセージを送信します。**inst.syslog=** は、リモート **syslog** プロセスが着信接続を受け入れるように設定されている場合にのみ使用できます。

inst.virtiolog=

inst.virtiolog = オプションを使用して、ログの転送に使用する virtio ポート (**/dev/virtio-ports/name** にある文字デバイス) を指定します。デフォルト値は、**org.fedoraproject.anaconda.log.0** です。

inst.zram=

インストール中の zRAM スワップの使用を制御します。このオプションは、圧縮したブロックデバイスをシステム RAM に作成し、ハードドライブではなくスワップ領域に使用します。この設定により、使用可能なメモリーが少ない状態でインストールプログラムを実行し、インストール速度を向上させることができます。次の値を使用して、**inst.zram=** オプションを設定できます。

- inst.zram=1 は、システムメモリーサイズに関係なく、zRAM スワップを有効にします。デフォルトでは、2GiB 以下の RAM を搭載したシステムで zRAM のスワップが有効になっています。
- inst.zram=0 は、システムメモリーサイズに関係なく、zRAM スワップを無効にします。デフォルトでは、2GiB を超えるメモリーを搭載したシステムでは zRAM のスワップが無効になっています。

rd.live.ram

images/install.img の **stage 2** イメージを RAM にコピーします。これにより、インストールに必要なメモリーがイメージのサイズ (通常は 400 ~ 800MB) だけ増加することに注意してください。

inst.nokill

致命的なエラーが発生したとき、またはインストールプロセスの最後に、インストールプログラムが再起動しないようにします。再起動時に失われるインストールログをキャプチャーするのに使用します。

inst.noshell

インストール中にターミナルセッション 2 (tty2) でシェルを防止します。

inst.notmux

インストール中に tmux を使用しないようにします。この出力は、ターミナル制御文字なしで生成され、非対話用になります。

inst.remotelog=

TCP 接続を使用してすべてのログをリモート **host:port** に送信します。リスナーがなく、インストールが通常通りに進まない場合は、接続が中断されます。

F.5. ストレージ起動オプション

このセクションでは、ストレージデバイスからの起動をカスタマイズするために指定できるオプションを説明します。

inst.nodmraid

dmraid サポートを無効にします。

警告

使用する場合は注意が必要です。ファームウェア RAID アレイの一部として誤って特定されたディスクがある場合は、古い RAID メタデータが存在する可能性があります。これらは、**dmraid** や **wipefs** などの適切なツールを使用して削除する必要があります。

inst.nompath

マルチパスデバイスのサポートを無効にします。このオプションは、システムに誤検知があり、通常のブロックデバイスをマルチパスデバイスとして誤って識別する場合にのみ使用してください。

警告

使用する場合は注意が必要です。マルチパスハードウェアではこのオプションを使用しないでください。このオプションを使用してマルチパスデバイスのシングルパスにインストールすることはサポートされていません。

inst.gpt

インストールプログラムがパーティション情報を Master Boot Record (MBR) ではなく GUID Partition Table (GPT) にインストールするように強制します。このオプションは、BIOS 互換モードである場合を除き、UEFI ベースのシステムでは有効ではありません。通常、BIOS 互換モードの BIOS ベースのシステムおよび UEFI ベースのシステムは、ディスクのサイズが 2^32 セクター以上でない限り、パーティション情報の格納に MBR スキーマを使用しようとします。ディスクセクターは通常 512 バイトで、通常これは 2 TiB に相当します。**inst.gpt** ブートオプションを使用すると、GPT をより小さなディスクに書き込むことができます。

F.6. 廃止予定の起動オプション

このセクションは、非推奨の起動オプションを説明します。これらのオプションはインストールプログラムでも使用できますが、非推奨とされています。また、Red Hat Enterprise Linux の今後のリリースで削除される予定です。

method

method オプションは、**inst.repo** のエイリアスです。

dns

dns の代わりに **nameserver** を使用します。ネームサーバーはコンマ区切りのリストを受け付けず、代わりに複数のネームサーバーオプションを使用することに注意してください。

netmask、gateway、hostname

netmask、**gateway**、および **hostname** オプションは、**ip** オプションの一部として利用できます。

ip=bootif

PXE 指定の **BOOTIF** オプションが自動的に使用されるため、**ip=bootif** を使用する必要はありません。

ksdevice

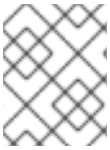
表F.5 ksdevice 起動オプションの値

値	情報
存在しない	該当なし
ksdevice=link	このオプションがデフォルトの動作と同じ場合に無視されます。

値	情報
ksdevice=bootif	**BOOTIF=** が存在する場合は、このオプションはデフォルトであるため無視されます。
ksdevice=ibft	**ip=ibft** に変更詳細は **ip** を参照してください。
ksdevice=<MAC>	**BOOTIF=${MAC/:/-}** に変更
ksdevice=<DEV>	**bootdev** に置き換え

F.7. 削除済みの起動オプション

このセクションでは、Red Hat Enterprise Linux から削除された起動オプションを説明します。



注記

dracut では、高度な起動オプションを利用できます。**dracut** の詳細は、man ページの **dracut.cmdline(7)** を参照してください。

askmethod、asknetwork

initramfs は完全に非対話的に実行されるため、**askmethod** と **asknetwork** のオプションは削除されました。**inst.repo** を使用して、適切なネットワークオプションを指定します。

blacklist、nofirewire

modprobe オプションは、カーネルモジュールのブロックリストを処理するようになりました。**modprobe.blacklist=<mod1>,<mod2>** を使用します。**modprobe.blacklist=firewire_ohci** を使用して、FireWire モジュールを拒否リストに入れることができます。

inst.headless=

headless= オプションでは、インストールしているシステムにディスプレイハードウェアがなく、インストールプログラムがディスプレイハードウェアを検索する必要がないことを指定しています。

inst.decorated

inst.decorated オプションは、装飾画面でのグラフィカルインストールの指定に指定されていまいた。デフォルトでは、この画面は装飾されないため、タイトルバーやサイズ変更などの機能はありません。このオプションは不要になりました。

repo=nfsiso

inst.repo=nfs: オプションを使用します。

serial

console=ttyS0 オプションを指定します。

updates

inst.updates オプションを指定します。

essid、wepkey、wpakey

dracut はワイヤレスネットワークをサポートしません。

ethtool

このオプションは不要になりました。

gdb

dracut ベースの **initramfs** のデバッグには多くのオプションが使用できるため、このオプションは削除されました。

inst.mediacheck

dracut オプションの rd.live.check オプション指定してください。

ks=floppy

inst.ks=hd:<device> オプションを指定します。

display

UI のリモートディスプレイには、**inst.vnc** オプションを指定します。

utf8

このオプションは、デフォルトの TERM 設定が期待通りに動作するため、不要になりました。

noipv6

IPv6 はカーネルに組み込まれたため、インストールプログラムによる削除はできません。**ipv6.disable=1** を使用して ipv6 を無効にすることができます。この設定は、インストール済みシステムによって使用されます。

upgradeany

インストールプログラムがアップグレードを処理しなくなるため、このオプションは不要になりました。

付録G サブスクリプションサービスの変更

サブスクリプションを管理するには、Red Hat Subscription Management Server または Red Hat Satellite Server に RHEL システムを登録します。必要に応じて、後でサブスクリプションサービスを変更できます。登録しているサブスクリプションサービスを変更するには、現在のサービスからシステムの登録を解除し、新しいサービスに登録します。

このセクションは、Red Hat Subscription Management Server および Red Hat Satellite Server から RHEL システムの登録を解除する方法を説明します。

前提条件

以下のいずれかでシステムを登録している。

- Red Hat Subscription Management Server
- Red Hat Satellite Server version 6.11

注記

システムの更新を受け取るには、いずれかの管理サーバーでシステムを登録します。

G.1. SUBSCRIPTION MANAGEMENT SERVER からの登録解除

このセクションでは、コマンドラインと Subscription Manager ユーザーインターフェイスを使用して、Red Hat Subscription Management Server から RHEL システムの登録を解除する方法を説明します。

G.1.1. コマンドラインでの登録解除

unregister コマンドを使用して、Red Hat Subscription Management Server から RHEL システムの登録を解除します。

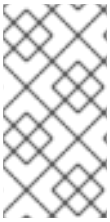
手順

1. root ユーザーで unregister コマンドにパラメーターを付けずに実行します。

```
# subscription-manager unregister
```

2. プロンプトが表示されたら、root パスワードを入力します。

システムが Subscription Management Server から登録解除され、ステータス The system is currently not registered が表示され、**登録** ボタンが有効になります。



注記

中断しなかったサービスを続けるには、いずれかの管理サービスでシステムの再登録を行います。管理サービスでシステムを登録しないと、システムの更新を受け取らないことがあります。システム登録の詳細は、コマンドラインでシステムの登録 を参照してください。

関連情報

- Using and Configuring Red Hat Subscription Manager

G.1.2. Subscription Manager ユーザーインターフェイスを使用した登録解除

このセクションでは、Subscription Manager ユーザーインターフェイスを使用して、Red Hat Subscription Management Server から RHEL システムの登録を解除する方法を説明します。

手順

1. システムにログインします。
2. 画面左上で、**アクティビティー** をクリックします。
3. メニューオプションから、**アプリケーションを表示する** アイコンをクリックします。
4. **Red Hat Subscription Manager** アイコンをクリックするか、検索に **Red Hat Subscription Manager** と入力します。
5. **認証が必要です** ダイアログボックスで管理者パスワードを入力します。**サブスクリプション** 画面が開き、サブスクリプションの現在のステータス、システムの目的、インストール済み製品が表示されます。未登録の製品には、赤い X 印が表示されます。

注記

システムで特権タスクを実行するには、認証が必要です。

6. **登録解除** ボタンをクリックします。

システムが Subscription Management Server から登録解除され、ステータス The system is currently not registered が表示され、**登録** ボタンが有効になります。

注記

中断しなかったサービスを続けるには、いずれかの管理サービスでシステムの再登録を行います。管理サービスでシステムを登録しないと、システムの更新を受け取らないことがあります。システム登録の詳細は、Subscription Manager ユーザーインターフェイスを使用したシステム登録 を参照してください。

関連情報

- Using and Configuring Red Hat Subscription Manager

G.2. SATELLITE SERVER からの登録解除

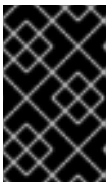
Satellite Server から Red Hat Enterprise Linux システムの登録を解除するには、Satellite Server からシステムを削除します。

詳細は、Satellite Server ドキュメントの **ホストの管理**の Satellite Server からのホストの削除 を参照してください。

付録H インストールプログラムの ISCSI ディスク

Red Hat Enterprise Linux インストーラーは、以下のいずれかの方法で iSCSI ディスクを検出し、ログインできます。

- インストーラーが起動すると、システムの BIOS またはアドオンブート ROM が iSCSI から起動できるシステムの BIOS 拡張である iBFT (iSCSI Boot Firmware Table) に対応しているかどうかを確認します。BIOS が iBFT に対応している場合は、インストーラーは BIOS から設定済みのブートディスクの iSCSI ターゲット情報を読み取り、このターゲットにログインして、インストールターゲットとして利用可能にします。



重要

iSCSI ターゲットに接続するには、ターゲットにアクセスするネットワークデバイスをアクティベートします。これを行うには、起動オプション **ip=ibft** を使用します。詳細は、Network boot options を参照してください。

- インストーラーのグラフィカルユーザーインターフェイスで iSCSI ターゲットの検出と追加を手動で行うことができます。詳細は、ストレージデバイスの設定 を参照してください。

重要

この方法を使用して手動で追加した iSCSI ターゲットには **/boot** パーティションを置くことができません。**/boot** パーティションを含む iSCSI ターゲットを iBFT で使用するように設定する必要があります。ただし、インストールされたシステムが、たとえば iPXE を使用して、ファームウェアの iBFT 以外の方法で提供された iBFT 設定で iSCSI から起動する場合は、**inst.nonibftiscsiboot** インストーラー起動オプションを使用して **/boot** パーティション制限を削除できます。

インストーラーは **iscsiadm** を使用して iSCSI ターゲットを検索し、ログインしますが、**iscsiadm** は自動的にこれらのターゲットに関する情報を **iscsiadm** iSCSI データベースに保存します。その後、インストーラーはこのデータベースをインストール済みシステムにコピーし、root パーティションに使用されていない iSCSI ターゲットをマークします。これにより、システムは起動時に自動的にそのターゲットにログインします。root パーティションを iSCSI ターゲットに置くと、initrd はこのターゲットにログインし、インストーラーは同じターゲットへのログインを複数回試行しないように、起動スクリプトにこのターゲットを含めません。

第6章 UEFI セキュアブートを使用したベータシステムの起動

オペレーティングシステムのセキュリティーを強化するには、UEFI セキュアブートが有効になっているシステムで Red Hat Enterprise Linux ベータ版リリースを起動したときに、署名の検証に UEFI セキュアブート機能を使用します。

6.1. UEFI セキュアブートおよび RHEL ベータ版リリース

UEFI セキュアブートでは、オペレーティングシステムカーネルが、認識された秘密キーで署名されている必要があります。UEFI セキュアブートは、対応する公開キーを使用して署名を検証します。

Red Hat Enterprise Linux 8 のベータリリースの場合には、カーネルは Red Hat ベータ固有の秘密鍵で署名されます。UEFI セキュアブートは、対応する公開鍵を使用して署名を検証しようとしますが、このハードウェアはベータ版の秘密鍵を認識しないため、Red Hat Enterprise Linux ベータ版のリリースシステムは起動に失敗します。そのため、ベータリリースで UEFI セキュアブートを使用するには、MOK (Machine Owner Key) 機能を使用して Red Hat ベータ公開キーをシステムに追加します。

6.2. UEFI セキュアブートのベータ公開鍵の追加

このセクションでは、UEFI セキュアブート用に Red Hat Enterprise Linux ベータ版の公開鍵を追加する方法を説明します。

前提条件

- システムで UEFI セキュアブートが無効になっています。
- Red Hat Enterprise Linux ベータ版リリースがインストールされており、システムの再起動もセキュアブートが無効になっている。
- システムにログインし、**初期セットアップ** 画面でタスクを完了します。

手順

1. システムの Machine Owner Key (MOK) リストに Red Hat ベータ版の公開鍵の登録を開始します。

   ```
   # mokutil --import /usr/share/doc/kernel-keys/$(uname -r)/kernel-signing-ca.cer
   ```

   $(uname -r) はカーネルバージョン (**4.18.0-80.el8.x86_64** など) に置き換えられます。

2. プロンプトが表示されたらパスワードを入力します。
3. システムを再起動し、任意のキーを押して起動を続行します。Shim UEFI キー管理ユーティリティーは、システム起動時に起動します。
4. **Enroll MOK** を選択します。
5. **Continue** を選択します。
6. **Yes** を選択し、パスワードを入力します。この鍵はシステムのファームウェアにインポートされます。
7. **Reboot** を選択します。
8. システムでセキュアブートを有効にします。

6.3. ベータ版公開鍵の削除

Red Hat Enterprise Linux ベータ版リリースを削除し、Red Hat Enterprise Linux General Availability (GA) リリースをインストールするか、別のオペレーティングシステムをインストールする予定の場合は、ベータ版の公開鍵を削除します。

この手順では、ベータ版の公開鍵を削除する方法を説明します。

手順

1. システムの Machine Owner Key (MOK) リストから Red Hat ベータ版の公開鍵の削除を開始します。

```
# mokutil --reset
```

2. プロンプトが表示されたらパスワードを入力します。
3. システムを再起動し、任意のキーを押して起動を続行します。Shim UEFI キー管理ユーティリティーは、システム起動時に起動します。
4. **Reset MOK** を選択します。
5. **Continue** を選択します。
6. **Yes** を選択し、手順 2 で指定したパスワードを入力します。この鍵はシステムのファームウェアから削除されます。
7. **Reboot** を選択します。

第7章 RHEL システムイメージのカスタマイズ

7.1. IMAGE BUILDER の説明

システムをクラウドプラットフォームにデプロイするには、システムイメージを作成します。RHEL システムイメージを作成するには、Image Builder ツールを使用します。

7.1.1. Image Builder とは?

Image Builder を使用することで、RHEL のカスタマイズされたシステムイメージを作成できます。これには、クラウドプラットフォームへのデプロイメント用に準備されたシステムイメージが含まれます。Image Builder は、各出力タイプのセットアップの詳細を自動的に処理するため、手動でイメージを作成する方法よりも使いやすく、作業が高速です。**composer-cli** ツールのコマンドラインインターフェイスで Image Builder 機能、または RHEL Web コンソールでグラフィカルインターフェイスにアクセスできます。

注記

RHEL 8.3 以降では、**osbuild-composer** バックエンドが **lorax-composer** に取って代わります。新しいサービスには、イメージビルド向けの REST API が含まれます。

7.1.2. Image Builder の用語

ブループリント

ブループリントは、カスタマイズされたシステムイメージの説明です。システムの一部となるパッケージとカスタマイズが一覧表示されます。ブループリントをカスタマイズして編集し、特定のバージョンとして保存できます。ブループリントからシステムイメージを作成すると、RHEL Web コンソールのイメージビルダーインターフェイスでイメージがブループリントに関連付けられます。
ブループリントは TOML 形式で作成できます。

Compose

コンポーズは、特定のブループリントの特定のバージョンに基づいた、システムイメージの個別のビルドです。用語としての Compose は、システムイメージと、その作成、入力、メタデータ、およびそのプロセス自体のログを指します。

カスタマイズ

カスタマイズは、パッケージではないイメージの仕様です。これには、ユーザー、グループ、および SSH 鍵が含まれます。

7.1.3. Image Builder の出力形式

Image Builder は、次の表に示す出力形式でイメージを作成できます。サポートされているタイプを確認するには、次のコマンドを実行します。

```
# composer-cli compose types
```

表7.1 Image Builder の出力形式

説明	CLI 名	ファイル拡張子
QEMU QCOW2 イメージ	**qcow2**	**.qcow2**
tar アーカイブ	**tar**	**.tar**
Amazon Machine Image ディスクの作成	**ami**	**.raw**
Azure ディスクイメージ	**vhd**	**.vhd**
Google Cloud Platform	**gce**	**.vhd**
VMware 仮想マシンディスク	**vmdk**	**.vmdk**
Openstack	**openstack**	**.qcow2**
RHEL for Edge のコミット	**edge-commit**	**.tar**
RHEL for Edge コンテナー	**edge-container**	**.tar**
RHEL for Edge インストーラー	**edge-installer**	**.iso**
RHEL for Edge Raw	**edge-raw-image**	**.tar**
RHEL for Edge Simplified Installer	**edge-simplified-installer**	**.iso**
ISO イメージ	**image-installer**	**.iso**

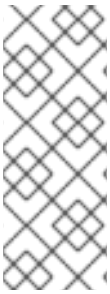
7.1.4. Image Builder のシステム要件

Image Builder を実行する環境 (専用の仮想マシンなど) は、次の表に記載されている要件を満たす必要があります。

表7.2 Image Builder のシステム要件

パラメーター	最低要求値
システムのタイプ	専用の仮想マシン。Image Builder は、Red Hat Universal Base Images (UBI) を含むコンテナーではサポートされていないことに注意してください。
プロセッサー	2 コア
メモリー	4 GiB
ディスク容量	**/var** ファイルシステムに 20 GiB の空き容量

パラメーター	最低要求値
アクセス権限	管理者レベル (root)
ネットワーク	インターネット接続



注記

インターネット接続がない場合は、Red Hat Content Delivery Network (CDN) に接続しないように再設定すれば、隔離されたネットワークで Image Builder を使用できます。そのためには、ローカルリポジトリーを指すようにデフォルトのリポジトリーをオーバーライドする必要があります。コンテンツが内部でミラーリングされていることを確認するか、Red Hat Satellite を使用してください。詳細は、Managing repositories を参照してください。

関連情報

- Provisioning to Satellite using a Red Hat image builder image

7.2. IMAGE BUILDER のインストール

Image Builder は、カスタムのシステムイメージを作成するツールです。Image Builder を使用する前に、仮想マシンで Image Builder をインストールする必要があります。

7.2.1. Image Builder のシステム要件

Image Builder を実行する環境 (専用の仮想マシンなど) は、次の表に記載されている要件を満たす必要があります。

表7.3 Image Builder のシステム要件

パラメーター	最低要求値
システムのタイプ	専用の仮想マシン。Image Builder は、Red Hat Universal Base Images (UBI) を含むコンテナーではサポートされていないことに注意してください。
プロセッサー	2 コア
メモリー	4 GiB
ディスク容量	**/var** ファイルシステムに 20 GiB の空き容量
アクセス権限	管理者レベル (root)
ネットワーク	インターネット接続

注記

インターネット接続がない場合は、Red Hat Content Delivery Network (CDN) に接続しないように再設定すれば、隔離されたネットワークで Image Builder を使用できます。そのためには、ローカルリポジトリーを指すようにデフォルトのリポジトリーをオーバーライドする必要があります。コンテンツが内部でミラーリングされていることを確認するか、Red Hat Satellite を使用してください。詳細は、Managing repositories を参照してください。

関連情報

- Provisioning to Satellite using a Red Hat image builder image

7.2.2. 仮想マシンへの Image Builder のインストール

Image Builder を専用の仮想マシン (VM) にインストールするには、以下の手順を行います。

前提条件

- RHEL VM に接続している。
- Image Builder 用の VM が実行中であり、Red Hat Subscription Manager (RHSM) または Red Hat Satellite にサブスクライブしている必要があります。

手順

1. Image Builder とその他の必要なパッケージを VM にインストールします。
   - **osbuild-composer** - RHEL 8.3 以降でサポート
   - **composer-cli**
   - **cockpit-composer**
   - **bash-completion**

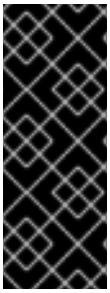
   ```
   # yum install osbuild-composer composer-cli cockpit-composer bash-completion
   ```

   Web コンソールは、**cockpit-composer** パッケージの依存関係としてインストールされます。
2. システムを再起動するたびに、Image Builder が起動するようにします。

   ```
   # systemctl enable --now osbuild-composer.socket
   # systemctl enable --now cockpit.socket
   ```

   osbuild-composer および **cockpit** サービスは、最初のアクセスで自動的に起動します。
3. システムを再起動しなくても、**composer-cli** コマンドのオートコンプリート機能がすぐに動作するように、シェル設定スクリプトを読み込みます。

   ```
   $ source /etc/bash_completion.d/composer-cli
   ```



重要

osbuild-composer パッケージは、Red Hat Enterprise Linux 8.3 以降の新機能すべてに焦点を当てた新しいバックエンドエンジンで、デフォルト設定として推奨されています。以前のバックエンドの **lorax-composer** は非推奨となり、Red Hat Enterprise Linux 8 ライフサイクルの残りの期間、一部の修正のみを受信し、今後のメジャーリリースから削除される予定です。osbuild-composer を優先するには、**lorax-composer** のアンインストールを推奨します。

検証

システムジャーナルを使用して、Image Builder サービスアクティビティーを追跡できます。さらに、ファイル内のログメッセージを見つけることができます。

- トレースバックのジャーナル出力を見つけるには、次のコマンドを実行します。

```
$ journalctl | grep osbuild
```

- リモートワーカーとローカルワーカーの両方を表示するには:

```
$ journalctl -u osbuild-worker*
```

- 実行中のサービスを表示するには:

```
$ journalctl -u osbuild-composer.service
```

7.2.3. lorax-composer Image Builder バックエンドに戻す手順

osbuild-composer バックエンドは、はるかに拡張性が高くなっていますが、現時点では以前の **lorax-composer** バックエンドとの機能パリティーがありません。

以前のバックエンドに戻すには、以下の手順に従います。

前提条件

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

手順

1. osbuild-composer バックエンドを削除します。

```
# yum remove osbuild-composer
# yum remove weldr-client
```

2. **/etc/yum.conf** ファイルで、**osbuild-composer** パッケージの除外エントリーを追加します。

```
# cat /etc/yum.conf
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
```

```
best=True
skip_if_unavailable=False
exclude=osbuild-composer weldr-client
```

3. **lorax-composer** パッケージをインストールします。

```
# yum install lorax-composer composer-cli
```

4. **lorax-composer** サービスを有効にして開始し、再起動するたびに開始します。

```
# systemctl enable --now lorax-composer.socket
# systemctl start lorax-composer
```

関連情報

- Red Hat サポートでケースを作成 します。

7.3. IMAGE BUILDER コマンドラインインターフェイスを使用したシステムイメージの作成

Image Builder は、カスタムのシステムイメージを作成するツールです。Image Builder を制御し、カスタムシステムイメージを作成するには、コマンドラインインターフェイス (CLI) または Web コンソールインターフェイスを使用できます。ただし、現在、Image Builder を使用するには CLI が推奨されます。

7.3.1. Image Builder コマンドラインインターフェイスの紹介

現在、Image Builder コマンドラインインターフェイス (CLI) は、Image Builder を使用するための推奨される方法です。Web コンソールインターフェイスよりも多くの機能を提供します。CLI を使用するには、適切なオプションとサブコマンドを指定して **composer-cli** コマンドを実行します。

コマンドラインインターフェイスのワークフローの概要は次のようになります。

1. 平文テキストファイルにブループリント定義をエクスポート (**保存**) する。
2. テキストエディターでこのファイルを編集する。
3. Image Builder でブループリントのテキストファイルをインポート (**プッシュ**) する。
4. compose を実行して、ブループリントからイメージを構築する。
5. イメージファイルをエクスポートして、ダウンロードする。

この手順を実行する基本的なサブコマンドとは別に、**composer-cli** コマンドには、設定したブループリントと Compose の状態を調べるサブコマンドが多数あります。

composer-cli コマンドを非 root として実行するには、ユーザーは、 **weldr** または **root** グループに属している必要があります。

- ユーザーを **weldr** または **root** グループに追加するには、次のコマンドを実行します:

```
$ sudo usermod -a -G weldr user
$ newgrp weldr
```

7.3.2. コマンドラインインターフェイスを使用した Image Builder ブループリントの作成

コマンドラインインターフェイス (CLI) を使用して、新しい Image Builder ブループリントを作成できます。ブループリントには、最終的なイメージと、パッケージやカーネルのカスタマイズなどのそのカスタマイズが記述されています。

前提条件

- イメージビルダーツールへのアクセス。

手順

1. 以下の内容で平文テキストファイルを作成します。

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "0.0.1"
modules = []
groups = []
```

BLUEPRINT-NAME および **LONG FORM DESCRIPTION TEXT** は、ブループリントの名前および説明に置き換えます。

Semantic Versioning スキームに従って、**0.0.1** をバージョン番号に置き換えます。

2. ブループリントに含まれるすべてのパッケージに、次の行をファイルに追加します。

```
[[packages]]
name = "package-name"
version = "package-version"
```

package-name は、パッケージ名 (**httpd**、**gdb-doc**、**coreutils** など) に置き換えます。

package-version を使用するバージョンに置き換えます。このフィールドは、**dnf** バージョンの指定に対応します。

- 特定のバージョンについては、**8.7.0** などの正確なバージョン番号を使用してください。
- 利用可能な最新バージョンについては、アスタリスク * を使用してください。
- 最新のマイナーバージョンを指定する場合は、**8.*** などの形式を使用してください。

3. ニーズに合わせてブループリントをカスタマイズします。たとえば、Simultaneous Multi Threading (SMT) を無効にするには、ブループリントファイルに次の行を追加します。

```
[customizations.kernel]
append = "nosmt=force"
```

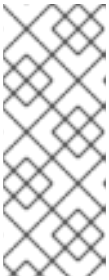
その他に利用できるカスタマイズについては、サポートされているイメージのカスタマイズ を参照してください。

4. たとえば、ファイルを **BLUEPRINT-NAME**.toml として保存し、テキストエディターを閉じます。

5. ブループリントをプッシュ (インポート) します。

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

BLUEPRINT-NAME は、前の手順で使用した値に置き換えます。



注記

composer-cli を非 root として使用してイメージを作成するには、ユーザーを **weldr** または **root** グループに追加します。

```
# usermod -a -G weldr user
$ newgrp weldr
```

検証

- ブループリントがプッシュされ存在していることを確認するには、既存のブループリントを一覧表示します。

```
# composer-cli blueprints list
```

- 追加したばかりのブループリント設定を表示します。

```
# composer-cli blueprints show BLUEPRINT-NAME
```

- ブループリントに記載されているコンポーネントおよびバージョンと、その依存関係が有効かどうかを確認します。

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

Image Builder がカスタムリポジトリーからパッケージを解決できない場合は、次の手順に従います。

- osbuild-composer キャッシュを削除します。

```
$ sudo rm -rf /var/cache/osbuild-composer/*
$ sudo systemctl restart osbuild-composer
```

関連情報

- **osbuild-composer** がカスタムリポジトリーからパッケージを depsolve できない
- プロキシーサーバーを使用してカスタマイズされた RHEL システムイメージを作成する

7.3.3. コマンドラインインターフェイスで Image Builder のブループリントの編集

コマンドライン (CLI) インターフェイスで既存の Image Builder ブループリントを編集して、たとえば、新しいパッケージを追加したり、新しいグループを定義したり、カスタマイズしたイメージを作成したりできます。そのためには、以下の手順に従います。

前提条件

- ブループリントを作成している。

手順

1. ローカルのテキストファイルにブループリントを保存 (エクスポート) します。

```
# composer-cli blueprints save BLUEPRINT-NAME
```

2. **BLUEPRINT-NAME** .toml ファイルをテキストエディターで編集し、変更を加えます。
3. 編集を完了する前に、ファイルが有効なブループリントであることを確認します。
   a. 次の行がある場合は削除します。

```
packages = []
```

   b. たとえば、バージョン番号を 0.0.1 から 0.1.0 に増やします。Image Builder のブループリントバージョンは Semantic Versioning スキームを使用する必要があります。また、バージョンを変更しない場合、**パッチ**バージョンコンポーネントが自動的に増加することにも注意してください。

   c. コンテンツが有効な TOML 仕様かどうかを確認します。詳細は、TOML のドキュメント を参照してください。

注記

TOML のドキュメントはコミュニティーが提供しているため、Red Hat のサポート対象外となります。このツールの問題は、https://github.com/toml-lang/toml/issues から報告できます。

4. ファイルを保存し、テキストエディターを閉じます。
5. ブループリントを Image Builder にプッシュ (インポート) します。

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

注記

ブループリントを Image Builder にインポートして戻すには、**.toml** 拡張子を含むファイル名を指定しますが、他のコマンドではブループリント名のみを使用します。

6. Image Builder にアップロードしたコンテンツが編集内容と一致することを確認するには、ブループリントのコンテンツの一覧を表示します。

```
# composer-cli blueprints show BLUEPRINT-NAME
```

7. ブループリントに記載されているコンポーネントおよびバージョンと、その依存関係が有効かどうかを確認します。

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

関連情報

- サポートされているイメージのカスタマイズ

7.3.4. Image Builder コマンドラインインターフェイスでシステムイメージの作成

Image Builder コマンドラインインターフェイスを使用して、カスタムイメージを作成できます。

前提条件

- イメージにブループリントを用意している。コマンドラインインターフェイスを使用した Image Builder ブループリントの作成 を参照してください。

手順

1. Compose を起動します。

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE
```

BLUEPRINT-NAME をブループリントの名前に、**IMAGE-TYPE** をイメージのタイプに置き換えます。使用可能な値は、**composer-cli compose types** コマンドの出力を参照してください。

作成プロセスはバックグラウンドで開始され、作成者の Universally Unique Identifier (UUID) が表示されます。

2. 作成プロセスが完了するまで待ちます。イメージの作成が完了するまでに最大 10 分かかる場合があります。
Compose 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。

```
# composer-cli compose status
```

終了した設定には、**FINISHED** ステータス値が表示されます。リストで設定を識別するには、その UUID を使用します。

3. 作成プロセスが完了したら、結果のイメージファイルをダウンロードします。

```
# composer-cli compose image UUID
```

UUID は、前の手順で示した UUID 値に置き換えます。

検証

イメージを作成したら、次のコマンドを使用してイメージ作成の進行状況を確認できます。

- 作成ステータスを確認します。

```
$ sudo composer-cli compose status
```

- イメージのメタデータをダウンロードします。

```
$ sudo composer-cli compose metadata UUID
```

- イメージのログをダウンロードします。
-

```
$ sudo composer-cli compose logs UUID
```

このコマンドは、イメージ作成のログを含む **.tar** ファイルを作成します。ログが空の場合は、ジャーナルを確認できます。

- ジャーナルを確認してください。

```
$ journalctl | grep osbuild
```

- マニフェストを確認します。

```
$ sudo cat /var/lib/osbuild-composer/jobs/job_UUID.json
```

ジャーナルで **job_UUID** .json を見つけることができます。

関連情報

- Tracing Image Builder

7.3.5. Image Builder コマンドラインの基本的なコマンド

Image Builder コマンドラインインターフェイスでは、以下のサブコマンドを利用できます。

ブループリント操作

利用可能なブループリント一覧の表示

```
# composer-cli blueprints list
```

TOML 形式でブループリントの内容の表示

```
# composer-cli blueprints show BLUEPRINT-NAME
```

TOML 形式のブループリントの内容を BLUEPRINT-NAME.toml ファイルに保存 (エクスポート)

```
# composer-cli blueprints save BLUEPRINT-NAME
```

ブループリントの削除

```
# composer-cli blueprints delete BLUEPRINT-NAME
```

TOML 形式のブループリントファイルを Image Builder へプッシュ (インポート)

```
# composer-cli blueprints push BLUEPRINT-NAME
```

ブループリントでイメージの設定

利用可能なイメージタイプをリスト表示します。

```
# composer-cli compose types
```

Compose の起動

```
# composer-cli compose start BLUEPRINT COMPOSE-TYPE
```

BLUEPRINT は、構築するブループリントの名前に、COMPOSE-TYPE は、出力イメージタイプに置き換えます。

Compose のリスト表示

```
# composer-cli compose list
```

Compose、およびそのステータスのリスト表示

```
# composer-cli compose status
```

実行中の Compose のキャンセル

```
# composer-cli compose cancel COMPOSE-UUID
```

完了した Compose の削除

```
# composer-cli compose delete COMPOSE-UUID
```

Compose の詳細情報の表示

```
# composer-cli compose info COMPOSE-UUID
```

Compose のイメージファイルのダウンロード

```
# composer-cli compose image COMPOSE-UUID
```

サブコマンドとオプションをもっと見る

```
# composer-cli help
```

関連情報

- **composer-cli**(1) man ページ

7.3.6. Image Builder のブループリント形式

Image Builder のブループリントは、TOML 形式のプレーンテキストとしてユーザーに表示されます。

一般的なブループリントファイルの要素には、次のものが含まれます。

ブループリントのメタデータ

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "VERSION"
```

BLUEPRINT-NAME および **LONG FORM DESCRIPTION TEXT** フィールドは、ブループリントの名前と説明です。

VERSION は、セマンティックバージョニング スキームに従ったバージョン番号です。

この部分は、ブループリントファイル全体に対して 1 回だけ存在します。

modules エントリーには、イメージにインストールされるパッケージの名前とバージョンが一覧表示されます。

group エントリーは、イメージにインストールするパッケージのグループを説明します。グループは、次のパッケージカテゴリーを使用します。

- 必須
- デフォルト
- オプション
  ブループリントは、必須パッケージとデフォルトパッケージをインストールします。オプションパッケージを選択するメカニズムはありません。

イメージに追加するグループ

```
[[groups]]
name = "group-name"
```

group-name は、**anaconda-tools**、**widget**、**wheel** または **users** などのグループの名前です。

イメージに追加するパッケージ

```
[[packages]]
name = "package-name"
version = "package-version"
```

package-name は、**httpd**、**gdb-doc**、または **coreutils** などのパッケージの名前です。

package-version は使用するバージョンです。このフィールドは、**dnf** バージョンの指定に対応します。

- 特定のバージョンについては、**8.7.0** などの正確なバージョン番号を使用してください。
- 利用可能な最新バージョンを指定する場合は、アスタリスク (*) を使用します。
- 最新のマイナーバージョンの場合は、**8.*** などの形式を使用します。

追加するすべてのパッケージにこのブロックを繰り返します。

注記

現在、Image Builder ツールのパッケージとモジュールの間に違いはありません。どちらも RPM パッケージの依存関係として扱われます。

7.3.7. サポートされているイメージのカスタマイズ

ブループリントに追加の RPM パッケージを追加するか、サービスを有効にするか、カーネルコマンドラインパラメーターをカスタマイズすることで、イメージをカスタマイズできます。ブループリント内でいくつかのイメージのカスタマイズを使用できます。これらのオプションを利用するには、最初にブループリントでカスタマイズを設定し、それを Image Builder にインポート (プッシュ) する必要があります。

注記

Web コンソールで Image Builder を使用する場合、これらのカスタマイズはサポートされません。

パッケージグループの選択

```
[[packages]]
name = "package_group_name"
```

"**package_group_name**" は、パッケージグループの名前に置き換えます。たとえば、"@server with gui" です。

イメージのホスト名の設定

```
[customizations]
hostname = "baseimage"
```

作成されるシステムイメージに対するユーザー指定

```
[[customizations.user]]
name = "USER-NAME"
description = "USER-DESCRIPTION"
password = "PASSWORD-HASH"
key = "PUBLIC-SSH-KEY"
home = "/home/USER-NAME/"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```

GID はオプションであり、イメージにすでに存在している必要があります。必要に応じて、パッケージで作成するか、ブループリントで **customizations.group** エントリーを使用して GID を作成します。

重要

password hash を生成するには、システムに **python3** をインストールする必要があります。

```
# yum install python3
```

PASSWORD-HASH は、実際の **password hash** に置き換えます。**password hash** を生成するには、次のようなコマンドを使用します。

```
$ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

PUBLIC-SSH-KEY を、実際の公開鍵に置き換えます。

その他のプレースホルダーを、適切な値に置き換えます。

name を入力する必要があります。不要な行は省略できます。

追加するすべてのユーザーにこのブロックを繰り返します。

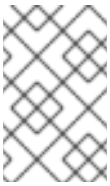
作成されるシステムイメージに対するグループ指定

```
[[customizations.group]]
name = "GROUP-NAME"
gid = NUMBER
```

追加するすべてのグループにこのブロックを繰り返します。

既存ユーザーの SSH 鍵を設定します。

```
[[customizations.sshkey]]
user = "root"
key = "PUBLIC-SSH-KEY"
```



注記

既存のユーザーの SSH キーの設定のカスタマイズは、既存ユーザーにのみ適用されます。ユーザーの作成と SSH キーの設定は、**システムイメージのカスタマイズに関するユーザー仕様** を参照してください。

デフォルトにカーネルの起動パラメーターオプションを追加

```
[customizations.kernel]
append = "KERNEL-OPTION"
```

デフォルトでは、Image Builder はデフォルトのカーネルをイメージにビルドします。ただし、ブループリントで次の設定を使用してカーネルをカスタマイズできます

```
[customizations.kernel]
name = "KERNEL-rt"
```

イメージで使用するカーネル名を定義

```
[customizations.kernel.name]
name = "KERNEL-NAME"
```

作成されたシステムイメージにタイムゾーンおよび Network Time Protocol (NTP) サーバーを設定

```
[customizations.timezone]
timezone = "TIMEZONE"
ntpservers = "NTP_SERVER"
```

- . –

タイムゾーンを設定しないと、システムはデフォルトとして Universal Time, Coordinated (UTC) を使用します。NTP サーバーの設定はオプションです。

作成されたシステムイメージのロケール設定

```
[customizations.locale]
languages = ["LANGUAGE"]
keyboard = "KEYBOARD"
```

言語とキーボードオプションの両方を設定する必要があります。他の多くの言語を追加できます。最初に追加する言語はプライマリー言語で、他の言語はセカンダリーになります。以下に例を示します。

```
[customizations.locale]
languages = ["en_US.UTF-8"]
keyboard = "us"
```

言語でサポートされている値を一覧表示するには、以下のコマンドを実行します。

```
$ localectl list-locales
```

キーボードでサポートされている値を一覧表示するには、以下のコマンドを実行します。

```
$ localectl list-keymaps
```

作成されたシステムイメージのファイアウォールを設定

```
[customizations.firewall]
port = ["PORTS"]
```

リストを有効にするには、数値ポートまたは **/etc/services** ファイルの名前を使用できます。

ファイアウォールサービスのカスタマイズ

利用可能なファイアウォールサービスを確認します。

```
$ firewall-cmd --get-services
```

ブループリントの **customizations.firewall.service** セクションで、カスタマイズするファイアウォールサービスを指定します。

```
[customizations.firewall.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

firewall.services にリストされているサービスは、**/etc/services** ファイルで使用可能なサービス名とは異なります。

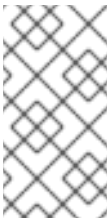
注記

ファイアウォールサービスをカスタマイズしない場合は、ブループリントの **[customizations.firewall]** セクションおよび **[customizations.firewall.services]** セクションを省略します。

システムの起動時に有効にするサービスの設定

```
[customizations.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

システムの起動時に有効にするサービスを制御することができます。一部のイメージタイプでは、イメージが正しく機能し、この設定を上書きできないようにするために、すでにサービスが有効または無効になっています。ブループリントの **customizations.services** カスタマイズは、これらのサービスを置き換えるのではなく、イメージテンプレートにすでに存在するサービスのリストに追加します。



注記

ビルドが開始されるたびに、ホストシステムのリポジトリーのクローンが作成されます。大量の履歴を持つリポジトリーを参照すると、クローンに時間がかかり、大量のディスク領域が使用される場合があります。また、クローンは一時的なものであり、RPM パッケージの作成後にビルドによって削除されます。

カスタムファイルシステム設定を指定します。

ブループリントでカスタムファイルシステム設定を指定できるため、デフォルトのレイアウト設定ではなく、特定のディスクレイアウトでイメージを作成できます。ブループリントでデフォルト以外のレイアウト設定を使用すると、次の利点が得られます。

- セキュリティーベンチマークコンプライアンス
- ディスク外エラーに対する保護
- 改良された性能
- 既存の設定との一貫性
  ブループリントでファイルシステム設定をカスタマイズするには:

  ```
  [[customizations.filesystem]]
  mountpoint = "MOUNTPOINT"
  size = MINIMUM-PARTITION-SIZE
  ```

  ブループリントは、次の **mountpoints** とそのサブディレクトリーをサポートしています。

  - / - ルートマウントポイント
  - /**var**
  - /**home**
  - /**opt**
  - /**srv**/

- **/usr**
- **/app**
- **/data**
- **/boot** - RHEL 8.7 および RHEL 9.1 以降でサポートされています。

注記

マウントポイントのカスタマイズは、CLI を使用することで、RHEL 8.5 および RHEL 9.0 ディストリビューション以降でのみサポートされます。以前のディストリビューションでは、**root** パーティションをマウントポイントとして指定し、size 引数をイメージ **size** のエイリアスとして指定することしかできません。

カスタマイズされたイメージに複数のパーティションがある場合、LVM でカスタマイズされたファイルシステムパーティションを使用してイメージを作成し、実行時にそれらのパーティションのサイズを変更できます。これを行うには、ブループリントでカスタマイズされたファイルシステム設定を指定して、目的のディスクレイアウトでイメージを作成します。デフォルトのファイルシステムレイアウトは変更されません。ファイルシステムをカスタマイズせずにプレーンイメージを使用すると、ルートパーティションは **cloud-init** によってサイズ変更されます。

注記

8.6 以降、**osbuild-composer-46.1-1.el8** RPM 以降のバージョンでは、物理パーティションは使用できなくなり、ファイルシステムのカスタマイズによって論理ボリュームが作成されます。

ブループリントは、ファイルシステムのカスタマイズを LVM パーティションに自動的に変換します。

MINIMUM-PARTITION-SIZE 値には、デフォルトのサイズ形式はありません。ブループリントのカスタマイズでは、kB から TB、および KiB から TiB の値と単位がサポートされています。たとえば、マウントポイントのサイズをバイト単位で定義できます。

```
[[customizations.filesystem]]
mountpoint = "/var"
size = 1073741824
```

単位を使用してマウントポイントのサイズを定義することもできます。

注記

RHEL 8.6 および RHEL 9.0 ディストリビューション以降に提供されているパッケージバージョンの単位を使用してのみ、マウントポイントサイズを定義できます。

以下に例を示します。

```
[[customizations.filesystem]]
mountpoint = "/opt"
```

```
size = "20 GiB"

or

[[customizations.filesystem]]
mountpoint = "/boot"
size = "1 GiB"
```

関連情報

- ファイルシステムのカスタマイズサイズを追加した後、ブループリントのインポートが失敗します。

7.3.8. Image Builder によってインストールされるパッケージ

Image Builder を使用してシステムイメージを作成すると、システムはベースパッケージのセットをインストールします。デフォルトでは、Image Builder は **Core** グループをパッケージの基本リストとして使用します。

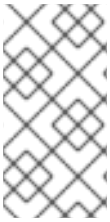
表7.4 イメージタイプの作成をサポートするデフォルトパッケージ

イメージタイプ	デフォルトパッケージ
ami	**checkpolicy、chrony、cloud-init、cloud-utils-growpart、@Core、dhcp-client、gdisk、insights-client、kernel、langpacks-en、net-tools、NetworkManager、redhat-release、redhat-release-eula、rng-tools、rsync、selinux-policy-targeted、tar、yum-utils**
openstack	**@core、langpacks-en**
qcow2	**@core, chrony, dnf, kernel, yum, nfs-utils, dnf-utils, cloud-init, python3-jsonschema, qemu-guest-agent, cloud-utils-growpart, dracut-norescue, tar, tcpdump, rsync, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhnsd, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release, redhat-release-eula, rng-tools, insights-client**
tar	**policycoreutils、selinux-policy-targeted**
vhd	**@core、langpacks-en**
vmdk	**@core、chrony、cloud-init、firewalld、langpacks-en、open-vm-tools、selinux-policy-targeted**

イメージタイプ	デフォルトパッケージ
edge-commit	attr、audit、basesystem、bash、bash-completion、chrony、clevis、clevis-dracut、clevis-luks、container-selinux、coreutils、criu、cryptsetup、curl、dnsmasq、dosfstools、dracut-config-generic、dracut-network、e2fsprogs、firewalld、fuse-overlayfs、fwupd、glibc、glibc-minimal-langpack、gnupg2、greenboot、gzip、hostname、ima-evm-utils、iproute、iptables、iputils、keyutils、less、lvm2、NetworkManager、NetworkManager-wif、NetworkManager-wwan、nss-altfiles、openssh-clients、openssh-server、passwd、pinentry、platform-python、podman、policycoreutils、policycoreutils-python-utils、polkit、procps-ng、redhat-release、rootfiles、rpm、rpm-ostree、rsync、selinux-policy-targeted、setools-console、setup、shadow-utils、shadow-utils、skopeo、slirp4netns、sudo、systemd、tar、tmux、traceroute、usbguard、util-linux、vim-minimal、wpa_supplicant、xz
edge-container	dnf、dosfstools、e2fsprogs、glibc、lorax-templates-generic、lorax-templates-rhel、lvm2、policycoreutils、python36、python3-iniparse、qemu-img、selinux-policy-targeted、systemd、tar、xfsprogs、xz

イメージタイプ	デフォルトパッケージ
edge-installer	**aajohan-comfortaa-fonts、abattis-cantarell-fonts、alsa-firmware、alsa-tools-firmware、anaconda、anaconda-install-env-deps、anaconda-widgets、audit、bind-utils、bitmap-fangsongti-fonts、bzip2、cryptsetup、dbus-x11、dejavu-sans-fonts、dejavu-sans-mono-fonts、device-mapper-persistent-data、dnf、dump、ethtool、fcoe-utils、ftp、gdb-gdbserver、gdisk、gfs2-utils、glibc-all-langpacks、google-noto-sans-cjk-ttc-fonts、gsettings-desktop-schemas、hdparm、hexedit、initscripts、ipmitool、iwl3945-firmware、iwl4965-firmware、iwl6000g2a-firmware、iwl6000g2b-firmware、jomolhari-fonts、kacst-farsi-fonts、kacst-qurn-fonts、kbd、kbd-misc、kdump-anaconda-addon、khmeros-base-fonts、libblockdev-lvm-dbus、libertas-sd8686-firmware、libertas-sd8787-firmware、libertas-usb8388-firmware、libertas-usb8388-olpc-firmware、libibverbs、libreport-plugin-bugzilla、libreport-plugin-reportuploader、libreport-rhel-anaconda-bugzilla、librsvg2、linux-firmware、lklug-fonts、lldpad、lohit-assamese-fonts、lohit-bengali-fonts、lohit-devanagari-fonts、lohit-gujarati-fonts、lohit-gurmukhi-fonts、lohit-kannada-fonts、lohit-odia-fonts、lohit-tamil-fonts、lohit-telugu-fonts、lsof、madan-fonts、metacity、mtr、mt-st、net-tools、nmap-ncat、nm-connection-editor、nss-tools、openssh-server、oscap-anaconda-addon、pciutils、perl-interpreter、pigz、python3-pyatspi、rdma-core、redhat-release-eula、rpm-ostree、rsync、rsyslog、sg3_utils、sil-abyssinica-fonts、sil-padauk-fonts、sil-scheherazade-fonts、smartmontools、smc-meera-fonts、spice-vdagent、strace、system-storage-manager、thai-scalable-waree-fonts、tigervnc-server-minimal、tigervnc-server-module、udisks2、udisks2-iscsi、usbutils、vim-minimal、volume_key、wget、xfsdump、xorg-x11-drivers、xorg-x11-fonts-misc、xorg-x11-server-utils、xorg-x11-server-Xorg、xorg-x11-xauth**

イメージタイプ	デフォルトパッケージ
edge-simplified-installer	**attr、basesystem、binutils、bsdtar、clevis-dracut、clevis-luks、cloud-utils-growpart、coreos-installer、coreos-installer-dracut、coreutils、device-mapper-multipath、dnsmasq、dosfstools、dracut-live、e2fsprogs、fcoe-utils、fdo-init、gzip、ima-evm-utils、iproute、iptables、iputils、iscsi-initiator-utils、keyutils、lldpad、lvm2、passwd、policycoreutils、policycoreutils-python-utils、procps-ng、rootfiles、setools-console、sudo、traceroute、util-linux**
image-installer	**anaconda-dracut、curl、dracut-config-generic、dracut-network、hostname、iwl100-firmware、iwl1000-firmware、iwl105-firmware、iwl135-firmware、iwl2000-firmware、iwl2030-firmware、iwl3160-firmware、iwl5000-firmware、iwl5150-firmware、iwl6000-firmware、iwl6050-firmware、iwl7260-firmware、kernel、less、nfs-utils、openssh-clients、ostree、plymouth、prefixdevname、rng-tools、rpcbind、selinux-policy-targeted、systemd、tar、xfsprogs、xz**
edge-raw-image	**dnf、dosfstools、e2fsprogs、glibc、lorax-templates-generic、lorax-templates-rhel、lvm2、policycoreutils、python36、python3-iniparse、qemu-img、selinux-policy-targeted、systemd、tar、xfsprogs、xz**
gce	**@core、langpacks-en、acpid、dhcp-client、dnf-automatic、net-tools、python3、rng-tools、tar、vim**



注記

ブループリントにコンポーネントを追加する場合は、追加したコンポーネント内のパッケージが他のパッケージコンポーネントと競合しないようにしてください。そうしないと、システムは依存関係を解決できず、カスタマイズされたイメージの作成に失敗します。次のコマンドを実行して、パッケージ間に競合がないかどうかを確認できます。

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

関連情報

- Image Builder の説明

7.3.9. カスタムイメージで有効なサービス

Image Builder を使用してカスタムイメージを設定する場合、イメージが使用するデフォルトのサービスは次のように決定されます。

- **osbuild-composer** ユーティリティーを使用する RHEL リリース
- イメージの種類

たとえば、**ami** イメージタイプは、デフォルトで **sshd**、**chronyd**、および **cloud-init** サービスを有効にします。これらのサービスが有効になっていない場合、カスタムイメージは起動しません。

表7.5 イメージタイプの作成をサポートするために有効になっているサービス

イメージタイプ	デフォルトで有効化されているサービス
ami	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
openstack	sshd, cloud-init, cloud-init-local, cloud-config, cloud-final
qcow2	cloud-init
rhel-edge-commit	デフォルトでは、追加のサービスは有効になりません。
tar	デフォルトでは、追加のサービスは有効になりません。
vhd	sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final
vmdk	sshd、chronyd、vmtoolsd、cloud-init

注記:システムの起動時に有効にするサービスをカスタマイズできます。ただし、カスタマイズは、前述のイメージタイプに対してデフォルトで有効になっているサービスを上書きしません。

関連情報

- サポートされているイメージのカスタマイズ

7.4. IMAGE BUILDER WEB コンソールインターフェイスを使用したシステムイメージの作成

Image Builder は、カスタムのシステムイメージを作成するツールです。Image Builder を制御してカスタムシステムイメージを作成する場合は、Web コンソールインターフェイスを使用できます。ただし、コマンドラインインターフェイスの方が提供している機能が多いため、コマンドラインインターフェイスを使用することが推奨されます。

7.4.1. RHEL Web コンソールでの Image Builder GUI へのアクセス

RHEL Web コンソール用の **cockpit-composer** プラグインを使用すると、グラフィカルインターフェイスを使用してイメージビルダーのブループリントと設定を管理できます。Image Builder を制御するための推奨される方法は、コマンドラインインターフェイスです。

前提条件

- システムへの root アクセス権限がある。
- Image Builder をインストールしました。

手順

1. Image Builder がインストールされている Web ブラウザーで **https://localhost:9090/** を開きます。
   Image Builder にリモートでアクセスする方法の詳細は、**Managing systems using the RHEL web console** を参照してください。
2. root ユーザーとして Web コンソールにログインします。
3. Image Builder コントロールを表示するには、ウィンドウの左上にある **Image Builder** アイコンをクリックします。
   Image Builder ビューが開き、既存のブループリントの一覧が表示されます。

7.4.2. Web コンソールインターフェイスで Image Builder のブループリントの作成

ブループリントの作成は、カスタマイズされたシステムイメージを説明する前に必要な手順です。

前提条件

- ブラウザーの Web コンソールから Image Builder アプリケーションを開いた。RHEL Web コンソールでの Image Builder GUI へのアクセス を参照してください。

手順

1. 右上隅にある **Create Blueprint** をクリックします。
   ブループリントの名前と説明のフィールドを含むダイアログウィザードが開きます。
2. ブループリントの名前と、必要に応じてその説明を入力します。
3. **Create** をクリックします。

Image Builder ビューが開き、既存のブループリントの一覧が表示されます。

7.4.3. Web コンソールインターフェイスで Image Builder を使用してシステムイメージを作成する

次の手順を実行して、ブループリントからシステムイメージを作成できます。

前提条件

- ブラウザーの Web コンソールから Image Builder アプリを開いている。
- ブループリントを作成している。

手順

1. ブループリントに **Back to blueprints** をクリックして、ブループリントテーブルを表示します。

2. ブループリントテーブルで、イメージを構築するブループリントを見つけます。

   a. 必要に応じて、検索ボックスを使用してブループリントを見つけることができます。ブループリント名を入力します。

3. 選択したブループリントの右側で、**Create Image** をクリックします。Create image ダイアログウィザードが開きます。

4. Image output ページで、次の手順を実行します。

   a. **イメージ出力タイプ** リストから、目的のイメージタイプを選択します。

      i. 一部のイメージは、Amazon Webservice や Oracle Cloud Infrastructure などのターゲットクラウド環境にアップロードできます。そのためには、**ターゲットクラウドにアップロード** ボックスをオンにします。

      ii. 次のページで、クラウド環境の認証情報を追加するよう求められます。

5. Image Size フィールドから、イメージサイズを入力します。最小サイズはイメージの種類によって異なります。**Next** をクリックします。

6. Targeted_Cloud **にアップロード** ページで、次の手順を実行します。
   注記: このページは、イメージをクラウド環境にアップロードするボックスをチェックしていない場合は表示されません。

   a. **認証** ページで、ターゲットクラウドアカウント ID に関連する情報を入力し、**次へ** をクリックします。

   b. **宛先ページ** で、ターゲットクラウドアカウントの種類に関連する情報を入力し、**次へ** をクリックします。

7. **カスタマイズ** ページで、次の手順を完了します。

   a. **システム** ページで、ホスト名を入力します。ホスト名を入力しない場合、オペレーティングシステムがシステムのホスト名を決定します。

   b. **ユーザー** ページで、ユーザーの **追加** をクリックします。

      i. 必須:ユーザー名を入力。

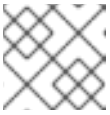
      ii. パスワードを入力します。

      iii. SSH キーを入力します。

      iv. ユーザーをサーバー管理者にする場合は、チェックボックスをオンにします。**Next** をクリックします。

8. **パッケージ** ページで、次の手順を完了します。

   a. **使用可能なパッケージ** 検索フィールドで、システムイメージに追加するパッケージ名を入力します。



注記

パッケージの検索が完了するまでに時間がかかる場合があります。

b. > 矢印をクリックして、選択したパッケージを追加します。**Next** をクリックします。

9. **確認** ページで、イメージの作成に関する詳細を確認します。**Save Blueprint** をクリックして、ブループリントに追加したカスタマイズを保存します。**Create image** をクリックします。イメージのビルドが開始され、完了するまでに最大 20 分かかります。

7.5. IMAGE BUILDER を使用したクラウドイメージの準備とアップロード

Image Builder は、さまざまなクラウドプラットフォームですぐに使用できるカスタムシステムイメージを作成できます。カスタマイズした RHEL システムイメージをクラウドで使用するには、各出力タイプを使用して Image Builder でシステムイメージを作成し、イメージをアップロードするようにシステムを設定し、クラウドアカウントへイメージをアップロードします。RHEL Web コンソールの **image builder** アプリケーションを介して、カスタマイズされたイメージクラウドをプッシュできます。これは、**AWS** や **Microsoft Azure** クラウドなど、サポートされているサービスプロバイダーのサブセットで利用できます。イメージを AWS クラウド AMI にプッシュする および VHD イメージを Microsoft Azure クラウドにプッシュするを 参照してください。

7.5.1. AWS AMI イメージのアップロードの準備

AWS AMI イメージをアップロードする前に、イメージをアップロードするためのシステムを設定する必要があります。

前提条件

- AWS IAM アカウントマネージャー にアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

手順

1. Python 3 および **pip** ツールをインストールします。

```
# yum install python3
# yum install python3-pip
```

2. pip で **AWS コマンドラインツール** をインストールします。

```
# pip3 install awscli
```

3. 以下のコマンドを実行してプロファイルを設定します。ターミナルで、認証情報、リージョン、および出力形式を指定するように求められます。

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

4. バケット名を定義し、以下のコマンドを使用してバケットを作成します。

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

bucketname は、バケット名に置き換えます。この名前は、グローバルで一意となるように指定する必要があります。上記で、バケットが作成されます。

5. S3 バケットへのアクセス許可を付与するには、AWS Identity and Access Management (IAM) で **vmimport** S3 ロールを作成します (まだ作成していない場合)。

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{ "sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[ "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s", "arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute", "ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET $BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document file://role-policy.json
```

関連情報

- Using high-level (s3) commands with the AWS CLI

7.5.2. CLI を使用して AMI イメージを AWS にアップロードする

Image Builder を使用して **ami** イメージを構築し、CLI を使用してそれらを Amazon AWS Cloud サービスプロバイダーに直接プッシュできます。

前提条件

- AWS IAM アカウントマネージャーに **Access Key ID** を設定している。
- 書き込み可能な S3 バケット を準備している。
- 定義済みの青写真がある。

手順

1. テキストエディターを使用して、次の内容の設定ファイルを作成します。

```
provider = "aws"

[settings]
accessKeyID = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

フィールドの値を **accessKeyID**、**secretAccessKey**、**bucket**、および **region** の認証情報に置き換えます。**IMAGE_KEY** 値は、EC2 にアップロードされる VM イメージの名前です。

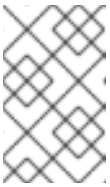
2. ファイルを **CONFIGURATION-FILE**.toml として保存し、テキストエディターを閉じます。

3. Compose を起動します。

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY
CONFIGURATION-FILE.toml
```

以下を置き換えます。

- **BLUEPRINT-NAME** は作成したブループリントの名前です。
- **IMAGE-TYPE**と **ami** イメージタイプ。
- EC2 にアップロードする VM イメージの名前を含む **IMAGE_KEY**。
- クラウドプロバイダーの設定ファイルの名前を持つ **CONFIGURATION-FILE**.toml。



注記

カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。イメージをアップロードする前にバケットにポリシーを設定しておく必要があります。

4. イメージビルドのステータスを確認し、AWS にアップロードします。

```
# composer-cli compose status
```

イメージのアップロードプロセスが完了すると、FINISHED ステータスが表示されます。

検証

イメージのアップロードが成功したことを確認するには、以下を行います。

1. メニューで EC2 にアクセスし、AWS コンソールで正しいリージョンを選択します。イメージが正常にアップロードされたことを示すには、イメージが **available** ステータスになっている必要があります。

2. ダッシュボードでイメージを選択し、**Launch** をクリックします。

関連情報

- 仮想マシンのインポートに必要なサービスロール

7.5.3. イメージの AWS Cloud AMI へのプッシュ

作成した出力イメージを **Amazon AWS Cloud AMI**サービスプロバイダーに直接プッシュできます。

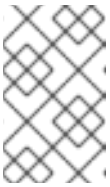
前提条件

- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェイスを開いている。
- ブループリントを作成している。Web コンソールインターフェイスで Image Builder のブループリントの作成 を参照してください。

- AWS IAM アカウントマネージャーにアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

手順

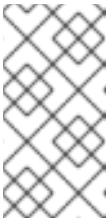
1. **ブループリント名** をクリックします。
2. **イメージ** タブを選択します。
3. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
   ポップアップウィンドウが開きます。
   a. **Type** ドロップダウンメニューから、**Amazon Machine Image Disk (.raw)** を選択します。
   b. **Upload to AWS** チェックボックスをチェックして、イメージを AWS Cloud にアップロードし、**Next** をクリックします。
   c. AWS へのアクセスを認証するには、対応するフィールドに A**AWS access key ID** および **AWS secret access key** と入力します。**Next** をクリックします。

   注記

   新規アクセスキー ID を作成する場合にのみ、AWS シークレットアクセスキーを表示できます。秘密鍵が分からない場合は、新しいアクセスキー ID を生成します。

   d. **Image name** フィールドにイメージ名を、**Amazon S3 bucket name** フィールドに Amazon バケット名を入力して、カスタマイズイメージを追加するバケットの **AWS region** フィールドを入力します。**Next** をクリックします。
   e. 情報を確認し、**Finish** をクリックします。
   必要に応じて、**戻る** をクリックして、誤った情報を変更できます。

   注記

   カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。この手順では IAM のインポートとエクスポートを使用するため、バケットにイメージをアップロードする前にバケットに **ポリシー** を設定する必要があります。詳細は、IAM ユーザーの必要なパーミッション を参照してください。

4. 右上の小さなポップアップで、保存の進行状況が通知されます。また、イメージの作成、イメージ作成の進捗、およびそれ以降の AWS Cloud にアップロードに関する情報も通知されます。
   プロセスが完了すると、**Image build complete** のステータスが表示されます。
5. メニューで Service→EC2 をクリックし、AWS コンソールで 正しいリージョン を選択します。
   イメージのステータスは、アップロードされていることを示す **Available** でなければなりません。
6. ダッシュボードでイメージを選択し、**Launch** をクリックします。
7. 新しいウィンドウが開きます。イメージを開始するために必要なリソースに応じて、インスタンスタイプを選択します。**Review and Launch** をクリックします。

8. インスタンスの開始の詳細を確認します。変更が必要な場合は、各セクションを編集できます。**Launch** をクリックします。

9. インスタンスを起動する前に、インスタンスにアクセスするための公開鍵を選択します。既存のキーペアを使用するか、キーペアーを新規作成します。**Image Builder** を使用して、既存の公開鍵でイメージにユーザーを追加します。詳細は SSH キーを使用したユーザーアカウントの作成 を参照してください。

   次の手順に従って、EC2 で新規キーペアを作成し、新規インスタンスにアタッチします。

   a. ドロップダウンメニューリストから、**Create a new key pair**を選択します。

   b. 新しいキーペアに名前を入力します。新しいキーペアが生成されます。

   c. **Download Key Pair** をクリックして、新しいキーペアをローカルシステムに保存します。

10. 次に、**Launch Instance** をクリックしてインスタンスを起動できます。**Initializing** と表示されるインスタンスのステータスを確認できます。

11. インスタンスのステータスが **running** になると、**Connect** ボタンが有効になります。

12. **Connect** をクリックします。ポップアップウィンドウが表示され、SSH を使用して接続する方法の説明が表示されます。

    a. 優先する接続方法として **スタンドアロン SSH クライアント** を選択し、ターミナルを開きます。

    b. 秘密鍵を保存する場所で、SSH が機能するために鍵が公開されていることを確認してください。これには、以下のコマンドを実行します。

    ```
    $ chmod 400 <your-instance-name.pem>_
    ```

    c. パブリック DNS を使用してインスタンスに接続します。

    ```
    $ ssh -i "<_your-instance-name.pem_"> ec2-user@<_your-instance-IP-address_>
    ```

    d. **yes** と入力して、接続の続行を確定します。
    これで、SSH でインスタンスに接続されました。

検証

1. SSH でインスタンスに接続している間にアクションが実行できるかどうかを確認します。

関連情報

- Red Hat カスタマーポータルでのケースの作成
- SSH を使用した Linux インスタンスへの接続
- サポートケースの作成

7.5.4. Microsoft Azure VHD イメージをアップロードする準備をしています

Image Builder を使用して、**Microsoft Azure** クラウドにアップロードできる VHD イメージを準備できます。

前提条件

- 使用可能な Microsoft Azure リソースグループとストレージアカウントが必要です。
- **AZ CLI** ツールは特に python 2.7 に依存しているため、python2 がインストールされています。

手順

1. Microsoft リポジトリーキーをインポートします。

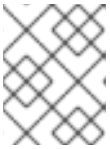
```
# rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. ローカルの **azure-cli** リポジトリー情報を作成します。

```
# sh -c 'echo -e "[azure-cli]\nname=Azure
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

3. Microsoft Azure CLI をインストールします。

```
# yumdownloader azure-cli
# rpm -ivh --nodeps azure-cli-2.0.64-1.el7.x86_64.rpm
```



注記

Microsoft Azure CLI パッケージのダウンロードバージョンは、現在利用可能なバージョンによって異なる場合があります。

4. Microsoft Azure CLI を実行します。

```
$ az login
```

ターミナルに次のメッセージが表示されます。**Note, we have launched a browser for you to login.For old experience with device code, use "az login --use-device-code**次に、ターミナルは、ログインできる場所から https://microsoft.com/devicelogin へのリンクのあるブラウザーを開きます。

注記

リモート (SSH) セッションを実行している場合、https://microsoft.com/devicelogin リンクはブラウザーで開きません。この場合、リンクをブラウザーにコピーしてログインし、リモートセッションを認証できます。サインインするには、Web ブラウザーを使用してページ https://microsoft.com/devicelogin を開き、デバイスコードを入力して認証します。

5. Microsoft Azure のストレージアカウントのキーをリスト表示します。

```
$ GROUP=resource-group-name
$ ACCOUNT=storage-account-name
$ az storage account keys list --resource-group $GROUP --account-name $ACCOUNT
```

resource-group-name を Microsoft Azure リソースグループの名前に置き換え、**storage-account-name** を Microsoft Azure ストレージアカウントの名前に置き換えます。

注記

次のコマンドを使用して、使用可能なリソースを一覧表示できます。

```
$ az resource list
```

6. 上記コマンドの出力の **key1** の値を書き留め、それを環境変数に割り当てます。

```
$ KEY1=value
```

7. ストレージコンテナーを作成します。

```
$ CONTAINER=storage-account-name
$ az storage container create --account-name $ACCOUNT \
--account-key $KEY1 --name $CONTAINER
```

storage-account-name は、ストレージアカウント名に置き換えます。

関連情報

- Microsoft Azure CLI.

7.5.5. Microsoft Azure クラウドへの VHD イメージのアップロード

カスタマイズした VHD イメージを作成したら、それを Microsoft Azure クラウドにアップロードできます。

前提条件

- Microsoft Azure VHD イメージをアップロードするには、システムをセットアップする必要があります。Microsoft Azure VHD イメージのアップロードの準備 を参照してください。
- Image Builder によって作成された Microsoft Azure VHD イメージが必要です。
  - CLI で、**vhd** 出力タイプを使用します。
  - GUI で、**Azure Disk Image (.vhd)** イメージタイプを使用します。

手順

1. イメージを Microsoft Azure にプッシュし、そこからインスタンスを作成します。

```
$ VHD=25ccb8dd-3872-477f-9e3d-c2970cd4bbaf-disk.vhd
$ az storage blob upload --account-name $ACCOUNT --container-name
$CONTAINER --file $VHD --name $VHD --type page
...
```

2. Microsoft Azure Blob ストレージへのアップロードが完了したら、そこから Microsoft Azure イメージを作成します。

```
$ az image create --resource-group $GROUP --name $VHD --os-type linux --
location eastus --source
https://$ACCOUNT.blob.core.windows.net/$CONTAINER/$VHD
 - Running ...
```

検証

1. Microsoft Azure ポータル、または以下のようなコマンドを使用して、インスタンスを作成します。

```
$ az vm create --resource-group $GROUP --location eastus --name $VHD --image $VHD --
admin-username azure-user --generate-ssh-keys
 - Running ...
```

2. 秘密鍵を使用して、SSH 経由で、作成されたインスタンスにアクセスします。**azure-user** としてログインします。

関連情報

- **.vhd** 形式のイメージの作成に失敗する

7.5.6. VMDK イメージのアップロードと vSphere での RHEL 仮想マシンの作成

CLI ツール **govc import.vmdk** を使用して、VMware vSphere に .vmdk イメージをアップロードします。

注記

UI を介したイメージのアップロードはサポートされていません。

前提条件

- ユーザー名とパスワードをカスタマイズして、ブループリントを作成しました。
- Image Builder を使用して、**.vmdk** イメージを作成し、ホストシステムにダウンロードしました。
- CLI ツール **govc import.vmdk** をインストールしました。
- CLI ツールクライアント **govc import.vmdk** を設定しました。
  - 環境に次の値を設定する必要があります。

```
GOVC_URL
GOVC_DATACENTER
GOVC_FOLDER
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

手順

1. **.vmdk** イメージをダウンロードしたディレクトリーに移動します。

2. 次の手順に従って、vSphere でイメージを起動します。

   a. **.vmdk** イメージを vSphere にインポートします。

   ```
   $ govc import.vmdk ./composer-api.vmdk foldername
   ```

   b. 電源をオンにせずに VSphere に仮想マシンを作成します。

   ```
   govc vm.create \
   -net.adapter=vmxnet3 \
   -m=4096 -c=2 -g=rhel8_64Guest \
   -firmware=efi -disk="foldername/composer-api.vmdk" \
   -disk.controller=scsi -on=false \
    vmname
   ```

   c. 仮想マシンの電源をオンにします。

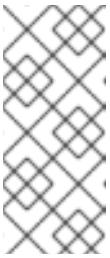
   ```
   govc vm.power -on vmname
   ```

   d. 仮想マシンの IP アドレスを取得します。

   ```
   HOST=$(govc vm.ip vmname)
   ```

   e. ブループリントで指定したユーザー名とパスワードで、SSH を使用して、VM にログインします。

   ```
   $ ssh admin@HOST
   ```



注記

govc datastore.upload コマンドを使用して、ローカルホストから宛先に **.vmdk** イメージをコピーした場合、そのイメージの使用はサポートされていません。vSphere GUI で **import.vmdk** コマンドを使用するオプションがないため、vSphere GUI は直接アップロードをサポートしていません。その結果、vSphere GUI から **.vmdk** イメージを直接使用することはできません。

7.5.7. Image Builder を使用して GCP にイメージをアップロードする

Image Builder を使用すると、**gce** イメージを構築し、ユーザーまたは GCP サービスアカウントの認証情報を提供してから、**gce** イメージを GCP 環境に直接アップロードできます。

7.5.7.1. CLI を使用して GCP に gce イメージをアップロードする

gce イメージを GCP にアップロードするための認証情報を含む設定ファイルをセットアップする手順に従います。

前提条件

- GCP にイメージをアップロードするためのユーザーまたはサービスアカウントの Google 認証情報を持っている。認証情報に関連付けられたアカウントには、少なくとも次の IAM ロールが割り当てられている必要があります。
  - **roles/storage.admin** - ストレージオブジェクトの作成と削除

- **roles/compute.storageAdmin** - VM イメージを Compute Engine にインポートします。

- 既存の GCP バケットがあります。

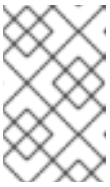
手順

1. テキストエディターを使用して、次の内容で **gcp-config.toml** 設定ファイルを作成します。

```
provider = "gcp"

[settings]
bucket = "GCP_BUCKET"
region = "GCP_STORAGE_REGION"
object = "OBJECT_KEY"
credentials = "GCP_CREDENTIALS"
```

ここでは、以下のようになります。

- **GCP_BUCKET** は既存のバケットを指します。アップロード中のイメージの中間ストレージオブジェクトを格納するために使用されます。
- **GCP_STORAGE_REGION** は、通常の Google ストレージリージョンであると同時に、デュアルリージョンまたはマルチリージョンでもあります。
- **OBJECT_KEY** は、中間ストレージオブジェクトの名前です。アップロード前に存在してはならず、アップロードプロセスが完了すると削除されます。オブジェクト名が **.tar.gz** で終わらない場合、拡張子がオブジェクト名に自動的に追加されます。
- **GCP_CREDENTIALS** は、GCP からダウンロードされた認証情報 JSON ファイルの Base64 エンコードスキームです。認証情報によって、GCP がイメージをアップロードするプロジェクトが決まります。



注記

GCP での認証に別のメカニズムを使用する場合、**gcp-config.toml** での **GCP_CREDENTIALS** の指定はオプションです。GCP で認証するさまざまな方法の詳細については、GCP での認証 をご覧ください。

2. 追加のイメージ名とクラウドプロバイダープロファイルを使用して設定を作成します。

```
$ sudo composer-cli compose start BLUEPRINT-NAME gce IMAGE_KEY gcp-config.toml
```

注記:イメージビルド、アップロード、およびクラウド登録プロセスは、完了に最大 10 分かかる場合があります。

検証

- イメージのステータスが FINISHED であることを確認します。

```
$ sudo composer-cli compose status
```

関連情報

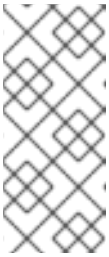
- アイデンティティーおよびアクセス管理

- ストレージバケットを作成します。

7.5.7.2. GCP による認証

Image Builder でいくつかの異なる種類の認証情報を使用して、GCP で認証できます。複数の認証情報セットを使用して GCP で認証するように Image Builder 設定が設定されている場合、次の優先順位で認証情報が使用されます。

1. 設定ファイルで **composer-cli** コマンドで指定された認証情報。
2. **osbuild-composer** ワーカー設定で設定された認証情報。
3. 次のオプションを使用して認証方法を自動的に見つけようとする、**Google GCP SDK** ライブラリーからのアプリケーションのデフォルト認証情報:
   a. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数が設定されている場合、Application Default Credentials は、変数が指すファイルから認証情報を読み込んで使用しようとします。
   b. Application Default Credentials は、コードを実行しているリソースに関連付けられたサービスアカウントを使用して認証を試みます。たとえば、Google Compute Engine VM です。



注記

イメージをアップロードする GCP プロジェクトを決定するには、GCP 認証情報を使用する必要があります。したがって、すべてのイメージを同じ GCP プロジェクトにアップロードする場合を除き、**composer-cli** コマンドを使用して **gcp-config.toml** 設定ファイルに認証情報を指定する必要があります。

7.5.7.2.1. composer-cli コマンドで認証情報を指定する

提供されたアップロードターゲット設定 **gcp-config.toml** で GCP 認証認証情報を指定できます。時間を節約するために、Google アカウント認証情報の JSON ファイルの **Base64** エンコードスキームを使用します。

手順

- 提供されたアップロードターゲット設定 **gcp-config.toml** で、認証情報を設定します。

```
provider = "gcp"

[settings]
provider = "gcp"

[settings]
...
credentials = "GCP_CREDENTIALS"
```

- **GOOGLE_APPLICATION_CREDENTIALS** 環境変数に保存されているパスを使用して、Google アカウント認証情報ファイルのエンコードされたコンテンツを取得するには、次のコマンドを実行します。

```
$ base64 -w 0 "${GOOGLE_APPLICATION_CREDENTIALS}"
```

7.5.7.2.2. osbuild-composer ワーカー設定で認証情報を指定する

すべてのイメージビルドでグローバルに GCP に使用される GCP 認証認証情報を設定できます。このようにして、イメージを同じ GCP プロジェクトにインポートする場合、GCP へのすべてのイメージのアップロードに同じ認証情報を使用できます。

手順

- **/etc/osbuild-worker/osbuild-worker.toml** ワーカー設定で、次の認証情報の値を設定します。

```
[gcp]
credentials = "PATH_TO_GCP_ACCOUNT_CREDENTIALS"
```

7.5.8. GUI イメージビルダーツールを使用して VMDK イメージを vSphere にプッシュする

GUI イメージビルダーツールを使用して VMware イメージを構築し、そのイメージを直接 vSphere インスタンスにプッシュすることで、イメージファイルをダウンロードして手動でプッシュする必要がなくなります。Image Builder を使用して直接 vSphere インスタンスサービスプロバイダーに **.vmdk** イメージを作成するには、次の手順に従います。

前提条件

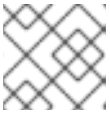
- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェイスを開いている。
- ブループリントを作成している。Web コンソールインターフェイスで Image Builder のブループリントの作成 を参照してください。
- vSphere アカウント がある。

手順

1. 作成したブループリントの **Images** タブをクリックします。
2. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
   イメージタイプウィンドウが開きます。
3. **Image type** ウィンドウで、以下を実行します。
   a. ドロップダウンメニューから、Type を選択します。VMware VSphere (.vmdk).
   b. **Upload to VMware** チェックボックスをチェックして、イメージを vSphere にアップロードします。
   c. オプション: インスタンス化するイメージのサイズを設定します。最小のデフォルトサイズは 2 GB です。
   d. **Next** をクリックします。
4. **Upload to VMware** ウィンドウの **Authentication** の下に以下の情報を入力します。
   a. **ユーザー名**: vSphere アカウントのユーザー名。
   b. **パスワード**: vSphere アカウントのパスワード。

5. **Upload to VMware** ウィンドウの **Destination** の下に以下の情報を入力します。
   a. **イメージ名**: アップロードするイメージの名前。
   b. **Host**:イメージをアップロードする VMware vSphere の URL。
   c. **Cluster**:イメージをアップロードするクラスターの名前。
   d. **Data center**:イメージがアップロードされるデータセンターの名前。
   e. **データストア**: イメージをアップロードするデータストアの名前。
   f. **Next** をクリックします。
6. **確認** ウィンドウで、イメージ作成の詳細を確認し、**Finish** をクリックします。
   Back をクリックして、誤った情報を変更できます。

   Image Builder は、RHEL vSphere イメージの Compose をキューに追加し、指定した vSphere インスタンスのクラスターにイメージを作成してアップロードします。

   注記

   イメージビルドおよびアップロードプロセスの完了には数分かかります。

   プロセスが完了すると、**Image build complete** のステータスが表示されます。

検証

イメージステータスのアップロードが正常に完了したら、アップロードしたイメージから仮想マシン (VM) を作成し、ログインできます。これを行うには、以下を行います。

1. VMware vSphere クライアントにアクセスします。
2. 指定した vSphere インスタンスのクラスターでイメージを検索します。
3. アップロードしたイメージから新しい仮想マシンを作成できます。
   a. アップロードしたイメージを選択します。
   b. 選択したイメージを右クリックします。
   c. **New Virtual Machine** をクリックします。
      New Virtual Machine ウィンドウが開きます。

      New Virtual Machine ウィンドウで、以下の詳細を指定します。

      i. **New Virtual Machine** を選択します。
      ii. VM の名前とフォルダーを選択します。
      iii. コンピューターリソースの選択: この操作の宛先コンピューターリソースを選択します
      iv. ストレージを選択:たとえば、NFS-Node1 を選択します。
      v. 互換性を選択:イメージは BIOS のみである必要があります。

vi. ゲストオペレーティングシステムを選択します。たとえば、**Linux** および **Red Hat Fedora (64-bit)** を選択します。

vii. **ハードウェアのカスタマイズ**:VM を作成するとき、右上の **デバイス設定** ボタンで、デフォルトの 新しいハードディスクを削除し、ドロップダウンを使用して 既存のハードディスクディスクイメージを選択します。

viii. 完了する準備ができました:詳細を確認し、**Finish** をクリックしてイメージを作成します。

d. **VMs** タブに移動します。

i. リストから、作成した仮想マシンを選択します。

ii. パネルから **Start** ボタンをクリックします。仮想マシンイメージを読み込み中であることを示す新しいウィンドウが表示されます。

iii. ブループリント用に作成した認証情報を使用してログインします。

iv. ブループリントに追加したパッケージがインストールされていることを確認できます。以下に例を示します。

```
$ rpm -qa | grep firefox
```

関連情報

- Installing the vSphere Client

7.5.9. GUI イメージビルダーツールを使用して VHD イメージを Microsoft Azure クラウドにプッシュする

Image Builder を使用して **.vhd** イメージを作成できます。次に、**.vhd** イメージを Microsoft Azure クラウドサービスプロバイダーの Blob Storage にプッシュできます。

前提条件

- システムへの root アクセス権があります。
- ブラウザーで、RHEL Web コンソールの Image Builder インターフェイスを開いている。
- ブループリントを作成している。Web コンソールインターフェイスで Image Builder のブループリントの作成 を参照してください。
- Microsoft ストレージアカウント が作成されました。
- 書き込み可能な Blob Storage が準備されました。

手順

1. **blueprint name** については、**Images** タブをクリックします。
2. **イメージの作成** をクリックして、カスタマイズしたイメージを作成します。
ポップアップウィンドウが開きます。

a. **Type** ドロップダウンメニューから、**Azure Disk Image(.vhd)** イメージを選択します。

b. **Upload to Microsoft Azure** チェックボックスをチェックして、イメージを Microsoft Azure Cloud にアップロードし、**Next** をクリックします。

c. Microsoft Azure へのアクセスを認証するには、対応するフィールドにストレージアカウントとストレージアクセスキーを入力します。**Next** をクリックします。
Microsoft ストレージアカウントの詳細 は、Settings → Access Key menu list で確認できます。

d. アップロードするイメージファイルに使用する **イメージ名** と、イメージのプッシュ先のイメージファイルに使用する Blob のストレージコンテナーを入力します。**Next** をクリックします。

e. 指定した情報を確認し、**Finish** をクリックします。
必要に応じて、**戻る** をクリックして、誤った情報を変更できます。

3. イメージ作成プロセスが開始されると、右上に小さなポップアップが表示され、次のメッセージが表示されます。**Image creation has been added to the queue**.
イメージプロセスの作成が完了したら、イメージを作成した Blueprint をクリックします。**images** タブで、作成したイメージの **イメージビルドの完了** ステータスを確認できます。

4. **Microsoft Azure Cloud** にプッシュしたイメージにアクセスするには、Microsoft Azure Portal にアクセスします。

5. 検索バーで **Images** と入力して、**Services** の下にある最初のエントリーを選択します。**Image Dashboard** にリダイレクトされます。

6. **Add** をクリックします。**Create an Image** ダッシュボードにリダイレクトされます。
以下の情報を追加します。

a. **名前**:新しいイメージの名前を選択します。

b. **リソースグループ**:**リソースグループ** を選択します。

c. **場所**:ストレージアカウントに割り当てられたリージョンと一致する **場所** を選択します。それ以外の場合は、Blob を選択できません。

d. **OS タイプ**:オペレーティングシステムの種類を **Linux** に設定します。

e. **VM Generation**:仮想マシンの生成は **Gen 1** に設定したままにします。

f. **Storage Blob**:**Storage blob input** の右側にある **参照** をクリックします。ダイアログを使用して、先ほどアップロードしたイメージを見つけます。
その他のフィールドはデフォルトのままにしておきます。

7. **作成** をクリックしてイメージを作成します。イメージが作成されたら、右上隅に **Successfully created image** というメッセージが表示されます。

8. **Refresh** をクリックして、新しく作成したイメージを表示し、開きます。

9. **+ Create VM** をクリックします。**Create a virtual machine** ダッシュボードにリダイレクトされます。

10. **Basic** タブの **Project Details** で、**Subscription** と **Resource Group** がすでに事前設定されています。
新しい **Resource Group** を作成する場合:

a. **Create new** をクリックします。

ポップアップで、**リソースグループ名** のコンテナーの作成が求められます。

b. 名前を入力して **OK** をクリックします。
事前に設定された **リソースグループ** をそのまま使用する場合は、以下を行います。

11. **インスタンスの詳細** で、次のように入力します。

a. **Virtual machine name**

b. **Region**

c. **イメージ**:作成したイメージがデフォルトで事前に選択されています。

d. **サイズ**:必要に応じて仮想マシンのサイズを選択します。
残りのフィールドはデフォルトのままにします。

12. **Administrator account** に、以下の情報を入力します。

a. **username**: アカウント管理者の名前。

b. **SSH Public Key source**: ドロップダウンメニューから、**Generate new key pair** を選択します。
既存のキーペアを使用するか、キーペアーを新規作成します。**Image Builder** を使用して、既存の公開鍵でイメージにユーザーを追加します。詳細は、SSH 鍵を持つユーザーアカウントの作成 を参照してください。

c. **key pair name**: キーペアの名前を挿入します。

13. **受信ポートのルール** で、各フィールドの値を選択します。

a. **Public inbound ports**:**Allow selected ports**.

b. **Select inbound ports**:デフォルト設定 **SSH (22)** を使用します。

14. **Review + Create** をクリックします。**Review + create** タブにリダイレクトされ、検証が正常に終了した旨の確認メッセージが表示されます。

15. 詳細を確認して **Create** をクリックします。
オプションで **Previous** をクリックして、以前に選択したオプションを修正できます。

16. **新しい鍵ペアを生成する** ウィンドウが開きます。**Download private key and create resources** をクリックします。
yourKey.pem として鍵ファイルを保存します。

17. デプロイメントが完了したら、**Go to resource** をクリックします。

18. 実際の仮想マシンの詳細を含む新規ウィンドウに、リダイレクトされます。ページの右上にあるパブリック IP アドレスを選択してクリップボードにコピーします。

次に、仮想マシンとの SSH 接続を作成して、仮想マシンに接続します。

1. 端末を開きます。

2. プロンプトで、VM への SSH 接続を開きます。IP アドレスは、仮想マシンの IP アドレスに、.pem へのパスは、キーファイルのダウンロード先のパスに置き換えます。

```
# ssh -i ./Downloads/yourKey.pem azureuser@10.111.12.123
```

3. 接続を続行するには確定する必要があります。続行するには **yes** と入力します。

上記の作業の結果、Microsoft Azure Storage Blob にプッシュした出力イメージをプロビジョニングする準備が整いました。

関連情報

- Microsoft Azure ストレージのドキュメント。
- Microsoft Azure ストレージアカウントを作成します。
- Red Hat カスタマーポータルでのケースの作成
- ヘルプとサポート
- Red Hat に問い合わせる

7.5.10. OpenStack への QCOW2 イメージのアップロード

Image Builder ツールを使用すると、OpenStack クラウドデプロイメントにアップロードしてそこでインスタンスを開始するのに適した、カスタマイズされた **.qcow2** イメージを作成できます。

警告

Image Builder を OpenStack イメージタイプで使用して作成する一般的な **QCOW2** イメージタイプの出力フォーマットを間違えないでください。これも QCOW2 フォーマットですが、OpenStack に固有の変更がさらに含まれています。

前提条件

- ブループリントを作成している。
- Image Builder を使用して **QCOW2** イメージを作成しました。詳細は、

手順

1. **QCOW2** イメージの作成を開始します。

```
# composer-cli compose start blueprint_name openstack
```

2. ビルドの状態を確認します。

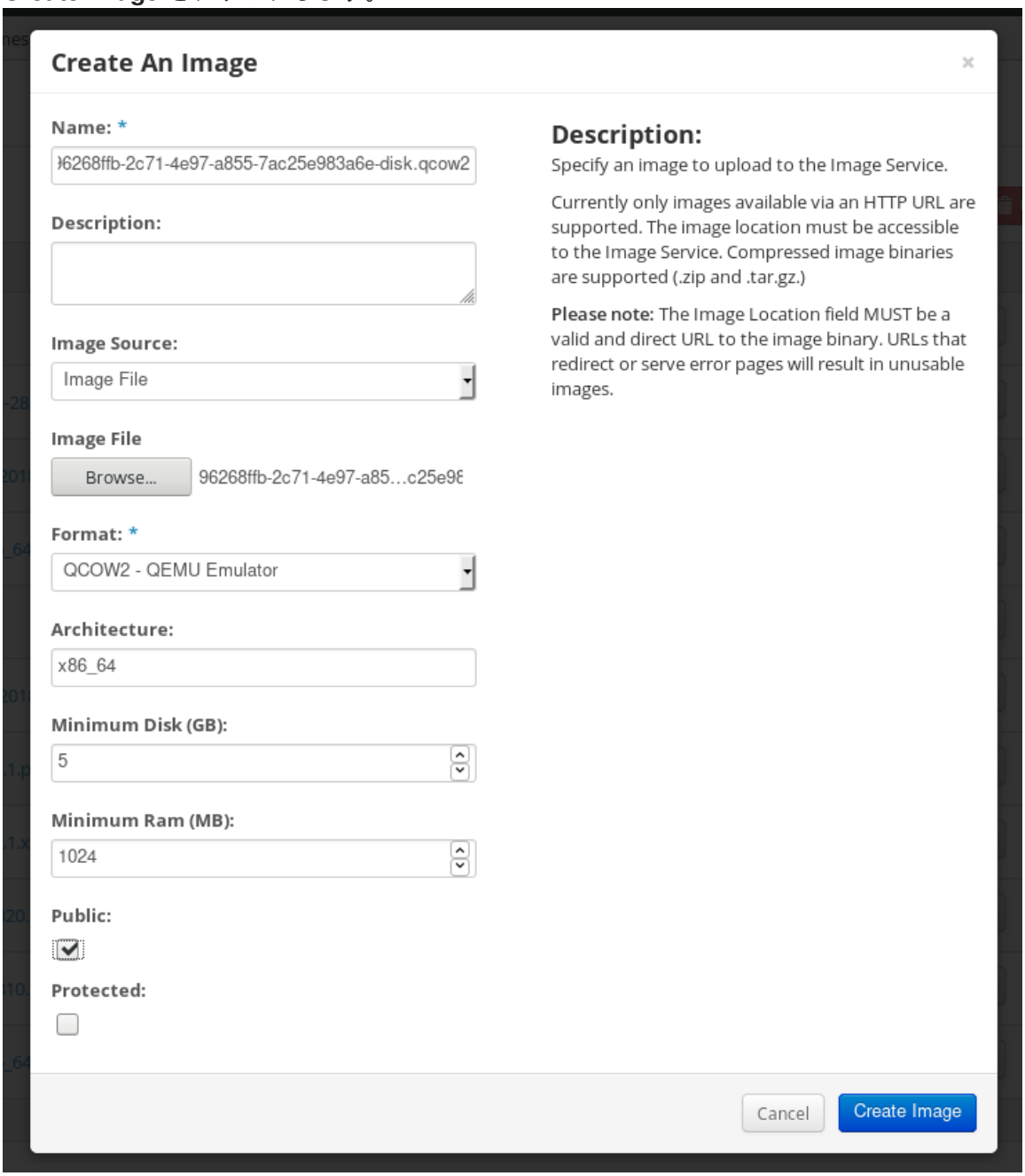
```
# composer-cli compose status
```

イメージのビルドが完了したら、イメージをダウンロードできます。

3. **QCOW2** イメージをダウンロードします。

```
# composer-cli compose image UUID
```

4. OpenStack ダッシュボードにアクセスし、**+Create Image** をクリックします。
5. 左側のメニューで、**Admin** タブを選択します。
   a. **System Panel** から **Image** をクリックします。
   Create An Image ウィザードが開きます。
6. **Create An Image** ウィザードで、以下を行います。
   a. イメージの名前を入力します。
   b. **Browse** をクリックして **QCOW2** イメージをアップロードします。
   c. **Format** ドロップダウンリストから、**QCOW2 - QEMU Emulator** を選択します。
   d. **Create Image** をクリックします。



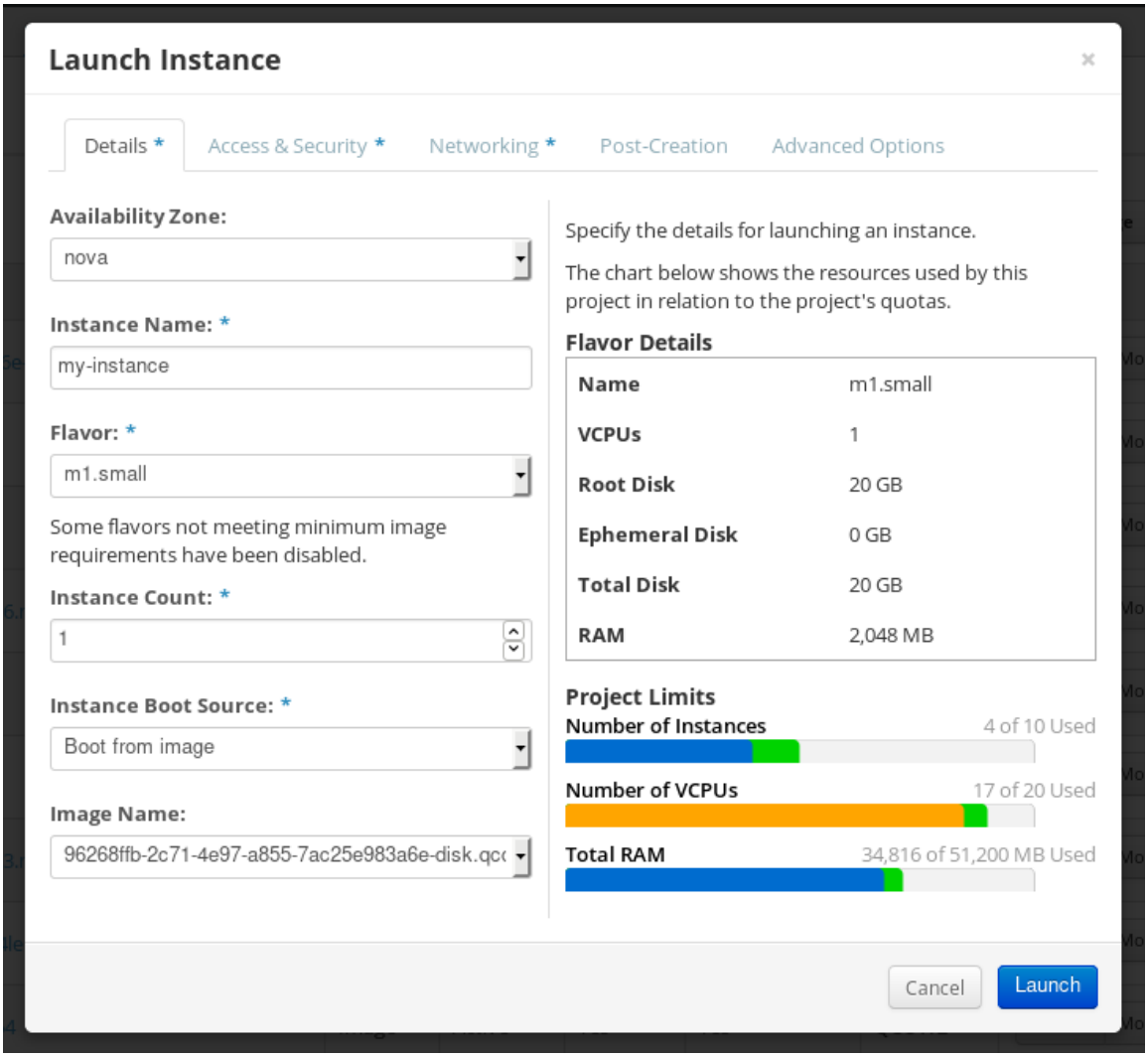
7. 左側のメニューで **Project** タブを選択します。
   a. **Compute** メニューから **Instances** を選択します。

b. **Launch Instance** ボタンをクリックします。
インスタンスの **Launch Instance** が開きます。

c. **Details** ページで、インスタンスの名前を入力します。**Next** をクリックします。

d. **Source** ページで、アップロードしたイメージの名前を選択します。**Next** をクリックします。

e. **Flavor** ページで、ニーズに最適なマシンリソースを選択します。**Launch** をクリックします。



8. イメージから任意のメカニズム (CLI または OpenStack Web UI) を使用して、イメージインスタンスを実行できます。秘密鍵を使用して、SSH 経由で、作成されたインスタンスにアクセスします。**cloud-user** としてログインします。

7.5.11. カスタマイズされた RHEL イメージを Alibaba にアップロードする準備

カスタマイズされた RHEL イメージを **Alibaba Cloud** にデプロイするには、まずカスタマイズされたイメージを検証する必要があります。Alibaba Cloud は、イメージを使用する前に特定の要件を満たすようにカスタムイメージを要求するため、イメージが正常に起動するように特別な設定が必要になります。

注記

Image Builder は、Alibaba の要件に準拠するイメージを生成します。ただし、Red Hat は、Alibaba **image_check ツール** を使用して、イメージのフォーマット準拠を確認することも推奨します。

前提条件

- Image Builder を使用して Alibaba イメージを作成しておく。

手順

1. Alibaba **image_check ツール** を使用して、チェックするイメージを含むシステムに接続します。
2. **image_check** ツールをダウンロードします。

```
$ curl -O http://docs-aliyun.cn-hangzhou.oss.aliyun-
inc.com/assets/attach/73848/cn_zh/1557459863884/image_check
```

3. イメージのコンプライアンスツールのファイルパーミッションを変更します。

```
# chmod +x image_check
```

4. 次のコマンドを実行して、イメージコンプライアンスツールのチェックを起動します。

```
# ./image_check
```

このツールは、システム設定を検証し、画面に表示されるレポートを生成します。image_check ツールは、イメージのコンプライアンスツールが実行しているフォルダーにこのレポートを保存します。

トラブルシューティング

いずれかの **検出項目** が失敗した場合は、ターミナルの指示に従って修正してください。リンクを参照してください。Detection items section.

関連情報

- Image Compliance Tool

7.5.12. カスタマイズされた RHEL イメージを Alibaba にアップロードする

Image Builder を使用して作成したカスタマイズされた **AMI** イメージを Object Storage Service (OSS) にアップロードできます。

前提条件

- Alibaba イメージのアップロードを設定している。Alibaba にイメージをアップロードするための準備 を参照してください。
- Image Builder を使用して **ami** イメージを作成しました。
- バケットがある。Creating a bucket を参照してください。

- アクティブな Alibaba アカウント がある。
- OSS をアクティベートしている。

手順

1. OSS コンソール にログインします。
2. 左側のバケットメニューで、イメージをアップロードするバケットを選択します。
3. 右上のメニューで、**Files** タブをクリックします。
4. **Upload** をクリックします。右側のダイアログウィンドウが開きます。以下を設定します。
   - **アップロード先**:これを選択すると、**現在** のディレクトリーまたは **指定した** ディレクトリーにファイルをアップロードします。
   - **ファイル ACL**:アップロードしたファイルのパーミッションのタイプを選択します。
5. **Upload** をクリックします。
6. アップロードするイメージを選択します。
7. **Open** をクリックします。

その結果、カスタマイズされた **AMI** イメージが OSS コンソールにアップロードされます。

関連情報

- Upload an object
- Creating an instance from custom images
- Importing images

7.5.13. イメージの Alibaba へのインポート

Image Builder を使用して作成したカスタマイズされた Alibaba RHEL イメージを Elastic Cloud Console (ECS) にインポートするには、次の手順に従います。

前提条件

- Alibaba イメージのアップロードを設定している。Alibaba にイメージをアップロードするための準備 を参照してください。
- Image Builder を使用して **ami** イメージを作成しました。
- バケットがある。Creating a bucket を参照してください。
- アクティブな Alibaba アカウント がある。
- OSS をアクティベートしている。
- イメージを OSS (Object Storage Service) にアップロードしている。Alibaba へのイメージのアップロード を参照してください。

手順

1. ECS コンソール にログインします。
   i. 左側のメニューで、**images** をクリックします。
   ii. 右上にある **Import Image** をクリックします。ダイアログウィンドウが開きます。
   iii. イメージが含まれる正しいリージョンを設定していることを確認します。以下の情報を入力します。
      a. **OSS Object Address**:OSS Object Address の取得方法を参照してください。
      b. **Image Name**
      c. **オペレーティングシステム**
      d. **System Disk Size**
      e. **システムアーキテクチャー**
      f. **プラットフォーム**:Red Hat
   iv. 必要に応じて、以下の情報を指定します。
      g. **Image Format** - アップロードしたイメージの形式に応じて **qcow2** または **ami**。
      h. **Image Description**
      i. **Add Images of Data Disks**
      アドレスは、OSS 管理コンソールで確認できます。左側のメニューで必要なバケットを選択した後:
2. **Files** セクションを選択します。
3. 適切なイメージの右側にある **Details** リンクをクリックします。
   画面右側にウィンドウが表示され、イメージの詳細が表示されます。**OSS** オブジェクトアドレスは **URL** ボックスにあります。
4. **OK** をクリックします。

注記

インポートプロセスの時間は、イメージのサイズによって異なります。

カスタマイズされたイメージが **ECS** コンソールにインポートされます。

関連情報

- Notes for importing images
- Creating an instance from custom images
- Upload an object

7.5.14. Alibaba を使用してカスタマイズされた RHEL イメージのインスタンスを作成する

Alibaba ECS Console を使用して、カスタマイズされた RHEL イメージのインスタンスを作成できます。

前提条件

- OSS をアクティベートして、カスタムイメージをアップロードしている。
- イメージを ECS コンソールに正常にインポートしている。Alibaba へのイメージのインポートを参照してください。

手順

1. ECS コンソール にログインします。
2. 左側のメニューで、**インスタンス** を選択します。
3. 右上隅にある **インスタンスの作成** をクリックします。新しいウィンドウにリダイレクトされます。
4. 必要な情報をすべて完了します。詳細は、Creating an instance by using the wizard を参照してください。
5. **Create Instance** をクリックして、順番を確認します。

注記

サブスクリプションによっては、**Create Instance** ではなく **Create Order** が表示されます。

その結果、アクティブなインスタンスを **Alibaba ECS Console** からデプロイする準備が整いました。

関連情報

- Creating an instance by using a custom image
- Create an instance by using the wizard

第8章 キックスタートを使用した自動インストールの実行

8.1. キックスタートインストールの基礎

以下は、キックスタートの基本情報と、それを使用して Red Hat Enterprise Linux のインストールを自動化する方法を説明します。

8.1.1. キックスタートを使用したインストールの概要

キックスタートは、RHEL インストールプロセスを部分的または完全に自動化する方法を提供します。

キックスタートファイルには、RHEL インストールオプションの一部またはすべてが含まれます。たとえば、タイムゾーン、ドライブのパーティション設定方法、インストールするパッケージなどです。事前に準備したキックスタートファイルを使用すると、ユーザーによる操作を必要としないインストールが可能になります。これは、Red Hat Enterprise Linux を多数のシステムに一度にデプロイする場合などに特に便利です。

キックスタートファイルによりソフトウェア選択の幅を広げることができます。グラフィカルインストールインターフェイスで Red Hat Enterprise Linux を手動でインストールする場合、ソフトウェアの選択は事前定義されている環境とアドオンの選択に限られます。キックスタートファイルを使用すると、パッケージを個別にインストールしたり、除外したりできます。

キックスタートファイルを 1 つのサーバーに置くことで、インストール時に各コンピューターが読み込むことができます。この方法を使用すると、1 つのキックスタートファイルで複数のマシンに Red Hat Enterprise Linux をインストールできるため、ネットワークおよびシステム管理者には理想的な方法になります。

キックスタートスクリプトおよびそのスクリプトの実行により生成されるログファイルは、インストール問題のデバッグの手助けとなるよう、新たにインストールしたシステムの **/tmp** ディレクトリーにすべて保存されます。インストールに使用されるキックスタートおよび Anaconda が生成した出力キックスタートは、ターゲットシステムの **/root** に保存され、キックスタートスクリプトレット実行のログは **/var/log/anaconda** に保存されます。

注記

キックスタートは、Red Hat Enterprise Linux の以前のバージョンではシステムをアップグレードするのに使用できました。Red Hat Enterprise Linux 7 以降では、この機能は削除されており、システムのアップグレードではなく、特殊なツールにより処理されます。Red Hat Enterprise Linux 8 へのアップグレードの詳細は、Upgrading from RHEL 7 to RHEL 8 および Considerations in adopting RHEL を参照してください。

8.1.2. 自動インストールのワークフロー

キックスタートを使用したインストールは、ローカルの DVD またはハードドライブを使用するか、NFS、FTP、HTTP、または HTTPS で実行できます。本セクションでは、キックスタートの使用方法の概要を説明します。

1. キックスタートファイルを作成します。手動で作成したり、手動インストール後に保存したキックファイルファイルをコピーしたり、オンライン生成ツールを使用してファイルを作成したりして、後で編集したりできます。Creating Kickstart files を参照してください。

2. リムーバブルメディア、ハードドライブ、ならびに HTTP (S) サーバー、FTP サーバー、または NFS サーバーに置いたインストールプログラムでキックスタートファイルを使用できるようにしてある。Making Kickstart files available to the installation program を参照してください。

3. インストール開始に使用する起動用メディアを作成します。起動可能なインストールメディアの作成 および PXE によるネットワークからのインストールの準備 を参照してください。

4. インストールソースをインストールプログラムに利用できるようにします。Creating installation sources for Kickstart installations を参照してください。

5. ブートメディアおよびキックスタートファイルを使用して、インストールを開始します。Starting Kickstart installations を参照してください。

これは、キックスタートファイルが必須のコマンドおよびセクションをすべて含む場合に、インストールが自動的に行われます。必須部分が 1 つ以上欠けている場合、またはエラーが発生した場合は、インストールを手動で行う必要があります。

注記

UEFI セキュアブートが有効になっているシステムに Red Hat Enterprise Linux のベータ版リリースをインストールする予定がある場合は、UEFI セキュアブートオプションを無効にしてから、インストールを開始します。

UEFI セキュアブートでは、オペレーティングシステムのカーネルが、対応する公開鍵を使用してシステムのファームウェアが検証する、認識済みの秘密鍵で署名されている必要があります。Red Hat Enterprise Linux ベータ版リリースの場合には、カーネルは Red Hat ベータ版固有の秘密鍵で署名されていますが、この秘密鍵はデフォルトではシステムで認識できません。その結果、システムはインストールメディアの起動に失敗します。

8.2. キックスタートファイルの作成

次の方法を使用してキックスタートファイルを作成できます。

- オンラインのキックスタート設定ツールを使用する。
- 手動インストールのログとして作成したキックスタートファイルをコピーする。
- キックスタートファイル全体を手動で書き込む。
- Red Hat Enterprise Linux 8 インストール用に Red Hat Enterprise Linux 7 キックスタートファイルを変換します。
  変換ツールの詳細については、Kickstart generator lab を参照してください。
- 仮想環境およびクラウド環境では、Image Builder を使用してカスタムシステムイメージを作成します。

一部の詳細なインストールオプションは、キックスタートファイルを手動で編集しないと設定できないことに注意してください。

8.2.1. キックスタート設定ツールを使用したキックスタートファイルの作成

Red Hat カスタマーポータルのアカウントをお持ちの場合は、カスタマーポータルで提供している Labs の Kickstart Generator ツールを使用して、キックスタートファイルをオンラインで生成できます。このツールは基本的な設定を段階的に説明し、作成したキックスタートファイルのダウンロードを可能にします。

前提条件

- Red Hat カスタマーポータルアカウントとアクティブな Red Hat サブスクリプションを持っている。

手順

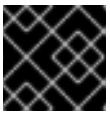
1. Lab で提供されている Kickstart Generator の情報は https://access.redhat.com/labsinfo/kickstartconfig を参照してください。
2. 見出しの左にある **Go to Application** ボタンをクリックし、次のページが読み込まれるのを待ちます。
3. ドロップダウンメニューで **Red Hat Enterprise Linux 8**を選択し、ページが更新するのを待ちます。
4. フォーム内のフィールドを使用して、インストールするシステムを記述します。
フォームの左側にあるリンクを使用すれば、フォームのセクション間をすばやく移動できます。
5. 生成されたキックスタートファイルをダウンロードするには、ページの先頭に戻り、赤色の **Download** ボタンをクリックします。
Web ブラウザーによりファイルが保存されます。

8.2.2. 手動インストールを実行したキックスタートファイルの作成

キックスタートファイルの作成方法としては、Red Hat Enterprise Linux の手動インストールにより作成されたファイルを使用することが推奨される方法となります。インストールが完了すると、インストール中に選択したものがすべて、インストール済みシステムの **/root/** ディレクトリーに置かれているキックスタートファイル **anaconda-ks.cfg** に保存されます。このファイルを使用して、以前とまったく同じ方法でインストールを行えます。または、このファイルをコピーして必要な変更を加え、その後のインストールで使用することもできます。

手順

1. RHEL をインストールします。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。
インストール時に、管理者権限を持つユーザーを作成します。
2. インストール済みシステムでインストールを完了し、再起動します。
3. 管理者アカウントでシステムにログインします。
4. **/root/anaconda-ks.cfg** ファイルを、任意の場所にコピーします。



重要

ファイルには、ユーザーとパスワードの情報が含まれます。

- 端末内のファイルの内容を表示するには、次のコマンドを実行します。

```
# cat /root/anaconda-ks.cfg
```

出力をコピーして、別のファイルに選択を保存できます。

- 別の場所にファイルをコピーするには、ファイルマネージャーを使用します。root 以外のユーザーがそのファイルを読み込めるように、コピーしたファイルのアクセス権を忘れずに変更してください。

関連情報

- 標準の RHEL 8 インストールの実行

8.2.3. 以前の RHEL インストールからキックスタートファイルを変換する

Kickstart Converter ツールを使用して、RHEL 7 Kickstart ファイルを RHEL 8 または 9 インストールで使用するために変換したり、RHEL 8 Kickstart ファイルを RHEL 9 で使用するために変換したりできます。ツールの詳細と、そのツールで RHEL キックスタートファイルを変換する方法は、https://access.redhat.com/labs/kickstartconvert/ を参照してください。

8.2.4. Image Builder を使用したカスタムイメージの作成

Red Hat Image Builder を使用して、仮想デプロイメント用およびクラウドデプロイメント用にカスタマイズされたシステムイメージを作成できます。

Image Builder を使用したカスタムイメージの作成の詳細は、**Composing a customized RHEL system image** を参照してください。

8.3. インストールプログラムでキックスタートファイルの準備

以下では、ターゲットシステムのインストールプログラムでキックスタートファイルを使用できるようにする方法を説明します。

8.3.1. ネットワークインストール用のポート

次の表は、ネットワークベースの各種インストールにファイルを提供するためにサーバーで開く必要があるポートの一覧です。

表8.1 ネットワークインストール用のポート

使用プロトコル	開くべきポート
HTTP	80
HTTPS	443
FTP	21
NFS	2049、111、20048
TFTP	69

関連情報

- ネットワークのセキュリティー保護

8.3.2. NFS サーバーでキックスタートファイルの準備

この手順では、キックスタートスクリプトファイルを NFS サーバーに格納する方法を説明します。この方法を使用すると、キックスタートファイルに物理メディアを使用することなく、1 つのソースから複数のシステムをインストールできます。

前提条件

- ローカルネットワーク上の Red Hat Enterprise Linux 8 を使用するサーバーへの管理者レベルのアクセス権がある。
- インストールするシステムがサーバーに接続できる。
- サーバー上のファイアウォールがインストール先のシステムからの接続を許可している。

手順

1. root で以下のコマンドを実行して、**nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

2. キックスタートファイルを、NFS サーバーのディレクトリーにコピーします。
3. テキストエディターで **/etc/exports** ファイルを開き、以下の構文の行を追加します。

```
/exported_directory/ clients
```

4. **/exported_directory/** を、キックスタートファイルを保存しているディレクトリーのフルパスに置き換えます。**clients** の代わりに、この NFS サーバーからインストールするコンピューターのホスト名または IP アドレス、すべてのコンピューターが ISO イメージにアクセスするためのサブネットワーク、またはネットワークアクセスのあるコンピューターが NFS サーバーにアクセスして ISO イメージを使用できるようにする場合はアスタリスク記号 (*) を使用します。このフィールドの形式に関する詳細は、man ページの **exports(5)** を参照してください。**/rhel8-install/** ディレクトリーを、すべてのクライアントに対する読み取り専用として使用できるようにする基本設定は次のようになります。

```
/rhel8-install *
```

5. **/etc/exports** ファイルを保存して、テキストエディターを終了します。
6. nfs サービスを起動します。

```
# systemctl start nfs-server.service
```

/etc/exports ファイルに変更を加える前にサービスを稼働していた場合は、以下のコマンドを実行して、稼働中の NFS サーバーで設定を再ロードします。

```
# systemctl reload nfs-server.service
```

キックスタートファイルは NFS 経由でアクセス可能になり、インストールに使用できるようになりました。

注記

キックスタートソースを指定する場合は、プロトコルに **nfs:** を使用して、サーバーのホスト名または IP アドレス、コロン記号 (**:**)、およびそのファイルを保存しているディレクトリーを指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、そのファイルを **/rhel8-install/my-ks.cfg** に保存した場合、指定するインストールソースの起動オプションは **inst.ks=nfs:myserver.example.com:/rhel8-install/my-ks.cfg** となります。

関連情報

- PXE を使用してネットワークからインストールするための準備

8.3.3. HTTP サーバーまたは HTTPS サーバーで使用できるキックスタートファイルの準備

この手順では、キックスタートスクリプトファイルを HTTP サーバーまたは HTTPS サーバーに格納する方法を説明します。この方法を使用すると、キックスタートファイルに物理メディアを使用することなく、1 つのソースから複数のシステムをインストールできます。

前提条件

- ローカルネットワーク上の Red Hat Enterprise Linux 8 を使用するサーバーへの管理者レベルのアクセス権がある。
- インストールするシステムがサーバーに接続できる。
- サーバー上のファイアウォールがインストール先のシステムからの接続を許可している。

手順

1. キックスタートファイルを HTTP に保存するには、**httpd** パッケージをインストールします。

```
# yum install httpd
```

HTTPS にキックスタートファイルを保存するには、**httpd** パッケージおよび **mod_ssl** パッケージをインストールします。

```
# yum install httpd mod_ssl
```

警告

Apache Web サーバー設定で SSL セキュリティーが有効になっている場合は、TLSv1 プロトコルのみが有効で、SSLv2 と SSLv3 は無効になっていることを確認してください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は https://access.redhat.com/solutions/1232413 を参照してください。

重要

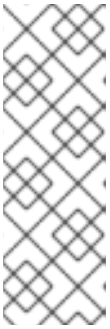
自己署名証明書付きの HTTPS サーバーを使用する場合は、**inst.noverifyssl** オプションを指定してインストールプログラムを起動する必要があります。

2. **/var/www/html/** ディレクトリーのサブディレクトリーに、HTTP(S) サーバーへのキックスタートファイルをコピーします。

3. httpd サービスを起動します。

```
# systemctl start httpd.service
```

キックスタートファイルはアクセス可能になり、インストールとして使用できるようになりました。



注記

キックスタートファイルの場所を指定する場合は、プロトコルに **http://** または **https://** を使用して、サーバーのホスト名または IP アドレス、キックスタートファイルのパス (HTTP サーバーの root への相対パス) を指定します。たとえば、HTTP を使用して、サーバーのホスト名が **myserver.example.com** で、キックスタートファイルを **/var/www/html/rhel8-install/my-ks.cfg** にコピーした場合、指定するインストールソースは **http://myserver.example.com/rhel8-install/my-ks.cfg** となります。

関連情報

- さまざまな種類のサーバーのデプロイメント

8.3.4. FTP サーバーでキックスタートファイルの準備

この手順では、キックスタートスクリプトファイルを FTP サーバーに格納する方法を説明します。この方法を使用すると、キックスタートファイルに物理メディアを使用することなく、1 つのソースから複数のシステムをインストールできます。

前提条件

- ローカルネットワーク上の Red Hat Enterprise Linux 8 を使用するサーバーへの管理者レベルのアクセス権がある。
- インストールするシステムがサーバーに接続できる。
- サーバー上のファイアウォールがインストール先のシステムからの接続を許可している。

手順

1. root で以下のコマンドを実行して、**vsftpd** パッケージをインストールします。

```
# yum install vsftpd
```

2. 必要に応じて、**/etc/vsftpd/vsftpd.conf** 設定ファイルをテキストエディターで開いて編集します。

   a. **anonymous_enable=NO** の行を **anonymous_enable=YES** に変更します。

b. **write_enable=YES** の行を **write_enable=NO** に変更します。

c. **pasv_min_port=min_port** と **pasv_max_port=max_port** の行を追加します。**min_port** および **max_port** は、パッシブモードの FTP サーバーで使用されるポート番号の範囲に置き換えます (例: **10021** および **10031**)。
このステップは、各種のファイアウォール/NAT 設定を採用するネットワーク環境に必要です。

d. オプションで、カスタムの変更を設定に追加します。利用可能なオプションは、**vsftpd.conf(5)** の man ページを参照してください。この手順では、デフォルトのオプションが使用されていることを前提としています。

警告

vsftpd.conf ファイルで SSL/TLS セキュリティーを設定している場合は、TLSv1 プロトコルのみを有効にし、SSLv2 と SSLv3 は無効にしてください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は、https://access.redhat.com/solutions/1234773 を参照してください。

3. サーバーのファイアウォールを設定します。

a. ファイアウォールを有効にします。

```
# systemctl enable firewalld
# systemctl start firewalld
```

b. 直前の手順の FTP ポートおよびポート範囲のファイアウォールで有効にします。

```
# firewall-cmd --add-port min_port-max_port/tcp --permanent
# firewall-cmd --add-service ftp --permanent
# firewall-cmd --reload
```

min_port-max_port を、**/etc/vsftpd/vsftpd.conf** 設定ファイルに入力したポート番号に置き換えます。

4. **/var/ftp/** ディレクトリーまたはそのサブディレクトリーに、FTP サーバーへのキックスタートファイルをコピーします。

5. 正しい SELinux コンテキストとアクセスモードがファイルに設定されていることを確認してください。

```
# restorecon -r /var/ftp/your-kickstart-file.ks
# chmod 444 /var/ftp/your-kickstart-file.ks
```

6. **vsftpd** サービスを開始します。

```
# systemctl start vsftpd.service
```

/etc/vsftpd/vsftpd.conf ファイルを変更する前から、このサービスがすでに実行されていた場合は、サービスを再起動して必ず編集後のファイルを読み込ませてください。

```
# systemctl restart vsftpd.service
```

vsftpd サービスを有効にして、システムの起動プロセス時に開始するようにします。

```
# systemctl enable vsftpd
```

キックスタートファイルはアクセス可能になり、同じネットワークのシステムからのインストールとして使用できるようになりました。

注記

インストールソースを設定するには、プロトコルに **ftp://** を使用して、サーバーのホスト名または IP アドレス、キックスタートファイルのパス (FTP サーバーの root への相対パス) を指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、ファイルを **/var/ftp/my-ks.cfg** にコピーした場合、指定するインストールソースは **ftp://myserver.example.com/my-ks.cfg** となります。

8.3.5. ローカルボリュームでキックスタートファイルの準備

この手順では、インストールするシステムのボリュームにキックスタートスクリプトファイルを保存する方法を説明します。この方法により、別のシステムは必要なくなります。

前提条件

- USB スティックなど、インストールするマシンに移動できるドライブがある。
- ドライブには、インストールプログラムで読み取ることができるパーティションが含まれている。対応しているタイプは、**ext2**、**ext3**、**ext4**、**xfs**、および **fat** です。
- ドライブがシステムに接続されており、そのボリュームがマウントされている。

手順

1. ボリューム情報のリストを表示し、キックスタートファイルをコピーするボリュームの UUID をメモします。

```
# lsblk -l -p -o name,rm,ro,hotplug,size,type,mountpoint,uuid
```

2. ボリュームのファイルシステムに移動します。
3. このファイルシステムにキックスタートファイルをコピーします。
4. **inst.ks=** オプションを使用して後で使用する文字列をメモしておきます。この文字列の形式は **hd:UUID=volume-UUID:path/to/kickstart-file.cfg** です。パスは、ファイルシステムシステム階層の / (root) ではなく、ファイルシステムの root に相対的になります。 **volume-UUID** を、上記の UUID に置き換えます。
5. ドライブボリュームのマウントをすべて解除します。

```
# umount /dev/xyz ...
```

スペースで区切って、コマンドにすべてのボリュームを追加します。

8.3.6. 自動読み込みのローカルボリュームでキックスタートファイルを使用可能に

特別な名前が付けられたキックスタートファイルを、インストールするシステムで特別な名前が付けられたボリュームの root に置くことができます。これにより、別のシステムが必要なくなり、インストールプログラムが自動的にファイルを読み込むことができるようになります。

前提条件

- USB スティックなど、インストールするマシンに移動できるドライブがある。
- ドライブには、インストールプログラムで読み取ることができるパーティションが含まれている。対応しているタイプは、**ext2**、**ext3**、**ext4**、**xfs**、および **fat** です。
- ドライブがシステムに接続されており、そのボリュームがマウントされている。

手順

1. キックスタートファイルをコピーするボリューム情報をリスト表示します。

```
# lsblk -l -p
```

2. ボリュームのファイルシステムに移動します。
3. このファイルシステムの root にキックスタートファイルをコピーします。
4. キックスタートファイルの名前を **ks.cfg** に変更します。
5. ボリュームの名前を **OEMDRV** に変更します。
   - **ext2**、**ext3**、および **ext4** のファイルシステムの場合:

```
# e2label /dev/xyz OEMDRV
```

   - XFS ファイルシステムの場合:

```
# xfs_admin -L OEMDRV /dev/xyz
```

   /dev/xyz を、ボリュームのブロックデバイスのパスに置き換えます。
6. ドライブボリュームのマウントをすべて解除します。

```
# umount /dev/xyz ...
```

   スペースで区切って、コマンドにすべてのボリュームを追加します。

8.4. キックスタートインストール用のインストールソースの作成

本セクションでは、必要なリポジトリーおよびソフトウェアパッケージを含む DVD ISO イメージを使用して、Boot ISO イメージのインストールソースを作成する方法を説明します。

8.4.1. インストールソースの種類

最小限のブートイメージには、以下のいずれかのインストールソースを使用できます。

- **DVD:**DVD に DVD ISO イメージを書き込みます。DVD はインストールソース (ソフトウェアパッケージソース) として自動的に使用されます。
- **ハードドライブまたは USB ドライブ:**DVD ISO イメージをドライブにコピーして、ドライブからソフトウェアパッケージをインストールするように、インストールプログラムを設定します。USB ドライブを使用する場合は、インストールを開始する前に、USB ドライブがシステムに接続されていることを確認してください。インストールプログラムは、インストールの開始後にメディアを検出することができません。
  - **ハードドライブの制限**: ハードドライブの DVD ISO イメージは、インストールプログラムがマウントできるファイルシステムを使用しているパーティションに置く必要があります。対応するファイルシステムは、**xfs**、**ext2**、**ext3**、**ext4**、および **vfat (FAT32)** となります。

警告

Microsoft Windows システムで、ハードドライブをフォーマットする際に使用されるデフォルトのファイルシステムは NTFS です。exFAT ファイルシステムも利用できます。ただし、このファイルシステムは、いずれもインストール時に変更することができません。Microsoft Windows のインストールソースとして、ハードドライブまたは USB ドライブを作成する場合は、ドライブを FAT32 としてフォーマットするようにしてください。FAT32 ファイルシステムは、4 GiB を超えるファイルを保存できません。

Red Hat Enterprise Linux 8 では、ローカルのハードドライブのディレクトリーからインストールできます。これを行うには、DVD ISO イメージの内容をハードドライブのディレクトリーにコピーし、ISO イメージの代わりに、そのディレクトリーをインストールソースとして指定します。たとえば、**inst.repo=hd:<device>:<path to the directory>** です。

- **ネットワークの場所:**DVD ISO イメージまたはインストールツリー (DVD ISO イメージから抽出したコンテンツ) をネットワーク上の場所にコピーし、次のプロトコルを使用して、ネットワーク経由でインストールを実行します。
  - **NFS:**DVD ISO イメージは、ネットワークファイルシステム (NFS) 共有にあります。
  - **HTTPS、HTTP、または FTP の場合:**インストールツリーは、HTTP、HTTPS、または FTP 経由でアクセス可能なネットワーク上にあります。

8.4.2. ネットワークインストール用のポート

次の表は、ネットワークベースの各種インストールにファイルを提供するためにサーバーで開く必要があるポートの一覧です。

表8.2 ネットワークインストール用のポート

使用プロトコル	開くべきポート
HTTP	80
HTTPS	443
FTP	21
NFS	2049、111、20048
TFTP	69

関連情報

- ネットワークのセキュリティー保護

8.4.3. NFS サーバーへのインストールソースの作成

この方法を使用して、物理メディアに接続しなくても、1 つのソースから複数のシステムをインストールできます。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。
- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。
- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。

手順

1. **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

2. DVD ISO イメージを、NFS サーバーのディレクトリーにコピーします。
3. テキストエディターで **/etc/exports** ファイルを開き、以下の構文の行を追加します。

```
/exported_directory/ clients
```

- **/exported_directory/** を、ISO イメージが含まれるディレクトリーのフルパスに置き換えます。
- **clients** を次のいずれかに置き換えます。

- ターゲットシステムのホスト名または IP アドレス
- すべてのターゲットシステムが ISO イメージへのアクセスに使用できるサブネットワーク
- NFS サーバーへのネットワークアクセスを持つすべてのシステムが ISO イメージを使用できるようにするためのアスタリスク記号 (*)

このフィールドの形式に関する詳細は、**exports(5)** の man ページを参照してください。

たとえば、**/rhel8-install/** ディレクトリーを、すべてのクライアントに対する読み取り専用として使用できるようにする基本設定は次のようになります。

```
/rhel8-install *
```

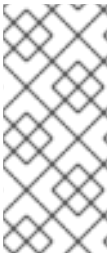
4. **/etc/exports** ファイルを保存して、テキストエディターを終了します。
5. nfs サービスを起動します。

```
# systemctl start nfs-server.service
```

/etc/exports ファイルを変更する前に サービスが稼働していた場合は、NFS サーバーの設定をリロードします。

```
# systemctl reload nfs-server.service
```

ISO イメージは、NFS 経由でアクセス可能になり、インストールソースとして使用できるようになりました。



注記

インストールソースを設定するには、プロトコルに **nfs:** を使用し、サーバーのホスト名または IP アドレス、コロン記号 **(:)**、および ISO イメージを保存しているディレクトリーを指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、ISO イメージを **/rhel8-install/** に保存した場合、指定するインストールソースは **nfs:myserver.example.com:/rhel8-install/** となります。

8.4.4. HTTP または HTTPS を使用するインストールソースの作成

インストールツリー (DVD ISO イメージから抽出したコンテンツと、有効な **.treeinfo** ファイル含むディレクトリー) を使用したネットワークベースのインストール用のインストールソースを作成できます。インストールソースには、HTTP、または HTTPS でアクセスします。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。
- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。

- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。
- **httpd** パッケージがインストールされている。
- **https** インストールソースを使用すると、**mod_ssl** パッケージがインストールされます。

警告

Apache Web サーバー設定で SSL セキュリティーが有効になっている場合は、TLSv1.3 プロトコルを有効にすることが推奨されます。デフォルトでは、TLSv1.2 が有効になっており、TLSv1 (LEGACY) プロトコルを使用できます。

重要

自己署名証明書付きの HTTPS サーバーを使用する場合は、**noverifyssl** オプションを指定してインストールプログラムを起動する必要があります。

手順

1. HTTP(S) サーバーに DVD ISO イメージをコピーします。
2. DVD ISO イメージをマウントするのに適したディレクトリーを作成します。以下はその例です。

```
# mkdir /mnt/rhel8-install/
```

3. DVD ISO イメージをディレクトリーにマウントします。

```
# mount -o loop,ro -t iso9660 /image_directory/image.iso /mnt/rhel8-install/
```

/image_directory/image.iso を DVD ISO イメージへのパスに置き換えます。

4. マウントされたイメージから、HTTP(S) サーバーの root にファイルをコピーします。

```
# cp -r /mnt/rhel8-install/ /var/www/html/
```

このコマンドにより、イメージに含まれるファイルが保存される **/var/www/html/rhel8-install/** ディレクトリーを作成します。他の一部のコピー方法は、有効なインストールソースに必要な **.treeinfo** ファイルを省略する可能性があることに注意してください。この手順で示されているように、ディレクトリー全体に対して **cp** コマンドを入力すると、**.treeinfo** が正しくコピーされます。

5. **httpd** サービスを起動します。

```
# systemctl start httpd.service
```

これにより、インストールツリーにアクセスできるようになり、インストールソースとして使用できるようになります。

注記

インストールソースを設定するには、プロトコルに **http://** または **https://** を使用して、サーバーのホスト名または IP アドレス、および ISO イメージのファイルを保存するディレクトリー (HTTP サーバーの root への相対パス) を指定します。たとえば、HTTP を使用し、サーバーのホスト名が **myserver.example.com** で、イメージのファイルが **/var/www/html/rhel8-install/** にコピーされた場合、指定するインストールソースは **http://myserver.example.com/rhel8-install/** となります。

関連情報

- さまざまな種類のサーバーのデプロイメント

8.4.5. FTP を使用するインストールソースの作成

インストールツリー (DVD ISO イメージから抽出したコンテンツと、有効な **.treeinfo** ファイル含むディレクトリー) を使用したネットワークベースのインストール用のインストールソースを作成できます。インストールソースには、FTP を使用してアクセスします。

前提条件

- Red Hat Enterprise Linux 8 を搭載したサーバーへの管理者レベルのアクセス権があり、このサーバーが、インストールするシステムと同じネットワーク上にある。
- Binary DVD イメージをダウンロードしている。詳しくは、インストール ISO イメージのダウンロード を参照してください。
- イメージファイルから、起動可能な CD、DVD、または USB デバイスを作成している。詳細は、インストールメディアの作成 を参照してください。
- ファイアウォールにより、インストールしようとしているシステムがリモートインストールソースにアクセスできることを確認している。詳細については、ネットワークベースのインストール用のポート を参照してください。
- **vsftpd** パッケージがインストールされている。

手順

1. 必要に応じて、**/etc/vsftpd/vsftpd.conf** 設定ファイルをテキストエディターで開いて編集します。
   a. **anonymous_enable=NO** の行を **anonymous_enable=YES** に変更します。
   b. **write_enable=YES** の行を **write_enable=NO** に変更します。
   c. **pasv_min_port=<min_port>** および **pasv_max_port=<max_port>** の行を追加します。<min_port> と <max_port> を、FTP サーバーがパッシブモードで使用するポート番号の範囲 (**10021** と **10031** など) に置き換えます。
   この手順は、各種のファイアウォール/NAT 設定を採用するネットワーク環境で必要になる可能性があります。
   d. オプション: カスタム変更を設定に追加します。利用可能なオプションは、**vsftpd.conf(5)** の man ページを参照してください。この手順では、デフォルトのオプションが使用されていることを前提としています。

警告

vsftpd.conf ファイルで SSL/TLS セキュリティーを設定している場合は、TLSv1 プロトコルのみを有効にし、SSLv2 と SSLv3 は無効にしてください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は、https://access.redhat.com/solutions/1234773 を参照してください。

2. サーバーのファイアウォールを設定します。
   a. ファイアウォールを有効にします。

```
# systemctl enable firewalld
```

   b. ファイアウォールを起動します。

```
# systemctl start firewalld
```

   c. 前の手順で設定した FTP ポートとポート範囲を許可するようにファイアウォールを設定します。

```
# firewall-cmd --add-port min_port-max_port/tcp --permanent
# firewall-cmd --add-service ftp --permanent
```

   <**min_port**> と <**max_port**> を **/etc/vsftpd/vsftpd.conf** 設定ファイルに入力したポート番号に置き換えます。

   d. ファイアウォールをリロードして、新しいルールを適用します。

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

3. DVD ISO イメージを FTP サーバーにコピーします。
4. DVD ISO イメージをマウントするのに適したディレクトリーを作成します。以下はその例です。

```
# mkdir /mnt/rhel8-install
```

5. DVD ISO イメージをディレクトリーにマウントします。

```
# mount -o loop,ro -t iso9660 /image-directory/image.iso /mnt/rhel8-install
```

   /image-directory/image.iso を DVD ISO イメージへのパスに置き換えます。

6. マウントされたイメージから、FTP サーバーのルートにファイルをコピーします。

```
# mkdir /var/ftp/rhel8-install
# cp -r /mnt/rhel8-install/ /var/ftp/
```

このコマンドは、イメージに含まれるファイルが保存される **/var/ftp/rhel8-install/** ディレクトリーを作成します。一部のコピー方法は、有効なインストールソースに必要な **.treeinfo** ファイルを省略できることに注意してください。この手順で示されているように、ディレクトリー全体に対して **cp** コマンドを入力しても、**.treeinfo** が正しくコピーされます。

7. 正しい SELinux コンテキストとアクセスモードが、コピーされたコンテンツに設定されていることを確認します。

```
# restorecon -r /var/ftp/rhel8-install
# find /var/ftp/rhel8-install -type f -exec chmod 444 {} \;
# find /var/ftp/rhel8-install -type d -exec chmod 755 {} \;
```

8. **vsftpd** サービスを開始します。

```
# systemctl start vsftpd.service
```

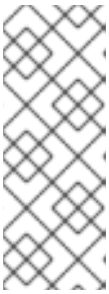
/etc/vsftpd/vsftpd.conf ファイルを変更する前から、このサービスがすでに実行されていた場合は、サービスを再起動して必ず編集後のファイルを読み込ませてください。

```
# systemctl restart vsftpd.service
```

vsftpd サービスを有効にして、システムの起動プロセス時に開始するようにします。

```
# systemctl enable vsftpd
```

これにより、インストールツリーにアクセスできるようになり、インストールソースとして使用できるようになります。



注記

インストールソースを設定するには、プロトコルに **ftp://** を使用して、サーバーのホスト名または IP アドレス、および ISO イメージのファイルを保存するディレクトリー (FTP サーバーの root への相対パス) を指定します。たとえば、サーバーのホスト名が **myserver.example.com** で、イメージからコピーしたファイルを **/var/ftp/rhel8-install/** に置いた場合、指定するインストールソースは **ftp://myserver.example.com/rhel8-install/** となります。

8.5. キックスタートインストールの開始

キックスタートインストールは、複数の方法で開始できます。

- 手動でインストールプログラムの起動メニューに入り、そこにキックスタートファイルを含むオプションを指定します。
- 自動的に PXE ブートで起動オプションを編集することもできます。
- 特定の名前を持つボリュームに、自動的にファイルを提供することもできます。

次のセクションでは、各メソッドの実行方法を説明します。

8.5.1. 手動でのキックスタートインストールの開始

本セクションでは、キックスタートを手動で起動する方法を説明します。この場合は、(**boot:** プロンプトで起動オプションを追加することで) ユーザーとの対話が必要になります。インストールシステムを

起動する場合は、起動オプション **inst.ks=location** を使用します。location は、キックスタートファイルの場所に置き換えます。ブートオプションとブートプロンプトの形式を指定する正確な方法は、システムのアーキテクチャーによって異なります。詳細は、RHEL インストーラーの起動オプション ガイドを参照してください。

前提条件

- インストールするシステムからアクセスできる場所に、キックスタートファイルを用意しておきます。

手順

1. ローカルメディア (CD、DVD、USB フラッシュドライブなど) を使用してシステムを起動します。
2. 起動プロンプトで、必要な起動オプションを指定します。
   a. キックスタートファイルまたは必要なリポジトリーがネットワークの場所にある場合は、**ip=** オプションを使用したネットワークの設定が必要になる場合があります。インストーラーは、このオプションを使用せずに、デフォルトで DHCP プロトコルを使用するすべてのネットワークデバイスを設定しようとします。
   b. 起動オプション **inst.ks=** と、キックスタートファイルの場所を追加します。
   c. 必要なパッケージがインストールされるソフトウェアソースにアクセスするには **inst.repo=** オプションを追加しないといけない場合があります。このオプションを指定しないと、キックスタートファイルでインストールソースを指定する必要があります。

   起動オプションの編集方法の詳細は、Editing boot options を参照してください。
3. 追加した起動オプションを確認してインストールを開始します。
   これにより、キックスタートファイルで指定されているインストールオプションを使用したインストールが開始します。キックスタートファイルに問題がなく、必要なコマンドがすべて含まれていれば、この時点からインストールは完全に自動化で行われます。

注記

UEFI セキュアブートが有効になっているシステムに、Red Hat Enterprise Linux ベータ版リリースをインストールした場合は、システムの Machine Owner Key (MOK) リストにベータ版の公開鍵を追加します。UEFI セキュアブートおよび Red Hat Enterprise Linux Beta リリースの詳細は、**標準の RHEL 8 インストールの実行** ドキュメントの インストール後のタスクの完了 セクションを参照してください。

8.5.2. PXE を使用した自動キックスタートインストールの開始

AMD64、Intel 64、および 64 ビット ARM システム、ならびに IBM Power Systems サーバーでは、PXE サーバーを使用して起動する機能があります。PXE サーバーの設定時に、ブートローダー設定ファイルに起動オプションを追加できます。これにより、インストールを自動的に開始できるようになります。このアプローチにより、ブートプロセスを含めたインストールを完全に自動化できるようになります。

この手順は一般的な参照です。詳細な手順はシステムのアーキテクチャーによって異なります。すべてのオプションが、すべてのアーキテクチャーで使用できるわけではありません (たとえば、64 ビットの IBM Z で PXE ブートを使用することはできません)。

前提条件

前提条件

- インストールするシステムからアクセスできる場所に、キックスタートファイルを用意しておきます。
- システムを起動してインストールを開始するために使用できる PXE サーバーが用意されています。

手順

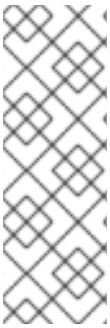
1. PXE サーバー上でブートローダー設定ファイルを開き、**inst.ks=** 起動オプションを適切な行に追加します。ファイル名と構文は、システムのアーキテクチャーおよびハードウェアにより異なります。
   - BIOS が搭載される AMD64 システムおよび Intel 64 システムのファイル名は、デフォルトまたはシステムの IP アドレスをベースにしたもののいずれかになります。このケースでは、インストールエントリーにある append 行に、**inst.ks=** オプションを追加します。設定ファイルの append 行は以下のようになります。

     ```
     append initrd=initrd.img inst.ks=http://10.32.5.1/mnt/archive/RHEL-8/8.x/x86_64/kickstarts/ks.cfg
     ```

   - GRUB2 ブートローダーを使用しているシステム (UEFI ファームウェアが搭載されている AMD64、Intel 64、および 64 ビット ARM システム、ならびに IBM Power Systems サーバー) のファイル名は **grub.cfg** になります。このファイルのインストールエントリーに含まれる kernel 行に、**inst.ks=** オプションを追加します。設定ファイルの kernel 行の例を以下に示します。

     ```
     kernel vmlinuz inst.ks=http://10.32.5.1/mnt/archive/RHEL-8/8.x/x86_64/kickstarts/ks.cfg
     ```

2. ネットワークサーバーからインストールを起動します。
   これでキックスタートファイルで指定されているインストールオプションを使用したインストールが開始します。キックスタートファイルに問題がなく、必要なコマンドがすべて含まれていれば、インストールは完全に自動で行われます。



注記

UEFI セキュアブートが有効になっているシステムに、Red Hat Enterprise Linux ベータ版リリースをインストールした場合は、システムの Machine Owner Key (MOK) リストにベータ版の公開鍵を追加します。

UEFI セキュアブートおよび Red Hat Enterprise Linux Beta リリースの詳細は、**標準の RHEL 8 インストールの実行** ドキュメントの インストール後のタスクの完了 セクションを参照してください。

8.5.3. ローカルボリュームを使用した自動キックスタートインストールの開始

特別にラベルが追加されたストレージボリュームで、特定の名前が付いたキックスタートファイルを置くことで、キックスタートインストールを開始できます。

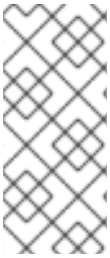
前提条件

- ラベル **OEMDRV** で準備されたボリューム、およびそのルートに **ks.cfg** として存在するキックスタートファイルがあります。

- このボリュームを含むドライブは、インストールプログラムの起動時にシステムで使用できます。

手順

1. ローカルメディア (CD、DVD、USB フラッシュドライブなど) を使用してシステムを起動します。
2. 起動プロンプトで、必要な起動オプションを指定します。
   a. 必要なリポジトリーがネットワーク上にある場合は、**ip=** オプションを使用したネットワークの設定が必要になる場合があります。インストーラーは、このオプションを使用せずに、デフォルトで DHCP プロトコルを使用するすべてのネットワークデバイスを設定しようとします。
   b. 必要なパッケージがインストールされるソフトウェアソースにアクセスするには **inst.repo=** オプションを追加しないといけない場合があります。このオプションを指定しないと、キックスタートファイルでインストールソースを指定する必要があります。インストールソースの詳細は、インストールプログラム設定およびフロー制御のためのキックスタートコマンド を参照してください。
3. 追加した起動オプションを確認してインストールを開始します。
   インストールが開始し、キックスタートファイルが自動的に検出され、自動化されたキックスタートインストールを開始します。



注記

UEFI セキュアブートが有効になっているシステムに、Red Hat Enterprise Linux ベータ版リリースをインストールした場合は、システムの Machine Owner Key (MOK) リストにベータ版の公開鍵を追加します。UEFI セキュアブートおよび Red Hat Enterprise Linux Beta リリースの詳細は、**標準の RHEL 8 インストールの実行** ドキュメントの インストール後のタスクの完了 セクションを参照してください。

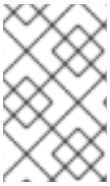
8.6. インストール中のコンソールとロギング

Red Hat Enterprise Linux インストーラーは、**tmux** 端末マルチプレクサーを使用して、メインのインターフェイスのほかに複数の画面を表示し、制御します。この画面は、それぞれ目的が異なり、インストールプロセス中に発生した問題をトラブルシューティングするのに使用できるさまざまなログを表示します。画面の 1 つでは、起動オプションまたはキックスタートコマンドを使用して明示的に無効にしない限り、**root** 権限で使用できる対話式シェルプロンプトを使用できます。

注記

一般的に、インストール関連の問題を診断する必要がなければ、デフォルトのグラフィカルインストール環境から、他の環境に移動する必要はありません。

端末マルチプレクサーは、仮想コンソール 1 で実行しています。インストール環境を、**tmux** に変更する場合は、**Ctrl**+**Alt**+**F1** を押します。仮想コンソール 6 で実行されているメインのインストールインターフェイスに戻るには、**Ctrl**+**Alt**+**F6** を押します。



注記

テキストモードのインストールを選択するには、仮想コンソール 1 (**tmux**) を開始し、その後にコンソール 6 に切り替えると、グラフィカルインターフェイスではなくシェルプロンプトが開きます。

tmux を実行しているコンソールには、利用可能な画面が 5 つあります。その内容と、キーボードショートカットは、以下の表で説明します。キーボードショートカットは 2 段階となっており、最初に **Ctrl**+**b** を押し、両方のキーを離してから、使用する画面で数字キーを押す必要があります。

また、**Ctrl**+**b n**、**Alt+ Tab**、および **Ctrl**+**b p** を使用して、次または前の **tmux** 画面に切り替えることもできます。

表8.3 利用可能な tmux 画面

ショートカット	内容
Ctrl+**b 1**	メインのインストールプログラム画面。テキストベースのプロンプト (テキストモードのインストール中もしくは VNC Direct モードを使用の場合) とデバッグ情報があります。
Ctrl+**b 2**	**root** 権限のある対話式シェルプロンプト。
Ctrl+**b 3**	インストールログ: **/tmp/anaconda.log** に保存されているメッセージを表示します。
Ctrl+**b 4**	ストレージログ - **/tmp/storage.log** に保存されているストレージデバイスおよび設定に関連するメッセージを表示します。
Ctrl+**b 5**	プログラムログ - **/tmp/program.log** に保存されている、インストールプロセス時に実行するユーティリティーのメッセージを表示します。

8.7. キックスタートファイルの維持

キックスタートファイルで自動チェックを実行できます。通常、新規または問題のあるキックスタートファイルが有効であることを確認します。

8.7.1. キックスタートのメンテナンスツールのインストール

キックスタートのメンテナンスツールを使用するには、それを含むパッケージをインストールする必要があります。

手順

- **pykickstart** パッケージをインストールします。

```
# yum install pykickstart
```

8.7.2. キックスタートファイルの確認

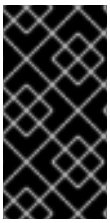
ksvalidator コマンドラインユーティリティーを使用して、キックスタートファイルが有効であることを確認します。これは、キックスタートファイルに大規模な変更を加える際に便利です。**ksvalidator** コマンドで **-v RHEL8** オプションを使用して、RHEL8 クラスの新しいコマンドを承認します。

手順

- キックスタートファイルで **ksvalidator** を実行します。

```
$ ksvalidator -v RHEL8 /path/to/kickstart.ks
```

/path/to/kickstart.ks を、確認するキックスタートファイルのパスに置き換えます。



重要

検証ツールは、インストールの成功を保証しているわけではありません。このツールは、構文が正しく、ファイルに非推奨のオプションが含まれていないことだけを保証します。キックスタートファイルの **%pre** セクション、**%post** セクション、および **%packages** セクションは検証されません。

関連情報

- **ksvalidator(1)** の man ページ

8.8. キックスタートを使用した CDN を介した RHEL の登録およびインストール

本セクションでは、キックスタートを使用して、システムを登録し、RHEL サブスクリプションを割り当て、Red Hat コンテンツ配信ネットワーク (CDN) からインストールする方法を説明します。

8.8.1. CDN から RHEL の登録およびインストール

この手順に従って、システムを登録して、RHEL サブスクリプションを割り当て、キックスタートコマンドの **rhsm** を使用して Red Hat コンテンツ配信ネットワーク (CDN) からインストールします。これは、**syspurpose** コマンドと Red Hat Insights に対応しています。キックスタートコマンド **rhsm** は、システムの登録時にカスタムの **%post** スクリプトを使用する要件を削除します。

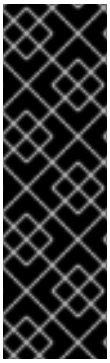
重要

CDN 機能は、**Boot ISO** および **DVD ISO** のイメージファイルでサポートされています。ただし、**Boot ISO** イメージファイルのインストールソースのデフォルトは CDN であるため、Boot ISO イメージファイルを使用することが推奨されます。

前提条件

- CDN にアクセスできるネットワークに接続されている。
- キックスタートファイルを作成し、リムーバブルメディア、ハードドライブ、または HTTP(S)、FTP、NFS サーバーを使用するネットワーク上の場所でインストールプログラムから使用できるようにしました。
- インストールするシステムからアクセス可能な場所にキックスタートファイルがある。

- インストールの開始に使用するブートメディアを作成し、インストールソースをインストールプログラムで使用できるようにしました。



重要

- システム登録後に使用されるインストールソースリポジトリーは、システムの起動方法により異なります。詳細は、**標準的な RHEL 8 インストールの実行** の**システム登録後のインストールソースリポジトリー** セクションを参照してください。
- サブスクリプションは、システムがアクセスできる CDN サブセットとリポジトリーを管理するため、キックスタートファイルではリポジトリー設定は必要ありません。

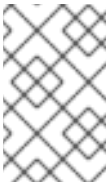
手順

1. キックスタートファイルを開きます。
2. このファイルに、**rhsm** キックスタートコマンドとそのオプションを追加します。

組織 (必須)

組織 ID を入力します。以下に例を示します。

```
--organization=1234567
```



注記

セキュリティー上の理由から、CDN から登録してインストールする場合、Red Hat のユーザー名およびパスワードアカウントの詳細はキックスタートでは対応していません。

アクティベーションキー (必須)

アクティベーションキーを入力します。サブスクリプションにアクティベーションキーが登録されている限り、複数の鍵を使用できます。以下に例を示します。

```
--activation-key="Test_key_1" --activation-key="Test_key_2"
```

Red Hat Insights (推奨)

ターゲットシステムを Red Hat Insights に接続します。

注記

Red Hat Insights は SaaS (Software-as-a-Service) 製品で、継続的に、登録済みの Red Hat ベースのシステムに詳細な分析を提供し、物理環境、仮想環境、クラウド環境、およびコンテナーデプロイメントでセキュリティー、パフォーマンス、および安定性に関する脅威をプロアクティブに特定します。インストーラー GUI を使用した手動インストールとは異なり、キックスタートの使用時には、Red Hat Insights への接続はデフォルトで有効になっていません。

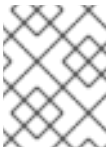
以下に例を示します。

```
--connect-to-insights
```

HTTP プロキシー (任意)

HTTP プロキシーを設定します。以下に例を示します。

```
--proxy="user:password@hostname:9000"
```



注記

ホスト名のみが必須です。認証のないデフォルトポートでプロキシーを実行する必要がある場合は、オプションが **--proxy="hostname"** になります。

システムの目的 (任意)

次のコマンドを使用して、システムの目的のロール、SLA、使用方法を設定します。

```
subscription-manager syspurpose role --set="Red Hat Enterprise Linux Server" --
sla="Premium" --usage="Production"
```

例

次の例では、すべてのキックスタートコマンドの **rhsm** オプションを含む最小限のキックスタートファイルを表示しています。

```
graphical
lang en_US.UTF-8
keyboard us
rootpw 12345
timezone America/New_York
zerombr
clearpart --all --initlabel
autopart
syspurpose --role="Red Hat Enterprise Linux Server" --sla="Premium" --
usage="Production"
rhsm --organization="12345" --activation-key="test_key" --connect-to-insights --
proxy="user:password@hostname:9000"
reboot
%packages
vim
%end
```

3. キックスタートファイルを保存し、インストールプロセスを開始します。

関連情報

- システムの目的の設定
- キックスタートインストールの開始
- Red Hat Insights product documentation
- Understanding Activation Keys

- Subscription Manager の HTTP プロキシーの設定は、**subscription-manager** man ページの **PROXY CONFIGURATION** セクションを参照してください。

8.8.2. CDN からシステム登録の確認

以下の手順に従って、システムが CDN に登録されていることを確認します。

前提条件

- CDN を使用した登録とインストール に記載されているように、登録とインストールのプロセスを完了している。
- Starting Kickstart installations に従って、キックスタートインストールを開始しています。
- インストール済みシステムを再起動して、端末画面が開いている。

手順

1. 端末画面で、**root** ユーザーとしてログインして、登録を確認します。

```
# subscription-manager list
```

出力には、割り当てられているサブスクリプションの詳細が表示されます。以下に例を示します。

```
Installed Product Status

Product Name: Red Hat Enterprise Linux for x86_64
Product ID: 486
Version: X
Arch: x86_64
Status: Subscribed
Status Details
Starts: 11/4/2019
Ends: 11/4/2020
```

2. 詳細なレポートを表示するには、次のコマンドを実行します。

```
# subscription-manager list --consumed
```

8.8.3. CDN からシステムの登録解除

以下の手順に従って、Red Hat CDN からシステムの登録を解除します。

前提条件

- Registering and installing RHEL from the CDN に従って、登録およびインストールプロセスを完了している。
- Starting Kickstart installations に従って、キックスタートインストールを開始しています。
- インストール済みシステムを再起動して、端末画面が開いている。

手順

- 端末画面で、**root** ユーザーとしてログインし、登録を解除します。

```
# subscription-manager unregister
```

システムに割り当てられていたサブスクリプションが削除され、CDN への接続が削除されます。

8.9. VNC を使用したリモート RHEL インストールの実行

このセクションでは、Virtual Network Computing (VNC) を使用して、リモート RHEL インストールを実行する方法を説明します。

8.9.1. 概要

CD、DVD、または USB フラッシュドライブから、または PXE を使用してネットワークからシステムを起動して RHEL をインストールする場合は、グラフィカルユーザーインターフェイスを使用することが推奨されます。ただし、IBM Power Systems や 64 ビットの IBM Z など、多くのエンタープライズシステムは、自動的に実行されても、ディスプレイ、キーボード、およびマウスには接続されていないリモートのデータセンター環境に置かれています。通常、これらのシステムは **ヘッドレスシステム** と呼ばれ 、通常はネットワーク接続で制御されます。RHEL インストールプログラムには、ターゲットマシンでグラフィカルインストールを実行する Virtual Network Computing (VNC) インストールが含まれます。ただし、グラフィカルインストールの制御はネットワーク上の別のシステムで処理されます。
RHEL インストールプログラムでは、2 つの VNC インストールモードを利用できます。**Direct** および **Connect** です。接続が確立されれば、この 2 つのモードに違いはありません。選択するモードは、環境によって異なります。

Direct モード

Direct モードでは、RHEL インストールプログラムがターゲットシステムで起動するように設定されています。また、続行前に別のシステムにインストールされている VNC ビューアーを待ちます。Direct モードインストールの一環として、IP アドレスとポートがターゲットシステムに表示されます。VNC ビューアーを使用することで、IP アドレスとポートを使用して、ターゲットシステムにリモートで接続し、グラフィカルインストールを完了できます。

Connect モード

Connect モードでは、VNC ビューアーが **リスニング** モードでリモートシステムで開始されます。VNC ビューアーは、指定したポート上のターゲットシステムからの着信接続を待ちます。RHEL インストールプログラムがターゲットシステムで起動すると、起動オプションまたはキックスタートコマンドを使用して、システムのホスト名とポート番号が提供されます。次に、インストールプログラムは指定したシステムホスト名およびポート番号を使用して、リスニング VNC ビューアーで接続を確立します。Connect モードを使用するには、リッスンしている VNC ビューアーを持つシステムが、着信ネットワーク接続を許可できる必要があります。

8.9.2. 留意事項

VNC を使用してリモート RHEL インストールを実行する場合は、以下の項目を考慮してください。

- **VNC クライアントアプリケーション**:VNC クライアントアプリケーションは、VNC Direct および Connect インストールの両方を実行する必要があります。VNC クライアントアプリケーションは多くの Linux ディストリビューションのリポジトリーで入手できます。また、Windows などの他のオペレーティングシステムにもには、無料の VNC クライアントアプリケーションを利用できます。RHEL では、以下の VNC クライアントアプリケーションを利用できます。

    a. **tigervnc** はデスクトップ環境から独立しており、**tigervnc** パッケージの一部としてインス

- **tigervnc** はデスクトップ環境から独立しており、**tigervnc** パッケージの一部としてインストールされます。
- **vinagre** は GNOME デスクトップ環境に含まれており、**vinagre** パッケージの一部としてインストールされます。

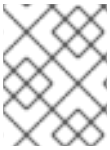
注記

VNC サーバーはインストールプログラムに含まれているため、インストールは不要です。

- **ネットワークおよびファイアウォール:**
  - ターゲットシステムがファイアウォールにより着信接続が許可されていない場合は、Connect モードを使用するかファイアウォールを無効にする必要があります。ファイアウォールを無効にすると、セキュリティーに影響を及ぼす可能性があります。
  - VNC ビューアーを実行しているシステムがファイアウォールにより着信接続を許可されていない場合は、Direct モードを使用するか、ファイアウォールを無効にする必要があります。ファイアウォールを無効にすると、セキュリティーに影響を及ぼす可能性があります。ファイアウォールの設定に関する詳細情報は、セキュリティーの強化 を参照してください。
- **カスタム起動オプション**VNC インストールを開始するにはカスタムの起動オプションを指定する必要があります。インストール手順はシステムのアーキテクチャーによって異なる可能性があります。
- **キックスタートインストールの VNC:**キックスタートでは、VNC 固有のコマンドを使用できます。**vnc** コマンドのみを使用すると、Direct モードで RHEL インストールを実行します。Connect モードでインストールを設定するために追加オプションを使用することができます。

8.9.3. VNC Direct モードでのリモート RHEL インストールの実行

この手順では、VNC Direct モードで、リモート RHEL インストールを実行します。Direct モードでは、RHEL にインストールされているターゲットシステムへの接続を VNC ビューアーにより開始されることが想定されます。この手順では、VNC ビューアーを使用するシステムが、**リモート** システムと呼ばれます。RHEL インストールプログラムにより、リモートシステムの VNC ビューアーからターゲットシステムへの接続を開始することが求められます。



注記

以下の手順では、VNC ビューアーとして **TigerVNC** を使用します。その他のビューアーの手順は異なる場合がありますが、一般的な原則が適用されます。

前提条件

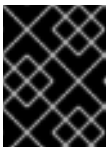
- root ユーザーとして、リモートシステムに VNC ビューアーをインストールした。
- ネットワークブートサーバーを設定して、ターゲットシステムでインストールを起動した。

手順

1. ターゲットシステムの RHEL ブートメニューから、キーボードの **Tab** キーを押して、起動オプションを編集します。

2. **inst.vnc** オプションをコマンドラインの最後に追加します。

    a. インストールしているシステムに VNC アクセスを制限する場合は、コマンドラインの末尾に **inst.vncpassword=PASSWORD** 起動オプションを追加します。**PASSWORD** をインストールに使用するパスワードに置き換えます。VNC パスワードは 6 文字から 8 文字に設定する必要があります。



重要

inst.vncpassword= オプションには一時的なパスワードを使用してください。既存のパスワードまたは root パスワードは指定しないでください。

3. **Enter** を押してインストールを開始します。ターゲットシステムはインストールプログラムを初期化し、必要なサービスを開始します。システムの準備ができると、システムの IP アドレスとポート番号を示すメッセージが表示されます。

4. リモートシステムで VNC ビューアーを開きます。

5. **VNC サーバー** フィールドに IP アドレスとポート番号を入力します。

6. **Connect** をクリックします。

7. VNC パスワードを入力して、**OK** をクリックします。新しいウィンドウが開き、VNC 接続が確立され、RHEL インストールメニューが表示されます。このウィンドウから、グラフィカルユーザーインターフェイスを使用して、ターゲットシステムに RHEL をインストールできます。

8.9.4. VNC Connect モードでのリモート RHEL インストールの実行

この手順を使用して、VNC Connect モードで、リモート RHEL インストールを実行します。Connect モードでは、RHEL でインストールするターゲットシステムが、別のシステムにインストールされている VNC ビューアーに接続を開始します。この手順では、VNC ビューアーを使用するシステムが、**リモート** システムと呼ばれます。

注記

以下の手順では、VNC ビューアーとして **TigerVNC** を使用します。その他のビューアーの手順は異なる場合がありますが、一般的な原則が適用されます。

前提条件

- root ユーザーとして、リモートシステムに VNC ビューアーをインストールした。
- ターゲットシステムでインストールを開始するよう、ネットワーク起動サーバーを設定している。
- VNC Connect インストールに対して起動オプションを使用するようにターゲットシステムを設定している。
- VNC ビューアーでリモートシステムが必要なポートで着信接続を受け入れるよう設定されていることを確認している。検証は、ネットワークとシステム設定によって異なります。詳細は、セキュリティーの強化 および ネットワークのセキュリティー保護 を参照してください。

手順

1. 以下のコマンドを実行して、リモートシステム上で VNC ビューアーを **リスニングモード** で開始します。

```
$ vncviewer -listen PORT
```

2. PORT は、接続に使用されるポート番号に置き換えます。

3. 端末には、ターゲットシステムからの着信接続を待機していることを示すメッセージが表示されます。

```
TigerVNC Viewer 64-bit v1.8.0
Built on: 2017-10-12 09:20
Copyright (C) 1999-2017 TigerVNC Team and many others (see README.txt)
See http://www.tigervnc.org for information on TigerVNC.

Thu Jun 27 11:30:57 2019
 main:       Listening on port 5500
```

4. ターゲットシステムをネットワークから起動します。

5. ターゲットシステムの RHEL ブートメニューから、キーボードの **Tab** キーを押して、起動オプションを編集します。

6. **inst.vnc inst.seLinuxOptions=HOST:PORT** オプションをコマンドラインの最後に追加します。

7. **HOST** には、リッスンしている VNC ビューアーを実行しているリモートシステムの IP アドレス、**PORT** には、VNC ビューアーがリッスンしているポート番号を入力します。

8. **Enter** を押してインストールを開始します。システムはインストールプログラムを初期化し、必要なサービスを開始します。初期化プロセスが終了すると、インストールプログラムは指定の IP アドレスとポートへの接続を試行します。

9. 接続に成功すると、新しいウィンドウが開き、VNC 接続が確立され、RHEL インストールメニューが表示されます。このウィンドウから、グラフィカルユーザーインターフェイスを使用して、ターゲットシステムに RHEL をインストールできます。

第9章 高度な設定オプション

9.1. システムの目的の設定

システムの目的を使用して、Red Hat Enterprise Linux 8 システムの使用目的を記録します。システムの目的を設定すると、エンタイトルメントサーバーは最も適切なサブスクリプションを自動接続できます。本セクションは、キックスタートを使用して、システムの目的を設定する方法を説明します。

次の利点があります。

- システム管理および事業運営に関する詳細なシステムレベルの情報
- システムを調達した理由とその目的を判断する際のオーバーヘッドを削減
- Subscription Manager の自動割り当てと、システムの使用状況の自動検出および調整のカスタマーエクスペリエンスの向上

9.1.1. 概要

以下のいずれかの方法でシステムの目的のデータを入力できます。

- イメージの作成時
- **Connect to Red Hat**画面を使用してシステムを登録し、Red Hat サブスクリプションを割り当てる際の GUI インストール時
- **syspurpose Kickstart** コマンドを使用したキックスタートインストール時
- **syspurpose** コマンドラインツール (CLI) を使用したインストール後

システムの目的を記録するために、システムの目的の以下のコンポーネントを設定できます。選択された値は、登録時にエンタイトルメントサーバーが、システムに最適なサブスクリプションを割り当てるのに使用されます。

ロール

- Red Hat Enterprise Linux Server
- Red Hat Enterprise Linux Workstation
- Red Hat Enterprise Linux Compute Node

サービスレベルアグリーメント

- Premium
- Standard
- Self-Support

用途

- Production
- Development/Test

- Disaster Recovery

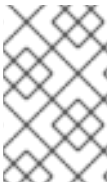
関連情報

- RHEL システムイメージのカスタマイズ
- 高度な RHEL 8 インストールの実行
- Red Hat Subscription Manager の使用および設定

9.1.2. キックスタートでシステムの目的の設定

以下の手順に従って、インストール時にシステムの目的を設定します。これを行うには、キックスタート設定ファイルで、キックスタートコマンドの **syspurpose** を使用します。

システムの目的は Red Hat Enterprise Linux インストールプログラムでは任意の機能ですが、最適なサブスクリプションを自動的にアタッチするためにシステムの目的を設定することを強く推奨します。



注記

インストール完了後にシステムの目的を有効にすることもできます。これを行うには、**syspurpose** コマンドラインツールを使用します。**syspurpose** ツールのコマンドは、**syspurpose** Kickstart コマンドとは異なります。

キックスタートコマンド **syspurpose** では、以下のアクションが利用可能です。

ロール

システムで計画しているロールを設定します。このアクションは以下の形式を使用します。

```
syspurpose --role=
```

割り当てられるロールは以下のとおりです。

- **Red Hat Enterprise Linux Server**
- **Red Hat Enterprise Linux Workstation**
- **Red Hat Enterprise Linux Compute Node**

SLA

システムで計画している SLA を設定します。このアクションは以下の形式を使用します。

```
syspurpose --sla=
```

割り当てられる SLA は以下の通りです。

- **Premium**
- **Standard**
- **Self-Support**

使用方法

システムで計画している使用目的を設定します。このアクションは以下の形式を使用します。

```
syspurpose --usage=
```

割り当てられる使用方法は以下の通りです。

- **Production**
- **Development/Test**
- **障害復旧**

アドオン

追加のレイヤード製品または機能。複数のアイテムを追加するには、階層化製品/機能ごとに1回使用する **--addon** を複数回指定します。このアクションは以下の形式を使用します。

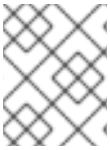
```
syspurpose --addon=
```

9.1.3. 関連情報

- Configuring System Purpose using the **subscription-manager** command-line tool

9.2. インストール時のドライバーの更新

本セクションは、Red Hat Enterprise Linux インストールプロセス時にドライバーの更新を完了する方法を説明します。



注記

これは、インストールプロセスの任意の手順です。Red Hat は、必要な場合を除いて、ドライバーの更新は行わないことを推奨します。

前提条件

- Red Hat、ハードウェアベンダー、または信頼できるサードパーティーベンダーから、Red Hat Enterprise Linux のインストール時に、ドライバー更新が必要になることが通知されている。

9.2.1. 概要

Red Hat Enterprise Linux は、多数のハードウェアデバイス用のドライバーに対応していますが、新たにリリースしたドライバーには対応していない可能性があります。ドライバーの更新は、そのドライバーが対応していないために、インストールが完了できない場合に限り、実行する必要があります。インストール中にドライバーを更新することは、通常、特定の設定に対応する場合に限り必要になります。たとえば、システムのストレージデバイスへのアクセスを提供するストレージアダプター用ドライバーをインストールします。

警告

ドライバー更新ディスクは、競合するカーネルドライバーを無効にする場合があります。この方法でカーネルモジュールをアンロードすると、インストールエラーが発生することがあります。

9.2.2. ドライバー更新の種類

Red Hat、ハードウェアベンダー、信頼できるサードパーティーは、ドライバー更新を ISO イメージファイルとして提供します。ISO イメージファイルを受け取ったら、ドライバー更新の種類を選択してください。

ドライバー更新の種類

自動

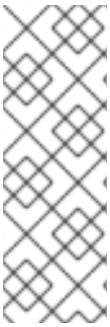
推奨されるドライバーの更新方法です。**OEMDRV** とラベルが付いたストレージデバイス (CD、DVD、または USB フラッシュドライブ) が、そのシステムに物理的に接続されます。インストールの開始時に、**OEMDRV** ストレージデバイスが存在する場合は、それがドライバー更新ディスクのように扱われ、インストールプログラムはそのドライバーを自動的に読み込みます。

アシスト付き

このインストールプログラムは、ドライバーの更新を指定するように促します。**OEMDRV** 以外の任意のローカルストレージデバイスラベルを使用できます。インストールを開始するときに、**inst.dd** ブートオプションが指定されます。このオプションにパラメーターを付けずに使用すると、インストールプログラムはシステムに接続されているすべてのストレージデバイスを表示し、ドライバー更新を含むデバイスを選択するように促します。

手動

ドライバー更新イメージまたは RPM パッケージのパスを手動で指定します。**OEMDRV** 以外のラベルを持つ任意のローカルストレージ、またはインストールシステムからアクセス可能なネットワークの場所を使用できます。**inst.dd=location** 起動オプションは、インストールの開始時に指定しますが、**location** は、ドライバー更新ディスクまたは ISO イメージのパスになります。このオプションを指定すると、インストールプログラムは特定の場所にあるドライバー更新を読み込みます。手動でドライバーを更新する場合は、ローカルストレージデバイス、またはネットワークの場所 (HTTP、HTTPS、または FTP サーバー) を指定できます。



注記

- **inst.dd=location** と **inst.dd** の両方を同時に使用できます。**location** は、ドライバー更新ディスクまたは ISO イメージのパスになります。このシナリオでは、インストールプログラムは、その場所から、利用可能なドライバーの更新を読み込み、ドライバーの更新が含まれるデバイスを選択するように求められます。
- ネットワークの場所からドライバーの更新を読み込むときは、**ip= option** を使用してネットワークを初期化します。

制限

セキュアブート技術を使用する UEFI システムでは、すべてのドライバーが有効な証明書で署名されている必要があります。Red Hat ドライバーは、Red Hat の秘密鍵のいずれかで署名され、カーネルで対応する公開鍵により認証されます。追加で別のドライバーを読み込む場合は、それが署名されているこ

とを確認してください。

9.2.3. ドライバー更新の準備

この手順では、CD および DVD でドライバー更新の準備を行う方法を説明します。

前提条件

- Red Hat、ハードウェアベンダー、または信頼できるサードパーティーベンダーからドライバー更新の ISO イメージを受け取っている。
- ドライバー更新の ISO イメージを CD または DVD に焼き付けている。

警告

CD または DVD で、**.iso** で終了する ISO イメージファイルが 1 つしか利用できない場合、書き込み処理は成功していません。CD または DVD に ISO イメージを作成する方法は、システムの書き込みソフトウェアのドキュメントを参照してください。

手順

1. ドライバー更新用 CD または DVD をシステムの CD/DVD ドライブに挿入し、システムのファイルマネージャーツールで参照します。
2. **rhdd3** ファイルが 1 つ利用できることを確認してください。**rhdd3** は、ドライバーの説明が含まれる署名ファイルと、ディレクトリーの **rpms** です。このディレクトリーには、さまざまなアーキテクチャー用のドライバーが同梱される RPM パッケージが含まれます。

9.2.4. 自動ドライバー更新の実行

この手順では、インストール時にドライバーの自動更新を行う方法を説明します。

前提条件

- **OEMDRV** ラベルの付いた標準のディスクパーティションにドライバーの更新イメージを置くか、**OEMDRV** ドライバー更新イメージを CD または DVD に作成します。RAID や LVM ボリュームなどの高度なストレージは、ドライバーの更新プロセス中はアクセスできない可能性があります。
- インストールプロセスを開始する前に、ボリュームラベル **OEMDRV** が付いたブロックデバイスをシステムに接続しているか、事前に準備した CD または DVD をシステムの CD/DVD ドライブに挿入している。

手順

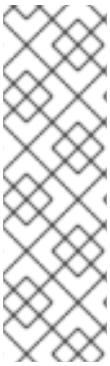
- 前提条件の手順を完了すると、インストールプログラムの起動時にドライバーが自動的にロードされ、システムのインストールプロセス中にインストールされます。

9.2.5. アシスト付きドライバー更新の実行

この手順では、インストール時に、ドライバーのアシスト付き更新を行う方法を説明します。

前提条件

- インストールプロセスを開始する前に、**OEMDRV** ボリュームラベルのないブロックデバイスをシステムに接続し、ドライバーディスクイメージをこのデバイスにコピーしたか、ドライバー更新の CD または DVD を準備して、システムの CD または DVD ドライブに挿入しました。

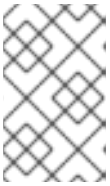


注記

CD または DVD に ISO イメージファイルを書き込んだにもかかわらず、**OEMDRV** ボリュームラベルがない場合は、引数を追加せずに **inst.dd** オプションを使用できます。インストールプログラムは、CD または DVD からドライバーをスキャンして選択するオプションを提供します。このシナリオでは、インストールプログラムから、ドライバー更新用 ISO イメージを選択するように求められません。別のシナリオでは、起動オプション **inst.dd=location** で CD または DVD を使用します。これにより、インストールプログラムが、ドライバー更新に CD または DVD を自動的にスキャンできるようになります。詳細は、ドライバーの手動更新の実行 を参照してください。

手順

1. ブートメニューウィンドウで **Tab** キーを押して、ブートコマンドラインを表示します。
2. 起動オプション **inst.dd** をコマンドラインに追加し、**Enter** を押して起動プロセスを実行します。
3. メニューから、ローカルディスクパーティション、もしくは CD デバイスまたは DVD デバイスを選択します。インストールプログラムが ISO ファイル、または ドライバー更新 RPM パッケージをスキャンします。
4. オプション:ドライバー更新 ISO ファイルを選択します。



注記

選択したデバイスまたはパーティション (ドライバー更新 CD または DVD を含む光学ドライブなど) に、ISO イメージファイルではなく、ドライバー更新 RPM パッケージが含まれる場合は、この手順は必要ありません。

5. 必要なドライバーを選択します。
   a. キーボードの数字キーを使用して、ドライバー選択を切り替えます。
   b. **c** を押して、選択したドライバーをインストールします。選択したドライバーが読み込まれ、インストールプロセスが始まります。

9.2.6. 手動によるドライバー更新の実行

この手順では、インストール時にドライバーを手動で更新する方法を説明します。

前提条件

- ドライバー更新の ISO イメージファイルを USB フラッシュドライブまたは Web サーバーに配置し、コンピューターに接続しました。

手順

1. ブートメニューウィンドウで **Tab** キーを押して、ブートコマンドラインを表示します。
2. **inst.dd=location** 起動オプションをコマンドに追加します。場所は、ドライバー更新のファイルがある場所です。通常、イメージファイルは Web サーバー (http://server.example.com/dd.iso など) または USB フラッシュドライブ (**/dev/sdb1** など) に置きます。ドライバー更新を含む RPM パッケージ (http://server.example.com/dd.rpm など) を指定することもできます。
3. **Enter** を押して、起動プロセスを実行してください。指定した場所で利用可能なドライバーが自動的に読み込まれ、インストールプロセスが始まります。

関連情報

- The **inst.dd** boot option

9.2.7. ドライバーの無効

この手順では、誤動作しているドライバーを無効にする方法を説明します。

前提条件

- インストールプログラムブートメニューを起動している。

手順

1. ブートメニューで **Tab** キーを押して、ブートコマンドラインを表示します。
2. 起動オプション **modprobe.blacklist=driver_name** をコマンドラインに追加します。
3. **driver_name** を、無効にするドライバーの名前に置き換えます。以下に例を示します。

```
modprobe.blacklist=ahci
```

起動オプション **modprobe.blacklist=** を使用して無効にしたドライバーは、インストール済みシステムで無効になり、**/etc/modprobe.d/anaconda-blacklist.conf** ファイルに表示されます。

4. **Enter** を押して、起動プロセスを実行してください。

9.3. PXE を使用してネットワークからインストールするための準備

本セクションでは、PXE 起動およびネットワークインストールを有効にするために、PXE サーバーで TFTP および DHCP を設定する方法を説明します。

9.3.1. ネットワークインストールの概要

ネットワークインストールでは、インストールサーバーへのアクセスがあるシステムに、Red Hat Enterprise Linux をインストールできます。ネットワークインストールには、少なくとも 2 つのシステムが必要です。

PXE サーバー

DHCP サーバー、TFTP サーバー、および HTTP サーバー、HTTPS サーバー、FTP サーバー、または NFS サーバーを実行しているシステム。各サーバーが稼働する物理システムが同じである必要は

ありませんが、本セクションの手順では、1つのシステムですべてのサーバーが稼働していることが想定されています。

クライアント

Red Hat Enterprise Linux をインストールしているシステム。インストールが開始すると、クライアントは DHCP サーバーに問い合わせ、TFTP サーバーからブートファイルを受け取り、HTTP サーバー、HTTPS サーバー、FTP サーバー、または NFS サーバーからインストールイメージをダウンロードします。その他のインストール方法とは異なり、クライアントはインストールを開始するのに物理的な起動メディアを必要としません。

注記

ネットワークからクライアントをブートするには、BIOS/UEFI またはクイックブートメニューで設定します。ハードウェアによっては、ネットワークから起動するオプションが無効になっていたり、利用できない場合があります。

PXE を使用してネットワークから Red Hat Enterprise Linux をインストールする準備を行う手順は次のとおりです。

手順

1. インストール ISO イメージまたはインストールツリーを NFS サーバー、HTTPS サーバー、HTTP サーバー、または FTP サーバーにエクスポートします。
2. TFTP サーバーおよび DHCP サーバーを設定し、PXE サーバーで TFTP サービスを開始します。
3. クライアントを起動して、インストールを開始します。

重要

GRUB2 ブートローダーは、TFTP サーバーの他に、HTTP からのネットワークブートをサポートします。このプロトコルからブートファイル (カーネルおよび初期 RAM ディスク (**vmlinuz** および **initrd**)) を送ると時間がかかり、タイムアウトが発生する場合もあります。HTTP サーバーにはこのリスクがありませんが、ブートファイルを送信する場合は TFTP サーバーを使用することが推奨されます。

関連情報

- キックスタートインストール用のインストールソースの作成
- BIOS ベースのクライアント向けに TFTP サーバーの設定
- UEFI ベースのクライアント向けに TFTP サーバーの設定
- IBM Power システム用のネットワークサーバーの設定
- Red Hat Satellite の製品ドキュメント

9.3.2. BIOS ベースのクライアント向けに TFTP サーバーの設定

この手順に従って、TFTP サーバーおよび DHCP サーバーを設定し、BIOS ベースの AMD および Intel の 64 ビットシステム用 PXE サーバーで、TFTP サービスを開始します。

重要

本セクションのすべての設定ファイルは例となります。設定の詳細は、アーキテクチャーや特定の要件によって異なります。

手順

1. root で、次のパッケージをインストールします。ネットワークに DHCP サーバーがすでに設定されている場合は、**dhcp-server** パッケージを除外します。

```
# yum install tftp-server dhcp-server
```

2. ファイアウォールで、**tftp service** サービスへの着信接続を許可します。

```
# firewall-cmd --add-service=tftp
```

注記

- このコマンドは、次にサーバーを再起動するまで、一時的にアクセスを有効にします。永続的アクセスを有効にするには、コマンドに **--permanent** オプションを追加します。
- ISO インストールファイルの場所によっては、HTTP などのサービスの着信接続を許可しないといけない場合があります。

3. 以下の例の **/etc/dhcp/dhcpd.conf** ファイルのように、**SYSLINUX** に同梱されているブートイメージを使用するように DHCP サーバーを設定します。DHCP サーバーがすでに設定されている場合は、DHCP サーバーでこの手順を実行します。

```
option space pxelinux;
option pxelinux.magic code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.reboottime code 211 = unsigned integer 32;
option architecture-type code 93 = unsigned integer 16;

subnet 10.0.0.0 netmask 255.255.255.0 {
 option routers 10.0.0.254;
 range 10.0.0.2 10.0.0.253;

 class "pxeclients" {
   match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
   next-server 10.0.0.1;

   if option architecture-type = 00:07 {
     filename "BOOTX64.EFI";
     } else {
     filename "pxelinux/pxelinux.0";
   }
 }
}
```

4. DVD ISO イメージファイルの **SYSLINUX** パッケージから **pxelinux.0** ファイルにアクセスします。ここで、**my_local_directory** は、作成するディレクトリーの名前です。

```
# mount -t iso9660 /path_to_image/name_of_image.iso /mount_point -o loop,ro
```

```
# cp -pr /mount_point/BaseOS/Packages/syslinux-tftpboot-version-architecture.rpm
/my_local_directory
```

```
# umount /mount_point
```

5. パッケージをデプロイメントします。

```
# rpm2cpio syslinux-tftpboot-version-architecture.rpm | cpio -dimv
```

6. **tftpboot**/ に **pxelinux**/ ディレクトリーを作成し、そのディレクトリーから **pxelinux**/ ディレクトリーにすべてのファイルをコピーします。

```
# mkdir /var/lib/tftpboot/pxelinux
```

```
# cp my_local_directory/tftpboot/* /var/lib/tftpboot/pxelinux
```

7. **pxelinux**/ ディレクトリー内に **pxelinux.cfg**/ ディレクトリーを作成します。

```
# mkdir /var/lib/tftpboot/pxelinux/pxelinux.cfg
```

8. **default** という名前の設定ファイルを作成し、以下の例のように **pxelinux.cfg**/ ディレクトリーに追加します。

```
default vesamenu.c32
prompt 1
timeout 600

display boot.msg

label linux
  menu label ^Install system
  menu default
  kernel images/RHEL-8/vmlinuz
  append initrd=images/RHEL-8/initrd.img ip=dhcp inst.repo=http://10.32.5.1/RHEL-
8/x86_64/iso-contents-root/
label vesa
  menu label Install system with ^basic video driver
  kernel images/RHEL-8/vmlinuz
  append initrd=images/RHEL-8/initrd.img ip=dhcp inst.xdriver=vesa nomodeset
inst.repo=http://10.32.5.1/RHEL-8/x86_64/iso-contents-root/
label rescue
  menu label ^Rescue installed system
  kernel images/RHEL-8/vmlinuz
  append initrd=images/RHEL-8/initrd.img rescue
label local
  menu label Boot from ^local drive
  localboot 0xffff
```

注記

- このランタイムイメージなしでは、インストールプログラムは起動できません。**inst.stage2** 起動オプションを使用して、イメージの場所を指定します。または、**inst.repo=** オプションを使用して、イメージおよびインストールソースを指定することも可能です。
- **inst.repo** で使用したインストールソースの場所には、有効な **treeinfo** ファイルが含まれている必要があります。
- インストールソースとして RHEL8 インストール DVD を選択すると、**.treeinfo** ファイルが BaseOS リポジトリーおよび AppStream リポジトリーを指定します。単一の **inst.repo** オプションを使用することで両方のリポジトリーを読み込むことができます。

9. **/var/lib/tftpboot/** ディレクトリーに、ブートイメージファイルを保存するサブディレクトリーを作成し、そのディレクトリーにブートイメージファイルをコピーします。この例のディレクトリーは、**/var/lib/tftpboot/pxelinux/images/RHEL-8/** になります。

```
# mkdir -p /var/lib/tftpboot/pxelinux/images/RHEL-8/
# cp /path_to_x86_64_images/pxeboot/{vmlinuz,initrd.img}
/var/lib/tftpboot/pxelinux/images/RHEL-8/
```

10. DHCP サーバーで **dhcpd** サービスを開始して有効にします。localhost で DHCP サーバーを設定している場合は、ローカルホストで **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

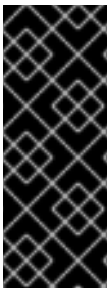
11. **tftp.socket** サービスを開始して有効にします。

```
# systemctl start tftp.socket
# systemctl enable tftp.socket
```

これにより、PXE 起動サーバーでは、PXE クライアントにサービスを提供する準備が整いました。クライアント (Red Hat Enterprise Linux のインストール先システム) を起動し、起動ソースを指定するように求められたら、**PXE ブート** を選択してネットワークインストールを開始できます。

9.3.3. UEFI ベースのクライアント向けに TFTP サーバーの設定

この手順に従って、TFTP サーバーおよび DHCP サーバーを設定し、UEFI ベースの AMD64、Intel 64、および 64 ビット ARM システム用に、PXE サーバーで TFTP サービスを開始する方法を説明します。



重要

- 本セクションのすべての設定ファイルは例となります。設定の詳細は、アーキテクチャーや特定の要件によって異なります。
- Red Hat Enterprise Linux 8 UEFI PXE ブートは、MAC ベースの grub メニューファイルで小文字のファイル形式に対応します。たとえば、grub2 の MAC アドレスのファイル形式は **grub.cfg-01-aa-bb-cc-dd-ee-ff** です。

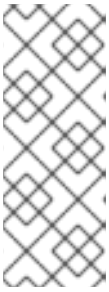
手順

1. root で、次のパッケージをインストールします。ネットワークに DHCP サーバーが設定されている場合は、dhcp-server パッケージを除外します。

```
# yum install tftp-server dhcp-server
```

2. ファイアウォールで、**tftp service** サービスへの着信接続を許可します。

```
# firewall-cmd --add-service=tftp
```



注記

- このコマンドは、次にサーバーを再起動するまで、一時的にアクセスを有効にします。永続的アクセスを有効にするには、コマンドに **--permanent** オプションを追加します。
- ISO インストールファイルの場所によっては、HTTP などのサービスの着信接続を許可しないといけない場合があります。

3. 以下の例の **/etc/dhcp/dhcpd.conf** ファイルのように、**shim** に同梱されているブートイメージを使用するように DHCP サーバーを設定します。DHCP サーバーがすでに設定されている場合は、DHCP サーバーでこの手順を実行します。

```
option space pxelinux;
option pxelinux.magic code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.reboottime code 211 = unsigned integer 32;
option architecture-type code 93 = unsigned integer 16;

subnet 10.0.0.0 netmask 255.255.255.0 {
 option routers 10.0.0.254;
 range 10.0.0.2 10.0.0.253;

 class "pxeclients" {
   match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
   next-server 10.0.0.1;

   if option architecture-type = 00:07 {
     filename "BOOTX64.EFI";
   } else {
     filename "pxelinux/pxelinux.0";
  }
  }
}
```

4. DVD ISO イメージファイルにある **shim** パッケージの **BOOTX64.EFI** ファイルと、**grub2-efi** パッケージの **grubx64.efi** ファイルにアクセスします。ここで、**my_local_directory** は作成するディレクトリーの名前になります。

```
# mount -t iso9660 /path_to_image/name_of_image.iso /mount_point -o loop,ro
```

```
# cp -pr /mount_point/BaseOS/Packages/shim-version-architecture.rpm /my_local_directory
```

```
# cp -pr /mount_point/BaseOS/Packages/grub2-efi-version-architecture.rpm
/my_local_directory
```

```
# umount /mount_point
```

5. パッケージを抽出します。

```
# rpm2cpio shim-version-architecture.rpm | cpio -dimv
```

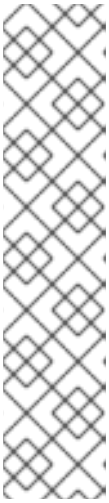
```
# rpm2cpio grub2-efi-version-architecture.rpm | cpio -dimv
```

6. ブートディレクトリーから、EFI ブートイメージをコピーします。**ARCH** を shim または grub に置き換え、その後にアーキテクチャーを追加します (**grubx64** など)。

```
# mkdir /var/lib/tftpboot/uefi
# cp my_local_directory/boot/efi/EFI/redhat/ARCH.efi /var/lib/tftpboot/uefi/
```

7. 以下の例のように、**grub.cfg** という名前の設定ファイルを **tftpboot**/ ディレクトリーに追加します。

```
set timeout=60
menuentry 'RHEL 8' {
  linuxefi images/RHEL-8.x/vmlinuz ip=dhcp inst.repo=http://10.32.5.1/RHEL-8.x/x86_64/iso-
contents-root/
  initrdefi images/RHEL-8.x/initrd.img
}
```



注記

- このランタイムイメージなしでは、インストールプログラムは起動できません。**inst.stage2** 起動オプションを使用して、イメージの場所を指定します。または、**inst.repo=** オプションを使用して、イメージおよびインストールソースを指定することも可能です。
- **inst.repo** で使用したインストールソースの場所には、有効な **treeinfo** ファイルが含まれている必要があります。
- インストールソースとして RHEL8 インストール DVD を選択すると、**.treeinfo** ファイルが BaseOS リポジトリーおよび AppStream リポジトリーを指定します。単一の **inst.repo** オプションを使用することで両方のリポジトリーを読み込むことができます。

8. **/var/lib/tftpboot/** ディレクトリーに、ブートイメージファイルを保存するサブディレクトリーを作成し、そのディレクトリーにブートイメージファイルをコピーします。この例のディレクトリーは、**/var/lib/tftpboot/images/RHEL-8.x/** になります。

```
# mkdir -p /var/lib/tftpboot/images/RHEL-8/
# cp /path_to_x86_64_images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftpboot/images/RHEL-8/
```

9. DHCP サーバーで **dhcpd** サービスを開始して有効にします。localhost で DHCP サーバーを設定している場合は、ローカルホストで **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

10. **tftp.socket** サービスを開始して有効にします。

```
# systemctl start tftp.socket
# systemctl enable tftp.socket
```

これにより、PXE 起動サーバーでは、PXE クライアントにサービスを提供する準備が整いました。クライアント (Red Hat Enterprise Linux のインストール先システム) を起動し、起動ソースを指定するように求められたら、**PXE ブート** を選択してネットワークインストールを開始できます。

関連情報

- Using the Shim Program

9.3.4. IBM Power システム用のネットワークサーバーの設定

この手順に従って、GRUB2 を使用して、IBM Power システム用のネットワーク起動サーバーを設定する方法を説明します。

重要

本セクションのすべての設定ファイルは例となります。設定の詳細は、アーキテクチャーや特定の要件によって異なります。

手順

1. root で、次のパッケージをインストールします。ネットワークに DHCP サーバーが設定されている場合は、dhcp-server パッケージを除外します。

```
# yum install tftp-server dhcp-server
```

2. ファイアウォールで、**tftp service** サービスへの着信接続を許可します。

```
# firewall-cmd --add-service=tftp
```

注記

- このコマンドは、次にサーバーを再起動するまで、一時的にアクセスを有効にします。永続的アクセスを有効にするには、コマンドに **--permanent** オプションを追加します。
- ISO インストールファイルの場所によっては、HTTP などのサービスの着信接続を許可しないといけない場合があります。

3. tftp のルートに、**GRUB2** ネットワーク起動ディレクトリーを作成します。

```
# grub2-mknetdir --net-directory=/var/lib/tftpboot
Netboot directory for powerpc-ieee1275 created. Configure your DHCP server to point to
/boot/grub2/powerpc-ieee1275/core.elf
```

注記

この手順で説明しているように、コマンドの出力は、DHCP 設定で設定する必要があるファイル名をユーザーに通知します。

a. PXE サーバーを x86 マシンで実行している場合は、tftp root に **GRUB2** ネットワーク起動ディレクトリーを作成する前に、**grub2-ppc64-modules** をインストールする必要があります。

```
# yum install grub2-ppc64-modules
```

4. 以下の例のように、**GRUB2** 設定ファイル (**/var/lib/tftpboot/boot/grub2/grub.cfg**) を作成します。

```
set default=0
set timeout=5

echo -e "\nWelcome to the Red Hat Enterprise Linux 8 installer!\n\n"

menuentry 'Red Hat Enterprise Linux 8' {
  linux grub2-ppc64/vmlinuz ro ip=dhcp inst.repo=http://10.32.5.1/RHEL-8/x86_64/iso-contents-root/
  initrd grub2-ppc64/initrd.img
}
```

注記

- このランタイムイメージなしでは、インストールプログラムは起動できません。**inst.stage2** 起動オプションを使用して、イメージの場所を指定します。または、**inst.repo=** オプションを使用して、イメージおよびインストールソースを指定することも可能です。
- **inst.repo** で使用したインストールソースの場所には、有効な **treeinfo** ファイルが含まれている必要があります。
- インストールソースとして RHEL8 インストール DVD を選択すると、**.treeinfo** ファイルが BaseOS リポジトリーおよび AppStream リポジトリーを指定します。単一の **inst.repo** オプションを使用することで両方のリポジトリーを読み込むことができます。

5. このコマンドを使用して DVD ISO イメージをマウントします。

```
# mount -t iso9660 /path_to_image/name_of_iso/ /mount_point -o loop,ro
```

6. ディレクトリーを作成し、DVD ISO イメージから **initrd.img** ファイルおよび **vmlinuz** ファイルをコピーします。以下に例を示します。

```
# cp /mount_point/ppc/ppc64/{initrd.img,vmlinuz} /var/lib/tftpboot/grub2-ppc64/
```

7. 以下の例のように、**GRUB2** に同梱されているブートイメージを使用するように DHCP サーバーを設定します。DHCP サーバーがすでに設定されている場合は、DHCP サーバーでこの手順を実行します。

```
subnet 192.168.0.1 netmask 255.255.255.0 {
  allow bootp;
  option routers 192.168.0.5;
  group { #BOOTP POWER clients
    filename "boot/grub2/powerpc-ieee1275/core.elf";
    host client1 {
    hardware ethernet 01:23:45:67:89:ab;
    fixed-address 192.168.0.112;
    }
  }
}
```

8. ネットワーク設定に合わせて、サンプルパラメーターの **subnet**、**netmask**、**routers**、**fixed-address**、および **hardware ethernet** を変更します。**file name** パラメーターは、この手順のステップで、**grub2-mknetdir** コマンドで出力したファイル名となります。

9. DHCP サーバーで **dhcpd** サービスを開始して有効にします。localhost で DHCP サーバーを設定している場合は、ローカルホストで **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

10. **tftp.socket** サービスを開始して有効にします。

```
# systemctl start tftp.socket
# systemctl enable tftp.socket
```

これにより、PXE 起動サーバーでは、PXE クライアントにサービスを提供する準備が整いました。クライアント (Red Hat Enterprise Linux のインストール先システム) を起動し、起動ソースを指定するように求められたら、**PXE ブート** を選択してネットワークインストールを開始できます。

9.4. 起動オプション

このセクションでは、インストールプログラムのデフォルトの動作を変更するために使用できる起動オプションについて説明します。すべての起動オプションは、アップストリームの Boot Option を参照してください。

9.4.1. 起動オプションの入力

起動オプションには、等号 (=) が付いているものと、付けていないものがあります。ブートオプションはブートコマンドラインに追加され、スペースで区切って複数のオプションを追加できます。インストールプログラムに固有の起動オプションは、常に **inst** から始まります。

等号 (=) 記号を使用するオプション

起動オプションに、**=** 記号を使用する値を指定する必要があります。たとえば、**inst.vncpassword=** オプションには値 (この場合はパスワード) を指定する必要があります。この例の正しい構文は **inst.vncpassword=password** です。

等号 (=) 記号を使用しないオプション

この起動オプションでは、値またはパラメーターを使用できません。たとえば、**rd.live.check** オプションでは、インストール開始前にインストールメディアの検証が強制されます。インストールプログラムは、このブートオプションが存在すると検証を実行します。ブートオプションが存在しないと、検証はスキップされます。

9.4.2. 起動オプションの編集

このセクションでは、起動メニューから起動オプションを編集するさまざまな方法を説明します。インストールメディアを起動すると、起動メニューが開きます。

9.4.2.1. BIOS で boot: プロンプトの編集

boot: プロンプトを使用すると、最初のオプションは、読み込むインストールプログラムのイメージファイルを常に指定する必要があります。ほとんどの場合、このイメージはキーワードを使用して指定できます。要件に応じて、追加オプションを指定できます。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューが開いたら、キーボードの **Esc** キーを押します。
2. **boot:** プロンプトにアクセスできるようになります。
3. キーボードの **Tab** キーを押して、ヘルプコマンドを表示します。
4. キーボードの **Enter** キーを押して、オプションでインストールを開始します。**boot:** プロンプトから起動メニュー画面に戻るには、システムを再起動して、インストールメディアから再度起動します。

注記

boot: プロンプトでは、**dracut** カーネルオプションも使用できます。利用可能なオプションの一覧は、**dracut.cmdline(7)** の man ページを参照してください。

9.4.2.2. > プロンプトを使用して事前定義されたブートオプションの編集

BIOS ベースの AMD64 および Intel64 システムでは、**>** プロンプトを使用して、事前定義されたブートオプションを編集できます。オプションの完全なセットを表示するには、ブートメニューから **Test this media and install RHEL 8** を選択します。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューでオプションを選択し、キーボードの **Tab** キーを押します。**>** プロンプトにアクセスし、利用可能なオプションを表示します。
2. **>** プロンプトに必要なオプションを追加します。
3. **Enter** を押してインストールを開始します。

4. **Esc** キーを押して編集をキャンセルし、ブートメニューに戻ります。

9.4.2.3. UEFI ベースのシステムの GRUB2 メニューの編集

GRUB2 メニューは、UEFI ベースの AMD64、Intel 64、および 64 ビット ARM システムで利用できます。

前提条件

- 起動可能なインストールメディア (USB、CD、または DVD) を作成している。
- メディアからインストールを起動し、起動メニュー画面が開いている。

手順

1. ブートメニューウィンドウから必要なオプションを選択し、**e** を押します。
2. UEFI システムでは、カーネルコマンドラインは **linuxefi** で始まります。カーソルを **linuxefi** カーネルコマンドラインの最後に移動します。
3. 必要に応じてパラメーターを編集します。たとえば、1 つ以上のネットワークインターフェイスを設定するには、**linuxefi** カーネルコマンドラインの最後に **ip=** パラメーターを追加し、その後に必要な値を追加します。
4. 編集が終了したら、**Ctrl** + **X** を押して、指定したオプションを使用してインストールを開始します。

9.4.3. インストールソースの起動オプション

このセクションでは、さまざまなインストールソースのブートオプションについて説明します。

inst.repo=

inst.repo= 起動オプションはインストールソースを指定します。つまり、パッケージリポジトリーと、そのリポジトリーを記述する有効な **.treeinfo** ファイルを提供する場所にあたります。たとえば、**inst.repo=cdrom** になります。**inst.repo=** オプションの対象は、以下のいずれかのインストールメディアになります。

- インストール可能なツリー (インストールプログラムのイメージ、パッケージ群、リポジトリーデータおよび有効な **.treeinfo** ファイルを含むディレクトリー設定)
- DVD (システムの DVD ドライブにある物理ディスク)
- Red Hat Enterprise Linux のフルインストール用 DVD の ISO イメージ (ハードドライブ、またはシステムにアクセスできるネットワーク上の場所)
  inst.repo= 起動オプションでは、さまざまなインストール方法を設定します。以下の表は、**inst.repo=** 起動オプションの詳細な構文を記載します。

表9.1 inst.repo= ブートオプションおよびインストールソースのタイプおよびフォーマット

ソースタイプ	起動オプションの形式	ソースの形式
CD/DVD ドライブ	**inst.repo=cdrom:<device>**	物理ディスクとしてのインストール DVD。[a]

ソースタイプ	起動オプションの形式	ソースの形式
マウント可能なデバイス (HDD および USB スティック)	**inst.repo=hd:<device>:/<path>**	インストール DVD のイメージファイル
NFS サーバー	**inst.repo=nfs:[options:]<server>:/<path>**	インストール DVD のイメージファイル、またはインストールツリー (インストール DVD にあるディレクトリーおよびファイルの完全なコピー)。[b]
HTTP サーバー	**inst.repo=http://<host>/<path>**	インストールツリー (インストール DVD 上にあるディレクトリーおよびファイルの完全なコピー)。
HTTPS サーバー	**inst.repo=https://<host>/<path>**	
FTP サーバー	**inst.repo=ftp://<username>:<password>@<host>/<path>**	
HMC	**inst.repo=hmc**	

[a] **device** が省略された場合、インストールプログラムはインストール DVD を含むドライブを自動的に検索します。

[b] NFS サーバーのオプションでは、デフォルトで NFS プロトコルのバージョン 3 が使用されます。別のバージョンを使用するには、**nfsvers=X** を **オプション** に追加し、**X** を、使用するバージョン番号に置き換えます。

ディスクデバイス名は、次の形式で設定します。

- カーネルデバイス名 (例: **/dev/sda1** または **sdb2**)
- ファイルシステムのラベル (例: **LABEL=Flash** または **LABEL=RHEL8**)
- ファイルシステムの UUID (例: **UUID=8176c7bf-04ff-403a-a832-9557f94e61db**)

英数字以外は **\xNN** で表す必要があります。**NN** は文字の 16 進数表示になります。たとえば、**\x20** なら空白 **(" ")** になります。

inst.addrepo=

inst.addrepo= 起動オプションを使用して、別のインストールソースとして、メインリポジトリー (**inst.repo=**) とともに追加のリポジトリーを追加します。起動時に、**inst.addrepo=** 起動オプションを複数回使用できます。以下の表では、**inst.addrepo=** 起動オプションの構文の詳細を記載します。

注記

REPO_NAME はリポジトリーの名前であり、インストールプロセスでは必須です。これらのリポジトリーは、インストールプロセス時にのみ使用され、インストールしたシステムにはインストールされません。

統一された ISO に関する詳細は、Unified ISO を参照してください。

表9.2 インストールソースおよびブートオプションの形式

インストールソース	起動オプションの形式	関連情報
URL にあるインストール可能なツリー	**inst.addrepo=REPO_NAME,[http,https,ftp]://<host>/<path>**	指定の URL にあるインストール可能なツリーを探します。
NFS パスにあるインストール可能なツリー	**inst.addrepo=REPO_NAME,nfs://<server>:/<path>**	指定した NFS パスのインストール可能なツリーを探します。コロンは、ホストの後に必要です。インストールプログラムは、RFC 2224 に従って URL の解析を行うのではなく、**nfs://** ディレクトリーの後のすべてを mount コマンドに渡します。
インストール環境でインストール可能なツリー	**inst.addrepo=REPO_NAME,file://<path>**	インストール環境の指定した場所にあるインストール可能なツリーを探します。このオプションを使用するには、インストールプログラムが利用可能なソフトウェアグループのロードを試行する前に、リポジトリーがマウントされる必要があります。このオプションの利点は、起動可能な ISO に複数のリポジトリーを利用でき、ISO からメインリポジトリーと追加のリポジトリーの両方をインストールできることです。追加のリポジトリーへのパスは **/run/install/source/REPO_ISO_PATH** です。また、キックスタートファイルの **%pre** セクションにリポジトリーディレクトリーをマウントできます。パスは、**inst.addrepo=REPO_NAME,file:///<path>** など、/ で始まる必要があります。

インストールソース	起動オプションの形式	関連情報
ハードドライブ	**inst.addrepo=REPO_NAME,hd:<device>:<path>**	指定した <device> パーティションをマウントして、<path> で指定した ISO からインストールします。<path> を指定しないと、インストールプログラムは <device> 上の有効なインストール ISO を探します。このインストール方法には、有効なインストール可能ツリーを持つ ISO が必要です。

inst.stage2=

inst.stage2= 起動オプションは、インストールプログラムのランタイムイメージの場所を指定します。このオプションは、有効な **.treeinfo** ファイルが含まれるディレクトリーへのパスを想定し、**.treeinfo** ファイルからランタイムイメージの場所を読み取ります。**.treeinfo** ファイルが利用できないと、インストールプログラムは、**images/install.img** からイメージを読み込もうとします。**inst.stage2** オプションを指定しない場合、インストールプログラムは **inst.repo** オプションで指定された場所を使用しようとします。

このオプションは、後でインストールプログラム内でインストールソースを手動で指定する場合に使用します。たとえば、インストールソースとしてコンテンツ配信ネットワーク (CDN) を選択する場合などに使用します。インストール DVD および Boot ISO には、それぞれの ISO からインストールプログラムを起動するための適切な **inst.stage2** オプションがすでに含まれています。

インストールソースを指定する場合は、代わりに **inst.repo=** オプションを使用します。

注記

デフォルトでは、インストールメデイアで **inst.stage2=** 起動オプションが使用され、これは特定のラベル (たとえば **inst.stage2=hd:LABEL=RHEL-x-0-0-BaseOS-x86_64**) に設定されています。ランタイムイメージが含まれるファイルシステムのデフォルトラベルを修正する場合、またはカスタマイズされた手順を使用してインストールシステムを起動する場合は、**inst.stage2=** 起動オプションに正しい値が設定されていることを確認してください。

inst.noverifyssl

inst.noverifyssl 起動オプションを使用して、追加のキックスタートリポジトリーを除き、すべての HTTPS 接続の SSL 証明書が検証されないようにします。ただし、**--noverifyssl** はリポジトリーごとに設定できます。
たとえば、リモートのインストールソースが自己署名 SSL 証明書を使用している場合には、**inst.noverifyssl** 起動オプションは、SSL 証明書を検証せずにインストーラーがインストールを完了できるようにします。

inst.stage2= を使用してソースを指定する場合の例

```
inst.stage2=https://hostname/path_to_install_image/ inst.noverifyssl
```

inst.repo= を使用してソースを指定する場合の例

-

```
inst.repo=https://hostname/path_to_install_repository/ inst.noverifyssl
```

inst.stage2.all

inst.stage2.all 起動オプションを使用して、複数の HTTP、HTTPS、または FTP ソースを指定します。**inst.stage2=** 起動オプションは、**inst.stage2.all** オプションとともに複数回使用して、成功するまで、イメージを順番にフェッチできます。以下に例を示します。

```
inst.stage2.all
inst.stage2=http://hostname1/path_to_install_tree/
inst.stage2=http://hostname2/path_to_install_tree/
inst.stage2=http://hostname3/path_to_install_tree/
```

inst.dd=

インストール時にドライバーの更新を実行する場合は、**inst.dd=** 起動オプションを使用します。インストール時にドライバーを更新する方法の詳細は、**高度な RHEL 8 インストールの実行** を参照してください。

inst.repo=hmc

このオプションにより、外部ネットワーク設定の必要がなくなるため、インストールのオプションが増えます。Binary DVD から起動すると、インストーラープログラムにより、追加のカーネルパラメーターを入力するように求められます。DVD をインストールソースとして設定するには、**inst.repo=hmc** オプションをカーネルパラメーターに追加します。インストールプログラムは、サポート要素 (SE) およびハードウェア管理コンソール (HMC) のファイルアクセスを有効にし、DVD から stage2 のイメージをフェッチし、ソフトウェア選択のために DVD のパッケージへのアクセスを提供します。

inst.proxy=

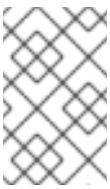
HTTP、HTTPS、および FTP プロトコルからインストールを実行する場合には、**inst.proxy=** 起動オプションが使用されます。以下に例を示します。

```
[PROTOCOL://][USERNAME[:PASSWORD]@]HOST[:PORT]
```

inst.nosave=

inst.nosave= 起動オプションを指定して、インストールログや関連ファイルがインストール済みのシステムに保存されないように制御します (例: **input_ks**、**output_ks**、**all_ks**、**logs**、**all**)。複数の値をコンマで区切って組み合わせることができます。以下に例を示します。

```
inst.nosave=Input_ks,logs
```



注記

inst.nosave 起動オプションは、インストール済みのシステムから、キックスタートのログや入力/出力などの Kickstart %post スクリプトで削除できないファイルの除外に使用されます。

input_ks

キックスタートによる入力を保存する機能を無効にします。

output_ks

インストールプログラムで生成されたキックスタートによる出力を保存する機能を無効にします。

all_ks

キックスタートによる入出力を保存する機能を無効にします。

logs

すべてのインストールログを保存する機能を無効にします。

all

すべてのキックスタート結果とすべてのログを保存する機能を無効にします。

inst.multilib

inst.multilib 起動オプションを使用して、DNF の **multilib_policy** を、**best** ではなく **all** に設定します。

inst.memcheck

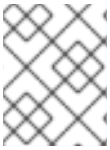
inst.memcheck 起動オプションは、インストールを完了するのにシステムに十分な RAM があることを確認するためのチェックを実行します。RAM が十分でない場合は、インストールプロセスが停止します。システムのチェックはおおよそのもので、インストールの際のメモリー使用率は、パッケージ選択やユーザーインターフェイス (グラフィカル、テキスト)、その他のパラメーターにより異なります。

inst.nomemcheck

inst.nomemcheck 起動オプションは、インストールを完了するのに十分な RAM があるかどうかの確認を実行しません。推奨よりも低いメモリー量でのインストールはサポートされていないため、インストールプロセスが失敗する場合があります。

9.4.4. ネットワーク起動オプション

シナリオでローカルイメージから起動するのではなく、ネットワーク経由でイメージから起動する必要がある場合は、次のオプションを使用してネットワーク起動をカスタマイズできます。



注記

dracut ツールを使用してネットワークを初期化します。**dracut** オプションの完全なリストについては、**dracut.cmdline(7)** の man ページを参照してください。

ip=

ip= 起動オプションは、1 つ以上のネットワークインターフェイスを設定します。複数のインターフェイスを設定するには、次のいずれかの方法を使用します。

- インターフェイスごとに 1 回ずつ、**ip** オプションを複数回使用します。これを行うには、**rd.neednet=1** オプションを使用し、**bootdev** オプションを使用してプライマリーブートインターフェイスを指定します。
- **ip** オプションを 1 回使用してから、Kickstart を使用してさらにインターフェイスを設定します。このオプションでは、複数の形式が使用できます。以下の表は、最も一般的なオプションの情報が含まれます。

以下の表では、下記の点を前提としています。

- **ip** パラメーターはクライアントの IP アドレスを指定し、**IPv6** には角括弧が必要です (例: 192.0.2.1 または [2001:db8::99])。
- **gateway** パラメーターはデフォルトのゲートウェイになります。**IPv6** には角括弧必要です。
- **netmask** パラメーターは使用するネットマスクです。完全ネットマスク (255.255.255.0 など) または接頭辞 (64 など) を使用できます。

- **hostname** パラメーターはクライアントシステムのホスト名です。このパラメーターは任意です。

表9.3 ネットワークインターフェイスを設定するためのブートオプション形式

起動オプションの形式	設定方法
ip=method	全インターフェイスの自動設定
ip=interface:method	特定インターフェイスの自動設定
ip=ip::gateway:netmask:hostname:interface:none	静的設定 (例: IPv4 **ip=192.0.2.1::192.0.2.254:255.255.255.0:server.example.com:enp1s0:none**) IPv6: **ip=[2001:db8::1]::[2001:db8::fffe]:64:server.example.com:enp1s0:none**
ip=ip::gateway:netmask:hostname:interface:method:mtu	オーバーライドを使用した特定インターフェイスの自動設定

自動インターフェイスの設定方法
オーバーライドを使用した特定インターフェイスの自動設定 では、**dhcp** など、指定した自動設定方法を使用してインターフェイスを起動しますが、自動取得した IP アドレス、ゲートウェイ、ネットマスク、ホスト名、他のパラメーターなどで指定したものは無効にします。パラメーターはすべて任意となるため、無効にするパラメーターだけを指定します。

method パラメーターには、以下のいずれかを使用します。

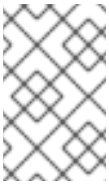
DHCP
dhcp
IPv6 DHCP
dhcp6
IPv6 自動設定
auto6
iBFT (iSCSI Boot Firmware Table)
ibft

注記

- **ip** オプションを指定せずに、**inst.ks=http://host/path** などのネットワークアクセスを必要とするブートオプションを使用する場合、**ip** オプションのデフォルト値は **ip=dhcp** です。
- iSCSI ターゲットに自動的に接続するには、**ip=ibft** ブートオプションを使用して、ターゲットにアクセスするネットワークデバイスをアクティブ化します。

nameserver=

nameserver= オプションは、ネームサーバーのアドレスを指定します。このオプションは複数回使用できます。



注記

ip= パラメーターには角括弧が必要です。ただし、IPv6 アドレスには角括弧が使用できません。IPv6 アドレスに使用する正しい構文は **nameserver=2001:db8::1** のようになります。

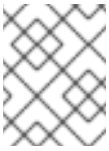
bootdev=

bootdev= オプションは、起動インターフェイスを指定します。このオプションは、**ip** オプションを複数回使用する場合に必要になります。

ifname=

ifname= オプションは、特定の MAC アドレスを持つネットワークデバイスにインターフェイス名を割り当てます。このオプションは複数回使用できます。構文は、**ifname=interface:MAC** です。以下に例を示します。

```
ifname=eth0:01:23:45:67:89:ab
```



注記

ifname= オプションは、インストール中にカスタムのネットワークインターフェイス名を設定する際にサポートされる唯一の方法となります。

inst.dhcpclass=

inst.dhcpclass= オプションは、DHCP のベンダークラス識別子を指定します。**dhcpd** サービスではこの値を **vendor-class-identifier** として認識します。デフォルト値は **anaconda-$(uname -srm)** です。

inst.waitfornet=

inst.waitfornet=SECONDS 起動オプションを使用すると、インストールシステムは、ネットワーク接続を待ってからインストールします。**SECONDS** 引数で指定する値は、ネットワーク接続がない場合でもすぐにはタイムアウトにせず、ネットワーク接続を待ち続け、インストールプロセスを継続する最大秒数を表します。

vlan=

vlan= オプションを使用して、仮想 LAN (VLAN) デバイスに特定の名前を付け、指定インターフェイスにそのデバイスを設定します。構文は **vlan=name:interface** です。以下に例を示します。

```
vlan=vlan5:enp0s1
```

これにより、**enp0s1** インターフェイスに **vlan5** という名前の VLAN デバイス が設定されます。name は以下のような形式をとります。

- VLAN_PLUS_VID: **vlan0005**
- VLAN_PLUS_VID_NO_PAD: **vlan5**
- DEV_PLUS_VID: **enp0s1.0005**
- DEV_PLUS_VID_NO_PAD: **enp0s1.5**

bond=

bond= オプションを使用して、**bond=name[:interfaces][:options]** 構文でボンディングデバイスを設定します。name はボンディングデバイス名に置き換え、interfaces は物理 (イーサネット) インターフェイスのコンマ区切りリストに置き換え、options はボンディングオプションのコンマ区切りリストに置き換えます。以下に例を示します。

```
bond=bond0:enp0s1,enp0s2:mode=active-backup,tx_queues=32,downdelay=5000
```

利用可能なオプションのリストは、ボンディングコマンド **modinfo** を実行します。

team=

team= オプションを使用して、**team=name:interfaces** 構文でチームデバイスを設定します。チームデバイスの基礎となるインターフェイスとして使用されるように、name はチームデバイスの望ましい名前に、interfaces は物理 (イーサネット) デバイスのコンマ区切りリストに置き換えます。以下に例を示します。

```
team=team0:enp0s1,enp0s2
```

bridge=

bridge= オプションを使用して、**bridge=name:interfaces** 構文でブリッジデバイスを設定します。ブリッジデバイスの基礎となるインターフェイスとして使用されるように、name はブリッジデバイスの望ましい名前に、interfaces は物理 (イーサネット) デバイスのコンマ区切りリストに置き換えます。以下に例を示します。

```
bridge=bridge0:enp0s1,enp0s2
```

関連情報

- **ネットワークの設定および管理**

9.4.5. コンソール起動オプション

このセクションでは、コンソール、モニターディスプレイ、およびキーボードの起動オプションを設定する方法を説明します。

console=

console= オプションを使用して、プライマリーコンソールとして使用するデバイスを指定します。たとえば、最初のシリアルポートでコンソールを使用するには、**console=ttyS0** を使用します。**console=** 引数を使用する場合、インストールはテキスト UI から始まります。 **console=** オプションを複数回使用する必要がある場合は、指定したすべてのコンソールにブートメッセージが表示されます。ただし、インストールプログラムは、最後に指定されたコンソールのみを使用します。たとえば、**console=ttyS0 console=ttyS1** と指定すると、インストールプログラムでは **ttyS1** が使用されます。

inst.lang=

inst.lang= オプションを使用して、インストール時に使用する言語を設定します。ロケールのリストを表示するには、コマンド **locale -a | grep _** または **localectl list-locales | grep _** コマンドを実行します。

inst.singlelang

inst.singlelang を指定して単一の言語モードでインストールを行うと、そのインストール言語と言語サポート設定に対する対話オプションを利用できません。**inst.lang** 起動オプションまたは **lang**

キックスタートコマンドを使用して言語を指定すると、オプションが指定されます。言語を指定しないと、インストールプログラムのロケールはデフォルトで **en_US.UTF-8** となります。

inst.geoloc=

インストールプログラムで、地理位置情報の使用方法を設定するには、**inst.geoloc=** オプションを使用します。地理位置情報は、言語およびタイムゾーンの事前設定に使用され、**inst.geoloc=value** 構文を使用します。**value** には、以下のいずれかのパラメーターを使用します。

- 地理位置情報の無効化: **inst.geoloc=0**
- Fedora GeoIP API (**inst.geoloc=provider_fedora_geoip**) を使用します。
- Hostip.info GeoIP API (**inst.geoloc=provider_hostip**) を使用します。

inst.geoloc= オプションを指定しない場合、デフォルトのオプションは **provider_fedora_geoip** です。

inst.keymap=

inst.keymap= オプションを使用して、インストールに使用するキーボードレイアウトを指定します。

inst.cmdline

inst.cmdline オプションを使用して、インストールプログラムをコマンドラインモードで強制的に実行します。このモードでは対話が使用できないため、キックスタートファイルまたはコマンドラインですべてのオプションを指定する必要があります。

inst.graphical

インストールプログラムをグラフィカルモードで強制的に実行するには、**inst.graphical** オプションを使用します。グラフィカルモードがデフォルトです。

inst.text

inst.text オプションを使用して、グラフィカルモードではなく、テキストモードでインストールプログラムを強制的に実行します。

inst.noninteractive

inst.noninteractive 起動オプションを使用して、非対話モードでインストールプログラムを実行します。非対話型モード (および **inst.noninteractive**) では、ユーザーとの対話は許可されていません。グラフィカルまたはテキストインストールで **inst.nointeractive** オプションを使用できます。**inst.noninteractive** オプションをテキストモードで使用すると、**inst.cmdline** オプションと同じように動作します。

inst.resolution=

inst.resolution= オプションを使用して、グラフィカルモードで、画面の解像度を指定します。形式は **NxM** です。**N** は画面の幅で、**M** は画面の高さ (ピクセル単位) です。サポートされる最小解像度は 1024x768 です。

inst.vnc

inst.vnc オプションを使用して、Virtual Network Computing (VNC) を使用したグラフィカルインストールを実行します。インストールプログラムと対話するには VNC クライアントアプリケーションを使用する必要があります。VNC 共有を有効にすると、複数のクライアントに接続できます。VNC を使用してインストールしたシステムは、テキストモードで起動します。

inst.vncpassword=

inst.vncpassword= オプションを使用して、インストールプログラムが使用する VNC サーバーにパスワードを設定します。

inst.vncconnect=

inst.vncconnect= オプションを使用して、指定されたホストの場所にあるリスニング VNC クライアントに接続します (例: **inst.vncconnect=<host>[:<port>]**)。デフォルトのポートは 5900 です。このオプションを使用するには、コマンド **vncviewer -listen** を入力します。

inst.xdriver=

inst.xdriver= オプションを使用して、インストール時およびインストール済みシステムで使用される X ドライバーの名前を指定します。

inst.usefbx

inst.usefbx オプションを使用して、ハードウェア固有のドライバーではなく、フレームバッファー X ドライバーを使用するようにインストールプログラムに要求します。このオプションは、**inst.xdriver=fbdev** オプションと同等です。

modprobe.blacklist=

modprobe.blacklist= オプションを使用して、1 つ以上のドライバーを拒否リストに追加するか、完全に無効にします。このオプションを使用して無効にしたドライバー (mods) は、インストールの開始時にロードできません。インストールが完了すると、インストールされたシステムはこれらの設定を保持します。拒否リストに指定したドライバーのリストは、**/etc/modprobe.d/** ディレクトリーにあります。複数のドライバーを無効にするには、コンマ区切りリストを使用します。以下に例を示します。

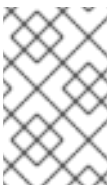
```
modprobe.blacklist=ahci,firewire_ohci
```

inst.xtimeout=

inst.xtimeout= オプションを使用して、X サーバーの起動のタイムアウトを秒単位で指定します。

inst.sshd

インストール時に、SSH を使用してシステムに接続し、インストールの進捗を監視できるように、**inst.sshd** オプションを使用して、**sshd** サービスを開始します。SSH の詳細は、man ページの **ssh(1)** を参照してください。デフォルトでは、**sshd** オプションは、64 ビットの IBM Z アーキテクチャーでのみ自動的に起動します。その他のアーキテクチャーでは、**sshd** は、**inst.sshd** オプションを使用しない限り起動しません。



注記

インストール中に、root アカウントにはデフォルトでパスワードが設定されていません。キックスタートコマンド **sshpw** を使用して、インストール時に root パスワードを設定できます。

inst.kdump_addon=

インストールプログラムで Kdump 設定画面 (アドオン) を有効または無効にするには、**inst.kdump_addon=** オプションを使用します。この画面はデフォルトで有効になっているため、無効にする場合は **inst.kdump_addon=off** を使用します。アドオンを無効にすると、グラフィカルおよびテキストベースのインターフェイスと、キックスタートコマンド **%addon com_redhat_kdump** の両方で Kdump 画面が無効になります。

9.4.6. 起動オプションのデバッグ

このセクションでは、問題をデバッグするときに使用できるオプションを説明します。

inst.rescue

inst.rescue オプションを使用して、システムの診断と修正のためのレスキュー環境を実行します。たとえば、レスキューモードでファイルシステムを修復 できます。

inst.updates=

inst.updates= オプションを使用して、インストール時に適用する **updates.img** ファイルの場所を指定します。**updated.img** ファイルは、いくつかのソースの1つから取得できます。

表9.4 updates.img ファイルソース

ソース	説明	例
ネットワークからの更新	**updates.img** のネットワーク上の場所を指定します。インストールツリーを変更する必要はありません。この方法を使用するには、カーネルコマンドラインを編集して **inst.updates** を追加します。	**inst.updates=http://website.com/path/to/updates.img**.
ディスクイメージからの更新	フロッピードライブまたは USB キーに **updates.img** を保存できます。これは、ファイルシステムタイプが **ext2** の **updates.img** でのみ可能です。イメージの内容をフロッピードライブに保存するには、フロッピーディスクを挿入し、次のコマンドを実行します。	**dd if=updates.img of=/dev/fd0 bs=72k count=20**USB キーまたはフラッシュメディアを使用するには、**/dev/fd0** を、USB キーのデバイス名に置き換えます。
インストールツリーからの更新	CD、ハードドライブ、HTTP、または FTP のインストールを使用する場合は、すべてのインストールツリーが **.img** ファイルを検出できるように、インストールツリーに **updates.img** を保存できます。このファイル名は、**updates.img** にする必要があります。	NFS インストールの場合は、ファイルを **images/** ディレクトリーまたは **RHupdates/** ディレクトリーに保存します。

inst.loglevel=

inst.loglevel= オプションを使用して、端末に記録するログメッセージの最小レベルを指定します。このオプションは、ターミナルログにのみ適用されます。ログファイルには、常にすべてのレベルのメッセージが含まれます。このオプションで可能な値は、最低レベルから最高レベルまで次のとおりです。

- **debug**
- **info**
- **warning**
- **error**
- **critical**

デフォルト値は **info** となるため、デフォルトでは、**info** から **critical** までのメッセージがログの端末に表示されます。

inst.syslog=

インストールの開始時に、指定されたホスト上の **syslog** プロセスにログメッセージを送信します。**inst.syslog=** は、リモート **syslog** プロセスが着信接続を受け入れるように設定されている場合にのみ使用できます。

inst.virtiolog=

inst.virtiolog = オプションを使用して、ログの転送に使用する virtio ポート (**/dev/virtio-ports/name** にある文字デバイス) を指定します。デフォルト値は、**org.fedoraproject.anaconda.log.0** です。

inst.zram=

インストール中の zRAM スワップの使用を制御します。このオプションは、圧縮したブロックデバイスをシステム RAM に作成し、ハードドライブではなくスワップ領域に使用します。この設定により、使用可能なメモリーが少ない状態でインストールプログラムを実行し、インストール速度を向上させることができます。次の値を使用して、**inst.zram=** オプションを設定できます。

- inst.zram=1 は、システムメモリーサイズに関係なく、zRAM スワップを有効にします。デフォルトでは、2GiB 以下の RAM を搭載したシステムで zRAM のスワップが有効になっています。
- inst.zram=0 は、システムメモリーサイズに関係なく、zRAM スワップを無効にします。デフォルトでは、2GiB を超えるメモリーを搭載したシステムでは zRAM のスワップが無効になっています。

rd.live.ram

images/install.img の **stage 2** イメージを RAM にコピーします。これにより、インストールに必要なメモリーがイメージのサイズ (通常は 400 ~ 800MB) だけ増加することに注意してください。

inst.nokill

致命的なエラーが発生したとき、またはインストールプロセスの最後に、インストールプログラムが再起動しないようにします。再起動時に失われるインストールログをキャプチャーするのに使用します。

inst.noshell

インストール中にターミナルセッション 2 (tty2) でシェルを防止します。

inst.notmux

インストール中に tmux を使用しないようにします。この出力は、ターミナル制御文字なしで生成され、非対話用になります。

inst.remotelog=

TCP 接続を使用してすべてのログをリモート **host:port** に送信します。リスナーがなく、インストールが通常通りに進まない場合は、接続が中断されます。

9.4.7. ストレージ起動オプション

このセクションでは、ストレージデバイスからの起動をカスタマイズするために指定できるオプションを説明します。

inst.nodmraid

dmraid サポートを無効にします。

警告

使用する場合は注意が必要です。ファームウェア RAID アレイの一部として誤って特定されたディスクがある場合は、古い RAID メタデータが存在する可能性があります。これらは、**dmraid** や **wipefs** などの適切なツールを使用して削除する必要があります。

inst.nompath

マルチパスデバイスのサポートを無効にします。このオプションは、システムに誤検知があり、通常のブロックデバイスをマルチパスデバイスとして誤って識別する場合にのみ使用してください。

警告

使用する場合は注意が必要です。マルチパスハードウェアではこのオプションを使用しないでください。このオプションを使用してマルチパスデバイスのシングルパスにインストールすることはサポートされていません。

inst.gpt

インストールプログラムがパーティション情報を Master Boot Record (MBR) ではなく GUID Partition Table (GPT) にインストールするように強制します。このオプションは、BIOS 互換モードである場合を除き、UEFI ベースのシステムでは有効ではありません。通常、BIOS 互換モードの BIOS ベースのシステムおよび UEFI ベースのシステムは、ディスクのサイズが 2^32 セクター以上でない限り、パーティション情報の格納に MBR スキーマを使用しようとします。ディスクセクターは通常 512 バイトで、通常これは 2 TiB に相当します。**inst.gpt** ブートオプションを使用すると、GPT をより小さなディスクに書き込むことができます。

9.4.8. キックスタート起動オプション

このセクションでは、インストールを自動化するのにキックスタートファイルに追加できるブートオプションを説明します。

inst.ks=

インストールの自動化に使用するキックスタートファイルの場所を定義します。その後、いずれかの **inst.repo** 形式を使用して、場所を指定できます。パスを指定せずにデバイスを指定すると、インストールプログラムは、指定したデバイスの **/ks.cfg** でキックスタートファイルを検索します。

デバイスを指定せずにこのオプションを使用する場合、インストールプログラムはオプションに次の値を使用します。

```
inst.ks=nfs:next-server:/filename
```

ここでは、**next-server** は DHCP の next-server オプション、または DHCP サーバーの IP アドレスで、**filename** は DHCP の filename オプションまたは **/kickstart/** です。指定のファイル名が / 文字で終了すると 、**ip-kickstart** が追加されます。次の表に例を示します。

表9.5 デフォルトのキックスタートファイルの場所

DHCP サーバーのアドレス	クライアントのアドレス	キックスタートファイルの場所
192.168.122.1	192.168.122.100	192.168.122.1:/kickstart/192.168.122.100-kickstart

OEMDRV のラベルが付いたボリュームが存在すると、インストールプログラムは、キックスタートファイル **ks.cfg** を読み込もうとします。キックスタートファイルがこの場所にある場合は、**inst.ks=** 起動オプションを使用する必要がありません。

inst.ks.all

複数の **inst.ks** オプションによる複数のキックスタートファイルの場所を順次試行するように **inst.ks.all** オプションを指定します。最初に成功した場所が使用されます。これは、**http**、**https**、または **ftp** タイプの場所のみ適用され、その他の場所は無視されます。

inst.ks.sendmac

inst.ks.sendmac オプションを使用して、すべてのネットワークインターフェイスの MAC アドレスを含む HTTP 送信リクエストにヘッダーを追加します。以下に例を示します。

```
X-RHN-Provisioning-MAC-0: eth0 01:23:45:67:89:ab
```

これは、**inst.ks=http** を使用してシステムをプロビジョニングする場合に便利です。

inst.ks.sendsn

inst.ks.sendsn オプションを使用して、HTTP 送信リクエストにヘッダーを追加します。このヘッダーには、**/sys/class/dmi/id/product_serial** から読み込まれたシステムのシリアル番号が含まれます。ヘッダーの構文は以下のとおりです。

```
X-System-Serial-Number: R8VA23D
```

関連情報

- Full list of boot options

9.4.9. 高度なインストール起動オプション

本セクションでは、高度なインストール起動オプションを説明します。

inst.kexec

再起動を実行する代わりに、インストールの最後に **kexec** システムコールを実行します。**inst.kexec** オプションは、新しいシステムを即座に読み込み、BIOS またはファームウェアが通常実行するハードウェアの初期化を回避します。

重要

このオプションは非推奨になっており、テクノロジープレビューとしてのみ利用できます。テクノロジープレビュー機能に対する Red Hat のサポート範囲の詳細は、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。

kexec を使用すると、通常はシステムの完全な再起動時にクリアされるデバイスレジスタがデータでいっぱいになる可能性があります。これにより、特定のデバイスドライバーに問題が発生する可能性があります。

inst.multilib

multilib パッケージ用にシステムを設定して、64 ビット AMD64 または Intel 64 システムに 32 ビットパッケージをインストールできるようにします。通常、AMD64 または Intel 64 システムでは、このアーキテクチャー専用となるパッケージ (x86_64 の印が付いている) と、全アーキテクチャー用のパッケージ (noarch の印が付いている) がインストールされます。**inst.multilib** 起動オプションを使用すると、32 ビットの AMD または Intel システム用のパッケージ (i686 の印が付いている) が自動的にインストールされます。
これは、**%packages** セクションで直接指定されているパッケージにのみ適用されます。パッケージが依存関係としてインストールされている場合は、正確に指定した依存関係のみがインストールされます。たとえば、**glibc** パッケージに依存する **bash** パッケージをインストールする場合、**bash** パッケージは複数のバリアントでインストールされますが、**glibc** パッケージは bash パッケージが必要とするバリアントにのみインストールされます。

selinux=0

インストールプログラムおよびインストールされたシステムでの SELinux の使用を無効にします。デフォルトでは、SELinux はインストールプログラムでは permissive モードで動作し、インストールされたシステムでは enforcing モードで動作します。

注記

inst.selinux=0 と selinux=0 オプションは同じではありません: * inst.selinux=0: インストールプログラムでのみ SELinux を無効にします。* selinux=0: インストールプログラムとインストールされたシステムで SELinux の使用を無効にします。SELinux を無効にすると、イベントがログに記録されなくなります。

inst.nonibftiscsiboot

iSCSI ブートファームウェアテーブル (iBFT) で設定されていない iSCSI デバイスにブートローダーを配置します。

9.4.10. 廃止予定の起動オプション

このセクションは、非推奨の起動オプションを説明します。これらのオプションはインストールプログラムでも使用できますが、非推奨とされています。また、Red Hat Enterprise Linux の今後のリリースで削除される予定です。

method

method オプションは、**inst.repo** のエイリアスです。

dns

dns の代わりに **nameserver** を使用します。ネームサーバーはコンマ区切りのリストを受け付けず、代わりに複数のネームサーバーオプションを使用することに注意してください。

netmask、gateway、hostname

netmask、**gateway**、および **hostname** オプションは、**ip** オプションの一部として利用できます。

ip=bootif

PXE 指定の **BOOTIF** オプションが自動的に使用されるため、**ip=bootif** を使用する必要はありません。

ksdevice

表9.6 ksdevice 起動オプションの値

値	情報
存在しない	該当なし
ksdevice=link	このオプションがデフォルトの動作と同じ場合に無視されます。
ksdevice=bootif	**BOOTIF=** が存在する場合は、このオプションはデフォルトであるため無視されます。
ksdevice=ibft	**ip=ibft** に変更詳細は **ip** を参照してください。
ksdevice=<MAC>	**BOOTIF=${MAC/:/-}** に変更
ksdevice=<DEV>	**bootdev** に置き換え

9.4.11. 削除済みの起動オプション

このセクションでは、Red Hat Enterprise Linux から削除された起動オプションを説明します。

注記

dracut では、高度な起動オプションを利用できます。**dracut** の詳細は、man ページの **dracut.cmdline(7)** を参照してください。

askmethod、asknetwork

initramfs は完全に非対話的に実行されるため、**askmethod** と **asknetwork** のオプションは削除されました。**inst.repo** を使用して、適切なネットワークオプションを指定します。

blacklist、nofirewire

modprobe オプションは、カーネルモジュールのブロックリストを処理するようになりました。**modprobe.blacklist=<mod1>,<mod2>** を使用します。**modprobe.blacklist=firewire_ohci** を使用して、FireWire モジュールを拒否リストに入れることができます。

inst.headless=

headless= オプションでは、インストールしているシステムにディスプレイハードウェアがなく、インストールプログラムがディスプレイハードウェアを検索する必要がないことを指定しています。

inst.decorated

inst.decorated オプションは、装飾画面でのグラフィカルインストールの指定に指定されていまいた。デフォルトでは、この画面は装飾されないため、タイトルバーやサイズ変更などの機能はありません。このオプションは不要になりました。

repo=nfsiso

inst.repo=nfs: オプションを使用します。

serial

console=ttyS0 オプションを指定します。

updates

inst.updates オプションを指定します。

essid、wepkey、wpakey

dracut はワイヤレスネットワークをサポートしません。

ethtool

このオプションは不要になりました。

gdb

dracut ベースの **initramfs** のデバッグには多くのオプションが使用できるため、このオプションは削除されました。

inst.mediacheck

dracut オプションの rd.live.check オプション指定してください。

ks=floppy

inst.ks=hd:<device> オプションを指定します。

display

UI のリモートディスプレイには、**inst.vnc** オプションを指定します。

utf8

このオプションは、デフォルトの TERM 設定が期待通りに動作するため、不要になりました。

noipv6

IPv6 はカーネルに組み込まれたため、インストールプログラムによる削除はできません。**ipv6.disable=1** を使用して ipv6 を無効にすることができます。この設定は、インストール済みシステムによって使用されます。

upgradeany

インストールプログラムがアップグレードを処理しなくなるため、このオプションは不要になりました。

第10章 キックスタートの参照

付録I キックスタートスクリプトのファイル形式の参照

この参照は、キックスタートファイルの形式を詳細に説明します。

I.1. キックスタートファイルの形式

キックスタートスクリプトは、インストールプログラムが認識するキーワードが含まれ、インストールの指示を提供するプレーンテキストのファイルです。ファイルを ASCII テキストとして保存できるテキストエディター (例: Linux システムの **Gedit** または **vim**、Windows システムの **メモ帳**) は、キックスタートファイルの作成や編集に使用できます。キックスタート設定ファイルには好きな名前を付けることができますが、後で他の設定ファイルやダイアログでこの名前を指定する必要があるため、シンプルな名前にしておくことが推奨されます。

コマンド

コマンドは、インストールの命令として役に立つキーワードです。各コマンドは 1 行で記載する必要があります。コマンドにはオプションを指定できます。コマンドとオプションの指定方法は、シェルで Linux コマンドを使用するのと似ています。

セクション

パーセント **%** 文字で始まる特殊コマンドは、セクションを開始します。セクションのコマンドの解釈は、セクションの外に置かれたコマンドとは異なります。すべてのセクションは、**%end** コマンドで終了する必要があります。

セクションタイプ

利用可能なセクションは以下のとおりです。

- **アドオンセクション**。これらのセクションは、**%addon addon_name** コマンドを使用します。
- **パッケージの選択セクション**。**%packages** から始まります。これを使用してインストールするパッケージを指定します。これには、パッケージグループやモジュールなど、間接的な指定も含まれます。
- **スクリプトセクション**。これは、**%pre**、**%pre-install**、**%post**、および **%onerror** で開始します。これらのセクションは必須ではありません。

コマンドセクション

コマンドセクションは、スクリプトセクションや **%packages** セクション以外の、キックスタートファイルのコマンドに使用される用語です。

スクリプトセクション数および順序付け

コマンドセクションを除くすべてのセクションはオプションであり、複数回表示できます。特定タイプのスクリプトセクションが評価される際に、キックスタートにあるそのタイプのセクションがすべて、表示順に評価されます。たとえば、**%post** が 2 つある場合は、表示されている順に評価されます。ただし、さまざまなタイプのスクリプトセクションを任意の順序で指定する必要はありません。**%pre** セクションの前に、**%post** セクションがあるかどうかは問題ありません。

コメント

キックスタートコマンドは、ハッシュ文字 **#** 始まる行です。このような行は、インストールプログラムには無視されます。

必須項目以外は省略しても構いません。必須項目を省略すると、インストールプログラムがインタラクティブモードに変更され、通常の対話型インストールと同じように、ユーザーが関連する項目に回答できるようになります。キックスタートスクリプトは、**cmdline** コマンドで非対話的に宣言することもできます。非対話モードでは、回答していない項目があるとインストールプロセスが中断します。

注記

テキストまたはグラフィカルモードのキックスタートインストール時にユーザーの対話が必要な場合は、インストールを完了するために更新が必須であるウィンドウのみに入力してください。スポークを入力すると、キックスタートの設定がリセットされる可能性があります。設定のリセットは、インストール先ウィンドウの入力後に、ストレージに関連するキックスタートコマンドに特化して適用されます。

I.2. キックスタートでのパッケージ選択

キックスタートは、インストールするパッケージを選択するために、**%packages** コマンドで始まるセクションを使用します。この方法で、パッケージ、グループ、環境、モジュールストリーム、およびモジュールプロファイルをインストールできます。

I.2.1. パッケージの選択セクション

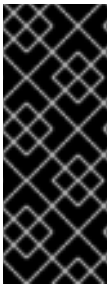
%packages コマンドを使用して、インストールするソフトウェアパッケージを説明するキックスタートセクションを開始します。**%packages** セクションは、**%end** コマンドで終了する必要があります。

パッケージは、環境、グループ、モジュールストリーム、モジュールプロファイル、またはパッケージ名で指定できます。関連パッケージを含むいくつかの環境およびグループが定義されます。環境およびグループのリストは、Red Hat Enterprise Linux 8 インストール DVD の **repository/repodata/*-comps-repository.architecture.xml** ファイルを参照してください。

***-comps-repository.architecture.xml** ファイルには、利用可能な環境 (**<environment>** タグでマーク) およびグループ (**<group>** タグ) を記述した構造が含まれています。各エントリーには、ID、ユーザー可視性の値、名前、説明、パッケージリストがあります。グループがインストールに選択されていると、パッケージリストで **mandatory** とマークされたパッケージが常にインストールされ、**default** とマークされたパッケージは、他で個別に除外されていない場合に限りインストールされます。また、**optional** とマークされたパッケージは、グループが選択されている場合でも、他で明確に含める必要があります。

パッケージグループや環境は、その ID (**<id>** タグ) もしくは名前 (**<name>** タグ) を使用して指定できます。

どのパッケージをインストールするべきかわからない場合は、Minimal Install 環境を選択することが推奨されます。**最小インストール** では、Red Hat Enterprise Linux 8 の実行に必須のパッケージのみが提供されます。これにより、システムが脆弱性の影響を受ける可能性が大幅に減ります。必要な場合は、インストール後に追加パッケージをインストールできます。**最小インストール** の詳細は、**セキュリティーの強化** の 必要なパッケージの最小限のインストール のセクションを参照してください。**初期セットアップ** は、デスクトップ環境と X Window System がインストールに含まれ、グラフィカルログインが有効になっていないと、キックスタートファイルからシステムをインストールしてから実行することができません。



重要

64 ビットシステムに 32 ビットパッケージをインストールするには、次を行います。

- **%packages** セクションに **--multilib** オプションを指定します。
- **glibc.i686** のように、そのパッケージの構築対象である 32 ビットアーキテクチャーをパッケージ名に追記します。

I.2.2. パッケージの選択コマンド

このコマンドは、キックスタートファイルの **%packages** セクションで使用できます。

環境の指定

@^ 記号で開始する行で、インストールする環境全体を指します。

```
%packages
@^Infrastructure Server
%end
```

これは、**インフラストラクチャーサーバー** 環境の一部となるパッケージをすべてインストールします。利用可能なすべての環境は、Red Hat Enterprise Linux 8 インストール DVD の **repository/repodata/*-comps-repository.architecture.xml** ファイルで説明されています。

キックスタートファイルに指定する必要があるのは、1 つの環境だけです。追加の環境を指定すると、最後に指定した環境のみが使用されます。

グループの指定

1 行に 1 エントリーずつグループを指定します。***-comps-repository.architecture.xml** ファイルに指定したとおりに、@ 記号に続いてグループのフルネームまたはグループ ID を指定します。以下に例を示します。

```
%packages
@X Window System
@Desktop
@Sound and Video
%end
```

Core グループは常に選択されるため、**%packages** セクションで指定する必要はありません。

個別パッケージの指定

1 行に 1 エントリーで、名前で個別のパッケージを指定します。アスタリスク記号 (*) をパッケージ名のワイルドカードとして使用できます。以下に例を示します。

```
%packages
sqlite
curl
aspell
docbook*
%end
```

docbook* エントリーには、ワイルドカードを使用したパターンに適合する **docbook-dtds** パッケージおよび **docbook-style** パッケージが含まれます。

モジュールストリームのプロファイルの指定

プロファイルの構文を使用して、モジュールストリープのポリシーを、1 行ごとに指定します。

```
%packages
@module:stream/profile
%end
```

これにより、モジュールストリームで指定したプロファイルに記載されているパッケージがすべてインストールされます。

- モジュールにデフォルトのストリームが指定されている場合は、削除できます。デフォルトのストリームが指定されていない場合は、指定する必要があります。
- モジュールストリームにデフォルトのプロファイルが指定されている場合は、削除できます。デフォルトのプロファイルが指定されていない場合は、指定する必要があります。
- 異なるストリームでモジュールを複数回インストールすることはできません。
- 同じモジュールおよびストリームの複数プロファイルをインストールできます。

モジュールおよびグループは、@ 記号で始まる同じ構文を使用します。同じ名前のモジュールとパッケージグループが存在する場合は、モジュールが優先されます。

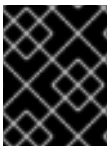
Red Hat Enterprise Linux 8 では、モジュールは AppStream リポジトリーにのみ存在します。利用可能なモジュールのリストを表示するには、インストールされている Red Hat Enterprise Linux 8 システムで **yum module list** コマンドを使用します。

キックスタートコマンド **module** を使用して、モジュールストリームを有効にし、直接命名して、モジュールストリームに含まれるパッケージをインストールすることもできます。

環境、グループ、パッケージの除外

ダッシュ (**-**) を先頭に付け、インストールから除外するパッケージやグループを指定します。以下に例を示します。

```
%packages
-@Graphical Administration Tools
-autofs
-ipa*compat
%end
```



重要

キックスタートファイルで * のみを使用して、利用可能なパッケージをすべてインストールする方法はサポートされていません。

%packages セクションのデフォルト動作は、オプションを使用して変更する方法がいくつかあります。オプションの中には、全パッケージの選択で機能するものと、特定のグループにのみ機能するものがあります。

関連情報

- ソフトウェアのインストール
- 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

I.2.3. 一般的なパッケージ選択のオプション

%packages では、以下のオプションが使用できます。オプションを使用するには、パッケージ選択セクションの最初に追加します。以下に例を示します。

```
%packages --multilib --ignoremissing
```

--default

パッケージのデフォルトセットをインストールします。これは、対話式インストールの **パッケージの選択** 画面でその他を選択しない場合にインストールされるパッケージセットに対応するものです。

--excludedocs

パッケージに含まれているドキュメンテーションをインストールしません。ほとんどの場合、通常 **/usr/share/doc** ディレクトリーにインストールされるファイルを除外しますが、個別に除外されるファイルは個別のパッケージによります。

--ignoremissing

インストールを停止してインストールの中断または続行を確認する代わりに、インストールソースにないパッケージ、グループ、モジュールストリーム、モジュールプロファイル、および環境を無視します。

--instLangs=

インストールする言語リストを指定します。これはパッケージグループレベルでの選択とは異なることに注意してください。このオプションでは、インストールするパッケージグループを記述するのではなく、RPM マクロを設定して、個別パッケージからインストールする翻訳ファイルを制御します。

--multilib

64 ビットのシステムに 32 ビットのパッケージをインストールできるように、multilib パッケージ用にインストールされたシステムを設定し、本セクションで説明しているようにパッケージをインストールします。
通常、AMD64 および Intel 64 のシステムでは、x86_64 パッケージおよび noarch パッケージのみをインストールできます。ただし、--multilib オプションを使用すると、32 ビット AMD および i686 Intel のシステムパッケージが存在する場合は自動的にインストールされます。

これは **%packages** セクションで明示的に指定されているパッケージにのみ適用されます。キックスタートファイルで指定されずに依存関係としてのみインストールされるパッケージは、他のアーキテクチャーで利用可能な場合でも、必要とされるアーキテクチャーのバージョンにのみインストールされます。

システムのインストール時に、Anaconda が **multilib** モードでパッケージをインストールするように設定できます。以下のいずれかのオプションを使用して **multilib** モードを有効にします。

1. 以下の行でキックスタートファイルを設定します。

```
%packages --multilib --default
%end
```

2. インストールイメージの起動中に、inst.multilib 起動オプションを追加します。

--nocore

@Core パッケージグループのインストールを無効にします。これを使用しない場合は、デフォルトでインストールされます。**--nocore** での **@Core** パッケージグループの無効化は、軽量コンテナーの作成にのみ使用してください。**--nocore** を指定してデスクトップやサーバーのシステムをインストールすると、システムが使用できなくなります。

注記

- **@Core** パッケージグループ内のパッケージを、**-@Core** を使用して除外することはできません。**@Core** パッケージグループを除外する唯一の方法は、**--nocore** オプションを使用することです。
- **@Core** パッケージグループは、作業 system のインストールに必要なパッケージの最小セットとして定義されています。これは、パッケージマニフェスト および 対象範囲の詳細 で定義されているコアパッケージには関係ありません。

--excludeWeakdeps

弱い依存関係からのパッケージのインストールを無効にします。これは、Recommends フラグおよび Supplements フラグで選択したパッケージセットにリンクされたパッケージです。デフォルトでは、弱い依存関係がインストールされます。

--retries=

YUM がパッケージのダウンロードを試みる回数を設定します (再試行)。デフォルト値は 10 です。このオプションはインストール時にのみ適用され、インストールされているシステムの YUM 設定には影響を及ぼしません。

--timeout=

YUM のタイムアウトを秒単位で設定します。デフォルト値は 30 です。このオプションはインストール時にのみ適用され、インストールされているシステムの YUM 設定には影響を及ぼしません。

I.2.4. 特定パッケージグループ用のオプション

以下のオプションは、単一パッケージグループにのみ適用されます。キックスタートファイルの **%packages** コマンドで使用する代わりに、グループ名に追加します。以下に例を示します。

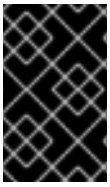
```
%packages
@Graphical Administration Tools --optional
%end
```

--nodefaults

デフォルト選択ではなく、グループの必須パッケージのみをインストールします。

--optional

デフォルトの選択に加えて、***-comps-repository.architecture.xml** ファイルのグループ定義でオプションの印が付けられているパッケージをインストールします。
Scientific Support のようなパッケージグループは、必須もしくはデフォルトのパッケージが指定されておらず、オプションのパッケージのみであることに注意してください。この場合は、**--optional** オプションを常に使用する必要があり、このオプションを使用しないと、このグループからパッケージがインストールできません。



重要

--nodefaults および **--optional** オプションは併用できません。**--nodefaults** を使用して、インストール中に必須パッケージのみをインストールし、インストール後にインストール済みシステムにオプションのパッケージをインストールできます。

I.3. キックスタートファイル内のスクリプト

キックスタートファイルには以下のスクリプトを追加できます。

- **%pre**
- **%pre-install**
- **%post**

本セクションでは、スクリプトに関する以下の情報を提供します。

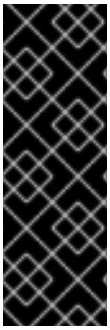
- 実行時間
- スクリプトに追加できるコマンドのタイプ
- スクリプトの目的
- スクリプトオプション

I.3.1. %pre スクリプト

%pre スクリプトは、キックスタートファイルの読み込み直後 (スクリプトが完全に解析され、インストールが開始する前) にシステムで実行されます。各セクションは、**%pre** で開始し、**%end** で終了する必要があります。

%pre スクリプトは、ネットワークおよびストレージデバイスのアクティベートおよび設定に使用できます。また、インストール環境で利用可能なインタープリターを使用して、スクリプトを実行することもできます。インストールを進める前に特定の設定を必要とするネットワークやストレージがある場合や、追加のログパラメーターや環境変数などを設定するスクリプトがある場合には、**%pre** スクリプトが便利になります。

%pre スクリプトでの問題のでバッグは難しくなる可能性があるので、**%pre** スクリプトは必要な場合にのみ使用することが推奨されます。



重要

キックスタートの **%pre** セクションは、インストーラーイメージ (**inst.stage2**) がフェッチされた後に発生するインストールの段階で実行されます。これは、root がインストーラー環境 (インストーラーイメージ) に切り替わった **後**、および **Anaconda** インストーラー自体が起動した **後** に実行されます。次に、**%pre** の設定が適用され、キックスタートの URL などで設定されたインストールリポジトリーからパッケージを取得するために使用できます。ただし、ネットワークからイメージ (**inst.stage2**) をフェッチするようにネットワークを設定するために使用する **ことはできません**。

インストール環境の **/sbin** ディレクトリーおよび **/bin** ディレクトリーにあるほとんどのユーティリティーの他に、**%pre** スクリプトでは、ネットワーク、ストレージ、およびファイルシステムに関連するコマンドを使用できます。

%pre セクションのネットワークにはアクセスできます。この時点では name サービスが設定されていないため、URL ではなく IP アドレスだけが有効です。

注記

pre スクリプトは、chroot 環境では実行しません。

I.3.1.1. %pre スクリプトセクションのオプション

以下のオプションを使用して、インストール前のスクリプトの動作を変更できます。オプションを使用するには、スクリプトの最初の部分で **%pre** 行にオプションを追加してください。以下に例を示します。

```
%pre --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

--interpreter=

Python などの別のスクリプト言語を指定できます。システムで利用可能なスクリプト言語は、どれでも使用できます。ほとんどの場合は、**/usr/bin/sh**、**/usr/bin/bash**、および **/usr/libexec/platform-python** になります。
platform-python インタープリターは、Python バージョン 3.6 を使用することに注意してください。新しいパスおよびバージョン用に、Python スクリプトを以前の RHEL バージョンから変更する必要があります。また、**platform-python** はシステムツール向けです。インストール環境外で **python36** を使用してください。Red Hat Enterprise Linux の Python の詳細は、 **Configuring basic system settings** の Introduction to Python を参照してください。

--erroronfail

スクリプトが失敗するとエラーを表示し、インストールを停止します。エラーメッセージは、失敗の原因がログ記録されている場所を示します。インストールされたシステムは、不安定で起動できない状態になる可能性があります。**inst.nokill** オプションを使用して、スクリプトをデバッグできます。

--log=

スクリプトの出力を、指定したログファイルに記録します。以下に例を示します。

```
%pre --log=/tmp/ks-pre.log
```

I.3.2. %pre-install スクリプト

pre-install スクリプトのコマンドは、以下のタスクの完了後に実行されます。

- システムのパーティションを設定した。
- ファイルシステムは /mnt/sysroot の下に作成およびマウントされます
- ネットワークが起動オプションとキックスタートコマンドに従って設定されている。

各 **%pre-install** セクションは、**%pre-install** で開始し、**%end** で終了します。

%pre-install スクリプトを使用してインストールを修正して、パッケージのインストール前に保証されている ID があるユーザーとグループを追加できます。

インストールに必要な変更には、**%post** スクリプトを使用することが推奨されます。**%pre-install** スクリプトは、**%post** スクリプトが必要な変更に満たない場合に限り使用します。

備考:**pre-install** スクリプトは、chroot 環境では実行しません。

I.3.2.1. %pre-install スクリプトセクションオプション

以下のオプションを使用して、**pre-install** のスクリプトの動作を変更できます。オプションを使用する場合は、スクリプトの先頭にある **%pre-install** 行に追加してください。以下に例を示します。

-

```
%pre-install --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

複数の **%pre-install** セクションを複数設定できます。インタープリターは同じものを複数回使用することもできます。設定したものは、キックスタートファイル内の参照順に評価されます。

--interpreter=

Python などの別のスクリプト言語を指定できます。システムで利用可能なスクリプト言語は、どれでも使用できます。ほとんどの場合は、**/usr/bin/sh**、**/usr/bin/bash**、および **/usr/libexec/platform-python** になります。
platform-python インタープリターは、Python バージョン 3.6 を使用することに注意してください。新しいパスおよびバージョン用に、Python スクリプトを以前の RHEL バージョンから変更する必要があります。また、**platform-python** はシステムツール向けです。インストール環境外で **python36** を使用してください。Red Hat Enterprise Linux の Python の詳細は、 **Configuring basic system settings** の Introduction to Python を参照してください。

--erroronfail

スクリプトが失敗するとエラーを表示し、インストールを停止します。エラーメッセージは、失敗の原因がログ記録されている場所を示します。インストールされたシステムは、不安定で起動できない状態になる可能性があります。**inst.nokill** オプションを使用して、スクリプトをデバッグできます。

--log=

スクリプトの出力を、指定したログファイルに記録します。以下に例を示します。

```
%pre-install --log=/mnt/sysroot/root/ks-pre.log
```

I.3.3. %post スクリプト

%post スクリプトは、インストールが完了した後、システムが最初に再起動する前に実行されるインストール後のスクリプトです。本セクションでは、システムのサブスクリプションなどのタスクを実行できます。

インストールが完了し、システムを最初に再起動する前に、システムで実行するコマンドを追加するオプションがあります。このセクションは、**%post** で始まり、**%end** で終了します。

%post セクションは、追加ソフトウェアのインストールや、追加のネームサーバーの設定といった機能に役に立ちます。インストール後のスクリプトは **chroot** 環境で実行するため、インストールメディアからスクリプトや RPM をコピーするなどの作業はデフォルトでは機能しません。この動作は、以下に記載されるように **--nochroot** オプションを使用することで変更できます。その後、**%post** スクリプトはインストール環境で実行し、インストール済みのターゲットシステムの **chroot** で実行することはありません。

インストール後のスクリプトは **chroot** 環境で実行されるため、ほとんどの **systemctl** コマンドはいかなるアクションも拒否します。

%post セクションの実行中にも、インストールメディアが挿入される必要があることに注意してください。

I.3.3.1. %post スクリプトセクションオプション

以下のオプションを使用して、インストール後のスクリプトの動作を変更できます。オプションを使用するには、スクリプトの最初の部分で **%post** 行にオプションを追加してください。以下に例を示します。

```
%post --interpreter=/usr/libexec/platform-python
-- Python script omitted --
%end
```

--interpreter=

Python などの別のスクリプト言語を指定できます。以下に例を示します。

```
%post --interpreter=/usr/libexec/platform-python
```

システムで利用可能なスクリプト言語は、どれでも使用できます。ほとんどの場合は、**/usr/bin/sh**、**/usr/bin/bash**、および **/usr/libexec/platform-python** になります。

platform-python インタープリターは、Python バージョン 3.6 を使用することに注意してください。新しいパスおよびバージョン用に、Python スクリプトを以前の RHEL バージョンから変更する必要があります。また、**platform-python** はシステムツール向けです。インストール環境外で **python36** を使用してください。Red Hat Enterprise Linux の Python の詳細は、 **Configuring basic system settings** の Introduction to Python を参照してください。

--nochroot

chroot 環境外で実行するコマンドを指定できます。
以下の例では、/etc/resolv.conf ファイルを、インストールしたばかりのファイルシステムにコピーします。

```
%post --nochroot
cp /etc/resolv.conf /mnt/sysroot/etc/resolv.conf
%end
```

--erroronfail

スクリプトが失敗するとエラーを表示し、インストールを停止します。エラーメッセージは、失敗の原因がログ記録されている場所を示します。インストールされたシステムは、不安定で起動できない状態になる可能性があります。**inst.nokill** オプションを使用して、スクリプトをデバッグできます。

--log=

スクリプトの出力を、指定したログファイルに記録します。ログファイルのパスは、ユーザーが **--nochroot** オプションを使用しているかどうかを考慮に入れる必要があることに注意して下さい。**--nochroot** がない場合の例を示します。

```
%post --log=/root/ks-post.log
```

--nochroot を使用した場合は、以下のようになります。

```
%post --nochroot --log=/mnt/sysroot/root/ks-post.log
```

I.3.3.2. たとえば、以下のようになります。インストール後スクリプトで NFS のマウント

この **%post** セクション例では、NFS 共有をマウントし、共有の **/usr/new-machines/** に置かれた **runme** スクリプトを実行します。キックスタートモードでは NFS ファイルのロックがサポートされていないため、**-o nolock** オプションが必要となることに注意してください。

```
# Start of the %post section with logging into /root/ks-post.log
%post --log=/root/ks-post.log

# Mount an NFS share
mkdir /mnt/temp
mount -o nolock 10.10.0.2:/usr/new-machines /mnt/temp
openvt -s -w -- /mnt/temp/runme
umount /mnt/temp

# End of the %post section
%end
```

I.3.3.3. たとえば、以下のようになります。インストール後のスクリプトで subscription-manager の実行

キックスタートを使用したインストールで最もよく使用されるインストール後のスクリプトの 1 つは、Red Hat Subscription Manager を使用したインストール済みシステムの自動登録です。以下は、**%post** スクリプトの自動サブスクリプションの例です。

```
%post --log=/root/ks-post.log
subscription-manager register --username=admin@example.com --password=secret --auto-attach
%end
```

subscription-manager のコマンドラインスクリプトで、システムが Red Hat Subscription Management サーバー (カスタマーポータルによるサブスクリプション管理、Satellite 6、CloudForms System Engine) に登録されます。このスクリプトは、システムに最も適したサブスクリプションをそのシステムに自動的に割り当てる場合にも使用できます。カスタマーポータルに登録する場合は、Red Hat Network ログイン認証情報を使用します。Satellite 6 または CloudForms System Engine に登録する場合は、ローカル管理者が提供する認証情報に加え、**--serverurl**、**--org**、**--environment** などの subscription-manager オプションも指定する必要があります。共有キックスタートファイルで、**--username --password** 値を公開しないようにするには、認証情報が、**--org --activationkey** の組み合わせの形式で使用されます。

登録コマンドで追加オプションを使用してシステムの優先サービスレベルを設定し、更新およびエラータを、以前のストリームで修正が必要な Extended Update Support サブスクリプションをお持ちのお客様の、特定のマイナーリリースバージョンの RHEL に制限することができます。

キックスタートの **%post** セクションで **subscription-manager** を使用する方法は、How do I use subscription-manager in a kickstart file? を参照してください。

I.4. ANACONDA 設定セクション

追加のインストールオプションは、キックスタートファイルの **%anaconda** セクションで設定できます。このセクションでは、インストールシステムのユーザーインターフェイスの動作を制御します。

本セクションは、キックスタートコマンドの後、キックスタートファイルの終わりの方に配置し、**%anaconda** で始まり **%end** で終了します。

現在、**%anaconda** セクションで使用できる唯一のコマンドは **pwpolicy** です。

例I.1%anaconda スクリプトのサンプル

以下は、%anaconda セクションの例です。

```
%anaconda
pwpolicy root --minlen=10 --strict
%end
```

上記の例では、**%anaconda** セクションではパスワードポリシーを設定します。root パスワードは 10 文字以上にする必要があり、この要件に一致しないものは厳密に禁止されます。

I.5. キックスタートでのエラー処理セクション

Red Hat Enterprise Linux 7 から、インストールプログラムが致命的なエラーに遭遇した場合に実行するカスタムスクリプトをキックスタートインストールに含めることができるようになりました。たとえば、インストールが要求されたパッケージにエラーがあったり、指定した VNC が起動に失敗したり、ストレージデバイスのスキャン中にエラーが発生する場合などです。このようなエラーが発生すると、インストールが続行できません。インストールプログラムは、キックスタートファイルで提供された順番で、すべての **%onerror** スクリプトを実行します。また、**%onerror** スクリプトは、トレースバックの際にも実行されます。

それぞれの **%onerror** スクリプトが、**%end** で終了する必要があります。

エラー処理のセクションでは、次のオプションを受け入れます。

--erroronfail

スクリプトが失敗するとエラーを表示し、インストールを停止します。エラーメッセージは、失敗の原因がログ記録されている場所を示します。インストールされたシステムは、不安定で起動できない状態になる可能性があります。**inst.nokill** オプションを使用して、スクリプトをデバッグできます。

--interpreter=

Python などの別のスクリプト言語を指定できます。以下に例を示します。

```
%onerror --interpreter=/usr/libexec/platform-python
```

システムで利用可能なスクリプト言語は、どれでも使用できます。ほとんどの場合は、**/usr/bin/sh**、**/usr/bin/bash**、および **/usr/libexec/platform-python** になります。

platform-python インタープリターは、Python バージョン 3.6 を使用することに注意してください。新しいパスおよびバージョン用に、Python スクリプトを以前の RHEL バージョンから変更する必要があります。また、**platform-python** はシステムツール向けです。インストール環境外で **python36** を使用してください。Red Hat Enterprise Linux の Python の詳細は、 **Configuring basic system settings** の Introduction to Python を参照してください。

--log=

スクリプトの出力を、指定したログファイルに記録します。

I.6. キックスタートのアドオンセクション

Red Hat Enterprise Linux 7 以降は、キックスタートインストールでアドオンをサポートするようになりました。これらのアドオンは、多くの方法で基本的なキックスタート (および Anaconda) の機能を拡張できます。

キックスタートファイルでアドオンを使用するには、**%addon addon_name options** コマンドを使用し、**%end** ステートメントでコマンドを終了します。これはインストール前およびインストール後スクリプトのセクションと似ています。たとえば、デフォルトで Anaconda で提供される Kdump アドオンを使用する場合は、次のコマンドを使用します。

```
%addon com_redhat_kdump --enable --reserve-mb=auto
%end
```

%addon コマンドには、独自のオプションが含まれていません。すべてのオプションは実際のアドオンに依存しています。

付録J キックスタートのコマンドおよびオプションの参照

ここでは、Red Hat Enterprise Linux インストールプログラムがサポートするキックスタートコマンドの一覧を提供します。コマンドは、いくつかのカテゴリーに分かれ、アルファベット順に記載されています。コマンドが複数のカテゴリーに該当する場合は、該当するすべてのカテゴリーに記載されます。

J.1. キックスタートの変更

以下のセクションでは、Red Hat Enterprise Linux 8 におけるキックスタートコマンドおよびオプションの変更を説明します。

RHEL 8 で auth または authconfig が非推奨に

authconfig ツールおよびパッケージが削除されたため、Red Hat Enterprise Linux 8 では、キックスタートコマンドの **auth** または **authconfig** が非推奨になっています。

コマンドラインで実行した **authconfig** コマンドと同様、キックスタートスクリプトの **authconfig** コマンドが **authselect-compat** ツールを使用して、新しい **authselect** ツールを実行するようになりました。この互換性層や、その既知の問題の説明は、**authselect-migration(7)** の man ページを参照してください。このインストールプログラムは、非推奨のコマンドの使用を自動的に検出し、互換性層を提供する **authselect-compat** パッケージをインストールします。

キックスタートで Btrfs がサポート対象外に

Red Hat Enterprise Linux 8 は、Btrfs ファイルシステムに対応していません。そのため、グラフィカルユーザーインターフェイス (GUI) およびキックスタートコマンドが Btrfs に対応しなくなりました。

以前の RHEL リリースのキックスタートファイルの使用

以前の RHEL リリースのキックスタートファイルを使用する場合は、Red Hat Enterprise Linux 8 BaseOS リポジトリーおよび AppStream リポジトリーの詳細について、Considerations in adopting RHEL 8 の **Repositories** のセクションを参照してください。

J.1.1. キックスタートで非推奨になったコマンドおよびオプション

次のキックスタートのコマンドとオプションが、Red Hat Enterprise Linux 8 で非推奨になりました。

特定のオプションだけがリスト表示されている場合は、基本コマンドおよびその他のオプションは引き続き利用でき、非推奨ではありません。

- **auth** または **authconfig** (代わりに **authselect** を使用)
- **device**
- **deviceprobe**
- **dmraid**
- **install** - サブコマンドまたはメソッドをそのままコマンドとして使用します。
- **multipath**
- **bootloader --upgrade**
- **ignoredisk --interactive**
- **partition --active**

- **reboot --kexec**
- **syspurpose** - 代わりに **subscription-manager syspurpose** を使用してください

auth コマンドまたは **authconfig** コマンドを除き、キックスタートファイルのコマンドを使用すると、ログに警告が出力されます。

inst.ksstrict ブートオプションで、**auth** コマンドまたは **authconfig** コマンドを除いた非推奨のコマンドの警告をエラーに変えることができます。

J.1.2. キックスタートから削除されたコマンドおよびオプション

次のキックスタートのコマンドとオプションが、Red Hat Enterprise Linux 8 から完全に削除されました。キックスタートファイルでこれを使用すると、エラーが発生します。

- **device**
- **deviceprobe**
- **dmraid**
- **install** - サブコマンドまたはメソッドをそのままコマンドとして使用します。
- **multipath**
- **bootloader --upgrade**
- **ignoredisk --interactive**
- **partition --active**
- **harddrive --biospart**
- **upgrade** (このコマンドはすでに非推奨になっています)
- **btrfs**
- **part/partition btrfs**
- **part --fstype btrfs** または **partition --fstype btrfs**
- **logvol --fstype btrfs**
- **raid --fstype btrfs**
- **unsupported_hardware**

特定のオプションおよび値だけが表示されている場合は、基本コマンドおよびその他のオプションは引き続き利用でき、削除されません。

J.2. インストールプログラムの設定とフロー制御のためのキックスタートコマンド

このリストのキックスタートコマンドは、インストールのモードとコースを制御し、最後に何が起こるかを制御します。

J.2.1. cdrom

キックスタートコマンドの **cdrom** は任意です。これは、システムの最初の光学ドライブからインストールを実行します。

構文

```
cdrom
```

備考

- **cdrom** コマンドは、以前は **install** コマンドとともに使用する必要がありました。**install** コマンドが非推奨になり、(**install** が暗黙的に使用されるようになったため) **cdrom** は独立して使用できるようになりました。
- このコマンドにはオプションはありません。
- 実際にインストールを実行するには、**cdrom**、**harddrive**、**hmc**、**nfs**、**liveimg**、または **url** のいずれかを指定する必要があります。

J.2.2. cmdline

キックスタートコマンドの **cmdline** は任意です。完全に非対話式のコマンドラインモードでインストールを実行します。対話のプロンプトがあるとインストールは停止します。

構文

```
cmdline
```

注記

- 完全に自動となるインストールでは、キックスタートファイルで利用可能なモード (**graphical**、**text**、または **cmdline**) のいずれかを指定するか、起動オプション **console=** を使用する必要があります。モードが指定されていないと、可能な場合はグラフィカルモードが使用されるか、VNC モードおよびテキストモードからの選択が求められます。
- このコマンドにはオプションはありません。
- このモードは、x3270 端末と共に 64 ビットの IBM Z システムで使用する場合に便利です。

J.2.3. driverdisk

キックスタートコマンドの **driverdisk** は任意です。このコマンドを使用して、インストールプログラムに追加ドライバーを提供します。

ドライバーディスクは、キックスタートを使用したインストール中に、デフォルトでは含まれていないドライバーを追加する場合に使用します。ドライバーディスクのコンテンツを、システムのハードドライブにあるパーティションのルートディレクトリーにコピーする必要があります。次に、**driverdisk** コマンドを使用して、インストールプログラムがドライバーディスクとその場所を検索するように指定する必要があります。

Syntax

```
driverdisk [partition|--source=url|--biospart=biospart]
```

オプション

この方法のいずれかで、ドライバーディスクの場所を指定する必要があります。

- partition - ドライバーディスクを含むパーティション。パーティションを指定する場合はパーティション名 (**sdb1** など) だけでは **なく**、完全パス (/**dev**/**sdb1** など) を使用してください。
- **--source=** - ドライバーディスクの URL。以下のようになります。

```
driverdisk --source=ftp://path/to/dd.img
driverdisk --source=http://path/to/dd.img
driverdisk --source=nfs:host:/path/to/dd.img
```

- **--biospart=** - ドライバーディスクを含む BIOS パーティション (**82p2** など)。

注記

ドライバーディスクは、ネットワーク経由や **initrd** から読み込むのではなく、ハードディスクドライブまたは同様のデバイスから読み込むこともできます。以下の手順に従います。

1. ハードディスクドライブ、USB、または同様のデバイスにドライバーディスクを読み込みます。
2. このデバイスに対して **DD** などのラベルを設定します。
3. キックスタートファイルに以下の行を追加します。

```
driverdisk LABEL=DD:/e1000.rpm
```

DD を具体的なラベルに置き換え、e1000.rpm を具体的な名前に置き換えます。LABEL ではなく、**inst.repo** コマンドがサポートするものを使用して、ハードディスクドライブを指定してください。

J.2.4. eula

キックスタートコマンドの **eula** は任意です。ユーザーとの対話なしでエンドユーザーライセンス契約 (EULA) に同意するには、このオプションを使用します。このオプションを使用すると、インストールを終了して、システムを最初に再起動した後に、ライセンス契約に同意するように求められなくなります。

Syntax

```
eula [--agreed]
```

オプション

- **--agreed** (必須) - EULA に同意します。このオプションは必ず使用する必要があります。使用しないと **eula** コマンド自体を使用する意味がなくなります。

J.2.5. firstboot

キックスタートコマンドの **firstboot** は任意です。初めてシステムを起動した時に、**初期セットアップ** アプリケーションを開始するかどうかを指定します。有効にする場合は、**initial-setup** パッケージをインストールする必要があります。何も指定しないとデフォルトで無効になるオプションです。

構文

```
firstboot OPTIONS
```

オプション

- **--enable** または **--enabled** - システムの初回起動時に、初期セットアップを開始します。
- **--disable** または **--disabled** - システムの初回起動時に、初期セットアップを開始しません。
- **--reconfig** - システムの起動時に、初期セットアップが再設定モードで開始します。このモードでは、デフォルトのオプションに加えて、root パスワード、時刻と日付、ネットワークとホスト名の設定オプションが有効になります。

J.2.6. graphical

キックスタートコマンドの **graphical** は任意です。これは、グラフィカルモードでインストールを実行します。これがデフォルトになります。

構文

```
graphical [--non-interactive]
```

オプション

- **--non-interactive** - 完全に非対話式のモードでインストールを実行します。このモードでは、ユーザーの対話が必要になるとインストールを終了します。

注記

- 完全に自動となるインストールでは、キックスタートファイルで利用可能なモード (**graphical**、**text**、または **cmdline**) のいずれかを指定するか、起動オプション **console=** を使用する必要があります。モードが指定されていないと、可能な場合はグラフィカルモードが使用されるか、VNC モードおよびテキストモードからの選択が求められます。

J.2.7. halt

キックスタートコマンドの **halt** は任意です。

インストールが正常に完了するとシステムを一時停止します。手動インストールと同じく、Anaconda のメッセージが表示され、ユーザーがキーを押すのを待ってから再起動が行われます。キックスタートを使用したインストールで、完了方法が指定されない場合は、このオプションがデフォルトとして使用されます。

構文

```
halt
```

注記

- **halt** コマンドは **shutdown -H** コマンドと同じです。詳細は、**shutdown(8)** の man ページを参照してください。

- 他の完了方法は、**poweroff**、**reboot**、**shutdown** などのコマンドをご覧ください。
- このコマンドにはオプションはありません。

J.2.8. harddrive

キックスタートコマンドの **harddrive** は任意です。ローカルドライブにある完全インストール用の ISO イメージまたは Red Hat インストールツリーからインストールします。ドライブは、インストールプログラムがマウントできるファイルシステムでフォーマットする必要があります (**ext2**、**ext3**、**ext4**、**vfat**、または **xfs**)。

Syntax

```
harddrive OPTIONS
```

オプション

- **--partition=** - インストールするパーティションを指定する場合に使用します (**sdb2** など)。
- **--dir=** - 完全インストール用 DVD の ISO イメージやインストールツリーの **variant** ディレクトリーを格納しているディレクトリーを指定する場合に使用します。

例

```
harddrive --partition=hdb2 --dir=/tmp/install-tree
```

注記

- **harddrive** コマンドは、**install** コマンドとともに使用する必要がありました。**install** コマンドが非推奨になり、(**install** が暗黙的に使用されるようになったため) **harddrive** は独立して使用できるようになりました。
- 実際にインストールを実行するには、**cdrom**、**harddrive**、**hmc**、**nfs**、**liveimg**、または **url** のいずれかを指定する必要があります。

J.2.9. install (非推奨)

重要

キックスタートコマンド **install** は、Red Hat Enterprise Linux 8 で非推奨になりました。そのメソッドは、別々のコマンドとして使用します。

キックスタートコマンドの **install** は任意です。デフォルトのインストールモードを指定します。

構文

```
install
installation_method
```

備考

- **install** コマンドに続いて、インストール方法のコマンドを指定する必要があります。インストール方法のコマンドは、別の行に指定する必要があります。
- 方法は次のとおりです。
  - **cdrom**
  - **harddrive**
  - **hmc**
  - **nfs**
  - **liveimg**
  - **url**

  メソッドの詳細は、個別のリファレンスページを参照してください。

J.2.10. liveimg

キックスタートコマンドの **liveimg** は任意です。パッケージの代わりに、ディスクイメージからインストールを実行します。

構文

```
liveimg --url=SOURCE [OPTIONS]
```

必須オプション

- **--url=** - インストール元となる場所です。**HTTP**、**HTTPS**、**FTP**、**file** が対応プロトコルになります。

任意のオプション

- **--url=** - インストール元となる場所です。**HTTP**、**HTTPS**、**FTP**、**file** が対応プロトコルになります。
- **--proxy=** - インストール実行時に使用する **HTTP**、**HTTPS**、または **FTP** プロキシーを指定します。
- **--checksum=** - 検証に使用するイメージファイルのチェックサム **SHA256** を使用するオプションの引数です。
- **--noverifyssl** - **HTTPS** サーバーへの接続の際に、SSL 確認を無効にします。

例

```
liveimg --url=file:///images/install/squashfs.img --
checksum=03825f567f17705100de3308a20354b4d81ac9d8bed4bb4692b2381045e56197 --
noverifyssl
```

注記

- イメージは、ライブ ISO イメージの **squashfs.img** ファイル、圧縮 tar ファイル

(**.tar**、**.tbz**、**.tgz**、**.txz**、**.tar.bz2**、**.tar.gz**、または **.tar.xz**)、もしくはインストールメディアでマウントできるファイルシステムであればどれでも構いません。**ext2**、**ext3**、**ext4**、**vfat**、**xfs** などが対応ファイルシステムになります。

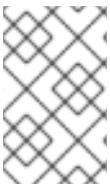
- ドライバーディスクで **liveimg** インストールモードを使用している場合、ディスク上のドライバーがインストールされるシステムに自動的に含まれることはありません。これらのドライバーが必要な場合は、手動でインストールするか、キックスタートスクリプトの **%post** セクションでインストールします。

- 実際にインストールを実行するには、**cdrom**、**harddrive**、**hmc**、**nfs**、**liveimg**、または **url** のいずれかを指定する必要があります。

- **liveimg** コマンドは、以前は **install** コマンドとともに使用する必要がありました。**install** コマンドが非推奨になり、(**install** が暗黙的に使用されるようになったため) **liveimg** は独立して使用できるようになりました。

J.2.11. logging

キックスタートコマンドの **logging** は任意です。インストール時に Anaconda に記録されるエラーログを制御します。インストール済みのシステムには影響しません。



注記

ロギングは TCP でのみサポートされています。リモートロギングの場合は、**--port=** オプションで指定するポート番号がリモートサーバーで開いていることを確認してください。デフォルトのポートは 514 です。

構文

```
logging OPTIONS
```

任意のオプション

- **--host=** - 指定したリモートホストにログ情報を送信します。ログを受け取るには、リモートホストで設定した syslogd プロセスが実行している必要があります。

- **--port=** - リモートの syslogd プロセスがデフォルト以外のポートを使用する場合は、このオプションを使用して設定します。

- **--level=** - tty3 に表示されるメッセージの最低レベルを指定します。ただし、このレベルに関係なくログファイルには全メッセージが送信されます。設定できるレベルは **debug**、**info**、**warning**、**error**、**critical** になります。

J.2.12. mediacheck

キックスタートコマンドの **mediacheck** は任意です。このコマンドを使用すると、インストール開始前にメディアチェックの実行が強制されます。インストール時の介入が必要となるため、デフォルトでは無効になっています。

構文

```
mediacheck
```

注記

注記

- このキックスタートコマンドは、**rd.live.check** 起動オプションに相当します。
- このコマンドにはオプションはありません。

J.2.13. nfs

キックスタートコマンドの **nfs** は任意です。指定した NFS サーバーからインストールを実行します。

構文

```
nfs OPTIONS
```

オプション

- **--server=** - インストール元となるサーバーを指定します (ホスト名または IP)。
- **--dir=** - インストールツリーの **variant** ディレクトリーを格納しているディレクトリーを指定する場合に使用します。
- **--opts=** - NFS エクスポートのマウントに使用するマウントポイントを指定します (オプション)。

例

```
nfs --server=nfsserver.example.com --dir=/tmp/install-tree
```

備考

- **nfs** コマンドは、以前は **install** コマンドとともに使用する必要がありました。**install** コマンドが非推奨になり、(**install** が暗黙的に使用されるようになったため) **nfs** は独立して使用できるようになりました。
- 実際にインストールを実行するには、**cdrom**、**harddrive**、**hmc**、**nfs**、**liveimg**、または **url** のいずれかを指定する必要があります。

J.2.14. ostreesetup

キックスタートコマンドの **ostreesetup** は任意です。これは、OStree ベースのインストールを設定するのに使用されます。

Syntax

```
ostreesetup --osname=OSNAME [--remote=REMOTE] --url=URL --ref=REF [--nogpg]
```

必須オプション:

- **--osname=OSNAME** - OS インストール用の root の管理
- **--url=URL** - インストール元となるリポジトリーの URL
- **--ref=REF** - インストールに使用するリポジトリーのブランチー名

任意のオプション:

- **--remote=REMOTE** - OS インストール用の管理ルート
- **--nogpg** - GPG 鍵の検証の無効化

注記

- OStree ツールの詳細は、アップストリームのドキュメント https://ostree.readthedocs.io/en/latest/ を参照してください。

J.2.15. poweroff

キックスタートコマンドの **poweroff** は任意です。インストールが正常に完了したら、システムをシャットダウンして電源を切ります。通常、手動のインストールでは Anaconda によりメッセージが表示され、ユーザーがキーを押すのを待ってから再起動が行われます。

構文

```
poweroff
```

注記

- **poweroff** オプションは **shutdown -P** コマンドと同じです。詳細は、**shutdown(8)** の man ページを参照してください。
- 他の完了方法は、**halt**、**reboot**、**shutdown** などのキックスタートコマンドをご覧ください。キックスタートファイルに完了方法が明示的には指定されていない場合は、**halt** オプションがデフォルトの完了方法になります。
- **poweroff** オプションは、使用中のハードウェアに大きく依存します。特に、BIOS、APM (advanced power management)、ACPI (advanced configuration and power interface) などの特定ハードウェアコンポーネントは、システムカーネルと対話できる状態にする必要があります。使用システムの APM/ACPI 機能は、製造元発行のドキュメントをご覧ください。
- このコマンドにはオプションはありません。

J.2.16. reboot

キックスタートコマンドの **reboot** は任意です。インストールが正常に完了したらシステムを再起動するように、インストールプログラムに指示します (引数なし)。通常、キックスタートは、メッセージを表示し、ユーザーがキーを押してから再起動します。

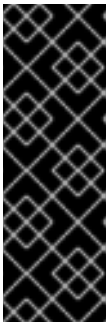
構文

```
reboot OPTIONS
```

オプション

- **--eject** - 再起動の前に起動可能なメディア (DVD、USB、またはその他のメディア) の取り出しを試みます。

- **--kexec** - 完全な再起動を実行する代わりに **kexec** システムコールを使用します。BIOS やファームウェアが通常実行するハードウェアの初期化を行わずに、インストールしたシステムを即座にメモリーに読み込みます。



重要

このオプションは非推奨になっており、テクノロジープレビューとしてのみ利用できます。テクノロジープレビュー機能に対する Red Hat のサポート範囲の詳細は、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。

kexec の使用時には、(完全なシステム再起動では通常クリアされる) デバイスレジスターにデータが残ります。デバイスドライバーによってはこれが問題になる可能性もあります。

注記

- インストールメディアやインストール方法によっては、**reboot** オプションを使用するとインストールプロセスがループして**完了しなくなる場合があります**。
- **reboot** オプションは **shutdown -r** コマンドと同じです。詳細は、**shutdown(8)** の man ページを参照してください。
- 64 ビットの IBM Z でコマンドラインによるインストールを行う際は、**reboot** を指定してインストールを完全自動化します。
- これ以外の完了方法については、**halt**、**poweroff**、**shutdown** などのキックスタートオプションをご覧ください。キックスタートファイルに完了方法が明示的には指定されていない場合は、**halt** オプションがデフォルトの完了方法になります。

J.2.17. rhsm

キックスタートコマンドの **rhsm** は任意です。ここでは、インストールプログラムにより、CDN から RHEL が登録されインストールされるようになっています。

注記

キックスタートコマンド **rhsm** は、システムの登録時にカスタムの **%post** スクリプトを使用する要件を削除します。

オプション

- **--organization=** - 組織 ID を使用して CDN から RHEL を登録してインストールします。
- **--activation-key=** - アクティベーションキーを使用して、CDN から RHEL を登録してインストールします。使用するアクティベーションキーがサブスクリプションに登録されている限り、アクティベーションキーごとに 1 回使用するオプションを複数回使用できます。
- **--connect-to-insights** - ターゲットシステムを Red Hat Insights に接続します。
- **--proxy=** - HTTP プロキシーを設定します。

J.2.18. shutdown

キックスタートコマンドの **shutdown** は任意です。インストールが正常に完了したら、システムをシャットダウンします。

構文

```
shutdown
```

注記

- キックスタートオプションの **shutdown** は、**shutdown** コマンドと同じです。詳細は、**shutdown(8)** の man ページを参照してください。
- その他の完了方法は、**halt**、**poweroff**、**reboot** などのキックスタートオプションをご覧ください。キックスタートファイルに完了方法が明示的には指定されていない場合は、**halt** オプションがデフォルトの完了方法になります。
- このコマンドにはオプションはありません。

J.2.19. sshpw

キックスタートコマンドの **sshpw** は任意です。

インストール中に、**SSH** 接続によりインストールプログラムと対話操作を行い、その進捗状況を監視できます。**sshpw** コマンドを使用して、ログオンするための一時的なアカウントを作成します。コマンドの各インスタンスにより、インストール環境でしか存在しない個別アカウントが作成されます。ここで作成されたアカウントは、インストールが完了したシステムには転送されません。

Syntax

```
sshpw --username=name [OPTIONS] password
```

必須オプション

- **--username**=name - ユーザー名を入力します。このオプションは必須です。
- password - このユーザーに使用するパスワードです。このオプションは必須です。

任意のオプション

- **--iscrypted** - このオプションを追加すると、パスワード引数はすでに暗号化済みと仮定されます。**--plaintext** と相互排他的になります。暗号化したパスワードを作成する場合は Python を使用します。

  ```
  $ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  上記の例では、ランダムの salt を使用して、パスワードの sha512 暗号と互換性があるハッシュが生成されます。

- **--plaintext** - このオプションを使用すると、パスワードの引数はプレーンテキストであると仮定されます。**--iscrypted** と相互排他的になります。
- **--lock** - このオプションを指定すると、このアカウントはデフォルトでロックされます。つまり、ユーザーはコンソールからログインできなくなります。
- **--sshkey** - このオプションを指定すると、**<password>** 文字列が ssh 鍵の値として解釈されます。

注記

- デフォルトでは、**ssh** サーバーは、インストール時に起動しません。インストール時に **ssh** を使用できるようにするには、カーネル起動オプション **inst.sshd** を使用してシステムを起動します。
- インストール中、別のユーザーの **ssh** アクセスを許可する一方で、root の **ssh** アクセスを無効にする場合は、以下のコマンドを実行します。

```
sshpw --username=example_username example_password --plaintext
sshpw --username=root example_password --lock
```

- 単に root の **ssh** アクセスを無効にするには、以下のコマンドを使用します。

```
sshpw --username=root example_password --lock
```

J.2.20. text

キックスタートコマンドの **text** は任意です。テキストモードでキックスタートインストールを実行します。キックスタートインストールは、デフォルトでグラフィカルモードで実行します。

構文

```
text [--non-interactive]
```

オプション

- **--non-interactive** - 完全に非対話式のモードでインストールを実行します。このモードでは、ユーザーの対話が必要になるとインストールを終了します。

注記

- 完全に自動となるインストールでは、キックスタートファイルで利用可能なモード (**graphical**、**text**、または **cmdline**) のいずれかを指定するか、起動オプション **console=** を使用する必要がある点に注意してください。モードが指定されていないと、可能な場合はグラフィカルモードが使用されるか、VNC モードおよびテキストモードからの選択が求められます。

J.2.21. url

キックスタートコマンドの **url** は任意です。これは、FTP、HTTP、または HTTPS プロトコルを使用して、リモートサーバーのインストールツリーイメージからインストールするのに使用されます。URL は 1 つだけ指定できます。

構文

```
url --url=FROM [OPTIONS]
```

必須オプション

- **--url=FROM** - インストール元となる **HTTP**、**HTTPS**、**FTP**、または **ファイル** の場所を指定します。

任意のオプション

- **--mirrorlist=** - インストール元となるミラー URL を指定します。
- **--proxy=** - インストール時に使用する **HTTP**、**HTTPS**、または **FTP** プロキシーを指定します。
- **--noverifyssl** - **HTTPS** サーバーへの接続時に SSL 検証を無効にします。
- **--metalink=URL** - インストール元となるメタリンク URL を指定します。変数の置換は、URLの **$releasever** および **$basearch** で行います。

例

- HTTP サーバーからインストールするには、以下を行います。

```
url --url=http://server/path
```

- FTP サーバーからインストールするには、以下を行います。

```
url --url=ftp://username:password@server/path
```

- ローカルファイルからインストールするには、以下を行います。

```
liveimg --url=file:///images/install/squashfs.img --noverifyssl
```

注記

- **url** コマンドは、以前は **install** コマンドとともに使用する必要がありました。**install** コマンドが非推奨になり、(**install** が暗黙的に使用されるようになったため) **url** は独立して使用できるようになりました。
- 実際にインストールを実行するには、**cdrom**、**harddrive**、**hmc**、**nfs**、**liveimg**、または **url** のいずれかを指定する必要があります。

J.2.22. vnc

キックスタートコマンドの **vnc** は任意です。これにより、VNC を介して、リモートにグラフィカルインストールを表示できます。

テキストインストールではサイズと言語の一部が制限されるため、通常はテキストモードよりもこの方法が好まれます。追加のオプション指定がないと、このコマンドは、パスワードを使用せずに、インストールシステムで VNC サーバーを開始し、接続に必要な詳細を表示します。

Syntax

```
vnc [--host=host_name] [--port=port] [--password=password]
```

オプション

--host=

指定したホスト名でリッスンしている VNC ビューアープロセスに接続します。

--port=

リモート VNC ビューアープロセスがリッスンしているポートを指定します。このオプションを使用しないと、Anaconda は VNC のデフォルトポートである 5900 を使用します。

--password=

VNC セッションへの接続に必要なパスワードを設定します。これはオプションですが、推奨されます。

関連情報

- PXE を使用してネットワークからインストールするための準備

J.2.23. %include

キックスタートコマンドの **%include** は任意です。

%include コマンドを使用して、キックスタートファイル内の別のファイルのコンテンツが、キックスタートファイルの **%include** コマンドの場所にあるかのように設定します。

この包含は、**%pre** スクリプトセクションの後にのみ評価されるため、**%pre** セクションでスクリプトにより生成されたファイルに使用できます。**%pre** セクションを評価する前にファイルを指定するには、**%ksappend** コマンドを使用します。

構文

```
%include path/to/file
```

J.2.24. %ksappend

キックスタートコマンドの **%ksappend** は任意です。

%ksappend コマンドを使用して、キックスタートファイル内の別のファイルのコンテンツが、キックスタートファイルの **%ksappend** コマンドの場所にあるかのように設定します。

この包含は、**%include** コマンドで使用するのとは異なり、**%pre** スクリプトセクションの前に評価されます。

構文

```
%ksappend path/to/file
```

J.3. システム設定用キックスタートコマンド

このリストのキックスタートコマンドは、ユーザー、リポジトリー、サーバーなど、システムの詳細を設定します。

J.3.1. auth または authconfig (非推奨)

重要

非推奨になったキックスタートコマンドの **auth** または **authconfig** を使用する代わりに **authselect** コマンドを使用します。**auth** および **authconfig** は、限定された後方互換性にのみ利用できます。

キックスタートコマンドの **auth** または **authconfig** は任意です。**authconfig** ツールを使用してシステムの認証オプションを設定します。インストール完了後もコマンドラインで実行できます。

Syntax

```
authconfig [OPTIONS]
```

注記

- キックスタートコマンドの **auth** または **authconfig** コマンドは、以前は **authconfig** ツールと呼ばれていました。このツールは、Red Hat Enterprise Linux 8 では非推奨になりました。このキックスタートコマンドは、**authselect-compat** ツールを使用して、新しい **authselect** ツールを呼び出せるようになりました。互換性層の説明と、その既知の問題は、**authselect-migration(7)** の man ページを参照してください。インストールプログラムが自動的に非推奨のコマンドの使用を検出し、互換性層を提供するために、システムに **authselect-compat** パッケージをインストールします。
- デフォルトでは、パスワードがシャドウ化されています。
- 安全対策上、**SSL** プロトコルで OpenLDAP を使用する場合はサーバー設定内の **SSLv2** および **SSLv3** のプロトコルを必ず無効にしてください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は https://access.redhat.com/solutions/1234843 を参照してください。

J.3.2. authselect

キックスタートコマンドの **authselect** は任意です。**authselect** コマンドを使用してシステムの認証オプションを設定します。インストール完了後もコマンドラインで実行できます。

Syntax

```
authselect [OPTIONS]
```

注記

- このコマンドは、すべてのオプションを **authselect** コマンドに渡します。詳細は、**authselect(8)** の man ページ、および **authselect --help** コマンドを参照してください。
- このコマンドは、Red Hat Enterprise Linux 8 で非推奨になった **auth** または **authconfig** コマンドを、**authconfig** ツールに置き換えます。
- デフォルトでは、パスワードがシャドウ化されています。
- 安全対策上、**SSL** プロトコルで OpenLDAP を使用する場合はサーバー設定内の **SSLv2** および **SSLv3** のプロトコルを必ず無効にしてください。POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受けないようにするためです。詳細は https://access.redhat.com/solutions/1234843 を参照してください。

J.3.3. firewall

キックスタートコマンドの **firewall** は任意です。インストール済みシステムにファイアウォール設定を指定します。

Syntax

```
firewall --enabled|--disabled [incoming] [OPTIONS]
```

必須オプション

- **--enabled** または **--enable** - DNS 応答や DHCP 要求など、発信要求に対する応答ではない着信接続を拒否します。このマシンで実行中のサービスへのアクセスが必要な場合は、特定サービスに対してファイアウォールの通過許可を選択できます。
- **--disabled** または **--disable** - iptable ルールを一切設定しません。

任意のオプション

- **--trust** - **em1** などのデバイスを指定することで、ファイアウォールを通過するこのデバイスへの着信トラフィックおよびこのデバイスからの発信トラフィックをすべて許可します。複数のデバイスをリスト表示するには、**--trust em1 --trust em2** などのオプションをさらに使用します。**--trust em1, em2** などのようなコンマ区切りは使用しないでください。
- **--remove-service** - サービスがファイアウォールを通過するのを許可しません。
- incoming - 指定したサービスがファイアウォールを通過できるように、以下のいずれかに置き換えます (複数のサービスを指定できます)。
  - **--ssh**
  - **--smtp**
  - **--http**
  - **--ftp**
- **--port=** - port:protocol の形式で指定したポートのファイアウォール通過を許可できます。たとえば、IMAP アクセスがファイアウォールを通過できるようにする場合は、**imap:tcp** と指定します。ポート番号を明示的に指定することもできます。ポート 1234 の UDP パケットを許可する場合は **1234:udp** と指定します。複数のポートを指定する場合は、コンマで区切って指定します。
- **--service=** - このオプションは、サービスがファイアウォールを通過できるように高レベルの方法を提供します。サービスの中には複数のポートを開く必要があったり (**cups**、**avahi** など)、サービスが正常に動作するよう特殊な設定を必要とするものがあります。このような場合は、**--port** オプションでポート単位での指定を行ったり、**--service=** を使用して必要なポートをすべて一度に開くことが可能です。
  firewalld パッケージ内の **firewall-offline-cmd** プログラムで認識できるオプションは、すべて使用できます。**firewalld** サービスを実行している場合は、**firewall-cmd --get-services** を実行すると、認識できるサービス名のリストが表示されます。
- **--use-system-defaults** - ファイアウォールを設定しません。このオプションにより、anaconda では何も実行せず、システムが、パッケージまたは ostree で提供されるデフォルトに依存するようになります。このオプションを他のオプションと共に使用すると、他のすべてのオプションは無視されます。

J.3.4. group

キックスタートコマンドの **group** は任意です。システムに新しいユーザーグループを作成します。

```
group --name=name [--gid=gid]
```

必須オプション

- **--name=** - グループ名を指定します。

任意のオプション

- **--gid=** - グループの GID です。指定しないとシステムの GID 以外で次に使用可能な GID がデフォルト設定されます。

注記

- 指定された名前や GID を持つグループが存在すると、このコマンドは失敗します。
- **user** コマンドは、新たに作成したユーザーに新しいグループを作成するのに使用できます。

J.3.5. keyboard (必須)

キックスタートコマンド **keyboard** が必要です。これは、システムに利用可能なキーボードレイアウトを 1 つまたは複数設定します。

構文

```
keyboard --vckeymap|--xlayouts OPTIONS
```

オプション

- **--vckeymap=** - 使用する **VConsole** キーマップを指定します。**/usr/lib/kbd/keymaps/xkb/** ディレクトリーの各ファイル名から **.map.gz** 拡張子を外したものが、有効なキーマップ名になります。
- **--xlayouts=** - 使用する X のレイアウトを、空白なしのコンマで区切ったリストで指定します。**setxkbmap(1)** と同じ形式 (**layout** 形式 (**cz** など)、または **layout (variant)** 形式 (**cz (qwerty)** など)) の値をとります。
  使用できるレイアウトは、**Layouts** の **xkeyboard-config(7)** man ページをご覧ください。
- **--switch=** - レイアウト切り替えのオプションリストを指定します (複数のキーボードレイアウト切り替え用のショートカット)。複数のオプションは、空白なしのコンマで区切ってください。**setxkbmap(1)** と同じ形式の値を受け取ります。
  使用できる切り替えオプションは、**xkeyboard-config(7)** の man ページの **Options** をご覧ください。

注記

- **--vckeymap=** オプションまたは **--xlayouts=** オプションのいずれかを使用する必要があります。

例

以下の例では、**--xlayouts=** オプションを使用して 2 種類のキーボードレイアウト (**English (US)** と **Czech (qwerty)**) を設定し、切り替えオプションは、**Alt**+**Shift** を使用するように指定しています。

```
keyboard --xlayouts=us,'cz (qwerty)' --switch=grp:alt_shift_toggle
```

J.3.6. lang (必須)

キックスタートコマンドの **lang** が必要です。これは、インストール時に使用する言語と、インストール済みシステムで使用するデフォルト言語を設定します。

Syntax

```
lang language [--addsupport=language,...]
```

必須オプション

- **language** - この言語のサポートをインストールし、システムのデフォルトとして設定します。

任意のオプション

- **--addsupport=** - 追加言語のサポートを指定します。空白を入れずコンマで区切った形式を受け取ります。以下に例を示します。

  ```
  lang en_US --addsupport=cs_CZ,de_DE,en_UK
  ```

注記

- **locale -a | grep _** コマンドまたは **localectl list-locales | grep _** コマンドは、ロケールのリストを返します。
- テキストモードのインストールでは、特定の言語には対応していません (中国語、日本語、韓国語、インド系言語など)。**lang** コマンドでこの言語を指定しても、インストールプロセスは英語で続行します。ただし、インストール後のシステムでは選択した言語がデフォルトの言語として使用されます。

例

言語を英語に設定するには、キックスタートファイルに次の行が含まれている必要があります。

```
lang en_US
```

J.3.7. module

キックスタートコマンドの **module** は任意です。このコマンドを使用すると、キックスタートスクリプトでパッケージのモジュールストリームが有効になります。

Syntax

```
module --name=NAME [--stream=STREAM]
```

必須オプション

--name=

有効にするモジュールの名前を指定します。**NAME** を、実際の名前に置き換えます。

任意のオプション

--stream=

有効にするモジュールストリームの名前を指定します。**STREAM** を、実際の名前に置き換えます。

デフォルトストリームが定義されているモジュールには、このオプションを指定する必要はありません。デフォルトストリームのないモジュールの場合、このオプションは必須であり省略するとエラーになります。異なるストリームでモジュールを複数回有効にすることはできません。

注記

- このコマンドと **%packages** セクションを組み合わせて使用すると、モジュールとストリームを明示的に指定せずに、有効なモジュールとストリームの組み合わせで提供されるパッケージをインストールできます。モジュールは、パッケージをインストールする前に有効にする必要があります。**module** コマンドでモジュールを有効にしたら、**%packages** セクションにパッケージのリストを追加することで、このモジュールで有効にしたパッケージをインストールできます。
- 1 つの **module** コマンドで、1 つのモジュールとストリームの組み合わせのみを有効にできます。複数のモジュールを有効にするには、複数の **module** コマンドを使用します。異なるストリームでモジュールを複数回有効にすることはできません。
- Red Hat Enterprise Linux 8 では、モジュールは AppStream リポジトリーにのみ存在します。利用可能なモジュールのリストを表示するには、インストールされている Red Hat Enterprise Linux 8 システムで **yum module list** コマンドを実行します。

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

J.3.8. repo

キックスタートコマンドの **repo** は任意です。パッケージインストール用のソースとして使用可能な追加の yum リポジトリーを設定します。複数の **repo** 行を追加できます。

Syntax

```
repo --name=repoid [--baseurl=url|--mirrorlist=url|--metalink=url] [OPTIONS]
```

必須オプション

- **--name=** - リポジトリー ID を入力します。このオプションは必須です。以前に追加したリポジトリーと名前が競合する場合は無視されます。インストールプログラムでは事前設定したリポジトリーのリストが使用されるため、このリストにあるリポジトリーと同じ名前のものは追加できません。

URL オプション

これらのオプションは相互排他的で、オプションです。ここでは、yum のリポジトリーの設定ファイル内で使用できる変数はサポートされません。文字列 **$releasever** および **$basearch** を使用できます。これは、URL の該当する値に置き換えられます。

- **--baseurl=** - リポジトリーの URL を入力します。
- **--mirrorlist=** - リポジトリーのミラーのリストを指す URL を入力します。
- **--metalink=** - リポジトリーのメタリンクを持つ URL です。

任意のオプション

- **--install** - 指定したリポジトリーの設定を、インストールしたシステムの **/etc/yum.repos.d/** ディレクトリーに保存します。このオプションを使用しない場合は、キックスタートファイルで設定したリポジトリーの使用はインストール中に限られ、インストール後のシステムでは使用できません。

- **--cost=** - このリポジトリーに割り当てるコストを整数で入力します。複数のリポジトリーで同じパッケージを提供している場合に、リポジトリーの使用優先順位がこの数値で決まります。コストの低いリポジトリーは、コストの高いリポジトリーよりも優先されます。

- **--excludepkgs=** - このリポジトリーからは読み出しては**ならない**パッケージ名のリストをコンマ区切りで指定します。複数のリポジトリーで同じパッケージが提供されていて、特定のリポジトリーから読み出す場合に便利なオプションです。(**publican** といった) 完全なパッケージ名と (**gnome-*** といった) グロブの両方が使えます。

- **--includepkgs=** - このリポジトリーから取得できるパッケージ名およびグロブのリストをコンマ区切りで指定します。リポジトリーが提供するその他のパッケージは無視されます。これは、リポジトリーが提供する他のパッケージをすべて除外しながら、リポジトリーから 1 つのパッケージまたはパッケージセットをインストールする場合に便利です。

- **--proxy=[protocol://][username[:password]@]host[:port]** - このリポジトリーにだけ使用する HTTP/HTTPS/FTP プロキシーを指定します。この設定は他のリポジトリーには影響しません。また、HTTP インストールでは **install.img** の読み込みについても影響はありません。

- **--noverifyssl** - **HTTPS** サーバーへの接続の際に、SSL 確認を無効にします。

注記

- インストールに使用するリポジトリーは安定した状態を維持してください。インストールが終了する前にリポジトリーに変更が加えられると、インストールが失敗する可能があります。

J.3.9. rootpw (必須)

キックスタートコマンドの **rootpw** が必要です。システムの root パスワードを **password** 引数に設定します。

構文

```
rootpw [--iscrypted|--plaintext] [--lock] password
```

必須オプション

- password - パスワード指定。プレーンテキストまたは暗号化された文字列。以下の **--iscrypted** および **--plaintext** を参照してください。

オプション

- **--iscrypted** - このオプションを追加すると、パスワード引数はすでに暗号化済みと仮定されます。**--plaintext** と相互排他的になります。暗号化したパスワードを作成する場合は python を使用します。

  ```
  $ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
  (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  上記の例では、ランダムの salt を使用して、パスワードの sha512 暗号と互換性があるハッシュが生成されます。

- **--plaintext** - このオプションを使用すると、パスワードの引数はプレーンテキストであると仮定されます。**--iscrypted** と相互排他的になります。
- **--lock** - このオプションを指定すると、root アカウントはデフォルトでロックされます。つまり、root ユーザーはコンソールからログインできなくなります。また、グラフィカルおよびテキストベースの手動インストールで、**Root Password** ウィンドウが無効になります。

J.3.10. selinux

キックスタートコマンドの **selinux** は任意です。インストール済みシステムの SELinux の状態を設定します。デフォルトの SELinux ポリシーは **enforcing** です。

構文

```
selinux [--disabled|--enforcing|--permissive]
```

オプション

--enforcing

SELinux をデフォルトの対象ポリシーである **enforcing** で有効にします。

--permissive

SELinux のポリシーに基づく警告を出力します。ただし、実際にはポリシーは実施されません。

--disabled

システムで SELinux を完全に無効にします。

関連情報

- Using SElinux

J.3.11. services

キックスタートコマンドの **services** は任意です。デフォルトの systemd ターゲット下で実行するデフォルトのサービスセットを変更します。無効にするサービスのリストは、有効にするサービスのリストの前に処理されます。したがって、サービスが両方のリストに記載されていると、そのサービスは有効になります。

Syntax

```
services [--disabled=list] [--enabled=list]
```

オプション

- **--disabled=** - 無効にするサービスをコンマ区切りで指定します。
- **--enabled=** - 有効にするサービスをコンマ区切りで指定します。

注記

- サービスのリストには空白文字を使用しないでください。空白があると、キックスタートでは、最初の空白の直前のサービスまでしか有効または無効になりません。以下に例を示します。
  -

```
services --disabled=auditd, cups,smartd, nfslock
```

この場合は、**auditd** サービスしか無効になりません。4 つのサービスをすべて無効にするには、エントリーから空白を取り除きます。

```
services --disabled=auditd,cups,smartd,nfslock
```

J.3.12. skipx

キックスタートコマンドの **skipx** は任意です。存在する場合は、インストール済みシステムで X が設定されていません。

パッケージ選択のオプションでディスプレイマネージャーをインストールすると、このパッケージにより X の設定が作成されるため、インストールが完了したシステムは **graphical.target** にデフォルト設定されることになります。これにより、**skipx** オプションが無効になります。

構文

```
skipx
```

注記

- このコマンドにはオプションはありません。

J.3.13. sshkey

キックスタートコマンドの **sshkey** は任意です。インストール済みシステムで、指定したユーザーの **authorized_keys** ファイルに SSH キーを追加します。

Syntax

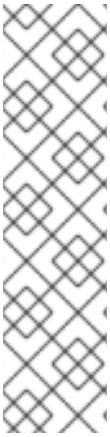
```
sshkey --username=user "ssh_key"
```

必須オプション

- **--username=** - 鍵をインストールするユーザー。
- **ssh_key** - 完全な SSH 鍵のフィンガープリント。引用符でラップする必要があります。

J.3.14. syspurpose

キックスタートコマンドの **syspurpose** は任意です。インストール後にシステムがどのように使用されるかを説明するシステムの目的を設定します。この情報により、適切なサブスクリプションエンタイトルメントがシステムに適用されます。



注記

Red Hat Enterprise Linux 8.6 以降では、1 つの **subscription-manager syspurpose** モジュールで **role**、**service-level**、**usage**、および **addons** サブコマンドを利用可能にすることで、1 つのモジュールでシステムの目的の属性を管理および表示できます。以前は、システム管理者は 4 つのスタンドアロンの **syspurpose** コマンドのいずれかを使用して各属性を管理していました。このスタンドアロンの **syspurpose** コマンドは RHEL 8.6 以降非推奨となり、RHEL 9 では削除される予定です。Red Hat は、現在のリリースのライフサイクル中にバグ修正とこの機能に対するバグ修正やサポートを提供しますが、この機能は機能強化の対象外となります。RHEL 9 以降、単一の **subscription-manager syspurpose** コマンドとその関連のサブコマンドは、システムの目的を使用する唯一の方法です。

Syntax

```
syspurpose [OPTIONS]
```

オプション

- **--role=** - 希望するシステムロールを設定します。利用できる値は次のとおりです。
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Workstation
  - Red Hat Enterprise Linux Compute Node
- **--sla=** - サービスレベルアグリーメントを設定します。利用できる値は次のとおりです。
  - Premium
  - Standard
  - Self-Support
- **--usage=** - システムの使用方法。利用できる値は次のとおりです。
  - Production
  - Disaster Recovery
  - Development/Test
- **--addon=** - のレイヤード製品または機能を指定します。このオプションは複数回使用できます。

注記

- スペースで値を入力し、二重引用符で囲みます。

  ```
  syspurpose --role="Red Hat Enterprise Linux Server"
  ```

- システムの目的を設定することが強く推奨されますが、Red Hat Enterprise Linux インストールプログラムでは任意の機能です。インストールが完了してからシステムの目的を有効にする場合は、コマンドラインツールの **syspurpose** を使用できます。

注記

Red Hat Enterprise Linux 8.6 以降では、1 つの **subscription-manager syspurpose** モジュールで **role**、**service-level**、**usage**、および **addons** サブコマンドを利用可能にすることで、1 つのモジュールでシステムの目的の属性を管理および表示できます。以前は、システム管理者は 4 つのスタンドアロンの **syspurpose** コマンドのいずれかを使用して各属性を管理していました。このスタンドアロンの **syspurpose** コマンドは RHEL 8.6 以降非推奨となり、RHEL 9 では削除される予定です。Red Hat は、現在のリリースのライフサイクル中にバグ修正とこの機能に対するバグ修正やサポートを提供しますが、この機能は機能強化の対象外となります。RHEL 9 以降、単一の **subscription-manager syspurpose** コマンドとその関連のサブコマンドは、システムの目的を使用する唯一の方法です。

J.3.15. timezone (必須)

キックスタートコマンド **timezone** が必要です。システムのタイムゾーンを設定します。

Syntax

```
timezone timezone [OPTIONS]
```

必須オプション

- **timezone** - システムに設定するタイムゾーン

任意のオプション

- **--utc** - これを指定すると、ハードウェアクロックが UTC (グリニッジ標準) 時間に設定されているとシステムは見なします。
- **--nontp** - NTP サービスの自動起動を無効にします。
- **--ntpservers=** - 使用する NTP サーバーを空白を入れないコンマ区切りのリストで指定します。

注記

Red Hat Enterprise Linux 8 では、タイムゾーン名は **pytz** パッケージにより提供される **pytz.all_timezones** のリストを使用して検証されます。以前のリリースでは、名前は現在使用されているリストのサブセットである **pytz.common_timezones** に対して検証されていました。グラフィックおよびテキストモードのインターフェイスには、引き続きより制限の多い **pytz.common_timezones** のリストが使用される点に注意してください。別のタイムゾーン定義を使用するには、キックスタートファイルを使用する必要があります。

J.3.16. user

キックスタートコマンドの **user** は任意です。システムに新しいユーザーを作成します。

Syntax

```
user --name=username [OPTIONS]
```

必須オプション

- **--name=** - ユーザー名を入力します。このオプションは必須です。

任意のオプション

- **--gecos=** - ユーザーの GECOS 情報を指定します。これは、コンマ区切りのさまざまなシステム固有フィールドの文字列です。ユーザーのフルネームやオフィス番号などを指定するのに使用されます。詳細は、**passwd(5)** の man ページを参照してください。

- **--groups=** - デフォルトグループの他にもユーザーが所属すべきグループ名のコンマ区切りのリストです。このグループは、ユーザーアカウントの作成前に存在する必要があります。詳細は、**group** コマンドを参照してください。

- **--homedir=** - ユーザーのホームディレクトリーです。これが設定されない場合は、**/home/username** がデフォルトになります。

- **--lock** - このオプションを指定すると、このアカウントはデフォルトでロックされます。つまり、ユーザーはコンソールからログインできなくなります。また、グラフィカルおよびテキストベースの手動インストールで、**ユーザーの作成** ウィンドウが無効になります。

- **--password=** - 新規のユーザーパスワードです。指定しないと、そのアカウントはデフォルトでロックされます。

- **--iscrypted** - このオプションを追加すると、パスワード引数はすでに暗号化済みと仮定されます。**--plaintext** と相互排他的になります。暗号化したパスワードを作成する場合は python を使用します。

  ```
  $ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
  (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  上記の例では、ランダムの salt を使用して、パスワードの sha512 暗号と互換性があるハッシュが生成されます。

- **--plaintext** - このオプションを使用すると、パスワードの引数はプレーンテキストであると仮定されます。**--iscrypted** と相互排他的になります。

- **--shell=** - ユーザーのログインシェルです。指定しないと、システムのデフォルトが使用されます。

- **--uid=** - ユーザーの UID (User ID) です。指定しないと、次に利用可能なシステム以外の UID をデフォルトにします。

- **--gid=** - ユーザーのグループで使用される GID (Group ID) です。指定しないと、次に利用可能なシステム以外のグループ ID をデフォルトにします。

注記

- **--uid** と **--gid** のオプションを使用して、通常のユーザーとそのデフォルトグループに **1000** ではなく **5000** から始まる範囲の ID を設定することを検討してください。これは、システムユーザーおよびグループに予約してある **0**-**999** の範囲が今後広がり、通常のユーザーの ID と重複する可能性があるためです。
  選択した UID および GID の範囲が、ユーザーの作成時に自動的に適用されるように、インストール後に UID および GID の最小制限を変更する場合は、**Configuring basic system settings** Setting default permissions for new files using umask を参照してください。

- ファイルおよびディレクトリーはさまざまなパーミッションで作成され、パーミッションは、ファイルまたはディレクトリーを作成するアプリケーションによる影響を受けます。たとえ

ば、**mkdir** コマンドは、すべてのパーミッションを有効にしてディレクトリーを作成します。ただし、**user file-creation mask** 設定で指定されたように、アプリケーションは、新規に作成したファイルに特定パーミッションを付与しません。
user file-creation mask は、**umask** コマンドで管理できます。新規ユーザー向けの **user file-creation mask** のデフォルト設定は、インストールシステム上の **/etc/login.defs** 設定ファイルの **UMASK** 変数で定義されます。これを設定しない場合は、デフォルト値 **022** を使用します。デフォルト値を使用し、アプリケーションがファイルを作成した場合は、ファイルの所有者以外のユーザーに書き込みパーミッションが付与されません。ただし、これは他の設定やスクリプトで無効にできます。

詳細は、**Configuring basic system settings** の Setting default permissions for new files using umask のセクションを参照してください。

J.3.17. xconfig

キックスタートコマンドの **xconfig** は任意です。X Window System を設定します。

構文

```
xconfig [--startxonboot]
```

オプション

- **--startxonboot** - インストール済みシステムでグラフィカルログインを使用します。

注記

- Red Hat Enterprise Linux 8 には KDE デスクトップ環境が含まれていないため、アップストリームに記載されている **--defaultdesktop=** を使用しないでください。

J.4. ネットワーク設定用キックスタートコマンド

このリストのキックスタートコマンドにより、システムにネットワークを設定できます。

J.4.1. ネットワーク (任意)

オプションの **network** キックスタートコマンドを使用して、ターゲットシステムのネットワーク情報を設定し、インストール環境でネットワークデバイスをアクティブにします。最初の **network** コマンドで指定しているデバイスが自動的にアクティベートされます。**--activate** オプションを使用して、デバイスを明示的にアクティブ化するように要求することもできます。

Syntax

```
network OPTIONS
```

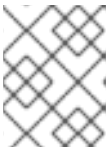
オプション

- **--activate** - インストール環境でこのデバイスをアクティブにします。
すでにアクティブ化しているデバイスに対して **--activate** オプションを使用すると (たとえば、キックスタートファイルを取得できるよう起動オプションで設定したインターフェイスなど)、キックスタートファイルで指定している詳細を使用するようデバイスが再アクティブ化されます。

デバイスにデフォルトのルートを使用しないようにするには、**--nodefroute** オプションを使用します。

- **--no-activate** - インストール環境でこのデバイスをアクティブにしません。
デフォルトでは、**--activate** オプションにかかわらず、Anaconda はキックスタートファイルの 1 番目のネットワークデバイスをアクティブにします。**--no-activate** オプションを使用して、デフォルトの設定を無効にできます。

- **--bootproto=** - **dhcp**、**bootp**、**ibft** または **static** のいずれかを指定します。**dhcp** がデフォルトのオプションになります。**dhcp** と **bootp** は同じように処理されます。デバイスの **ipv4** 設定を無効にするには、**--noipv4** オプションを使用します。



注記

このオプションは、デバイスの ipv4 設定を行います。ipv6 の設定には、**--ipv6** オプションおよび **--ipv6gateway** オプションを使用します。

DHCP メソッドでは、DHCP サーバーシステムを使用してネットワーク設定を取得します。BOOTP メソッドも同様で、BOOTP サーバーがネットワーク設定を提供する必要があります。システムが DHCP を使用するようにする場合は、以下のように指定します。

```
network --bootproto=dhcp
```

BOOTP を使用してネットワーク設定を取得する場合は、キックスタートファイルで次の行を使用します。

```
network --bootproto=bootp
```

iBFT で指定されている設定を使用する場合は、以下のようにします。

```
network --bootproto=ibft
```

static メソッドの場合は、キックスタートファイルに IP アドレスおよびネットマスクを指定する必要があります。これらの情報は静的となるため、インストール時およびインストール後にも使用されます。

静的なネットワーク設定情報はすべて **一行で** 指定する必要があります。コマンドラインのようにバックスラッシュ (\) を使用して行を折り返すことはできません。

```
network --bootproto=static --ip=10.0.2.15 --netmask=255.255.255.0 --gateway=10.0.2.254 --
nameserver=10.0.2.1
```

ネームサーバーは同時に複数設定することもできます。以下のように、1 つの **--nameserver=** オプションに対して、ネームサーバーの IP アドレスをコンマ区切りで指定します。

```
network --bootproto=static --ip=10.0.2.15 --netmask=255.255.255.0 --gateway=10.0.2.254 --
nameserver=192.168.2.1,192.168.3.1
```

- **--device=** - **network** コマンドで設定する (また最終的に Anaconda でアクティベートさせる) デバイスを指定します。
最初 に使用される **network** コマンドに **--device=** オプションがない場合は、Anaconda の起動オプション **inst.ks.device=** の値が使用されます (使用可能な場合)。ただし、この動作は廃止が予定されているため注意してください。ほとんどの場合において、すべての **network** コマンド

には必ず **--device=** オプションを指定してください。

同じキックスタートファイルに記載される 2 番目以降の **network** コマンドの動作は、**--device=** オプションを指定しないと詳細が不明になります。1 番目以降の **network** コマンドに、このオプションを指定していることを確認してください。

起動するデバイスは、以下のいずれかの方法で指定します。

- インターフェイスのデバイス名を使用して指定する (**em1** など)
- インターフェイスの MAC アドレスを使用して指定する (**01:23:45:67:89:ab** など)
- **link** キーワードを使用する (リンクが **up** 状態になっている 1 番目のインターフェイス)。
- キーワード **bootif** を使用する。これは、pxelinux が **BOOTIF** 変数に設定した MAC アドレスを使用します。pxelinux に **BOOTIF** 変数を設定する場合は、**pxelinux.cfg** ファイルに **IPAPPEND 2** を設定します。

以下に例を示します。

```
network --bootproto=dhcp --device=em1
```

- **--ip=** - デバイスの IP アドレスを指定します。
- **--ipv6=** - デバイスの IPv6 アドレスを **address**[/**prefix length**] の形式で指定します (例: **3ffe:ffff:0:1::1/128**)。**prefix** を省略すると、**64** が使用されます。**auto** を使用すると自動設定に、**dhcp** を使用すると DHCPv6 限定の設定 (ルーター広告なし) となります。
- **--gateway=** - 単一 IPv4 アドレスのデフォルトゲートウェイを指定します。
- **--ipv6gateway=** - 単一 IPv6 アドレスのデフォルトゲートウェイを指定します。
- **--nodefroute** - インターフェイスがデフォルトのルートとして設定されないようにします。iSCSI ターゲット用に別のサブネットにある NIC など、**--activate=** オプションで追加デバイスをアクティブにする場合は、このオプションを使用します。
- **--nameserver=** - IP アドレスに DNS ネームサーバーを指定します。複数のネームサーバーを指定する場合は、1 つの オプションに対して、IP アドレスをコンマ区切りで指定します。
- **--netmask=** - インストール後のシステムのネットワークマスクを指定します。
- **--hostname=** - ターゲットシステムのホスト名を設定するために使用されます。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、このマシン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てください。
  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。

ターゲットシステムのホスト名のみを設定する場合は、**network** コマンドで **--hostname** オプションを使用し、他のオプションは含めないでください。

ホスト名の設定時に追加オプションを指定すると、**network** コマンドは指定したオプションを使用してデバイスを設定します。**--device** オプションを使用して設定するデバイスを指定しないと、デフォルトの **--device link** の値が使用されます。また、**--bootproto** オプションを使用してプロトコルを指定しないと、デバイスはデフォルトで DHCP を使用するように設定されます。

- **--ethtool=** - ethtool プログラムに渡されるネットワークデバイスの低レベルの追加設定を指定します。
- **--onboot=** - システムの起動時にデバイスを有効にするかどうかを指定します。
- **--dhcpclass=** - DHCP クラスを指定します。
- **--mtu=** - デバイスの MTU を指定します。
- **--noipv4** - このデバイスで IPv4 を無効にします。
- **--noipv6** - このデバイスで IPv6 を無効にします。
- **--bondslaves=** - このオプションを使用すると、**--bondslaves=** オプションで定義されたセカンダリーデバイスを使用して、**--device=** オプションで指定したボンディングデバイスが作成されます。以下に例を示します。

```
network --device=bond0 --bondslaves=em1,em2
```

上記のコマンドは、インターフェイスの **em1** および **em2** をセカンダリーデバイスとして使用し、ボンドデバイス **bond0** を作成します。

- オプションの **--bondopts=** - **--bondslaves=** および **--device=** を使用して指定されるボンドインターフェイス用のオプションパラメーターのリストです。このリスト内のオプションは必ずコンマ (,) またはセミコロン (;) で区切ってください。オプション自体にコンマが含まれている場合はセミコロンを使用してください。以下に例を示します。

```
network --bondopts=mode=active-backup,balance-rr;primary=eth1
```

重要

--bondopts=mode= パラメーターは、**balance-rr** や **broadcast** などのフルモード名にしか対応しません。**0** や **3** などの数値による表記には対応していません。利用可能なモードおよびサポートされるモードの一覧は、Configuring and Managing Networking Guide を参照してください。

- **--vlanid=** - **--device=** で指定したデバイスを親として作成する仮想デバイスの仮想 LAN (VLAN) の ID 番号 (802.1q タグ) を指定します。たとえば、**network --device=em1 --vlanid=171** を使用すると仮想 LAN デバイスの **em1.171** が作成されます。

- **--interfacename=** - 仮想 LAN デバイスのカスタムのインターフェイス名を指定します。**--vlanid=** オプションで生成されるデフォルト名が望ましくない場合に使用してください。**--vlanid=** と併用する必要があります。以下に例を示します。

```
network --device=em1 --vlanid=171 --interfacename=vlan171
```

上記のコマンドにより、**em1** デバイスに ID **171** の仮想 LAN インターフェイス **vlan171** が作成されます。

インターフェイスには任意の名前 (**my-vlan** など) を付けることができますが、場合によっては次の命名規則に従う必要があります。

  - 名前にドット (**.**) を含める場合は、**NAME.ID** の形式にする必要があります。NAME は任意の名前で構いませんが **ID** は VLAN ID にする必要があります。たとえば、**em1.171**、**my-vlan.171** などにします。

  - **vlan** で開始する名前を付ける場合は、**vlanID** の形式にする必要があります。たとえば、**vlan171** などにします。

- **--teamslaves=** - このオプションで指定したセカンダリーデバイスを使用して、**--device=** オプションで指定したチームデバイスが作成されます。セカンダリーデバイスはコンマで区切ります。各セカンダリーデバイスの後ろにその設定を指定できます。\ 記号でエスケープした二重引用符で、一重引用符の JSON 文字列を囲っている部分が実際の設定になります。以下に例を示します。

```
network --teamslaves="p3p1'{\"prio\": -10, \"sticky\": true}',p3p2'{\"prio\": 100}'"
```

--teamconfig= オプションも参照してください。

- **--teamconfig=** - チームデバイスの設定を二重引用符で囲って指定します。これは、二重引用符と \ 記号でエスケープした JSON 文字列になります。デバイス名は、**--device=** オプションで指定し、セカンダリーデバイスとその設定は、**--teamslaves=** オプションで指定します。以下に例を示します。

```
network --device team0 --activate --bootproto static --ip=10.34.102.222 --
netmask=255.255.255.0 --gateway=10.34.102.254 --nameserver=10.34.39.2 --
teamslaves="p3p1'{\"prio\": -10, \"sticky\": true}',p3p2'{\"prio\": 100}'" --teamconfig="
{\"runner\": {\"name\": \"activebackup\"}}"
```

- **--bridgeslaves=** - このオプションを使用すると、**--device=** オプションで指定したデバイス名でネットワークブリッジが作成され、このネットワークブリッジに、**--bridgeslaves=** オプションで指定したデバイスが追加されます。以下に例を示します。

```
network --device=bridge0 --bridgeslaves=em1
```

- **--bridgeopts=** - オプションでブリッジしたインターフェイス用パラメーターのリストをコンマで区切って指定します。使用できる値は **stp**、**priority**、**forward-delay**、**hello-time**、**max-age**、**ageing-time** などです。これらのパラメーターの詳細は、**nm-settings(5)** man ページまたは ネットワーク設定仕様 にある **ブリッジ設定** の表を参照してください。
ネットワークブリッジの一般情報は、Configuring and managing networking も参照してください。

- **--bindto=mac** - インストールされたシステムのデバイス設定ファイルをインターフェイス名 (**DEVICE**) へのデフォルトのバインドではなく、デバイスの MAC アドレス (**HWADDR**) にバインドします。このオプションは **--device=** オプションとは独立している点に注意してくださ

い。同じ **network** コマンドでデバイス名、**リンク**、または **bootif** が指定されていても、**--bindto=mac** が適用されます。

注記

- 命名方法の変更により、Red Hat Enterprise Linux では **eth0** などの **ethN** デバイス名を使用できなくなりました。デバイスの命名スキームの詳細は、アップストリームドキュメント Predictable Network Interface Names を参照してください。
- キックスタートのオプションまたは起動オプションを使用して、ネットワークにあるインストールリポジトリーを指定したものの、インストール開始時にネットワークが利用できない状態になっている場合は、**インストール概要** ウィンドウが表示される前に、ネットワーク接続の設定を求める **ネットワークの設定** ウィンドウが表示されます。詳細については、**RHEL 8 の標準インストールの実行** ドキュメントの ネットワークとホスト名のオプションの設定 セクションを参照してください。

J.4.2. realm

キックスタートコマンドの **realm** は任意です。Active Directory や IPA ドメインを参加させます。このコマンドの詳細は、man ページ **realm(8)** の **join** のセクションを参照してください。

Syntax

```
realm join [OPTIONS] domain
```

必須オプション

- **domain** - 参加するドメイン。

オプション

- **--computer-ou=OU=** - コンピューターアカウントを作成するために、組織単位の識別名を指定します。識別名の形式は、クライアントソフトウェアおよびメンバーシップのソフトウェアにより異なります。通常、識別名のルート DSE の部分は省略できます。
- **--no-password** - パスワードの入力なしで自動的に参加します。
- **--one-time-password=** - ワンタイムパスワードを使用して参加します。すべてのレルムで使用できるとは限りません。
- **--client-software=** - ここで指定したクライアントソフトウェアを実行できるレルムにしか参加しません。使用できる値は **sssd** や **winbind** などになります。すべてのレルムがすべての値に対応しているとは限りません。デフォルトでは、クライアントソフトウェアは自動的に選択されます。
- **--server-software=** - ここで指定したサーバーソフトウェアを実行できるレルムにしか参加しません。使用できる値は **active-directory** や **freeipa** などになります。
- **--membership-software=** - レルムに参加する際に、このソフトウェアを使用します。使用できる値は **samba** や **adcli** などになります。すべてのレルムがすべての値に対応しているとは限りません。デフォルトでは、メンバーシップソフトウェアは自動的に選択されます。

J.5. ストレージを処理するキックスタートコマンド

本セクションのキックスタートコマンドは、デバイス、ディスク、パーティション、LVM、ファイルシステムなど、ストレージの設定を行います。

J.5.1. device (非推奨)

キックスタートコマンドの **device** は任意です。追加のカーネルモジュールを読み込むのに使用します。

ほとんどの PCI システムでは、イーサネットカードや SCSI カードが自動検出されます。ただし、旧式のシステムや一部の PCI では、適切なデバイスを検出できるようキックスタートにヒントを追加する必要があります。追加モジュールをインストールするようにインストールプログラムに指示する **device** コマンドは、以下の形式を使用します。

Syntax

```
device moduleName --opts=options
```

オプション

- moduleName - インストールが必要なカーネルモジュール名に置き換えます。
- **--opts=**: カーネルモジュールに渡すオプションです。以下に例を示します。

  ```
  device --opts="aic152x=0x340 io=11"
  ```

J.5.2. autopart

キックスタートコマンドの **autopart** は任意です。自動的にパーティションを作成します。

自動的に作成されるパーティション - ルート (/) パーティション (1 GB 以上)、**swap** パーティション、アーキテクチャーに応じた **/boot** パーティション。容量が十分にあるドライブの場合 (50 GiB 以上)、**/home** パーティションも作成されます。

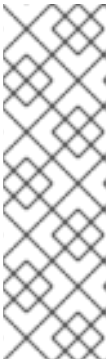
Syntax

```
autopart OPTIONS
```

オプション

- **--type=** - 事前定義済み自動パーティション設定スキームの中から、使用するスキームを選択します。次の値を取ります。
  - **lvm**:LVM パーティション設定スキーム
  - **plain**:LVM がない普通のパーティション
  - **thinp**:LVM シンプロビジョニングのパーティション設定スキーム
- **--fstype=** - 利用可能なファイルシステムのタイプを選択します。利用可能な値は、**ext2**、**ext3**、**ext4**、**xfs**、および **vfat** です。デフォルトのファイルシステムは **xfs** です。
- **--nohome** - **/home** パーティションの自動作成を無効にします。

- **--nolvm** - 自動パーティション設定に LVM を使用しません。このオプションは **--type=plain** と同じです。
- **--noboot** - **/boot** パーティションを作成しません。
- **--noswap** - swap パーティションを作成しません。
- **--encrypted** - Linux Unified Key Setup (LUKS) ですべてのパーティションを暗号化します。手動によるグラフィカルインストールを行った際の初期パーティション設定ウィンドウで表示される **Encrypt partitions (パーティションの暗号化)** のチェックボックスと同じです。



注記

1 つまたは複数のパーティションを暗号化する際には、安全な暗号化を行うため、Anaconda が 256 ビットのエントロピーを収集しようとします。エントロピーの収集には時間がかかる場合があります。十分なエントロピーが収集されたかどうかにかかわらず、このプロセスは最大 10 分後に終了します。

プロセスは、インストールシステムと対話することにより高速化できます (キーボードで入力またはマウスの移動)。仮想マシンにインストールしている場合は、**virtio-rng** デバイス (仮想乱数ジェネレーター) をゲストに登録できます。

- **--luks-version=LUKS_VERSION** - ファイルシステムの暗号化に使用する LUKS 形式のバージョンを指定します。**--encrypted** と併用しないと有効ではありません。
- **--passphrase=** - 暗号化した全デバイスに、デフォルトのシステムワイドパスフレーズを指定します。
- **--escrowcert=URL_of_X.509_certificate** - 暗号化した全ボリュームのデータ暗号化の鍵を **/root** 配下にファイル形式で格納します。**URL_of_X.509_certificate** で指定した URL の X.509 証明書を使用して暗号化します。鍵は暗号化したボリュームごとに別のファイルとして格納されます。**--encrypted** と併用しないと有効ではありません。
- **--backuppassphrase** - 暗号化されたボリュームにそれぞれランダムに生成されたパスフレーズを追加します。パスフレーズは、**/root** 配下に別々のファイルで格納され、**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと有効ではありません。
- **--cipher=** - Anaconda のデフォルトである **aes-xts-plain64** では十分ではない場合に使用する暗号化の種類を指定します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。使用できる暗号化の種類は セキュリティーの強化 に記載されていますが、Red Hat では、**aes-xts-plain64** または **aes-cbc-essiv:sha256** のいずれかの使用を強く推奨しています。
- **--pbkdf=PBKDF** - LUKS 鍵スロット用の PBKDF (Password-Based Key Derivation Function) アルゴリズムを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。
- **--pbkdf-memory=PBKDF_MEMORY** - PBKDF のメモリーコストを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。
- **--pbkdf-time=PBKDF_TIME** - PBKDF パスフレーズ処理にかかるミリ秒数を設定します。**cryptsetup(8)** の man ページの **--iter-time** も併せて参照してください。このオプションは、**--encrypted** が指定される場合に限り有効になり、**--pbkdf-iterations** と相互に排他的になります。

- **--pbkdf-iterations=PBKDF_ITERATIONS** - 反復の数を直接設定し、PBKDF ベンチマークを回避します。**cryptsetup(8)** の man ページの **--pbkdf-force-iterations** も併せて参照してください。このオプションは、**--encrypted** が指定されている場合に限り有効になり、**--pbkdf-time** と相互に排他的になります。

注記

- **autopart** オプションは、同じキックスタートファイル内で は、**part/partition**、**raid**、**logvol**、**volgroup** などのオプションとは併用できません。
- **autopart** コマンドは必須ではありませんが、キックスタートスクリプトに **part** コマンドまたは **mount** コマンドがない場合は、このコマンドを組み込む必要があります。
- CMS タイプの 1 つの FBA DASD にインストールする場合は、**autopart --nohome** のキックスタートオプションを使用することが推奨されます。これを使用すると、インストールプログラムが別の **/home** パーティションを作成しません。その後、インストールは成功します。
- LUKS パスフレーズが分からなくなると、暗号化されたパーティションと、その上にあるデータには完全にアクセスできなくなります。分からなくなったパスフレーズを復元する方法はありません。ただし、**--escrowcert** を使用して暗号パスフレーズを保存し、**--backuppassphrase** オプションを使用してバックアップの暗号化パスフレーズを作成できます。
- **autopart**、**autopart --type=lvm**、または **autopart=thinp** を使用する場合は、ディスクのセクターサイズに一貫性があることを確認してください。

J.5.3. bootloader (必須)

キックスタートコマンド の **bootloader** が必要です。ブートローダーをインストールする方法を指定します。

Syntax

```
bootloader [OPTIONS]
```

オプション

- **--append=** - 追加のカーネルパラメーターを指定します。複数のパラメーターを指定する場合は空白で区切ります。以下に例を示します。

  ```
  bootloader --location=mbr --append="hdd=ide-scsi ide=nodma"
  ```

  plymouth パッケージをインストールすると、**rhgb** パラメーターおよび **quiet** パラメーターをここで指定しなくても、もしくは **--append=** コマンドを使用しなくても、自動的に追加されます。この動作を無効にするには、**plymouth** のインストールを明示的に拒否します。

  ```
  %packages
  -plymouth
  %end
  ```

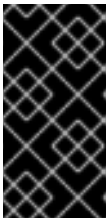
  このオプションは、Meltdown および Spectre に起因する脆弱性の問題を軽減するために実装されたメカニズムを無効にする場合に便利です。投機的実行を悪用するもので、今日のほとんどのプロセッサーで確認されています (CVE-2017-5754、CVE-2017-5753、および CVE-2017-5715)。場合によっては、これらのメカニズムは不要で、有効にしてもセキュリティーは向上せ

ずパフォーマンスが低下する可能性があります。これらのメカニズムを無効にするには、無効にするオプションをキックスタートファイルに追加します (AMD64/Intel 64 システムの例: **bootloader --append="nopti noibrs noibpb"**)。

警告

脆弱性の問題を軽減するメカニズムを無効にする場合は、システムが攻撃の危険にさらされていないことを確認する必要があります。Meltdown および Spectre に起因する脆弱性については、Red Hat vulnerability response article の記事を参照してください。

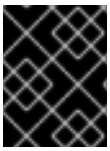
- **--boot-drive=** - ブートローダーの書き込み先のドライブを指定します。つまり、コンピューターが起動するドライブです。ブートドライブにマルチパスデバイスを使用する場合は、disk/by-id/dm-uuid-mpath-WWID 名を使用してデバイスを指定します。



重要

現在、**zipl** ブートローダーを使用する 64 ビットの IBM Z システムの Red Hat Enterprise Linux インストールでは、**--boot-drive=** オプションが無視されます。**zipl** をインストールすると、それ自体に起動ドライブがあると判断されます。

- **--leavebootorder** - インストールプログラムが、ブートローダーのインストール済みシステムリストの最上位に Red Hat Enterprise Linux 8 を追加し、その順番と既存の全エントリーを保持します。



重要

このオプションは、Power システムのみに適用されます。UEFI システムにはこのオプションを使用しないでください。

- **--driveorder=** - BIOS の起動順序で最初のドライブを指定します。以下に例を示します。

```
bootloader --driveorder=sda,hda
```

- **--location=** - ブートレコードの書き込み先を指定します。使用できる値は以下のとおりです。
  - **mbr** - デフォルトのオプションです。ドライブが使用しているのが Master Boot Record (MBR) スキームか GUID Partition Table (GPT) スキームかによって、動作が異なります。GPT フォーマット済みディスクの場合は、ブートローダーのステージ 1.5 が BIOS 起動パーティションにインストールされます。

    MBR フォーマット済みディスクの場合は、MBR と 1 番目のパーティションの間にある空白領域にステージ 1.5 がインストールされます。
  - **partition** - カーネルを置くパーティションの 1 番目のセクターに、ブートローダーをインストールします。
  - **none** - ブートローダーをインストールしません。

ほとんどの場合、このオプションは指定する必要がありません。

- **--nombr** - MBR にブートローダーをインストールしません。
- **--password=** - GRUB2 を使用する場合は、このオプションで指定したパスワードを、ブートローダーのパスワードとして設定します。任意のカーネルオプションが渡される可能性のある GRUB2 シェルへのアクセスを限定する場合に使用してください。
  パスワードを指定すると、GRUB2 ではユーザー名の入力も求められます。ユーザー名は常に **root** です。
- **--iscrypted** - **--password=** オプションを使用してブートローダーのパスワードを指定すると、通常、キックスタートファイルにプレーンテキスト形式で保存されます。このパスワードを暗号化する場合に、このオプションを使用して暗号化パスワードを生成します。
  暗号化したパスワードを生成するには、**grub2-mkpasswd-pbkdf2** コマンドを使用し、使用するパスワードを入力し、コマンドからの出力 (**grub.pbkdf2** で始まるハッシュ) をキックスタートファイルにコピーします。暗号化したパスワードがあるキックスタートエントリーの **bootloader** の例を以下に示します。

```
bootloader --iscrypted --
password=grub.pbkdf2.sha512.10000.5520C6C9832F3AC3D149AC0B24BE69E2D4FB0DBE
EDBD29CA1D30A044DE2645C4C7A291E585D4DC43F8A4D82479F8B95CA4BA4381F8550
510B75E8E0BB2938990.C688B6F0EF935701FF9BD1A8EC7FE5BD2333799C98F28420C5
CC8F1A2A233DE22C83705BB614EA17F3FDFDF4AC2161CEA3384E56EB38A2E39102F53
34C47405E
```

- **--timeout=** - ブートローダーがデフォルトオプションで起動するまでの待ち時間を指定します (秒単位)。
- **--default=** - ブートローダー設定内のデフォルトのブートイメージを設定します。
- **--extlinux** - GRUB2 の代わりに extlinux ブートローダーを使用します。このオプションが動作するには、extlinux が対応しているシステムのみです。
- **--disabled** - このオプションは、**--location=none** のより強力なバージョンになります。**--location=none** は単にブートローダーのインストールを無効にしますが、**--disabled** だとブートローダーのインストールを無効にするほか、ブートローダーを含むパッケージのインストールを無効にするため、領域が節約できます。

注記

- Red Hat は、全マシンにブートローダーのパスワードを設定することを強く推奨します。ブートローダーが保護されていないと、攻撃者によりシステムの起動オプションが修正され、システムへの不正アクセスが許可されてしまう可能性があります。
- AMD64、Intel 64、および 64 ビット ARM のシステムにブートローダーをインストールするのに、特殊なパーティションが必要になります。このパーティションの種類とサイズは、ブートローダーをインストールしているディスクが、MBR (Master Boot Record) または GPT (GUID Partition Table) スキーマを使用しているかどうかにより異なります。詳細は、**標準的な RHEL 8 インストールの実行** の ブートローダーの設定 セクションを参照してください。
- **sdX** (または **/dev/sdX**) 形式でのデバイス名がシステムの再起動後に維持される保証がないため、一部のキックスタートコマンドを複雑にします。コマンドがデバイスノード名を呼び出す際には、代わりに **/dev/disk** からのアイテムを使用することができます。以下に例を示します。

```
part / --fstype=xfs --onpart=sda1
```

上記のコマンドの代わりに、以下のいずれかを使用します。

```
part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
```

```
part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
```

上記の手順により、コマンドは常に同じストレージデバイスをターゲットとします。これは特に大型のストレージ環境で便利なものです。ストレージデバイスを連続して参照するさまざまな方法についての詳細は、**Managing storage devices** の Overview of persistent naming attributes を参照してください。

- **--upgrade** オプションは、Red Hat Enterprise Linux 8 で非推奨となりました。

J.5.4. zipl

キックスタートコマンドの **zipl** は任意です。これは、64 ビットの IBM Z の ZIPL 設定を指定します。

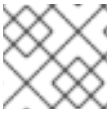
オプション

- **--secure-boot** - インストールシステムで対応しているかどうかを、セキュアな起動を有効にします。

注記

インストールシステムは、IBM z14 以降のシステムにインストールする場合、IBM z14 またはそれ以前のモデルからは起動できません。

- **--force-secure-boot** - セキュアな起動を無条件で有効にします。



注記

IBM z14 以前のモデルでは、インストールに対応していません。

- **--no-secure-boot** - セキュアな起動を無効にします。

注記

Secure Boot は、IBM z14 とそれ以前のモデルでは対応していません。IBM z14 以前のモデルでインストール済みシステムを起動する場合は、**--no-secure-boot** を使用します。

J.5.5. clearpart

キックスタートコマンドの **clearpart** は任意です。新しいパーティションを作成する前に、システムからパーティションを削除します。デフォルトでは、パーティションは削除されません。

構文

```
clearpart OPTIONS
```

オプション

- **--all** - システムにあるすべてのパーティションを消去します。
このオプションを使用すると、接続しているネットワークストレージなど、インストールプログラムでアクセスできるディスクがすべて消去されます。使用する場合は注意が必要です。

  clearpart に **--drives=** オプションを使用して消去するドライブのみを指定する、ネットワークストレージは後で接続する (キックスタートファイルの **%post** セクションを利用するなど)、ネットワークストレージのアクセスに使用されるカーネルモジュールを拒否リストに記載するなどの手段を取ると、保持したいストレージが消去されるのを防ぐことができます。

- **--drives=** - ドライブを指定してパーティションを消去します。次の例では、プライマリー IDE コントローラーの 1 番目と 2 番目のドライブにあるパーティションをすべて消去することになります。

  ```
  clearpart --drives=hda,hdb --all
  ```

  マルチパスのデバイスを消去する場合は、**disk/by-id/scsi-WWID** の形式を使用します。**WWID** はデバイスの World-Wide Identifier になります。WWID **58095BEC5510947BE8C0360F604351918** のディスクを消去する場合は以下を使用します。

  ```
  clearpart --drives=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
  ```

  マルチパスのデバイスを消去する場合はこの形式が適しています。ただし、エラーが発生する場合は、そのマルチパスデバイスが論理ボリューム管理 (LVM) を使用していなければ、**disk/by-id/dm-uuid-mpath-WWID** の形式を使用して消去することもできます。**WWID** はデバイスの World-Wide Identifier です。WWID **2416CD96995134CA5D787F00A5AA11017** のディスクを消去する場合は以下を使用します。

  ```
  clearpart --drives=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
  ```

  mpatha などのデバイス名でマルチパスデバイスを指定しないでください。このようなデバイス名は、特定のディスクに固有の名前ではありません。インストール中の **/dev/mpatha** という名前のディスクは必ずしも期待したディスクを指すとは限りません。したがって、**clearpart** コマンドを使用する際は、間違ったディスクが対象となる可能性があります。

- **--initlabel** - フォーマット用に指定されたそれぞれのアーキテクチャーで全ディスクに対してデフォルトのディスクラベルを作成して、ディスクを初期化します。たとえば、x86 の場合は msdos になります。**--initlabel** ではすべてのディスクが表示されてしまうため、フォーマット対象のドライブだけを接続することが重要です。**--initlabel** が使用されていない場合でも、**clearpart** によってクリアされたディスクにはラベルが作成されます。

  ```
  clearpart --initlabel --drives=names_of_disks
  ```

  以下に例を示します。

  ```
  clearpart --initlabel --drives=dasda,dasdb,dasdc
  ```

- **--list=** - 消去するパーティションを指定します。このオプションを使用すると、**--all** および **--linux** のオプションが上書きされます。異なるドライブ間で使用できます。以下に例を示します。

  ```
  clearpart --list=sda2,sda3,sdb1
  ```

- **--disklabel=LABEL** - 使用するデフォルトのディスクラベルを設定します。そのプラットフォームでサポートされるディスクラベルのみが設定できます。たとえば、64 ビットの Intel

アーキテクチャーおよび AMD アーキテクチャーでは、**msdos** ディスクラベルおよび **gpt** ディスクラベルが使用できますが、**dasd** は使用できません。

- **--linux** - すべての Linux パーティションを消去します。
- **--none** (デフォルト) - パーティションを消去しません。
- **--cdl** - LDL DASD を CDL 形式に再フォーマットします。

注意

- **sdX** (または **/dev/sdX**) 形式でのデバイス名がシステムの再起動後に維持される保証がないため、一部のキックスタートコマンドを複雑にします。コマンドがデバイスノード名を呼び出す際には、代わりに **/dev/disk** からのアイテムを使用することができます。以下に例を示します。

```
part / --fstype=xfs --onpart=sda1
```

以下のいずれかのようなエントリーを使用します。

```
part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
```

```
part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
```

上記の手順により、コマンドは常に同じストレージデバイスをターゲットとします。これは特に大型のストレージ環境で便利なものです。ストレージデバイスを連続して参照するさまざまな方法についての詳細は、**Managing storage devices** の Overview of persistent naming attributes を参照してください。

- **clearpart** コマンドを使用する場合は、論理パーティションには **part --onpart** コマンドは使用できません。

J.5.6. fcoe

キックスタートコマンドの **fcoe** は任意です。Enhanced Disk Drive Services (EDD) で検出されたデバイス以外で、自動的にアクティベートする FCoE デバイスを指定します。

Syntax

```
fcoe --nic=name [OPTIONS]
```

オプション

- **--nic=** (必須) - アクティベートするデバイス名です。
- **--dcb=** - データセンターブリッジ (DCB) の設定を確立します。
- **--autovlan** - VLAN を自動的に検出します。このオプションはデフォルトで有効になっています。

J.5.7. ignoredisk

キックスタートコマンドの **ignoredisk** は任意です。インストールプログラムが、指定したディスクを無視するようになります。

自動パーティション設定を使用して、特定のディスクを無視したい場合に便利なオプションです。たとえば、**ignoredisk** を使用せずに SAN クラスターに導入しようとすると、インストールプログラムが SAN へのパッシブパスを検出し、パーティションテーブルがないことを示すエラーが返されるため、キックスタートが失敗します。

Syntax

```
ignoredisk --drives=drive1,drive2,... | --only-use=drive
```

オプション

- **--drives=driveN,...** - driveN を、 **sda**、 **sdb** のいずれかに置き換えます。, **hda**,... などになります。
- **--only-use=driveN,...** - インストールプログラムで使用するディスクのリストを指定します。これ以外のディスクはすべて無視されます。たとえば、インストール中に **sda** ディスクを使用し、他はすべて無視する場合は以下のコマンドを使用します。

  ```
  ignoredisk --only-use=sda
  ```

  LVM を使用しないマルチパスのデバイスを指定する場合は、次のコマンドを実行します。

  ```
  ignoredisk --only-use=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
  ```

  LVM を使用するマルチパスのデバイスを指定する場合は、次のコマンドを実行します。

  ```
  ignoredisk --only-use==/dev/disk/by-id/dm-uuid-mpath-
  ```

  ```
  bootloader --location=mbr
  ```

--drives または **--only-use** のいずれかのみを指定する必要があります。

備考

- **--interactive** オプションは、Red Hat Enterprise Linux 8 で非推奨となりました。このオプションにより、高度なストレージ画面を手動で操作できます。
- 論理ボリューム管理 (LVM) を使用していないマルチパスデバイスを無視する場合は、**disk/by-id/dm-uuid-mpath-WWID** の形式を使用します。**WWID** はデバイスの World-Wide Identifier です。たとえば、WWID **2416CD96995134CA5D787F00A5AA11017** のディスクを無視する場合は以下を使用します。

  ```
  ignoredisk --drives=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
  ```

- **mpatha** などのデバイス名でマルチパスデバイスを指定しないでください。このようなデバイス名は、特定のディスクに固有の名前ではありません。インストール中の **/dev/mpatha** という名前のディスクは必ずしも期待したディスクを指すとは限りません。したがって、**clearpart** コマンドを使用する際は、間違ったディスクが対象となる可能性があります。
- **sdX** (または **/dev/sdX**) 形式でのデバイス名がシステムの再起動後に維持される保証がないため、一部のキックスタートコマンドを複雑にします。コマンドがデバイスノード名を呼び出す際には、代わりに **/dev/disk** からのアイテムを使用することができます。以下に例を示します。

```
part / --fstype=xfs --onpart=sda1
```

上記のコマンドの代わりに、以下のいずれかを使用します。

```
part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
```

```
part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
```

上記の手順により、コマンドは常に同じストレージデバイスをターゲットとします。これは特に大型のストレージ環境で便利なものです。ストレージデバイスを連続して参照するさまざまな方法についての詳細は、**Managing storage devices** の Overview of persistent naming attributes を参照してください。

J.5.8. iscsi

キックスタートコマンドの **iscsi** は任意です。インストール時に接続する追加の iSCSI ストレージを指定します。

Syntax

```
iscsi --ipaddr=address [OPTIONS]
```

必須オプション

- **--ipaddr=** (必須) - 接続先ターゲットの IP アドレスを指定します。

任意のオプション

- **--port=** (必須) - ポート番号を指定します。存在しない場合は、**--port=3260** がデフォルトで自動的に使用されます。
- **--target=** - ターゲットの IQN (iSCSI 修飾名) を指定します。
- **--iface=** - ネットワーク層で確定されるデフォルトのネットワークインターフェイスではなく、特定のネットワークインターフェイスに接続をバインドします。これを一度使用したら、キックスタート内の **iscsi** コマンドのインスタンスではすべて指定する必要があります。
- **--user=** - ターゲットでの認証に必要なユーザー名を指定します。
- **--password=** - ターゲットに指定したユーザー名のパスワードを指定します。
- **--reverse-user=** - 逆 CHAP 認証を使用するターゲットのイニシエーターでの認証に必要なユーザー名を指定します。
- **--reverse-password=** - イニシエーターに指定したユーザー名のパスワードを指定します。

注記

- また、**iscsi** コマンドを使用する場合は、**iscsiname** コマンドで iSCSI ノードに名前を割り当てる必要があります。**iscsiname** コマンドは、キックスタートファイルで、**iscsi** コマンドより先に指定してください。
- iSCSI ストレージは、できる限り **iscsi** コマンドではなくシステムの BIOS またはファームウェア (Intel システムの場合は iBFT) 内で設定してください。BIOS またはファームウェア内で設定

されたディスクは Anaconda で自動的に検出されて使用されるため、キックスタートファイルで特に設定する必要がありません。

- **iscsi** コマンドを使用する必要がある場合は、インストールの開始時にネットワークがアクティブであること、**iscsi** コマンドが、キックスタートファイルで **clearpart** や **ignoredisk** などのコマンドによる iSCSI ディスクの参照よりも**前に**指定されていることを確認してください。

J.5.9. iscsiname

キックスタートコマンドの **iscsiname** は任意です。これは、**iscsi** コマンドが指定した iSCSI ノードに名前を割り当てます。

構文

```
iscsiname iqname
```

オプション

- **iqname** - iSCSI ノードに割り当てる名前。

注記

- キックスタートファイルで **iscsi** コマンドを使用する場合は、キックスタートファイルで **iscsiname** earlier を指定する必要があります。

J.5.10. logvol

キックスタートコマンドの **logvol** は任意です。論理ボリューム管理 (LVM) に論理ボリュームを作成します。

Syntax

```
logvol mntpoint --vgname=name --name=name [OPTIONS]
```

必須オプション

mntpoint

パーティションがマウントされているマウントポイント。次のいずれかの形式になります。

- **/path**
  / または /**home** など
- **swap**
  このパーティションは、swap 領域として使用されます。

  自動的に swap パーティションのサイズを確定させる場合は、**--recommended** オプションを使用します。

  ```
  swap --recommended
  ```

  自動的に swap パーティションサイズを確定し、ハイバネート用に追加領域も配分するには、**--hibernation** オプションを使用します。

```
swap --hibernation
```

割り当てられるサイズは、**--recommended** で割り当てられる swap 領域に加え、システムの RAM の容量が割り当てられるサイズになります。

AMD64、Intel 64、および 64 ビット ARM のシステムで、このコマンドが割り当てたスワップサイズは、推奨されるパーティション設定スキーム を参照してください。

--vgname=name

ボリュームグループの名前。

--name=name

論理ボリュームの名前。

任意のオプション

--noformat

既存の論理ボリュームを使用し、フォーマットは行いません。

--useexisting

既存の論理ボリュームを使用し、再フォーマットします。

--fstype=

論理ボリュームのファイルシステムのタイプを設定します。**xfs**、**ext2**、**ext3**、**ext4**、**swap**、および **vfat** が使用できる値になります。

--fsoptions=

ファイルシステムをマウントする場合に使用するオプションの文字列を自由形式で指定します。この文字列はインストール後の **/etc/fstab** ファイルにコピーされるため、引用符で囲んでください。

注記

EFI システムパーティション (**/boot/efi**) では、anaconda が値をハードコードし、ユーザー指定の **--fsoptions** 値を無視します。

--mkfsoptions=

このパーティションでファイルシステムを作成するプログラムに渡す追加のパラメーターを指定します。引数のリストでは処理が行われないため、mkfs プログラムに直接渡すことが可能な形式で提供する必要があります。つまり、複数のオプションはコンマ区切りにするか、二重引用符で囲む必要があります (ファイルシステムによって異なります)。

--fsprofile=

このパーティションでファイルシステムを作成するプログラムに渡すのに使用するタイプを指定します。ファイルシステムの作成時に使用されるさまざまなチューニングパラメーターは、この使用タイプにより定義されます。ファイルシステム側で使用タイプという概念に対応し、有効なタイプを指定する設定ファイルがないと、このオプションは正しく機能しません。**ext2**、**ext3**、および **ext4** の場合、この設定ファイルは **/etc/mke2fs.conf** になります。

--label=

論理ボリュームのラベルを設定します。

--grow

論理ボリュームを拡張して、利用可能なサイズ (存在する場合) を埋めるか、指定されている場合は最大サイズまで埋めます。このオプションを使用する必要があるのは、ディスクイメージに最小限のストレージ領域を事前に割り当てており、ボリュームを拡大して使用可能な領域を埋める場合の

みです。物理的な環境では、これは 1 回限りのアクションです。ただし、仮想環境では、仮想マシンが仮想ディスクにデータを書き込むとボリュームサイズが増加します。

--size=

論理ボリュームのサイズを MiB 単位で指定します。このオプションは **--percent=** オプションと併用することはできません。

--percent=

サイズを静的に指定した論理ボリュームを考慮に入れた後のボリュームグループにある空き領域を表すパーセンテージとして、論理ボリュームのサイズを指定します。このオプションは **--size=** オプションと併用することはできません。

重要

論理ボリュームの新規作成時には、**--size=** オプションで静的なサイズを指定するか、**--percent=** オプションで残りの空き領域をパーセンテージとして指定する必要があります。1 つの論理ボリュームで、両方のオプションを使用することはできません。

--maxsize=

論理ボリュームを grow に設定した場合の最大サイズを MiB 単位で指定します。**500** などの整数値を使用してください (単位は不要)。

--recommended

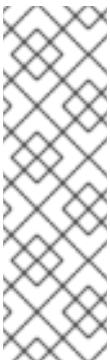
論理ボリュームを作成して、システムのハードウェアに基づいてそのボリュームのサイズを自動的に確定するために、このオプションを使用します。
AMD64、Intel 64、および 64 ビットの ARM システムで推奨されるスキームの詳細は、推奨されるパーティション設定スキーム を参照してください。

--resize

論理ボリュームのサイズを変更します。このオプションを使用する場合は、**--useexisting** と **--size** も指定する必要があります。

--encrypted

この論理ボリュームを、**--passphrase=** オプションで入力したパスフレーズを使用する LUKS (Linux Unified Key Setup) で暗号化します。このパスフレーズを指定しない場合、インストールプログラムは **autopart --passphrase** コマンドで設定されるデフォルトのシステムワイドパスフレーズを使用します。このデフォルトのパスフレーズも設定されていない場合は、インストールプロセスが中断されてパスフレーズの入力が求められます。



注記

1 つまたは複数のパーティションを暗号化する際には、安全な暗号化を行うため、Anaconda が 256 ビットのエントロピーを収集しようとします。エントロピーの収集には時間がかかる場合があります。十分なエントロピーが収集されたかどうかにかかわらず、このプロセスは最大 10 分後に終了します。

プロセスは、インストールシステムと対話することにより高速化できます (キーボードで入力またはマウスの移動)。仮想マシンにインストールしている場合は、**virtio-rng** デバイス (仮想乱数ジェネレーター) をゲストに登録できます。

--passphrase=

この論理ボリュームを暗号化する際に使用するパスフレーズを指定します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。

--cipher=

Anaconda のデフォルトである **aes-xts-plain64** では十分ではない場合に使用する暗号化の種類を指定します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。使用できる暗号化の種類は セキュリティーの強化 に記載されていますが、Red Hat では、**aes-xts-plain64** または **aes-cbc-essiv:sha256** のいずれかの使用を強く推奨しています。

--escrowcert=URL_of_X.509_certificate

暗号化した全ボリュームのデータ暗号化の鍵を **/root** 配下にファイルとして格納します。**URL_of_X.509_certificate** で指定した URL の X.509 証明書を使用して暗号化します。鍵は暗号化したボリュームごとに別のファイルとして格納されます。**--encrypted** と併用しないと有効ではありません。

--luks-version=LUKS_VERSION

ファイルシステムの暗号化に使用する LUKS 形式のバージョンを指定します。**--encrypted** と併用しないと有効ではありません。

--backuppassphrase

暗号化されたボリュームにそれぞれランダムに生成されたパスフレーズを追加します。パスフレーズは、**/root** 配下に別々のファイルで格納され、**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと有効ではありません。

--pbkdf=PBKDF

LUKS 鍵スロット用の PBKDF (Password-Based Key Derivation Function) アルゴリズムを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。

--pbkdf-memory=PBKDF_MEMORY

PBKDF のメモリーコストを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。

--pbkdf-time=PBKDF_TIME

PBKDF パスフレーズ処理にかかるミリ秒数を設定します。**cryptsetup(8)** の man ページの **--iter-time** も併せて参照してください。このオプションは、**--encrypted** が指定される場合に限り有効になり、**--pbkdf-iterations** と相互に排他的になります。

--pbkdf-iterations=PBKDF_ITERATIONS

反復の数を直接設定し、PBKDF ベンチマークを回避します。**cryptsetup(8)** の man ページの **--pbkdf-force-iterations** も併せて参照してください。このオプションは、**--encrypted** が指定されている場合に限り有効になり、**--pbkdf-time** と相互に排他的になります。

--thinpool

シンプール論理ボリュームを作成します。(**none** のマウントポイントの使用)

--metadatasize=size

新しいシンプールデバイスのメタデータ領域のサイズ (MiB 単位) を指定します。

--chunksize=size

新しいシンプールデバイスのチャンクサイズ (KiB) を指定します。

--thin

シン論理ボリュームを作成します。(**--poolname** が必要です。)

--poolname=name

シン論理ボリュームを作成するシンプールの名前を指定します。**--thin** オプションが必要です。

--profile=name

シン論理ボリュームで使用する設定プロファイル名を指定します。これを使用する場合は、この名前は特定の論理ボリュームのメタデータにも含まれることになります。デフォルトで使用できるプロファイルは **default** と **thin-performance** で、**/etc/lvm/profile/** ディレクトリーで定義します。詳

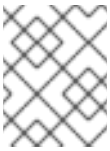
細は **lvm(8)** の man ページを参照してください。

--cachepvs=

該当ボリュームのキャッシュとして使用する物理ボリュームをコンマ区切りで記入します。

--cachemode=

当該論理ボリュームのキャッシュに使用するモードを指定します (**writeback** または **writethrough** になります)。



注記

キャッシュ済み論理ボリュームおよびそのモードの詳細は、**lvmcache(7)** の man ページを参照してください。

--cachesize=

論理ボリュームにアタッチするキャッシュのサイズを MiB 単位で指定します。このオプションは、**--cachepvs=** オプションと併用する必要があります。

注記

- キックスタートを使用して Red Hat Enterprise Linux をインストールする場合は、論理ボリューム名およびボリュームグループ名にダッシュ (**-**) 記号を使用しないでください。この文字を使用すると、インストール自体は正常に完了しますが、**/dev/mapper/** ディレクトリー内の論理ボリューム名とボリュームグループ名にダッシュが二重に付いてしまうことになります。たとえば、ボリュームグループ **volgrp-01** に論理ボリューム **logvol-01** が格納されている場合は、以下のような表記になります。**/dev/mapper/volgrp—01-logvol—01**.
この制約が適用されるのは、新規作成の論理ボリュームおよびボリュームグループ名のみです。既存の論理ボリュームまたはボリュームグループを **--noformat** オプションを使用して再利用する場合は、名前が変更されません。

- LUKS パスフレーズが分からなくなると、暗号化されたパーティションと、その上にあるデータには完全にアクセスできなくなります。分からなくなったパスフレーズを復元する方法はありません。ただし、**--escrowcert** を使用して暗号パスフレーズを保存し、**--backuppassphrase** オプションを使用してバックアップの暗号化パスフレーズを作成できます。

例

- まずパーティションを作成します。次に論理ボリュームグループを作成して、論理ボリュームを作成します。

```
part pv.01 --size 3000
volgroup myvg pv.01
logvol / --vgname=myvg --size=2000 --name=rootvol
```

- 最初にパーティションを作成します。次に論理ボリュームグループを作成して、ボリュームグループに残っている領域の 90 % を占める論理ボリュームを作成します。

```
part pv.01 --size 1 --grow
volgroup myvg pv.01
logvol / --vgname=myvg --name=rootvol --percent=90
```

関連情報

- 論理ボリュームの設定および管理

J.5.11. mount

キックスタートコマンドの **mount** は任意です。これは、既存のブロックデバイスにマウントポイントを割り当て、必要に応じて、指定の形式で再フォーマットします。

構文

```
mount [OPTIONS] device mountpoint
```

必須オプション:

- **device** - マウントするブロックデバイス。
- **mountpoint** - **device** をマウントする場所。/、/**usr** などの有効なマウントポイントを指定する必要があります。デバイスがマウントできない場合 (**swap** など) は **none** と指定します。

任意のオプション:

- **--reformat=** - デバイスを再フォーマットする際の新しいフォーマット (例: **ext4**) を指定します。
- **--mkfsoptions=** - **--reformat=** で指定した新しいファイルシステムを作成するコマンドに渡す追加のオプションを指定します。ここで指定するオプションのリストは処理されません。したがって、直接 **mkfs** プログラムに渡すことのできる形式で指定する必要があります。オプションのリストは、コンマ区切りとするか、二重引用符で囲む必要があります (ファイルシステムによって異なります)。詳細は、作成するファイルシステムの **mkfs** の man ページで確認してください (例: **mkfs.ext4(8)** または **mkfs.xfs(8)**)。
- **--mountoptions=** - ファイルシステムをマウントする場合に使用するオプションを含む文字列を、自由形式で指定します。この文字列はインストールされたシステムの /**etc**/**fstab** ファイルにコピーされるため、二重引用符で囲んでください。マウントオプションの全リストは **mount(8)** の man ページを、概要は **fstab(5)** を参照してください。

注記

- キックスタートの他の多くのストレージ設定コマンドとは異なり、**mount** の場合には、キックスタートファイルにすべてのストレージ設定を記述する必要がありません。確認する必要があるのは、記述されたブロックデバイスがシステムに存在することだけです。ただし、すべてのデバイスがマウントされたストレージスタックを **作成する** 場合には、**part** 等の他のコマンドを使用する必要があります。
- 同じキックスタートファイル内で、**mount** を **part**、**logvol**、または **autopart** などの他のストレージ関連コマンドと併用することはできません。

J.5.12. nvdimm

キックスタートコマンドの **nvdimm** は任意です。これは、NVDIMM (Non-Volatile Dual In-line Memory Module) デバイスでアクションを実行します。

Syntax

```
nvdimm action [OPTIONS]
```

アクション

- **reconfigure** - 指定した NVDIMM デバイスを特定のモードに再設定します。なお、指定したデバイスは暗示的にインストール先と識別されるため、同じデバイスに対するこれ以降の **nvdimm use** コマンドは冗長になります。このアクションは以下の形式を使用します。

```
nvdimm reconfigure [--namespace=NAMESPACE] [--mode=MODE] [--
sectorsize=SECTORSIZE]
```

  - **--namespace=** - 名前空間でデバイスを指定します。以下に例を示します。

```
nvdimm reconfigure --namespace=namespace0.0 --mode=sector --sectorsize=512
```

  - **--mode=** - モードを指定します。現在、利用できる値は **sector** だけです。
  - **--sectorsize=** - セクターサイズ (セクターモードの場合)。以下に例を示します。

```
nvdimm reconfigure --namespace=namespace0.0 --mode=sector --sectorsize=512
```

    サポートされるセクターサイズは 512 バイトおよび 4096 バイトです。

- **use** - NVDIMM デバイスを、インストールのターゲットに指定します。デバイスは **nvdimm reconfigure** コマンドでセクターモードに設定されている必要があります。このアクションは以下の形式を使用します。

```
nvdimm use [--namespace=NAMESPACE|--blockdevs=DEVICES]
```

  - **--namespace=** - 名前空間でデバイスを指定します。以下に例を示します。

```
nvdimm use --namespace=namespace0.0
```

  - **--blockdevs=** - 使用する NVDIMM デバイスに対応するブロックデバイスをコンマ区切りリストで指定します。ワイルドカードとしてアスタリスク * が使用できます。以下に例を示します。

```
nvdimm use --blockdevs=pmem0s,pmem1s
nvdimm use --blockdevs=pmem*
```

注記

- デフォルトでは、インストールプログラムはすべての NVDIMM デバイスを無視します。このデバイスでのインストールを有効にするには、**nvdimm** コマンドを使用する必要があります。

J.5.13. part または partition

キックスタートコマンド **part** または **partition** が必要です。このコマンドは、システムにパーティションを作成します。

Syntax

```
part|partition mntpoint --name=name --device=device --rule=rule [OPTIONS]
```

オプション

- **mntpoint** - パーティションをマウントする場所です。値は次のいずれかの形式になります。
  - **/path**
    /、**/usr**、**/home** など。
  - **swap**
    このパーティションは、swap 領域として使用されます。

    自動的に swap パーティションのサイズを確定させる場合は、**--recommended** オプションを使用します。

    ```
    swap --recommended
    ```

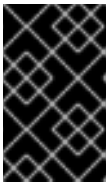
    有効なサイズが割り当てられますが、システムに対して正確に調整されたサイズではありません。

    自動的に swap パーティションサイズを確定しながら、ハイバネート用に余剰領域も割り当てる場合は、**--hibernation** オプションを使用します。

    ```
    swap --hibernation
    ```

    --recommended で割り当てられる swap 領域に加え、システムの RAM 容量が加算されたサイズが割り当てられるようになります。

    AMD64、Intel 64、および 64 ビット ARM のシステムで、このコマンドが割り当てたスワップサイズは「推奨されるパーティション設定スキーム」を参照してください。
  - **raid.id**
    このパーティションはソフトウェア RAID に使用されます (**raid** を参照)。
  - **pv.id**
    このパーティションは LVM に使用されます (**logvol** を参照)。
  - **biosboot**
    このパーティションは、BIOS 起動パーティションに使用されます。GPT (GUID Partition Table) を使用する BIOS ベースの AMD64 および Intel 64 のシステムには、1 MiB の BIOS 起動パーティションが必要になります。ブートローダーは、このパーティションにインストールされます。UEFI システムには必要ありません。詳細は **bootloader** コマンドをご覧ください。
  - **/boot/efi**
    EFI システムパーティションです。UEFI ベースの AMD64、Intel 64、および 64 ビットの ARM には 50 MiB の EFI パーティションが必要になります。推奨サイズは 200 MiB です。BIOS システムには必要ありません。詳細は **bootloader** コマンドをご覧ください。
- **--size=** - パーティションの最小サイズを MiB 単位で指定します。**500** などの整数値を使用してください (単位は不要)。

  重要

  --size の値が小さすぎると、インストールに失敗します。**--size** の値は、必要となる領域の最小値として指定します。推奨されるサイズは、「推奨されるパーティション設定スキーム」 を参照してください。

- **--grow** - パーティションを拡張して、利用可能なサイズ (存在する場合) を埋めるか、指定され

- **--grow** - パーティションを拡張して、利用可能なディスク (存在する場合) を埋めるか、指定されている場合は最大サイズまで埋めます。

注記

swap パーティションに **--maxsize=** を設定せずに **--grow=** を使用すると、swap パーティションの最大サイズは、Anaconda により制限されます。物理メモリーが 2 GiB 未満のシステムの場合は、物理メモリー量の 2 倍に制限されます。物理メモリーが 2 GiB 以上のシステムの場合は、物理メモリー量に 2GiB を足した量に制限されます。

- **--maxsize=** - パーティションが grow に設定されている場合の最大サイズを MiB 単位で指定します。**500** などの整数値を使用してください (単位は不要)。

- **--noformat** - パーティションをフォーマットしない場合に指定します。**--onpart** コマンドと併用してください。

- **--onpart=** または **--usepart=** - パーティションを配置するデバイスを指定します。既存の空のデバイスを使用し、新たに指定したタイプにフォーマットします。以下に例を示します。

```
partition /home --onpart=hda1
```

上記では、**/home** パーティションが **/dev/hda1** に配置されます。

このオプションを使用して、パーティションを論理ボリュームに追加することもできます。以下に例を示します。

```
partition pv.1 --onpart=hda2
```

この場合は、デバイスがシステムに存在している必要があります。**--onpart** オプションでデバイスを作成するわけではありません。

パーティションではなく、ドライブ全体を指定することも可能です。その場合、Anaconda はパーティションテーブルを作成せずに、ドライブをフォーマットして使用します。ただし、この方法でフォーマットしたデバイスでは GRUB2 のインストールがサポートされないため、パーティションテーブルのあるドライブに置かれる必要があります。

```
partition pv.1 --onpart=hdb
```

- **--ondisk=** または **--ondrive=** - 既存ディスクに (**part** コマンドで指定した) パーティションを作成します。このコマンドは常にパーティションを作成します。特定のディスクに強制的にパーティションを作成します。たとえば、**--ondisk=sdb** を使用すると、パーティションは 2 番目の SCSI ディスクに作成されます。
論理ボリューム管理 (LVM) を使用しないマルチパスデバイスを指定する場合は、**disk/by-id/dm-uuid-mpath-WWID** の形式を使用します。**WWID** は、デバイスの World-Wide Identifier です。たとえば、WWID **2416CD96995134CA5D787F00A5AA11017** のディスクを指定する場合は以下を使用します。

```
part / --fstype=xfs --grow --asprimary --size=8192 --ondisk=disk/by-id/dm-uuid-mpath-
2416CD96995134CA5D787F00A5AA11017
```

警告

mpatha などのデバイス名でマルチパスデバイスを指定しないでください。このようなデバイス名は、特定のディスクに固有の名前ではありません。インストール中の **/dev/mpatha** という名前のディスクは必ずしも期待したディスクを指すとは限りません。したがって、**part** コマンドを使用する際は、間違ったディスクが対象となる可能性があります。

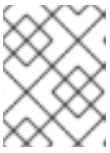
- **--asprimary** - パーティションが **プライマリー** パーティションとして割り当てられるように強制実行します。(通常、すでに割り当てられているプライマリーパーティションが多すぎるという理由で) パーティションをプライマリーとして割り当てられない場合は、パーティション設定のプロセスが失敗します。このオプションは、Master Boot Record (MBR) をディスクが使用する場合にのみ有効で、GUID Partition Table (GPT) ラベルが付いたディスクでは有効ではありません。

- **--fsprofile=** - このパーティションでファイルシステムを作成するプログラムに渡すのに使用するタイプを指定します。ファイルシステムの作成時に使用されるさまざまなチューニングパラメーターは、この使用タイプにより定義されます。ファイルシステム側で使用タイプという概念に対応し、有効なタイプを指定する設定ファイルがないと、このオプションは正しく機能しません。**ext2**、**ext3**、**ext4** の場合、この設定ファイルは **/etc/mke2fs.conf** になります。

- **--mkfsoptions=** - このパーティションでファイルシステムを作成するプログラムに渡す追加のパラメーターを指定します。これは **--fsprofile** と似ていますが、プロフィールの概念に対応するものだけではなく、すべてのファイルシステムで機能するものです。引数のリストでは処理が行われないため、mkfs プログラムに直接渡すことが可能な形式で提供する必要があります。つまり、複数のオプションはコンマ区切りにするか、二重引用符で囲む必要があります (ファイルシステムによって異なります)。

- **--fstype=** - パーティションのファイルシステムタイプを設定します。使用できる値は、**xfs**、**ext2**、**ext3**、**ext4**、**swap**、**vfat**、**efi**、および **biosboot** になります。

- **--fsoptions=** - ファイルシステムをマウントする場合に使用するオプションの文字列を自由形式で指定します。この文字列はインストール後の **/etc/fstab** ファイルにコピーされるため、引用符で囲んでください。



注記

EFI システムパーティション (**/boot/efi**) では、anaconda が値をハードコードし、ユーザー指定の **--fsoptions** 値を無視します。

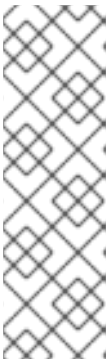
- **--label=** - 個別パーティションにラベルを割り当てます。

- **--recommended** - パーティションのサイズを自動的に確定します。AMD64、Intel 64、および 64 ビット ARM のシステムで推奨されるスキームは、「推奨されるパーティション設定スキーム」 を参照してください。

重要

このオプションは、**/boot** パーティションや **swap** 領域といったファイルシステムになるパーティションにのみ使用できます。LVM 物理ボリュームや RAID メンバーの作成には使用できません。

- **--onbiosdisk** - BIOS で検出された特定のディスクに強制的にパーティションを作成します。
- **--encrypted** - **--passphrase** オプションで入力したパスフレーズを使用して、LUKS (Linux Unified Key Setup) でこのパーティションを暗号化するように指定します。このパスフレーズを指定しないと、Anaconda は、**autopart --passphrase** コマンドで設定されるデフォルトのシステムワイドパスフレーズを使用します。このデフォルトのパスフレーズも設定されていない場合は、インストールプロセスが中断して、パスフレーズの入力が求められます。



注記

1 つまたは複数のパーティションを暗号化する際には、安全な暗号化を行うため、Anaconda が 256 ビットのエントロピーを収集しようとします。エントロピーの収集には時間がかかる場合があります。十分なエントロピーが収集されたかどうかにかかわらず、このプロセスは最大 10 分後に終了します。

プロセスは、インストールシステムと対話することにより高速化できます (キーボードで入力またはマウスの移動)。仮想マシンにインストールしている場合は、**virtio-rng** デバイス (仮想乱数ジェネレーター) をゲストに登録できます。

- **--luks-version=LUKS_VERSION** - ファイルシステムの暗号化に使用する LUKS 形式のバージョンを指定します。**--encrypted** と併用しないと有効ではありません。
- **--passphrase=** - このパーティションの暗号化を行う際に使用するパスフレーズを入力します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。
- **--cipher=** - Anaconda のデフォルトである **aes-xts-plain64** では十分ではない場合に使用する暗号化の種類を指定します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。使用できる暗号化の種類は セキュリティーの強化 に記載されていますが、Red Hat では、**aes-xts-plain64** または **aes-cbc-essiv:sha256** のいずれかの使用を強く推奨しています。
- **--escrowcert=URL_of_X.509_certificate** - 暗号化した全パーティションのデータ暗号化の鍵を **/root** 配下にファイルとして格納します。**URL_of_X.509_certificate** で指定した URL の X.509 証明書を使用して暗号化します。鍵は、暗号化したパーティションごとに別のファイルとして格納されます。**--encrypted** と併用しないと有効ではありません。
- **--backuppassphrase** - 暗号化されたパーティションにそれぞれランダムに生成されたパスフレーズを追加します。パスフレーズは、**/root** 配下に別々のファイルで格納され、**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと有効ではありません。
- **--pbkdf=PBKDF** - LUKS 鍵スロット用の PBKDF (Password-Based Key Derivation Function) アルゴリズムを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。
- **--pbkdf-memory=PBKDF_MEMORY** - PBKDF のメモリーコストを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。
- **--pbkdf-time=PBKDF_TIME** - PBKDF パスフレーズ処理にかかるミリ秒数を設定しま

す。cryptsetup(8) の man ページの **--iter-time** も併せて参照してください。このオプションは、**--encrypted** が指定される場合に限り有効になり、**--pbkdf-iterations** と相互に排他的になります。

- **--pbkdf-iterations=PBKDF_ITERATIONS** - 反復の数を直接設定し、PBKDF ベンチマークを回避します。cryptsetup(8) の man ページの **--pbkdf-force-iterations** も併せて参照してください。このオプションは、**--encrypted** が指定されている場合に限り有効になり、**--pbkdf-time** と相互に排他的になります。

- **--resize=** - 既存パーティションのサイズを変更します。このオプションを使用する際は、**--size=** オプションで目的のサイズ (MiB 単位) を指定し、 **--onpart=** オプションで目的のパーティションを指定します。

注記

- **part** コマンドは必須ではありませんが、キックスタートスクリプトには **part**、**autopart**、または **mount** のいずれかを指定する必要があります。

- **--active** オプションは、Red Hat Enterprise Linux 8 で非推奨となりました。

- 何らかの理由でパーティションの設定ができなかった場合には、診断メッセージが仮想コンソール 3 に表示されます。

- **--noformat** および **--onpart** を使用しないと、作成されたパーティションはすべてインストールプロセスの一部としてフォーマット化されます。

- **sdX** (または **/dev/sdX**) 形式でのデバイス名がシステムの再起動後に維持される保証がないため、一部のキックスタートコマンドを複雑にします。コマンドがデバイスノード名を呼び出す際には、代わりに **/dev/disk** からのアイテムを使用することができます。以下に例を示します。

```
part / --fstype=xfs --onpart=sda1
```

以下のいずれかのようなエントリーを使用します。

```
part / --fstype=xfs --onpart=/dev/disk/by-path/pci-0000:00:05.0-scsi-0:0:0:0-part1
```

```
part / --fstype=xfs --onpart=/dev/disk/by-id/ata-ST3160815AS_6RA0C882-part1
```

上記の手順により、コマンドは常に同じストレージデバイスをターゲットとします。これは特に大型のストレージ環境で便利なものです。ストレージデバイスを連続して参照するさまざまな方法についての詳細は、**Managing storage devices** の Overview of persistent naming attributes を参照してください。

- LUKS パスフレーズが分からなくなると、暗号化されたパーティションと、その上にあるデータには完全にアクセスできなくなります。分からなくなったパスフレーズを復元する方法はありません。ただし、**--escrowcert** を使用して暗号パスフレーズを保存し、**--backuppassphrase** オプションを使用してバックアップの暗号化パスフレーズを作成できます。

J.5.14. raid

キックスタートコマンドの **raid** は任意です。ソフトウェアの RAID デバイスを組み立てます。

Syntax

```
raid mntpoint --level=level --device=device-name partitions*
```

オプション

- **mntpoint** - RAID ファイルシステムをマウントする場所です。/ にマウントする場合、boot パーティション (**/boot**) がなければ RAID レベルは 1 にする必要があります。boot パーティションがある場合は、**/boot** パーティションをレベル 1 にしてください。ルート (/) パーティションのタイプはどれでも構いません。**partitions*** (複数パーティションの指定が可能) には RAID アレイに追加する RAID 識別子を指定します。

重要

  - IBM Power Systems で RAID デバイスの準備は行ったものの、インストール中に再フォーマットを行っていない場合で、この RAID デバイスに **/boot** パーティションおよび PReP パーティションの配置を予定している場合は、RAID メタデータのバージョンが **0.90** または **1.0** になっていることを確認してください。**mdadm** メタデータバージョン **1.1** および **1.2** は、**/boot** および PReP パーティションではサポートされていません。
  - PowerNV システムでは、**PReP** Boot パーティションは必要ありません。

- **--level=**: 使用する RAID レベルを指定します (0、1、4、5、6、10 のいずれか)。
利用可能な RAID レベルの詳細は、「対応する RAID のタイプ」 を参照してください。

- **--device=** - 使用する RAID デバイス名を指定します (例: **--device=root**)。

重要

mdraid 名を **md0** の形式で使用しないでください。このような名前は永続性が保証されていません。代わりに、**root**、**swap** など意味のある名前にしてください。意味のある名前を使用すると、**/dev/md/name** から、アレイに割り当てられている **/dev/mdX** ノードへのシンボリックリンクが作成されます。

名前を割り当てることができない古い (v0.90 メタデータ) アレイがある場合は、ファイルシステムラベルまたは UUID でアレイを指定できます。たとえば、**--device=LABEL=root** または **--device=UUID=93348e56-4631-d0f0-6f5b-45c47f570b88** です。

RAID デバイス上のファイルシステムの UUID または RAID デバイス自体の UUID を使用できます。RAID デバイスの UUID は **8-4-4-4-12** 形式である必要があります。mdadm によって報告される UUID は、変更する必要がある **8:8:8:8** 形式です。たとえば、**93348e56:4631d0f0:6f5b45c4:7f570b88** は **93348e56-4631-d0f0-6f5b-45c47f570b88** に変更する必要があります。

- **--chunksize=** - RAID ストレージのチャンクサイズを KiB 単位で設定します。場合によっては、デフォルトのサイズ (**512 Kib**) 以外のチャンクサイズを使用すると、RAID のパフォーマンスが向上することもあります。

- **--spares=** - RAID アレイに割り当てられるスペアドライブの数を指定します。スペアドライブは、ドライブに障害が発生した場合にアレイの再設定に使用されます。

- **--fsprofile=** - このパーティションでファイルシステムを作成するプログラムに渡すのに使用するタイプを指定します。ファイルシステムの作成時に使用されるさまざまなチューニングパラメーターは、この使用タイプにより定義されます。ファイルシステム側で使用タイプという概

念に対応し、有効なタイプを指定する設定ファイルがないと、このオプションは正しく機能しません。ext2、ext3、ext4 の場合、この設定ファイルは **/etc/mke2fs.conf** にあります。

- **--fstype=** - RAID アレイのファイルシステムタイプを設定します。**xfs**、**ext2**、**ext3**、**ext4**、**swap**、および **vfat** が使用できる値になります。

- **--fsoptions=** - ファイルシステムをマウントする場合に使用するオプションの文字列を自由形式で指定します。この文字列はインストール後の **/etc/fstab** ファイルにコピーされるため、引用符で囲んでください。

注記

EFI システムパーティション (**/boot/efi**) では、anaconda が値をハードコードし、ユーザー指定の **--fsoptions** 値を無視します。

- **--mkfsoptions=** - このパーティションでファイルシステムを作成するプログラムに渡す追加のパラメーターを指定します。引数のリストでは処理が行われないため、mkfs プログラムに直接渡すことが可能な形式で提供する必要があります。つまり、複数のオプションはコンマ区切りにするか、二重引用符で囲む必要があります (ファイルシステムによって異なります)。

- **--label=** - 作成するファイルシステムのラベルを指定します。指定ラベルが別のファイルシステムですでに使用されている場合は、新しいラベルが作成されます。

- **--noformat** - 既存の RAID デバイスを使用し、RAID アレイのフォーマットは行いません。

- **--useexisting** - 既存の RAID デバイスを使用し、再フォーマット化を行います。

- **--encrypted** - **--passphrase** オプションで入力したパスフレーズを使用して、LUKS (Linux Unified Key Setup) でこの RAID デバイスを暗号化するように指定します。このパスフレーズを指定しないと、Anaconda は、**autopart --passphrase** コマンドで設定されるデフォルトのシステムワイドパスフレーズを使用します。このデフォルトのパスフレーズも設定されていない場合は、インストールプロセスが中断して、パスフレーズの入力が求められます。

注記

1 つまたは複数のパーティションを暗号化する際には、安全な暗号化を行うため、Anaconda が 256 ビットのエントロピーを収集しようとします。エントロピーの収集には時間がかかる場合があります。十分なエントロピーが収集されたかどうかにかかわらず、このプロセスは最大 10 分後に終了します。

プロセスは、インストールシステムと対話することにより高速化できます (キーボードで入力またはマウスの移動)。仮想マシンにインストールしている場合は、**virtio-rng** デバイス (仮想乱数ジェネレーター) をゲストに登録できます。

- **--luks-version=LUKS_VERSION** - ファイルシステムの暗号化に使用する LUKS 形式のバージョンを指定します。**--encrypted** と併用しないと有効ではありません。

- **--cipher=** - Anaconda のデフォルトである **aes-xts-plain64** では十分ではない場合に使用する暗号化の種類を指定します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。使用できる暗号化の種類は セキュリティーの強化 に記載されていますが、Red Hat では、**aes-xts-plain64** または **aes-cbc-essiv:sha256** のいずれかの使用を強く推奨しています。

- **--passphrase=** - この RAID デバイスの暗号化を行う際に使用するパスフレーズを入力します。**--encrypted** オプションと併用してください。単独で使用しても暗号化されません。

- **--escrowcert=URL_of_X.509_certificate** - このデバイス用のデータ暗号化の鍵を **/root** 配下にファイルとして格納します。鍵は、**URL_of_X.509_certificate** で指定した URL の X.509 証明書を使用して暗号化します。**--encrypted** と併用しないと有効ではありません。

- **--backuppassphrase** - このデバイスにランダムに生成されたパスフレーズを追加します。パスフレーズは **/root** 配下にファイルとして保存し、**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと有効ではありません。

- **--pbkdf=PBKDF** - LUKS 鍵スロット用の PBKDF (Password-Based Key Derivation Function) アルゴリズムを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。

- **--pbkdf-memory=PBKDF_MEMORY** - PBKDF のメモリーコストを設定します。**cryptsetup(8)** の man ページも併せて参照してください。**--encrypted** と併用しないと有効ではありません。

- **--pbkdf-time=PBKDF_TIME** - PBKDF パスフレーズ処理にかかるミリ秒数を設定します。**cryptsetup(8)** の man ページの **--iter-time** も併せて参照してください。このオプションは、**--encrypted** が指定される場合に限り有効になり、**--pbkdf-iterations** と相互に排他的になります。

- **--pbkdf-iterations=PBKDF_ITERATIONS** - 反復の数を直接設定し、PBKDF ベンチマークを回避します。**cryptsetup(8)** の man ページの **--pbkdf-force-iterations** も併せて参照してください。このオプションは、**--encrypted** が指定されている場合に限り有効になり、**--pbkdf-time** と相互に排他的になります。

例

以下の例では、/ には RAID レベル 1 のパーティション、 **/home** には RAID レベル 5 のパーティションを作成します。ここでは、システムには SCSI ディスクが 3 つあることが前提です。各ドライブに 1 つずつ、3 つの swap パーティションを作成します。

```
part raid.01 --size=6000 --ondisk=sda
part raid.02 --size=6000 --ondisk=sdb
part raid.03 --size=6000 --ondisk=sdc
part swap --size=512 --ondisk=sda
part swap --size=512 --ondisk=sdb
part swap --size=512 --ondisk=sdc
part raid.11 --size=1 --grow --ondisk=sda
part raid.12 --size=1 --grow --ondisk=sdb
part raid.13 --size=1 --grow --ondisk=sdc
raid / --level=1 --device=rhel8-root --label=rhel8-root raid.01 raid.02 raid.03
raid /home --level=5 --device=rhel8-home --label=rhel8-home raid.11 raid.12 raid.13
```

注記

- LUKS パスフレーズが分からなくなると、暗号化されたパーティションと、その上にあるデータには完全にアクセスできなくなります。分からなくなったパスフレーズを復元する方法はありません。ただし、**--escrowcert** を使用して暗号パスフレーズを保存し、**--backuppassphrase** オプションを使用してバックアップの暗号化パスフレーズを作成できます。

J.5.15. reqpart

キックスタートコマンドの **reqpart** は任意です。使用中のハードウェアプラットホームで必要となるパーティションを自動的に作成します。UEFI ファームウェアのシステム向けに **/boot/efi** パーティショ

ン、BIOS ファームウェアおよび GPT のシステム向けに **biosboot** パーティション、IBM Power Systems 向けに **PRePBoot** パーティションが作成されます。

構文

```
reqpart [--add-boot]
```

オプション

- **--add-boot** - ベースコマンドが作成するプラットホーム固有のパーティションとは別に、**/boot** パーティションを作成します。

注記

- このコマンドは、**autopart** と併用することはできません。**autopart** は **reqpart** コマンドの実行内容に加えて、他のパーティションや、/、**swap** といった論理ボリュームも作成するためです。**autopart** とは異なり、このコマンドは、プラットホーム固有のパーティションの作成のみを行い、ドライブの残りは空のままにするため、カスタムレイアウトの作成が可能になります。

J.5.16. snapshot

キックスタートコマンドの **snapshot** は任意です。インストールプロセス時に、このコマンドを使用して LVM のシンボリュームのスナップショットを作成できます。これにより、インストール前後の論理ボリュームのバックアップ作成が可能になります。

複数のスナップショットを作成するには、キックスタートコマンドの **snaphost** を複数回追加します。

Syntax

```
snapshot vg_name/lv_name --name=snapshot_name --when=pre-install|post-install
```

オプション

- **vg_name/lv_name** - スナップショットの作成元となるボリュームグループや論理ボリュームの名前を設定します。
- **--name=snapshot_name** - スナップショットの名前を設定します。この名前は、ボリュームグループ内で一意のものにする必要があります。
- **--when=pre-install|post-install** - インストール前もしくは完了後にスナップショットを作成することを指定します。

J.5.17. volgroup

キックスタートコマンドの **volgroup** は任意です。論理ボリューム管理 (LVM) グループを作成します。

Syntax

```
volgroup name [OPTIONS] [partition*]
```

必須オプション

- **name** - 新しいボリュームグループの名前。

オプション

- **partition** - ボリュームグループのバッキングストレージとして使用する物理ボリュームパーティション。
- **--noformat** - 既存のボリュームグループを使用し、フォーマットは行いません。
- **--useexisting** - 既存のボリュームグループを使用し、そのボリュームグループを再フォーマットします。このオプションを使用する場合は **partition** は指定しないでください。以下に例を示します。

```
volgroup rhel00 --useexisting --noformat
```

- **--pesize=** - ボリュームグループの物理エクステントのサイズをキビバイト (KiB) 単位で設定します。デフォルト値は 4096 (4 MiB) で、最小値は 1024 (1 MiB) になります。
- **--reserved-space=** - ボリュームグループに未使用で残す領域を MiB 単位で指定します。新規作成のボリュームグループにのみ適用されます。
- **--reserved-percent=** - 未使用で残すボリュームグループ領域全体の割合を指定します。新規作成のボリュームグループにのみ適用されます。

注記

- まずパーティションを作成します。次に論理ボリュームグループを作成して、論理ボリュームを作成します。以下に例を示します。

```
part pv.01 --size 10000
volgroup my_volgrp pv.01
logvol / --vgname=my_volgrp --size=2000 --name=root
```

- キックスタートを使用して Red Hat Enterprise Linux をインストールする場合は、論理ボリューム名およびボリュームグループ名にダッシュ (**-**) 記号を使用しないでください。この文字を使用すると、インストール自体は正常に完了しますが、**/dev/mapper/** ディレクトリー内の論理ボリューム名とボリュームグループ名にダッシュが二重に付いてしまうことになります。たとえば、**logvol-01** という名前の論理ボリュームを格納する **volgrp-01** という名前のボリュームグループなら、**/dev/mapper/volgrp--01-logvol--01** というような表記になってしまいます。この制約が適用されるのは、新規作成の論理ボリュームおよびボリュームグループ名のみです。既存の論理ボリュームまたはボリュームグループを **--noformat** オプションを使用して再利用する場合は、名前が変更されません。

J.5.18. zerombr

キックスタートコマンドの **zerombr** は任意です。**zerombr** は、ディスク上で見つかった無効なパーティションテーブルを初期化し、無効なパーティションテーブルがあるディスクの内容をすべて破棄します。このコマンドは、フォーマットされていない DASD (Direct Access Storage Device) ディスクを備えた 64 ビットの IBM Z システムでインストールを実行する場合に必要です。このコマンドを使用しないと、フォーマットされていないディスクがインストール時にフォーマットされず、使用されません。

Syntax

```
zerombr
```

注記

- 64 ビットの IBM Z では **zerombr** が指定された場合、インストールプログラムに見えている Direct Access Storage Device (DASD) でまだ低レベルフォーマット処理がなされていないものは、自動的に dasdfmt で低レベルフォーマット処理がなされます。このコマンドでは、対話型インストール中のユーザー選択も行われません。
- **zerombr** が指定されておらず、少なくとも 1 つの未フォーマットの DASD がインストールプログラムに見えている場合、非対話形式のキックスタートを使用したインストールは失敗に終わります。
- **zerombr** が指定されておらず、少なくとも 1 つの未フォーマットの DASD がインストールプログラムに見えている場合、ユーザーがすべての見えている未フォーマットの DASD のフォーマットに同意しなければ、対話形式のインストールは終了します。この状況を回避するには、インストール中に使用する DASD のみをアクティベートします。DASD は、インストール完了後にいつでも追加できます。
- このコマンドにはオプションはありません。

J.5.19. zfcp

キックスタートコマンドの **zfcp** は任意です。Fibre チャンネルデバイスを定義します。

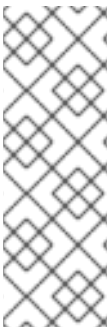
このオプションは、64 ビットの IBM Z にのみ適用されます。下記のオプションをすべて指定する必要があります。

Syntax

```
zfcp --devnum=devnum [--wwpn=wwpn --fcplun=lun]
```

オプション

- **--devnum=** - デバイス番号 (zFCP アダプターデバイスバス ID)。
- **--wwpn=** - デバイスの WWPN (ワールドワイドポートネーム)。**0x** で始まる 16 桁の番号になります。
- **--fcplun=** - デバイスの論理ユニット番号 (LUN)。**0x** で始まる 16 桁の番号になります。



注記

自動 LUN スキャンが利用できる場合や、8 以降のリリースをインストールする場合は、FCP デバイスバス ID を指定するだけで十分です。それ以外の場合は、3 つのパラメーターがすべて必要になります。自動 LUN スキャンは、**zfcp.allow_lun_scan** モジュールパラメーターで無効にされていない場合 (デフォルトでは有効)、NPIV モードで動作する FCP デバイスで使用できます。これは、指定されたバス ID を持つ FCP デバイスに接続されたストレージエリアネットワークで見つかったすべての SCSI デバイスへのアクセスを提供します。

Example

```
zfcp --devnum=0.0.4000 --wwpn=0x5005076300C213e9 --fcplun=0x5022000000000000
zfcp --devnum=0.0.4000
```

J.6. RHEL インストールプログラムで提供されるアドオン向けキックスタートコマンド

本セクションのキックスタートコマンドは、Red Hat Enterprise Linux インストールプログラム(Kdump および OpenSCAP) にデフォルトで提供されるアドオンに関連しています。

J.6.1. %addon com_redhat_kdump

キックスタートコマンドの **%addon com_redhat_kdump** は任意です。このコマンドは、kdump カーネルクラッシュのダンプメカニズムを設定します。

Syntax

```
%addon com_redhat_kdump [OPTIONS]
%end
```

注記

このコマンドは、ビルトインのキックスタートコマンドではなくアドオンであることから、構文は通常のものとは異なります。

注記

Kdump とは、システムのメモリーの内容を保存して後で分析できるように、カーネルのクラッシュをダンプするメカニズムを指します。これは、**kexec** に依存し、別のカーネルのコンテキストから、システムを再起動することなく Linux カーネルを起動し、通常は失われてしまう 1 番目のカーネルメモリーの内容を維持できます。

システムクラッシュが発生すると、**kexec** は 2 番目のカーネルで起動します (キャプチャーカーネル)。このキャプチャーカーネルは、システムメモリーの予約部分に収納されています。このため、Kdump は、クラッシュしたカーネルメモリーの内容 (クラッシュダンプ) をキャプチャーして、指定した場所に保存します。この場所は、このキックスタートコマンドを使用して設定することはできません。インストール後に **/etc/kdump.conf** 設定ファイルを編集して設定する必要があります。

Kdump の詳細は、**Managing, monitoring and updating the kernel**の Installing kdump の章を参照してください。

オプション

- **--enable** - インストール済みのシステムで kdump を有効にします。
- **--disable** - インストール済みのシステムで kdump を無効にします。
- **--reserve-mb=** - kdump 用に予約するメモリーの量 (MiB 単位)。以下に例を示します。

```
%addon com_redhat_kdump --enable --reserve-mb=128
%end
```

数値の代わりに **auto** と指定することもできます。その場合は、インストールプログラムは、**Managing, monitoring and updating the kernel**の Memory requirements for kdump のセクションに記載の基準に基づいて、メモリーの量を自動的に決定します。

kdump を有効にして、**--reserve-mb=** オプションを指定しないと、**auto** の値が使用されます。

- **--enablefadump** - 対応するシステム (特に IBM Power Systems サーバー) へのファームウェア補助によるダンピングを有効にします。

J.6.2. %addon org_fedora_oscap

キックスタートコマンドの **%addon org_fedora_oscap** は任意です。

OpenSCAP インストールプログラムのアドオンは、インストールシステム上で SCAP (Security Content Automation Protocol) のコンテンツ (セキュリティーポリシー) を適用するために使用されます。Red Hat Enterprise Linux 7.2 以降、このアドオンがデフォルトで有効になりました。有効にすると、この機能の提供に必要なパッケージが自動的にインストールされます。ただし、デフォルトではポリシーが強制されることがなく、明確に設定されている場合を除いて、インストール時およびインストール後にチェックが行われません。

重要

セキュリティーポリシーの適用はすべてのシステムで必要なわけではありません。この画面は、特定のポリシーが業務規定や法令で義務付けられている場合に限り使用してください。

多くのコマンドとは異なり、このアドオンは通常のオプションを受け付けず、**%addon** 定義の本文で鍵と値のペアを使用します。空白は無視されます。値は一重引用符 (') または二重引用符 (") で囲みます。

構文

```
%addon org_fedora_oscap
key = value
%end
```

鍵

アドオンは以下の鍵を認識します。

content-type

セキュリティーコンテンツのタイプ。値は、**datastream**、**archive**、**rpm**、または **scap-security-guide** になります。
content-type を **scap-security-guide** にすると、アドオンは scap-security-guide パッケージが提供するコンテンツを使用します。このパッケージは起動用メディアにあります。つまり、**profile** を除く他のすべての鍵の影響がなくなります。

content-url

セキュリティーコンテンツの場所。コンテンツは、HTTP、HTTPS、FTP のいずれかを使用してアクセスできるようにする必要があります。ローカルストレージは現在、サポートされていません。リモートの場所にあるコンテンツ定義に到達するネットワーク接続が必要になります。

datastream-id

content-url で参照されているデータストリームの ID。**content-type** が **datastream** の場合にのみ使用します。

xccdf-id

使用するベンチマークの ID。

content-path

使用するデータストリームまたは XCCDF ファイルのパスを、アーカイブ内の相対パスで指定します。

profile

適用するプロファイルの ID。デフォルトのプロファイルを使用する場合は **default** を使用してください。

フィンガープリント (fingerprint)

content-url で参照されるコンテンツの MD5、SHA1、または SHA2 のチェックサム。

tailoring-path

使用するテーラリングファイルのパスを、アーカイブの相対パスで指定します。

例

- インストールメディアの **scap-security-guide** のコンテンツを使用する **%addon org_fedora_oscap** セクションの例は、以下のようになります。

  例J.1 SCAP Security Guide を使用した OpenSCAP アドオン定義の例

  ```
  %addon org_fedora_oscap
  content-type = scap-security-guide
  profile = xccdf_org.ssgproject.content_profile_pci-dss
  %end
  ```

- Web サーバーからカスタムプロファイルを読み込むより複雑な例は、以下のようになります。

  例J.2 データストリームを使用した OpenSCAP アドオン定義の例

  ```
  %addon org_fedora_oscap
  content-type = datastream
  content-url = http://www.example.com/scap/testing_ds.xml
  datastream-id = scap_example.com_datastream_testing
  xccdf-id = scap_example.com_cref_xccdf.xml
  profile =  xccdf_example.com_profile_my_profile
  fingerprint = 240f2f18222faa98856c3b4fc50c4195
  %end
  ```

関連情報

- セキュリティーの強化
- OpenSCAP installation program add-on
- OpenSCAP Portal

J.7. ANACONDA で使用されるコマンド

pwpolicy コマンドは、キックスタートファイルの **%anaconda** セクションでのみ使用できる Anaconda UI 固有のコマンドです。

J.7.1. pwpolicy

キックスタートコマンドの **pwpolicy** は任意です。このコマンドを使用して、インストール中にカスタムパスワードポリシーを適用します。このポリシーでは、ユーザーアカウントの root、ユーザー、または luks ユーザーのパスワードを作成する必要があります。パスワードの長さや強度などの要因により、パスワードの有効性が決まります。

Syntax

```
pwpolicy name [--minlen=length] [--minquality=quality] [--strict|--nostrict] [--emptyok|--noempty] [--changesok|--nochanges]
```

必須オプション

- name - **root**、**user**、または **luks** に置き換え、それぞれ **root** パスワード、ユーザーパスワード、もしくは LUKS パスフレーズのポリシーを強制します。

任意のオプション

- **--minlen=** - パスワードの最低文字数を設定します。デフォルト値は **6** です。
- **--minquality=** - **libpwquality** ライブラリーで定義されるパスワードの最低限の質を設定します。デフォルト値は **1** です。
- **--strict** - 厳密なパスワード強制を有効にします。**--minquality=** と **--minlen=** で指定された要件を満たさないパスワードは拒否されます。このオプションはデフォルトで無効になっています。
- **--notstrict** - **--minquality=** オプションおよび **-minlen=** オプションで指定した最小要件を **満たさない** パスワードは、GUI で **完了** を 2 回クリックすると可能になります。テキストモードインターフェイスでは、同様のメカニズムが使用されます。
- **--emptyok** - 空のパスワードの使用を許可します。デフォルトでユーザーパスワードに有効となっています。
- **--notempty** - 空のパスワードの使用を許可しません。root パスワードと LUKS パスフレーズについて、デフォルトで有効になっています。
- **--changesok** - キックスタートファイルでパスワードが設定されていても、ユーザーインターフェイスでのパスワード変更を許可します。デフォルトでは無効です。
- **--nochanges** - キックスタートファイルで設定されているパスワードの変更を許可しません。デフォルトでは有効です。

注記

- **pwpolicy** コマンドは、キックスタートファイルの **%anaconda** セクションでのみ使用できる Anaconda UI 固有のコマンドです。
- **libpwquality** ライブラリーは、パスワードの最低要件 (長さおよび質) の確認に使用されます。**libpwquality** パッケージが提供する **pwscore** コマンドおよび **pwmake** コマンドを使用してパスワードの質のスコアを確認するか、特定スコアのパスワードをランダムに作成できま

す。これらのコマンドの詳細は、**pwscore(1)** および **pwmake(1)** の man ページを参照してください。

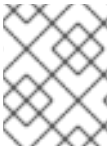
J.8. システム復旧用キックスタートコマンド

このセクションのキックスタートコマンドは、インストールされたシステムを修復します。

J.8.1. rescue

キックスタートコマンドの **rescue** は任意です。これは、root 権限を備えたシェル環境と、インストールを修復して次のような問題のトラブルシューティングを行うための一連のシステム管理ツールを提供します。

- ファイルシステムを読み取り専用としてマウントする
- ドライバーディスクで提供されているドライバーを拒否リスト登録または追加する
- システムパッケージをインストールまたはアップグレードする
- パーティションを管理する



注記

キックスタートレスキューモードは、systemd およびサービスマネージャーの一部として提供されるレスキューモードおよび緊急モードとは異なります。

rescue コマンドは、システム自体を変更することはありません。読み取り/書き込みモードでシステムを /mnt/sysimage の下にマウントすることにより、レスキュー環境を設定するだけです。システムをマウントしないか、読み取り専用モードでマウントするかを選択できます。

構文

```
rescue [--nomount|--romount]
```

オプション

- **--nomount** または **--romount** - インストールを完了したシステムをレスキュー環境でマウントする方法を制御します。デフォルトでは、インストールプログラムによりシステムの検出が行われてから、読み取りと書き込みのモードでシステムのマウントが行われ、マウントされた場所が通知されます。オプションでマウントを行わない (**--nomount** オプション)、または読み取り専用モードでマウントする (**--romount** オプション) のいずれかを選択できます。指定できるのはどちらか一方です。

注記

レスキューモードを実行するには、キックスタートファイルのコピーを作成し、それに **rescue** コマンドを含めます。

rescue コマンドを使用すると、インストーラーは次の手順を実行します。

1. **%pre** スクリプトを実行します。
2. レスキューモードの環境をセットアップします。
   以下のキックスタートコマンドが有効になります。

a. updates

b. sshpw

c. logging

d. lang

e. network

3. 高度なストレージ環境を設定します。
   以下のキックスタートコマンドが有効になります。

   a. fcoe

   b. iscsi

   c. iscsiname

   d. nvdimm

   e. zfcp

4. システムをマウントします。

```
rescue [--nomount|--romount]
```

5. %post スクリプトを実行します。
   この手順は、インストールされたシステムが読み取り/書き込みモードでマウントされている場合にのみ実行されます。

6. シェルを開始します。

7. システムを再起動します。

パート II. セキュリティーの設計

第11章 RHEL におけるセキュリティーの強化の概要

ビジネスの運営や個人情報の把握ではネットワーク化された強力なコンピューターへの依存度が高まっていることから、各種業界ではネットワークとコンピューターのセキュリティーの実践に関心が向けられています。企業は、システム監査を適正に行い、ソリューションが組織の運営要件を満たすようにするために、セキュリティーの専門家の知識と技能を求めてきました。多くの組織はますます動的になってきていることから、従業員は、会社の重要な IT リソースに、ローカルまたはリモートからアクセスするようになっています。このため、セキュアなコンピューティング環境に対するニーズはより顕著になっています。

にも関わらず、多くの組織 (個々のユーザーも含む) は、機能性、生産性、便利さ、使いやすさ、および予算面の懸念事項にばかり目を向け、セキュリティーをその結果論と見なし、セキュリティーのプロセスが見過ごされています。したがって、適切なセキュリティーの確保は、無許可の侵入が発生してはじめて徹底されることも少なくありません。多くの侵入の試みを阻止する効果的な方法は、インターネットなどの信頼できないネットワークにサイトを接続する前に、適切な措置を講じることです。

11.1. コンピューターセキュリティーとは

コンピューターセキュリティーは、コンピューティングと情報処理の幅広い分野で使用される一般的な用語です。コンピューターシステムとネットワークを使用して日々の業務を行い、重要な情報へアクセスしている業界では、企業データを総体的資産の重要な部分であると見なしています。総保有コスト (Total Cost of Ownership: TCO)、投資利益率 (Return on Investment: ROI)、サービスの品質 (Quality of Service: QoS) などの用語や評価指標は日常的なビジネス用語として用いられるようになっています。各種の業界が、計画およびプロセス管理コストの一環として、これらの評価指標を用いてデータ保全性や可用性などを算出しています。電子商取引などの業界では、データの可用性と信頼性は、成功と失敗の違いを意味します。

11.2. セキュリティーの標準化

企業はどの業界でも、米国医師会 (AMA: American Medical Association)、米国電気電子学会 (IEEE: Institute of Electrical and Electronics Engineers) などの標準化推進団体が作成する規制やルールに従っています。情報セキュリティーにも同じことが言えます。多くのセキュリティーコンサルタントやベンダーが **機密性 (Confidentiality)、保全性 (Integrity)、可用性 (Availability)** の頭文字をとった CIA として知られる標準セキュリティーモデルを採用しています。この 3 階層モデルは、機密情報のリスク評価やセキュリティー方針の確立において、一般的に採用されているモデルです。以下でこの CIA モデルを説明します。

- 機密性 - 機密情報は、事前に定義された個人だけが利用できるようにする必要があります。許可されていない情報の送信や使用は、制限する必要があります。たとえば、情報に機密性があれば、権限のない個人が顧客情報や財務情報を悪意のある目的 (ID 盗難やクレジット詐欺など) で入手できません。

- 保全性 - 情報は、改ざんして不完全または不正確なものにすべきではありません。承認されていないユーザーが、機密情報を変更したり破壊したりする機能を使用できないように制限する必要があります。

- 可用性 - 情報は、認証されたユーザーが必要な時にいつでもアクセスできるようにする必要があります。可用性は、合意した頻度とタイミングで情報を入手できることを保証します。これは、パーセンテージで表されることが多く、ネットワークサービスプロバイダーやその企業顧客が使用するサービスレベルアグリーメント (SLA) で正式に合意となります。

11.3. 暗号化ソフトウェアおよび認定

Red Hat Enterprise Linux は、業界のベストプラクティスに従い、**FIPS 140-2**、**Common Criteria** (CC) などのセキュリティー認証を受けています。

ナレッジベースの記事RHEL 8 core crypto componentsでは、Red Hat Enterprise Linux 8 コア暗号化コンポーネントの概要 (どのコンポーネントか選択されているか、どのように選択されているか、オペレーティングシステムにどのように統合されているかどうか、ハードウェアセキュリティーモジュールおよびスマートカードにどのように対応しているか、および暗号化認証がどのように適用されているか) を説明します。

11.4. セキュリティーコントロール

多くの場合、コンピューターセキュリティーは、一般に **コントロール** と呼ばれる 3 つの主要カテゴリーに分類されます。

- 物理的
- 技術的
- 管理的

この 3 つのカテゴリーは、セキュリティーの適切な実施における主な目的を定義するものです。このコントロールには、コントロールと、その実装方法を詳細化するサブカテゴリーがあります。

11.4.1. 物理的コントロール

物理的コントロールは、機密資料への非認証アクセスの抑止または防止のために、明確な構造でセキュリティー対策を実施します。物理的コントロールの例は以下のとおりです。

- 有線監視カメラ
- 動作または温度の感知アラームシステム
- 警備員
- 写真付き身分証明書
- 施錠された、デッドボルト付きのスチールドア
- バイオメトリック (本人確認を行うための指紋、声、顔、虹彩、筆跡などの自動認識方法が含まれます)

11.4.2. 技術的コントロール

技術的コントロールでは、物理的な構造物やネットワークにおける機密データのアクセスや使用を制御する基盤となる技術を使用します。技術的コントロールは広範囲に及び、以下のような技術も含まれます。

- 暗号化
- スマートカード
- ネットワーク認証
- アクセス制御リスト (ACL)
- ファイルの完全性監査ソフトウェア

11.4.3. 管理的コントロール

管理的コントロールは、セキュリティーの人的要素を定義します。これは組織内のあらゆるレベルの職員や社員に関連するもので、誰がどのリソースや情報にアクセスするかを、次のような手段で決定します。

- トレーニングおよび認識の向上
- 災害準備および復旧計画
- 人員採用と分離の戦略
- 人員登録とアカウンティング

11.5. 脆弱性のアセスメント

時間やリソースがあり、その気になれば、攻撃者はほとんどすべてのシステムに侵入できます。現在利用できるセキュリティーの手順と技術をすべて駆使しても、すべてのシステムを侵入から完全に保護できる訳ではありません。ルーターは、インターネットへのセキュアなゲートウェイを提供します。ファイアウォールは、ネットワークの境界を保護します。仮想プライベートネットワーク (VPN) では、データが、暗号化されているストリームで安全に通過できます。侵入検知システムは、悪意のある活動を警告します。しかし、これらの技術が成功するかどうかは、以下のような数多くの要因によって決まります。

- 技術の設定、監視、および保守を行うスタッフの専門知識
- サービスとカーネルのパッチ、および更新を迅速かつ効率的に行う能力
- ネットワーク上での警戒を常に怠らない担当者の能力

データシステムと各種技術が動的であることを考えると、企業リソースを保護するタスクは極めて複雑になる可能性もあります。この複雑さゆえに、使用するすべてのシステムの専門家を見つけることは、多くの場合困難になります。情報セキュリティーの多くの分野によく精通している人材を確保することはできても、多くの分野を専門とするスタッフを確保することは容易ではありません。これは、情報セキュリティーの各専門分野で、継続的な注意と重点が必要となるためです。情報セキュリティーは、常に変化しています。

脆弱性アセスメントは、お使いのネットワークとシステムのセキュリティーに関する内部監査です。このアセスメントの結果により、ネットワークの機密性、完全性、および可用性の状態が明らかになります。通常、脆弱性アセスメントは、対象システムとリソースに関する重要なデータを収集する調査フェーズから開始します。その後システム準備フェーズとなります。基本的にこのフェーズでは、対象を絞り、すべての既知の脆弱性を調べます。準備フェーズが終わると報告フェーズになります。ここでは、調査結果が高中低のカテゴリーに分類され、対象のセキュリティーを向上させる (または脆弱性のリスクを軽減する) 方法が話し合われます。

たとえば、自宅の脆弱性アセスメントを実施することを想定してみましょう。まずは自宅のドアを点検し、各ドアが閉まっていて、かつ施錠されていることを確認します。また、すべての窓が完全に閉まっていて鍵が閉まっていることも確認します。これと同じ概念が、システム、ネットワーク、および電子データにも適用されます。悪意のあるユーザーはデータを盗んで、破壊します。悪意のあるユーザーが使用するツール、思考、動機に注目すると、彼らの行動にすばやく反応することが可能になります。

11.5.1. アセスメントとテストの定義

脆弱性アセスメントは、**外部からの視点** と **内部からの視点** の 2 種類に分類できます。

外部からの視点で脆弱性アセスメントを実施する場合は、外部からシステムに攻撃を試みます。会社を

外から見ることで、クラッカーの視点に立つことができます。一般にルーティング可能な IP アドレス、**DMZ** にあるシステム、ファイアウォールの外部インターフェイスなど、クラッカーが目を付けるものに着目します。DMZ は非武装地帯 (demilitarized zone) を表し、企業のプライベート LAN などの信頼できる内部ネットワークと、公的なインターネットなどの信頼できない外部ネットワークの間にあるコンピューターまたは小さなサブネットワークに相当します。通常、DMZ には Web (HTTP) サーバー、FTP サーバー、SMTP (e-mail) サーバー、DNS サーバーなど、インターネットのトラフィックにアクセスできるデバイスが含まれます。

内部からの視点で脆弱性アセスメントを実施する場合、実行者は内部関係者であり、信頼されるステータスにあることから、有利な立場になります。内部からの視点は、実行者やその同僚がシステムにログオンした時点で得られるものです。プリントサーバー、ファイルサーバー、データベースなどのリソースを見ることができます。

これら 2 種類の脆弱性アセスメントには大きな違いがあります。社内のユーザーには、部外者が得られない多くの特権が付与されています。多くの組織では、侵入者を締め出すようにセキュリティーが設定されています。しかし、組織内の細かい部分 (部門内ファイアウォール、ユーザーレベルのアクセス制御および内部リソースに対する認証手順など) には、セキュリティー対策がほとんど行われていません。また、一般的にほとんどのシステムは社内にあるため、内部からの方がより多くのリソースを確認できます。いったん社外に移動すると、ステータスは信頼されない状態になります。通常、外部から利用できるシステムやリソースは、非常に限られたものになります。

脆弱性アセスメントと **侵入テスト** の違いを考えてみましょう。脆弱性アセスメントを、侵入テストの第一歩と捉えてください。このアセスメントで得られる情報は、その後のテストで使用します。アセスメントは抜け穴や潜在的な脆弱性を検査する目的で行われるのに対し、侵入テストでは調査結果を実際に使用する試みがなされます。

ネットワークインフラストラクチャーのアセスメントは動的なプロセスです。セキュリティー (情報セキュリティーおよび物理的なセキュリティー) は動的なものです。アセスメントを実施することで概要が明らかになり、誤検出 (False positives) および検出漏れ (False negatives) が示される場合があります。誤検出は、実際には存在しない脆弱性をツールが検出することを指します。検出漏れは、実際の脆弱性が検出されないことを指します。

セキュリティー管理者の力量は、使用するツールとその管理者が有する知識で決まります。現在使用できるアセスメントツールのいずれかを選び、それらをシステムに対して実行すると、ほぼ間違いなく誤検出がいくつか見つかります。プログラム障害でもユーザーエラーでも、結果は同じです。ツールは、誤検出することもあれば、さらに悪い場合は、検出漏れをすることもあります。

脆弱性アセスメントと侵入テストの違いが定義されたところで、新たなベストプラクティスの一環として侵入テストを実施する前に、アセスメントの結果を注意深く確認し、検討してみましょう。

警告

実稼働システムで脆弱性を悪用する試みを行わないでください。システムおよびネットワークの生産性ならびに効率に悪影響を与える可能性があります。

脆弱性アセスメントの実施には、以下のような利点があります。

- 情報セキュリティーに事前にフォーカスできる
- クラッカーが発見する前に潜在的な不正使用を発見できる

- システムを最新の状態に維持し、パッチを適用できる
- スタッフの成長と専門知識の開発を促す
- 経済的な損失や否定的な評判を減らす

11.5.2. 脆弱性評価に関する方法論の確立

脆弱性アセスメントの方法論が確立されれば、脆弱性アセスメント用のツール選択に役立ちます。現時点では、事前定義の方法論や業界で承認された方法論はありませんが、一般常識やベストプラクティスを適切なガイドとして活用できます。

ターゲットとは何を指していますか ?1 台のサーバー、またはネットワーク全体およびネットワーク内にあるすべてのサーバーを確認しますか ?会社外ですか？それとも内部ですか？ この質問に対する回答は、選択したツールだけでなく、そのツールの使用方法を決定する際に重要です。

方法論の確立の詳細は、以下の Web サイトを参照してください。

- https://www.owasp.org/ – **The Open Web Application Security Project**

11.5.3. 脆弱性アセスメントのツール

アセスメントは、情報収集ツールを使用することから始まります。ネットワーク全体を評価する際は、最初にレイアウトを描いて、稼働しているホストを把握します。ホストの場所を確認したら、それぞれのホストを個別に検査します。各ホストにフォーカスするには別のツールセットが必要になります。どのツールを使用すべきかを知っておくことは、脆弱性の発見において最も重要なステップになる可能性があります。

以下で、利用可能なツールを一部紹介します。

- **Nmap** は、ホストシステムを見つけて、そのシステムでポートを開くことができる一般的なツールです。**AppStream** リポジトリーから **Nmap** をインストールするには、**root** で **yum install nmap** コマンドを実行します。詳細は **nmap(1)** の man ページを参照してください。
- **oscap** コマンドラインユーティリティー、**scap-workbench** グラフィカルユーティリティーなどの **OpenSCAP** スイートのツールは、完全に自動化されたコンプライアンス監査を提供します。詳細は セキュリティーコンプライアンスおよび脆弱性スキャンの開始 を参照してください。
- **AIDE** (Advanced Intrusion Detection Environment) は、システムのファイルのデータベースを作成し、そのデータベースを使用してファイルの整合性を確保し、システムの侵入を検出します。詳細は AIDE で整合性のチェック を参照してください。

11.6. セキュリティーへの脅威

11.6.1. ネットワークセキュリティーへの脅威

ネットワークの以下の要素を設定する際に不適当なプラクティスが行われると、攻撃のリスクが増大します。

セキュリティーが十分ではないアーキテクチャー

間違った設定のネットワークは、未承認ユーザーの主要なエントリーポイントになります。信頼に基づいたオープンなローカルネットワークを、安全性が非常に低いインターネットに対して無防備な状態にしておくことは、犯罪の多発地区でドアを半開きにしておくようなものです。すぐに何かが起きること

はないかもしれませんが、**いずれ**、誰かが、このチャンスを悪用するでしょう。

ブロードキャストネットワーク

システム管理者は、セキュリティー計画においてネットワーキングハードウェアの重要性を見落としがちです。ハブやルーターなどの単純なハードウェアは、ブロードキャストやノンスイッチの仕組みに基づいています。つまり、あるノードがネットワークを介して受信ノードにデータを送信するときは常に、受信ノードがデータを受信して処理するまで、ハブやルーターがデータパケットのブロードキャストを送信します。この方式は、外部侵入者やローカルホストの未認証ユーザーが仕掛けるアドレス解決プロトコル (**ARP**) およびメディアアクセスコントロール (**MAC**) アドレスの偽装に対して最も脆弱です。

集中化サーバー

ネットワーキングのもうひとつの落とし穴は、集中化されたコンピューティングの使用にあります。多くの企業では、一般的なコスト削減手段として、すべてのサービスを 1 台の強力なマシンに統合しています。集中化は、複数サーバーを設定するよりも管理が簡単で、コストを大幅に削減できるので便利です。ただし、集中化されたサーバーはネットワークにおける単一障害点となります。中央のサーバーが攻撃されると、ネットワークが完全に使用できなくなるか、データの不正操作や盗難が起きやすくなる可能性があります。このような場合は、中央サーバーがネットワーク全体へのアクセスを許可することになります。

11.6.2. サーバーセキュリティーへの脅威

サーバーには組織の重要情報が数多く含まれることが多いため、サーバーのセキュリティーは、ネットワークのセキュリティーと同様に重要です。サーバーが攻撃されると、クラッカーが意のままにすべてのコンテンツを盗んだり、不正に操作したりできるようになる可能性があります。以下のセクションでは、主要な問題の一部を詳述します。

未使用のサービスと開かれたポート

Red Hat Enterprise Linux 8 のフルインストールを行うと、アプリケーションとライブラリーのパッケージが 1000 個以上含まれます。ただし、サーバー管理者が、ディストリビューションに含まれるすべての個別パッケージをインストールすることはほとんどありません。代わりに、複数のサーバーアプリケーションを含むパッケージのベースインストールを行います。

システム管理者は、インストールに含まれるプログラムに注意を向けずにオペレーティングシステムをインストールしてしまうことがよくあります。これにより、不要なサービスがインストールされ、デフォルト設定でオンになっていることで、問題が発生する場合があります。つまり、管理者が気が付かないところで、Telnet、DHCP、DNS などの不要なサービスがサーバーやワークステーションで実行し、その結果、サーバーへの不要なトラフィックが発生したり、クラッカーがシステムのパスを悪用できてしまう可能性があります。

パッチが適用されないサービス

デフォルトのインストールに含まれるほとんどのサーバーアプリケーションは、ソフトウェアの細部まで徹底的にテストされており、堅牢な作りになっています。何年も実稼働環境で使用される中で、そのコードは入念に改良され、数多くのバグが発見されて修正されてきました。

しかし、完璧なソフトウェアというものはなく、改良の余地は常にあります。または、比較的新しいソフトウェアは、実稼働環境に導入されてから日が浅く、他のサーバーソフトウェアほど普及していないこともあるため、厳密なテストが期待通りに行われていない状況も少なくありません。

開発者やシステム管理者が、サーバーアプリケーションで悪用される可能性のあるバグを発見することも多々あり、Bugtraq メーリングリスト (http://www.securityfocus.com)、Computer Emergency Response Team (CERT) Web サイト (http://www.cert.org) などで、バグ追跡やセキュリティー関連の Web サイトに関連する情報が公開されています。このような情報発信は、コミュニティーにセキュリティーの脆弱性を警告する効果的な方法ではありますが、システムに速やかにパッチを当てるかどうか

は個々のシステム管理者が決定します。クラッカーも、パッチが適用されていないシステムがあればクラッキングできるように、脆弱性トラッキングサービスにアクセスし、関連情報を利用できることを考慮すると、速やかな対応がとりわけ重要になります。優れたシステム管理を行うには、警戒を怠らず、バグ追跡を絶えず行い、適切なシステム保守を実行して、よりセキュアなコンピューティング環境を維持することが求められます。

管理における不注意

管理者がシステムにパッチを当てないことが、サーバーのセキュリティーに対する最大の脅威の 1 つになります。これは、管理者の経験の少なさだけでなく、管理者の過信やモチベーションの低さなども原因となります。

管理者が、サーバーやワークステーションにパッチを当てることを忘れたり、システムのカーネルやネットワーク通信のログメッセージを見落とす場合もあります。その他にも、よく起こるケースとして、サービスのデフォルトパスワードや鍵を変更しないまま放置しておくことが挙げられます。たとえば、データベースにはデフォルトの管理パスワードが設定されているものがありますが、ここでは、システム管理者がインストール後すぐにデフォルトパスワードを変更することを、データベース開発者は想定しています。しかし、データベース管理者がパスワードを変更することを忘れると、クラッカーの経験が浅くても、周知のデフォルトパスワードを使用してデータベースの管理者権限を得ることができます。この他に、管理者の不注意によりサーバーが危険にさらされる場合もあります。

本質的に安全ではないサービス

どんなに注意深い組織であっても、選択するネットワークサービスが本質的に安全でない限り、攻撃を受けやすくなります。たとえば、多くのサービスは、信頼できるネットワークでの使用を想定して開発されますが、このサービスが (本質的に信頼できない) インターネットで利用可能になる時点で、この仮定は成立しなくなります。

安全ではないネットワークサービスの例として、暗号化されていないユーザー名とパスワードを認証時に要求するサービスが挙げられます。具体例としては、Telnet や FTP の 2 つがあげられます。パケット盗聴ソフトウェアがリモートユーザーとこのようなサービスの間のトラフィックを監視していれば、ユーザー名とパスワードは簡単に傍受される可能性があります。

また、基本的にこのようなサービスはセキュリティー業界で **中間者** 攻撃と呼ばれる攻撃の被害者になりやすくなります。この種の攻撃では、クラッカーが、ネットワーク上でクラッキングしたネームサーバーを操って、目標のサーバーではなくクラッカーのマシンを指定して、ネットワークトラフィックをリダイレクトします。誰かがサーバーへのリモートセッションを開くと、攻撃者のマシンがリモートサービスと無防備なユーザーとの間に存在する目に見えないパイプとして機能し、この間を流れる情報を取り込みます。このようにして、クラッカーはサーバーやユーザーに気付かれることなく、管理パスワードや生データを収集できるようになります。

安全ではないサービスの例としては、他にも NFS、NIS などのネットワークファイルシステムおよび情報サービスが挙げられます。このサービスは、LAN 利用を目的として開発されましたが、(リモートユーザー用の) WAN も対象に含まれるように拡張されました。NFS では、クラッカーによる NFS 共有のマウントやそこに格納されているものへのアクセスを防ぐ認証やセキュリティーの仕組みがデフォルトで設定されていません。NIS も、プレーンテキストの ASCII または DBM (ASCII から派生) データベースに、パスワードやファイルパーミッションなど、ネットワーク上の全コンピューターへの周知が必要となる重要な情報を保持しています。クラッカーがこのデータベースのアクセス権を取得すると、管理者のアカウントを含む、ネットワークのすべてのユーザーアカウントにアクセスできるようになります。

Red Hat Enterprise Linux 8 では、デフォルトでは、上記のサービスがすべて無効になっています。ただし、管理者は、このようなサービスを使用しないといけない場合があるため、注意して設定することが重要となります。

11.6.3. ワークステーションおよび家庭用 PC のセキュリティーに対する脅威

ワークステーションや自宅用 PC はネットワークやサーバーほど攻撃される可能性はありませんが、クレジットカード情報などの機密データが含まれるため、システムクラッカーの標的になります。知らない間にワークステーションが攻撃者により選択され、一連の攻撃でボットマシンとして使用される可能性もあります。このため、ユーザーはワークステーションの脆弱性を理解しておくと、オペレーティングシステムの再インストールや、深刻な場合はデータ盗難からの回復といった問題から免れることができます。

不適切なパスワード

攻撃者が最も簡単にシステムへのアクセスを得る方法の 1 つとして、パスワードが適切でないことが挙げられます。

脆弱なクライアントアプリケーション

管理者がサーバーに十分な安全対策を施し、パッチを当てている場合でも、リモートユーザーによるアクセスが安全であるわけではありません。たとえば、サーバーが公開ネットワーク上で Telnet や FTP のサービスを提供している場合、攻撃者はネットワークを通過するプレーンテキストのユーザー名とパスワードを取り込み、アカウント情報を使用してリモートユーザーのワークステーションにアクセスすることが可能です。

SSH などのセキュアなプロトコルを使用している場合であっても、クライアントアプリケーションを定期的に更新していないと、リモートユーザーは特定の攻撃を受けやすくなる可能性があります。たとえば、SSH プロトコルのバージョン 1 のクライアントは、悪意のある SSH サーバーからの X 転送攻撃に対して脆弱です。クライアントがサーバーに接続すると、攻撃者はネットワーク上でクライアントによるキー入力やマウス操作をひそかに収集できます。この問題は SSH プロトコルのバージョン 2 で修正されましたが、ユーザーはどのアプリケーションにこのような脆弱性があるかを追跡し、必要に応じてアプリケーションを更新する必要があります。

11.7. 一般的な不正使用と攻撃

以下の表では、侵入者が組織のネットワークリソースにアクセスするために使用する最も一般的な不正使用とエントリーポイントの例を挙げて詳しく説明します。この一般的な不正使用では、それがどのように実行され、管理者がその攻撃からネットワークをどのように適切に保護できるかを理解していることが重要になります。

表11.1 一般的な不正使用

不正使用	説明	備考

不正使用	説明	備考
空またはデフォルトのパスワード	管理パスワードを空白のままにしたり、製品ベンダーが設定したデフォルトのパスワードをそのまま使用します。これは、ルーターやファイアウォールなどのハードウェアで最もよく見られますが、Linux で実行するサービスにはデフォルトの管理者パスワードが指定されているものがあります (ただし Red Hat Enterprise Linux 8 には含まれません)。	一般的に、ルーター、ファイアウォール、VPN、ネットワーク接続ストレージ (NAS) の機器など、ネットワークハードウェアに関連するものです。 多数のレガシーオペレーティングシステム、特にサービスをバンドルしたオペレーティングシステム (UNIX や Windows など) でよく見られます。 管理者が急いで特権ユーザーアカウントを作成したためにパスワードが空白のままになっていることがありますが、このような空白のパスワードは、このアカウントを発見した悪意のあるユーザーが利用できる絶好のエントリーポイントとなります。
デフォルトの共有鍵	セキュアなサービスでは、開発や評価テスト向けにデフォルトのセキュリティー鍵がパッケージ化されていることがあります。この鍵を変更せずにインターネットの実稼働環境に置いた場合は、同じデフォルトの鍵を持つ **すべての** ユーザーがその共有鍵のリソースや、そこにあるすべての機密情報にアクセスできるようになります。	無線アクセスポイントや、事前設定済みでセキュアなサーバー機器に最も多く見られます。
IP スプーフィング	リモートマシンがローカルネットワークのノードのように動作し、サーバーに脆弱性を見つけるとバックドアプログラムまたはトロイの木馬をインストールして、ネットワークリソース全体へのコントロールを得ようとします。	スプーフィングは、攻撃者が標的となるシステムへの接続を調整するのに、TCP/IP シーケンス番号を予測しなければならないため、かなり難しくなりますが、クラッカーの脆弱性の攻撃を支援する利用可能なツールがいくつかあります。 標的となるシステムで実行している source-based 認証技術を使用するサービス (**rsh**、**telnet**、FTP など) により異なりますが、このようなサービスは、**ssh**、または SSL/TLS で使用される PKI などの形式の暗号化認証と比較すると推奨されません。

不正使用	説明	備考
盗聴	2 つのノード間の接続を盗聴することにより、ネットワーク上のアクティブなノード間を行き交うデータを収集します。	この種類の攻撃には大抵、Telnet、FTP、HTTP 転送などのプレーンテキストの転送プロトコルが使用されます。 リモートの攻撃者がこのような攻撃を仕掛けるには、LAN で、攻撃するシステムへのアクセス権が必要になります。通常、クラッカーは、LAN 上にあるシステムを危険にさらすためにアクティブ攻撃 (IP スプーフィングや中間者攻撃など) を仕掛けます。 パスワードのなりすましに対する防護策としては、暗号化鍵交換、ワンタイムパスワード、または暗号化された認証によるサービス使用が挙げられます。通信中は強力な暗号化を実施することを推奨します。

不正使用	説明	備考
サービスの脆弱性	攻撃者はインターネットで実行しているサービスの欠陥や抜け穴を見つけます。攻撃者がこの脆弱性を利用する場合は、システム全体と格納されているデータを攻撃するだけでなく、ネットワーク上の他のシステムも攻撃する可能性があります。	CGI などの HTTP ベースのサービスは、リモートのコマンド実行やインタラクティブなシェルアクセスに対しても脆弱です。HTTP サービスが nobody などの権限のないユーザーとして実行している場合でも、設定ファイルやネットワークマップなどの情報が読み取られる可能性があります。または、攻撃者がサービス拒否攻撃を開始して、システムのリソースを浪費させたり、他のユーザーが利用できないようにする可能性もあります。 開発時およびテスト時には気が付かない脆弱性がサービスに含まれることがあります。(アプリケーションのメモリーバッファー領域をあふれさせ、任意のコマンドを実行できるようなインタラクティブなコマンドプロンプトを攻撃者に提供するように、攻撃者が任意の値を使用してサービスをクラッシュさせる**バッファーオーバーフロー**などの) 脆弱性は、完全な管理コントロールを攻撃者に与えるものとなる可能性があります。 管理者は、root 権限でサービスが実行されないようにし、ベンダー、または CERT、CVE などのセキュリティー組織がアプリケーション用のパッチやエラータ更新を提供していないかを常に注意する必要があります。

不正使用	説明	備考
アプリケーションの脆弱性	攻撃者は、デスクトップやワークステーションのアプリケーション (電子メールクライアントなど) に欠陥を見つけ出し、任意のコードを実行したり、将来のシステム侵害のためにトロイの木馬を移植したり、システムを破壊したりします。攻撃を受けたワークステーションがネットワークの残りの部分に対して管理特権を持っている場合は、さらなる不正使用が起こる可能性があります。	ワークステーションとデスクトップは、ユーザーが侵害を防いだり検知するための専門知識や経験を持たないため、不正使用の対象になりやすくなります。認証されていないソフトウェアをインストールしたり、要求していないメールの添付ファイルを開く際には、それに伴うリスクについて個々に通知することが必須です。 電子メールクライアントソフトウェアが添付ファイルを自動的に開いたり、実行したりしないようにするといった、予防手段を取ることが可能です。さらに、Red Hat Network や他のシステム管理サービスなどからワークステーションのソフトウェアを自動更新することにより、マルチシートのセキュリティーデプロイメントの負担を軽減できます。
サービス拒否攻撃 (DoS: Denial of Service)	単独の攻撃者または攻撃者のグループは、目標のホスト (サーバー、ルーター、ワークステーションのいずれか) に認証されていないパケットを送り、組織のネットワークまたはサーバーのリソースに対して攻撃を仕掛けます。これにより、正当なユーザーがリソースを使用できなくなります。	米国で最も多く報告された DOS の問題は、2000 年に発生しました。この時、通信量が非常に多い民間および政府のサイトが一部が利用できなくなりました。**ゾンビ (zombie)** や、リダイレクトされたブロードキャストノードとして動作する高帯域幅接続を有し、セキュリティー侵害された複数のシステムを使用して、調整された ping フラッド攻撃が行われたためです。 通常ソースパケットは、真の攻撃元を調査するのが難しくなるよう、偽装 (または再ブロードキャスト) されています。 **nftables** パケットフィルタリングフレームワークを使用したイングレスフィルタリング (RFC 2267) や、**snort** などのネットワーク侵入検知システムにおける進歩は、管理者が分散型サービス拒否攻撃を追跡し、これを防止するのに役立っています。

第12章 インストール時の RHEL の保護

セキュリティーへの対応は、Red Hat Enterprise Linux をインストールする前にすでに始まっています。最初からシステムのセキュリティーを設定することで、追加のセキュリティー設定を実装することがより簡単になります。

12.1. BIOS および UEFI のセキュリティー

BIOS (もしくは BIOS に相当するもの) およびブートローダーをパスワードで保護することで、システムに物理的にアクセス可能な未承認ユーザーがリムーバブルメディアを使用して起動したり、シングルユーザーモードで root 権限を取得することを防ぐことができます。このような攻撃に対するセキュリティー対策は、ワークステーションの情報の機密性とマシンの場所によって異なります。

たとえば、見本市で使用されていて機密情報を含んでいないマシンでは、このような攻撃を防ぐことが重要ではないかもしれません。しかし、同じ見本市で、企業ネットワークに対して暗号化されていない SSH 秘密鍵のある従業員のノートパソコンが、誰の監視下にもなく置かれていた場合は、重大なセキュリティー侵害につながり、その影響は企業全体に及ぶ可能性があります。

一方で、ワークステーションが権限のあるユーザーもしくは信頼できるユーザーのみがアクセスできる場所に置かれてるのであれば、BIOS もしくはブートローダーの安全確保は必要ない可能性もあります。

12.1.1. BIOS パスワード

コンピューターの BIOS をパスワードで保護する主な 2 つの理由を以下に示します。[1]:

1. **BIOS 設定の変更を防止する** - 侵入者が BIOS にアクセスした場合は、CD-ROM やフラッシュドライブから起動するように設定できます。このようにすると、侵入者がレスキューモードやシングルユーザーモードに入ることが可能になり、システムで任意のプロセスを開始したり、機密性の高いデータをコピーできるようになってしまいます。

2. **システムの起動を防止する** - BIOS の中には起動プロセスをパスワードで保護できるものもあります。これを有効にすると、攻撃者は BIOS がブートローダーを開始する前にパスワード入力を求められます。

BIOS パスワードの設定方法はコンピューターメーカーで異なるため、具体的な方法はコンピューターのマニュアルを参照してください。

BIOS パスワードを忘れた場合は、マザーボードのジャンパーでリセットするか、CMOS バッテリーを外します。このため、可能な場合はコンピューターのケースをロックすることが推奨されます。ただし、CMOS バッテリーを外す前にコンピューターもしくはマザーボードのマニュアルを参照してください。

12.1.2. 非 BIOS ベースシステムのセキュリティー

その他のシステムやアーキテクチャーでは、異なるプログラムを使用して x86 システムの BIOS とほぼ同等の低レベルのタスクを実行します。**UEFI** (**Unified Extensible Firmware Interface**) シェルなどがこの例になります。

BIOS のようなプログラムをパスワード保護する方法は、メーカーにお問い合わせください。

12.2. ディスクのパーティション設定

Red Hat は、**/boot**、/、**/home**、**/tmp**、および **/var/tmp/** の各ディレクトリーに別々のパーティションを作成することを推奨します。

/boot

このパーティションは、システムの起動時にシステムが最初に読み込むパーティションです。Red Hat Enterprise Linux 8 でシステムを起動するのに使用するブートローダーとカーネルイメージはこのパーティションに保存されます。このパーティションは暗号化しないでください。このパーティションが /` に含まれており、そのパーティションが暗号化されているなどの理由で利用できなくなると、システムを起動できなくなります。

/home

ユーザーデータ (**/home**) が別のパーティションではなく / に保存されていると、このパーティションが満杯になり、オペレーティングシステムが不安定になる可能性があります。また、システムを、Red Hat Enterprise Linux 8 の次のバージョンにアップグレードする際に、**/home** パーティションにデータを保存できると、このデータはインストール時に上書きされないため、アップグレードが非常に簡単になります。root パーティション (/) が破損すると、データが完全に失われます。したがって、パーティションを分けることが、データ損失に対する保護につながります。また、このパーティションを、頻繁にバックアップを作成する対象にすることも可能です。

/tmp および /var/tmp/

/tmp ディレクトリーおよび **/var/tmp/** ディレクトリーは、どちらも長期保存の必要がないデータを保存するのに使用されます。しかし、このいずれかのディレクトリーでデータがあふれると、ストレージ領域がすべて使用されてしまう可能性があります。このディレクトリーは / に置かれているため、こうした状態が発生すると、システムが不安定になり、クラッシュする可能性があります。そのため、このディレクトリーは個別のパーティションに移動することが推奨されます。

注記

インストールプロセス時に、パーティションを暗号化するオプションがあります。パスフレーズを入力する必要があります。これは、パーティションのデータを保護するのに使用されるバルク暗号鍵を解除する鍵として使用されます。

12.3. インストールプロセス時のネットワーク接続の制限

Red Hat Enterprise Linux 8 をインストールする際に使用するインストールメディアは、特定のタイミングで作成されたスナップショットです。そのため、セキュリティー修正が最新のものではなく、このインストールメディアで設定するシステムが公開されてから修正された特定の問題に対して安全性に欠ける場合があります。

脆弱性が含まれる可能性のあるオペレーティングシステムをインストールする場合には、必ず、公開レベルを、必要最小限のネットワークゾーンに限定してください。最も安全な選択肢は、インストールプロセス時にマシンをネットワークから切断した状態にするネットワークなしのゾーンです。インターネット接続からのリスクが最も高く、一方で LAN またはイントラネット接続で十分な場合もあります。セキュリティーのベストプラクティスに従い、ネットワークから Red Hat Enterprise Linux 8 をインストールする場合は、お使いのリポジトリーに最も近いゾーンを選択するようにしてください。

12.4. 必要なパッケージの最小限のインストール

コンピューターの各ソフトウェアには脆弱性が潜んでいる可能性があるため、実際に使用するパッケージのみをインストールすることがベストプラクティスになります。インストールを DVD から行う場合は、インストールしたいパッケージのみを選択するようにします。その他のパッケージが必要になる場合は、後でいつでもシステムに追加できます。

12.5. インストール後の手順

以下は、Red Hat Enterprise Linux 8 のインストール直後に実行する必要があるセキュリティー関連の手順です。

- システムを更新します。root で以下のコマンドを実行します。

```
# yum update
```

- ファイアウォールサービスの **firewalld** は、Red Hat Enterprise Linux のインストールで自動的に有効になっていますが、キックスタート設定などで明示的に無効となっている場合もあります。このような場合は、ファイアウォールを再度有効にすることが推奨されます。
firewalld を開始するには、root で次のコマンドを実行します。

```
# systemctl start firewalld
# systemctl enable firewalld
```

- セキュリティーを強化するために、不要なサービスは無効にしてください。たとえば、使用中のコンピューターにプリンターがインストールされていなければ、以下のコマンドを使用して **cups** サービスを無効にします。

```
# systemctl disable cups
```

アクティブなサービスを確認するには、次のコマンドを実行します。

```
$ systemctl list-units | grep service
```

---

[1] システム BIOS はメーカーによって異なるため、いずれかのタイプのパスワード保護のみをサポートするものもあれば、いずれのタイプのパスワード保護もサポートしないものもあります。

第13章 システム全体の暗号化ポリシーの使用

システム全体の暗号化ポリシーは、コア暗号化サブシステムを設定するシステムコンポーネントで、TLS、IPsec、SSH、DNSSec、および Kerberos の各プロトコルに対応します。これにより、管理者が選択できる小規模セットのポリシーを提供します。

13.1. システム全体の暗号化ポリシー

システム全体のポリシーを設定すると、RHEL のアプリケーションはそのポリシーに従い、ポリシーを満たしていないアルゴリズムやプロトコルを使用するように明示的に要求されない限り、その使用を拒否します。つまり、システムが提供した設定で実行する際に、デフォルトのアプリケーションの挙動にポリシーを適用しますが、必要な場合は上書きできます。

RHEL 8 には、以下の定義済みポリシーが含まれています。

DEFAULT	デフォルトのシステム全体の暗号化ポリシーレベルで、現在の脅威モデルに対して安全なものです。TLS プロトコルの 1.2 と 1.3、IKEv2 プロトコル、および SSH2 プロトコルが使用できます。RSA 鍵と Diffie-Hellman パラメーターは長さが 2048 ビット以上であれば許容されます。
LEGACY	このポリシーは、Red Hat Enterprise Linux 5 以前のリリースとの互換性を最大化しますが、攻撃領域が大きくなるため脆弱になります。**DEFAULT** レベルでのアルゴリズムとプロトコルに加えて、TLS プロトコル 1.0 および 1.1 を許可します。アルゴリズム DSA、3DES、および RC4 が許可され、RSA 鍵と Diffie-Hellman パラメーターの長さが 1023 ビット以上であれば許容されます。
FUTURE	近い将来の攻撃に耐えられると考えられている保守的なセキュリティーレベルです。このレベルは、署名アルゴリズムに SHA-1 の使用を許可しません。TLS プロトコルの 1.2 と 1.3、IKEv2 プロトコル、および SSH2 プロトコルが使用できます。RSA 鍵と Diffie-Hellman パラメーターは、ビット長が 3072 以上だと許可されます。
FIPS	FIPS140-2 要件に準拠するポリシールールです。これは、**fips-mode-setup** ツールの内部で使用され、RHEL システムを FIPS モードに切り替えます。

Red Hat は常に、レガシーポリシーの使用時以外、全ライブラリーにセキュアなデフォルト値が提供されるように、全ポリシーレベルを調節します。レガシープロファイルではセキュアなデフォルト値が提供されませんが、簡単に悪用できるアルゴリズムは含まれません。このため、提供されたポリシーで有効なアルゴリズムのセットまたは許容可能な鍵サイズは、Red Hat Enterprise Linux の存続期間中に変更する可能性があります。

このような変更は、新しいセキュリティー標準や新しいセキュリティー調査を反映しています。Red Hat Enterprise Linux の有効期間中、特定のシステムとの相互運用性を確保する必要がある場合は、そのシステムと相互作用するコンポーネントの暗号化ポリシーから除外し、カスタムポリシーを使用して特定のアルゴリズムを再度有効にする必要があります。

重要

カスタマーポータル API の証明書が使用する暗号化鍵は **FUTURE** のシステム全体の暗号化ポリシーが定義する要件を満たさないので、現時点で **redhat-support-tool** ユーティリティーは、このポリシーレベルでは機能しません。

この問題を回避するには、カスタマーポータル API への接続中に **DEFAULT** 暗号化ポリシーを使用します。

注記

ポリシーレベルで許可されていると記載されている特定のアルゴリズムと暗号は、アプリケーションがそれに対応している場合に限り使用できます。

暗号化ポリシーを管理するツール

現在のシステム全体の暗号化ポリシーを表示または変更するには、**update-crypto-policies** ツールを使用します。以下に例を示します。

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

暗号化ポリシーの変更を確実に適用するには、システムを再起動します。

安全ではない暗号スイートおよびプロトコルを削除した、強力な暗号デフォルト

以下の一覧は、Red Hat Enterprise Linux 8 のコア暗号ライブラリーから削除された暗号スイートとプロトコルを示しています。このアプリケーションはソースには存在しないか、ビルド時にサポートを無効にしているため、アプリケーションは使用できません。

- DES (RHEL 7 以降)
- すべてのエクスポートグレードの暗号化スイート (RHEL 7 以降)
- 署名内の MD5 (RHEL 7 以降)
- SSLv2 (RHEL 7 以降)
- SSLv3 (RHEL 8 以降)
- 224 ビットより小さいすべての ECC 曲線 (RHEL 6 以降)
- すべてのバイナリーフィールドの ECC 曲線 (RHEL 6 以降)

すべてのポリシーレベルで無効になっている暗号スイートおよびプロトコル

以下の暗号スイートおよびプロトコルは、すべての暗号化ポリシーレベルで無効になっています。これは、各アプリケーションで明示的に有効にした場合に限り利用可能にできます。

- パラメーターが 1024 ビットより小さい DH
- 鍵のサイズが 1024 ビットより小さい RSA
- Camellia
- ARIA

- SEED
- IDEA
- 完全性のみの暗号スイート
- SHA-384 HMAC を使用した TLS CBC モード暗号化スイート
- AES-CCM8
- TLS 1.3 と互換性がないすべての ECC 曲線 (secp256k1 を含む)
- IKEv1 (RHEL 8 以降)

暗号ポリシーレベルで有効な暗号スイートおよびプロトコル
次の表は、暗号ポリシーの各レベルで有効な暗号化スイートおよびプロトコルを示しています。

	LEGACY	DEFAULT	FIPS	FUTURE
IKEv1	いいえ	いいえ	いいえ	いいえ
3DES	はい	いいえ	いいえ	いいえ
RC4	はい	いいえ	いいえ	いいえ
DH	最低 1024 ビット	最低 2048 ビット	最低 2048 ビット	最低 3072 ビット
RSA	最低 1024 ビット	最低 2048 ビット	最低 2048 ビット	最低 3072 ビット
DSA	はい	いいえ	いいえ	いいえ
TLS v1.0	はい	いいえ	いいえ	いいえ
TLS v1.1	はい	いいえ	いいえ	いいえ
デジタル署名における SHA-1	はい	はい	いいえ	いいえ
CBC モード暗号	はい	はい	はい	任意[a]
256 ビットより小さい鍵を持つ対称暗号	はい	はい	はい	いいえ
証明書における SHA-1 および SHA-224 の署名	はい	はい	はい	いいえ

[a] TLS の CBC 暗号が無効になっています。TLS 以外のシナリオでは、**AES-128-CBC** は無効になっていますが、**AES-256-CBC** が有効になります。**AES-256-CBC** も無効にするには、カスタムのサブポリシーを適用します。

関連情報

- **update-crypto-policies(8)** の man ページ

13.2. システム全体の暗号化ポリシーを、以前のリリースと互換性のあるモードに切り替え

Red Hat Enterprise Linux 8 におけるデフォルトのシステム全体の暗号化ポリシーでは、現在は古くて安全ではないプロトコルは許可されません。Red Hat Enterprise Linux 6 およびそれ以前のリリースとの互換性が必要な場合には、安全でない **LEGACY** ポリシーレベルを利用できます。

警告

LEGACY ポリシーレベルに設定すると、システムおよびアプリケーションの安全性が低下します。

手順

1. システム全体の暗号化ポリシーを **LEGACY** レベルに切り替えるには、**root** で以下のコマンドを実行します。

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

関連情報

- 利用可能な暗号化ポリシーのレベルは、**update-crypto-policies(8)** の man ページを参照してください。
- カスタム暗号化ポリシーの定義については、man ページの **update-crypto-policies (8)** の **Custom Policies** セクションと、**crypto-policies(7)** の **Crypto Policy Definition Format** セクションを参照してください。

13.3. WEB コンソールでシステム全体の暗号化ポリシーを設定する

事前定義されたシステム全体の暗号化ポリシーレベルから選択し、Red Hat Enterprise Linux Web コンソールインターフェイスでそれらを直接切り替えることができます。システムにカスタムポリシーを設定すると、Web コンソールの **Overview** ページと **Change crypto policy** ダイアログウィンドウにポリシーが表示されます。

前提条件

- RHEL 8 Web コンソールがインストールされている。詳細は、Web コンソールのインストールおよび有効化 を参照してください。
- 管理者権限があります。

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. **Overview** ページの **Configuration** カードで、**Crypto policy** の横にある現在のポリシー値をクリックします。
3. **Change crypto policy** ダイアログウィンドウで、使用を開始するポリシーレベルをクリックします。
4. **Apply and reboot** ボタンをクリックします。

検証

- 再度ログインして、**Crypto policy** の値が選択した値に対応していることを確認します。

13.4. FIPS モードへのシステムの切り替え

システム全体の暗号化ポリシーには、連邦情報処理規格 (FIPS) 公開文書 140-2 の要件に準拠した暗号化モジュールのセルフチェックを有効にするポリシーレベルが含まれます。FIPS モードを有効または無効にする **fips-mode-setup** ツールは、内部的に **FIPS** のシステム全体の暗号化ポリシーレベルを使用します。

重要

Red Hat は、後で FIPS モードを有効にするのではなく、FIPS モードを有効にして Red Hat Enterprise Linux 8 をインストールすることを推奨します。インストール時に FIPS モードを有効にすると、システムは FIPS で承認されるアルゴリズムと継続的な監視テストですべてのキーを生成するようになります。

手順

1. システムを FIPS モードに切り替えるには、以下のコマンドを実行します。

```
# fips-mode-setup --enable
Kernel initramdisks are being regenerated. This might take some time.
Setting system policy to FIPS
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
FIPS mode will be enabled.
Please reboot the system for the setting to take effect.
```

2. システムを再起動して、カーネルを FIPS モードに切り替えます。

```
# reboot
```

検証

1. システムが再起動したら、FIPS モードの現在の状態を確認できます。

```
# fips-mode-setup --check
FIPS mode is enabled.
```

関連情報

関連情報

- **FIPS-mode-setup(8)** の man ページ
- FIPS モードが有効な RHEL 8 システムのインストール
- List of RHEL applications using cryptography that is not compliant with FIPS 140-2
- NIST (National Institute of Standards and Technology) の Web サイトの Security Requirements for Cryptographic Modules。

13.5. コンテナーでの FIPS モードの有効化

Federal Information Processing Standard Publication 140-2 (FIPS モード) で義務付けられている暗号化モジュールのセルフチェックの完全なセットを有効にするには、ホストシステムのカーネルが FIPS モードで実行されている必要があります。ホストシステムのバージョンに応じて、コンテナーで FIPS モードを有効にすることが完全に自動で行われるか、コマンドが 1 つだけ必要になります。

注記

コンテナーで **fips-mode-setup** コマンドが正しく機能せず、このシナリオでこのコマンドを使用して FIPS モードを有効にしたり確認することができません。

前提条件

- ホストシステムが FIPS モードである必要があります。

手順

- **RHEL 8.1 および 8.2 を実行しているホストの場合**: 次のコマンドを使用してコンテナー内の FIPS 暗号化ポリシーレベルを設定します。**fips-mode-setup** コマンドを使用するというアドバイスは無視します。

```
$ update-crypto-policies --set FIPS
```

- **RHEL 8.4 以降を実行しているホストの場合**: FIPS モードが有効になっているシステムでは、**podman** ユーティリティーはサポートされているコンテナーで FIPS モードを自動的に有効にします。

関連情報

- FIPS モードへのシステムの切り替え
- FIPS モードが有効な RHEL 8 システムのインストール

13.6. LIST OF RHEL APPLICATIONS USING CRYPTOGRAPHY THAT IS NOT COMPLIANT WITH FIPS 140-2

コア暗号化コンポーネントでは、FIPS 140-2 などの関連する暗号化証明書すべてを渡して RHEL システム全体の暗号化ポリシーにも準拠することが保証されているので、Red Hat はこのコンポーネントからライブラリーを使用することを推奨します。

RHEL 8 コア暗号化コンポーネントの概要、このコンポーネントの選択方法、オペレーティングシステムへの統合方法、ハードウェアセキュリティーモジュールおよびスマートカードのサポート方法、暗号化による認定の適用方法の概要は、RHEL 8 コア暗号化コンポーネント を参照してください。

以下の表に加えて、一部の RHEL 8 Z-stream リリース (例: 8.1.1) では Firefox ブラウザーパッケージが更新され、別の NSS 暗号化ライブラリーが含まれています。このように、Red Hat では、パッチリリースでこのような詳細レベルのコンポーネントをリベースするなどの混乱を回避できればと考えています。そのため、この Firefox パッケージは FIPS 140-2 検証モジュールを使用しません。

表13.1 List of RHEL 8 applications using cryptography that is not compliant with FIPS 140-2

アプリケーション	詳細
FreeRADIUS	MD5 を使用する RADIUS プロトコル
ghostscript	ドキュメントを暗号化および復号化するためのカスタムの cryptogtaphy 実装 (MD5、RC4、SHA-2、AES)
ipxe	TLS の暗号化スタックはコンパイルされていますが、使用されません。
libica	CPACF 命令から RSA や ECDH などのさまざまなアルゴリズムのソフトウェアフォールバック
OVMF (UEFI ファームウェア)、Edk2、shim	完全な暗号スタック (OpenSSL ライブラリーの埋め込みコピー)
perl-Digest-HMAC	HMAC、HMAC-SHA1、HMAC-MD5
perl-Digest-SHA	SHA-1, SHA-224, ...
pidgin	DES、RC4
qatengine	暗号化プリミティブの混在ハードウェアおよびソフトウェア実装 (RSA、EC、DH、AES、...)
samba[a]	AES、DES、RC4
valgrind	AES、ハッシュ[b]

[a] RHEL 8.3 以降、samba は FIPS 準拠の暗号を使用します。

[b] AES-NI などのソフトウェア/ハードウェアオフロード操作に再実装します。

13.7. システム全体の暗号化ポリシーに従わないようにアプリケーションを除外

アプリケーションで使用される暗号化関連の設定をカスタマイズする必要がある場合は、サポートされる暗号スイートとプロトコルをアプリケーションで直接設定することが推奨されます。

/etc/crypto-policies/back-ends ディレクトリーからアプリケーション関連のシンボリックリンクを削除することもできます。カスタマイズした暗号化設定に置き換えることもできます。この設定により、除外されたバックエンドを使用するアプリケーションに対するシステム全体の暗号化ポリシーが使用できなくなります。この修正は、Red Hat ではサポートされていません。

13.7.1. システム全体の暗号化ポリシーを除外する例

wget

wget ネットワークダウンローダーで使用される暗号化設定をカスタマイズするには、**--secure-protocol** オプションおよび **--ciphers** オプションを使用します。以下に例を示します。

```
$ wget --secure-protocol=TLSv1_1 --ciphers="SECURE128" https://example.com
```

詳細は、**wget(1)** man ページの HTTPS (SSL/TLS) Options のセクションを参照してください。

curl

curl ツールで使用する暗号を指定するには、**--ciphers** オプションを使用して、その値に、コロンで区切った暗号化のリストを指定します。以下に例を示します。

```
$ curl https://example.com --ciphers '@SECLEVEL=0:DES-CBC3-SHA:RSA-DES-CBC3-SHA'
```

詳細は、**curl(1)** の man ページを参照してください。

Firefox

Web ブラウザーの **Firefox** でシステム全体の暗号化ポリシーをオプトアウトできない場合でも、Firefox の設定エディターで、対応している暗号と TLS バージョンをさらに詳細に制限できます。アドレスバーに **about:config** と入力し、必要に応じて **security.tls.version.min** の値を変更します。たとえば、**security.tls.version.min** を **1** に設定すると、最低でも TLS 1.0 が必要になり、**security.tls.version.min 2** が TLS 1.1 になります。

OpenSSH

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 クライアントに対する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、以下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、辞書学的に **50-redhat.conf** ファイルよりも前に来るように、50 未満の 2 桁の接頭辞と、**.conf** の接尾辞で **/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ポリシーを指定します (例: **49-crypto-policy-override.conf** など)。

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

Libreswan

詳細は、Securing networks の Configuring IPsec connections that opt out of the system-wide crypto policies を参照してください。

関連情報

- **update-crypto-policies(8)** の man ページ

13.8. 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

この手順を使用して、有効な暗号化アルゴリズムまたはプロトコルのセットを調整します。

既存のシステム全体の暗号化ポリシーの上にカスタムサブポリシーを適用するか、そのようなポリシーを最初から定義することができます。

スコープが設定されたポリシーの概念により、バックエンドごとに異なるアルゴリズムセットを有効にできます。各設定ディレクティブは、特定のプロトコル、ライブラリー、またはサービスに限定できます。

また、ディレクティブでは、ワイルドカードを使用して複数の値を指定する場合にアスタリスクを使用できます。

/etc/crypto-policies/state/CURRENT.pol ファイルには、ワイルドカードデプロイメント後に現在適用されているシステム全体の暗号化ポリシーのすべての設定がリスト表示されます。暗号化ポリシーをより厳密にするには、**/usr/share/crypto-policies/policies/FUTURE.pol** ファイルにリストされている値を使用することを検討してください。

サブポリシーの例は、**/usr/share/crypto-policies/policies/modules/** ディレクトリーにあります。このディレクトリーのサブポリシーファイルには、コメントアウトされた行に説明が含まれています。

注記

システム全体の暗号化ポリシーのカスタマイズは、RHEL 8.2 から利用できます。スコープ指定ポリシーの概念と、RHEL 8.5 以降でワイルドカードを使用するオプションを使用できます。

手順

1. **/etc/crypto-policies/policies/modules/** ディレクトリーをチェックアウトします。

```
# cd /etc/crypto-policies/policies/modules/
```

2. 調整用のサブポリシーを作成します。次に例を示します。

```
# touch MYCRYPTO-1.pmod
# touch SCOPES-AND-WILDCARDS.pmod
```

重要

ポリシーモジュールのファイル名には大文字を使用します。

3. 任意のテキストエディターでポリシーモジュールを開き、システム全体の暗号化ポリシーを変更するオプションを挿入します。次に例を示します。

```
# vi MYCRYPTO-1.pmod
```

```
min_rsa_size = 3072
hash = SHA2-384 SHA2-512 SHA3-384 SHA3-512
```

```
# vi SCOPES-AND-WILDCARDS.pmod
```

```
# Disable the AES-128 cipher, all modes
cipher = -AES-128-*

# Disable CHACHA20-POLY1305 for the TLS protocol (OpenSSL, GnuTLS, NSS, and
OpenJDK)
cipher@TLS = -CHACHA20-POLY1305

# Allow using the FFDHE-1024 group with the SSH protocol (libssh and OpenSSH)
group@SSH = FFDHE-1024+

# Disable all CBC mode ciphers for the SSH protocol (libssh and OpenSSH)
cipher@SSH = -*-CBC

# Allow the AES-256-CBC cipher in applications using libssh
cipher@libssh = AES-256-CBC+
```

4. 変更をモジュールファイルに保存します。
5. ポリシーの調整を、システム全体の暗号化ポリシーレベル **DEFAULT** に適用します。

```
# update-crypto-policies --set DEFAULT:MYCRYPTO-1:SCOPES-AND-WILDCARDS
```

6. 暗号化設定を実行中のサービスやアプリケーションで有効にするには、システムを再起動します。

```
# reboot
```

検証

- **/etc/crypto-policies/state/CURRENT.pol** ファイルに変更が含まれていることを確認します。以下に例を示します。

```
$ cat /etc/crypto-policies/state/CURRENT.pol | grep rsa_size
min_rsa_size = 3072
```

関連情報

- **update-crypto-policies(8)** man ページの **Custom Policies** セクション
- **crypto-policies(7)** man ページの **Crypto Policy Definition Format** セクション
- Red Hat ブログ記事 How to customize crypto policies in RHEL 8.2

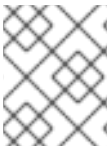
13.9. システム全体の暗号化ポリシーをカスタマイズして SHA-1 を無効化

SHA-1 ハッシュ関数は本質的に設計面で弱く、暗号解析機能によって攻撃を受けやすいため、RHEL 8 ではデフォルトで SHA-1 を使用していません。ただし、一部のサードパーティーアプリケーション (公開署名など) は SHA-1 を使用します。システムの署名アルゴリズムで SHA-1 の使用を無効にするには、**NO-SHA1** ポリシーモジュールを使用できます。

重要

NO-SHA1 ポリシーモジュールが無効にするのは、署名の SHA-1 ハッシュ関数だけでそれ以外は無効になりません。特に、**NO-SHA1** モジュールでも、ハッシュベースのメッセージ認証コード (HMAC) とともに SHA-1 を使用できます。HMAC セキュリティープロパティーは、対応するハッシュ関数の衝突耐性に依存しないので、HMAC に SHA-1 を使用している場合に最近の SHA-1 への攻撃による影響は大幅に低くなっています。

特定の鍵交換 (KEX) アルゴリズムの組み合わせ (例: **diffie-hellman-group-exchange-sha1**) を無効にする必要があるものの、関連する KEX とアルゴリズムの両方を継続して使用する場合は、SSH のシステム全体の暗号化ポリシーをオプトアウトして SSH を直接設定する手順について、SSH で diffie-hellman-group1-sha1 アルゴリズムを無効にする手順 を参照してください。



注記

SHA-1 を無効にするモジュールは、RHEL 8.3 で利用できます。システム全体の暗号化ポリシーのカスタマイズは、RHEL 8.2 から利用できます。

手順

1. ポリシーの調整を、システム全体の暗号化ポリシーレベル **DEFAULT** に適用します。

```
# update-crypto-policies --set DEFAULT:NO-SHA1
```

2. 暗号化設定を実行中のサービスやアプリケーションで有効にするには、システムを再起動します。

```
# reboot
```

関連情報

- **update-crypto-policies(8)** man ページの **Custom Policies** セクション
- **crypto-policies(7)** man ページの **Crypto Policy Definition Format** セクション
- Red Hat ブログ記事 How to customize crypto policies in RHEL

13.10. システム全体のカスタム暗号化ポリシーの作成および設定

以下の手順は、完全なポリシーファイルでシステム全体の暗号化ポリシーをカスタマイズする方法を示しています。

注記

システム全体の暗号化ポリシーのカスタマイズは、RHEL 8.2 から利用できます。

手順

1. カスタマイズのポリシーファイルを作成します。

```
# cd /etc/crypto-policies/policies/
# touch MYPOLICY.pol
```

または、定義されている 4 つのポリシーレベルのいずれかをコピーします。

```
# cp /usr/share/crypto-policies/policies/DEFAULT.pol /etc/crypto-
policies/policies/MYPOLICY.pol
```

2. 必要に応じて、テキストエディターでファイルを編集します。以下のようにしてカスタム暗号化ポリシーを使用します。

```
# vi /etc/crypto-policies/policies/MYPOLICY.pol
```

3. システム全体の暗号化ポリシーをカスタムレベルに切り替えます。

```
# update-crypto-policies --set MYPOLICY
```

4. 暗号化設定を実行中のサービスやアプリケーションで有効にするには、システムを再起動します。

```
# reboot
```

関連情報

- **update-crypto-policies (8)** の man ページの **Custom Policies** セクションと、**crypto-policies(7)** の **Crypto Policy Definition Format** セクションを参照してください。
- Red Hat ブログ記事 How to customize crypto policies in RHEL

13.11. 関連情報

- ナレッジベースの記事 System-wide crypto policies in RHEL 8 および Strong crypto defaults in RHEL 8 and deprecation of weak crypto algorithms

第14章 PKCS #11 で暗号化ハードウェアを使用するようにアプリケーションを設定

スマートカードや、エンドユーザー認証用の暗号化トークン、サーバーアプリケーション用のハードウェアセキュリティーモジュール (HSM) など、専用の暗号化デバイスで秘密情報の一部を分離することで、セキュリティー層が追加されます。RHEL では、PKCS #11 API を使用した暗号化ハードウェアへの対応がアプリケーション間で統一され、暗号ハードウェアでの秘密の分離が複雑なタスクではなくなりました。

14.1. PKCS #11 による暗号化ハードウェアへの対応

PKCS #11 (Public-Key Cryptography Standard) は、暗号化情報を保持する暗号化デバイスに、アプリケーションプログラミングインターフェイス (API) を定義し、暗号化機能を実行します。このデバイスはトークンと呼ばれ、ハードウェアまたはソフトウェアの形式で実装できます。

PKCS #11 トークンには、証明書、データオブジェクト、公開鍵、秘密鍵、または秘密鍵を含むさまざまなオブジェクトタイプを保存できます。このオブジェクトは、PKCS #11 の URI スキームにより一意に識別できます。

PKCS #11 の URI は、オブジェクト属性に従って、PKCS #11 モジュールで特定のオブジェクトを識別する標準的な方法です。これにより、URI の形式で、すべてのライブラリーとアプリケーションを同じ設定文字列で設定できます。

RHEL では、デフォルトでスマートカード用に OpenSC PKCS #11 ドライバーが提供されています。ただし、ハードウェアトークンと HSM には、システムにカウンターパートを持たない独自の PKCS #11 モジュールがあります。この PKCS #11 モジュールは **p11-kit** ツールで登録できます。これは、システムの登録済みスマートカードドライバーにおけるラッパーとして機能します。

システムで独自の PKCS #11 モジュールを有効にするには、新しいテキストファイルを **/etc/pkcs11/modules/** ディレクトリーに追加します。

/etc/pkcs11/modules/ ディレクトリーに新しいテキストファイルを作成すると、独自の PKCS #11 モジュールをシステムに追加できます。たとえば、**p11-kit** の OpenSC 設定ファイルは、以下のようになります。

```
$ cat /usr/share/p11-kit/modules/opensc.module
module: opensc-pkcs11.so
```

関連情報

- Consistent PKCS #11 support in Red Hat Enterprise Linux 8
- The PKCS #11 URI Scheme
- Controlling access to smart cards

14.2. 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

14.3. スマートカードから証明書を使用して認証するアプリケーションの設定

アプリケーションでスマートカードを使用した認証は、セキュリティーを強化し、自動化を簡素化する可能性があります。

- **wget** ネットワークダウンローダーでは、ローカルに保存された秘密鍵へのパスの代わりに PKCS #11 の URI を指定できるため、安全に保存された秘密鍵と証明書を必要とするタスク用のスクリプトの作成が容易になります。以下に例を示します。

  ```
  $ wget --private-key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --certificate 'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
  ```

  詳細は、**wget(1)** の man ページを参照してください。

- **curl** ツールで使用する PKCS #11 の URI は、以下のように指定します。

  ```
  $ curl --key 'pkcs11:token=softhsm;id=%01;type=private?pin-value=111111' --cert 'pkcs11:token=softhsm;id=%01;type=cert' https://example.com/
  ```

  詳細は、**curl(1)** の man ページを参照してください。

- Web ブラウザーの **Firefox** は、**p11-kit-proxy** モジュールを自動的に読み込みます。つまり、システムで対応しているすべてのスマートカードが自動的に検出されます。TLS クライアント認証を使用した場合、その他に必要な設定はありません。また、サーバーがスマートカードを要求する際に、スマートカードの鍵が自動的に使用されます。

カスタムアプリケーションで PKCS #11 の URI の使用

アプリケーションが **GnuTLS** ライブラリーまたは **NSS** ライブラリーを使用する場合、PKCS #11 の URI は PKCS #11 の組み込みサポートで保証されます。また、**OpenSSL** ライブラリーに依存するアプリケーションは、**openssl-pkcs11** エンジンが生成する暗号化ハードウェアモジュールにアクセスできます。

アプリケーションでスマートカードの秘密鍵を使用する必要があり、**NSS**、**GnuTLS**、および **OpenSSL** は使用しない場合は、**p11-kit** を使用して PKCS #11 モジュールの登録を実装します。

関連情報

- **p11-kit(8)** の man ページ

14.4. APACHE で秘密鍵を保護する HSM の使用

Apache HTTP サーバーは、ハードウェアセキュリティーモジュール (HSM) に保存されている秘密鍵と連携できます。これにより、鍵の漏えいや中間者攻撃を防ぐことができます。通常、これを行うには、ビジーなサーバーに高パフォーマンスの HSM が必要になります。

HTTPS プロトコルの形式でセキュアな通信を行うために、**Apache** HTTP サーバー (**httpd**) は OpenSSL ライブラリーを使用します。OpenSSL は、PKCS #11 にネイティブに対応しません。HSM を使用するには、エンジンインターフェイスを介して PKCS #11 モジュールへのアクセスを提供する

openssl-pkcs11 パッケージをインストールする必要があります。通常のファイル名ではなく PKCS #11 の URI を使用すると、**/etc/httpd/conf.d/ssl.conf** 設定ファイルでサーバーの鍵と証明書を指定できます。以下に例を示します。

```
SSLCertificateFile    "pkcs11:id=%01;token=softhsm;type=cert"
SSLCertificateKeyFile "pkcs11:id=%01;token=softhsm;type=private?pin-value=111111"
```

httpd-manual パッケージをインストールして、TLS 設定を含む **Apache** HTTP サーバーの完全ドキュメントを取得します。**/etc/httpd/conf.d/ssl.conf** 設定ファイルで利用可能なディレクティブの詳細は、**/usr/share/httpd/manual/mod/mod_ssl.html** を参照してください。

14.5. NGINX で秘密鍵を保護する HSM の使用

Nginx HTTP サーバーは、ハードウェアセキュリティーモジュール (HSM) に保存されている秘密鍵と連携できます。これにより、鍵の漏えいや中間者攻撃を防ぐことができます。通常、これを行うには、ビジーなサーバーに高パフォーマンスの HSM が必要になります。

Nginx は暗号化操作に OpenSSL を使用するため、PKCS #11 への対応は **openssl-pkcs11** エンジンを介して行う必要があります。**Nginx** は現在、HSM からの秘密鍵の読み込みのみに対応します。また、証明書は通常のファイルとして個別に提供する必要があります。**/etc/nginx/nginx.conf** 設定ファイルの **server** セクションで **ssl_certificate** オプションおよび **ssl_certificate_key** オプションを変更します。

```
ssl_certificate     /path/to/cert.pem
ssl_certificate_key "engine:pkcs11:pkcs11:token=softhsm;id=%01;type=private?pin-value=111111";
```

Nginx 設定ファイルの PKCS #11 URI に接頭辞 **engine:pkcs11:** が必要なことに注意してください。これは、他の **pkcs11** 接頭辞がエンジン名を参照するためです。

14.6. 関連情報

- **pkcs11.conf(5)** の man ページ

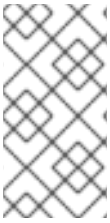
第15章 共通システム証明書の使用

共有システム証明書ストレージは、NSS、GnuTLS、OpenSSL、および Java が、システムの証明書アンカーと、拒否リスト情報を取得するデフォルトソースを共有します。トラストストアには、デフォルトで、Mozilla CA のリスト (信頼されるリストおよび信頼されないリスト) を含みます。システムは、コア Mozilla CA リストを更新したり、証明書リストを選択したりできます。

15.1. システム全体でトラストストアの使用

RHEL では、統合されたシステム全体のトラストストアが **/etc/pki/ca-trust/** ディレクトリーおよび **/usr/share/pki/ca-trust-source/** ディレクトリーに置かれています。**/usr/share/pki/ca-trust-source/** のトラスト設定は、**/etc/pki/ca-trust/** の設定よりも低い優先順位で処理されます。

証明書ファイルは、インストールされているサブディレクトリーによって扱われ方が異なります。たとえば、トラストアンカーは **/usr/share/pki/ca-trust-source/anchors/** または **/etc/pki/ca-trust/source/anchors/** ディレクトリーに属します。



注記

階層暗号化システムでは、トラストアンカーとは、他のパーティーが信頼できると想定する権威あるエンティティーです。X.509 アーキテクチャーでは、ルート証明書はトラストチェーンの元となるトラストアンカーです。チェーンの検証を有効にするには、信頼元がまずトラストアンカーにアクセスできる必要があります。

関連情報

- **update-ca-trust(8)** および **trust(1)** の man ページ

15.2. 新しい証明書の追加

新しいソースでシステムのアプリケーションを確認するには、対応する証明書をシステム全体のストアに追加し、**update-ca-trust** コマンドを使用します。

前提条件

- **ca-certificates** パッケージがシステムにインストールされている。

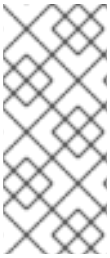
手順

1. システムで信頼されている CA のリストに、シンプルな PEM または DER のファイルフォーマットに含まれる証明書を追加するには、**/usr/share/pki/ca-trust-source/anchors/** ディレクトリーまたは **/etc/pki/ca-trust/source/anchors/** ディレクトリーに証明書ファイルをコピーします。以下に例を示します。

```
# cp ~/certificate-trust-examples/Cert-trust-test-ca.pem /usr/share/pki/ca-trust-source/anchors/
```

2. システム全体のトラストストア設定を更新するには、**update-ca-trust** コマンドを実行します。

```
# update-ca-trust
```



注記

update-ca-trust を事前に実行しなくても、Firefox ブラウザーは追加された証明書を使用できますが、CA を変更するたびに **update-ca-trust** コマンドを入力してください。Firefox、Chromium および GNOME Web などのブラウザーはファイルをキャッシュするので、ブラウザーのキャッシュをクリアするか、ブラウザーを再起動して、現在のシステム証明書の設定を読み込む必要がある場合があります。

関連情報

- **update-ca-trust(8)** および **trust(1)** の man ページ

15.3. 信頼されているシステム証明書の管理

trust コマンドを使用すると、共有システム全体のトラストストアで証明書を便利な方法で管理できます。

- トラストアンカーのリスト表示、抽出、追加、削除、または変更を行うには、**trust** コマンドを使用します。このコマンドの組み込みヘルプを表示するには、引数を付けずに、または **--help** ディレクティブを付けて実行します。

```
$ trust
usage: trust command <args>...

Common trust commands are:
  list            List trust or certificates
  extract         Extract certificates and trust
  extract-compat  Extract trust compatibility bundles
  anchor          Add, remove, change trust anchors
  dump            Dump trust objects in internal format

See 'trust <command> --help' for more information
```

- すべてのシステムのトラストアンカーおよび証明書のリストを表示するには、**trust list** コマンドを実行します。

```
$ trust list
pkcs11:id=%d2%87%b4%e3%df%37%27%93%55%f6%56%ea%81%e5%36%cc%8c%1e%3
f%bd;type=cert
    type: certificate
    label: ACCVRAIZ1
    trust: anchor
    category: authority

pkcs11:id=%a6%b3%e1%2b%2b%49%b6%d7%73%a1%aa%94%f5%01%e7%73%65%4c%
ac%50;type=cert
    type: certificate
    label: ACEDICOM Root
    trust: anchor
    category: authority
...
```

- トラストアンカーをシステム全体のトラストストアに保存するには、**trust anchor** サブコマンドを使用し、証明書のパスを指定します。**<path.to/certificate.crt>** を、証明書およびそのファイル名へのパスに置き換えます。

```
# trust anchor <path.to/certificate.crt>
```

- 証明書を削除するには、証明書のパス、または証明書の ID を使用します。

```
# trust anchor --remove <path.to/certificate.crt>
# trust anchor --remove "pkcs11:id=<%AA%BB%CC%DD%EE>;type=cert"
```

関連情報

- **trust** コマンドのすべてのサブコマンドは、以下のような詳細な組み込みヘルプを提供します。

```
$ trust list --help
usage: trust list --filter=<what>

  --filter=<what>     filter of what to export
                      ca-anchors        certificate anchors
...
  --purpose=<usage>   limit to certificates usable for the purpose
                      server-auth       for authenticating servers
...
```

関連情報

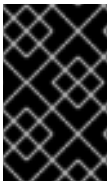
- **update-ca-trust(8)** および **trust(1)** の man ページ

第16章 セキュリティーコンプライアンスおよび脆弱性スキャンの開始

16.1. RHEL における設定コンプライアンスツール

Red Hat Enterprise Linux は、コンプライアンス監査を完全に自動化できるツールを提供します。このツールは SCAP (Security Content Automation Protocol) 規格に基づいており、コンプライアンスポリシーの自動化に合わせるように設計されています。

- SCAP Workbench - **scap-workbench** グラフィカルユーティリティーは、1 台のローカルシステムまたはリモートシステムで設定スキャンと脆弱性スキャンを実行するように設計されています。これらのスキャンと評価に基づくセキュリティーレポートの生成にも使用できます。
- OpenSCAP: **OpenSCAP** ライブラリーは、付随する **oscap** コマンドラインユーティリティーとともに、ローカルシステムで設定スキャンと脆弱性スキャンを実行するように設計されています。これにより、設定コンプライアンスのコンテンツを検証し、スキャンおよび評価に基づいてレポートおよびガイドを生成します。



重要

OpenSCAP の使用中にメモリー消費の問題が発生する可能性があります。これにより、プログラムが途中で停止し、結果ファイルが生成されない可能性があります。詳細については、ナレッジベース記事 OpenSCAP のメモリー消費の問題 を参照してください。

- SCAP Security Guide (SSG) - **scap-security-guide** パッケージは、Linux システム向けの最新のセキュリティーポリシーコレクションを提供します。このガイダンスは、セキュリティー強化に関する実践的なアドバイスのカタログで構成されています (該当する場合は、法規制要件へのリンクが含まれます)。このプロジェクトは、一般的なポリシー要件と特定の実装ガイドラインとの間にあるギャップを埋めることを目的としています。
- **Script Check Engine (SCE)** - SCE は SCAP プロトコルの拡張機能であり、この機能を使用すると管理者が Bash、Python、Ruby などのスクリプト言語を使用してセキュリティーコンテンツを記述できるようになります。SCE 拡張機能は、**openscap-engine-sce** パッケージで提供されます。SCE 自体は SCAP 標準規格の一部ではありません。

複数のリモートシステムで自動コンプライアンス監査を実行する必要がある場合は、Red Hat Satellite 用の OpenSCAP ソリューションを利用できます。

関連情報

- **oscap(8)**、**scap-workbench(8)**、および **scap-security-guide(8)** の man ページ
- Red Hat Security Demos:Creating Customized Security Policy Content to Automate Security Compliance
- Red Hat Security Demos:Defend Yourself with RHEL Security Technologies
- Red Hat Satellite の管理ガイドのセキュリティーコンプライアンスの管理

16.2. RED HAT SECURITY ADVISORIES OVAL フィード

Red Hat Enterprise Linux のセキュリティー監査機能は、標準規格セキュリティー設定共通化手順 (Security Content Automation Protocol (SCAP)) を基にしています。SCAP は、自動化された設定、脆

弱性およびパッチの確認、技術的な制御コンプライアンスアクティビティー、およびセキュリティーの測定に対応している多目的な仕様のフレームワークです。

SCAP 仕様は、スキャナーまたはポリシーエディターの実装が義務付けられていなくても、セキュリティーコンテンツの形式がよく知られて標準化されているエコシステムを作成します。これにより、組織は、採用しているセキュリティーベンダーの数に関係なく、セキュリティーポリシー (SCAP コンテンツ) を構築するのは一度で済みます。

セキュリティー検査言語 OVAL (Open Vulnerability Assessment Language) は、SCAP に不可欠で最も古いコンポーネントです。その他のツールやカスタマイズされたスクリプトとは異なり、OVAL は、宣言型でリソースが必要な状態を記述します。OVAL コードは、スキャナーと呼ばれる OVAL インタープリターツールを使用して直接実行されることは決してありません。OVAL が宣言型であるため、評価されるシステムの状態が偶然修正されることはありません。

他のすべての SCAP コンポーネントと同様に、OVAL は XML に基づいています。SCAP 標準規格は、いくつかのドキュメント形式を定義します。この形式にはそれぞれ異なる種類の情報が記載され、異なる目的に使用されます。

Red Hat 製品セキュリティー を使用すると、Red Hat 製品をお使いのお客様に影響を及ぼすセキュリティー問題をすべて追跡して調査します。Red Hat カスタマーポータルで簡潔なパッチやセキュリティーアドバイザリーを適時提供します。Red Hat は OVAL パッチ定義を作成してサポートし、マシンが判読可能なセキュリティーアドバイザリーを提供します。

プラットフォーム、バージョン、およびその他の要因が異なるため、Red Hat 製品セキュリティーによる脆弱性の重大度定性評価は、サードパーティーが提供する Common Vulnerability Scoring System (CHC) のベースライン評価と完全に一致しているわけではありません。したがって、サードパーティーが提供する定義ではなく、RHSA OVAL 定義を使用することが推奨されます。

各 RHSA OVAL 定義 は完全なパッケージとして利用でき、新しいセキュリティーアドバイザリーが Red Hat カスタマーポータルで利用可能になってから 1 時間以内に更新されます。

各 OVAL パッチ定義は、Red Hat セキュリティーアドバイザリー (RHSA) と 1 対 1 にマッピングしています。RHSA には複数の脆弱性に対する修正が含まれるため、各脆弱性は、共通脆弱性識別子 (Common Vulnerabilities and Exposures (CVE)) 名ごとに表示され、公開バグデータベースの該当箇所へのリンクが示されます。

RHSA OVAL 定義は、システムにインストールされている RPM パッケージで脆弱なバージョンを確認するように設計されています。この定義は拡張でき、パッケージが脆弱な設定で使用されているかどうかを見つけるなど、さらに確認できるようにすることができます。この定義は、Red Hat が提供するソフトウェアおよび更新に対応するように設計されています。サードパーティーソフトウェアのパッチ状態を検出するには、追加の定義が必要です。

注記

Red Hat Insights for Red Hat Enterprise Linux コンプライアンスサービス は、IT セキュリティーおよびコンプライアンス管理者が Red Hat Enterprise Linux システムのセキュリティーポリシーのコンプライアンスを評価、監視、およびレポートするのに役立ちます。また、コンプライアンスサービス UI 内で完全に SCAP セキュリティーポリシーを作成および管理することもできます。

関連情報

- Red Hat and OVAL compatibility
- Red Hat and CVE compatibility

- 製品セキュリティーの概要 の 通知およびアドバイザリー
- Security Data Metrics

16.3. 脆弱性スキャン

16.3.1. Red Hat Security Advisories OVAL フィード

Red Hat Enterprise Linux のセキュリティー監査機能は、標準規格セキュリティー設定共通化手順 (Security Content Automation Protocol (SCAP)) を基にしています。SCAP は、自動化された設定、脆弱性およびパッチの確認、技術的な制御コンプライアンスアクティビティー、およびセキュリティーの測定に対応している多目的な仕様のフレームワークです。

SCAP 仕様は、スキャナーまたはポリシーエディターの実装が義務付けられていなくても、セキュリティーコンテンツの形式がよく知られて標準化されているエコシステムを作成します。これにより、組織は、採用しているセキュリティーベンダーの数に関係なく、セキュリティーポリシー (SCAP コンテンツ) を構築するのは一度で済みます。

セキュリティー検査言語 OVAL (Open Vulnerability Assessment Language) は、SCAP に不可欠で最も古いコンポーネントです。その他のツールやカスタマイズされたスクリプトとは異なり、OVAL は、宣言型でリソースが必要な状態を記述します。OVAL コードは、スキャナーと呼ばれる OVAL インタープリターツールを使用して直接実行されることは決してありません。OVAL が宣言型であるため、評価されるシステムの状態が偶然修正されることはありません。

他のすべての SCAP コンポーネントと同様に、OVAL は XML に基づいています。SCAP 標準規格は、いくつかのドキュメント形式を定義します。この形式にはそれぞれ異なる種類の情報が記載され、異なる目的に使用されます。

Red Hat 製品セキュリティー を使用すると、Red Hat 製品をお使いのお客様に影響を及ぼすセキュリティー問題をすべて追跡して調査します。Red Hat カスタマーポータルで簡潔なパッチやセキュリティーアドバイザリーを適時提供します。Red Hat は OVAL パッチ定義を作成してサポートし、マシンが判読可能なセキュリティーアドバイザリーを提供します。

プラットフォーム、バージョン、およびその他の要因が異なるため、Red Hat 製品セキュリティーによる脆弱性の重大度定性評価は、サードパーティーが提供する Common Vulnerability Scoring System (CHC) のベースライン評価と完全に一致しているわけではありません。したがって、サードパーティーが提供する定義ではなく、RHSA OVAL 定義を使用することが推奨されます。

各 RHSA OVAL 定義 は完全なパッケージとして利用でき、新しいセキュリティーアドバイザリーが Red Hat カスタマーポータルで利用可能になってから 1 時間以内に更新されます。

各 OVAL パッチ定義は、Red Hat セキュリティーアドバイザリー (RHSA) と 1 対 1 にマッピングしています。RHSA には複数の脆弱性に対する修正が含まれるため、各脆弱性は、共通脆弱性識別子 (Common Vulnerabilities and Exposures (CVE)) 名ごとに表示され、公開バグデータベースの該当箇所へのリンクが示されます。

RHSA OVAL 定義は、システムにインストールされている RPM パッケージで脆弱なバージョンを確認するように設計されています。この定義は拡張でき、パッケージが脆弱な設定で使用されているかどうかを見つけるなど、さらに確認できるようにすることができます。この定義は、Red Hat が提供するソフトウェアおよび更新に対応するように設計されています。サードパーティーソフトウェアのパッチ状態を検出するには、追加の定義が必要です。

注記

Red Hat Insights for Red Hat Enterprise Linux コンプライアンスサービス は、IT セキュリティーおよびコンプライアンス管理者が Red Hat Enterprise Linux システムのセキュリティーポリシーのコンプライアンスを評価、監視、およびレポートするのに役立ちます。また、コンプライアンスサービス UI 内で完全に SCAP セキュリティーポリシーを作成および管理することもできます。

関連情報

- Red Hat and OVAL compatibility
- Red Hat and CVE compatibility
- 製品セキュリティーの概要 の 通知およびアドバイザリー
- Security Data Metrics

16.3.2. システムの脆弱性のスキャン

oscap コマンドラインユーティリティーを使用すると、ローカルシステムのスキャン、設定コンプライアンスコンテンツの確認、ならびにスキャンおよび評価を基にしたレポートとガイドの生成が可能です。このユーティリティーは、OpenSCAP ライブラリーのフロントエンドとしてサービスを提供し、その機能を処理する SCAP コンテンツのタイプに基づいてモジュール (サブコマンド) にグループ化します。

前提条件

- **openscap-scanner** および **bzip2** パッケージがインストールされます。

手順

1. システムに最新 RHSA OVAL 定義をダウンロードします。

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. システムの脆弱性をスキャンし、**vulnerability.html** ファイルに結果を保存します。

```
# oscap oval eval --report vulnerability.html rhel-8.oval.xml
```

検証

- 結果をブラウザーで確認します。以下に例を示します。

```
$ firefox vulnerability.html &
```

関連情報

- **oscap(8)** の man ページ
- Red Hat OVAL 定義
- OpenSCAP のメモリー消費の問題

16.3.3. リモートシステムの脆弱性のスキャン

SSH プロトコルで **oscap-ssh** ツールを使用して、OpenSCAP スキャナーでリモートシステムの脆弱性を確認することもできます。

前提条件

- **openscap-utils** および **bzip2** パッケージは、スキャンに使用するシステムにインストールされます。
- リモートシステムに **openscap-scanner** パッケージがインストールされている。
- リモートシステムで SSH サーバーが実行している。

手順

1. システムに最新 RHSA OVAL 定義をダウンロードします。

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. 脆弱性に対して、ホスト名 **machine1**、ポート 22 で実行する SSH、およびユーザー名 **joesec** でリモートシステムをスキャンし、結果を **remote-vulnerability.html** ファイルに保存します。

```
# oscap-ssh joesec@machine1 22 oval eval --report remote-vulnerability.html rhel-8.oval.xml
```

関連情報

- **oscap-ssh(8)**
- Red Hat OVAL 定義
- OpenSCAP のメモリー消費の問題

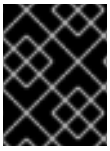
16.4. 設定コンプライアンススキャン

16.4.1. RHEL の設定コンプライアンス

設定コンプライアンススキャンを使用して、特定の組織で定義されているベースラインに準拠できます。たとえば、米国政府と協力している場合は、システムを Operating System Protection Profile (OSPP) に準拠させ、支払い処理業者の場合は、システムを Payment Card Industry Data Security Standard (PCI-DSS) に準拠させなければならない場合があります。設定コンプライアンススキャンを実行して、システムセキュリティーを強化することもできます。

Red Hat は、対象コンポーネント向けの Red Hat のベストプラクティスに従っているため、SCAP Security Guide パッケージで提供される Security Content Automation Protocol (SCAP) コンテンツに従うことを推奨します。

SCAP Security Guide パッケージは、SCAP 1.2 および SCAP 1.3 標準規格に準拠するコンテンツを提供します。**openscap scanner** ユーティリティーは、SCAP Security Guide パッケージで提供される SCAP 1.2 および SCAP 1.3 コンテンツの両方と互換性があります。



重要

設定コンプライアンススキャンを実行しても、システムが準拠しているとは限りません。

SCAP セキュリティーガイドスイートは、データストリームのドキュメント形式で、複数のプラットフォームのプロファイルを提供します。データストリームは、定義、ベンチマーク、プロファイル、および個々のルールが含まれるファイルです。各ルールでは、コンプライアンスの適用性と要件を指定します。RHEL は、セキュリティーポリシーを扱う複数のプロファイルを提供します。Red Hat データストリームには、業界標準の他に、失敗したルールの修正に関する情報も含まれます。

コンプライアンススキャンリソースの構造

```
Data stream
   ├── xccdf
   |     ├── benchmark
   |          ├── profile
   |          |    ├──rule reference
   |          |    └──variable
   |          ├── rule
   |               ├── human readable data
   |               ├── oval reference
   ├── oval         ├── ocil reference
   ├── ocil         ├── cpe reference
   └── cpe          └── remediation
```

プロファイルは、OSPP、PCI-DSS、Health Insurance Portability and Accountability Act (HIPAA) などのセキュリティーポリシーに基づく一連のルールです。これにより、セキュリティー標準規格に準拠するために、システムを自動で監査できます。

プロファイルを変更 (調整) して、パスワードの長さなどの特定のルールをカスタマイズできます。プロファイルの調整の詳細は、SCAP Workbench を使用したセキュリティープロファイルのカスタマイズ を参照してください。

16.4.2. OpenSCAP スキャン結果の例

システムのさまざまなプロパティーと、OpenSCAP スキャンに適用されるデータストリームおよびプロファイルによっては、ルールごとに固有の結果が生成されることがあります。以下は、考えられる結果のリストで、その意味を簡単に説明します。

表16.1 OpenSCAP スキャン結果の例

結果	説明
Pass	スキャンでは、このルールとの競合が見つかりませんでした。
Fail	スキャンで、このルールとの競合が検出されました。
Not checked	OpenSCAP はこのルールの自動評価を実行しません。システムがこのルールに手動で準拠しているかどうかを確認してください。

結果	説明
Not applicable	このルールは、現在の設定には適用されません。
Not selected	このルールはプロファイルには含まれません。OpenSCAP はこのルールを評価せず、結果にこのようなルールは表示されません。
Error	スキャンでエラーが発生しました。詳細は、**--verbose DEVEL** オプションを指定して **oscap** コマンドで確認できます。バグレポート を作成することを検討してください。
Unknown	スキャンで予期しない状況が発生しました。詳細は、`**--verbose DEVEL** オプションを指定して **oscap** コマンドを入力できます。バグレポート を作成することを検討してください。

16.4.3. 設定コンプライアンスのプロファイルの表示

スキャンまたは修復にプロファイルを使用することを決定する前に、**oscap info** サブコマンドを使用して、プロファイルを一覧表示し、詳細な説明を確認できます。

前提条件

- **openscap-scanner** パッケージおよび **scap-security-guide** パッケージがインストールされている。

手順

1. SCAP Security Guide プロジェクトが提供するセキュリティーコンプライアンスプロファイルで利用可能なファイルをすべて表示します。

```
$ ls /usr/share/xml/scap/ssg/content/
ssg-firefox-cpe-dictionary.xml  ssg-rhel6-ocil.xml
ssg-firefox-cpe-oval.xml        ssg-rhel6-oval.xml
...
ssg-rhel6-ds-1.2.xml          ssg-rhel8-oval.xml
ssg-rhel8-ds.xml              ssg-rhel8-xccdf.xml
...
```

2. **oscap info** サブコマンドを使用して、選択したデータストリームに関する詳細情報を表示します。データストリームを含む XML ファイルは、名前に **-ds** 文字列で示されます。**Profiles** セクションでは、利用可能なプロファイルと、その ID のリストを確認できます。

```
$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
Profiles:
...
  Title: Health Insurance Portability and Accountability Act (HIPAA)
    Id: xccdf_org.ssgproject.content_profile_hipaa
  Title: PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 8
    Id: xccdf_org.ssgproject.content_profile_pci-dss
```

```
    Title: OSPP - Protection Profile for General Purpose Operating Systems
      Id: xccdf_org.ssgproject.content_profile_ospp
...
```

3. データストリームファイルからプロファイルを選択し、選択したプロファイルに関する追加情報を表示します。そのためには、**oscap info** に **--profile** オプションを指定した後に、直前のコマンドの出力で表示された ID の最後のセクションを指定します。たとえば、HIPPA プロファイルの ID は **xccdf_org.ssgproject.content_profile_hipaa** で、**--profile** オプションの値は **hipaa** です。

```
$ oscap info --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
...
Profile
 Title: Health Insurance Portability and Accountability Act (HIPAA)

 Description: The HIPAA Security Rule establishes U.S. national standards to protect
individuals' electronic personal health information that is created, received, used, or
maintained by a covered entity.
...
```

関連情報

- **scap-security-guide (8)** の man ページ
- OpenSCAP のメモリー消費の問題

16.4.4. 特定のベースラインによる設定コンプライアンスの評価

システムが特定のベースラインに準拠しているかどうかを確認するには、次の手順に従います。

前提条件

- **openscap-scanner** パッケージおよび **scap-security-guide** パッケージがインストールされている。
- システムが準拠する必要があるベースライン内のプロファイルの ID を知っている必要があります。ID を見つけるには、設定コンプライアンスのプロファイルの表示 を参照してください。

手順

1. 選択したプロファイルでそのシステムがどのように複雑であるかを評価し、スキャン内容を保存すると、以下のように HTML ファイル (**report.html**) に結果が表示されます。

```
$ oscap xccdf eval --report report.html --profile hipaa /usr/share/xml/scap/ssg/content/ssg-
rhel8-ds.xml
```

2. オプション: コンプライアンスに対して、ホスト名 **machine1**、ポート **22** で実行する SSH、およびユーザー名 **joesec** でリモートシステムをスキャンし、結果を **remote-report.html** ファイルに保存します。

```
$ oscap-ssh joesec@machine1 22 xccdf eval --report remote_report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

関連情報

- **scap-security-guide (8)** の man ページ
- **/usr/share/doc/scap-security-guide/** ディレクトリーにある **SCAP Security Guide** ドキュメント
- **/usr/share/doc/scap-security-guide/guides/ssg-rhel8-guide-index.html** - **scap-security-guide-doc** パッケージでインストールされた Red Hat Enterprise Linux 8 のセキュアな設定ガイド
- OpenSCAP のメモリー消費の問題

16.5. 特定のベースラインに合わせたシステムの修復

この手順を使用して、特定のベースラインに合わせて RHEL システムを修正します。この例では、Health Insurance Portability and Accountability Act (HIPAA) プロファイルを使用します。

警告

修正 オプションが有効な状態でのシステム評価は、慎重に行わないとシステムが機能不全に陥る場合があります。Red Hat は、セキュリティーを強化した修正で加えられた変更を元に戻す自動手段は提供していません。修復は、デフォルト設定の RHEL システムで対応しています。インストール後にシステムが変更した場合は、修正を実行しても、必要なセキュリティープロファイルに準拠しない場合があります。

前提条件

- RHEL システムに、**scap-security-guide** パッケージがインストールされている。

手順

1. **oscap** コマンドに **--remediate** オプションを指定して使用します。

```
# oscap xccdf eval --profile hipaa --remediate /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

2. システムを再起動します。

検証

1. システムの OSPP プロファイルへのコンプライアンスを評価し、スキャン結果を **ospp_report.html** ファイルに保存します。

```
$ oscap xccdf eval --report hipaa_report.html --profile hipaa /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

関連情報

- **scap-security-guide(8)** および **oscap(8)** の man ページ

16.6. SSG ANSIBLE PLAYBOOK を使用して、特定のベースラインに合わせてシステムを修正する

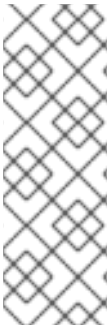
この手順では、SCAP Security Guide プロジェクトの Ansible Playbook ファイルを使用して、特定のベースラインでシステムを修正します。この例では、Health Insurance Portability and Accountability Act (HIPAA) プロファイルを使用します。

警告

修正 オプションが有効な状態でのシステム評価は、慎重に行わないとシステムが機能不全に陥る場合があります。Red Hat は、セキュリティーを強化した修正で加えられた変更を元に戻す自動手段は提供していません。修復は、デフォルト設定の RHEL システムで対応しています。インストール後にシステムが変更した場合は、修正を実行しても、必要なセキュリティープロファイルに準拠しない場合があります。

前提条件

- **scap-security-guide** パッケージがインストールされている。
- **ansible-core** パッケージがインストールされている。詳細は、Ansible インストールガイドを参照してください。



注記

RHEL 8.6 以降では、Ansible Engine は、組み込みモジュールのみを含む **ansible-core** パッケージに置き換えられました。Ansible 修復の多くは、community コレクションおよび Portable Operating System Interface (POSIX) コレクションのモジュールを使用することに注意してください。これは組み込みモジュールには含まれていません。この場合は、Ansible 修復の代わりに Bash 修復を使用できます。RHEL 8 の Red Hat Connector には、Ansible Core で修復 Playbook を機能させるために必要な Ansible モジュールが含まれています。

手順

1. Ansible を使用して OSPP に合わせてシステムを修正します。

```
# ansible-playbook -i localhost, -c local /usr/share/scap-security-guide/ansible/rhel8-playbook-hipaa.yml
```

2. システムを再起動します。

検証

1. システムの OSPP プロファイルへのコンプライアンスを評価し、スキャン結果を **ospp_report.html** ファイルに保存します。

```
# oscap xccdf eval --profile hipaa --report hipaa_report.html
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

関連情報

- **scap-security-guide(8)** および **oscap(8)** の man ページ
- Ansible ドキュメント

16.7. システムを特定のベースラインに合わせるための修復用 ANSIBLE PLAYBOOK の作成

以下の手順に従って、必要な修復のみを含む Ansible Playbook を作成し、システムを特定のベースラインに合わせます。この例では、Health Insurance Portability and Accountability Act (HIPAA) プロファイルを使用します。この手順では、要件を満たしていない小規模の Playbook を作成します。以下の手順に従うと、システムを変更せずに、後のアプリケーション用にファイルの準備を行うだけです。

注記

RHEL 8.6 では、Ansible Engine は、組み込みモジュールのみを含む **ansible-core** パッケージに置き換えられました。Ansible 修復の多くは、community コレクションおよび Portable Operating System Interface (POSIX) コレクションのモジュールを使用することに注意してください。これは組み込みモジュールには含まれていません。この場合は、Bash 修復を Ansible 修復の代わりに使用できます。RHEL 8.6 の Red Hat Connector には、Ansible Core で修復 Playbook を機能させるために必要な Ansible モジュールが含まれています。

前提条件

- **scap-security-guide** パッケージがインストールされている。

手順

1. システムをスキャンして結果を保存します。

```
# oscap xccdf eval --profile hipaa --results hipaa-results.xml
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

2. 前の手順で生成されたファイルに基づいて Ansible Playbook を生成します。

```
# oscap xccdf generate fix --fix-type ansible --profile hipaa --output hipaa-remediations.yml
hipaa-results.xml
```

3. **hippa-remediations.yml** ファイルには、手順 1 で実行されたスキャン中に失敗したルールの Ansible 修復が含まれています。この生成されたファイルを確認した後、**ansible-playbook hipaa-remediations.yml** コマンドで適用できます。

検証

- お使いのテキストエディターで、手順 1 で実行したスキャンで失敗したルールが **hipaa-remediations.yml** ファイルに含まれていることを確認します。

関連情報

- **scap-security-guide(8)** および **oscap(8)** の man ページ
- Ansible ドキュメント

16.8. 後でアプリケーションを修復するための BASH スクリプトの作成

この手順を使用して、システムを HIPAA などのセキュリティープロファイルと調整する修正を含む Bash スクリプトを作成します。次の手順では、システムに変更を加えることなく、後のアプリケーション用にファイルを準備する方法を説明します。

前提条件

- RHEL システムに、**scap-security-guide** パッケージがインストールされている。

手順

1. **oscap** コマンドを使用してシステムをスキャンし、結果を XML ファイルに保存します。以下の例では、**oscap** は **hipaa** プロファイルに対してシステムを評価します。

```
# oscap xccdf eval --profile hipaa --results hipaa-results.xml
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

2. 前の手順で生成された結果ファイルに基づいて Bash スクリプトを生成します。

```
# oscap xccdf generate fix --profile hipaa --fix-type bash --output hipaa-remediations.sh
hipaa-results.xml
```

3. **hippa-dss-remediations.sh** ファイルには、手順 1 で実行されたスキャン中に失敗したルールの修復が含まれます。この生成されたファイルを確認したら、このファイルと同じディレクトリー内で **./hipaa-remediations.sh** コマンドを使用して適用できます。

検証

- お使いのテキストエディターで、手順 1 で実行したスキャンで失敗したルールが **hipaa-remediations.sh** ファイルに含まれていることを確認します。

関連情報

- **scap-security-guide(8)**、**oscap(8)**、および **bash(1)** の man ページ

16.9. SCAP WORKBENCH を使用したカスタムプロファイルでシステムのスキャン

SCAP Workbench (**scap-workbench**) パッケージはグラフィカルユーティリティーで、1 台のローカルシステムまたはリモートシステムで設定スキャンと脆弱性スキャンを実行し、システムの修復を実行して、スキャン評価に基づくレポートを生成します。**oscap** コマンドラインユーティリティーとの比較は、**SCAP Workbench** には限定的な機能しかないことに注意してください。**SCAP Workbench** は、データストリームファイルの形式でセキュリティーコンテンツを処理します。

16.9.1. SCAP Workbench を使用したシステムのスキャンおよび修復

選択したセキュリティーポリシーに対してシステムを評価するには、以下の手順に従います。

前提条件

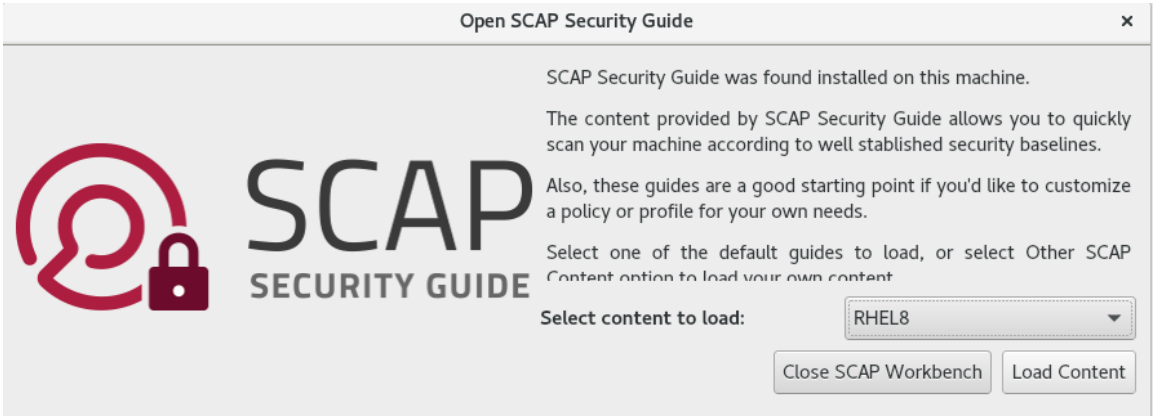
- **scap-workbench** パッケージがシステムにインストールされている。

手順

1. **GNOME Classic** デスクトップ環境から **SCAP Workbench** を実行するには、**Super** キーを押して **アクティビティーの概要** を開き、**scap-workbench**と入力して **Enter**を押します。または、次のコマンドを実行します。

```
$ scap-workbench &
```

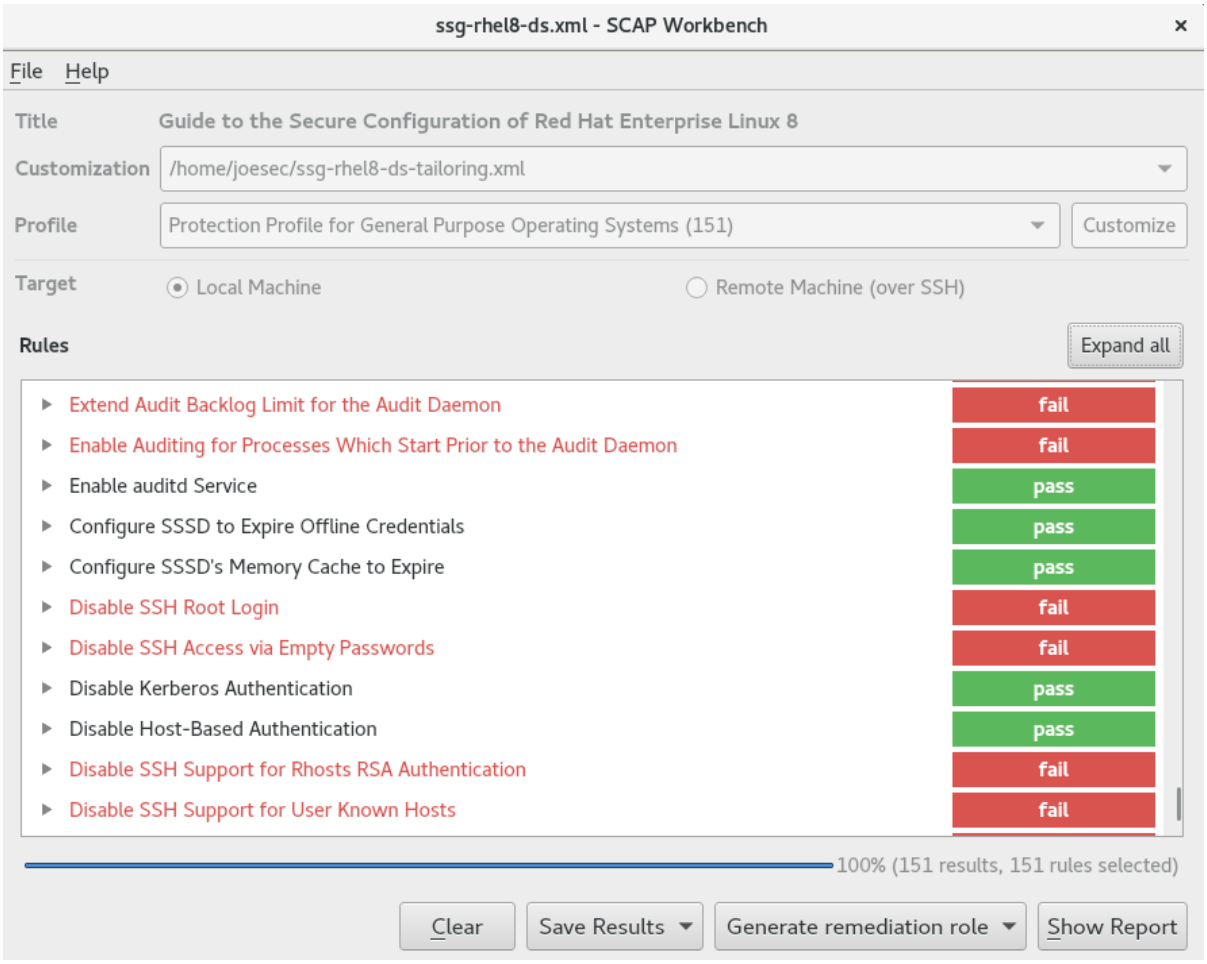
2. 以下のオプションのいずれかを使用してセキュリティーポリシーを選択します。
   - 開始ウィンドウの **Load Content** ボタン
   - **Open content from SCAP Security Guide**
   - **File** メニューの **Open Other Content** で、XCCDF、SCAP RPM、またはデータストリームファイルの各ファイルを検索します。



3. **Remediate** チェックボックスを選択して、システム設定の自動修正を行うことができます。このオプションを有効にすると、**SCAP Workbench** は、ポリシーにより適用されるセキュリティールールに従ってシステム設定の変更を試みます。このプロセスは、システムスキャン時に失敗した関連チェックを修正する必要があります。

> **警告**
>
> **修正** オプションが有効な状態でのシステム評価は、慎重に行わないとシステムが機能不全に陥る場合があります。Red Hat は、セキュリティーを強化した修正で加えられた変更を元に戻す自動手段は提供していません。修復は、デフォルト設定の RHEL システムで対応しています。インストール後にシステムが変更した場合は、修正を実行しても、必要なセキュリティープロファイルに準拠しない場合があります。

4. **Scan** ボタンをクリックし、選択したプロファイルでシステムをスキャンします。



5. スキャン結果を XCCDF ファイル、ARF ファイル、または HTML ファイルの形式で保存するには、**Save Results** コンボボックスをクリックします。**HTML Report** オプションを選択して、スキャンレポートを、人間が判読できる形式で生成します。XCCDF 形式および ARF (データストリーム) 形式は、追加の自動処理に適しています。3 つのオプションはすべて繰り返し選択できます。

6. 結果ベースの修復をファイルにエクスポートするには、ポップアップメニューの **Generate remediation role** を使用します。

16.9.2. SCAP Workbench を使用したセキュリティープロファイルのカスタマイズ

セキュリティープロファイルをカスタマイズするには、特定のルール (パスワードの最小長など) のパラメーターを変更し、別の方法で対象とするルールを削除し、追加のルールを選択して内部ポリシーを実装できます。プロファイルをカスタマイズして新しいルールの定義はできません。

以下の手順は、プロファイルをカスタマイズ (調整) するための **SCAP Workbench** の使用を示しています。**oscap** コマンドラインユーティリティーで使用するようにカスタマイズしたプロファイルを保存することもできます。

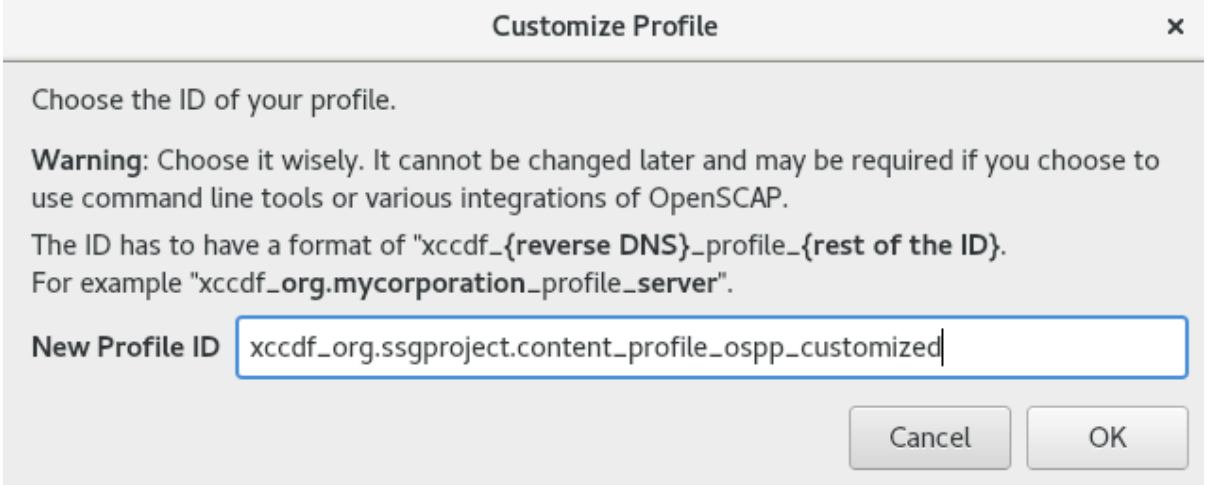
前提条件

- **scap-workbench** パッケージがシステムにインストールされている。

手順

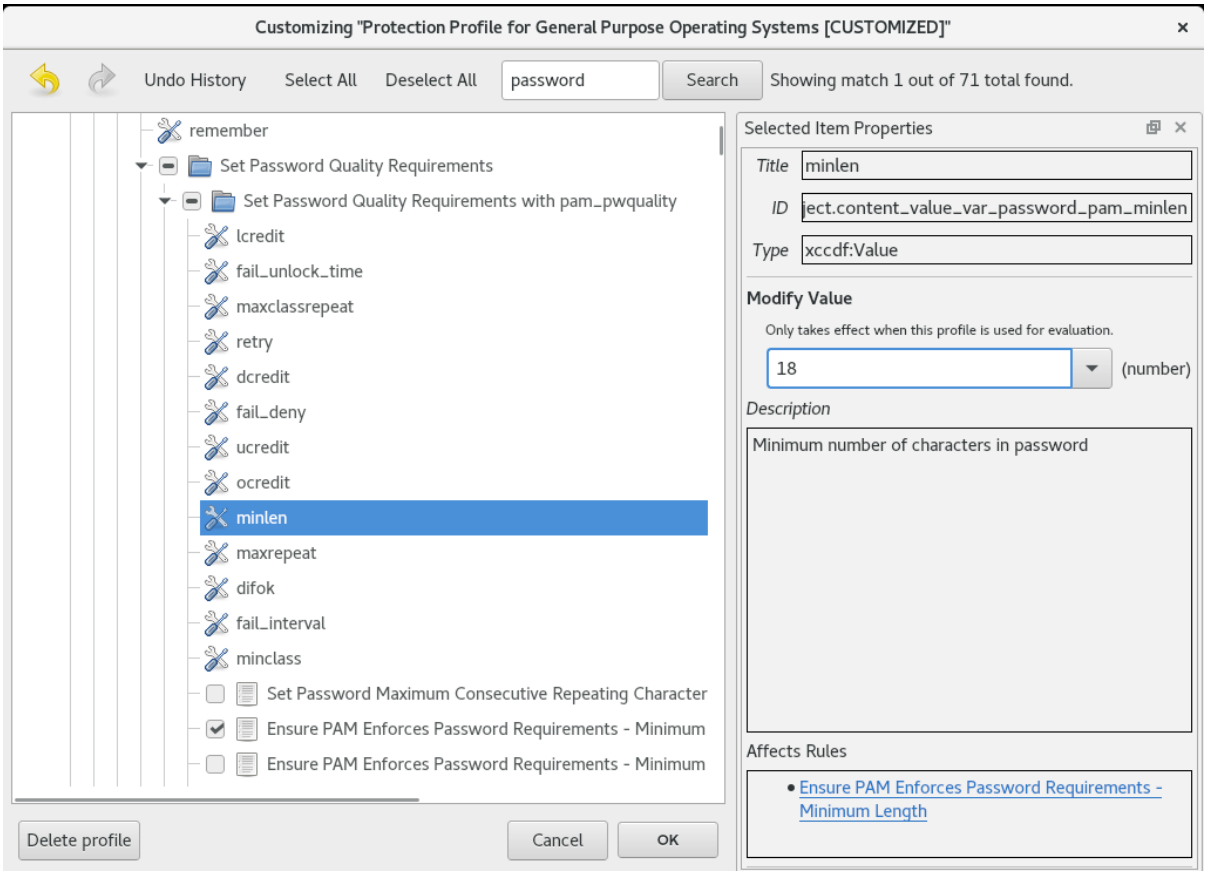
1. **SCAP Workbench** を実行し、**Open content from SCAP Security Guide** または **File** メニューの **Open Other Content** を使用してカスタマイズするプロファイルを選択します。

2. 選択したセキュリティープロファイルを必要に応じて調整するには、**Customize** ボタンをクリックします。
これにより、元のデータストリームファイルを変更せずに現在選択されているプロファイルを変更できる新しいカスタマイズウィンドウが開きます。新しいプロファイル ID を選択します。



3. 論理グループに分けられたルールを持つツリー構造を使用するか、**Search** フィールドを使用して変更するルールを検索します。

4. ツリー構造のチェックボックスを使用した include ルールまたは exclude ルール、または必要に応じてルールの値を変更します。



5. **OK** ボタンをクリックして変更を確認します。

6. 変更内容を永続的に保存するには、以下のいずれかのオプションを使用します。

   - **File** メニューの **Save Customization Only** を使用して、カスタマイズファイルを別途保存します。

- **File** メニュー **Save All** を選択して、すべてのセキュリティーコンテンツを一度に保存します。
  Into a directory オプションを選択すると、**SCAP Workbench** は、データストリームファイルおよびカスタマイズファイルの両方を、指定した場所に保存します。これはバックアップソリューションとして使用できます。

  As RPM オプションを選択すると、**SCAP Workbench** に、データストリームファイル、ならびにカスタマイズファイルを含む RPM パッケージの作成を指示できます。これは、リモートでスキャンできないシステムにセキュリティーコンテンツを配布したり、詳細な処理のためにコンテンツを配信するのに便利です。

注記

SCAP Workbench は、カスタマイズしたプロファイル向けの結果ベースの修正に対応していないため、**oscap** コマンドラインユーティリティーでエクスポートした修正を使用します。

16.9.3. 関連情報

- **scap-workbench (8)** の man ページ
- **scap-workbench** パッケージで提供される **/usr/share/doc/scap-workbench/user_manual.html** ファイル
- カスタマイズされた SCAP ポリシーを Satellite 6.x KCS でデプロイする 記事

16.10. コンテナーおよびコンテナーイメージの脆弱性スキャン

以下の手順を使用して、コンテナーまたはコンテナーイメージのセキュリティー脆弱性を検索します。

注記

oscap-podman コマンドは、RHEL 8.2 で利用できます。RHEL 8.1 および 8.0 の場合は、ナレッジベースの記事 Using OpenSCAP for scanning containers in RHEL 8 で説明されている回避策を利用します。

前提条件

- **openscap-utils** および **bzip2** パッケージがインストールされます。

手順

1. システムに最新 RHSA OVAL 定義をダウンロードします。

```
# wget -O - https://www.redhat.com/security/data/oval/v2/RHEL8/rhel-8.oval.xml.bz2 | bzip2 --decompress > rhel-8.oval.xml
```

2. コンテナーまたはコンテナーイメージの ID を取得します。以下に例を示します。

```
# podman images
REPOSITORY                          TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi8/ubi   latest   096cae65a207   7 weeks ago   239 MB
```

3. コンテナーまたはコンテナーイメージで脆弱性をスキャンし、結果を **vulnerability.html** ファイルに保存します。

```
# oscap-podman 096cae65a207 oval eval --report vulnerability.html rhel-8.oval.xml
```

oscap-podman コマンドには root 権限が必要で、コンテナーの ID は最初の引数であることに注意してください。

検証

- 結果をブラウザーで確認します。以下に例を示します。

```
$ firefox vulnerability.html &
```

関連情報

- 詳細は、**oscap-podman(8)** および **oscap(8)** の man ページを参照してください。

16.11. 特定のベースラインを使用したコンテナーまたはコンテナーイメージのセキュリティーコンプライアンスの評価

以下の手順に従い、OSPP (Operating System Protection Profile) や PCI-DSS (Payment Card Industry Data Security Standard)、Health Insurance Portability and Accountability Act (HIPAA) などの特定のセキュリティーベースラインのあるコンテナーまたはコンテナーイメージのコンプライアンスを評価します。

注記

oscap-podman コマンドは、RHEL 8.2 で利用できます。RHEL 8.1 および 8.0 の場合は、ナレッジベースの記事 Using OpenSCAP for scanning containers in RHEL 8 で説明されている回避策を利用します。

前提条件

- **openscap-utils** パッケージおよび **scap-security-guide** パッケージがインストールされている。

手順

1. コンテナーまたはコンテナーイメージの ID を取得します。以下に例を示します。

```
# podman images
REPOSITORY                         TAG      IMAGE ID       CREATED       SIZE
registry.access.redhat.com/ubi8/ubi   latest   096cae65a207   7 weeks ago   239 MB
```

2. HIPAA プロファイルでコンテナーイメージのコンプライアンスを評価し、スキャン結果を **report.html** ファイルに保存します。

```
# oscap-podman 096cae65a207 xccdf eval --report report.html --profile hipaa
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

OSPP または PCI-DSS ベースラインでセキュリティーコンプライアンスを評価する場合は、**096cae65a207** をコンテナーイメージの ID に、 **hipaa** の値を **ospp** または **pci-dss** に置き換えます。**oscap-podman** コマンドには、root 権限が必要なことに注意してください。

検証

- 結果をブラウザーで確認します。以下に例を示します。

```
$ firefox report.html &
```

注記

notapplicable が付いているルールは、コンテナー化されたシステムには適用されないルールです。これらのルールは、ベアメタルおよび仮想化システムにのみ適用されます。

関連情報

- **oscap-podman(8)** および **scap-security-guide(8)** の man ページ。
- **/usr/share/doc/scap-security-guide/** ディレクトリー。

16.12. AIDE で整合性の確認

AIDE (Advanced Intrusion Detection Environment) は、システムのファイルのデータベースを作成し、そのデータベースを使用してファイルの整合性を確保し、システムの侵入を検出します。

16.12.1. AIDE のインストール

以下の手順は、**AIDE** をインストールして、そのデータベースを開始するのに必要です。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **aide** 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 aide
```

2. 初期データベースを生成するには、次のコマンドを実行します。

```
# aide --init
```

注記

デフォルト設定では、**aide --init** コマンドは、**/etc/aide.conf** ファイルで定義するディレクトリーとファイルのセットのみを確認します。ディレクトリーまたはファイルを **AIDE** データベースに追加し、監視パラメーターを変更するには、**/etc/aide.conf** を変更します。

3. データベースの使用を開始するには、初期データベースのファイル名から末尾の **.new** を削除します。

```
# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

4. **AIDE** データベースの場所を変更するには、**/etc/aide.conf** ファイルを編集して、**DBDIR** 値を変更します。追加のセキュリティーのデータベース、設定、**/usr/sbin/aide** バイナリーファイルを、読み取り専用メディアなどの安全な場所に保存します。

16.12.2. AIDE を使用した整合性チェックの実行

前提条件

- **AIDE** が適切にインストールされ、そのデータベースが初期化されている。AIDE のインストール を参照してください。

手順

1. 手動でチェックを開始するには、以下を行います。

```
# aide --check
Start timestamp: 2018-07-11 12:41:20 +0200 (AIDE 0.16)
AIDE found differences between database and filesystem!!
...
[trimmed for clarity]
```

2. 最低でも、**AIDE** は週ごとに実行するようにシステムを設定します。最適な設定としては、**AIDE** を毎日実行します。たとえば、**AIDE** を毎日午前 04:05 に実行するようにスケジュールするには、**cron** コマンドを使用して、次の行を **/etc/crontab** ファイルを追加します。

```
05 4 * * * root /usr/sbin/aide --check
```

16.12.3. AIDE データベースの更新

Red Hat は、システムの変更 (パッケージの更新、設定ファイルの修正など) を確認してから、基本となる **AIDE** データベースを更新することを推奨します。

前提条件

- **AIDE** が適切にインストールされ、そのデータベースが初期化されている。AIDE のインストール を参照してください。

手順

1. 基本となる **AIDE** データベースを更新します。

```
# aide --update
```

aide --update コマンドは、**/var/lib/aide/aide.db.new.gz** データベースファイルを作成します。

2. 整合性チェックで更新したデータベースを使用するには、ファイル名から末尾の **.new** を削除します。

16.12.4. ファイル整合性ツール:AIDE および IMA

Red Hat Enterprise Linux は、システム上のファイルとディレクトリーの整合性をチェックおよび保持するためのさまざまなツールを提供します。次の表は、シナリオに適したツールを決定するのに役立ちます。

表16.2 AIDE と IMA の比較

比較項目	Advanced Intrusion Detection Environment (AIDE)	Integrity Measurement Architecture (IMA)
確認対象	AIDE は、システム上のファイルとディレクトリーのデータベースを作成するユーティリティーです。このデータベースは、ファイルの整合性をチェックし、侵入を検出するのに役立ちます。	IMA は、以前に保存された拡張属性と比較してファイル測定値 (ハッシュ値) をチェックすることにより、ファイルが変更されているかどうかを検出します。
確認方法	AIDE はルールを使用して、ファイルとディレクトリーの整合性状態を比較します。	IMA は、ファイルハッシュ値を使用して侵入を検出します。
理由	検出- AIDE は、ルールを検証することにより、ファイルが変更されているかどうかを検出します。	検出と防止- IMA は、ファイルの拡張属性を置き換えることにより、攻撃を検出および防止します。
Usage	AIDE は、ファイルまたはディレクトリーが変更されたときに脅威を検出します。	誰かがファイル全体の変更を試みた時に、IMA は脅威を検出します。
範囲	AIDE は、ローカルシステム上のファイルとディレクトリーの整合性をチェックします。	IMA は、ローカルシステムとリモートシステムのセキュリティーを確保します。

16.12.5. 関連情報

- **aide(1)** の man ページ
- Kernel integrity subsystem

16.13. LUKS を使用したブロックデバイスの暗号化

ディスクの暗号化は、それを暗号化することにより、ブロックデバイス上のデータを保護します。デバイスで復号したコンテンツにアクセスするには、パスフレーズまたは鍵を認証として提供する必要があります。これは、モバイルコンピューターや、リムーバブルメディアの場合に特に重要になります。これにより、デバイスをシステムから物理的に削除した場合でも、デバイスのコンテンツを保護するのに役立ちます。LUKS 形式は、RHEL におけるブロックデバイスの暗号化のデフォルト実装です。

16.13.1. LUKS ディスクの暗号化

LUKS (Linux Unified Key Setup-on-disk-format) は、ブロックデバイスを暗号化でき、暗号化したデバイスの管理を簡素化するツールセットを提供します。LUKS を使用すれば、複数のユーザー鍵が、パーティションのバルク暗号化に使用されるマスター鍵を複号できるようになります。

RHEL は、LUKS を使用してブロックデバイスの暗号化を行います。デフォルトではインストール時に、ブロックデバイスを暗号化するオプションが指定されていません。ディスクを暗号化するオプションを選択すると、コンピューターを起動するたびにパスフレーズの入力が求められます。このパスフレーズは、パーティションの複号に用いられるバルク暗号化鍵のロックを解除します。デフォルトのパーティションテーブルの変更を選択すると、暗号化するパーティションを選択できます。この設定は、パーティションテーブル設定で行われます。

LUKS の機能

- LUKS は、ブロックデバイス全体を暗号化するため、脱着可能なストレージメディアやノート PC のディスクドライブといった、モバイルデバイスのコンテンツを保護するのに適しています。
- 暗号化されたブロックデバイスの基本的な内容は任意であり、スワップデバイスの暗号化に役立ちます。また、とりわけデータストレージ用にフォーマットしたブロックデバイスを使用する特定のデータベースに関しても有用です。
- LUKS は、既存のデバイスマッパーのカーネルサブシステムを使用します。
- LUKS はパスフレーズのセキュリティーを強化し、辞書攻撃から保護します。
- LUKS デバイスには複数のキースロットが含まれ、ユーザーはこれを使用してバックアップキーやパスフレーズを追加できます。

LUKS が行わない こと

- LUKS などのディスク暗号化ソリューションは、システムの停止時にしかデータを保護しません。システムの電源がオンになり、LUKS がディスクを復号すると、そのディスクのファイルは、通常、そのファイルにアクセスできるすべてのユーザーが使用できます。
- LUKS は、多くのユーザーが、同じデバイスにアクセスする鍵をそれぞれ所有することが必要となるシナリオには適していません。LUKS1 形式は鍵スロットを 8 個提供し、LUKS2 形式は鍵スロットを最大 32 個提供します。
- LUKS は、ファイルレベルの暗号化を必要とするアプリケーションには適していません。

暗号化

LUKS に使用されるデフォルトの暗号は **aes-xts-plain64** です。LUKS のデフォルトの鍵サイズは 512 ビットです。**Anaconda** (XTS モード) を使用した LUKS のデフォルトの鍵サイズは 512 ビットです。利用可能な暗号は以下のとおりです。

- AES: Advanced Encryption Standard
- Twofish (128 ビットブロック暗号)
- Serpent

関連情報

- LUKS プロジェクトのホームページ
- LUKS オンディスクフォーマットの仕様

- FIPS PUB 197

16.13.2. RHEL の LUKS バージョン

RHEL では、LUKS 暗号化のデフォルト形式は LUKS2 です。従来の LUKS1 形式は、完全にサポートされ、以前の RHEL リリースと互換性のある形式として提供されます。

LUKS2 形式は、今後も、バイナリー構造を変更することなく、さまざまな要素を更新できるように設計されています。LUKS2 は、内部的にメタデータに JSON テキスト形式を使用し、メタデータの冗長性を提供し、メタデータの破損を検出し、メタデータコピーからの自動修正を可能にします。

重要

LUKS1 にのみ対応する以前のシステムとの互換性を必要とするシステムでは、LUKS2 を使用しないでください。RHEL 7 は、バージョン 7.6 以降の LUKS2 形式に対応していることに注意してください。

警告

LUKS2 および LUKS1 は、異なるコマンドを使用してディスクを暗号化します。LUKS バージョンに誤ったコマンドを使用すると、データが失われる可能性があります。

LUKS バージョン	暗号化コマンド
LUKS2	**cryptsetup reencrypt**
LUKS1	**cryptsetup-reencrypt**

オンラインの再暗号化

LUKS2 形式は、デバイスが使用中の間に、暗号化したデバイスの再暗号化に対応します。たとえば、以下のタスクを実行するにあたり、デバイスでファイルシステムをアンマウントする必要はありません。

- ボリュームキーの変更
- 暗号化アルゴリズムの変更

暗号化されていないデバイスを暗号化する場合は、ファイルシステムのマウントを解除する必要があります。暗号化の短い初期化後にファイルシステムを再マウントできます。

LUKS1 形式は、オンライン再暗号化に対応していません。

変換

LUKS2 形式は、LUKS1 により提供されます。特定の状況では、LUKS1 を LUKS2 に変換できます。具体的には、以下のシナリオでは変換ができません。

- LUKS1 デバイスが、Policy-Based Decryption (PBD - Clevis) ソリューションにより使用されているとマークされている。**cryptsetup** ツールは、**luksmeta** メタデータが検出されると、そのデバイスを変換することを拒否します。
- デバイスがアクティブになっている。デバイスが非アクティブ状態でなければ、変換することはできません。

16.13.3. LUKS2 再暗号化中のデータ保護のオプション

LUKS2 では、再暗号化プロセスで、パフォーマンスやデータ保護の優先度を設定する複数のオプションを選択できます。

checksum

これがデフォルトのモードです。データ保護とパフォーマンスのバランスを取ります。
このモードは、セクターの個々のチェックサムを再暗号化領域に保存するため、復旧プロセスでは、LUKS2 がすでに再暗号化しているセクターを検出できます。このモードでは、ブロックデバイスセクターの書き込みがアトミックである必要があります。

journal

このモードが最も安全ですが、最も遅くなります。このモードは、バイナリー領域の再暗号化領域をジャーナル化するため、LUKS2 はデータを 2 回書き込みます。

none

このモードはパフォーマンスを優先し、データ保護は提供しません。これは、**SIGTERM** シグナルや、**Ctrl+C** を押すユーザーなど、安全なプロセス終了からのみデータを保護します。予期しないシステムクラッシュやアプリケーションのクラッシュが発生すると、データが破損する可能性があります。

cryptsetup の **--resilience** オプションを使用してモードを選択できます。

LUKS2 の再暗号化プロセスが強制的に突然終了した場合、LUKS2 は以下のいずれかの方法で復旧を実行できます。

- 自動的に実行 (LUKS2 デバイスの次回のオープン動作時)。この動作は、**cryptsetup open** コマンドまたは **systemd-cryptsetup** でデバイスを割り当てると発生します。
- LUKS2 デバイスで **cryptsetup repair** コマンドを使用して手動で実行。

16.13.4. LUKS2 を使用したブロックデバイスの既存データの暗号化

この手順では、LUKS2 形式を使用して、暗号化されていないデバイスの既存データを暗号化します。新しい LUKS ヘッダーは、デバイスのヘッドに保存されます。

前提条件

- ブロックデバイスにファイルシステムが含まれている。
- データのバックアップを作成している。

警告

ハードウェア、カーネル、または人的ミスにより、暗号化プロセス時にデータが失われる場合があります。データの暗号化を開始する前に、信頼性の高いバックアップを作成してください。

手順

1. 暗号化するデバイスにあるファイルシステムのマウントをすべて解除します。以下に例を示します。

```
# umount /dev/sdb1
```

2. LUKS ヘッダーを保存するための空き容量を確認します。以下のいずれかのオプションを選択します。

   - 論理ボリュームを暗号化する場合は、以下のように、ファイルシステムのサイズを変更せずに、論理ボリュームを拡張できます。以下に例を示します。

```
# lvextend -L+32M vg00/lv00
```

   - **parted** などのパーティション管理ツールを使用してパーティションを拡張します。
   - このデバイスのファイルシステムを縮小します。ext2、ext3、または ext4 のファイルシステムには **resize2fs** ユーティリティーを使用できます。XFS ファイルシステムは縮小できないことに注意してください。

3. 暗号化を初期化します。以下に例を示します。

```
# cryptsetup reencrypt \
        --encrypt \
        --init-only \
        --reduce-device-size 32M \
        /dev/sdb1 sdb1_encrypted
```

   このコマンドを実行するとパスフレーズの入力が求められ、暗号化プロセスが開始します。

4. デバイスをマウントします。

```
# mount /dev/mapper/sdb1_encrypted /mnt/sdb1_encrypted
```

5. 永続的なマッピングのエントリーを **/etc/crypttab** に追加します。

   a. **luksUUID** を見つけます。

```
# cryptsetup luksUUID /dev/mapper/sdb1_encrypted
```

   これにより、選択したデバイスの **luksUUID** が表示されます。

b. 任意のテキストエディターで **/etc/crypttab** ファイルを開き、このファイルにデバイスを追加します。

```
$ vi /etc/crypttab
```

```
/dev/mapper/sdb1_encrypted luks_uuid none
```

c. **dracut** で initramfs を更新します。

```
$ dracut -f --regenerate-all
```

6. **/etc/fstab** ファイルに永続的なマウントのエントリーを追加します。

a. アクティブな LUKS ブロックデバイスの **FS UUID** を見つけます。

```
$ blkid  -p /dev/mapper/sdb1_encrypted
```

b. 任意のテキストエディターで **/etc/fstab** ファイルを開き、このファイルにデバイスを追加します。次に例を示します。

```
$ vi  /etc/fstab
fs__uuid /home auto rw,user,auto 0 0
```

7. オンライン暗号化を開始します。

```
# cryptsetup reencrypt --resume-only /dev/sdb1
```

関連情報

- **cryptsetup (8)**、**lvextend (8)**、**resize2fs(8)**、および **parted(8)** の man ページ

16.13.5. 独立したヘッダーがある LUKS2 を使用してブロックデバイスの既存データの暗号化

この手順では、LUKS ヘッダーを保存するための空き領域を作成せずに、ブロックデバイスの既存データを暗号化します。ヘッダーは、追加のセキュリティー層としても使用できる、独立した場所に保存されます。この手順では、LUKS2 暗号化形式を使用します。

前提条件

- ブロックデバイスにファイルシステムが含まれている。
- データのバックアップを作成している。

警告

ハードウェア、カーネル、または人的ミスにより、暗号化プロセス時にデータが失われる場合があります。データの暗号化を開始する前に、信頼性の高いバックアップを作成してください。

手順

1. プールにあるファイルシステムのマウントをすべて解除します。以下に例を示します。

```
# umount /dev/sdb1
```

2. 暗号化を初期化します。

```
# cryptsetup reencrypt \
        --encrypt \
        --init-only \
        --header /path/to/header \
        /dev/sdb1 sdb1_encrypted
```

/path/to/header を、独立した LUKS ヘッダーのあるファイルへのパスに置き換えます。暗号化したデバイスを後でアンロックできるように、接続解除した LUKS ヘッダーにアクセスできる必要があります。

このコマンドを実行するとパスフレーズの入力が求められ、暗号化プロセスが開始します。

3. デバイスをマウントします。

```
# mount /dev/mapper/sdb1_encrypted /mnt/sdb1_encrypted
```

4. オンライン暗号化を開始します。

```
# cryptsetup reencrypt --resume-only --header /path/to/header /dev/sdb1
```

関連情報

- **cryptsetup(8)** の man ページ

16.13.6. LUKS2 を使用した空のブロックデバイスの暗号化

この手順では、LUKS2 形式を使用して空のブロックデバイスを暗号化する方法を説明します。

前提条件

- 空のブロックデバイス。

手順

1. 暗号化した LUKS パーティションとしてパーティションを設定します。

```
# cryptsetup luksFormat /dev/sdb1
```

2. 暗号化した LUKS パーティションを開きます。

```
# cryptsetup open /dev/sdb1 sdb1_encrypted
```

これにより、パーティションのロックが解除され、デバイスマッパーを使用して新しいデバイスにマッピングされます。これは、 **デバイス** が暗号化されたデバイスであり、暗号化されたデータを上書きしないように **/dev/mapper/device_mapped_name** を使用して LUKS を通じてアドレス指定する必要があることをカーネルに警告します。

3. パーティションに暗号化されたデータを書き込むには、デバイスをマッピングした名前でアクセスする必要があります。これを実行するには、ファイルシステムを作成する必要があります。以下に例を示します。

```
# mkfs -t ext4 /dev/mapper/sdb1_encrypted
```

4. デバイスをマウントします。

```
# mount /dev/mapper/sdb1_encrypted mount-point
```

関連情報

- **cryptsetup(8)** の man ページ

16.13.7. storage RHEL System Role を使用して LUKS 暗号化ボリュームを作成する

storage ロールを使用し、Ansible Playbook を実行して、LUKS で暗号化されたボリュームを作成および設定できます。

前提条件

- **crypto_policies** システムロールで設定するシステムである 1 つ以上の **管理対象ノード** へのアクセスとパーミッション。
- コントロールノード (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- マネージドノードが記載されているインベントリーファイルがある。

手順

1. 以下の内容を含む新しい **playbook.yml** ファイルを作成します。

```
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: your-password
  roles:
   - rhel-system-roles.storage
```

2. オプション:Playbook の構文を確認します。

```
# ansible-playbook --syntax-check playbook.yml
```

3. インベントリーファイルで Playbook を実行します。

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

関連情報

- LUKS を使用したブロックデバイスの暗号化
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** file

16.14. ポリシーベースの複号を使用して暗号化ボリュームの自動アンロックの設定

ポリシーベースの複号 (PBD) は、物理マシンおよび仮想マシンにおいて、ハードドライブで暗号化した root ボリュームおよびセカンダリーボリュームのロックを解除できるようにする一連の技術です。PBD は、ユーザーパスワード、TPM (Trusted Platform Module) デバイス、システムに接続する PKCS #11 デバイス (たとえばスマートカード) などのさまざまなロックの解除方法、もくしは特殊なネットワークサーバーを使用します。

PBD を使用すると、ポリシーにさまざまなロックの解除方法を組み合わせて、さまざまな方法で同じボリュームのロックを解除できるようにすることができます。RHEL における PBD の現在の実装は、Clevis フレームワークと、**ピン** と呼ばれるプラグインで構成されます。各ピンは、個別のアンロック機能を提供します。現在利用できるピンは以下のとおりです。

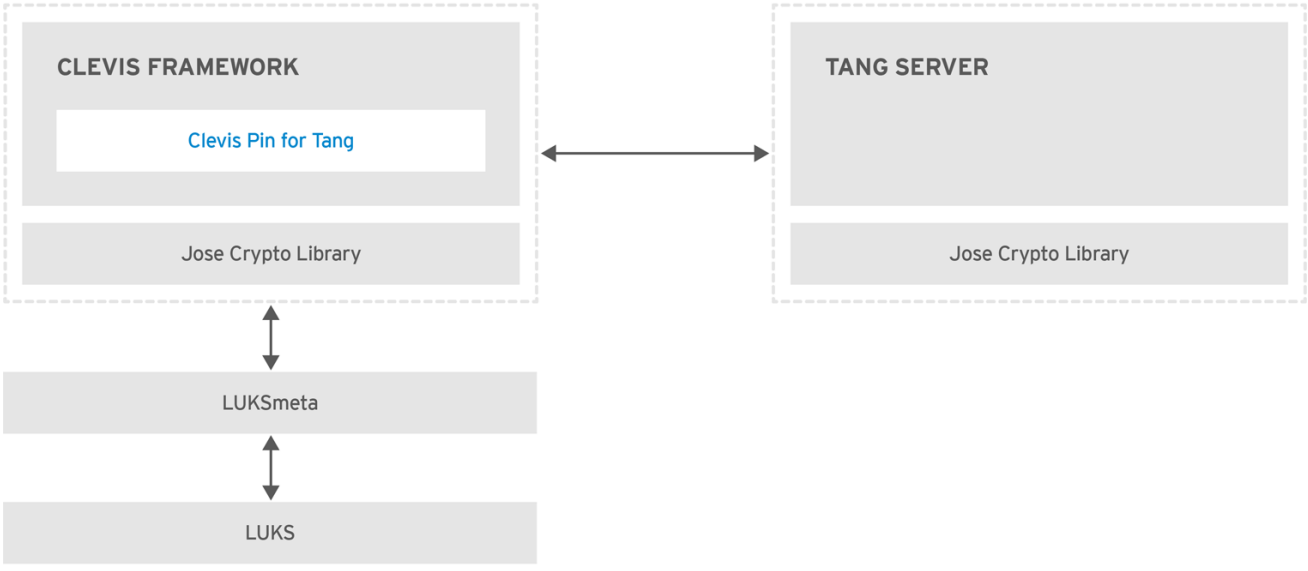
- **tang** - ネットワークサーバーを使用してボリュームのロックを解除できます
- **tpm2** - TPM2 ポリシーを使用してボリュームのロックを解除できます
- **sss** - Shamir's Secret Sharing (SSS) 暗号方式を使用して高可用性システムをデプロイできます

16.14.1. NBDE (Network-Bound Disk Encryption)

Network Bound Disc Encryption (NBDE) は、ポリシーベースの復号 (PBD) のサブカテゴリーであり、暗号化されたボリュームを特別なネットワークサーバーにバインドできるようにします。NBDE の現在の実装には、Tang サーバー自体と、Tang サーバー用の Clevis ピンが含まれます。

RHEL では、NBDE は次のコンポーネントとテクノロジーによって実装されます。

図16.1 LUKS1 で暗号化したボリュームを使用する場合の NBDE スキーム(luksmeta パッケージは、LUKS2 ボリュームには使用されません)



Tang は、ネットワークのプレゼンスにデータをバインドするためのサーバーです。セキュリティーが保護された特定のネットワークにシステムをバインドする際に利用可能なデータを含めるようにします。Tang はステートレスで、TLS または認証は必要ありません。エスクローベースのソリューション (サーバーが暗号鍵をすべて保存し、使用されたことがあるすべての鍵に関する知識を有する) とは異なり、Tang はクライアントの鍵と相互作用することはないため、クライアントから識別情報を得ることがありません。

Clevis は、自動化された復号用のプラグイン可能なフレームワークです。NBDE では、Clevis は、LUKS ボリュームの自動アンロックを提供します。**clevis** パッケージは、クライアントで使用される機能を提供します。

Clevis ピン は、Clevis フレームワークへのプラグインです。このようなピンの 1 つは、NBDE サーバー (Tang) との相互作用を実装するプラグインです。

Clevis および Tang は、一般的なクライアントおよびサーバーのコンポーネントで、ネットワークがバインドされた暗号化を提供します。RHEL では、LUKS と組み合わせて使用され、ルートおよび非ルートストレージボリュームを暗号化および復号して、ネットワークにバインドされたディスク暗号化を実現します。

クライアントおよびサーバーのコンポーネントはともに **José** ライブラリーを使用して、暗号化および複号の操作を実行します。

NBDE のプロビジョニングを開始すると、Tang サーバーの Clevis ピンは、Tang サーバーの、アドバタイズされている非対称鍵のリストを取得します。もしくは、鍵が非対称であるため、Tang の公開鍵のリストを帯域外に配布して、クライアントが Tang サーバーにアクセスしなくても動作できるようにします。このモードは **オフラインプロビジョニング** と呼ばれます。

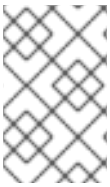
Tang 用の Clevis ピンは、公開鍵のいずれかを使用して、固有で、暗号論的に強力な暗号鍵を生成します。この鍵を使用してデータを暗号化すると、この鍵は破棄されます。Clevis クライアントは、使いやすい場所に、このプロビジョニング操作で生成した状態を保存する必要があります。データを暗号化するこのプロセスは **プロビジョニング手順** と呼ばれています。

LUKS バージョン 2 (LUKS2) は、RHEL のデフォルトのディスク暗号化形式であるため、NBDE のプロビジョニング状態は、LUKS2 ヘッダーにトークンとして保存されます。**luksmeta** パッケージによる NBDE のプロビジョニング状態は、LUKS1 で暗号化したボリュームにのみ使用されます。

Tang 用の Clevis ピンは、規格を必要とせずに LUKS1 と LUKS2 の両方をサポートします。Clevis はプレーンテキストファイルを暗号化できますが、ブロックデバイスの暗号化には **cryptsetup** ツールを使用する必要があります。詳細については、 Encrypting block devices using LUKS を参照してください。

クライアントがそのデータにアクセスする準備ができると、プロビジョニング手順で生成したメタデータを読み込み、応答して暗号鍵を戻します。このプロセスは **復旧手順** と呼ばれます。

Clevis は、NBDE ではピンを使用して LUKS ボリュームをバインドしているため、自動的にロックが解除されます。バインドプロセスが正常に終了すると、提供されている Dracut アンロックを使用してディスクをアンロックできます。



注記

kdump カーネルクラッシュのダンプメカニズムが、システムメモリーのコンテンツを LUKS で暗号化したデバイスに保存するように設定されている場合には、2 番目のカーネル起動時にパスワードを入力するように求められます。

関連情報

- NBDE (Network-Bound Disk Encryption) テクノロジーの ナレッジベース記事
- **tang(8)**、**clevis(1)**、**jose(1)** および **clevis-luks-unlockers(7)** の man ページ
- ナレッジベースの記事 How to set up Network-Bound Disk Encryption with multiple LUKS devices(Clevis + Tang unlocking)

16.14.2. 暗号化クライアント (Clevis) のインストール

この手順に従って、システムに Clevis プラグ可能フレームワークを使用してデプロイと起動を行います。

手順

1. 暗号化されたボリュームを持つシステムに Clevis とそのピン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 clevis
```

2. データを複号するには、**clevis decrypt** コマンドを実行して、JWE (JSON Web Encryption) 形式で暗号文を指定します。以下に例を示します。

```
$ clevis decrypt < secret.jwe
```

関連情報

- **clevis(1)** の man ページ
- 引数を指定せずに **clevis** コマンドを実行した後の組み込み CLI ヘルプ

```
$ clevis
Usage: clevis COMMAND [OPTIONS]

clevis decrypt      Decrypts using the policy defined at encryption time
clevis encrypt sss  Encrypts using a Shamir's Secret Sharing policy
clevis encrypt tang Encrypts using a Tang binding server policy
clevis encrypt tpm2 Encrypts using a TPM2.0 chip binding policy
clevis luks bind    Binds a LUKS device using the specified policy
clevis luks edit    Edit a binding from a clevis-bound slot in a LUKS device
clevis luks list    Lists pins bound to a LUKSv1 or LUKSv2 device
clevis luks pass    Returns the LUKS passphrase used for binding a particular slot.
clevis luks regen   Regenerate clevis binding
clevis luks report  Report tang keys' rotations
clevis luks unbind  Unbinds a pin bound to a LUKS volume
clevis luks unlock  Unlocks a LUKS volume
```

16.14.3. SELinux を Enforcing モードで有効にした Tang サーバーのデプロイメント

この手順では、Enforcing モードの SELinux で限定サービスとして、カスタムポートで実行する Tang サーバーをデプロイします。

前提条件

- **policycoreutils-python-utils** パッケージおよび依存関係がインストールされている。
- **firewalld** サービスが実行している。

手順

1. **tang** パッケージとその依存関係をインストールするには、**root** で以下のコマンドを実行します。

```
# yum install tang
```

2. 7500/tcp などの不要なポートを選択し、**tangd** サービスがそのポートにバインドできるようにします。

```
# semanage port -a -t tangd_port_t -p tcp 7500
```

ポートは1つのサービスのみで一度に使用できるため、すでに使用しているポートを使用しようとすると、**ValueError:Port already defined** エラーが発生します。

3. ファイアウォールのポートを開きます。

```
# firewall-cmd --add-port=7500/tcp
# firewall-cmd --runtime-to-permanent
```

4. **tangd** サービスを有効にします。

```
# systemctl enable tangd.socket
```

5. オーバーライドファイルを作成します。

```
# systemctl edit tangd.socket
```

6. 以下のエディター画面で、**/etc/systemd/system/tangd.socket.d/** ディレクトリーにある空の **override.conf** ファイルを開き、次の行を追加して、Tang サーバーのデフォルトのポートを、80 から、以前取得した番号に変更します。

```
[Socket]
ListenStream=
ListenStream=7500
```

ファイルを保存して、エディターを終了します。

7. 変更した設定を再読み込みします。

```
# systemctl daemon-reload
```

8. 設定が機能していることを確認します。

```
# systemctl show tangd.socket -p Listen
Listen=[::]:7500 (Stream)
```

9. **tangd** サービスを開始します。

```
# systemctl restart tangd.socket
```

tangd が、**systemd** のソケットアクティベーションメカニズムを使用しているため、最初に接続するとすぐにサーバーが起動します。最初の起動時に、一組の暗号鍵が自動的に生成されます。鍵の手動生成などの暗号化操作を実行するには、**jose** ユーティリティーを使用します。

関連情報

- **tang(8)**、**semanage(8)**、**firewall-cmd(1)**、**jose(1)**、**systemd.unit(5)** および **systemd.socket(5)** の man ページ

16.14.4. Tang サーバーの鍵のローテーションおよびクライアントでのバインディングの更新

以下の手順に従って、Tang サーバーの鍵をローテーションし、クライアントの既存のバインディングを更新します。鍵をローテートするのに適した間隔は、アプリケーション、鍵のサイズ、および組織のポリシーにより異なります。

したがって、**nbde_server** RHEL システムロールを使用して、Tang 鍵をローテーションできます。詳細は 複数の Tang サーバー設定での nbde_server システムロールの使用 を参照してください。

前提条件

- Tang サーバーが実行している。
- **clevis** パッケージおよび **clevis-luks** パッケージがクライアントにインストールされている。
- RHEL 8.2 に、**clevis luks list**、**clevis luks report**、および **clevis luks regen** が追加されていることに注意してください。

手順

1. **/var/db/tang** 鍵データベースディレクトリーのすべての鍵の名前の前に **.** を指定して、アドバタイズメントに対して非表示にします。以下の例のファイル名は、Tang サーバーの鍵データベースディレクトリーにある一意のファイル名とは異なります。

```
# cd /var/db/tang
# ls -l
-rw-r--r--. 1 root root 349 Feb  7 14:55 UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
-rw-r--r--. 1 root root 354 Feb  7 14:55 y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
# mv UV6dqXSwe1bRKG3KbJmdiR020hY.jwk .UV6dqXSwe1bRKG3KbJmdiR020hY.jwk
# mv y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk .y9hxLTQSiSB5jSEGWnjhY8fDTJU.jwk
```

2. 名前が変更され、Tang サーバーのアドバタイズに対してすべての鍵が非表示になっていることを確認します。

```
# ls -l
total 0
```

3. Tang サーバーの **/var/db/tang** で **/usr/libexec/tangd-keygen** コマンドを使用して新しい鍵を生成します。

```
# /usr/libexec/tangd-keygen /var/db/tang
# ls /var/db/tang
3ZWS6-cDrCG61UPJS2BMmPU4I54.jwk zyLuX6hijUy_PSeUEFDi7hi38.jwk
```

4. Tang サーバーが、以下のように新規キーペアから署名キーを公開していることを確認します。

```
# tang-show-keys 7500
3ZWS6-cDrCG61UPJS2BMmPU4I54
```

5. NBDE クライアントで **clevis luks report** コマンドを使用して、Tang サーバーでアドバタイズされた鍵が同じままかどうかを確認します。**clevis luks list** コマンドを使用すると、関連するバインディングのあるスロットを特定できます。以下に例を示します。

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv"}'
# clevis luks report -d /dev/sda2 -s 1
```

```
...
Report detected that some keys were rotated.
Do you want to regenerate luks metadata with "clevis luks regen -d /dev/sda2 -s 1"? [ynYN]
```

6. 新しい鍵の LUKS メタデータを再生成するには、直前のコマンドプロンプトで **y** を押すか、**clevis luks regen** コマンドを使用します。

```
# clevis luks regen -d /dev/sda2 -s 1
```

7. すべての古いクライアントが新しい鍵を使用することを確認したら、Tang サーバーから古い鍵を削除できます。次に例を示します。

```
# cd /var/db/tang
# rm .*.jwk
```

警告

クライアントが使用している最中に古い鍵を削除すると、データが失われる場合があります。このような鍵を誤って削除した場合は、クライアントで **clevis luks regen** コマンドを実行し、LUKS パスワードを手動で提供します。

関連情報

- **tang-show-keys(1)**、**clevis-luks-list(1)**、**clevis-luks-report(1)**、および **clevis-luks-regen(1)** の man ページ

16.14.5. Web コンソールで Tang 鍵を使用した自動アンロックの設定

Tang サーバーが提供する鍵を使用して、LUKS で暗号化したストレージデバイスの自動ロック解除を設定します。

前提条件

- RHEL 8 Web コンソールがインストールされている。
  詳細は、Web コンソールのインストールおよび有効化 を参照してください。
- **cockpit-storaged** パッケージがシステムにインストールされている。
- **cockpit.socket** サービスがポート 9090 で実行されている。
- **clevis** パッケージ、**tang** パッケージ、および **clevis-dracut** パッケージがインストールされている。
- Tang サーバーが実行している。

手順

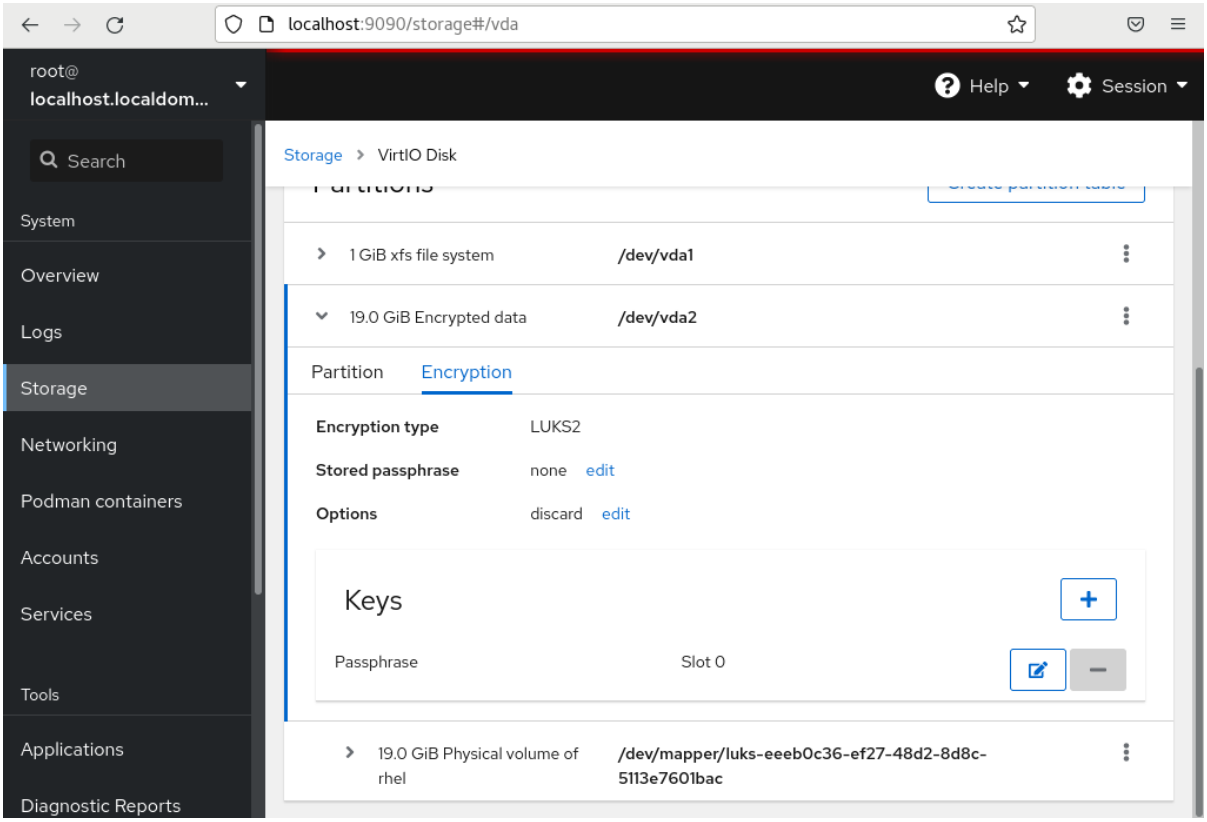
1. Web ブラウザーに以下のアドレスを入力して、RHEL Web コンソールを開きます。

> https://**localhost**:9090

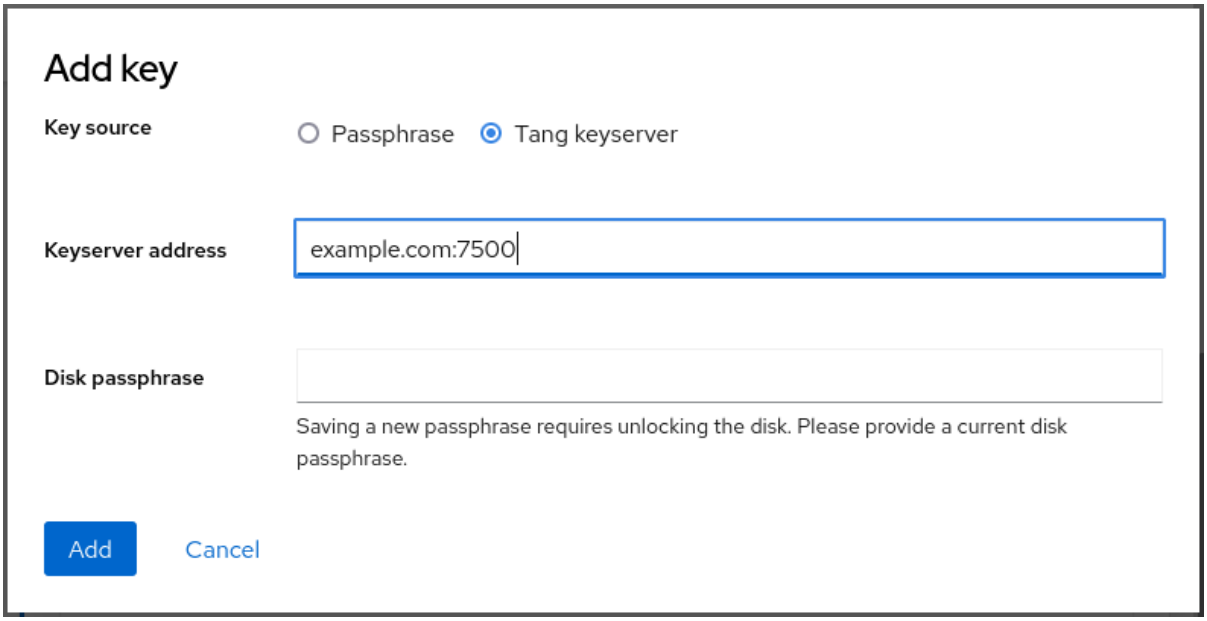
リモートシステムに接続する際に、**localhost** の部分をリモートサーバーのホスト名または IP アドレスに置き換えます。

2. 認証情報を指定して、**ストレージ** をクリックします。**>** をクリックして、Tang サーバーを使用してロックを解除する暗号化されたデバイスの詳細をデプロイメントし、**Encryption** をクリックします。

3. **Keys** セクションの **+** をクリックして Tang キーを追加します。



4. Tang サーバーのアドレスと、LUKS で暗号化したデバイスのロックを解除するパスワードを指定します。**Add** をクリックして確定します。

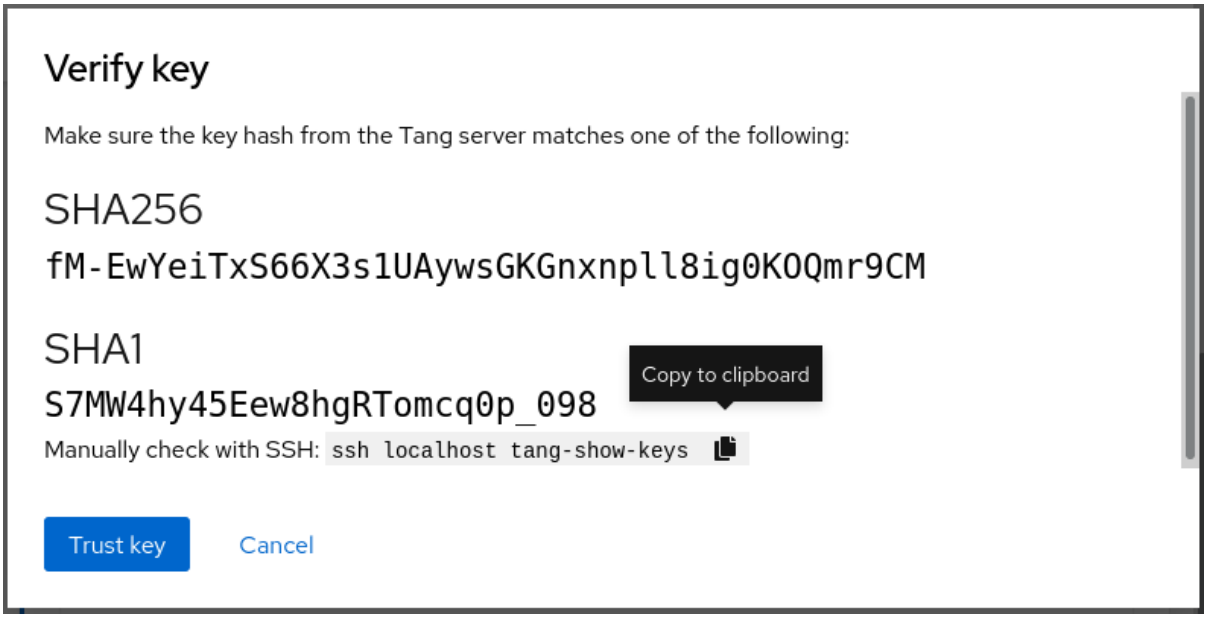


以下のダイアログウインドウは、鍵ハッシュが一致することを確認するコマンドを提供します。

5. Tang サーバーのターミナルで、**tang-show-keys** コマンドを使用して、比較のためにキーハッシュを表示します。この例では、Tang サーバーはポート 7500 で実行されています。

```
# tang-show-keys 7500
fM-EwYeiTxS66X3s1UAywsGKGnxnpll8ig0KOQmr9CM
```

6. Web コンソールと前述のコマンドの出力のキーハッシュが同じ場合は、**Trust key** をクリックします。



7. 初期ブートシステムでディスクバインディングを処理できるようにするには、左側のナビゲーションバーの下部にある **Terminal** をクリックし、次のコマンドを入力します。

```
# yum install clevis-dracut
# grubby --update-kernel=ALL --args="rd.neednet=1"
# dracut -fv --regenerate-all
```

検証

1. 新規に追加された Tang キーが **Keyserver** タイプの **Keys** セクションにリスト表示されていることを確認します。

2. バインディングが初期ブートで使用できることを確認します。次に例を示します。

```
# lsinitrd | grep clevis
clevis
clevis-pin-sss
clevis-pin-tang
clevis-pin-tpm2
-rwxr-xr-x   1 root     root         1600 Feb 11 16:30 usr/bin/clevis
-rwxr-xr-x   1 root     root         1654 Feb 11 16:30 usr/bin/clevis-decrypt
...
-rwxr-xr-x   2 root     root           45 Feb 11 16:30 usr/lib/dracut/hooks/initqueue/settled/60-clevis-hook.sh
-rwxr-xr-x   1 root     root         2257 Feb 11 16:30 usr/libexec/clevis-luks-askpass
```

関連情報

- ポリシーベースの複号を使用して暗号化ボリュームの自動アンロックの設定

16.14.6. 基本的な NBDE および TPM2 暗号化クライアント操作

Clevis フレームワークは、プレーンテキストファイルを暗号化し、JSON Web Encryption (JWE) 形式の暗号化テキストと LUKS 暗号化ブロックデバイスの両方を復号できます。Clevis クライアントは、暗号化操作に Tang ネットワークサーバーまたは Trusted Platform Module 2.0(TPM 2.0) チップのいずれかを使用できます。

次のコマンドは、プレーンテキストファイルが含まれる例で Clevis が提供する基本的な機能を示しています。また、NBDE または Clevis + TPM のデプロイメントのトラブルシューティングにも使用できます。

Tang サーバーにバインドされた暗号化クライアント

- Clevis 暗号化クライアントが Tang サーバーにバインドサれることを確認するには、**clevis encrypt tang** サブコマンドを使用します。

```
$ clevis encrypt tang '{"url":"http://tang.srv:port"}' < input-plain.txt > secret.jwe
```

```
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
```

この例の URL (**http://tang.srv:port**) を、 **tang** がインストールされているサーバーの URL に変更します。**secret.jwe** 出力ファイルには、JWE 形式で暗号化した暗号文が含まれます。この暗号文は **input-plain.txt** 入力ファイルから読み込まれます。

また、設定に SSH アクセスなしで Tang サーバーとの非対話型の通信が必要な場合は、アドバタイズメントをダウンロードしてファイルに保存できます。

```
$ curl -sfg http://tang.srv:port/adv -o adv.jws
```

ファイルやメッセージの暗号化など、次のタスクには **adv.jws** ファイルのアドバタイズメントを使用します。

```
$ echo 'hello' | clevis encrypt tang '{"url":"http://tang.srv:port","adv":"adv.jws"}'
```

- データを複号するには、**clevis decrypt** コマンドを実行して、暗号文 (JWE) を提供します。

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

TPM2.0 を使用する暗号化クライアント

- TPM 2.0 チップを使用して暗号化するには、JSON 設定オブジェクト形式の引数のみが使用されている **clevis encrypt tpm2** サブコマンドを使用します。

```
$ clevis encrypt tpm2 '{}' < input-plain.txt > secret.jwe
```

別の階層、ハッシュ、および鍵アルゴリズムを選択するには、以下のように、設定プロパティーを指定します。

```
$ clevis encrypt tpm2 '{"hash":"sha256","key":"rsa"}' < input-plain.txt > secret.jwe
```

- データを復号するには、JSON Web Encryption (JWE) 形式の暗号文を提供します。

```
$ clevis decrypt < secret.jwe > output-plain.txt
```

ピンは、PCR (Platform Configuration Registers) 状態へのデータのシーリングにも対応します。このように、PCP ハッシュ値が、シーリング時に使用したポリシーと一致する場合にのみ、データのシーリングを解除できます。

たとえば、SHA-256 バンクに対して、インデックス 0 および 7 の PCR にデータをシールするには、以下を行います。

```
$ clevis encrypt tpm2 '{"pcr_bank":"sha256","pcr_ids":"0,7"}' < input-plain.txt > secret.jwe
```

警告

PCR のハッシュは書き換えることができ、暗号化されたボリュームのロックを解除することはできなくなりました。このため、PCR の値が変更された場合でも、暗号化されたボリュームのロックを手動で解除できる強力なパスフレーズを追加します。

shim-x64 パッケージのアップグレード後にシステムが暗号化されたボリュームのロックを自動的に解除できない場合は、KCS の記事Clevis TPM2 no longer decrypts LUKS devices after a restartの手順に従ってください。

関連情報

- **clevis-encrypt-tang(1)**、**clevis-luks-unlockers(7)**、**clevis(1)**、および **clevis-encrypt-tpm2(1)** の man ページ
- 以下のように引数指定せずに **clevis**、**clevis decrypt** および **clevis encrypt tang** コマンドを入力したときに表示される組み込み CLI。

```
$ clevis encrypt tang
Usage: clevis encrypt tang CONFIG < PLAINTEXT > JWE
...
```

16.14.7. LUKS で暗号化したボリュームの手動登録の設定

以下の手順に従って、NBDE を使用して LUKS で暗号化されたボリュームのロック解除を設定します。

前提条件

- Tang サーバーが実行されていて、使用できるようにしてある。

手順

1. LUKS で暗号化した既存のボリュームを自動的にアンロックするには、サブパッケージの **clevis-luks** をインストールします。

```
# yum install clevis-luks
```

2. PBD 用 LUKS 暗号化ボリュームを特定します。次の例では、ブロックデバイスは **/dev/sda2** と呼ばれています。

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. **clevis luks bind** コマンドを使用して、ボリュームを Tang サーバーにバインドします。

```
# clevis luks bind -d /dev/sda2 tang '{"url":"http://tang.srv"}'
The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y
You are about to initialize a LUKS device for metadata storage.
Attempting to initialize it may result in data loss if data was
already written into the LUKS header gap in a different format.
A backup is advised before initialization is performed.

Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

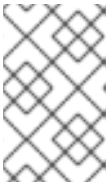
このコマンドは、以下の 4 つの手順を実行します。

a. LUKS マスター鍵と同じエントロピーを使用して、新しい鍵を作成します。

b. Clevis で新しい鍵を暗号化します。

c. LUKS2 ヘッダートークンに Clevis JWE オブジェクトを保存するか、デフォルト以外の LUKS1 ヘッダーが使用されている場合は LUKSMeta を使用します。

d. LUKS を使用する新しい鍵を有効にします。



注記

バインド手順では、空き LUKS パスワードスロットが少なくとも 1 つあることが前提となっています。そのスロットの 1 つを **clevis luks bind** コマンドが使用します。

ボリュームは、現在、既存のパスワードと Clevis ポリシーを使用してロックを解除できます。

4. システムの起動プロセスの初期段階でディスクバインディングを処理するようにするには、インストール済みのシステムで **dracut** ツールを使用します。

```
# yum install clevis-dracut
```

RHEL 8 では、Clevis はホスト固有の設定オプションを指定せずに汎用 **initrd** (initial ramdisk) を生成し、カーネルコマンドラインに **rd.neednet=1** などのパラメーターを自動的に追加しません。初期の起動時にネットワークを必要とする Tang ピンを使用する場合は、**--hostonly-cmdline** 引数を使用し、**dracut** が Tang バインディングを検出すると **rd.neednet=1** を追加します。

```
# dracut -fv --regenerate-all --hostonly-cmdline
```

または、**/etc/dracut.conf.d/** に .conf ファイルを作成し、以下のように **hostonly_cmdline=yes** オプションを追加します。

```
# echo "hostonly_cmdline=yes" > /etc/dracut.conf.d/clevis.conf
```

注記

Clevis がインストールされているシステムで **grubby** ツールを使用して、システム起動時の早い段階で Tang ピンのネットワークを利用できるようにすることができます。

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

次に、**--hostonly-cmdline** なしで **dracut** を使用できます。

```
# dracut -fv --regenerate-all
```

検証

1. Clevis JWE オブジェクトが LUKS ヘッダーに適切に置かれていることを確認するには、**clevis luks list** コマンドを使用します。

```
# clevis luks list -d /dev/sda2
1: tang '{"url":"http://tang.srv:port"}'
```

重要

(DHCP を使用しない) 静的な IP 設定を持つクライアントに NBDE を使用するには、以下のように、手動でネットワーク設定を **dracut** ツールに渡します。

```
# dracut -fv --regenerate-all --kernel-cmdline
"ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none"
```

もしくは、静的ネットワーク情報を使用して **/etc/dracut.conf.d/** ディレクトリーに .conf ファイルを作成します。以下に例を示します。

```
# cat /etc/dracut.conf.d/static_ip.conf
kernel_cmdline="ip=192.0.2.10::192.0.2.1:255.255.255.0::ens3:none"
```

初期 RAM ディスクイメージを再生成します。

```
# dracut -fv --regenerate-all
```

関連情報

- **clevis-luks-bind(1)** および **dracut.cmdline(7)** の man ページ。
- RHEL Network boot options

16.14.8. TPM2.0 ポリシーを使用した LUKS で暗号化したボリュームの手動登録の設定

次の手順を使用して、Trusted Platform Module 2.0 (TPM 2.0) ポリシーを使用して LUKS 暗号化ボリュームのロック解除を設定します。

前提条件

- アクセス可能な TPM2.0 互換デバイス。
- システムが 64 ビット Intel アーキテクチャー、または 64 ビット AMD アーキテクチャーである。

手順

1. LUKS で暗号化した既存のボリュームを自動的にアンロックするには、サブパッケージの **clevis-luks** をインストールします。

```
# yum install clevis-luks
```

2. PBD 用 LUKS 暗号化ボリュームを特定します。次の例では、ブロックデバイスは **/dev/sda2** と呼ばれています。

```
# lsblk
NAME                                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                             8:0    0    12G  0 disk
├─sda1                                          8:1    0     1G  0 part  /boot
└─sda2                                          8:2    0    11G  0 part
  └─luks-40e20552-2ade-4954-9d56-565aa7994fb6 253:0    0    11G  0 crypt
    ├─rhel-root                               253:0    0   9.8G  0 lvm   /
    └─rhel-swap                               253:1    0   1.2G  0 lvm   [SWAP]
```

3. **clevis luks bind** コマンドを使用して、ボリュームを TPM 2.0 デバイスにバインドします。以下に例を示します。

```
# clevis luks bind -d /dev/sda2 tpm2 '{"hash":"sha256","key":"rsa"}'
...
Do you wish to initialize /dev/sda2? [yn] y
Enter existing LUKS password:
```

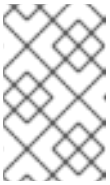
このコマンドは、以下の 4 つの手順を実行します。

a. LUKS マスター鍵と同じエントロピーを使用して、新しい鍵を作成します。

b. Clevis で新しい鍵を暗号化します。

c. LUKS2 ヘッダートークンに Clevis JWE オブジェクトを保存するか、デフォルト以外の LUKS1 ヘッダーが使用されている場合は LUKSMeta を使用します。

d. LUKS を使用する新しい鍵を有効にします。



注記

バインド手順では、空き LUKS パスワードスロットが少なくとも 1 つあることが前提となっています。そのスロットの 1 つを **clevis luks bind** コマンドが使用します。

あるいは、特定の Platform Configuration Registers (PCR) の状態にデータをシールする場合は、**clevis luks bind** コマンドに **pcr_bank** と **pcr_ids** 値を追加します。以下に例を示します。

```
# clevis luks bind -d /dev/sda2 tpm2
'{"hash":"sha256","key":"rsa","pcr_bank":"sha256","pcr_ids":"0,1"}'
```

警告

PCR ハッシュ値がシール時に使用されるポリシーと一致し、ハッシュを書き換えることができる場合にのみ、データをアンシールできるため、PCR の値が変更された場合、暗号化されたボリュームのロックを手動で解除できる強力なパスフレーズを追加します。

shim-x64 パッケージのアップグレード後にシステムが暗号化されたボリュームのロックを自動的に解除できない場合は、KCS の記事Clevis TPM2 no longer decrypts LUKS devices after a restartの手順に従ってください。

4. ボリュームは、現在、既存のパスワードと Clevis ポリシーを使用してロックを解除できます。
5. システムの起動プロセスの初期段階でディスクバインディングを処理するようにするには、インストール済みのシステムで **dracut** ツールを使用します。

```
# yum install clevis-dracut
# dracut -fv --regenerate-all
```

検証

1. Clevis JWE オブジェクトが LUKS ヘッダーに適切に置かれていることを確認するには、**clevis luks list** コマンドを使用します。

```
# clevis luks list -d /dev/sda2
1: tpm2 '{"hash":"sha256","key":"rsa"}'
```

関連情報

- **clevis-luks-bind(1)**、**clevis-encrypt-tpm2(1)**、および **dracut.cmdline(7)** の man ページ

16.14.9. LUKS で暗号化したボリュームからの Clevis ピンの手動削除

clevis luks bind コマンドで作成されたメタデータを手動で削除する場合や、Clevis が追加したパスフレーズを含む鍵スロットを一掃するには、以下の手順を行います。

重要

LUKS で暗号化したボリュームから Clevis ピンを削除する場合は、**clevis luks unbind** コマンドを使用することが推奨されます。**clevis luks unbind** を使用した削除手順は、1 回のステップで構成され、LUKS1 ボリュームおよび LUKS2 ボリュームの両方で機能します。以下のコマンド例は、バインディング手順で作成されたメタデータを削除し、**/dev/sda2** デバイスの鍵スロット **1** を削除します。

```
# clevis luks unbind -d /dev/sda2 -s 1
```

前提条件

- Clevis バインディングを使用した LUKS 暗号化ボリューム。

手順

1. **/dev/sda2** などのボリュームがどの LUKS バージョンであるかを確認し、Clevis にバインドされているスロットおよびトークンを特定します。

```
# cryptsetup luksDump /dev/sda2
LUKS header information
Version:        2
...
Keyslots:
  0: luks2
...
1: luks2
      Key:        512 bits
      Priority:   normal
      Cipher:     aes-xts-plain64
...
      Tokens:
        0: clevis
              Keyslot:  1
...
```

上記の例では、Clevis トークンは **0** で識別され、関連付けられた鍵スロットは **1** です。

2. LUKS2 暗号化の場合は、トークンを削除します。

```
# cryptsetup token remove --token-id 0 /dev/sda2
```

3. デバイスが LUKS1 で暗号化されていて、**Version:1** という文字列が **cryptsetup luksDump** コマンドの出力に含まれている場合は、**luksmeta wipe** コマンドでこの追加手順を実行します。

```
# luksmeta wipe -d /dev/sda2 -s 1
```

4. Clevis パスフレーズを含む鍵スロットを削除します。

```
# cryptsetup luksKillSlot /dev/sda2 1
```

関連情報

- **clevis-luks-unbind(1)**、**cryptsetup(8)**、および **luksmeta(8)** の man ページ

16.14.10. キックスタートを使用して、LUKS で暗号化したボリュームの自動登録の設定

この手順に従って、LUKS で暗号化されたボリュームの登録に Clevis を使用する自動インストールプロセスを設定します。

手順

1. 一時パスワードを使用して、LUKS 暗号化が有効になっているディスクを、**/boot** 以外のすべてのマウントポイントで分割するように、キックスタートに指示します。パスワードは、登録プロセスの手順に使用するための一時的なものです。

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --grow --encrypted --passphrase=temppass
```

   OSPP 準拠のシステムには、より複雑な設定が必要であることに注意してください。次に例を示します。

```
part /boot --fstype="xfs" --ondisk=vda --size=256
part / --fstype="xfs" --ondisk=vda --size=2048 --encrypted --passphrase=temppass
part /var --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /tmp --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /home --fstype="xfs" --ondisk=vda --size=2048 --grow --encrypted --
passphrase=temppass
part /var/log --fstype="xfs" --ondisk=vda --size=1024 --encrypted --passphrase=temppass
part /var/log/audit --fstype="xfs" --ondisk=vda --size=1024 --encrypted --
passphrase=temppass
```

2. 関連する Clevis パッケージを **%packages** セクションに追加して、インストールします。

```
%packages
clevis-dracut
clevis-luks
clevis-systemd
%end
```

3. オプションで、必要に応じて暗号化されたボリュームのロックを手動で解除できるようにするには、一時パスフレーズを削除する前に強力なパスフレーズを追加します。詳細については、How to add a passphrase, key, or keyfile to an existing LUKS device の記事を参照してください。

4. **clevis luks bind** を呼び出して、**%post** セクションのバインディングを実行します。その後、一時パスワードを削除します。

```
%post
clevis luks bind -y -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv"}' <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

設定が起動初期にネットワークを必要とする Tang ピンに依存している場合、または静的 IP 設定の NBDE クライアントを使用している場合は、Configuring manual enrollment of LUKS-encrypted volumesに従って **dracut** コマンドを変更する必要があります。

clevis luks bind コマンドの **-y** オプションは、RHEL 8.3 から使用できることに注意してください。RHEL 8.2 以前では、 **clevis luks bind** コマンドで **-y** を **-f** に置き換え、Tang サーバーからアドバタイズメントをダウンロードします。

```
%post
curl -sfg http://tang.srv/adv -o adv.jws
clevis luks bind -f -k - -d /dev/vda2 \
tang '{"url":"http://tang.srv","adv":"adv.jws"}' <<< "temppass"
cryptsetup luksRemoveKey /dev/vda2 <<< "temppass"
dracut -fv --regenerate-all
%end
```

警告

cryptsetup luksRemoveKey コマンドは、それを適用する LUKS2 デバイスがそれ以上に管理されるのを防ぎます。LUKS1 デバイスに対してのみ **dmsetup** コマンドを使用して、削除されたマスターキーを回復できます。

Tang サーバーの代わりに TPM 2.0 ポリシーを使用する場合は、同様の手順を使用できます。

関連情報

- **clevis(1)**、**clevis-luks-bind(1)**、**cryptsetup(8)**、および **dmsetup(8)** の man ページ
- キックスタートを使用して Red Hat Enterprise Linux 8 のインストール

16.14.11. LUKS で暗号化されたリムーバブルストレージデバイスの自動アンロックの設定

この手順に従って、LUKS 暗号化 USB ストレージデバイスの自動ロック解除プロセスを設定します。

手順

1. USB ドライブなど、LUKS で暗号化したリムーバブルストレージデバイスを自動的にアンロックするには、**clevis-udisks2** パッケージをインストールします。

```
# yum install clevis-udisks2
```

2. システムを再起動し、**LUKS で暗号化したボリュームの手動登録の設定** に従って、**clevis luks bind** コマンドを使用したバインディング手順を実行します。以下に例を示します。

```
# clevis luks bind -d /dev/sdb1 tang '{"url":"http://tang.srv"}'
```

3. LUKS で暗号化したリムーバブルデバイスは、GNOME デスクトップセッションで自動的にアンロックできるようになりました。Clevis ポリシーにバインドするデバイスは、**clevis luks unlock** コマンドでアンロックできます。

```
# clevis luks unlock -d /dev/sdb1
```

Tang サーバーの代わりに TPM 2.0 ポリシーを使用する場合は、同様の手順を使用できます。

関連情報

- **clevis-luks-unlockers(7)** man ページ

16.14.12. 高可用性 NBDE システムのデプロイメント

Tang は、高可用性デプロイメントを構築する方法を 2 つ提供します。

クライアントの冗長性 (推奨)

クライアントは、複数の Tang サーバーにバインドする機能を使用して設定する必要があります。この設定では、各 Tang サーバーに独自の鍵があり、クライアントは、このサーバーのサブセットに接続することで復号できます。Clevis はすでに、**sss** プラグインを使用してこのワークフローに対応しています。Red Hat は、高可用性のデプロイメントにこの方法を推奨します。

鍵の共有

冗長性を確保するために、Tang のインスタンスは複数デプロイできます。2 つ目以降のインスタンスを設定するには、**tang** パッケージをインストールし、**SSH** 経由で **rsync** を使用してその鍵ディレクトリーを新規ホストにコピーします。鍵を共有すると鍵への不正アクセスのリスクが高まり、追加の自動化インフラストラクチャーが必要になるため、Red Hat はこの方法を推奨していません。

16.14.12.1. シャミアの秘密分散を使用した高可用性 NBDE

シャミアの秘密分散 (SSS) は、秘密を複数の固有のパーツに分割する暗号スキームです。秘密を再構築するには、いくつかのパーツが必要になります。数値はしきい値と呼ばれ、SSS はしきい値スキームとも呼ばれます。

Clevis は、SSS の実装を提供します。鍵を作成し、これをいくつかのパーツに分割します。各パーツは、SSS も再帰的に含む別のピンを使用して暗号化されます。また、しきい値 **t** も定義します。NBDE デプロイメントで少なくとも **t** の部分を復号すると、暗号化鍵が復元され、復号プロセスが成功します。Clevis がしきい値で指定されている数よりも小さい部分を検出すると、エラーメッセージが出力されます。

16.14.12.1.1. 例 1:2 台の Tang サーバーを使用した冗長性

次のコマンドは、2 台の Tang サーバーのうち少なくとも 1 台が使用可能な場合に、LUKS で暗号化されたデバイスを復号します。

```
# clevis luks bind -d /dev/sda1 sss '{"t":1,"pins":{"tang":[{"url":"http://tang1.srv"},
{"url":"http://tang2.srv"}]}}'
```

上記のコマンドでは、以下の設定スキームを使用していました。

```
{
    "t":1,
    "pins":{
```

```
        "tang":[
           {
              "url":"http://tang1.srv"
           },
           {
              "url":"http://tang2.srv"
           }
        ]
     }
}
```

この設定では、リストに記載されている 2 台の **tang** サーバーのうち少なくとも 1 つが利用可能であれば、SSS しきい値 **t** が **1** に設定され、**clevis luks bind** コマンドが秘密を正常に再構築します。

16.14.12.1.2. 例 2:Tang サーバーと TPM デバイスで共有している秘密

次のコマンドは、**tang** サーバーと **tpm2** デバイスの両方が利用可能な場合に、LUKS で暗号化したデバイスを正常に復号します。

```
# clevis luks bind -d /dev/sda1 sss '{"t":2,"pins":{"tang":[{"url":"http://tang1.srv"}], "tpm2":
{"pcr_ids":"0,7"}}}'
```

SSS しきい値 't' が '2' に設定されている設定スキームは以下のようになります。

```
{
   "t":2,
   "pins":{
      "tang":[
         {
            "url":"http://tang1.srv"
         }
      ],
      "tpm2":{
         "pcr_ids":"0,7"
      }
   }
}
```

関連情報

- **tang(8)** (**High Availability** セクション)、**clevis(1)** (**Shamir's Secret Sharing** セクション)、および **clevis-encrypt-sss(1)** の man ページ

16.14.13. NBDE ネットワークで仮想マシンのデプロイメント

clevis luks bind コマンドは、LUKS マスター鍵を変更しません。これは、仮想マシンまたはクラウド環境で使用する、LUKS で暗号化したイメージを作成する場合に、このイメージを実行するすべてのインスタンスがマスター鍵を共有することを意味します。これにはセキュリティーの観点で大きな問題があるため、常に回避する必要があります。

これは、Clevis の制限ではなく、LUKS の設計原理です。シナリオでクラウド内のルートボリュームを暗号化する必要がある場合は、クラウド内の Red Hat Enterprise Linux の各インスタンスに対しても (通常はキックスタートを使用して) インストールプロセスを実行します。このイメージは、LUKS マスター鍵を共有しなければ共有できません。

仮想化環境で自動ロック解除をデプロイメントするには、**lorax** や **virt-install** などのシステムとキックスタートファイル (キックスタートを使用した LUKS 暗号化ボリュームの自動登録の設定参照) またはその他の自動プロビジョニングツールを使用して、各暗号化 VM に固有のマスターキーを確実に付与します。

関連情報

- **clevis-luks-bind(1)** man ページ

16.14.14. NBDE を使用してクラウド環境に自動的に登録可能な仮想マシンイメージの構築

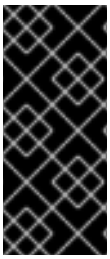
自動登録可能な暗号化イメージをクラウド環境にデプロイすると、特有の課題が発生する可能性があります。他の仮想化環境と同様に、LUKS マスター鍵を共有しないように、1 つのイメージから起動するインスタンス数を減らすことが推奨されます。

したがって、ベストプラクティスは、どのパブリックリポジトリーでも共有されず、限られたインスタンスのデプロイメントのベースを提供するように、イメージをカスタマイズすることです。作成するインスタンスの数は、デプロイメントのセキュリティーポリシーで定義する必要があります。また、LUKS マスター鍵の攻撃ベクトルに関連するリスク許容度に基づいて決定する必要があります。

LUKS に対応する自動デプロイメントを構築するには、Lorax、virt-install などのシステムとキックスタートファイルを一緒に使用し、イメージ構築プロセス中にマスター鍵の一意性を確保する必要があります。

クラウド環境では、ここで検討する 2 つの Tang サーバーデプロイメントオプションが利用できます。まず、クラウド環境そのものに Tang サーバーをデプロイできます。もしくは、2 つのインフラストラクチャー間で VPN リンクを使用した独立したインフラストラクチャーで、クラウドの外に Tang サーバーをデプロイできます。

クラウドに Tang をネイティブにデプロイすると、簡単にデプロイできます。ただし、別のシステムの暗号文のデータ永続化層でインフラストラクチャーを共有します。Tang サーバーの秘密鍵および Clevis メタデータは、同じ物理ディスクに保存できる場合があります。この物理ディスクでは、暗号文データへのいかなる不正アクセスが可能になります。



重要

このため、Red Hat は、データを保存する場所と、Tang が実行しているシステムを、物理的に分離させることを強く推奨します。クラウドと Tang サーバーを分離することで、Tang サーバーの秘密鍵が、Clevis メタデータと誤って結合することがないようにします。さらに、これにより、クラウドインフラストラクチャーが危険にさらされている場合に、Tang サーバーのローカル制御を提供します。

16.14.15. コンテナーとしての Tang のデプロイ

tang コンテナーイメージは、OpenShift Container Platform (OCP) クラスターまたは別の仮想マシンで実行する Clevis クライアントの Tang-server 復号化機能を提供します。

前提条件

- **podman** パッケージとその依存関係がシステムにインストールされている。
- **podman login registry.redhat.io** コマンドを使用して **registry.redhat.io** コンテナーカタログにログインしている。詳細は、Red Hat コンテナーレジストリーの認証 を参照してください。

- Clevis クライアントは、Tang サーバーを使用して、自動的にアンロックする LUKS で暗号化したボリュームを含むシステムにインストールされている。

手順

1. **registry.redhat.io** レジストリーから **tang** コンテナーイメージをプルします。

```
# podman pull registry.redhat.io/rhel8/tang
```

2. コンテナーを実行し、そのポートを指定して Tang 鍵へのパスを指定します。上記の例では、**tang** コンテナーを実行し、ポート **7500** を指定し、**/var/db/tang** ディレクトリーの Tang 鍵へのパスを示します。

```
# podman run -d -p 7500:7500 -v tang-keys:/var/db/tang --name tang
registry.redhat.io/rhel8/tang
```

Tang はデフォルトでポート 80 を使用しますが、Apache HTTP サーバーなどの他のサービスと共存する可能性があることに注意してください。

3. (必要に応じて) セキュリティーを強化する場合は、Tang 鍵を定期的にローテーションします。**tangd-rotate-keys** スクリプトを使用できます。以下に例を示します。

```
# podman run --rm -v tang-keys:/var/db/tang registry.redhat.io/rhel8/tang tangd-rotate-keys -
v -d /var/db/tang
Rotated key 'rZAMKAseaXBe0rcKXL1hCCIq-DY.jwk' -> .'rZAMKAseaXBe0rcKXL1hCCIq-
DY.jwk'
Rotated key 'x1AIpc6WmnCU-CabD8_4q18vDuw.jwk' -> .'x1AIpc6WmnCU-
CabD8_4q18vDuw.jwk'
Created new key GrMMX_WfdqomIU_4RyjpcdlXb0E.jwk
Created new key _dTTfn17sZZqVAp80u3ygFDHtjk.jwk
Keys rotated successfully.
```

検証

- Tang サーバーが存在しているために自動アンロック用に LUKS で暗号化したボリュームが含まれているシステムで、Clevis クライアントが Tang を使用してプレーンテキストのメッセージを暗号化および復号化できることを確認します。

```
# echo test | clevis encrypt tang '{"url":"http://localhost:7500"}' | clevis decrypt
The advertisement contains the following signing keys:

x1AIpc6WmnCU-CabD8_4q18vDuw

Do you wish to trust these keys? [ynYN] y
test
```

上記のコマンド例は、**localhost** URL で Tang サーバーが利用できる場合にその出力の最後に **テスト** 文字列を示し、ポート **7500** 経由で通信します。

関連情報

- **podman(1)**、**clevis(1)** および **tang(8)** の man ページ

- Clevis および Tang を使用して LUKS で暗号化したボリュームの自動ロック解除の詳細は、ボリシーベースの複号を使用して暗号化ボリュームの自動アンロックの設定 を参照してください。

16.14.16. nbde_client および nbde_server システムロールの概要 (Clevis および Tang)

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集合です。

RHEL 8.3 では、Clevis および Tang を使用した PBD (Policy-Based Decryption) ソリューションの自動デプロイメント用 Ansible ロールが導入されました。**rhel-system-roles** パッケージには、これらのシステムロール、関連する例、リファレンスドキュメントが含まれます。

nbde_client システムロールにより、複数の Clevis クライアントを自動的にデプロイできます。**nbde_client** ロールは、Tang バインディングのみをサポートしており、現時点では TPM2 バインディングには使用できない点に留意してください。

nbde_client ロールには、LUKS を使用して暗号化済みのボリュームが必要です。このロールは、LUKS 暗号化ボリュームの 1 つ以上の Network-Bound (NBDE) サーバー (Tang サーバー) へのバインドに対応します。パスフレーズを使用して既存のボリュームの暗号化を保持するか、削除できます。パスフレーズを削除したら、NBDE だけを使用してボリュームのロックを解除できます。これは、システムのプロビジョニング後に削除する必要がある一時鍵またはパスワードを使用して、ボリュームが最初に暗号化されている場合に役立ちます。

パスフレーズと鍵ファイルの両方を指定する場合には、ロールは最初に指定した内容を使用します。有効なバインディングが見つからない場合は、既存のバインディングからパスフレーズの取得を試みます。

PBD では、デバイスをスロットにマッピングするものとしてバインディングを定義します。つまり、同じデバイスに複数のバインディングを指定できます。デフォルトのスロットは 1 です。

nbde_client ロールでは、**state** 変数も指定できます。新しいバインディングを作成するか、既存のバインディングを更新する場合は、**present** を使用します。**clevis luks bind** とは異なり、**state: present** を使用してデバイススロットにある既存のバインディングを上書きすることもできます。**absent** に設定すると、指定したバインディングが削除されます。

nbde_client システムロールを使用すると、自動ディスク暗号化ソリューションの一部として、Tang サーバーをデプロイして管理できます。このロールは以下の機能をサポートします。

- Tang 鍵のローテーション
- Tang 鍵のデプロイおよびバックアップ

関連情報

- NBDE (Network-Bound Disk Encryption) 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/nbde_client/** と **/usr/share/doc/rhel-system-roles/nbde_server/** ディレクトリーの **README.md** と **README.html** ファイルを参照してください。
- たとえば、system-roles Playbook の場合は、**rhel-system-roles** パッケージをインストールし、**/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** ディレクトリーを参照してください。
- RHEL システムロールの詳細は、Introduction to RHEL System Roles を参照してください。

16.14.17. 複数の Tang サーバーをセットアップするための nbde_server システムロールの使用

以下の手順に従って、Tang サーバー設定を含む Ansible Playbook を準備および適用します。

前提条件

- 1 つ以上の **マネージドノード** (**nbde_server** システムロールで設定するシステム) へのアクセスおよびパーミッション。
- コントロールノード (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
  コントロールノードでは、
  - **ansible-core** パッケージおよび **rhel-system-roles** パッケージがインストールされている。

重要

RHEL 8.0-8.5 では、別の Ansible リポジトリーへのアクセス権を指定されており、Ansible をベースにする自動化用の Ansible Engine 2.9 が含まれています。Ansible Engine には、**ansible**、**ansible-playbook** などのコマンドラインユーティリティー、**docker** や **podman** などのコネクター、プラグインとモジュールが多く含まれています。Ansible Engine を入手してインストールする方法については、ナレッジベースの How to download and install Red Hat Ansible Engine を参照してください。

RHEL 8.6 および 9.0 では、Ansible Core (**ansible-core** パッケージとして提供) が導入されました。これには、Ansible コマンドラインユーティリティー、コマンド、およびビルトイン Ansible プラグインのセットが含まれています。RHEL は、AppStream リポジトリーを介してこのパッケージを提供し、サポート範囲は限定的です。詳細については、ナレッジベースの Scope of support for the Ansible Core package included in the RHEL 9 and RHEL 8.6 and later AppStream repositories を参照してください。

- マネージドノードが記載されているインベントリーファイルがある。

手順

1. Tang サーバーの設定が含まれる Playbook を準備します。ゼロから開始するか、**/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** ディレクトリーにある Playbook のいずれかのサンプルを使用することができます。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2. 選択したテキストエディターで Playbook を編集します。以下に例を示します。

```
# vi my-tang-playbook.yml
```

3. 必要なパラメーターを追加します。以下の Playbook の例では、Tang サーバーのデプロイと鍵のローテーションを確実に実行します。

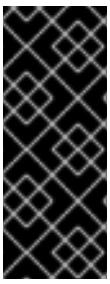
```
---
- hosts: all
```

```
vars:
  nbde_server_rotate_keys: yes

roles:
  - rhel-system-roles.nbde_server
```

4. 終了した Playbook を適用します。

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

ここで * **inventory-file** はインベントリーファイル、* **logging-playbook.yml** は Playbook も置き換えます。



重要

Clevis がインストールされているシステムで **grubby** ツールを使用して、システム起動時の早い段階で Tang ピンのネットワークを利用できるようにするには、次のコマンドを実行します。

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

関連情報

- 詳細は、**rhel-system-roles** パッケージをインストールして、**/usr/share/doc/rhel-system-roles/nbde_server/** ディレクトリーおよび **usr/share/ansible/roles/rhel-system-roles.nbde_server/** ディレクトリーを参照してください。

16.14.18. 複数の Clevis クライアントの設定に nbde_client システムロールを使用

手順に従って、Clevis クライアント設定を含む Ansible Playbook を準備および適用します。

注記

nbde_client システムロールは、Tang バインディングのみをサポートします。これは、現時点では TPM2 バインディングに使用できないことを意味します。

前提条件

- 1 つ以上の **マネージドノード** (**nbde_client** システムロールで設定するシステム) へのアクセスおよびパーミッション。
- **コントロールノード** (このシステムから Red Hat Ansible Core は他のシステムを設定) へのアクセスおよびパーミッション。
- Ansible Core パッケージがコントロールマシンにインストールされている。
- **rhel-system-roles** パッケージが、Playbook を実行するシステムにインストールされている。

手順

1. Clevis クライアントの設定が含まれる Playbook を準備します。ゼロから開始するか、**/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/** ディレクトリーにある Playbook のいずれかのサンプルを使用することができます。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2. 選択したテキストエディターで Playbook を編集します。以下に例を示します。

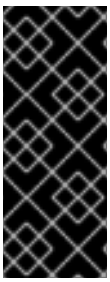
```
# vi my-clevis-playbook.yml
```

3. 必要なパラメーターを追加します。以下の Playbook の例では、2 つの Tang サーバーのうち少なくとも 1 台が利用可能な場合に、LUKS で暗号化した 2 つのボリュームを自動的にアンロックするように Clevis クライアントを設定します。

```
---
- hosts: all

  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com
      - device: /dev/rhel/swap
        encryption_key_src: /etc/luks/keyfile
        servers:
          - http://server1.example.com
          - http://server2.example.com

  roles:
    - rhel-system-roles.nbde_client
```

4. 終了した Playbook を適用します。

```
# ansible-playbook -i host1,host2,host3 my-clevis-playbook.yml
```



重要

Clevis がインストールされているシステムで **grubby** ツールを使用して、システム起動時の早い段階で Tang ピンのネットワークを利用できるようにするには、次のコマンドを実行します。

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

関連情報

- パラメーターの詳細と、NBDE Client システムロールに関する追加情報は、**rhel-system-roles** パッケージをインストールし、**/usr/share/doc/rhel-system-roles/nbde_client/** および **/usr/share/ansible/roles/rhel-system-roles.nbde_client/** ディレクトリーを参照してください。

第17章 SELINUX の使用

17.1. SELINUX の使用

Security Enhanced Linux (SELinux) は、新たにシステムセキュリティーの層を提供します。SELinux は、基本的に次の質問に答えます。**May <subject> do <action> to <object>?**。以下に例を示します。**Web サーバーが、ユーザーのホームディレクトリーにあるファイルにアクセスできる可能性がありますか？**

17.1.1. SELinux の概要

ユーザー、グループ、およびその他のアクセス権に基づいた標準のアクセスポリシーは Discretionary Access Control (DAC) として知られており、システム管理者が、包括的で詳細なセキュリティーポリシー (たとえば、特定のアプリケーションではログファイルの表示だけを許可し、その他のアプリケーションではログファイルに新しいデータを追加するのを許可するなど) を作成することはできません。

Security Enhanced Linux (SELinux) は Mandatory Access Control (MAC) を実装します。それぞれのプロセスおよびシステムリソースには、**SELinux コンテキスト** と呼ばれる特別なセキュリティーラベルがあります。SELinux コンテキストは **SELinux ラベル** として参照されることがありますが、システムレベルの詳細を抽象化し、エンティティーのセキュリティープロパティーに焦点を当てた識別子です。これにより、SELinux ポリシーでオブジェクトを参照する方法に一貫性を持たせ、他の識別方法に含まれる曖昧さがなくなりました。たとえば、バインドマウントを使用するシステムで、ファイルに、有効なパス名を複数設定できます。

SELinux ポリシーは、プロセスが、互いに、またはさまざまなシステムリソースと相互作用する方法を定義する一連のルールにこのコンテキストを使用します。デフォルトでは、最初にルールが明示的にアクセスを許可し、その後ポリシーが任意の対話を許可します。

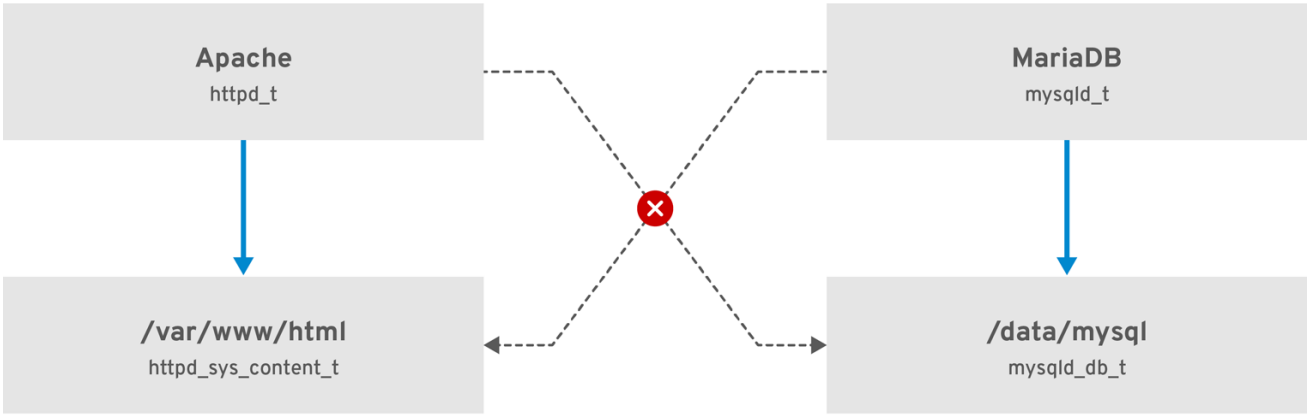
注記

SELinux ポリシールールが DAC ルールの後に確認されている点に注意してください。DAC ルールがアクセスを拒否すると、SELinux ポリシールールは使用されません。これは、従来の DAC ルールがそのアクセスを拒否すると、SELinux 拒否がログに記録されないということを示しています。

SELinux コンテキストには、複数のフィールド (ユーザー、ロール、タイプ、セキュリティーレベル) があります。プロセスとシステムリソースとの間で許可される相互作用を定義する最も一般的なポリシールールは、完全な SELinux コンテキストではなく、SELinux タイプを使用するため、SELinux ポリシーでは、SELinux のタイプ情報がおそらく最も重要です。SELinux のタイプの名前は、最後に **_t** が付きます。たとえば、Web サーバーのタイプ名は **httpd_t** です。**/var/www/html/** にあるファイルおよびディレクトリーのタイプコンテキストは、通常 **httpd_sys_content_t** です。**/tmp** および **/var/tmp/** にあるファイルおよびディレクトリーのタイプコンテキストは、通常 **tmp_t** です。Web サーバーポートのタイプコンテキストは **http_port_t** です。

Apache (**httpd_t** として実行する Web サーバープロセス) を許可するポリシールールがあります。このルールでは、通常 **/var/www/html/** にあるコンテキストを持つファイルおよびディレクトリーと、その他の Web サーバーディレクトリー (**httpd_sys_content_t**) へのアクセスを許可します。通常、**/tmp** および **/var/tmp/** に含まれるファイルのポリシーには、許可ルールがないため、アクセスは許可されません。SELinux を使用すれば、Apache が危険にさらされ、悪意のあるスクリプトがアクセスを得た場合でも、**/tmp** ディレクトリーにアクセスすることはできなくなります。

図17.1 Apache と MariaDB を安全に実行する SELinux の例



このスキーマが示すように、SELinux は、**httpd_t** として実行している Apache プロセスが **/var/www/html/** ディレクトリーにアクセスするのは許可しますが、同じ Apache プロセスが **/data/mysql/** ディレクトリーにアクセスするのは拒否します。これは、**httpd_t** タイプコンテキストと **mysqld_db_t** タイプコンテキストに許可ルールがないのが原因です。一方、**mysqld_t** として実行する MariaDB プロセスは **/data/mysql/** ディレクトリーにアクセスできますが、SELinux により、**mysqld_t** タイプを持つプロセスが、**httpd_sys_content_t** とラベルが付いた **/var/www/html/** ディレクトリーにアクセスするのは拒否されます。

関連情報

- **apropos selinux** コマンドで表示される man ページの **selinux(8)**
- **selinux-policy-doc** パッケージをインストールしている場合は、**man -k _selinux** コマンドで表示された man ページ。
- The SELinux Coloring Book、SELinux の基本概念をよりよく理解するのに役立ちます。
- SELinux Wiki FAQ

17.1.2. SELinux を実行する利点

SELinux は、次のような利点を提供します。

- プロセスとファイルにはすべてラベルが付いています。SELinux ポリシーにより、プロセスがファイルと相互作用する方法と、プロセスが互いに相互作用する方法が定義されます。アクセスは、それを特別に許可する SELinux ポリシールールが存在する場合に限り許可されます。
- アクセス制御がより詳細に設定できるようになりました。SELinux のアクセスは、ユーザーの裁量と、Linux のユーザー ID およびグループ ID に基づいて制御される従来の UNIX アクセス権だけでなく、SELinux のユーザー、ロール、タイプなど (必要に応じてセキュリティーレベルも) の、入手可能なすべての情報に基づいて決定されます。
- SELinux ポリシーは管理者が定義し、システム全体に適用されます。
- 権限昇格攻撃に対する軽減策が向上しました。プロセスはドメインで実行するため、互いに分離しています。SELinux ポリシールールは、プロセスがどのようにファイルやその他のプロセスにアクセスするかを定義します。プロセスへのアクセスが不正に行われても、攻撃者は、そのプロセスの通常の機能と、そのプロセスがアクセスするように設定されているファイルにしかアクセスできません。たとえば、Apache HTTP Server へのアクセスが不正に行われても、そ

のアクセスを許可する特別な SELinux ポリシールールが追加されたり、設定された場合を除き、ユーザーのホームディレクトリーにあるファイルを読み込むプロセスを攻撃者が利用することはできません。

- SELinux は、データの機密性と完全性、並びに信頼されていない入力からの保護プロセスを強化するのに使用できます。

ただし、SELinux は以下の機能とは異なります。

- ウイルス対策ソフトウェア
- パスワード、ファイアウォールなどのセキュリティーシステムの代替
- 一体型のセキュリティーソリューション

SELinux は、既存のセキュリティーソリューションを強化するために作られており、代わりに使用されるものではありません。SELinux を実行している場合でも、ソフトウェアを最新の状態にする、推測が困難なパスワードを使用する、ファイアウォールを使用するなど、優れたセキュリティー対策を続けることが重要です。

17.1.3. SELinux の例

以下の例は、SELinux がどのようにセキュリティーを向上するかを説明します。

- デフォルトのアクションは拒否です。アクセスを許可する SELinux のポリシールール (ファイルを開くプロセスなど) が存在しない場合は、アクセスが拒否されます。
- SELinux は、Linux ユーザーに制限をかけられます。SELinux ポリシーには、制限がかけられた SELinux ユーザーが多数含まれます。Linux ユーザーを、制限がかけられた SELinux ユーザーにマッピングして、SELinux ユーザーに適用されているセキュリティールールおよびメカニズムを利用できます。たとえば、Linux ユーザーを SELinux **user_u** ユーザーにマッピングすると、その Linux ユーザーは、**sudo** や **su** などの setuid (set user ID) アプリケーションを実行できなくなります。
- プロセスとデータの分離が向上します。SELinux **ドメイン** の概念により、特定のファイルやディレクトリーにアクセスできるプロセスの定義が可能になります。たとえば、SELinux を実行している場合に、(許可が設定されていない限り) 攻撃者は Samba サーバーを危険にさらすことはできず、その Samba サーバーを攻撃ベクトルとして使用して、その他のプロセス (MariaDB など) が使用するファイルの読み書きを行うことはできません。
- SELinux は、設定ミスによるダメージを軽減します。ドメインネームシステム (DNS) サーバーは、多くの場合、ゾーン転送で相互に情報をレプリケートします。攻撃者は、ゾーン転送を使用して、虚偽の情報で DNS サーバーを更新できます。RHEL で Berkeley Internet Name Domain (BIND) を DNS サーバーとして実行している場合、管理者がゾーン転送を実行できるサーバーを制限するのを忘れた場合でも、デフォルトの SELinux ポリシーによってゾーンファイルの更新が妨げられます。[2] BIND **named** のデーモン自体、およびその他のプロセスによって、ゾーン転送を使用します。
- SELinux を使用しない場合、攻撃者は脆弱性を悪用して Apache Web サーバーのパストラバーサルを行い、**../** などの特別な要素を使用してファイルシステムに保存されているファイルやディレクトリーにアクセスできます。SELinux を enforcing モードで実行しているサーバーに対して攻撃者が攻撃を試みた場合、SELinux は、**httpd** プロセスがアクセスしてはならないファイルへのアクセスを拒否します。SELinux はこのタイプの攻撃を完全にブロックすることはできませんが、効果的に緩和します。
- Enforcing モードの SELinux は、非 SMAP プラットフォームでのカーネル NULL ポインター逆

参照 Operator の悪用を正常に防止します (CVE-2019-9213)。攻撃者は、null ページのマッピングをチェックしない **mmap** 関数の脆弱性を利用して、このページに任意のコードを配置します。

- **deny_ptrace** SELinux ブール値と Enforcing モードの SELinux は、システムを **PTRACE_TRACEME** 脆弱性 (CVE-2019-13272) から保護します。このような設定により、攻撃者が **root** 権限を取得できるシナリオが防止されます。

- **nfs_export_all_rw** および **nfs_export_all_ro** SELinux ブール値は、**/home** ディレクトリーの偶発的な共有など、ネットワークファイルシステム (NFS) の設定ミスを防ぐための使いやすいツールを提供します。

関連情報

- SELinux as a security pillar of an operating system - Real-world benefits and examples ナレッジベースの記事

17.1.4. SELinux のアーキテクチャーおよびパッケージ

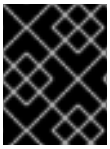
SELinux は、Linux カーネルに組み込まれる Linux セキュリティーモジュール (LSM) です。カーネルの SELinux サブシステムは、管理者が制御し、システムの起動時に読み込まれるセキュリティーポリシーにより動作します。システムにおけるセキュリティー関連の、カーネルレベルのアクセス操作はすべて SELinux により傍受され、読み込んだセキュリティーポリシーのコンテキストに従って検討されます。読み込んだポリシーが操作を許可すると、その操作は継続します。許可しないと、その操作はブロックされ、プロセスがエラーを受け取ります。

アクセスの許可、拒否などの SELinux の結果はキャッシュされます。このキャッシュは、アクセスベクトルキャッシュ (AVC) として知られています。このように結果がキャッシュされると、確認が必要な量が減るため、SELinux ポリシーのパフォーマンスが向上します。DAC ルールがアクセスを拒否した場合は、SELinux ポリシールールが適用されないことに注意してください。未加工の監査メッセージのログは、行頭に **type=AVC** 文字列が追加されて、**/var/log/audit/audit.log** に記録されます。

RHEL 8 では、システムサービスは **systemd** デーモンによって制御されています。**systemd** はすべてのサービスを起動および停止し、ユーザーとプロセスは **systemctl** ユーティリティーを使用して **systemd** と通信します。**systemd** デーモンは、SELinux ポリシーを調べ、呼び出しているプロセスのラベルと、呼び出し元が管理するユニットファイルのラベルを確認してから、呼び出し元のアクセスを許可するかどうかを SELinux に確認します。このアプローチにより、システムサービスの開始や停止などの、重要なシステム機能へのアクセス制御が強化されます。

また、**systemd** デーモンは SELinux Access Manager としても起動します。 **systemctl** を実行しているプロセス、または **systemd** に **D-Bus** メッセージを送信したプロセスのラベルを取得します。次に、デーモンは、プロセスが設定するユニットファイルのラベルを探します。最後に、SELinux ポリシーでプロセスラベルとユニットファイルラベルとの間で特定のアクセスが許可されている場合、**systemd** はカーネルから情報を取得できます。これは、特定のサービスに対して、**systemd** と相互作用を必要とする、危険にさらされたアプリケーションを SELinux が制限できることを意味します。ポリシー作成者は、このような粒度の細かい制御を使用して、管理者を制限することもできます。

プロセスが別のプロセスに **D-Bus** メッセージを送信しているときに、この 2 つのプロセスの **D-Bus** 通信が SELinux ポリシーで許可されていない場合は、システムが **USER_AVC** 拒否メッセージを出力し、D-Bus 通信がタイムアウトになります。2 つのプロセス間の D-Bus 通信は双方向に機能することに注意してください。



重要

SELinux ラベリングが誤っているために問題が発生するのを回避するには、**systemctl start** コマンドを使用してサービスを開始するようにしてください。

RHEL 8 では、SELinux を使用するために以下のパッケージが提供されています。

- ポリシー - **selinux-policy-targeted**、**selinux-policy-mls**
- ツール - **policycoreutils**、**policycoreutils-gui**、**libselinux-utils**、**policycoreutils-python-utils**、**setools-console**, **checkpolicy**

17.1.5. SELinux のステータスおよびモード

SELinux は、3 つのモード (Enforcing、Permissive、または Disabled) のいずれかで実行できます。

- Enforcing モードは、デフォルトのモードで、推奨される動作モードです。SELinux は、Enforcing モードでは正常に動作し、読み込んだセキュリティーポリシーをシステム全体に強制します。
- Permissive モードでは、システムは、読み込んだセキュリティーポリシーを SELinux が強制しているように振る舞い、オブジェクトのラベリングや、アクセスを拒否したエントリーをログに出力しますが、実際に操作を拒否しているわけではありません。Permissive モードは、実稼働システムで使用することは推奨されませんが、SELinux ポリシーの開発やデバッグには役に立ちます。
- Disabled モードを使用することは推奨されません。システムは、SELinux ポリシーの強制を回避するだけでなく、ファイルなどの任意の永続オブジェクトにラベルを付けなくなり、将来的に SELinux を有効にすることが難しくなります。

Enforcing モードと Permissive モードとの間を切り替えるには **setenforce** ユーティリティーを使用してください。**setenforce** で行った変更は、システムを再起動すると元に戻ります。Enforcing モードに変更するには、Linux の root ユーザーで、**setenforce 1** コマンドを実行します。Permissive モードに変更するには、**setenforce 0** コマンドを実行します。**getenforce** ユーティリティーを使用して、現在の SELinux モードを表示します。

```
# getenforce
Enforcing
```

```
# setenforce 0
# getenforce
Permissive
```

```
# setenforce 1
# getenforce
Enforcing
```

Red Hat Enterprise Linux では、システムを Enforcing モードで実行している場合に、個々のドメインを Permissive モードに設定できます。たとえば、**httpd_t** ドメインを Permissive に設定するには、以下のコマンドを実行します。

```
# semanage permissive -a httpd_t
```

Permissive ドメインは、システムのセキュリティーを侵害できる強力なツールです。Red Hat は、特定のシナリオのデバッグ時など、Permissive ドメインを使用する場合は注意することを推奨します。

17.2. SELINUX のステータスおよびモードの変更

SELinux が有効になっている場合は、Enforcing モードまたは Permissive モードのいずれかで実行できます。以下のセクションでは、これらのモードに永続的に変更する方法を説明します。

17.2.1. SELinux のステータスおよびモードの永続的変更

SELinux のステータスおよびモード で説明されているように、SELinux は有効または無効にできます。有効にした場合の SELinux のモードには、Enforcing および Permissive の 2 つがあります。

getenforce コマンド、または **sestatus** コマンドを使用して、SELinux が実行しているモードを確認できます。**getenforce** コマンドは、**Enforcing**、**Permissive**、または **Disabled** を返します。

sestatus コマンドは SELinux のステータスと、使用されている SELinux ポリシーを返します。

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      31
```

警告

Permissive モードで SELinux を実行すると、ユーザーやプロセスにより、さまざまなファイルシステムオブジェクトのラベルが間違って設定される可能性があります。SELinux が無効になっている間に作成されたファイルシステムのオブジェクトには、ラベルが追加されません。ただし、SELinux では、ファイルシステムオブジェクトのラベルが正しいことが必要になるため、これにより Enforcing モードに変更したときに問題が発生します。

SELinux では、誤ったラベル付けやラベル付けされていないファイルが問題を引き起こすことを防ぐため、Disabled 状態から Permissive モードまたは Enforcing モードに変更すると、ファイルシステムのラベルが自動的に再設定されます。root で **fixfiles -F onboot** コマンドを使用して、**-F** オプションを含む **/.autorelabel** ファイルを作成し、次回のシステムの再起動時にファイルに再ラベル付けされるようにします。

再ラベル付けのためにシステムを再起動する前に、**enforcing=0** カーネルオプションを使用するなどして、システムが Permissive モードで起動することを確認します。これにより、**selinux-autorelabel** サービスを起動する前に、**systemd** が必要とするラベルのないファイルがシステムにある場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

17.2.2. Permissive モードへの変更

以下の手順を使用して、SELinux モードを永続的に Permissive に変更します。SELinux を Permissive

モードで実行していると、SELinux ポリシーは強制されません。システムは動作し続け、SELinux がオペレーションを拒否せず AVC メッセージをログに記録できるため、このログを使用して、トラブルシューティングやデバッグ、ならびに SELinux ポリシーの改善に使用できます。この場合、各 AVC は一度だけログに記録されます。

前提条件

- **selinux-policy-targeted** パッケージ、**libselinux-utils** パッケージ、および **policycoreutils** パッケージがインストールされている。
- **selinux=0** または **enforcing=0** カーネルパラメーターは使用されません。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=permissive** オプションを設定します。

```
# 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=permissive
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. システムを再起動します。

```
# reboot
```

検証

1. システムの再起動後に、**getenforce** コマンドが **Permissive** を返すことを確認します。

```
$ getenforce
Permissive
```

17.2.3. Enforcing モードへの変更

以下の手順に従って、SELinux を Enforcing モードに切り替えます。SELinux を Enforcing モードで実行している場合は、SELinux ポリシーが強制され、SELinux ポリシールールに基づいてアクセスが拒否されます。RHEL では、システムに SELinux を最初にインストールした時に、Enforcing モードがデフォルトで有効になります。

前提条件

- **selinux-policy-targeted** パッケージ、**libselinux-utils** パッケージ、および **policycoreutils** パッケージがインストールされている。

- **selinux=0** または **enforcing=0** カーネルパラメーターは使用されません。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=enforcing** オプションを設定します。

```
# 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
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 変更を保存して、システムを再起動します。

```
# reboot
```

次にシステムを起動する際に、SELinux はシステム内のファイルおよびディレクトリーのラベルを再設定し、SELinux が無効になっている間に作成したファイルおよびディレクトリーに SELinux コンテキストを追加します。

検証

1. システムの再起動後に、**getenforce** コマンドが **Enforcing** を返すことを確認します。

```
$ getenforce
Enforcing
```

注記

Enforcing モードに変更したあと、SELinux ポリシールールが間違っていたか、設定されていなかったため、SELinux が一部のアクションを拒否する場合があります。SELinux に拒否されるアクションを表示するには、root で以下のコマンドを実行します。

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts today
```

setroubleshoot-server パッケージがインストールされている場合は、次のコマンドも使用できます。

```
# grep "SELinux is preventing" /var/log/messages
```

SELinux が有効で、Audit デーモン (**auditd**) がシステムで実行していない場合は、**dmesg** コマンドの出力で SELinux メッセージを検索します。

```
# dmesg | grep -i -e type=1300 -e type=1400
```

詳細は Troubleshooting problems related to SELinux を参照してください。

17.2.4. 以前は無効にしていたシステムで SELinux を有効にする

SELinux が無効になっていたシステムで、SELinux を有効にする際に、システムが起動できない、プロセスが失敗するなどの問題を回避するには、この手順に従ってください。

警告

Permissive モードで SELinux を実行すると、ユーザーやプロセスにより、さまざまなファイルシステムオブジェクトのラベルが間違って設定される可能性があります。SELinux が無効になっている間に作成されたファイルシステムのオブジェクトには、ラベルが追加されません。ただし、SELinux では、ファイルシステムオブジェクトのラベルが正しいことが必要になるため、これにより Enforcing モードに変更したときに問題が発生します。

SELinux では、誤ったラベル付けやラベル付けされていないファイルが問題を引き起こすことを防ぐため、Disabled 状態から Permissive モードまたは Enforcing モードに変更すると、ファイルシステムのラベルが自動的に再設定されます。

再ラベル付けのためにシステムを再起動する前に、**enforcing=0** カーネルオプションを使用するなどして、システムが Permissive モードで起動することを確認します。これにより、**selinux-autorelabel** サービスを起動する前に、**systemd** が必要とするラベルのないファイルがシステムにある場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

手順

1. SELinux を Permissive モードで有効にします。詳細は Permissive モードへの変更 を参照してください。
2. システムを再起動します。

```
# reboot
```

3. SELinux 拒否メッセージを確認します。詳細は SELinux 拒否の特定 を参照してください。
4. 次の再起動時に、ファイルが再ラベル付けされていることを確認します。

```
# fixfiles -F onboot
```

これにより、**-F** オプションを含む **/.autorelabel** ファイルが作成されます。

警告

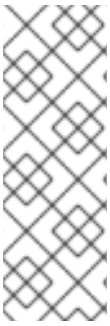
fixfiles -F onboot コマンドを入力する前に、必ず Permissive モードに切り替えてください。これにより、ラベルのないファイルがシステムに含まれている場合に、システムが起動に失敗することを防ぎます。詳細は、RHBZ#2021835 を参照してください。

5. 拒否がない場合は、Enforcing モードに切り替えます。詳細は システムの起動時に SELinux モードの変更 を参照してください。

検証

1. システムの再起動後に、**getenforce** コマンドが **Enforcing** を返すことを確認します。

```
$ getenforce
Enforcing
```



注記

Enforcing モードで SELinux を使用してカスタムアプリケーションを実行するには、次のいずれかのシナリオを選択してください。

- **unconfined_service_t** ドメインでアプリケーションを実行します。
- アプリケーションに新しいポリシーを記述します。詳細は、カスタム SELinux ポリシーの作成 のセクションを参照してください。

関連情報

- SELinux states and modes section covers temporary changes in modes.

17.2.5. SELinux の無効化

以下の手順に従って、SELinux を永続的に無効にします。

重要

SELinux が無効になっていると、SELinux ポリシーは読み込まれません。ポリシーは強制されず、AVC メッセージはログに記録されません。したがって、SELinux を実行する利点 はすべて失われます。

Red Hat は、SELinux を永続的に無効にする代わりに、Permissive モードを使用することを強く推奨します。Permissive モードの詳細は Permissive モードへの変更 を参照してください。

警告

/etc/selinux/config で **SELINUX=disabled** オプションを使用して SELinux を無効にすると、カーネルが SELinux を有効にして起動し、その後のブートプロセスで無効化モードに切り替わります。メモリーリークおよび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、SELinux を完全に無効にする必要がある場合に システムの起動時に SELinux モードの変更 で説明されているように、**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 を無効にすることが推奨されます。

手順

1. 任意のテキストエディターで **/etc/selinux/config** ファイルを開きます。以下に例を示します。

```
# vi /etc/selinux/config
```

2. **SELINUX=disabled** オプションを設定します。

```
# 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=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 変更を保存して、システムを再起動します。

```
# reboot
```

検証

1. 再起動したら、**getenforce** コマンドが **Disabled** を返すことを確認します。

```
$ getenforce
Disabled
```

17.2.6. システムの起動時に SELinux モードの変更

システムの起動時に、SELinux の実行方法を変更するカーネルパラメーターを設定できます。

enforcing=0

このパラメーターを設定すると、システムを起動する際に、Permissive モードで起動します。これは、問題のトラブルシューティングを行うときに便利です。ファイルシステムの破損がひどい場合は、Permissive モードを使用することが、問題を検出するための唯一の選択肢となるかもしれません。また、Permissive モードでは、ラベルの作成が適切に行われます。このモードで作成した AVC メッセージは、Enforcing モードと同じになるとは限りません。
Permissive モードでは、一連の同じ拒否の最初の拒否のみが報告されます。一方、Enforcing モードでは、ディレクトリーの読み込みに関する拒否が発生し、アプリケーションが停止する場合がします。Permissive モードでは、表示される AVC メッセージは同じですが、アプリケーションは、ディレクトリー内のファイルを読み続け、拒否が発生するたびに AVC を取得します。

selinux=0

このパラメーターにより、カーネルは、SELinux インフラストラクチャーのどの部分も読み込まないようになります。init スクリプトは、システムが **selinux=0** パラメーターで起動したことを認識し、**/.autorelabel** ファイルのタイムスタンプを変更します。これにより、次回 SELinux を有効にしてシステムを起動する際にシステムのラベルが自動的に再設定されます。

重要

Red Hat では、**selinux=0** パラメーターを使用することは推奨されません。システムをデバッグする場合は、Permissive モードを使用することが推奨されます。

autorelabel=1

このパラメーターにより、システムで、以下のコマンドと同様の再ラベルが強制的に行われます。

```
# touch /.autorelabel
# reboot
```

ファイルシステムに間違ったラベルが付いたオブジェクトが大量に含まれる場合は、システムを Permissive モードで起動して自動再ラベルプロセスを正常に実行します。

関連情報

- **checkreqprot** などの追加の SELinux 関連のカーネル起動パラメーターは、**kernel-doc** パッケージと一緒にインストールされる **/usr/share/doc/kernel-doc-<KERNEL_VER>/Documentation/admin-guide/kernel-parameters.txt** ファイルを参照してください。**<KERNEL_VER>** 文字列をインストール済みカーネルのバージョン番号に置き換えます。以下に例を示します。

```
# yum install kernel-doc
$ less /usr/share/doc/kernel-doc-4.18.0/Documentation/admin-guide/kernel-parameters.txt
```

17.3. SELINUX 関連の問題のトラブルシューティング

SELinux が無効になっていたシステムで SELinux を有効にする場合や、標準以外の設定でサービスを実行した場合は、SELinux がブロックできる状況のトラブルシューティングを行う必要がある可能性があります。ほとんどの場合、SELinux の拒否は、設定間違いによるものになります。

17.3.1. SELinux 拒否の特定

この手順で必要な手順のみを行います。ほとんどの場合は、ステップ 1 のみを実行する必要があります。

手順

1. SELinux がシナリオをブロックしたときに、最初に拒否に関する詳細情報を確認するのは **/var/log/audit/audit.log** ファイルとなります。Audit ログのクエリーには、**ausearch** ツールを使用します。アクセスを許可する、または許可しないといった SELinux の決定はキャッシュされ、このキャッシュは AVC (アクセスベクターキャッシュ) として知られています。たとえば、メッセージタイプパラメーターには **AVC** および **USER_AVC** の値を使用します。

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

一致するものがない場合は、Audit デーモンが実行しているかどうかを確認します。実行していない場合は、**auditd** を起動して Audit ログを再確認してから、拒否されたシナリオを繰り返します。

2. **auditd** が実行中で、**ausearch** の出力に一致がない場合は、**systemd** ジャーナルが出力するメッセージを確認してください。

```
# journalctl -t setroubleshoot
```

3. SELinux が有効で、Audit デーモンがシステムで実行していない場合は、**dmesg** コマンドの出力で特定の SELinux メッセージを検索します。

```
# dmesg | grep -i -e type=1300 -e type=1400
```

4. 以上の 3 つを確認しても、何も見つからない場合もあります。この場合、**dontaudit** ルールが原因で AVC 拒否を非表示にできます。
一時的に **dontaudit** ルールを無効にし、すべての拒否をログに記録できるようにするには、以下のコマンドを実行します。

```
# semodule -DB
```

以前の手順で、拒否されたシナリオを再実行し、拒否メッセージを取得したら、次のコマンドはポリシーで **dontaudit** ルールを再度有効にします。

```
# semodule -B
```

5. これまでの 4 つのステップをすべて試し、それでも問題が解決しない場合は、SELinux がシナリオをブロックするかどうかを検討してください。

   - Permissive モードに切り替えます。

```
# setenforce 0
$ getenforce
Permissive
```

   - シナリオを繰り返します。

   上記を試しても問題が発生する場合は、SELinux 以外に原因があります。

17.3.2. SELinux 拒否メッセージの分析

SELinux がシナリオをブロックしていることを 特定 したら、修正する前に原因分析が必要になる場合があります。

前提条件

- **policycoreutils-python-utils** パッケージおよび **setroubleshoot-server** パッケージがシステムにインストールされている。

手順

1. 以下のように、**sealert** コマンドを実行して、ログに記録されている拒否の詳細をリスト表示します。

```
$ sealert -l "*"
SELinux is preventing /usr/bin/passwd from write access on the file
/root/test.

*****  Plugin leaks (86.2 confidence) suggests ******************************

If you want to ignore passwd trying to write access the test file,
because you believe it should not need this access.
Then you should report this as a bug.
You can generate a local policy module to dontaudit this access.
Do
# ausearch -x /usr/bin/passwd --raw | audit2allow -D -M my-passwd
# semodule -X 300 -i my-passwd.pp

*****  Plugin catchall (14.7 confidence) suggests **************************

...

Raw Audit Messages
type=AVC msg=audit(1553609555.619:127): avc:  denied  { write } for
pid=4097 comm="passwd" path="/root/test" dev="dm-0" ino=17142697
scontext=unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

...

Hash: passwd,passwd_t,admin_home_t,file,write
```

2. 前の手順で取得した出力に明確な提案が含まれていない場合は、以下のコマンドを実行します。

   - 完全パス監査を有効にして、アクセスしたオブジェクトの完全パスを表示し、追加の Linux Audit イベントフィールドが表示されるようにします。

     ```
     # auditctl -w /etc/shadow -p w -k shadow-write
     ```

   - **setroubleshoot** キャッシュを削除します。

     ```
     # rm -f /var/lib/setroubleshoot/setroubleshoot.xml
     ```

- 問題を再現します。
- ステップ 1 を繰り返します。
  プロセスが完了したら、完全パスの監査を無効にします。

```
# auditctl -W /etc/shadow -p w -k shadow-write
```

3. **sealert** が **catchall** 提案を返すか、**audit2allow** ツールを使用して新しいルールを追加するように提案した場合は、Audit ログの SELinux 拒否 で説明されている例と問題を一致させます。

関連情報

- **sealert(8)** の man ページ

17.3.3. 分析した SELinux 拒否の修正

ほとんどの場合、**sealert** ツールが提供する提案により、SELinux ポリシーに関連する問題を修正するための適切なガイドが提供されます。SELinux 拒否のメッセージを解析 を参照してください。**sealert** を使用して SELinux 拒否の解析方法を参照してください。

ツールが、**audit2allow** ツールを使用して設定変更を提案している場合は注意が必要です。**audit2allow** を使用して、SELinux 拒否を確認する際に、最初のオプションとしてローカルポリシーモジュールを生成することはできません。トラブルシューティングは、ラベル付けの問題があるかどうかを最初に確認します。2 番目に多いのが、SELinux が、プロセスの設定変更を認識していない場合です。

ラベル付けの問題

ラベル付けの問題の一般的な原因として、非標準ディレクトリーがサービスに使用される場合が挙げられます。たとえば、管理者が、Web サイト **/var/www/html/** ではなく、**/srv/myweb/** を使用したい場合があります。Red Hat Enterprise Linux では、**/srv** ディレクトリーには **var_t** タイプのラベルが付けられます。**/srv** で作成されるファイルおよびディレクトリーは、このタイプを継承します。また、**/myserver** などの最上位のディレクトリーに新規作成したオブジェクトには、**default_t** タイプのラベルが付けられます。SELinux は、Apache HTTP Server (**httpd**) がこの両方のタイプにアクセスできないようにします。アクセスを許可するには、SELinux では、**/srv/myweb/** のファイルが **httpd** からアクセス可能であることを認識する必要があります。

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/myweb(/.*)?"
```

この **semanage** コマンドは、**/srv/myweb/** ディレクトリーおよびその下のすべてのファイルおよびディレクトリーのコンテキストを SELinux ファイルコンテキストの設定に追加します。**semanage** ユーティリティーはコンテキストを変更しません。root で **restorecon** ユーティリティーを使用して変更を適用します。

```
# restorecon -R -v /srv/myweb
```

コンテキストの誤り

matchpathcon ユーティリティーは、ファイルパスのコンテキストを確認し、そのパスのデフォルトラベルと比較します。以下の例では、ラベルが間違っているファイルを含むディレクトリーにおける **matchpathcon** の使用方法を説明します。

```
$ matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

```
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0, should be
system_u:object_r:httpd_sys_content_t:s0
```

この例では、**index.html** ファイルおよび **page1.html** ファイルに、**user_home_t** タイプのラベルが付けられています。このタイプは、ユーザーのホームディレクトリーのファイルに使用されます。**mv** コマンドを使用してファイルをホームディレクトリーから移動すると、ファイルに **user_home_t** タイプのラベルが付けられることがあります。このタイプは、ホームディレクトリーの外に存在してはなりません。このようなファイルを正しいタイプに復元するには、**restorecon** ユーティリティーを使用します。

```
# restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context unconfined_u:object_r:user_home_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

ディレクトリー下の全ファイルのコンテキストを復元するには、**-R** オプションを使用します。

```
# restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context unconfined_u:object_r:samba_share_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
```

標準以外の方法で設定された制限のあるアプリケーション

サービスはさまざまな方法で実行できます。この場合は、サービスの実行方法を指定する必要があります。これは、SELinux ポリシーの一部をランタイム時に変更できるようにする SELinux ブール値を使用して実行できます。これにより、SELinux ポリシーの再読み込みや再コンパイルを行わずに、サービスが NFS ボリュームにアクセスするのを許可するなどの変更が可能になります。また、デフォルト以外のポート番号でサービスを実行するには、**semanage** コマンドを使用してポリシー設定を更新する必要があります。

たとえば、Apache HTTP Server が MariaDB と接続するのを許可する場合は、**httpd_can_network_connect_db** のブール値を有効にします。

```
# setsebool -P httpd_can_network_connect_db on
```

-P オプションを使用すると、システムの再起動後も設定が永続化されることに注意してください。

特定のサービスでアクセスが拒否される場合は、**getsebool** ユーティリティーおよび **grep** ユーティリティーを使用して、アクセスを許可するブール値が利用できるかどうかを確認します。たとえば、**getsebool -a | grep ftp** コマンドを使用して FTP 関連のブール値を検索します。

```
$ getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_nfs --> off

ftpd_connect_db --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
```

ブール値のリストを取得し、ブール値が有効または無効かどうかを確認する場合は、**getsebool -a** コマンドを使用します。ブール値のリストとその意味を取得し、有効かどうかを調べるには、**selinux-policy-devel** パッケージをインストールして、root で **semanage boolean -l** コマンドを実行します。

ポート番号

ポリシー設定によっては、サービスは特定のポート番号でのみ実行できます。ポリシーを変更せずにサービスが実行するポートを変更しようとすると、サービスが起動できなくなる可能性があります。たとえば、root で **semanage port -l | grep http** コマンドを実行して、**http** 関連のポートを一覧表示します。

```
# semanage port -l | grep http
http_cache_port_t              tcp      3128, 8080, 8118
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

http_port_t ポートタイプは、Apache HTTP サーバーがリッスンできるポートを定義します。この場合は、TCP ポート 80、443、488、8008、8009、および 8443 です。**httpd** がポート 9876 (**Listen 9876**) でリッスンするように管理者が **httpd.conf** を設定しても、これを反映するようにポリシーが更新されていない場合、以下のコマンドは失敗します。

```
# systemctl start httpd.service
Job for httpd.service failed. See 'systemctl status httpd.service' and 'journalctl -xn' for details.

# systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: failed (Result: exit-code) since Thu 2013-08-15 09:57:05 CEST; 59s ago
  Process: 16874 ExecStop=/usr/sbin/httpd $OPTIONS -k graceful-stop (code=exited,
status=0/SUCCESS)
  Process: 16870 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
```

以下のような SELinux 拒否メッセージのログは、**/var/log/audit/audit.log** に記録されます。

```
type=AVC msg=audit(1225948455.061:294): avc:  denied  { name_bind } for  pid=4997
comm="httpd" src=9876 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:port_t:s0 tclass=tcp_socket
```

httpd が **http_port_t** ポートタイプに追加されていないポートをリッスンできるようにするには、**semanage port** コマンドを使用して別のラベルをポートに割り当てます。

```
# semanage port -a -t http_port_t -p tcp 9876
```

-a オプションは新規レコードを追加します。**-t** オプションはタイプを定義し、**-p** オプションはプロトコルを定義します。最後の引数は、追加するポート番号です。

進化または破損したアプリケーション、および侵害されたシステム (稀に発生する難しいケース)

アプリケーションにバグが含まれ、SELinux がアクセスを拒否する可能性があります。また、SELinux ルールは進化しています。アプリケーションが特定の方法で実行しているのを SELinux が認識しておらず、アプリケーションが期待どおりに動作していても、アクセスを拒否してしまうような場合もあります。たとえば、PostgreSQL の新規バージョンがリリースされると、現在のポリシーが考慮されないアクションが実行される可能性があり、アクセスが許可される場合でもアクセスが拒否されます。

このような状況では、アクセスが拒否された後に **audit2allow** ユーティリティーを使用して、アクセス

を許可するカスタムポリシーモジュールを作成します。SELinux ポリシーで足りないルールは Red Hat Bugzilla から報告できます。Red Hat Enterprise Linux 8 の場合は、製品で **Red Hat Enterprise Linux 8** を選択し、**selinux-policy** コンポーネントを選択してバグを作成します。このバグレポートに、**audit2allow -w -a** コマンドおよび **audit2allow -a** コマンドの出力を追加します。

アプリケーションが主要なセキュリティー特権を要求すると、そのアプリケーションが危険にさらされたことを示す警告が発生することがあります。侵入検出ツールを使用して、このような疑わしい動作を検証します。

また、Red Hat カスタマーポータル の Solution Engine では、同じ問題または非常に類似する問題に関する記事が提供されます。ここでは、問題の解決策と考えられる方法が示されています。関連する製品とバージョンを選択し、**selinux**、**avc** などの SELinux 関連のキーワードと、ブロックされたサービスまたはアプリケーションの名前 (**selinux samba** など) を使用します。

17.3.4. Audit ログの SELinux 拒否

Linux Audit システムは、デフォルトで **/var/log/audit/audit.log** ファイルにログエントリーを保存します。

SELinux 関連の記録のみをリスト表示するには、メッセージタイプパラメーターの **AVC** および **AVC_USER** (並びに必要に応じたパラメーター) を付けて **ausearch** コマンドを実行します。

```
# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR
```

Audit ログファイルの SELinux 拒否エントリーは次のようになります。

```
type=AVC msg=audit(1395177286.929:1638): avc:  denied  { read } for  pid=6591 comm="httpd"
name="webpages" dev="0:37" ino=2112 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:nfs_t:s0 tclass=dir
```

このエントリーで最も重要な部分は以下の通りです。

- **avc: denied** - SELinux によって実行され、アクセスベクターキャッシュ (AVC) で記録されるアクション
- **{ read }** - 拒否の動作
- **pid=6591** - 拒否されたアクションの実行を試みたサブジェクトのプロセス ID
- **comm="httpd"** - 分析しているプロセスを呼び出すのに使用されたコマンドの名前
- **httpd_t** - プロセスの SELinux タイプ
- **nfs_t** - プロセスのアクションに影響するオブジェクトの SELinux タイプ
- **tclass=dir** - ターゲットオブジェクトクラス

このログエントリーは、以下のように解釈できます。

SELinux が、nfs_t タイプのディレクトリーから読み込む PID 6591 および httpd_t タイプの httpd プロセスを拒否します。

Apache HTTP Server が Samba スイートのタイプでラベル付けされたディレクトリーにアクセスしようとすると、以下の SELinux 拒否メッセージが発生します。

```
type=AVC msg=audit(1226874073.147:96): avc:  denied  { getattr } for  pid=2465 comm="httpd"
path="/var/www/html/file1" dev=dm-0 ino=284133 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file
```

- **{ getattr }** - **getattr** エントリーは、ターゲットファイルのステータス情報をソースプロセスが読み取ろうとしているのを示します。これは、ファイルを読み取る前に発生します。プロセスがファイルにアクセスし、適切なラベルがないため、SELinux はこのアクションを拒否します。一般的に表示されるパーミッションには、**getattr**、**read**、**write** などが含まれます。
- **path="/var/www/html/file1"** - アクセスを試みたオブジェクト (ターゲット) へのパス。
- **scontext="unconfined_u:system_r:httpd_t:s0"** - 拒否されたアクションを試みたプロセス (ソース) の SELinux コンテキスト。この場合、Apache HTTP Server は **httpd_t** タイプで実行している SELinux コンテキストです。
- **tcontext="unconfined_u:object_r:samba_share_t:s0"** - プロセスがアクセスを試みたオブジェクト (ターゲット) の SELinux コンテキストです。この例では、これが **file1** の SELinux コンテキストです。

この SELinux 拒否は、以下のように解釈できます。

SELinux は、samba_share_t タイプの /var/www/html/file1 ファイルにアクセスする PID 2465 の httpd プロセスを拒否し、その他に許可するような設定がない場合は、httpd_t ドメインで実行しているプロセスにアクセスできません。

関連情報

- **auditd(8)** および **ausearch(8)** の man ページ

17.3.5. 関連情報

- CLI での基本的な SELinux トラブルシューティング
- Fedora People のプレゼンテーション - What is SELinux trying to tell me?The 4 key causes of SELinux エラー

[2] ホスト名から IP アドレスへのマッピングなど、DNS 情報を含むテキストファイル。

パート III. ネットワークの設計

第18章 IFCFG ファイルで IP ネットワークの設定

インターフェイス設定(**ifcfg**)ファイルは、個々のネットワークデバイスのソフトウェアインターフェイスを制御します。これは、システムの起動時に、このファイルを使用して、どのインターフェイスを起動するかと、どのように設定するかを決定します。これらのファイルの名前は **ifcfg-name_pass** です。接尾辞 **name** は、設定ファイルが制御するデバイスの名前を指します。通常、**ifcfg** ファイルの接尾辞は、設定ファイル自体の **DEVICE** ディレクティブが指定する文字列と同じです。

重要

NetworkManager は、鍵ファイル形式で保存されたプロファイルに対応します。ただし、NetworkManager の API を使用してプロファイルを作成または更新する場合、NetworkManager はデフォルトで **ifcfg** 形式を使用します。

将来のメジャーリリースの RHEL では、鍵ファイル形式がデフォルトになります。設定ファイルを手動で作成して管理する場合は、鍵ファイル形式の使用を検討してください。詳細は、鍵ファイル形式で NetworkManager プロファイルの手動による作成 を参照してください。

18.1. IFCFG ファイルの静的ネットワーク設定でインタフェースの設定

NetworkManager ユーティリティーおよびアプリケーションを使用しない場合は、**ifcfg** ファイルを作成してネットワークインターフェイスを手動で設定できます。

手順

- **ifcfg** ファイルを使用して、静的ネットワークで、インターフェイス **enp1s0** を設定するには、**/etc/sysconfig/network-scripts/** ディレクトリー内に、以下のような内容で **ifcfg-enp1s0** という名前のファイルを作成します。
  - **IPv4** 設定の場合は、以下のようになります。

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
GATEWAY=10.0.1.1
```

  - **IPv6** 設定の場合は、以下のようになります。

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=2001:db8:1::2/64
```

関連情報

- **nm-settings-ifcfg-rh(5)** man ページ

18.2. IFCFG ファイルの動的ネットワーク設定でインタフェースの設定

NetworkManager ユーティリティーおよびアプリケーションを使用しない場合は、**ifcfg** ファイルを作成してネットワークインターフェイスを手動で設定できます。

手順

1. **ifcfg** ファイルの動的ネットワークを使用して、インターフェイス **em1** を設定するには、**/etc/sysconfig/network-scripts/** ディレクトリーに、以下のような内容で、**ifcfg-em1** という名前のファイルを作成します。

```
DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes
```

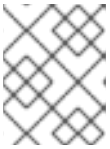
2. 送信するインターフェイスを設定するには、以下を行います。

   - **DHCP** サーバーに別のホスト名を追加し、**ifcfg** ファイルに以下の行を追加します。

```
DHCP_HOSTNAME=hostname
```

   - **DHCP** サーバーに、別の完全修飾ドメイン名 (FQDN) を追加し、 **ifcfg** ファイルに以下の行を追加します。

```
DHCP_FQDN=fully.qualified.domain.name
```



注記

この設定は、いずれか一方のみを使用できます。**DHCP_HOSTNAME** と **DHCP_FQDN** の両方を指定すると、**DHCP_FQDN** のみが使用されます。

3. インターフェイスが特定の **DNS** サーバーを使用するよう設定するには、**ifcfg** ファイルに以下の行を追加します。

```
PEERDNS=no
DNS1=ip-address
DNS2=ip-address
```

ここで、ip-address は **DNS** サーバーのアドレスです。これにより、ネットワークサービスは指定した **DNS** サーバーで **/etc/resolv.conf** を更新します。**DNS** サーバーアドレスは、1 つだけ必要です。もう 1 つは任意です。

18.3. IFCFG ファイルでシステム全体およびプライベート接続プロファイルの管理

デフォルトでは、ホスト上のすべてのユーザーが **ifcfg** ファイルで定義された接続を使用できます。**ifcfg** ファイルに **USERS** パラメーターを追加すると、この動作を特定ユーザーに制限できます。

前提条件

- **ifcfg** ファイルがすでに存在します。

手順

1. 特定のユーザーに制限する **/etc/sysconfig/network-scripts/** ディレクトリーの **ifcfg** ファイルを編集し、以下を追加します。

```
USERS="username1 username2 ..."
```

2. 接続をリアクティブにします。

```
# nmcli connection up connection_name
```

第19章 IPVLAN の使用

IPVLAN は、仮想ネットワークデバイス用のドライバーで、コンテナー環境でホストネットワークにアクセスするのに使用できます。IPVLAN は外部ネットワークに対し、ホストネットワーク内で作成された IPVLAN デバイスの数に関わらず、MAC アドレスを 1 つ公開します。つまり、ユーザーは複数コンテナーに複数の IPVLAN デバイスを持つことができますが、対応するスイッチは MAC アドレスを 1 つ読み込むということです。IPVLAN ドライバーは、ローカルスイッチで管理できる MAC アドレスの数に制限がある場合に役立ちます。

19.1. IPVLAN モード

IPVLAN では、次のモードが使用できます。

- **L2 モード**
  IPVLAN の **L2 モード** では、仮想デバイスは アドレス解決プロトコル (ARP) リクエストを受信して応答します。**netfilter** フレームワークは、仮想デバイスを所有するコンテナー内でのみ動作します。**netfilter** チェーンは、コンテナー化したトラッフィクにあるデフォルトの名前空間では実行されません。**L2 モード**を使用すると、パフォーマンスは高くなりますが、ネットワークトラフィックの制御性は低下します。

- **L3 モード**
  L3 モードでは、仮想デバイスは **L3** 以上のトラフィックのみを処理します。仮想デバイスは ARP リクエストに応答せず、関連するピアの IPVLAN IP アドレスは、隣接エントリーをユーザーが手動で設定する必要があります。関連するコンテナーの送信トラフィックはデフォルトの名前空間の **netfilter** の POSTROUTING および OUTPUT チェーンに到達する一方、ingress トラフィックは **L2 モード** と同様にスレッド化されます。**L3 モード** を使用すると、制御性は高くなりますが、ネットワークトラフィックのパフォーマンスは低下します。

- **L3S モード**
  L3S モード では、仮想デバイスは **L3 モード** と同様の処理をしますが、関連するコンテナーの egress トラフィックと ingress トラフィックの両方がデフォルトの名前空間の **netfilter** チェーンに到達する点が異なります。**L3S モード** は、**L3 モード** と同様の動作をしますが、ネットワークの制御が強化されます。

注記

IPVLAN 仮想デバイスは、**L3** モードおよび **L3S** モードでは、ブロードキャストトラフィックおよびマルチキャストトラフィックを受信しません。

19.2. IPVLAN および MACVLAN の比較

以下の表は、MACVLAN と IPVLAN の主な相違点を示しています。

MACVLAN	IPVLAN
各 MACVLAN デバイスに対して、MAC アドレスを使用します。スイッチの MAC テーブルの MAC アドレスの過剰制限により、接続が悪化する可能性があります。	IPVLAN デバイスの数を制限しない MAC アドレスを 1 つ使用します。

MACVLAN	IPVLAN
グローバル名前空間の netfilter ルールは、子名前空間の MACVLAN デバイスへのトラフィックに影響を及ぼしません。	**L3 モード** および **L3S モード** の IPVLAN デバイスとのトラフィックを制御できます。

IPVLAN および MACVLAN の両方には、カプセル化のレベルは必要ありません。

19.3. IPROUTE2 を使用した IPVLAN デバイスの作成および設定

この手順では、**iproute2** を使用して IPVLAN デバイスを設定する方法を説明します。

手順

1. IPVLAN デバイスを作成するには、次のコマンドを実行します。

```
# ip link add link real_NIC_device name IPVLAN_device type ipvlan mode l2
```

ネットワークインターフェイスコントローラー (NIC) は、コンピューターをネットワークに接続するハードウェアコンポーネントです。

例19.1 IPVLAN デバイスの作成

```
# ip link add link enp0s31f6 name my_ipvlan type ipvlan mode l2
# ip link
47: my_ipvlan@enp0s31f6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state
DOWN mode DEFAULT group default qlen 1000 link/ether e8:6a:6e:8a:a2:44 brd
ff:ff:ff:ff:ff:ff
```

2. **IPv4** アドレスまたは **IPv6** アドレスをインターフェイスに割り当てるには、次のコマンドを実行します。

```
# ip addr add dev IPVLAN_device IP_address/subnet_mask_prefix
```

3. **L3 モード** または **L3S モード** の IPVLAN デバイスを設定する場合は、以下の設定を行います。

   a. リモートホストのリモートピアのネイバー設定を行います。

```
# ip neigh add dev peer_device IPVLAN_device_IP_address lladdr MAC_address
```

MAC_address は、IPVLAN デバイスのベースである実際の NIC の MAC アドレスになります。

   b. **L3 モード** の IPVLAN デバイスを設定する場合は、次のコマンドを実行します。

```
# ip route add dev <real_NIC_device> <peer_IP_address/32>
```

L3S モード の場合は、次のコマンドを実行します。

```
# ip route add dev real_NIC_device peer_IP_address/32
```

IP アドレスは、リモートピアのアドレスを使用します。

4. IPVLAN デバイスをアクティブに設定するには、次のコマンドを実行します。

```
# ip link set dev IPVLAN_device up
```

5. IPVLAN デバイスがアクティブであることを確認するには、リモートホストで次のコマンドを実行します。

```
# ping IP_address
```

IP_address には、IPVLAN デバイスの IP アドレスを使用します。

第20章 異なるインターフェイスでの同じ IP アドレスの再利用

VRF (Virtual Routing and Forwarding) を使用すると、管理者は、同じホストで複数のルーティングテーブルを同時に使用できます。このため、VRF はレイヤー 3 でネットワークをパーティションで区切ります。これにより、管理者は、VRF ドメインごとに個別の独立したルートテーブルを使用してトラフィックを分離できるようになります。この技術は、レイヤー 2 でネットワークのパーティションを作成する仮想 LAN (VLAN) に類似しており、ここではオペレーティングシステムが異なる VLAN タグを使用して、同じ物理メディアを共有するトラフィックを分離させます。

レイヤー 2 のパーティションにある VRF の利点は、関与するピアの数に対して、ルーティングが適切にスケーリングすることです。

Red Hat Enterprise Linux は、各 VRF ドメインに仮想 **vrt** デバイスを使用し、既存のネットワークデバイスを VRF デバイスに追加して、VRF ドメインにルートを含めます。元のデバイスに接続していたアドレスとルートは、VRF ドメイン内に移動します。

各 VRF ドメインが互いに分離しているることに注意してください。

20.1. 別のインターフェイスで同じ IP アドレスを永続的に再利用する

VRF (Virtual Routing and Forwarding) 機能を使用して、1 台のサーバーの異なるインターフェイスで同じ IP アドレスを永続的に使用できます。

重要

同じ IP アドレスを再利用しながら、リモートのピアが VRF インターフェイスの両方に接続するようにするには、ネットワークインターフェイスが異なるブロードキャストドメインに属する必要があります。ネットワークのブロードキャストドメインは、ノードのいずれかによって送信されたブロードキャストトラフィックを受信するノードセットです。ほとんどの設定では、同じスイッチに接続されているすべてのノードが、同じブロードキャストドメインに属するようになります。

前提条件

- **root** ユーザーとしてログインしている。
- ネットワークインターフェイスが設定されていない。

手順

1. 最初の VRF デバイスを作成して設定します。
   a. VRF デバイスの接続を作成し、ルーティングテーブルに割り当てます。たとえば、ルーティングテーブル **1001** に割り当てられた **vrf0** という名前の VRF デバイスを作成するには、次のコマンドを実行します。

      ```
      # nmcli connection add type vrf ifname vrf0 con-name vrf0 table 1001 ipv4.method disabled ipv6.method disabled
      ```

   b. **vrf0** デバイスを有効にします。

      ```
      # nmcli connection up vrf0
      ```

c. 上記で作成した VRF にネットワークデバイスを割り当てます。たとえば、イーサネットデバイス **enp1s0** を **vrf0** VRF デバイスに追加し、IP アドレスとサブネットマスクを **enp1s0** に割り当てるには、次のコマンドを実行します。

```
# nmcli connection add type ethernet con-name vrf.enp1s0 ifname enp1s0 master vrf0 ipv4.method manual ipv4.address 192.0.2.1/24
```

d. **vrf.enp1s0** 接続をアクティベートします。

```
# nmcli connection up vrf.enp1s0
```

2. 次の VRF デバイスを作成して設定します。

a. VRF デバイスを作成し、ルーティングテーブルに割り当てます。たとえば、ルーティングテーブル **1002** に割り当てられた **vrf1** という名前の VRF デバイスを作成するには、次のコマンドを実行します。

```
# nmcli connection add type vrf ifname vrf1 con-name vrf1 table 1002 ipv4.method disabled ipv6.method disabled
```

b. **vrf1** デバイスをアクティベートします。

```
# nmcli connection up vrf1
```

c. 上記で作成した VRF にネットワークデバイスを割り当てます。たとえば、イーサネットデバイス **enp7s0** を **vrf1** VRF デバイスに追加し、IP アドレスとサブネットマスクを **enp7s0** に割り当てるには、次のコマンドを実行します。

```
# nmcli connection add type ethernet con-name vrf.enp7s0 ifname enp7s0 master vrf1 ipv4.method manual ipv4.address 192.0.2.1/24
```

d. **vrf.enp7s0** デバイスをアクティベートします。

```
# nmcli connection up vrf.enp7s0
```

20.2. 複数のインターフェイスで同じ IP アドレスを一時的に再利用

VRF (Virtual Routing and Forwarding) 機能を使用して、1台のサーバーの異なるインターフェイスで同じ IP アドレスを一時的に使用できます。この手順は、システムの再起動後に設定が一時的で失われてしまうため、テスト目的にのみ使用します。

重要

同じ IP アドレスを再利用しながら、リモートのピアが VRF インターフェイスの両方に接続するようにするには、ネットワークインターフェイスが異なるブロードキャストドメインに属する必要があります。ネットワークのブロードキャストドメインは、ノードのいずれかによって送信されたブロードキャストトラフィックを受信するノードセットです。ほとんどの設定では、同じスイッチに接続されているすべてのノードが、同じブロードキャストドメインに属するようになります。

前提条件

- **root** ユーザーとしてログインしている。

- ネットワークインターフェイスが設定されていない。

手順

1. 最初の VRF デバイスを作成して設定します。

   a. VRF デバイスを作成し、ルーティングテーブルに割り当てます。たとえば、**1001** ルーティングテーブルに割り当てられた **blue** という名前の VRF デバイスを作成するには、次のコマンドを実行します。

   ```
   # ip link add dev blue type vrf table 1001
   ```

   b. **blue** デバイスを有効にします。

   ```
   # ip link set dev blue up
   ```

   c. VRF デバイスにネットワークデバイスを割り当てます。たとえば、イーサネットデバイス **enp1s0** を、VRF デバイス **blue** に追加するには、次のコマンドを実行します。

   ```
   # ip link set dev enp1s0 master blue
   ```

   d. **enp1s0** デバイスを有効にします。

   ```
   # ip link set dev enp1s0 up
   ```

   e. IP アドレスとサブネットマスクを **enp1s0** デバイスに割り当てます。たとえば、**192.0.2.1/24** に設定するには、以下を実行します。

   ```
   # ip addr add dev enp1s0 192.0.2.1/24
   ```

2. 次の VRF デバイスを作成して設定します。

   a. VRF デバイスを作成し、ルーティングテーブルに割り当てます。たとえば、ルーティングテーブル **1002** に割り当てられた **red** という名前の VRF デバイスを作成するには、次のコマンドを実行します。

   ```
   # ip link add dev red type vrf table 1002
   ```

   b. **red** デバイスを有効にします。

   ```
   # ip link set dev red up
   ```

   c. VRF デバイスにネットワークデバイスを割り当てます。たとえば、イーサネットデバイス **enp7s0** を、VRF デバイス **red** に追加するには、次のコマンドを実行します。

   ```
   # ip link set dev enp7s0 master red
   ```

   d. **enp7s0** デバイスを有効にします。

   ```
   # ip link set dev enp7s0 up
   ```

e. VRF ドメイン **blue** の **enp1s0** に使用したものと同じ IP アドレスとサブネットマスクを **enp7s0** デバイスに割り当てます。

```
# ip addr add dev enp7s0 192.0.2.1/24
```

3. 必要に応じて、上記のとおりに、VRF デバイスをさらに作成します。

20.3. 関連情報

- **kernel-doc** パッケージの **/usr/share/doc/kernel-doc-<kernel_version>/Documentation/networking/vrf.txt**

第21章 ネットワークのセキュリティー保護

21.1. 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。

21.1.1. 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

21.1.2. 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. デフォルトの**0.0.0.0** (IPv4) または **::** とは異なるアドレスを指定するには、以下を行います。(IPv6) **/etc/ssh/sshd_config** 設定ファイルの **ListenAddress** ディレクティブ、および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。

21.1.3. 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。

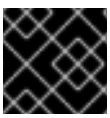
```
# systemctl reload sshd
```

関連情報

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

21.1.4. 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

21.1.5. 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

21.1.6. 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
```

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 --runtime-to-permanent
```

このコマンドで、**port_number** を、**Port** ディレクティブで指定された新しいポート番号に置き換えます。

root ログインなし

- 特定のユースケースで 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** に設定することを検討してください。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。

  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
  ```

- 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。このタスクには、暗号化オプションの設定に関する深い専門知識が必要になることに注意してください。

- 詳細は、Security hardening の Using system-wide cryptographic policies を参照してください

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

21.1.7. 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@ju
mp3.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

21.1.8. 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

- 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。

手順

1. オプション: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。

21.1.9. 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

21.2. TLS の計画および実施

TLS (トランスポート層セキュリティー) は、ネットワーク通信のセキュリティー保護に使用する暗号化プロトコルです。優先する鍵交換プロトコル、認証方法、および暗号化アルゴリズムを設定してシステムのセキュリティー設定を強化する際には、対応するクライアントの範囲が広ければ広いほど、セキュリティーのレベルが低くなることを認識しておく必要があります。反対に、セキュリティー設定を厳密にすると、クライアントとの互換性が制限され、システムからロックアウトされるユーザーが出てくる可能性もあります。可能な限り厳密な設定を目指し、互換性に必要な場合に限り、設定を緩めるようにしてください。

21.2.1. SSL プロトコルおよび TLS プロトコル

Secure Sockets Layer (SSL) プロトコルは、元々はインターネットを介した安全な通信メカニズムを提供するために、Netscape Corporation により開発されました。その後、このプロトコルは、Internet Engineering Task Force (IETF) により採用され、Transport Layer Security (TLS) に名前が変更になりました。

TLS プロトコルは、アプリケーションプロトコル層と、TCP/IP などの信頼性の高いトランスポート層の間にあります。これは、アプリケーションプロトコルから独立しているため、さまざまなプロトコルの下に階層化できます。(HTTP、FTP、SMTP など)

プロトコルのバージョン	推奨される使用方法

プロトコルのバージョン	推奨される使用方法
SSL v2	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 7 以降、コア暗号ライブラリーから削除されました。
SSL v3	使用しないでください。深刻なセキュリティー上の脆弱性があります。RHEL 8 以降、コア暗号ライブラリーから削除されました。
TLS 1.0	使用は推奨されません。相互運用性を保証した方法では軽減できない既知の問題があり、最新の暗号スイートには対応しません。RHEL 8 では、**LEGACY** システム全体の暗号化ポリシープロファイルでのみ有効です。
TLS 1.1	必要に応じて相互運用性の目的で使用します。最新の暗号スイートには対応しません。RHEL 8 では、**LEGACY** ポリシーでのみ有効になります。
TLS 1.2	最新の AEAD 暗号スイートに対応します。このバージョンは、システム全体のすべての暗号化ポリシーで有効になっていますが、このプロトコルの必須ではない部分に脆弱性があります。また、TLS 1.2 では古いアルゴリズムも使用できます。
TLS 1.3	推奨されるバージョン。TLS 1.3 は、既知の問題があるオプションを取り除き、より多くのネゴシエーションハンドシェイクを暗号化することでプライバシーを強化し、最新の暗号アルゴリズムをより効果的に使用することで速度を速めることができます。TLS 1.3 は、システム全体のすべての暗号化ポリシーでも有効になっています。

関連情報

- IETF:The Transport Layer Security (TLS) Protocol Version 1.3

21.2.2. RHEL 8 における TLS のセキュリティー上の検討事項

RHEL 8 では、システム全体の暗号化ポリシーにより、暗号化に関する検討事項が大幅に簡素化されています。**DEFAULT** 暗号化ポリシーは TLS 1.2 および 1.3 のみを許可します。システムが以前のバージョンの TLS を使用して接続をネゴシエートできるようにするには、アプリケーション内で次の暗号化ポリシーから除外するか、**update-crypto-policies** コマンドで **LEGACY** ポリシーに切り替える必要があります。詳細は、システム全体の暗号化ポリシーの使用 を参照してください。

大概のデプロイメントは、RHEL 8 に含まれるライブラリーが提供するデフォルト設定で十分に保護されます。TLS 実装は、可能な場合は、安全なアルゴリズムを使用する一方で、レガシーなクライアントまたはサーバーとの間の接続は妨げません。セキュリティーが保護されたアルゴリズムまたはプロトコルに対応しないレガシーなクライアントまたはサーバーの接続が期待できないまたは許可されない場合に、厳密なセキュリティー要件の環境で、強化設定を適用します。

TLS 設定を強化する最も簡単な方法は、**update-crypto-policies --set FUTURE** コマンドを実行して、システム全体の暗号化ポリシーレベルを **FUTURE** に切り替えます。

警告

LEGACY 暗号化ポリシーで無効にされているアルゴリズムは、Red Hat の RHEL 8 セキュリティーのビジョンに準拠しておらず、それらのセキュリティープロパティーは信頼できません。これらのアルゴリズムを再度有効化するのではなく、使用しないようにすることを検討してください。たとえば、古いハードウェアとの相互運用性のためにそれらを再度有効化することを決めた場合は、それらを安全でないものとして扱い、ネットワークの相互作用を個別のネットワークセグメントに分離するなどの追加の保護手段を適用します。パブリックネットワーク全体では使用しないでください。

RHEL システム全体の暗号化ポリシーに従わない場合、またはセットアップに適したカスタム暗号化ポリシーを作成する場合は、カスタム設定で必要なプロトコル、暗号スイート、および鍵の長さについて、以下の推奨事項を使用します。

21.2.2.1. プロトコル

最新バージョンの TLS は、最高のセキュリティーメカニズムを提供します。古いバージョンの TLS に対応しないといけないような特別な事態がない限り、システムは、TLS バージョン 1.2 以上を使用して接続をネゴシエートできるようにしてください。

RHEL 8 は TLS バージョン 1.3 をサポートしていますが、このプロトコルのすべての機能が RHEL 8 コンポーネントで完全にサポートされているわけではない点に注意してください。たとえば、接続レイテンシーを短縮する 0-RTT (Zero Round Trip Time) 機能は、Apache Web サーバーではまだ完全にはサポートされていません。

21.2.2.2. 暗号化スイート

旧式で、安全ではない暗号化スイートではなく、最近の、より安全なものを使用してください。暗号化スイートの eNULL および aNULL は、暗号化や認証を提供しないため、常に無効にしてください。RC4 や HMAC-MD5 をベースとした暗号化スイートには深刻な欠陥があるため、可能な場合はこれも無効にしてください。いわゆるエクスポート暗号化スイートも同様です。エクスポート暗号化スイートは意図的に弱くなっているため、侵入が容易になっています。

128 ビット未満のセキュリティーしか提供しない暗号化スイートでは直ちにセキュリティーが保護されなくなるというわけではありませんが、使用できる期間が短いため考慮すべきではありません。アルゴリズムが 128 ビット以上のセキュリティーを使用している場合は、少なくとも数年間は解読不可能であることが期待されているため、強く推奨されます。3DES 暗号は 168 ビットを使用していると言われていますが、実際に提供されているのは 112 ビットのセキュリティーであることに注意してください。

サーバーの鍵が危険にさらされた場合でも、暗号化したデータの機密性を保証する (完全な) 前方秘匿性 (PFS) に対応する暗号スイートを常に優先します。ここでは、速い RSA 鍵交換は除外されますが、ECDHE および DHE は使用できます。この 2 つを比べると、ECDHE の方が速いため推奨されます。

また、AES-GCM などの AEAD 暗号は、パディングオラクル攻撃の影響は受けないため、CBC モード暗号よりも推奨されます。さらに、多くの場合、特にハードウェアに AES 用の暗号化アクセラレーターがある場合、AES-GCM は CBC モードの AES よりも高速です。

ECDSA 証明書で ECDHE 鍵交換を使用すると、トランザクションは純粋な RSA 鍵交換よりもさらに高速になります。レガシークライアントに対応するため、サーバーには証明書と鍵のペアを 2 つ (新しいクライアント用の ECDSA 鍵と、レガシー用の RSA 鍵) インストールできます。

21.2.2.3. 公開鍵の長さ

RSA 鍵を使用する際は、SHA-256 以上で署名され、鍵の長さが 3072 ビット以上のものが常に推奨されます (これは、実際に 128 ビットであるセキュリティーに対して十分な大きさです)。

警告

システムのセキュリティー強度は、チェーンの中の最も弱いリンクが示すものと同じになります。たとえば、強力な暗号化だけではすぐれたセキュリティーは保証されません。鍵と証明書も同様に重要で、認証機関 (CA) が鍵の署名に使用するハッシュ機能と鍵もまた重要になります。

関連情報

- System-wide crypto policies in RHEL 8 .
- **update-crypto-policies(8)** の man ページ。

21.2.3. アプリケーションで TLS 設定の強化

RHEL では、システム全体の暗号化ポリシー は、暗号化ライブラリーを使用するアプリケーションが、既知の安全でないプロトコル、暗号化、またはアルゴリズムを許可しないようにするための便利な方法を提供します。

暗号化設定をカスタマイズして、TLS 関連の設定を強化する場合は、このセクションで説明する暗号化設定オプションを使用して、必要最小量でシステム全体の暗号化ポリシーを上書きできます。

いずれの設定を選択しても、サーバーアプリケーションが **サーバー側が指定した順序** で暗号を利用することを確認し、使用される暗号化スイートの選択がサーバーでの設定順に行われるように設定してください。

21.2.3.1. TLS を使用するように Apache HTTP サーバーを設定

Apache HTTP Server は、TLS のニーズに **OpenSSL** ライブラリーおよび **NSS** ライブラリーの両方を使用できます。RHEL 8 では、mod_ss パッケージで **mod_ssl** 機能が提供されます。

```
# yum install mod_ssl
```

mod_ssl パッケージは、**/etc/httpd/conf.d/ssl.conf** 設定ファイルをインストールします。これは、**Apache HTTP Server** の TLS 関連の設定を変更するのに使用できます。

httpd-manual パッケージをインストールして、TLS 設定を含む **Apache HTTP Server** の完全ドキュメントを取得します。**/etc/httpd/conf.d/ssl.conf** 設定ファイルで利用可能なディレクティブの詳細は、**/usr/share/httpd/manual/mod/mod_ssl.html** を参照してください。さまざまな設定の例は、**/usr/share/httpd/manual/ssl/ssl_howto.html** ファイルに記載されています。

/etc/httpd/conf.d/ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

SSLProtocol

このディレクティブを使用して、許可する TLS または SSL のバージョンを指定します。

SSLCipherSuite

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

SSLHonorCipherOrder

コメントを解除して、このディレクティブを **on** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、TLS 1.2 プロトコルおよび 1.3 プロトコルだけを使用する場合は、以下を実行します。

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
```

詳細は、Deploying different types of servers の Configuring TLS encryption on an Apache HTTP Server の章を参照してください。

21.2.3.2. TLS を使用するように Nginx HTTP およびプロキシーサーバーを設定

Nginx で TLS 1.3 サポートを有効にするには、**/etc/nginx/nginx.conf** 設定ファイルの **server** セクションで、**ssl_protocols** オプションに **TLSv1.3** 値を追加します。

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ....
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers
    ....
}
```

詳細は、Deploying different types of servers の Adding TLS encryption to an Nginx web server の章を参照してください。

21.2.3.3. TLS を使用するように Dovecot メールサーバーを設定

Dovecot メールサーバーのインストールが TLS を使用するように設定するには、**/etc/dovecot/conf.d/10-ssl.conf** 設定ファイルを修正します。このファイルで利用可能な基本的な設定ディレクティブの一部は、**/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt** ファイルで説明されています。このファイルは **Dovecot** の標準インストールに含まれています。

/etc/dovecot/conf.d/10-ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

ssl_protocols

このディレクティブを使用して、許可または無効にする TLS または SSL のバージョンを指定します。

ssl_cipher_list

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

ssl_prefer_server_ciphers

コメントを解除して、このディレクティブを **yes** に設定すると、接続先のクライアントは指定した暗号化の順序に従います。

たとえば、**/etc/dovecot/conf.d/10-ssl.conf** 内の次の行が、TLS 1.1 以降だけを許可します。

```
ssl_protocols = !SSLv2 !SSLv3 !TLSv1
```

関連情報

- Deploying different types of servers on RHEL 8
- **config(5)** および **ciphers(1)** の man ページ
- Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)
- Mozilla SSL Configuration Generator
- SSL Server Test

21.3. IPSEC を使用した VPN の設定

RHEL 8 では、仮想プライベートネットワーク (VPN) は **IPsec** プロトコルを使用して設定できます。これは、**Libreswan** アプリケーションによりサポートされます。

21.3.1. IPsec VPN 実装としての Libreswan

RHEL では、仮想プライベートネットワーク (VPN) は IPsec プロトコルを使用して設定できます。これは、Libreswan アプリケーションによりサポートされます。Libreswan は、Openswan アプリケーションの延長であり、Openswan ドキュメントの多くの例は Libreswan でも利用できます。

VPN の IPsec プロトコルは、IKE (Internet Key Exchange) プロトコルを使用して設定されます。IPsec と IKE は同義語です。IPsec VPN は、IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN、または IKE/IPsec VPN とも呼ばれます。Layer 2 Tunneling Protocol (L2TP) も使用する IPsec VPN のバリアントは、通常、L2TP/IPsec VPN と呼ばれ、**optional** のリポジトリーによって提供される **xl2tpd** パッケージが必要です。

Libreswan は、オープンソースのユーザー空間の IKE 実装です。IKE v1 および v2 は、ユーザーレベルのデーモンとして実装されます。IKE プロトコルも暗号化されています。IPsec プロトコルは Linux カーネルで実装され、Libreswan は、VPN トンネル設定を追加および削除するようにカーネルを設定します。

IKE プロトコルは、UDP ポート 500 および 4500 を使用します。IPsec プロトコルは、以下の 2 つのプロトコルで設定されます。

- 暗号セキュリティーペイロード (ESP) (プロトコル番号が 50)
- 認証ヘッダー (AH) (プロトコル番号 51)

AH プロトコルの使用は推奨されていません。AH のユーザーは、null 暗号化で ESP に移行することが推奨されます。

IPsec プロトコルは、以下の 2 つの操作モードを提供します。

- トンネルモード (デフォルト)
- トランスポートモード

IKE を使用せずに IPsec を使用してカーネルを設定できます。これは、**手動キーリング** と呼ばれます。また、**ip xfrm** コマンドを使用して手動キーを設定できますが、これはセキュリティー上の理由からは強く推奨されません。Libreswan では、netlink を使用する Linux カーネルで相互作用が行われます。Linux カーネルでパケットの暗号化と復号が行われます。

Libreswan は、ネットワークセキュリティーサービス (NSS) 暗号化ライブラリーを使用します。Libreswan および NSS はともに、**連邦情報処理標準** (**FIPS**) の公開文書 140-2 での使用が認定されています。

重要

Libreswan および Linux カーネルが実装する IKE/IPsec の VPN は、RHEL で使用することが推奨される唯一の VPN 技術です。その他の VPN 技術は、そのリスクを理解せずに使用しないでください。

RHEL では、Libreswan はデフォルトで **システム全体の暗号化ポリシー** に従います。これにより、Libreswan は、デフォルトのプロトコルとして IKEv2 を含む現在の脅威モデルに対して安全な設定を使用するようになります。詳細は、Using system-wide crypto policies を参照してください。

IKE/IPsec はピアツーピアプロトコルであるため、Libreswan では、ソースおよび宛先、またはサーバーおよびクライアントという用語を使用しません。終了点 (ホスト) を参照する場合は、代わりに左と右という用語を使用します。これにより、ほとんどの場合、両方の終了点で同じ設定も使用できます。ただし、管理者は通常、ローカルホストに左を使用し、リモートホストに右を使用します。

leftid と **rightid** オプションは、認証プロセス内の各ホストの識別として機能します。詳細は、man ページの **ipsec.conf(5)** を参照してください。

21.3.2. Libreswan の認証方法

Libreswan は複数の認証方法をサポートしますが、それぞれは異なるシナリオとなっています。

事前共有キー (PSK)

事前共有キー (PSK) は、最も簡単な認証メソッドです。セキュリティー上の理由から、PSK は 64 文字未満は使用しないでください。FIPS モードでは、PSK は、使用される整合性アルゴリズムに応じて、、最低強度の要件に準拠する必要があります。**authby=secret** 接続を使用して PSK を設定できます。

Raw RSA 鍵

Raw RSA 鍵 は、静的なホスト間またはサブネット間の IPsec 設定で一般的に使用されます。各ホストは、他のすべてのホストのパブリック RSA 鍵を使用して手動で設定され、Libreswan はホストの各ペア間で IPsec トンネルを設定します。この方法は、多数のホストでは適切にスケーリングされません。

ipsec newhostkey コマンドを使用して、ホストで Raw RSA 鍵を生成できます。 **ipsec showhostkey** コマンドを使用して、生成された鍵をリスト表示できます。**leftrsasigkey=** の行は、CKA ID キーを使用する接続設定に必要です。Raw RSA 鍵に **authby=rsasig** 接続オプションを使用します。

X.509 証明書

X.509 証明書 は、共通の IPsec ゲートウェイに接続するホストが含まれる大規模なデプロイメントに一般的に使用されます。中央の **認証局** (CA) は、ホストまたはユーザーの RSA 証明書に署名します。この中央 CA は、個別のホストまたはユーザーの取り消しを含む、信頼のリレーを行います。

たとえば、**openssl** コマンドおよび NSS **certutil** コマンドを使用して X.509 証明書を生成できます。Libreswan は、**leftcert=** 設定オプションの証明書のニックネームを使用して NSS データベースからユーザー証明書を読み取るため、証明書の作成時にニックネームを指定します。

カスタム CA 証明書を使用する場合は、これを Network Security Services(NSS) データベースにインポートする必要があります。**ipsec import** コマンドを使用して、PKCS #12 形式の証明書を Libreswan NSS データベースにインポートできます。

警告

Libreswan は、section 3.1 of RFC 4945 で説明されているように、すべてのピア証明書のサブジェクト代替名 (SAN) としてインターネット鍵 Exchange(IKE) ピア ID を必要とします。**require-id-on-certificated=** オプションを変更してこのチェックを無効にすると、システムが中間者攻撃に対して脆弱になる可能性があります。

SHA-1 および SHA-2 で RSA を使用した X.509 証明書に基づく認証に **authby=rsasig** 接続オプションを使用します。**authby=** を **ecdsa** に設定し、**authby=rsa-sha2** を介した SHA-2 による RSA Probabilistic Signature Scheme (RSASSA-PSS) デジタル署名ベースの認証を設定することにより、SHA-2 を使用する ECDSA デジタル署名に対してさらに制限することができます。デフォルト値は **authby=rsasig,ecdsa** です。

証明書と **authby=** 署名メソッドが一致する必要があります。これにより、相互運用性が向上し、1 つのデジタル署名システムでの認証が維持されます。

NULL 認証

null 認証 は、認証なしでメッシュの暗号化を取得するために使用されます。これは、パッシブ攻撃は防ぎますが、アクティブ攻撃は防ぎません。ただし、IKEv2 は非対称認証メソッドを許可するため、NULL 認証はインターネットスケールのオポチュニスティック IPsec にも使用できます。このモデルでは、クライアントはサーバーを認証しますが、サーバーはクライアントを認証しません。このモデルは、TLS を使用して Web サイトのセキュリティーを保護するのと似ています。NULL 認証に **authby=null** を使用します。

量子コンピューターに対する保護

上記の認証方法に加えて、**Post-quantum Pre-shared Key** (PPK) メソッドを使用して、量子コンピューターによる潜在的な攻撃から保護することができます。個々のクライアントまたはクライアントグループは、帯域外で設定された事前共有鍵に対応する PPK ID を指定することにより、独自の PPK を使用できます。

事前共有鍵が設定されている IKEv1 を使用すると、量子攻撃者からの保護が提供されます。IKEv2 の再設計は、この保護をネイティブに提供しません。Libreswan は、**Post-quantum Pre-shared Key** (PPK) を使用して、量子攻撃に対して IKEv2 接続を保護します。

任意の PPK 対応を有効にする場合は、接続定義に **ppk=yes** を追加します。PPK が必要な場合は **ppk=insist** を追加します。次に、各クライアントには、帯域外で通信する (および可能であれば量子攻撃に対して安全な) シークレット値を持つ PPK ID を付与できます。PPK はランダム性において非常に強力で、辞書の単語に基づいていません。PPK ID および PPK データは **ipsec.secrets** に保存されます。以下に例を示します。

```
@west @east : PPKS "user1" "thestringismeanttobearandomstr"
```

PPKS オプションは、静的な PPK を参照します。実験的な関数は、ワンタイムパッドに基づいた動的 PPK を使用します。各接続では、ワンタイムパッドの新しい部分が PPK として使用されます。これを使用すると、ファイル内の動的な PPK の部分がゼロで上書きされ、再利用を防ぐことができます。複

数のタイムパッドマテリアルが残っていないと、接続は失敗します。詳細は、man ページの **ipsec.secrets(5)** を参照してください。

警告

動的 PPK の実装はサポート対象外のテクノロジープレビューとして提供されます。注意して使用してください。

21.3.3. Libreswan のインストール

この手順では、Libreswan IPsec/IKE VPN 実装をインストールおよび起動を行う手順を説明します。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

2. Libreswan を再インストールする場合は、古いデータベースファイルを削除し、新しいデータベースを作成します。

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
# ipsec initnss
```

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

```
# systemctl enable ipsec --now
```

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

21.3.4. ホスト間の VPN の作成

Libreswan が、Raw RSA 鍵による認証を使用して、**left** と **right**と呼ばれる 2 つのホスト間にホスト間の IPsec VPN を作成するように設定するには、両方のホストに以下のコマンドを入力します。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。

手順

1. 各ホストで Raw RSA 鍵ペアを生成します。

```
# ipsec newhostkey
```

2. 前の手順で生成した鍵の **ckaid** を返します。**左** で次のコマンドを実行して、その **ckaid** を使用します。以下に例を示します。

```
# ipsec showhostkey --left --ckaid 2d3ea57b61c9419dfd6cf43a1eb6cb306c0e857d
```

上のコマンドの出力により、設定に必要な **leftrsasigkey=** 行が生成されます。次のホスト (**右**) でも同じ操作を行います。

```
# ipsec showhostkey --right --ckaid a9e1f6ce9ecd3608c24e8f701318383f41798f03
```

3. **/etc/ipsec.d/** ディレクトリーで、新しい **my_host-to-host.conf** ファイルを作成します。上の手順の **ipsec showhostkey** コマンドの出力から、RSA ホストの鍵を新規ファイルに書き込みます。以下に例を示します。

```
conn mytunnel
    leftid=@west
    left=192.1.2.23
    leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
    rightid=@east
    right=192.1.2.45
    rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    authby=rsasig
```

4. 鍵をインポートしたら、**ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

5. 接続を読み込みます。

```
# ipsec auto --add mytunnel
```

6. トンネルを確立します。

```
# ipsec auto --up mytunnel
```

7. **ipsec** サービスの開始時に自動的にトンネルを開始するには、以下の行を接続定義に追加します。

```
auto=start
```

21.3.5. サイト間 VPN の設定

2 つのネットワークを結合してサイト間の IPsec VPN を作成する場合は、その 2 つのホスト間の IPsec トンネルを作成します。これにより、ホストは終了点として動作し、1 つまたは複数のサブネットからのトラフィックが通過できるように設定されます。したがって、ホストを、ネットワークのリモート部分にゲートウェイとして見なすことができます。

サイト間の VPN の設定は、設定ファイル内で複数のネットワークまたはサブネットを指定する必要がある点のみが、ホスト間の VPN とは異なります。

前提条件

- ホスト間の VPN が設定されている。

手順

1. ホスト間の VPN の設定が含まれるファイルを、新規ファイルにコピーします。以下に例を示します。

```
# cp /etc/ipsec.d/my_host-to-host.conf /etc/ipsec.d/my_site-to-site.conf
```

2. 上の手順で作成したファイルに、サブネット設定を追加します。以下に例を示します。

```
conn mysubnet
    also=mytunnel
    leftsubnet=192.0.1.0/24
    rightsubnet=192.0.2.0/24
    auto=start

conn mysubnet6
    also=mytunnel
    leftsubnet=2001:db8:0:1::/64
    rightsubnet=2001:db8:0:2::/64
    auto=start

# the following part of the configuration file is the same for both host-to-host and site-to-site connections:

conn mytunnel
   leftid=@west
   left=192.1.2.23
   leftrsasigkey=0sAQOrlo+hOafUZDlCQmXFrje/oZm [...] W2n417C/4urYHQkCvuIQ==
   rightid=@east
   right=192.1.2.45
   rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
   authby=rsasig
```

21.3.6. リモートアクセスの VPN の設定

ロードウォーリアーとは、モバイルクライアントと動的に割り当てられた IP アドレスを使用する移動するユーザーのことです。モバイルクライアントは、X.509 証明書を使用して認証します。

以下の例では、**IKEv2** の設定を示しています。**IKEv1** XAUTH プロトコルは使用していません。

サーバー上では以下の設定になります。

```
conn roadwarriors
   ikev2=insist
   # support (roaming) MOBIKE clients (RFC 4555)
   mobike=yes
   fragmentation=yes
```

```
    left=1.2.3.4
    # if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
    # leftsubnet=10.10.0.0/16
    leftsubnet=0.0.0.0/0
    leftcert=gw.example.com
    leftid=%fromcert
    leftxauthserver=yes
    leftmodecfgserver=yes
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # pick an IP address pool to assign to remote users
    # 100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
    rightaddresspool=100.64.13.100-100.64.13.254
    # if you want remote clients to use some local DNS zones and servers
    modecfgdns="1.2.3.4, 5.6.7.8"
    modecfgdomains="internal.company.com, corp"
    rightxauthclient=yes
    rightmodecfgclient=yes
    authby=rsasig
    # optionally, run the client X.509 ID through pam to allow or deny client
    # pam-authorize=yes
    # load connection, do not initiate
    auto=add
    # kill vanished roadwarriors
    dpddelay=1m
    dpdtimeout=5m
    dpdaction=clear
```

ロードウォーリアーのデバイスであるモバイルクライアントでは、上記の設定に多少変更を加えて使用します。

```
conn to-vpn-server
    ikev2=insist
    # pick up our dynamic IP
    left=%defaultroute
    leftsubnet=0.0.0.0/0
    leftcert=myname.example.com
    leftid=%fromcert
    leftmodecfgclient=yes
    # right can also be a DNS hostname
    right=1.2.3.4
    # if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
    # rightsubnet=10.10.0.0/16
    rightsubnet=0.0.0.0/0
    fragmentation=yes
    # trust our own Certificate Agency
    rightca=%same
    authby=rsasig
    # allow narrowing to the server's suggested assigned IP and remote subnet
    narrowing=yes
    # support (roaming) MOBIKE clients (RFC 4555)
    mobike=yes
    # initiate connection
    auto=start
```

21.3.7. メッシュ VPN の設定

any-to-any VPN とも呼ばれるメッシュ VPN ネットワークは、全ノードが IPsec を使用して通信するネットワークです。この設定では、IPsec を使用できないノードの例外が許可されます。メッシュの VPN ネットワークは、以下のいずれかの方法で設定できます。

- IPSec を必要とする。
- IPsec を優先するが、平文通信へのフォールバックを可能にする。

ノード間の認証は、X.509 証明書または DNSSEC (DNS Security Extensions) を基にできます。

以下の手順では、X.509 証明書を使用します。これらの証明書は、Dongtag Certificate System などのいかなる種類の認証局 (CA) 管理システムを使用して生成できます。Dogtag は、各ノードの証明書が PKCS #12 形式 (.p12 ファイル) で利用可能であることを前提としています。これには、秘密鍵、ノード証明書、およびその他のノードの X.509 証明書を検証するのに使用されるルート CA 証明書が含まれます。

各ノードでは、その X.509 証明書を除いて、同じ設定を使用します。これにより、ネットワーク内で既存ノードを再設定せずに、新規ノードを追加できます。PKCS #12 ファイルには分かりやすい名前が必要であるため、名前には node を使用します。これにより、すべてのノードに対して、この名前を参照する設定ファイルが同一になります。

前提条件

- Libreswan がインストールされ、**ipsec** サービスが各ノードで開始している。

手順

1. 各ノードで PKCS #12 ファイルをインポートします。この手順では、PKCS #12 ファイルの生成に使用するパスワードが必要になります。

```
# ipsec import nodeXXX.p12
```

2. **IPsec required** (private)、**IPsec optional** (private-or-clear)、および **No IPsec** (clear) プロファイルに、以下のような 3 つの接続定義を作成します。

```
# cat /etc/ipsec.d/mesh.conf
conn clear
 auto=ondemand
 type=passthrough
 authby=never
 left=%defaultroute
 right=%group

conn private
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=drop
 negotiationshunt=drop
 # left
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert
        leftrsasigkey=%cert
```

```
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup

conn private-or-clear
 auto=ondemand
 type=transport
 authby=rsasig
 failureshunt=passthrough
 negotiationshunt=passthrough
 # left
 left=%defaultroute
 leftcert=nodeXXXX
 leftid=%fromcert
       leftrsasigkey=%cert
 # right
 rightrsasigkey=%cert
 rightid=%fromcert
 right=%opportunisticgroup
```

3. 適切なカテゴリーに、ネットワークの IP アドレスを追加します。たとえば、すべてのノードが 10.15.0.0/16 ネットワークにある場合は、すべてのノードに IPsec 暗号が必要です。

```
# echo "10.15.0.0/16" >> /etc/ipsec.d/policies/private
```

4. 特定のノード (例: 10.15.34.0/24) を、IPsec を使用または使用せずに機能させるには、以下の設定を使用して、これらのノードを private-or-clear グループに追加します。

```
# echo "10.15.34.0/24" >> /etc/ipsec.d/policies/private-or-clear
```

5. ホストを、10.15.1.2 など、IPsec の機能がない clear グループに定義する場合は、次のコマンドを実行します。

```
# echo "10.15.1.2/32" >> /etc/ipsec.d/policies/clear
```

/etc/ipsec.d/policies ディレクトリーのファイルは、各新規ノードのテンプレートから作成することも、Puppet または Ansible を使用してプロビジョニングすることもできます。

すべてのノードでは、例外のリストが同じか、異なるトラフィックフローが期待される点に注意してください。したがって、あるノードで IPsec が必要になり、別のノードで IPsec を使用できないために、ノード間の通信ができない場合もあります。

6. ノードを再起動して、設定したメッシュに追加します。

```
# systemctl restart ipsec
```

7. ノードを追加したら、**ping** コマンドで IPsec トンネルを開くだけで十分です。ノードが開くトンネルを確認するには、次のコマンドを実行します。

```
# ipsec trafficstatus
```

21.3.8. FIPS 準拠の IPsec VPN のデプロイメント

この手順を使用して、Libreswan に基づく FIPS 準拠の IPsec VPN ソリューションをデプロイします。次の手順では、FIPS モードの Libreswan で使用可能な暗号化アルゴリズムと無効になっている暗号化アルゴリズムを識別することもできます。

前提条件

- **AppStream** リポジトリーが有効になっている。

手順

1. **libreswan** パッケージをインストールします。

```
# yum install libreswan
```

2. Libreswan を再インストールする場合は、古い NSS データベースを削除します。

```
# systemctl stop ipsec
# rm /etc/ipsec.d/*db
```

3. **ipsec** サービスを開始して有効にし、システムの起動時にサービスを自動的に開始できるようにします。

```
# systemctl enable ipsec --now
```

4. ファイアウォールで、**ipsec** サービスを追加して、IKE プロトコル、ESP プロトコル、および AH プロトコルの 500/UDP ポートおよび 4500/UDP ポートを許可するように設定します。

```
# firewall-cmd --add-service="ipsec"
# firewall-cmd --runtime-to-permanent
```

5. システムを FIPS モードに切り替えます。

```
# fips-mode-setup --enable
```

6. システムを再起動して、カーネルを FIPS モードに切り替えます。

```
# reboot
```

検証

1. Libreswan が FIPS モードで実行していることを確認するには、次のコマンドを実行します。

```
# ipsec whack --fipsstatus
000 FIPS mode enabled
```

2. または、**systemd** ジャーナルで **ipsec** ユニットのエントリーを確認します。

```
$ journalctl -u ipsec
...
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Product: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Kernel: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Mode: YES
```

3. FIPS モードで使用可能なアルゴリズムを表示するには、次のコマンドを実行します。

```
# ipsec pluto --selftest 2>&1 | head -11
FIPS Product: YES
FIPS Kernel: YES
FIPS Mode: YES
NSS DB directory: sql:/etc/ipsec.d
Initializing NSS
Opening NSS database "sql:/etc/ipsec.d" read-only
NSS initialized
NSS crypto library initialized
FIPS HMAC integrity support [enabled]
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity verification self-test passed
```

4. FIPS モードで無効化されたアルゴリズムをクエリーするには、次のコマンドを実行します。

```
# ipsec pluto --selftest 2>&1 | grep disabled
Encryption algorithm CAMELLIA_CTR disabled; not FIPS compliant
Encryption algorithm CAMELLIA_CBC disabled; not FIPS compliant
Encryption algorithm SERPENT_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_SSH disabled; not FIPS compliant
Encryption algorithm NULL disabled; not FIPS compliant
Encryption algorithm CHACHA20_POLY1305 disabled; not FIPS compliant
Hash algorithm MD5 disabled; not FIPS compliant
PRF algorithm HMAC_MD5 disabled; not FIPS compliant
PRF algorithm AES_XCBC disabled; not FIPS compliant
Integrity algorithm HMAC_MD5_96 disabled; not FIPS compliant
Integrity algorithm HMAC_SHA2_256_TRUNCBUG disabled; not FIPS compliant
Integrity algorithm AES_XCBC_96 disabled; not FIPS compliant
DH algorithm MODP1024 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant
```

5. FIPS モードで許可されているすべてのアルゴリズムと暗号のリストを表示するには、次のコマンドを実行します。

```
# ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
{256,192,*128}  aes_ccm, aes_ccm_c
{256,192,*128}  aes_ccm_b
{256,192,*128}  aes_ccm_a
[*192]  3des
{256,192,*128}  aes_gcm, aes_gcm_c
{256,192,*128}  aes_gcm_b
{256,192,*128}  aes_gcm_a
{256,192,*128}  aesctr
{256,192,*128}  aes
{256,192,*128}  aes_gmac
sha, sha1, sha1_96, hmac_sha1
sha512, sha2_512, sha2_512_256, hmac_sha2_512
sha384, sha2_384, sha2_384_192, hmac_sha2_384
sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
```

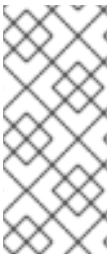
```
null
null, dh0
dh14
dh15
dh16
dh17
dh18
ecp_256, ecp256
ecp_384, ecp384
ecp_521, ecp521
```

関連情報

- Using system-wide cryptographic policies.

21.3.9. パスワードによる IPsec NSS データベースの保護

デフォルトでは、IPsec サービスは、初回起動時に空のパスワードを使用して Network Security Services (NSS) データベースを作成します。パスワード保護を追加するには、以下の手順を実行します。



注記

以前の RHEL 6.6 リリースでは、NSS 暗号化ライブラリーが FIPS 140-2 Level 2 標準で認定されているため、FIPS 140-2 要件を満たすパスワードで IPsec NSS データベースを保護する必要がありました。RHEL 8 では、この規格の NIST 認定 NSS がこの規格のレベル 1 に認定されており、このステータスではデータベースのパスワード保護は必要ありません。

前提条件

- **/etc/ipsec.d/** ディレクトリーに NSS データベースファイルが含まれます。

手順

1. Libreswan の **NSS** データベースのパスワード保護を有効にします。

```
# certutil -N -d sql:/etc/ipsec.d
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
```

2. 前の手順で設定したパスワードを追加した **/etc/ipsec.d/nsspassword** ファイルを作成します。以下に例を示します。

```
# cat /etc/ipsec.d/nsspassword
NSS Certificate DB:MyStrongPasswordHere
```

nsspassword ファイルは以下の構文を使用することに注意してください。

```
token_1_name:the_password
token_2_name:the_password
```

デフォルトの NSS ソフトウェアトークンは **NSS Certificate DB** です。システムが FIPS モードで実行し場合は、トークンの名前が **NSS FIPS 140-2 Certificate DB** になります。

3. 選択したシナリオに応じて、**nsspassword** ファイルの完了後に **ipsec** サービスを起動または再起動します。

```
# systemctl restart ipsec
```

検証

1. NSS データベースに空でないパスワードを追加した後に、**ipsec** サービスが実行中であることを確認します。

```
# systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
  Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor preset: disable>
  Active: active (running)...
```

2. 必要に応じて、初期化の成功を示すエントリーが **Journal** ログに含まれていることを確認します。

```
# journalctl -u ipsec
...
pluto[6214]: Initializing NSS using read-write database "sql:/etc/ipsec.d"
pluto[6214]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate
DB" with length 20 passed to NSS
pluto[6214]: NSS crypto library initialized
...
```

関連情報

- **certutil(1)** man ページ。
- Government Standards ナレッジベースアーティクル

21.3.10. TCP を使用するように IPsec VPN を設定

Libreswan は、RFC 8229 で説明されているように、IKE パケットおよび IPsec パケットの TCP カプセル化に対応します。この機能により、UDP 経由でトラフィックが転送されないように、IPsec VPN をネットワークに確立し、セキュリティーのペイロード (ESP) を強化できます。フォールバックまたはメインの VPN トランスポートプロトコルとして TCP を使用するように VPN サーバーおよびクライアントを設定できます。TCP カプセル化にはパフォーマンスコストが大きくなるため、UDP がシナリオで永続的にブロックされている場合に限り、TCP を主な VPN プロトコルとして使用してください。

前提条件

- リモートアクセス VPN が設定されている。

手順

1. **config setup** セクションの **/etc/ipsec.conf** ファイルに以下のオプションを追加します。

```
listen-tcp=yes
```

2. UDP で最初の試行に失敗した場合に TCP カプセル化をフォールバックオプションとして使用するには、クライアントの接続定義に以下の 2 つのオプションを追加します。

```
enable-tcp=fallback
tcp-remoteport=4500
```

または、UDP を永続的にブロックしている場合は、クライアントの接続設定で以下のオプションを使用します。

```
enable-tcp=yes
tcp-remoteport=4500
```

関連情報

- IETF RFC 8229:TCP Encapsulation of IKE and IPsec Packets

21.3.11. IPsec 接続を高速化するために、ESP ハードウェアオフロードの自動検出と使用を設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、Ethernet で IPsec 接続が加速します。デフォルトでは、Libreswan は、ハードウェアがこの機能に対応しているかどうかを検出するため、ESP ハードウェアのオフロードを有効にします。機能が無効になっているか、明示的に有効になっている場合は、自動検出に戻すことができます。

前提条件

- ネットワークカードは、ESP ハードウェアオフロードに対応します。
- ネットワークドライバーは、ESP ハードウェアのオフロードに対応します。
- IPsec 接続が設定され、動作する。

手順

1. ESP ハードウェアオフロードサポートの自動検出を使用する接続の **/etc/ipsec.d/** ディレクトリーにある Libreswan 設定ファイルを編集します。
2. 接続の設定で **nic-offload** パラメーターが設定されていないことを確認します。
3. **nic-offload** を削除した場合は、**ipsec** を再起動します。

```
# systemctl restart ipsec
```

検証

ネットワークカードが ESP ハードウェアオフロードサポートに対応している場合は、以下の手順に従って結果を検証します。

1. IPsec 接続が使用するイーサネットデバイスの **tx_ipsec** および **rx_ipsec** カウンターを表示します。

```
# ethtool -S enp1s0 | egrep "_ipsec"
    tx_ipsec: 10
    rx_ipsec: 10
```

2. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

```
# ping -c 5 remote_ip_address
```

3. イーサネットデバイスの **tx_ipsec** および **rx_ipsec** カウンターを再度表示します。

```
# ethtool -S enp1s0 | egrep "_ipsec"
    tx_ipsec: 15
    rx_ipsec: 15
```

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- IPsec を使用した VPN の設定

21.3.12. IPsec 接続を加速化するためにボンディングでの ESP ハードウェアオフロードの設定

Encapsulating Security Payload (ESP) をハードウェアにオフロードすると、IPsec 接続が加速します。フェイルオーバーの理由でネットワークボンディングを使用する場合、ESP ハードウェアオフロードを設定する要件と手順は、通常のイーサーネットデバイスを使用する要件と手順とは異なります。たとえば、このシナリオでは、ボンディングでオフロードサポートを有効にし、カーネルはボンディングのポートに設定を適用します。

前提条件

- ボンディングのすべてのネットワークカードが、ESP ハードウェアオフロードをサポートしている。
- ネットワークドライバーが、ボンドデバイスで ESP ハードウェアオフロードに対応している。RHEL では、**ixgbe** ドライバーのみがこの機能をサポートします。
- ボンディングが設定されており動作する。
- ボンディングで **active-backup** モードを使用している。ボンディングドライバーは、この機能の他のモードはサポートしていません。
- IPsec 接続が設定され、動作する。

手順

1. ネットワークボンディングで ESP ハードウェアオフロードのサポートを有効にします。

```
# nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

このコマンドにより、**bond0** 接続での ESP ハードウェアオフロードのサポートが有効になります。

2. **bond0** 接続を再度アクティブにします。

```
# nmcli connection up bond0
```

3. ESP ハードウェアオフロードに使用すべき接続の **/etc/ipsec.d/** ディレクトリーにある Libreswan 設定ファイルを編集し、**nic-offload=yes** ステートメントを接続エントリーに追加します。

```
conn example
    ...
    nic-offload=yes
```

4. **ipsec** サービスを再起動します。

```
# systemctl restart ipsec
```

検証

1. ボンディングのアクティブなポートを表示します。

```
# grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

2. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを表示します。

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 10
     rx_ipsec: 10
```

3. IPsec トンネルを介してトラフィックを送信します。たとえば、リモート IP アドレスに ping します。

```
# ping -c 5 remote_ip_address
```

4. アクティブなポートの **tx_ipsec** カウンターおよび **rx_ipsec** カウンターを再度表示します。

```
# ethtool -S enp1s0 | egrep "_ipsec"
     tx_ipsec: 15
     rx_ipsec: 15
```

カウンターの値が増えると、ESP ハードウェアオフロードが動作します。

関連情報

- ネットワークボンディングの設定
- ネットワークのセキュリティー保護ドキュメントの Configuring a VPN with IPsec セクション

21.3.13. システム全体の暗号化ポリシーをオプトアウトする IPsec 接続の設定

接続向けのシステム全体の暗号化ポリシーのオーバーライド

RHEL のシステム全体の暗号化ポリシーでは、**%default** と呼ばれる特別な接続が作成されます。この接続には、**ikev2** オプション、**esp** オプション、および **ike** オプションのデフォルト値が含まれます。ただし、接続設定ファイルに上記のオプションを指定すると、デフォルト値を上書きできます。

たとえば、次の設定では、AES および SHA-1 または SHA-2 で IKEv1 を使用し、AES-GCM または AES-CBC で IPsec (ESP) を使用する接続が可能です。

```
conn MyExample
  ...
  ikev2=never
  ike=aes-sha2,aes-sha1;modp2048
  esp=aes_gcm,aes-sha2,aes-sha1
  ...
```

AES-GCM は IPsec (ESP) および IKEv2 で利用できますが、IKEv1 では利用できません。

全接続向けのシステム全体の暗号化ポリシーの無効化

すべての IPsec 接続のシステム全体の暗号化ポリシーを無効にするには、**/etc/ipsec.conf** ファイルで次の行をコメントアウトします。

```
include /etc/crypto-policies/back-ends/libreswan.config
```

次に、接続設定ファイルに **ikev2=never** オプションを追加してください。

関連情報

- Using system-wide cryptographic policies.

21.3.14. IPsec VPN 設定のトラブルシューティング

IPsec VPN 設定に関連する問題は主に、一般的な理由が原因で発生する可能性が高くなっています。このような問題が発生した場合は、問題の原因が以下のシナリオのいずれかに該当するかを確認して、対応するソリューションを適用します。

基本的な接続のトラブルシューティング

VPN 接続関連の問題の多くは、管理者が不適当な設定オプションを指定してエンドポイントを設定した新しいデプロイメントで発生します。また、互換性のない値が新たに実装された場合に、機能していた設定が突然動作が停止する可能性があります。管理者が設定を変更した場合など、このような結果になることがあります。また、管理者が暗号化アルゴリズムなど、特定のオプションに異なるデフォルト値を使用して、ファームウェアまたはパッケージの更新をインストールした場合などです。

IPsec VPN 接続が確立されていることを確認するには、次のコマンドを実行します。

```
# ipsec trafficstatus
006 #8: "vpn.example.com"[1] 192.0.2.1, type=ESP, add_time=1595296930, inBytes=5999,
outBytes=3231, id='@vpn.example.com', lease=100.64.13.5/32
```

出力が空の場合や、エントリーで接続名が表示されない場合など、トンネルが破損します。

接続に問題があることを確認するには、以下を実行します。

1. **vpn.example.com** 接続をもう一度読み込みます。

```
# ipsec auto --add vpn.example.com
002 added connection description "vpn.example.com"
```

2. 次に、VPN 接続を開始します。

```
# ipsec auto --up vpn.example.com
```

ファイアウォール関連の問題

最も一般的な問題は、IPSec エンドポイントの 1 つ、またはエンドポイント間にあるルーターにあるファイアウォールで Internet Key Exchange (IKE) パケットがドロップされるという点が挙げられます。

- IKEv2 の場合には、以下の例のような出力は、ファイアウォールに問題があることを示しています。

```
# ipsec auto --up vpn.example.com
181 "vpn.example.com"[1] 192.0.2.2 #15: initiating IKEv2 IKE SA
181 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: sent v2I1, expected v2R1
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 0.5
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 1
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 2
seconds for
...
```

- IKEv1 の場合は、最初のコマンドの出力は以下のようになります。

```
# ipsec auto --up vpn.example.com
002 "vpn.example.com" #9: initiating Main Mode
102 "vpn.example.com" #9: STATE_MAIN_I1: sent MI1, expecting MR1
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 0.5 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 1 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 2 seconds for
response
...
```

IPsec の設定に使用される IKE プロトコルは暗号化されているため、**tcpdump** ツールを使用して、トラブルシューティングできるサブセットは一部のみです。ファイアウォールが IKE パケットまたは IPsec パケットをドロップしている場合は、**tcpdump** ユーティリティーを使用して原因を見つけることができます。ただし、**tcpdump** は IPsec VPN 接続に関する他の問題を診断できません。

- **eth0** インターフェイスで VPN および暗号化データすべてのネゴシエーションを取得するには、次のコマンドを実行します。

```
# tcpdump -i eth0 -n -n esp or udp port 500 or udp port 4500 or tcp port 4500
```

アルゴリズム、プロトコル、およびポリシーが一致しない場合

VPN 接続では、エンドポイントが IKE アルゴリズム、IPsec アルゴリズム、および IP アドレス範囲に一致する必要があります。不一致が発生した場合には接続は失敗します。以下の方法のいずれかを使用して不一致を特定した場合は、アルゴリズム、プロトコル、またはポリシーを調整して修正します。

- リモートエンドポイントが IKE/IPsec を実行していない場合は、そのパケットを示す ICMP パケットが表示されます。以下に例を示します。

```
# ipsec auto --up vpn.example.com
...
000 "vpn.example.com"[1] 192.0.2.2 #16: ERROR: asynchronous network error report on
wlp2s0 (192.0.2.2:500), complainant 198.51.100.1: Connection refused [errno 111, origin
ICMP type 3 code 3 (not authenticated)]
...
```

- IKE アルゴリズムが一致しない例:

```
# ipsec auto --up vpn.example.com
...
003 "vpn.example.com"[1] 193.110.157.148 #3: dropping unexpected IKE_SA_INIT message
containing NO_PROPOSAL_CHOSEN notification; message payloads: N; missing payloads:
SA,KE,Ni
```

- IPsec アルゴリズムが一致しない例:

```
# ipsec auto --up vpn.example.com
...
182 "vpn.example.com"[1] 193.110.157.148 #5: STATE_PARENT_I2: sent v2I2, expected
v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_256
group=MODP2048}
002 "vpn.example.com"[1] 193.110.157.148 #6: IKE_AUTH response contained the error
notification NO_PROPOSAL_CHOSEN
```

また、IKE バージョンが一致しないと、リモートエンドポイントが応答なしの状態でリクエストをドロップする可能性がありました。これは、すべての IKE パケットをドロップするファイアウォールと同じです。

- IKEv2 (Traffic Selectors - TS) の IP アドレス範囲が一致しない例:

```
# ipsec auto --up vpn.example.com
...
1v2 "vpn.example.com" #1: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
002 "vpn.example.com" #2: IKE_AUTH response contained the error notification
TS_UNACCEPTABLE
```

- IKEv1 の IP アドレス範囲で一致しない例:

```
# ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0
retransmits.  No acceptable response to our first Quick Mode message: perhaps peer likes
no proposal
```

- IKEv1 で PreSharedKeys (PSK) を使用する場合には、どちらでも同じ PSK に配置されなければ、IKE メッセージ全体の読み込みができなくなります。

```
# ipsec auto --up vpn.example.com
...
```

```
003 "vpn.example.com" #1: received Hash Payload does not match computed value
223 "vpn.example.com" #1: sending notification INVALID_HASH_INFORMATION to
192.0.2.23:500
```

- IKEv2 では、 mismatched-PSK エラーが原因で AUTHENTICATION_FAILED メッセージが表示されます。

```
# ipsec auto --up vpn.example.com
...
002 "vpn.example.com" #1: IKE SA authentication request rejected by peer:
AUTHENTICATION_FAILED
```

最大伝送単位 (MTU)

ファイアウォールが IKE または IPSec パケットをブロックする以外で、ネットワークの問題の原因として、暗号化パケットのパケットサイズの増加が最も一般的です。ネットワークハードウェアは、最大伝送単位 (MTU) を超えるパケットを 1500 バイトなどのサイズに断片化します。多くの場合、断片化されたパケットは失われ、パケットの再アセンブルに失敗します。これにより、小さいサイズのパケットを使用する ping テスト時には機能し、他のトラフィックでは失敗するなど、断続的な問題が発生します。このような場合に、SSH セッションを確立できますが、リモートホストに 'ls -al /usr' コマンドに入力した場合など、すぐにターミナルがフリーズします。

この問題を回避するには、トンネル設定ファイルに **mtu=1400** のオプションを追加して、MTU サイズを縮小します。

または、TCP 接続の場合は、MSS 値を変更する iptables ルールを有効にします。

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

各シナリオで上記のコマンドを使用して問題が解決されない場合は、**set-mss** パラメーターで直接サイズを指定します。

```
# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1380
```

ネットワークアドレス変換 (NAT)

IPsec ホストが NAT ルーターとしても機能すると、誤ってパケットが再マッピングされる可能性があります。以下の設定例はこの問題について示しています。

```
conn myvpn
    left=172.16.0.1
    leftsubnet=10.0.2.0/24
    right=172.16.0.2
    rightsubnet=192.168.0.0/16
…
```

アドレスが 172.16.0.1 のシステムには NAT ルールが 1 つあります。

```
iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
```

アドレスが 10.0.2.33 のシステムがパケットを 192.168.0.1 に送信する場合に、ルーターは IPsec 暗号化を適用する前にソースを 10.0.2.33 から 172.16.0.1 に変換します。

次に、ソースアドレスが 10.0.2.33 のパケットは **conn myvpn** 設定と一致しなくなるので、IPsec ではこのパケットが暗号化されません。

この問題を解決するには、ルーターのターゲット IPsec サブネット範囲の NAT を除外するルールを挿入します。以下に例を示します。

```
iptables -t nat -I POSTROUTING -s 10.0.2.0/24 -d 192.168.0.0/16 -j RETURN
```

カーネル IPsec サブシステムのバグ

たとえば、バグが原因で IKE ユーザー空間と IPsec カーネルの同期が解除される場合など、カーネル IPsec サブシステムに問題が発生する可能性があります。このような問題がないかを確認するには、以下を実行します。

```
$ cat /proc/net/xfrm_stat
XfrmInError               0
XfrmInBufferError         0
...
```

上記のコマンドの出力でゼロ以外の値が表示されると、問題があることを示しています。この問題が発生した場合は、新しい サポートケース を作成し、1 つ前のコマンドの出力と対応する IKE ログを添付してください。

Libreswan のログ

デフォルトでは、Libreswan は **syslog** プロトコルを使用してログに記録します。**journalctl** コマンドを使用して、IPsec に関連するログエントリーを検索できます。ログへの対応するエントリーは **pluto** IKE デーモンにより送信されるため、以下のように、キーワード pluto を検索します。

```
$ journalctl -b | grep pluto
```

ipsec サービスのライブログを表示するには、次のコマンドを実行します。

```
$ journalctl -f -u ipsec
```

ロギングのデフォルトレベルで設定問題が解決しない場合は、**/etc/ipsec.conf** ファイルの **config setup** セクションに **plutodebug=all** オプションを追加してデバッグログを有効にします。

デバッグロギングは多くのエントリーを生成し、**journald** サービスまたは **syslogd** サービスレートのいずれかが **syslog** メッセージを制限する可能性があることに注意してください。完全なログを取得するには、ロギングをファイルにリダイレクトします。**/etc/ipsec.conf** を編集し、**config setup** セクションに **logfile=/var/log/pluto.log** を追加します。

関連情報

- 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
- **tcpdump(8)** および **ipsec.conf(5)** の man ページ
- firewalld の使用および設定

21.3.15. 関連情報

- **ipsec(8)**、**ipsec.conf(5)**、**ipsec.secrets(5)**、**ipsec_auto(8)**、および **ipsec_rsasigkey(8)** の man ページ
- **/usr/share/doc/libreswan-version/** ディレクトリー

- アップストリームプロジェクトの Web サイト
- The Libreswan プロジェクトの Wiki
- All Libreswan のすべての man ページ
- NIST Special Publication 800-77:Guide to IPsec VPNs

21.4. MACSEC を使用した同じ物理ネットワーク内のレイヤー 2 トラフィックの暗号化

MACsec を使用して、2 つのデバイス間の通信を (ポイントツーポイントで) セキュリティー保護できます。たとえば、ブランチオフィスがメトロイーサネット接続を介してセントラルオフィスに接続されている場合、オフィスを接続する 2 つのホストで MACsec を設定して、セキュリティーを強化できます。

Media Access Control Security (MACsec) は、イーサーネットリンクで異なるトラフィックタイプを保護するレイヤー 2 プロトコルです。これには以下が含まれます。

- DHCP (Dynamic Host Configuration Protocol)
- アドレス解決プロトコル (ARP)
- インターネットプロトコルのバージョン 4 / 6 (**IPv4** / **IPv6**)
- TCP や UDP などの IP 経由のトラフィック

MACsec はデフォルトで、LAN 内のすべてのトラフィックを GCM-AES-128 アルゴリズムで暗号化および認証し、事前共有キーを使用して参加者ホスト間の接続を確立します。共有前の鍵を変更する場合は、MACsec を使用するネットワーク内のすべてのホストで NM 設定を更新する必要があります。

MACsec 接続は、親としてイーサネットネットワークカード、VLAN、トンネルデバイスなどのイーサネットデバイスを使用します。暗号化した接続のみを使用して他のホストと通信するように、MACsec デバイスでのみ IP 設定を指定するか、親デバイスに IP 設定を指定することもできます。後者の場合、親デバイスを使用して、暗号化されていない接続と暗号化された接続用の MACsec デバイスで他のホストと通信できます。

MACsec には特別なハードウェアは必要ありません。たとえば、ホストとスイッチの間のトラフィックのみを暗号化する場合を除き、任意のスイッチを使用できます。このシナリオでは、スイッチが MACsec もサポートする必要があります。

つまり、MACsec を設定する方法は 2 つあります。

- ホスト対ホスト
- 他のホストに切り替えるホスト

重要

MACsec は、同じ (物理または仮想) LAN のホスト間でのみ使用することができます。

21.4.1. nmcli を使用した MACsec 接続の設定

nmcli ツールを使用して、MACsec を使用するようにイーサーネットインターフェイスを設定できます。たとえば、イーサネット経由で接続された 2 つのホスト間に MACsec 接続を作成できます。

手順

1. MACsec を設定する最初のホストで:

   - 事前共有鍵の接続アソシエーション鍵 (CAK) と接続アソシエーション鍵名 (CKN) を作成します。

     a. 16 バイトの 16 進 CAK を作成します。

     ```
     # dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
     50b71a8ef0bd5751ea76de6d6c98c03a
     ```

     b. 32 バイトの 16 進 CKN を作成します。

     ```
     # dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
     f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
     ```

2. 両方のホストで、MACsec 接続を介して接続します。

3. MACsec 接続を作成します。

   ```
   # nmcli connection add type macsec con-name macsec0 ifname macsec0
   connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-
   cak 50b71a8ef0bd5751ea76de6d6c98c03a macsec.mka-ckn
   f2b4297d39da7330910a7abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
   ```

   前の手順で生成された CAK および CKN を **macsec.mka-cak** および **macsec.mka-ckn** パラメーターで使用します。この値は、MACsec で保護されるネットワーク内のすべてのホストで同じである必要があります。

4. MACsec 接続で IP を設定します。

   a. **IPv4** 設定を指定します。たとえば、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、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

   ```
   # nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
   '192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
   ```

   b. **IPv6** 設定を指定しますたとえば、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、および DNS サーバーを **macsec0** 接続に設定するには、以下のコマンドを実行します。

   ```
   # nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
   '2001:db8:1::1/32' ipv6.gateway '2001:db8:1::fffe' ipv6.dns '2001:db8:1::fffd'
   ```

5. 接続をアクティベートします。

   ```
   # nmcli connection up macsec0
   ```

検証

1. トラフィックが暗号化されていることを確認します。

```
# tcpdump -nn -i enp1s0
```

2. オプション: 暗号化されていないトラフィックを表示します。

```
# tcpdump -nn -i macsec0
```

3. MACsec の統計を表示します。

```
# ip macsec show
```

4. integrity-only (encrypt off) および encryption (encrypt on) の各タイプの保護に対して個々のカウンターを表示します。

```
# ip -s macsec show
```

21.4.2. 関連情報

- MACsec: a different solution to encrypt network traffic ブログ

21.5. FIREWALLD の使用および設定

ファイアウォール は、外部からの不要なトラフィックからマシンを保護する方法です。**ファイアウォールルール** セットを定義することで、ホストマシンへの着信ネットワークトラフィックを制御できます。このようなルールは、着信トラフィックを分類して、拒否または許可するために使用されます。

firewalld は、D-Bus インターフェイスを使用して、動的にカスタマイズできるホストベースのファイアウォールを提供するファイアウォールサービスデーモンです。ルールが変更するたびに、ファイアウォールデーモンを再起動しなくても、ルールの作成、変更、および削除を動的に可能にします。

firewalld は、ゾーンおよびサービスの概念を使用して、トラフィック管理を簡素化します。ゾーンは、事前定義したルールセットです。ネットワークインターフェイスおよびソースをゾーンに割り当てることができます。許可されているトラフィックは、コンピューターが接続するネットワークと、このネットワークが割り当てられているセキュリティーレベルに従います。ファイアウォールサービスは、特定のサービスに着信トラフィックを許可するのに必要なすべての設定を扱う事前定義のルールで、ゾーンに適用されます。

サービスは、ネットワーク接続に 1 つ以上のポートまたはアドレスを使用します。ファイアウォールは、ポートに基づいて接続のフィルターを設定します。サービスに対してネットワークトラフィックを許可するには、そのポートを開く必要があります。**firewalld** は、明示的に開いていないポートのトラフィックをすべてブロックします。trusted などのゾーンでは、デフォルトですべてのトラフィックを許可します。

nftables バックエンドを使用した **firewalld** が、**--direct** オプションを使用して、カスタムの **nftables** ルールを **firewalld** に渡すことに対応していないことに注意してください。

21.5.1. firewalld の使用

以下は、サービスやゾーンなどの **firewalld** 機能の概要と、**firewalld** systemd サービスの管理方法を示しています。

21.5.1.1. firewalld、nftables、または iptables を使用する場合

以下は、次のユーティリティーのいずれかを使用する必要があるシナリオの概要です。

- **firewalld**:簡単なファイアウォールのユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。

- **nftables**:**nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。

- **iptables**:Red Hat Enterprise Linux の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

ファイアウォールサービスがそれぞれに影響し合うことを回避するためには、RHEL ホストでそのうちの 1 つだけを実行し、他のサービスを無効にします。

21.5.1.2. ゾーン

firewalld は、インターフェイスに追加する信頼レベルと、そのネットワークのトラフィックに従って、複数のネットワークを複数のゾーンに分類できます。接続は、1 つのゾーンにしか指定できませんが、ゾーンは多くのネットワーク接続に使用できます。

NetworkManager は、**firewalld** にインターフェイスのゾーンを通知します。以下を使用して、ゾーンをインターフェイスに割り当てることができます。

- **NetworkManager**
- **firewall-config** ツール
- **firewall-cmd** コマンドラインツール
- RHEL Web コンソール

後者の 3 つは、適切な **NetworkManager** 設定ファイルの編集のみを行います。Web コンソールを使用してインターフェイスのゾーンを変更する (**firewall-cmd** または **firewall-config**) と、リクエストが **NetworkManager** に転送され、**firewalld** では処理されません。

事前定義したゾーンは **/usr/lib/firewalld/zones/** ディレクトリーに保存され、利用可能なネットワークインターフェイスに即座に適用されます。このファイルは、修正しないと **/etc/firewalld/zones/** ディレクトリーにコピーされません。事前定義したゾーンのデフォルト設定は以下のようになります。

block

IPv4 の場合は icmp-host-prohibited メッセージ、 **IPv6** の場合は icmp6-adm-prohibited メッセージで、すべての着信ネットワーク接続が拒否されます。システムで開始したネットワーク接続のみが可能です。

dmz

公開アクセスは可能ですが、内部ネットワークへのアクセスに制限がある非武装地帯にあるコンピューター向けです。選択した着信接続のみが許可されます。

drop

着信ネットワークパケットは、通知なしで遮断されます。発信ネットワーク接続だけが可能です。

external

マスカレードをルーター用に特別に有効にした外部ネットワークでの使用向けです。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターを信頼しません。選択した着信接続のみが許可されます。

home

そのネットワークでその他のコンピューターをほぼ信頼できる自宅での使用向けです。選択した着信接続のみが許可されます。

internal

そのネットワークでその他のコンピューターをほぼ信頼できる内部ネットワーク向けです。選択した着信接続のみが許可されます。

public

そのネットワークでその他のコンピューターを信頼できないパブリックエリーア向けです。選択した着信接続のみが許可されます。

trusted

すべてのネットワーク接続が許可されます。

work

そのネットワークで、その他のコンピューターをほぼ信頼できる職場での使用向けです。選択した着信接続のみが許可されます。

このゾーンのいずれかを **デフォルト** ゾーンに設定できます。インターフェイス接続を **NetworkManager** に追加すると、デフォルトゾーンに割り当てられます。**firewalld** のデフォルトゾーンは、インストール時に **public** ゾーンに設定されます。デフォルトゾーンは変更できます。

注記

ネットワークゾーン名は、分かりやすく、ユーザーが妥当な決定をすばやく下せるような名前が付けられています。セキュリティー問題を回避するために、ニーズおよびリスク評価に合わせて、デフォルトゾーンの設定の見直しを行ったり、不要なサービスを無効にしてください。

関連情報

- **firewalld.zone(5)** の man ページ

21.5.1.3. 事前定義サービス

サービスが、ローカルポート、プロトコル、ソースポート、宛先、そしてサービスが有効になると自動的に読み込まれるファイアウォールのヘルパーモジュールのリストを指す場合があります。サービスを使用すると、ポートのオープン、プロトコルの定義、パケット転送の有効化などを 1 つ 1 つ行うのではなく、1 回のステップで定義できます。

サービス設定オプションと、一般的なファイル情報は、man ページの **firewalld.service(5)** で説明されています。サービスは、個々の XML 設定ファイルを使用して指定し、名前は、**service-name.xml** のような形式になります。プロトコル名は、**firewalld** のサービス名またはアプリケーション名よりも優先されます。

サービスは、グラフィカルな **firewall-config** ツールと、**firewall-cmd** および **firewall-offline-cmd** を使用して追加または削除できます。

または、**/etc/firewalld/services/** ディレクトリーの XML ファイルを変更できます。ユーザーがサービスを追加または変更しないと、**/etc/firewalld/services/** には、対応する XML ファイルが記載されません。**/usr/lib/firewalld/services/** ディレクトリーのファイルは、サービスを追加または変更する場合にテンプレートとして使用できます。

関連情報

- **firewalld.service(5)** の man ページ

21.5.1.4. firewalld の起動

手順

1. **firewalld** を開始するには、**root** で次のコマンドを実行します。

```
# systemctl unmask firewalld
# systemctl start firewalld
```

2. システムの起動時に **firewalld** を自動的に起動するように設定するには、**root** で次のコマンドを実行します。

```
# systemctl enable firewalld
```

21.5.1.5. firewalld の停止

手順

1. **firewalld** を停止するには、**root** で次のコマンドを実行します。

```
# systemctl stop firewalld
```

2. システムの起動時に **firewalld** を自動的に起動しないように設定するには、次のコマンドを実行します。

```
# systemctl disable firewalld
```

3. **firewalld D-Bus** インターフェイスにアクセスして firewalld を起動していないこと、そしてその他のサービスが **firewalld** を求めているかどうかを確認するには、次のコマンドを実行します。

```
# systemctl mask firewalld
```

21.5.1.6. 永続的な firewalld 設定の確認

firewalld 設定ファイルを手動で編集した後など、特定の状況では、変更が正しいことを管理者が確認します。**firewall-cmd** ユーティリティーを使用して設定を確認することができます。

前提条件

- **firewalld** サービスが実行している。

手順

1. **firewalld** サービスの永続的な設定を確認します。

```
# firewall-cmd --check-config
success
```

永続的な設定が有効になると、コマンドが **success** を返します。その他の場合は、以下のような詳細で、コマンドがエラーを返します。

```
# firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
```

21.5.2. firewalld の現在の状況および設定の表示

firewalld サービスを監視するために、ステータス、許可されたサービス、および設定を表示できます。

21.5.2.1. firewalld の現在の状況の表示

ファイアウォールサービス **firewalld** は、システムにデフォルトでインストールされています。CLI インターフェイス **firewalld** を使用して、サービスが実行していることを確認します。

手順

1. サービスの状況を表示するには、次のコマンドを実行します。

```
# firewall-cmd --state
```

2. サービスの状況の詳細は、**systemctl status** サブコマンドを実行します。

```
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
   Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
     Docs: man:firewalld(1)
 Main PID: 705 (firewalld)
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/firewalld.service
           └─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

21.5.2.2. GUI を使用した許可されたサービスの表示

グラフィカルの **firewall-config** ツールを使用してサービスの一覧を表示する場合は、**Super** キーを押してアクティビティーの概要を開き、**firewall** と入力して **Enter** を押します。**firewall-config** ツールが表示されます。**Services** タブの下にサービスのリストが表示されます。

グラフィカルなファイアウォール設定ツールは、コマンドラインを使用して起動できます。

前提条件

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

手順

- コマンドラインを使用してグラフィカルなファイアウォール設定ツールを起動するには、次のコマンドを実行します。

```
$ firewall-config
```

Firewall Configuration ウィンドウが開きます。このコマンドは通常のユーザーとして実行できますが、監理者パスワードが求められる場合もあります。

21.5.2.3. CLI を使用した firewalld 設定の表示

CLI クライアントで、現在のファイアウォール設定を、複数の方法で表示できます。**--list-all** オプションは、**firewalld** 設定の完全概要を表示します。

firewalld は、ゾーンを使用してトラフィックを管理します。**--zone** オプションでゾーンを指定しないと、コマンドは、アクティブネットワークインターフェイスおよび接続に割り当てたデフォルトゾーンに対して有効になります。

手順

- デフォルトゾーンに関連する情報をすべて表示するには、次のコマンドを実行します。

```
# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

- 設定を表示するゾーンを指定するには、たとえば、**--zone=zone-name** 引数を **firewall-cmd --list-all** コマンドに指定します。

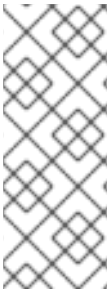
```
# firewall-cmd --list-all --zone=home
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh mdns samba-client dhcpv6-client
  ...
```

- サービス、ポートなど、特定情報の設定を確認するには、特定のオプションを使用します。**firewalld** の man ページか、コマンドの help でオプションのリストを表示します。

```
# firewall-cmd --help
```

- 現在のゾーンで許可されているサービスを表示するには、次のコマンドを実行します。

```
# firewall-cmd --list-services
ssh dhcpv6-client
```



注記

CLI ツールを使用してリスト表示した特定のサブパートの設定は、解釈が難しいことがしばしばあります。たとえば、**firewalld** で **SSH** サービスを許可し、そのサービスに必要なポート (22) を開くことができます。許可されたサービスをリスト表示すると、リストには **SSH** サービスが表示されますが、開いているポートをリスト表示しても、何も表示されません。したがって、**--list-all** オプションを使用して、完全な情報を取得することが推奨されます。

21.5.3. firewalld でネットワークトラフィックの制御

firewalld パッケージは、事前定義された多数のサービスファイルをインストールし、それらをさらに追加したり、カスタマイズしたりできます。さらに、これらのサービス定義を使用して、サービスが使用するプロトコルとポート番号を知らなくても、サービスのポートを開いたり閉じたりできます。

21.5.3.1. 緊急時に CLI を使用してすべてのトラフィックの無効化

システムへの攻撃などの緊急な状態にあるとき、すべてのネットワークトラフィックを無効にし、攻撃を遮断できます。

手順

1. ネットワークトラフィックを直ちに無効にするには、パニックモードをオンにします。

```
# firewall-cmd --panic-on
```

重要

パニックモードを有効にすると、ネットワークトラフィックがすべて停止します。したがって、そのマシンへの物理アクセスがある場合、またはシリアルコンソールを使用してログインする場合に限り使用してください。

2. パニックモードをオフにし、ファイアウォールを永続設定に戻します。パニックモードを無効にするには、次のコマンドを実行します。

```
# firewall-cmd --panic-off
```

検証

- パニックモードを有効または無効にするには、次のコマンドを実行します。

```
# firewall-cmd --query-panic
```

21.5.3.2. CLI を使用して事前定義されたサービスでトラフィックの制御

トラフィックを制御する最も簡単な方法は、事前定義したサービスを **firewalld** に追加する方法です。これにより、必要なすべてのポートが開き、service definition file に従ってその他の設定が変更されます。

手順

1. サービスが許可されていないことを確認します。

```
# firewall-cmd --list-services
ssh dhcpv6-client
```

2. 事前定義したサービスのリストを表示します。

```
# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6
dhcpv6-client dns docker-registry ...
```

3. サービスを、許可されたサービスに追加します。

```
# firewall-cmd --add-service=<service_name>
```

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.3.3. GUI を使用して事前定義サービスでトラフィックを制御

グラフィカルユーザーインターフェイスを使用して、事前定義サービスでネットワークトラフィックを制御できます。

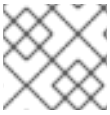
前提条件

- **firewall-config** パッケージがインストールされている

手順

1. 事前定義したサービスまたはカスタマイズしたサービスを有効または無効にするには、以下を行います。
   a. firewall-config ツールを起動して、サービスを設定するネットワークゾーンを選択します。
   b. **Zones** タブを選択してから、下の **Services** タブを選択します。
   c. 信頼するサービスのタイプごとにチェックボックスをオンにするか、チェックボックスをオフにして、選択したゾーンのサービスをブロックします。
2. サービスを編集するには、以下を行います。
   a. firewall-config ツールを起動します。
   b. **Configuration** メニューから **Permanent** を選択します。**Services** ウィンドウの下部に、その他のアイコンおよびメニューボタンが表示されます。
   c. 設定するサービスを選択します。

Ports、**Protocols**、**Source Port** のタブでは、選択したサービスのポート、プロトコル、およびソースポートの追加、変更、ならびに削除が可能です。モジュールタブは、Netfilter ヘルパーモジュールの設定を行います。**Destination** タブは、特定の送信先アドレスとインターネットプロトコル (**IPv4** または **IPv6**) へのトラフィックが制限できます。



注記

Runtime モードでは、サービス設定を変更できません。

21.5.3.4. 新しいサービスの追加

サービスは、グラフィカルな firewall-config ツールと、**firewall-cmd** および **firewall-offline-cmd** を使用して追加または削除できます。または、**/etc/firewalld/services/** にある XML ファイルを編集できます。ユーザーがサービスを追加または変更しないと、対応する XML ファイルが **/etc/firewalld/services/** に作成されません。**/usr/lib/firewalld/services/** のファイルは、サービスを追加または変更する際にテンプレートとして使用できます。

注記

サービス名は英数字にする必要があります。_ (下線) 文字および **-** (ハイフン) 文字も使用できます。

手順

firewalld がアクティブでない場合に、ターミナルで新しいサービスを追加するには、**firewall-cmd** または **firewall-offline-cmd** を使用します。

1. 新しい、空のサービスを追加するには、次のコマンドを実行します。

```
$ firewall-cmd --new-service=<service_name> --permanent
```

2. ローカルファイルを使用して新規サービスを追加するには、次のコマンドを使用します。

```
$ firewall-cmd --new-service-from-file=<service_xml_file> --permanent
```

--name= <service_name> オプションを追加して、サービス名を変更できます。

3. サービス設定を変更すると、直ちにサービスの更新コピーが **/etc/firewalld/services/** に作成できます。
root で次のコマンドを実行して、サービスを手動でコピーします。

```
# cp /usr/lib/firewalld/services/service-name.xml /etc/firewalld/services/service-name.xml
```

firewalld は、最初に **/usr/lib/firewalld/services** のファイルを読み込みます。ファイルは **/etc/firewalld/services** に置かれ、そのファイルが有効な場合は、**/usr/lib/firewalld/services** で一致するファイルを上書きします。**/usr/lib/firewalld/services** で上書きしたファイルは、**/etc/firewalld/services** で一致するファイルが削除されるとすぐに、もしくはサービスのデフォルトを読み込むように **firewalld** が求められた場合に使用されます。これに該当するのは永続環境のみです。ランタイム環境でフォールバックさせるには、再読み込みが必要です。

21.5.3.5. GUI を使用してポートを開く

特定のポートへのファイアウォールを通過するトラフィックを許可する場合は、GUI でポートを開くことができます。

前提条件

- **firewall-config** パッケージがインストールされている

手順

1. **firewall-config** ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側の **Ports** タブを選択し、**Add** ボタンをクリックします。**Port and Protocol** ウィンドウが開きます。
3. 許可するポート番号またはポートの範囲を入力します。
4. リストから **tcp** または **udp** を選択します。

21.5.3.6. GUI を使用してプロトコルを使用したトラフィックの制御

特定のプロトコルを使用してファイアウォールを経由したトラフィックを許可するには、GUI を使用できます。

前提条件

- **firewall-config** パッケージがインストールされている

手順

1. **firewall-config** ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側で **Protocols** タブを選択し、**Add** ボタンをクリックします。**Protocol** ウィンドウが開きます。
3. リストからプロトコルを選択するか、**Other Protocol** チェックボックスを選択し、そのフィールドにプロトコルを入力します。

21.5.3.7. GUI を使用してソースポートを開く

特定ポートからファイアウォールを経由したトラフィックを許可するには、GUI を使用できます。

前提条件

- **firewall-config** パッケージがインストールされている

手順

1. firewall-config ツールを起動し、設定を変更するネットワークゾーンを選択します。
2. 右側の **Source Port** タブを選択し、**Add** ボタンをクリックします。**Source Port** ウィンドウが開きます。
3. 許可するポート番号またはポートの範囲を入力します。リストから **tcp** または **udp** を選択します。

21.5.4. CLI を使用したポートの制御

ポートは、オペレーティングシステムが、ネットワークトラフィックを受信し、区別し、システムサービスに従って転送する論理デバイスです。これは、通常、ポートをリッスンするデーモンにより示されますが、このポートに入るトラフィックを待ちます。

通常、システムサービスは、サービスに予約されている標準ポートでリッスンします。**httpd** デーモン

は、たとえば、ポート 80 をリッスンします。ただし、デフォルトでは、システム管理者は、セキュリティーを強化するため、またはその他の理由により、別のポートをリッスンするようにデーモンを設定します。

21.5.4.1. ポートを開く

開かれたポートを介して、システムが外部からアクセスできます。これはセキュリティーリスクでもあります。一般的に、ポートを閉じたままにし、特定サービスに要求される場合に限り開きます。

手順

現在のゾーンで開かれたポートのリストを表示するには、以下を行います。

1. 許可されているポートのリストを表示します。

```
# firewall-cmd --list-ports
```

2. 許可されているポートにポートを追加して、着信トラフィックに対してそのポートを開きます。

```
# firewall-cmd --add-port=port-number/port-type
```

ポートタイプは、**tcp**、**udp**、**sctp**、または **dccp** になります。このタイプは、ネットワーク接続の種類と一致させる必要があります。

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

ポートタイプは、**tcp**、**udp**、**sctp**、または **dccp** になります。このタイプは、ネットワーク接続の種類と一致させる必要があります。

21.5.4.2. ポートを閉じる

開しているポートが必要なくなった場合に、**firewalld** のポートを閉じます。ポートをそのままにするとセキュリティーリスクとなるため、使用されなくなったらすぐに不要なポートを閉じることが強く推奨されます。

手順

ポートを閉じるには、許可されているポートのリストからそれを削除します。

1. 許可されているポートのリストを表示します。

```
# firewall-cmd --list-ports
```

警告

このコマンドにより、ポートとして開かれているポートのみが表示されます。サービスとして開いているポートは表示されません。したがって、**--list-ports** ではなく **--list-all** オプションの使用を検討してください。

2. 許可されているポートからポートを削除し、着信トラフィックに対して閉じます。

```
# firewall-cmd --remove-port=port-number/port-type
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.5. ファイアウォールゾーンでの作業

ゾーンは、着信トラフィックをより透過的に管理する概念を表しています。ゾーンはネットワークインターフェイスに接続されているか、ソースアドレスの範囲に割り当てられます。各ゾーンは個別にファイアウォールルールを管理しますが、これにより、複雑なファイアウォール設定を定義してトラフィックに割り当てることができます。

21.5.5.1. ゾーンのリスト

コマンドラインを使用してゾーンを一覧表示できます。

手順

1. システムで利用可能なゾーンを確認するには、次のコマンドを実行します。

```
# firewall-cmd --get-zones
```

firewall-cmd --get-zones コマンドは、システムで利用可能な全てのゾーンを表示し、特定ゾーンの詳細は表示しません。

2. すべてのゾーンで詳細情報を表示する場合は、次のコマンドを実行します。

```
# firewall-cmd --list-all-zones
```

3. 特定ゾーンに関する詳細情報を表示する場合は、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --list-all
```

21.5.5.2. 特定ゾーンに対する firewalld 設定の修正

CLI を使用して事前定義されたサービスでトラフィックの制御 および CLI を使用したポートの制御 では、サービスを追加する方法や、現在のワークゾーンの範囲内でポートを変更する方法について説明します。別のゾーンへのルールの設定が必要になる場合もあります。

手順

- 別のゾーンで作業するには、**--zone= <zone_name>** オプションを使用します。たとえば、**public** ゾーンで **SSH** サービスを許可するには、次のようにします。

```
# firewall-cmd --add-service=ssh --zone=public
```

21.5.5.3. デフォルトゾーンの変更

システム管理者は、設定ファイルのネットワークインターフェイスにゾーンを割り当てます。特定のゾーンに割り当てられないインターフェイスは、デフォルトゾーンに割り当てられます。**firewalld** サービスを再起動するたびに、**firewalld** は、デフォルトゾーンの設定を読み込み、それをアクティブにします。

手順

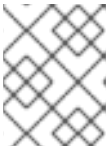
デフォルトゾーンを設定するには、以下を行います。

1. 現在のデフォルトゾーンを表示します。

```
# firewall-cmd --get-default-zone
```

2. 新しいデフォルトゾーンを設定します。

```
# firewall-cmd --set-default-zone <zone_name>
```



注記

この手順では、**--permanent** オプションを使用しなくても、設定は永続化します。

21.5.5.4. ゾーンへのネットワークインターフェイスの割り当て

複数のゾーンに複数のルールセットを定義して、使用されているインターフェイスのゾーンを変更することで、迅速に設定を変更できます。各インターフェイスに特定のゾーンを設定して、そのゾーンを通過するトラフィックを設定できます。

手順

特定インターフェイスにゾーンを割り当てるには、以下を行います。

1. アクティブゾーン、およびそのゾーンに割り当てられているインターフェイスをリスト表示します。

```
# firewall-cmd --get-active-zones
```

2. 別のゾーンにインターフェイスを割り当てます。

```
# firewall-cmd --zone=zone_name --change-interface=interface_name --permanent
```

21.5.5.5. nmcli を使用して接続にゾーンを割り当て

nmcli ユーティリティーを使用して、**firewalld** ゾーンを **NetworkManager** 接続に追加できます。

手順

1. ゾーンを **NetworkManager** 接続プロファイルに割り当てます。

```
# nmcli connection modify profile connection.zone zone_name
```

2. 接続をアクティベートします。

```
# nmcli connection up profile
```

21.5.5.6. ifcfg ファイルでゾーンをネットワーク接続に手動で割り当て

NetworkManager で接続を管理する場合は、NetworkManager が使用するゾーンを認識する必要があります。すべてのネットワーク接続にゾーンを指定できます。これにより、ポータブルデバイスを使用したコンピューターの場所に従って、様々なファイアウォールを柔軟に設定できるようになります。したがって、ゾーンおよび設定には、会社または自宅など、様々な場所を指定できます。

手順

- 接続のゾーンを設定するには、**/etc/sysconfig/network-scripts/ifcfg-connection_name** ファイルを変更して、この接続にゾーンを割り当てる行を追加します。

```
ZONE=zone_name
```

21.5.5.7. 新しいゾーンの作成

カスタムゾーンを使用するには、新しいゾーンを作成したり、事前定義したゾーンなどを使用したりします。新しいゾーンには **--permanent** オプションが必要となり、このオプションがなければコマンドは動作しません。

手順

1. 新しいゾーンを作成します。

```
# firewall-cmd --permanent --new-zone=zone-name
```

2. 作成したゾーンが永続設定に追加されたかどうかを確認します。

```
# firewall-cmd --get-zones
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.5.8. ゾーンの設定ファイル

また、**ゾーンの設定ファイル** を使用してゾーンを作成できます。このアプローチは、新しいゾーンを作成する必要がある場合に、別のゾーンの設定を変更して利用する場合に便利です。

firewalld ゾーン設定ファイルには、ゾーンに対する情報があります。これは、XML ファイル形式で、ゾーンの説明、サービス、ポート、プロトコル、icmp-block、マスカレード、転送ポート、およびリッチ言語ルールです。ファイル名は **zone-name.xml** となります。**zone-name** の長さは 17 文字に制限さ

れます。ゾーンの設定ファイルは、**/usr/lib/firewalld/zones/** ディレクトリーおよび **/etc/firewalld/zones/** ディレクトリーに置かれています。

以下の例は、**TCP** プロトコルまたは **UDP** プロトコルの両方に、1つのサービス (**SSH**) および1つのポート範囲を許可する設定を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>My Zone</short>
  <description>Here you can describe the characteristic features of the zone.</description>
  <service name="ssh"/>
  <port protocol="udp" port="1025-65535"/>
  <port protocol="tcp" port="1025-65535"/>
</zone>
```

そのゾーンの設定を変更するには、セクションを追加または削除して、ポート、転送ポート、サービスなどを追加します。

関連情報

- **firewalld.zone** man ページ

21.5.5.9. 着信トラフィックにデフォルトの動作を設定するゾーンターゲットの使用

すべてのゾーンに対して、特に指定されていない着信トラフィックを処理するデフォルト動作を設定できます。そのような動作は、ゾーンのターゲットを設定することで定義されます。4つのオプションがあります。

- **ACCEPT**:指定したルールで許可されていないパケットを除いた、すべての着信パケットを許可します。
- **REJECT**:指定したルールで許可されているパケット以外の着信パケットをすべて拒否します。**firewalld** がパケットを拒否すると、送信元マシンに拒否について通知されます。
- **DROP**:指定したルールで許可されているパケット以外の着信パケットをすべて破棄します。**firewalld** がパケットを破棄すると、ソースマシンにパケット破棄の通知がされません。
- **default**:**REJECT** と似ていますが、特定のシナリオで特別な意味を持ちます。詳細は、**firewall-cmd(1)** man ページの **適応およびクエリーゾーンとポリシーのオプション** セクションを参照してください。

手順

ゾーンにターゲットを設定するには、以下を行います。

1. 特定ゾーンに対する情報をリスト表示して、デフォルトゾーンを確認します。

```
# firewall-cmd --zone=zone-name --list-all
```

2. ゾーンに新しいターゲットを設定します。

```
# firewall-cmd --permanent --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

関連情報

- **firewall-cmd(1)** man ページ

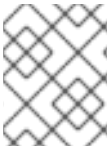
21.5.6. ゾーンを使用し、ソースに応じた着信トラフィックの管理

ゾーンを使用して、そのソースに基づいて着信トラフィックを管理するゾーンを使用できます。これにより、着信トラフィックを分類し、複数のゾーンに向け、トラフィックにより到達できるサービスを許可または拒否できます。

ソースをゾーンに追加する場合は、ゾーンがアクティブになり、そのソースからの着信トラフィックは、それを介して行われます。各ゾーンに異なる設定を指定できますが、それは指定したソースから順次トラフィックに適用されます。ネットワークインターフェイスが 1 つしかない場合でも、複数のゾーンを使用できます。

21.5.6.1. ソースの追加

着信トラフィックを特定のゾーンに転送する場合は、そのゾーンにソースを追加します。ソースは、CIDR (Classless Inter-domain Routing) 表記法の IP アドレスまたは IP マスクになります。



注記

ネットワーク範囲が重複している複数のゾーンを追加する場合は、ゾーン名で順序付けされ、最初のゾーンのみが考慮されます。

- 現在のゾーンにソースを設定するには、次のコマンドを実行します。

```
# firewall-cmd --add-source=<source>
```

- 特定ゾーンのソース IP アドレスを設定するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-source=<source>
```

以下の手順は、**信頼される** ゾーンで 192.168.2.15 からのすべての着信トラフィックを許可します。

手順

1. 利用可能なゾーンの一覧を表示します。

```
# firewall-cmd --get-zones
```

2. 永続化モードで、信頼ゾーンにソース IP を追加します。

```
# firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.6.2. ソースの削除

ゾーンからソースを削除すると、そのゾーンからのトラフィックを遮断します。

手順

1. 必要なゾーンに対して許可されているソースのリストを表示します。

```
# firewall-cmd --zone=zone-name --list-sources
```

2. ゾーンからソースを永続的に削除します。

```
# firewall-cmd --zone=zone-name --remove-source=<source>
```

3. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.6.3. ソースポートの追加

発信源となるポートに基づいたトラフィックの分類を有効にするには、**--add-source-port** オプションを使用してソースポートを指定します。**--add-source** オプションと組み合わせて、トラフィックを特定の IP アドレスまたは IP 範囲に制限できます。

手順

- ソースポートを追加するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --add-source-port=<port-name>/<tcp|udp|sctp|dccp>
```

21.5.6.4. ソースポートの削除

ソースポートを削除して、送信元ポートに基づいてトラフィックの分類を無効にします。

手順

- ソースポートを削除するには、次のコマンドを実行します。

```
# firewall-cmd --zone=zone-name --remove-source-port=<port-
name>/<tcp|udp|sctp|dccp>
```

21.5.6.5. ゾーンおよびソースを使用して特定ドメインのみに対してサービスの許可

特定のネットワークからのトラフィックを許可して、マシンのサービスを使用するには、ゾーンおよびソースを使用します。以下の手順では、他のトラフィックがブロックされている間に **192.0.2.0/24** ネットワークからの HTTP トラフィックのみを許可します。

警告

このシナリオを設定する場合は、**default** のターゲットを持つゾーンを使用します。**192.0.2.0/24** からのトラフィックではネットワーク接続がすべて許可されるため、ターゲットが **ACCEPT** に設定されたゾーンを使用することは、セキュリティー上のリスクになります。

手順

1. 利用可能なゾーンのリストを表示します。

```
# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

2. IP 範囲を **internal** ゾーンに追加し、ソースから発信されるトラフィックをゾーン経由でルーティングします。

```
# firewall-cmd --zone=internal --add-source=192.0.2.0/24
```

3. **http** サービスを **internal** ゾーンに追加します。

```
# firewall-cmd --zone=internal --add-service=http
```

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

検証

- **internal** ゾーンがアクティブで、サービスが許可されていることを確認します。

```
# firewall-cmd --zone=internal --list-all
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.0.2.0/24
  services: cockpit dhcpv6-client mdns samba-client ssh http
  ...
```

関連情報

- **firewalld.zones(5)** の man ページ

21.5.7. ゾーン間で転送されるトラフィックのフィルタリング

ポリシーオブジェクトを使用すると、ユーザーはポリシーで同様のパーミッションを必要とする異なるアイデンティティーをグループ化できます。トラフィックの方向に応じてポリシーを適用できます。

ポリシーオブジェクト policy objects 機能は、firewalld で正引きフィルターと出力フィルターを提供します。firewalld を使用して、異なるゾーン間のトラフィックをフィルタリングし、ローカルでホストされている仮想マシンへのアクセスを許可して、ホストを接続できます。

21.5.7.1. ポリシーオブジェクトとゾーンの関係

ポリシーオブジェクトを使用すると、サービス、ポート、リッチルールなどの firewalld のプリミティブをポリシーに割り当てることができます。ポリシーオブジェクトは、ステートフルおよび一方向の方法でゾーン間を通過するトラフィックに適用することができます。

```
# firewall-cmd --permanent --new-policy myOutputPolicy
```

```
# firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST

# firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

HOST および **ANY** は、入出力ゾーンリストで使用されるシンボリックゾーンです。

- **HOST** シンボリックゾーンは、firewalld を実行しているホストから発信されるトラフィック、またはホストへの宛先を持つトラフィックのポリシーを許可します。
- **ANY** シンボリックゾーンは、現行および将来のすべてのゾーンにポリシーを適用します。**ANY** シンボリックゾーンは、すべてのゾーンのワイルドカードとして機能します。

21.5.7.2. 優先度を使用したポリシーのソート

同じトラフィックセットに複数のポリシーを適用できるため、優先度を使用して、適用される可能性のあるポリシーの優先順位を作成する必要があります。

ポリシーをソートする優先度を設定するには、次のコマンドを実行します。

```
# firewall-cmd --permanent --policy mypolicy --set-priority -500
```

この例では、**-500** の優先度は低くなりますが、優先度は高くなります。したがって、-500 は、-100 より前に実行されます。優先度の高い値は、低い値よりも優先されます。

ポリシーの優先度には、以下のルールが適用されます。

- 負の優先度を持つポリシーは、ゾーンのルールの前に適用されます。
- 正の優先度を持つポリシーは、ゾーンのルールの後に適用されます。
- 優先度 0 は予約されているため、使用できません。

21.5.7.3. ポリシーオブジェクトを使用した、ローカルでホストされているコンテナーと、ホストに物理的に接続されているネットワークとの間でのトラフィックのフィルタリング

ポリシーオブジェクト機能を使用すると、コンテナーと仮想マシンのトラフィックをフィルターにかけることができます。

手順

1. 新しいポリシーを作成します。

```
# firewall-cmd --permanent --new-policy podmanToHost
```

2. すべてのトラフィックをブロックします。

```
# firewall-cmd --permanent --policy podmanToHost --set-target REJECT

# firewall-cmd --permanent --policy podmanToHost --add-service dhcp

# firewall-cmd --permanent --policy podmanToHost --add-service dns
```

注記

Red Hat では、デフォルトでホストへのすべてのトラフィックをブロックしてから、ホストに必要なサービスを選択的に開くことを推奨しています。

3. ポリシーで使用する入力ゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

4. ポリシーで使用する出力ゾーンを定義します。

```
# firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

検証

- ポリシーに関する情報を確認します。

```
# firewall-cmd --info-policy podmanToHost
```

21.5.7.4. ポリシーオブジェクトのデフォルトターゲットの設定

ポリシーには --set-target オプションを指定できます。以下のターゲットを使用できます。

- **ACCEPT** - パケットを受け入れます
- **DROP** - 不要なパケットを破棄します
- **REJECT** - ICMP 応答で不要なパケットを拒否します
- **CONTINUE** (デフォルト) - パケットは、次のポリシーとゾーンのルールに従います。

```
# firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

検証

- ポリシーに関する情報の確認

```
# firewall-cmd --info-policy mypolicy
```

21.5.8. firewalld を使用した NAT の設定

firewalld では、以下のネットワークアドレス変換 (NAT) タイプを設定できます。

- マスカレーディング
- ソース NAT (SNAT)
- 宛先 NAT (DNAT)
- リダイレクト

21.5.8.1. NAT タイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダーは、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードと SNAT は互いに非常に似ています。相違点は次のとおりです。

- マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。
- SNAT は、パケットのソース IP アドレスを指定された IP に設定し、出力インターフェイスの IP アドレスを動的に検索しません。そのため、SNAT の方がマスカレードよりも高速です。出力インターフェイスが固定 IP アドレスを使用する場合は、SNAT を使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、チェーンフックに応じてパケットをローカルマシンにリダイレクトする DNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

21.5.8.2. IP アドレスのマスカレードの設定

システムで IP マスカレードを有効にできます。IP マスカレードは、インターネットにアクセスする際にゲートウェイの向こう側にある個々のマシンを隠します。

手順

1. **external** ゾーンなどで IP マスカレーディングが有効かどうかを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --query-masquerade
```

このコマンドでは、有効な場合は **yes** と出力され、終了ステータスは **0** になります。無効の場合は **no** と出力され、終了ステータスは **1** になります。**zone** を省略すると、デフォルトのゾーンが使用されます。

2. IP マスカレードを有効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --add-masquerade
```

3. この設定を永続化するには、**--permanent** オプションをコマンドに渡します。

4. IP マスカレードを無効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --zone=external --remove-masquerade
```

この設定を永続化するには、**--permanent** をコマンドラインに渡します。

21.5.9. DNAT を使用して HTTPS トラフィックを別のホストに転送する

Web サーバーがプライベート IP アドレスを持つ DMZ で実行されている場合は、宛先ネットワークアドレス変換 (DNAT) を設定して、インターネット上のクライアントがこの Web サーバーに接続できるようにすることができます。この場合、Web サーバーのホスト名はルーターのパブリック IP アドレスに解決されます。クライアントがルーターの定義済みポートへの接続を確立すると、ルーターはパケットを内部 Web サーバーに転送します。

前提条件

- DNS サーバーが、Web サーバーのホスト名をルーターの IP アドレスに解決している。
- 次の設定を把握している。
  - 転送するプライベート IP アドレスおよびポート番号
  - 使用する IP プロトコル
  - パケットをリダイレクトする Web サーバーの宛先 IP アドレスおよびポート

手順

1. ファイアウォールポリシーを作成します。

```
# firewall-cmd --permanent --new-policy ExamplePolicy
```

   ポリシーは、ゾーンとは対照的に、入力、出力、および転送されるトラフィックのパケットフィルタリングを許可します。ローカルで実行されている Web サーバー、コンテナー、または仮想マシン上のエンドポイントにトラフィックを転送するには、このような機能が必要になるため、これは重要です。

2. 受信トラフィックと送信トラフィックのシンボリックゾーンを設定して、ルーター自体がローカル IP アドレスに接続し、このトラフィックを転送できるようにします。

```
# firewall-cmd --permanent --policy=ExamplePolicy --add-ingress-zone=HOST
# firewall-cmd --permanent --policy=ExamplePolicy --add-egress-zone=ANY
```

   --add-ingress-zone=HOST オプションは、ローカルで生成され、ローカルホストから送信されるパケットを参照します。**--add-egress-zone=ANY** オプションは、任意のゾーン宛てのトラフィックを参照します。

3. トラフィックを Web サーバーに転送するリッチルールを追加します。

```
# firewall-cmd --permanent --policy=ExamplePolicy --add-rich-rule='rule family="ipv4"
destination address="192.0.2.1" forward-port port="443" protocol="tcp" to-port="443"
to-addr="192.51.100.20"'
```

   リッチルールは、ルーターの IP アドレス 192.0.2.1 のポート 443 から Web サーバーの IP 192.51.100.20 のポート 443 に TCP トラフィックを転送します。ルールは **ExamplePolicy** を使用して、ルーターがローカル IP アドレスにも接続できるようにします。

4. ファイアウォール設定ファイルをリロードします。

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

5. カーネルで 127.0.0.0/8 のルーティングを有効にします。

```
# echo "net.ipv4.conf.all.route_localnet=1" > /etc/sysctl.d/90-enable-route-localnet.conf
# sysctl -p /etc/sysctl.d/90-enable-route-localnet.conf
```

検証

1. Web サーバーに転送したルーターの IP アドレスおよびポートに接続します。

```
# curl https://192.0.2.1:443
```

2. オプション: **net.ipv4.conf.all.route_localnet** がアクティブであることを確認します。

```
# sysctl net.ipv4.conf.all.route_localnet
net.ipv4.conf.all.route_localnet = 1
```

3. **ExamplePolicy** がアクティブで、必要な設定が含まれていることを確認します。特に、送信元 IP アドレスとポート、使用するプロトコル、および宛先 IP アドレスとポート:

```
# firewall-cmd --info-policy=ExamplePolicy
ExamplePolicy (active)
  priority: -1
  target: CONTINUE
  ingress-zones: HOST
  egress-zones: ANY
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
 rule family="ipv4" destination address="192.0.2.1" forward-port port="443" protocol="tcp" to-port="443" to-addr="192.51.100.20"
```

関連情報

- **firewall-cmd(1)**、**firewalld.policies(5)**、**firewalld.richlanguage(5)**、**sysctl(8)**、および **sysctl.conf(5)** の man ページ
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

21.5.10. ICMP リクエストの管理

Internet Control Message Protocol (**ICMP**) は、接続問題 (要求されているサービスが利用できないなど) を示すエラーメッセージと運用情報を送信するために、様々なネットワークデバイスにより使用されているサポート対象のプロトコルです。**ICMP** は、システム間でデータを交換するのに使用されていないため、TCP、UDP などの転送プロトコルとは異なります。

ただし、**ICMP** メッセージ (特に **echo-request** および **echo-reply**) を利用して、ネットワークに関する情報を明らかにし、その情報をさまざまな不正行為に悪用することが可能です。したがって、**firewalld** は、ネットワーク情報を保護するため、**ICMP** リクエストをブロックできます。

21.5.10.1. ICMP リクエストのリスト表示およびブロック

ICMP リクエストのリスト表示

ICMP リクエストは、**/usr/lib/firewalld/icmptypes/** ディレクトリーにある各 XML ファイルで説明されています。リクエストの説明は、このファイルを参照してください。**firewall-cmd** コマンドは、**ICMP** リクエストの操作を制御します。

- 利用可能な **ICMP** タイプのリストを表示するには、次のコマンドを実行します。

```
# firewall-cmd --get-icmptypes
```

- **ICMP** リクエストは、IPv4、IPv6、またはその両方のプロトコルで使用できます。**ICMP** リクエストが使用されているプロトコルを表示するには、次のコマンドを実行します。

```
# firewall-cmd --info-icmptype=<icmptype>
```

- **ICMP** リクエストのステータスは、リクエストが現在ブロックされている場合は **yes**、ブロックされていない場合は **no** となります。**ICMP** リクエストが現在ブロックされているかどうかを確認するには、次のコマンドを実行します。

```
# firewall-cmd --query-icmp-block=<icmptype>
```

ICMP リクエストのブロックまたはブロックの解除

サーバーが **ICMP** リクエストをブロックした場合は、通常の情報が提供されません。ただし、情報が全く提供されないというわけではありません。クライアントは、特定の **ICMP** リクエストがブロックされている (拒否されている) 情報を受け取ります。**ICMP** リクエストは、特に IPv6 トラフィックを使用すると、接続問題が発生することがあるため、注意深く検討する必要があります。

- **ICMP** リクエストが現在ブロックされているかどうかを確認するには、次のコマンドを実行します。

```
# firewall-cmd --query-icmp-block=<icmptype>
```

- **ICMP** リクエストをブロックするには、次のコマンドを実行します。

```
# firewall-cmd --add-icmp-block=<icmptype>
```

- **ICMP** リクエストのブロックを削除するには、次のコマンドを実行します。

```
# firewall-cmd --remove-icmp-block=<icmptype>
```

情報をまったく指定せずに ICMP リクエストのブロック

通常、**ICMP** リクエストをブロックすると、ブロックしていることをクライアントは認識します。したがって、ライブの IP アドレスを傍受している潜在的な攻撃者は、IP アドレスがオンラインであることを確認できます。この情報を完全に非表示にするには、**ICMP** リクエストをすべて破棄する必要があります。

- すべての **ICMP** リクエストをブロックして破棄するには、次のコマンドを実行します。
- ゾーンのターゲットを **DROP** に設定します。

```
# firewall-cmd --permanent --set-target=DROP
```

これで、明示的に許可されるトラフィックを除き、**ICMP** リクエストを含むすべてのトラフィックが破棄されます。

特定の **ICMP** リクエストをブロックして破棄し、その他のリクエストを許可するには、以下を行います。

1. ゾーンのターゲットを **DROP** に設定します。

```
# firewall-cmd --permanent --set-target=DROP
```

2. すべての **ICMP** リクエストを一度にブロックする、ICMP ブロックの反転を追加します。

```
# firewall-cmd --add-icmp-block-inversion
```

3. 許可する **ICMP** リクエストに ICMP ブロックを追加する場合は、次のコマンドを実行します。

```
# firewall-cmd --add-icmp-block=<icmptype>
```

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

ブロックの反転 は、**ICMP** リクエストブロックの設定を反転します。そのため、ゾーンのターゲットが **DROP** に変更されたため、ブロックされていないリクエストはすべてブロックされます。ブロックされているリクエストはブロックされません。これは、リクエストのブロックを解除する場合は、ブロックコマンドを使用する必要があることを示しています。

ブロックの反転を、完全許可の設定に戻すには、以下を行います。

1. ゾーンのターゲットを **default** または **ACCEPT** に戻すには、次のコマンドを設定します。

```
# firewall-cmd --permanent --set-target=default
```

2. **ICMP** リクエストに追加したすべてのブロックを削除します。

```
# firewall-cmd --remove-icmp-block=<icmptype>
```

3. **ICMP** ブロックの反転を削除します。

```
# firewall-cmd --remove-icmp-block-inversion
```

4. 新しい設定を永続化します。

```
# firewall-cmd --runtime-to-permanent
```

21.5.10.2. GUI を使用した ICMP フィルターの設定

- **ICMP** フィルターを有効または無効にするには、**firewall-config** ツールを起動して、フィルターをかけるメッセージのネットワークゾーンを選択します。**ICMP フィルター** タブを選択し、フィルターをかける **ICMP** メッセージの各タイプのチェックボックスを選択します。フィルターを無効にするには、チェックボックスの選択を外します。これは方向ごとに設定され、デフォルトではすべてが許可されます。

- **ICMP フィルター** の反転を有効にするには、右側の **フィルターの反転** チェックボックスをクリックします。マークがついた **ICMP** タイプだけが許可され、その他はすべて拒否されます。DROP ターゲットを使用するゾーンでは破棄されます。

21.5.11. firewalld を使用した IP セットの設定および制御

firewalld で対応する IP セットタイプのリストを表示するには、root で次のコマンドを実行します。

```
# firewall-cmd --get-ipset-types
hash:ip hash:ip,mark hash:ip,port hash:ip,port,ip hash:ip,port,net hash:mac hash:net hash:net,iface
hash:net,net hash:net,port hash:net,port,net
```

警告

Red Hat は、**firewalld** を介して管理していない IP セットを使用することは推奨しません。このような IP セットを使用すると、そのセットを参照する永続的なダイレクトルールが必要で、IP セットを作成するカスタムサービスを追加する必要があります。このサービスは、**firewalld** を起動する前に起動する必要があります。先に起動しておかないと、**firewalld** が、このセットを使用してダイレクトルールを追加できません。**/etc/firewalld/direct.xml** ファイルを使用して、永続的なダイレクトルールを追加できます。

21.5.11.1. CLI を使用した IP セットオプションの設定

IP セットは、**firewalld** ゾーンでソースとして使用でき、リッチルールでソースとして使用できます。Red Hat Enterprise Linux で推奨される方法は、ダイレクトルールで **firewalld** を使用して作成した IP セットを使用する方法です。

- 永続的な環境で **firewalld** に認識されている IP セットのリストを表示するには、次のコマンドを **root** で実行します。

  ```
  # firewall-cmd --permanent --get-ipsets
  ```

- 新しい IP セットを追加するには、永続化環境を使用し、**root** で次のコマンドを実行します。

  ```
  # firewall-cmd --permanent --new-ipset=test --type=hash:net
  success
  ```

  上記のコマンドは、名前 **test** とタイプ **hash:net** で、**IPv4** の新しい IP セットを作成します。**IPv6** で使用する IP セットを作成する場合は、**--option=family=inet6** オプションを追加します。ランタイム環境で新しい設定を有効にするには、**firewalld** を再読み込みします。

- **root** で次のコマンドを実行して、新しい IP セットのリストを表示します。

```
# firewall-cmd --permanent --get-ipsets
test
```

- IP セットの詳細は、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --info-ipset=test
test
type: hash:net
options:
entries:
```

この時点では IP セットにエントリーがありません。

- IP セット **test** にエントリーを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --add-entry=192.168.0.1
success
```

上記のコマンドは、IP アドレス **192.168.0.1** を IP セットに追加します。

- IP セットの現在のエントリーをリスト表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

- IP アドレスのリストを含む **iplist.txt** ファイルを作成します。次に例を示します。

```
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

IP セットの IP アドレスのリストが含まれるファイルには、行ごとにエントリーが含まれている必要があります。ハッシュ、セミコロン、また空の行から始まる行は無視されます。

- **iplist.txt** ファイルからアドレスを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --add-entries-from-file=iplist.txt
success
```

- 拡張された IP セットのエントリーリストを表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
```

- IP セットからアドレスを削除し、更新したエントリーリストを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --permanent --ipset=pass:_test_ --remove-entries-from-file=iplist.txt
success
```

```
# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

- IP セットをゾーンへのソースとして追加し、ゾーンを使用して、IP セットに記載されるアドレスから受信するすべてのトラフィックを処理します。たとえば、IP セットの **test** をソースとして **drop** ゾーンに追加し、IP セットの **test** のリストに表示されるすべてのエントリーから発信されるパケットをすべて破棄するには、**root** で次のコマンドを実行します。

  ```
  # firewall-cmd --permanent --zone=drop --add-source=ipset:test
  success
  ```

  ソースの **ipset:** 接頭辞は、ソースが IP セットで、IP アドレスまたはアドレス範囲ではない **firewalld** を示しています。

IP セットの作成および削除は、永続環境に限定されますが、その他の IP セットオプションは、**--permanent** オプションを使用しないランタイム環境で使用できます。

21.5.12. リッチルールの優先度設定

デフォルトでは、リッチルールはルールアクションに基づいて設定されます。たとえば、**許可** ルールよりも **拒否** ルールが優先されます。リッチルールで **priority** パラメーターを使用すると、管理者はリッチルールとその実行順序をきめ細かく制御できます。

21.5.12.1. priority パラメーターを異なるチェーンにルールを整理する方法

リッチルールの **priority** パラメーターは、**-32768** ~ **32767** の任意の数値に設定でき、値が小さい方が優先されます。

firewalld サービスは、優先度の値に基づいて、ルールを異なるチェーンに整理します。

- 優先度が 0 未満 - ルールは **_pre** 接尾辞が付いたチェーンにリダイレクトされます。
- 優先度が 0 を超える - ルールは **_post** 接尾辞が付いたチェーンにリダイレクトされます。
- 優先度が 0 - アクションに基づいて、ルールは、**_log**、**_deny**、または **_allow** のアクションを使用してチェーンにリダイレクトされます。

このサブチェーンでは、**firewalld** は優先度の値に基づいてルールを分類します。

21.5.12.2. リッチルールの優先度の設定

以下は、**priority** パラメーターを使用して、他のルールで許可または拒否されていないすべてのトラフィックをログに記録するリッチルールを作成する方法を示しています。このルールを使用して、予期しないトラフィックにフラグを付けることができます。

手順

- 優先度が非常に低いルールを追加して、他のルールと一致していないすべてのトラフィックをログに記録します。

  ```
  # firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"'
  ```

  このコマンドでは、ログエントリーの数を、毎分 **5** に制限します。

検証

- 前の手順のコマンドで作成した **nftables** ルールを表示します。

```
# nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
  chain filter_IN_public_post {
    log prefix "UNEXPECTED: " limit rate 5/minute
  }
}
```

21.5.13. ファイアウォールロックダウンの設定

ローカルのアプリケーションやサービスは、**root** で実行していれば、ファイアウォール設定を変更できます (たとえば **libvirt**)。管理者は、この機能を使用してファイアウォール設定をロックし、すべてのアプリケーションでファイアウォール変更を要求できなくするか、ロックダウンの許可リストに追加されたアプリケーションのみがファイアウォール変更を要求できるようにすることが可能になります。ロックダウン設定はデフォルトで無効になっています。これを有効にすると、ローカルのアプリケーションやサービスによるファイアウォールへの望ましくない設定変更を確実に防ぐことができます。

21.5.13.1. CLI を使用したロックダウンの設定

コマンドラインでロックダウン機能を有効または無効にすることができます。

手順

1. ロックダウンが有効になっているかどうかを確認するには、**root** で次のコマンドを使用します。

```
# firewall-cmd --query-lockdown
```

   ロックダウンが有効な場合は、**yes** と出力され、終了ステータスは **0** になります。無効の場合は **no** と出力され、終了ステータスは **1** になります。

2. ロックダウンを有効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --lockdown-on
```

3. ロックダウンを無効にするには、**root** で次のコマンドを実行します。

```
# firewall-cmd --lockdown-off
```

21.5.13.2. CLI を使用したロックダウン許可リストオプションの設定

ロックダウンの許可リストには、コマンド、セキュリティーのコンテキスト、ユーザー、およびユーザー ID を追加できます。許可リストのコマンドエントリーがアスタリスク*で終了している場合は、そのコマンドで始まるすべてのコマンドラインが一致することになります。*がなければ、コマンドと引数が完全に一致する必要があります。

- ここでのコンテキストは、実行中のアプリケーションやサービスのセキュリティー (SELinux) コンテキストです。実行中のアプリケーションのコンテキストを確認するには、次のコマンドを実行します。

```
$ ps -e --context
```

このコマンドは、実行中のアプリケーションをすべて返します。grep ツールを使用して、出力から目的のアプリケーションをパイプ処理します。以下に例を示します。

```
$ ps -e --context | grep example_program
```

- 許可リストにあるコマンドラインの一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-commands
```

- 許可リストに command コマンドを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

- 許可リストから command コマンドを削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

- command コマンドが許可リストに含まれるかどうかを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --query-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

このコマンドでは、含まれる場合は **yes** が出力され、終了ステータスは **0** になります。無効の場合は **no** と出力され、終了ステータスは **1** になります。

- 許可リストにあるセキュリティーコンテキストの一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-contexts
```

- 許可リストに context コンテキストを追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-context=context
```

- 許可リストから context コンテキストを削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-context=context
```

- context コンテキストが許可リストに含まれるかどうかを確認するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --query-lockdown-whitelist-context=context
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

- 許可リストにあるユーザー ID すべての一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-uids
```

- 許可リストにユーザー ID (**uid**) を追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-uid=uid
```

- 許可リストからユーザー ID (**uid**) を削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-uid=uid
```

- 許可リストにユーザー ID (**uid**) があるかどうかを確認するには、次のコマンドを実行します。

```
$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

- 許可リストにある全ユーザー名の一覧を表示するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --list-lockdown-whitelist-users
```

- 許可リストにユーザー名 (**user**) を追加するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --add-lockdown-whitelist-user=user
```

- 許可リストからユーザー名 (**user**) を削除するには、**root** で次のコマンドを実行します。

```
# firewall-cmd --remove-lockdown-whitelist-user=user
```

- ユーザー名 (**user**) が許可リストに含まれるかどうかを確認するには、次のコマンドを実行します。

```
$ firewall-cmd --query-lockdown-whitelist-user=user
```

含まれる場合は、**yes** と出力され、終了ステータスは **0** になります。含まれない場合は、**no** が出力され、終了ステータスは **1** になります。

21.5.13.3. 設定ファイルを使用したロックダウンの許可リストオプションの設定

デフォルトの許可リスト設定ファイルには、**NetworkManager** コンテキストと、**libvirt** のデフォルトコンテキストが含まれます。リストには、ユーザー ID (0) もあります。

+ 許可リスト設定ファイルは **/etc/firewalld/** ディレクトリーに保存されます。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

以下の許可リスト設定ファイルの例では、**firewall-cmd** ユーティリティーのコマンドと、ユーザー ID が **815** である **user** のコマンドをすべて有効にしています。

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <command name="/usr/libexec/platform-python -s /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

この例では、**user id** と **user name** の両方が使用されていますが、実際にはどちらか一方のオプションだけが必要です。Python はインタープリターとしてコマンドラインに追加されています。または、以下のような明確なコマンドも使用できます。

```
# /usr/bin/python3 /bin/firewall-cmd --lockdown-on
```

この例では、**--lockdown-on** コマンドだけが許可されます。

Red Hat Enterprise Linux では、すべてのユーティリティーが **/usr/bin/** ディレクトリーに格納されており、**/bin/** ディレクトリーは **/usr/bin/** ディレクトリーへのシンボリックリンクとなります。つまり、**root** で **firewall-cmd** のパスを実行すると **/bin/firewall-cmd** に対して解決しますが、**/usr/bin/firewall-cmd** が使用できるようになっています。新たなスクリプトは、すべて新しい格納場所を使用する必要があります。ただし、**root** で実行するスクリプトが **/bin/firewall-cmd** へのパスを使用するようになっているのであれば、これまでは **root** 以外のユーザーにのみ使用されていた **/usr/bin/firewall-cmd** パスに加え、このコマンドのパスも許可リストに追加する必要があります。

コマンドの名前属性の最後にある * は、その名前で始まるすべてのコマンドが一致することを意味します。*がなければ、コマンドと引数が完全に一致する必要があります。

21.5.14. firewalld ゾーン内の異なるインターフェイスまたはソース間でのトラフィック転送の有効化

ゾーン内転送は、**firewalld** ゾーン内のインターフェイスまたはソース間のトラフィック転送を可能にする **firewalld** 機能です。

21.5.14.1. ゾーン内転送と、デフォルトのターゲットが ACCEPT に設定されているゾーンの違い

ゾーン内転送を有効にすると、1 つの **firewalld** ゾーン内のトラフィックは、あるインターフェイスまたはソースから別のインターフェイスまたはソースに流れることができます。ゾーンは、インターフェイスおよびソースの信頼レベルを指定します。信頼レベルが同じである場合、インターフェイスまたはソース間の通信が可能です。

firewalld のデフォルトゾーンでゾーン内転送を有効にすると、現在のデフォルトゾーンに追加されたインターフェイスおよびソースにのみ適用されることに注意してください。

firewalld の **trusted** ゾーンは、**ACCEPT** に設定されたデフォルトのターゲットを使用します。このゾーンは、転送されたすべてのトラフィックを受け入れ、ゾーン内転送は適用されません。

他のデフォルトのターゲット値の場合、転送されたトラフィックはデフォルトでドロップされます。これは、信頼済みゾーンを除くすべての標準ゾーンに適用されます。

21.5.14.2. ゾーン内転送を使用したイーサネットと Wi-Fi ネットワーク間でのトラフィックの転送

ゾーン内転送を使用して、同じ **firewalld** ゾーン内のインターフェイスとソース間のトラフィックを転送することができます。たとえば、この機能を使用して、**enp1s0** に接続されたイーサネットネットワークと、**wlp0s20** に接続された Wi-Fi ネットワーク間のトラフィックを転送するには、この機能を使用します。

手順

1. カーネルでパケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2. ゾーン内転送を有効にするインターフェイスが、**internal** ゾーンと異なるゾーンに割り当てられていないことを確認してください。

```
# firewall-cmd --get-active-zones
```

3. 現在、インターフェイスが **internal** 以外のゾーンに割り当てられている場合は、以下のように再割り当てします。

```
# firewall-cmd --zone=internal --change-interface=interface_name --permanent
```

4. **enp1s0** および **wlp0s20** インターフェイスを **internal** ゾーンに追加します。

```
# firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5. ゾーン内転送を有効にします。

```
# firewall-cmd --zone=internal --add-forward
```

検証

以下の検証手順では、**nmap-ncat** パッケージが両方のホストにインストールされている必要があります。

1. ゾーン転送を有効にしたホストの **enp1s0** インターフェイスと同じネットワーク内にあるホストにログインします。

2. **ncat** で echo サービスを起動し、接続をテストします。

```
# ncat -e /usr/bin/cat -l 12345
```

3. **wlp0s20** インターフェイスと同じネットワークにあるホストにログインします。

4. **enp1s0** と同じネットワークにあるホスト上で実行している echo サーバーに接続します。

```
# ncat <other_host> 12345
```

5. 試しに何かを入力して **Enter** キーを押し、テキストが返送されることを確認します。

関連情報

- **firewalld.zones(5)** の man ページ

21.5.15. システムロールを使用した firewalld の設定

firewall システムロールを使用すると、一度に複数のクライアントに **firewalld** サービスを設定できます。この解決策は以下のとおりです。

- 入力設定が効率的なインターフェイスを提供する。
- 目的の **firewalld** パラメーターを 1 か所で保持する。

コントロールノードで **firewall** ロールを実行すると、システムロールは **firewalld** パラメーターをマネージドノードに即座に適用し、再起動後も維持されます。

21.5.15.1. RHEL システムロール firewall の概要

RHEL システムロールは、Ansible 自動化ユーティリティーのコンテンツセットです。このコンテンツは、Ansible 自動化ユーティリティーとともに、複数の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します。

firewalld サービスの自動設定に、RHEL システムロールからの **rhel-system-roles.firewall** ロールが導入されました。**rhel-system-roles** パッケージには、このシステムロールと参考ドキュメントも含まれます。

firewalld パラメーターを自動化された方法で 1 つ以上のシステムに適用するには、Playbook で **firewall** システムロール変数を使用します。Playbook は、テキストベースの YAML 形式で記述された 1 つ以上のプレイのリストです。

インベントリーファイルを使用して、Ansible が設定するシステムセットを定義できます。

firewall ロールを使用すると、以下のような異なる **firewalld** パラメーターを設定できます。

- ゾーン。
- パケットが許可されるサービス。
- ポートへのトラフィックアクセスの付与、拒否、または削除。
- ゾーンのポートまたはポート範囲の転送。

関連情報

- **/usr/share/doc/rhel-system-roles/firewall/** ディレクトリーの **README.md** ファイルおよび **README.html** ファイル
- Playbook の使用
- インベントリーの構築方法

21.5.15.2. ファイアウォール RHEL システムロールを使用した firewalld 設定のリセット

firewall RHEL システムロールを使用すると、**firewalld** 設定をデフォルトの状態にリセットできます。**previous:replaced** パラメーターを変数リストに追加すると、システムロールは既存のユーザー定義の設定をすべて削除し、**firewalld** をデフォルトにリセットします。**previous:replaced** パラメーター

を他の設定と組み合わせると、**firewall** ロールは新しい設定を適用する前に既存の設定をすべて削除します。

Ansible コントロールノードで以下の手順を実行します。

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は、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/reset-firewalld.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
  - name: Reset firewalld
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - previous: replaced
```

2. Playbook を実行します。

```
# ansible-playbook ~/configuring-a-dmz.yml
```

検証

- 管理対象ノードで **root** として次のコマンドを実行し、すべてのゾーンを確認します。

```
# firewall-cmd --list-all-zones
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**
- **ansible-playbook(1)**
- **firewalld(1)**

21.5.15.3. 別のローカルポートへの着信トラフィックの転送

firewall ロールを使用すると、複数の管理対象ホストで設定が永続化されるので **firewalld** パラメーターをリモートで設定できます。

Ansible コントロールノードで以下の手順を実行します。

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は、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/port_forwarding.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Forward incoming traffic on port 8080 to 443
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. Playbook を実行します。

```
# ansible-playbook ~/port_forwarding.yml
```

検証

- 管理対象ホストで、**firewalld** 設定を表示します。

```
# firewall-cmd --list-forward-ports
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

21.5.15.4. システムロールを使用したポートの設定

RHEL **firewall** システムロールを使用すると、着信トラフィックに対してローカルファイアウォールでポートを開くか閉じて、再起動後に新しい設定を永続化できます。たとえば、HTTPS サービスの着信トラフィックを許可するようにデフォルトゾーンを設定できます。

Ansible コントロールノードで以下の手順を実行します。

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は、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/opening-a-port.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Allow incoming HTTPS traffic to the local host
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - port: 443/tcp
          service: http
          state: enabled
          runtime: true
          permanent: true
```

permanent: true オプションを使用すると、再起動後も新しい設定が維持されます。

2. Playbook を実行します。

```
# ansible-playbook ~/opening-a-port.yml
```

検証

- 管理対象ノードで、**HTTPS** サービスに関連付けられた **443/tcp** ポートが開いていることを確認します。

```
# firewall-cmd --list-ports
443/tcp
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

21.5.15.5. firewalld RHEL システムロールを使用した DMZ firewalld ゾーンの設定

システム管理者は、**firewall** システムロールを使用して、**enp1s0** インターフェイスで **dmz** ゾーンを設定し、ゾーンへの **HTTPS** トラフィックを許可できます。これにより、外部ユーザーが Web サーバーにアクセスできるようにします。

Ansible コントロールノードで以下の手順を実行します。

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は、そのノードに対する **sudo** 権限がある。
- この Playbook を実行する管理対象ノードまたは管理対象ノードのグループが、Ansible インベントリーファイルにリストされている。

手順

1. **~/configuring-a-dmz.yml** などの Playbook ファイルを次の内容で作成します。

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - zone: dmz
          interface: enp1s0
          service: https
          state: enabled
          runtime: true
          permanent: true
```

2. Playbook を実行します。

```
# ansible-playbook ~/configuring-a-dmz.yml
```

検証

- 管理ノードで、**dmz** ゾーンに関する詳細情報を表示します。

```
# firewall-cmd --zone=dmz --list-all
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
```

```
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

21.5.16. 関連情報

- **firewalld(1)** の man ページ
- **firewalld.conf(5)** の man ページ
- **firewall-cmd(1)** man ページ
- **firewall-config(1)** の man ページ
- **firewall-offline-cmd(1)** の man ページ
- **firewalld.icmptype(5)** の man ページ
- **firewalld.ipset(5)** の man ページ
- **firewalld.service(5)** の man ページ
- **firewalld.zone(5)** の man ページ
- **firewalld.direct(5)** の man ページ
- **firewalld.lockdown-whitelist(5)**
- **firewalld.richlanguage(5)**
- **firewalld.zones(5)** の man ページ
- **firewalld.dbus(5)** の man ページ

21.6. NFTABLES の使用

nftables フレームワークはパケットを分類し、**iptables**、**ip6tables**、**arptables**、**ebtables**、および **ipset** ユーティリティーの後継です。利便性、機能、パフォーマンスにおいて、以前のパケットフィルタリングツールに多くの改良が追加されました。以下に例を示します。

- 線形処理の代わりに組み込みルックアップテーブルを使用
- **IPv4** プロトコルおよび **IPv6** プロトコルに対する 1 つのフレームワーク
- 完全ルールセットのフェッチ、更新、および保存を行わず、すべてアトミックに適用されるルール
- ルールセットにおけるデバッグおよびトレースへの対応 (**nftrace**) およびトレースイベントの監視 (**nft** ツール)

- より統一されたコンパクトな構文、プロトコル固有の拡張なし
- サードパーティーのアプリケーション用 Netlink API

nftables フレームワークは、テーブルを使用してチェーンを保存します。このチェーンには、アクションを実行する個々のルールが含まれます。**nft** ユーティリティーは、以前のパケットフィルタリングフレームワークのツールをすべて置き換えます。**libmnl** ライブラリーを介して、**nftables** Netlink API との低レベルの対話に **libnftnl** ライブラリーを使用できます。

ルールセット変更が適用されていることを表示するには、**nft list ruleset** コマンドを使用します。これらのユーティリティーはテーブル、チェーン、ルール、セット、およびその他のオブジェクトを **nftables** ルールセットに追加するため、**nft flush ruleset** コマンドなどの **nftables** ルールセット操作は、**iptables** コマンドを使用してインストールされたルールセットに影響を与える可能性があることに注意してください。

21.6.1. iptables から nftables への移行

ファイアウォール設定が依然として **iptables** ルールを使用している場合は、**iptables** ルールを **nftables** に移行できます。

21.6.1.1. firewalld、nftables、または iptables を使用する場合

以下は、次のユーティリティーのいずれかを使用する必要があるシナリオの概要です。

- **firewalld**:簡単なファイアウォールのユースケースには、**firewalld** ユーティリティーを使用します。このユーティリティーは、使いやすく、このようなシナリオの一般的な使用例に対応しています。
- **nftables**:**nftables** ユーティリティーを使用して、ネットワーク全体など、複雑なパフォーマンスに関する重要なファイアウォールを設定します。
- **iptables**:Red Hat Enterprise Linux の **iptables** ユーティリティーは、**legacy** バックエンドの代わりに **nf_tables** カーネル API を使用します。**nf_tables** API は、**iptables** コマンドを使用するスクリプトが、Red Hat Enterprise Linux で引き続き動作するように、後方互換性を提供します。新しいファイアウォールスクリプトの場合には、Red Hat は **nftables** を使用することを推奨します。

重要

ファイアウォールサービスがそれぞれに影響し合うことを回避するためには、RHEL ホストでそのうちの 1 つだけを実行し、他のサービスを無効にします。

21.6.1.2. iptables および ip6tables ルールセットの nftables への変換

iptables-restore-translate ユーティリティーおよび **ip6tables-restore-translate** ユーティリティーを使用して、**iptables** および **ip6tables** ルールセットを **nftables** に変換します。

前提条件

- **nftables** パッケージおよび **iptables** パッケージがインストールされている。
- システムに **iptables** ルールおよび **ip6tables** ルールが設定されている。

手順

1. **iptables** ルールおよび **ip6tables** ルールをファイルに書き込みます。

```
# iptables-save >/root/iptables.dump
# ip6tables-save >/root/ip6tables.dump
```

2. ダンプファイルを **nftables** 命令に変換します。

```
# iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-from-iptables.nft
# ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-from-ip6tables.nft
```

3. 必要に応じて、生成された **nftables** ルールを手動で更新して、確認します。

4. **nftables** サービスが生成されたファイルをロードできるようにするには、以下を **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5. **iptables** サービスを停止し、無効にします。

```
# systemctl disable --now iptables
```

カスタムスクリプトを使用して **iptables** ルールを読み込んだ場合は、スクリプトが自動的に開始されなくなったことを確認し、再起動してすべてのテーブルをフラッシュします。

6. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

- **nftables** ルールセットを表示します。

```
# nft list ruleset
```

関連情報

- システムの起動時に nftables ルールの自動読み込み

21.6.1.3. 単一の iptables および ip6tables ルールセットの nftables への変換

Red Hat Enterprise Linux は、**iptables** ルールまたは **ip6tables** ルールを、**nftables** で同等のルールに変換する **iptables-translate** ユーティリティーおよび **ip6tables-translate** ユーティリティーを提供します。

前提条件

- **nftables** パッケージがインストールされている。

手順

- 以下のように、**iptables** または **ip6tables** の代わりに **iptables-translate** ユーティリティーまたは **ip6tables-translate** ユーティリティーを使用して、対応する **nftables** ルールを表示します。

```
# iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

拡張機能によっては変換機能がない場合もあります。このような場合には、ユーティリティーは、以下のように、前に **#** 記号が付いた未変換ルールを出力します。

```
# iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

関連情報

- **iptables-translate --help**

21.6.1.4. 一般的な iptables コマンドと nftables コマンドの比較

以下は、一般的な **iptables** コマンドと **nftables** コマンドの比較です。

- すべてのルールをリスト表示します。

iptables	nftables
iptables-save	**nft list ruleset**

- 特定のテーブルおよびチェーンをリスト表示します。

iptables	nftables
iptables -L	**nft list table ip filter**
iptables -L INPUT	**nft list chain ip filter INPUT**
iptables -t nat -L PREROUTING	**nft list chain ip nat PREROUTING**

nft コマンドは、テーブルおよびチェーンを事前に作成しません。これらは、ユーザーが手動で作成した場合にのみ存在します。

firewalld によって生成されたルールの一覧表示:

```
# nft list table inet firewalld
# nft list table ip firewalld
# nft list table ip6 firewalld
```

21.6.1.5. 関連情報

- iptables:2 つのバリアントと nftables の関係

21.6.2. nftables スクリプトの作成および実行

nftables` フレームワークを使用する主な利点は、スクリプトの実行がアトミックであることです。つまり、システムがスクリプト全体を適用するか、エラーが発生した場合には実行を阻止することを意味します。これにより、ファイアウォールは常に一貫した状態になります。

さらに、**nftables** スクリプト環境を使用すると、次のことができます。

- コメントの追加
- 変数の定義
- 他のルールセットファイルの組み込み

nftables パッケージをインストールすると、Red Hat Enterprise Linux が自動的に ***.nft** スクリプトを **/etc/nftables/** ディレクトリーに作成します。このスクリプトには、さまざまな目的でテーブルと空のチェーンを作成するコマンドが含まれます。

21.6.2.1. 対応している nftables スクリプトの形式

nftables スクリプト環境では、次の形式でスクリプトを記述できます。

- **nft list ruleset** コマンドと同じ形式でルールセットが表示されます。

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
  chain example_chain {
    # Chain for incoming packets that drops all packets that
    # are not explicitly allowed by any rule in this chain
    type filter hook input priority 0; policy drop;

    # Accept connections to port 22 (ssh)
    tcp dport ssh accept
  }
}
```

- **nft** コマンドと同じ構文:

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table

# Create a chain for incoming packets that drops all packets
# that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0 ; policy drop ; }

# Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

21.6.2.2. nftables スクリプトの実行

nftables スクリプトは、**nft** ユーティリティーに渡すか、スクリプトを直接実行することで実行できます。

手順

- **nftables** スクリプトを **nft** ユーティリティーに渡して実行するには、次のコマンドを実行します。

```
# nft -f /etc/nftables/<example_firewall_script>.nft
```

- **nftables** スクリプトを直接実行するには、次のコマンドを実行します。

  a. 1 回だけ実行する場合:

  i. スクリプトが以下のシバンシーケンスで始まることを確認します。

```
#!/usr/sbin/nft -f
```

重要

-f パラメーターを省略すると、**nft** ユーティリティーはスクリプトを読み込まず、**Error: syntax error, unexpected newline, expecting string** のように表示されます。

  ii. オプション: スクリプトの所有者を **root** に設定します。

```
# chown root /etc/nftables/<example_firewall_script>.nft
```

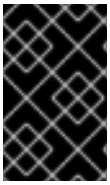
  iii. 所有者のスクリプトを実行ファイルに変更します。

```
# chmod u+x /etc/nftables/<example_firewall_script>.nft
```

  b. スクリプトを実行します。

```
# /etc/nftables/<example_firewall_script>.nft
```

出力が表示されない場合は、システムがスクリプトを正常に実行します。



重要

nft はスクリプトを正常に実行しますが、ルールの配置やパラメーター不足、またはスクリプト内のその他の問題により、ファイアウォールが期待通りの動作を起こさない可能性があります。

関連情報

- **chown(1)** の man ページ
- **chmod(1)** の man ページ
- システムの起動時に nftables ルールの自動読み込み

21.6.2.3. nftables スクリプトでコメントの使用

nftables スクリプト環境は、**#** 文字の右側から行末までのすべてをコメントとして解釈します。

コメントは、行の先頭またはコマンドの横から開始できます。

```
...
# Flush the rule set
flush ruleset

add table inet example_table  # Create a table
...
```

21.6.2.4. nftables スクリプトでの変数の使用

nftables スクリプトで変数を定義するには、**define** キーワードを使用します。シングル値および匿名セットを変数に保存できます。より複雑なシナリオの場合は、セットまたは決定マップを使用します。

値を 1 つ持つ変数

以下の例は、値が **enp1s0** の **INET_DEV** という名前の変数を定義します。

```
define INET_DEV = enp1s0
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

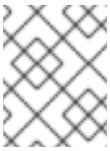
匿名セットを含む変数

以下の例では、匿名セットを含む変数を定義します。

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

スクリプトで変数を使用するには、**$** 記号と、それに続く変数名を指定します。

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```



注記

中括弧は、変数がセットを表していることを示すため、ルールで使用する場合は、特別なセマンティクスを持ちます。

関連情報

- nftables コマンドでのセットの使用
- nftables コマンドにおける決定マップの使用

21.6.2.5. nftables スクリプトへのファイルの追加

nftables スクリプト環境では、**include** ステートメントを使用して他のスクリプトを含めることができます。

絶対パスまたは相対パスのないファイル名のみを指定すると、**nftables** には、デフォルトの検索パスのファイルが含まれます。これは、Red Hat Enterprise Linux では **/etc** に設定されています。

例21.1 デフォルト検索ディレクトリーからのファイルを含む

デフォルトの検索ディレクトリーからファイルを指定するには、次のコマンドを実行します。

```
include "example.nft"
```

例21.2 ディレクトリーの *.nft ファイルをすべて含む

***.nft** で終わるすべてのファイルを **/etc/nftables/rulesets/** ディレクトリーに保存するには、次のコマンドを実行します。

```
include "/etc/nftables/rulesets/*.nft"
```

include ステートメントは、ドットで始まるファイルに一致しないことに注意してください。

関連情報

- **nft(8)** の man ページの **Include files** セクション

21.6.2.6. システムの起動時に nftables ルールの自動読み込み

systemd サービス **nftables** は、**/etc/sysconfig/nftables.conf** ファイルに含まれるファイアウォールスクリプトを読み込みます。

前提条件

- **nftables** スクリプトは、**/etc/nftables/** ディレクトリーに保存されます。

手順

1. **/etc/sysconfig/nftables.conf** ファイルを編集します。
   - **nftables** パッケージのインストールで **/etc/nftables/** に作成された ***.nft** スクリプトを変更した場合は、これらのスクリプトの **include** ステートメントのコメントを解除します。
   - 新しいスクリプトを作成した場合は、**include** ステートメントを追加してこれらのスクリプトを含めます。たとえば、**nftables** サービスの起動時に **/etc/nftables/example.nft** スクリプトを読み込むには、以下を追加します。

     ```
     include "/etc/nftables/_example_.nft"
     ```

2. オプション: **nftables** サービスを開始して、システムを再起動せずにファイアウォールルールを読み込みます。

   ```
   # systemctl start nftables
   ```

3. **nftables** サービスを有効にします。

```
# systemctl enable nftables
```

関連情報

- 対応している nftables スクリプトの形式

21.6.3. nftables テーブル、チェーン、およびルールの作成および管理

nftables ルールセットを表示して管理できます。

21.6.3.1. nftables テーブルの基本

nftables のテーブルは、チェーン、ルール、セットなどのオブジェクトを含む名前空間です。

各テーブルにはアドレスファミリーが割り当てられている必要があります。アドレスファミリーは、このテーブルが処理するパケットタイプを定義します。テーブルを作成する際に、以下のいずれかのアドレスファミリーを設定できます。

- **ip**:IPv4 パケットだけに一致します。アドレスファミリーを指定しないと、これがデフォルトになります。
- **ip6**:IPv6 パケットだけに一致します。
- **inet**:IPv4 パケットと IPv6 パケットの両方に一致します。
- **arp**:IPv4 アドレス解決プロトコル (ARP) パケットに一致します。
- **bridge**:ブリッジデバイスを通過するパケットに一致します。
- **netdev**:ingress からのパケットに一致します。

テーブルを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
}
```

- シェルスクリプトで、以下を使用します。

```
nft add table <table_address_family> <table_name>
```

21.6.3.2. nftables チェーンの基本

テーブルは、ルールのコンテナーであるチェーンで構成されます。次の 2 つのルールタイプが存在します。

- **ベースチェーン**:ネットワークスタックからのパケットのエントリーポイントとしてベースチェーンを使用できます。
- **通常のチェーン**:**jump** ターゲットとして通常のチェーンを使用し、ルールをより適切に整理できます。

ベースチェーンをテーブルに追加する場合に使用する形式は、ファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority>
    policy <policy> ;
  }
}
```

- シェルスクリプトで、以下を使用します。

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook <hook> priority <priority> \; policy <policy> \; }
```

シェルがセミコロンをコマンドの最後として解釈しないようにするには、セミコロンの前にエスケープ文字 \ を配置します。

どちらの例でも、**ベースチェーン** を作成します。**通常のチェーン** を作成する場合、中括弧内にパラメーターを設定しないでください。

チェーンタイプ
チェーンタイプとそれらを使用できるアドレスファミリーとフックの概要を以下に示します。

タイプ	アドレスファミリー	フック	説明
filter	all	all	標準のチェーンタイプ
nat	**ip、ip6、inet**	**prerouting、input、output、postrouting**	このタイプのチェーンは、接続追跡エントリーに基づいてネイティブアドレス変換を実行します。最初のパケットのみがこのチェーンタイプをトラバースします。
ルート	**ip、ip6**	**出力 (output)**	このチェーンタイプを通過する許可済みパケットは、IP ヘッダーの関連部分が変更された場合に、新しいルートルックアップを引き起こします。

チェーンの優先度
priority パラメーターは、パケットが同じフック値を持つチェーンを通過する順序を指定します。このパラメーターは、整数値に設定することも、標準の priority 名を使用することもできます。

以下のマトリックスは、標準的な priority 名とその数値の概要、それらを使用できるファミリーおよびフックの概要です。

テキストの値	数値	アドレスファミリー	フック
raw	**-300**	**ip、ip6、inet**	all

テキストの値	数値	アドレスファミリー	フック
mangle	**-150**	**ip、ip6、inet**	all
dstnat	**-100**	**ip、ip6、inet**	**prerouting**
	-300	**bridge**	**prerouting**
filter	**0**	**ip、ip6、inet、arp、netdev**	all
	-200	**bridge**	all
security	**50**	**ip、ip6、inet**	all
srcnat	**100**	**ip、ip6、inet**	**postrouting**
	300	**bridge**	**postrouting**
out	**100**	**bridge**	**出力 (output)**

チェーンポリシー
チェーンポリシーは、このチェーンのルールでアクションが指定されていない場合に、**nftables** がパケットを受け入れるかドロップするかを定義します。チェーンには、以下のいずれかのポリシーを設定できます。

- **accept** (デフォルト)
- **drop**

21.6.3.3. nftables ルールの基本

ルールは、このルールを含むチェーンを渡すパケットに対して実行するアクションを定義します。ルールに一致する式も含まれる場合、**nftables** は、以前の式がすべて適用されている場合にのみアクションを実行します。

チェーンにルールを追加する場合、使用する形式はファイアウォールスクリプトにより異なります。

- ネイティブ構文のスクリプトでは、以下を使用します。

```
table <table_address_family> <table_name> {
  chain <chain_name> {
    type <type> hook <hook> priority <priority> ; policy <policy> ;
      <rule>
  }
}
```

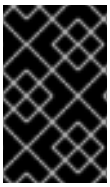
- シェルスクリプトで、以下を使用します。

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

このシェルコマンドは、チェーンの最後に新しいルールを追加します。チェーンの先頭にルールを追加する場合は、**nft add** の代わりに **nft insert** コマンドを使用します。

21.6.3.4. nft コマンドを使用したテーブル、チェーン、ルールの管理

コマンドラインまたはシェルスクリプトで **nftables** ファイアウォールを管理するには、**nft** ユーティリティーを使用します。



重要

この手順のコマンドは通常のワークフローを表しておらず、最適化されていません。この手順では、**nft** コマンドを使用して、一般的なテーブル、チェーン、およびルールを管理する方法を説明します。

手順

1. テーブルが IPv4 パケットと IPv6 パケットの両方を処理できるように、**inet** アドレスファミリーを使用して **nftables_svc** という名前のテーブルを作成します。

```
# nft add table inet nftables_svc
```

2. 受信ネットワークトラフィックを処理する **INPUT** という名前のベースチェーンを **inet nftables_svc** テーブルに追加します。

```
# nft add chain inet nftables_svc INPUT { type filter hook input priority filter \; policy accept \; }
```

シェルがセミコロンをコマンドの最後として解釈しないようにするには、\ 文字を使用してセミコロンをエスケープします。

3. **INPUT** チェーンにルールを追加します。たとえば、**INPUT** チェーンの最後のルールとして、ポート 22 および 443 で着信 TCP トラフィックを許可し、Internet Control Message Protocol (ICMP)ポートに到達できないメッセージで他の着信トラフィックを拒否します。

```
# nft add rule inet nftables_svc INPUT tcp dport 22 accept
# nft add rule inet nftables_svc INPUT tcp dport 443 accept
# nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

ここで示されたように **nft add rule** コマンドを実行すると、**nft** はコマンド実行と同じ順序でルールをチェーンに追加します。

4. ハンドルを含む現在のルールセットを表示します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

5. ハンドル 3 で既存ルールの前にルールを挿入します。たとえば、ポート 636 で TCP トラ

5. ハンドル 3 で既存ルールの前にルールを挿入します。たとえば、ポート 636 で TCP トラフィックを許可するルールを挿入するには、以下を入力します。

```
# nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6. ハンドル 3 で、既存ルールの後ろにルールを追加します。たとえば、ポート 80 で TCP トラフィックを許可するルールを追加するには、以下を入力します。

```
# nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7. ハンドルでルールセットを再表示します。後で追加したルールが指定の位置に追加されていることを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    tcp dport 80 accept # handle 6
    reject # handle 4
  }
}
```

8. ハンドル 6 でルールを削除します。

```
# nft delete rule inet nftables_svc INPUT handle 6
```

ルールを削除するには、ハンドルを指定する必要があります。

9. ルールセットを表示し、削除されたルールがもう存在しないことを確認します。

```
# nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
  chain INPUT { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport 22 accept # handle 2
    tcp dport 636 accept # handle 5
    tcp dport 443 accept # handle 3
    reject # handle 4
  }
}
```

10. **INPUT** チェーンから残りのルールをすべて削除します。

```
# nft flush chain inet nftables_svc INPUT
```

11. ルールセットを表示し、**INPUT** チェーンが空であることを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
  chain INPUT {
```

```
    type filter hook input priority filter; policy accept
  }
}
```

12. **INPUT** チェーンを削除します。

```
# nft delete chain inet nftables_svc INPUT
```

このコマンドを使用して、まだルールが含まれているチェーンを削除することもできます。

13. ルールセットを表示し、**INPUT** チェーンが削除されたことを確認します。

```
# nft list table inet nftables_svc
table inet nftables_svc {
}
```

14. **nftables_svc** テーブルを削除します。

```
# nft delete table inet nftables_svc
```

このコマンドを使用して、まだルールが含まれているテーブルを削除することもできます。

注記

ルールセット全体を削除するには、個別のコマンドですべてのルール、チェーン、およびテーブルを手動で削除するのではなく、**nft flush ruleset** コマンドを使用します。

関連情報

nft(8) の man ページ

21.6.4. nftables を使用した NAT の設定

nftables を使用すると、以下のネットワークアドレス変換 (NAT) タイプを設定できます。

- マスカレーディング
- ソース NAT (SNAT)
- 宛先 NAT (DNAT)
- リダイレクト

重要

iifname パラメーターおよび **oifname** パラメーターでは実インターフェイス名のみを使用でき、代替名 (**altname**) には対応していません。

21.6.4.1. NAT タイプ

以下は、ネットワークアドレス変換 (NAT) タイプになります。

マスカレードおよびソースの NAT (SNAT)

この NAT タイプのいずれかを使用して、パケットのソース IP アドレスを変更します。たとえば、インターネットサービスプロバイダーは、プライベート IP 範囲 (**10.0.0.0/8** など) をルーティングしません。ネットワークでプライベート IP 範囲を使用し、ユーザーがインターネット上のサーバーにアクセスできるようにする必要がある場合は、この範囲のパケットのソース IP アドレスをパブリック IP アドレスにマップします。
マスカレードと SNAT は互いに非常に似ています。相違点は次のとおりです。

- マスカレードは、出力インターフェイスの IP アドレスを自動的に使用します。したがって、出力インターフェイスが動的 IP アドレスを使用する場合は、マスカレードを使用します。
- SNAT は、パケットのソース IP アドレスを指定された IP に設定し、出力インターフェイスの IP アドレスを動的に検索しません。そのため、SNAT の方がマスカレードよりも高速です。出力インターフェイスが固定 IP アドレスを使用する場合は、SNAT を使用します。

宛先 NAT (DNAT)

この NAT タイプを使用して、着信パケットの宛先アドレスとポートを書き換えます。たとえば、Web サーバーがプライベート IP 範囲の IP アドレスを使用しているため、インターネットから直接アクセスできない場合は、ルーターに DNAT ルールを設定し、着信トラフィックをこのサーバーにリダイレクトできます。

リダイレクト

このタイプは、チェーンフックに応じてパケットをローカルマシンにリダイレクトする DNAT の特殊なケースです。たとえば、サービスが標準ポートとは異なるポートで実行する場合は、標準ポートからこの特定のポートに着信トラフィックをリダイレクトすることができます。

21.6.4.2. nftables を使用したマスカレードの設定

マスカレードを使用すると、ルーターは、インターフェイスを介して送信されるパケットのソース IP を、インターフェイスの IP アドレスに動的に変更できます。これは、インターフェイスに新しい IP が割り当てられている場合に、**nftables** はソース IP の置き換え時に新しい IP を自動的に使用することを意味します。

ens3 インターフェイスを介してホストから出るパケットの送信元 IP を、**ens3** で設定された IP に置き換えます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

重要

prerouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、着信パケット返信に一致するようにこのチェーンが必要になります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **postrouting** チェーンに、**ens3** インターフェイスの出力パケットに一致するルールを追加します。

```
# nft add rule nat postrouting oifname "ens3" masquerade
```

21.6.4.3. nftables を使用したソース NAT の設定

ルーターでは、ソース NAT (SNAT) を使用して、インターフェイスを介して特定の IP アドレスに送信するパケットの IP を変更できます。次に、ルーターは送信パケットのソース IP を置き換えます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```

重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **ens3** を介した発信パケットのソース IP を **192.0.2.1** に置き換えるルールを **postrouting** チェーンに追加します。

```
# nft add rule nat postrouting oifname "ens3" snat to 192.0.2.1
```

関連情報

- 特定のローカルポートで着信パケットを別のホストに転送

21.6.4.4. nftables を使用した宛先 NAT の設定

宛先 NAT (DNAT)を使用すると、ルーター上のトラフィックをインターネットから直接アクセスできないホストにリダイレクトできます。

たとえば、DNAT を使用すると、ルーターはポート **80** および **443** に送信された受信トラフィックを、IP アドレス **192.0.2.1** の Web サーバーにリダイレクトします。

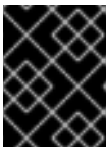
手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
```



重要

postrouting チェーンにルールを追加しなくても、**nftables** フレームワークでは、このチェーンが発信パケット返信に一致するようにする必要があります。

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **prerouting** チェーンに、ルーターの **ens3** インターフェイスのポート **80** および **443** に受信トラフィックを、IP アドレス **192.0.2.1** を持つ Web サーバーにリダイレクトするルールを追加します。

```
# nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4. 環境に応じて、SNAT ルールまたはマスカレードルールを追加して、Web サーバーから返されるパケットのソースアドレスを送信者に変更します。

   a. **ens3** インターフェイスが動的 IP アドレスを使用している場合は、マスカレードルールを追加します。

   ```
   # nft add rule nat postrouting oifname "ens3" masquerade
   ```

   b. **ens3** インターフェイスが静的 IP アドレスを使用する場合は、SNAT ルールを追加します。たとえば、**ens3** が IP アドレス **198.51.100.1** を使用している場合は、以下のようになります。

   ```
   # nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
   ```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

関連情報

- NAT タイプ

21.6.4.5. nftables を使用したリダイレクトの設定

redirect 機能は、チェーンフックに応じてパケットをローカルマシンにリダイレクトする宛先ネットワークアドレス変換 (DNAT) の特殊なケースです。

たとえば、ローカルホストのポート **22** に送信された着信および転送されたトラフィックを **2222** ポートにリダイレクトすることができます。

手順

1. テーブルを作成します。

```
# nft add table nat
```

2. テーブルに **prerouting** チェーンを追加します。

```
# nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにする必要があることに注意してください。

3. **22** ポートの着信トラフィックを **2222** ポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule nat prerouting tcp dport 22 redirect to 2222
```

関連情報

- NAT タイプ

21.6.5. nftables コマンドでのセットの使用

nftables フレームワークは、セットをネイティブに対応します。たとえば、ルールが複数の IP アドレス、ポート番号、インターフェイス、またはその他の一致基準に一致する必要がある場合など、セットを使用できます。

21.6.5.1. nftables での匿名セットの使用

匿名セットには、ルールで直接使用する **{ 22, 80, 443 }** などの中括弧で囲まれたコンマ区切りの値が含まれます。IP アドレスやその他の一致基準にも匿名セットを使用できます。

匿名セットの欠点は、セットを変更する場合はルールを置き換える必要があることです。動的なソリューションの場合は、nftables で名前付きセットの使用 で説明されているように名前付きセットを使用します。

前提条件

- **inet** ファミリーに **example_chain** チェーンおよび **example_table** テーブルがある。

手順

1. たとえば、ポート **22**、**80**、および **443** に着信トラフィックを許可するルールを、**example_table** の **example_chain** に追加するには、次のコマンドを実行します。

```
# nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2. オプション: **example_table** 内のすべてのチェーンとそのルールを表示します。

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport { ssh, http, https } accept
  }
}
```

21.6.5.2. nftables で名前付きセットの使用

nftables フレームワークは、変更可能な名前付きセットに対応します。名前付きセットは、テーブル内の複数のルールで使用できる要素のリストまたは範囲です。匿名セットに対する別の利点として、セットを使用するルールを置き換えることなく、名前付きセットを更新できます。

名前付きセットを作成する場合は、セットに含まれる要素のタイプを指定する必要があります。以下のタイプを設定できます。

- **192.0.2.1** や **192.0.2.0/24** など、IPv4 アドレスまたは範囲を含むセットの場合は **ipv4_addr**。
- **2001:db8:1::1** や **2001:db8:1::1/64** など、IPv6 アドレスまたは範囲を含むセットの場合は **ipv6_addr**。
- **52:54:00:6b:66:42** など、メディアアクセス制御 (MAC) アドレスの一覧を含むセットの場合は **ether_addr**。
- **tcp** など、インターネットプロトコルタイプの一覧が含まれるセットの場合は **inet_proto**。
- **ssh** など、インターネットサービスの一覧を含むセットの場合は **inet_service**。
- パケットマークの一覧を含むセットの場合は **mark**。パケットマークは、任意の 32 ビットの正の整数値 (**0** から **2147483647**) にすることができます。

前提条件

- **example_chain** チェーンと **example_table** テーブルが存在する。

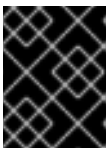
手順

1. 空のファイルを作成します。以下の例では、IPv4 アドレスのセットを作成します。
   - 複数の IPv4 アドレスを格納することができるセットを作成するには、次のコマンドを実行します。

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; }
     ```

   - IPv4 アドレス範囲を保存できるセットを作成するには、次のコマンドを実行します。

     ```
     # nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
     ```

     重要

     シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

2. オプション: セットを使用するルールを作成します。たとえば、次のコマンドは、**example_set** の IPv4 アドレスからのパケットをすべて破棄するルールを、**example_table** の **example_chain** に追加します。

   ```
   # nft add rule inet example_table example_chain ip saddr @example_set drop
   ```

   example_set が空のままなので、ルールには現在影響がありません。

3. IPv4 アドレスを **example_set** に追加します。

- 個々の IPv4 アドレスを保存するセットを作成する場合は、次のコマンドを実行します。

```
# nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
```

- IPv4 範囲を保存するセットを作成する場合は、次のコマンドを実行します。

```
# nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
```

IP アドレス範囲を指定する場合は、上記の例の **192.0.2.0/24** のように、CIDR (Classless Inter-Domain Routing) 表記を使用することもできます。

21.6.5.3. 関連情報

- **nft(8)** の man ページの **Sets** セクション

21.6.6. nftables コマンドにおける決定マップの使用

ディクショナリーとしても知られている決定マップにより、**nft** は一致基準をアクションにマッピングすることで、パケット情報に基づいてアクションを実行できます。

21.6.6.1. nftables での匿名マップの使用

匿名マップは、ルールで直接使用する **{ match_criteria : action }** ステートメントです。ステートメントには、複数のコンマ区切りマッピングを含めることができます。

匿名マップの欠点は、マップを変更する場合には、ルールを置き換える必要があることです。動的なソリューションの場合は、nftables での名前付きマップの使用 で説明されているように名前付きマップを使用します。

たとえば、匿名マップを使用して、IPv4 プロトコルおよび IPv6 プロトコルの TCP パケットと UDP パケットの両方を異なるチェーンにルーティングし、着信 TCP パケットと UDP パケットを個別にカウントできます。

手順

1. 新しいテーブルを作成します。

```
# nft add table inet example_table
```

2. **example_table** に **tcp_packets** チェーンを作成します。

```
# nft add chain inet example_table tcp_packets
```

3. このチェーンのトラフィックをカウントする **tcp_packets** にルールを追加します。

```
# nft add rule inet example_table tcp_packets counter
```

4. **example_table** で **udp_packets** チェーンを作成します。

```
# nft add chain inet example_table udp_packets
```

5. このチェーンのトラフィックをカウントする **udp_packets** にルールを追加します。

```
# nft add rule inet example_table udp_packets counter
```

6. 着信トラフィックのチェーンを作成します。たとえば、**example_table** に、着信トラフィックをフィルタリングする **incoming_traffic** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7. 匿名マップを持つルールを **incoming_traffic** に追加します。

```
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

匿名マップはパケットを区別し、プロトコルに基づいて別のカウンターチェーンに送信します。

8. トラフィックカウンターの一覧を表示する場合は、**example_table** を表示します。

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
    counter packets 36379 bytes 2103816
  }

  chain udp_packets {
    counter packets 10 bytes 1559
  }

  chain incoming_traffic {
    type filter hook input priority filter; policy accept;
    ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
  }
}
```

tcp_packets チェーンおよび **udp_packets** チェーンのカウンターは、受信パケットとバイトの両方を表示します。

21.6.6.2. nftables での名前付きマップの使用

nftables フレームワークは、名前付きマップに対応します。テーブルの複数のルールでこのマップを使用できます。匿名マップに対する別の利点は、名前付きマップを使用するルールを置き換えることなく、名前付きマップを更新できることです。

名前付きマップを作成する場合は、要素のタイプを指定する必要があります。

- 一致する部分に **192.0.2.1** などの IPv4 アドレスが含まれるマップの場合は **ipv4_addr**。
- 一致する部分に **2001:db8:1::1** などの IPv6 アドレスが含まれるマップの場合は **ipv6_addr**。
- **52:54:00:6b:66:42** などのメディアアクセス制御 (MAC) アドレスを含むマップの場合は **ether_addr**。
- 一致する部分に **tcp** などのインターネットプロトコルタイプが含まれるマップの場合は **inet_proto**。

- 一致する部分に **ssh** や **22** などのインターネットサービス名のポート番号が含まれるマップの場合は **inet_service**。
- 一致する部分にパケットマークが含まれるマップの場合は **mark**。パケットマークは、任意の正の 32 ビットの整数値 (**0** ~ **2147483647**) にできます。
- 一致する部分にカウンターの値が含まれるマップの場合は **counter**。カウンター値は、正の値の 64 ビットであれば任意の値にすることができます。
- 一致する部分にクォータ値が含まれるマップの場合は **quota**。クォータの値は、64 ビットの整数値にできます。

たとえば、送信元 IP アドレスに基づいて着信パケットを許可または拒否できます。名前付きマップを使用すると、このシナリオを設定するのに必要なルールは 1 つだけで、IP アドレスとアクションがマップに動的に保存されます。

手順

1. テーブルを作成します。たとえば、IPv4 パケットを処理する **example_table** という名前のテーブルを作成するには、次のコマンドを実行します。

```
# nft add table ip example_table
```

2. チェーンを作成します。たとえば、**example_table** に、**example_chain** という名前のチェーンを作成するには、次のコマンドを実行します。

```
# nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```

重要

シェルがセミコロンをコマンドの終わりとして解釈しないようにするには、バックスラッシュでセミコロンをエスケープする必要があります。

3. 空のマップを作成します。たとえば、IPv4 アドレスのマッピングを作成するには、次のコマンドを実行します。

```
# nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4. マップを使用するルールを作成します。たとえば、次のコマンドは、両方とも **example_map** で定義されている IPv4 アドレスにアクションを適用するルールを、 **example_table** の **example_chain** に追加します。

```
# nft add rule example_table example_chain ip saddr vmap @example_map
```

5. IPv4 アドレスと対応するアクションを **example_map** に追加します。

```
# nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

以下の例では、IPv4 アドレスのアクションへのマッピングを定義します。上記で作成したルールと組み合わせて、ファイアウォールは **192.0.2.1** からのパケットを許可し、**192.0.2.2** からのパケットを破棄します。

6. オプション: 別の IP アドレスおよび action ステートメントを追加してマップを拡張します。

```
# nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7. オプション: マップからエントリーを削除します。

```
# nft delete element ip example_table example_map { 192.0.2.1 }
```

8. オプション: ルールセットを表示します。

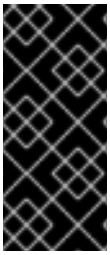
```
# nft list ruleset
table ip example_table {
  map example_map {
    type ipv4_addr : verdict
    elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
  }

  chain example_chain {
    type filter hook input priority filter; policy accept;
    ip saddr vmap @example_map
  }
}
```

21.6.6.3. 関連情報

- **nft(8)** の man ページの **Maps** セクション

21.6.7. 以下に例を示します。nftables スクリプトを使用した LAN および DMZ の保護

RHEL ルーターで **nftables** フレームワークを使用して、内部 LAN 内のネットワーククライアントと DMZ の Web サーバーを、インターネットやその他のネットワークからの不正アクセスから保護するファイアウォールスクリプトを作成およびインストールします。



重要

この例はデモ目的専用で、特定の要件があるシナリオを説明しています。

ファイアウォールスクリプトは、ネットワークインフラストラクチャーとセキュリティー要件に大きく依存します。この例を使用して、独自の環境用のスクリプトを作成する際に **nftables** ファイアウォールの概念を理解してください。

21.6.7.1. ネットワークの状態

この例のネットワークは、以下の条件下にあります。

- ルーターは以下のネットワークに接続されています。
  - インターフェイス **enp1s0**を介したインターネット
  - インターフェイス **enp7s0**を介した内部 LAN
  - **enp8s0**までの DMZ
- ルーターのインターネットインターフェイスには、静的 IPv4 アドレス (**203.0.113.1**) と IPv6 アドレス (**2001:db8:a::1**) の両方が割り当てられています。

- 内部 LAN のクライアントは **10.0.0.0/24** の範囲のプライベート IPv4 アドレスのみを使用します。その結果、LAN からインターネットへのトラフィックには、送信元ネットワークアドレス変換 (SNAT) が必要です。
- 内部 LAN の管理者用 PC は、IP アドレス **10.0.0.100** および **10.0.0.200** を使用します。
- DMZ は、**198.51.100.0/24** および **2001:db8:b::/56** の範囲のパブリック IP アドレスを使用します。
- DMZ の Web サーバーは、IP アドレス **198.51.100.5** および **2001:db8:b::5** を使用します。
- ルーターは、LAN および DMZ 内のホストのキャッシング DNS サーバーとして機能します。

21.6.7.2. ファイアウォールスクリプトのセキュリティー要件

以下は、サンプルネットワークにおける **nftables** ファイアウォールの要件です。

- ルーターは以下を実行できる必要があります。
  - DNS クエリーを再帰的に解決します。
  - ループバックインターフェイスですべての接続を実行します。
- 内部 LAN のクライアントは以下を実行できる必要があります。
  - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
  - DMZ の HTTPS サーバーにアクセスします。
  - インターネット上の任意の HTTPS サーバーにアクセスします。
- 管理者用の PC は、SSH を使用してルーターと DMZ 内のすべてのサーバーにアクセスできる必要があります。
- DMZ の Web サーバーは以下を実行できる必要があります。
  - ルーターで実行しているキャッシング DNS サーバーをクエリーします。
  - インターネット上の HTTPS サーバーにアクセスして更新をダウンロードします。
- インターネット上のホストは以下を実行できる必要があります。
  - DMZ の HTTPS サーバーにアクセスします。
- さらに、以下のセキュリティー要件が存在します。
  - 明示的に許可されていない接続の試行はドロップする必要があります。
  - ドロップされたパケットはログに記録する必要があります。

21.6.7.3. ドロップされたパケットをファイルにロギングするための設定

デフォルトでは、**systemd** は、ドロップされたパケットなどのカーネルメッセージをジャーナルに記録します。さらに、このようなエントリーを別のファイルに記録するように **rsyslog** サービスを設定することもできます。ログファイルが無限に大きくならないようにするために、ローテーションポリシーを設定します。

前提条件

- **rsyslog** パッケージがインストールされている。
- **rsyslog** サービスが実行されている。

手順

1. 以下の内容で **/etc/rsyslog.d/nftables.conf** ファイルを作成します。

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

この設定を使用すると、**rsyslog** サービスはドロップされたパケットを **/var/log/messages** ではなく **/var/log/nftables.log** ファイルに記録します。

2. **rsyslog** サービスを再起動します。

```
# systemctl restart rsyslog
```

3. サイズが 10 MB を超える場合は、以下の内容で **/etc/logrotate.d/nftables** ファイルを作成し、**/var/log/nftables.log** をローテーションします。

```
/var/log/nftables.log {
  size +10M
  maxage 30
  sharedscripts
  postrotate
    /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
  endscript
}
```

maxage 30 設定は、次のローテーション操作中に **logrotate** が 30 日経過したローテーション済みログを削除することを定義します。

関連情報

- **rsyslog.conf(5)** の man ページ
- **logrotate(8)** の man ページ

21.6.7.4. nftables スクリプトの作成とアクティブ化

この例は、RHEL ルーターで実行され、DMZ の内部 LAN および Web サーバーのクライアントを保護する **nftables** ファイアウォールスクリプトです。この例で使用されているネットワークとファイアウォールの要件について、詳しくはファイアウォールスクリプトの ネットワークの状態 および ファイアウォールスクリプトのセキュリティー要件 を参照してください。

警告

この **nftables** ファイアウォールスクリプトは、デモ専用です。お使いの環境やセキュリティー要件に適応させて使用してください。

前提条件

- ネットワークは、ネットワークの状態 で説明されているとおりに設定されます。

手順

1. 以下の内容で **/etc/nftables/firewall.nft** スクリプトを作成します。

```
# Remove all rules
flush ruleset


# Table for both IPv4 and IPv6 rules
table inet nftables_svc {

  # Define variables for the interface name
  define INET_DEV = enp1s0
  define LAN_DEV  = enp7s0
  define DMZ_DEV  = enp8s0


  # Set with the IPv4 addresses of admin PCs
  set admin_pc_ipv4 {
    type ipv4_addr
    elements = { 10.0.0.100, 10.0.0.200 }
  }


  # Chain for incoming trafic. Default policy: drop
  chain INPUT {
    type filter hook input priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # Accept incoming traffic on loopback interface
    iifname lo accept

    # Allow request from LAN and DMZ to local DNS server
    iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

    # Allow admins PCs to access the router using SSH
    iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

    # Last action: Log blocked packets
```

```
    # (packets that were not accepted in previous rules in this chain)
    log prefix "nft drop IN : "
  }


  # Chain for outgoing traffic. Default policy: drop
  chain OUTPUT {
    type filter hook output priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # Accept outgoing traffic on loopback interface
    oifname lo accept

    # Allow local DNS server to recursively resolve queries
    oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

    # Last action: Log blocked packets
    log prefix "nft drop OUT: "
  }


  # Chain for forwarding traffic. Default policy: drop
  chain FORWARD {
    type filter hook forward priority filter
    policy drop

    # Accept packets in established and related state, drop invalid packets
    ct state vmap { established:accept, related:accept, invalid:drop }

    # IPv4 access from LAN and Internet to the HTTPS server in the DMZ
    iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp dport
443 accept

    # IPv6 access from Internet to the HTTPS server in the DMZ
    iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443 accept

    # Access from LAN and DMZ to HTTPS servers on the Internet
    iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

    # Last action: Log blocked packets
    log prefix "nft drop FWD: "
  }


  # Postrouting chain to handle SNAT
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;

    # SNAT for IPv4 traffic from LAN to Internet
    iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
  }
}
```

2. **/etc/nftables/firewall.nft** スクリプトを **/etc/sysconfig/nftables.conf** ファイルに追加します。

```
include "/etc/nftables/firewall.nft"
```

3. IPv4 転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4. **nftables** サービスを有効にして起動します。

```
# systemctl enable --now nftables
```

検証

1. オプション: **nftables** ルールセットを確認します。

```
# nft list ruleset
...
```

2. ファイアウォールが阻止するアクセスの実行を試みます。たとえば、DMZ から SSH を使用してルーターにアクセスします。

```
# ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. ロギング設定に応じて、以下を検索します。

- ブロックされたパケットの **systemd** ジャーナル:

```
# journalctl -k -g "nft drop"
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

- ブロックされたパケットの **/var/log/nftables.log** ファイル:

```
Oct 14 17:27:18 router kernel: nft drop IN : IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

21.6.8. nftables を使用したポート転送の設定

ポート転送を使用すると、管理者は特定の宛先ポートに送信されたパケットを、別のローカルまたはリモートポートに転送できます。

たとえば、Web サーバーにパブリック IP アドレスがない場合は、ファイアウォールの **80** ポートおよび **443** ポートの着信パケットを Web サーバーに転送するファイアウォールのポート転送ルールを設定できます。このファイアウォールルールを使用すると、インターネットのユーザーは、ファイアウォールの IP またはホスト名を使用して Web サーバーにアクセスできます。

21.6.8.1. 着信パケットの別のローカルポートへの転送

nftables を使用してパケットを転送できます。たとえば、ポート **8022** の着信 IPv4 パケットを、ローカルシステムのポート **22** に転送できます。

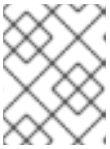
手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

```
# nft add table ip nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
```



注記

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **8022** ポートの着信パケットを、ローカルポート **22** にリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule ip nat prerouting tcp dport 8022 redirect to :22
```

21.6.8.2. 特定のローカルポートで着信パケットを別のホストに転送

宛先ネットワークアドレス変換 (DNAT) ルールを使用して、ローカルポートの着信パケットをリモートホストに転送できます。これにより、インターネット上のユーザーは、プライベート IP アドレスを持つホストで実行しているサービスにアクセスできるようになります。

たとえば、ローカルポート **443** の着信 IPv4 パケットを、IP アドレス **192.0.2.1** を持つリモートシステムの同じポート番号に転送できます。

前提条件

- パケットを転送するシステムに **root** ユーザーとしてログインしている。

手順

1. **ip** アドレスファミリーを使用して、**nat** という名前のテーブルを作成します。

```
# nft add table ip nat
```

2. テーブルに、**prerouting** チェーンおよび **postrouting** チェーンを追加します。

```
# nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
# nft add chain ip nat postrouting { type nat hook postrouting priority 100 \; }
```

注記

-- オプションを **nft** コマンドに渡して、シェルが負の priority 値を **nft** コマンドのオプションとして解釈しないようにします。

3. **443** ポートの着信パケットを **192.0.2.1** 上の同じポートにリダイレクトするルールを **prerouting** チェーンに追加します。

```
# nft add rule ip nat prerouting tcp dport 443 dnat to 192.0.2.1
```

4. 出力トラフィックをマスカレードするルールを **postrouting** チェーンに追加します。

```
# nft add rule ip nat postrouting daddr 192.0.2.1 masquerade
```

5. パケット転送を有効にします。

```
# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
# sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

21.6.9. nftables を使用した接続の量の制限

nftables を使用して、接続の数を制限したり、一定の数の接続の確立を試みる IP アドレスをブロックして、システムリソースを過剰に使用されないようにします。

21.6.9.1. nftables を使用した接続数の制限

nft ユーティリティーの **ct count** パラメーターを使用すると、管理者は接続数を制限することができます。

前提条件

- **example_table** にベースの **example_chain** が存在する。

手順

1. IPv4 アドレスの動的セットを作成します。

```
# nft add set inet example_table example_meter { type ipv4_addr\; flags dynamic \;}
```

2. IPv4 アドレスから SSH ポート (22) への 2 つの同時接続のみを許可し、同じ IP からのすべての接続を拒否するルールを追加します。

```
# nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
```

3. オプション: 前の手順で作成したセットを表示します。

```
# nft list set inet example_table example_meter
table inet example_table {
  meter example_meter {
    type ipv4_addr
    size 65535
    elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2  }
  }
}
```

elements エントリーは、現時点でルールに一致するアドレスを表示します。この例では、**elements** は、SSH ポートへのアクティブな接続がある IP アドレスを一覧表示します。出

力には、アクティブな接続の数を表示しないため、接続が拒否された場合は表示されないことに注意してください。

21.6.9.2. 1 分以内に新しい着信 TCP 接続を 11 個以上試行する IP アドレスのブロック

1 分以内に 11 個以上の IPv4 TCP 接続を確立しているホストを一時的にブロックできます。

手順

1. **ip** アドレスファミリーを使用して **filter** テーブルを作成します。

```
# nft add table ip filter
```

2. **input** チェーンを **filter** テーブルに追加します。

```
# nft add chain ip filter input { type filter hook input priority 0 \; }
```

3. 1 分以内に 10 を超える TCP 接続を確立しようとするソースアドレスからのすべてのパケットを破棄するルールを追加します。

```
# nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip saddr timeout 5m limit rate over 10/minute } drop
```

timeout 5m パラメーターは、**nftables** が、メーターが古いエントリーで一杯にならないように、5 分後にエントリーを自動的に削除することを定義します。

検証

- メーターのコンテンツを表示するには、以下のコマンドを実行します。

```
# nft list meter ip filter ratemeter
table ip filter {
  meter ratemeter {
    type ipv4_addr
    size 65535
    flags dynamic,timeout
    elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
  }
}
```

21.6.10. nftables ルールのデバッグ

nftables フレームワークは、管理者がルールをデバッグし、パケットがそれに一致するかどうかを確認するためのさまざまなオプションを提供します。

21.6.10.1. カウンターによるルールの作成

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- 既存のルールにカウンターを追加する手順の詳細は、**Configuring and managing networking** の Adding a counter to an existing rule を参照してください。

前提条件

- ルールを追加するチェーンが存在する。

手順

1. **counter** パラメーターで新しいルールをチェーンに追加します。以下の例では、ポート 22 で TCP トラフィックを許可し、このルールに一致するパケットとトラフィックをカウントするカウンターを使用するルールを追加します。

```
# nft add rule inet example_table example_chain tcp dport 22 *counter accept*
```

2. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

21.6.10.2. 既存のルールへのカウンターの追加

ルールが一致しているかどうかを確認するには、カウンターを使用できます。

- カウンターで新しいルールを追加する手順の詳細は、**Configuring and managing networking** の Creating a rule with the counter を参照してください。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールの代わりに、**counter** パラメーターを使用してカウンターを追加します。以下の例は、前の手順で表示したルールの代わりに、カウンターを追加します。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter accept
```

3. カウンター値を表示するには、次のコマンドを実行します。

```
# nft list ruleset
table inet example_table {
```

```
  chain example_chain {
    type filter hook input priority filter; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}
```

21.6.10.3. 既存のルールに一致するパケットの監視

nftables のトレース機能と、**nft monitor** コマンドを組み合わせることにより、管理者はルールに一致するパケットを表示できます。このルールに一致するパケットを監視するために、ルールのトレースを有効にできます。

前提条件

- カウンターを追加するルールがある。

手順

1. チェーンのルール (ハンドルを含む) を表示します。

```
# nft --handle list chain inet example_table example_chain
table inet example_table {
  chain example_chain { # handle 1
    type filter hook input priority filter; policy accept;
    tcp dport ssh accept # handle 4
  }
}
```

2. ルールを置き換えてトレース機能を追加しますが、**meta nftrace set 1** パラメーターを使用します。以下の例は、前の手順で表示したルールの代わりに、トレースを有効にします。

```
# nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
```

3. **nft monitor** コマンドを使用して、トレースを表示します。以下の例は、コマンドの出力をフィルタリングして、**inet example_table example_chain** が含まれるエントリーのみを表示します。

```
# nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr 52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728 tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1 accept (verdict accept)
...
```

警告

nft monitor コマンドは、トレースが有効になっているルールの数と、一致するトラフィックの量に応じて、大量の出力を表示できます。**grep** などのユーティリティーを使用して出力をフィルタリングします。

21.6.11. nftables ルールセットのバックアップおよび復元

nftables ルールをファイルにバックアップし、後で復元できます。また、管理者はルールが含まれるファイルを使用して、たとえばルールを別のサーバーに転送できます。

21.6.11.1. ファイルへの nftables ルールセットのバックアップ

nft ユーティリティーを使用して、**nftables** ルールセットをファイルにバックアップできます。

手順

- **nftables** ルールのバックアップを作成するには、次のコマンドを実行します。
  - **nft list ruleset** 形式で生成された形式の場合:

```
# nft list ruleset > file.nft
```

  - JSON 形式の場合は、以下のようになります。

```
# nft -j list ruleset > file.json
```

21.6.11.2. ファイルからの nftables ルールセットの復元

ファイルから **nftables** ルールセットを復元できます。

手順

- **nftables** ルールを復元するには、以下を行います。
  - 復元するファイルが、**nft list ruleset** が生成した形式であるか、**nft** コマンドを直接含んでいる場合は、以下のコマンドを実行します。

```
# nft -f file.nft
```

  - 復元するファイルが JSON 形式の場合は、次のコマンドを実行します。

```
# nft -j -f file.json
```

21.6.12. 関連情報

- Using nftables in Red Hat Enterprise Linux 8
- What comes after iptables?Its successor, of course: nftables

- Firewalld:The Future is nftables

パート IV. ハードディスクの設計

第22章 利用可能なファイルシステムの概要

利用可能な選択肢と、関連するトレードオフが多数あるため、アプリケーションに適したファイルシステムを選択することが重要になります。

次のセクションでは、Red Hat Enterprise Linux 8 にデフォルトで含まれるファイルシステムと、アプリケーションに最適なファイルシステムに関する推奨事項について説明します。

22.1. ファイルシステムの種類

Red Hat Enterprise Linux 8 は、さまざまなファイルシステム (FS) に対応します。さまざまな種類のファイルシステムがさまざまな問題を解決し、その使用はアプリケーションによって異なります。最も一般的なレベルでは、利用可能なファイルシステムを以下の主要なタイプにまとめることができます。

表22.1 ファイルシステムの種類とそのユースケース

タイプ	ファイルシステム	属性とユースケース
ディスクまたはローカルのファイルシステム	XFS	XFS は、RHEL におけるデフォルトファイルシステムです。ファイルをエクステントとして配置するため、ext4 と比べて断片化に対する影響はありません。Red Hat は、互換性や、パフォーマンスおいて稀に発生する難しい問題などの特別な理由がない限り、XFS をローカルファイルシステムとしてデプロイすることを推奨します。
	ext4	ext4 には、Linux の寿命が長いという利点があります。したがって、ほとんどすべての Linux アプリケーションでサポートされます。多くの場合は、パフォーマンスで XFS に匹敵します。ext4 は、一般的にホームディレクトリーに使用されます。
ネットワーク、またはクライアント/サーバーのファイルシステム	NFS	NFS は、同じネットワークにある複数のシステムでのファイル共有に使用します。
	SMB	SMB は、Microsoft Windows システムとのファイル共有に使用します。
共有ストレージまたは共有ディスクのファイルシステム	GFS2	GFS2 は、コンピュートクラスターのメンバーに共有書き込みアクセスを提供します。可能な限り、ローカルファイルシステムの機能的経験を備えた安定性と信頼性に重点が置かれています。SAS Grid、Tibco MQ、IBM Websphere MQ、および Red Hat Active MQ は、GFS2 に問題なくデプロイされています。

タイプ	ファイルシステム	属性とユースケース
ボリューム管理ファイルシステム	Stratis (テクノロジープレビュー)	Stratis は、XFS と LVM を組み合わせて構築されたボリュームマネージャーです。Stratis の目的は、Btrfs や ZFS などのボリューム管理ファイルシステムにより提供される機能をエミュレートすることです。このスタックを手動で構築することは可能ですが、Stratis は設定の複雑さを軽減し、ベストプラクティスを実装し、エラー情報を統合します。

22.2. ローカルファイルシステム

ローカルファイルシステムは、1 台のローカルサーバーで実行し、ストレージに直接接続されているファイルシステムです。

たとえば、ローカルファイルシステムは、内部 SATA ディスクまたは SAS ディスクにおける唯一の選択肢であり、ローカルドライブを備えた内蔵ハードウェア RAID コントローラーがサーバーにある場合に使用されます。ローカルファイルシステムは、SAN にエクスポートされたデバイスが共有されていない場合に、SAN が接続したストレージに最もよく使用されているファイルシステムです。

ローカルファイルシステムはすべて POSIX に準拠しており、サポートされているすべての Red Hat Enterprise Linux リリースと完全に互換性があります。POSIX 準拠のファイルシステムは、**read()**、**write()**、**seek()** など、明確に定義されたシステムコールのセットに対応します。

アプリケーションプログラマーの観点では、ローカルファイルシステム間の違いは比較的少なくなります。ユーザーの観点で最も重要な違いは、スケーラビリティーとパフォーマンスに関するものです。ファイルシステムの選択を検討する際に、ファイルシステムのサイズ、必要な固有の機能、実際のワークロードにおける実行方法を考慮してください。

利用可能なローカルファイルシステム

- XFS
- ext4

22.3. XFS ファイルシステム

XFS は、拡張性が高く、高性能で堅牢な、成熟した 64 ビットのジャーナリングファイルシステムで、1 台のホストで非常に大きなファイルおよびファイルシステムに対応します。Red Hat Enterprise Linux 8 ではデフォルトのファイルシステムになります。XFS は、元々 1990 年代の前半に SGI により開発され、極めて大規模なサーバーおよびストレージアレイで実行されてきた長い歴史があります。

XFS の機能は次のとおりです。

信頼性

- メタデータジャーナリング - システムの再起動時、およびファイルシステムの再マウント時に再生できるファイルシステム操作の記録を保持することで、システムクラッシュ後のファイルシステムの整合性を確保します。
- 広範囲に及ぶランタイムメタデータの整合性チェック
- 拡張性が高く、高速な修復ユーティリティー

- クォータジャーナリングクラッシュ後に行なわれる、時間がかかるクォータの整合性チェックが不要になります。

スケーラビリティーおよびパフォーマンス

- 対応するファイルシステムのサイズが最大 1024 TiB
- 多数の同時操作に対応する機能
- 空き領域管理のスケーラビリティーに関する B-Tree インデックス
- 高度なメタデータ先読みアルゴリズム
- ストリーミングビデオのワークロードの最適化

割り当てスキーム

- エクステント (領域) ベースの割り当て
- ストライプを認識できる割り当てポリシー
- 遅延割り当て
- 領域の事前割り当て
- 動的に割り当てられる inode

その他の機能

- Reflink ベースのファイルのコピー
- 密接に統合されたバックアップおよび復元のユーティリティー
- オンラインのデフラグ
- オンラインのファイルシステム拡張
- 包括的な診断機能
- 拡張属性 (**xattr**)。これにより、システムが、ファイルごとに、名前と値の組み合わせを追加で関連付けられるようになります。
- プロジェクトまたはディレクトリーのクォータ。ディレクトリーツリー全体にクォータ制限を適用できます。
- サブセカンド (一秒未満) のタイムスタンプ

パフォーマンスの特徴

XFS は、エンタープライズレベルのワークロードがある大規模なシステムで優れたパフォーマンスを発揮します。大規模なシステムとは、相対的に CPU 数が多く、さらには複数の HBA、および外部ディスクアレイへの接続を備えたシステムです。XFS は、マルチスレッドの並列 I/O ワークロードを備えた小規模のシステムでも適切に実行します。

XFS は、シングルスレッドで、メタデータ集約型のワークロードのパフォーマンスが比較的低くなります。たとえば、シングルスレッドで小さなファイルを多数作成し、削除するワークロードがこれに当てはまります。

22.4. EXT4 ファイルシステム

ext4 ファイルシステムは、ext ファイルシステムファミリーの第 4 世代です。これは、Red Hat Enterprise Linux 6 でデフォルトのファイルシステムです。

ext4 ドライバーは、ext2 および ext3 のファイルシステムの読み取りと書き込みが可能ですが、ext4 ファイルシステムのフォーマットは、ext2 ドライバーおよび ext3 ドライバーと互換性がありません。

ext4 には、以下のような新機能、および改善された機能が追加されました。

- 対応するファイルシステムのサイズが最大 50 TiB
- エクステントベースのメタデータ
- 遅延割り当て
- ジャーナルのチェックサム
- 大規模なストレージサポート

エクステントベースのメタデータと遅延割り当て機能は、ファイルシステムで使用されている領域を追跡する、よりコンパクトで効率的な方法を提供します。このような機能により、ファイルシステムのパフォーマンスが向上し、メタデータが使用する領域が低減します。遅延割り当てにより、ファイルシステムは、データがディスクにフラッシュされるまで、新しく書き込まれたユーザーデータの永続的な場所の選択を保留できます。これにより、より大きく、より連続した割り当てが可能になり、より優れた情報に基づいてファイルシステムが決定を下すことができるため、パフォーマンスが向上します。

ext4 で **fsck** ユーティリティーを使用するファイルシステムの修復時間は、ext2 と ext3 よりも高速です。一部のファイルシステムの修復では、最大 6 倍のパフォーマンスの向上が実証されています。

22.5. XFS と EXT4 の比較

XFS は、RHEL におけるデフォルトファイルシステムです。このセクションでは、XFS および ext4 の使用方法と機能を比較します。

メタデータエラーの動作

ext4 では、ファイルシステムがメタデータのエラーに遭遇した場合の動作を設定できます。デフォルトの動作では、操作を継続します。XFS が復旧できないメタデータエラーに遭遇すると、ファイルシステムをシャットダウンし、**EFSCORRUPTED** エラーを返します。

クォータ

ext4 では、既存のファイルシステムにファイルシステムを作成する場合にクォータを有効にできます。次に、マウントオプションを使用してクォータの適用を設定できます。
XFS クォータは再マウントできるオプションではありません。初期マウントでクォータをアクティブにする必要があります。

XFS ファイルシステムで **quotacheck** コマンドを実行すると影響しません。クォータアカウンティングを初めてオンにすると、XFS はクォータを自動的にチェックします。

ファイルシステムのサイズ変更

XFS には、ファイルシステムのサイズを縮小するユーティリティーがありません。XFS ファイルシステムのサイズのみを増やすことができます。ext4 は、ファイルシステムの拡張と縮小の両方をサポートします。

Inode 番号

ext4 ファイルシステムは、2^{32} 個を超える inode をサポートしません。
XFS は inode を動的に割り当てます。XFS ファイルシステムは、ファイルシステムに空き領域がある限り、inode からは実行できません。

特定のアプリケーションは、XFS ファイルシステムで 2^{32} 個を超える inode を適切に処理できません。このようなアプリケーションでは、戻り値 **EOVERFLOW** で 32 ビットの統計呼び出しに失敗する可能性があります。Inode 番号は、以下の条件下で 2^{32} 個を超えます。

- ファイルシステムが 256 バイトの inode を持つ 1 TiB を超える。
- ファイルシステムが 512 バイトの inode を持つ 2 TiB を超える。

inode 番号が大きくてアプリケーションが失敗した場合は、**-o inode32** オプションを使用して XFS ファイルシステムをマウントし、2^{32} 未満の inode 番号を実施します。**inode32** を使用しても、すでに 64 ビットの数値が割り当てられている inode には影響しません。

重要

特定の環境に必要な場合を除き、**inode32** オプション は**使用しないで**ください。**inode32** オプションは割り当ての動作を変更します。これにより、下層のディスクブロックに inode を割り当てるための領域がない場合に、**ENOSPC** エラーが発生する可能性があります。

22.6. ローカルファイルシステムの選択

アプリケーションの要件を満たすファイルシステムを選択するには、ファイルシステムをデプロイするターゲットシステムを理解する必要があります。以下の項目で、選択肢を確認できます。

- 大容量のサーバーがあるか
- ストレージの要件は大きいか、ローカルで低速な SATA ドライブが存在するか
- アプリケーションで期待される I/O ワークロードの種類
- スループットとレイテンシーの要件
- サーバーおよびストレージハードウェアの安定性
- ファイルとデータセットの標準的なサイズ
- システムで障害が発生した場合のダウンタイムの長さ

サーバーとストレージデバイスの両方が大きい場合は、XFS が最適です。ストレージアレイが小さくても、XFS は、平均のファイルサイズが大きい場合 (たとえば、数百メガバイト) に、非常に優れたパフォーマンスを発揮します。

既存のワークロードが ext4 で良好に機能している場合は、ext4 を引き続き使用することで、ユーザーとアプリケーションに非常に馴染みのある環境を提供できます。

ext4 ファイルシステムは、I/O 機能が制限されているシステムでパフォーマンスが向上する傾向があります。限られた帯域幅 (200MB/s 未満) と、最大約 1000 の IOPS 機能でパフォーマンスが向上します。より高い機能を備えたものであれば、XFS はより高速になる傾向があります。

XFS は、ext4 と比較して、メタデータあたりの CPU の動作を約 2 倍消費します。そのため、同時に処

理できることがほとんどない、CPU にバインドされたワークロードがあると、ext4 の方が高速になります。通常、アプリケーションが 1 つの読み取り/書き込みスレッドと小さなファイルを使用する場合は ext4 の方が優れていますが、アプリケーションが複数の読み取り/書き込みスレッドと大きなファイルを使用する場合は、XFS の方が優れています。

XFS ファイルシステムを縮小することはできません。ファイルシステムを縮小できるようにする必要がある場合は、オフライン縮小に対応する ext4 を使用することを検討してください。

通常、Red Hat は、ext4 に対する特別なユースケースがない限り、XFS を使用することを推奨します。また、ターゲットサーバーとストレージシステムで特定のアプリケーションのパフォーマンスを測定して、適切なタイプのファイルシステムを選択するようにしてください。

表22.2 ローカルファイルシステムに関する推奨事項の概要

シナリオ	推奨されるファイルシステム
特別なユースケースなし	XFS
大規模サーバー	XFS
大規模なストレージデバイス	XFS
大規模なファイル	XFS
マルチスレッド I/O	XFS
シングルスレッド I/O	ext4
制限された I/O 機能 (1000 IOPS 未満)	ext4
制限された帯域幅 (200MB/s 未満)	ext4
CPU にバインドされているワークロード	ext4
オフラインの縮小への対応	ext4

22.7. ネットワークファイルシステム

クライアント/サーバーファイルシステムとも呼ばれるネットワークファイルシステムにより、クライアントシステムは、共有サーバーに保存されているファイルにアクセスできます。これにより、複数のシステムの、複数のユーザーが、ファイルやストレージリソースを共有できます。

このようなファイルシステムは、ファイルシステムのセットを 1 つ以上のクライアントにエクスポートする、1 つ以上のサーバーから構築されます。クライアントノードは、基盤となるブロックストレージにアクセスできませんが、より良いアクセス制御を可能にするプロトコルを使用してストレージと対話します。

利用可能なネットワークファイルシステム

- RHEL で最も一般的なクライアント/サーバーファイルシステムは、NFS ファイルシステムです。RHEL は、ネットワーク経由でローカルファイルシステムをエクスポートする NFS サーバーコンポーネントと、このようなファイルシステムをインポートする NFS クライア

ントの両方を提供します。

- RHEL には、Windows の相互運用性で一般的に使用されている Microsoft SMB ファイルサーバーに対応する CIFS クライアントも含まれています。ユーザー空間 Samba サーバーは、RHEL サーバーから Microsoft SMB サービスを使用する Windows クライアントを提供します。

22.8. 共有ストレージファイルシステム

クラスターファイルシステムとも呼ばれる共有ストレージファイルシステムにより、クラスター内の各サーバーは、ローカルストレージエリアネットワーク (SAN) を介して共有ブロックデバイスに直接アクセスできます。

ネットワークファイルシステムとの比較

クライアント/サーバーのファイルシステムと同様、共有ストレージファイルシステムは、クラスターのすべてのメンバーであるサーバーのセットで機能します。ただし、NFS とは異なり、1 台のサーバーでは、その他のメンバーにデータまたはメタデータへのアクセスを提供しません。クラスターの各メンバーが同じストレージデバイス (**共有ストレージ**) に直接アクセスし、すべてのクラスターメンバーノードが同じファイルセットにアクセスできるようになります。

同時並行性

キャッシュの一貫性は、データの一貫性と整合性を確保するためにクラスター化されたファイルシステムで重要になります。クラスター内のすべてのノードに表示される、クラスター内のすべてのファイルのバージョンが 1 つ必要です。ファイルシステムは、クラスターのメンバーが同じストレージブロックを同時に更新して、データ破損を引き起こさないようにする必要があります。共有ストレージファイルシステムは、クラスター全体のロックメカニズムを使用して、同時実行制御メカニズムとしてストレージへのアクセスを調整します。たとえば、新しいファイルを作成したり、複数のサーバーで開いているファイルに書き込む前に、サーバーにあるファイルシステムコンポーネントが正しいロックを取得する必要があります。
クラスターファイルシステムの要件は、Apache Web サーバーのような可用性の高いサービスを提供することです。クラスターのすべてのメンバーに、共有ディスクのファイルシステムに保存されているデータに関する、完全に一貫した表示が提供され、すべての更新がロックメカニズムにより正しく調整されます。

パフォーマンスの特徴

共有ディスクファイルシステムは、ロックオーバーヘッドの計算コストのため、同じシステムで実行しているローカルファイルシステムと同じように機能するとは限りません。共有ディスクのファイルシステムは、各ノードが、その他のノードと共有していない特定のファイルセットにほぼ排他的に書き込むか、ファイルセットが、ノードセット間でほぼ排他的に読み取り専用で共有されるワークロードで良好に機能します。これにより、ノード間のキャッシュの無効化が最小限に抑えられ、パフォーマンスを最大化できます。
共有ディスクファイルシステムの設定は複雑で、共有ディスクのファイルシステムで適切に動作するようにアプリケーションを調整することが困難な場合があります。

利用可能な共有ストレージファイルシステム

- Red Hat Enterprise Linux は、GFS2 ファイルシステムを提供します。GFS2 は、Red Hat Enterprise Linux High Availability Add-On および Resilient Storage Add-On と密接に統合されています。

Red Hat Enterprise Linux は、サイズが 2 ノードから 16 ノードのクラスターで GFS2 に対応します。

22.9. ネットワークと共有ストレージファイルシステムの選択

ネットワークと共有ストレージのファイルシステムのいずれかを選択する際は、以下の点を考慮してください。

- NFS ベースのネットワークファイルシステムは、NFS サーバーを提供する環境において、ごく一般的で評判が良い選択肢です。
- ネットワークファイルシステムは、Infiniband や 10 ギガビットイーサネットなど、非常に高性能なネットワークテクノロジーを使用してデプロイできます。これは、ストレージに、生の帯域幅を取得するだけのために、共有ストレージのファイルシステムを有効にすべきではないことを意味します。アクセスの速度が非常に重要な場合は、NFS を使用して、XFS などのローカルファイルシステムをエクスポートします。
- 共有ストレージのファイルシステムは、設定や維持が容易ではないため、ローカルまたはネットワークのファイルシステムのいずれかで必要な可用性を提供できない場合に限りデプロイしてください。
- クラスター環境の共有ストレージのファイルシステムは、高可用性サービスの再配置を伴う一般的なフェイルオーバーシナリオで、マウント解除およびマウントに必要な手順を省くことで、ダウンタイムを短縮できます。

Red Hat は、共有ストレージのファイルシステムに対する特別なユースケースがない限り、ネットワークのファイルシステムを使用することを推奨します。共有ストレージのファイルシステムは、主に、最小限のダウンタイムで高可用性サービスを提供する必要があり、サービスレベルの要件が厳しいデプロイメントに使用します。

22.10. ボリューム管理ファイルシステム

ボリューム管理ファイルシステムは、簡素化とスタック内の最適化の目的で、ストレージスタック全体を統合します。

利用可能なボリューム管理ファイルシステム

- Red Hat Enterprise Linux 8 は、Stratis ボリュームマネージャーがテクノロジープレビューとして提供します。Stratis は、ファイルシステム層に XFS を使用し、LVM、Device Mapper、およびその他のコンポーネントと統合します。

Stratis は、Red Hat Enterprise Linux 8.0 で初めてリリースされました。Red Hat が Btrfs を非推奨にした時に生じたギャップを埋めると考えられています。Stratis 1.0 は、ユーザーによる複雑さを隠しつつ、重要なストレージ管理操作を実行できる直感的なコマンドラインベースのボリュームマネージャーです。

- ボリュームの管理
- プールの作成
- シンストレージプール
- スナップショット
- 自動化読み取りキャッシュ

Stratis は強力な機能を提供しますが、現時点では Btrfs や ZFS といったその他の製品と比較される可能性がある機能をいつくか欠いています。たとえば、セルフ修復を含む CRC には対応していません。

第23章 NFS 共有のマウント

システム管理者は、システムにリモート NFS 共有をマウントすると、共有データにアクセスできます。

23.1. NFS の概要

本セクションでは、NFS サービスの基本概念を説明します。

ネットワークファイルシステム (NFS) を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムと同じように操作できるようになります。また、リソースを、ネットワークの集中化サーバーに統合できるようになります。

NFS サーバーは、**/etc/exports** 設定ファイルを参照して、そのクライアントがエクスポート済みファイルシステムにアクセスできるかどうかを確認します。アクセスが可能なことが確認されると、そのユーザーは、ファイルおよびディレクトリーへの全操作を行えるようになります。

23.2. 対応している NFS バージョン

本セクションでは、Red Hat Enterprise Linux でサポートされている NFS のバージョンと、その機能の一覧を紹介します。

現在、Red Hat Enterprise Linux 8 は、以下の NFS のメジャーバージョンに対応しています。

- NFS バージョン 3 (NFSv3) は安全な非同期書き込みに対応しており、以前の NFSv2 よりもエラー処理において安定しています。64 ビットのファイルサイズとオフセットにも対応しているため、クライアントは 2 GB を超えるファイルデータにアクセスできます。
- NFS バージョン 4 (NFSv4) は、ファイアウォールやインターネットを介して動作し、**rpcbind** サービスを必要とせず、アクセス制御リスト (ACL) に対応し、ステートフルな操作を利用します。

NFS バージョン 2 (NFSv2) は、Red Hat のサポート対象外になりました。

デフォルトの NFS バージョン

Red Hat Enterprise Linux 8 のデフォルトの NFS バージョンは 4.2 です。NFS クライアントは、デフォルトで NFSv4.2 を使用してマウントを試行し、サーバーが NFSv4.2 に対応していない場合は NFSv4.1 にフォールバックします。マウントは後で NFSv4.0 に戻り、次に NFSv3 に戻ります。

NFS のマイナーバージョンの機能

以下は、Red Hat Enterprise Linux 8 における NFSv4.2 の機能です。

サーバー側コピー

NFS クライアントが **copy_file_range()** システムコールを使用してネットワークリソースを無駄にすることなく、データを効率的にコピーできるようにします。

スパースファイル

ファイルに 1 つ以上の **ホール** を持たせることができます。ホールとは、割り当てられていない、またはゼロのみで設定される未初期化データブロックです。NFSv4.2 の **lseek()** 操作は **seek_hole()** と **seek_data()** に対応しています。これにより、アプリケーションはスパースファイルのホールの場所をマップできます。

領域の予約

ストレージサーバーが空き領域を予約することを許可します。これにより、サーバーで領域が不足

することがなくなります。NFSv4.2 は、領域を予約するための **allocate()** 操作、領域の予約を解除するための **deallocate()** 操作、およびファイル内の領域の事前割り当てまたは割り当て解除を行う **fallocate()** 操作に対応しています。

ラベル付き NFS

データアクセス権を強制し、NFS ファイルシステム上の個々のファイルに対して、クライアントとサーバーとの間の SELinux ラベルを有効にします。

レイアウトの機能強化

一部の Parallel NFS (pNFS) サーバーがより良いパフォーマンス統計を収集できるようにする **layoutstats()** 操作が提供されます。

NFSv4.1 の機能は次のとおりです。

- ネットワークのパフォーマンスおよびセキュリティーを強化し、pNFS のクライアント側サポートも含みます。
- コールバックに個別の TCP 接続を必要としなくなりました。これにより、NAT やファイアウォールが干渉した場合など、クライアントと通信できない場合でも NFS サーバーは委任を許可できます。
- 応答が失われ、操作が 2 回送信された場合に特定の操作が不正確な結果を返すことがあるという以前の問題を防ぐために、1 回限りのセマンティクスを提供します (再起動操作を除く)。

23.3. NFS が必要とするサービス

本セクションでは、NFS サーバーの実行または NFS 共有のマウントに必要なシステムサービスのリストを紹介します。Red Hat Enterprise Linux は、このサービスを自動的に開始します。

Red Hat Enterprise Linux では、NFS ファイル共有を提供するのに、カーネルレベルのサポートとサービスのプロセスの組み合わせを使用します。NFS のすべてのバージョンは、クライアントとサーバーとの間の RPC (Remote Procedure Call) に依存します。NFS ファイルシステムの共有やマウントには、実装されている NFS のバージョンに応じて、次のようなサービスが連携して動作することになります。

nfsd

共有 NFS ファイルシステムに対する要求を処理する NFS サーバーカーネルモジュールです。

rpcbind

ローカルの RPC サービスからポート予約を受け取ります。その後、これらのポートは、対応するリモートの RPC サービスによりアクセス可能であることが公開されます。**rpcbind** サービスは、RPC サービスへの要求に応答し、要求された RPC サービスへの接続を設定します。このプロセスは NFSv4 では使用されません。

rpc.mountd

NFS サーバーは、このプロセスを使用して NFSv3 クライアントの **MOUNT** 要求を処理します。要求されている NFS 共有が現在 NFS サーバーによりエクスポートされているか、またその共有へのクライアントのアクセスが許可されているかを確認します。マウントの要求が許可されると、**nfs-mountd** サービスは Success ステータスで応答し、この NFS 共有用の File-Handle を NFS クライアントに戻します。

rpc.nfsd

このプロセスでは、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な要求に対応するため、Linux カーネルと連携して動作します。このプロセスは、**nfs-server** サービスに対応します。

lockd

クライアントとサーバーの両方で実行するカーネルスレッドです。Network Lock Manager (NLM) プロトコルを実装し、NFSv3 のクライアントが、サーバーでファイルのロックを行えるようにします。NFS サーバーが実行中で、NFS ファイルシステムがマウントされていれば、このプロセスは常に自動的に起動します。

rpc.statd

このプロセスは、Network Status Monitor (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされずに再起動すると、NFS クライアントに通知します。**rpc-statd** サービスは、**nfs-server** サービスにより自動的に起動されるため、ユーザー設定は必要ありません。このプロセスは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォーター情報を提供します。**quota-rpc** パッケージが提供する **rpc-rquotad** サービスは、**nfs-server** の起動時にユーザーが起動する必要があります。

rpc.idmapd

このプロセスは、ネットワークの NFSv4 の名前 (**user@domain** 形式の文字列) と、ローカルの UID および GID のマッピングを行う NFSv4 のクライアントおよびサーバーのアップコールを提供します。**idmapd** を NFSv4 で正常に動作させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。少なくとも、NFSv4 マッピングドメインを定義する **Domain** パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じ場合は、このパラメーターは必要ありません。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに合意しないと、適切に動作しません。
rpc.idmapd を使用するのは NFSv4 サーバーだけで、 **nfs-idmapd** サービスにより起動します。NFSv4 クライアントは、キーリングベースの **nfsidmap** ユーティリティーを使用します。これはカーネルによりオンデマンドで呼び出され、ID マッピングを実行します。**nfsidmap** に問題がある場合は、クライアントが **rpc.idmapd** の使用にフォールバックします。

NFSv4 を使用する RPC サービス

NFSv4 プロトコルには、マウントとロックのプロトコルが組み込まれています。サーバーは、既知の TCP ポート 2049 もリッスンします。そのため、NFSv4 は **rpcbind** サービス、**lockd** サービス、および **rpc-statd** サービスと対話する必要はありません。**nfs-mountd** サービスは、エクスポートを設定するために NFS サーバーで引き続き必要となりますが、ネットワーク上の操作には関与しません。

関連情報

- **rpcbind** を使用しない NFSv4 のみのサーバーの設定

23.4. NFS ホスト名の形式

本セクションでは、NFS 共有をマウントまたはエクスポートするときにホストの指定に使用するさまざまな形式を説明します。

次の形式でホストを指定できます。

単独のマシン

次のいずれかになります。

- 完全修飾ドメイン名 (サーバーにより解決)
- ホスト名 (サーバーにより解決)
- IP アドレス

IP ネットワーク

以下のいずれかの形式が有効です。

- **a.b.c.d/z** - **a.b.c.d** がネットワークで、**z** がネットマスクのビット数になります (例: **192.168.0.0/24**)。
- **a.b.c.d/netmask** - **a.b.c.d** がネットワークで、**netmask** がネットマスクになります (例: **192.168.100.8/255.255.255.0**)。

Netgroup

@group-name 形式 - **group-name** は NIS netgroup 名です。

23.5. NFS のインストール

この手順では、NFS 共有のマウントまたはエクスポートに必要なすべてのパッケージをインストールします。

手順

- **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

23.6. NFS エクスポートの検出

この手順では、特定の NFSv3 または NFSv4 サーバーがエクスポートしているファイルシステムを検出します。

手順

- NFSv3 に対応しているサーバーで、**showmount** ユーティリティーを使用します。

```
$ showmount --exports my-server

Export list for my-server
/exports/foo
/exports/bar
```

- NFSv4 に対応しているサーバーで、root ディレクトリーをマウントして確認します。

```
# mount my-server:/ /mnt/
# ls /mnt/

exports

# ls /mnt/exports/

foo
bar
```

NFSv4 と NFSv3 の両方に対応するサーバーでは、上記の方法はいずれも有効で、同じ結果となります。

関連情報

- **showmount(8)** man ページ

23.7. MOUNT で NFS 共有のマウント

mount ユーティリティーを使用して、サーバーからエクスポートされた NFS 共有をマウントします。

警告

NFS クライアントが同じ短いホスト名を使用している場合は、NFSv4 **clientid** で競合が発生し、それらが突然期限切れになることがあります。NFSv4 **clientid** が突然期限切れになる可能性を回避するには、使用しているシステムに応じて、NFS クライアントに一意のホスト名を使用するか、各コンテナーで識別子を設定する必要があります。詳細については、ナレッジベース記事 NFSv4 clientid was expired suddenly due to use same hostname on several NFS clients を参照してください。

手順

- NFS 共有をマウントするには、次のコマンドを使用します。

```
# mount -t nfs -o options host:/remote/export /local/directory
```

このコマンドは、以下のような変数を使用します。

options
: マウントオプションのコンマ区切りリスト

host
: マウントするファイルシステムをエクスポートするサーバーのホスト名、IP アドレス、または完全修飾ドメイン名。

/remote/export
: サーバーからエクスポートされるファイルシステムまたはディレクトリー、つまりマウントするディレクトリー。

/local/directory
: **/remote/export** がマウントされているクライアントの場所

関連情報

- 一般的な NFS マウントオプション
- NFS ホスト名の形式
- mount でファイルシステムのマウント
- **mount(8)** の man ページ
- **exports(5)** man ページ

23.8. 一般的な NFS マウントオプション

以下は、NFS 共有をマウントするときに一般的に使用されるオプションです。これらのオプションは、手動 **mount** コマンド、**/etc/fstab** 設定、および **autofs** で使用できます。

lookupcache=mode

任意のマウントポイントに対して、カーネルがディレクトリーエントリーのキャッシュを管理する方法を指定します。**mode** の有効な引数は、**all**、**none**、または **positive** です。

nfsvers=version

使用する NFS プロトコルのバージョンを指定します。**version** は、**3**、**4**、**4.0**、**4.1**、または **4.2** になります。これは、複数の NFS サーバーを実行しているホストや、より低いバージョンでのマウントの再試行を無効にするのに役立ちます。バージョンを指定しないと、NFS により、カーネルと **mount** ユーティリティーで対応している最新バージョンが使用されます。
vers オプションは **nfsvers** と同じで、互換性のためにこのリリースに含まれています。

noacl

ACL の処理をすべてオフにします。古いバージョンの Red Hat Enterprise Linux、Red Hat Linux、または Solaris と連動させる場合に必要となることがあります。こうした古いシステムには、最新の ACL テクノロジーに対する互換性がないためです。

nolock

ファイルのロック機能を無効にします。この設定は、非常に古いバージョンの NFS サーバーに接続する場合に必要となる場合があります。

noexec

マウントしたファイルシステムでバイナリーが実行されないようにします。互換性のないバイナリーを含む、Linux 以外のファイルシステムをマウントしている場合に便利です。

nosuid

set-user-identifier ビットおよび **set-group-identifier** ビットを無効にします。これにより、リモートユーザーは、**setuid** プログラムを実行してより高い権限を取得できなくなります。

port=num

NFS サーバーポートの数値を指定します。**num** が **0** (デフォルト値) の場合、**mount** は、使用するポート番号を、リモートホストの **rpcbind** サービスに問い合わせます。リモートホストの NFS サービスがその **rpcbind** サービスに登録されていない場合は、代わりに TCP 2049 の標準 NFS ポート番号が使用されます。

rsize=num and **wsize=num**

このオプションは、1 回の NFS 読み取り操作または書き込み操作で転送される最大バイト数を設定します。
rsize と **wsize** には、固定のデフォルト値がありません。デフォルトでは、NFS はサーバーとクライアントの両方がサポートしている最大の値を使用します。Red Hat Enterprise Linux 8 では、クライアントとサーバーの最大値は 1,048,576 バイトです。詳細は、What are the default and maximum values for rsize and wsize with NFS mounts? 参照してください。KBase の記事。

sec=flavors

マウントされたエクスポート上のファイルにアクセスするために使用するセキュリティーフレーバーです。**flavors** の値は、複数のセキュリティーフレーバーのコロンで区切られたリストです。
デフォルトでは、クライアントは、クライアントとサーバーの両方をサポートするセキュリティーフレーバーの検索を試みます。サーバーが選択したフレーバーのいずれかに対応していない場合、マウント操作は失敗します。

利用可能なフレーバー:

- **sec=sys** は、ローカルの UNIX UID および GID を使用します。 **AUTH_SYS** を使用して NFS 操作を認証します。
- **sec=krb5** は、ユーザー認証に、ローカルの UNIX の UID と GID ではなく、Kerberos V5 を使用します。
- **sec=krb5i** は、ユーザー認証に Kerberos V5 を使用し、データの改ざんを防ぐ安全なチェックサムを使用して、NFS 操作の整合性チェックを行います。
- **sec=krb5p** は、ユーザー認証に Kerberos V5 を使用し、整合性チェックを実行し、トラフィックの傍受を防ぐため NFS トラフィックの暗号化を行います。これが最も安全な設定になりますが、パフォーマンスのオーバーヘッドも最も高くなります。

tcp

NFS マウントが TCP プロトコルを使用するよう指示します。

関連情報

- **mount(8)** の man ページ
- **nfs(5)** man ページ

23.9. 関連情報

- Linux NFS wiki
- NFS 共有の永続的なマウント
- オンデマンドで NFS 共有のマウント

第24章 NFS 共有のエクスポート

システム管理者は、NFS サーバーを使用して、ネットワーク上のシステムのディレクトリーを共有できます。

24.1. NFS の概要

本セクションでは、NFS サービスの基本概念を説明します。

ネットワークファイルシステム (NFS) を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムと同じように操作できるようになります。また、リソースを、ネットワークの集中化サーバーに統合できるようになります。

NFS サーバーは、**/etc/exports** 設定ファイルを参照して、そのクライアントがエクスポート済みファイルシステムにアクセスできるかどうかを確認します。アクセスが可能なことが確認されると、そのユーザーは、ファイルおよびディレクトリーへの全操作を行えるようになります。

24.2. 対応している NFS バージョン

本セクションでは、Red Hat Enterprise Linux でサポートされている NFS のバージョンと、その機能の一覧を紹介します。

現在、Red Hat Enterprise Linux 8 は、以下の NFS のメジャーバージョンに対応しています。

- NFS バージョン 3 (NFSv3) は安全な非同期書き込みに対応しており、以前の NFSv2 よりもエラー処理において安定しています。64 ビットのファイルサイズとオフセットにも対応しているため、クライアントは 2 GB を超えるファイルデータにアクセスできます。
- NFS バージョン 4 (NFSv4) は、ファイアウォールやインターネットを介して動作し、**rpcbind** サービスを必要とせず、アクセス制御リスト (ACL) に対応し、ステートフルな操作を利用します。

NFS バージョン 2 (NFSv2) は、Red Hat のサポート対象外になりました。

デフォルトの NFS バージョン

Red Hat Enterprise Linux 8 のデフォルトの NFS バージョンは 4.2 です。NFS クライアントは、デフォルトで NFSv4.2 を使用してマウントを試行し、サーバーが NFSv4.2 に対応していない場合は NFSv4.1 にフォールバックします。マウントは後で NFSv4.0 に戻り、次に NFSv3 に戻ります。

NFS のマイナーバージョンの機能

以下は、Red Hat Enterprise Linux 8 における NFSv4.2 の機能です。

サーバー側コピー

NFS クライアントが **copy_file_range()** システムコールを使用してネットワークリソースを無駄にすることなく、データを効率的にコピーできるようにします。

スパースファイル

ファイルに 1 つ以上の **ホール** を持たせることができます。ホールとは、割り当てられていない、またはゼロのみで設定される未初期化データブロックです。NFSv4.2 の **lseek()** 操作は **seek_hole()** と **seek_data()** に対応しています。これにより、アプリケーションはスパースファイルのホールの場所をマップできます。

領域の予約

ストレージサーバーが空き領域を予約することを許可します。これにより、サーバーで領域が不足

することがなくなります。NFSv4.2 は、領域を予約するための **allocate()** 操作、領域の予約を解除するための **deallocate()** 操作、およびファイル内の領域の事前割り当てまたは割り当て解除を行う **fallocate()** 操作に対応しています。

ラベル付き NFS

データアクセス権を強制し、NFS ファイルシステム上の個々のファイルに対して、クライアントとサーバーとの間の SELinux ラベルを有効にします。

レイアウトの機能強化

一部の Parallel NFS (pNFS) サーバーがより良いパフォーマンス統計を収集できるようにする **layoutstats()** 操作が提供されます。

NFSv4.1 の機能は次のとおりです。

- ネットワークのパフォーマンスおよびセキュリティーを強化し、pNFS のクライアント側サポートも含みます。
- コールバックに個別の TCP 接続を必要としなくなりました。これにより、NAT やファイアウォールが干渉した場合など、クライアントと通信できない場合でも NFS サーバーは委任を許可できます。
- 応答が失われ、操作が 2 回送信された場合に特定の操作が不正確な結果を返すことがあるという以前の問題を防ぐために、1 回限りのセマンティクスを提供します (再起動操作を除く)。

24.3. NFSV3 と NFSV4 の TCP プロトコルと UDP プロトコル

NFSv4 は、IP ネットワークで TCP (Transmission Control Protocol) の実行が必要です。

NFSv3 は、Red Hat Enterprise Linux の以前のバージョンで User Datagram Protocol (UDP) を使用することもできます。Red Hat Enterprise Linux 8 では、NFS over UDP に対応しなくなりました。デフォルトでは、UDP は、NFS サーバーで無効になります。

24.4. NFS が必要とするサービス

本セクションでは、NFS サーバーの実行または NFS 共有のマウントに必要なシステムサービスのリストを紹介します。Red Hat Enterprise Linux は、このサービスを自動的に開始します。

Red Hat Enterprise Linux では、NFS ファイル共有を提供するのに、カーネルレベルのサポートとサービスのプロセスの組み合わせを使用します。NFS のすべてのバージョンは、クライアントとサーバーとの間の RPC (Remote Procedure Call) に依存します。NFS ファイルシステムの共有やマウントには、実装されている NFS のバージョンに応じて、次のようなサービスが連携して動作することになります。

nfsd

共有 NFS ファイルシステムに対する要求を処理する NFS サーバーカーネルモジュールです。

rpcbind

ローカルの RPC サービスからポート予約を受け取ります。その後、これらのポートは、対応するリモートの RPC サービスによりアクセス可能であることが公開されます。**rpcbind** サービスは、RPC サービスへの要求に応答し、要求された RPC サービスへの接続を設定します。このプロセスは NFSv4 では使用されません。

rpc.mountd

NFS サーバーは、このプロセスを使用して NFSv3 クライアントの **MOUNT** 要求を処理します。要求されている NFS 共有が現在 NFS サーバーによりエクスポートされているか、またその共有へのクライアントのアクセスが許可されているかを確認します。マウントの要求が許可されると、**nfs-**

mountd サービスは Success ステータスで応答し、この NFS 共有用の File-Handle を NFS クライアントに戻します。

rpc.nfsd

このプロセスでは、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な要求に対応するため、Linux カーネルと連携して動作します。このプロセスは、**nfs-server** サービスに対応します。

lockd

クライアントとサーバーの両方で実行するカーネルスレッドです。Network Lock Manager (NLM) プロトコルを実装し、NFSv3 のクライアントが、サーバーでファイルのロックを行えるようにします。NFS サーバーが実行中で、NFS ファイルシステムがマウントされていれば、このプロセスは常に自動的に起動します。

rpc.statd

このプロセスは、Network Status Monitor (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされずに再起動すると、NFS クライアントに通知します。**rpc-statd** サービスは、**nfs-server** サービスにより自動的に起動されるため、ユーザー設定は必要ありません。このプロセスは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォーター情報を提供します。**quota-rpc** パッケージが提供する **rpc-rquotad** サービスは、**nfs-server** の起動時にユーザーが起動する必要があります。

rpc.idmapd

このプロセスは、ネットワークの NFSv4 の名前 (**user@domain** 形式の文字列) と、ローカルの UID および GID のマッピングを行う NFSv4 のクライアントおよびサーバーのアップコールを提供します。**idmapd** を NFSv4 で正常に動作させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。少なくとも、NFSv4 マッピングドメインを定義する **Domain** パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じ場合は、このパラメーターは必要ありません。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに合意しないと、適切に動作しません。

rpc.idmapd を使用するのは NFSv4 サーバーだけで、**nfs-idmapd** サービスにより起動します。NFSv4 クライアントは、キーリングベースの **nfsidmap** ユーティリティーを使用します。これはカーネルによりオンデマンドで呼び出され、ID マッピングを実行します。**nfsidmap** に問題がある場合は、クライアントが **rpc.idmapd** の使用にフォールバックします。

NFSv4 を使用する RPC サービス

NFSv4 プロトコルには、マウントとロックのプロトコルが組み込まれています。サーバーは、既知の TCP ポート 2049 もリッスンします。そのため、NFSv4 は **rpcbind** サービス、**lockd** サービス、および **rpc-statd** サービスと対話する必要はありません。**nfs-mountd** サービスは、エクスポートを設定するために NFS サーバーで引き続き必要となりますが、ネットワーク上の操作には関与しません。

関連情報

- **rpcbind** を使用しない NFSv4 のみのサーバーの設定

24.5. NFS ホスト名の形式

本セクションでは、NFS 共有をマウントまたはエクスポートするときにホストの指定に使用するさまざまな形式を説明します。

次の形式でホストを指定できます。

単独のマシン

次のいずれかになります。

- 完全修飾ドメイン名 (サーバーにより解決)
- ホスト名 (サーバーにより解決)
- IP アドレス

IP ネットワーク

以下のいずれかの形式が有効です。

- **a.b.c.d/z** - **a.b.c.d** がネットワークで、**z** がネットマスクのビット数になります (例: **192.168.0.0/24**)。
- **a.b.c.d/netmask** - **a.b.c.d** がネットワークで、**netmask** がネットマスクになります (例: **192.168.100.8/255.255.255.0**)。

Netgroup

@group-name 形式 - **group-name** は NIS netgroup 名です。

24.6. NFS サーバーの設定

本セクションでは、NFS サーバーでエクスポートを設定する 2 種類の構文およびオプションを説明します。

- 設定ファイル **/etc/exports** を手動で編集する方法
- コマンドラインで **exportfs** ユーティリティーを使用する方法

24.6.1. /etc/exports 設定ファイル

/etc/exports ファイルは、リモートホストにどのファイルシステムをエクスポートするかを制御し、オプションを指定します。以下の構文ルールに従います。

- 空白行は無視する。
- コメント行は、ハッシュ記号 (**#**) で始める。
- 長い行は、バックスラッシュ (\) で改行できる。
- エクスポートするファイルシステムは、それぞれ 1 行で指定する。
- 許可するホストのリストは、エクスポートするファイルシステムの後に空白文字を追加し、その後に追加する。
- 各ホストのオプションは、ホストの識別子の直後に括弧を追加し、その中に指定する。ホストと最初の括弧の間には空白を使用しない。

エクスポートエントリー
エクスポートするファイルシステムの各エントリーは、以下のように指定します。

```
export host(options)
```

各ホストにそれぞれオプションを付けて、複数のホストを 1 行で指定することもできます。この場合は、以下のように、各ホスト名の後に、そのホストに対するオプションを括弧を付けて追加します。ホストは空白文字で区切ります。

```
export host1(options1) host2(options2) host3(options3)
```

この構造では、次のようになります。

export
エクスポートするディレクトリー

host
エクスポートを共有するホストまたはネットワーク

options
ホストに使用されるオプション

例24.1簡潔な /etc/exports ファイル

最も簡単な方法は、**/etc/exports** ファイルに、エクスポートするディレクトリーと、そのディレクトリーへのアクセスを許可するホストを指定することです。

```
/exported/directory bob.example.com
```

ここで、**bob.example.com** は、NFS サーバーから **/exported/directory/** をマウントできます。この例ではオプションが指定されていないため、NFS はデフォルトのオプションを使用します。

重要

/etc/exports ファイルの形式では、特に空白文字の使用が非常に厳しく扱われます。ホストからエクスポートするファイルシステムの間、そしてホスト同士の間には、必ず空白文字を挿入してください。また、それ以外の場所 (コメント行を除く) には、空白文字を追加しないでください。

たとえば、以下の 2 つの行は意味が異なります。

```
/home bob.example.com(rw)
/home bob.example.com (rw)
```

最初の行は、**bob.example.com** からのユーザーにのみ、**/home** ディレクトリーへの読み取り/書き込みアクセスを許可します。2 番目の行では、**bob.example.com** からのユーザーにディレクトリーを読み取り専用 (デフォルト) でマウントすることを許可し、その他のユーザーに読み取り/書き込みでマウントすることを許可します。

デフォルトのオプション
エクスポートエントリーのデフォルトオプションは次のとおりです。

ro
エクスポートするファイルシステムは読み取り専用です。リモートホストは、このファイルシステムで共有されているデータを変更できません。このファイルシステムで変更 (読み取り/書き込み) を可能にするには、rw オプションを指定します。

sync

NFS サーバーは、以前の要求で発生した変更がディスクに書き込まれるまで、要求に応答しません。代わりに非同期書き込みを有効にするには、**async** オプションを指定します。

wdelay

NFS サーバーは、別の書き込み要求が差し迫っていると判断すると、ディスクへの書き込みを遅らせます。これにより、複数の書き込みコマンドが同じディスクにアクセスする回数を減らすことができるため、書き込みのオーバーヘッドが低下し、パフォーマンスが向上します。これを無効にするには、**no_wdelay** オプションを指定します。これは、デフォルトの sync オプションが指定されている場合に限り利用できます。

root_squash

(ローカルからではなく) リモートから接続している root ユーザーが root 権限を持つことを阻止します。代わりに、そのユーザーには、NFS サーバーにより、ユーザー ID **nobody** が割り当てられます。これにより、リモートの root ユーザーの権限を、最も低いローカルユーザーレベルにまで下げて (squash)、高い確率でリモートサーバーへの書き込む権限を与えないようにすることができます。この root squashing を無効にするには、**no_root_squash** オプションを指定します。
(root を含む) すべてのリモートユーザーの権限を下げるには、**all_squash** オプションを使用します。特定ホストのリモートユーザーに対して、NFS サーバーが割り当てるユーザー ID とグループ ID を指定するには、**anonuid** オプションと **anongid** オプションを以下のように使用します。

```
export host(anonuid=uid,anongid=gid)
```

uid と **gid** は、それぞれユーザー ID とグループ ID の番号になります。 **anonuid** オプションと **anongid** オプションにより、共有するリモート NFS ユーザー用に、特別なユーザーアカウントおよびグループアカウントを作成できます。

デフォルトでは、アクセス制御リスト (ACL) は、Red Hat Enterprise Linux では NFS が対応しています。この機能を無効にするには、ファイルシステムをエクスポートする際に **no_acl** オプションを指定します。

デフォルトオプションと上書きオプション
エクスポートするすべてのファイルシステムの各デフォルトは、明示的に上書きする必要があります。たとえば、**rw** オプションを指定しないと、エクスポートするファイルシステムが読み取り専用として共有されます。以下は、**/etc/exports** の例になりますが、ここでは 2 つのデフォルトオプションを上書きします。

```
/another/exported/directory 192.168.0.3(rw,async)
```

この例では、**192.168.0.3** は **/another/exported/directory/** の読み書きをマウントでき、ディスクへの書き込みはすべて非同期になります。

24.6.2. exportfs ユーティリティー

root ユーザーは、**exportfs** ユーティリティーを使用すると、NFS サービスを再起動せずにディレクトリーを選択してエクスポートまたはアンエクスポートできます。適切なオプションが指定されると、**exportfs** ユーティリティーは、エクスポートされたファイルシステムを **/var/lib/nfs/xtab** に書き込みます。ファイルシステムへのアクセス権を決定する際には、**nfs-mountd** サービスが **xtab** ファイルを参照するため、エクスポートしたファイルシステムのリストの変更が直ちに反映されます。

一般的な exportfs オプション
exportfs で利用できる一般的なオプションのリストは以下のようになります。

-r

/var/lib/nfs/etab に新しいエクスポートリストを作成して、**/etc/exports** にリスト表示されているディレクトリーをすべてエクスポートします。このオプションにより、**/etc/exports** に変更が加えられると、エクスポートリストが効果的に更新されます。

-a

exportfs に渡されるその他のオプションに応じて、すべてのディレクトリーをエクスポートするかどうかを判断します。その他のオプションが指定されていないと、**exportfs** は、**/etc/exports** で指定されたすべてのファイルシステムをエクスポートします。

-o file-systems

/etc/exports 内に記載されていない、エクスポートされるディレクトリーを指定します。**file-systems** の部分を、エクスポートされる追加のファイルシステムに置き換えます。これらのファイルシステムは、**/etc/exports** で指定されたものと同じフォーマットでなければなりません。このオプションは、多くの場合、エクスポートされるファイルシステムのリストに永続的に追加する前に、エクスポートされるファイルシステムをテストするために使用されます。

-i

/etc/exports を無視します。コマンドラインで指定されたオプションのみが、エクスポート用ファイルシステムの定義に使用されます。

-u

すべての共有ディレクトリーをエクスポートしなくなります。**exportfs -ua** コマンドは、すべての NFS サービスを稼働状態に維持しながら、NFS ファイル共有を保留します。NFS 共有を再度有効にするには、**exportfs -r** を使用します。

-v

詳細な表示です。**exportfs** コマンドを実行するときに表示されるエクスポート、または非エクスポートのファイルシステムの情報が、より詳細に表示されます。

exportfs ユーティリティーにオプションが渡されていない場合は、現在エクスポートされているファイルシステムのリストが表示されます。

関連情報

- NFS ホスト名の形式

24.7. NFS および RPCBIND

rpcbind サービスは、RPC (Remote Procedure Call) サービスを、そのサービスがリッスンするポートにマッピングします。RPC のプロセスが開始すると、その開始が **rpcbind** に通知され、そのプロセスがリッスンしているポートと、そのプロセスが処理することが予想される RPC プログラム番号が登録されます。クライアントシステムは、特定の RPC プログラム番号でサーバーの **rpcbind** と通信します。**rpcbind** サービスは、クライアントを適切なポート番号にリダイレクトし、要求されたサービスと通信できるようにします。

ネットワークファイルシステムバージョン 3 (NFSv3)には **rpcbind** サービスが必要です。

RPC ベースのサービスは、**rpcbind** を使用して、クライアントの受信要求ですべての接続を確立します。したがって、RPC ベースのサービスが起動する前に、**rpcbind** を利用可能な状態にする必要があります。

rpcbind のアクセス制御ルールは、すべての RPC ベースのサービスに影響します。あるいは、NFS RPC デーモンごとにアクセス制御ルールを指定することもできます。

関連情報

- **rpc.mountd(8)** man ページ

- **rpc.statd(8)** man ページ

24.8. NFS のインストール

この手順では、NFS 共有のマウントまたはエクスポートに必要なすべてのパッケージをインストールします。

手順

- **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

24.9. NFS サーバーの起動

この手順では、NFS 共有をエクスポートするために必要な NFS サーバーの起動方法を説明します。

前提条件

- NFSv3 の接続に対応しているサーバーで、**rpcbind** サービスを実行している。**rpcbind** がアクティブであることを確認するには、次のコマンドを実行します。

```
$ systemctl status rpcbind
```

サービスが停止している場合は、起動して有効にします。

```
$ systemctl enable --now rpcbind
```

手順

- システムの起動時に、NFS サーバーを起動して自動的に起動するようにするには、次のコマンドを使用します。

```
# systemctl enable --now nfs-server
```

関連情報

- NFSv4 専用サーバーの設定

24.10. NFS と RPCBIND のトラブルシューティング

rpcbind サービスでは通信に使用するポート番号と RPC サービス間の調整を行うため、トラブルシューティングを行う際は、**rpcbind** を使用して現在の RPC サービスの状態を表示させると便利です。**rpcinfo** ユーティリティーは、RPC ベースの各サービスとそのポート番号、RPC プログラム番号、バージョン番号、および IP プロトコルタイプ (TCP または UDP) が表示されます。

手順

1. **rpcbind** に対して適切な RPC ベースの NFS サービス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。

```
# rpcinfo -p
```

例24.2 rpcinfo -p コマンドの出力

以下に、上記コマンドの出力例を示します。

```
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp  20048  mountd
    100005    1   tcp  20048  mountd
    100005    2   udp  20048  mountd
    100005    2   tcp  20048  mountd
    100005    3   udp  20048  mountd
    100005    3   tcp  20048  mountd
    100024    1   udp  37769  status
    100024    1   tcp  49349  status
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049  nfs_acl
    100021    1   udp  56691  nlockmgr
    100021    3   udp  56691  nlockmgr
    100021    4   udp  56691  nlockmgr
    100021    1   tcp  46193  nlockmgr
    100021    3   tcp  46193  nlockmgr
    100021    4   tcp  46193  nlockmgr
```

NFS サービスの 1 つが正しく起動しないと、**rpcbind** は、そのサービスに対するクライアントからの RPC 要求を、正しいポートにマッピングできません。

2. 多くの場合は、NFS が **rpcinfo** の出力に表示されていない時に NFS を再起動すると、サービスが **rpcbind** に正しく登録され、動作を開始します。

```
# systemctl restart nfs-server
```

関連情報

- NFSv4 専用サーバーの設定

24.11. ファイアウォールの背後で動作するように NFS サーバーを設定する手順

NFS には **rpcbind** サービスが必要です。このサービスは RPC サービスのポートを動的に割り当て、ファイアウォールルールの設定で問題が発生する可能性があります。以下のセクションでは、サポートが必要な場合に、ファイアウォールの内側で機能するように NFS バージョンを設定する方法を説明します。

- NFSv3

これには、NFSv3 をサポートするサーバーがすべて含まれます。

- NFSv3 専用サーバー
- NFSv3 と NFSv4 の両方に対応するサーバー

- NFSv4 専用

24.11.1. ファイアウォールの内側で動作するように NFSv3 対応サーバーを設定する手順

次の手順では、NFSv3 に対応するサーバーを設定し、ファイアウォールの内側で実行する方法を説明します。これには、NFSv3 専用サーバー、および NFSv3 と NFSv4 の両方をサポートするサーバーが含まれます。

手順

1. クライアントがファイアウォールの背後にある NFS 共有にアクセスできるようにするには、NFS サーバーで次のコマンドを実行してファイアウォールを設定します。

```
firewall-cmd --permanent --add-service mountd
firewall-cmd --permanent --add-service rpc-bind
firewall-cmd --permanent --add-service nfs
```

2. **/etc/nfs.conf** ファイルで、RPC サービス **nlockmgr** が使用するポートを次のように指定します。

```
[lockd]

port=tcp-port-number
udp-port=udp-port-number
```

   または、**/etc/modprobe.d/lockd.conf** ファイルで、**nlm_tcpport** および **nlm_udpport** を指定することもできます。

3. NFS サーバーで以下のコマンドを実行して、ファイアウォールで指定したポートを開きます。

```
firewall-cmd --permanent --add-port=<lockd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<lockd-udp-port>/udp
```

4. 以下のように、**/etc/nfs.conf** ファイルの **[statd]** セクションを編集して、**rpc.statd** の静的ポートを追加します。

```
[statd]

port=port-number
```

5. NFS サーバーで以下のコマンドを実行して、ファイアウォールに追加したポートを開きます。

```
firewall-cmd --permanent --add-port=<statd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<statd-udp-port>/udp
```

6. ファイアウォール設定を再読み込みします。

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

-

7. **rpc-statd** サービスを最初に再起動してから、**nfs-server** サービスを再起動します。

```
# systemctl restart rpc-statd.service
# systemctl restart nfs-server.service
```

または、**/etc/modprobe.d/lockd.conf** ファイルの **lockd** ポートを指定した場合は、次のコマンドを実行します。

a. **/proc/sys/fs/nfs/nlm_tcpport** と **/proc/sys/fs/nfs/nlm_udpport** の現在の値を更新します。

```
# sysctl -w fs.nfs.nlm_tcpport=<tcp-port>
# sysctl -w fs.nfs.nlm_udpport=<udp-port>
```

b. **rpc-statd** サービスおよび **nfs-server** サービスを再起動します。

```
# systemctl restart rpc-statd.service
# systemctl restart nfs-server.service
```

24.11.2. ファイアウォールの内側で実行されるように NFSv4 専用サーバーを設定する手順

次の手順では、NFSv4 専用サーバーをファイアウォールの内側で実行するように設定する方法を説明します。

手順

1. クライアントがファイアウォールの背後にある NFS 共有にアクセスできるようにするには、NFS サーバーで次のコマンドを実行してファイアウォールを設定します。

```
firewall-cmd --permanent --add-service nfs
```

2. ファイアウォール設定を再読み込みします。

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

3. NFS サーバーを再起動します。

```
# systemctl restart nfs-server
```

24.11.3. ファイアウォールの内側で動作するように NFSv3 クライアントを設定する手順

ファイアウォールの内側で実行するように NFSv3 クライアントを設定する手順は、ファイアウォールの内側で実行するように NFSv3 サーバーを設定する手順と似ています。

設定するマシンが NFS クライアントとサーバーの両方である場合には、NFSv3 対応サーバーがファイアウォールの内側で実行されるように設定する手順 で説明されている手順に従います。

以下の手順では、ファイアウォールの内側でのみ NFS クライアントマシンを設定する方法を説明します。

手順

1. クライアントがファイアウォールの内側で NFS クライアントにコールバックを実行できるようにするには、NFS クライアントで以下のコマンドを実行して **rpc-bind** サービスをファイアウォールに追加します。

```
firewall-cmd --permanent --add-service rpc-bind
```

2. **/etc/nfs.conf** ファイルで、RPC サービス **nlockmgr** が使用するポートを次のように指定します。

```
[lockd]

port=port-number
udp-port=upd-port-number
```

   または、**/etc/modprobe.d/lockd.conf** ファイルで、**nlm_tcpport** および **nlm_udpport** を指定することもできます。

3. NFS クライアントで以下のコマンドを実行して、ファイアウォールで指定したポートを開きます。

```
firewall-cmd --permanent --add-port=<lockd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<lockd-udp-port>/udp
```

4. 以下のように、**/etc/nfs.conf** ファイルの **[statd]** セクションを編集して、**rpc.statd** の静的ポートを追加します。

```
[statd]

port=port-number
```

5. NFS クライアントで以下のコマンドを実行して、ファイアウォールに追加したポートを開きます。

```
firewall-cmd --permanent --add-port=<statd-tcp-port>/tcp
firewall-cmd --permanent --add-port=<statd-udp-port>/udp
```

6. ファイアウォール設定を再読み込みします。

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

7. **rpc-statd** サービスを再起動します。

```
# systemctl restart rpc-statd.service
```

   または、**/etc/modprobe.d/lockd.conf** ファイルの **lockd** ポートを指定した場合は、次のコマンドを実行します。

   a. **/proc/sys/fs/nfs/nlm_tcpport** と **/proc/sys/fs/nfs/nlm_udpport** の現在の値を更新します。

```
# sysctl -w fs.nfs.nlm_tcpport=<tcp-port>
# sysctl -w fs.nfs.nlm_udpport=<udp-port>
```

b. **rpc-statd** サービスを再起動します。

```
# systemctl restart rpc-statd.service
```

24.11.4. ファイアウォールの内側で動作するように NFSv4 クライアントを設定する手順

この手順は、クライアントが NFSv4.0 を使用している場合に限り行います。その場合は、NFSv4.0 コールバックのポートを開く必要があります。

この手順は、NFSv4.1 以降では必要ありません。これは、新しいプロトコルバージョンでは、クライアントによって開始された同じ接続でサーバーがコールバックを実行するためです。

手順

1. NFSv4.0 コールバックがファイアウォールを通過できるようにするには、以下のように **/proc/sys/fs/nfs/nfs_callback_tcpport** を設定して、サーバーがクライアントのそのポートに接続できるようにします。

```
# echo "fs.nfs.nfs_callback_tcpport = <callback-port>" >/etc/sysctl.d/90-nfs-callback-port.conf
# sysctl -p /etc/sysctl.d/90-nfs-callback-port.conf
```

2. NFS クライアントで以下のコマンドを実行して、ファイアウォールの指定のポートを開きます。

```
firewall-cmd --permanent --add-port=<callback-port>/tcp
```

3. ファイアウォール設定を再読み込みします。

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

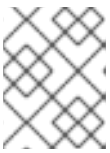
24.12. ファイアウォールからの RPC クォータのエクスポート

ディスククォータを使用するファイルシステムをエクスポートする場合は、クォータの RPC (Remote Procedure Call) サービスを使用して、NFS クライアントにディスククォータデータを提供できます。

手順

1. **rpc-rquotad** サービスを有効にして起動します。

```
# systemctl enable --now rpc-rquotad
```



注記

rpc-rquotad サービスが有効になっている場合は、nfs-server サービスが起動した後に自動的に起動されます。

2. ファイアウォールの内側で、クォータの RPC サービスにアクセスできるようにするには、TCP (UDP が可能な場合は UDP) の 875 ポートを開く必要があります。デフォルトのポート番号は **/etc/services** ファイルで定義します。
デフォルトのポート番号は、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-p port-number** を追加すると上書きできます。

3. デフォルトで、リモートホストはクォータのみを読み取ることができます。クライアントにクォータの設定を許可したい場合は、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-S** オプションを追加します。

4. **rpc-rquotad** を再起動して、**/etc/sysconfig/rpc-rquotad** ファイルの変更を反映します。

```
# systemctl restart rpc-rquotad
```

24.13. NFS OVER RDMA の有効化 (NFSORDMA)

Red Hat Enterprise Linux 8 では、RDMA 対応ハードウェア上の Remote Direct Memory Access (RDMA) サービスは、ネットワーク上でファイルを高速で転送するためのネットワークファイルシステム (NFS) プロトコルサポートを提供します。

手順

1. **rdma-core** パッケージをインストールします。

```
# yum install rdma-core
```

2. **xprtrdma** および **svcrdma** の行が **/etc/rdma/modules/rdma.conf** ファイルでコメント化されていることを確認します。

```
# NFS over RDMA client support
xprtrdma
# NFS over RDMA server support
svcrdma
```

3. NFS サーバーで、ディレクトリー **/mnt/nfsordma** を作成し、それを **/etc/exports** にエクスポートします。

```
# mkdir /mnt/nfsordma
# echo "/mnt/nfsordma *(fsid=0,rw,async,insecure,no_root_squash)" >> /etc/exports
```

4. NFS クライアントで、サーバーの IP アドレスを使用して nfs-share をマウントします (例: **172.31.0.186**)。

```
# mount -o rdma,port=20049 172.31.0.186:/mnt/nfs-share /mnt/nfs
```

5. **nfs-server** サービスを再起動します。

```
# systemctl restart nfs-server
```

関連情報

- RFC 5667 規格

24.14. 関連情報

- Linux NFS wiki

第25章 RED HAT ENTERPRISE LINUX での SMB 共有のマウント

Server Message Block (SMB) プロトコルは、アプリケーション層のネットワークプロトコルを実装します。これは、ファイル共有や共有プリンターなど、サーバー上のリソースにアクセスするために使用されます。

注記

SMB のコンテキストでは、SMB ダイアレクトである CIFS (Common Internet File System) プロトコルが言及されています。SMB と CIFS の両方のプロトコルがサポートされており、SMB 共有と CIFS 共有のマウントに関連するカーネルモジュールとユーティリティーはどちらも **cifs** という名前を使用します。

本セクションでは、SMB サーバーから共有をマウントする方法を説明します。Samba を使用して Red Hat Enterprise Linux に SMB サーバーを設定する方法は、Samba をサーバーとして使用 を参照してください。

前提条件

SMB は、Microsoft Windows にはデフォルトで実装されています。Red Hat Enterprise Linux では、カーネルの **cifs.ko** ファイルシステムモジュールが SMB 共有のマウントに対応します。したがって、**cifs-utils** パッケージをインストールします。

```
# yum install cifs-utils
```

cifs-utils パッケージには、以下を行うユーティリティーがあります。

- SMB 共有と CIFS 共有をマウントする
- カーネルのキーリングで、NT Lan Manager (NTLM) の認証情報を管理する
- SMB 共有および CIFS 共有のセキュリティー記述子で、アクセス制御リスト (ACL) を設定して、表示する

25.1. 対応している SMB プロトコルのバージョン

cifs.ko カーネルモジュールは、以下の SMB プロトコルバージョンをサポートします。

- SMB 1

警告

SMB1 プロトコルは既知のセキュリティー問題により非推奨となり、**プライベートネットワークでのみ安全に使用する** ことができます。SMB1 がサポートされているオプションとして推奨される主な理由は、現在 UNIX 拡張機能をサポートする唯一の SMB プロトコルバージョンであるためです。SMB で UNIX 拡張を使用する必要がない場合は、Red Hat は、SMB2 以降を使用することを強く推奨します。

- SMB 2.0
- SMB 2.1
- SMB 3.0
- SMB 3.1.1

注記

プロトコルのバージョンによっては、一部の SMB 機能しか実装されていません。

25.2. UNIX 拡張機能のサポート

Samba は、SMB プロトコルの **CAP_UNIX** 機能ビットを使用して UNIX 拡張機能を提供します。この拡張機能は、**cifs.ko** カーネルモジュールにも対応しています。ただし、Samba とカーネルモジュールはいずれも、SMB 1 プロトコルでのみ UNIX 拡張機能に対応します。

UNIX 拡張機能を使用するには、以下の手順を実行します。

1. **/etc/samba/smb.conf** ファイルの **[global]** セクションにある **server min protocol** パラメーターを **NT1** に設定します。
2. マウントコマンドに **-o vers=1.0** オプションを指定し、SMB 1 プロトコルを使用して共有をマウントします。以下に例を示します。

   ```
   # mount -t cifs -o vers=1.0,username=user_name //server_name/share_name /mnt/
   ```

   デフォルトで、カーネルモジュールは、SMB 2 またはサーバーでサポートされている最新のプロトコルバージョンを使用します。**-o vers=1.0** オプションを **mount** コマンドに渡すと、UNIX 拡張機能の使用に必要な SMB 1 プロトコルをカーネルモジュールが使用することが強制されます。

UNIX 拡張機能が有効になっているかどうかを確認するには、マウントされた共有のオプションを表示します。

```
# mount
...
//server/share on /mnt type cifs (...,unix,...)
```

マウントオプションのリストに **unix** エントリーが表示されている場合は、UNIX 拡張機能が有効になっています。

25.3. SMB 共有の手動マウント

SMB 共有のみを一時的にマウントする必要がある場合は、**mount** ユーティリティーを使用して手動でマウントできます。

注記

手動でマウントされた共有は、システムを再起動しても自動的にはマウントされません。システムの起動時に、Red Hat Enterprise Linux が自動的に共有をマウントするように設定する場合は、システムの起動時に自動的に SMB 共有をマウントする を参照してください。

前提条件

- **cifs-utils** パッケージがインストールされている。

手順

SMB 共有を手動でマウントする場合は、**mount** ユーティリティーに **-t cifs** パラメーターを指定して実行します。

```
# mount -t cifs -o username=user_name //server_name/share_name /mnt/
Password for user_name@//server_name/share_name:  password
```

-o パラメーターでは、共有のマウントに使用されるオプションを指定できます。詳細は、**mount.cifs(8)** の man ページおよび 頻繁に使用されるマウントオプション の **OPTIONS** セクションを参照してください。

例25.1 暗号化された SMB 3.0 接続を使用した共有のマウント

暗号化された SMB 3.0 接続で、**DOMAIN\Administrator** ユーザーとして **server\example**\ 共有を **/mnt/** ディレクトリーにマウントする場合は、次の手順を実行します。

```
# mount -t cifs -o username=DOMAIN\Administrator,seal,vers=3.0 //server/example /mnt/
Password for DOMAIN\Administrator@//server_name/share_name:  password
```

25.4. システム起動時の SMB 共有の自動マウント

マウントされた SMB 共有へのアクセスがサーバー上で恒久的に必要とされる場合は、システムの起動時に共有を自動的にマウントします。

前提条件

- **cifs-utils** パッケージがインストールされている。

手順

システムの起動時に SMB 共有を自動的にマウントするには、共有のエントリーを **/etc/fstab** ファイルに追加します。以下に例を示します。

```
//server_name/share_name  /mnt  cifs  credentials=/root/smb.cred  0 0
```

重要

システムが自動的に共有をマウントできるようにするには、ユーザー名、パスワード、およびドメイン名を認証情報ファイルに保存する必要があります。詳細は、認証情報ファイルを使用した SMB 共有への認証 を参照してください。

/etc/fstab の行の 4 つ目のフィールドで、認証情報ファイルへのパスなど、マウントオプションを指定します。詳細は、**mount.cifs(8)** の man ページおよび 頻繁に使用されるマウントオプション の **OPTIONS** セクションを参照してください。

共有が正常にマウントされたことを確認する場合は、次のコマンドを実行します。

```
# mount /mnt/
```

25.5. 認証情報ファイルを使用した SMB 共有への認証

特定の状況 (システムの起動時に共有を自動的にマウントする場合など) では、ユーザー名とパスワードを入力することなく共有がマウントされる必要があります。これを実装するには、認証情報ファイルを作成します。

前提条件

- **cifs-utils** パッケージがインストールされている。

手順

1. **/root/smb.cred** などのファイルを作成し、そのファイルのユーザー名、パスワード、およびドメイン名を指定します。

```
username=user_name
password=password
domain=domain_name
```

2. 所有者だけがファイルにアクセスできるようにパーミッションを設定します。

```
# chown user_name /root/smb.cred
# chmod 600 /root/smb.cred
```

mount ユーティリティーに **credentials=file_name** マウントオプションを渡すか、**/etc/fstab** ファイルでこのオプションを使用して、ユーザー名とパスワードの入力を求められずに共有をマウントできます。

25.6. よく使用されるマウントオプション

SMB 共有をマウントすると、マウントオプションにより次のことが決まります。

- サーバーとの接続がどのように確立されるか。たとえば、サーバーに接続するときに使用される SMB プロトコルバージョンはどれか。
- 共有が、ローカルファイルシステムにどのようにマウントされるか。たとえば、複数のローカルユーザーが、サーバーのコンテンツにアクセスできるようにするために、システムがリモートファイルとディレクトリーのパーミッションを上書きする場合など。

/etc/fstab ファイルの 4 番目のフィールド、またはマウントコマンドの **-o** パラメーターで複数のオプションを設定するには、オプションをコンマで区切ります。たとえば、multiuser オプションを使用した共有のマウント を参照してください。

次のリストは、よく使用されるマウントオプションを示しています。

オプション	説明
credentials=**file_name**	認証情報ファイルへのパスを設定します。認証情報ファイルを使用した SMB 共有への認証 を参照してください。

オプション	説明
dir_mode=**mode**	サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ディレクトリーモードを設定します。
file_mode=**mode**	サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ファイルモードを設定します。
password=**password**	SMB サーバーへの認証に使用されるパスワードを設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。
seal	SMB 3.0 以降のプロトコルバージョンを使用した接続に対する暗号化サポートを有効にします。そのため、**seal** は **3.0** 以降に設定された **vers** マウントオプションと一緒に使用します。SMB 共有の手動マウント の例を参照してください。
sec=**security_mode**	**ntlmsspi** などのセキュリティーモードを設定して、NTLMv2 パスワードハッシュとパケット署名を有効にします。対応している値のリストは、man ページの **mount.cifs(8)** にあるオプションの説明を参照してください。 サーバーが **ntlmv2** セキュリティーモードに対応していない場合は、**sec=ntlmssp** (デフォルト) を使用します。 セキュリティー上の理由から、安全でない**ntlm**セキュリティーモードは使用しないでください。
username=**user_name**	SMB サーバーへの認証に使用されるユーザー名を設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。
vers=**SMB_protocol_version**	サーバーとの通信に使用される SMB プロトコルバージョンを設定します。

完全なリストは、man ページの **mount.cifs(8)** の **OPTIONS** セクションを参照してください。

第26章 永続的な命名属性の概要

システム管理者は、永続的な命名属性を使用してストレージボリュームを参照し、再起動を何度も行っても信頼できるストレージ設定を構築する必要があります。

26.1. 非永続的な命名属性のデメリット

Red Hat Enterprise Linux では、ストレージデバイスを識別する方法が複数あります。特にドライブへのインストール時やドライブの再フォーマット時に誤ったデバイスにアクセスしないようにするため、適切なオプションを使用して各デバイスを識別することが重要になります。

従来、**/dev/sd(メジャー番号)(マイナー番号)** の形式の非永続的な名前は、ストレージデバイスを参照するために Linux 上で使用されます。メジャー番号とマイナー番号の範囲、および関連する **sd** 名は、検出されると各デバイスに割り当てられます。つまり、デバイスの検出順序が変わると、メジャー番号とマイナー番号の範囲、および関連する **sd** 名の関連付けが変わる可能性があります。

このような順序の変更は、以下の状況で発生する可能性があります。

- システム起動プロセスの並列化により、システム起動ごとに異なる順序でストレージデバイスが検出された場合。
- ディスクが起動しなかったり、SCSI コントローラーに応答しなかった場合。この場合は、通常のデバイスプローブにより検出されません。ディスクはシステムにアクセスできなくなり、後続のデバイスは関連する次の **sd** 名が含まれる、メジャー番号およびマイナー番号の範囲があります。たとえば、通常 **sdb** と呼ばれるディスクが検出されないと、**sdc** と呼ばれるディスクが **sdb** として代わりに表示されます。
- SCSI コントローラー (ホストバスアダプターまたは HBA) が初期化に失敗し、その HBA に接続されているすべてのディスクが検出されなかった場合。後続のプローブされた HBA に接続しているディスクは、別のメジャー番号およびマイナー番号の範囲、および関連する別の **sd** 名が割り当てられます。
- システムに異なるタイプの HBA が存在する場合は、ドライバー初期化の順序が変更する可能性があります。これにより、HBA に接続されているディスクが異なる順序で検出される可能性があります。また、HBA がシステムの他の PCI スロットに移動した場合でも発生する可能性があります。
- ストレージアレイや干渉するスイッチの電源が切れた場合など、ストレージデバイスがプローブされたときに、ファイバーチャネル、iSCSI、または FCoE アダプターを持つシステムに接続されたディスクがアクセスできなくなる可能性があります。システムが起動するまでの時間よりもストレージアレイがオンラインになるまでの時間の方が長い場合に、電源の障害後にシステムが再起動すると、この問題が発生する可能性があります。一部のファイバーチャネルドライバーは WWPN マッピングへの永続 SCSI ターゲット ID を指定するメカニズムをサポートしますが、メジャー番号およびマイナー番号の範囲や関連する **sd** 名は予約されず、一貫性のある SCSI ターゲット ID 番号のみが提供されます。

そのため、**/etc/fstab** ファイルなどにあるデバイスを参照するときにメジャー番号およびマイナー番号の範囲や関連する **sd** 名を使用することは望ましくありません。誤ったデバイスがマウントされ、データが破損する可能性があります。

しかし、場合によっては他のメカニズムが使用される場合でも **sd** 名の参照が必要になる場合もあります (デバイスによりエラーが報告される場合など)。これは、Linux カーネルはデバイスに関するカーネルメッセージで **sd** 名 (および SCSI ホスト、チャネル、ターゲット、LUN タプル) を使用するためです。

26.2. ファイルシステムおよびデバイスの識別子

このセクションでは、ファイルシステムおよびブロックデバイスを識別する永続的な属性の相違点を説明します。

ファイルシステムの識別子
ファイルシステムの識別子は、ブロックデバイス上に作成された特定のファイルシステムに関連付けられます。識別子はファイルシステムの一部としても格納されます。ファイルシステムを別のデバイスにコピーしても、ファイルシステム識別子は同じです。一方、**mkfs** ユーティリティーでフォーマットするなどしてデバイスを書き換えると、デバイスはその属性を失います。

ファイルシステムの識別子に含まれるものは、次のとおりです。

- 一意の ID (UUID)
- ラベル

デバイスの識別子
デバイス識別子は、ブロックデバイス (ディスクやパーティションなど) に関連付けられます。**mkfs** ユーティリティーでフォーマットするなどしてデバイスを書き換えた場合、デバイスはファイルシステムに格納されていないため、属性を保持します。

デバイスの識別子に含まれるものは、次のとおりです。

- World Wide Identifier (WWID)
- パーティション UUID
- シリアル番号

推奨事項

- 論理ボリュームなどの一部のファイルシステムは、複数のデバイスにまたがっています。Red Hat は、デバイスの識別子ではなくファイルシステムの識別子を使用してこのファイルシステムにアクセスすることを推奨します。

26.3. /DEV/DISK/ にある UDEV メカニズムにより管理されるデバイス名

udev メカニズムは、Linux のすべてのタイプのデバイスに使用され、ストレージデバイスだけに限定されません。**/dev/disk/** ディレクトリーにさまざまな種類の永続的な命名属性を提供します。ストレージデバイスの場合、Red Hat Enterprise Linux には **/dev/disk/** ディレクトリーにシンボリックリンクを作成する **udev** ルールが含まれています。これにより、次の方法でストレージデバイスを参照できます。

- ストレージデバイスのコンテンツ
- 一意の ID
- シリアル番号

udev の命名属性は永続的なものですが、システムを再起動しても自動的には変更されないため、設定可能なものもあります。

26.3.1. ファイルシステムの識別子

/dev/disk/by-uuid/ の UUID 属性

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内の **一意の ID** (UUID) によりストレージデバイスを参照するシンボリック名を提供します。以下に例を示します。

```
/dev/disk/by-uuid/3e6be9de-8139-11d1-9106-a43f08d823a6
```

次の構文を使用することで、UUID を使用して **/etc/fstab** ファイルのデバイスを参照できます。

```
UUID=3e6be9de-8139-11d1-9106-a43f08d823a6
```

ファイルシステムを作成する際に UUID 属性を設定できます。後で変更することもできます。

/dev/disk/by-label/ のラベル属性

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内の **ラベル** により、ストレージデバイスを参照するシンボリック名を提供します。

以下に例を示します。

```
/dev/disk/by-label/Boot
```

次の構文を使用することで、ラベルを使用して **/etc/fstab** ファイルのデバイスを参照できます。

```
LABEL=Boot
```

ファイルシステムを作成するときにラベル属性を設定できます。また、後で変更することもできます。

26.3.2. デバイスの識別子

/dev/disk/by-id/ の WWID 属性

World Wide Identifier (WWID) は永続的で、SCSI 規格によりすべての SCSI デバイスが必要とする **システムに依存しない識別子** です。各ストレージデバイスの WWID 識別子は一意となることが保証され、デバイスのアクセスに使用されるパスに依存しません。この識別子はデバイスのプロパティーですが、デバイスのコンテンツ (つまりデータ) には格納されません。

この識別子は、SCSI Inquiry を発行して Device Identification Vital Product Data (**0x83** ページ) または Unit Serial Number (**0x80** ページ) を取得することにより獲得できます。

Red Hat Enterprise Linux では、WWID ベースのデバイス名から、そのシステムの現在の **/dev/sd** 名への正しいマッピングを自動的に維持します。デバイスへのパスが変更したり、別のシステムからそのデバイスへのアクセスがあった場合にも、アプリケーションはディスク上のデータ参照に **/dev/disk/by-id/** 名を使用できます。

例26.1 WWID マッピング

WWID シンボリックリンク	非永続的なデバイス	備考
/dev/disk/by-id/scsi-3600508b400105e210000900000490000	**/dev/sda**	ページ **0x83** の識別子を持つデバイス
/dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6	**/dev/sdb**	ページ **0x80** の識別子を持つデバイス

WWID シンボリックリンク	非永続的なデバイス	備考
/dev/disk/by-id/ata-SAMSUNG_MZNLN256HMHQ-000L7_S2WDNX0J336519-part3	**/dev/sdc3**	ディスクパーティション

システムにより提供される永続的な名前のほかに、**udev** ルールを使用して独自の永続的な名前を実装し、ストレージの WWID にマップすることもできます。

/dev/disk/by-partuuid のパーティション UUID 属性

パーティション UUID (PARTUUID) 属性は、GPT パーティションテーブルにより定義されているパーティションを識別します。

例26.2 パーティション UUID のマッピング

PARTUUID シンボリックリンク	非永続的なデバイス
/dev/disk/by-partuuid/4cd1448a-01	**/dev/sda1**
/dev/disk/by-partuuid/4cd1448a-02	**/dev/sda2**
/dev/disk/by-partuuid/4cd1448a-03	**/dev/sda3**

/dev/disk/by-path/ のパス属性

この属性は、デバイスへのアクセスに使用される **ハードウェアパス** がストレージデバイスを参照するシンボル名を提供します。

ハードウェアパス (PCI ID、ターゲットポート、LUN 番号など) の一部が変更されると、パス属性に失敗します。このため、パス属性は信頼性に欠けます。ただし、パス属性は以下のいずれかのシナリオで役に立ちます。

- 後で置き換える予定のディスクを特定する必要があります。
- 特定の場所にあるディスクにストレージサービスをインストールする予定です。

26.4. DM MULTIPATH を使用した WORLD WIDE IDENTIFIER

Device Mapper (DM) Multipath を設定して、World Wide Identifier (WWID) と非永続的なデバイス名をマッピングできます。

システムからデバイスへのパスが複数ある場合、DM Multipath はこれを検出するために WWID を使用します。その後、DM Multipath は **/dev/mapper/wwid** ディレクトリー (例: **/dev/mapper/3600508b400105df70000e00000ac0000**) に単一の "疑似デバイス" を表示します。

コマンド **multipath -l** は、非永続的な識別子へのマッピングを示します。

- **Host:Channel:Target:LUN**

- **/dev/sd** 名
- **major:minor** 数値

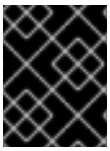
例26.3 マルチパス設定での WWID マッピング

multipath -l コマンドの出力例:

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

DM Multipath は、各 WWID ベースのデバイス名から、システムで対応する **/dev/sd** 名への適切なマッピングを自動的に維持します。これらの名前は、パスが変更しても持続し、他のシステムからデバイスにアクセスする際に一貫性を保持します。

DM Multipath の **user_friendly_names** 機能を使用すると、WWID は **/dev/mapper/mpathN** 形式の名前にマップされます。デフォルトでは、このマッピングは **/etc/multipath/bindings** ファイルに保持されています。これらの **mpathN** 名は、そのファイルが維持されている限り永続的です。



重要

user_friendly_names を使用する場合は、クラスター内で一貫した名前を取得するために追加の手順が必要です。

26.5. UDEV デバイス命名規則の制約

udev 命名規則の制約の一部は次のとおりです。

- **udev** イベントに対して **udev** ルールが処理されるときに、**udev** メカニズムはストレージデバイスをクエリーする機能に依存する可能性があるため、クエリーの実行時にデバイスにアクセスできない可能性があります。これは、ファイバーチャネル、iSCSI、または FCoE ストレージデバイスといった、デバイスがサーバーシャーシにない場合に発生する可能性が高くなります。
- カーネルは **udev** イベントをいつでも送信する可能性があるため、デバイスにアクセスできない場合に **/dev/disk/by-*/** リンクが削除される可能性があります。
- **udev** イベントが生成されそのイベントが処理されるまでに遅延が生じる場合があります (大量のデバイスが検出され、ユーザー空間の **udev** サービスによる各デバイスのルールを処理するのにある程度の時間がかかる場合など)。これにより、カーネルがデバイスを検出してから、**/dev/disk/by-*/** の名前が利用できるようになるまでに遅延が生じる可能性があります。
- ルールに呼び出される **blkid** などの外部プログラムによってデバイスが短期間開き、他の目的でデバイスにアクセスできなくなる可能性があります。
- /dev/disk/ の **udev** メカニズムで管理されるデバイス名は、メジャーリリース間で変更される可能性があるため、リンクの更新が必要になる場合があります。

26.6. 永続的な命名属性のリスト表示

この手順では、非永続的なストレージデバイスの永続命名属性を確認する方法を説明します。

手順

- UUID 属性とラベル属性をリスト表示するには、**lsblk** ユーティリティーを使用します。

```
$ lsblk --fs storage-device
```

以下に例を示します。

例26.4 ファイルシステムの UUID とラベルの表示

```
$ lsblk --fs /dev/sda1

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  afa5d5e3-9050-48c3-acc1-bb30095f3dc4 /boot
```

- PARTUUID 属性をリスト表示するには、**--output +PARTUUID** オプションを指定して **lsblk** ユーティリティーを使用します。

```
$ lsblk --output +PARTUUID
```

以下に例を示します。

例26.5 パーティションの PARTUUID 属性の表示

```
$ lsblk --output +PARTUUID /dev/sda1

NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT PARTUUID
sda1   8:1    0  512M  0 part /boot      4cd1448a-01
```

- WWID 属性をリスト表示するには、**/dev/disk/by-id/** ディレクトリーのシンボリックリンクのターゲットを調べます。以下に例を示します。

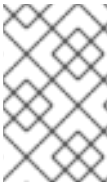
例26.6 システムにある全ストレージデバイスの WWID の表示

```
$ file /dev/disk/by-id/*

/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001
symbolic link to ../../sda
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1
symbolic link to ../../sda1
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part2
symbolic link to ../../sda2
/dev/disk/by-id/dm-name-rhel_rhel8-root
symbolic link to ../../dm-0
/dev/disk/by-id/dm-name-rhel_rhel8-swap
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhP0RMFsNyySVihqEl2cWWbR7MjXJolD6g
```

```
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhXqH2M45hD2H9nAf2qfWSrlRLhzfMyOKd
symbolic link to ../../dm-0
/dev/disk/by-id/lvm-pv-uuid-atlr2Y-vuMo-ueoH-CpMG-4JuH-AhEF-wu4QQm
symbolic link to ../../sda2
```

26.7. 永続的な命名属性の変更

この手順では、ファイルシステムの UUID またはラベルの永続的な命名属性を変更する方法を説明します。



注記

udev 属性の変更はバックグラウンドで行われ、時間がかかる場合があります。**udevadm settle** コマンドは変更が完全に登録されるまで待機します。これにより、次のコマンドが新しい属性を正しく利用できるようになります。

以下のコマンドでは、次を行います。

- **new-uuid** を、設定する UUID (例: **1cdfbc07-1c90-4984-b5ec-f61943f5ea50**) に置き換えます。**uuidgen** コマンドを使用して UUID を生成できます。
- **new-label** を、ラベル (例: **backup_data**) に置き換えます。

前提条件

- XFS ファイルシステムをアンマウントしている (XFS ファイルシステムの属性を変更する場合)。

手順

- **XFS** ファイルシステムの UUID またはラベル属性を変更するには、**xfs_admin** ユーティリティーを使用します。

```
# xfs_admin -U new-uuid -L new-label storage-device
# udevadm settle
```

- **ext4** ファイルシステム、**ext3** ファイルシステム、**ext2** ファイルシステムの UUID またはラベル属性を変更するには、**tune2fs** ユーティリティーを使用します。

```
# tune2fs -U new-uuid -L new-label storage-device
# udevadm settle
```

- スワップボリュームの UUID またはラベル属性を変更するには、**swaplabel** ユーティリティーを使用します。

```
# swaplabel --uuid new-uuid --label new-label swap-device
# udevadm settle
```

第27章 パーティションの使用

ディスクパーティション設定を使用して、ディスクを 1 つ以上の論理領域に分割し、各パーティションで個別に作業できるようにします。ハードディスクは、パーティションテーブルの各ディスクパーティションの場所とサイズに関する情報を保存します。このテーブルを使用すると、各パーティションはオペレーティングシステムへの論理ディスクとして表示されます。その後、それらの個々のディスクで読み取りと書き込みを行うことができます。

ブロックデバイスでパーティションを使用する場合のメリットとデメリットの概要については、利点と欠点の概要については 直接または LVM を間に入れて、LUN でパーティション設定を使用するメリットとデメリットは何ですか？を参照してください。

27.1. PARTED でディスクにパーティションテーブルを作成

parted ユーティリティーを使用して、より簡単にパーティションテーブルでブロックデバイスをフォーマットできます。

警告

パーティションテーブルを使用してブロックデバイスをフォーマットすると、そのデバイスに保存されているすべてのデータが削除されます。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

2. デバイスにパーティションテーブルがあるかどうかを確認します。

```
# (parted) print
```

デバイスにパーティションが含まれている場合は、次の手順でパーティションを削除します。

3. 新しいパーティションテーブルを作成します。

```
# (parted) mklabel table-type
```

- **table-type** を、使用するパーティションテーブルのタイプに置き換えます。
  - **msdo** (MBR の場合)
  - **gpt** (GPT の場合)

例27.1 GUID パーティションテーブル (GPT) テーブルの作成

ディスクに GPT テーブルを作成するには、次のコマンドを使用します。

```
# (parted) mklabel gpt
```

```
```

このコマンドを入力すると、変更の適用が開始されます。

4. パーティションテーブルを表示して、作成されたことを確認します。

```
# (parted) print
```

5. **parted** シェルを終了します。

```
# (parted) quit
```

関連情報

- **parted(8)** man ページ

27.2. PARTED でパーティションテーブルの表示

ブロックデバイスのパーティションテーブルを表示して、パーティションレイアウトと個々のパーティションの詳細を確認します。**parted** ユーティリティーを使用して、ブロックデバイスのパーティションテーブルを表示できます。

手順

1. **parted** ユーティリティーを起動します。たとえば、次の出力は、デバイス **/dev/sda** をリストします。

```
# parted /dev/sda
```

2. パーティションテーブルを表示します。

```
# (parted) print

Model: ATA SAMSUNG MZNLN256 (scsi)
Disk /dev/sda: 256GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  269MB   268MB   primary   xfs          boot
 2      269MB   34.6GB  34.4GB  primary
 3      34.6GB  45.4GB  10.7GB  primary
 4      45.4GB  256GB   211GB   extended
 5      45.4GB  256GB   211GB   logical
```

3. オプション:次に調べるデバイスに切り替えます。

```
# (parted) select block-device
```

print コマンドの出力の詳細については、以下を参照してください。

モデル:ATA SAMSUNG MZNLN256 (scsi)

ディスクタイプ、製造元、モデル番号、およびインターフェイス。

Disk /dev/sda:256GB

ブロックデバイスへのファイルパスとストレージ容量。

Partition Table: msdos

ディスクラベルの種類。

Number

パーティション番号。たとえば、マイナー番号 1 のパーティションは、**/dev/sda1** に対応します。

Start および End

デバイスにおけるパーティションの開始場所と終了場所。

Type

有効なタイプは、メタデータ、フリー、プライマリー、拡張、または論理です。

File system

ファイルシステムの種類。ファイルシステムの種類が不明な場合は、デバイスの **File system** フィールドに値が表示されません。**parted** ユーティリティーは、暗号化されたデバイスのファイルシステムを認識できません。

Flags

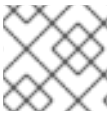
パーティションのフラグ設定リスト。利用可能なフラグは、**boot**、**root**、**swap**、**hidden**、**raid**、**lvm**、または **lba** です。

関連情報

- **parted(8)** man ページ

27.3. PARTED でパーティションの作成

システム管理者は、**parted** ユーティリティーを使用してディスクに新しいパーティションを作成できます。



注記

必要なパーティションは、**swap**、**/boot/**、および **/(root)** です。

前提条件

- ディスクのパーティションテーブル。
- 2TiB を超えるパーティションを作成する場合は、GUID Partition Table (GPT)でディスクをフォーマットしておく。

手順

1. **parted** ユーティリティーを起動します。

```
# parted block-device
```

2. 現在のパーティションテーブルを表示し、十分な空き領域があるかどうかを確認します。

```
# (parted) print
```

- 十分な空き容量がない場合は、パーティションのサイズを変更してください。
- パーティションテーブルから、以下を確認します。
  - 新しいパーティションの開始点と終了点
  - MBR で、どのパーティションタイプにすべきか

3. 新しいパーティションを作成します。

```
# (parted) mkpart part-type name fs-type start end
```

- **part-type** を **primary**、**logical**、または **extended** に置き換えます。これは MBR パーティションテーブルにのみ適用されます。
- **name** を任意のパーティション名に置き換えます。これは GPT パーティションテーブルに必要です。
- **fs-type** を、**xfs**、**ext2**、**ext3**、**ext4**、**fat16**、**fat32**、**hfs**、**hfs+**、**linux-swap**、**ntfs**、または **reiserfs** に置き換えます。**fs-type** パラメーターは任意です。**parted** ユーティリティーは、パーティションにファイルシステムを作成しないことに注意してください。
- **start** と **end** を、パーティションの開始点と終了点を決定するサイズに置き換えます (ディスクの開始からカウントします)。**512MiB**、**20GiB**、**1.5TiB** などのサイズ接尾辞を使用できます。デフォルトサイズの単位はメガバイトです。

例27.2 小さなプライマリーパーティションの作成

MBR テーブルに 1024MiB から 2048MiB までのプライマリーパーティションを作成するには、次のコマンドを使用します。

```
# (parted) mkpart primary 1024MiB 2048MiB
```

コマンドを入力すると、変更の適用が開始されます。

4. パーティションテーブルを表示して、作成されたパーティションのパーティションタイプ、ファイルシステムタイプ、サイズが、パーティションテーブルに正しく表示されていることを確認します。

```
# (parted) print
```

5. **parted** シェルを終了します。

```
# (parted) quit
```

6. 新規デバイスノードを登録します。

```
# udevadm settle
```

7. カーネルが新しいパーティションを認識していることを確認します。

```
# cat /proc/partitions
```

関連情報

- **parted(8)** man ページ
- parted でディスクにパーティションテーブルを作成
- parted でパーティションのサイズ変更

27.4. FDISK でパーティションタイプの設定

fdisk ユーティリティーを使用して、パーティションタイプまたはフラグを設定できます。

前提条件

- ディスク上のパーティション。

手順

1. インタラクティブな **fdisk** シェルを起動します。

```
# fdisk block-device
```

2. 現在のパーティションテーブルを表示して、パーティションのマイナー番号を確認します。

```
Command (m for help): print
```

現在のパーティションタイプは **Type** 列で、それに対応するタイプ ID は **Id** 列で確認できます。

3. パーティションタイプコマンドを入力し、マイナー番号を使用してパーティションを選択します。

```
Command (m for help): type
Partition number (1,2,3 default 3): 2
```

4. オプション:16 進数コードでリストを表示します。

```
Hex code (type L to list all codes): L
```

5. パーティションタイプを設定します。

```
Hex code (type L to list all codes): 8e
```

6. 変更を書き込み、**fdisk** シェルを終了します。

```
Command (m for help): write
The partition table has been altered.
Syncing disks.
```

7. 変更を確認します。

```
# fdisk --list block-device
```

27.5. PARTED でパーティションのサイズ変更

parted ユーティリティーを使用して、パーティションを拡張して未使用のディスク領域を利用したり、パーティションを縮小してその容量をさまざまな目的に使用したりできます。

前提条件

- パーティションを縮小する前にデータをバックアップする。
- 2TiB を超えるパーティションを作成する場合は、**GUID Partition Table (GPT)**でディスクをフォーマットしておく。
- パーティションを縮小する場合は、サイズを変更したパーティションより大きくならないように、最初にファイルシステムを縮小しておく。

注記

XFS は縮小に対応していません。

手順

1. **parted** ユーティリティーを起動します。

```
# parted block-device
```

2. 現在のパーティションテーブルを表示します。

```
# (parted) print
```

パーティションテーブルから、以下を確認します。

- パーティションのマイナー番号。
- 既存のパーティションの位置とサイズ変更後の新しい終了点。

3. パーティションのサイズを変更します。

```
# (parted) resizepart 1 2GiB
```

- **1** を、サイズを変更するパーティションのマイナー番号に置き換えます。
- **2** を、サイズを変更するパーティションの新しい終了点を決定するサイズに置き換えます (ディスクの開始からカウントします)。**512MiB**、**20GiB**、**1.5TiB** などのサイズ接尾辞を使用できます。デフォルトサイズの単位はメガバイトです。

4. パーティションテーブルを表示して、サイズ変更したパーティションのサイズが、パーティションテーブルで正しく表示されていることを確認します。

```
# (parted) print
```

5. **parted** シェルを終了します。

```
# (parted) quit
```

6. カーネルが新しいパーティションを登録していることを確認します。

```
# cat /proc/partitions
```

7. オプション:パーティションを拡張した場合は、そこにあるファイルシステムも拡張します。

関連情報

- **parted(8)** man ページ
- parted でディスクにパーティションテーブルを作成
- ext3 ファイルシステムのサイズ変更
- XFS ファイルシステムのサイズの拡大

27.6. PARTED でパーティションの削除

parted ユーティリティーを使用すると、ディスクパーティションを削除して、ディスク領域を解放できます。

警告

パーティションを削除すると、そのパーティションに保存されているすべてのデータが削除されます。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

   - **block-device** を、パーティションを削除するデバイスへのパス (例: **/dev/sda**) に置き換えます。

2. 現在のパーティションテーブルを表示して、削除するパーティションのマイナー番号を確認します。

```
(parted) print
```

3. パーティションを削除します。

```
(parted) rm minor-number
```

   - **minor-number** を、削除するパーティションのマイナー番号に置き換えます。

   このコマンドを実行すると、すぐに変更の適用が開始されます。

4. パーティションテーブルからパーティションが削除されたことを確認します。

```
(parted) print
```

5. **parted** シェルを終了します。

```
(parted) quit
```

6. パーティションが削除されたことをカーネルが登録していることを確認します。

```
# cat /proc/partitions
```

7. パーティションが存在する場合は、**/etc/fstab** ファイルからパーティションを削除します。削除したパーティションを宣言している行を見つけ、ファイルから削除します。

8. システムが新しい **/etc/fstab** 設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

9. スワップパーティション、または LVM の一部を削除した場合は、カーネルコマンドラインからパーティションへの参照をすべて削除します。

   a. アクティブなカーネルオプションを一覧表示し、削除されたパーティションを参照するオプションがないか確認します。

   ```
   # grubby --info=ALL
   ```

   b. 削除されたパーティションを参照するカーネルオプションを削除します。

   ```
   # grubby --update-kernel=ALL --remove-args="option"
   ```

10. アーリーブートシステムに変更を登録するには、**initramfs** ファイルシステムを再構築します。

```
# dracut --force --verbose
```

関連情報

- **parted(8)** man ページ

第28章 XFS の使用

これは、XFS ファイルシステムを作成および維持する方法の概要です。

28.1. XFS ファイルシステム

XFS は、拡張性が高く、高性能で堅牢な、成熟した 64 ビットのジャーナリングファイルシステムで、1 台のホストで非常に大きなファイルおよびファイルシステムに対応します。Red Hat Enterprise Linux 8 ではデフォルトのファイルシステムになります。XFS は、元々 1990 年代の前半に SGI により開発され、極めて大規模なサーバーおよびストレージアレイで実行されてきた長い歴史があります。

XFS の機能は次のとおりです。

信頼性

- メタデータジャーナリング - システムの再起動時、およびファイルシステムの再マウント時に再生できるファイルシステム操作の記録を保持することで、システムクラッシュ後のファイルシステムの整合性を確保します。
- 広範囲に及ぶランタイムメタデータの整合性チェック
- 拡張性が高く、高速な修復ユーティリティー
- クォータジャーナリングクラッシュ後に行なわれる、時間がかかるクォータの整合性チェックが不要になります。

スケーラビリティーおよびパフォーマンス

- 対応するファイルシステムのサイズが最大 1024 TiB
- 多数の同時操作に対応する機能
- 空き領域管理のスケーラビリティーに関する B-Tree インデックス
- 高度なメタデータ先読みアルゴリズム
- ストリーミングビデオのワークロードの最適化

割り当てスキーム

- エクステント (領域) ベースの割り当て
- ストライプを認識できる割り当てポリシー
- 遅延割り当て
- 領域の事前割り当て
- 動的に割り当てられる inode

その他の機能

- Reflink ベースのファイルのコピー
- 密接に統合されたバックアップおよび復元のユーティリティー

- オンラインのデフラグ
- オンラインのファイルシステム拡張
- 包括的な診断機能
- 拡張属性 (**xattr**)。これにより、システムが、ファイルごとに、名前と値の組み合わせを追加で関連付けられるようになります。
- プロジェクトまたはディレクトリーのクォータ。ディレクトリーツリー全体にクォータ制限を適用できます。
- サブセカンド (一秒未満) のタイムスタンプ

パフォーマンスの特徴

XFS は、エンタープライズレベルのワークロードがある大規模なシステムで優れたパフォーマンスを発揮します。大規模なシステムとは、相対的に CPU 数が多く、さらには複数の HBA、および外部ディスクアレイへの接続を備えたシステムです。XFS は、マルチスレッドの並列 I/O ワークロードを備えた小規模のシステムでも適切に実行します。

XFS は、シングルスレッドで、メタデータ集約型のワークロードのパフォーマンスが比較的低くなります。たとえば、シングルスレッドで小さなファイルを多数作成し、削除するワークロードがこれに当てはまります。

28.2. EXT4 および XFS で使用されるツールの比較

本セクションでは、ext4 ファイルシステムおよび XFS ファイルシステムで一般的なタスクを行うのに使用するツールを比較します。

タスク	ext4	XFS
ファイルシステムを作成する	**mkfs.ext4**	**mkfs.xfs**
ファイルシステム検査	**e2fsck**	**xfs_repair**
ファイルシステムのサイズを変更する	**resize2fs**	**xfs_growfs**
ファイルシステムのイメージを保存する	**e2image**	**xfs_metadump** および **xfs_mdrestore**
ファイルシステムのラベル付けまたはチューニングを行う	**tune2fs**	**xfs_admin**
ファイルシステムのバックアップを作成する	**dump** および **restore**	**xfsdump** および **xfsrestore**
クォータ管理	**quota**	**xfs_quota**
ファイルマッピング	**filefrag**	**xfs_bmap**

第29章 ファイルシステムのマウント

システム管理者は、システムにファイルシステムをマウントすると、ファイルシステムのデータにアクセスできます。

29.1. LINUX のマウントメカニズム

本セクションでは、Linux でのファイルシステムのマウントに関する基本概念を説明します。

Linux、UNIX、および類似のオペレーティングシステムでは、さまざまなパーティションおよびリムーバブルデバイス (CD、DVD、USB フラッシュドライブなど) にあるファイルシステムをディレクトリーツリーの特定のポイント (マウントポイント) に接続して、再度切り離すことができます。ファイルシステムがディレクトリーにマウントされている間は、そのディレクトリーの元の内容にアクセスすることはできません。

Linux では、ファイルシステムがすでに接続されているディレクトリーにファイルシステムをマウントできます。

マウント時には、次の方法でデバイスを識別できます。

- UUID (universally unique identifier): **UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb** など
- ボリュームラベル - **LABEL=home** など
- 非永続的なブロックデバイスへのフルパス - **/dev/sda3** など

デバイス名、目的のディレクトリー、ファイルシステムタイプなど、必要な情報をすべて指定せずに **mount** コマンドを使用してファイルシステムをマウントすると、**mount** ユーティリティーは **/etc/fstab** ファイルの内容を読み取り、指定のファイルシステムが記載されているかどうかを確認します。**/etc/fstab** ファイルには、選択したファイルシステムがマウントされるデバイス名およびディレクトリーのリスト、ファイルシステムタイプ、およびマウントオプションが含まれます。そのため、**/etc/fstab** で指定されたファイルシステムをマウントする場合は、以下のコマンド構文で十分です。

- マウントポイントによるマウント:

```
# mount directory
```

- ブロックデバイスによるマウント:

```
# mount device
```

関連情報

- **mount(8)** の man ページ
- UUID などの永続的な命名属性のリストを表示する方法。

29.2. 現在マウントされているファイルシステムのリスト表示

この手順では、コマンドラインに、現在マウントされているファイルシステムのリストを表示する方法を説明します。

手順

- マウントされているファイルシステムのリストを表示するには、**findmnt** ユーティリティーを使用します。

```
$ findmnt
```

- リスト表示されているファイルシステムを、特定のファイルシステムタイプに制限するには、**--types** オプションを追加します。

```
$ findmnt --types fs-type
```

以下に例を示します。

例29.1 XFS ファイルシステムのみを表示

```
$ findmnt --types xfs

TARGET  SOURCE                                          FSTYPE OPTIONS
/       /dev/mapper/luks-5564ed00-6aac-4406-bfb4-c59bf5de48b5 xfs    rw,relatime
├─/boot /dev/sda1                                       xfs    rw,relatime
└─/home /dev/mapper/luks-9d185660-7537-414d-b727-d92ea036051e xfs    rw,relatime
```

関連情報

- **findmnt(8)** man ページ

29.3. MOUNT でファイルシステムのマウント

この手順では、**mount** ユーティリティーを使用してファイルシステムをマウントする方法を説明します。

前提条件

- 選択したマウントポイントにファイルシステムがマウントされていない。

```
$ findmnt mount-point
```

手順

1. 特定のファイルシステムを添付する場合は、**mount** ユーティリティーを使用します。

```
# mount device mount-point
```

例29.2 XFS ファイルシステムのマウント

たとえば、UUID により識別されるローカル XFS ファイルシステムをマウントするには、次のコマンドを実行します。

```
# mount UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /mnt/data
```

2. **mount** がファイルシステムタイプを自動的に認識できない場合は、**--types** オプションで指定します。

```
# mount --types type device mount-point
```

例29.3 NFS ファイルシステムのマウント

たとえば、リモートの NFS ファイルシステムをマウントするには、次のコマンドを実行します。

```
# mount --types nfs4 host:/remote-export /mnt/nfs
```

関連情報

- **mount(8)** の man ページ

29.4. マウントポイントの移動

この手順では、マウントされたファイルシステムのマウントポイントを、別のディレクトリーに変更する方法を説明します。

手順

1. ファイルシステムがマウントされているディレクトリーを変更するには、以下のコマンドを実行します。

```
# mount --move old-directory new-directory
```

例29.4 ホームファイルシステムの移動

たとえば、**/mnt/userdirs/** ディレクトリーにマウントされたファイルシステムを **/home/** マウントポイントに移動するには、以下のコマンドを実行します。

```
# mount --move /mnt/userdirs /home
```

2. ファイルシステムが想定どおりに移動したことを確認します。

```
$ findmnt
$ ls old-directory
$ ls new-directory
```

関連情報

- **mount(8)** の man ページ

29.5. UMOUNT でファイルシステムのアンマウント

この手順では、**umount** ユーティリティーを使用してファイルシステムをアンマウントする方法を説明します。

手順

1. 次のいずれかのコマンドを使用してファイルシステムをアンマウントします。

   - マウントポイントで行う場合は、以下のコマンドを実行します。

     ```
     # umount mount-point
     ```

   - デバイスで行う場合は、以下のコマンドを実行します。

     ```
     # umount device
     ```

   コマンドが次のようなエラーで失敗した場合は、プロセスがリソースを使用しているため、ファイルシステムが使用中であることを意味します。

   ```
   umount: /run/media/user/FlashDrive: target is busy.
   ```

2. ファイルシステムが使用中の場合は、**fuser** ユーティリティーを使用して、ファイルシステムにアクセスしているプロセスを特定します。以下に例を示します。

   ```
   $ fuser --mount /run/media/user/FlashDrive

   /run/media/user/FlashDrive: 18351
   ```

   その後、ファイルシステムを使用してプロセスを終了し、マウント解除を再度試みます。

29.6. 一般的なマウントオプション

次の表に、**mount** ユーティリティーの最も一般的なオプションを示します。次の構文を使用して、これらのマウントオプションを適用できます。

```
# mount --options option1,option2,option3 device mount-point
```

表29.1 一般的なマウントオプション

オプション	説明
async	ファイルシステムで非同期の入出力を可能にします。
auto	**mount -a** コマンドを使用したファイルシステムの自動マウントを可能にします。
defaults	オプション **async、auto、dev、exec、nouser、rw、suid** のエイリアスを指定します。
exec	特定のファイルシステムでのバイナリーファイルの実行を許可します。
loop	イメージをループデバイスとしてマウントします。
noauto	デフォルトでは、**mount -a** コマンドを使用したファイルシステムの自動マウントを無効します。

オプション	説明
noexec	特定のファイルシステムでのバイナリーファイルの実行は許可しません。
nouser	普通のユーザー (つまり root 以外のユーザー) によるファイルシステムのマウントおよびアンマウントは許可しません。
remount	ファイルシステムがすでにマウントされている場合は再度マウントを行います。
ro	読み取り専用でファイルシステムをマウントします。
rw	ファイルシステムを読み取りと書き込み両方でマウントします。
user	普通のユーザー (つまり root 以外のユーザー) によるファイルシステムのマウントおよびアンマウントを許可します。

第30章 複数のマウントポイントでのマウント共有

システム管理者は、マウントポイントを複製して、複数のディレクトリーからファイルシステムにアクセスするようにできます。

30.1. 共有マウントのタイプ

使用できる共有マウントには複数のタイプがあります。共有マウントポイントの種類によって、マウントポイントに別のファイルシステムをマウントしたときに発生する内容がなります。共有マウントは、**共有サブツリー** 機能を使用して実装されます。

次のマウントタイプを使用できます。

プライベート

このタイプは、伝播イベントを受信または転送しません。
複製マウントポイントまたは元のマウントポイントのどちらかに別のファイルシステムをマウントしても、それは他方には反映されません。

shared

このタイプは、指定したマウントポイントの正確なレプリカを作成します。
マウントポイントが **shared** マウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントが複製マウントポイントに反映されます (その逆も同様です)。

これは、root ファイルシステムのデフォルトのマウントタイプです。

slave

このタイプは、指定したマウントポイントの限定的な複製を作成します。
マウントポイントが **slave** マウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントがそれに反映されますが、**slave** マウント内のマウントは元のマウントに反映されません。

unbindable

このタイプは、指定のマウントポイントの複製をまったく行いません。

関連情報

- Linux Weekly News の **Shared subtrees** 記事

30.2. プライベートマウントポイントの複製の作成

この手順では、マウントポイントをプライベートマウントとして複製します。複製後に、複製または元のマウントポイントにマウントするファイルシステムは、他方のマウントポイントには反映されません。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントをプライベートとしてマークします。

```
# mount --make-private original-dir
```

あるいは、選択したマウントポイントと、その下のすべてのマウントポイントのマウントタイプを変更するには、**--make-private** ではなく、**--make-rprivate** オプションを使用します。

3. 複製を作成します。

```
# mount --bind original-dir duplicate-dir
```

例30.1 プライベートマウントポイントとして /mnt に /media を複製

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーをプライベートとしてマークします。

```
# mount --make-private /media
```

3. そのコピーを **/mnt** に作成します。

```
# mount --bind /media /mnt
```

4. これで、**/media** と **/mnt** はコンテンツを共有してますが、**/media** 内のマウントはいずれも **/mnt** に現れていないことが確認できます。たとえば、CD-ROM ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
#
```

5. また、**/mnt** ディレクトリーにマウントされているファイルシステムが **/media** に反映されていないことを確認することもできます。たとえば、**/dev/sdc1** デバイスを使用する、空でない USB フラッシュドライブをプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のコマンドを実行します。

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連情報

- **mount(8)** の man ページ

30.3. 共有マウントポイントの複製の作成

この手順では、マウントポイントを共有マウントとして複製します。複製後に、元のディレクトリーまたは複製にマウントしたファイルシステムは、他方のマウントポイントに常に反映されます。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントを共有としてマークします。

```
# mount --make-shared original-dir
```

あるいは、選択したマウントポイントとその下のすべてのマウントポイントのマウントタイプを変更する場合は、**--make-shared** ではなく、**--make-rshared** オプションを使用します。

3. 複製を作成します。

```
# mount --bind original-dir duplicate-dir
```

例30.2 共有マウントポイントとして /mnt に /media を複製

/media ディレクトリーと **/mnt** ディレクトリーが同じコンテンツを共有するようにするには、次の手順を行います。

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーを共有としてマークします。

```
# mount --make-shared /media
```

3. そのコピーを **/mnt** に作成します。

```
# mount --bind /media /mnt
```

4. これで、**/media** 内のマウントが **/mnt** にも現れていることを確認できます。たとえば、CD-ROM ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5. 同様に、**/mnt** ディレクトリー内にマウントされているファイルシステムが **/media** に反映されていることを確認することもできます。たとえば、**/dev/sdc1** デバイスを使用する、空でない USB フラッシュドライブをプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のコマンドを実行します。

```
# mount /dev/sdc1 /mnt/flashdisk
```

```
# ls /media/flashdisk
en-US  publican.cfg
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連情報

- **mount(8)** の man ページ

30.4. スレーブマウントポイントの複製の作成

この手順では、マウントポイントを **slave** マウントタイプとして複製します。複製後に、元のマウントポイントにマウントしたファイルシステムは複製に反映されますが、その逆は反映されません。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントを共有としてマークします。

```
# mount --make-shared original-dir
```

あるいは、選択したマウントポイントとその下のすべてのマウントポイントのマウントタイプを変更する場合は、**--make-shared** ではなく、**--make-rshared** オプションを使用します。

3. 複製を作成し、これを **slave** タイプとしてマークします。

```
# mount --bind original-dir duplicate-dir
# mount --make-slave duplicate-dir
```

例30.3 スレーブマウントポイントとして /mnt に /media を複製

この例は、**/media** ディレクトリーのコンテンツが **/mnt** にも表示され、**/mnt** ディレクトリーのマウントが **/media** に反映されないようにする方法を示しています。

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーを共有としてマークします。

```
# mount --make-shared /media
```

3. その複製を **/mnt** に作成し、**slave** としてマークします。

```
# mount --bind /media /mnt
# mount --make-slave /mnt
```

4. **/media** 内のマウントが **/mnt** にも表示されていることを確認します。たとえば、CD-ROM ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5. また、**/mnt** ディレクトリー内にマウントされているファイルシステムが **/media** に反映されていないことを確認します。たとえば、**/dev/sdc1** デバイスを使用する、空でない USB フラッシュドライブをプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のコマンドを実行します。

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連情報

- **mount(8)** の man ページ

30.5. マウントポイントが複製されないようにする

この手順では、別のマウントポイントに複製されないように、マウントポイントをバインド不可としてマークします。

手順

- マウントポイントのタイプをバインド不可なマウントに変更するには、以下のコマンドを使用します。

```
# mount --bind mount-point mount-point
# mount --make-unbindable mount-point
```

あるいは、選択したマウントポイントとその下のすべてのマウントポイントのマウントタイプを変更する場合は、**--make-unbindable** の代わりに、**--make-runbindable** オプションを使用します。

これ以降、このマウントの複製を作成しようとすると、以下のエラーが出て失敗します。

```
# mount --bind mount-point duplicate-dir

mount: wrong fs type, bad option, bad superblock on mount-point,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail  or so
```

例30.4 /media が複製されないようにする

- **/media** ディレクトリーが共有されないようにするには、以下のコマンドを実行します。

```
# mount --bind /media /media
# mount --make-unbindable /media
```

関連情報

- **mount(8)** の man ページ

第31章 ファイルシステムの永続的なマウント

システム管理者は、ファイルシステムを永続的にマウントして、非リムーバブルストレージを設定できます。

31.1. /ETC/FSTAB ファイル

/etc/fstab 設定ファイルを使用して、ファイルシステムの永続的なマウントポイントを制御します。**/etc/fstab** ファイルの各行は、ファイルシステムのマウントポイントを定義します。

空白で区切られた 6 つのフィールドが含まれています。

1. **/dev** ディレクトリーの永続的な属性またはパスで識別されるブロックデバイス。
2. デバイスがマウントされるディレクトリー。
3. デバイス上のファイルシステム。
4. ファイルシステムのマウントオプション。これには、ブート時にデフォルトオプションでパーティションをマウントする **defaults** オプションが含まれます。マウントオプションフィールドは、**x-systemd.option** 形式の **systemd** マウントユニットオプションも認識します。
5. **dump** ユーティリティーのオプションのバックアップを作成します。
6. **fsck** ユーティリティーの順序を確認します。

注記

systemd-fstab-generator は、エントリーを **/etc/fstab** ファイルから **systemd-mount** ユニットに動的に変換します。**systemd-mount** ユニットがマスクされていない限り、**systemd** は手動アクティベーション中に **/etc/fstab** から LVM ボリュームを自動マウントします。

例31.1 /etc/fstab の /boot ファイルシステム

ブロックデバイス	マウントポイント	ファイルシステム	オプション	バックアップ	チェック
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b	**/boot**	**xfs**	**defaults**	**0**	**0**

systemd サービスは、**/etc/fstab** のエントリーからマウントユニットを自動的に生成します。

関連情報

- **fstab(5)** および **systemd.mount(5)** の man ページ

31.2. /ETC/FSTAB へのファイルシステムの追加

この手順では、**/etc/fstab** 設定ファイルでファイルシステムの永続マウントポイントを設定する方法を説明します。

手順

1. ファイルシステムの UUID 属性を調べます。

```
$ lsblk --fs storage-device
```

以下に例を示します。

例31.2 パーティションの UUID の表示

```
$ lsblk --fs /dev/sda1

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot
```

2. このマウントポイントのディレクトリーがない場合は、作成します。

```
# mkdir --parents mount-point
```

3. root で **/etc/fstab** ファイルを編集し、ファイルシステムに行を追加します (UUID で識別されます)。
以下に例を示します。

例31.3 /etc/fstab の /boot マウントポイント

```
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot xfs defaults 0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

5. ファイルシステムをマウントして、設定が機能することを確認します。

```
# mount mount-point
```

関連情報

- 永続的な命名属性の概要

第32章 RHEL システムロールを使用したファイルシステムの永続的なマウント

ファイルシステムを永続的にマウントするには、**storage** ロールを使用します。

前提条件

- **storage** ロールを使用する Ansible Playbook がある。

32.1. ファイルシステムを永続的にマウントする ANSIBLE PLAYBOOK の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、**storage** ロールをすぐに適用して、XFS ファイルシステムを永続的にマウントします。

例32.1 /dev/sdb のファイルシステムを /mnt/data にマウントする Playbook

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- この Playbook では、ファイルシステムが **/etc/fstab** ファイルに追加され、すぐにファイルシステムをマウントします。
- **/dev/sdb** デバイス上のファイルシステム、またはマウントポイントのディレクトリーが存在しない場合は、Playbook により作成されます。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル

第33章 オンデマンドでのファイルシステムのマウント

システム管理者は、NFS などのファイルシステムをオンデマンドで自動的にマウントするように設定できます。

33.1. AUTOFS サービス

本セクションでは、ファイルシステムをオンデマンドでマウントするのに使用する **autofs** サービスの利点と基本概念を説明します。

/etc/fstab 設定を使用した永続的なマウントの欠点の 1 つは、マウントされたファイルシステムにユーザーがアクセスする頻度に関わらず、マウントされたファイルシステムを所定の場所で維持するために、システムがリソースを割り当てる必要があることです。これは、システムが一度に多数のシステムへの NFS マウントを維持している場合などに、システムのパフォーマンスに影響を与える可能性があります。

/etc/fstab に代わるのは、カーネルベースの **autofs** サービスの使用です。これは以下のコンポーネントで設定されています。

- ファイルシステムを実装するカーネルモジュール
- 他のすべての機能を実行するユーザー空間サービス

autofs サービスは、ファイルシステムの自動マウントおよび自動アンマウントが可能なため (オンデマンド)、システムのリソースを節約できます。このサービスは、NFS、AFS、SMBFS、CIFS、およびローカルなどのファイルシステムをマウントする場合にも使用できます。

関連情報

- man ページの **autofs(8)**

33.2. AUTOFS 設定ファイル

本セクションでは、**autofs** サービスで使用される設定ファイルの使用方法と構文を説明します。

マスターマップファイル

autofs サービスは、デフォルトの主要設定ファイルとして、**/etc/auto.master** (マスターマップ) を使用します。これは、**/etc/autofs.conf** 設定ファイルの **autofs** 設定を Name Service Switch (NSS) メカニズムとともに使用することで、対応している別のネットワークソースと名前を使用するように変更できます。

すべてのオンデマンドマウントポイントはマスターマップで設定する必要があります。マウントポイント、ホスト名、エクスポートされたディレクトリー、オプションはすべて、ホストごとに手動で設定するのではなく、一連のファイル (またはサポートされているその他のネットワークソース) で指定できます。

マスターマップファイルには、**autofs** により制御されるマウントポイントと、それに対応する設定ファイルまたは自動マウントマップと呼ばれるネットワークソースがリスト表示されます。マスターマップの形式は次のとおりです。

```
mount-point  map-name  options
```

この形式で使用されている変数を以下に示します。

mount-point

autofs マウントポイント (例: **/mnt/data/**) です。

map-file

マウントポイントのリストと、マウントポイントがマウントされるファイルシステムの場所が記載されているマップソースファイルです。

options

指定した場合に、エントリーにオプションが指定されていなければ、指定されたマップ内のすべてのエントリーに適用されます。

例33.1 /etc/auto.master ファイル

以下は **/etc/auto.master** ファイルのサンプル行です。

```
/mnt/data  /etc/auto.data
```

マップファイル

マップファイルは、個々のオンデマンドマウントポイントのプロパティーを設定します。

ディレクトリーが存在しない場合、自動マウント機能はディレクトリーを作成します。ディレクトリーが存在している状況で自動マウント機能が起動した場合は、自動マウント機能の終了時にディレクトリーが削除されることはありません。タイムアウトを指定した場合は、タイムアウト期間中ディレクトリーにアクセスしないと、ディレクトリーが自動的にアンマウントされます。

マップの一般的な形式は、マスターマップに似ています。ただし、マスターマップでは、オプションフィールドはエントリーの末尾ではなく、マウントポイントと場所の間に表示されます。

```
mount-point  options  location
```

この形式で使用されている変数を以下に示します。

mount-point

これは、**autofs** のマウントポイントを参照しています。これは 1 つのインダイレクトマウント用の 1 つのディレクトリー名にすることも、複数のダイレクトマウント用のマウントポイントの完全パスにすることもできます。ダイレクトマップとインダイレクトマップの各エントリーキー (**mount-point**) の後に空白で区切られたオフセットディレクトリー (/ で始まるサブディレクトリー名) が記載されます。これがマルチマウントエントリーと呼ばれるものです。

options

このオプションを指定すると、マスターマップエントリーのオプション (存在する場合) に追加されます。設定エントリーの **append_options** が **no** に設定されている場合は、マスターマップのオプションの代わりにこのオプションが使用されます。

location

ローカルファイルシステムのパス (Sun マップ形式のエスケープ文字 **:** が先頭に付き、マップ名が / で始まります)、NFS ファイルシステム、他の有効なファイルシステムの場所などのファイルシステムの場所を参照します。

例33.2 マップファイル

以下は、マップファイルのサンプルです (例: **/etc/auto.misc**)。

```
payroll  -fstype=nfs4  personnel:/exports/payroll
sales    -fstype=xfs   :/dev/hda4
```

マップファイルの最初の列は、**autofs** マウントポイント (**personnel** サーバーからの **sales** と **payroll**) を示しています。2 列目は、**autofs** マウントのオプションを示しています。3 列目はマウントのソースを示しています。

任意の設定に基づき、**autofs** マウントポイントは、**/home/payroll** と **/home/sales** になります。**-fstype=** オプションは多くの場合省略されており、ファイルシステムが NFS の場合は必要ありません。これには、システムのデフォルトが NFS マウント用の NFSv4 である場合の NFSv4 のマウントも含まれます。

与えられた設定を使用して、プロセスが **/home/payroll/2006/July.sxc** などのアンマウントされたディレクトリー **autofs** へのアクセスを要求すると、**autofs** サービスは自動的にディレクトリーをマウントします。

amd マップ形式

autofs サービスは、**amd** 形式のマップ設定も認識します。これは Red Hat Enterprise Linux から削除された、**am-utils** サービス用に書き込まれた既存の自動マウント機能の設定を再利用する場合に便利です。

ただし、Red Hat は、前述のセクションで説明した簡単な **autofs** 形式の使用を推奨しています。

関連情報

- **autofs(5)** man ページ
- **autofs.conf(5)** man ページ
- **auto.master(5)** man ページ
- **/usr/share/doc/autofs/README.amd-maps** ファイル

33.3. AUTOFS マウントポイントの設定

この手順では、**autofs** サービスを使用してオンデマンドマウントポイントを設定する方法を説明します。

前提条件

- **autofs** パッケージをインストールしている。

```
# yum install autofs
```

- **autofs** サービスを起動して有効にしている。

```
# systemctl enable --now autofs
```

手順

1. **/etc/auto.identifier** にあるオンデマンドマウントポイント用のマップファイルを作成します。**identifier** を、マウントポイントを識別する名前に置き換えます。

2. マップファイルで、autofs 設定ファイル の説明に従って、マウントポイント、オプション、および場所の各フィールドを入力します。

3. autofs 設定ファイル セクションの説明に従って、マップファイルをマスターマップファイルに登録します。

4. 設定の再読み込みを許可し、新しく設定した **autofs** マウントを管理できるようにします。

```
# systemctl reload autofs.service
```

5. オンデマンドディレクトリーのコンテンツへのアクセスを試みます。

```
# ls automounted-directory
```

33.4. AUTOFS サービスを使用した NFS サーバーユーザーのホームディレクトリーの自動マウント

この手順では、ユーザーのホームディレクトリーを自動的にマウントするように autofs サービスを設定する方法を説明します。

前提条件

- autofs パッケージがインストールされている。
- autofs サービスが有効で、実行している。

手順

1. ユーザーのホームディレクトリーをマウントする必要があるサーバーの **/etc/auto.master** ファイルを編集して、マップファイルのマウントポイントと場所を指定します。これを行うには、以下の行を **/etc/auto.master** ファイルに追加します。

```
/home /etc/auto.home
```

2. ユーザーのホームディレクトリーをマウントする必要があるサーバー上で、**/etc/auto.home** という名前のマップファイルを作成し、以下のパラメーターでファイルを編集します。

```
* -fstype=nfs,rw,sync host.example.com:/home/&
```

fstype パラメーターはデフォルトで **nfs** であるため、このパラメーターは飛ばして次に進むことができます。詳細は、**autofs(5)** man ページを参照してください。

3. **autofs** サービスを再読み込みします。

```
# systemctl reload autofs
```

33.5. AUTOFS サイトの設定ファイルの上書き/拡張

クライアントシステムの特定のマウントポイントで、サイトのデフォルトを上書きすることが役に立つ場合があります。

例33.3 初期条件

たとえば、次の条件を検討します。

- 自動マウント機能マップは NIS に保存され、**/etc/nsswitch.conf** ファイルには以下のディレクティブがあります。

```
automount:    files nis
```

- **auto.master** ファイルには以下が含まれます。

```
+auto.master
```

- NIS の **auto.master** マップファイルに以下が含まれます。

```
/home auto.home
```

- NIS の **auto.home** マップには以下が含まれます。

```
beth    fileserver.example.com:/export/home/beth
joe     fileserver.example.com:/export/home/joe
*       fileserver.example.com:/export/home/&
```

- **autofs** 設定オプションの **BROWSE_MODE** は **yes** に設定されています。

```
BROWSE_MODE="yes"
```

- ファイルマップ **/etc/auto.home** は存在しません。

手順

本セクションでは、別のサーバーからホームディレクトリーをマウントし、選択したエントリーのみで **auto.home** を強化する例を説明します。

例33.4 別のサーバーからのホームディレクトリーのマウント

上記の条件で、クライアントシステムが NIS マップの **auto.home** を上書きして、別のサーバーからホームディレクトリーをマウントする必要があるとします。

- この場合、クライアントは次の **/etc/auto.master** マップを使用する必要があります。

```
/home /etc/auto.home
+auto.master
```

- **/etc/auto.home** マップにエントリーが含まれています。

```
*    host.example.com:/export/home/&
```

自動マウント機能は最初に出現したマウントポイントのみを処理するため、**/home** ディレクトリーには NIS **auto.home** マップではなく、**/etc/auto.home** の内容が含まれます。

例33.5 選択されたエントリーのみを使用した auto.home の拡張

別の方法として、サイト全体の **auto.home** マップを少しのエントリーを使用して拡張するには、次の手順を行います。

1. **/etc/auto.home** ファイルマップを作成し、そこに新しいエントリーを追加します。最後に、NIS の **auto.home** マップを含めます。これにより、**/etc/auto.home** ファイルマップは次のようになります。

```
mydir someserver:/export/mydir
+auto.home
```

2. この NIS の **auto.home** マップ条件で、**/home** ディレクトリーの出力内容をリスト表示すると次のようになります。

```
$ ls /home

beth joe mydir
```

autofs は、読み取り中のファイルマップと同じ名前のファイルマップの内容を組み込まないため、上記の例は期待どおりに動作します。このように、**autofs** は、**nsswitch** 設定内の次のマップソースに移動します。

33.6. LDAP で自動マウント機能マップの格納

この手順では、**autofs** マップファイルではなく、LDAP 設定で自動マウント機能マップを格納するように **autofs** を設定します。

前提条件

- LDAP から自動マウント機能マップを取得するように設定されているすべてのシステムに、LDAP クライアントライブラリーをインストールする必要があります。Red Hat Enterprise Linux では、**openldap** パッケージは、**autofs** パッケージの依存関係として自動的にインストールされます。

手順

1. LDAP アクセスを設定するには、**/etc/openldap/ldap.conf** ファイルを変更します。**BASE**、**URI**、**schema** の各オプションがサイトに適切に設定されていることを確認します。

2. 自動マウント機能マップを LDAP に格納するためにデフォルトされた最新のスキーマが、**rfc2307bis** ドラフトに記載されています。このスキーマを使用する場合は、スキーマの定義のコメント文字を取り除き、**/etc/autofs.conf** 設定ファイル内に設定する必要があります。以下に例を示します。

例33.6 autofs の設定

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

3. 他のすべてのスキーマエントリーが設定内でコメントされていることを確認してください。**rfc2307bis** スキーマの **automountKey** 属性は、**rfc2307** スキーマの **cn** 属性に置き換わります。以下は、LDAP データ交換形式 (LDIF) 設定の例です。

例33.7 LDIF 設定

```
# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.master

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
automountKey: /home
automountInformation: auto.home

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: foo
automountInformation: filer.example.com:/export/foo

# /, auto.home, example.com
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

関連情報

- **rfc2307bis** ドラフト

33.7. SYSTEMD.AUTOMOUNT を使用して、/ETC/FSTAB を使用してオンデマンドでファイルシステムをマウントします

この手順は、マウントポイントが **/etc/fstab** で定義されている場合に、automount systemd ユニットを使用してオンデマンドでファイルシステムをマウントする方法を示しています。マウントごとに自動マウントユニットを追加して有効にする必要があります。

手順

1. 第 30 章 ファイルシステムの永続的なマウント に記載されているように、必要な fstab エントリーを追加します。以下に例を示します。

```
/dev/disk/by-id/da875760-edb9-4b82-99dc-5f4b1ff2e5f4  /mount/point  xfs  defaults  0 0
```

2. 前の手順で作成したエントリーの options フィールドに **x-systemd.automount** を追加します。
3. システムが新しい設定を登録するように、新しく作成されたユニットをロードします。

```
# systemctl daemon-reload
```

4. 自動マウントユニットを起動します。

```
# systemctl start mount-point.automount
```

検証

1. **mount-point.automount** が実行されていることを確認します。

```
# systemctl status mount-point.automount
```

2. 自動マウントされたディレクトリーに目的のコンテンツが含まれていることを確認します。

```
# ls /mount/point
```

関連情報

- **systemd.automount(5)** man ページ
- **systemd.mount(5)** man ページ
- systemd の概要

33.8. SYSTEMD.AUTOMOUNT を使用して、マウントユニットを使用してファイルシステムをオンデマンドでマウントします

この手順は、マウントポイントがマウントユニットによって定義されている場合に、automount systemd ユニットを使用してオンデマンドでファイルシステムをマウントする方法を示しています。マウントごとに自動マウントユニットを追加して有効にする必要があります。

手順

1. マウントユニットを作成します。以下に例を示します。

```
mount-point.mount
[Mount]
What=/dev/disk/by-uuid/f5755511-a714-44c1-a123-cfde0e4ac688
Where=/mount/point
Type=xfs
```

2. マウントユニットと同じ名前で、拡張子が **.automount** のユニットファイルを作成します。
3. ファイルを開き、**[Automount]** セクションを作成します。**Where=** オプションをマウントパスに設定します。

```
[Automount]
Where=/mount/point
```

```
[Install]
WantedBy=multi-user.target
```

4. システムが新しい設定を登録するように、新しく作成されたユニットをロードします。

```
# systemctl daemon-reload
```

5. 代わりに、自動マウントユニットを有効にして起動します。

```
# systemctl enable --now mount-point.automount
```

検証

1. **mount-point.automount** が実行されていることを確認します。

```
# systemctl status mount-point.automount
```

2. 自動マウントされたディレクトリーに目的のコンテンツが含まれていることを確認します。

```
# ls /mount/point
```

関連情報

- **systemd.automount(5)** man ページ
- **systemd.mount(5)** man ページ
- systemd の概要

第34章 IDM からの SSSD コンポーネントを使用した AUTOFS マップのキャッシュ

システムセキュリティーサービスデーモン (System Security Services Daemon: SSSD) は、リモートサービスディレクトリーと認証メカニズムにアクセスするシステムサービスです。データキャッシュは、ネットワーク接続が遅い場合に役立ちます。SSSD サービスが autofs マップをキャッシュするように設定するには、本セクションの以下の手順に従います。

34.1. IDM サーバーを LDAP サーバーとして使用するように AUTOFS を手動で設定する

この手順では、IdM サーバーを LDAP サーバーとして使用するように **autofs** を設定する方法を説明します。

手順

1. **/etc/autofs.conf** ファイルを編集し、**autofs** が検索するスキーマ属性を指定します。

```
#
# Other common LDAP naming
#
map_object_class = "automountMap"
entry_object_class = "automount"
map_attribute = "automountMapName"
entry_attribute = "automountKey"
value_attribute = "automountInformation"
```

注記

ユーザーは、**/etc/autofs.conf** ファイルに小文字と大文字の両方で属性を書き込むことができます。

2. オプションで、LDAP 設定を指定します。これには 2 通りの方法があります。最も簡単な方法は、自動マウントサービスが LDAP サーバーと場所を自分で発見するようにすることです。

```
ldap_uri = "ldap:///dc=example,dc=com"
```

このオプションでは、DNS に検出可能なサーバーの SRV レコードが含まれている必要があります。

別の方法では、使用する LDAP サーバーと LDAP 検索のベース DN を明示的に設定します。

```
ldap_uri = "ldap://ipa.example.com"
search_base = "cn=location,cn=automount,dc=example,dc=com"
```

3. autofs が IdM LDAP サーバーによるクライアント認証を許可するように **/etc/autofs_ldap_auth.conf** ファイルを編集します。

- **authrequired** を yes に変更します。

- プリンシパルを IdM LDAP サーバー (**host/fqdn@REALM**) の Kerberos ホストプリンシパルに設定します。プリンシパル名は、GSS クライアント認証の一部として IdM ディレクトリーへの接続に使用されます。

```
<autofs_ldap_sasl_conf
    usetls="no"
    tlsrequired="no"
    authrequired="yes"
    authtype="GSSAPI"
    clientprinc="host/server.example.com@EXAMPLE.COM"
    />
```

ホストプリンシパルの詳細は、IdM での正規化された DNS ホスト名の使用 を参照してください。

必要に応じて **klist -k** を実行して、正確なホストプリンシパル情報を取得します。

34.2. AUTOFS マップをキャッシュする SSSD の設定

SSSD サービスを使用すると、IdM サーバーに保存されている **autofs** マップを、IdM サーバーを使用するように **autofs** を設定することなくキャッシュできます。

前提条件

- **sssd** パッケージがインストールされている。

手順

1. SSSD 設定ファイルを開きます。

```
# vim /etc/sssd/sssd.conf
```

2. SSSD が処理するサービスリストに **autofs** サービスを追加します。

```
[sssd]
domains = ldap
services = nss,pam,autofs
```

3. **[autofs]** セクションを新規作成します。**autofs** サービスのデフォルト設定はほとんどのインフラストラクチャーに対応するため、これを空白のままにすることができます。

```
[nss]

[pam]

[sudo]

[autofs]

[ssh]

[pac]
```

詳細は man ページの **sssd.conf** を参照してください。

4. オプションとして、**autofs** エントリーの検索ベースを設定します。デフォルトでは、これは LDAP 検索ベースですが、**ldap_autofs_search_base** パラメーターでサブツリーを指定できます。

```
[domain/EXAMPLE]

ldap_search_base = "dc=example,dc=com"
ldap_autofs_search_base = "ou=automount,dc=example,dc=com"
```

5. SSSD サービスを再起動します。

```
# systemctl restart sssd.service
```

6. SSSD が自動マウント設定のソースとしてリスト表示されるように、**/etc/nsswitch.conf** ファイルを確認します。

```
automount: sss files
```

7. **autofs** サービスを再起動します。

```
# systemctl restart autofs.service
```

8. **/home** のマスターマップエントリーがあると想定し、ユーザーの **/home** ディレクトリーをリスト表示して設定をテストします。

```
# ls /home/userName
```

リモートファイルシステムをマウントしない場合は、**/var/log/messages** ファイルでエラーを確認します。必要に応じて、**logging** パラメーターを **debug** に設定して、**/etc/sysconfig/autofs** ファイルのデバッグレベルを増やします。

第35章 ROOT ファイルシステムに対する読み取り専用パーミッションの設定

場合によっては、root ファイルシステム (/) を読み取り専用パーミッションでマウントする必要があります。ユースケースの例には、システムの予期せぬ電源切断後に行うセキュリティーの向上またはデータ整合性の保持が含まれます。

35.1. 書き込みパーミッションを保持するファイルおよびディレクトリー

システムが正しく機能するためには、一部のファイルやディレクトリーで書き込みパーミッションが必要とされます。root ファイルシステムが読み取り専用モードでマウントされると、このようなファイルは、**tmpfs** 一時ファイルシステムを使用して RAM にマウントされます。

このようなファイルおよびディレクトリーのデフォルトセットは、**/etc/rwtab** ファイルから読み込まれます。このファイルをシステムに存在させるには、**readonly-root** パッケージが必要であることに注意してください。

```
dirs /var/cache/man
dirs /var/gdm
<content truncated>

empty /tmp
empty /var/cache/foomatic
<content truncated>

files /etc/adjtime
files /etc/ntp.conf
<content truncated>
```

/etc/rwtab ファイルーのエントリーは次の形式になります。

```
copy-method    path
```

この構文で、以下のことを行います。

- **copy-method** を、ファイルまたはディレクトリーを tmpfs にコピーする方法を指定するキーワードの 1 つに置き換えます。
- **path** を、ファイルまたはディレクトリーへのパスに置き換えます。

/etc/rwtab ファイルは、ファイルまたはディレクトリーを **tmpfs** にコピーする方法として以下を認識します。

empty

空のパスが **tmpfs** にコピーされます。以下に例を示します。

```
empty /tmp
```

dirs

ディレクトリーツリーが空の状態で **tmpfs** にコピーされます。以下に例を示します。

```
dirs /var/run
```

files

ファイルやディレクトリーツリーはそのまま **tmpfs** にコピーされます。以下に例を示します。

```
files /etc/resolv.conf
```

カスタムパスを **/etc/rwtab.d/** に追加する場合も同じ形式が適用されます。

35.2. ブート時に読み取り専用パーミッションでマウントするように ROOT ファイルシステムの設定

この手順を行うと、今後システムが起動するたびに、root ファイルシステムが読み取り専用としてマウントされます。

手順

1. **/etc/sysconfig/readonly-root** ファイルで、**READONLY** オプションを **yes** に設定します。

```
# Set to 'yes' to mount the file systems as read-only.
READONLY=yes
```

2. **/etc/fstab** ファイルの root エントリー (/) に **ro** オプションを追加します。

```
/dev/mapper/luks-c376919e... /  xfs  x-systemd.device-timeout=0,ro  1  1
```

3. **ro** kernel オプションを有効にします。

```
# grubby --update-kernel=ALL --args="ro"
```

4. **rw** カーネルオプションが無効になっていることを確認します。

```
# grubby --update-kernel=ALL --remove-args="rw"
```

5. **tmpfs** ファイルシステムに書き込みパーミッションでマウントするファイルとディレクトリーを追加する必要がある場合は、**/etc/rwtab.d/** ディレクトリーにテキストファイルを作成し、そこに設定を置きます。
たとえば、**/etc/example/file** ファイルを書き込みパーミッションでマウントするには、この行を **/etc/rwtab.d/example** ファイルに追加します。

```
files /etc/example/file
```

重要

tmpfs のファイルおよびディレクトリーの変更内容は、再起動後は持続しません。

6. システムを再起動して変更を適用します。

トラブルシューティング

- 誤って読み取り専用パーミッションで root ファイルシステムをマウントした場合は、次のコマンドを使用して、読み書きパーミッションで再度マウントできます。

```
# mount -o remount,rw /
```

第36章 ストレージデバイスの管理

36.1. STRATIS ファイルシステムの設定

Stratis は、物理ストレージデバイスのプールを管理するためにサービスとして実行され、複雑なストレージ設定のセットアップと管理を支援しながら、ローカルストレージ管理を使いやすく簡素化します。

重要

Stratis はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

36.1.1. Stratis とは

Stratis は、Linux 用のローカルストレージ管理ソリューションです。これは、シンプルさと使いやすさに力を入れており、高度なストレージ機能にアクセスできます。

Stratis を使用すると、以下の活動をより簡単に行うことができます。

- ストレージの初期設定
- その後の変更
- 高度なストレージ機能の使用

Stratis は、高度なストレージ機能に対応する、ユーザーとカーネルのハイブリッドローカルストレージ管理システムです。Stratis は、ストレージ **プール** の概念を中心としています。このプールは 1 つ以上のローカルディスクまたはパーティションから作成され、ボリュームはプールから作成されます。

プールにより、次のような多くの便利な機能を使用できます。

- ファイルシステムのスナップショット
- シンプロビジョニング
- 階層化

関連情報

- Stratis Web サイト

36.1.2. Stratis ボリュームの設定要素

Stratis ボリュームを設定するコンポーネントについて説明します。

外部的には、Stratis は、コマンドラインインターフェイスおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで設定されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は、各プールの **/dev/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。
ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratis に更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再設定しないでください。

Stratis は、**/dev/stratis/my-pool/my-fs** パスにファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映します。

36.1.3. Stratis で使用可能なブロックデバイス

Stratis で使用可能なストレージデバイス。

対応デバイス
Stratis プールは、次の種類のブロックデバイスで動作するかどうかをテスト済みです。

- LUKS
- LVM 論理ボリューム
- MD RAID
- DM Multipath
- iSCSI

- HDD および SSD
- NVMe デバイス

対応していないデバイス
Stratis にはシンプロビジョニングレイヤーが含まれているため、Red Hat はすでにシンプロビジョニングされているブロックデバイスに Stratis プールを配置することを推奨しません。

36.1.4. Stratis のインストール

Stratis に必要なパッケージをインストールします。

手順

1. Stratis サービスとコマンドラインユーティリティーを提供するパッケージをインストールします。

```
# yum install stratisd stratis-cli
```

2. **stratisd** サービスが有効になっていることを確認します。

```
# systemctl enable --now stratisd
```

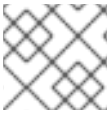
36.1.5. 暗号化されていない Stratis プールの作成

1 つ以上のブロックデバイスから暗号化されていない Stratis プールを作成できます。

前提条件

- Stratis がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis プールを作成するブロックデバイスは使用されておらず、マウントされていない。
- Stratis プールを作成する各ブロックデバイスが、1 GB 以上である。
- IBM Z アーキテクチャーでは、**/dev/dasd*** ブロックデバイスをパーティションに分割している。Stratis プールでパーティションを使用します。

DASD デバイスのパーティション分割については、IBM Z への Linux インスタンスの設定 を参照してください。



注記

暗号化されていない Stratis プールを暗号化することはできません。

手順

1. Stratis プールで使用する各ブロックデバイスに存在するファイルシステム、パーティションテーブル、または RAID 署名をすべて削除します。

```
# wipefs --all block-device
```

ここで、**block-device** は、ブロックデバイスへのパスになります (例: **/dev/sdb**)。

2. 選択したブロックデバイスに新しい暗号化されていない Stratis プールを作成します。

```
# stratis pool create my-pool block-device
```

ここで、**block-device** は、空のブロックデバイスまたは消去したブロックデバイスへのパスになります。

注記

1 行に複数のブロックデバイスを指定します。

```
# stratis pool create my-pool block-device-1 block-device-2
```

3. 新しい Stratis プールが作成されていることを確認します。

```
# stratis pool list
```

36.1.6. 暗号化された Stratis プールの作成

データを保護するには、1 つ以上のブロックデバイスから暗号化された Stratis プールを作成します。

暗号化された Stratis プールを作成すると、カーネルキーリングはプライマリー暗号化メカニズムとして使用されます。その後のシステムを再起動すると、このカーネルキーリングは、暗号化された Stratis プールのロックを解除します。

1 つ以上のブロックデバイスから暗号化された Stratis プールを作成する場合は、次の点に注意してください。

- 各ブロックデバイスは **cryptsetup** ライブラリーを使用して暗号化され、**LUKS2** 形式を実装します。
- 各 Stratis プールは、一意の鍵を持つか、他のプールと同じ鍵を共有できます。これらのキーはカーネルキーリングに保存されます。
- Stratis プールを設定するブロックデバイスは、すべて暗号化または暗号化されていないデバイスである必要があります。同じ Stratis プールに、暗号化したブロックデバイスと暗号化されていないブロックデバイスの両方を含めることはできません。
- 暗号化 Stratis プールのデータ層に追加されるブロックデバイスは、自動的に暗号化されます。

前提条件

- Stratis v2.1.0 以降がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis プールを作成するブロックデバイスは使用されておらず、マウントされていない。
- Stratis プールを作成するブロックデバイスが、それぞれ 1GB 以上である。

- IBM Z アーキテクチャーでは、**/dev/dasd*** ブロックデバイスをパーティションに分割している。Stratis プールでパーティションを使用します。

DASD デバイスのパーティション分割については、IBM Z への Linux インスタンスの設定 を参照してください。

手順

1. Stratis プールで使用する各ブロックデバイスに存在するファイルシステム、パーティションテーブル、または RAID 署名をすべて削除します。

```
# wipefs --all block-device
```

ここで、**block-device** は、ブロックデバイスへのパスになります (例: **/dev/sdb**)。

2. キーセットをまだ作成していない場合には、以下のコマンドを実行してプロンプトに従って、暗号化に使用するキーセットを作成します。

```
# stratis key set --capture-key key-description
```

ここでの **key-description** は、カーネルキーリングで作成されるキーへの参照になります。

3. 暗号化した Stratis プールを作成し、暗号化に使用する鍵の説明を指定します。**key-description** オプションを使用する代わりに、**--keyfile-path** オプションを使用してキーパスを指定することもできます。

```
# stratis pool create --key-desc key-description my-pool block-device
```

ここでは、以下のようになります。

key-description
直前の手順で作成したカーネルキーリングに存在するキーを参照します。

my-pool
新しい Stratis プールの名前を指定します。

block-device
空のブロックデバイスまたは消去したブロックデバイスへのパスを指定します。

注記

1 行に複数のブロックデバイスを指定します。

```
# stratis pool create --key-desc key-description my-pool block-device-1
block-device-2
```

4. 新しい Stratis プールが作成されていることを確認します。

```
# stratis pool list
```

36.1.7. Stratis ファイルシステムでのシンプロビジョニング層の設定

ストレージスタックは、オーバープロビジョニングの状態になる可能性があります。ファイルシステムのサイズが、そのファイルシステムをサポートするプールよりも大きい場合には、プールがいっぱいになります。これを回避するには、オーバープロビジョニングを無効にし、プール上のすべてのファイルシステムのサイズが、プールが提供する利用可能な物理ストレージを超えないようにします。重要なアプリケーションまたは root ファイルシステムに Stratis を使用する場合は、このモードでは特定の障害ケースが阻止されます。

オーバープロビジョニングを有効にすると、ストレージが完全に割り当てられたことを API シグナルに通知します。通知は、残りのプールスペースがすべていっぱいになると、Stratis に拡張するスペースが残っていないことをユーザーに通知する警告として機能します。

前提条件

- Stratis がインストールされている。詳細は、Stratis のインストール を参照してください。

手順

プールを正しく設定するには、次の 2 つの方法があります。

1. 1 つ以上のブロックデバイスからプールを作成します。

```
# stratis pool create --no-overprovision pool-name /dev/sdb
```

   - **--no-overprovision** オプションを使用すると、プールは実際に利用可能な物理領域よりも多くの論理領域を割り当てることができません。

2. 既存のプールにオーバープロビジョニングモードを設定します。

```
# stratis pool overprovision pool-name <yes|no>
```

   - yes に設定すると、プールへのオーバープロビジョニングが有効になります。これは、プールによってサポートされる Stratis ファイルシステムの論理サイズの合計が、利用可能なデータ領域の量を超える可能性があることを意味します。

検証

1. 以下のコマンドを実行し、Stratis プールの全一覧を表示します。

```
# stratis pool list

Name         Total Physical                Properties   UUID                                 Alerts
pool-name    1.42 TiB / 23.96 MiB / 1.42 TiB  ~Ca,~Cr,~Op  cb7cb4d8-9322-4ac4-a6fd-eb7ae9e1e540
```

2. ubuntu pool **list** の出力に、プールのオーバープロビジョニングモードフラグが表示されているかどうかを確認します。~ は NOT を表す数学記号であるため、**~Op** はオーバープロビジョニングなしという意味です。

3. オプション:以下のコマンドを実行して、特定のプールでオーバープロビジョニングを確認します。

```
# stratis pool overprovision pool-name yes

# stratis pool list
```

```
Name         Total Physical                 Properties   UUID                                   Alerts
pool-name    1.42 TiB / 23.96 MiB / 1.42 TiB  ~Ca,~Cr,~Op  cb7cb4d8-9322-4ac4-a6fd-eb7ae9e1e540
```

関連情報

- Stratis Storage の Web ページ

36.1.8. Stratis プールの NBDE へのバインド

暗号化された Stratis プールを Network Bound Disk Encryption (NBDE) にバインドするには、Tang サーバーが必要です。Stratis プールを含むシステムが再起動すると、Tang サーバーに接続して、カーネルキーリングの説明を指定しなくても、暗号化したプールのロックを自動的に解除します。

注記

Stratis プールを補助 Clevis 暗号化メカニズムにバインドすると、プライマリーカーネルキーリング暗号化は削除されません。

前提条件

- Stratis v2.3.0 以降がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- 暗号化した Stratis プールを作成し、暗号化に使用されたキーの説明がある。詳細は、暗号化された Stratis プールの作成 を参照してください。
- Tang サーバーに接続できる。詳細は、SELinux を Enforcing モードで有効にした Tang サーバーのデプロイメント を参照してください。

手順

- 暗号化された Stratis プールを NBDE にバインドする。

```
# stratis pool bind nbde --trust-url my-pool tang-server
```

ここでは、以下のようになります。

my-pool
: 暗号化された Stratis プールの名前を指定します。

tang-server
: Tang サーバーの IP アドレスまたは URL を指定します。

関連情報

- ポリシーベースの復号を使用して暗号化ボリュームの自動アンロックの設定

36.1.9. Stratis プールの TPM へのバインド

暗号化 Stratis プールを Trusted Platform Module (TPM) 2.0 にバインドすると、プールを含むシステムの再起動時に、カーネルキーリングの説明を提供することなくプールが自動的アンロックされます。

前提条件

- Stratis v2.3.0 以降がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- 暗号化された Stratis プールを作成している。詳細は、暗号化された Stratis プールの作成 を参照してください。

手順

- 暗号化された Stratis プールを TPM にバインドします。

```
# stratis pool bind tpm my-pool key-description
```

ここでは、以下のようになります。

my-pool
暗号化された Stratis プールの名前を指定します。

key-description
暗号化された Stratis プールの作成時に生成されたカーネルキーリングに存在するキーを参照します。

36.1.10. カーネルキーリングを使用した暗号化 Stratis プールのロック解除

システムの再起動後、暗号化した Stratis プール、またはこれを設定するブロックデバイスが表示されない場合があります。プールの暗号化に使用したカーネルキーリングを使用して、プールのロックを解除できます。

前提条件

- Stratis v2.1.0 がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- 暗号化された Stratis プールを作成している。詳細は、暗号化された Stratis プールの作成 を参照してください。

手順

1. 以前使用したものと同じキー記述を使用して、キーセットを再作成します。

```
# stratis key set --capture-key key-description
```

ここで、**key-description** は、暗号化された Stratis プールの作成時に生成されたカーネルキーリングに存在するキーを参照します。

2. Stratis プールと、それを設定するブロックデバイスをアンロックします。

```
# stratis pool unlock keyring
```

3. Stratis プールが表示されることを確認します。

```
# stratis pool list
```

36.1.11. Clevis を使用した暗号化された Stratis プールのロック解除

システムの再起動後、暗号化した Stratis プール、またはこれを設定するブロックデバイスが表示されない場合があります。プールがバインドされている補助暗号化メカニズムを使用して、暗号化した Stratis プールをアンロックできます。

前提条件

- Stratis v2.3.0 以降がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- 暗号化された Stratis プールを作成している。詳細は、暗号化された Stratis プールの作成 を参照してください。
- 暗号化した Stratis プールは、サポート対象の補助暗号化メカニズムにバインドされます。詳細は、暗号化された Stratis プールを NBDE にバインドする を参照してください。

または、暗号化された Stratis プールの TPM へのバインド を参照してください。

手順

1. Stratis プールと、それを設定するブロックデバイスをアンロックします。

```
# stratis pool unlock clevis
```

2. Stratis プールが表示されることを確認します。

```
# stratis pool list
```

36.1.12. 補助暗号化からの Stratis プールのバインド解除

暗号化した Stratis プールを、サポート対象の補助暗号化メカニズムからバインドを解除すると、プライマリーカーネルキーリングの暗号化はそのまま残ります。

前提条件

- Stratis v2.3.0 以降がシステムにインストールされている。詳細は、Stratis のインストール を参照してください。
- 暗号化された Stratis プールを作成している。詳細は、暗号化された Stratis プールの作成 を参照してください。
- 暗号化した Stratis プールは、サポート対象の補助暗号化メカニズムにバインドされます。

手順

- 補助暗号化メカニズムから暗号化された Stratis プールのバインドを解除します。

```
# stratis pool unbind clevis my-pool
```

ここでは、以下のようになります。

my-pool は、バインドを解除する Stratis プールの名前を指定します。

関連情報

- 暗号化された Stratis プールの NBDE へのバインド
- 暗号化された Stratis プールの TPM へのバインド

36.1.13. Stratis プールの開始および停止

Stratis プールを開始および停止できます。これにより、ファイルシステム、キャッシュデバイス、シンプール、暗号化されたデバイスなど、プールの構築に使用されたすべてのオブジェクトをオプションとして分解するか、停止できます。プールがデバイスまたはファイルシステムをアクティブに使用している場合は、警告が表示され、停止できない可能性があることに注意してください。

停止したプールは、停止状態をメタデータに記録します。これらのプールは、プールが開始コマンドを受信するまで、次のブートでは開始されません。

暗号化されていない場合、以前に開始されたプールは起動時に自動的に開始されます。このバージョンの Stratis では、プールの **pool unlock** が **pool start** に置き換えられるため、暗号化されたプールには起動時に常に **pool start** コマンドが必要です。

前提条件

- Stratis がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- 暗号化されていない、または暗号化された Stratis プールを作成している。暗号化されていない Stratis プールの作成 を参照してください。

または、暗号化された Stratis プールの作成 を参照してください。

手順

- 以下のコマンドを使用して Stratis プールを起動します。**--unlock-method** オプションは、プールが暗号化されている場合にプールのロックを解除する方法を指定します。

```
# stratis pool start pool-uuid --unlock-method <keyring|clevis>
```

- または、以下のコマンドを使用して Stratis プールを停止します。これにより、ストレージスタックが切断されますが、メタデータはすべて保持されます。

```
# stratis pool stop pool-name
```

検証手順

- 以下のコマンドを使用して、システム上のプールを一覧表示します。

```
# stratis pool list
```

- 以下のコマンドを使用して、以前に起動していないプールの一覧を表示します。UUID を指定すると、このコマンドは UUID に対応するプールに関する詳細情報を出力します。

```
# stratis pool list --stopped --uuid UUID
```

36.1.14. Stratis ファイルシステムの作成

既存の Stratis プールに Stratis ファイルシステムを作成します。

前提条件

- Stratis がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis プールを作成している。暗号化されていない Stratis プールの作成 を参照してください。

または、暗号化された Stratis プールの作成 を参照してください。

手順

1. Stratis ファイルシステムをプールに作成するには、次のコマンドを実行します。

```
# stratis filesystem create --size number-and-unit my-pool my-fs
```

ここでは、以下のようになります。

number-and-unit
ファイルシステムのサイズを指定します。仕様形式は、入力の標準サイズ指定形式 (B、KiB、MiB、GiB、TiB、または PiB) に準拠する必要があります。

my-pool
Stratis プールの名前を指定します。

my-fs
ファイルシステムの任意名を指定します。
以下に例を示します。

例36.1 Stratis ファイルシステムの作成

```
# stratis filesystem create --size 10GiB pool1 filesystem1
```

検証手順

- プール内のファイルシステムを一覧表示して、Stratis ファイルシステムが作成されているかどうかを確認します。

```
# stratis fs list my-pool
```

関連情報

- Stratis ファイルシステムのマウント

36.1.15. Stratis ファイルシステムのマウント

既存の Stratis ファイルシステムをマウントして、コンテンツにアクセスします。

前提条件

- Stratis がインストールされている。詳細は、Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis ファイルシステムを作成している。詳細は、Stratis ファイルシステムの作成 を参照してください。

手順

- ファイルシステムをマウントするには、**/dev/stratis/** ディレクトリーに Stratis が維持するエントリーを使用します。

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

これでファイルシステムは **mount-point** ディレクトリーにマウントされ、使用できるようになりました。

関連情報

- Stratis ファイルシステムの作成

36.1.16. Stratis ファイルシステムの永続的なマウント

この手順では、Stratis ファイルシステムを永続的にマウントして、システムが起動した後に自動的に利用できるようにします。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis ファイルシステムを作成している。Stratis ファイルシステムの作成 を参照してください。

手順

1. ファイルシステムの UUID 属性を調べます。

```
$ lsblk --output=UUID /dev/stratis/my-pool/my-fs
```

以下に例を示します。

例36.2 Stratis ファイルシステムの UUID の表示

```
$ lsblk --output=UUID /dev/stratis/my-pool/fs1

UUID
a1f0b64a-4ebb-4d4e-9543-b1d79f600283
```

2. このマウントポイントのディレクトリーがない場合は、作成します。

```
# mkdir --parents mount-point
```

3. root で **/etc/fstab** ファイルを編集し、ファイルシステムに行を追加します (UUID で識別されます)。**xfs** をファイルシステムのタイプとして使用し、**x-systemd.requires=stratisd.service** オプションを追加します。
以下に例を示します。

例36.3 /etc/fstab の /fs1 マウントポイント

```
UUID=a1f0b64a-4ebb-4d4e-9543-b1d79f600283 /fs1 xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

5. ファイルシステムをマウントして、設定が機能することを確認します。

```
# mount mount-point
```

関連情報

- ファイルシステムの永続的なマウント

36.1.17. systemd サービスを使用した /etc/fstab での非 root Stratis ファイルシステムの設定

systemd サービスを使用して、/etc/fstab で非 root ファイルシステムの設定を管理できます。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis ファイルシステムを作成している。Stratis ファイルシステムの作成 を参照してください。

手順

- すべての非 root Stratis ファイルシステムでは、次を使用します。

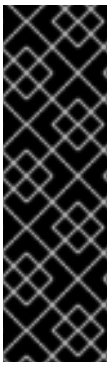
```
# /dev/stratis/[STRATIS_SYMLINK] [MOUNT_POINT] xfs defaults, x-
systemd.requires=stratis-fstab-setup@[POOL_UUID].service,x-systemd.after=stratis-stab-
setup@[POOL_UUID].service <dump_value> <fsck_value>
```

関連情報

- ファイルシステムの永続的なマウント

36.2. 追加のブロックデバイスでの STRATIS ボリュームの拡張

Stratis ファイルシステムのストレージ容量を増やすために、追加のブロックデバイスを Stratis プールに追加できます。



重要

Stratis はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

36.2.1. Stratis ボリュームの設定要素

Stratis ボリュームを設定するコンポーネントについて説明します。

外部的には、Stratis は、コマンドラインインターフェイスおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで設定されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は、各プールの **/dev/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。
ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratis に更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再設定しないでください。

Stratis は、**/dev/stratis/my-pool/my-fs** パスにファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映します。

36.2.2. Stratis プールへのブロックデバイスの追加

この手順では、Stratis ファイルシステムで使用できるように、1 つ以上のブロックデバイスを Stratis プールに追加します。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis プールに追加するブロックデバイスは使用されておらず、マウントされていない。
- Stratis プールに追加するブロックデバイスは使用されておらず、それぞれ 1 GiB 以上である。

手順

- 1 つ以上のブロックデバイスをプールに追加するには、以下を使用します。

```
# stratis pool add-data my-pool device-1 device-2 device-n
```

関連情報

- **stratis(8)** man ページ

36.2.3. 関連情報

- Stratis Storage の Web サイト

36.3. STRATIS ファイルシステムの監視

Stratis ユーザーは、システムにある Stratis ボリュームに関する情報を表示して、その状態と空き容量を監視できます。

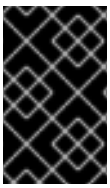
重要

Stratis はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

36.3.1. さまざまなユーティリティーが報告する Stratis のサイズ

本セクションでは、**df** などの標準的なユーティリティーと、**stratis** ユーティリティーにより報告される Stratis サイズの相違点を説明します。

df などの標準的な Linux ユーティリティーは、Stratis 上の 1TiB の XFS ファイルシステムレイヤーのサイズを報告します。これは 1 TiB です。Stratis の実際のストレージ使用量は、シンプロビジョニングにより少なくなっており、また XFS レイヤーが満杯に近くなると Stratis が自動的にファイルシステムを拡張するため、これは特に有用な情報ではありません。



重要

Stratis ファイルシステムに書き込まれているデータ量を定期的に監視します。これは **Total Physical Used** の値として報告されます。これが **Total Physical Size** の値を超えていないことを確認してください。

関連情報

- **stratis (8)** man ページ

36.3.2. Stratis ボリュームの情報表示

この手順では、Stratis ボリュームに関する合計サイズ、使用済みサイズ、空きサイズ、ファイルシステム、プールに属するブロックデバイスなどの統計情報をリスト表示します。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。

手順

- システムで Stratis に使用されているすべての **ブロックデバイス** に関する情報を表示する場合は、次のコマンドを実行します。

```
# stratis blockdev

Pool Name  Device Node    Physical Size   State  Tier
my-pool    /dev/sdb           9.10 TiB  In-use  Data
```

- システムにあるすべての Stratis **プール** に関する情報を表示するには、次のコマンドを実行します。

```
# stratis pool

Name    Total Physical Size  Total Physical Used
my-pool           9.10 TiB             598 MiB
```

- システムにあるすべての Stratis **ファイルシステム** に関する情報を表示するには、次のコマンドを実行します。

```
# stratis filesystem

Pool Name  Name  Used     Created            Device
my-pool    my-fs 546 MiB  Nov 08 2018 08:03  /dev/stratis/my-pool/my-fs
```

関連情報

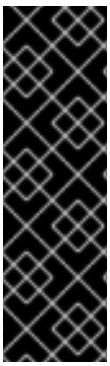
- **stratis (8)** man ページ

36.3.3. 関連情報

- Stratis Storage の Web サイト

36.4. STRATIS ファイルシステムでのスナップショットの使用

Stratis ファイルシステムのスナップショットを使用して、ファイルシステムの状態を任意の時点でキャプチャーし、後でそれを復元できます。



重要

Stratis はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

36.4.1. Stratis スナップショットの特徴

Stratis では、スナップショットは、別の Stratis ファイルシステムのコピーとして作成した通常の Stratis ファイルシステムです。スナップショットには、元のファイルシステムと同じファイルの内容が含まれていますが、スナップショットが変更するときにファイル内容が変更する可能性があります。スナップショットにどんな変更を加えても、元のファイルシステムには反映されません。

Stratis の現在のスナップショット実装は、次のような特徴があります。

- ファイルシステムのスナップショットは別のファイルシステムです。
- スナップショットと元のファイルシステムのリンクは、有効期間中は行われません。スナップショットされたファイルシステムは、元のファイルシステムよりも長く存続します。
- スナップショットを作成するためにファイルシステムをマウントする必要はありません。

- 各スナップショットは、XFS ログに必要となる実際のバッキングストレージの約半分のギガバイトを使用します。

36.4.2. Stratis スナップショットの作成

この手順では、既存の Stratis ファイルシステムのスナップショットとして Stratis ファイルシステムを作成します。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis ファイルシステムを作成している。Stratis ファイルシステムの作成 を参照してください。

手順

- Stratis スナップショットを作成するには、次のコマンドを実行します。

```
# stratis fs snapshot my-pool my-fs my-fs-snapshot
```

関連情報

- **stratis (8)** man ページ

36.4.3. Stratis スナップショットのコンテンツへのアクセス

この手順では、Stratis ファイルシステムのスナップショットをマウントして、読み書き操作にアクセスできるようにします。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis スナップショットを作成している。Stratis ファイルシステムの作成 を参照してください。

手順

- スナップショットにアクセスするには、**/dev/stratis/my-pool/** ディレクトリーから通常のファイルシステムとしてマウントします。

```
# mount /dev/stratis/my-pool/my-fs-snapshot mount-point
```

関連情報

- Stratis ファイルシステムのマウント
- **mount(8)** man ページ。

36.4.4. Stratis ファイルシステムを以前のスナップショットに戻す

この手順では、Stratis ファイルシステムの内容を、Stratis スナップショットでキャプチャーされた状態に戻します。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis スナップショットを作成している。Stratis スナップショットの作成 を参照してください。

手順

1. 必要に応じて、後でそれにアクセスできるように、ファイルシステムの現在の状態のバックアップを作成します。

```
# stratis filesystem snapshot my-pool my-fs my-fs-backup
```

2. 元のファイルシステムをアンマウントして削除します。

```
# umount /dev/stratis/my-pool/my-fs
# stratis filesystem destroy my-pool my-fs
```

3. 元のファイルシステムの名前でスナップショットのコピーを作成します。

```
# stratis filesystem snapshot my-pool my-fs-snapshot my-fs
```

4. 元のファイルシステムと同じ名前でアクセスできるようになったスナップショットをマウントします。

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

my-fs という名前のファイルシステムの内容は、スナップショット **my-fs-snapshot** と同じになりました。

関連情報

- **stratis (8)** man ページ

36.4.5. Stratis スナップショットの削除

この手順では、Stratis スナップショットをプールから削除します。スナップショットのデータは失われます。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。

- Stratis スナップショットを作成している。Stratis スナップショットの作成 を参照してください。

手順

1. スナップショットをアンマウントします。

```
# umount /dev/stratis/my-pool/my-fs-snapshot
```

2. スナップショットを破棄します。

```
# stratis filesystem destroy my-pool my-fs-snapshot
```

関連情報

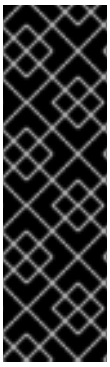
- **stratis (8)** man ページ

36.4.6. 関連情報

- Stratis Storage の Web サイト

36.5. STRATIS ファイルシステムの削除

既存の Stratis ファイルシステムまたは Stratis プールは、そこに含まれるデータを破棄することで削除できます。



重要

Stratis はテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

36.5.1. Stratis ボリュームの設定要素

Stratis ボリュームを設定するコンポーネントについて説明します。

外部的には、Stratis は、コマンドラインインターフェイスおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで設定されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は、各プールの **/dev/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratis に更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再設定しないでください。

Stratis は、**/dev/stratis/my-pool/my-fs** パスにファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映します。

36.5.2. Stratis ファイルシステムの削除

この手順では、既存の Stratis ファイルシステムを削除します。そこに保存されているデータは失われます。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis ファイルシステムを作成している。Stratis ファイルシステムの作成 を参照してください。

手順

1. ファイルシステムをアンマウントします。

```
# umount /dev/stratis/my-pool/my-fs
```

2. ファイルシステムを破棄します。

```
# stratis filesystem destroy my-pool my-fs
```

3. ファイルシステムがもう存在しないことを確認します。

```
# stratis filesystem list my-pool
```

関連情報

- **stratis (8)** man ページ

36.5.3. Stratis プールの削除

この手順では、既存の Stratis プールを削除します。そこに保存されているデータは失われます。

前提条件

- Stratis がインストールされている。Stratis のインストール を参照してください。
- **stratisd** サービスを実行している。
- Stratis プールを作成している。
  - 暗号化されていないプールを作成するには、暗号化されていない Stratis プールの作成 を参照してください。
  - 暗号化されたプールを作成するには、暗号化された Stratis プールの作成 を参照してください。

手順

1. プールにあるファイルシステムのリストを表示します。

```
# stratis filesystem list my-pool
```

2. プール上のすべてのファイルシステムをアンマウントします。

```
# umount /dev/stratis/my-pool/my-fs-1 \
        /dev/stratis/my-pool/my-fs-2 \
        /dev/stratis/my-pool/my-fs-n
```

3. ファイルシステムを破棄します。

```
# stratis filesystem destroy my-pool my-fs-1 my-fs-2
```

4. プールを破棄します。

```
# stratis pool destroy my-pool
```

5. プールがなくなったことを確認します。

```
# stratis pool list
```

関連情報

- **stratis (8)** man ページ

36.5.4. 関連情報

- Stratis Storage の Web サイト

36.6. スワップの使用

スワップ領域を使用して、非アクティブなプロセスとデータに一時的なストレージを提供し、物理メモリーがいっぱいになった場合に発生するメモリー不足エラーを防ぎます。スワップ領域は物理メモリーの拡張として機能し、物理メモリーが使い果たされた場合でもシステムがスムーズに動作し続けることを可能にします。スワップ領域を使用するとシステムのパフォーマンスが低下する可能性があるため、スワップ領域を利用する前に物理メモリーの使用を最適化するほうが望ましい場合があることに注意してください。

36.6.1. スワップ領域の概要

Linux の **スワップ領域** は、物理メモリー (RAM) が不足すると使用されます。システムに多くのメモリーリソースが必要で、RAM が不足すると、メモリーの非アクティブなページがスワップ領域に移動します。スワップ領域は、RAM が少ないマシンで役に立ちますが、RAM の代わりに使用しないようにしてください。

スワップ領域はハードドライブにあり、そのアクセス速度は物理メモリーに比べると遅くなります。スワップ領域の設定は、専用のスワップパーティション (推奨)、スワップファイル、またはスワップパーティションとスワップファイルの組み合せが考えられます。

過去数年、推奨されるスワップ領域のサイズは、システムの RAM サイズに比例して増加していました。しかし、最近のシステムには通常、数百ギガバイトの RAM が含まれます。結果として、推奨されるスワップ領域は、システムのメモリーではなく、システムメモリーのワークロードの機能とみなされます。

スワップ領域の追加

以下は、さまざまな方法でスワップ領域を追加する方法です。

- LVM2 論理ボリュームでのスワップ領域の拡張
- スワップの LVM2 論理ボリュームの作成
- スワップファイルの作成
  たとえば、システムの RAM 容量を 1 GB から 2 GB にアップグレードするときに、スワップスペースが 2 GB しかないとします。メモリーを大幅に消費する操作を実行している場合や、大量のメモリーを必要とするアプリケーションを実行する場合は、スワップ領域を 4 GB に増やすことが有益となる可能性があります。

スワップ領域の削除

スワップ領域を削除するには、以下の異なる方法を使用します。

- LVM2 論理ボリュームでのスワップ領域の縮小
- スワップの LVM2 論理ボリュームの削除
- スワップファイルの削除
  たとえば、システムの RAM 容量を 1 GB から 512 MB にダウングレードするとします。しかし、依然として 2 GB のスワップ容量が割り当てられています。ディスク領域が大きくなる (2 GB など) と無駄になる可能性があるため、スワップ領域を 1 GB に減らすことでメリットを得られることがあります。

36.6.2. システムの推奨スワップ領域

このセクションでは、システム内の RAM の量と、システムがハイバネートになるために十分なメモリーを必要とするかどうかに応じた、スワップパーティションの推奨サイズについて説明します。推奨されるスワップパーティションのサイズは、インストール時に自動的に確定されます。ハイバネートを可能にするには、カスタムのパーティション分割段階でスワップ領域を編集する必要があります。

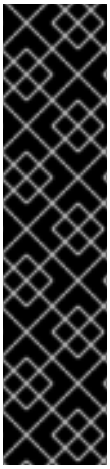
以下の推奨は、1GB 以下など、メモリーが少ないシステムで特に重要です。このようなシステムで十分なスワップ領域を割り当てられないと、不安定になる問題が生じたり、インストールしたシステムが起動できなくなる可能性があります。

表36.1 推奨されるスワップ領域

システム内の RAM の容量	推奨されるスワップ領域	ハイバネートを許可する場合に推奨されるスワップ領域
⩽ 2 GB	RAM 容量の 2 倍	RAM 容量の 3 倍
> 2 GB ~ 8 GB	RAM 容量と同じ	RAM 容量の 2 倍
> 8 GB ~ 64 GB	最低 4GB	RAM 容量の 1.5 倍
> 64 GB	最低 4GB	ハイバネートは推奨されない

値が、この表の範囲の境界線上にある場合 (システムの RAM が 2GB、8GB、または 64GB などの場合)、選択したスワップ領域とハイバネートへのサポートに関しては、適宜判断してください。システムリソースに余裕がある場合は、スワップ領域を増やすとパフォーマンスが向上することがあります。

高速のドライブ、コントローラー、およびインターフェイスを搭載したシステムでは、複数のストレージデバイスにスワップ領域を分散すると、スワップ領域のパフォーマンスも向上します。



重要

スワップ領域として割り当てたファイルシステムおよび LVM2 ボリュームは、変更時に **使用しない** でください。システムプロセスまたはカーネルがスワップ領域を使用していると、スワップの修正に失敗します。**free** コマンドおよび **cat /proc/swaps** コマンドを使用して、スワップの使用量と、使用中の場所を確認します。

スワップ領域のサイズを変更すると、システムのスワップ領域が一時的に削除されます。これは、実行中のアプリケーションが追加のスワップ領域に依存し、メモリーが不足する可能性がある場合に問題になる可能性があります。レスキューモードからスワップのサイズを変更することが推奨されます。**Performing an advanced RHEL 8 installation** の Debug boot options を参照してください。ファイルシステムをマウントするように指示されたら、**スキップ** を選択します。

36.6.3. LVM2 論理ボリュームでのスワップ領域の拡張

この手順では、既存の LVM2 論理ボリュームでスワップ領域を拡張する方法を説明します。ここでは、**2 GB** 拡張するボリュームを **/dev/VolGroup00/LogVol01** とします。

前提条件

- 十分なディスク領域がある。

手順

1. 関連付けられている論理ボリュームのスワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームのサイズを 2 GB 増やします。

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 拡張論理ボリュームを有効にします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

検証

- スワップ論理ボリュームが正常に拡張され、アクティブになったかをテストするには、次のコマンドを使用して、アクティブなスワップ領域を調べます。

```
$ cat /proc/swaps
$ free -h
```

36.6.4. スワップの LVM2 論理ボリュームの作成

この手順では、スワップ用に LVM2 論理ボリュームを作成する方法を説明します。ここでは、追加するスワップボリュームを /dev/VolGroup00/LogVol02 とします。

前提条件

- 十分なディスク領域がある。

手順

1. サイズが 2 GB の LVM2 論理ボリュームを作成します。

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol02
```

3. 次のエントリーを /etc/fstab ファイルに追加します。

```
/dev/VolGroup00/LogVol02 none swap defaults 0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

5. 論理ボリュームでスワップをアクティブにします。

```
# swapon -v /dev/VolGroup00/LogVol02
```

検証

- スワップ論理ボリュームが正常に作成され、アクティブになったかをテストするには、次のコマンドを使用して、アクティブなスワップ領域を調べます。

```
$ cat /proc/swaps
$ free -h
```

36.6.5. スワップファイルの作成

この手順では、スワップファイルの作成方法を説明します。

前提条件

- 十分なディスク領域がある。

手順

1. 新しいスワップファイルのサイズをメガバイト単位で指定してから、そのサイズに 1024 をかけてブロック数を指定します。たとえばスワップファイルのサイズが 64 MB の場合は、ブロック数が 65536 になります。
2. 空のファイルの作成:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

   65536 を、目的のブロックサイズと同じ値に置き換えます。

3. 次のコマンドでスワップファイルをセットアップします。

```
# mkswap /swapfile
```

4. スワップファイルのセキュリティーを変更して、全ユーザーで読み込みができないようにします。

```
# chmod 0600 /swapfile
```

5. システムの起動時にスワップファイルを有効にするには、次のエントリーを使用して **/etc/fstab** ファイルを編集します。

```
/swapfile none swap defaults 0 0
```

   次にシステムが起動すると新しいスワップファイルが有効になります。

6. システムが新しい **/etc/fstab** 設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

7. すぐにスワップファイルをアクティブにします。

```
# swapon /swapfile
```

検証

- 新しいスワップファイルが正常に作成され、有効になったかをテストするには、次のコマンドを使用して、アクティブなスワップ領域を調べます。

```
$ cat /proc/swaps
$ free -h
```

36.6.6. LVM2 論理ボリュームでのスワップ領域の縮小

この手順では、LVM2 論理ボリュームでスワップを減らす方法を説明します。ここでは、縮小するボリュームを **/dev/VolGroup00/LogVol01** とします。

手順

1. 関連付けられている論理ボリュームのスワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームのサイズを変更して 512 MB 削減します。

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 論理ボリュームでスワップをアクティブにします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

検証

- スワップ論理ボリュームが正常に削減されたかをテストするには、次のコマンドを使用して、アクティブなスワップ領域を調べます。

```
$ cat /proc/swaps
$ free -h
```

36.6.7. スワップの LVM2 論理ボリュームの削除

この手順では、スワップ用に LVM2 論理ボリュームを削除する方法を説明します。削除するスワップボリュームを **/dev/VolGroup00/LogVol02** とします。

手順

1. 関連付けられている論理ボリュームのスワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2. LVM2 論理ボリュームを削除します。

```
# lvremove /dev/VolGroup00/LogVol02
```

3. 次の関連エントリーを **/etc/fstab** ファイルから削除します。

```
/dev/VolGroup00/LogVol02 none swap defaults 0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

検証

- 論理ボリュームが正常に削除されたかをテストするには、次のコマンドを使用して、アクティブなスワップ領域を調べます。

```
$ cat /proc/swaps
$ free -h
```

36.6.8. スワップファイルの削除

この手順では、スワップファイルを削除する方法を説明します。

手順

1. シェルプロンプトで次のコマンドを実行してスワップファイルを無効にします (スワップファイルの場所が **/swapfile** であるとします)。

```
# swapoff -v /swapfile
```

2. **/etc/fstab** ファイルからエントリーを削除します。

3. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

4. 実際のファイルを削除します。

```
# rm /swapfile
```

第37章 ストレージの重複排除および圧縮

37.1. VDO のデプロイメント

システム管理者は、VDO を使用してストレージプールの重複を排除して、圧縮できます。

37.1.1. VDO の概要

VDO (Virtual Data Optimizer) は、重複排除、圧縮、およびシンプロビジョニングの形で、Linux でインラインのデータ削減を行います。VDO ボリュームを設定する場合は、VDO ボリュームを構築するブロックデバイスと、作成する論理ストレージのサイズを指定します。

- アクティブな仮想マシンまたはコンテナーをホストする場合、Red Hat は、物理と論理の割合を 1 対 10 にすることを推奨します。つまり、物理ストレージを 1 TB にした場合は、論理ストレージを 10 TB にします。
- Ceph が提供するタイプなどのオブジェクトストレージの場合、Red Hat は、物理と論理の割合を 1 対 3 にすることを推奨します。つまり、物理ストレージを 1 TB にした場合は、論理ストレージを 3 TB にします。

いずれの場合も、VDO が作成する論理デバイスにファイルシステムを置くだけで、直接使用することも、分散クラウドストレージアーキテクチャーの一部として使用することもできます。

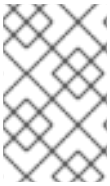
VDO はシンプロビジョニングされているため、ファイルシステムとアプリケーションは、使用中の論理領域だけを認識し、実際に利用可能な物理領域は認識しません。スクリプトを使用して、実際に利用可能な領域を監視し、使用量がしきい値を超えた場合 (たとえば、VDO ボリュームの使用量が 80% になった場合) にアラートを生成します。

37.1.2. VDO デプロイメントシナリオ

VDO は、様々な方法でデプロイして、以下に対して、重複排除したストレージを提供できます。

- ブロックおよびファイルアクセスの両方
- ローカルストレージおよびリモートストレージの両方

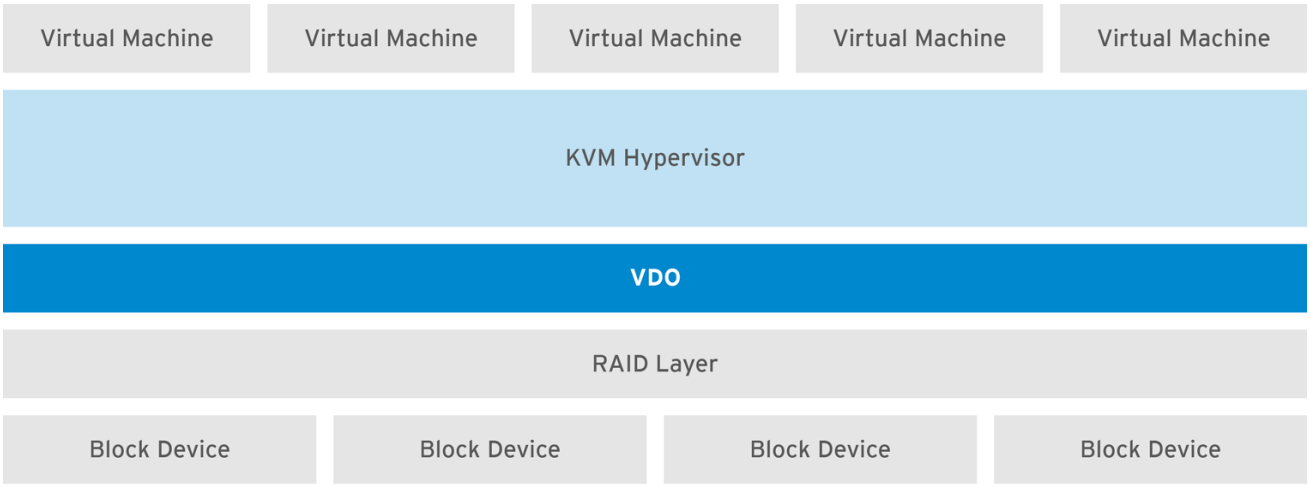
VDO は、標準の Linux ブロックデバイスとして重複排除したストレージを公開するため、そのストレージを標準ファイルシステム、iSCSI および FC のターゲットドライバー、または統合ストレージとして使用できます。



注記

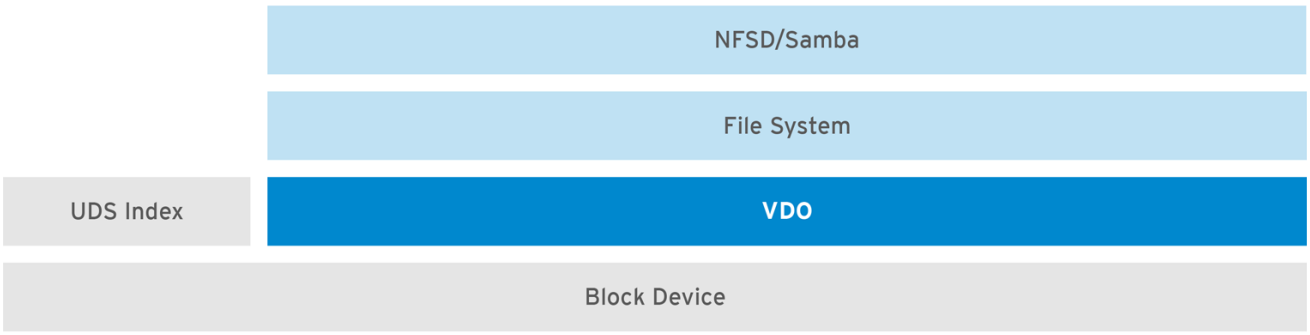
現在、Ceph RADOS ブロックデバイス (RBD) 上での VDO ボリュームのデプロイがサポートされています。ただし、VDO ボリューム上での Red Hat Ceph Storage クラスターコンポーネントのデプロイは現在サポートされていません。

KVM
DAS (Direct Attached Storage) を使用して設定した KVM サーバーに VDO をデプロイできます。



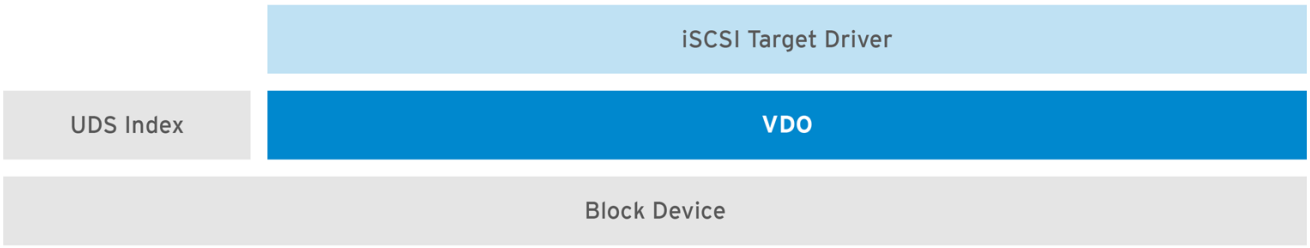
RHEL_462492_1117

ファイルシステム
VDO にファイルシステムを作成して、NFS サーバーまたは Samba で、NFS ユーザーまたは CIFS ユーザーに公開します。



RHEL_466924_0218

iSCSI への VDO の配置
VDO ストレージターゲット全体を、iSCSI ターゲットとしてリモート iSCSI イニシエーターにエクスポートできます。



RHEL_466924_0218

iSCSI で VDO ボリュームを作成する場合は、VDO ボリュームを iSCSI レイヤーの上または下に配置できます。考慮すべき点はたくさんありますが、ここでは、環境に最適な方法を選択するのに役立つガイドラインをいくつか示します。

VDO ボリュームを iSCSI レイヤーの下の iSCSI サーバー (ターゲット) に配置する場合:

- VDO ボリュームは、他の iSCSI LUN と同様に、イニシエーターに対して透過的です。シンプロビジョニングとスペースの節約をクライアントから隠すことで、LUN の外観の監視と保守が容易になります。

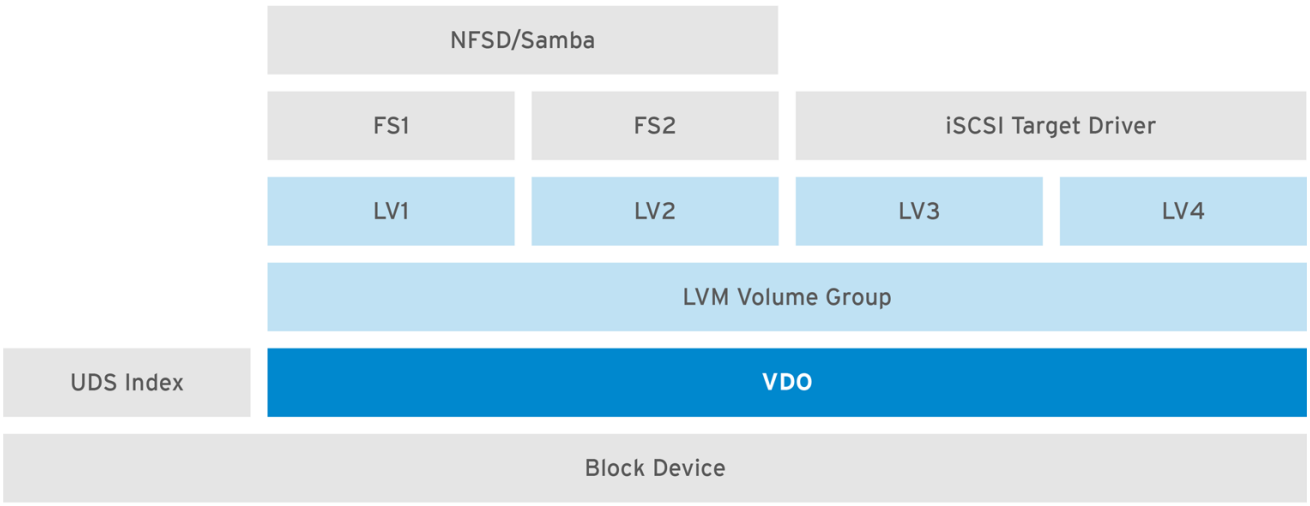
- VDO メタデータの読み取りまたは書き込みがないため、ネットワークトラフィックが減少し、重複排除アドバイスの読み取り検証がネットワーク全体で発生しません。
- iSCSI ターゲットで使用されているメモリーと CPU リソースにより、パフォーマンスが向上する可能性があります。たとえば、iSCSI ターゲットでボリュームの削減が行われているため、ハイパーバイザーの数を増やすことができます。
- クライアントがイニシエーターに暗号化を実装し、ターゲットの下に VDO ボリュームがある場合、スペースの節約は実現しません。

VDO ボリュームを iSCSI レイヤーの上の iSCSI クライアント (イニシエーター) に配置する場合:

- 高いスペース節約率を達成する場合、ASYNC モードでネットワーク全体のネットワークトラフィックが低下する可能性があります。
- スペースの節約を直接表示および制御し、使用状況を監視できます。
- たとえば、**dm-crypt** を使用してデータを暗号化する場合は、暗号の上に VDO を実装して、スペース効率を利用できます。

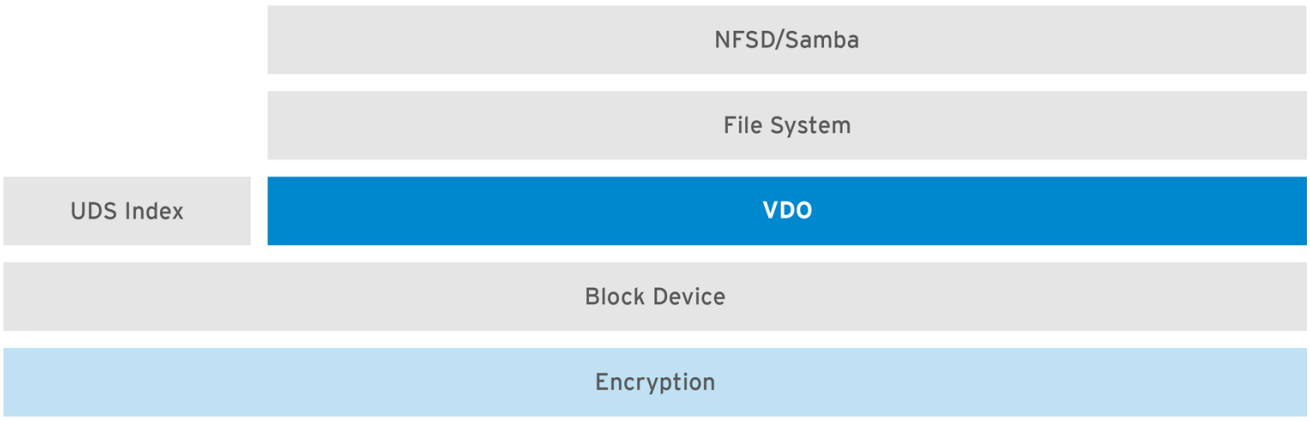
LVM
より機能豊富なシステムでは、LVM を使用して、重複排除した同じストレージプールですべて対応している複数の論理ユニット番号 (LUN) を提供できます。

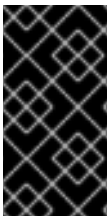
以下の図は、VDO ターゲットが物理ボリュームとして登録されるため、LVM で管理できます。複数の論理ボリューム (**LV1** から **LV4**) が、重複排除したストレージプールから作成されます。これにより、VDO は、基となる重複排除したストレージプールへのマルチプロトコル統合ブロックまたはファイルアクセスに対応できます。



重複排除した統合ストレージ設計により、複数のファイルシステムが、LVM ツールを介して同じ重複排除ドメインを共同で使用できます。また、ファイルシステムは、LVM スナップショット、コピーオンライト、縮小機能、拡大機能、および VDO にある全機能を利用できます。

暗号化
DM Crypt などのデバイスマッパー (DM) メカニズムは VDO と互換性があります。VDO ボリュームの暗号化により、データセキュリティーと、VDO にある全ファイルシステムが重複排除されるようになります。





重要

VDO で暗号化層を適用すると、データの重複排除が行われてもほとんど行われません。暗号化により、VDO が重複を排除する前に、重複ブロックを変更します。

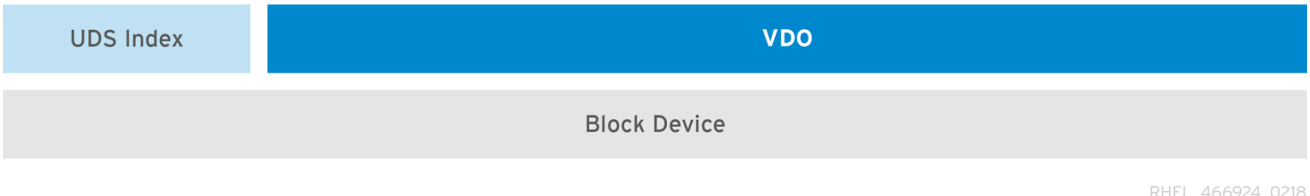
常に VDO の下に暗号化層を配置します。

iSCSI で VDO ボリュームを作成する場合は、VDO ボリュームを iSCSI レイヤーの上または下に配置できます。考慮すべき点はたくさんありますが、ここでは、環境に最適な方法を選択するのに役立つガイドラインをいくつか示します。

37.1.3. VDO ボリュームのコンポーネント

VDO は、バッキングストアとしてブロックデバイスを使用します。これは、複数のディスク、パーティション、またはフラットファイルで設定される物理ストレージの集約を含めることができます。ストレージ管理ツールが VDO ボリュームを作成すると、VDO は、UDS インデックスおよび VDO ボリュームのボリューム領域を予約します。UDS インデックスと VDO ボリュームは対話して、重複排除したブロックストレージを提供します。

図37.1 VDO ディスク組織



VDO ソリューションは、以下のコンポーネントで設定されます。

kvdo

Linux Device Mapper 層に読み込まれるカーネルモジュールは、重複排除され、圧縮され、シンプロビジョニングされたブロックストレージボリュームを提供します。
kvdo モジュールはブロックデバイスを公開します。ブロックストレージ用にこのブロックデバイスに直接アクセスするか、XFS や ext4 などの Linux ファイルシステムを介して提示することができます。

kvdo が VDO ボリュームからデータ論理ブロックを読み取る要求を受信すると、要求された論理ブロックを基礎となる物理ブロックにマッピングし、要求したデータを読み取り、返します。

kvdo が VDO ボリュームにデータブロックを書き込む要求を受信すると、まず要求が DISCARD または TRIM のものであるか、またはデータが一貫してゼロかどうかを確認します。これらの条件のいずれかが true の場合、**kvdo** はブロックマップを更新し、リクエストを承認します。そうでない場合は、VDO はデータを処理して最適化します。

uds

ボリューム上の Universal Deduplication Service (UDS) インデックスと通信し、データの重複を分析するカーネルモジュール。新しい各データについて、その部分が保存してあるデータ内容と同一であるかどうかを UDS が素早く判断します。インデックスが一致すると、ストレージシステムは、同じ情報を複数格納しないように、既存の項目を内部的に参照できます。
UDS インデックスは、**uds** カーネルモジュールとしてカーネル内で実行します。

コマンドラインツール

最適化されたストレージの設定および管理

37.1.4. VDO ボリュームの物理サイズおよび論理サイズ

VDO は、物理サイズ、利用可能な物理サイズ、および論理サイズを次の方法で利用します。

物理サイズ

これは、基礎となるブロックデバイスと同じサイズです。VDO は、以下の目的でこのストレージを使用します。

- 重複排除および圧縮される可能性があるユーザーデータ
- UDS インデックスなどの VDO メタデータ

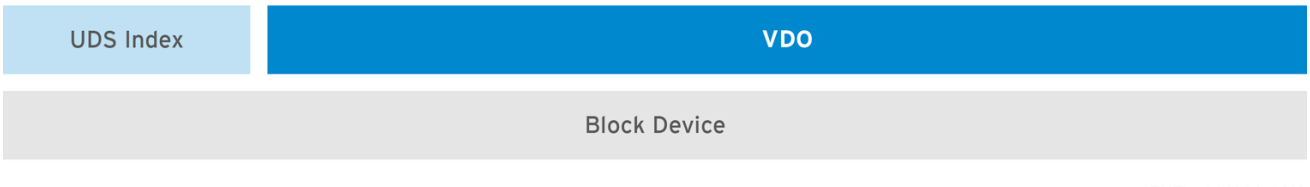
利用可能な物理サイズ

これは、VDO がユーザーデータに使用できる物理サイズの一部です。
これは、メタデータのサイズを引いた物理サイズと同等で、指定のスラブサイズでボリュームをスラブに分割した後の残りを引いたものと同じです。

論理サイズ

これは、VDO ボリュームがアプリケーションに提示するプロビジョニングされたサイズです。通常、これは利用可能な物理サイズよりも大きくなります。**--vdoLogicalSize** オプションを指定しないと、論理ボリュームのプロビジョニングが **1:1** の比率にプロビジョニングされます。たとえば、VDO ボリュームが 20 GB ブロックデバイスの上に置かれている場合は、2.5 GB が UDS インデックス用に予約されます (デフォルトのインデックスサイズが使用される場合)。残りの 17.5 GB は、VDO メタデータおよびユーザーデータに提供されます。そのため、消費する利用可能なストレージは 17.5 GB を超えません。実際の VDO ボリュームを設定するメタデータにより、これよりも少なくなる可能性があります。
VDO は現在、絶対最大論理サイズ 4PB の物理ボリュームの最大 254 倍の論理サイズに対応します。

図37.2 VDO ディスク組織



この図では、VDO で重複排除したストレージターゲットがブロックデバイス上に完全に配置されています。つまり、VDO ボリュームの物理サイズは、基礎となるブロックデバイスと同じサイズになります。

関連情報

- さまざまなサイズのブロックデバイスに必要なストレージ VDO メタデータのサイズは、「物理サイズ別の VDO 要件の例」を参照してください。

37.1.5. VDO のスラブサイズ

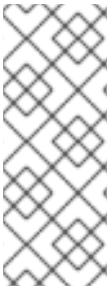
VDO ボリュームの物理ストレージは、複数のスラブに分割されます。各スラブは、物理領域における連続した領域です。特定のボリュームのスラブはすべて同じサイズで、128 MB の 2 のべき乗倍のサイズ (最大 32 GB) になります。

小規模なテストシステムで VDO を評価しやすくするため、デフォルトのスラブサイズは 2 GB です。1 つの VDO ボリュームには、最大 8192 個のスラブを含めることができます。したがって、デフォルト設定の 2 GB のスラブを使用する場合、許可される物理ストレージは最大 16 TB です。32 GB のスラブを使用する場合、許可される物理ストレージは最大 256 TB です。VDO は常に少なくとも 1 つのスラブ全体をメタデータ用に予約します。そのため、この予約されたスラブをユーザーデータの保存に使用することはできません。

スラブサイズは、VDO ボリュームのパフォーマンスには影響しません。

表37.1 物理ボリュームサイズ別の推奨 VDO スラブサイズ

物理ボリュームのサイズ	推奨されるスラブサイズ
10 - 99 GB	1 GB
100 GB - 1 TB	2 GB
2 - 256 TB	32 GB



注記

VDO ボリュームの最小ディスク使用量は、デフォルト設定のスラブサイズ 2 GB、dense インデックス 0.25 を使用した場合、約 4.7 GB が必要です。これにより、0% の重複排除または圧縮で書き込むための 2 GB 弱の物理データが提供されます。

ここでの最小のディスク使用量は、デフォルトのスラブサイズと dense インデックスの合計です。

lvcreate コマンドに **--config 'allocation/vdo_slab_size_mb=size-in-megabytes'** オプションを指定すると、スラブサイズを制御できます。

37.1.6. VDO 要件

VDO は、配置とシステムリソースに特定の要件があります。

37.1.6.1. VDO メモリー要件

各 VDO ボリュームには、2 つの異なるメモリー要件があります。

VDO モジュール

VDO には、固定メモリー 38 MB と変動用に容量を確保する必要があります。

- 設定済みのブロックマップキャッシュサイズ 1 MB ごとに 1.15 MB のメモリー。ブロックマップキャッシュには、少なくとも 150 MB のメモリーが必要です。
- 1 TB の論理領域ごとに 1.6 MB のメモリー。
- ボリュームが管理する物理ストレージの 1 TB ごとに 268 MB のメモリー。

UDS インデックス

Universal Deduplication Service (UDS) には、最低 250 MB のメモリーが必要です。このメモリー量は、重複排除が使用するデフォルトの容量です。この値は、インデックスに必要なストレージ容量にも影響するため、VDO ボリュームをフォーマットするときに設定できます。
UDS インデックスに必要なメモリーは、インデックスタイプと、重複排除ウィンドウに必要なサイズで決定されます。

インデックスタイプ	重複排除ウィンドウ	備考
Dense	RAM 1 GB あたり 1 TB	通常、最大 4 TB の物理ストレージには、1 GB の dense インデックスで十分です。
Sparse	RAM 1 GB あたり 10 TB	通常、最大 40 TB の物理ストレージには、1 GB の sparse インデックスで十分です。

注記

VDO ボリュームの最小ディスク使用量は、デフォルト設定のスラブサイズ 2 GB、dense インデックス 0.25 を使用した場合、約 4.7 GB が必要です。これにより、0% の重複排除または圧縮で書き込むための 2 GB 弱の物理データが提供されます。

ここでの最小のディスク使用量は、デフォルトのスラブサイズと dense インデックスの合計です。

VDO で推奨されるモードは、UDS の sparse インデックス機能です。この機能は、データの一時的な局所性に依存し、メモリー内で最も関連性の高いインデックスエントリーのみを保持しようとします。sparse インデックスでは、UDS は、同じ量のメモリーを使用しながら、dense を使用したときの 10 倍以上長い重複排除ウィンドウを維持できます。

sparse インデックスを使用すると対象範囲が広くなりますが、dense インデックスの方が提供する重複排除アドバイスが多くなります。ほとんどのワークロードでは、メモリー量が同じであれば、dense インデックスと sparse インデックスの重複排除率の差はごくわずかです。

関連情報

- 物理サイズ別の VDO 要件の例

37.1.6.2. VDO ストレージの領域要件

VDO ボリュームを設定して、最大 256 TB の物理ストレージを使用するように設定できます。データを格納するのに使用できるのは、物理ストレージの一部のみです。このセクションでは、VDO に管理されるボリュームで使用可能なサイズを特定するための計算方法を説明します。

VDO では、2 種類の VDO メタデータと UDS インデックスにストレージが必要です。

- 最初のタイプの VDO メタデータは、4 GB の **物理ストレージ** ごとに約 1 MB を使用し、スラブごとにさらに 1 MB を使用します。
- 2 番目のタイプの VDO メタデータは、**論理ストレージ** 1 GB ごとに約 1.25 MB を消費し、最も近いスラブに切り上げられます。
- UDS インデックスに必要なストレージの容量は、インデックスの種類と、インデックスに割り当てられている RAM の容量によって異なります。RAM 1 GB ごとに、dense の UDS インデックスはストレージを 17 GB 使用し、sparse の UDS インデックスはストレージを 170 GB 使用します。

関連情報

- 物理サイズ別の VDO 要件の例
- VDO のスラブサイズ

37.1.6.3. ストレージスタックの VDO の配置

配置要件に合わせて、Virtual Data Optimizer (VDO) の上または下にストレージ層を配置します。

VDO ボリュームは、シンプロビジョニングしたブロックデバイスです。後で拡張できるストレージ層の上にボリュームを配置することで、物理スペースの不足を防ぐことができます。このような拡張可能なストレージの例として、論理ボリュームマネージャー (LVM) ボリューム、または Multiple Device Redundant Array of Inexpensive/Independent Disks (MD RAID) アレイがあります。

VDO の上にシックプロビジョニングレイヤーを配置できます。シックプロビジョニングレイヤーについては、次の 2 つの側面を考慮する必要があります。

- シックデバイスの未使用の論理領域への新しいデータの書き込み。VDO またはその他のシンプロビジョニングストレージを使用している場合、この種の書き込み中にデバイスの領域が不足していると報告されることがあります。
- 新しいデータによるシックデバイスの使用済み論理領域の上書き。VDO を使用している場合、データを上書きすると、デバイスの領域が不足しているという報告が表示されることがあります。

これらの制限は、VDO 層より上のすべてのレイヤーに影響します。VDO デバイスが監視されていない場合には、VDO 層の上にあるシックプロビジョニングのボリュームで、物理領域が予期せず不足する可能性があります。

次のサポート対象およびサポート対象外の VDO ボリューム設定の例を参照してください。

図37.3 サポートされている VDO ボリューム設定

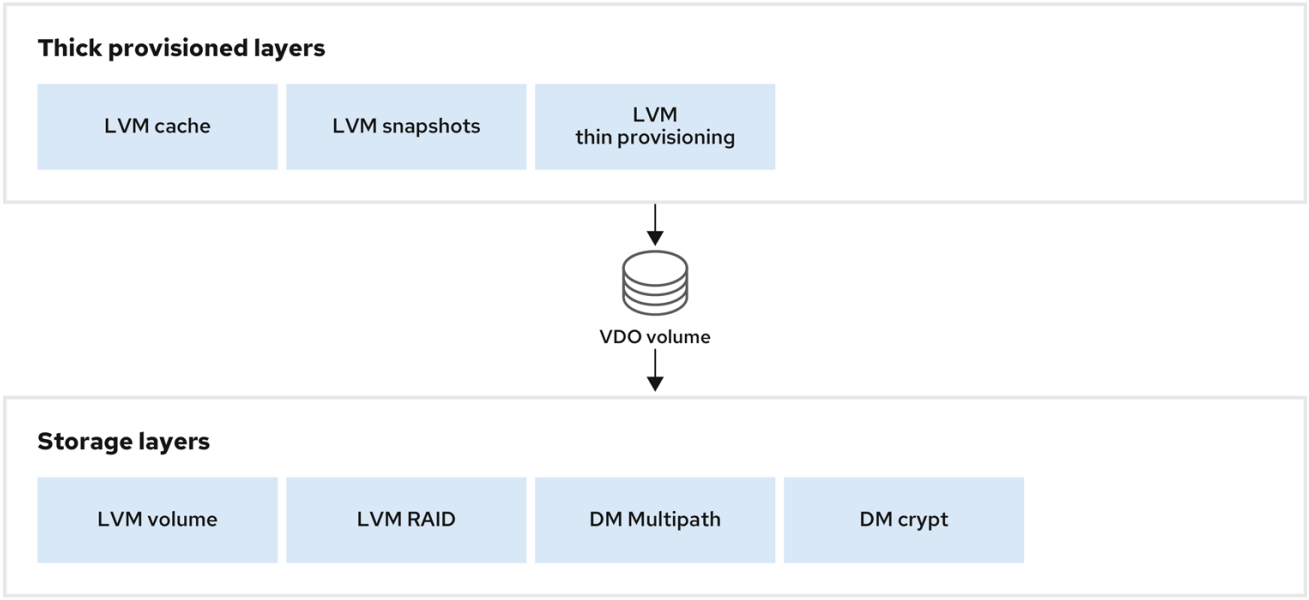
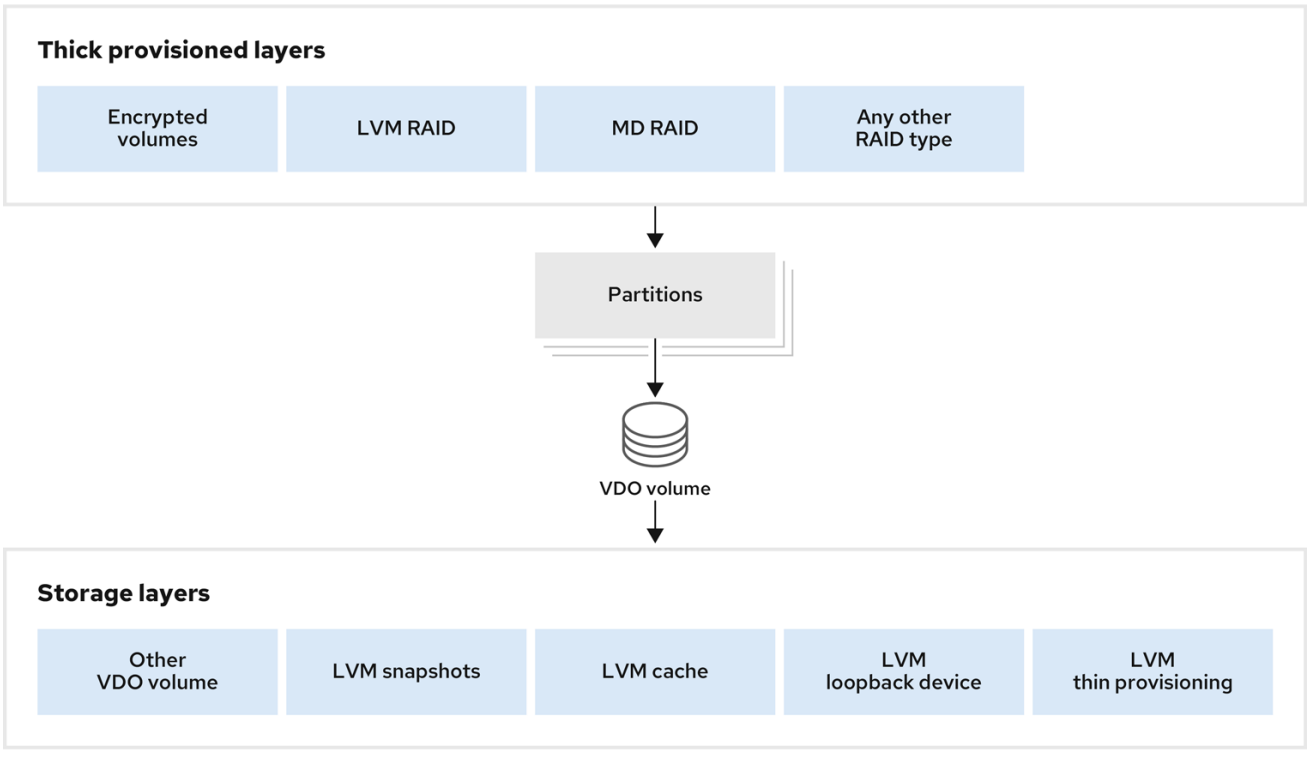


図37.4 サポートされていない VDO ボリューム設定



関連情報

- LVM レイヤーによる VDO のスタックの詳細は、LVM ボリュームのスタック を参照してください。

37.1.6.4. 物理サイズ別の VDO 要件の例

以下の表は、基盤となるボリュームの物理サイズに基づいた、VDO のシステム要件の概算を示しています。それぞれの表には、プライマリーストレージ、バックアップストレージなどの、目的のデプロイメントに適した要件が記載されています。

正確な数値は、VDO ボリュームの設定により異なります。

プライマリーストレージのデプロイメント

プライマリーストレージの場合、UDS インデックスのサイズは、物理サイズの 0.01% から 25% になります。

表37.2 プライマリーストレージのストレージ要件およびメモリー要件

物理サイズ	RAM の使用量:UDS	RAM の使用量:VDO	ディスク使用量	インデックスタイプ
10 GB - 1 TB	250 MB	472 MB	2.5 GB	Dense
2 - 10 TB	1 GB	3 GB	10 GB	Dense
	250 MB		22 GB	Sparse
11 - 50 TB	2 GB	14 GB	170 GB	Sparse
51 - 100 TB	3 GB	27 GB	255 GB	Sparse
101 - 256 TB	12 GB	69 GB	1020 GB	Sparse

バックアップストレージのデプロイメント

バックアップストレージの場合、UDS インデックスは、バックアップセットのサイズよりは大きくなりますが、物理サイズと同じか、より小さくなります。バックアップセットや物理サイズが今後大きくなる可能性がある場合は、これをインデックスサイズに組み込んでください。

表37.3 バックアップストレージのストレージ要件およびメモリー要件

物理サイズ	RAM の使用量:UDS	RAM の使用量:VDO	ディスク使用量	インデックスタイプ
10 GB - 1 TB	250 MB	472 MB	2.5 GB	Dense
2 - 10 TB	2 GB	3 GB	170 GB	Sparse
11 - 50 TB	10 GB	14 GB	850 GB	Sparse
51 - 100 TB	20 GB	27 GB	1700 GB	Sparse

物理サイズ	RAM の使用量:UDS	RAM の使用量:VDO	ディスク使用量	インデックスタイプ
101 - 256 TB	26 GB	69 GB	3400 GB	Sparse

37.1.7. VDO のインストール

この手順では、VDO ボリュームの作成、マウント、および管理に必要なソフトウェアをインストールします。

手順

- VDO ソフトウェアをインストールします。

```
# yum install lvm2 kmod-kvdo vdo
```

37.1.8. VDO ボリュームの作成

この手順では、ブロックデバイスに VDO ボリュームを作成します。

前提条件

- VDO ソフトウェアをインストールしている。「VDO のインストール」 を参照してください。
- 拡張可能なストレージをバッキングブロックデバイスとして使用している。詳細は、「ストレージスタックの VDO の 配置」 を参照してください。

手順

以下のすべての手順で、**vdo-name** を、VDO ボリュームに使用する識別子 (**vdo1** など) に置き換えます。システムの VDO の各インスタンスに、それぞれ別の名前とデバイスを使用する必要があります。

1. VDO ボリュームを作成するブロックデバイスの永続的な名前を確認してください。永続的な名前の詳細は、*26章 永続的な命名属性の概要* を参照してください。
   永続的なデバイス名を使用しないと、今後デバイスの名前が変わった場合に、VDO が正しく起動しなくなることがあります。

2. VDO ボリュームを作成します。

```
# vdo create \
    --name=vdo-name \
    --device=block-device \
    --vdoLogicalSize=logical-size
```

   - **block-device** を、VDO ボリュームを作成するブロックデバイスの永続名に置き換えます。たとえば、**/dev/disk/by-id/scsi-3600508b1001c264ad2af21e903ad031f** です。
   - **logical-size** を、VDO ボリュームが含まれる論理ストレージのサイズに置き換えます。
     - アクティブな仮想マシンまたはコンテナーストレージの場合は、使用する論理サイズが、ブロックデバイスの物理サイズの 10 倍になるようにします。たとえば、ブロックデバイスのサイズが 1 TB の場合は、**10T** を使用します。

- オブジェクトストレージの場合は、使用する論理サイズを、ブロックデバイスの物理サイズの 3 倍になるようにします。たとえば、ブロックデバイスのサイズが 1 TB の場合は、**3T** を使用します。

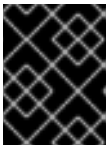
- 物理ブロックデバイスが 16TiB を超える場合は、**--vdoSlabSize=32G** オプションを指定して、ボリューム上のスラブサイズを 32GiB に増やします。
16TiB を超えるブロックデバイスでデフォルトのスラブサイズ 2GiB を使用すると、**vdo create** コマンドが失敗し、以下のエラーが出力されます。

```
vdo: ERROR - vdoformat: formatVDO failed on '/dev/device': VDO Status: Exceeds maximum number of slabs supported
```

例37.1 コンテナーストレージ用に VDO の作成

たとえば、1 TB ブロックデバイスでコンテナーストレージ用に VDO ボリュームを作成するには、次のコマンドを実行します。

```
# vdo create \
    --name=vdo1 \
    --device=/dev/disk/by-id/scsi-3600508b1001c264ad2af21e903ad031f \
    --vdoLogicalSize=10T
```



重要

VDO ボリュームの作成中に問題が発生した場合は、ボリュームを削除してください。詳細は、作成に失敗した VDO ボリュームの削除 を参照してください。

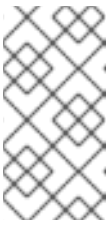
3. VDO ボリュームにファイルシステムを作成します。

- XFS ファイルシステムの場合:

```
# mkfs.xfs -K /dev/mapper/vdo-name
```

- ext4 ファイルシステムの場合:

```
# mkfs.ext4 -E nodiscard /dev/mapper/vdo-name
```



注記

新しく作成された VDO ボリュームでの **-K** および **-E nodiscard** オプションの目的は、割り当てられていないブロックには影響しないため、リクエストの送信に時間を費やさないようにすることです。新しい VDO ボリュームは、100% 割り当てられていない状態で開始されます。

4. 次のコマンドを使用して、システムが新しいデバイスノードを登録するまで待機します。

```
# udevadm settle
```

次のステップ

1. ファイルシステムをマウントします。詳しくは 「VDO ボリュームのマウント」 を参照してください。
2. VDO デバイスのファイルシステムで **discard** 機能を有効にします。詳しくは 「定期的なブロック破棄の有効化」 を参照してください。

関連情報

- man ページの **vdo(8)**

37.1.9. VDO ボリュームのマウント

この手順では、手動で、または永続的に、VDO ボリュームにファイルシステムをマウントします。

前提条件

- システムで VDO ボリュームが作成されている。手順は、「VDO ボリュームの作成」 を参照してください。

手順

- VDO ボリュームに手動でファイルシステムをマウントするには、以下のコマンドを使用します。

```
# mount /dev/mapper/vdo-name mount-point
```

- システムの起動時にファイルシステムを自動的にマウントするように設定するには、**/etc/fstab** ファイルに以下の行を追加します。
  - XFS ファイルシステムの場合:

```
/dev/mapper/vdo-name mount-point xfs defaults 0 0
```

  - ext4 ファイルシステムの場合:

```
/dev/mapper/vdo-name mount-point ext4 defaults 0 0
```

  VDO ボリュームが、iSCSI などのネットワークを必要とするブロックデバイスに配置されている場合は、**_netdev** マウントオプションを追加します。

関連情報

- **vdo(8)** man ページ
- iSCSI や、ネットワークを必要とするその他のブロックデバイスの **_netdev** マウントオプションに関する情報は、man ページの **systemd.mount(5)** を参照してください。

37.1.10. 定期的なブロック破棄の有効化

この手順では、対応するすべてのファイルシステムで、未使用のブロックを定期的に破棄する **systemd** タイマーを有効にします。

手順

- **systemd** タイマーを有効にして起動します。

```
# systemctl enable --now fstrim.timer
```

37.1.11. VDO の監視

この手順では、VDO ボリュームから、使用方法と効率に関する情報を取得する方法を説明します。

前提条件

- VDO ソフトウェアをインストールしている。VDO のインストール を参照してください。

手順

- **vdostats** ユーティリティーは、VDO ボリュームに関する情報を取得します。

```
# vdostats --human-readable

Device                 1K-blocks     Used     Available    Use%    Space saving%
/dev/mapper/node1osd1    926.5G       21.0G    905.5G       2%      73%
/dev/mapper/node1osd2    926.5G       28.2G    898.3G       3%      64%
```

関連情報

- man ページの **vdostats(8)**

37.2. VDO のメンテナンス

VDO ボリュームのデプロイ後、特定のタスクを実行して、ボリュームを維持または最適化することができます。VDO ボリュームの適切な機能には、以下の一部のタスクが必要です。

前提条件

- VDO がインストールされ、デプロイされます。「VDO のデプロイメント」 を参照してください。

37.2.1. VDO ボリューム上の空き領域の管理

VDO は、シンプロビジョニングされたブロックストレージターゲットです。そのため、VDO ボリュームで領域の使用状況をアクティブに監視し、管理する必要があります。

37.2.1.1. VDO ボリュームの物理サイズおよび論理サイズ

VDO は、物理サイズ、利用可能な物理サイズ、および論理サイズを次の方法で利用します。

物理サイズ

これは、基礎となるブロックデバイスと同じサイズです。VDO は、以下の目的でこのストレージを使用します。

- 重複排除および圧縮される可能性があるユーザーデータ
- UDS インデックスなどの VDO メタデータ

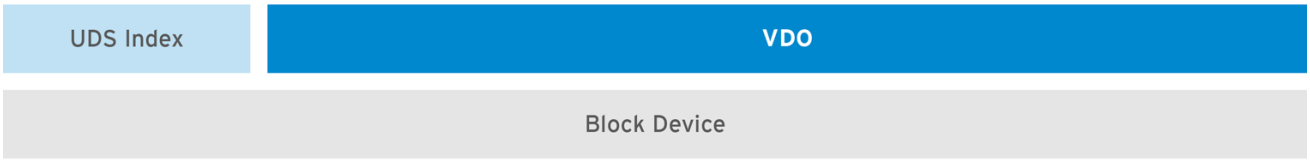
利用可能な物理サイズ

これは、VDO がユーザーデータに使用できる物理サイズの一部です。
これは、メタデータのサイズを引いた物理サイズと同等で、指定のスラブサイズでボリュームをスラブに分割した後の残りを引いたものと同じです。

論理サイズ

これは、VDO ボリュームがアプリケーションに提示するプロビジョニングされたサイズです。通常、これは利用可能な物理サイズよりも大きくなります。**--vdoLogicalSize** オプションを指定しないと、論理ボリュームのプロビジョニングが **1:1** の比率にプロビジョニングされます。たとえば、VDO ボリュームが 20 GB ブロックデバイスの上に置かれている場合は、2.5 GB が UDS インデックス用に予約されます (デフォルトのインデックスサイズが使用される場合)。残りの 17.5 GB は、VDO メタデータおよびユーザーデータに提供されます。そのため、消費する利用可能なストレージは 17.5 GB を超えません。実際の VDO ボリュームを設定するメタデータにより、これよりも少なくなる可能性があります。
VDO は現在、絶対最大論理サイズ 4PB の物理ボリュームの最大 254 倍の論理サイズに対応します。

図37.5 VDO ディスク組織



RHEL_466924_0218

この図では、VDO で重複排除したストレージターゲットがブロックデバイス上に完全に配置されています。つまり、VDO ボリュームの物理サイズは、基礎となるブロックデバイスと同じサイズになります。

関連情報

- さまざまなサイズのブロックデバイスに必要なストレージ VDO メタデータのサイズは、「物理サイズ別の VDO 要件の例」 を参照してください。

37.2.1.2. VDO でのシンプロビジョニング

VDO は、シンプロビジョニングされたブロックストレージターゲットです。VDO ボリュームが使用する物理領域のサイズは、ストレージのユーザーに示されるボリュームのサイズとは異なる可能性があります。この相違を活用して、ストレージのコストを削減できます。

容量不足の条件

書き込んだデータが、予想される最適化率に到達できない場合は、ストレージ領域が予想外に不足しないように注意してください。

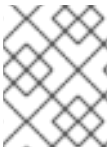
論理ブロック (仮想ストレージ) の数が物理ブロック (実際のストレージ) の数を超えると、ファイルシステムおよびアプリケーションで領域が予想外に不足する可能性があります。このため、VDO を使用するストレージシステムは、VDO ボリュームの空きプールのサイズを監視する方法で提供する必要があります。

vdostats ユーティリティーを使用すると、この空きプールのサイズを確認できます。このユーティリティーのデフォルト出力には、Linux の **df** ユーティリティーと同様の形式で稼働しているすべての VDO ボリュームの情報が記載されます。以下に例を示します。

```
Device                1K-blocks   Used       Available   Use%
/dev/mapper/vdo-name  211812352   105906176   105906176   50%
```

VDO ボリュームの物理ストレージ領域が不足しそうになると、VDO は、システムログに、以下のような警告を出力します。

```
Oct  2 17:13:39 system lvm[13863]: Monitoring VDO pool vdo-name.
Oct  2 17:27:39 system lvm[13863]: WARNING: VDO pool vdo-name is now 80.69% full.
Oct  2 17:28:19 system lvm[13863]: WARNING: VDO pool vdo-name is now 85.25% full.
Oct  2 17:29:39 system lvm[13863]: WARNING: VDO pool vdo-name is now 90.64% full.
Oct  2 17:30:29 system lvm[13863]: WARNING: VDO pool vdo-name is now 96.07% full.
```



注記

この警告メッセージは、**lvm2-monitor** サービスが実行している場合に限り表示されます。これは、デフォルトで有効になっています。

容量不足の状況を防ぐ方法

空きプールのサイズが特定のレベルを下回る場合は、以下を行うことができます。

- データの削除。これにより、削除したデータが重複していないと、領域が回収されます。データを削除しても、破棄が行われないと領域を解放しません。
- 物理ストレージの追加

重要

VDO ボリュームで物理領域を監視し、領域不足を回避します。物理ブロックが不足すると、VDO ボリュームに最近書き込まれたデータや、未承認のデータが失われることがあります。

シンプロビジョニング、TRIM コマンド、および DISCARD コマンド

シンプロビジョニングによるストレージの削減から利益を得るには、データを削除するタイミングを物理ストレージ層が把握する必要があります。シンプロビジョニングされたストレージで動作するファイルシステムは、**TRIM** コマンドまたは **DISCARD** コマンドを送信して、論理ブロックが不要になったときにストレージシステムに通知します。

TRIM コマンドまたは **DISCARD** コマンドを送信する方法はいくつかあります。

- **discard** マウントオプションを使用すると、ブロックが削除されるたびに、ファイルシステムがそのコマンドを送信できます。
- **fstrim** などのユーティリティーを使用すると、制御された方法でコマンドを送信できます。このようなユーティリティーは、どの論理ブロックが使用されていないかを検出し、**TRIM** コマンドまたは **DISCARD** コマンドの形式でストレージシステムに情報を送信するようにファイルシステムに指示します。

未使用のブロックで **TRIM** または **DISCARD** を使用する必要は、VDO に特有のものではありません。シンプロビジョニングしたストレージシステムでも、同じ課題があります。

37.2.1.3. VDO の監視

この手順では、VDO ボリュームから、使用方法と効率に関する情報を取得する方法を説明します。

前提条件

- VDO ソフトウェアをインストールしている。VDO のインストール を参照してください。

手順

- **vdostats** ユーティリティーは、VDO ボリュームに関する情報を取得します。

```
# vdostats --human-readable

Device                 1K-blocks    Used    Available    Use%    Space saving%
/dev/mapper/node1osd1    926.5G      21.0G    905.5G      2%      73%
/dev/mapper/node1osd2    926.5G      28.2G    898.3G      3%      64%
```

関連情報

- man ページの **vdostats(8)**

37.2.1.4. ファイルシステムで VDO の領域の回収

この手順では、ファイルシステムをホストする VDO ボリュームのストレージの領域を回収する方法を説明します。

ファイルシステムが、**DISCARD** コマンド、**TRIM** コマンド、または **UNMAP** コマンドを使用して、ブロックが空いていることを伝えない限り、VDO は領域を回収できません。

手順

- VDO ボリュームのファイルシステムが破棄操作に対応している場合は、その操作を有効化してください。未使用ブロックの破棄 を参照してください。

- ファイルシステムが、**DISCARD**、**TRIM**、または **UNMAP** を使用していない場合は、空き領域を手動で回収できます。バイナリーゼロで設定されるファイルを保存し、空き領域を埋め、そのファイルを削除します。

37.2.1.5. ファイルシステムを使用しない VDO の領域の回収

この手順では、ファイルシステムを使用せずにブロックストレージターゲットとして使用される VDO ボリュームでストレージ領域を回収します。

手順

- **blkdiscard** ユーティリティーを使用します。
  たとえば、VDO ボリュームは、LVM をその上にデプロイすることで、1つの VDO を複数のサブボリュームに分類できます。論理ボリュームのプロビジョニングを解除する前に、**blkdiscard** ユーティリティーを使用して、その論理ボリュームにより使用されていた領域を解放します。

  LVM は、**REQ_DISCARD** コマンドに対応し、領域を解放するために適切な論理ブロックアドレスで VDO にリクエストを転送します。その他のボリュームマネージャーを使用する場合

は、**REQ_DISCARD** に対応する必要があります。同じように、SCSI デバイスの場合は **UNMAP**、または ATA デバイスの場合は **TRIM** に対応している必要があります。

関連情報

- man ページの **blkdiscard(8)**

37.2.1.6. ファイバーチャンネルまたはイーサネットネットワーク上で VDO の領域の回収

この手順では、LIO、SCST などの SCSI ターゲットフレームワークを使用して、ファイバーチャネルストレージファブリック、またはイーサネットネットワークでホストにプロビジョニングされる VDO ボリューム (またはボリュームの一部) でストレージ領域を回収します。

手順

- SCSI イニシエーターは、**UNMAP** コマンドを使用して、シンプロビジョニングしたストレージターゲットで領域を解放できますが、SCSI ターゲットフレームワークに、このコマンドのサポートを通知するように設定する必要があります。これは、通常、このボリュームでシンプロビジョニングを有効にすることで行います。
  次のコマンドを実行して、Linux ベースの SCSI イニシエーターで **UNMAP** のサポートを検証します。

  ```
  # sg_vpd --page=0xb0 /dev/device
  ```

  出力では、**Maximum unmap LBA count(未マッピングの LBA の最大数)**の値がゼロより大きいことを確認します。

37.2.2. VDO ボリュームの開始または停止

指定した VDO ボリューム、またはすべての VDO ボリューム、および関連の UDS インデックスを起動または停止できます。

37.2.2.1. 起動して有効にした VDO ボリューム

システムの起動時に、**vdo systemd** ユニットは、**有効** に設定されている VDO デバイスをすべて自動的に **起動** します。

vdo パッケージをインストールすると、**vdo systemd** ユニットがインストールされ、デフォルトで有効になっています。このユニットは、システム起動時に **vdo start --all** コマンドを自動的に実行して、有効になっている VDO ボリュームをすべて起動します。

--activate=disabled オプションを **vdo create** コマンドに指定することで、自動的に起動しない VDO ボリュームを作成できます。

開始順序

システムによっては、LVM ボリュームを、VDO ボリュームの上または下の両方に配置できるものがあります。このようなシステムでは、適切な順序でサービスを開始する必要があります。

1. 下層の LVM を最初に起動する必要があります。大概のシステムでは、LVM パッケージがインストールされていれば、この層が起動するように自動的に設定されます。

2. 次に、**vdo systemd** ユニットを起動する必要があります。

3. 最後に、稼働している VDO の上にある LVM ボリュームやその他のサービスを実行するために、その他のスクリプトを実行する必要があります。

ボリュームの停止にかかる時間

VDO ボリュームの停止にかかる時間は、ストレージデバイスの速度と、ボリュームへの書き込みが必要なデータ量により異なります。

- ボリュームは、UDS インデックスの 1GiB ごとに、約 1 GiB のデータを常に書き込みます。
- ボリュームはブロックマップキャッシュのサイズと、スラブごとに最大 8MiB のデータ量を書き込みます。
- ボリュームは、未処理のすべての IO 要求の処理を終了する必要があります。

37.2.2.2. VDO ボリュームの作成

この手順では、システムにある一部の VDO ボリューム、またはすべての VDO ボリュームを起動します。

手順

- 特定の VDO ボリュームを起動するには、以下のコマンドを使用します。

```
# vdo start --name=my-vdo
```

- すべての VDO ボリュームを起動するには、次のコマンドを実行します。

```
# vdo start --all
```

関連情報

- man ページの **vdo(8)**

37.2.2.3. VDO ボリュームの停止

この手順では、システムで一部の VDO ボリュームまたはすべての VDO ボリュームを停止します。

手順

1. ボリュームを停止します。
   - 特定の VDO ボリュームを停止するには、以下のコマンドを使用します。

```
# vdo stop --name=my-vdo
```

   - すべての VDO ボリュームを停止するには、以下のコマンドを使用します。

```
# vdo stop --all
```

2. ボリュームが、ディスクへのデータの書き込みを終了するまで待ちます。

関連情報

- man ページの **vdo(8)**

37.2.2.4. 関連情報

- シャットダウンが正常に行われなかった場合は、システムを再起動したときに VDO が再構築を行いメタデータの一貫性を確認し、必要であれば修復します。再ビルドプロセスの詳細は、「シャットダウンが適切に行われない場合の VDO ボリュームの復旧」 を参照してください。

37.2.3. システムブートで VDO ボリュームを自動的に起動

VDO ボリュームを設定し、システムの起動時に自動的に起動するようにします。自動起動を無効にすることもできます。

37.2.3.1. 起動して有効にした VDO ボリューム

システムの起動時に、**vdo systemd** ユニットは、**有効** に設定されている VDO デバイスをすべて自動的に **起動** します。

vdo パッケージをインストールすると、**vdo systemd** ユニットがインストールされ、デフォルトで有効になっています。このユニットは、システム起動時に **vdo start --all** コマンドを自動的に実行して、有効になっている VDO ボリュームをすべて起動します。

--activate=disabled オプションを **vdo create** コマンドに指定することで、自動的に起動しない VDO ボリュームを作成できます。

開始順序

システムによっては、LVM ボリュームを、VDO ボリュームの上または下の両方に配置できるものがあります。このようなシステムでは、適切な順序でサービスを開始する必要があります。

1. 下層の LVM を最初に起動する必要があります。大概のシステムでは、LVM パッケージがインストールされていれば、この層が起動するように自動的に設定されます。
2. 次に、**vdo systemd** ユニットを起動する必要があります。
3. 最後に、稼働している VDO の上にある LVM ボリュームやその他のサービスを実行するために、その他のスクリプトを実行する必要があります。

ボリュームの停止にかかる時間

VDO ボリュームの停止にかかる時間は、ストレージデバイスの速度と、ボリュームへの書き込みが必要なデータ量により異なります。

- ボリュームは、UDS インデックスの 1GiB ごとに、約 1 GiB のデータを常に書き込みます。
- ボリュームはブロックマップキャッシュのサイズと、スラブごとに最大 8MiB のデータ量を書き込みます。
- ボリュームは、未処理のすべての IO 要求の処理を終了する必要があります。

37.2.3.2. VDO ボリュームの有効化

この手順では、VDO ボリュームを有効にして、自動的に開始できるようにします。

手順

- 特定のボリュームを有効にするには、以下のコマンドを実行します。

```
# vdo activate --name=my-vdo
```

- すべてのボリュームを有効にするには、以下のコマンドを実行します。

```
# vdo activate --all
```

関連情報

- man ページの **vdo(8)**

37.2.3.3. VDO ボリュームの無効化

この手順では、VDO ボリュームを無効にして、自動的に起動しないようにします。

手順

- 特定のボリュームを無効にするには、以下のコマンドを実行します。

```
# vdo deactivate --name=my-vdo
```

- すべてのボリュームを無効にするには、以下のコマンドを実行します。

```
# vdo deactivate --all
```

関連情報

- man ページの **vdo(8)**

37.2.4. VDO 書き込みモードの選択

基となるブロックデバイスでの要件に応じて、VDO ボリュームに書き込みモードを設定できます。デフォルトでは、VDO は書き込みモードを自動的に選択します。

37.2.4.1. VDO 書き込みモード

VDO は、以下の書き込みモードに対応します。

sync

VDO が **sync** モードの場合、その上の層は、書き込みコマンドがデータを永続ストレージに書き込むことを想定します。したがって、このモードは、ファイルシステムやアプリケーションには必要ありません。FLUSH リクエストまたは FUA (強制ユニットアクセス) リクエストを発行すると、データは、重要な点で持続します。
VDO は、書き込みコマンドが完了したときに、基となるストレージが、データが永続ストレージに書き込まれることを保証する場合に限り、**sync** モードに設定する必要があります。つまり、ストレージには揮発性の書き込みキャッシュがないか、ライトスルーキャッシュが存在する必要があります。

async

VDO が **async** モードの場合は、書き込みコマンドが承認されたときに、データが永続ストレージに書き込まれることを VDO が保証しません。ファイルシステムまたはアプリケーションは、各トラン

ザクションの重要な点でデータの永続性を保証するために、FLUSH リクエストまたは FUA リクエストを発行する必要があります。
書き込みコマンドが完了したときに、基となるストレージが永続ストレージに対するデータの書き込みを保証しない場合は、VDO を **async** モードに設定する必要があります。これは、ストレージに揮発性のあるライトバックキャッシュがある場合です。

async-unsafe

このモードには、**async** と同じプロパティーがありますが、ACID (Atomicity, Consistency, Isolation, Durability) に準拠していません。**async** と比較して、**async-unsafe** のパフォーマンスは向上します。

警告

VDO ボリュームに関する ACID コンプライアンスを想定するアプリケーションまたはファイルシステムが稼働する場合は、**async-unsafe** モードにより予想外のデータ損失が生じる可能性があります。

auto

auto モードは、各デバイスの性質に基づいて、**sync** または **async** を自動的に選択します。以下はデフォルトのオプションになります。

37.2.4.2. VDO 書き込みモードの内部処理

VDO の書き込みモードは **sync** と **async** です。以下では、これらのモードの操作について説明します。

kvdo モジュールが同期 (**synch**) モードで動作している場合は、以下を行います。

1. リクエストのデータを一時的に、割り当てられたブロックに書き込み、リクエストを承認します。
2. 承認が完了すると、ブロックデータの MurmurHash-3 署名を計算してブロックの重複排除が試行されます。これは、VDO インデックスに送信されます。
3. VDO インデックスに同じ署名とともにブロックのエントリーが含まれる場合、**kvdo** は示されたブロックを読み込み、同一であるかを検証するために 2 つのブロックのバイト対バイトの比較を行います。
4. 同一であることが確認されると、**kvdo** はブロックマップを更新して、論理ブロックが、一致する物理ブロックを指定し、割り当てられた物理ブロックをリリースします。
5. VDO インデックスに、書き込まれているブロックの署名のエントリーを含まない場合や、示されたブロックが同じデータを含まない場合は、**kvdo** はブロックマップを更新して、一時的な物理ブロックを永続的にします。

kvdo が非同期 (**async**) モードで動作している場合は、以下のコマンドを実行します。

1. データを書き込む代わりに、リクエストをすぐに承認します。
2. 上記の説明と同じように、ブロックの重複排除試行が行われます。

3. ブロックが重複していることになると、**kvdo** はブロックマップを更新し、割り当てられたブロックを解放します。解放しない場合は、リクエストのデータが、割り当てられたブロックに書き込み、ブロックマップを更新して物理ブロックを永続的にします。

37.2.4.3. VDO ボリュームにおける書き込みモードの確認

この手順では、選択した VDO ボリュームに対するアクティブな書き込みモードをリスト表示します。

手順

- 以下のコマンドを使用して、VDO ボリュームが使用する書き込みモードを表示します。

```
# vdo status --name=my-vdo
```

出力されるファイルには、以下が記載されます。

- **設定した書き込みポリシー** - **sync**、**async**、または **auto** から選択されるオプションです。
- **書き込みポリシー** - VDO が適用される特定の書き込みモードで、**sync** または **async** になります。

37.2.4.4. 揮発性キャッシュの確認

この手順では、ブロックデバイスに揮発性キャッシュがあるかどうかを確認します。情報を使用して、VDO 書き込みモードである **sync** と **async** のいずれかを選択できます。

手順

1. デバイスにライトバックキャッシュがあるかどうかを判断する場合は、以下のいずれか方法を使用します。
   - **sysfs** ファイルの **/sys/block/block-device/device/scsi_disk/identifier/cache_type** を読み込みます。以下に例を示します。

```
$ cat '/sys/block/sda/device/scsi_disk/7:0:0:0/cache_type'

write back
```

```
$ cat '/sys/block/sdb/device/scsi_disk/1:2:0:0/cache_type'

None
```

   - また、カーネルブートログでは、上記のデバイスに書き込みキャッシュがあるかどうかを調べることができます。

```
sd 7:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or
FUA
sd 1:2:0:0: [sdb] Write cache: disabled, read cache: disabled, supports DPO and FUA
```

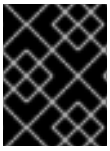
2. 上の例では、以下のようになります。
   - デバイス **sda** には、ライトバックキャッシュが**ある** ことが示されています。**async** モードを使用します。

- デバイス **sdb** には、ライトバックキャッシュが **ない** ことが示されています。**sync** モードを使用します。

cache_type の値が **None** または **write through** である場合は、**sync** 書き込みモードを使用するように VDO を設定する必要があります。

37.2.4.5. VDO 書き込みモードの設定

この手順では、既存のボリュームの場合、またはボリュームの新規作成時に、VDO ボリュームに書き込みモードを設定します。



重要

誤った書き込みモードを使用すると、停電、システムクラッシュ、またはディスクとの接続が予期せず失われた時に、データが失われることがあります。

前提条件

- デバイスに対してどの書き込みモードが正しいかを確認している。「揮発性キャッシュの確認」 を参照してください。

手順

- 既存の VDO ボリュームまたは新規ボリュームの作成時に、書き込みモードを設定できます。
  - 既存の VDO ボリュームを変更するには、以下のコマンドを使用します。

```
# vdo changeWritePolicy --writePolicy=sync|async|async-unsafe|auto \
                        --name=vdo-name
```

  - VDO ボリュームの作成時に書き込みモードを指定するには、**--writePolicy=sync|async|async-unsafe|auto** オプションを **vdo create** コマンドに追加します。

37.2.5. シャットダウンが適切に行われない場合の VDO ボリュームの復旧

シャットダウンが適切に行われない場合に VDO ボリュームを復旧して、動作を継続できます。多くのタスクは自動化されています。また、プロセスの障害により VDO ボリュームの作成に失敗した場合は、クリーンアップできます。

37.2.5.1. VDO 書き込みモード

VDO は、以下の書き込みモードに対応します。

sync

VDO が **sync** モードの場合、その上の層は、書き込みコマンドがデータを永続ストレージに書き込むことを想定します。したがって、このモードは、ファイルシステムやアプリケーションには必要ありません。FLUSH リクエストまたは FUA (強制ユニットアクセス) リクエストを発行すると、データは、重要な点で持続します。
VDO は、書き込みコマンドが完了したときに、基となるストレージが、データが永続ストレージに書き込まれることを保証する場合に限り、**sync** モードに設定する必要があります。つまり、ストレージには揮発性の書き込みキャッシュがないか、ライトスルーキャッシュが存在する必要があります。

async

VDO が **async** モードの場合は、書き込みコマンドが承認されたときに、データが永続ストレージに書き込まれることを VDO が保証しません。ファイルシステムまたはアプリケーションは、各トランザクションの重要な点でデータの永続性を保証するために、FLUSH リクエストまたは FUA リクエストを発行する必要があります。
書き込みコマンドが完了したときに、基となるストレージが永続ストレージに対するデータの書き込みを保証しない場合は、VDO を **async** モードに設定する必要があります。これは、ストレージに揮発性のあるライトバックキャッシュがある場合です。

async-unsafe

このモードには、**async** と同じプロパティーがありますが、ACID (Atomicity, Consistency, Isolation, Durability) に準拠していません。**async** と比較して、**async-unsafe** のパフォーマンスは向上します。

警告

VDO ボリュームに関する ACID コンプライアンスを想定するアプリケーションまたはファイルシステムが稼働する場合は、**async-unsafe** モードにより予想外のデータ損失が生じる可能性があります。

auto

auto モードは、各デバイスの性質に基づいて、**sync** または **async** を自動的に選択します。以下はデフォルトのオプションになります。

37.2.5.2. VDO ボリュームの復旧

シャットダウンが適切に行われなかった場合に VDO ボリュームを再起動すると、VDO は以下の操作を実行します。

- ボリューム上のメタデータの一貫性を検証する
- メタデータの一部を再構築して、必要に応じて修復する

再構築は自動で行われ、ユーザーの介入は必要ありません。

VDO は、アクティブな書き込みモードに依存する各種書き込みを再構築します。

sync

VDO が同期ストレージで稼働していて、書き込みポリシーが **sync** に設定されていた場合は、ボリュームに書き込まれたすべてのデータが完全に復元されます。

async

書き込みポリシーが **async** であった場合、一部の書き込みは永続性が保たれないと復元されないことがあります。これは、VDO に **FLUSH** コマンド、または FUA (強制ユニットアクセス) フラグでタグ付けされた書き込み I/O を送信することで行われます。これは、**fsync**、**fdatasync**、**sync**、**umount** などのデータ整合性の操作を呼び出すことで、ユーザーモードから実行できます。

どちらのモードであっても、フラッシュによって承認されていない、あるいは確認されていない一部の書き込みも再構築されることがあります。

自動復元および手動復元
VDO ボリュームが **復旧** 操作モードになると、VDO は、オンラインに戻ってから、適切ではない VDO ボリュームを自動的に再構築します。これは **オンラインリカバリー** と呼ばれます。

VDO が正常に VDO ボリュームを復元できない場合は、ボリュームの再起動後も持続する **読み取り専用** モードにボリュームを置きます。再構築が強制されるため、問題を手動で修正する必要があります。

関連情報

- 自動リカバリーおよび手動リカバリー、ならびに VDO 操作モードの詳細は、「VDO 操作モード」 を参照してください。

37.2.5.3. VDO 操作モード

ここでは、VDO ボリュームが正常に動作しているか、エラーからの復旧であることを示すモードを説明します。

vdostats --verbose device コマンドを使用すると、VDO ボリュームの現在の操作モードを表示できます。出力内の **Operating mode** 属性を参照してください。

normal
: これがデフォルトの操作モードです。以下のいずれかのステータスにより別のモードが強制されない限りに、VDO ボリュームは常に **normal** モードになります。新規作成された VDO ボリュームは **normal** モードで起動します。

recovering
: シャットダウンの前に、VDO ボリュームがすべてのメタデータを保存しない場合は、次に起動した時に自動的に **recovering** モードになります。このモードになる一般的な理由は、突然の停電や、基となるストレージデバイスの問題です。
recovering モードでは、VDO は、デバイスのデータの物理ブロックごとに参照カウントを修正します。通常、リカバリーにはかなり時間がかかります。その時間は、VDO ボリュームの大きさ、基となるストレージデバイスの速度、VDO が同時に処理するリクエスト数により異なります。VDO ボリュームは、通常は以下の例外で動作します。

  - 最初に、ボリュームに対する書き込み要求に利用できる領域の量が制限される場合があります。復旧するメタデータの数が多いと、より多くの空き領域が利用可能になります。
  - VDO ボリュームの復旧中に書き込まれたデータは、そのデータが、復旧していないボリュームに含まれる場合に、クラッシュする前に書き込まれたデータに対する重複排除に失敗することがあります。VDO は、ボリュームの復旧中にデータを圧縮できます。圧縮したブロックの読み取りや上書きは可能です。
  - オンラインリカバリーを行う際、一部の統計 (**blocks in use** や **blocks free** など) は利用できません。この統計は、再構築が完了すると利用できます。
  - 継続中の復旧作業により、読み取りと書き込みの応答時間が通常よりも遅いことがあります。

  recovering モードでは、VDO ボリュームを問題なくシャットダウンできます。シャットダウンする前に復元が完了しないと、デバイスは、次回起動時に再度 **recovering** モードになります。

VDO ボリュームは自動的に **recovering** モードを終了し、すべての参照カウントが修正されると、**normal** モードに移行します。管理者アクションは必要ありません。詳細は、「VDO ボリュームのオンラインリカバリー」 を参照してください。

read-only

VDO ボリュームが致命的な内部エラーに遭遇すると、**read-only** モードになります。**read-only** モードになるイベントには、メタデータの破損や、バッキングストレージデバイスが読み取り専用になるなどが挙げられます。このモードはエラー状態です。
read-only モードでは、データ読み取りは正常に機能しますが、データの書き込みは常に失敗します。管理者が問題を修正するまで、VDO ボリュームは **read-only** モードのままになります。

VDO ボリュームを **read-only** モードで問題なくシャットダウンできます。通常、モードは、VDO ボリュームが再起動した後も持続します。まれに、VDO ボリュームはバッキングストレージデバイスに **read-only** 状態を記録することができません。このような場合、VDO は代わりに復旧を試みます。

ボリュームが読み取り専用モードの場合、ボリュームのデータが損失または破損していないという保証はありません。このような場合、Red Hat は、読み取り専用のボリュームからデータをコピーして、バックアップからボリュームを復旧することを推奨します。

データ破損のリスクを許容できる場合は、VDO ボリュームのメタデータのオフライン再構築を強制することで、ボリュームをオンラインに戻して利用できるようにできます。再構築されたデータの整合性は保証できません。詳細は、「VDO ボリュームメタデータのオフライン再構築の強制」 を参照してください。

37.2.5.4. VDO ボリュームのオンラインリカバリー

この手順では、シャットダウンが正常に行われなかったときに、VDO ボリュームでオンラインリカバリーを実行して、メタデータを復旧します。

手順

1. VDO ボリュームを起動していない場合は、起動します。

```
# vdo start --name=my-vdo
```

   その他に何か行う必要はありません。復元は、バックグラウンドで実行します。

2. **blocks in use**、**blocks free** などのボリューム統計を使用する場合は、利用可能になるまで待ちます。

37.2.5.5. VDO ボリュームメタデータのオフライン再構築の強制

この手順では、シャットダウンが正常に行われなかった場合に、VDO ボリュームメタデータの強制的なオフライン再構築を実行して、復旧します。

警告

この手順では、ボリュームでデータが失われることがあります。

前提条件

- VDO ボリュームが起動している。

手順

1. ボリュームが読み取り専用モードかどうかを確認します。コマンド出力で **operating mode** 属性を確認します。

```
# vdo status --name=my-vdo
```

ボリュームが読み取り専用モードではない場合は、オフラインの再構築を強制する必要はありません。「VDO ボリュームのオンラインリカバリー」 に従って、オンラインリカバリーを実行します。

2. ボリュームが稼働している場合は停止します。

```
# vdo stop --name=my-vdo
```

3. **--forceRebuild** オプションを指定して、ボリュームを再起動します。

```
# vdo start --name=my-vdo --forceRebuild
```

37.2.5.6. 作成に失敗した VDO ボリュームの削除

この手順では、中間状態で VDO ボリュームをクリーンアップします。ボリュームの作成時に障害が発生した場合、ボリュームは中間状態になります。たとえば、以下のような場合に発生する可能性があります。

- システムのクラッシュ
- 停電
- 管理者が、実行中の **vdo create** コマンドに割り込み

手順

- クリーンアップを行う場合は、**--force** オプションを使用して、作成に失敗したボリュームを削除します。

```
# vdo remove --force --name=my-vdo
```

ボリュームの作成に失敗して、管理者がシステム設定を変更して競合を発生させたため、**--force** オプションが必要となります。

--force オプションを指定しないと、**vdo remove** コマンドが失敗して、以下のメッセージが表示されます。

```
[...]
A previous operation failed.
Recovery from the failure either failed or was interrupted.
Add '--force' to 'remove' to perform the following cleanup.
Steps to clean up VDO my-vdo:
umount -f /dev/mapper/my-vdo
```

```
udevadm settle
dmsetup remove my-vdo
vdo: ERROR - VDO volume my-vdo previous operation (create) is incomplete
```

37.2.6. UDS インデックスの最適化

UDS インデックスを特定の設定を行い、システムで最適化できます。

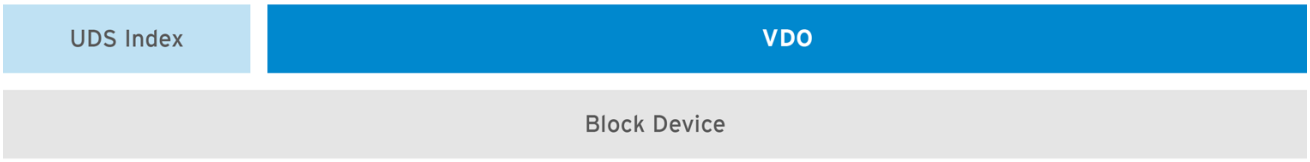
重要

VDO ボリュームを **作成したら**、UDS インデックスのプロパティーを変更することはできません。

37.2.6.1. VDO ボリュームのコンポーネント

VDO は、バッキングストアとしてブロックデバイスを使用します。これは、複数のディスク、パーティション、またはフラットファイルで設定される物理ストレージの集約を含めることができます。ストレージ管理ツールが VDO ボリュームを作成すると、VDO は、UDS インデックスおよび VDO ボリュームのボリューム領域を予約します。UDS インデックスと VDO ボリュームは対話して、重複排除したブロックストレージを提供します。

図37.6 VDO ディスク組織



RHEL_466924_0218

VDO ソリューションは、以下のコンポーネントで設定されます。

kvdo

Linux Device Mapper 層に読み込まれるカーネルモジュールは、重複排除され、圧縮され、シンプロビジョニングされたブロックストレージボリュームを提供します。
kvdo モジュールはブロックデバイスを公開します。ブロックストレージ用にこのブロックデバイスに直接アクセスするか、XFS や ext4 などの Linux ファイルシステムを介して提示することができます。

kvdo が VDO ボリュームからデータ論理ブロックを読み取る要求を受信すると、要求された論理ブロックを基礎となる物理ブロックにマッピングし、要求したデータを読み取り、返します。

kvdo が VDO ボリュームにデータブロックを書き込む要求を受信すると、まず要求が DISCARD または TRIM のものであるか、またはデータが一貫してゼロかどうかを確認します。これらの条件のいずれかが true の場合、**kvdo** はブロックマップを更新し、リクエストを承認します。そうでない場合は、VDO はデータを処理して最適化します。

uds

ボリューム上の Universal Deduplication Service (UDS) インデックスと通信し、データの重複を分析するカーネルモジュール。新しい各データについて、その部分が保存してあるデータ内容と同一であるかどうかを UDS が素早く判断します。インデックスが一致すると、ストレージシステムは、同じ情報を複数格納しないように、既存の項目を内部的に参照できます。
UDS インデックスは、**uds** カーネルモジュールとしてカーネル内で実行します。

コマンドラインツール

最適化されたストレージの設定および管理

37.2.6.2. UDS インデックス

VDO は、UDS と呼ばれる高パフォーマンスの重複排除インデックスを使用して、格納されているときにデータの重複ブロックを検出します。

UDS インデックスは、VDO 製品の基盤を提供します。新しい各データについて、その部分が保存してあるデータ内容と同一であるかどうかを素早く判断します。インデックスが一致すると、ストレージシステムは、同じ情報を複数格納しないように、既存の項目を内部的に参照できます。

UDS インデックスは、**uds** カーネルモジュールとしてカーネル内で実行します。

重複排除ウィンドウ は、以前書き込んだことをインデックスが記憶しているブロックの数です。重複排除ウィンドウのサイズは設定可能です。特定のウィンドウサイズでは、インデックスに特定の RAM のサイズと、特定のディスク領域が必要です。ウィンドウのサイズは、通常、**--indexMem=size** オプションを使用してインデックスメモリーのサイズを指定して決定されます。VDO は、自動的に使用するディスク領域を決定します。

UDS インデックスは 2 つの部分から成ります。

- 一意のブロックごとに最大 1 つのエントリーを含むメモリーでは、コンパクトな表が用いられています。
- 発生する際に、インデックスに対して示される関連のブロック名を順に記録するオンディスクコンポーネント。

UDS は、メモリーの各エントリーに対して平均 4 バイトを使用します (キャッシュを含む)。

オンディスクコンポーネントは、UDS に渡されるデータの境界履歴を維持します。UDS は、直近で確認されたブロックの名前を含む、この重複排除ウィンドウ内のデータの重複排除アドバイスを提供します。重複排除ウィンドウでは、大規模なデータリポジトリーに対して必要なメモリーの量を制限する際に、UDS はできるだけ効率的にデータのインデックスを作成します。重複排除ウィンドウの境界の特徴により、重複排除レベルが高い多くのデータセットでは、一時的な局所性も多く確認されます。つまり、多くの重複排除は、同時に書き込まれたブロックセット間で発生します。さらに、通常、書き込まれたデータは、以前に書き込まれたデータよりも、最近書き込まれたデータを複製する可能性が高くなります。したがって、特定の時間間隔での特定のワークロードでは、UDS が最新のデータのみをインデックス付けしても、すべてのデータをインデックス付けしても、重複排除率は同じになることがよくあります。

重複データの場合では、一時的な局所性が示される傾向もあるため、ストレージシステム内のすべてのブロックにインデックスを作成する必要はほとんどありません。そうでない場合、インデックスメモリーのコストは、重複排除によるストレージコストの削減を上回ります。インデックスサイズの要件は、データの摂取率に密接に関連します。たとえば、合計容量が 100 TB のストレージシステムには、毎週 1 TB の摂取率を指定します。UDS は、4 TB の重複排除ウィンドウにより、前の月に書き込まれたデータで、最も冗長性が大きいものを検出できます。

37.2.6.3. 推奨される UDS インデックス設定

本セクションでは、目的のユースケースに基づいて、UDS インデックスとともに使用することが推奨されるオプションを説明します。

一般的には、Red Hat は、すべての実稼働環境に **sparse** の UDS インデックスを使用することを推奨します。これは、非常に効率的なインデックスデータ構造であり、重複排除ウィンドウでブロックごとに RAM の約 10 分の 1 バイトが必要です。ディスクの場合は、ブロックごとに約 72 バイトのディスク領

域が必要です。このインデックスの最小設定は、ディスクで 256 MB の RAM と約 25 GB の領域を使用します。

この設定を使用するには、**--sparseIndex=enabled --indexMem=0.25** オプションを **vdo create** コマンドに指定します。この設定により、2.5 TB の重複排除ウィンドウが作成されます (つまり、2.5 TB の履歴が記録されます)。ほとんどのユースケースでは、2.5 TB の重複排除ウィンドウは、サイズが 10 TB までのストレージプールを重複排除するのに適しています。

ただし、インデックスのデフォルト設定は、**dense** のインデックスを使用します。このインデックスは RAM では非常に効率が悪い (10 倍) ですが、必要なディスク領域もはるかに少ない (10 倍) ため、制約された環境での評価に便利です。

一般に、推奨される設定は、VDO ボリュームの物理サイズの 4 分の 1 の重複排除ウィンドウです。ただし、これは実際の要件ではありません。小さな重複排除ウィンドウ (物理ストレージの量と比較) であっても、多くのユースケースで大量の重複データを確認できます。大概のウィンドウも使用できますが、ほとんどの場合、それを行う利点はほとんどありません。

関連情報

- この重要なシステムパラメーターの調整は、Red Hat テクニカルアカウントマネージャーまでご相談ください。

37.2.7. VDO での重複排除の有効化または無効化

一部の例では、ボリュームへの読み書きを行う機能を維持しながら、VDO ボリュームに書き込まれているデータの重複排除を一時的に無効にする場合があります。重複排除を無効にすると、後続の書き込みが重複排除できなくなりますが、すでに重複排除されたデータが残ります。

37.2.7.1. VDO での重複排除

重複排除とは、重複ブロックの複数のコピーを削除することで、ストレージリソースの消費を低減させるための技術です。

同じデータを複数回書き込むのではなく、VDO は各重複ブロックを検出し、元のブロックへの参照として記録します。VDO は、VDO の上にあるストレージ層により使用されている論理ブロックアドレスから、VDO の下にあるストレージ層で使用される物理ブロックアドレスへのマッピングを維持します。

重複排除を行った後、複数の論理ブロックアドレスが同じ物理ブロックアドレスにマッピングできます。共有ブロックと呼ばれます。ブロックストレージの共有は、VDO が存在しない場合に、読み込みブロックと書き込みブロックが行われるストレージのユーザーには表示されません。

共有ブロックが上書きされると、VDO は新しいブロックデータを保存する新しい物理ブロックを割り当て、共有物理ブロックにマッピングされたその他の論理ブロックアドレスが変更されないようにします。

37.2.7.2. VDO ボリュームでの重複排除の有効化

これにより、関連の UDS インデックスを再起動し、重複排除が再びアクティブになったことを VDO ボリュームに認識させます。

注記

重複排除はデフォルトで有効になっています。

手順

- VDO ボリュームでの重複排除を再起動するには、以下のコマンドを使用します。

```
# vdo enableDeduplication --name=my-vdo
```

37.2.7.3. VDO ボリュームでの重複排除の無効化

これにより、関連の UDS インデックスを停止し、重複排除がアクティブでなくなったことを VDO ボリュームに認識させます。

手順

- VDO ボリュームでの重複排除を停止するには、以下のコマンドを使用します。

```
# vdo disableDeduplication --name=my-vdo
```

- **--deduplication=disabled** オプションを **vdo create** コマンドに指定することで、新しい VDO ボリュームを作成するときに重複排除を無効にできます。

37.2.8. VDO で圧縮の有効化または無効化

VDO は、データ圧縮を提供します。これを無効にすると、パフォーマンスが最大化され、圧縮される可能性が低いデータの処理が高速化されます。再度有効にすると、スペースを節約できます。

37.2.8.1. VDO の圧縮

VDO は、ブロックレベルの重複排除の他に、HIOPS compression™ 技術を使用してインラインブロックレベルの圧縮も提供します。

VDO ボリューム圧縮はデフォルトでオンになっています。

重複排除は、仮想マシン環境とバックアップアプリケーションに最適なソリューションですが、圧縮は、通常、ログファイルやデータベースなどのブロックレベルの冗長性を見せない構造化および非構造化のファイル形式に非常に適しています。

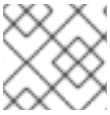
圧縮は、重複として認識されていないブロックで動作します。VDO が初めて一意のデータを見つけた場合は、データを圧縮します。その後の、保存しているデータのコピーは、追加の圧縮手順なしに重複排除されます。

圧縮機能は、同時に多くの圧縮操作に対応できるようにする並列化されたパッケージアルゴリズムに基づいています。最初にブロックを保存し、リクエストに応答したら、圧縮時に複数のブロックを検出する最適なパックアルゴリズムが、1 つの物理ブロックに適合します。特定の物理ブロックが追加の圧縮ブロックを保持していないと判断すると、そのブロックはストレージに書き込まれます。圧縮されていないブロックは解放され、再利用されます。

要求したものに応答し、圧縮およびパッケージ化の操作を実行することにより、圧縮を使用することで課せられるレイテンシーが最小限に抑えられます。

37.2.8.2. VDO ボリュームでの圧縮の有効化

この手順では、VDO ボリュームで圧縮を有効化して、削減する領域を大きくします。



注記

デフォルトでは圧縮が有効になっています。

手順

- 再び起動するには、以下のコマンドを使用します。

```
# vdo enableCompression --name=my-vdo
```

37.2.8.3. VDO ボリュームでの圧縮の無効化

この手順では、VDO ボリュームで圧縮を停止して、パフォーマンスを最大化します。もしくは、圧縮できないと思われるデータの処理を高速化します。

手順

- 既存の VDO ボリュームで圧縮を停止するには、次のコマンドを使用します。

```
# vdo disableCompression --name=my-vdo
```

- もしくは、新規ボリュームの作成時に、**--compression=disabled** オプションを **vdo create** コマンドに指定して実行すると、圧縮を無効にできます。

37.2.9. VDO ボリュームのサイズの拡大

VDO ボリュームの物理サイズを増やして、基となるストレージの容量をさらに利用したり、ボリュームの論理サイズを大きくして、ボリュームが多くの領域を使用できるようにできます。

37.2.9.1. VDO ボリュームの物理サイズおよび論理サイズ

VDO は、物理サイズ、利用可能な物理サイズ、および論理サイズを次の方法で利用します。

物理サイズ

これは、基礎となるブロックデバイスと同じサイズです。VDO は、以下の目的でこのストレージを使用します。

- 重複排除および圧縮される可能性があるユーザーデータ
- UDS インデックスなどの VDO メタデータ

利用可能な物理サイズ

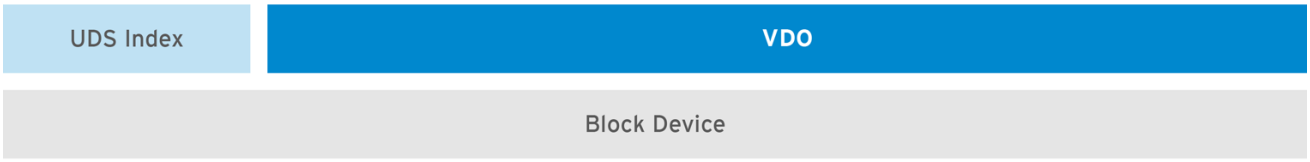
これは、VDO がユーザーデータに使用できる物理サイズの一部です。
これは、メタデータのサイズを引いた物理サイズと同等で、指定のスラブサイズでボリュームをスラブに分割した後の残りを引いたものと同じです。

論理サイズ

これは、VDO ボリュームがアプリケーションに提示するプロビジョニングされたサイズです。通常、これは利用可能な物理サイズよりも大きくなります。**--vdoLogicalSize** オプションを指定しないと、論理ボリュームのプロビジョニングが **1:1** の比率にプロビジョニングされます。たとえば、VDO ボリュームが 20 GB ブロックデバイスの上に置かれている場合は、2.5 GB が UDS インデックス用に予約されます (デフォルトのインデックスサイズが使用される場合)。残りの 17.5 GB は、

VDO メタデータおよびユーザーデータに提供されます。そのため、消費する利用可能なストレージは 17.5 GB を超えません。実際の VDO ボリュームを設定するメタデータにより、これよりも少なくなる可能性があります。
VDO は現在、絶対最大論理サイズ 4PB の物理ボリュームの最大 254 倍の論理サイズに対応します。

図37.7 VDO ディスク組織



RHEL_466924_0218

この図では、VDO で重複排除したストレージターゲットがブロックデバイス上に完全に配置されています。つまり、VDO ボリュームの物理サイズは、基礎となるブロックデバイスと同じサイズになります。

関連情報

- さまざまなサイズのブロックデバイスに必要なストレージ VDO メタデータのサイズは、「物理サイズ別の VDO 要件の例」 を参照してください。

37.2.9.2. VDO でのシンプロビジョニング

VDO は、シンプロビジョニングされたブロックストレージターゲットです。VDO ボリュームが使用する物理領域のサイズは、ストレージのユーザーに示されるボリュームのサイズとは異なる可能性があります。この相違を活用して、ストレージのコストを削減できます。

容量不足の条件

書き込んだデータが、予想される最適化率に到達できない場合は、ストレージ領域が予想外に不足しないように注意してください。

論理ブロック (仮想ストレージ) の数が物理ブロック (実際のストレージ) の数を超えると、ファイルシステムおよびアプリケーションで領域が予想外に不足する可能性があります。このため、VDO を使用するストレージシステムは、VDO ボリュームの空きプールのサイズを監視する方法で提供する必要があります。

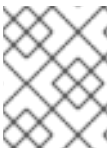
vdostats ユーティリティーを使用すると、この空きプールのサイズを確認できます。このユーティリティーのデフォルト出力には、Linux の **df** ユーティリティーと同様の形式で稼働しているすべての VDO ボリュームの情報が記載されます。以下に例を示します。

```
Device                1K-blocks   Used       Available   Use%
/dev/mapper/vdo-name  211812352   105906176   105906176   50%
```

VDO ボリュームの物理ストレージ領域が不足しそうになると、VDO は、システムログに、以下のような警告を出力します。

```
Oct  2 17:13:39 system lvm[13863]: Monitoring VDO pool vdo-name.
Oct  2 17:27:39 system lvm[13863]: WARNING: VDO pool vdo-name is now 80.69% full.
Oct  2 17:28:19 system lvm[13863]: WARNING: VDO pool vdo-name is now 85.25% full.
```

```
Oct  2 17:29:39 system lvm[13863]: WARNING: VDO pool vdo-name is now 90.64% full.
Oct  2 17:30:29 system lvm[13863]: WARNING: VDO pool vdo-name is now 96.07% full.
```



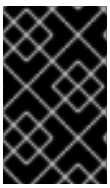
注記

この警告メッセージは、**lvm2-monitor** サービスが実行している場合に限り表示されます。これは、デフォルトで有効になっています。

容量不足の状況を防ぐ方法

空きプールのサイズが特定のレベルを下回る場合は、以下を行うことができます。

- データの削除。これにより、削除したデータが重複していないと、領域が回収されます。データを削除しても、破棄が行われないと領域を解放しません。
- 物理ストレージの追加



重要

VDO ボリュームで物理領域を監視し、領域不足を回避します。物理ブロックが不足すると、VDO ボリュームに最近書き込まれたデータや、未承認のデータが失われることがあります。

シンプロビジョニング、TRIM コマンド、および DISCARD コマンド

シンプロビジョニングによるストレージの削減から利益を得るには、データを削除するタイミングを物理ストレージ層が把握する必要があります。シンプロビジョニングされたストレージで動作するファイルシステムは、**TRIM** コマンドまたは **DISCARD** コマンドを送信して、論理ブロックが不要になったときにストレージシステムに通知します。

TRIM コマンドまたは **DISCARD** コマンドを送信する方法はいくつかあります。

- **discard** マウントオプションを使用すると、ブロックが削除されるたびに、ファイルシステムがそのコマンドを送信できます。
- **fstrim** などのユーティリティーを使用すると、制御された方法でコマンドを送信できます。このようなユーティリティーは、どの論理ブロックが使用されていないかを検出し、**TRIM** コマンドまたは **DISCARD** コマンドの形式でストレージシステムに情報を送信するようにファイルシステムに指示します。

未使用のブロックで **TRIM** または **DISCARD** を使用する必要は、VDO に特有のものではありません。シンプロビジョニングしたストレージシステムでも、同じ課題があります。

37.2.9.3. VDO ボリュームの論理サイズの拡大

この手順では、指定した VDO ボリュームの論理サイズを増やします。最初に、領域が不足しないサイズの論理が含まれる VDO ボリュームを作成できます。しばらくすると、データ消費の実際のレートを評価することができ、十分な場合には、VDO ボリュームの論理サイズを拡張して、削減して得られた領域を活用することができます。

VDO ボリュームの論理サイズを縮小することはできません。

手順

- 論理サイズを拡張するには、次のコマンドを実行します。

```
# vdo growLogical --name=my-vdo \
                  --vdoLogicalSize=new-logical-size
```

論理サイズが増大すると、VDO は新しいサイズのボリュームの上にデバイスまたはファイルシステムに通知します。

37.2.9.4. VDO ボリュームの物理サイズの拡大

この手順では、VDO ボリュームに利用できる物理ストレージの量を増やします。

このように VDO ボリュームを縮小することはできません。

前提条件

- 基となるブロックデバイスのサイズが、VDO ボリュームの現在の物理サイズよりも大きい。そうでない場合は、デバイスのサイズを大きくできます。正確な手順は、デバイスの種類によって異なります。たとえば、MBR パーティションまたは GPT パーティションのサイズ変更は、**ストレージデバイスの管理**の パーティションの使用 を参照してください。

手順

- 新しい物理ストレージ領域を VDO ボリュームに追加します。

```
# vdo growPhysical --name=my-vdo
```

37.2.10. VDO ボリュームの削除

システムで既存の VDO ボリュームを削除できます。

37.2.10.1. 作業中の VDO ボリュームの削除

この手順では、VDO ボリュームと、関連する UDS インデックスを削除します。

手順

1. ファイルシステムのマウントを解除し、VDO ボリュームでストレージを使用しているアプリケーションを停止します。
2. システムから VDO ボリュームを削除するには、次のコマンドを使用します。

```
# vdo remove --name=my-vdo
```

37.2.10.2. 作成に失敗した VDO ボリュームの削除

この手順では、中間状態で VDO ボリュームをクリーンアップします。ボリュームの作成時に障害が発生した場合、ボリュームは中間状態になります。たとえば、以下のような場合に発生する可能性があります。

- システムのクラッシュ
- 停電
- 管理者が、実行中の **vdo create** コマンドに割り込み

手順

- クリーンアップを行う場合は、**--force** オプションを使用して、作成に失敗したボリュームを削除します。

```
# vdo remove --force --name=my-vdo
```

ボリュームの作成に失敗して、管理者がシステム設定を変更して競合を発生させたため、**--force** オプションが必要となります。

--force オプションを指定しないと、**vdo remove** コマンドが失敗して、以下のメッセージが表示されます。

```
[...]
A previous operation failed.
Recovery from the failure either failed or was interrupted.
Add '--force' to 'remove' to perform the following cleanup.
Steps to clean up VDO my-vdo:
umount -f /dev/mapper/my-vdo
udevadm settle
dmsetup remove my-vdo
vdo: ERROR - VDO volume my-vdo previous operation (create) is incomplete
```

37.2.11. 関連情報

- **Ansible** ツールを使用することで、VDO デプロイメントと管理を自動化できます。詳細は、次を参照してください。
  - Ansible ドキュメント -https://docs.ansible.com/
  - VDO Ansible モバイルドキュメンテーション - https://docs.ansible.com/ansible/latest/modules/vdo_module.html

37.3. 未使用ブロックの破棄

破棄操作に対応するブロックデバイスで破棄操作を実行するか、そのスケジュールを設定できます。

37.3.1. ブロックの破棄操作

ブロック破棄操作では、マウントされたファイルシステムで使用されていないブロックを破棄します。これは以下に役立ちます。

- ソリッドステートドライブ (SSD)
- シンプロビジョニングされたストレージ

要件
ファイルシステムの基礎となるブロックデバイスは、物理的な破棄操作に対応している必要があります。

/sys/block/device/queue/discard_max_bytes ファイルの値がゼロではない場合は、物理的な破棄操作に対応しています。

37.3.2. ブロック破棄操作のタイプ

以下のような、さまざまな方法で破棄操作を実行できます。

バッチ破棄

ユーザーが明示的に実行します。選択したファイルシステムのすべての未使用ブロックを破棄します。

オンライン破棄

マウント時に指定されます。ユーザーによる介入なしでリアルタイムで実行されます。オンライン破棄操作は、使用中から空きに移行しているブロックのみを破棄します。

定期的な破棄

systemd サービスが定期的に実行するバッチ操作です。

すべてのタイプは、XFS ファイルシステム、ext4 ファイルシステム、および VDO で対応されています。

推奨事項

Red Hat は、バッチ破棄または周期破棄を使用することを推奨します。

以下の場合にのみ、オンライン破棄を使用してください。

- システムのワークロードでバッチ破棄が実行できない場合
- パフォーマンス維持にオンライン破棄操作が必要な場合

37.3.3. バッチブロック破棄の実行

この手順では、バッチによるブロック破棄操作を実行し、マウントされたファイルシステムの未使用ブロックを破棄します。

前提条件

- ファイルシステムがマウントされている。
- ファイルシステムの基礎となるブロックデバイスが物理的な破棄操作に対応している。

手順

- **fstrim** ユーティリティーを使用します。
  - 選択したファイルシステムでのみ破棄を実行するには、次のコマンドを使用します。

    ```
    # fstrim mount-point
    ```

  - マウントされているすべてのファイルシステムで破棄を実行するには、次のコマンドを使用します。

    ```
    # fstrim --all
    ```

fstrim コマンドを以下のいずれかで実行している場合は、

- 破棄操作に対応していないデバイス
- 複数のデバイスから設定され、そのデバイスの 1 つが破棄操作に対応していない論理デバイス (LVM または MD)

次のメッセージが表示されます。

```
# fstrim /mnt/non_discard

fstrim: /mnt/non_discard: the discard operation is not supported
```

関連情報

- **fstrim(8)** man ページ。

37.3.4. オンラインブロック破棄の有効化

この手順は、サポートされるすべてのファイルシステムで、未使用のブロックを自動的に破棄するオンラインブロック破棄操作を有効にします。

手順

- マウント時のオンライン破棄を有効にします。
  - ファイルシステムを手動でマウントするには、**-o discard** マウントオプションを追加します。

    ```
    # mount -o discard device mount-point
    ```

  - ファイルシステムを永続的にマウントするには、**/etc/fstab** ファイルのマウントエントリーに **discard** オプションを追加します。

関連情報

- **mount(8)** man ページ。
- **fstab(5)** man ページ

37.3.5. 定期的なブロック破棄の有効化

この手順では、対応するすべてのファイルシステムで、未使用のブロックを定期的に破棄する **systemd** タイマーを有効にします。

手順

- **systemd** タイマーを有効にして起動します。

  ```
  # systemctl enable --now fstrim.timer
  ```

37.4. WEB コンソールを使用した VIRTUAL DATA OPTIMIZER ボリュームの管理

RHEL 8 Web コンソールを使用して、VDO (Virtual Data Optimizer) を設定します。

以下の方法について説明します。

- VDO ボリュームの作成

- VDO ボリュームのフォーマット
- VDO ボリュームの拡張

前提条件

- RHEL 8 Web コンソールをインストールし、アクセスできる。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。
- **cockpit-storaged** パッケージがシステムにインストールされている。

37.4.1. Web コンソールでの VDO ボリューム

Red Hat Enterprise Linux 8 では、Virtual Data Optimizer (VDO) がサポートされます。

VDO は、以下を組み合わせたブロック仮想化テクノロジーです。

圧縮
詳細は、Enabling or disabling compression in VDO を参照してください。

重複排除
詳細は、Enabling or disabling compression in VDO を参照してください。

シンプロビジョニング
詳細は、シンプロビジョニングボリューム (シンボリューム) の作成と管理 を参照してください。

このような技術を使用して、VDO は、以下を行います。

- ストレージ領域をインラインに保存します。
- ファイルを圧縮します。
- 重複を排除します。
- 物理ストレージまたは論理ストレージが提供するサイズよりも多くの仮想領域を割り当てることができます。
- 拡大して仮想ストレージを拡張できます。

VDO は、さまざまなタイプのストレージに作成できます。RHEL 8 Web コンソールでは、以下に VDO を設定できます。

- LVM

注記

シンプロビジョニングされたボリュームに VDO を設定することはできません。

- 物理ボリューム
- ソフトウェア RAID

ストレージスタックにおける VDO の配置の詳細は、System Requirements を参照してください。

関連情報

- VDO の詳細は、Deduplicating and compressing storage を参照してください。

37.4.2. Web コンソールで VDO ボリュームの作成

RHEL Web コンソールで VDO ボリュームを作成します。

前提条件

- VDO の作成元となる物理ドライブ、LVM、または RAID

手順

1. RHEL 8 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. **Storage** をクリックします。
3. **VDO Devices** ボックスの **+** ボタンをクリックします。
4. **名前** フィールドに、VDO ボリュームの名前 (スペースなし) を入力します。
5. 使用するドライブを選択します。
6. **論理サイズ** バーに、VDO ボリュームのサイズを設定します。10 回以上拡張できますが、VDO ボリュームを作成する目的を検討してください。
   - アクティブな仮想マシンまたはコンテナーストレージの場合は、使用する論理のサイズを、ボリュームの物理サイズの 10 倍になるようにします。
   - オブジェクトストレージの場合は、使用する論理のサイズを、ボリュームの物理サイズの 3 倍になるようにします。

   詳細は、Deploying VDO を参照してください。
7. **インデックスメモリー** バーで、VDO ボリュームにメモリーを割り当てます。
   VDO システム要件の詳細は、System Requirements を参照してください。
8. **圧縮** オプションを選択します。このオプションを使用すると、さまざまなファイル形式を効率的に減らすことができます。
   詳細は、Enabling or disabling compression in VDO を参照してください。
9. **重複排除** オプションを選択します。
   このオプションは、重複ブロックのコピーを削除して、ストレージリソースが使用されなくなるようにします。詳細は、Enabling or disabling compression in VDO を参照してください。
10. 必要に応じて、512 バイトのブロックサイズを必要とするアプリケーションで VDO ボリュームを使用する場合は、**512 バイトのエミュレーションを使用** を選択します。これにより、VDO ボリュームのパフォーマンスは低下しますが、その必要はほとんどありません。不明な場合は、無効にします。
11. **Create** をクリックします。

検証手順

- **ストレージ** セクションに新しい VDO ボリュームが表示されることを確認します。そして、ファイルシステムでフォーマットすることができます。

37.4.3. Web コンソールで VDO ボリュームのフォーマット

VDO ボリュームは物理ドライブとして動作します。論理ボリュームを使用するには、ファイルシステムでフォーマットする必要があります。

警告

VDO をフォーマットすると、ボリュームのデータがすべて消去されます。

次の手順では、VDO ボリュームをフォーマットする手順を説明します。

前提条件

- VDO ボリュームが作成されている。詳細については、Web コンソールでの VDO ボリュームの作成 を参照してください。

手順

1. RHEL 8 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. **ストレージ** をクリックします。
3. VDO ボリュームをクリックします。
4. **Unrecognized Data** タブをクリックします。
5. **Format** をクリックします。
6. **Erase** ドロップダウンメニューから、以下を選択します。

   既存データを上書きしない

   RHEL Web コンソールは、ディスクヘッダーのみを書き換えます。このオプションの利点は、フォーマットの速度です。

   既存のデータをゼロで上書きする

   RHEL Web コンソールは、ディスク全体をゼロで書き直します。このプログラムはディスク全体を調べるため、このオプションを使用すると遅くなります。ディスクにデータが含まれていて、書き直す必要がある場合は、このオプションを使用します。

7. **種類** ドロップダウンメニューで、ファイルシステムを選択します。

   - **XFS** ファイルシステムは大規模な論理ボリュームをサポートし、オンラインの物理ドライブを停止せずに、既存のファイルシステムの拡大および縮小を行うことができます。別のストレージの使用を希望しない場合は、このファイルシステムを選択したままにしてください。
     XFS は、ボリュームの縮小に対応していません。したがって、XFS でフォーマットしたボリュームを縮小することはできません。

   - **ext4** ファイルシステムは論理ボリュームをサポートし、オンラインの物理ドライブを停止せずに、既存のファイルシステムの拡大および縮小を行うことができます。

LUKS (Linux Unified Key Setup) 暗号を使用したバージョンも選択できます。パスフレーズを使用してボリュームの暗号化を行えます。

8. **名前** フィールドに、論理ボリューム名を入力します。
9. **マウント** ドロップダウンメニューで、**カスタム** を選択します。
   デフォルト オプションでは、システムを次回起動したときにファイルシステムがマウントされているとは限りません。
10. **マウントポイント** フィールドに、マウントパスを追加します。
11. **起動時にマウント** を選択します。
12. **Format** をクリックします。
    フォーマットに使用されるオプションや、ボリュームのサイズによって、フォーマットに数分かかることがあります。

    成功すると、**ファイルシステム** タブに、フォーマットされた VDO ボリュームの詳細が表示されます。
13. VDO ボリュームを使用するには、**マウント** をクリックします。

この時点で、システムが、マウントされてフォーマットされた VDO ボリュームを使用します。

37.4.4. Web コンソールで VDO ボリュームの拡張

RHEL 8 Web コンソールで VDO ボリュームを拡張します。

前提条件

- **cockpit-storaged** パッケージがシステムにインストールされている。
- VDO ボリュームが作成されている。

手順

1. RHEL 8 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. **ストレージ** をクリックします。
3. **VDO デバイス** で、VDO ボリュームをクリックします。
4. VDO ボリュームの詳細で、**Grow** ボタンをクリックします。
5. **VDO の論理サイズを増加** ダイアログボックスで、VDO ボリュームの論理サイズを増やします。

1. **Grow** をクリックします。

検証手順

- 新しいサイズの VDO ボリュームの詳細を確認し、変更が正常に行われたことを確認します。

パート V. ログファイルの設計

第38章 システムの監査

Audit は、追加のセキュリティー機能をシステムに提供するのではありません。システムで使用されるセキュリティーポリシーの違反を発見するために使用できます。このような違反は、SELinux などの別のセキュリティー対策で防ぐことができます。

38.1. LINUX の AUDIT

Linux の Audit システムは、システムのセキュリティー関連情報を追跡する方法を提供します。事前設定されたルールに基づき、Audit は、ログエントリーを生成し、システムで発生しているイベントに関する情報をできるだけ多く記録します。この情報は、ミッションクリティカルな環境でセキュリティーポリシーの違反者と、違反者によるアクションを判断する上で必須のものです。

以下は、Audit がログファイルに記録できる情報の概要です。

- イベントの日時、タイプ、結果
- サブジェクトとオブジェクトの機密性のラベル
- イベントを開始したユーザーの ID とイベントの関連性
- Audit 設定の全修正および Audit ログファイルへのアクセス試行
- SSH、Kerberos、およびその他の認証メカニズムの全使用
- 信頼できるデータベース (**/etc/passwd** など) への変更
- システムからの情報のインポート、およびシステムへの情報のエクスポートの試行
- ユーザー ID、サブジェクトおよびオブジェクトラベルなどの属性に基づく include または exclude イベント

Audit システムの使用は、多くのセキュリティー関連の認定における要件でもあります。Audit は、以下の認定またはコンプライアンスガイドの要件に合致するか、それを超えるように設計されています。

- Controlled Access Protection Profile (CAPP)
- Labeled Security Protection Profile (LSPP)
- Rule Set Base Access Control (RSBAC)
- NISPOM (National Industrial Security Program Operating Manual)
- Federal Information Security Management Act (FISMA)
- PCI DSS (Payment Card Industry Data Security Standard)
- セキュリティー技術実装ガイド (Security Technical Implementation Guide (STIG))

Audit は以下でも認定されています。

- National Information Assurance Partnership (NIAP) および Best Security Industries (BSI) による評価
- Red Hat Enterprise Linux 5 における LSPP/CAPP/RSBAC/EAL4 以降の認定

- Red Hat Enterprise Linux 6 における OSPP/EAL4 以降 (Operating System Protection Profile / Evaluation Assurance Level 4 以降) の認定

ユースケース

ファイルアクセスの監視

Audit は、ファイルやディレクトリーがアクセス、修正、または実行されたか、もしくはファイル属性が変更されたかを追跡できます。これはたとえば、重要なファイルへのアクセスを検出し、これらのファイルが破損した場合に監査証跡を入手可能とする際に役に立ちます。

システムコールの監視

Audit は、一部のシステムコールが使用されるたびにログエントリーを生成するように設定できます。これを使用すると、**settimeofday** や **clock_adjtime**、その他の時間関連のシステムコールを監視することで、システム時間への変更を追跡できます。

ユーザーが実行したコマンドの記録

Audit はファイルが実行されたかどうかを追跡できるため、特定のコマンドの実行を毎回記録するようにルールを定義できます。たとえば、**/bin** ディレクトリー内のすべての実行可能ファイルにルールを定義できます。これにより作成されるログエントリーをユーザー ID で検索すると、ユーザーごとに実行されたコマンドの監査証跡を生成できます。

システムのパス名の実行の記録

ルールの呼び出し時にパスを inode に変換するファイルアクセスをウォッチする以外に、ルールの呼び出し時に存在しない場合や、ルールの呼び出し後にファイルが置き換えられた場合でも、Audit がパスの実行をウォッチできるようになりました。これにより、ルールは、プログラム実行ファイルをアップグレードした後、またはインストールされる前にも機能を継続できます。

セキュリティーイベントの記録

pam_faillock 認証モジュールは、失敗したログイン試行を記録できます。Audit で失敗したログイン試行も記録するように設定すると、ログインを試みたユーザーに関する追加情報が提供されます。

イベントの検索

Audit は **ausearch** ユーティリティーを提供します。これを使用すると、ログエントリーをフィルターにかけ、いくつかの条件に基づく完全な監査証跡を提供できます。

サマリーレポートの実行

aureport ユーティリティーを使用すると、記録されたイベントのデイリーレポートを生成できます。システム管理者は、このレポートを分析し、疑わしいアクティビティーをさらに調べることができます。

ネットワークアクセスの監視

nftables、**iptables**、および **ebtables** ユーティリティーは、Audit イベントを発生するように設定できるため、システム管理者がネットワークアクセスを監視できるようになります。

注記

システムのパフォーマンスは、Audit が収集する情報量によって影響される可能性があります。

38.2. AUDIT システムのアーキテクチャー

Audit システムは、ユーザー空間アプリケーションおよびユーティリティーと、カーネル側のシステムコール処理という 2 つの主要部分で構成されます。カーネルコンポーネントは、ユーザー空間アプリケーションからシステムコールを受け、これを **user**、**task**、**fstype**、または **exit** のいずれかのフィルターで振り分けます。

システムコールが **exclude** フィルターを通過すると、前述のフィルターのいずれかに送られます。このフィルターにより、Audit ルール設定に基づいてシステムコールが Audit デーモンに送信され、さらに処理されます。

ユーザー空間の Audit デーモンは、カーネルから情報を収集し、ログファイルのエントリーを作成します。他のユーザー空間ユーティリティーは、Audit デーモン、カーネルの Audit コンポーネント、または Audit ログファイルと相互作用します。

- **auditctl** - Audit 制御ユーティリティーはカーネル Audit コンポーネントと相互作用し、ルールを管理するだけでなくイベント生成プロセスの多くの設定やパラメーターも制御します。
- 残りの Audit ユーティリティーは、Audit ログファイルのコンテンツを入力として受け取り、ユーザーの要件に基づいて出力を生成します。たとえば、**aureport** ユーティリティーは、記録された全イベントのレポートを生成します。

RHEL 8 では、Audit dispatcher デーモン (**audisp**) 機能は、Audit デーモン (**auditd**) に統合されています。監査イベントと、リアルタイムの分析プログラムの相互作用に使用されるプラグイン設定ファイルは、デフォルトで **/etc/audit/plugins.d/** ディレクトリーに保存されます。

38.3. 環境を保護するための AUDITD の設定

デフォルトの **auditd** 設定は、ほとんどの環境に適しています。ただし、環境が厳格なセキュリティーポリシーを満たす必要がある場合は、**/etc/audit/auditd.conf** ファイル内の Audit デーモン設定に次の設定が推奨されます。

log_file

Audit ログファイル (通常は **/var/log/audit/**) を保持するディレクトリーは、別のマウントポイントにマウントされている必要があります。これにより、その他のプロセスがこのディレクトリー内の領域を使用しないようにし、Audit デーモンの残りの領域を正確に検出します。

max_log_file

1 つの Audit ログファイルの最大サイズを指定します。Audit ログファイルを保持するパーティションで利用可能な領域をすべて使用するように設定する必要があります。**max_log_file`** パラメーターは、最大ファイルサイズをメガバイト単位で指定します。指定する値は、数値にする必要があります。

max_log_file_action

max_log_file に設定した制限に達したときに実行するアクションを決定します。Audit ログファイルが上書きされないように **keep_logs** に設定する必要があります。

space_left

space_left_action パラメーターで設定されたアクションがトリガーされるディスクに残っている空き領域の量を指定します。管理者は、ディスクの領域を反映して解放するのに十分な時間を設定する必要があります。**space_left** の値は、Audit ログファイルが生成されるレートによって異なります。space_left の値が整数として指定されている場合は、メガバイト (MiB) 単位の絶対サイズとして解釈されます。値が 1 〜 99 の数値の後にパーセント記号を付けて指定されている場合 (5% など)、Audit デーモンは、**log_file** を含むファイルシステムのサイズに基づいて、メガバイト単位で絶対サイズを計算します。

space_left_action

適切な通知方法を使用して、**space_left_action** パラメーターを **email** または **exec** に設定することを推奨します。

admin_space_left

admin_space_left_action パラメーターで設定されたアクションがトリガーされる空きスペースの絶対最小量を指定します。これは、管理者によって実行されたアクションをログに記録するのに十分なスペースを残す値に設定する必要があります。このパラメーターの数値は、space_left の数値

より小さくする必要があります。また、数値にパーセント記号を追加 (1% など) して、Audit デーモンが、ディスクパーティションサイズに基づいて、数値を計算するようにすることもできます。

admin_space_left_action

single を、システムをシングルユーザーモードにし、管理者がディスク領域を解放できるようにします。

disk_full_action

Audit ログファイルが含まれるパーティションに空き領域がない場合に発生するアクションを指定します (**halt** または **single** に設定する必要があります)。これにより、Audit がイベントをログに記録できなくなると、システムは、シングルユーザーモードでシャットダウンまたは動作します。

disk_error_action

Audit ログファイルが含まれるパーティションでエラーが検出された場合に発生するアクションを指定します。このパラメーターは、ハードウェアの機能不全処理に関するローカルのセキュリティーポリシーに基づいて、**syslog**、**single**、**halt** のいずれかに設定する必要があります。

flush

incremental_async に設定する必要があります。これは **freq** パラメーターと組み合わせて機能します。これは、ハードドライブとのハード同期を強制する前にディスクに送信できるレコードの数を指定します。**freq** パラメーターは **100** に設定する必要があります。このパラメーターにより、アクティビティーが集中した際に高いパフォーマンスを保ちつつ、Audit イベントデータがディスクのログファイルと確実に同期されるようになります。

残りの設定オプションは、ローカルのセキュリティーポリシーに合わせて設定します。

38.4. AUDITD の開始および制御

auditd が設定されたら、サービスを起動して Audit 情報を収集し、ログファイルに保存します。root ユーザーで次のコマンドを実行し、**auditd** を起動します。

```
# service auditd start
```

システムの起動時に **auditd** が起動するように設定するには、次のコマンドを実行します。

```
# systemctl enable auditd
```

 # auditctl -e 0 で **auditd** を一時的に無効にし、**# auditctl -e 1** で再度有効にできます。

service auditd action コマンドを使用すると、**auditd** でさまざまなアクションを実行できます。ここでの **アクション** は以下のいずれかになります。

stop

auditd を停止します。

restart

auditd を再起動します。

reload または force-reload

/etc/audit/auditd.conf ファイルから **auditd** の設定を再ロードします。

rotate

/var/log/audit/ ディレクトリーのログファイルをローテーションします。

resume

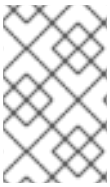
Audit イベントのログが一旦停止した後、再開します。たとえば、Audit ログファイルが含まれるディスクパーティションの未使用領域が不足している場合などです。

condrestart または try-restart

auditd がすでに起動している場合にのみ、これを再起動します。

status

auditd の稼働状況を表示します。



注記

service コマンドは、**auditd** デーモンと正しく相互作用する唯一の方法です。**auid** 値が適切に記録されるように、**service** コマンドを使用する必要があります。**systemctl** コマンドは、2 つのアクション (**enable** および **status**) にのみ使用できます。

38.5. AUDIT ログファイルについて

デフォルトでは、Audit システムはログエントリーを **/var/log/audit/audit.log** ファイルに保存します。ログローテーションが有効になっていれば、ローテーションされた **audit.log** ファイルは同じディレクトリーに保存されます。

下記の Audit ルールを追加して、**/etc/ssh/sshd_config** ファイルの読み取りまたは修正の試行をすべてログに記録します。

```
# auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

auditd デーモンが実行している場合は、たとえば次のコマンドを使用して、Audit ログファイルに新しいイベントを作成します。

```
$ cat /etc/ssh/sshd_config
```

このイベントは、**audit.log** ファイルでは以下のようになります。

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no exit=-13
a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538 auid=1000 uid=1000
gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1
comm="cat" exe="/bin/cat" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="sshd_config"
type=CWD msg=audit(1364481363.243:24287):  cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0 name="/etc/ssh/sshd_config" inode=409248
dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0
type=PROCTITLE msg=audit(1364481363.243:24287) :
proctitle=636174002F6574632F7373682F737368645F636F6E666967
```

上記のイベントは 4 つのレコードで構成されており、タイムスタンプとシリアル番号を共有します。レコードは、常に **type=** で始まります。各レコードには、スペースまたはコンマで区切られた名前と値のペア (**name=value**) が複数使用されています。上記のイベントの詳細な分析は以下のようになります。

1つ目のレコード

type=SYSCALL

type フィールドには、レコードのタイプが記載されます。この例の **SYSCALL** 値は、カーネルへのシステムコールによりこれが記録されたことを示しています。

msg=audit(1364481363.243:24287):

msg フィールドには以下が記録されます。

- **audit(time_stamp:ID)** 形式のレコードのタイムスタンプおよび一意の ID。複数のレコードが同じ Audit イベントの一部として生成されている場合は、同じタイムスタンプおよび ID を共有できます。タイムスタンプは Unix の時間形式です (1970 年 1 月 1 日 00:00:00 UTC からの秒数)。
- カーネル空間およびユーザー空間のアプリケーションが提供するさまざまなイベント固有の **name=value** ペア。

arch=c000003e

arch フィールドには、システムの CPU アーキテクチャーに関する情報が含まれます。**c000003e** の値は 16 進数表記で記録されます。**ausearch** コマンドで Audit レコードを検索する場合は、**-i** オプションまたは **--interpret** オプションを使用して、16 進数の値を人間が判読できる値に自動的に変換します。**c000003e** 値は **x86_64** として解釈されます。

syscall=2

syscall フィールドは、カーネルに送信されたシステムコールのタイプを記録します。値が **2** の場合は、**/usr/include/asm/unistd_64.h** ファイルに、人間が判読できる値を指定できます。この場合の **2** は、**オープン** なシステムコールです。**ausyscall** ユーティリティーでは、システムコール番号を、人間が判読できる値に変換できます。**ausyscall --dump** コマンドを使用して、システムコールのリストとその数字を表示します。詳細は、**ausyscall**(8) の man ページを参照してください。

success=no

success フィールドは、その特定のイベントで記録されたシステムコールが成功したかどうかを記録します。この例では、呼び出しが成功しませんでした。

exit=-13

exit フィールドには、システムコールが返した終了コードを指定する値が含まれます。この値は、システムコールにより異なります。次のコマンドを実行すると、この値を人間が判読可能なものに変換できます。

```
# ausearch --interpret --exit -13
```

この例では、監査ログに、終了コード **-13** で失敗したイベントが含まれていることが前提となります。

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

a0 から **a3** までのフィールドは、このイベントにおけるシステムコールの最初の 4 つの引数を、16 進法で記録します。この引数は、使用されるシステムコールにより異なります。**ausearch** ユーティリティーで解釈できます。

items=1

items フィールドには、システムコールのレコードに続く PATH 補助レコードの数が含まれます。

ppid=2686

ppid フィールドは、親プロセス ID (PPID) を記録します。この例では、 **2686** は、**bash** などの親プロセスの PPID です。

pid=3538

pid フィールドは、プロセス ID (PID) を記録します。この例の **3538** は **cat** プロセスの PID です。

auid=1000

auid フィールドには、loginuid である Audit ユーザー ID が記録されます。この ID は、ログイン時にユーザーに割り当てられ、ユーザーの ID が変更した後でもすべてのプロセスに引き継がれます (たとえば、**su - john** コマンドでユーザーアカウントを切り替えた場合)。

uid=1000

uid フィールドは、解析しているプロセスを開始したユーザーのユーザー ID を記録します。ユーザー ID は、**ausearch -i --uid UID** のコマンドを使用するとユーザー名に変換されます。

gid=1000

gid フィールドは、解析しているプロセスを開始したユーザーのグループ ID を記録します。

euid=1000

euid フィールドは、解析しているプロセスを開始したユーザーの実効ユーザー ID を記録します。

suid=1000

suid フィールドは、解析しているプロセスを開始したユーザーのセットユーザー ID を記録します。

fsuid=1000

fsuid フィールドは、解析しているプロセスを開始したユーザーのファイルシステムユーザー ID を記録します。

egid=1000

egid フィールドは、解析しているプロセスを開始したユーザーの実効グループ ID を記録します。

sgid=1000

sgid フィールドは、解析しているプロセスを開始したユーザーのセットグループ ID を記録します。

fsgid=1000

fsgid フィールドは、解析しているプロセスを開始したユーザーのファイルシステムグループ ID を記録します。

tty=pts0

tty フィールドは、解析しているプロセスが開始したターミナルを記録します。

ses=1

ses フィールドは、解析しているプロセスが開始したセッションのセッション ID を記録します。

comm="cat"

comm フィールドは、解析しているプロセスを開始するために使用したコマンドのコマンドライン名を記録します。この例では、この Audit イベントを発生するのに、**cat** コマンドが使用されました。

exe="/bin/cat"

exe フィールドは、解析しているプロセスを開始するために使用した実行可能ファイルへのパスを記録します。

subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

subj フィールドは、解析しているプロセスの実行時にラベル付けされた SELinux コンテンツを記録します。

key="sshd_config"

key フィールドは、Audit ログでこのイベントを生成したルールに関連付けられている管理者による定義の文字列を記録します。

2 つ目のレコード

type=CWD

2 つ目のレコードの **type** フィールドの値は、**CWD** (現在の作業ディレクトリー) です。このタイプは、最初のレコードで指定されたシステムコールを開始したプロセスの作業ディレクトリーを記録するために使用されます。
この記録の目的は、相対パスが関連する PATH 記録に保存された場合に、現行プロセスの位置を記録することにあります。これにより、絶対パスを再構築できます。

msg=audit(1364481363.243:24287)

msg フィールドは、最初のレコードと同じタイムスタンプと ID の値を保持します。タイムスタンプは Unix の時間形式です (1970 年 1 月 1 日 00:00:00 UTC からの秒数)。

cwd="/home/user_name"

cwd フィールドは、システムコールが開始したディレクトリーのパスになります。

3 つ目のレコード

type=PATH

3 つ目のレコードでは、**type** フィールドの値は **PATH** です。Audit イベントには、システムコールに引数として渡されたすべてのパスに **PATH** タイプのレコードが含まれます。この Audit イベントでは、1 つのパス (**/etc/ssh/sshd_config**) のみが引数として使用されます。

msg=audit(1364481363.243:24287):

msg フィールドは、1 つ目と 2 つ目のレコードと同じタイムスタンプと ID になります。

item=0

item フィールドは、**SYSCALL** タイプレコードで参照されているアイテムの合計数のうち、現在のレコードがどのアイテムであるかを示します。この数はゼロベースで、**0** は最初の項目であることを示します。

name="/etc/ssh/sshd_config"

name フィールドは、システムコールに引数として渡されたファイルまたはディレクトリーのパスを記録します。この場合、これは **/etc/ssh/sshd_config** ファイルです。

inode=409248

inode フィールドには、このイベントで記録されたファイルまたはディレクトリーに関連する inode 番号が含まれます。以下のコマンドは、inode 番号 **409248** に関連するファイルまたはディレクトリーを表示します。

```
# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

dev フィールドは、このイベントで記録されたファイルまたはディレクトリーを含むデバイスのマイナーおよびメジャーの ID を指定します。ここでは、値が **/dev/fd/0** デバイスを示しています。

mode=0100600

mode フィールドは、ファイルまたはディレクトリーのパーミッションを、**st_mode** フィールドの **stat** コマンドが返す数字表記で記録します。詳細は、**stat(2)** の man ページを参照してください。この場合、**0100600** は **-rw-------** として解釈されます。つまり、root ユーザーにのみ、**/etc/ssh/sshd_config** ファイルに読み取りおよび書き込みのパーミッションが付与されます。

ouid=0

ouid フィールドは、オブジェクトの所有者のユーザー ID を記録します。

ogid=0

ogid フィールドは、オブジェクトの所有者のグループ ID を記録します。

rdev=00:00

rdev フィールドには、特定ファイルにのみ記録されたデバイス識別子が含まれます。ここでは、記録されたファイルは通常のファイルであるため、このフィールドは使用されません。

obj=system_u:object_r:etc_t:s0

obj フィールドは、実行時に、記録されているファイルまたはディレクトリーにラベル付けする SELinux コンテキストを記録します。

nametype=NORMAL

nametype フィールドは、指定したシステムコールのコンテキストで各パスのレコード操作の目的を記録します。

cap_fp=none

cap_fp フィールドは、ファイルまたはディレクトリーオブジェクトで許可されたファイルシステムベースの機能の設定に関連するデータを記録します。

cap_fi=none

cap_fi フィールドは、ファイルまたはディレクトリーオブジェクトの継承されたファイルシステムベースの機能の設定に関するデータを記録します。

cap_fe=0

cap_fe フィールドは、ファイルまたはディレクトリーオブジェクトのファイルシステムベースの機能の有効ビットの設定を記録します。

cap_fver=0

cap_fver フィールドは、ファイルまたはディレクトリーオブジェクトのファイルシステムベースの機能のバージョンを記録します。

4 つ目のレコード

type=PROCTITLE

type フィールドには、レコードのタイプが記載されます。この例の **PROCTITLE** 値は、このレコードにより、カーネルへのシステムコールにより発生するこの監査イベントを発生させた完全なコマンドラインを提供することが指定されることを示しています。

proctitle=636174002F6574632F7373682F737368645F636F6E666967

proctitle フィールドは、解析しているプロセスを開始するために使用したコマンドのコマンドラインを記録します。このフィールドは 16 進数の表記で記録され、Audit ログパーサーに影響が及ばないようにします。このテキストは、この Audit イベントを開始したコマンドに復号します。**ausearch** コマンドで Audit レコードを検索する場合は、**-i** オプションまたは **--interpret** オプションを使用して、16 進数の値を人間が判読できる値に自動的に変換します。**636174002F6574632F7373682F737368645F636F6E666967** 値は、**cat /etc/ssh/sshd_config** として解釈されます。

38.6. AUDITCTL で AUDIT ルールを定義および実行

Audit システムは、ログファイルで取得するものを定義する一連のルールで動作します。Audit ルールは、**auditctl** ユーティリティーを使用してコマンドラインで設定するか、**/etc/audit/rules.d/** ディレクトリーで設定できます。

auditctl コマンドを使用すると、Audit システムの基本的な機能を制御し、どの Audit イベントをログに記録するかを指定するルールを定義できます。

ファイルシステムのルールの例

1. すべての書き込みアクセスと **/etc/passwd** ファイルのすべての属性変更をログに記録するルールを定義するには、次のコマンドを実行します。

```
# auditctl -w /etc/passwd -p wa -k passwd_changes
```

2. すべての書き込みアクセスと、**/etc/selinux/** ディレクトリー内の全ファイルへのアクセスと、その属性変更をすべてログに記録するルールを定義するには、次のコマンドを実行します。

```
# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

システムロールのルールの例

1. システムで 64 ビットアーキテクチャーが使用され、システムコールの **adjtimex** または **settimeofday** がプログラムにより使用されるたびにログエントリーを作成するルールを定義するには、次のコマンドを実行します。

```
# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

2. ユーザー ID が 1000 以上のシステムユーザーがファイルを削除したりファイル名を変更するたびに、ログエントリーを作成するルールを定義するには、次のコマンドを実行します。

```
# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000 -F
auid!=4294967295 -k delete
```

-F auid!=4294967295 オプションが、ログイン UID が設定されていないユーザーを除外するために使用されています。

実行可能なファイルルール

/bin/id プログラムのすべての実行をログに取得するルールを定義するには、次のコマンドを実行します。

```
# auditctl -a always,exit -F exe=/bin/id -F arch=b64 -S execve -k execution_bin_id
```

関連情報

- **auditctl(8)** man ページ

38.7. 永続的な AUDIT ルールの定義

再起動後も持続するように Audit ルールを定義するには、**/etc/audit/rules.d/audit.rules** ファイルに直接追加するか、**/etc/audit/rules.d/** ディレクトリーにあるルールを読み込む **augenrules** プログラムを使用する必要があります。

auditd サービスを開始すると、**/etc/audit/audit.rules** ファイルが生成されることに注意してください。**/etc/audit/rules.d/** のファイルは、同じ **auditctl** コマンドライン構文を使用してルールを指定します。ハッシュ記号 (#) に続く空の行とテキストは無視されます。

また、**auditctl** コマンドは、以下のように **-R** オプションを使用して指定したファイルからルールを読み込むのに使用することもできます。

```
# auditctl -R /usr/share/audit/sample-rules/30-stig.rules
```

38.8. 事前に設定されたルールファイルの使用

/usr/share/audit/sample-rules ディレクトリーには、**audit** パッケージが各種の証明書規格に従って、事前設定ルールのファイル一式が提供されています。

30-nispom.rules

NISPOM (National Industrial Security Program Operating Manual) の Information System Security の章で指定している要件を満たす Audit ルール設定

30-ospp-v42*.rules

OSPP (Protection Profile for General Purpose Operating Systems) プロファイルバージョン 4.2 に定義されている要件を満たす監査ルール設定

30-pci-dss-v31.rules

PCI DSS (Payment Card Industry Data Security Standard) v3.1 に設定されている要件を満たす監査ルール設定

30-stig.rules

セキュリティー技術実装ガイド (STIG: Security Technical Implementation Guide) で設定されている要件を満たす Audit ルール設定

上記の設定ファイルを使用するには、**/etc/audit/rules.d/** ディレクトリーにコピーして、以下のように **augenrules --load** コマンドを使用します。

```
# cd /usr/share/audit/sample-rules/
# cp 10-base-config.rules 30-stig.rules 31-privileged.rules 99-finalize.rules /etc/audit/rules.d/
# augenrules --load
```

番号指定スキームを使用して監査ルールを順序付けできます。詳細は、**/usr/share/audit/sample-rules/README-rules** ファイルを参照してください。

関連情報

- **audit.rules(7)** man ページ

38.9. 永続ルールを定義する AUGENRULES の使用

augenrules スクリプトは、**/etc/audit/rules.d/** ディレクトリーにあるルールを読み込み、**audit.rules** ファイルにコンパイルします。このスクリプトは、自然なソート順序の特定の順番で、**.rules** で終わるすべてのファイルを処理します。このディレクトリーのファイルは、以下の意味を持つグループに分類されます。

- 10 - カーネルおよび auditctl の設定
- 20 - 一般的なルールに該当してしまう可能性もあるが、ユーザー側で独自ルールを作成することも可能
- 30 - 主なルール
- 40 - 任意のルール
- 50 - サーバー固有のルール
- 70 - システムのローカルルール
- 90 - ファイナライズ (不変)

ルールは、すべてを一度に使用することは意図されていません。ルールは考慮すべきポリシーの一部であり、個々のファイルは **/etc/audit/rules.d/** にコピーされます。たとえば、STIG 設定でシステムを設定し、**10-base-config**、**30-stig**、**31-privileged**、**99-finalize** の各ルールをコピーします。

/etc/audit/rules.d/ ディレクトリーにルールを置いたら、**--load** ディレクティブで **augenrules** スクリプトを実行することでそれを読み込みます。

```
# augenrules --load
/sbin/augenrules: No change
No rules
enabled 1
failure 1
pid 742
rate_limit 0
...
```

関連情報

- **audit.rules(8)** および **augenrules(8)** の man ページ

38.10. AUGENRULES の無効化

augenrules ユーティリティーを無効にするには、以下の手順に従います。これにより、Audit が **/etc/audit/audit.rules** ファイルで定義されたルールを使用するように切り替えます。

手順

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

```
# cp -f /usr/lib/systemd/system/auditd.service /etc/systemd/system/
```

2. 任意のテキストエディターで **/etc/systemd/system/auditd.service** ファイルを編集します。以下に例を示します。

```
# vi /etc/systemd/system/auditd.service
```

3. **augenrules** を含む行をコメントアウトし、**auditctl -R** コマンドを含む行のコメント設定を解除します。

```
#ExecStartPost=-/sbin/augenrules --load
ExecStartPost=-/sbin/auditctl -R /etc/audit/audit.rules
```

4. **systemd** デーモンを再読み込みして、**auditd.service** ファイルの変更を取得します。

```
# systemctl daemon-reload
```

5. **auditd** サービスを再起動します。

```
# service auditd restart
```

関連情報

- **augenrules(8)** および **audit.rules(8)** の man ページ
- auditd service restart overrides changes made to /etc/audit/audit.rules .

38.11. ソフトウェアの更新を監視するための AUDIT の設定

RHEL 8.6 以降のバージョンでは、事前設定されたルール **44-installers.rules** を使用して、ソフトウェアをインストールする次のユーティリティーを監視するように Audit を設定できます。

- **dnf** [3]
- **yum**
- **pip**
- **npm**
- **cpan**
- **gem**
- **luarocks**

デフォルトでは、**rpm** はパッケージのインストールまたは更新時に監査 **SOFTWARE_UPDATE** イベントをすでに提供しています。これらをリスト表示するには、コマンドラインで **ausearch -m SOFTWARE_UPDATE** と入力します。

RHEL 8.5 以前のバージョンでは、手動でルールを追加して、**/etc/audit/rules.d/** ディレクトリーの **.rules** ファイルにソフトウェアをインストールするユーティリティーを監視できます。

注記

事前設定されたルールファイルは、**ppc64le** および **aarch64** アーキテクチャーを備えたシステムでは使用できません。

前提条件

- **auditd** が、環境を保護するための auditd の設定で提供される設定に従って定義されている。

手順

1. RHEL 8.6 以降では、事前設定されたルールファイル **44-installers.rules** を **/usr/share/audit/sample-rules/** ディレクトリーから **/etc/audit/rules.d/** ディレクトリーにコピーします。

```
# cp /usr/share/audit/sample-rules/44-installers.rules /etc/audit/rules.d/
```

   RHEL 8.5 以前では、**/etc/audit/rules.d/** ディレクトリーに、**44-installers.rules** という名前の新規ファイルを作成し、以下のルールを挿入します。

```
-a always,exit -F perm=x -F path=/usr/bin/dnf-3 -F key=software-installer
-a always,exit -F perm=x -F path=/usr/bin/yum -F
```

   同じ構文を使用して、**pip** や **npm** などのソフトウェアをインストールする他のユーティリティー用に、さらにルールを追加できます。

2. 監査ルールを読み込みます。

```
# augenrules --load
```

検証

検証

1. 読み込まれたルールをリスト表示します。

```
# auditctl -l
-p x-w /usr/bin/dnf-3 -k software-installer
-p x-w /usr/bin/yum -k software-installer
-p x-w /usr/bin/pip -k software-installer
-p x-w /usr/bin/npm -k software-installer
-p x-w /usr/bin/cpan -k software-installer
-p x-w /usr/bin/gem -k software-installer
-p x-w /usr/bin/luarocks -k software-installer
```

2. インストールを実行します。以下に例を示します

```
# yum reinstall -y vim-enhanced
```

3. Audit ログで最近のインストールイベントを検索します。次に例を示します。

```
# ausearch -ts recent -k software-installer
----
time->Thu Dec 16 10:33:46 2021
type=PROCTITLE msg=audit(1639668826.074:298):
proctitle=2F7573722F6C6962657865632F706C6174666F726D2D707974686F6E002F75737
22F62696E2F646E66007265696E7374616C6C002D790076696D2D656E68616E636564
type=PATH msg=audit(1639668826.074:298): item=2 name="/lib64/ld-linux-x86-64.so.2"
inode=10092 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=1 name="/usr/libexec/platform-python"
inode=4618433 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=0 name="/usr/bin/dnf" inode=6886099
dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:rpm_exec_t:s0
nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1639668826.074:298): cwd="/root"
type=EXECVE msg=audit(1639668826.074:298): argc=5 a0="/usr/libexec/platform-python"
a1="/usr/bin/dnf" a2="reinstall" a3="-y" a4="vim-enhanced"
type=SYSCALL msg=audit(1639668826.074:298): arch=c000003e syscall=59 success=yes
exit=0 a0=55c437f22b20 a1=55c437f2c9d0 a2=55c437f2aeb0 a3=8 items=3 ppid=5256
pid=5375 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=3
comm="dnf" exe="/usr/libexec/platform-python3.6"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="software-installer"
```

38.12. AUDIT によるユーザーログイン時刻の監視

特定の時刻にログインしたユーザーを監視するために、特別な方法で Audit を設定する必要はありません。同じ情報を表示する異なる方法を提供する **ausearch** または **aureport** ツールを使用できます。

前提条件

- **auditd** が、環境を保護するための auditd の設定で提供される設定に従って定義されている。

手順

ユーザーのログイン時刻を表示するには、次のいずれかのコマンドを使用します。

- 監査ログで **USER_LOGIN** メッセージタイプを検索します。

```
# ausearch -m USER_LOGIN -ts '12/02/2020' '18:00:00' -sv no
time->Mon Nov 22 07:33:22 2021
type=USER_LOGIN msg=audit(1637584402.416:92): pid=1939 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login acct="
(unknown)" exe="/usr/sbin/sshd" hostname=? addr=10.37.128.108 terminal=ssh res=failed'
```

  - **-ts** オプションを使用して日付と時刻を指定できます。このオプションを使用しない場合、**ausearch** は今日の結果を提供し、時刻を省略すると、 **ausearch** は午前 0 時からの結果を提供します。

  - 成功したログイン試行を除外するには **-sv yes** オプションを、失敗したログイン試行を除外するには **-sv no** を、それぞれ使用することができます。

- **ausearch** コマンドの生の出力を **aulast** ユーティリティーにパイプで渡します。このユーティリティーは、**last** コマンドの出力と同様の形式で出力を表示します。以下に例を示します。

```
# ausearch --raw | aulast --stdin
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.22.16.106     Mon Nov 22 07:40 - 07:40  (00:00)
reboot   system boot  4.18.0-348.6.el8 Mon Nov 22 07:33
```

- **--login -i** オプションを指定して **aureport** コマンドを使用し、ログインイベントのリストを表示します。

```
# aureport --login -i

Login Report
============================================
# date time auid host term exe success event
============================================
1. 11/16/2021 13:11:30 root 10.40.192.190 ssh /usr/sbin/sshd yes 6920
2. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6925
3. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6930
4. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6935
5. 11/16/2021 13:11:33 root 10.40.192.190 ssh /usr/sbin/sshd yes 6940
6. 11/16/2021 13:11:33 root 10.40.192.190 /dev/pts/0 /usr/sbin/sshd yes 6945
```

関連情報

- **ausearch(8)** の man ページ。
- **aulast(8)** の man ページ。
- **aureport(8)** の man ページ。

38.13. 関連情報

- ナレッジベースアーティクルの RHEL Audit System Reference
- ナレッジベースアーティクルの Auditd execution options in a container

- Linux Audit ドキュメントのプロジェクトページ
- **audit** パッケージが提供するドキュメントは、**/usr/share/doc/audit/** ディレクトリーにあります。
- **auditd(8)**、**auditctl(8)**、**ausearch(8)**、**audit.rules(7)**、**audispd.conf(5)**、**audispd(8)**、**auditd.conf(5)**、**ausearch-expression(5)**、**aulast(8)**、**aulastlog(8)**、**aureport(8)**、**ausyscall(8)**、**autrace(8)**、および **auvirt(8)** の man ページ。

---

[3] **dnf** は RHEL ではシンボリックリンクであるため、**dnf** Audit ルールのパスにはシンボリックリンクのターゲットが含まれている必要があります。正しい Audit イベントを受信するには、**path=/usr/bin/dnf** パスを **/usr/bin/dnf-3** に変更して、**44-installers.rules** ファイルを変更します。

パート VI. カーネルの設計

第39章 LINUX カーネル

Linux カーネルと、Red Hat が提供および管理する Linux カーネル RPM パッケージ (Red Hat カーネル) について学びます。Red Hat カーネルを最新の状態に保ちます。これにより、オペレーティングシステムに最新のバグ修正、パフォーマンス強化、およびパッチがすべて適用され、新しいハードウェアとの互換性が保たれます。

39.1. カーネルとは

カーネルは Linux オペレーティングシステムのコア部分で、システムリソースを管理し、ハードウェアアプリケーションおよびソフトウェアアプリケーション間のインターフェイスを確立します。Red Hat カーネルは、アップストリームの Linux メインラインカーネルをベースにしたカスタムカーネルです。Red Hat のエンジニアは、安定性と、最新のテクノロジーおよびハードウェアとの互換性に重点を置き、さらなる開発と強化を行っています。

Red Hat が新しいカーネルバージョンをリリースする前に、カーネルは厳格な品質保証テストをクリアしなければなりません。

Red Hat カーネルは RPM 形式でパッケージ化されているため、**yum** パッケージマネージャーにより簡単にアップグレードおよび検証できます。

警告

Red Hat によってコンパイルされていないカーネルは、Red Hat ではサポートされ**ていません**。

39.2. RPM パッケージ

RPM パッケージは、他のファイルとそのメタデータ (システムが必要とするファイルに関する情報) を含むファイルです。

特に、RPM パッケージは **cpio** アーカイブで設定されています。

cpio アーカイブには以下が含まれます。

- ファイル
- RPM ヘッダー (パッケージのメタデータ)
  rpm パッケージマネージャーはこのメタデータを使用して依存関係、ファイルのインストール先、およびその他の情報を決定します。

RPM パッケージの種類
RPM パッケージには 2 つの種類があります。いずれも、同じファイル形式とツールを使用しますが、コンテンツが異なるため、目的が異なります。

- ソース RPM (SRPM)
  SRPM には、ソースコードと SPEC ファイルが含まれます。これには、ソースコードをバイナリー RPM にビルドする方法が書かれています。必要に応じて、ソースコードへのパッチも含まれます。

- バイナリー RPM
  バイナリー RPM には、ソースおよびパッチから構築されたバイナリーが含まれます。

39.3. LINUX カーネル RPM パッケージの概要

カーネル RPM は、ファイルを含まないメタパッケージで、以下の必須サブパッケージが正しくインストールされるようにします。

- **kernel-core** - コア機能を確保するために、カーネルのバイナリーイメージ、システムを起動するためのすべての **initramfs** 関連オブジェクト、および最小限のカーネルモジュールが含まれます。このサブパッケージ単体は、仮想環境およびクラウド環境で使用して、Red Hat Enterprise Linux 8 カーネルのブート時間を短縮し、ディスクサイズを抑えます。

- **kernel-modules** - **kernel-core** にない残りのカーネルモジュールが含まれます。

上記の **kernel** サブパッケージをいくつか用意することで、特に仮想環境やクラウド環境でのシステム管理者へのメンテナンス面を低減させることを目指します。

任意のカーネルパッケージは、以下の例のようになります。

- **kernel-modules-extra** - まれなハードウェア用のカーネルモジュールと、読み込みがデフォルトで無効になっているモジュールが含まれます。

- **kernel-debug** – カーネル診断ができるように複数のデバッグオプションが有効になっているカーネルが含まれます。 デバッグオプションが有効になっているとパフォーマンスが低下します。

- **kernel-tools** – Linux カーネル操作のツールとサポートドキュメントが含まれています。

- **kernel-devel** – **kernel** パッケージに対して、モジュールを構築するのに十分なカーネルヘッダーと makefiles を含んでいます。

- **kernel-abi-stablelists** – RHEL カーネル ABI に関連する情報が含まれています。これには、強化を支援するための外部 Linux カーネルモジュールおよび **yum** プラグインで必要なカーネルシンボルのリストが含まれます。

- **kernel-headers** – Linux カーネルと、ユーザー空間ライブラリーおよびプログラムとの間のインターフェイスを指定する C ヘッダーファイルが含まれます。ヘッダーファイルは、ほとんどの標準プログラムを構築するのに必要な構造と定数を定義します。

関連情報

- **kernel-core パッケージ、kernel-modules パッケージ、および kernel-modules-extras パッケージは何ですか？**

39.4. カーネルパッケージの内容の表示

rpm コマンドを使用してインストールせずに、カーネルパッケージとそのサブパッケージの内容を表示します。

前提条件

- 使用している CPU アーキテクチャー用の **kernel**、**kernel-core**、**kernel-modules**、**kernel-modules-extra** RPM パッケージを取得していること

手順

手順

- **kernel** 用のモジュールをリスト表示します。

```
$ rpm -qlp <kernel_rpm>
(contains no files)
…
```

- **kernel-core** のモジュールをリスト表示します。

```
$ rpm -qlp <kernel-core_rpm>
…
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/udf/udf.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/xfs
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/xfs/xfs.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel/trace
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel/trace/ring_buffer_benchmark.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/lib
/lib/modules/4.18.0-80.el8.x86_64/kernel/lib/cordic.ko.xz
…
```

- **kernel-modules** のモジュールをリスト表示します。

```
$ rpm -qlp <kernel-modules_rpm>
…
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx4/mlx4_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx5/mlx5_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/qedr/qedr.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/usnic/usnic_verbs.ko.xz
/lib/modules/4.18.0-
80.el8.x86_64/kernel/drivers/infiniband/hw/vmw_pvrdma/vmw_pvrdma.ko.xz
…
```

- **kernel-modules-extra** のモジュールをリスト表示します。

```
$ rpm -qlp <kernel-modules-extra_rpm>
…
/lib/modules/4.18.0-80.el8.x86_64/extra/net/sched/sch_cbq.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/extra/net/sched/sch_choke.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/extra/net/sched/sch_drr.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/extra/net/sched/sch_dsmark.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/extra/net/sched/sch_gred.ko.xz
…
```

関連情報

- **rpm(8)** man ページ
- RPM パッケージ

39.5. カーネルの更新

yum パッケージマネージャーを使用してカーネルを更新します。

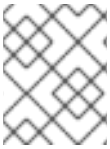
手順

1. カーネルを更新するには、次のコマンドを入力します。

```
# yum update kernel
```

このコマンドは、カーネルと、利用可能な最新バージョンへのすべての依存関係を更新します。

2. システムを再起動して、変更を有効にします。



注記

RHEL 7 から RHEL 8 にアップグレードする場合は、**RHEL 7 から RHEL 8 へのアップグレード** ドキュメントの関連のセクションを参照してください。

関連情報

- ソフトウェアパッケージの管理

39.6. 特定のカーネルバージョンのインストール

yum パッケージマネージャーを使用して新しいカーネルをインストールします。

手順

- 特定のカーネルバージョン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 kernel-{version}
```

関連情報

- Red Hat Code Browser
- Red Hat Enterprise Linux Release Dates

第40章 カーネルコマンドラインパラメーターの設定

カーネルコマンドラインパラメーターは、システムの起動時に Red Hat Enterprise Linux カーネルの特定の側面の動作を変更する手段のひとつです。システム管理者は、システムの起動時に設定されるオプションを完全に制御できます。特定のカーネルの動作はシステムの起動時にのみ設定できるため、このような変更を行う方法を理解することが管理スキルの鍵となります。

重要

カーネルコマンドラインパラメーターを変更することで、システムの動作が変更するオプションは、システムに悪影響を及ぼす可能性があります。したがって、実稼働環境にデプロイする前に変更をテストする必要があります。詳細なガイダンスは、Red Hat サポートまでご連絡ください。

40.1. カーネルコマンドラインパラメーターの概要

カーネルコマンドラインパラメーターは、以下のシステムの起動時間設定に使用されます。

- Red Hat Enterprise Linux カーネル
- 初期 RAM ディスク
- ユーザー領域機能

カーネルの起動時間パラメーターは、デフォルト値を上書きしたり、特定のハードウェア設定にするために使用されます。

デフォルトでは、GRUB ブートローダーを使用するシステムのカーネルコマンドラインパラメーターは、カーネルブートエントリーごとに **/boot/grub2/grubenv** ファイルの **kernelopts** 変数で定義されます。

注記

IBM Z では、**zipl** ブートローダーは環境変数に対応していないため、カーネルコマンドラインパラメーターはブートエントリー設定ファイルに保存されます。したがって、**kernelopts** 環境変数は使用できません。

関連情報

- **kernel-command-line(7)**、**bootparam(7)**、および **dracut.cmdline (7)** の man ページ
- How to install and boot custom kernels in Red Hat Enterprise Linux 8

40.2. GRUBBY とは

grubby は、ブートローダーの設定ファイルを操作するためのユーティリティーです。

grubby を使用して、デフォルトのブートエントリーを変更したり、GRUB2 メニューエントリーから引数を追加または削除したりすることもできます。

関連情報

- **grubby(8)** の man ページ

40.3. ブートエントリーとは

ブートエントリーは設定ファイルに格納され、特定のカーネルバージョンに関連付けられるオプションの集合です。実際には、ブートエントリーは、システムにカーネルがインストールされているのと同じ数だけあります。ブートエントリーの設定ファイルは、**/boot/loader/entries/** ディレクトリーにあり、以下のようになります。

```
6f9cc9cb7d7845d49698c9537337cedc-4.18.0-5.el8.x86_64.conf
```

上記のファイル名は、**/etc/machine-id** ファイルに保存されているマシン ID と、カーネルバージョンから設定されます。

ブートエントリーの設定ファイルには、カーネルバージョン、初期 ramdisk イメージ、および **kernelopts** 環境変数 (カーネルコマンドラインパラメーターを含む) に関する情報が含まれます。ブートエントリー設定例の内容は、以下のようになります。

```
title Red Hat Enterprise Linux (4.18.0-74.el8.x86_64) 8.0 (Ootpa)
version 4.18.0-74.el8.x86_64
linux /vmlinuz-4.18.0-74.el8.x86_64
initrd /initramfs-4.18.0-74.el8.x86_64.img $tuned_initrd
options $kernelopts $tuned_params
id rhel-20190227183418-4.18.0-74.el8.x86_64
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

kernelopts 環境変数は **/boot/grub2/grubenv** ファイルで定義されます。

関連情報

- How to install and boot custom kernels in Red Hat Enterprise Linux 8

40.4. すべてのブートエントリーでカーネルコマンドラインパラメーターの変更

システム上のすべてのブートエントリーのカーネルコマンドラインパラメーターを変更します。

前提条件

- **gruby** ユーティリティーがシステムにインストールされていることを確認してください。
- **zipl** ユーティリティーが IBM Z システムにインストールされていることを確認してください。

手順

- パラメーターを追加するには、以下を行います。

  ```
  # grubby --update-kernel=ALL --args="<NEW_PARAMETER>"
  ```

  GRUB ブートローダーを使用するシステムの場合は、**/boot/grub2/grubenv** の **kernelopts** 変数に新しいカーネルパラメーターを追加して、ファイルを更新します。

  - IBM Z で、起動メニューを更新するオプションを指定せずに **zipl** コマンドを実行します。

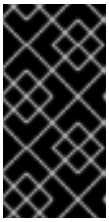
- パラメーターを削除するには、次のコマンドを実行します。

```
# grubby --update-kernel=ALL --remove-args="<PARAMETER_TO_REMOVE>"
```

  - IBM Z で、起動メニューを更新するオプションを指定せずに **zipl** コマンドを実行します。

- カーネルパッケージの更新ごとに、設定したカーネルオプションを新しいカーネルに反映させます。

```
# grub2-mkconfig -o /etc/grub2.cfg
```



重要

新しくインストールされたカーネルは、以前に設定されたカーネルからカーネルコマンドラインパラメーターを継承しません。新しくインストールされたカーネルで **grub2-mkconfig** コマンドを実行して、必要なパラメーターを新しいカーネルに反映させる必要があります。

関連情報

- カーネルコマンドラインパラメーターの概要
- **grubby(8)** および **zipl(8)** の man ページ
- grubby ツール

40.5. 1 つのブートエントリーでカーネルコマンドラインパラメーターの変更

システム上の単一のブートエントリーのカーネルコマンドラインパラメーターを変更します。

前提条件

- **grubby** ユーティリティーおよび **zipl** ユーティリティーがシステムにインストールされている。

手順

- パラメーターを追加するには、以下を行います。

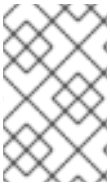
```
#  grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="<NEW_PARAMETER>"
```

  - IBM Z で、起動メニューを更新するオプションを指定せずに **zipl** コマンドを実行します。

- パラメーターを削除するには、以下のコマンドを使用します。

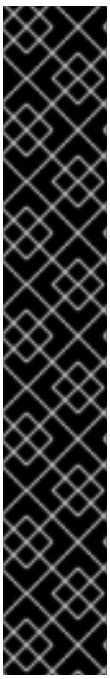
```
#  grubby --update-kernel=/boot/vmlinuz-$(uname -r) --remove-args="
<PARAMETER_TO_REMOVE>"
```

  - IBM Z で、起動メニューを更新するオプションを指定せずに **zipl** コマンドを実行します。



注記

grub.cfg ファイルを使用するシステムでは、デフォルトで、カーネルブートエントリーごとに **options** パラメーターがあり、これは **kernelopts** 変数に設定されます。この変数は、**/boot/grub2/grubenv** 設定ファイルで定義されます。



重要

GRUB2 システムの場合:

- すべてのブートエントリーに対してカーネルコマンドラインパラメーターを変更する場合には、**grubby** ユーティリティーは **/boot/grub2/grubenv** ファイルの **kernelopts** 変数を更新します。
- 1 つのブートエントリーのカーネルコマンドラインパラメーターが変更されると、**kernelopts** 変数の拡張やカーネルパラメーターの変更が行われ、得られた値は各ブートエントリーの **/boot/loader/entries/<RELEVANT_KERNEL_BOOT_ENTRY.conf>** ファイルに保存されます。

zIPL システムの場合:

- **grubby** は、個別のカーネルブートエントリーのカーネルコマンドラインパラメーターを変更して、**/boot/loader/entries/<ENTRY>.conf** ファイルに保存します。

関連情報

- カーネルコマンドラインパラメーターの概要
- **grubby(8)** および **zipl(8)** の man ページ
- **grubby ツール**

40.6. 起動時の一時的なカーネルコマンドラインパラメーターの変更

1 回の起動プロセス中にのみカーネルパラメーターを変更することで、カーネルメニューエントリーを一時的に変更します。

手順

1. GRUB 2 ブートメニューが表示されたら起動するカーネルを選択し、**e** キーを押してカーネルパラメーターを編集します。
2. カーソルを下に移動してカーネルコマンドラインを見つけます。カーネルコマンドラインは、64 ビット IBM Power シリーズおよび x86-64 BIOS ベースのシステムの場合は **linux** で始まり、UEFI システムの場合は **linuxefi** で始まります。
3. カーソルを行の最後に移動します。

注記

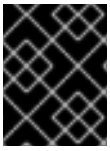
行の最初に移動するには **Ctrl+a** を押します。行の最後に移動するには **Ctrl+e** を押します。システムによっては、**Home** キーおよび **End** キーも機能する場合があります。

4. 必要に応じてカーネルパラメーターを編集します。たとえば、緊急モードでシステムを実行するには、**linux** 行の最後に **emergency** パラメーターを追加します。

```
linux   ($root)/vmlinuz-4.18.0-348.12.2.el8_5.x86_64 root=/dev/mapper/rhel-root ro
crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap
rhgb quiet emergency
```

システムメッセージを有効にするには、**rhgb** および **quiet** パラメーターを削除します。

5. **Ctrl**+**x** を押して、選択したカーネルと変更したコマンドラインパラメーターで起動します。



重要

Esc キーを押すと、コマンドラインの編集を終了し、ユーザーの加えた変更をすべて破棄します。

注記

この手順は単一ブートにのみ適用され、変更は永続的に行われません。

40.7. シリアルコンソール接続を有効にする GRUB 設定

シリアルコンソールは、ネットワークがダウンしている場合にヘッドレスサーバーまたは埋め込みシステムに接続する際に便利です。あるいは、セキュリティールールを回避し、別のシステムへのログインアクセスを取得する必要がある場合などです。

シリアルコンソール接続を使用するように、デフォルトの GRUB 設定の一部を設定する必要があります。

前提条件

- root 権限がある。

手順

1. **/etc/default/grub** ファイルに以下の 2 つの行を追加します。

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"
```

最初の行は、グラフィカルターミナルを無効にします。**GRUB_TERMINAL** キーは、**GRUB_TERMINAL_INPUT** および **GRUB_TERMINAL_OUTPUT** の値を上書きします。

2 行目は、ボーレート (**--speed**)、パリティー、および他の値を使用中の環境とハードウェアに適合するように調整します。以下のログファイルのようなタスクには、115200 のように非常に高いボーレートが推奨されます。

2. GRUB 設定ファイルを更新します。

- BIOS ベースのマシンの場合:

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

- UEFI ベースのマシンの場合:

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

3. システムを再起動して、変更を有効にします。

第41章 ランタイム時のカーネルパラメーターの設定

システム管理者は、ランタイム時に Red Hat Enterprise Linux カーネルの動作を多数変更できます。**sysctl** コマンドを使用し、**/etc/sysctl.d/** および **/proc/sys/** ディレクトリー内の設定ファイルを変更して、実行時にカーネルパラメーターを設定します。

41.1. カーネルパラメーターとは

カーネルパラメーターは、システムの実行中に調整できる調整可能な値です。変更を有効にする場合でも、カーネルの再起動や再コンパイルは不要です。

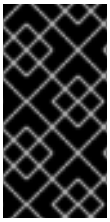
以下を使用してカーネルパラメーターに対応できます。

- **sysctl** コマンド
- **/proc/sys/** ディレクトリーにマウントされている仮想ファイルシステム
- **/etc/sysctl.d/** ディレクトリー内の設定ファイル

調整可能パラメーターは、カーネルサブシステムでクラスに分割されます。Red Hat Enterprise Linux には、以下の調整可能のクラスがあります。

表41.1 sysctl クラスの表

調整パラメーターのクラス	サブシステム
abi	実行ドメインおよびパーソナリティー
crypto	暗号化インターフェイス
debug	カーネルのデバッグインターフェイス
dev	デバイス固有の情報
fs	グローバルおよび固有の調整可能なファイルシステム
kernel	グローバルなカーネルの設定項目
net	ネットワークの設定項目
sunrpc	Sun Remote Procedure Call (NFS)
user	ユーザー名前空間の制限
vm	メモリー、バッファー、およびキャッシュのチューニングと管理



重要

プロダクションシステムでカーネルパラメーターを設定するには、慎重なプランニングが必要です。プランニングが欠如した変更では、カーネルが不安定になり、システムの再起動が必要とることがあります。カーネル値を変更する前に、有効なオプションを使用していることを確認してください。

関連情報

- **sysctl(8)** および **sysctl.d(5)** の man ページ

41.2. SYSCTL でカーネルパラメーターの一時的な設定

sysctl コマンドを使用して、実行時に一時的にカーネルパラメーターを設定します。このコマンドは、調整可能パラメーターのリスト表示およびフィルタリングにも便利です。

前提条件

- root 権限

手順

1. すべてのパラメーターとその値をリストします。

```
# sysctl -a
```

注記

 # sysctl -a コマンドは、ランタイム時およびシステムの起動時に調整できるカーネルパラメーターを表示します。

2. パラメーターを一時的に設定するには、次のように入力します。

```
# sysctl <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
```

上記のサンプルコマンドは、システムの実行中にパラメーター値を変更します。この変更は、再起動なしですぐに適用されます。

注記

変更は、システムの再起動後にデフォルトに戻ります。

関連情報

- **sysctl(8)** の man ページ
- sysctl でカーネルパラメーターを永続的に設定
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

41.3. SYSCTL でカーネルパラメーターを永続的に設定

sysctl コマンドを使用して、カーネルパラメーターを永続的に設定します。

前提条件

- root 権限

手順

1. すべてのパラメーターをリストします。

```
# sysctl -a
```

このコマンドは、ランタイム時に設定できるカーネルパラメーターをすべて表示します。

2. パラメーターを永続的に設定すします。

```
# sysctl -w <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE> >> /etc/sysctl.conf
```

サンプルコマンドは、調整可能な値を変更して、**/etc/sysctl.conf** ファイルに書き込みます。これにより、カーネルパラメーターのデフォルト値が上書きされます。変更は、再起動なしで即座に永続的に反映されます。

注記

カーネルパラメーターを永続的に変更するには、**/etc/sysctl.d/** ディレクトリーの設定ファイルに手動で変更を行ってください。

関連情報

- **sysctl(8)** および **sysctl.conf(5)** の man ページ
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

41.4. /ETC/SYSCTL.D/ の設定ファイルでカーネルパラメーターの調整

/etc/sysctl.d/ ディレクトリーの設定ファイルを手動で変更して、カーネルパラメーターを永続的に設定します。

前提条件

- root 権限

手順

1. **/etc/sysctl.d/** に新しい設定ファイルを作成します。

```
# vim /etc/sysctl.d/<some_file.conf>
```

2. カーネルパラメーターを 1 行に 1 つずつ含めます。

```
<TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
<TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
```

3. 設定ファイルを作成します。
4. マシンを再起動して、変更を有効にします。
   - また、再起動せずに変更を適用するには、以下を実行します。

     ```
     # sysctl -p /etc/sysctl.d/<some_file.conf>
     ```

     このコマンドにより、以前に作成した設定ファイルから値を読み取ることができます。

関連情報

- **sysctl(8)**、**sysctl.d(5)** の man ページ

41.5. /PROC/SYS/ でカーネルパラメーターの一時的な設定

/proc/sys/ 仮想ファイルシステムディレクトリー内のファイルを使用して、一時的にカーネルパラメーターを設定します。

前提条件

- root 権限

手順

1. 設定するカーネルパラメーターを特定します。

   ```
   # ls -l /proc/sys/<TUNABLE_CLASS>/
   ```

   コマンドが返した書き込み可能なファイルは、カーネルの設定に使用できます。読み取り専用権限を持つユーザーは、現在の設定についてフィードバックを提供します。

2. カーネルパラメーターにターゲットの値を割り当てます。

   ```
   # echo <TARGET_VALUE> > /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
   ```

   このコマンドでは、システムが再起動すると設定変更が消えます。

3. 必要に応じて、新しく設定した設定したカーネルパラメーターの値を確認します。

   ```
   # cat /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
   ```

関連情報

- sysctl でカーネルパラメーターを永続的に設定
- /etc/sysctl.d/ の設定ファイルでカーネルパラメーターの調整

第42章 KDUMP のインストールと設定

42.1. KDUMP のインストール

Red Hat Enterprise Linux の新規インストールでは、デフォルトで **kdump** サービスがインストールされ有効になっています。**kdump** について、およびデフォルトで有効になっていない場合に **kdump** をインストールする方法を説明します。

42.1.1. kdump とは

kdump は、クラッシュダンプのメカニズムを提供するサービスです。このサービスを使用すると、システムのメモリーの内容を分析するために保存できます。**kdump** は **kexec** システムコールを使用して再起動せずに別のカーネル (**capture kernel**) で起動し、クラッシュしたカーネルメモリーの内容 (**crash dump** または **vmcore**) をキャプチャーして保存します。この第 2 のカーネルは、システムメモリーの予約部分に収納されています。

重要

カーネルクラッシュダンプは、システム障害 (重大なバグ) 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、システム管理者が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が実装されてい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。

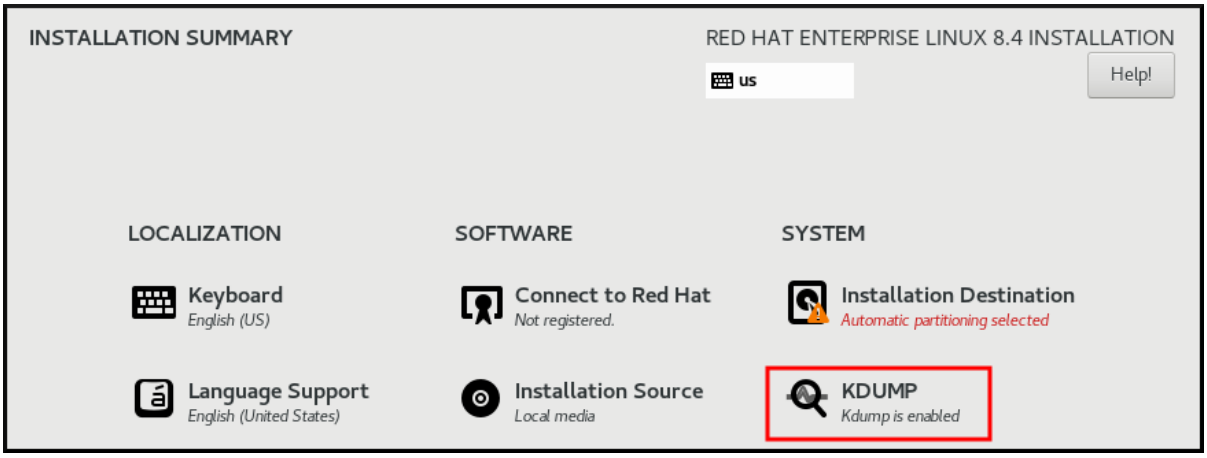
kdump をインストールすると、デフォルトの **/etc/kdump.conf** が作成されます。このファイルには、デフォルトの最小限の **kdump** 設定が含まれます。このファイルを編集して、**kdump** 設定をカスタマイズできますが、必須ではありません。

42.1.2. Anaconda を使用した kdump のインストール

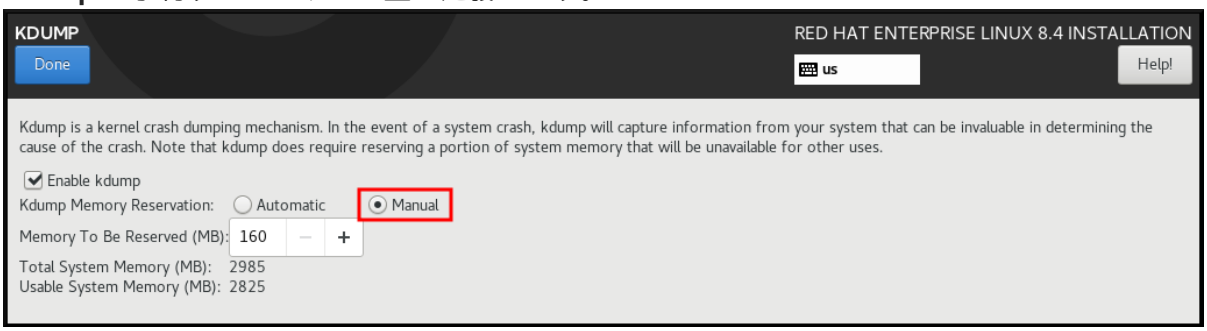
Anaconda インストーラーでは、対話式インストール時に **kdump** 設定用のグラフィカルインターフェイス画面が表示されます。インストーラー画面のタイトルは KDUMP で、メインの **インストールの概要** 画面から利用できます。**kdump** を有効にして、必要な量のメモリーを予約できます。

手順

1. **Kdump** 項目に移動します。
2. **kdump** が有効になっていない場合は有効にします。



3. **kdump** に予約するメモリーの量を定義します。



42.1.3. コマンドラインで kdump のインストール

カスタムの Kickstart インストールなどの一部のインストールオプションでは、デフォルトで **kdump** がインストールまたは有効化**されない**場合があります。この場合は、以下の手順を行ってください。

前提条件

- アクティブな RHEL サブスクリプション
- **kexec-tools** パッケージ
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **kdump** がシステムにインストールされているかどうかを確認します。

```
# rpm -q kexec-tools
```

このパッケージがインストールされている場合は以下を出力します。

```
kexec-tools-2.0.17-11.el8.x86_64
```

このパッケージがインストールされていない場合は以下を出力します

```
package kexec-tools is not installed
```

2. **kdump** および必要なパッケージをインストールします。

```
# dnf install kexec-tools
```

重要

kernel-3.10.0-693.el7 以降、**kdump** では **Intel IOMMU** ドライバーがサポートされています。以前のバージョン (kernel-3.10.0-514[.XYZ].el7 以前) では、**Intel IOMMU** のサポートを無効にすることが推奨されています。無効にしないと、キャプチャーカーネルが応答しなくなる可能性が高くなります。

42.2. コマンドラインで KDUMP の設定

kdump 環境を計画して構築します。

42.2.1. kdump サイズの見積もり

kdump 環境の計画および構築を行う際に、クラッシュダンプファイルに必要な領域を把握しておくことが重要です。

makedumpfile --mem-usage コマンドは、クラッシュダンプファイルに必要な領域を推定し、メモリー使用量に関するレポートを生成します。このレポートは、ダンプレベルと、除外して問題ないページを判断するのに役立ちます。

手順

- 次のコマンドを実行して、メモリー使用量に関するレポートを生成します。

```
# makedumpfile --mem-usage /proc/kcore


TYPE        PAGES    EXCLUDABLE    DESCRIPTION
----------------------------------------------------------------
ZERO          501635      yes        Pages filled with zero
CACHE         51657       yes        Cache pages
CACHE_PRIVATE 5442        yes        Cache pages + private
USER          16301       yes        User process pages
FREE          77738211    yes        Free pages
KERN_DATA     1333192     no         Dumpable kernel data
```

重要

makedumpfile --mem-usage は、必要なメモリーをページ単位で報告します。つまり、カーネルページサイズを元に、使用するメモリーのサイズを計算する必要があります。

42.2.2. メモリー使用量の設定

kdump のメモリー予約は、システムの起動中に行われます。メモリーサイズは、システムの GRUB (Grand Unified Bootloader) 設定で設定されます。メモリーサイズは、設定ファイルで指定された **crashkernel=** オプションの値と、システムの物理メモリーのサイズにより異なります。

crashkernel= オプションはさまざまな方法で定義できます。**crashkernel=** 値を指定するか、**auto** オプションを設定できます。**crashkernel=auto** パラメーターは、システムの物理メモリーの合計量に基づいて、メモリーを自動的に予約します。これを設定すると、カーネルは、キャプチャーカーネルに必要

な適切な量のメモリーを自動的に予約します。これにより、OOM (Out-of-Memory) エラーの回避に役立ちます。

注記

kdump の自動メモリー割り当ては、システムのハードウェアアーキテクチャーと利用可能なメモリーサイズによって異なります。

システムに、自動割り当ての最小メモリーしきい値より少ないメモリーしかない場合は、手動で予約メモリーの量を設定できます。

前提条件

- システムの root 権限がある。
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **crashkernel=** オプションを準備してください。
   - たとえば、128 MB のメモリーを予約するには、以下を使用します。

     ```
     crashkernel=128M
     ```

   - または、インストールされているメモリーの合計量に応じて、予約メモリーサイズを変数に設定できます。変数へのメモリー予約の構文は **crashkernel=<range1>:<size1>,<range2>:<size2>** です。以下に例を示します。

     ```
     crashkernel=512M-2G:64M,2G-:128M
     ```

     システムメモリーの合計量が 512 MB - 2 GB の範囲にある場合、64 MB のメモリーを予約します。メモリーの合計量が 2 GB を超える場合、メモリー予約は 128 MB になります。

   - 予約メモリーのオフセット。
     一部のシステムでは、**crashkernel** の予約が早い段階で行われるため、特定の固定オフセットでメモリーを予約する必要があります。また、特別な用途のために、さらに多くのメモリーの予約が必要になることもあります。オフセットを定義すると、予約メモリーはそこから開始されます。予約メモリーをオフセットするには、以下の構文を使用します。

     ```
     crashkernel=128M@16M
     ```

     この例では、**kdump** は 16 MB (物理アドレス **0x01000000**) から始まる 128 MB のメモリーを予約します。offset パラメーターを 0 に設定するか、完全に省略すると、**kdump** は予約メモリーを自動的にオフセットします。変数のメモリー予約を設定する場合は、この構文を使用することもできます。その場合、オフセットは常に最後に指定されます。以下に例を示します。

     ```
     crashkernel=512M-2G:64M,2G-:128M@16M
     ```

2. **crashkernel=** オプションをブートローダー設定に適用します。

   ```
   # grubby --update-kernel=ALL --args="crashkernel=<value>"
   ```

<value> は、前のステップで準備した **crashkernel=** オプションの値に置き換えます。

関連情報

- kdump メモリー要件
- カーネルコマンドラインパラメーターの設定
- システムを起動する前に、grub で boot パラメーターを手動で変更する
- How to install and boot custom kernels in Red Hat Enterprise Linux 8
- **grubby(8)** の man ページ

42.2.3. kdump ターゲットの設定

クラッシュダンプは通常、ローカルファイルシステムにファイルとして保存され、デバイスに直接書き込まれます。または、**NFS** プロトコルまたは **SSH** プロトコルを使用して、ネットワーク経由でクラッシュダンプを送信するように設定できます。クラッシュダンプファイルを保存するオプションは、一度に 1 つだけ設定できます。デフォルトの動作では、ローカルファイルシステムの **/var/crash/** ディレクトリーに保存されます。

前提条件

- **root** 権限
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

- ローカルファイルシステムの **/var/crash/** ディレクトリーにクラッシュダンプファイルを保存するには、**/etc/kdump.conf** ファイルを変更して、パスを指定します。

```
path /var/crash
```

path /var/crash オプションは、**kdump** がクラッシュダンプファイルを保存するファイルシステムへのパスを表します。

注記

- **/etc/kdump.conf** ファイルでダンプターゲットを指定すると、**path** は指定されたダンプ出力先に対する相対パスになります。
- **/etc/kdump.conf** ファイルでダンプターゲットを指定しない場合、パスはルートディレクトリーからの **絶対** パスを表します。

現在のシステムにマウントされている内容に応じて、ダンプターゲットと調整されたダンプパスが自動的に適用されます。

例42.1 kdump ターゲット設定

```
# grep -v ^# /etc/kdump.conf | grep -v ^$
ext4 /dev/mapper/vg00-varcrashvol
```

```
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

ここでは、ダンプターゲットが指定されているため (**ext4**/**dev**/**mapper**/**vg00-varcrashvol**)、/**var**/**crash** にマウントされます。**path** オプションも /**var**/**crash** に設定されているため、**kdump** は **vmcore** ファイルを /**var**/**crash**/**var**/**crash** ディレクトリーに保存します。

- クラッシュダンプを保存するローカルディレクトリーを変更するには、**root** として /**etc**/**kdump.conf** 設定ファイルを編集します。

  1. **#path** /**var**/**crash** の行頭にあるハッシュ記号 ("#") を削除します。
  2. 値を対象のディレクトリーパスに置き換えます。以下に例を示します。

     ```
     path /usr/local/cores
     ```

     重要

     RHEL 8 では、**path** ディレクティブを使用して kdump ターゲットとして定義されたディレクトリーは、**kdump systemd** サービスの開始時に存在する必要があります。存在しない場合、サービスは失敗します。この動作は、サービスの起動時にディレクトリーが存在しなかった場合は自動的に作成されていた RHEL の以前のリリースとは異なります。

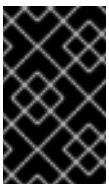
- ファイルを別のパーティションに書き込むには、/**etc**/**kdump.conf** 設定ファイルを編集します。

  1. 必要に応じて **#ext4** の行頭にあるハッシュ記号 ("#") を削除します。
     - デバイス名 (**#ext4** /**dev**/**vg**/**lv_kdump** 行)
     - ファイルシステムラベル (**#0ext4 LABEL=**/**boot** 行)
     - UUID (**#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** の行)
  2. ファイルシステムタイプと、デバイス名、ラベル、UUID を希望の値に変更します。以下に例を示します。

     ```
     ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
     ```

     注記

     UUID 値を指定するための正しい構文は、**UUID="correct-uuid"** と **UUID=correct-uuid** の両方です。

     重要

     LABEL= または **UUID=** を使用してストレージデバイスを指定することが推奨されます。/**dev**/**sda3** などのディスクデバイス名は、再起動した場合に一貫性が保証されません。

- クラッシュダンプを直接書き込むには、/**etc**/**kdump.conf** 設定ファイルを修正します。

1. **#raw /dev/vg/lv_kdump** の行頭にあるハッシュ記号 ("#") を削除します。
2. 値を対象のデバイス名に置き換えます。以下に例を示します。

```
raw /dev/sdb1
```

- **NFS** プロトコルを使用してクラッシュダンプをリモートマシンに保存するには、次の手順を実行します。
  1. **#nfs my.server.com:/export/tmp** の行頭にあるハッシュ記号 ("#") を削除します。
  2. 値を、正しいホスト名およびディレクトリーパスに置き換えます。以下に例を示します。

```
nfs penguin.example.com:/export/cores
```

- **SSH** プロトコルを使用してクラッシュダンプをリモートマシンに保存するには、次の手順を実行します。
  1. **#ssh user@my.server.com** の行頭にあるハッシュ記号 ("#") を削除します。
  2. 値を正しいユーザー名およびホスト名に置き換えます。
  3. **SSH** キーを設定に含めます。
     - **#sshkey /root/.ssh/kdump_id_rsa** の行頭にあるハッシュ記号 ("#") を削除します。
     - 値を、ダンプ先のサーバー上の正しいキーの場所に変更します。以下に例を示します。

```
ssh john@penguin.example.com
sshkey /root/.ssh/mykey
```

42.2.4. kdump コアコレクターの設定

kdump では、**core_collector** を使用してクラッシュダンプイメージをキャプチャーします。RHEL では、**makedumpfile** ユーティリティーがデフォルトのコアコレクターです。これは、以下に示すプロセスによりダンプファイルを縮小するのに役立ちます。

- クラッシュダンプファイルのサイズを圧縮し、さまざまなダンプレベルを使用して必要なページのみをコピーする
- 不要なクラッシュダンプページを除外する
- クラッシュダンプに含めるページタイプをフィルタリングする

Syntax

```
core_collector makedumpfile -l --message-level 1 -d 31
```

オプション

- **-c**、**-l**、または **-p**: **zlib** (**-c** オプションの場合)、**lzo** (**-l** オプションの場合)、または **snappy** (**-p** オプションの場合) のいずれかを使用して、ページごとに圧縮ダンプファイルの形式を指定します。
- **-d (dump_level)**: ページを除外して、ダンプファイルにコピーされないようにします。

- **--message-level**: メッセージタイプを指定します。このオプションで **message_level** を指定すると、出力の表示量を制限できます。たとえば、**message_level** で 7 を指定すると、一般的なメッセージとエラーメッセージを出力します。**message_level** の最大値は 31 です。

前提条件

- システムの root 権限がある。
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **root** で、**/etc/kdump.conf** 設定ファイルを編集し、**#core_collector makedumpfile -l --message-level 1 -d 31** の行頭にあるハッシュ記号 ("#") を削除します。
2. クラッシュダンプファイルの圧縮を有効にするには、以下のコマンドを実行します。

```
core_collector makedumpfile -l --message-level 1 -d 31
```

-l オプションにより、**dump** の圧縮ファイル形式を指定します。**-d** オプションで、ダンプレベルを 31 に指定します。**--message-level** オプションで、メッセージレベルを 1 に指定します。

また、**-c** オプションおよび **-p** オプションを使用した以下の例を検討してください。

- **-c** を使用してクラッシュダンプファイルを圧縮するには、以下のコマンドを実行します。

```
core_collector makedumpfile -c -d 31 --message-level 1
```

- **-p** を使用してクラッシュダンプファイルを圧縮するには、以下のコマンドを実行します。

```
core_collector makedumpfile -p -d 31 --message-level 1
```

関連情報

- **makedumpfile(8)** の man ページ
- kdump 設定ファイル

42.2.5. kdump のデフォルト障害応答の設定

デフォルトでは、設定したターゲットの場所で **kdump** がクラッシュダンプファイルの作成に失敗すると、システムが再起動し、ダンプがプロセス内で失われます。この場合は、以下の手順を実施します。

前提条件

- root 権限
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **root** で、**/etc/kdump.conf** 設定ファイルの **#failure_action** の行頭にあるハッシュ記号 ("#") を削除します。
2. 値を任意のアクションに置き換えます。

```
failure_action poweroff
```

関連情報

- kdump ターゲットの設定

42.2.6. kdump 設定のテスト

マシンが実稼働に入る前に、クラッシュダンププロセスが機能し、有効であることをテストできます。

警告

以下のコマンドでは、カーネルがクラッシュします。以下の手順に従う場合は、注意を払ってください。アクティブな実稼働システムで、不注意に使用しないでください。

手順

1. **kdump** を有効にしてシステムを再起動します。
2. **kdump** が動作していることを確認します。

```
# systemctl is-active kdump
active
```

3. Linux カーネルを強制的にクラッシュさせます。

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

警告

上記のコマンドによりカーネルがクラッシュし、再起動が必要になります。

もう一度起動すると、**/etc/kdump.conf** ファイルで指定した場所 (デフォルトでは **/var/crash/**) に **address-YYYY-MM-DD-HH:MM:SS/vmcore** ファイルが作成されます。

注記

このアクションで、設定の妥当性が確認されます。また、このアクションを使用して、一般的な作業負荷でクラッシュダンプの完了にかかる時間を記録することもできます。

関連情報

- kdump ターゲットの設定

42.3. KDUMP の有効化

この手順を使用すると、インストールされているすべてのカーネルまたは特定のカーネルに対して **kdump** サービスを有効または無効にすることができます。

42.3.1. インストールされているすべてのカーネルでの kdump の有効化

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、**kdump** を有効にして起動できます。

前提条件

- 管理者権限

手順

1. インストールしたすべてのカーネルに **crashkernel=auto** コマンドラインパラメーターを追加します。

```
# grubby --update-kernel=ALL --args="crashkernel=auto"
```

2. **kdump** を有効にします。

```
# systemctl enable --now kdump.service
```

検証

- **kdump** が実行されていることを確認します。

```
# systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
     Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
disabled)
     Active: active (live)
```

42.3.2. 特定のインストール済みカーネルでの kdump の有効化

マシン上の特定カーネルに対して、**kdump** を有効にできます。

前提条件

- 管理者権限

手順

1. マシンにインストールされているカーネルをリスト表示します。

```
# ls -a /boot/vmlinuz-*
/boot/vmlinuz-0-rescue-2930657cd0dc43c2b75db480e5e5b4a9 /boot/vmlinuz-4.18.0-
330.el8.x86_64 /boot/vmlinuz-4.18.0-330.rt7.111.el8.x86_64
```

2. 特定の **kdump** カーネルを、システムの GRUB (Grand Unified Bootloader) 設定ファイルに追加します。
以下に例を示します。

```
# grubby --update-kernel=vmlinuz-4.18.0-330.el8.x86_64 --args="crashkernel=auto"
```

3. **kdump** を有効にします。

```
# systemctl enable --now kdump.service
```

検証

- **kdump** が実行されていることを確認します。

```
# systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
     Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
disabled)
     Active: active (live)
```

42.3.3. kdump サービスの無効化

システムの起動時に **kdump** を無効にするには、以下の手順を行います。

前提条件

- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。
- **kdump** のインストール用のオプションがすべて、要件に応じて設定されている。詳細は、kdump のインストール を参照してください。

手順

1. 現在のセッションで **kdump** を停止するには、以下のコマンドを実行します。

```
# systemctl stop kdump.service
```

2. **kdump** を無効にするには、以下を行います。

```
# systemctl disable kdump.service
```

警告

kptr_restrict=1 を設定することが推奨されます。これにより、Kernel Address Space Layout (KASLR) が有効かどうかにかかわらず、**kdumpctl** はクラッシュカーネルを読み込みます。

トラブルシューティングの手順

kptr_restrict が (1) に設定されておらず、KASLR が有効になっている場合は、 **/proc/kcore** ファイルの内容がすべてゼロとして生成されます。したがって、**kdumpctl** サービスは **/proc/kcore** にアクセスしてクラッシュカーネルを読み込むことができません。

この問題を回避するために、**kptr_restrict=1** の設定を推奨する警告メッセージが **/usr/share/doc/kexec-tools/kexec-kdump-howto.txt** ファイルに表示されます。

kdumpctl サービスが必ずクラッシュカーネルを読み込むように、**kernel.kptr_restrict=1** が **sysctl.conf** ファイルに含まれていることを確認します。

関連情報

- RHEL の 基本的なシステム設定の設定

42.4. WEB コンソールで KDUMP の設定

RHEL 8 Web コンソールで **kdump** 設定を指定してテストします。

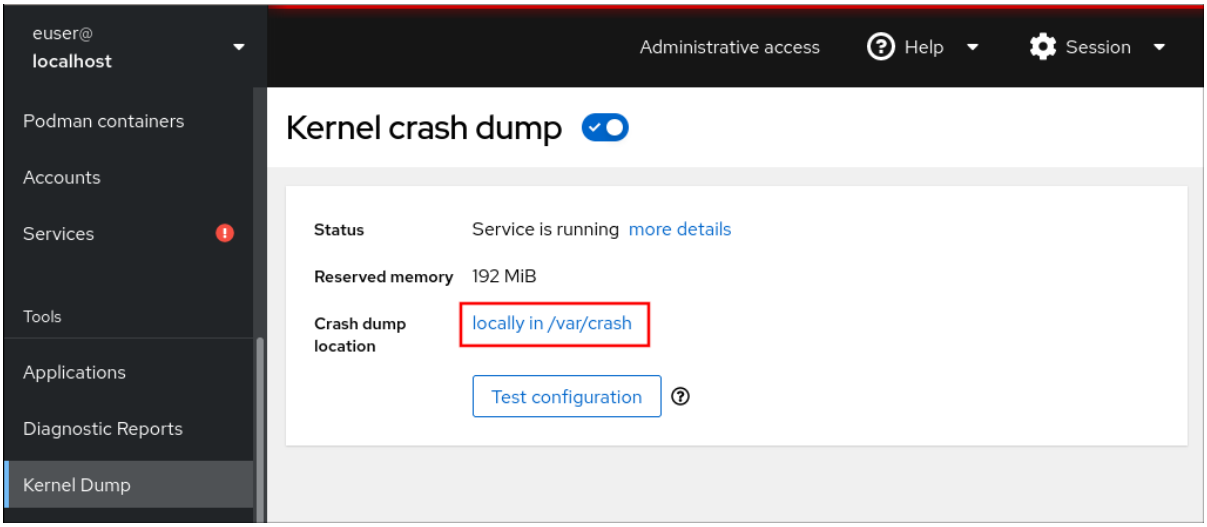
Web コンソールは、RHEL 8 のデフォルトインストールの一部で、システムの起動時に **kdump** を有効または無効にします。さらには、**kdump** に予約メモリーを設定したり、非圧縮または圧縮の形式で **vmcore** の保存場所を選択したりすることもできます。

42.4.1. 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

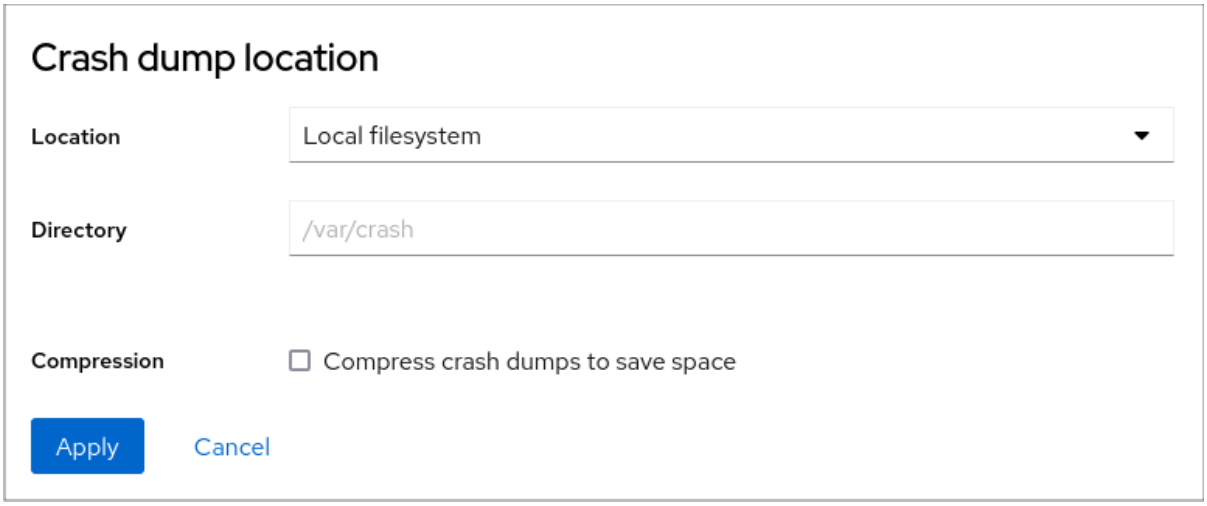
以下の手順では、RHEL Web コンソールインターフェイスの **Kernel Dump** タブを使用して、**kdump** カーネルに予約されているメモリー容量を設定する方法を示しています。この手順では、**vmcore** ダンプファイルのターゲットの場所を指定する方法と、設定をテストする方法を説明します。

手順

1. **Kernel Dump** タブを開き、**kdump** サービスを開始します。
2. コマンドラインで **kdump** のメモリー使用量を設定します。
3. **クラッシュダンプの場所** オプションの横にあるリンクをクリックします。



4. ドロップダウンメニューから **ローカルファイルシステム** を選択し、ダンプを保存するディレクトリーを指定します。



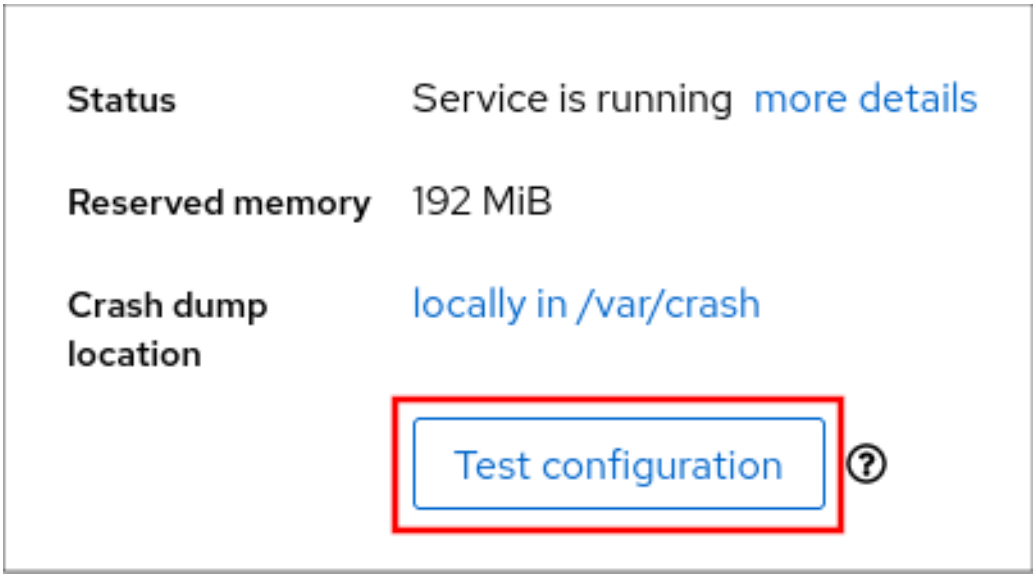
- または、ドロップダウンから **SSH 経由のリモート** オプションを選択し、SSH プロトコルを使用して、vmcore をリモートマシンに送信します。
  Server、**ssh key**、**Directory** の各フィールドに、リモートマシンのアドレス、ssh キーの場所、およびターゲットディレクトリーを入力します。

- または、ドロップダウンから **NFS 経由のリモート** オプションを選択し、**マウント** フィールドに入力して、NFS プロトコルを使用して vmcore をリモートマシンに送信することもできます。

注記

圧縮 チェックボックスにチェックマークを入れ、vmcore ファイルのサイズを小さくします。

5. カーネルをクラッシュして、設定をテストします。



a. **Test configuration** をクリックします。

b. **Test kdump settings** フィールドで、**Crash system** をクリックします。

> **警告**
>
> この手順では、カーネルの実行を中断し、システムクラッシュやデータの損失が発生します。

関連情報

- サポートしている kdump のダンプ出力先
- 2 台のシステム間で OpenSSH を使用した安全な通信の使用

42.4.2. 関連情報

- RHEL Web コンソールの使用

42.5. サポートしている KDUMP の設定とダンプ出力先

42.5.1. kdump メモリー要件

kdump でカーネルクラッシュのダンプをキャプチャーして分析のため保存するには、キャプチャーカーネル用にシステムメモリーの一部を永続的に予約しておく必要があります。予約されている場合、システムメモリーのこの部分はメインカーネルでは使用できません。

メモリー要件は、特定のシステムパラメーターによって異なります。主な要因は、システムのハードウェアアーキテクチャーです。正確なマシンアーキテクチャー (Intel 64 や AMD64 (x86_64) など) を調べ、それを標準出力に出力するには、以下のコマンドを使用します。

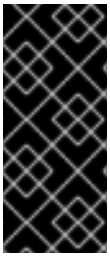
```
$ uname -m
```

kdump に必要な予約メモリーの最小量 の表には、利用可能な最新バージョンで **kdump** のメモリーサイズを自動的に予約するための最小メモリー要件が含まれています。システムのアーキテクチャーと利用可能な物理メモリーの合計に応じて、サイズが変更されます。

表42.1 kdump 用に必要な最小予約メモリー

アーキテクチャー	使用可能なメモリー	最小予約メモリー
AMD64 と Intel 64 (**x86_64**)	1 GB から 4 GB	192 MB のメモリー
	4 GB から 64 GB	256 MB のメモリー
	64 GB 以上	512 MB のメモリー
64 ビット ARM アーキテクチャー (**arm64**)	2 GB 以上	480 MB のメモリー
IBM Power Systems (**ppc64le**)	2 GB から 4 GB	384 MB のメモリー
	4 GB から 16 GB	512 MB のメモリー
	16 GB から 64 GB	1 GB のメモリー
	64 GB から 128 GB	2 GB のメモリー
	128 GB 以上	4 GB のメモリー
IBM Z (**s390x**)	1 GB から 4 GB	192 MB のメモリー
	4 GB から 64 GB	256 MB のメモリー
	64 GB 以上	512 MB のメモリー

多くのシステムでは、**kdump** は必要なメモリー量を予測して、自動的に予約できます。この動作はデフォルトで有効になっていますが、利用可能な合計メモリーサイズが一定以上搭載されているシステムに限られます。この自動割り当て動作に必要なメモリーサイズはシステムのアーキテクチャーによって異なります。



重要

システムのメモリー合計量に基づく予約メモリーの自動設定は、ベストエフォート予測です。実際に必要なメモリーは、I/O デバイスなどの他の要素により異なる場合があります。メモリーが十分でない場合は、カーネルパニックが発生したときにデバッグカーネルがキャプチャーカーネルとして起動できなくなる可能性があります。この問題を回避するには、クラッシュカーネルメモリーを十分なサイズにします。

関連情報

- RHEL8 マイナーリリース間で crashkernel パラメーターがどのように変化してきたか？

- テクノロジープレビュー機能および制限の表
- メモリー自動予約の最小しきい値

42.5.2. メモリー自動予約の最小しきい値

一部のシステムでは、ブートローダー設定ファイルで **crashkernel=auto** パラメーターを使用するか、グラフィカル設定ユーティリティーでこのオプションを有効にすることで、**kdump** 用のメモリーを自動的に割り当てることができます。ただし、この自動予約が機能するには、合計メモリーの特定量のメモリーを利用できる必要があります。必要な容量は、システムのアーキテクチャーによって異なります。

次の表は、自動メモリー割り当てのしきい値のリストです。システムのメモリーが指定のしきい値よりも小さい場合は、メモリーを手動で設定する必要があります。

表42.2 自動メモリー予約に必要な最小メモリーサイズ

アーキテクチャー	必要なメモリー
AMD64 と Intel 64 (**x86_64**)	2 GB
IBM Power Systems (**ppc64le**)	2 GB
IBM Z (**s390x**)	4 GB

42.5.3. サポートしている kdump のダンプ出力先

カーネルクラッシュがキャプチャされたら、vmcore ダンプファイルはデバイスに直接書き込むか、ローカルファイルシステム上でファイルとして保存されるか、ネットワークで送信されます。以下の表に、現在対応のダンプ出力先、または **kdump** が明示的に対応していないダンプ出力先の完全なリストを示します。

タイプ	対応しているダンプ出力先	対応していないダンプ出力先
Raw デバイス	ローカルで添付されたすべての raw ディスクとパーティション	
ローカルファイルシステム	直接接続されているディスクドライブ、ハードウェア RAID 論理ドライブ、LVM デバイス、**mdraid** アレイ上の **ext2**、**ext 3**、**ext4**、および **xfs** ファイルシステム。	**auto** タイプ (自動ファイルシステム検出) など、この表で明示的にサポート対象とされていないローカルファイルシステム。
リモートディレクトリー	**IPv4** で **NFS** または **SSH** プロトコルを使用してアクセスしたリモートディレクトリー。	**NFS** プロトコルを使用してアクセスした **rootfs** ファイルシステム上のリモートディレクトリー。
ハードウェアおよびソフトウェアイニシエーター上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー。	**be2iscsi** ハードウェア上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー。	マルチパスベースのストレージ

タイプ	対応しているダンプ出力先	対応していないダンプ出力先
		IPv6 上でアクセスするリモートディレクトリー
		SMB または **CIFS** を使用してアクセスするリモートディレクトリー。
		FCoE (Fibre Channel over Ethernet) プロトコルを使用してアクセスするリモートディレクトリー。
		ワイヤレスネットワークインターフェイスを使用してアクセスするリモートディレクトリー

重要

fadump (firmware assisted dump) を使用して vmcore を取得し、SSH プロトコルまたは NFS プロトコルを使用してリモートマシンに保存すると、ネットワークインターフェイスの名前が **kdump-<interface-name>** に変更になります。名前変更は、**<interface-name>** が ***eth#**、**net#** などのように一般的な場合に発生します。この問題は、初期 RAM ディスク (**initrd**) の vmcore 取得スクリプトが、ネットワークインターフェイス名に接尾辞 **kdump-** を追加して、永続的な名前付けを保護するために発生します。同じ **initrd** が通常の起動にも使用されるため、実稼働環境のカーネルのインターフェイス名も変更されます。

関連情報

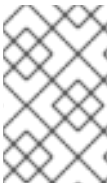
- kdump ターゲットの設定

42.5.4. 対応している kdump のフィルターレベル

ダンプファイルのサイズを縮小するために、**kdump** は **makedumpfile** コアコレクターを使用してデータを圧縮し、必要に応じて不要な情報を省略します。以下の表に、**makedumpfile** ユーティリティーで現在対応しているフィルターレベルの完全なリストを示します。

オプション	説明
1	ゼロページ
2	キャッシュページ
4	キャッシュプライベート
8	ユーザーページ

オプション	説明
16	フリーページ



注記

makedumpfile コマンドは、透過的な大規模ページおよび hugetlbfs ページの削除に対応しています。これらのタイプの hugepages User Page の両方を考えて、**-8** レベルを使用して削除します。

関連情報

- コアコレクターの設定

42.5.5. 対応しているデフォルトの障害応答

デフォルトでは、**kdump** がコアダンプを作成できない場合、オペレーティングシステムが再起動します。ただし、コアダンプをプライマリーターゲットに保存できない場合は、**kdump** が別の操作を実行するように設定できます。次の表は、現在対応しているすべてのデフォルトアクションのリストです。

オプション	説明
dump_to_rootfs	root ファイルシステムにコアダンプの保存を試行します。ネットワーク上のダンプ出力先と併用する場合に特に便利なオプションです。ネットワーク上のダンプ出力先にアクセスできない場合、ローカルにコアダンプを保存するよう kdump の設定を行います。システムは、後で再起動します。
reboot	システムを再起動します。コアダンプは失われます。
halt	システムを停止します。コアダンプは失われます。
poweroff	システムの電源を切ります。コアダンプは失われます。
shell	initramfs 内から shell セッションを実行して、ユーザーが手動でコアダンプを記録できるようにします。
final_action	**kdump** の成功後、または **shell** または **dump_to_rootfs** の失敗アクションの完了時に、**reboot**、**halt** および **poweroff** アクションなどの追加の操作を有効にします。デフォルトの **final_action** オプションは **reboot** です。

関連情報

- kdump のデフォルト障害応答の設定

42.5.6. final_action パラメーターの使用

final_action パラメーターを使用すると、**kdump** の成功後、または、 **shell** または **dump_to_rootfs** を使用した無効な **failure_response** メカニズムの完了時に、**reboot**、**halt** および **poweroff** アクションなど、特定の操作を追加で使用できます。**final_action** オプションを指定しない場合には、この値はデフォルトで **reboot** になります。

手順

1. **/etc/kdump.conf** ファイルを編集し、**final_action** パラメーターを追加します。

```
final_action <reboot | halt | poweroff>
```

2. **kdump** サービスを再起動します。

```
kdumpctl restart
```

42.6. KDUMP 設定のテスト

マシンが実稼働に入る前に、クラッシュダンププロセスが機能し、有効であることをテストできます。

警告

以下のコマンドでは、カーネルがクラッシュします。以下の手順に従う場合は、注意を払ってください。アクティブな実稼働システムで、不注意に使用しないでください。

手順

1. **kdump** を有効にしてシステムを再起動します。
2. **kdump** が動作していることを確認します。

```
# systemctl is-active kdump
active
```

3. Linux カーネルを強制的にクラッシュさせます。

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

警告

上記のコマンドによりカーネルがクラッシュし、再起動が必要になります。

もう一度起動すると、**/etc/kdump.conf** ファイルで指定した場所 (デフォルトでは **/var/crash/**) に **address-YYYY-MM-DD-HH:MM:SS/vmcore** ファイルが作成されます。

注記

このアクションで、設定の妥当性が確認されます。また、このアクションを使用して、一般的な作業負荷でクラッシュダンプの完了にかかる時間を記録することもできます。

関連情報

- kdump ターゲットの設定

42.7. KEXEC を使用した別のカーネルの起動

kexec システムコールを使用すると現在実行中のカーネルから別のカーネルを読み込んだり、起動したりすることが可能で、カーネル内のブートローダーとして機能します。

kexec ユーティリティーは、**kexec** システムコールのカーネルおよび **initramfs** イメージを読み込み、別のカーネルで起動します。

以下の手順では、**kexec** ユーティリティーを使用して別のカーネルに再起動する時に、**kexec** システムコールを手動で呼び出す方法を説明します。

手順

1. **kexec** ユーティリティーを実行します。

```
# kexec -l /boot/vmlinuz-3.10.0-1040.el7.x86_64 --initrd=/boot/initramfs-3.10.0-1040.el7.x86_64.img --reuse-cmdline
```

このコマンドは、**kexec** システムコールのカーネルおよび initramfs イメージを手動で読み込みます。

2. システムを再起動します。

```
# reboot
```

このコマンドはカーネルを検出し、すべてのサービスをシャットダウンしてから、**kexec** システムコールを呼び出して直前の手順で指定したカーネルに再起動します。

警告

kexec -3 コマンドを使用して、マシンを別のカーネルで再起動すると、システムは、次のカーネルを起動する前に標準のシャットダウンシーケンスを通過しません。これにより、データが失われたり、システムが応答しなくなったりする可能性があります。

42.8. カーネルドライバーが KDUMP を読み込まないようにする設定

/etc/sysconfig/kdump 設定ファイルに **KDUMP_COMMANDLINE_APPEND=** 変数を追加することで、キャプチャーカーネルが特定のカーネルドライバーをロードしないように制御できます。この方法を使用すると、**kdump** 初期 RAM ディスクイメージ **initramfs** が、指定されたカーネルモジュールをロードするのを防ぐことができます。これにより、メモリー不足 (oom) killer エラーやその他のクラッシュカーネル障害を防ぐことができます。

以下の設定オプションのいずれかを使用して、**KDUMP_COMMANDLINE_APPEND=** 変数を追加することができます。

- **rd.driver.blacklist=<modules>**
- **modprobe.blacklist=<modules>**

手順

1. 読み込みをブロックするカーネルモジュールを選択します。

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  1
uinput                 20480  1
xt_conntrack           16384  1
```

lsmod コマンドは、現在実行中のカーネルに読み込まれているモジュールのリストを表示します。

2. **/etc/sysconfig/kdump** ファイルの **KDUMP_COMMANDLINE_APPEND=** 変数を更新します。

```
#
KDUMP_COMMANDLINE_APPEND="rd.driver.blacklist=hv_vmbus,hv_storvsc,hv_utils,
hv_netvsc,hid-hyperv"
```

modprobe.blacklist=<modules> 設定オプションを使用した以下の例も検討してください。

```
# KDUMP_COMMANDLINE_APPEND="modprobe.blacklist=emcp
modprobe.blacklist=bnx2fc modprobe.blacklist=libfcoe modprobe.blacklist=fcoe"
```

3. **kdump** サービスを再起動します。

```
# systemctl restart kdump
```

関連情報

- **dracut.cmdline** の man ページ

42.9. 暗号化されたディスクがあるシステムでの KDUMP の実行

LUKS 暗号化パーティションを実行すると、システムで利用可能なメモリーが一定量必要になります。システムが必要なメモリー量を下回ると、**cryptsetup** ユーティリティーがパーティションのマウントに失敗します。その結果、2 番目のカーネル (キャプチャーカーネル) で、暗号化したターゲットの場所に **vmcore** ファイルをキャプチャーできませんでした。

kdumpctl estimate コマンドは、**kdump** に必要なメモリーの量を見積もるのに役立ちます。**kdumpctl estimate** 値は、推奨される **crashkernel** 値を出力します。これは、**kdump** に必要な最適なメモリーサイズです。

推奨の **crashkernel** 値は、現在のカーネルサイズ、カーネルモジュール、initramfs、および暗号化したターゲットメモリー要件に基づいて計算されます。

カスタムの **crashkernel=** オプションを使用している場合には、**kdumpctl estimate** は **LUKS required size** 値を出力します。この値は、LUKS 暗号化ターゲットに必要なメモリーサイズです。

手順

1. **crashkernel=** の推定値を出力します。

```
# kdumpctl estimate

Encrypted kdump target requires extra memory, assuming using the keyslot with minimum memory requirement
  Reserved crashkernel:    256M
  Recommended crashkernel: 652M

  Kernel image size:   47M
  Kernel modules size: 8M
  Initramfs size:      20M
  Runtime reservation: 64M
  LUKS required size:  512M
  Large modules: <none>
  WARNING: Current crashkernel size is lower than recommended size 652M.
```

2. **crackkernel =** を目的の値に増やして、必要なメモリーの量を設定します。

3. システムを再起動します。

注記

それでも **kdump** がダンプファイルを暗号化したターゲットに保存できない場合は、必要に応じて **crashkernel=** を増やしてください。

42.10. ファームウェア支援ダンプの仕組み

ファームウェア支援ダンプ (fadump) は、IBM POWER システムの **kdump** メカニズムの代わりに提供されるダンプ取得メカニズムです。**kexec** および **kdump** のメカニズムは、AMD64 および Intel 64 システムでコアダンプを取得する際に役立ちます。ただし、最小システムやメインフレームコンピューターなどの一部のハードウェアでは、オンボードファームウェアを活用してメモリー領域を分離して、クラッシュ分析に重要なデータが誤って上書きされないようにします。**fadump** ユーティリティーは、**fadump** メカニズムと IBM POWER システム上の RHEL との統合向けに最適化されています。

42.10.1. IBM PowerPC ハードウェアにおけるファームウェア支援ダンプ

fadump ユーティリティーは、PCI デバイスおよび I/O デバイスが搭載され、完全にリセットされたシステムから **vmcore** ファイルをキャプチャーします。この仕組みでは、クラッシュするとファームウェアを使用してメモリー領域を保存し、**kdump** ユーザー空間スクリプトをもう一度使用して **vmcore** ファイルを保存します。このメモリー領域には、ブートメモリー、システムレジスター、およびハードウェアのページテーブルエントリー (PTE) を除く、すべてのシステムメモリーコンテンツが含まれます。

fadump メカニズムは、パーティションを再起動し、新規カーネルを使用して以前のカーネルクラッシュからのデータをダンプすることで従来のダンプタイプに比べて信頼性が向上されています。**fadump** には、IBM POWER6 プロセッサーベースまたはそれ以降バージョンのハードウェアプラットフォームが必要です。

PowerPC 固有のハードウェアのリセット方法など、**fadump** メカニズムの詳細は、**/usr/share/doc/kexec-tools/fadump-howto.txt** ファイルを参照してください。

注記

保持されないメモリー領域はブートメモリーと呼ばれており、この領域はクラッシュ後にカーネルを正常に起動するのに必要なメモリー容量です。デフォルトのブートメモリーサイズは、256 MB または全システム RAM の 5% のいずれか大きい方です。

kexec で開始されたイベントとは異なり、**fadump** メカニズムでは実稼働用のカーネルを使用してクラッシュダンプを復元します。PowerPC ハードウェアは、クラッシュ後の起動時に、デバイスノード **/proc/device-tree/rtas/ibm.kernel-dump** が **proc** ファイルシステム (**procfs**) で利用できるようにします。**fadump-aware kdump** スクリプトでは、保存された **vmcore** があるかを確認してから、システムの再起動を正常に完了させます。

42.10.2. ファームウェア支援ダンプメカニズムの有効化

IBM POWER システムのクラッシュダンプ機能は、ファームウェア支援ダンプ (**fadump**) メカニズムを有効にすることで強化できます。

セキュアブート環境では、**GRUB2** ブートローダーは、Real Mode Area (RMA) と呼ばれるブートメモリー領域を割り当てます。RMA のサイズは 512 MB で、ブートコンポーネント間で分割されます。コンポーネントがサイズの割り当てを超えると、**GRUB2** はメモリー不足 (**OOM**) エラーで失敗します。

警告

RHEL 8.7 および 8.6 バージョンのセキュアブート環境では、ファームウェア支援ダンプ (**fadump**) メカニズムを有効にしないでください。**GRUB2** ブートローダーが次のエラーで失敗します。

```
error: ../../grub-core/kern/mm.c:376:out of memory.
Press any key to continue…
```

システムは、**fadump** 設定のためにデフォルトの **initramfs** サイズを増やした場合にのみ回復可能です。

システムを回復するための回避策については、記事 System boot ends in GRUB Out of Memory (OOM) を参照してください。

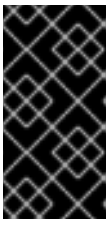
手順

1. **kdump** のインストールと設定
2. **fadump=on** カーネルオプションを有効にします。

```
# grubby --update-kernel=ALL --args="fadump=on"
```

3. (オプション) デフォルトを使用する代わりに、予約済みのブートメモリーを指定する場合は、**crashkernel=xxM** オプションを有効にします。ここで、**xx** は必要なメモリーの量 (メガバイト単位) です。

```
# grubby --update-kernel=ALL --args="crashkernel=xxM fadump=on"
```



重要

ブート設定オプションを指定するときは、実行する前にすべてのブート設定オプションをテストしてください。**kdump** カーネルの起動に失敗した場合は、**crashkernel=** 引数に指定した値を徐々に増やして、適切な値を設定します。

42.10.3. IBM Z ハードウェアにおけるファームウェア支援ダンプの仕組み

IBM Z システムは、以下のファームウェア支援ダンプメカニズムをサポートします。

- **スタンドアロンダンプ (sadump)**
- **VMDUMP**

IBM Z システムでは、**kdump** インフラストラクチャーはサポート対象で、使用されています。ただし、IBM Z にファームウェア支援ダンプ (fadump) を使用すると、さまざまな利点が得られます。

- **sadump** メカニズムはシステムコンソールで開始および制御され、**IPL** 起動可能なデバイスに保存されます。

- **VMDUMP** メカニズムは **sadump** に似ています。このツールもシステムコンソールから開始しますが、ハードウェアから生成されたダンプを取得して解析用にシステムにコピーします。
- (他のハードウェアベースのダンプメカニズムと同様に) これらの手法では、(**kdump** サービスが開始される前の) 起動初期段階におけるマシンの状態をキャプチャーできます。
- **VMDUMP** には、ハードウェアからコピーしたダンプファイルを Red Hat Enterprise Linux システムに格納する仕組みがありますが、IBM Z ハードウェアコンソールから、**VMDUMP** の設定および制御が管理されます。

IBM は、 **sadump** について **Stand-alone dump program** 記事で 、 **VMDUMP** について **Creating dumps on z/VM with VMDUMP** 記事で詳説しています。

また、IBM は、**Using the Dump Tools on Red Hat Enterprise Linux 7.4** 記事で Red Hat Linux Enterprise Linux 7 でダンプツールを使用するための一連のドキュメントも用意しています。

関連情報

- スタンドアロンダンププログラム
- VMDUMP を使用した z/VM でのダンプの作成
- Red Hat Enterprise Linux 7.4 でのダンプツールの使用

42.10.4. Fujitsu PRIMEQUEST システムにおける sadump の使用

Fujitsu **sadump** メカニズムは、**kdump** が正常に完了しないイベントで **fallback** ダンプキャプチャーを提供するように設計されています。**sadump** メカニズムは、システムの ManageMent Board (MMB) インターフェイスから手動で呼び出します。MMB を使用して、Intel 64 サーバーまたは AMD 64 サーバーのように **kdump** を設定し、**sadump** の有効化手順を追加で実施します。

手順

1. **sadump** に対して **kdump** が予想どおりに起動するように **/etc/sysctl.conf** ファイルで以下の行を追加または編集します。

```
kernel.panic=0
kernel.unknown_nmi_panic=1
```

> **警告**
>
> 特に、**kdump** の後にシステムが再起動しないようにする必要があります。**kdump** が **vmcore** ファイルの保存失敗後にシステムを再起動すると、**sadump** を呼び出すことができません。

2. **/etc/kdump.conf** の **failure_action** パラメーターを **halt** または **shell** として適切に設定します。

```
failure_action shell
```

関連情報

- FUJITSU Server PRIMEQUEST 2000 Series インストールマニュアル

42.11. コアダンプの分析

システムクラッシュの原因を確認するには、**crash** ユーティリティーを使用します。これにより、GDB (GNU Debugger) と非常によく似たインタラクティブなプロンプトを利用できます。このユーティリティーでは、**kdump**、**netdump**、**diskdump**、または **xendump** によって作成されたコアダンプ、実行中の Linux システムなどをインタラクティブに分析できます。または、Kernel Oops Analyzer または Kdump Helper ツールを使用する選択肢もあります。

42.11.1. crash ユーティリティーのインストール

crash ツールをインストールして、コア分析スイートを取得します。

手順

1. 関連するリポジトリーを有効にします。

```
# subscription-manager repos --enable baseos repository
```

```
# subscription-manager repos --enable appstream repository
```

```
# subscription-manager repos --enable rhel-8-for-x86_64-baseos-debug-rpms
```

2. **crash** パッケージをインストールします。

```
# yum install crash
```

3. **kernel-debuginfo** パッケージをインストールします。

```
# yum install kernel-debuginfo
```

パッケージは実行中のカーネルに対応し、ダンプ分析に必要なデータを提供します。

関連情報

- 基本的なシステム設定の設定

42.11.2. crash ユーティリティーの実行および終了

システムクラッシュの原因を分析するため、**crash** ユーティリティーを起動します。

前提条件

- 現在実行しているカーネルを特定します (**4.18.0-5.el8.x86_64** など)。

手順

1. **crash** ユーティリティーを起動するには、2 つの必要なパラメーターをコマンドに渡す必要があります。

- debug-info (圧縮解除された vmlinuz イメージ) (特定の **kernel-debuginfo** パッケージに含まれる **/usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinux** など)
- 実際の vmcore ファイル。**/var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore**
  その結果の **crash** コマンドの以下のようになります。

```
# crash /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinux /var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore
```

kdump で取得したのと同じ **<kernel>** のバージョンを使用します。

例42.2 crash ユーティリティーの実行

以下の例は、4.18.0-5.el8.x86_64 カーネルを使用して、2018 年 10 月 6 日 (14:05 PM) に作成されたコアダンプの分析を示しています。

```
...
WARNING: kernel relocated [202MB]: patching 90160 gdb minimal_symbol values

      KERNEL: /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore  [PARTIAL DUMP]
        CPUS: 2
        DATE: Sat Oct  6 14:05:16 2018
      UPTIME: 01:03:57
LOAD AVERAGE: 0.00, 0.00, 0.00
       TASKS: 586
    NODENAME: localhost.localdomain
     RELEASE: 4.18.0-5.el8.x86_64
     VERSION: #1 SMP Wed Aug 29 11:51:55 UTC 2018
     MACHINE: x86_64  (2904 Mhz)
      MEMORY: 2.9 GB
       PANIC: "sysrq: SysRq : Trigger a crash"
         PID: 10635
     COMMAND: "bash"
        TASK: ffff8d6c84271800  [THREAD_INFO: ffff8d6c84271800]
         CPU: 1
       STATE: TASK_RUNNING (SYSRQ)

crash>
```

2. 対話型プロンプトを終了して **crash** を終了するには、**crash** または **q** を使用します。

例42.3 crash ユーティリティーの終了

```
crash> exit
~]#
```

注記

crash コマンドは、ライブシステムをデバッグする強力なツールとして使用することもできます。ただし、システムを破損しないように注意してください。

関連情報

- 予期しないシステムの再起動のガイド

42.11.3. crash ユーティリティーのさまざまなインジケーターの表示

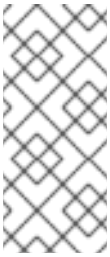
crash ユーティリティーを使用して、カーネルメッセージバッファー、バックトレース、プロセスステータス、仮想メモリー情報、開いているファイルなど、さまざまなインジケータを表示します。

メッセージバッファーの表示

- カーネルメッセージバッファーを表示するには、以下の例で示されているように対話式プロンプトで **log** コマンドを実行します。

```
crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
 DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
 c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> fffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002 b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560 ef4dbfb4
Call Trace:
 [<c068146b>] ? __handle_sysrq+0xfb/0x160
 [<c06814d0>] ? write_sysrq_trigger+0x0/0x50
 [<c068150f>] ? write_sysrq_trigger+0x3f/0x50
 [<c0569ec4>] ? proc_reg_write+0x64/0xa0
 [<c0569e60>] ? proc_reg_write+0x0/0xa0
 [<c051de50>] ? vfs_write+0xa0/0x190
 [<c051e8d1>] ? sys_write+0x41/0x70
 [<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0 88 41 03 f3 c3 90 c7 05
c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05 00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50
d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000
```

コマンドの使用の詳細は **help log** を実行します。



注記

カーネルメッセージバッファーには、システムクラッシュに関する最も重要な情報が含まれています。したがって、これは常に最初に **vmcore-dmesg.txt** ファイルにダンプされます。これは、たとえば、ターゲットの場所にスペースがないために、**vmcore** ファイル全体の取得試行に失敗するときに便利です。デフォルトでは、**vmcore-dmesg.txt** は **/var/pkcs/directory/** あります。

バックトレースの表示

- カーネルスタックトレースを表示するには、**bt** コマンドを使用します。

```
crash> bt
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
 #0 [ef4dbdcc] crash_kexec at c0494922
 #1 [ef4dbe20] oops_end at c080e402
 #2 [ef4dbe34] no_context at c043089d
 #3 [ef4dbe58] bad_area at c0430b26
 #4 [ef4dbe6c] do_page_fault at c080fb9b
 #5 [ef4dbee4] error_code (via page_fault) at c080d809
    EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP: 00000000
    DS:  007b      ESI: c0a09ca0  ES:  007b      EDI: 00000286  GS:  00e0
    CS:  0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
 #6 [ef4dbf18] sysrq_handle_crash at c068124f
 #7 [ef4dbf24] __handle_sysrq at c0681469
 #8 [ef4dbf48] write_sysrq_trigger at c068150a
 #9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: fffffffda  EBX: 00000001  ECX: b7776000  EDX: 00000002
    DS:  007b      ESI: 00000002  ES:  007b      EDI: b7776000
    SS:  007b      ESP: bfcb2088  EBP: bfcb20b4  GS:  0033
    CS:  0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246
```

bt <pid> を入力して特定のプロセスのバックトレースを表示するか、**help bt** を実行して、**bt** の使用についての詳細を表示します。

プロセスの状態表示

- システム内のプロセスの状態を表示するには、**ps** コマンドを使用します。

```
crash> ps
   PID    PPID  CPU   TASK    ST  %MEM     VSZ    RSS  COMM
>     0      0   0  c09dc560  RU   0.0       0      0  [swapper]
>     0      0   1  f7072030  RU   0.0       0      0  [swapper]
      0      0   2  f70a3a90  RU   0.0       0      0  [swapper]
>     0      0   3  f70ac560  RU   0.0       0      0  [swapper]
      1      0   1  f705ba90  IN   0.0    2828   1424  init
... several lines omitted ...
   5566      1   1  f2592560  IN   0.0   12876    784  auditd
   5567      1   2  ef427560  IN   0.0   12876    784  auditd
   5587   5132   0  f196d030  IN   0.0   11064   3184  sshd
>  5591   5587   2  f196d560  RU   0.0    5084   1648  bash
```

ps <pid> を使用して、単一プロセスのステータスを表示します。**ps** の詳細な使用方法は、**help ps** を使用します。

仮想メモリー情報の表示

- 基本的な仮想メモリー情報を表示するには、対話式プロンプトで **vm** コマンドを入力します。

```
crash> vm
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
   MM       PGD      RSS    TOTAL_VM
f19b5900  ef9c6000  1648k    5084k
```

```
  VMA       START      END    FLAGS  FILE
f1bb0310    242000    260000 8000875  /lib/ld-2.12.so
f26af0b8    260000    261000 8100871  /lib/ld-2.12.so
efbc275c    261000    262000 8100873  /lib/ld-2.12.so
efbc2a18    268000    3ed000 8000075  /lib/libc-2.12.so
efbc23d8    3ed000    3ee000 8000070  /lib/libc-2.12.so
efbc2888    3ee000    3f0000 8100071  /lib/libc-2.12.so
efbc2cd4    3f0000    3f1000 8100073  /lib/libc-2.12.so
efbc243c    3f1000    3f4000 100073
efbc28ec    3f6000    3f9000 8000075  /lib/libdl-2.12.so
efbc2568    3f9000    3fa000 8100071  /lib/libdl-2.12.so
efbc2f2c    3fa000    3fb000 8100073  /lib/libdl-2.12.so
f26af888    7e6000    7fc000 8000075  /lib/libtinfo.so.5.7
f26aff2c    7fc000    7ff000 8100073  /lib/libtinfo.so.5.7
efbc211c    d83000    d8f000 8000075  /lib/libnss_files-2.12.so
efbc2504    d8f000    d90000 8100071  /lib/libnss_files-2.12.so
efbc2950    d90000    d91000 8100073  /lib/libnss_files-2.12.so
f26afe00    edc000    edd000 4040075
f1bb0a18   8047000   8118000 8001875  /bin/bash
f1bb01e4   8118000   811d000 8101873  /bin/bash
f1bb0c70   811d000   8122000 100073
f26afae0   9fd9000   9ffa000 100073
... several lines omitted ...
```

vm <pid> を実行して、1 つのプロセスの情報を表示するか、**help vm** を実行して、**vm** の使用方法を表示します。

オープンファイルの表示

- オープンファイルの情報を表示するには、**files** コマンドを実行します。

```
crash> files
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
ROOT: /    CWD: /root
 FD    FILE     DENTRY    INODE    TYPE  PATH
  0  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
  1  efade5c0  eee14090  f00431d4  REG   /proc/sysrq-trigger
  2  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
 10  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
255  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
```

files <pid> を実行して、選択した 1 つのプロセスによって開かれたファイルを表示するか、**help files** を実行して、**files** の使用方法を表示します。

42.11.4. Kernel Oops Analyzer の使用

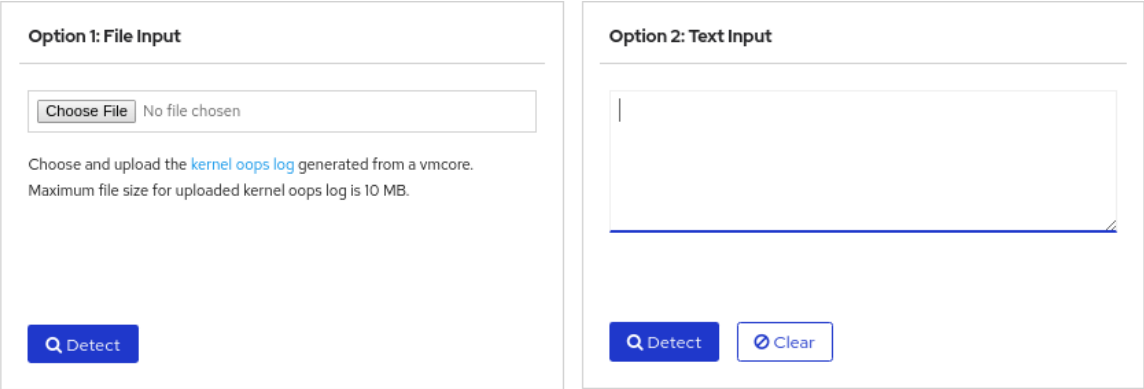
Kernel Oops Analyzer ツールでは、ナレッジベースの既知の問題で oops メッセージを比較することでクラッシュダンプを分析します。

前提条件

- oops メッセージのセキュリティーを保護し、Kernel Oops Analyzer にフィードしている。

手順

1. Kernel Oops Analyzer ツールにアクセスします。
2. カーネルクラッシュの問題を診断するには、**vmcore** に生成されたカーネルの oops ログをアップロードします。
   - テキストメッセージまたは **vmcore-dmesg.txt** を入力として指定して、カーネルクラッシュの問題を診断することも可能です。



3. **DETECT** をクリックして、**makedumpfile** からの情報に基づいて既知のソリューションと oops メッセージを比較します。

関連情報

- Kernel Oops Analyzer の記事
- 予期しないシステムの再起動のガイド

42.11.5. Kdump Helper ツール

Kdump ヘルパーツールは、提供された情報を使用して **kdump** を設定するのに役立ちます。Kdump Helper は、ユーザーの設定に基づいて設定スクリプトを生成します。サーバーでスクリプトを開始して実行すると、**kdump** サービスが設定されます。

関連情報

- Kdump ヘルパー

42.12. EARLY KDUMP を使用した起動時間クラッシュの取得

kdump サービスの **early kdump** メカニズムを使用して、ブートプロセスの初期段階で **vmcore** ファイルをキャプチャーします。次の情報と手順を使用すると、**early kdump** メカニズム、**early** kdump の設定、ステータスの確認を理解できます。

42.12.1. early kdump の概要

kdump サービスが起動していないと、起動段階でカーネルがクラッシュし、クラッシュしたカーネルメモリーの内容を取得して保存できません。そのため、トラブルシューティングの重要な情報は失われます。

この問題を対処するために、RHEL 8 では、**early kdump** 機能が **kdump** サービスの一部として導入されました。

42.12.2. early kdump の有効化

early kdump 機能は、初期クラッシュの **vmcore** 情報をキャプチャーするため、十分に早めにロードされるように、クラッシュカーネルと初期 RAM ディスクイメージ (**initramfs**) をセットアップします。これにより、初期のブートカーネルクラッシュに関する情報が失われるリスクを排除できます。

前提条件

- アクティブな RHEL サブスクリプション。
- システムの CPU アーキテクチャー用の **kexec-tools** パッケージを含むリポジトリー
- **kdump** の設定とターゲットの要件を満たしている

手順

1. **kdump** サービスが有効でアクティブであることを確認します。

```
# systemctl is-enabled kdump.service && systemctl is-active kdump.service enabled
active
```

kdump が有効ではなく、実行されていない場合は、必要な設定をすべて設定し、**kdump** サービスが有効化されていることを確認します。

2. 起動カーネルの **initramfs** イメージを、**early kdump** 機能で再構築します。

```
# dracut -f --add earlykdump
```

3. **rd.earlykdump** カーネルコマンドラインパラメーターを追加します。

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="rd.earlykdump"
```

4. システムを再起動して、変更を反映させます。

```
# reboot
```

検証手順

- **rd.earlykdump** が正常に追加され、**early kdump** 機能が有効になっていることを確認します。

```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-187.el8.x86_64 root=/dev/mapper/rhel-root ro
crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap
rhgb quiet rd.earlykdump

# journalctl -x | grep early-kdump
Mar 20 15:44:41 redhat dracut-cmdline[304]: early-kdump is enabled.
Mar 20 15:44:42 redhat dracut-cmdline[304]: kexec: loaded early-kdump kernel
```

関連情報

- **/usr/share/doc/kexec-tools/early-kdump-howto.txt** ファイル
- What is early kdump support and how do I configure it?

42.13. 関連情報

次のセクションでは、クラッシュ情報の取得について詳しく説明します。

- **kdump.conf**(5) - 利用できるオプションの完全なドキュメンテーションを含む **/etc/kdump.conf** 設定ファイルの man ページ。
- **zipl.conf**(5): **/etc/zipl.conf** 設定ファイルの man ページです。
- **zipl**(8) - IBM System z 用の **zipl** ブートローダーユーティリティーの man ページ。
- **makedumpfile**(8) - **makedumpfile** コアコレクターの man ページ。
- **kexec**(8) – **kexec** の man ページ。
- **crash**(8) – **crash** ユーティリティーの man ページ。
- **/usr/share/doc/kexec-tools/kexec-kdump-howto.txt** –**kdump** と **kexec** インストールと使用の概要。
- How to troubleshoot kernel crashes, hangs, or reboots with kdump on Red Hat Enterprise Linux
- kdump のターゲットとしてサポートされているもの

第43章 カーネルライブパッチでパッチの適用

Red Hat Enterprise Linux カーネルのライブパッチソリューションを使用して、システムの再起動またはプロセスの再起動を行わずに、実行中のカーネルにパッチを当てることができます。

このソリューションでは、システム管理者は以下を行うことができます。

- 重大なセキュリティーパッチをカーネルに即座に適用することが可能。
- 長時間実行しているタスクの完了、ユーザーのログオフ、スケジュールダウンタイムを待つ必要がない。
- システムのアップタイムをより制御し、セキュリティーや安定性を犠牲にしない。

重要な、重要なすべての CVE は、カーネルライブパッチソリューションで解決されるわけではありません。この目的は、セキュリティー関連パッチに必要な再起動を減らすことであり、完全になくすことではありません。ライブパッチの範囲の詳細は、**RHEL 7 はライブカーネルパッチ (kpatch) をサポートしていますか？**を参照してください。

警告

カーネルのライブマイグレーションパッチと、その他のカーネルサブコンポーネントとの間に、いくらか非互換性が存在します。以下を参照してください。

カーネルのライブパッチを使用する前に、kpatch の制限 を慎重に確認してください。

43.1. KPATCH の制限

- **kpatch** 機能は、汎用のカーネルアップグレードメカニズムではありません。システムをすぐに再起動できない場合など、単純なセキュリティーおよびバグ修正の更新を適用する場合に使用します。
- パッチの読み込み中または読み込み後は、**SystemTap** ツールまたは **kprobe** ツールを使用しないでください。このようなプローブが削除されるまでは、パッチが適用できなくなる可能性があります。

43.2. サードパーティーのライブパッチサポート

kpatch ユーティリティーは、Red Hat リポジトリー提供の RPM モジュールを含む、Red Hat がサポートする唯一のカーネルライブパッチユーティリティーです。Red Hat は、Red Hat 提供でないライブカーネルパッチはサポートしません。

サードパーティーのライブパッチで発生する問題に対応する必要がある場合、Red Hat では、原因発見を必要とする調査のアウトセットで、ライブパッチベンダーにケースを開くことを奨していますこれにより、ベンダーが許可すれば、ソースコードの供給が可能になり、Red Hat サポートに調査を依頼する前に、サポート組織への原因追及を支援することがになります。

サードパーティーのライブパッチを実行しているシステムの場合、Red Hat は、Red Hat が同梱し、サポートしているソフトウェアの複製を求める権利を有します。これが可能でない場合、Red Hat は、同

じ動作が発生するかどうかを確認するために、ライブパッチを適用せずに、お使いのテスト環境で同じようなシステムとワークロードのデプロイメントを求めます。

サードパーティーソフトウェアサポートポリシーの詳細は、Red Hat グローバルサポートサービスは、サードパーティーのソフトウェア、ドライバー、そして認定されていないハードウェアおよびハイパーバイザー、もしくはゲストのオペレーティングシステムについてどのようなサポートを提供していますか？を参照してください。

43.3. カーネルライブパッチへのアクセス

ライブのカーネルパッチ機能は、RPM パッケージとして提供されるカーネルモジュール (**kmod**) として実装されます。

すべてのお客様は、通常のチャンネルから提供されるカーネルライブパッチにアクセスできます。ただし、延長サポートサービスにサブスクライブしていないお客様は、次のマイナーリリースが利用可能になると、現行のマイナーリリースに対する新しいパッチへのアクセスを失うことになります。たとえば、標準のサブスクリプションを購入しているお客様は、RHEL 8.3 カーネルがリリースされるまで RHEL 8.2 カーネルのライブパッチのみを行うことができます。

43.4. カーネルライブパッチのコンポーネント

カーネルのライブパッチのコンポーネントは、以下のようになります。

カーネルパッチモジュール

- カーネルライブパッチの配信メカニズム
- パッチが適用されるカーネル用に構築したカーネルモジュール。
- パッチモジュールには、カーネルに必要な修正のコードが含まれます。
- パッチモジュールは、**kpatch** カーネルサブシステムで登録し、置き換えられるオリジナル機能の情報を提供します。また、置換される機能に一致するポインターも含まれます。カーネルパッチモジュールは RPM として提供されます。
- 命名規則は、**kpatch_<kernel version>_<kpatch version>_<kpatch release>** です。名前の kernel version 部分の **ドット** は、**アンダースコア** に置き換えます。

kpatch ユーティリティー

パッチモジュールを管理するためのコマンドラインユーティリティー。

kpatch サービス

multiuser.target で必要な **systemd** サービス。このターゲットは、システムの起動時にカーネルパッチをロードします。

kpatch-dnf パッケージ

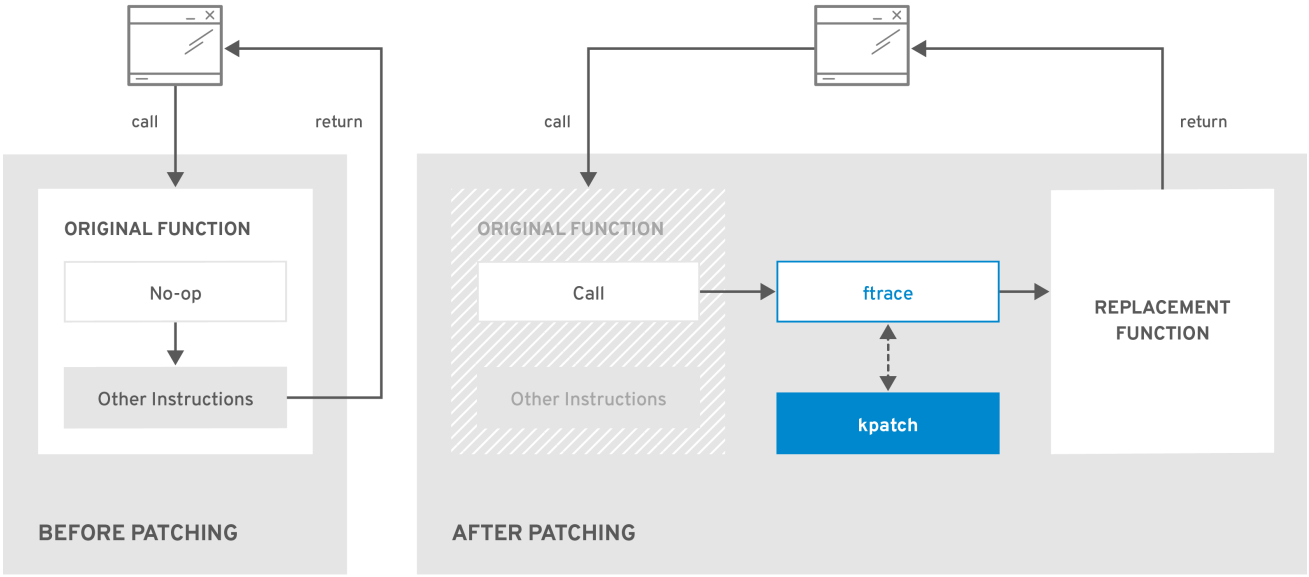
RPM パッケージ形式で配信される DNF プラグイン。このプラグインは、カーネルライブパッチへの自動サブスクリプションを管理します。

43.5. カーネルライブパッチの仕組み

kpatch カーネルパッチソリューションは、**livepatch** カーネルサブシステムを使用して、古い機能の出力先を新しい機能に変更します。ライブカーネルパッチがシステムに適用されると、以下が発生します。

1. カーネルパッチモジュールは、**/var/lib/kpatch/** ディレクトリーにコピーされ、次回の起動時に **systemd** を介して、カーネルへの再適用として登録されます。
2. 実行中のカーネルに kpatch モジュールがロードされ、新しいコードのメモリー内の場所を指定するポインターを使用して、新しい機能が **ftrace** メカニズムに登録されます。
3. パッチが当てられた機能にカーネルがアクセスすると、**ftrace** メカニズムにリダイレクトされます。これにより、元々の機能は回避され、パッチを当てたバージョンの機能にカーネルをリダイレクトします。

図43.1 カーネルライブパッチの仕組み



43.6. 現在インストールされているカーネルをライブパッチストリームにサブスクライブする手順

カーネルパッチモジュールは RPM パッケージに含まれ、パッチが適用されたカーネルバージョンに固有のものとなります。各 RPM パッケージは、徐々に蓄積されていきます。

以下の手順では、指定のカーネルに対して、今後の累積パッチ更新をすべてサブスクライブする方法を説明します。ライブパッチは累積的であるため、特定のカーネルにデプロイされている個々のパッチを選択できません。

警告

Red Hat は、Red Hat がサポートするシステムに適用されたサードパーティーのライブパッチをサポートしません。

前提条件

- root 権限

手順

1. 必要に応じて、カーネルバージョンを確認します。

```
# uname -r
4.18.0-94.el8.x86_64
```

2. カーネルのバージョンに一致するライブパッチパッケージを検索します。

```
# yum search $(uname -r)
```

3. ライブパッチパッケージをインストールします。

```
# yum install "kpatch-patch = $(uname -r)"
```

上記のコマンドでは、特定カーネルにのみに最新の累積パッチをインストールし、適用します。

ライブパッチパッケージのバージョンが 1-1 以上である場合には、パッケージにパッチモジュールが含まれます。この場合、ライブパッチパッケージのインストール時に、カーネルにパッチが自動的に適用されます。

カーネルパッチモジュールは、今後の再起動時に **systemd** システムおよびサービスマネージャーにより読み込まれる **/var/lib/kpatch/** ディレクトリーにもインストールされます。

注記

指定のカーネルに利用可能なライブパッチがない場合は、空のライブパッチパッケージがインストールされます。空のライブパッチパッケージには、kpatch_version-kpatch_release 0-0 (例: **kpatch-patch-4_18_0-94-0-0.el8.x86_64.rpm**) が含まれます。空の RPM のインストールを行うと、指定のカーネルの将来のすべてのライブパッチにシステムがサブスクライブされます。

4. 必要に応じて、カーネルがパッチを当てていることを確認します。

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_94_1_1 [enabled]

Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
…
```

この出力は、カーネルパッチモジュールがカーネルに読み込まれていることを示しています。つまり、カーネルは現在、**kpatch-patch-4_18_0-94-1-1.el8.x86_64.rpm** パッケージの最新の修正でパッチが当てられています。

関連情報

- **kpatch(1)** の man ページ
- RHEL の **基本的なシステム設定の設定**

43.7. ライブパッチストリームに新しいカーネルを自動的にサブスクライブする手順

kpatch-dnf YUM プラグインを使用して、カーネルパッチモジュール (別称 カーネルライブパッチ) が提供する修正にシステムをサブスクライブできます。このプラグインは、現在システムが使用するカーネルと、**今後インストールされる** カーネルの **自動** サブスクリプションを有効にします。

前提条件

- root 権限がある。

手順

1. 必要に応じて、インストール済みの全カーネルと、現在実行中のカーネルを確認します。

```
# yum list installed | grep kernel
Updating Subscription Management repositories.
Installed Packages
...
kernel-core.x86_64        4.18.0-240.10.1.el8_3        @rhel-8-for-x86_64-baseos-rpms
kernel-core.x86_64        4.18.0-240.15.1.el8_3        @rhel-8-for-x86_64-baseos-rpms
...

# uname -r
4.18.0-240.10.1.el8_3.x86_64
```

2. **kpatch-dnf** プラグインをインストールします。

```
# yum install kpatch-dnf
```

3. カーネルライブパッチの自動サブスクリプションを有効にします。

```
# yum kpatch auto
Updating Subscription Management repositories.
Last metadata expiration check: 19:10:26 ago on Wed 10 Mar 2021 04:08:06 PM CET.
Dependencies resolved.
================================================
 Package                    Architecture
================================================
Installing:
 kpatch-patch-4_18_0-240_10_1       x86_64
 kpatch-patch-4_18_0-240_15_1       x86_64

Transaction Summary
================================================
Install  2 Packages
…
```

このコマンドは、現在インストールされているすべてのカーネルをサブスクライブして、カーネルライブパッチを受け取ります。このコマンドは、インストールされている全カーネルに、最新の累積パッチ (存在する場合) をインストールして適用します。

今後、カーネルを更新すると、新しいカーネルのインストールプロセス中にライブパッチが自動的にインストールされます。

カーネルパッチモジュールは、今後の再起動時に **systemd** システムおよびサービスマネージャーにより読み込まれる **/var/lib/kpatch/** ディレクトリーにもインストールされます。

注記

指定のカーネルに利用可能なライブパッチがない場合は、空のライブパッチパッケージがインストールされます。空のライブパッチパッケージには、**kpatch_version-kpatch_release** 0-0 (例: **kpatch-patch-4_18_0-240-0-0.el8.x86_64.rpm**) が含まれます。空の RPM のインストールを行うと、指定のカーネルの将来のすべてのライブパッチにシステムがサブスクライブされます。

検証手順

- インストールされているすべてのカーネルにパッチが当てられていることを確認します。

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_240_10_1_0_1 [enabled]

Installed patch modules:
kpatch_4_18_0_240_10_1_0_1 (4.18.0-240.10.1.el8_3.x86_64)
kpatch_4_18_0_240_15_1_0_2 (4.18.0-240.15.1.el8_3.x86_64)
```

この出力から、実行中のカーネルとインストールされている他のカーネル両方に **kpatch-patch-4_18_0-240_10_1-0-1.rpm** と **kpatch-patch-4_18_0-240_15_1-0-1.rpm** パッケージそれぞれからの修正が適用されたことが分かります。

関連情報

- **kpatch(1)** および **dnf-kpatch(8)** の man ページ
- RHEL の **基本的なシステム設定の設定**

43.8. ライブパッチストリームへの自動サブスクリプションの無効化

カーネルパッチモジュールが提供する修正にシステムをサブスクライブする場合、サブスクリプションは **自動的** です。この機能 (したがって、**kpatch-patch** パッケージの自動インストール) を無効にできます。

前提条件

- root 権限がある。

手順

1. 必要に応じて、インストール済みの全カーネルと、現在実行中のカーネルを確認します。

```
# yum list installed | grep kernel
Updating Subscription Management repositories.
Installed Packages
...
kernel-core.x86_64          4.18.0-240.10.1.el8_3          @rhel-8-for-x86_64-baseos-rpms
kernel-core.x86_64          4.18.0-240.15.1.el8_3          @rhel-8-for-x86_64-baseos-rpms
...
```

```
# uname -r
4.18.0-240.10.1.el8_3.x86_64
```

2. カーネルライブパッチへの自動サブスクリプションを無効にします。

```
# yum kpatch manual
Updating Subscription Management repositories.
```

検証手順

- 成功した出力を確認できます。

```
# yum kpatch status
...
Updating Subscription Management repositories.
Last metadata expiration check: 0:30:41 ago on Tue Jun 14 15:59:26 2022.
Kpatch update setting: manual
```

関連情報

- **kpatch(1)** および **dnf-kpatch(8)** の man ページ

43.9. カーネルパッチモジュールの更新

カーネルパッチモジュールが配信され、RPM パッケージを通じて適用されているため、累計のカーネルパッチモジュール更新は、他の RPM パッケージの更新と似ています。

前提条件

- 現在インストールされているカーネルをライブパッチストリームにサブスクライブする手順 に従って、システムがライブパッチストリームにサブスクライブされている。

手順

- 現在のカーネルの新しい累積バージョンを更新します。

```
# yum update "kpatch-patch = $(uname -r)"
```

上記のコマンドは、現在実行中のカーネルに利用可能な更新を自動的にインストールし、適用します。これには、新たにリリースされた累計なライブパッチが含まれます。

- もしくは、インストールしたすべてのカーネルパッチモジュールを更新します。

```
# yum update "kpatch-patch"
```

注記

システムが同じカーネルで再起動すると、**kpatch.service** systemd サービスにより、カーネルが自動的に再適用されます。

関連情報

- RHEL の **基本的なシステム設定の設定**

43.10. ライブパッチパッケージの削除

ライブパッチパッケージを削除して、Red Hat Enterprise Linux カーネルライブパッチソリューションを無効にします。

前提条件

- root 権限
- ライブパッチパッケージがインストールされている。

手順

1. ライブパッチパッケージを選択します。

```
# yum list installed | grep kpatch-patch
kpatch-patch-4_18_0-94.x86_64        1-1.el8        @@commandline
…
```

上記の出力例は、インストールしたライブパッチパッケージをリスト表示します。

2. ライブパッチパッケージを削除します。

```
# yum remove kpatch-patch-4_18_0-94.x86_64
```

ライブパッチパッケージが削除されると、カーネルは次回の再起動までパッチが当てられたままになりますが、カーネルパッチモジュールはディスクから削除されます。今後の再起動では、対応するカーネルにはパッチが適用されなくなります。

3. システムを再起動します。

4. ライブパッチパッケージが削除されたことを確認します。

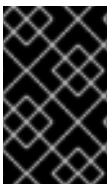
```
# yum list installed | grep kpatch-patch
```

パッケージが正常に削除された場合、このコマンドでは何も出力されません。

5. 必要に応じて、カーネルのライブパッチソリューションが無効になっていることを確認します。

```
# kpatch list
Loaded patch modules:
```

この出力例では、現在読み込まれているパッチモジュールがないため、カーネルにパッチが適用されておらず、ライブパッチソリューションがアクティブでないことが示されています。



重要

現在、Red Hat はシステムの再起動なしで、ライブパッチを元に戻すことはサポートしていません。ご不明な点がございましたら、サポートチームまでお問い合わせください。

関連情報

- **kpatch(1)** の man ページ
- RHEL の **基本的なシステム設定の設定**

43.11. カーネルパッチモジュールのアンインストール

Red Hat Enterprise Linux カーネルライブパッチソリューションが、以降の起動時にカーネルパッチモジュールを適用しないようにします。

前提条件

- root 権限がある。
- ライブパッチパッケージがインストールされている。
- カーネルパッチモジュールがインストールされ、ロードされている。

手順

1. カーネルパッチモジュールを選択します。

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_94_1_1 [enabled]

Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
…
```

2. 選択したカーネルパッチモジュールをアンインストールします。

```
# kpatch uninstall kpatch_4_18_0_94_1_1
uninstalling kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

   - アンインストールしたカーネルモジュールが読み込まれていることに注意してください。

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_94_1_1 [enabled]

Installed patch modules:
<NO_RESULT>
```

   選択したモジュールをアンインストールすると、カーネルは次回の再起動までパッチが当てられますが、カーネルパッチモジュールはディスクから削除されます。

3. システムを再起動します。
4. 必要に応じて、カーネルパッチモジュールがアンインストールされていることを確認します。

```
# kpatch list
Loaded patch modules:
…
```

上記の出力例では、ロードまたはインストールされたカーネルパッチモジュールが表示されていません。したがって、カーネルにパッチが適用されておらず、カーネルのライブパッチソリューションはアクティブではありません。

重要

現在、Red Hat はシステムの再起動なしで、ライブパッチを元に戻すことはサポートしていません。ご不明な点がございましたら、サポートチームまでお問い合わせください。

関連情報

- **kpatch(1)** の man ページ

43.12. KPATCH.SERVICE の無効化

Red Hat Enterprise Linux カーネルライブパッチソリューションが、以降の起動時にすべてのカーネルパッチモジュールをシステム全体に適用しないようにします。

前提条件

- root 権限がある。
- ライブパッチパッケージがインストールされている。
- カーネルパッチモジュールがインストールされ、ロードされている。

手順

1. **kpatch.service** が有効化されていることを確認します。

```
# systemctl is-enabled kpatch.service
enabled
```

2. **kpatch.service** を無効にします。

```
# systemctl disable kpatch.service
Removed /etc/systemd/system/multi-user.target.wants/kpatch.service.
```

   - 適用されたカーネルモジュールが依然としてロードされていることに注意してください。

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_94_1_1 [enabled]

Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

3. システムを再起動します。
4. 必要に応じて、**kpatch.service** のステータスを確認します。

```
# systemctl status kpatch.service
● kpatch.service - "Apply kpatch kernel patches"
  Loaded: loaded (/usr/lib/systemd/system/kpatch.service; disabled; vendor preset: disabled)
  Active: inactive (dead)
```

この出力テストサンプルでは、**kpatch.service** が無効になっており、実行されていないことを証明しています。したがって、カーネルのライブパッチソリューションはアクティブではありません。

5. カーネルパッチモジュールがアンロードされたことを確認します。

```
# kpatch list
Loaded patch modules:
<NO_RESULT>

Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

上記の出力例では、カーネルパッチモジュールがインストールされていても、カーネルにパッチが適用されていないことを示しています。

重要

現在、Red Hat はシステムの再起動なしで、ライブパッチを元に戻すことはサポートしていません。ご不明な点がございましたら、サポートチームまでお問い合わせください。

関連情報

- **kpatch(1)** の man ページ
- RHEL の **基本的なシステム設定の設定**

第44章 アプリケーションの制限の設定

コントロールグループ (**cgroup**) のカーネル機能を使用して、プロセスのハードウェアリソースの制限や優先順位を設定したりまたは分離できます。これにより、アプリケーションのリソース使用状況をより詳細に制御して、効率的に使用できます。

44.1. コントロールグループについて

コントロールグループは、プロセスを階層的に順序付けされた **cgroups** グループに編成できる Linux カーネル機能です。階層 (コントロールグループツリー) は、**cgroups** 仮想ファイルシステムに構造を提供して定義されます。デフォルトでは **/sys/fs/cgroup** ディレクトリーにマウントされます。**systemd** システムとサービスマネージャーは、**cgroups** を利用して、それが管理するすべてのユニットとサービスを組織化します。または、**/sys/fs/cgroup/** ディレクトリーでサブディレクトリーを作成および削除することにより、**cgroups** 階層を手動で管理できます。

リソースコントローラー (カーネルコンポーネント) は、これらのプロセスのシステムリソース (CPU 時間、メモリー、ネットワーク帯域幅、各種組み合わせなど) を制限、優先順位付け、または割り当てることで **cgroup** 内のプロセスの動作を変更します。

cgroups に追加された値はプロセスアグリゲーションで、アプリケーションとユーザー間のハードウェアリソースの分割を可能にします。その結果、ユーザー環境の全体的な効率、安定性、およびセキュリティーが向上します。

コントロールグループ 1

コントロールグループバージョン 1 (**cgroups-v1**) はリソースごとのコントローラー階層を提供します。これは、CPU、メモリー、I/O などの各リソースに、独自のコントロールグループ階層があることを意味します。あるコントローラーが、各リソースの管理において別のものと連携できるように、別のコントロールグループ階層を組み合わせることができます。ただし、2 つのコントローラーは異なるプロセス階層に属する可能性があり、適切な調整はできません。
cgroups-v1 コントローラーは、長期間に渡って開発されているため、制御ファイルの動作と命名は均一ではありません。

コントロールグループ 2

階層の柔軟性から生じるコントローラー連携の問題は、**control groups version 2** の発展につながっていました。
Control groups version 2 (**cgroups-v2**) では、すべてのリソースコントロールがマウントされることに対して単一のコントロールグループ階層を指定します。

コントロールファイルの動作と命名は、さまざまなコントローラーにおいて一貫性があります。

注記

cgroups-v2 は、RHEL 8.2 以降のバージョンで完全にサポートされています。詳細は Control Group v2 is now fully supported in RHEL 8 を参照してください。

このサブセクションは、Devconf.cz 2019 プレゼンテーションにを基にしています。[4]

関連情報

- Linux カーネルリソースコントローラーとは
- **cgroups(7)** の man ページ

- コントロールグループ内の systemd のロール

44.2. LINUX カーネルリソースコントローラーとは

コントロールグループの機能は、カーネルリソースコントローラーで有効化します。RHEL 8 は、**コントロールグループバージョン 1** (**cgroups-v1**) および **コントロールグループバージョン 2** (**cgroups-v2**) のさまざまなコントローラーをサポートします。

コントロールグループサブシステムとも呼ばれるリソースコントローラーは、1 つのリソース (CPU 時間、メモリー、ネットワーク帯域幅、ディスク I/O など) を表すカーネルサブシステムです。Linux カーネルは、**systemd** システムおよびサービスマネージャーが自動的にマウントされるリソースコントローラーの範囲を指定します。**/proc/cgroups** エントリーで現在マウントされているリソースコントローラーのリストを検索します。

cgroups-v1 では、以下のコントローラーを使用できます。

- **blkio** - ブロックデバイスへの入出力アクセスに制限を設定できます。
- **cpu** - コントロールグループのタスクに対して、Completely Fair Scheduler (CFS) スケジューラーのパラメーターを調整できます。これは、同じマウントで **cpuacct** コントローラーとともにマウントされます。
- **cpuacct** - コントロールグループ内のタスクが使用する CPU リソースに関する自動レポートを作成します。これは、同じマウント上の **cpu** コントローラーとともにマウントされます。
- **cpuset** - コントロールグループタスクが CPU の特定のサブセットでのみ実行されるように制限したり、指定メモリーノードでのみメモリーを使用できるようにタスクに指示したりできます。
- **デバイス** - コントロールグループのタスクに関してデバイスへのアクセスを制御できます。
- **freezer** - コントロールグループのタスクを一時停止または再開するのに使用できます。
- **memory** - コントロールグループ内のタスクでメモリー使用を設定し、そのタスクによって使用されるメモリーリソースに関する自動レポートを生成するのに使用できます。
- **net_cls** - 特定のコントロールグループタスクから発信されたパケットを識別できるようにするために Linux トラフィックコントローラー (**tc** コマンド) を有効にするクラス識別子 (**classic**) でネットワークパケットをタグ付けします。**net_cls** のサブシステム **net_ filter** (iptables) でも、このタグを使用して、そのようなパケットに対するアクションを実行することができます。**net_filter** は、ファイアウォール識別子 (**fwid**) でネットワークソケットをタグ付けします。これにより、(**iptables** コマンドで) Linux ファイアウォールが、特定のコントールグループタスクから発信されたパケットを識別できるようになります。
- **net_prio** - ネットワークトラフィックの優先度を設定します。
- **pids** - コントロールグループ内の多数のプロセスと子に制限を設定できます。
- **perf_event** - **perf** パフォーマンス監視およびレポートユーティリティーにより、監視するタスクをグループ化できます。
- **rdma** - コントロールグループ内のリモートダイレクトメモリーアクセス/InfiniB 固有のリソースに制限を設定できます。
- **hugetlb** - コントロールグループ内のタスクで大容量の仮想メモリーページの使用を制限するのに使用できます。

cgroups-v2 では、次のモードを使用できます。

- **io** - **cgroups-v1** の **blkio** へのフォローアップ
- **memory** - **cgroups-v1** の **メモリー** へのフォローアップ
- **pids** - **cgroups-v1** の **pids** と同じ
- **RDMA** - **cgroups-v1** の **rdma** と同じ
- **cpu - cpu** - **cgroups-v1** の **cpu** と **cpuacct** へのフォローアップ
- **cpuset** - コア機能 (**cpus{,.effective}**, **mems{,.effective}**) のみを新しいパーティション機能でサポートします。
- **perf_event** - サポートは継承され、明示的な制御ファイルはありません。**perf** コマンドに **v2 cgroup** をパラメーターとして指定でき、対象の **cgroup** 内のタスクすべてがプロファイリングされます。

重要

リソースコントローラーは、**cgroups-v1** 階層または **cgroups-v 2** 階層のいずれかで使用できますが、両方を同時に使用することはできません。

関連情報

- **cgroups(7)** の man ページ
- **/usr/share/doc/kernel-doc-<kernel_version>/Documentation/cgroups-v1/** ディレクトリー内のドキュメント (**kernel-doc** パッケージをインストールした後)。

44.3. NAMESPACE とは

名前空間は、ソフトウェアオブジェクトを整理および特定するための最も重要な方法の 1 つです。

名前空間は、グローバルシステムリソース (マウントポイント、ネットワークデバイス、ホスト名など) を抽象化してラップすることで、グローバルリソースごとに分離されたインスタンスを含む対象の名前空間にプロセスを表示させることができます。名前空間を使用する最も一般的なテクノロジーの 1 つとしてコンテナーが挙げられます。

特定のグローバルリソースへの変更は、その名前空間のプロセスにのみ表示され、残りのシステムまたは他の名前空間には影響しません。

プロセスがどの名前空間に所属するかを確認するには、**/proc/<PID>/ns/** ディレクトリーのシンボリックリンクを確認します。

以下の表は、分離されるサポート対象の名前空間およびリソースを示しています。

名前空間	分離
Mount	マウントポイント
UTS	ホスト名および NIS ドメイン名

名前空間	分離
IPC	System V IPC、POSIX メッセージキュー
PID	プロセス ID
Network	ネットワークデバイス、スタック、ポートなど。
User	ユーザーおよびグループ ID
Control groups	コントロールグループの root ディレクトリー

関連情報

- **namespaces(7)** および **cgroup_namespaces(7)** man ページ
- コントロールグループについて

44.4. CGROUPS-V1 を使用したアプリケーションへの CPU 制限の設定

アプリケーションが CPU 時間を過剰に消費すると、環境の全体的な健全性に悪影響を及ぼす可能性があります。**/sys/fs/** 仮想ファイルシステムを使用して、**コントロールグループバージョン 1** (**cgroups-v1**) を使用してアプリケーションへの CPU 制限を設定します。

前提条件

- root 権限がある。
- CPU 消費を制限するアプリケーションがある。
- **cgroups-v1** コントローラーがマウントされていることを確認している。

```
# mount -l | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-
cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,cpu,cpuacct)
cgroup on /sys/fs/cgroup/perf_event type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,pids)
...
```

手順

1. CPU 消費を制限するアプリケーションのプロセス ID (PID) を特定します。

```
# top
top - 11:34:09 up 11 min,  1 user,  load average: 0.51, 0.27, 0.22
Tasks: 267 total,   3 running, 264 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s): 49.0 us,  3.3 sy,  0.0 ni, 47.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.0 st
MiB Mem :   1826.8 total,    303.4 free,   1046.8 used,    476.5 buff/cache
MiB Swap:   1536.0 total,   1396.0 free,    140.0 used.    616.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  99.3   0.1   0:32.71 sha1sum
 5760 jdoe      20   0 3603868 205188  64196 S   3.7  11.0   0:17.19 gnome-shell
 6448 jdoe      20   0  743648  30640  19488 S   0.7   1.6   0:02.73 gnome-terminal-
    1 root      20   0  245300   6568   4116 S   0.3   0.4   0:01.87 systemd
  505 root      20   0       0      0      0 I   0.3   0.0   0:00.75 kworker/u4:4-events_unbound
...
```

top プログラムの出力例は、**PID 6955** (具体例のアプリケーション **sha1sum**) が CPU リソースを大量に消費することを示しています。

2. **cpu** リソースコントローラーディレクトリーにサブディレクトリーを作成します。

```
# mkdir /sys/fs/cgroup/cpu/Example/
```

上記のディレクトリーは、特定のプロセスを配置でき、特定の CPU 制限をプロセスに適用できるコントロールグループです。同時に、一部の **cgroups-v1** インターフェイスファイルと **cpu** コントローラー固有のファイルがディレクトリーに作成されます。

3. 必要に応じて、新しく作成されたコントロールグループを確認します。

```
# ll /sys/fs/cgroup/cpu/Example/
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.clone_children
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.procs
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_all
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_user
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_user
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_quota_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_runtime_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.shares
-r—r—r--. 1 root root 0 Mar 11 11:42 cpu.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 notify_on_release
-rw-r—r--. 1 root root 0 Mar 11 11:42 tasks
```

この出力例は、特定の設定や制限を表す **cpuacct.usage**、**cpu.cfs._period_us** などのファイルを示しています。これは、**Example** コントロールグループのプロセスに設定できます。各ファイル名の前に、そのファイルが属するコントロールグループコントローラーの名前が接頭辞として追加されることに注意してください。

デフォルトでは、新しく作成されたコントロールグループは、システムの CPU リソース全体へのアクセスを制限なしで継承します。

4. コントロールグループの CPU 制限を設定します。

```
# echo "1000000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
# echo "200000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
```

cpu.cfs_period_us ファイルは、制御グループの CPU リソースへのアクセスが再割り当てされる頻度について、マイクロ秒単位の期間 (ここでは us と表示されますが μs) を表します。上限は 1 秒で、下限は 1000 マイクロ秒です。

cpu.cfs_quota_us ファイルは、(**cpu.cfs_period_us** で定義されるように) コントロールグループのすべてのプロセスを 1 期間中に実行できる合計時間をマイクロ秒単位で表します。1 期間中にコントロールグループ内のプロセスがクォータで指定された時間をすべて使いきるとすぐに、残りの期間はスロットルされて、次の期間まで実行できなくなります。下限は 1000 マイクロ秒です。

上記のコマンド例は、CPU 時間制限を設定して、**Example** コントロールグループでまとめられているすべてのプロセスは、1 秒ごと (**cpu.cfs_period_us** に定義されている) に、0.2 秒間だけ (**cpu.cfs_quota_us** に定義されている) 実行できるようにしています。

5. 必要に応じて、制限を確認します。

```
# cat /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
/sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
1000000
200000
```

6. アプリケーションの PID を **Example** コントロールグループに追加します。

```
# echo "6955" > /sys/fs/cgroup/cpu/Example/cgroup.procs

or

# echo "6955" > /sys/fs/cgroup/cpu/Example/tasks
```

上記のコマンドは、目的のアプリケーションが **Example** コントロールグループのメンバーとなるようするので、**Example** コントロールグループに設定された CPU 制限は超えません。PID は、システム内の既存のプロセスを表します。ここで **PID 6955** は、プロセス **sha1sum /dev/zero &** に割り当てられ、**cpu** コントローラーの使用例を示すために使用されました。

7. アプリケーションが指定のコントロールグループで実行されていることを確認します。

```
# cat /proc/6955/cgroup
12:cpuset:/
11:hugetlb:/
10:net_cls,net_prio:/
9:memory:/user.slice/user-1000.slice/user@1000.service
8:devices:/user.slice
7:blkio:/
6:freezer:/
5:rdma:/
4:pids:/user.slice/user-1000.slice/user@1000.service
3:perf_event:/
2:cpu,cpuacct:/Example
1:name=systemd:/user.slice/user-1000.slice/user@1000.service/gnome-terminal-
server.service
```

上記の出力例は、目的のアプリケーションのプロセスが **Example** のコントロールグループで実行され、アプリケーションのプロセスに CPU 制限が適用されることを示しています。

8. スロットルしたアプリケーションの現在の CPU 使用率を特定します。

```
# top
top - 12:28:42 up  1:06,  1 user,  load average: 1.02, 1.02, 1.00
Tasks: 266 total,   6 running, 260 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11.0 us,  1.2 sy,  0.0 ni, 87.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.2 st
MiB Mem :   1826.8 total,    287.1 free,   1054.4 used,    485.3 buff/cache
MiB Swap:   1536.0 total,   1396.7 free,    139.2 used.    608.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  20.6   0.1  47:11.43 sha1sum
 5760 jdoe      20   0 3604956 208832  65316 R   2.3  11.2   0:43.50 gnome-shell
 6448 jdoe      20   0  743836  31736  19488 S   0.7   1.7   0:08.25 gnome-terminal-
  505 root      20   0       0      0      0 I   0.3   0.0   0:03.39 kworker/u4:4-events_unbound
 4217 root      20   0   74192   1612   1320 S   0.3   0.1   0:01.19 spice-vdagentd
...
```

PID 6955 の CPU 消費が 99% から 20% に減少していることに注意してください。

重要

cpu.cfs_period_us および **cpu.cfs_quota_us** に対応する **cgroups-v2** は、**cpu.max** ファイルです。**cpu.max** ファイルは、**cpu** コントローラーから入手できます。

関連情報

- コントロールグループについて
- Linux カーネルリソースコントローラーとは
- **cgroups(7)**、**sysfs(5)** の man ページ

[4] Linux Control Group v2 - An Introduction, Devconf.cz 2019 presentation by Waiman Long

第45章 BPF コンパイラーコレクションでシステムパフォーマンスの分析

システム管理者として BPF コンパイラーコレクション (BCC) ライブラリーで Linux オペレーティングシステムのパフォーマンスを分析するツールを作成します。ただし、他のインターフェイス経由での取得は困難な場合があります。

45.1. BCC-TOOLS パッケージのインストール

bcc-tools パッケージをインストールします。これにより、依存関係として BPF Compiler Collection (BCC) ライブラリーもインストールされます。

手順

1. **bcc-tools** をインストールします。

```
# yum install bcc-tools
```

   BCC ツールは、**/usr/share/bcc/tools/** ディレクトリーにインストールされます。

2. 必要に応じて、ツールを検証します。

```
# ll /usr/share/bcc/tools/
...
-rwxr-xr-x. 1 root root  4198 Dec 14 17:53 dcsnoop
-rwxr-xr-x. 1 root root  3931 Dec 14 17:53 dcstat
-rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
-rw-r--r--. 1 root root  7105 Dec 14 17:53 deadlock_detector.c
drwxr-xr-x. 3 root root  8192 Mar 11 10:28 doc
-rwxr-xr-x. 1 root root  7588 Dec 14 17:53 execsnoop
-rwxr-xr-x. 1 root root  6373 Dec 14 17:53 ext4dist
-rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
...
```

   上記のリストにある **doc** ディレクトリーには、各ツールのドキュメントが含まれます。

45.2. BCC-TOOLS でパフォーマンスの分析

BPF Compiler Collection (BCC) ライブラリーから事前に作成された特定のプログラムを使用して、システムパフォーマンスをイベントごとに効率的かつセキュアに分析します。BCC ライブラリーで事前作成されたプログラムセットは、追加プログラム作成の例として使用できます。

前提条件

- bcc-tools パッケージがインストールされている
- root 権限がある。

execsnoop を使用したシステムプロセスの検証

1. 1 つの端末で **execsnoop** プログラムを実行します。

```
# /usr/share/bcc/tools/execsnoop
```

2. たとえば、別のターミナルで次のように実行します。

```
$ ls /usr/share/bcc/tools/doc/
```

これにより、**ls** コマンドの短命プロセスが作成されます。

3. **execsnoop** を実行している端末は、以下のような出力を表示します。

```
PCOMM PID    PPID   RET ARGS
ls    8382   8287     0 /usr/bin/ls --color=auto /usr/share/bcc/tools/doc/
...
```

execsnoop プログラムは、新しいプロセスごとに出力行を出力するため、システムリソースを消費します。また、**ls** などの非常に短期間に実行されるプログラムのプロセスを検出します。なお、ほとんどの監視ツールはそれらを登録しません。

execsnoop 出力には以下のフィールドが表示されます。

- PCOMM - 親プロセス名。(**ls**)
- PID - プロセス ID(**8382**)
- ppid: 親プロセス ID。(**8287**)
- RET - **exec()** のシステム呼び出しの戻り値 (**0**)。プログラムコードを新規プロセスに読み込みます。
- ARGS - 引数を使用して開始したプログラムの場所。

execsnoop の詳細、例、およびオプションを確認するには、**/usr/share/bcc/tools/doc/execsnoop_example.txt** ファイルを参照してください。

exec() の詳細は、**exec(3)** man ページを参照してください。

opensnoop を使用した、コマンドにより開かれるファイルの追跡

1. 1 つのターミナルで **opensnoop** プログラムを実行します。

```
# /usr/share/bcc/tools/opensnoop -n uname
```

上記の出力では、**uname** コマンドのプロセスによってのみ開かれるファイルの内容が出力されます。

2. 別のターミナルで、次のように実行します。

```
$ uname
```

上記のコマンドは、特定のファイルを開きます。このファイルは次のステップでキャプチャーされます。

3. **opensnoop** を実行している端末は、以下のような出力を表示します。

```
PID    COMM  FD ERR PATH
8596   uname  3  0   /etc/ld.so.cache
8596   uname  3  0   /lib64/libc.so.6
```

```
8596   uname  3  0   /usr/lib/locale/locale-archive
...
```

opensnoop プログラムは、システム全体で **open()** システム呼び出しを監視し、**uname** が開こうとしたファイルごとに出力行を出力します。

opensnoop 出力には、以下のフィールドが表示されます。

- **PID** - プロセス ID(**8596**)
- **COMM** - プロセス名 (**uname**)
- **FD** - ファイルの記述子。開いたファイルを参照するために **open()** が返す値。(**3**)
- **ERR** - すべてのエラー
- **PATH** - **open()** で開こうとしたファイルの場所。
  コマンドが、存在しないファイルを読み込もうとすると、**FD** コラムは **-1** を返し、**ERR** コラムは関連するエラーに対応する値を出力します。その結果、**opennoop** は、適切に動作しないアプリケーションの特定に役立ちます。

opensnoop の詳細、例、およびオプションを確認するには、**/usr/share/bcc/tools/doc/opensnoop_example.txt** ファイルを参照してください。

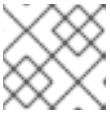
open() の詳細は、**open(2)** man ページを参照してください。

ディスク上の I/O 操作を調べるための biotop の使用

1. 1 つのターミナルで **biotop** プログラムを実行します。

```
# /usr/share/bcc/tools/biotop 30
```

このコマンドにより、ディスク上で I/O 操作を実行する上位のプロセスを監視できます。この引数は、コマンドが 30 秒の概要を生成するようにします。



注記

引数を指定しないと、デフォルトでは 1 秒ごとに出力画面が更新されます。

2. 別の端末で、たとえば次のように実行します。

```
# dd if=/dev/vda of=/dev/zero
```

上記のコマンドは、ローカルのハードディスクデバイスからコンテンツを読み込み、出力を **/dev/zero** ファイルに書き込みます。この手順では、**biotop** を示す特定の I/O トラフィックを生成します。

3. **biotop** を実行している端末は、以下のような出力を表示します。

```
PID    COMM             D MAJ MIN DISK       I/O  Kbytes     AVGms
9568   dd               R 252 0   vda      16294 14440636.0  3.69
48     kswapd0          W 252 0   vda       1763 120696.0    1.65
7571   gnome-shell      R 252 0   vda        834 83612.0     0.33
1891   gnome-shell      R 252 0   vda       1379 19792.0     0.15
7515   Xorg             R 252 0   vda        280  9940.0     0.28
```

```
7579  llvmpipe-1      R 252 0   vda       228  6928.0     0.19
9515  gnome-control-c  R 252 0   vda        62  6444.0     0.43
8112  gnome-terminal-  R 252 0   vda        67  2572.0     1.54
7807  gnome-software   R 252 0   vda        31  2336.0     0.73
9578  awk             R 252 0   vda        17  2228.0     0.66
7578  llvmpipe-0      R 252 0   vda       156  2204.0     0.07
9581  pgrep           R 252 0   vda        58  1748.0     0.42
7531  InputThread     R 252 0   vda        30  1200.0     0.48
7504  gdbus           R 252 0   vda         3  1164.0     0.30
1983  llvmpipe-1      R 252 0   vda        39   724.0     0.08
1982  llvmpipe-0      R 252 0   vda        36   652.0     0.06
...
```

biotop 出力には、以下のフィールドが表示されます。

- **PID** - プロセス ID(**9568**)
- **COMM** - プロセス名。(**dd**)
- **DISK** - 読み取り操作を実行するディスク。(**vda**)
- **I/O** - 実行された読み取り操作の数。(16294)
- **kbytes** - 読み取り操作によって使用したバイト数 (KB)。(14,440,636)
- **AVGms** - 読み取り操作の平均 I/O 時間。(3.69)

biotop の詳細、例、およびオプションを確認するには、**/usr/share/bcc/tools/doc/biotop_example.txt** ファイルを参照してください。

dd の詳細は、**dd(1)** man ページを参照してください。

xfsslower を使用した、予想外に遅いファイルシステム動作の明確化

1. 1 つのターミナルで **xfsslower** プログラムを実行します。

```
# /usr/share/bcc/tools/xfsslower 1
```

上記のコマンドは、XFS ファイルシステムが、読み込み、書き込み、開く、または同期 (**fsync**) 操作を実行するのに費やした時間を測定します。**1** 引数を指定すると、1 ms よりも遅い操作のみが表示されます。

注記

引数を指定しないと、**xfsslower** はデフォルトで 10 ms よりも低速な操作を表示します。

2. 別のターミナルで、たとえば次のように入力します。

```
$ vim text
```

上記のコマンドは、**vim** エディターでテキストファイルを作成し、XFS ファイルシステムと特定の対話を開始します。

3. **xfsslower** を実行している端末は、前の手順でファイルを保存した場合と同様の内容を示しています。

```
TIME     COMM          PID    T BYTES   OFF_KB   LAT(ms) FILENAME
13:07:14 b'bash'       4754   R 256     0          7.11 b'vim'
13:07:14 b'vim'        4754   R 832     0          4.03 b'libgpm.so.2.1.0'
13:07:14 b'vim'        4754   R 32      20         1.04 b'libgpm.so.2.1.0'
13:07:14 b'vim'        4754   R 1982    0          2.30 b'vimrc'
13:07:14 b'vim'        4754   R 1393    0          2.52 b'getscriptPlugin.vim'
13:07:45 b'vim'        4754   S 0       0          6.71 b'text'
13:07:45 b'pool'       2588   R 16      0          5.58 b'text'
...
```

上記の各行はファイルシステム内の操作を表し、特定のしきい値よりも時間がかかります。**xfsslower** は、ファイルシステムの問題の明確化に適しており、動作が予期せずに低速になります。

xfsslower 出力には、以下のフィールドが表示されます。

- COMM - プロセス名。(**b'bash'**)
- T - 操作の種類。(**R**)
  - Read
  - Write
  - Sync
- OFF_KB - ファイルオフセット (KB)。(0)
- FILENAME - 読み取り、書き込み、または同期するファイルです。

xfsslower の詳細、例、およびオプションについては、**/usr/share/bcc/tools/doc/xfsslower_example.txt** ファイルを参照してください。

fsync の詳細は、**fsync(2)** の man ページを参照してください。

パート VII. 高可用性システムの設計

第46章 HIGH AVAILABILITY ADD-ON の概要

High Availability Add-On は、基幹実稼働サービスに、信頼性、スケーラビリティー、および可用性を提供するクラスターシステムです。

クラスターは、連携してタスクを実行する 2 つ以上のコンピューター (**ノード** または **メンバー** と呼ばれています) を指します。クラスターを使用すると、可用性の高いサービスまたはリソースを提供できます。複数のマシンによ r 冗長性は、様々な障害から保護するために使用されます。

高可用性クラスターは、単一障害点を排除し、ノードが稼働しなくなった場合に、あるクラスターノードから別のクラスターノードにサービスをフェイルオーバーして、可用性が高いサービスを提供します。通常、高可用性クラスターのサービスは、(read-write でマウントされたファイルシステム経由で) データの読み取りや書き込みを行います。したがって、あるクラスターノードが別のクラスターノードからサービスの制御を引き継ぐ際に、高可能性クラスターでデータ整合性を維持する必要があります。高可用性クラスター内のノードの障害は、クラスター外にあるクライアントからは確認できません。また、高可用性クラスターはフェイルオーバークラスターと呼ばれることがあります。 High Availability Add-On は、高可用性サービス管理コンポーネントの **Pacemaker** を介して、高可用性クラスタリングを提供します。

46.1. HIGH AVAILABILITY ADD-ON コンポーネント

Red Hat High Availability Add-On は、高可用性サービスを提供する複数のコンポーネントで構成されます。

High Availability Add-On の主なコンポーネントは以下のとおりです。

- クラスターインフラストラクチャー - クラスターとして連携するように、ノード群に基本的な機能 (設定ファイル管理、メンバーシップ管理、ロック管理、およびフェンシング) を提供します。
- 高可用性サービス管理 - 1 つのクラスターノードが動作不能になった場合は、そのクラスターノードから別のノードにサービスのフェイルオーバーを提供します。
- クラスター管理ツール - High Availability Add-On のセットアップ、設定、および管理を行うツール。このツールは、クラスターインフラストラクチャーのコンポーネント、高可用性およびサービス管理のコンポーネント、ならびにストレージで使用されます。

以下のコンポーネントで、High Availability Add-On を補完できます。

- Red Hat GFS2 (Global File System 2) - Resilient Storage Add-On に同梱され、High Availability Add-On で使用するクラスターファイルシステムを提供します。GFS2 により、ストレージがローカルで各クラスターノードに接続されているかのように、ブロックレベルにおいて、複数ノードでストレージを共有できるようになります。GFS2 クラスターファイルシステムを使用する場合は、クラスターインフラストラクチャーが必要になります。
- LVM ロッキングデーモン (**lvmlockd**) - Resilient Storage Add-On に同梱され、クラスターストレージのボリューム管理を提供します。**lvmlockd** に対応するには、クラスターインフラストラクチャーも必要になります。
- HAProxy - レイヤー 4 (TCP) およびレイヤー 7 (HTTP および HTTPS) サービスで高可用性負荷分散とフェイルオーバーを提供するルーティングソフトウェアです。

46.2. HIGH AVAILABILITY ADD-ON の概念

Red Hat High Availability Add-On クラスターの主要な概念を以下に示します。

46.2.1. フェンシング

クラスター内のノードの1つと通信が失敗した場合に、障害が発生したクラスターノードがアクセスする可能性があるリソースへのアクセスを、その他のノードが制限したり、解放したりできるようにする必要があります。クラスターノードが応答しない可能性があるため、そのクラスターノードと通信しても成功しません。代わりに、フェンスエージェントを使用した、フェンシングと呼ばれる外部メソッドを指定する必要があります。フェンスデバイスは、クラスターが使用する外部デバイスのことで、このデバイスを使用して、不安定なノードによる共有リソースへのアクセスを制限したり、クラスターノードでハードリブートを実行します。

フェンスデバイスが設定されていないと、以前使用していたリソースが解放されていることを切断されているクラスターノードが把握できず、他のクラスターノードでサービスを実行できなくなる可能性があります。また、クラスターノードがそのリソースを解放したとシステムが誤って想定し、データが破損または損失する可能性もあります。フェンスデバイスが設定されていないと、データの整合性は保証できず、クラスター設定はサポートされません。

フェンシングの進行中は、他のクラスター操作を実行できません。クラスターノードの再起動後にフェンシングが完了するか、クラスターノードがクラスターに再度参加するまで、クラスターの通常の動作を再開することができません。

フェンシングの詳細は Fencing in a Red Hat High Availability Cluster を参照してください。

46.2.2. クォーラム

クラスターの整合性と可用性を維持するために、クラスターシステムは、**クォーラム** と呼ばれる概念を使用してデータの破損や損失を防ぎます。クラスターノードの過半数がオンラインになると、クラスターでクォーラムが確立されます。クラスターでクォーラムが確立されない場合は、障害によるデータ破損の可能性を小さくするために、Pacemaker はデフォルトですべてのリソースを停止します。

クォーラムは、投票システムを使用して確立されます。クラスターノードが通常どおり機能しない場合や、クラスターの他の部分との通信が失われた場合に、動作している過半数のノードが、問題のあるノードを分離するように投票し、必要に応じて、接続を切断して別のノードに切り替えてサービスを継続 (フェンス) します。

たとえば、6 ノードクラスターで、4 つ以上のクラスターノードが動作している場合にクォーラムが確立されます。過半数のノードがオフラインまたは利用できない状態になると、クラスターでクォーラムが確立されず、Pacemaker がクラスター化サービスを停止します。

Pacemaker におけるクォーラム機能は、**スプレットブレイン** と呼ばれる状況が発生しないようにします。スプレットブレインは、クラスターが通信から分離されたあとも、各部分が別のクラスターとして機能し続けることで、同じデータの書き込みや、データの破壊または損失が発生する可能性がある現象です。スプリットブレイン状態の詳細と、一般的なクォーラムの概念は Exploring Concepts of RHEL High Availability Clusters - Quorum を参照してください。

Red Hat High Availability Add-On クラスターは、スプリットブレインの状況を回避するために、**votequorum** サービスをフェンシングと併用します。クラスターの各システムには多くの投票数が割り当てられ、過半数の票を取得しているものだけがクラスターの操作を継続できます。

46.2.3. クラスターリソース

クラスターリソース は、クラスターサービスで管理するプログラム、データ、またはアプリケーションのインスタンスです。このようなリソースは、クラスター環境でリソースを管理する標準インターフェイスを提供する **エージェント** により抽象化されます。

リソースを健全な状態に保つために、リソースの定義に監視操作を追加できます。リソースの監視操作を指定しない場合は、デフォルトで監視操作が追加されます。

クラスター内のリソースの動作は、**制約** を指定することで設定できます。以下の制約のカテゴリーを設定できます。

- 場所の制約 - リソースを実行できるノードを設定する
- 順序の制約 - リソースを実行する順序を設定する
- コロケーションの制約 - 他のリソースに対して相対的なリソースの配置先を設定する

クラスターの最も一般的な設定要素の 1 つがリソースセットです。リソースセットはまとめて配置し、順番に起動し、その逆順で停止する必要があります。この設定を簡略化するために、Pacemaker では **グループ** という概念がサポートされます。

46.3. PACEMAKER の概要

Pacemaker は、クラスターリソースマネージャーです。クラスターインフラストラクチャーのメッセージング機能およびメンバーシップ機能を使用して、ノードおよびリソースレベルの障害を防ぎ、障害から復旧することで、クラスターサービスおよびリソースの可用性を最大化します。

46.3.1. Pacemaker アーキテクチャーコンポーネント

Pacemaker で設定されたクラスターは、クラスターメンバーシップを監視する個別のコンポーネントデーモン、サービスを管理するスクリプト、および異なるリソースを監視するリソース管理サブシステムで設定されます。

Pacemaker アーキテクチャーを形成するコンポーネントは、以下のとおりです。

Cluster Information Base (CIB)

XML を内部的に使用して、DC (Designated Coordinator) (CIB を介してクラスターのステータスと動作を格納および分散するために、Pacemaker により割り当てられたノード) から、他のすべてのクラスターノードに対して現在の設定とステータスの情報を分散し、同期する Pacemaker 情報デーモン。

Cluster Resource Management Daemon (CRMd)

Pacemaker クラスターリソースの動作は、このデーモンを介してルーティングされます。CRMd により管理されるリソースは、必要に応じてクライアントシステムが問い合わせることができます。また、リソースを移動したり、インスタンス化したり、変更したりできます。
各クラスターノードには、CRMd とリソースの間のインターフェイスとして動作する LRMd (Local Resource Manager daemon) も含まれます。LRMd は、起動、停止、ステータス情報のリレーなどのコマンドを、CRMd からエージェントに渡します。

Shoot the Other Node in the Head (STONITH)

STONITH は Pacemaker フェンシングの実装です。STONITH は、フェンス要求を処理する Pacemaker のクラスターリソースとして動作し、強制的にノードをシャットダウンし、クラスターからノードを削除してデータの整合性を確保します。STONITH は、CIB で設定し、通常のクラスターリソースとして監視できます。

corosync

corosync は、コアメンバーシップと、高可用性クラスターのメンバー間の通信ニーズに対応するコンポーネントで、デーモンも同じ名前になります。これは、High Availability Add-On が機能するのに必要です。
corosync は、このようなメンバーシップとメッセージング機能のほかに、以下も提供します。

- クォーラムのルールおよび決定を管理します。

- クラスターの複数のメンバーに渡って調整または動作するアプリケーションへのメッセージング機能を提供します。そのため、インスタンス間で、ステートフルな情報またはその他の情報を通信できる必要があります。
- **kronosnet** ライブラリーをネットワークトランスポートとして使用し、複数の冗長なリンクおよび自動フェイルオーバーを提供します。

46.3.2. Pacemaker の設定および管理ツール

High Availability Add-On には、クラスターのデプロイメント、監視、および管理に使用する 2 つの設定ツールが含まれます。

pcs

pcs コマンドラインインターフェイスは、Pacemaker および **corosync** ハートビートデーモンを制御し、設定します。コマンドラインベースのプログラムである **pcs** は、以下のクラスター管理タスクを実行できます。

- Pacemaker/Corosync クラスターの作成および設定
- 実行中のクラスターの設定変更
- Pacemaker と Corosync の両方のリモートでの設定、ならびにクラスターの起動、停止、およびステータス情報の表示

pcsd Web UI

Pacemaker/Corosync クラスターを作成および設定するグラフィカルユーザーインターフェイスです。

46.3.3. クラスターおよび Pacemaker の設定ファイル

Red Hat High Availability Add-On の設定ファイルは、**corosync.conf** および **cib.xml** です。

corosync.conf ファイルは、Pacemaker を構築するクラスターマネージャー (**corosync**) が使用するクラスターパラメーターを提供します。通常は、直接 **corosync.conf** を編集するのではなく、**pcs** インターフェイスまたは **pcsd** インターフェイスを使用します。

cib.xml ファイルは、クラスターの設定、およびクラスターの全リソースにおいて現在の状態を表す XML ファイルです。このファイルは、Pacemaker のクラスター情報ベース (CIB) により使用されます。CIB の内容は、自動的にクラスター全体に同期されます。**cib.xml** ファイルは直接編集せず、代わりに **pcs** インターフェイスまたは **pcsd** インターフェイスを使用してください。

46.4. RED HAT HIGH AVAILABILITY クラスターの LVM 論理ボリューム

Red Hat High Availability Add-On は、2 つの異なるクラスター設定で LVM ボリュームをサポートします。

以下のクラスター設定を選択できます。

- アクティブ/パッシブのフェイルオーバー設定の HA-LVM (High Availability LVM) ボリューム。クラスターで同時にストレージにアクセスするノードは 1 つだけになります。
- アクティブ/アクティブ設定でストレージデバイスを管理する **lvmlockd** を使用する LVM ボリューム。クラスターで、1 つ以上のクラスターが同時にストレージにアクセスする必要があります。**lvmlockd** デーモンは、Resilient Storage Add-On で提供されます。

46.4.1. HA-LVM または共有ボリュームの選択

HA-LVM、または **lvmlockd** デーモンが管理する共有論理ボリュームを使用するタイミングは、デプロイされるアプリケーションまたはサービスのニーズに基づいて決定する必要があります。

- クラスターの複数のノードが、アクティブ/アクティブシステムで LVM ボリュームへの同時読み取りまたは書き込みを必要とする場合に、**lvmlockd** デーモンを使用して、ボリュームを共有ボリュームとして設定します。**lvmlockd** デーモンは、クラスターのノード全体で、LVM ボリュームのアクティベーションおよび変更を同時に調整するシステムを提供します。**lvmlockd** デーモンのロックサービスでは、クラスターのさまざまなノードがボリュームと対話し、レイアウトに変更を加えて、LVM メタデータを保護します。この保護は、複数のクラスターノードで同時にアクティブにされるボリュームグループを共有ボリュームとして設定することにより決まります。

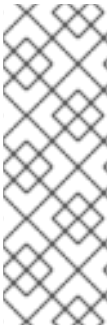
- アクティブ/パッシブで共有リソースを管理するように HA クラスターを設定し、指定した LVM ボリュームに同時にアクセスするメンバーを 1 つのみにした場合は、HA-LVM で **lvmlockd** ロックサービスを使用する必要はありません。

ほとんどのアプリケーションは、その他のインスタンスと同時に実行するように設計または最適化されていないため、アクティブ/パッシブ設定での実行により適しています。共有論理ボリュームで、クラスターに対応していないアプリケーションを実行すると、パフォーマンスが低下することがあります。これは、論理ボリューム自体にクラスター通信のオーバーヘッドが発生するためです。クラスター対応のアプリケーションは、クラスターファイルシステムとクラスター対応の論理ボリュームにより発生するパフォーマンスの低下を上回るパフォーマンスの向上を実現できるようにする必要があります。実現が容易かどうかは、アプリケーションやワークロードによって異なります。クラスターの要件を判断し、アクティブ/アクティブのクラスターを最適化する努力に価値があるかどうかを判断して、どちらの LVM を使用するかを選択します。ほとんどの場合は、HA-LVM を使用すると HA を最適化できます。

HA-LVM および **lvmlockd** を使用する共有論理ボリュームは、複数のマシンが変更を重複して行うと発生する、LVM メタデータとその論理ボリュームの破損を防ぐという点で似ています。HA-LVM では、論理ボリュームは、アクティベートする場合は排他的に行うように制限されているため、一度に 1 つのマシンでしかアクティブになりません。そのため、ストレージドライバーのローカル (非クラスター) 実装のみが使用されます。このようにクラスターの調整オーバーヘッドが発生しないようにすると、パフォーマンスが向上します。**lvmlockd** を使用する共有ボリュームにはこのような制限はなく、ユーザーは、クラスターのすべてのマシンで論理ボリュームをアクティベートできます。これにより、クラスター対応のストレージドライバーの使用が強制され、クラスター対応のファイルシステムとアプリケーションが優先されます。

46.4.2. クラスター内での LVM ボリュームの設定

クラスターは Pacemaker で管理されます。HA-LVM および共有論理ボリュームは、Pacemaker クラスターと併用される場合のみサポートされ、クラスターリソースとして設定する必要があります。



注記

Pacemaker クラスターが使用する LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように、ターゲット用に **systemd resource-agents-deps** ターゲットと **systemd** ドロップインユニットを設定することを推奨します。**systemd resource-agents-deps** ターゲットを設定する方法は、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

- HA-LVM ボリュームを Pacemaker クラスターの一部として設定する手順は、Red Hat High

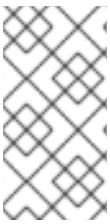
Availability クラスターでのアクティブ/パッシブ Apache HTTP サーバーの設定 および Red Hat High Availability クラスターのアクティブ/パッシブな NFS サーバーの設定 を参照してください。
この手順には、以下の手順が含まれます。

- クラスターのみがボリュームグループをアクティベートできるようにする
- LVM 論理ボリュームを設定する
- LVM ボリュームをクラスターリソースとして設定する

- **lvmlockd** デーモンを使用してストレージデバイスをアクティブ/アクティブ設定で管理する共有 LVM ボリュームを設定する手順は、クラスター内の GFS2 ファイルシステム および Red Hat High Availability クラスターでのアクティブ/アクティブ Samba サーバーの設定 を参照してください。

第47章 PACEMAKER の使用の開始

Pacemaker クラスターの作成に使用するツールとプロセスに慣れるために、次の手順を実行できます。ここで説明する内容は、クラスターソフトウェアの概要と、作業用のクラスターを設定せずに管理する方法に関心のあるユーザーを対象としています。



注記

ここで説明する手順では、2 つ以上のノードとフェンシングデバイスの設定が必要となるサポート対象の Red Hat クラスターは作成されません。Red Hat のサポートポリシー、要件、および制限の詳細は、RHEL 高可用性クラスターのサポートポリシー を参照してください。

47.1. PACEMAKER の使用方法

ここでは、Pacemaker を使用してクラスターを設定する方法、クラスターのステータスを表示する方法、およびクラスターサービスを設定する方法を学習します。この例では、Apache HTTP サーバーをクラスターリソースとして作成し、リソースに障害が発生した場合のクラスターの応答方法を表示します。

この例では、以下のように設定されています。

- ノード: **z1.example.com**
- Floating IP アドレス: 192.168.122.120

前提条件

- RHEL 8 を実行しているノード 1 つ
- このノードで静的に割り当てられている IP アドレスの 1 つと同じネットワーク上にあるフローティング IP アドレス
- **/etc/hosts** ファイルに、実行中のノード名が含まれている

手順

1. High Availability チャンネルから Red Hat High Availability Add-On ソフトウェアパッケージをインストールし、**pcsd** サービスを起動して有効にします。

```
# yum install pcs pacemaker fence-agents-all
...
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

firewalld デーモンを実行している場合は、Red Hat High Availability Add-On で必要なポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

2. クラスターの各ノードにユーザー **hacluster** のパスワードを設定し、**pcs** コマンドを実行するノードにあるクラスターの各ノードに対して、**hacluster** ユーザーの認証を行います。この例では、ノードを 1 つだけ使用し、そのノードからコマンドを実行していますが、このステップ

はサポート対象の Red Hat High Availability マルチノードクラスターを設定する際に必要となるため、この手順に含まれています。

```
# passwd hacluster
...
# pcs host auth z1.example.com
```

3. メンバーを 1 つ含む **my_cluster** という名前のクラスターを作成し、クラスターのステータスを確認します。この 1 つのコマンドで、クラスターが作成され、起動します。

```
# pcs cluster setup my_cluster --start z1.example.com
...
# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z1.example.com
 1 node configured
 0 resources configured

PCSD Status:
  z1.example.com: Online
```

4. Red Hat High Availability クラスターでは、クラスターのフェンシングを設定することが必要になります。この要件が必要になる理由は Fencing in a Red Hat High Availability Cluster を参照してください。ただし、ここでは基本的な Pacemaker コマンドの使用方法を説明することを目的としているため、**stonith-enabled** クラスターのオプションを **false** に設定し、フェンシングを無効にします。

警告

stonith-enabled=false の使用は、実稼働クラスターには完全に適していません。これにより、障害が発生したノードが適切にフェンスされていることを装うようにクラスターに指示されます。

```
# pcs property set stonith-enabled=false
```

5. システムに Web ブラウザーを設定し、Web ページを作成して簡単なテキストメッセージを表示します。**firewalld** デーモンを実行している場合は、**httpd** で必要なポートを有効にします。

注記

システムの起動時に使用する場合は、**systemctl enable** で、クラスターが管理するサービスを有効にしないでください。

```
# yum install -y httpd wget
...
```

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

# cat <<-END >/var/www/html/index.html
<html>
<body>My Test Site - $(hostname)</body>
</html>
END
```

Apache リソースエージェントが Apache のステータスを取得できるようにするため、既存の設定に以下の内容を追加して、ステータスサーバーの URL を有効にします。

```
# cat <<-END > /etc/httpd/conf.d/status.conf
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from 127.0.0.1
Allow from ::1
</Location>
END
```

6. クラスターが管理するリソース **IPaddr2** および **apache** を作成します。IPaddr2 はフローティング IP であるため、物理ノードに関連付けられている IP アドレスは使用できません。IPaddr2 の NIC デバイスを指定しない場合は、そのノードで使用される、静的に割り当てられた IP アドレスと同じネットワークにフローティング IP が存在する必要があります。
利用可能なリソースタイプのリストを表示する場合は、**pcs resource list** コマンドを使用します。指定したリソースタイプに設定できるパラメーターを表示する場合は、**pcs resource describe resourcetype** コマンドを使用します。たとえば、以下のコマンドは、**apache** タイプのリソースに設定できるパラメーターを表示します。

```
# pcs resource describe apache
...
```

この例では、IP アドレスリソースと apache リソースの両方が **apachegroup** グループに含まれるように設定します。これにより、両リソースが一緒に保存され、作業用のマルチノードクラスターを設定する際に、同じノードで実行できます。

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.122.120 --group
apachegroup

# pcs resource create WebSite ocf:heartbeat:apache
configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" --group
apachegroup

# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
Last updated: Fri Oct 12 09:54:33 2018
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

1 node configured
2 resources configured
```

```
Online: [ z1.example.com ]

Full list of resources:

Resource Group: apachegroup
    ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
    WebSite    (ocf::heartbeat:apache):        Started z1.example.com

PCSD Status:
  z1.example.com: Online
...
```

クラスターリソースを設定したら、**pcs resource config** コマンドを使用して、そのリソースに設定したオプションを表示します。

```
# pcs resource config WebSite
Resource: WebSite (class=ocf provider=heartbeat type=apache)
 Attributes: configfile=/etc/httpd/conf/httpd.conf statusurl=http://localhost/server-status
 Operations: start interval=0s timeout=40s (WebSite-start-interval-0s)
             stop interval=0s timeout=60s (WebSite-stop-interval-0s)
             monitor interval=1min (WebSite-monitor-interval-1min)
```

7. ブラウザーで、設定済みのフローティング IP アドレスを使用して作成した Web サイトを開くように指定します。定義したテキストメッセージが表示されるはずです。

8. Apache Web サービスを停止し、クラスターのステータスを確認します。**killall -9** を使用して、アプリケーションレベルのクラッシュをシミュレートします。

```
# killall -9 httpd
```

クラスターのステータスを確認します。Web サービスは停止したためアクションは失敗しますが、クラスターソフトウェアがサービスを再起動したため、Web サイトに引き続きアクセスできることが確認できるはずです。

```
# pcs status
Cluster name: my_cluster
...
Current DC: z1.example.com (version 1.1.13-10.el7-44eb2dd) - partition with quorum
1 node and 2 resources configured

Online: [ z1.example.com ]

Full list of resources:

Resource Group: apachegroup
    ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
    WebSite    (ocf::heartbeat:apache):        Started z1.example.com

Failed Resource Actions:
* WebSite_monitor_60000 on z1.example.com 'not running' (7): call=13, status=complete,
exitreason='none',
    last-rc-change='Thu Oct 11 23:45:50 2016', queued=0ms, exec=0ms

PCSD Status:
    z1.example.com: Online
```

サービスが再開すると、障害が発生したリソースの障害 (failure) ステータスが削除されるため、クラスターステータスを確認する際に、障害が発生したアクションの通知が表示されなくなります。

```
# pcs resource cleanup WebSite
```

9. クラスターと、クラスターのステータスを確認したら、ノードでクラスターサービスを停止します。(この手順を試すために、実際にサービスを起動したノードが 1 つだけであっても) **--all** パラメーターを追加してください。これにより実際のマルチノードクラスターの全ノードでクラスターサービスが停止します。

```
# pcs cluster stop --all
```

47.2. フェイルオーバーの設定方法

以下の手順では、サービスを実行する Pacemaker クラスターの作成方法を紹介します。このサービスは、サービスを実行しているノードが利用できなくなると、現在のノードから別のノードにフェイルオーバーします。この手順を行って、2 ノードクラスターでサービスを作成する方法と、サービスを実行しているノードでサービスが失敗するとどうなるかを確認します。

この手順では、Apache HTTP サーバーを実行する 2 ノード Pacemaker クラスターを設定します。その後、1 つのノードで Apache サービスを停止し、どのようにしてサービスを利用可能のままにしているかを確認できます。

この例では、以下のように設定されています。

- ノード: **z1.example.com** および **z2.example.com**
- Floating IP アドレス: 192.168.122.120

前提条件

- 相互に通信が可能な RHEL 8 を実行するノード 2 つ
- このノードで静的に割り当てられている IP アドレスの 1 つと同じネットワーク上にあるフローティング IP アドレス
- **/etc/hosts** ファイルに、実行中のノード名が含まれている

手順

1. 両方のノードで、High Availability チャンネルから Red Hat High Availability Add-On ソフトウェアパッケージをインストールし、**pcsd** サービスを起動して有効にします。

```
# yum install pcs pacemaker fence-agents-all
...
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

firewalld デーモンを実行している場合は、両方のノードで、Red Hat High Availability Add-On で必要なポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

2. クラスター内の両方のノードに、**hacluster** ユーザーのパスワードを設定します。

```
# passwd hacluster
```

3. **pcs** コマンドを実行するノードで、クラスター内の各ノードに対して **hacluster** ユーザーの認証を行います。

```
# pcs host auth z1.example.com z2.example.com
```

4. 両方のノードで、クラスターメンバーとなるクラスター **my_cluster** を作成します。この1つのコマンドで、クラスターが作成され、起動します。**pcs** 設定コマンドはクラスター全体に適用されるため、このコマンドは、クラスター内のいずれかのノードで実行してください。
クラスター内のいずれかのノードで、以下のコマンドを実行します。

```
# pcs cluster setup my_cluster --start z1.example.com z2.example.com
```

5. Red Hat High Availability クラスターでは、クラスターのフェンシングを設定することが必要になります。この要件が必要になる理由は Fencing in a Red Hat High Availability Cluster を参照してください。ただし、この設定でフェイルオーバーがどのように機能するかを示す場合は、**stonith-enabled** クラスターのオプションを **false** に設定し、フェンシングを無効にします。

警告

stonith-enabled=false の使用は、実稼働クラスターには完全に適していません。これにより、障害が発生したノードが適切にフェンスされていることを装うようにクラスターに指示されます。

```
# pcs property set stonith-enabled=false
```

6. クラスターを作成し、フェンシングを無効にしたら、クラスターのステータスを確認します。

注記

pcs cluster status コマンドを実行したときの出力は、一時的に、システムコンポーネントの起動時の例とは若干異なる場合があります。

```
# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z1.example.com
 2 nodes configured
 0 resources configured
```

```
PCSD Status:
  z1.example.com: Online
  z2.example.com: Online
```

7. 両方のノードに Web ブラウザーを設定し、Web ページを作成して簡単なテキストメッセージを表示します。**firewalld** デーモンを実行している場合は、**httpd** で必要なポートを有効にします。

注記

システムの起動時に使用する場合は、**systemctl enable** で、クラスターが管理するサービスを有効にしないでください。

```
# yum install -y httpd wget
...
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload

# cat <<-END >/var/www/html/index.html
<html>
<body>My Test Site - $(hostname)</body>
</html>
END
```

Apache リソースエージェントが、クラスターの各ノードで Apache のステータスを取得できるようにするため、既存の設定に以下の内容を追加して、ステータスサーバーの URL を有効にします。

```
# cat <<-END > /etc/httpd/conf.d/status.conf
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from 127.0.0.1
Allow from ::1
</Location>
END
```

8. クラスターが管理するリソース **IPaddr2** および **apache** を作成します。IPaddr2 はフローティング IP であるため、物理ノードに関連付けられている IP アドレスは使用できません。IPaddr2 の NIC デバイスを指定しない場合は、そのノードで使用される、静的に割り当てられた IP アドレスと同じネットワークにフローティング IP が存在する必要があります。
利用可能なリソースタイプのリストを表示する場合は、**pcs resource list** コマンドを使用します。指定したリソースタイプに設定できるパラメーターを表示する場合は、**pcs resource describe resourcetype** コマンドを使用します。たとえば、以下のコマンドは、**apache** タイプのリソースに設定できるパラメーターを表示します。

```
# pcs resource describe apache
...
```

この例では、IP アドレスリソースおよび apache リソースの両方が、**apachegroup** という名前のグループに含まれるように設定します。これにより、両リソースが一緒に保存され、同じノードで実行できます。

クラスター内のいずれかのノードで、次のコマンドを実行します。

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.122.120 --group
apachegroup

# pcs resource create WebSite ocf:heartbeat:apache
configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" --group
apachegroup

# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
Last updated: Fri Oct 12 09:54:33 2018
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

2 nodes configured
2 resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:

Resource Group: apachegroup
    ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
    WebSite    (ocf::heartbeat:apache):        Started z1.example.com

PCSD Status:
  z1.example.com: Online
  z2.example.com: Online
...
```

このインスタンスでは、**apachegroup** サービスが z1.example.com ノードで実行していることに注意してください。

9. 作成した Web サイトにアクセスし、サービスを実行しているノードでそのサービスを停止し、2 番目のノードにサービスがフェイルオーバーする方法を確認してください。

   a. ブラウザーで、設定済みのフローティング IP アドレスを使用して作成した Web サイトを開くように指定します。定義したテキストメッセージが表示され、Web サイトを実行しているノードの名前が表示されるはずです。

   b. Apache Web サービスを停止します。**killall -9** を使用して、アプリケーションレベルのクラッシュをシミュレートします。

      ```
      # killall -9 httpd
      ```

      クラスターのステータスを確認します。Web サービスを停止したためにアクションが失敗したものの、サービスが実行していたノードでクラスターソフトウェアがサービスを再起動するため、Web サイトに引き続きアクセスできるはずです。

      ```
      # pcs status
      Cluster name: my_cluster
      Stack: corosync
      Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
      ```

```
Last updated: Fri Oct 12 09:54:33 2018
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

2 nodes configured
2 resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:

Resource Group: apachegroup
   ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
   WebSite    (ocf::heartbeat:apache):        Started z1.example.com

Failed Resource Actions:
* WebSite_monitor_60000 on z1.example.com 'not running' (7): call=31,
status=complete, exitreason='none',
   last-rc-change='Fri Feb  5 21:01:41 2016', queued=0ms, exec=0ms
```

サービスが再開したら、障害 (failure) ステータスを削除します。

```
# pcs resource cleanup WebSite
```

c. サービスを実行しているノードをスタンバイモードにします。フェンシングを無効にしているため、ノードレベルの障害 (電源ケーブルを引き抜くなど) を効果的にシミュレートできません。クラスターがこのような状態から復旧するにはフェンシングが必要になるためです。

```
# pcs node standby z1.example.com
```

d. クラスターのステータスを確認し、サービスを実行している場所をメモします。

```
# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
Last updated: Fri Oct 12 09:54:33 2018
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

2 nodes configured
2 resources configured

Node z1.example.com: standby
Online: [ z2.example.com ]

Full list of resources:

Resource Group: apachegroup
   ClusterIP  (ocf::heartbeat:IPaddr2):       Started z2.example.com
   WebSite    (ocf::heartbeat:apache):        Started z2.example.com
```

e. Web サイトにアクセスします。サービスの切断はありません。表示メッセージには、サービスを実行しているノードが含まれるはずです。

10. クラスターサービスを最初のノードに復元するには、そのノードをスタンドバイモードから回復します。ただし、必ずしもそのサービスが最初のノードに戻るわけではありません。

```
# pcs node unstandby z1.example.com
```

11. 最終的なクリーンナップを行うために、両方のノードでクラスターサービスを停止します。

```
# pcs cluster stop --all
```

第48章 PCS コマンドラインインターフェイス

pcs コマンドラインインターフェイスを使用すると、**corosync**、**pacemaker**、**booth**、**sbd** などのクラスターサービスを制御し、設定を簡単に行うことができます。

cib.xml 設定ファイルは直接編集しないでください。ほとんどの場合、Pacemaker は、直接編集した **cib.xml** ファイルを受け付けません。

48.1. PCS HELP DISPLAY

pcs コマンドで **-h** オプションを使用すると、**pcs** コマンドのパラメーターと、その説明が表示されます。

次のコマンドは、**pcs resource** コマンドのパラメーターを表示します。

```
# pcs resource -h
```

48.2. 未編集のクラスター設定の表示

クラスター設定ファイルは直接編集しないようにしてください。未編集のクラスター設定は、**pcs cluster cib** コマンドで表示できます。

pcs cluster cib filename コマンドを使用すると、未編集のクラスター設定を、指定したファイルに保存できます。クラスターを事前に設定していて、アクティブな CIB が存在する場合は、以下のコマンドを実行して、未編集の xml ファイルを保存します。

```
pcs cluster cib filename
```

たとえば、次のコマンドを使用すると、**testfile** という名前のファイルに、未編集の CIB の xml ファイルが保存されます。

```
# pcs cluster cib testfile
```

48.3. 作業ファイルへの設定変更の保存

クラスターを設定する際に、アクティブな CIB に影響を及ぼさずに、指定したファイルに設定変更を保存できます。これにより、個々の更新を使用して実行中のクラスター設定を直ちに更新することなく、設定の更新を指定できます。

CIB をファイルに保存する方法は、未編集のクラスター設定の表示 を参照してください。そのファイルを作成したら、**pcs** コマンドの **-f** オプションを使用したアクティブな CIB ではなく、ファイルに設定変更を保存できます。変更を完了し、アクティブな CIB ファイルへの更新が用意できたら、**pcs cluster cib-push** コマンドでファイルの更新をプッシュできます。

手順

以下は、CIB のファイルに変更をプッシュする際に推奨される手順です。この手順は、保存した CIB ファイルのコピーを作成し、そのコピーを変更します。アクティブな CIB にその変更をプッシュする場合は、ここで、**pcs cluster cib-push** コマンドの **diff-against** オプションを指定して、元のファイルと、変更したファイルの差異だけが CIB にプッシュされるようにします。これにより、ユーザーが互いを上書きしないように、並列に変更を加えることができます。ここでは、設定ファイル全体を解析する必要はないため、Pacemaker への負荷が減ります。

1. ファイルへのアクティブな CIB を保存します。この例では、**original.xml** という名前のファイルに CIB が保存されます。

```
# pcs cluster cib original.xml
```

2. 設定の更新に使用する作業ファイルに、保存したファイルをコピーします。

```
# cp original.xml updated.xml
```

3. 必要に応じて設定を更新します。以下のコマンドは、**updated.xml** ファイルにリソースを作成しますが、現在実行しているクラスター設定にはそのリソースを追加しません。

```
# pcs -f updated.xml resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120
op monitor interval=30s
```

4. 更新したファイルを、アクティブな CIB にプッシュします。元のファイルに加えた変更のみをプッシュするように指定します。

```
# pcs cluster cib-push updated.xml diff-against=original.xml
```

もしくは、次のコマンドを使用して、CIB ファイルの現在のコンテンツ全体をプッシュできます。

```
pcs cluster cib-push filename
```

CIB ファイル全体をプッシュすると、Pacemaker はバージョンを確認して、クラスターにあるものよりも古い場合は CIB ファイルをプッシュしません。クラスターにあるものよりも古いバージョンで CIB ファイル全体を更新する必要がある場合は、**pcs cluster cib-push** コマンドの **--config** オプションを使用します。

```
pcs cluster cib-push --config filename
```

48.4. クラスターのステータス表示

さまざまなコマンドを使用して、クラスターおよびそのコンポーネントのステータスを表示できます。

次のコマンドで、クラスターおよびクラスターリソースのステータスを表示します。

```
# pcs status
```

pcs status コマンドの commands パラメーター (**resources**、**cluster**、**nodes**、または **pcsd**) を指定すると、特定のクラスターコンポーネントのステータスを表示できます。

```
pcs status commands
```

たとえば、次のコマンドは、クラスターリソースのステータスを表示します。

```
# pcs status resources
```

このコマンドはクラスターの状態を表示しますが、クラスターリソースの状態は表示しません。

```
# pcs cluster status
```

48.5. クラスターの全設定の表示

現在のクラスター設定をすべて表示する場合は、次のコマンドを実行します。

```
# pcs config
```

48.6. PCS コマンドによる COROSYNC.CONF ファイルの変更

Red Hat Enterprise Linux 8.4 では、**pcs** コマンドを使用して **corosync.conf** ファイルのパラメーターを変更できます。

次のコマンドは、**corosync.conf** ファイルのパラメーターを変更します。

```
pcs cluster config update [transport pass:quotes[transport options]] [compression pass:quotes[compression options]] [crypto pass:quotes[crypto options]] [totem pass:quotes[totem options]] [--corosync_conf pass:quotes[path]]
```

以下のコマンド例は、**knet_pmtud_interval** トランスポート値と、**token** と **join** の totem の値を更新します。

```
# pcs cluster config update transport knet_pmtud_interval=35 totem token=10000 join=100
```

関連情報

- 既存クラスターにノードを追加および削除する方法は、クラスターノードの管理 を参照してください。
- 既存クラスターのリンクの追加および変更の詳細は、既存クラスターへのリンクの追加および変更 を参照してください。
- クラスターでクォーラムオプションを変更する方法およびクォーラムデバイスの設定を管理する方法は、クラスタークォーラムの設定 および クォーラムデバイスの設定 を参照してください。

48.7. PCS コマンドでの COROSYNC.CONF ファイルの表示

次のコマンドは、**corosync.conf** クラスター設定ファイルの内容を表示します。

```
# pcs cluster corosync
```

Red Hat Enterprise Linux 8.4 では、以下のように **pcs cluster config** コマンドで、人間が判読できる形式で **corosync.conf** ファイルの内容を出力できます。

クラスターが RHEL 8.7 以降で作成された場合、または UUID によるクラスターの識別 で説明されているように UUID が手動で追加された場合、このコマンドの出力にはクラスターの UUID が含まれます。

```
[root@r8-node-01 ~]# pcs cluster config
Cluster Name: HACluster
Cluster UUID: ad4ae07dcafe4066b01f1cc9391f54f5
Transport: knet
Nodes:
  r8-node-01:
```

```
    Link 0 address: r8-node-01
    Link 1 address: 192.168.122.121
    nodeid: 1
  r8-node-02:
    Link 0 address: r8-node-02
    Link 1 address: 192.168.122.122
    nodeid: 2
Links:
  Link 1:
    linknumber: 1
    ping_interval: 1000
    ping_timeout: 2000
    pong_count: 5
Compression Options:
  level: 9
  model: zlib
  threshold: 150
Crypto Options:
  cipher: aes256
  hash: sha256
Totem Options:
  downcheck: 2000
  join: 50
  token: 10000
Quorum Device: net
  Options:
    sync_timeout: 2000
    timeout: 3000
  Model Options:
    algorithm: lms
    host: r8-node-03
  Heuristics:
    exec_ping: ping -c 1 127.0.0.1
```

RHEL 8.4 では、**--output-format=cmd** オプションを指定して **pcs cluster config show** コマンドを実行し、以下の例のように、既存の **corosync.conf** ファイルの再作成に使用できる **pcs** 設定コマンドを表示できます。

```
[root@r8-node-01 ~]# pcs cluster config show --output-format=cmd
pcs cluster setup HACluster \
  r8-node-01 addr=r8-node-01 addr=192.168.122.121 \
  r8-node-02 addr=r8-node-02 addr=192.168.122.122 \
  transport \
  knet \
    link \
      linknumber=1 \
      ping_interval=1000 \
      ping_timeout=2000 \
      pong_count=5 \
    compression \
      level=9 \
      model=zlib \
      threshold=150 \
    crypto \
      cipher=aes256 \
      hash=sha256 \
```

```
totem \
  downcheck=2000 \
  join=50 \
  token=10000
```

第49章 PACEMAKER を使用した RED HAT HIGH AVAILABILITY クラスターの作成

以下の手順では、**pcs** コマンドラインインターフェイスを使用して、2 ノードの Red Hat High Availability クラスターを作成します。

この例では、クラスターを設定するために、システムに以下のコンポーネントを追加する必要があります。

- クラスターを作成するのに使用する 2 つのノード。この例では、使用されるノードは **z1.example.com** および **z2.example.com** です。
- プライベートネットワーク用のネットワークスイッチ。クラスターノード同士の通信、およびその他のクラスターハードウェア (ネットワーク電源スイッチやファイバーチャネルスイッチなど) との通信にプライベートネットワークを使用することが推奨されますが、必須ではありません。
- クラスターの各ノード用のフェンスデバイス。この例では、APC 電源スイッチの 2 ポートを使用します。ホスト名は **zapc.example.com** です。

49.1. クラスターソフトウェアのインストール

以下の手順では、クラスターソフトウェアをインストールし、システムでクラスターの作成を設定するように設定します。

手順

1. クラスター内の各ノードで、システムアーキテクチャーに対応する高可用性のリポジトリーを有効にします。たとえば、x86_64 システムの高可用性リポジトリーを有効にするには、以下の **subscription-manager** コマンドを入力します。

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
```

2. クラスターの各ノードに、Red Hat High Availability Add-On ソフトウェアパッケージと、使用可能なすべてのフェンスエージェントを、High Availability チャンネルからインストールします。

```
# yum install pcs pacemaker fence-agents-all
```

または、次のコマンドを実行して、Red Hat High Availability Add-On ソフトウェアパッケージと、必要なフェンスエージェントのみをインストールすることもできます。

```
# yum install pcs pacemaker fence-agents-model
```

次のコマンドは、利用できるフェンスエージェントのリストを表示します。

```
# rpm -q -a | grep fence
fence-agents-rhevm-4.0.2-3.el7.x86_64
fence-agents-ilo-mp-4.0.2-3.el7.x86_64
fence-agents-ipmilan-4.0.2-3.el7.x86_64
...
```

警告

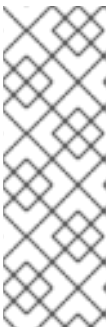
Red Hat High Availability Add-On パッケージをインストールしたら、自動的に何もインストールされないように、ソフトウェア更新設定を行う必要があります。実行中のクラスターにインストールすると、予期しない動作が発生する可能性があります。詳細は RHEL 高可用性またはレジリエントストレージクラスターにソフトウェア更新を適用するのに推奨されるプラクティス を参照してください。

3. **firewalld** デーモンを実行している場合は、次のコマンドを実行して、Red Hat High Availability Add-On で必要なポートを有効にします。

注記

firewalld デーモンがシステムにインストールされているかどうかを確認する場合は、**rpm -q firewalld** コマンドを実行します。firewalld デーモンがインストールされている場合は、**firewall-cmd --state** コマンドで、実行しているかどうかを確認できます。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```



注記

クラスターコンポーネントの理想的なファイアウォール設定は、ローカル環境によって異なります。ここでは、ノードに複数のネットワークインターフェイスがあるかどうか、オフホストのファイアウォールがあるかどうかを検討しないといけない場合があります。この例では、Pacemaker クラスターで通常必要となるポートを開きますが、ローカル条件に合わせて変更する必要があります。High Availability Add-On のポートの有効化 では、Red Hat High Availability Add-On で有効にするポートと、各ポートの使用目的が説明されています。

4. **pcs** を使用してクラスターの設定やノード間の通信を行うため、**pcs** の管理アカウントとなるユーザー ID **hacluster** のパスワードを各ノードに設定する必要があります。**hacluster** ユーザーのパスワードは、各ノードで同じにすることが推奨されます。

```
# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

5. クラスターを設定する前に、各ノードで、システムの起動時に **pcsd** デーモンを起動できるように、デーモンを起動して有効にしておく必要があります。このデーモンは、**pcs** コマンドで動作し、クラスターのノード全体で設定を管理します。
クラスターの各ノードで次のコマンドを実行して、システムの起動時に **pcsd** サービスが起動し、**pcsd** が有効になるように設定します。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

49.2. PCP-ZEROCONF パッケージのインストール (推奨)

クラスターを設定する際に、PCP (Performance Co-Pilot) ツールの **pcp-zeroconf** パッケージをインストールすることが推奨されます。PCP は、RHEL システムに推奨される Red Hat の resource-monitoring ツールです。**pcp-zeroconf** パッケージをインストールすると、PCP を実行してパフォーマンス監視データを収集して、フェンシング、リソース障害、およびクラスターを中断するその他のイベントの調査に役立てることができます。

注記

PCP が有効になっているクラスターデプロイメントには、**/var/log/pcp/** を含むファイルシステムで PCP が取得したデータ用に十分な領域が必要です。PCP による一般的な領域使用はデプロイメントごとに異なりますが、通常 **pcp-zeroconf** のデフォルト設定を使用する場合は 10Gb で十分であり、環境によっては必要な量が少なくなることがあります。一般的なアクティビティーの 14 日間におけるこのディレクトリーの使用状況を監視すると、より正確な使用状況の概算値が得られます。

手順

pcp-zeroconf 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 pcp-zeroconf
```

このパッケージは **pmcd** を有効にし、10 秒間隔でデータキャプチャーを設定します。

PCP データを確認する方法は、Red Hat カスタマーポータルの Why did a RHEL High Availability cluster node reboot - how can I prevent it again を参照してください。

49.3. 高可用性クラスターの作成

以下の手順では、Red Hat High Availability Add-On クラスターを作成します。この手順の例では、ノード **z1.example.com** および **z2.example.com** で構成されるクラスターを作成します。

手順

1. **pcs** を実行するノードで、クラスター内の各ノードに対して、**pcs** ユーザー **hacluster** を認証します。
   次のコマンドは、**z1.example.com** と **z2.example.com** で構成される 2 ノードクラスターの両ノードに対して、**z1.example.com** の **hacluster** ユーザーを認証します。

   ```
   [root@z1 ~]# pcs host auth z1.example.com z2.example.com
   Username: hacluster
   Password:
   z1.example.com: Authorized
   z2.example.com: Authorized
   ```

2. **z1.example.com** で以下のコマンドを実行し、2 つのノード **z1.example.com** と **z2.example.com** で構成される 2 ノードクラスター **my_cluster** を作成します。これにより、クラスター設定ファイルが、クラスターの両ノードに伝搬されます。このコマンドには **--start**

オプションが含まれます。このオプションを使用すると、クラスターの両ノードでクラスターサービスが起動します。

```
[root@z1 ~]# pcs cluster setup my_cluster --start z1.example.com z2.example.com
```

3. クラスターサービスを有効にし、ノードの起動時にクラスターの各ノードでクラスターサービスが実行するようにします。

注記

使用している環境でクラスターサービスを無効のままにしておきたい場合などは、この手順を省略できます。この手順を行うことで、ノードがダウンした場合にクラスターやリソース関連の問題をすべて解決してから、そのノードをクラスターに戻すことができます。クラスターサービスを無効にしている場合には、ノードを再起動する時に **pcs cluster start** コマンドを使用して手作業でサービスを起動しなければならないので注意してください。

```
[root@z1 ~]# pcs cluster enable --all
```

pcs cluster status コマンドを使用するとクラスターの現在の状態を表示できます。**pcs cluster setup** コマンドで **--start** オプションを使用してクラスターサービスを起動した場合は、クラスターが稼働するのに時間が少しかかる可能性があるため、クラスターとその設定で後続の動作を実行する前に、クラスターが稼働していることを確認する必要があります。

```
[root@z1 ~]# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z2.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z2.example.com
 2 Nodes configured
 0 Resources configured

...
```

49.4. 複数のリンクを使用した高可用性クラスターの作成

pcs cluster setup コマンドを使用して、各ノードにリンクをすべて指定することで、複数のリンクを持つ Red Hat High Availability クラスターを作成できます。

2 つのリンクを持つ 2 ノードクラスターを作成する基本的なコマンドの形式は、以下のとおりです。

```
pcs cluster setup pass:quotes[cluster_name] pass:quotes[node1_name]
addr=pass:quotes[node1_link0_address] addr=pass:quotes[node1_link1_address]
pass:quotes[node2_name] addr=pass:quotes[node2_link0_address]
addr=pass:quotes[node2_link1_address]
```

このコマンドの完全な構文は、**pcs**(8) の man ページを参照してください。

複数のリンクを持つクラスターを作成する場合は、次に示す内容を検討してください。

- **addr=address** パラメーターの順番は重要です。ノード名の後に指定する最初のアドレスは **link0** に使用され、2 番目以降のアドレスは **link1** 以降に順に使用されます。

- デフォルトでは、リンクに **link_priority** が指定されていない場合、リンクの優先度はリンク番号と同じになります。リンクの優先度は、指定された順序に従って 0、1、2、3 などになり、0 はリンクの優先順位が最も高くなります。
- デフォルトのリンクモードは **passive** で、リンク優先度の数字が最も小さいアクティブなリンクが使用されます。
- **link_mode** および **link_priority** のデフォルト値では、指定された最初のリンクが最も優先順位の高いリンクとして使用され、そのリンクが失敗した場合は、指定された次のリンクが使用されます。
- デフォルトのトランスポートプロトコルである **knet** トランスポートプロトコルを使用して、リンクを 8 つまで指定できます。
- **addr=** パラメーターの数は、すべてのノードで同じでなければなりません。
- RHEL 8.1 の時点では、**pcs cluster link add**、**pcs cluster link remove**、**pcs cluster link delete** および **pcs cluster link update** コマンドを使用して既存のクラスターのリンクを追加、削除、変更できます。
- シングルリンククラスターと同様、1 つのリンクで IPv4 アドレスと IPv6 アドレスを混在させないでください。ただし、1 つのリンクで IPv4 を実行し、別のリンクで IPv6 を実行することはできます。
- シングルリンククラスターと同様、1 つのリンクで IPv4 アドレスと IPv6 アドレスが混在しない IPv4 アドレスまたは IPv6 アドレスで名前が解決される限り、アドレスを IP アドレスまたは名前として指定できます。

この例では、2 つのノード **rh80-node1** と **rh80-node2** で設定される 2 ノードクラスター **my_twolink_cluster** を作成します。**rh80-node1** のインターフェイスとして、IP アドレス 192.168.122.201 を **link0**、192.168.123.201 を **link1** に設定します。**rh80-node2** のインターフェイスとして、IP アドレス 192.168.122.202 を **link0**、192.168.123.202 を **link1** に設定します。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202
```

リンク優先度を、リンク番号であるデフォルト値とは異なる値に設定するには、**pcs cluster setup** コマンドの **link_priority** オプションを使用してリンクの優先度を設定します。以下の 2 つのコマンド例のそれぞれは、2 つのインターフェイスを持つ 2 ノードクラスターを作成し、最初のリンクであるリンク 0 のリンク優先度は 1 で、2 番目のリンクであるリンク 1 のリンク優先度は 0 です。リンク 1 が最初に使用され、リンク 0 はフェイルオーバーリンクとして機能します。リンクモードが指定されていないため、デフォルトの passive に設定されます。

これら 2 つのコマンドは同等です。**link** キーワードの後にリンク番号を指定しないと、**pcs** インターフェイスは使用されていない最も小さいリンク番号からリンク番号を自動的に追加します。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link link_priority=1 link link_priority=0

# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link linknumber=1 link_priority=0 link link_priority=1
```

以下の例のように、**pcs cluster setup** コマンドの **link_mode** オプションを使用して、リンクモードをデフォルト値の **passive** とは異なる値に設定できます。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link_mode=active
```

以下の例では、リンクモードとリンク優先度の両方を設定します。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link_mode=active link link_priority=1 link link_priority=0
```

リンクが複数ある既存クラスターにノードを追加する方法は、複数のリンクを持つクラスターへのノードの追加 を参照してください。

リンクが複数ある既存クラスターのリンクを変更する方法は、既存クラスターへのリンクの追加および変更 を参照してください。

49.5. フェンシングの設定

クラスターの各ノードにフェンスデバイスを設定する必要があります。フェンスの設定コマンドおよびオプションに関する情報は Red Hat High Availability クラスターでのフェンシングの設定 を参照してください。

フェンシングの概要と、Red Hat High Availability クラスターにおけるフェンシングの重要性は Fencing in a Red Hat High Availability Cluster を参照してください。

注記

フェンスデバイスを設定する場合は、そのデバイスが、クラスター内のノードまたはデバイスと電源を共有しているかどうかに注意する必要があります。ノードとそのフェンスデバイスが電源を共有していると、その電源をフェンスできず、フェンスデバイスが失われた場合は、クラスターがそのノードをフェンスできない可能性があります。このようなクラスターには、フェンスデバイスおよびノードに冗長電源を提供するか、電源を共有しない冗長フェンスデバイスが存在する必要があります。SBD やストレージフェンシングなど、その他のフェンシング方法でも、分離した電源供給の停止時に冗長性を得られます。

手順

ここでは、ホスト名が **zapc.example.com** の APC 電源スイッチを使用してノードをフェンスし、**fence_apc_snmp** フェンスエージェントを使用します。どちらのノードも同じフェンスエージェントでフェンシングされるため、**pcmk_host_map** オプションを使用して、両方のフェンスデバイスを 1 つのリソースとして設定できます。

pcs stonith create コマンドを使用して、**stonith** リソースとしてデバイスを設定し、フェンスデバイスを作成します。以下のコマンドは、**z1.example.com** ノードおよび **z2.example.com** ノードの **fence_apc_snmp** フェンスエージェントを使用する、**stonith** リソース **myapc** を設定します。**pcmk_host_map** オプションにより、**z1.example.com** がポート 1 にマップされ、**z2.example.com** がポート 2 にマップされます。APC デバイスのログイン値とパスワードはいずれも **apc** です。デフォルトでは、このデバイスは各ノードに対して、60 秒間隔で監視を行います。

また、ノードのホスト名を指定する際に、IP アドレスを使用できます。

```
[root@z1 ~]# pcs stonith create myapc fence_apc_snmp ipaddr="zapc.example.com"
pcmk_host_map="z1.example.com:1;z2.example.com:2" login="apc" passwd="apc"
```

次のコマンドは、既存の STONITH デバイスのパラメーターを表示します。

```
[root@rh7-1 ~]# pcs stonith config myapc
 Resource: myapc (class=stonith type=fence_apc_snmp)
  Attributes: ipaddr=zapc.example.com pcmk_host_map=z1.example.com:1;z2.example.com:2
login=apc passwd=apc
  Operations: monitor interval=60s (myapc-monitor-interval-60s)
```

フェンスデバイスの設定後に、デバイスをテストする必要があります。フェンスデバイスをテストする方法は フェンスデバイスのテスト を参照してください。

注記

ネットワークインターフェイスを無効にしてフェンスデバイスのテストを実行しないでください。フェンシングが適切にテストされなくなります。

注記

フェンシングを設定してクラスターが起動すると、タイムアウトに到達していなくても、ネットワークの再起動時に、ネットワークを再起動するノードのフェンシングが発生します。このため、ノードで意図しないフェンシングが発生しないように、クラスターサービスの実行中はネットワークサービスを再起動しないでください。

49.6. クラスター設定のバックアップおよび復元

以下のコマンドは、tar アーカイブのクラスター設定のバックアップを作成し、バックアップからすべてのノードのクラスター設定ファイルを復元します。

手順

以下のコマンドを使用して、tar アーカイブでクラスター設定をバックアップします。ファイル名を指定しないと、標準出力が使用されます。

```
pcs config backup filename
```

注記

pcs config backup コマンドは、CIB に設定したようにクラスターの設定だけをバックアップします。リソースデーモンの設定は、このコマンドに含まれません。たとえば、クラスターで Apache リソースを設定すると、(CIB にある) リソース設定のバックアップが作成されますが、(/etc/httpd に設定したとおり) Apache デーモン設定と、そこで使用されるファイルのバックアップは作成されません。同様に、クラスターに設定されているデータベースリソースがある場合は、データベースそのもののバックアップが作成されません。ただし、データベースのリソース設定のバックアップ (CIB) は作成されます。

以下のコマンドを使用して、バックアップからすべてのクラスターノードのクラスター設定ファイルを復元します。**--local** オプションを指定すると、このコマンドを実行したノードでのみクラスター設定ファイルが復元されます。ファイル名を指定しないと、標準入力が使用されます。

```
pcs config restore [--local] [filename]
```

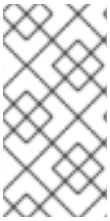
49.7. HIGH AVAILABILITY ADD-ON のポートの有効化

クラスターコンポーネントの理想的なファイアウォール設定は、ローカル環境によって異なります。ここでは、ノードに複数のネットワークインターフェイスがあるかどうか、オフホストのファイアウォールがあるかどうかを検討しないといけない場合があります。

firewalld デーモンを実行している場合は、次のコマンドを実行して、Red Hat High Availability Add-On で必要なポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

ローカルの状況に合わせて開くポートを変更することが必要になる場合があります。



注記

firewalld デーモンがシステムにインストールされているかどうかを確認する場合は、**rpm -q firewalld** コマンドを実行します。**firewalld** デーモンをインストールしている場合は、**firewall-cmd --state** コマンドを使用して、そのデーモンが実行しているかどうかを確認できます。

以下の表は、Red Hat High Availability Add-On で有効にするポートとポートの使用目的を説明します。

表49.1 High Availability Add-On で有効にするポート

ポート	必要になる場合
TCP 2224	すべてのノードに必要なデフォルトの **pcsd** ポートで必須 (pcsd Web UI で必要、ノード間通信で必須) です。**/etc/sysconfig/pcsd** ファイルの **PCSD_PORT** パラメーターを使用して **pcsd** を設定できます。 ポート 2224 を開いて、任意のノードの **pcs** が、それ自体も含め、クラスター内のすべてのノードに通信できるようにする必要があります。Booth クラスターチケットマネージャーまたはクォーラムデバイスを使用する場合は、Booth Arbiter、クォーラムデバイスなどのすべての関連ホストで、ポート 2224 を開く必要があります。

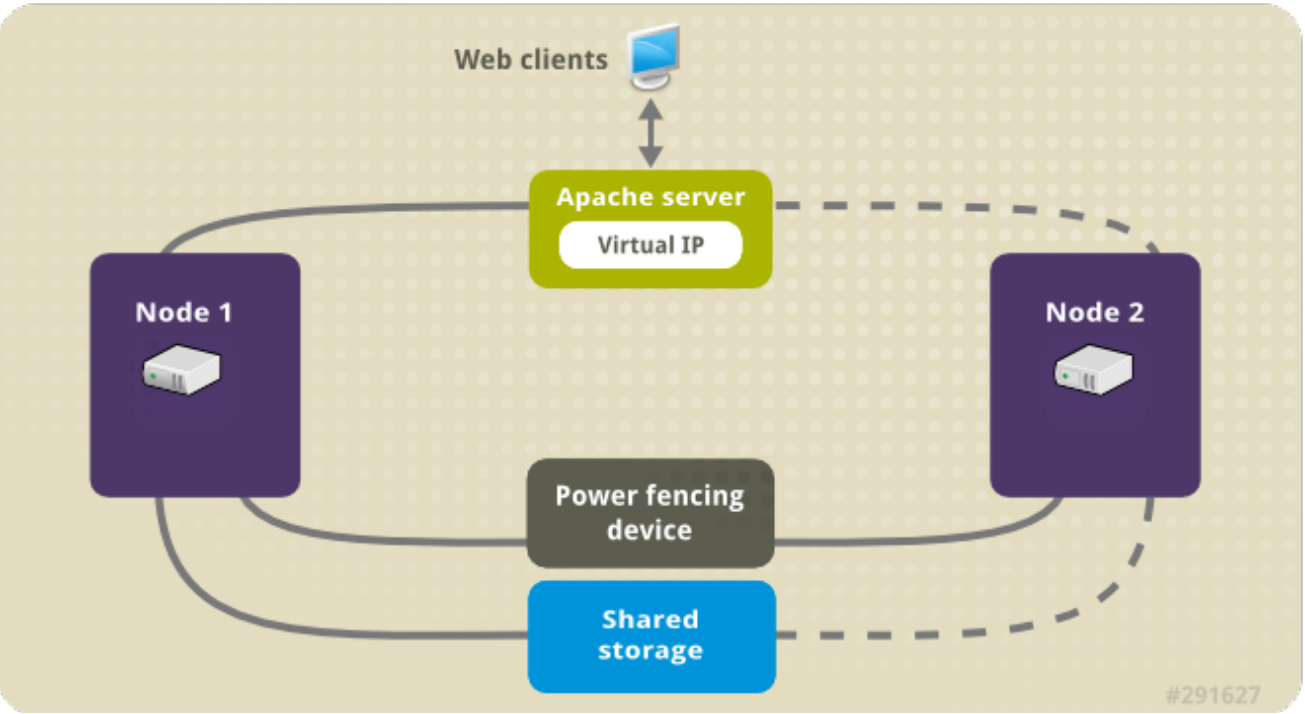
ポート	必要になる場合
TCP 3121	クラスターに Pacemaker リモートノードがある場合に、すべてのノードで必須です。 完全なクラスターノードにある Pacemaker の **pacemaker-based** デーモンは、ポート 3121 で Pacemaker リモートノードの **pacemaker_remoted** デーモンへの通信を行います。クラスター通信に別のインターフェイスを使用する場合は、そのインターフェイスでポートを開くことのみが必要になります。少なくとも、ポートは、Pacemaker リモートノードの全クラスターノードに対して開いている必要があります。ユーザーは完全なノードとリモートノード間でホストを変換する可能性があるか、ホストのネットワークを使用してコンテナー内でリモートノードを実行する可能性があるため、すべてのノードにポートを開くと便利になる場合があります。ノード以外のホストにポートを開く必要はありません。
TCP 5403	**corosync-qnetd** で、クォーラムデバイスを使用するクォーラムデバイスホストで必須です。デフォルト値は、**corosync-qnetd** コマンドの **-p** オプションで変更できます。
UDP 5404-5412	ノード間の通信を容易にするために corosync ノードで必須です。ポート 5404-5412 を開いて、任意のノードの **corosync** が、それ自体も含め、すべてのノードと通信できるようにする必要があります。
TCP 21064	DLM が必要なリソースがクラスターに含まれる場合に、すべてのノードで必須です (例: **GFS2**)。
TCP 9929、UDP 9929	Booth チケットマネージャーを使用してマルチサイトクラスターを確立する場合に、同じノードから接続するために、すべてのクラスターノードと、Booth 仲裁ノードで開いている必要があります。

第50章 RED HAT HIGH AVAILABILITY クラスターでのアクティブ/パッシブ APACHE HTTP サーバーの設定

以下の手順では、2 ノードの Red Hat Enterprise Linux High Availability Add-On クラスターでアクティブ/パッシブ Apache HTTP サーバーを設定します。このユースケースでは、クライアントはフローティング IP アドレスを使用して Apache HTTP サーバーにアクセスします。Web サーバーは、クラスターにある 2 つのノードのいずれかで実行します。Web サーバーが実行しているノードが正常に動作しなくなると、Web サーバーはクラスターの 2 番目のノードで再起動し、サービスの中断は最小限に抑えられます。

以下の図は、クラスターがネットワーク電源スイッチと共有ストレージで設定された 2 ノードの Red Hat High Availability クラスターであるクラスターの高レベルの概要を示しています。クライアントは仮想 IP を使用して Apache HTTP サーバーにアクセスするため、クラスターノードはパブリックネットワークに接続されます。Apache サーバーは、ノード 1 またはノード 2 のいずれかで実行します。いずれのノードも、Apache のデータが保持されるストレージにアクセスできます。この図では、Web サーバーが ノード 1 で実行しており、ノード 1 が正常に動作しなくなると、ノード 2 がサーバーを実行できます。

図50.1 2 ノードの Red Hat High Availability クラスターの Apache



このユースケースでは、システムに以下のコンポーネントが必要です。

- 各ノードに電源フェンスが設定されている 2 ノードの Red Hat High Availability クラスター。プライベートネットワークが推奨されますが、必須ではありません。この手順では、Pacemaker を使用した Red Hat High Availability クラスターの作成 で説明されているサンプルのクラスターを使用します。
- Apache に必要なパブリック仮想 IP アドレス。
- iSCSI、ファイバーチャネル、またはその他の共有ネットワークデバイスを使用する、クラスター内のノードの共有ストレージ。

クラスターは、Web サーバーで必要な LVM リソース、ファイルシステムリソース、IP アドレスリソース、Web サーバーリソースなどのクラスターコンポーネントを含む Apache リソースグループで設定されます。このリソースグループは、クラスター内のあるノードから別のノードへのフェイルオーバーが

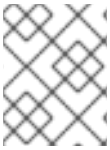
可能なため、いずれのノードでも Web サーバーを実行できます。このクラスターのリソースグループを作成する前に、以下の手順を実行します。

1. 論理ボリューム **my_lv** 上に XFS ファイルシステムを設定します。
2. Web サーバーを設定します。

上記の手順をすべて完了したら、リソースグループと、そのグループに追加するリソースを作成します。

50.1. PACEMAKER クラスターで XFS ファイルシステムを使用して LVM ボリュームを設定する

この手順では、クラスターのノード間で共有されているストレージに LVM 論理ボリュームを作成します。



注記

LVM ボリュームと、クラスターノードで使用するパーティションおよびデバイスは、クラスターノード以外には接続しないでください。

次の手順では、LVM 論理ボリュームを作成し、そのボリューム上に Pacemaker クラスターで使用する XFS ファイルシステムを作成します。この例では、LVM 論理ボリュームを作成する LVM 物理ボリュームを保管するのに、共有パーティション **/dev/sdb1** が使用されます。

手順

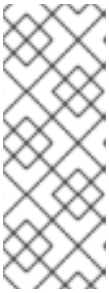
1. クラスターの両ノードで以下の手順を実行し、LVM システム ID の値を、システムの **uname** 識別子の値に設定します。LVM システム ID を使用すると、クラスターのみがボリュームグループをアクティブにできるようになります。
   a. **/etc/lvm/lvm.conf** 設定ファイルの **system_id_source** 設定オプションを **uname** に設定します。

   ```
   # Configuration option global/system_id_source.
   system_id_source = "uname"
   ```

   b. ノードの LVM システム ID が、ノードの **uname** に一致することを確認します。

   ```
   # lvm systemid
     system ID: z1.example.com
   # uname -n
     z1.example.com
   ```

2. LVM ボリュームを作成し、そのボリューム上に XFS ファイルシステムを作成します。**/dev/sdb1** パーティションは共有されるストレージであるため、この手順のこの部分は、1 つのノードでのみ実行してください。



注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターによって使用されるリモート物理ボリュームの起動順序の設定については、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

a. パーティション **/dev/sdb1** に LVM 物理ボリュームを作成します。

```
[root@z1 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターによって使用されるリモート物理ボリュームの起動順序の設定については、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

b. 物理ボリューム **/dev/sdb1** で構成されるボリュームグループ **my_vg** を作成します。RHEL 8.5 以降では、**--setautoactivation n** フラグを指定して、クラスターで Pacemaker が管理するボリュームグループが起動時に自動的にアクティブにならないようにします。作成する LVM ボリュームに既存のボリュームグループを使用している場合は、ボリュームグループで **vgchange --setautoactivation n** コマンドを使用して、このフラグをリセットできます。

```
[root@z1 ~]# vgcreate --setautoactivation n my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

RHEL 8.4 以前の場合は、以下のコマンドでボリュームグループを作成します。

```
[root@z1 ~]# vgcreate my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

RHEL 8.4 以前の起動時に、クラスターの Pacemaker が管理するボリュームグループが自動的にアクティベートされないようにする方法は、ボリュームグループが複数のクラスターノードでアクティベートされていないこと (RHEL 8.4 以前) を参照してください。

c. 新規ボリュームグループには、実行中のノードで、かつボリュームグループの作成元であるノードのシステム ID があることを確認します。

```
[root@z1 ~]# vgs -o+systemid
  VG    #PV #LV #SN Attr   VSize  VFree  System ID
  my_vg   1   0   0 wz--n- <1.82t <1.82t z1.example.com
```

d. ボリュームグループ **my_vg** を使用して、論理ボリュームを作成します。

```
[root@z1 ~]# lvcreate -L450 -n my_lv my_vg
  Rounding up size to full physical extent 452.00 MiB
  Logical volume "my_lv" created
```

lvs コマンドを使用して論理ボリュームを表示してみます。

```
[root@z1 ~]# lvs
  LV      VG      Attr      LSize   Pool Origin Data%  Move Log Copy%  Convert
  my_lv   my_vg   -wi-a---- 452.00m
  ...
```

e. 論理ボリューム **my_lv** 上に XFS ファイルシステムを作成します。

```
[root@z1 ~]# mkfs.xfs /dev/my_vg/my_lv
meta-data=/dev/my_vg/my_lv       isize=512    agcount=4, agsize=28928 blks
         =                       sectsz=512   attr=2, projid32bit=1
...
```

3. RHEL 8.5 以降でサポートされる LVM デバイスファイルを使用している場合は、クラスターの 2 番目のノードのデバイスファイルに共有デバイスを追加します。

```
[root@z2 ~]# lvmdevices --adddev /dev/sdb1
```

50.2. 複数のクラスターノードでボリュームグループがアクティブにならないようにする方法 (RHEL 8.4 以前)

以下の手順で、クラスター内の Pacemaker が管理するボリュームグループが起動時に自動でアクティベートされないようにすることができます。Pacemaker ではなく、システムの起動時にボリュームグループが自動的にアクティブになる場合は、ボリュームグループが同時に複数のノードでアクティブになるリスクがあり、ボリュームグループのメタデータが破損する可能性があります。

注記

RHEL 8.5 以降では、**vgcreate** コマンドに **--setautoactivation n** フラグを指定して、ボリュームグループを作成する場合、ボリュームグループの自動アクティベーションを無効にすることができます (Pacemaker クラスターで XFS ファイルシステムを使用して、LVM ボリュームを設定する を参照)。

この手順では、**/etc/lvm/lvm.conf** 設定ファイルの **auto_activation_volume_list** エントリーを変更します。**auto_activation_volume_list** エントリーは、自動アクティブ化を特定の論理ボリュームに制限するために使用されます。**auto_activation_volume_list** を空のリストに設定すると、自動アクティベーションは完全に無効になります。

ノードのローカルにある root およびホームディレクトリーに関係のあるボリュームグループなど、Pacemaker で管理されていないまたは共有されていないローカルのボリュームは、**auto_activation_volume_list** に含める必要があります。クラスターマネージャーが管理するボリュームグループはすべて、**auto_activation_volume_list** エントリーから除外する必要があります。

手順

クラスター内の各ノードで以下の手順を実行します。

1. 以下のコマンドを使用して、ローカルストレージに現在設定されているボリュームグループを

確認します。これにより、現在設定されているボリュームグループのリストが出力されます。このノードの root とホームディレクトリーに、別のボリュームグループの領域を割り当てると、この例のように以下のボリュームが出力に表示されます。

```
# vgs --noheadings -o vg_name
  my_vg
  rhel_home
  rhel_root
```

2. **/etc/lvm/lvm.conf** 設定ファイルの **auto_activation_volume_list** へのエントリーとして、**my_vg** 以外のボリュームグループ (クラスターに定義したボリュームグループ) を追加します。
   たとえば、root とホームディレクトリーに別のボリュームグループの領域が割り当てられている場合は、以下のように **lvm.conf** ファイルの **auto_activation_volume_list** 行をアンコメントし、これらのボリュームグループをエントリーとして **auto_activation_volume_list** に追加します。クラスターに対してだけ定義したボリュームグループ (この例では **my_vg**) は、このリストは含まれない点に注意してください。

```
auto_activation_volume_list = [ "rhel_root", "rhel_home" ]
```

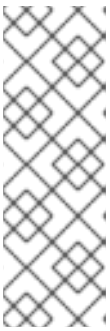
注記

クラスターマネージャーの外部でアクティベートするノードにローカルボリュームグループが存在しない場合は、 **auto_activation_volume_list** エントリーを **auto_activation_volume_list = []** として初期化する必要があります。

3. **initramfs** ブートイメージを再構築して、クラスターが制御するボリュームグループがブートイメージによりアクティベートされないようにします。以下のコマンドを使用して、**initramfs** デバイスを更新します。このコマンドが完了するまで最大1分かかる場合があります。

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

4. ノードを再起動します。



注記

ブートイメージを作成したノードを起動してから、新しい Linux カーネルをインストールした場合は、新しい **initrd** イメージは、作成時に実行していたカーネル用で、ノードの再起動時に実行している新しいカーネル用ではありません。再起動の前後で **uname -r** コマンドを使用して実行しているカーネルリリースを確認し必ず正しい **initrd** デバイスを使用するよう注意してください。リリースが同じでない場合には、新規カーネルで再起動した後に **initrd** ファイルを更新して、ノードを再起動します。

5. ノードが再起動したら **pcs cluster status** コマンドを実行し、クラスターサービスがそのノードで再度開始されたかどうかを確認します。**Error: cluster is not running on this node** というメッセージが表示される場合は、以下のコマンドを入力します。

```
# pcs cluster start
```

または、クラスターの各ノードを再起動して、クラスターの全ノードでクラスターサービスを開始するまで待機するには、次のコマンドを使用します。

```
# pcs cluster start --all
```

50.3. APACHE HTTP サーバーの設定

次の手順に従って Apache HTTP サーバーを設定します。

手順

1. クラスターの各ノードに、Apache HTTP サーバーがインストールされていることを確認します。Apache HTTP サーバーのステータスを確認するには、クラスターに **wget** ツールがインストールされている必要があります。
   各ノードで、以下のコマンドを実行します。

   ```
   # yum install -y httpd wget
   ```

   firewalld デーモンを実行している場合は、クラスターの各ノードで、Red Hat High Availability Add-On に必要なポートを有効にし、**httpd** の実行に必要なポートを有効にします。以下の例では、一般からのアクセス用に **httpd** のポートを有効にしていますが、**httpd** 用に有効にする具体的なポートは、本番環境のユースケースでは異なる場合があります。

   ```
   # firewall-cmd --permanent --add-service=http
   # firewall-cmd --permanent --zone=public --add-service=http
   # firewall-cmd --reload
   ```

2. Apache リソースエージェントが、クラスターの各ノードで Apache のステータスを取得できるようにするため、既存の設定に以下の内容を追加して、ステータスサーバーの URL を有効にします。

   ```
   # cat <<-END > /etc/httpd/conf.d/status.conf
   <Location /server-status>
      SetHandler server-status
      Require local
   </Location>
   END
   ```

3. **apache** リソースエージェントを使用して Apache を管理する場合は **systemd** が使用されません。そのため、Apache のリロードに **systemctl** が使用されないようにするため、Apache によって提供される **logrotate** スクリプトを編集する必要があります。
   クラスター内の各ノード上で、**/etc/logrotate.d/httpd** ファイルから以下の行を削除します。

   ```
   /bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
   ```

   削除した行を以下の 3 行に置き換えます。

   ```
   /usr/bin/test -f /run/httpd.pid >/dev/null 2>/dev/null &&
   /usr/bin/ps -q $(/usr/bin/cat /run/httpd.pid) >/dev/null 2>/dev/null &&
   /usr/sbin/httpd -f /etc/httpd/conf/httpd.conf \
   -c "PidFile /run/httpd.pid" -k graceful > /dev/null 2>/dev/null || true
   ```

4. Apache で提供する Web ページを作成します。
   クラスター内の 1 つのノードで、XFS ファイルシステムを使用した LVM ボリュームの設定 で作成した論理ボリュームがアクティブになっていることを確認し、作成したファイルシステム

をその論理ボリュームにマウントし、そのファイルシステムにファイル **index.html** を作成します。次に、ファイルシステムをアンマウントします。

```
# lvchange -ay my_vg/my_lv
# mount /dev/my_vg/my_lv /var/www/
# mkdir /var/www/html
# mkdir /var/www/cgi-bin
# mkdir /var/www/error
# restorecon -R /var/www
# cat <<-END >/var/www/html/index.html
<html>
<body>Hello</body>
</html>
END
# umount /var/www
```

50.4. リソースおよびリソースグループの作成

次の手順でクラスターのリソースを作成します。すべてのリソースが必ず同じノードで実行するように、このリソースを、リソースグループ **apachegroup** に追加します。作成するリソースは以下のとおりで、開始する順に記載されています。

1. XFS ファイルシステムを使用した LVM ボリュームの設定 で作成した LVM ボリュームグループを使用する **my_lvm** という名前の **LVM-activate** リソース。
2. XFS ファイルシステムを使用した LVM ボリュームの設定 で作成したファイルシステムデバイス **/dev/my_vg/my_lv** を使用する、**my_fs** という名前の **Filesystem** リソース。
3. **apachegroup** リソースグループのフローティング IP アドレスである **IPaddr2** リソース。物理ノードに関連付けられている IP アドレスは使用できません。**IPaddr2** リソースの NIC デバイスを指定していない場合は、そのノードに静的に割り当てられている IP アドレスの 1 つと同じネットワークにフローティング IP が存在していないと、フローティング IP アドレスを割り当てる NIC デバイスが適切に検出されません。
4. Apache HTTP サーバーの設定 で定義した **index.html** ファイルと Apache 設定を使用する **Website** という名前の **apache** リソース

以下の手順で、**apachegroup** リソースグループと、このグループに追加するリソースを作成します。リソースは、グループに追加された順序で起動し、その逆の順序で停止します。この手順は、クラスター内のいずれかのノードで実行してください。

手順

1. 次のコマンドは、**LVM が有効** なリソース **my_lvm** を作成します。リソースグループ **apachegroup** は存在しないため、このコマンドによりリソースグループが作成されます。

注記

アクティブ/パッシブの HA 設定で、同じ LVM ボリュームグループを使用する **LVM が有効** なリソースを複数設定するとデータが破損する場合があるため、そのようなリソースは 1 つ以上設定しないでください。また、**LVM が有効** なリソースは、アクティブ/パッシブの HA 設定のクローンリソースとして設定しないでください。

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group apachegroup
```

リソースを作成すると、そのリソースは自動的に起動します。以下のコマンドを使用すると、リソースが作成され、起動していることを確認できます。

```
# pcs resource status
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): Started
```

pcs resource disable と **pcs resource enable** のコマンドを使用すると手作業によるリソースの停止と起動をリソースごと個別に行うことができます。

2. 以下のコマンドでは、設定に必要な残りのリソースを作成し、作成したリソースを既存の **apachegroup** リソースグループに追加します。

```
[root@z1 ~]# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv"
directory="/var/www" fstype="xfs" --group apachegroup

[root@z1 ~]# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 cidr_netmask=24 --
group apachegroup

[root@z1 ~]# pcs resource create Website apache
configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-status" --
group apachegroup
```

3. リソースと、そのリソースを含むリソースグループの作成が完了したら、クラスターのステータスを確認します。4 つのリソースがすべて同じノードで実行していることに注意してください。

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): Started z1.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z1.example.com
     Website (ocf::heartbeat:apache): Started z1.example.com
```

クラスターのフェンスデバイスを設定していないと、リソースがデフォルトで起動しません。

4. クラスターが稼働したら、ブラウザーで、**IPaddr2** リソースとして定義した IP アドレスを指定して、Hello と単語が表示されるサンプル表示を確認します。

```
Hello
```

設定したリソースが実行していない場合は、**pcs resource debug-start resource** コマンドを実行して、リソースの設定をテストします。

50.5. リソース設定のテスト

次の手順でクラスター内のリソース設定をテストします。

リソースおよびリソースグループの作成 で説明するクラスターのステータス表示では、すべてのリソースが **z1.example.com** ノードで実行されます。以下の手順に従い、1 番目のノードを **スタンバイ** モードにし、リソースグループが **z2.example.com** ノードにフェイルオーバーするかどうかをテストします。1 番目のノードをスタンバイモードにすると、このノードはリソースをホストできなくなります。

手順

1. 以下のコマンドは、**z1.example.com** ノードを **スタンバイ** モードにします。

```
[root@z1 ~]# pcs node standby z1.example.com
```

2. **z1** を **スタンバイ** モードにしたら、クラスターのステータスを確認します。リソースはすべて **z2** で実行しているはずです。

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Node z1.example.com (1): standby
Online: [ z2.example.com ]

Full list of resources:

 myapc (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): Started z2.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z2.example.com
     Website (ocf::heartbeat:apache): Started z2.example.com
```

定義している IP アドレスの Web サイトは、中断せず表示されているはずです。

3. **スタンバイ** モードから **z1** を削除するには、以下のコマンドを実行します。

```
[root@z1 ~]# pcs node unstandby z1.example.com
```

注記

ノードを **スタンバイ** モードから削除しても、リソースはそのノードにフェイルオーバーしません。これは、リソースの **resource-stickiness** の値により異なります。**resource-stickiness** メタ属性の詳細は、現在のノードを優先するようにリソースを設定 を参照してください。

第51章 RED HAT HIGH AVAILABILITY クラスターのアクティブ/パッシブな NFS サーバーの設定

Red Hat High Availability Add-On は、共有ストレージを使用して Red Hat Enterprise Linux High Availability アドオンクラスターで高可用性アクティブ/パッシブ NFS サーバーを実行するためのサポートを提供します。次の例では、クライアントが Floating IP アドレスを介して NFS ファイルシステムにアクセスする 2 ノードクラスターを設定します。NFS サービスは、クラスターにある 2 つのノードのいずれかで実行します。NFS サーバーが実行しているノードが正常に動作しなくなると、NFS サーバーはクラスターの 2 番目のノードで再起動し、サービスの中断が最小限に抑えられます。

このユースケースでは、システムに以下のコンポーネントが必要です。

- 各ノードに電源フェンスが設定されている 2 ノードの Red Hat High Availability クラスター。プライベートネットワークが推奨されますが、必須ではありません。この手順では、Pacemaker を使用した Red Hat High Availability クラスターの作成 で説明されているサンプルのクラスターを使用します。
- NFS サーバーに必要なパブリック仮想 IP アドレス。
- iSCSI、ファイバーチャネル、またはその他の共有ネットワークデバイスを使用する、クラスター内のノードの共有ストレージ。

既存の 2 ノードの Red Hat Enterprise Linux High Availability クラスターで高可用性アクティブ/パッシブ NFS サーバーを設定するには、以下の手順を実行する必要があります。

1. クラスターのノード用に、共有ストレージの LVM 論理ボリュームにファイルシステムを設定します。
2. 共有ストレージの LVM 論理ボリュームで NFS 共有を設定する。
3. クラスターリソースを作成する。
4. 設定した NFS サーバーをテストする。

51.1. PACEMAKER クラスターで XFS ファイルシステムを使用して LVM ボリュームを設定する

この手順では、クラスターのノード間で共有されているストレージに LVM 論理ボリュームを作成します。

注記

LVM ボリュームと、クラスターノードで使用するパーティションおよびデバイスは、クラスターノード以外には接続しないでください。

次の手順では、LVM 論理ボリュームを作成し、そのボリューム上に Pacemaker クラスターで使用する XFS ファイルシステムを作成します。この例では、LVM 論理ボリュームを作成する LVM 物理ボリュームを保管するのに、共有パーティション **/dev/sdb1** が使用されます。

手順

1. クラスターの両ノードで以下の手順を実行し、LVM システム ID の値を、システムの **uname** 識別子の値に設定します。LVM システム ID を使用すると、クラスターのみがボリュームグループをアクティブにできるようになります。

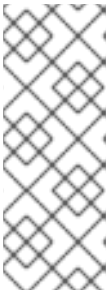
a. **/etc/lvm/lvm.conf** 設定ファイルの **system_id_source** 設定オプションを **uname** に設定します。

```
# Configuration option global/system_id_source.
system_id_source = "uname"
```

b. ノードの LVM システム ID が、ノードの **uname** に一致することを確認します。

```
# lvm systemid
  system ID: z1.example.com
# uname -n
  z1.example.com
```

2. LVM ボリュームを作成し、そのボリューム上に XFS ファイルシステムを作成します。**/dev/sdb1** パーティションは共有されるストレージであるため、この手順のこの部分は、1 つのノードでのみ実行してください。



注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターによって使用されるリモート物理ボリュームの起動順序の設定については、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

a. パーティション **/dev/sdb1** に LVM 物理ボリュームを作成します。

```
[root@z1 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターによって使用されるリモート物理ボリュームの起動順序の設定については、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

b. 物理ボリューム **/dev/sdb1** で構成されるボリュームグループ **my_vg** を作成します。RHEL 8.5 以降では、**--setautoactivation n** フラグを指定して、クラスターで Pacemaker が管理するボリュームグループが起動時に自動的にアクティブにならないようにします。作成する LVM ボリュームに既存のボリュームグループを使用している場合は、ボリュームグループで **vgchange --setautoactivation n** コマンドを使用して、このフラグをリセットできます。

```
[root@z1 ~]# vgcreate --setautoactivation n my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

RHEL 8.4 以前の場合は、以下のコマンドでボリュームグループを作成します。

```
[root@z1 ~]# vgcreate my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

RHEL 8.4 以前の起動時に、クラスターの Pacemaker が管理するボリュームグループが自動的にアクティベートされないようにする方法は、ボリュームグループが複数のクラスターノードでアクティベートされていないこと (RHEL 8.4 以前) を参照してください。

c. 新規ボリュームグループには、実行中のノードで、かつボリュームグループの作成元であるノードのシステム ID があることを確認します。

```
[root@z1 ~]# vgs -o+systemid
  VG    #PV #LV #SN Attr   VSize  VFree  System ID
  my_vg   1   0   0 wz--n- <1.82t <1.82t z1.example.com
```

d. ボリュームグループ **my_vg** を使用して、論理ボリュームを作成します。

```
[root@z1 ~]# lvcreate -L450 -n my_lv my_vg
  Rounding up size to full physical extent 452.00 MiB
  Logical volume "my_lv" created
```

lvs コマンドを使用して論理ボリュームを表示してみます。

```
[root@z1 ~]# lvs
  LV      VG      Attr      LSize   Pool Origin Data%  Move Log Copy%  Convert
  my_lv   my_vg   -wi-a---- 452.00m
  ...
```

e. 論理ボリューム **my_lv** 上に XFS ファイルシステムを作成します。

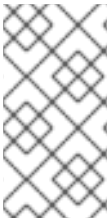
```
[root@z1 ~]# mkfs.xfs /dev/my_vg/my_lv
meta-data=/dev/my_vg/my_lv       isize=512    agcount=4, agsize=28928 blks
         =                       sectsz=512   attr=2, projid32bit=1
...
```

3. RHEL 8.5 以降でサポートされる LVM デバイスファイルを使用している場合は、クラスターの 2 番目のノードのデバイスファイルに共有デバイスを追加します。

```
[root@z2 ~]# lvmdevices --adddev /dev/sdb1
```

51.2. 複数のクラスターノードでボリュームグループがアクティブにならないようにする方法 (RHEL 8.4 以前)

以下の手順で、クラスター内の Pacemaker が管理するボリュームグループが起動時に自動でアクティベートされないようにすることができます。Pacemaker ではなく、システムの起動時にボリュームグループが自動的にアクティブになる場合は、ボリュームグループが同時に複数のノードでアクティブになるリスクがあり、ボリュームグループのメタデータが破損する可能性があります。



注記

RHEL 8.5 以降では、**vgcreate** コマンドに **--setautoactivation n** フラグを指定して、ボリュームグループを作成する場合、ボリュームグループの自動アクティベーションを無効にすることができます (Pacemaker クラスターで XFS ファイルシステムを使用して、LVM ボリュームを設定する を参照)。

この手順では、**/etc/lvm/lvm.conf** 設定ファイルの **auto_activation_volume_list** エントリーを変更します。**auto_activation_volume_list** エントリーは、自動アクティブ化を特定の論理ボリュームに制限するために使用されます。**auto_activation_volume_list** を空のリストに設定すると、自動アクティベーションは完全に無効になります。

ノードのローカルにある root およびホームディレクトリーに関係のあるボリュームグループなど、Pacemaker で管理されていないまたは共有されていないローカルのボリュームは、**auto_activation_volume_list** に含める必要があります。クラスターマネージャーが管理するボリュームグループはすべて、**auto_activation_volume_list** エントリーから除外する必要があります。

手順

クラスター内の各ノードで以下の手順を実行します。

1. 以下のコマンドを使用して、ローカルストレージに現在設定されているボリュームグループを確認します。これにより、現在設定されているボリュームグループのリストが出力されます。このノードの root とホームディレクトリーに、別のボリュームグループの領域を割り当てると、この例のように以下のボリュームが出力に表示されます。

```
# vgs --noheadings -o vg_name
  my_vg
  rhel_home
  rhel_root
```

2. **/etc/lvm/lvm.conf** 設定ファイルの **auto_activation_volume_list** へのエントリーとして、**my_vg** 以外のボリュームグループ (クラスターに定義したボリュームグループ) を追加します。
たとえば、root とホームディレクトリーに別のボリュームグループの領域が割り当てられている場合は、以下のように **lvm.conf** ファイルの **auto_activation_volume_list** 行をアンコメントし、これらのボリュームグループをエントリーとして **auto_activation_volume_list** に追加します。クラスターに対してだけ定義したボリュームグループ (この例では **my_vg**) は、このリストは含まれない点に注意してください。

```
auto_activation_volume_list = [ "rhel_root", "rhel_home" ]
```

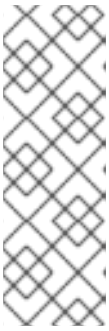
注記

クラスターマネージャーの外部でアクティベートするノードにローカルボリュームグループが存在しない場合は、 **auto_activation_volume_list** エントリーを **auto_activation_volume_list = []** として初期化する必要があります。

3. **initramfs** ブートイメージを再構築して、クラスターが制御するボリュームグループがブートイメージによりアクティベートされないようにします。以下のコマンドを使用して、**initramfs** デバイスを更新します。このコマンドが完了するまで最大 1 分かかる場合があります。

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

4. ノードを再起動します。



注記

ブートイメージを作成したノードを起動してから、新しい Linux カーネルをインストールした場合は、新しい **initrd** イメージは、作成時に実行していたカーネル用で、ノードの再起動時に実行している新しいカーネル用ではありません。再起動の前後で **uname -r** コマンドを使用して実行しているカーネルリリースを確認し必ず正しい **initrd** デバイスを使用するよう注意してください。リリースが同じでない場合には、新規カーネルで再起動した後に **initrd** ファイルを更新して、ノードを再起動します。

5. ノードが再起動したら **pcs cluster status** コマンドを実行し、クラスターサービスがそのノードで再度開始されたかどうかを確認します。**Error: cluster is not running on this node** というメッセージが表示される場合は、以下のコマンドを入力します。

```
# pcs cluster start
```

または、クラスターの各ノードを再起動して、クラスターの全ノードでクラスターサービスを開始するまで待機するには、次のコマンドを使用します。

```
# pcs cluster start --all
```

51.3. NFS 共有の設定

次の手順では、NFS サービスのフェイルオーバー用の NFS 共有を設定します。

手順

1. クラスターの両方のノードに、**/nfsshare** ディレクトリーを作成します。

```
# mkdir /nfsshare
```

2. クラスター内の 1 ノードで、以下の手順を行います。

   a. XFS ファイルシステムを使用した LVM ボリュームの設定 で作成した論理ボリュームを確認します。アクティブ化されたら、**/nfsshare** ディレクトリーの論理ボリューム上に作成したファイルシステムをマウントします。

```
[root@z1 ~]# lvchange -ay my_vg/my_lv
[root@z1 ~]# mount /dev/my_vg/my_lv /nfsshare
```

   b. **/nfsshare** ディレクトリー内に **exports** ディレクトリーツリーを作成します。

```
[root@z1 ~]# mkdir -p /nfsshare/exports
[root@z1 ~]# mkdir -p /nfsshare/exports/export1
[root@z1 ~]# mkdir -p /nfsshare/exports/export2
```

   c. NFS クライアントがアクセスするファイルを、**exports** ディレクトリーに置きます。この例では、**clientdatafile1** および **clientdatafile2** という名前のテストファイルを作成します。

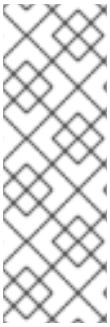
```
[root@z1 ~]# touch /nfsshare/exports/export1/clientdatafile1
[root@z1 ~]# touch /nfsshare/exports/export2/clientdatafile2
```

d. ファイルシステムをアンマウントし、LVM ボリュームグループを非アクティブ化します。

```
[root@z1 ~]# umount /dev/my_vg/my_lv
[root@z1 ~]# vgchange -an my_vg
```

51.4. クラスターの NFS サーバーへリソースおよびリソースグループを設定

以下の手順で、クラスター内の NFS サーバーのクラスターリソースを設定します。



注記

クラスターにフェンスデバイスを設定していないと、リソースはデフォルトでは起動しないことに注意してください。

設定したリソースが実行していない場合は、**pcs resource debug-start resource** コマンドを実行して、リソースの設定をテストします。このコマンドは、クラスターの制御や認識の範囲外でサービスを起動します。設定したリソースが再稼働したら、**pcs resource cleanup resource** を実行して、クラスターが更新を認識するようにします。

手順

以下の手順では、システムリソースを設定します。これらのリソースがすべて同じノードで実行するように、これらのリソースはリソースグループ **nfsgroup** に含まれます。リソースは、グループに追加された順序で起動し、その逆の順序で停止します。この手順は、クラスター内のいずれかのノードで実行してください。

1. LVM が有効なリソース **my_lvm** を作成します。リソースグループ **my_lvm** は存在しないため、このコマンドによりリソースグループが作成されます。

警告

データ破損のリスクとなるため、アクティブ/パッシブの HA 設定で、同じ LVM ボリュームグループを使用する **LVM が有効** なリソースを複数設定しないでください。また、**LVM が有効** なリソースは、アクティブ/パッシブの HA 設定のクローンリソースとして設定しないでください。

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group nfsgroup
```

2. クラスターのステータスを確認し、リソースが実行していることを確認します。

```
root@z1 ~]#  pcs status
Cluster name: my_cluster
Last updated: Thu Jan  8 11:13:17 2015
Last change: Thu Jan  8 11:13:08 2015
```

```
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.12-a14efad
2 Nodes configured
3 Resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com

PCSD Status:
  z1.example.com: Online
  z2.example.com: Online

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

3. クラスターに **Filesystem** リソースを設定します。
次のコマンドは、**nfsshare** という名前の XFS **Filesystem** リソースを **nfsgroup** リソースグループの一部として設定します。このファイルシステムは、XFS ファイルシステムを使用した LVM ボリュームの設定 で作成した LVM ボリュームグループと XFS ファイルシステムを使用し、NFS 共有の設定 で作成した **/nfsshare** ディレクトリーにマウントされます。

```
[root@z1 ~]# pcs resource create nfsshare Filesystem device=/dev/my_vg/my_lv
directory=/nfsshare fstype=xfs --group nfsgroup
```

options=options パラメーターを使用すると、**Filesystem** リソースのリソース設定の一部としてマウントオプションを指定できます。すべての設定オプションを確認する場合は、**pcs resource describe Filesystem** コマンドを実行します。

4. **my_lvm** リソースおよび **nfsshare** リソースが実行していることを確認します。

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
...
```

5. **nfsgroup** リソースグループに、**nfs-daemon** という名前の **nfsserver** リソースを作成します。

注記

nfsserver リソースを使用して、**nfs_shared_infodir** パラメーターを指定できます。これは、NFS サーバーが、NFS 関連のステートフル情報を保管するのに使用するディレクトリーです。

この属性は、このエクスポートのコレクションで作成した **Filesystem** リソースのいずれかのサブディレクトリーに設定することが推奨されます。これにより、NFS サーバーは、このリソースグループを再配置する必要がある場合に別のノードで使用できるデバイスに、ステートフル情報を保存します。この例では、以下のように設定されています。

- **/nfsshare** は、**Filesystem** リソースにより管理される shared-storage ディレクトリーです。
- **/nfsshare/exports/export1** および **/nfsshare/exports/export2** は、エクスポートディレクトリーです。
- **/nfsshare/nfsinfo** は、**nfsserver** リソースの共有情報ディレクトリーです。

```
[root@z1 ~]# pcs resource create nfs-daemon nfsserver
nfs_shared_infodir=/nfsshare/nfsinfo nfs_no_notify=true --group nfsgroup

[root@z1 ~]# pcs status
...
```

6. **exportfs** リソースを追加して、**/nfsshare/exports** ディレクトリーをエクスポートします。このリソースは、**nfsgroup** リソースグループに含まれます。これにより、NFSv4 クライアントの仮想ディレクトリーが構築されます。このエクスポートには、NFSv3 クライアントもアクセスできます。

注記

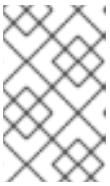
fsid=0 オプションは、NFSv4 クライアントに仮想ディレクトリーを作成する場合にのみ必要です。詳細は、How do I configure the fsid option in an NFS server's /etc/exports file? を参照してください。

```
[root@z1 ~]# pcs resource create nfs-root exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports fsid=0 --group nfsgroup

[root@z1 ~]# pcs resource create nfs-export1 exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports/export1 fsid=1 --group nfsgroup

[root@z1 ~]# pcs resource create nfs-export2 exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports/export2 fsid=2 --group nfsgroup
```

7. NFS 共有にアクセスするために、NFS クライアントが使用するフローティング IP アドレスリソースを追加します。このリソースは、リソースグループ **nfsgroup** に含まれます。このデプロイメント例では、192.168.122.200 をフローティング IP アドレスとして使用します。

```
[root@z1 ~]# pcs resource create nfs_ip IPaddr2 ip=192.168.122.200 cidr_netmask=24 --group nfsgroup
```

8. NFS デプロイメント全体が初期化されたら、NFSv3 の再起動通知を送信する **nfsnotify** リソースを追加します。このリソースは、リソースグループ **nfsgroup** に含まれます。



注記

NFS の通知が適切に処理されるようにするには、フローティング IP アドレスにホスト名が関連付けられており、それが NFS サーバーと NFS クライアントで同じである必要があります。

```
[root@z1 ~]# pcs resource create nfs-notify nfsnotify source_host=192.168.122.200 --group nfsgroup
```

9. リソースとリソースの制約を作成したら、クラスターのステータスを確認できます。すべてのリソースが同じノードで実行していることに注意してください。

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z1.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export1        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export2        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z1.example.com
     nfs-notify (ocf::heartbeat:nfsnotify):     Started z1.example.com
...
```

51.5. NFS リソース設定のテスト

以下の手順を使用して、高可用性クラスターで NFS リソース設定を検証できます。NFSv3 または NFSv4 のいずれかで、エクスポートされたファイルシステムをマウントできるはずです。

51.5.1. NFS エクスポートのテスト

1. クラスターノードで **firewalld** デーモンを実行している場合は、システムが NFS アクセスに必要とするポートがすべてのノードで有効になっていることを確認してください。

2. デプロイメントと同じネットワークにあるクラスター外部のノードで NFS 共有をマウントして、NFS 共有が表示されることを確認します。この例では、192.168.122.0/24 ネットワークを使用します。

```
# showmount -e 192.168.122.200
Export list for 192.168.122.200:
/nfsshare/exports/export1 192.168.122.0/255.255.255.0
/nfsshare/exports         192.168.122.0/255.255.255.0
/nfsshare/exports/export2 192.168.122.0/255.255.255.0
```

3. NFSv4 で NFS 共有をマウントできることを確認する場合は、クライアントノードのディレクトリーに NFS 共有をマウントします。マウントしたら、エクスポートディレクトリーの内容が表示されることを確認します。テスト後に共有をアンマウントします。

```
# mkdir nfsshare
# mount -o "vers=4" 192.168.122.200:export1 nfsshare
# ls nfsshare
clientdatafile1
# umount nfsshare
```

4. NFSv3 で NFS 共有をマウントできることを確認します。マウントしたら、テストファイル **clientdatafile1** が表示されていることを確認します。NFSv4 とは異なり、NFSv3 は仮想ファイルシステムを使用しないため、特定のエクスポートをマウントする必要があります。テスト後に共有をアンマウントします。

```
# mkdir nfsshare
# mount -o "vers=3" 192.168.122.200:/nfsshare/exports/export2 nfsshare
# ls nfsshare
clientdatafile2
# umount nfsshare
```

51.5.2. フェイルオーバーのテスト

1. クラスター外のノードで、NFS 共有をマウントし、NFS 共有の設定 で作成した **clientdatafile1** ファイルへのアクセスを確認します。

```
# mkdir nfsshare
# mount -o "vers=4" 192.168.122.200:export1 nfsshare
# ls nfsshare
clientdatafile1
```

2. クラスター内で、**nfsgroup** を実行しているノードを確認します。この例では、**nfsgroup** が **z1.example.com** で実行しています。

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z1.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export1        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export2        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z1.example.com
     nfs-notify (ocf::heartbeat:nfsnotify):     Started z1.example.com
...
```

3. クラスター内のノードから、**nfsgroup** を実行しているノードをスタンバイモードにします。

```
[root@z1 ~]# pcs node standby z1.example.com
```

4. **nfsgroup** が、別のクラスターノードで正常に起動することを確認します。

```
[root@z1 ~]# pcs status
...
Full list of resources:
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z2.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z2.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z2.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs-export1        (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs-export2        (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z2.example.com
     nfs-notify (ocf::heartbeat:nfsnotify):     Started z2.example.com
...
```

5. NFS 共有をマウントしたクラスターの外部のノードから、この外部ノードが NFS マウント内のテストファイルに引き続きアクセスできることを確認します。

```
# ls nfsshare
clientdatafile1
```

フェイルオーバー時に、クライアントに対するサービスが一時的に失われますが、クライアントはユーザーが介入しなくても回復します。デフォルトでは、NFSv4 を使用するクライアントの場合は、マウントの復旧に最大 90 秒かかることがあります。この 90 秒は、システムの起動時にサーバーが監視する NFSv4 ファイルのリースの猶予期間です。NFSv3 クライアントでは、数秒でマウントへのアクセスが回復します。

6. クラスター内のノードから、最初に **nfsgroup** を実行していたノードをスタンバイモードから削除します。

注記

ノードを **スタンバイ** モードから削除しても、リソースはそのノードにフェイルオーバーしません。これは、リソースの **resource-stickiness** の値により異なります。**resource-stickiness** メタ属性の詳細は、現在のノードを優先するようにリソースを設定 を参照してください。

```
[root@z1 ~]# pcs node unstandby z1.example.com
```

第52章 クラスター内の GFS2 ファイルシステム

Red Hat 高可用性クラスターで GFS2 ファイルシステムを設定するには、次の管理手順を使用します。

52.1. クラスターに GFS2 ファイルシステムを設定

次の手順で、GFS2 ファイルシステムを含む Pacemaker クラスターをセットアップできます。この例では、2 ノードクラスター内の 3 つの論理ボリューム上に 3 つの GFS2 ファイルシステムを作成します。

前提条件

- 両方のクラスターノードにクラスターソフトウェアをインストールして起動し、基本的な 2 ノードクラスターを作成している。
- クラスターのフェンシングを設定している。

Pacemaker クラスターの作成とクラスターのフェンシングの設定については、Pacemaker を使用した Red Hat High Availability クラスターの作成 を参照してください。

手順

1. クラスター内の両方のノードで、システムアーキテクチャーに対応する Resilient Storage のリポジトリーを有効にします。たとえば、x86_64 システムの Resilient Storage リポジトリーを有効にするには、以下の **subscription-manager** コマンドを入力します。

   ```
   # subscription-manager repos --enable=rhel-8-for-x86_64-resilientstorage-rpms
   ```

   Resilient Storage リポジトリーは、High Availability リポジトリーのスーパーセットであることに注意してください。Resilient Storage リポジトリーを有効にする場合は、High Availability リポジトリーを有効にする必要はありません。

2. クラスターの両方のノードで、**lvm2-lockd** パッケージ、**gfs2-utils** パッケージ、および **dlm** パッケージをインストールします。AppStream チャンネルおよび Resilient Storage チャンネルにサブスクライブして、これらのパッケージをサポートする必要があります。

   ```
   # yum install lvm2-lockd gfs2-utils dlm
   ```

3. クラスターの両方のノードで、**/etc/lvm/lvm.conf** ファイルの **use_lvmlockd** 設定オプションを **use_lvmlockd=1** に設定します。

   ```
   ...
   use_lvmlockd = 1
   ...
   ```

4. グローバル Pacemaker パラメーター **no-quorum-policy** を **freeze** に設定します。

注記

デフォルトでは、**no-quorum-policy** の値は **stop** に設定され、定足数が失われると、残りのパーティションのリソースがすべて即座に停止されます。通常、このデフォルト設定は最も安全なオプションで最適なおプションですが、ほとんどのリソースとは異なり、GFS2 が機能するにはクォーラムが必要です。クォーラムが失われると、GFS2 マウントを使用したアプリケーション、GFS2 マウント自体の両方が正しく停止できません。クォーラムなしでこれらのリソースを停止しようとすると失敗し、最終的にクォーラムが失われるたびにクラスター全体がフェンスされます。

この状況に対処するには、GFS2 の使用時の **no-quorum-policy** を **freeze** に設定します。この設定では、クォーラムが失われると、クォーラムが回復するまで残りのパーティションは何もしません。

```
[root@z1 ~]# pcs property set no-quorum-policy=freeze
```

5. **dlm** リソースをセットアップします。これは、クラスター内で GFS2 ファイルシステムを設定するために必要な依存関係です。この例では、**dlm** リソースを作成し、リソースグループ **locking** に追加します。

```
[root@z1 ~]# pcs resource create dlm --group locking ocf:pacemaker:controld op
monitor interval=30s on-fail=fence
```

6. リソースグループがクラスターの両方のノードでアクティブになるように、**locking** リソースグループのクローンを作成します。

```
[root@z1 ~]# pcs resource clone locking interleave=true
```

7. **locking** リソースグループの一部として **lvmlockd** リソースを設定します。

```
[root@z1 ~]# pcs resource create lvmlockd --group locking ocf:heartbeat:lvmlockd op
monitor interval=30s on-fail=fence
```

8. クラスターのステータスを確認し、クラスターの両方のノードで **locking** リソースグループが起動していることを確認します。

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Online: [ z1.example.com (1) z2.example.com (2) ]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Started: [ z1.example.com z2.example.com ]
```

9. クラスターの 1 つのノードで、2 つの共有ボリュームグループを作成します。一方のボリュームグループには GFS2 ファイルシステムが 2 つ含まれ、もう一方のボリュームグループには GFS2 ファイルシステムが 1 つ含まれます。

注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターによって使用されるリモート物理ボリュームの起動順序の設定については、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

以下のコマンドは、共有ボリュームグループ **shared_vg1** を **/dev/vdb** に作成します。

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/vdb
  Physical volume "/dev/vdb" successfully created.
  Volume group "shared_vg1" successfully created
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

以下のコマンドは、共有ボリュームグループ **shared_vg2** を **/dev/vdc** に作成します。

```
[root@z1 ~]# vgcreate --shared shared_vg2 /dev/vdc
  Physical volume "/dev/vdc" successfully created.
  Volume group "shared_vg2" successfully created
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

10. クラスター内の 2 番目のノードで以下を実行します。

    a. RHEL 8.5 以降でサポートされる LVM デバイスファイルを使用している場合は、共有デバイスをデバイスファイルに追加します。

    ```
    [root@z2 ~]# lvmdevices --adddev /dev/vdb
    [root@z2 ~]# lvmdevices --adddev /dev/vdc
    ```

    b. 共有ボリュームグループごとにロックマネージャーを起動します。

    ```
    [root@z2 ~]# vgchange --lockstart shared_vg1
      VG shared_vg1 starting dlm lockspace
      Starting locking.  Waiting until locks are ready...
    [root@z2 ~]# vgchange --lockstart shared_vg2
      VG shared_vg2 starting dlm lockspace
      Starting locking.  Waiting until locks are ready...
    ```

11. クラスター内の 1 つのノードで、共有論理ボリュームを作成し、ボリュームを GFS2 ファイルシステムでフォーマットします。ファイルシステムをマウントするノードごとに、ジャーナルが 1 つ必要になります。クラスター内の各ノードに十分なジャーナルを作成してください。ロックテーブル名の形式は、**ClusterName:FSName** です。**ClusterName** は、GFS2 ファイルシステムが作成されているクラスターの名前です。**FSname** はファイルシステム名です。これは、クラスター経由のすべての **lock_dlm** ファイルシステムで一意である必要があります。

```
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg1
```

```
  Logical volume "shared_lv1" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv2 shared_vg1
  Logical volume "shared_lv2" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg2
  Logical volume "shared_lv1" created.

[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo1
/dev/shared_vg1/shared_lv1
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo2
/dev/shared_vg1/shared_lv2
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo3
/dev/shared_vg2/shared_lv1
```

12. すべてのノードで論理ボリュームを自動的にアクティブにするために、各論理ボリュームに **LVM が有効** なリソースを作成します。

    a. ボリュームグループ **shared_vg1** の論理ボリューム **shared_lv1** に、**LVM が有効** なリソース **sharedlv1** を作成します。このコマンドは、リソースを含むリソースグループ **shared_vg1** も作成します。この例のリソースグループの名前は、論理ボリュームを含む共有ボリュームグループと同じになります。

    ```
    [root@z1 ~]# pcs resource create sharedlv1 --group shared_vg1 ocf:heartbeat:LVM-
    activate lvname=shared_lv1 vgname=shared_vg1 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

    b. ボリュームグループ **shared_vg1** の論理ボリューム **shared_lv2** に、**LVM が有効** なリソース **sharedlv2** を作成します。このリソースは、リソースグループ **shared_vg1** に含まれます。

    ```
    [root@z1 ~]# pcs resource create sharedlv2 --group shared_vg1 ocf:heartbeat:LVM-
    activate lvname=shared_lv2 vgname=shared_vg1 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

    c. ボリュームグループ **shared_vg2** の論理ボリューム **shared_lv1** に、**LVM が有効** なリソース **sharedlv3** を作成します。このコマンドは、リソースを含むリソースグループ **shared_vg2** も作成します。

    ```
    [root@z1 ~]# pcs resource create sharedlv3 --group shared_vg2 ocf:heartbeat:LVM-
    activate lvname=shared_lv1 vgname=shared_vg2 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

13. リソースグループのクローンを新たに 2 つ作成します。

```
[root@z1 ~]# pcs resource clone shared_vg1 interleave=true
[root@z1 ~]# pcs resource clone shared_vg2 interleave=true
```

14. **dlm** リソースおよび **lvmlockd** リソースを含む **locking** リソースグループが最初に起動するように、順序の制約を設定します。

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg1-clone
Adding locking-clone shared_vg1-clone (kind: Mandatory) (Options: first-action=start then-
action=start)
```

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg2-clone
Adding locking-clone shared_vg2-clone (kind: Mandatory) (Options: first-action=start then-action=start)
```

15. コロケーション制約を設定して、**vg1** および **vg2** のリソースグループが **locking** リソースグループと同じノードで起動するようにします。

```
[root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
[root@z1 ~]# pcs constraint colocation add shared_vg2-clone with locking-clone
```

16. クラスターの両ノードで、論理ボリュームがアクティブであることを確認します。数秒の遅延が生じる可能性があります。

```
[root@z1 ~]# lvs
  LV         VG         Attr       LSize
  shared_lv1 shared_vg1 -wi-a----- 5.00g
  shared_lv2 shared_vg1 -wi-a----- 5.00g
  shared_lv1 shared_vg2 -wi-a----- 5.00g

[root@z2 ~]# lvs
  LV         VG         Attr       LSize
  shared_lv1 shared_vg1 -wi-a----- 5.00g
  shared_lv2 shared_vg1 -wi-a----- 5.00g
  shared_lv1 shared_vg2 -wi-a----- 5.00g
```

17. ファイルシステムリソースを作成し、各 GFS2 ファイルシステムをすべてのノードに自動的にマウントします。
このファイルシステムは Pacemaker のクラスターリソースとして管理されるため、**/etc/fstab** ファイルには追加しないでください。マウントオプションは、**options=options** を使用してリソース設定の一部として指定できます。すべての設定オプションを確認する場合は、**pcs resource describe Filesystem** コマンドを実行します。

    以下のコマンドは、ファイルシステムのリソースを作成します。これらのコマンドは各リソースを、そのファイルシステムの論理ボリュームを含むリソースグループに追加します。

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1 ocf:heartbeat:Filesystem device="/dev/shared_vg1/shared_lv1" directory="/mnt/gfs1" fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
[root@z1 ~]# pcs resource create sharedfs2 --group shared_vg1 ocf:heartbeat:Filesystem device="/dev/shared_vg1/shared_lv2" directory="/mnt/gfs2" fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
[root@z1 ~]# pcs resource create sharedfs3 --group shared_vg2 ocf:heartbeat:Filesystem device="/dev/shared_vg2/shared_lv1" directory="/mnt/gfs3" fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
```

検証手順

1. GFS2 ファイルシステムが、クラスターの両方のノードにマウントされていることを確認します。

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)
```

```
[root@z2 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)
```

2. クラスターのステータスを確認します。

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
 Clone Set: shared_vg1-clone [shared_vg1]
     Resource Group: shared_vg1:0
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedlv2      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
         sharedfs2      (ocf::heartbeat:Filesystem):    Started z2.example.com
     Resource Group: shared_vg1:1
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedlv2      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z1.example.com
         sharedfs2      (ocf::heartbeat:Filesystem):    Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
 Clone Set: shared_vg2-clone [shared_vg2]
     Resource Group: shared_vg2:0
         sharedlv3      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedfs3      (ocf::heartbeat:Filesystem):    Started z2.example.com
     Resource Group: shared_vg2:1
         sharedlv3      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         sharedfs3      (ocf::heartbeat:Filesystem):    Started z1.example.com
     Started: [ z1.example.com z2.example.com ]

...
```

関連情報

- GFS2 ファイルシステムの設定
- Microsoft Azure での Red Hat High Availability クラスターの設定
- AWS での Red Hat High Availability クラスターの設定
- Google Cloud Platform での Red Hat High Availability クラスターの設定

- Configuring Shared Block Storage for a Red Hat High Availability Cluster on Alibaba Cloud

52.2. クラスターでの暗号化 GFS2 ファイルシステムの設定

(RHEL 8.4 以降) 次の手順で、LUKS で暗号化した GFS2 ファイルシステムを含む Pacemaker クラスターを作成できます。この例では、論理ボリュームに 1 つの GFS2 ファイルシステムを作成し、そのファイルシステムを暗号化します。暗号化された GFS2 ファイルシステムは、LUKS 暗号化に対応する **crypt** リソースエージェントを使用してサポートされます。

この手順は、以下の 3 つの部分で設定されます。

- Pacemaker クラスター内で共有論理ボリュームを設定する
- 論理ボリュームを暗号化して **crypt** リソースを作成する
- GFS2 ファイルシステムで暗号化された論理ボリュームをフォーマットしてクラスター用のファイルシステムリソースを作成する

52.2.1. Pacemaker クラスター内での共有論理ボリュームの設定

前提条件

- 2 つのクラスターノードにクラスターソフトウェアをインストールして起動し、基本的な 2 ノードクラスターを作成している。
- クラスターのフェンシングを設定している。

Pacemaker クラスターの作成とクラスターのフェンシングの設定については、Pacemaker を使用した Red Hat High Availability クラスターの作成 を参照してください。

手順

1. クラスター内の両方のノードで、システムアーキテクチャーに対応する Resilient Storage のリポジトリーを有効にします。たとえば、x86_64 システムの Resilient Storage リポジトリーを有効にするには、以下の **subscription-manager** コマンドを入力します。

   ```
   # subscription-manager repos --enable=rhel-8-for-x86_64-resilientstorage-rpms
   ```

   Resilient Storage リポジトリーは、High Availability リポジトリーのスーパーセットであることに注意してください。Resilient Storage リポジトリーを有効にする場合は、High Availability リポジトリーを有効にする必要はありません。

2. クラスターの両方のノードで、**lvm2-lockd** パッケージ、**gfs2-utils** パッケージ、および **dlm** パッケージをインストールします。AppStream チャンネルおよび Resilient Storage チャンネルにサブスクライブして、これらのパッケージをサポートする必要があります。

   ```
   # yum install lvm2-lockd gfs2-utils dlm
   ```

3. クラスターの両方のノードで、**/etc/lvm/lvm.conf** ファイルの **use_lvmlockd** 設定オプションを **use_lvmlockd=1** に設定します。

   ```
   ...
   use_lvmlockd = 1
   ...
   ```

4. グローバル Pacemaker パラメーター **no-quorum-policy** を **freeze** に設定します。

注記

デフォルトでは、**no-quorum-policy** の値は **stop** に設定され、定足数が失われると、残りのパーティションのリソースがすべて即座に停止されます。通常、このデフォルト設定は最も安全なオプションで最適なおプションですが、ほとんどのリソースとは異なり、GFS2 が機能するにはクォーラムが必要です。クォーラムが失われると、GFS2 マウントを使用したアプリケーション、GFS2 マウント自体の両方が正しく停止できません。クォーラムなしでこれらのリソースを停止しようとすると失敗し、最終的にクォーラムが失われるたびにクラスター全体がフェンスされます。

この状況に対処するには、GFS2 の使用時の **no-quorum-policy** を **freeze** に設定します。この設定では、クォーラムが失われると、クォーラムが回復するまで残りのパーティションは何もしません。

```
[root@z1 ~]# pcs property set no-quorum-policy=freeze
```

5. **dlm** リソースをセットアップします。これは、クラスター内で GFS2 ファイルシステムを設定するために必要な依存関係です。この例では、**dlm** リソースを作成し、リソースグループ **locking** に追加します。

```
[root@z1 ~]# pcs resource create dlm --group locking ocf:pacemaker:controld op
monitor interval=30s on-fail=fence
```

6. リソースグループがクラスターの両方のノードでアクティブになるように、**locking** リソースグループのクローンを作成します。

```
[root@z1 ~]# pcs resource clone locking interleave=true
```

7. **lvmlockd** リソースを、**locking** グループに追加します。

```
[root@z1 ~]# pcs resource create lvmlockd --group locking ocf:heartbeat:lvmlockd op
monitor interval=30s on-fail=fence
```

8. クラスターのステータスを確認し、クラスターの両方のノードで **locking** リソースグループが起動していることを確認します。

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Online: [ z1.example.com (1) z2.example.com (2) ]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Resource Group: locking:1
```

```
     dlm    (ocf::pacemaker:controld):      Started z2.example.com
     lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
 Started: [ z1.example.com z2.example.com ]
```

9. クラスターの 1 つのノードで、共有ボリュームグループを作成します。

注記

LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Red Hat は、Pacemaker が起動する前にサービスが開始されるように設定することを推奨します。Pacemaker クラスターが使用するリモート物理ボリュームの起動順序を設定する方法は、Pacemaker で管理されないリソース依存関係の起動順序の設定 を参照してください。

以下のコマンドは、共有ボリュームグループ **shared_vg1** を **/dev/sda1** に作成します。

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/sda1
  Physical volume "/dev/sda1" successfully created.
  Volume group "shared_vg1" successfully created
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

10. クラスター内の 2 番目のノードで以下を実行します。

    a. RHEL 8.5 以降でサポートされる LVM デバイスファイルを使用している場合は、共有デバイスをデバイスファイルに追加します。

    ```
    [root@z2 ~]# lvmdevices --adddev /dev/sda1
    ```

    b. 共有ボリュームグループのロックマネージャーを起動します。

    ```
    [root@z2 ~]# vgchange --lockstart shared_vg1
      VG shared_vg1 starting dlm lockspace
      Starting locking.  Waiting until locks are ready...
    ```

11. クラスター内の 1 つのノードで、共有論理ボリュームを作成します。

```
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg1
  Logical volume "shared_lv1" created.
```

12. すべてのノードで論理ボリュームを自動的にアクティブにするために、論理ボリュームに **LVM が有効** なリソースを作成します。
以下のコマンドは、ボリュームグループ **shared_vg1** の論理グループ **shared_lv1** に、名前が **sharedlv1** で、**LVM が有効な** リソースを作成します。このコマンドは、リソースを含むリソースグループ **shared_vg1** も作成します。この例のリソースグループの名前は、論理ボリュームを含む共有ボリュームグループと同じになります。

```
[root@z1 ~]# pcs resource create sharedlv1 --group shared_vg1 ocf:heartbeat:LVM-activate lvname=shared_lv1 vgname=shared_vg1 activation_mode=shared vg_access_mode=lvmlockd
```

13. 新しいリソースグループのクローンを作成します。

```
[root@z1 ~]# pcs resource clone shared_vg1 interleave=true
```

14. **dlm** および **lvmlockd** リソースを含む **locking** リソースグループが最初に起動するように、順序の制約を設定します。

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg1-clone
Adding locking-clone shared_vg1-clone (kind: Mandatory) (Options: first-action=start then-action=start)
```

15. コロケーション制約を設定して、**vg1** および **vg2** のリソースグループが **locking** リソースグループと同じノードで起動するようにします。

```
[root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
```

検証手順

クラスターの両ノードで、論理ボリュームがアクティブであることを確認します。数秒の遅延が生じる可能性があります。

```
[root@z1 ~]# lvs
  LV         VG         Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g

[root@z2 ~]# lvs
  LV         VG         Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
```

52.2.2. 論理ボリュームの暗号化および暗号化リソースの作成

前提条件

- Pacemaker クラスターに共有論理ボリュームを設定している。

手順

1. クラスター内の 1 つのノードで、crypt キーを含めて新しいファイルを作成し、ファイルにパーミッションを設定して root でのみ読み取りできるようにします。

```
[root@z1 ~]# touch /etc/crypt_keyfile
[root@z1 ~]# chmod 600 /etc/crypt_keyfile
```

2. crypt キーを作成します。

```
[root@z1 ~]# dd if=/dev/urandom bs=4K count=1 of=/etc/crypt_keyfile
1+0 records in
1+0 records out
4096 bytes (4.1 kB, 4.0 KiB) copied, 0.000306202 s, 13.4 MB/s
[root@z1 ~]# scp /etc/crypt_keyfile root@z2.example.com:/etc/
```

3. **-p** パラメーターを使用して設定したパーミッションを保持した状態で、cript キーファイルをクラスター内の他のノードに配布します。

```
[root@z1 ~]# scp -p /etc/crypt_keyfile root@z2.example.com:/etc/
```

4. LVM ボリュームに暗号化デバイスを作成して、暗号化された GFS2 ファイルシステムを設定します。

```
[root@z1 ~]# cryptsetup luksFormat /dev/shared_vg1/shared_lv1 --type luks2 --key-file=/etc/crypt_keyfile
WARNING!
========
This will overwrite data on /dev/shared_vg1/shared_lv1 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
```

5. **shared_vg1** ボリュームグループの一部として crypt リソースを作成します。

```
[root@z1 ~]# pcs resource create crypt --group shared_vg1 ocf:heartbeat:crypt crypt_dev="luks_lv1" crypt_type=luks2 key_file=/etc/crypt_keyfile encrypted_dev="/dev/shared_vg1/shared_lv1"
```

検証手順

crypt リソースが crypt デバイスを作成していることを確認します。この例では crypt デバイスは **/dev/mapper/luks_lv1** です。

```
[root@z1 ~]# ls -l /dev/mapper/
...
lrwxrwxrwx 1 root root 7 Mar 4 09:52 luks_lv1 -> ../dm-3
...
```

52.2.3. GFS2 ファイルシステムで暗号化された論理ボリュームをフォーマットしてクラスター用のファイルシステムリソースを作成します。

前提条件

- 論理ボリュームを暗号化し、crypt リソースを作成している。

手順

1. クラスター内の 1 つのノードで、GFS2 ファイルシステムを使用してボリュームをフォーマットします。ファイルシステムをマウントするノードごとに、ジャーナルが 1 つ必要になります。クラスター内の各ノードに十分なジャーナルを作成してください。ロックテーブル名の形式は、**ClusterName:FSName** です。**ClusterName** は、GFS2 ファイルシステムが作成されているクラスターの名前です。**FSname** はファイルシステム名です。これは、クラスター経由のすべての **lock_dlm** ファイルシステムで一意である必要があります。

```
[root@z1 ~]# mkfs.gfs2 -j3 -p lock_dlm -t my_cluster:gfs2-demo1 /dev/mapper/luks_lv1
/dev/mapper/luks_lv1 is a symbolic link to /dev/dm-3
This will destroy any data on /dev/dm-3
Are you sure you want to proceed? [y/n] y
Discarding device contents (may take a while on large devices): Done
Adding journals: Done
Building resource groups: Done
Creating quota file: Done
```

```
Writing superblock and syncing: Done
Device:                    /dev/mapper/luks_lv1
Block size:                4096
Device size:               4.98 GB (1306624 blocks)
Filesystem size:           4.98 GB (1306622 blocks)
Journals:                  3
Journal size:              16MB
Resource groups:           23
Locking protocol:          "lock_dlm"
Lock table:                "my_cluster:gfs2-demo1"
UUID:                      de263f7b-0f12-4d02-bbb2-56642fade293
```

2. ファイルシステムリソースを作成し、GFS2 ファイルシステムをすべてのノードに自動的にマウントします。
ファイルシステムは Pacemaker のクラスターリソースとして管理されるため、**/etc/fstab** ファイルには追加しないでください。マウントオプションは、**options=options** を使用してリソース設定の一部として指定できます。すべての設定オプションを確認する場合は、**pcs resource describe Filesystem** コマンドを実行します。

   以下のコマンドは、ファイルシステムのリソースを作成します。このコマンドは、対象のファイルシステムの論理ボリュームリソースを含むリソースグループに、リソースを追加します。

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1
ocf:heartbeat:Filesystem device="/dev/mapper/luks_lv1" directory="/mnt/gfs1"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
```

検証手順

1. GFS2 ファイルシステムが、クラスターの両方のノードにマウントされていることを確認します。

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)

[root@z2 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
```

2. クラスターのステータスを確認します。

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
    Started: [ z1.example.com z2.example.com ]
```

```
  Clone Set: shared_vg1-clone [shared_vg1]
    Resource Group: shared_vg1:0
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         crypt       (ocf::heartbeat:crypt) Started z2.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
    Resource Group: shared_vg1:1
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z1.example.com
         crypt      (ocf::heartbeat:crypt)  Started z1.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z1.example.com
       Started:  [z1.example.com z2.example.com ]
...
```

関連情報

- GFS2 ファイルシステムの設定

52.3. RHEL7 から RHEL8 へ GFS2 ファイルシステムの移行

GFS2 ファイルシステムを含む RHEL 8 クラスターを設定する場合は、既存の Red Hat Enterprise 7 論理ボリュームを使用できます。

Red Hat Enterprise Linux 8 では、LVM は、**clvmd** の代わりに LVM ロックデーモン **lvmlockd** を使用して、アクティブ/アクティブのクラスターで共有ストレージデバイスを管理します。これにより、アクティブ/アクティブのクラスターに共有論理ボリュームとして使用する必要がある論理ボリュームを設定する必要があります。また、これにより、**LVM が有効** なリソースを使用して LVM ボリュームを管理し、**lvmlockd** リソースエージェントを使用して **lvmlockd** デーモンを管理する必要があります。共有論理ボリュームを使用して、GFS2 ファイルシステムを含む Pacemaker クラスターを設定する手順は、クラスターに GFS2 ファイルシステムを設定 を参照してください。

GFS2 ファイルシステムを含む RHEL8 クラスターを設定する際に、既存の Red Hat Enterprise Linux 7 論理ボリュームを使用する場合は、RHEL8 クラスターから以下の手順を実行します。この例では、クラスター化された RHEL 7 論理ボリュームが、ボリュームグループ **upgrade_gfs_vg** に含まれます。

注記

既存のファイルシステムを有効にするために、RHEL8 クラスターの名前は、GFS2 ファイルシステムに含まれる RHEL7 クラスターと同じになります。

手順

1. GFS2 ファイルシステムを含む論理ボリュームが現在非アクティブであることを確認してください。すべてのノードがボリュームグループを使用して停止した場合にのみ、この手順は安全です。

2. クラスター内の 1 つのノードから、強制的にボリュームグループをローカルに変更します。

```
[root@rhel8-01 ~]# vgchange --lock-type none --lock-opt force upgrade_gfs_vg
Forcibly change VG lock type to none? [y/n]: y
  Volume group "upgrade_gfs_vg" successfully changed
```

3. クラスター内の 1 つのノードから、ローカルボリュームグループを共有ボリュームグループに変更します。

```
[root@rhel8-01 ~]# vgchange --lock-type dlm upgrade_gfs_vg
  Volume group "upgrade_gfs_vg" successfully changed
```

4. クラスター内の各ノードで、ボリュームグループのロックを開始します。

```
[root@rhel8-01 ~]# vgchange --lockstart upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@rhel8-02 ~]# vgchange --lockstart upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

この手順を実行すると、各論理ボリュームに、**LVM が有効** なリソースを作成できます。

第53章 RED HAT HIGH AVAILABILITY クラスターでのフェンシングの設定

応答しないノードがデータへのアクセスを続けている可能性があります。データが安全であることを確認する場合は、STONITH を使用してノードをフェンシングすることが唯一の方法になります。STONITH は Shoot The Other Node In The Head の頭字語で、不安定なノードや同時アクセスによるデータの破損を防ぐことができます。STONITH を使用すると、別のノードからデータをアクセスする前に、そのノードが完全にオフラインであることを確認できます。

STONITH はクラスター化したサービスを停止できない場合にも役に立ちます。この場合は、クラスターが STONITH を使用してノード全体を強制的にオフラインにし、その後サービスを別の場所で開始すると安全です。

フェンシングの概要と、Red Hat High Availability クラスターにおけるフェンシングの重要性は Fencing in a Red Hat High Availability Cluster を参照してください。

クラスターのノードにフェンスデバイスを設定して、Pacemaker クラスターに STONITH を実装します。

53.1. 利用可能なフェンスエージェントと、そのオプションの表示

以下のコマンドは、利用可能なフェンシングエージェントと、特定のフェンスエージェントで利用可能なオプションを表示できます。

このコマンドは、利用可能な全フェンシングエージェントをリスト表示します。フィルターを指定すると、フィルターに一致するフェンシングエージェントのみが表示されます。

```
pcs stonith list [filter]
```

このコマンドにより、指定したフェンスエージェントのオプションが表示されます。

```
pcs stonith describe [stonith_agent]
```

次のコマンドでは Telnet または SSH 経由の APC 用フェンスエージェントのオプションを表示します。

```
# pcs stonith describe fence_apc
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  passwd_script: Script to retrieve password
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number or name of virtual machine
  identity_file: Identity file for ssh
  switch: Physical switch number on device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  ipport: TCP port to use for connection with device
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
```

```
help: Display help and exit
separator: Separator for CSV created by operation list
power_timeout: Test X seconds for status change after ON/OFF
shell_timeout: Wait X seconds for cmd prompt after issuing command
login_timeout: Wait X seconds for cmd prompt after login
power_wait: Wait X seconds after issuing ON/OFF
delay: Wait X seconds before fencing is started
retry_on: Count of attempts to retry power on
```

警告

method オプションを提供するフェンスエージェントでは **cycle** がサポートされず、データの破損が生じる可能性があるため、この値は指定できません。

53.2. フェンスデバイスの作成

フェンスデバイスを作成するコマンドの形式は、以下のとおりです。利用可能なフェンスデバイス作成オプションのリストは、**pcs stonith -h** の出力を参照してください。

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options] [op
operation_action operation_options]
```

以下のコマンドは、1つのノードに対して、1つのフェンスデバイスを作成します。

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor interval=30s
```

1つのノードのみをフェンスできるフェンスデバイスや、複数のノードをフェンスできるデバイスもあります。フェンスデバイスの作成時に指定するパラメーターは、フェンスデバイスが対応しているか、必要としているかにより異なります。

- フェンスデバイスの中には、フェンスできるノードを自動的に判断できるものがあります。
- フェンスデバイスの作成時に **pcmk_host_list** パラメーターを使用すると、フェンスデバイスで制御されるすべてのマシンを指定できます。
- フェンスデバイスによっては、フェンスデバイスが理解する仕様へのホスト名のマッピングが必要となるものがあります。フェンスデバイスの作成時に、**pcmk_host_map** パラメーターを使用して、ホスト名をマッピングできます。

pcmk_host_list パラメーターおよび **pcmk_host_map** パラメーターの詳細は、フェンスデバイスの一般的なプロパティー を参照してください。

フェンスデバイスを設定したら、デバイスをテストして正しく機能していることを確認してください。フェンスデバイスをテストする方法は、フェンスデバイスのテスト を参照してください。

53.3. フェンスデバイスの一般的なプロパティー

フェンスデバイスにも設定可能な一般的なプロパティーや、フェンスの動作を決定するさまざまなクラスタープロパティーがあります。

クラスターノードは、フェンスリソースが開始しているかどうかに関わらず、フェンスデバイスでその他のクラスターノードをフェンスできます。以下の例外を除き、リソースが開始しているかどうかは、デバイスの定期的なモニターのみを制御するものとなり、使用可能かどうかは制御しません。

- フェンスデバイスは、**pcs stonith disable stonith_id** コマンドを実行して無効にできます。これにより、ノードがそのデバイスを使用できないように設定できます。
- 特定のノードがフェンスデバイスを使用できないようにするには、**pcs constraint location … avoids** コマンドで、フェンスリソースの場所制約を設定できます。
- **stonith-enabled=false** を設定すると、フェンシングがすべて無効になります。ただし、実稼働環境でフェンシングを無効にすることは適していないため、フェンシングが無効になっている場合は、Red Hat ではクラスターがサポートされないことに注意してください。

以下の表は、フェンスデバイスに設定できる一般的なプロパティーを説明します。

表53.1 フェンスデバイスの一般的なプロパティー

フィールド	タイプ	デフォルト	説明
pcmk_host_map	文字列		ホスト名を、ホスト名に対応していないデバイスのポート番号へマッピングします。たとえば、**node1:1;node2:2,3** の場合は、node1 にはポート 1 を使用し、node2 にはポート 2 と 3 を使用するようにクラスターに指示します。RHEL 8.7 以降、**pcmk_host_map** プロパティーは、値の前にバックスラッシュを使用して **pcmk_host_map** 値内の特殊文字をサポートします。たとえば、**pcmk_host_map="node3:plug\ 1"** を指定して、ホストエイリアスにスペースを含めることができます。
pcmk_host_list	文字列		このデバイスで制御するマシンのリストです (**pcmk_host_check=static-list** 以外は任意)。

フィールド	タイプ	デフォルト	説明
pcmk_host_check	文字列	* **pcmk_host_list** または **pcmk_host_map** が設定されている場合は **static-list** * それが設定されておらず、フェンスデバイスが **list** アクションに対応する場合は **dynamic-list** になります。 * それ以外で、フェンスデバイスが **status** アクションに対応している場合は **status** になります。 * それ以外は、**none** になります。	デバイスで制御するマシンを指定します。使用できる値は、**dynamic-list** (デバイスへの問い合わせ)、**static-list** (**pcmk_host_list** 属性の確認)、なし (すべてのデバイスで全マシンのフェンスが可能と見なされる) です。

以下の表では、フェンスデバイスに設定できるその他のプロパティーをまとめています。これらのオプションは高度な設定を行う場合にのみ使用されます。

表53.2 フェンスデバイスの高度なプロパティー

フィールド	タイプ	デフォルト	説明
pcmk_host_argument	文字列	port	port の代替パラメーターです。デバイスによっては、標準の port パラメーターに対応していない場合や、そのデバイス固有のパラメーターも提供している場合があります。このパラメーターを使用して、デバイス固有の代替パラメーターを指定します。これは、フェンシングするマシンを示します。クラスターが追加パラメーターを提供しないようにする場合は、**none** 値を使用します。
pcmk_reboot_action	文字列	reboot	**reboot** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このパラメーターを使用して、再起動を実行するデバイス固有の代替コマンドを指定します。

フィールド	タイプ	デフォルト	説明
pcmk_reboot_timeout	時間	60s	**stonith-timeout** の代替コマンドで、再起動にタイムアウトを指定します。再起動が完了するまでに通常より長い時間を要するデバイスもあれば、通常より短い時間で完了するデバイスもあります。このパラメーターを使用して、再起動にデバイス固有のタイムアウトを指定します。
pcmk_reboot_retries	整数	2	タイムアウト期間内に、**reboot** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による再起動の動作の再試行回数を変更する場合に使用します。
pcmk_off_action	文字列	off	**off** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、オフ操作を実行するデバイス固有のコマンドを指定します。
pcmk_off_timeout	時間	60s	**stonith-timeout** の代替コマンドで、オフ操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、オフ操作にデバイス固有のタイムアウトを指定します。
pcmk_off_retries	整数	2	タイムアウト期間内に、off コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker によるオフ動作の再試行回数を変更する場合に使用します。
pcmk_list_action	文字列	list	**list** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、list 操作を実行するデバイス固有のコマンドを指定します。

フィールド	タイプ	デフォルト	説明
pcmk_list_timeout	時間	60s	list 操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、list 操作にデバイス固有のタイムアウトを指定します。
pcmk_list_retries	整数	2	タイムアウト期間内に、**list** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による list 動作の再試行回数を変更する場合に使用します。
pcmk_monitor_action	文字列	monitor	**monitor** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、監視操作を実行するデバイス固有のコマンドを指定します。
pcmk_monitor_timeout	時間	60s	**stonith-timeout** の代替コマンドで、監視にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、監視操作にデバイス固有のタイムアウトを指定します。
pcmk_monitor_retries	整数	2	タイムアウト期間内に、**monitor** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による監視操作の再試行回数を変更する場合に使用します。
pcmk_status_action	文字列	status	**status** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、status 操作を実行するデバイス固有のコマンドを指定します。

フィールド	タイプ	デフォルト	説明
pcmk_status_timeout	時間	60s	**stonith-timeout** の代替コマンドで、status 操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、status 操作にデバイス固有のタイムアウトを指定します。
pcmk_status_retries	整数	2	タイムアウト期間内に、status コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による status 動作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明
pcmk_delay_base	文字列	0s	stonith 操作のベース遅延を有効にし、ベース遅延の値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、ランダムな遅延値に静的遅延を加算します。**pcmk_delay_base** を設定し、**pcmk_delay_max** を設定しない場合は、遅延にランダムなコンポーネントはなく、**pcmk_delay_base** の値となります。 Red Hat Enterprise Linux 8.6 では、**pcmk_delay_base** パラメーターを使用して、ノードごとに異なる値を指定できます。これにより、ノードごとに異なる遅延を使用して、単一のフェンスデバイスを 2 ノードクラスターで使用できます。これは、各ノードが同時に他のノードをフェンスしようとする状況を防ぐのに役立ちます。ノードごとに異なる値を指定するには、**pcmk_host_map** と同様の構文を使用して、ホスト名をそのノードの遅延値にマップします。たとえば、**node1:0;node2:10s** は、**node1** をフェンシングするときに遅延を使用せず、**node2** をフェンシングするときに 10 秒の遅延を使用します。 各フェンスエージェントには delay パラメーターが実装されています。これは、**pcmk_delay_*** プロパティーで設定された遅延の影響は受けません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。

フィールド	タイプ	デフォルト	説明
pcmk_delay_max	時間	0s	stonith 動作のランダムな遅延を有効にし、ランダムな遅延の最大値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、このランダムな遅延値に静的遅延を加算します。**pcmk_delay_max** を設定し、**pcmk_delay_base** を設定しない場合は、静的なコンポーネントが遅延に含まれません。 各フェンスエージェントには delay パラメーターが実装されています。これは、**pcmk_delay_*** プロパティーで設定された遅延の影響は受けません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。
pcmk_action_limit	整数	1	このデバイスで並行して実行できる操作の上限です。最初に、クラスタープロパティーの **concurrent-fencing=true** を設定する必要があります (これは、RHEL 8.1 以降のデフォルト値です)。値を -1 にすると無制限になります。
pcmk_on_action	文字列	on	高度な使用のみ**on** の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、**on** 操作を実行するデバイス固有のコマンドを指定します。
pcmk_on_timeout	時間	60s	高度な使用のみ**stonith-timeout** の代替コマンドで、**on** 操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、**on** 操作にデバイス固有のタイムアウトを指定します。

フィールド	タイプ	デフォルト	説明
pcmk_on_retries	整数	2	高度な使用のみタイムアウト期間内に、**on** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が **失敗** する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による **on** 動作の再試行回数を変更する場合に使用します。

個々のフェンスデバイスに設定できるプロパティーのほかにも、以下の表で説明しているように、フェンス動作を判断するクラスタープロパティーも設定できます。

表53.3 フェンスの動作を決定するクラスタープロパティー

オプション	デフォルト	説明
stonith-enabled	true	障害が発生したノードと、停止できないリソースが含まれるノードをフェンスする必要があることを示します。データを保護するには、**true** に設定する必要があります。 **true** または未設定の場合は、STONITH リソースが設定されていない限り、クラスターによりリソースの起動が拒否されます。 Red Hat は、この値が **true** に設定されたクラスターのみをサポートします。
stonith-action	reboot	STONITH デバイスに送るアクション。使用できる値は **reboot**、**off** です。**poweroff** 値も使用できますが、レガシーデバイスでのみ使用されます。
stonith-timeout	60s	STONITH アクションが完了するのを待つ時間。
stonith-max-attempts	10	クラスターがすぐに再起動できなくなるまで、ターゲットでフェンシングが失敗する回数。
stonith-watchdog-timeout		ノードがハードウェアウォッチドッグによって強制終了すまで待機する最大時間。この値は、ハードウェアウォッチドッグのタイムアウト値の倍に設定することが推奨されます。このオプションは、ウォッチドッグのみの SBD 設定がフェンシングに使用される場合にのみ必要です。

オプション	デフォルト	説明
concurrent-fencing	true (RHEL 8.1 以降)	フェンシング操作を並行して実行できるようにします。
fence-reaction	stop	(Red Hat Enterprise Linux 8.2 以降) 独自のフェンシングの通知を受信した場合は、クラスターノードがどのように反応するかを決定します。クラスターノードは、フェンシングの設定が間違っている場合に独自のフェンシングの通知を受信するか、ファブリックフェンシングがクラスター通信を遮断しない状態である可能性があります。許可される値は、Pacemaker をすぐに停止し、停止したままにする **stop** と、ローカルノードを直ちに再起動して失敗した場合に停止する **panic** です。 このプロパティーのデフォルト値は **stop** ですが、この値に最も安全な選択肢は **panic** であり、ローカルノードを直ちに再起動しようとします。停止動作を希望する場合は、おそらくファブリックフェンシングと併用する場合は、明示的に指定することが推奨されます。

クラスターのプロパティーの設定は、クラスターのプロパティーの設定と削除 を参照してください。

53.4. フェンスデバイスのテスト

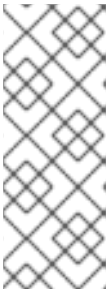
フェンシングは、Red Hat Cluster インフラストラクチャーの基本的な部分を設定しているため、フェンシングが適切に機能していることを確認またはテストすることは重要です。

手順

以下の手順で、フェンスデバイスをテストします。

1. デバイスへの接続に使用する ssh、telnet、HTTP などのリモートプロトコルを使用して、手動でログインしてフェンスデバイスをテストしたり、出力される内容を確認します。たとえば、IPMI 対応デバイスのフェンシングを設定する場合は、**ipmitool** を使用してリモートでのログインを試行します。手動でログインする際に使用するオプションに注意してください。これらのオプションは、フェンスエージェントを使用する際に必要になる場合があります。
   フェンスデバイスにログインできない場合は、そのデバイスが ping 可能であること、ファイアウォール設定がフェンスデバイスへのアクセスを妨げていないこと、フェンスデバイスでリモートアクセスが有効になっていること、認証情報が正しいことなどを確認します。

2. フェンスエージェントスクリプトを使用して、フェンスエージェントを手動で実行します。フェンスエージェントを実行するのに、クラスターサービスが実行している必要はないため、デバイスをクラスターに設定する前にこのステップを完了できます。これにより、先に進む前に、フェンスデバイスが適切に応答することを確認できます。



注記

これらの例では、iLO デバイスの **fence_ipmilan** フェンスエージェントスクリプトを使用します。実際に使用するフェンスエージェントと、そのエージェントを呼び出すコマンドは、お使いのサーバーハードウェアによって異なります。指定するオプションを確認するには、フェンスエージェントの man ページを参照してください。通常は、フェンスデバイスのログイン、パスワードなどの情報と、その他のフェンスデバイスに関する情報を把握しておく必要があります。

以下の例は、**-o status** パラメーターを指定して **fence_ipmilan** フェンスエージェントスクリプトを実行する場合に使用する形式になります。このコマンドを実行すると、フェンシングは実行せずに、別のノードのフェンスデバイスインターフェイスのステータスを確認します。ノードの再起動を試行する前にデバイスをテストして、動作させることができます。このコマンドを実行する際に、iLO デバイスの電源をオン/オフにするパーミッションを持つ iLO ユーザーの名前およびパスワードを指定します。

```
# fence_ipmilan -a ipaddress -l username -p password -o status
```

以下の例は、**-o reboot** パラメーターを指定して **fence_ipmilan** フェンスエージェントスクリプトを実行するのに使用する形式になります。このコマンドを 1 つのノードで実行すると、この iLO デバイスで管理するノードが再起動します。

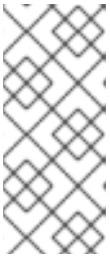
```
# fence_ipmilan -a ipaddress -l username -p password -o reboot
```

フェンスエージェントがステータス、オフ、オン、または再起動の動作を適切に実行しない場合は、ハードウェア、フェンスデバイスの設定、およびコマンドの構文を確認する必要があります。さらに、デバッグ出力を有効にした状態で、フェンスエージェントスクリプトを実行できます。デバッグ出力は、一部のフェンスエージェントで、フェンスデバイスにログインする際に、フェンスエージェントスクリプトに問題が発生しているイベントシーケンスの場所を確認するのに役に立ちます。

```
# fence_ipmilan -a ipaddress -l username -p password -o status -D /tmp/$(hostname)-fence_agent.debug
```

発生した障害を診断する際に、フェンスデバイスに手動でログインする際に指定したオプションが、フェンスエージェントスクリプトでフェンスエージェントに渡した内容と同一であることを確認する必要があります。

フェンスエージェントが、暗号化した接続に対応する場合は、証明書の検証で障害が生じているためにエラーが出力される場合があり、ホストを信頼することや、フェンスエージェントの **ssl-insecure** パラメーターを使用することが求められます。同様に、ターゲットデバイスで SSL/TLS を無効にした場合は、フェンスエージェントに SSL パラメーターを設定する際に、これを考慮しないといけない場合があります。



注記

テストしているフェンスエージェントが **fence_drac** または **fence_ilo** の場合、もしくはその他の、継続して失敗しているシステム管理デバイスのフェンスエージェントの場合は、フォールバックして **fence_ipmilan** を試行します。大半のシステム管理カードは IPMI リモートログインに対応しており、フェンスエージェントとしては **fence_ipmilan** だけに対応しています。

3. フェンスデバイスを、手動で機能したオプションと同じオプションでクラスターに設定し、クラスターを起動したら、以下の例にあるように、任意のノードから **pcs stonith fence** コマン

ドを実行してフェンシングをテストします (または複数のノードから複数回実行します)。**pcs stonith fence** コマンドは m クラスター設定を CIB から読み取り、フェンス動作を実行するように設定したフェンスエージェントを呼び出します。これにより、クラスター設定が正確であることが確認できます。

```
# pcs stonith fence node_name
```

pcs stonith fence コマンドに成功した場合は、フェンスイベントの発生時に、クラスターのフェンシング設定が機能します。このコマンドが失敗すると、クラスター管理が取得した設定でフェンスデバイスを起動することができません。以下の問題を確認し、必要に応じてクラスター設定を更新します。

- フェンス設定を確認します。たとえば、ホストマップを使用したことがある場合は、指定したホスト名を使用して、システムがノードを見つけられるようにする必要があります。
- デバイスのパスワードおよびユーザー名に、bash シェルが誤って解釈する可能性がある特殊文字が含まれるかどうかを確認します。パスワードとユーザー名を引用符で囲んで入力すると、この問題に対処できます。
- **pcs stonith** コマンドで IP アドレスまたはホスト名を使用してデバイスに接続できるかどうかを確認してください。たとえば、stonith コマンドでホスト名を指定し、IP アドレスを使用して行ったテストは有効ではありません。
- フェンスデバイスが使用するプロトコルにアクセスできる場合は、そのプロトコルを使用してデバイスへの接続を試行します。たとえば、多くのエージェントが ssh または telnet を使用します。デバイスへの接続は、デバイスの設定時に指定した認証情報を使用して試行する必要があります。これにより、有効なプロンプトを取得し、そのデバイスにログインできるかどうかを確認できます。
  すべてのパラメーターが適切であることが確認できたものの、フェンスデバイスには接続できない時に、フェンスデバイスでログ機能が使用できる場合は、ログを確認できます。これにより、ユーザーが接続したかどうかと、ユーザーが実行したコマンドが表示されます。**/var/log/messages** ファイルで stonith やエラーを確認すれば、発生している問題のヒントが得られる可能性もあります。また、エージェントによっては、より詳細な情報が得られる場合があります。

4. フェンスデバイステストに成功し、クラスターが稼働したら、実際の障害をテストします。このテストでは、クラスターで、トークンの損失を生じさせる動作を実行します。
   - ネットワークを停止します。ネットワークの利用方法は、設定により異なります。ただし、多くの場合は、ネットワークケーブルまたは電源ケーブルをホストから物理的に抜くことができます。ネットワーク障害をシミュレートする方法は What is the proper way to simulate a network failure on a RHEL Cluster? を参照してください。

     注記

     ネットワークや電源ケーブルを物理的に切断せずに、ローカルホストのネットワークインターフェイスを無効にすることは、フェンシングのテストとしては推奨されません。実際に発生する障害を正確にシミュレートしていないためです。

   - ローカルのファイアウォールを使用して、corosync の受信トラフィックおよび送信トラフィックをブロックします。
     以下の例では corosync をブロックします。ここでは、デフォルトの corosync ポートと、ローカルのファイアウォールとして **firewalld** が使用されていることと、corosync が使用するネットワークインターフェイスがデフォルトのファイアウォールゾーンにあることが

前提となっています。

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 2 -p udp --dport=5405 -j DROP
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="5405" protocol="udp"
drop
```

- **sysrq-trigger** でクラッシュをシミュレートし、マシンをクラッシュします。ただし、カーネルパニックを発生させると、データが損失する可能性があることに注意してください。クラッシュする前に、クラスターリソースを無効にすることが推奨されます。

```
# echo c > /proc/sysrq-trigger
```

53.5. フェンスレベルの設定

Pacemaker は、フェンストポロジーと呼ばれる機能を用いて、複数デバイスでのノードのフェンシングに対応します。トポロジーを実装するには、通常の方法で各デバイスを作成し、設定のフェンストポロジーセクションでフェンスレベルを 1 つ以上定義します。

Pacemaker は、以下のようにフェンシングレベルを処理します。

- レベルは、1 から昇順で試行されていきます。
- デバイスに障害が発生すると、現在のレベルの処理が終了します。同レベルのデバイスには試行されず、次のレベルが試行されます。
- すべてのデバイスのフェンシングが正常に完了すると、そのレベルが継承され、他のレベルは試行されなくなります。
- いずれかのレベルで成功するか、すべてのレベルが試行され失敗すると、操作は終了します。

ノードにフェンスレベルを追加する場合は、次のコマンドを使用します。デバイスは、stonith ID をコンマで区切って指定します。stonith ID が、指定したレベルで試行されます。

```
pcs stonith level add level node devices
```

次のコマンドを使用すると現在設定されている全フェンスレベルが表示されます。

```
pcs stonith level
```

以下の例では、ノード **rh7-2** に、2 つのフェンスデバイス (il0 フェンスデバイス **my_ilo** と、apc フェンスデバイス **my_apc**) が設定されています。このコマンドはフェンスレベルを設定し、デバイス **my_ilo** に障害が発生し、ノードがフェンスできない場合に、Pacemaker がデバイス **my_apc** の使用を試行できるようにします。この例では、レベル設定後の **pcs stonith level** コマンドの出力も表示されています。

```
# pcs stonith level add 1 rh7-2 my_ilo
# pcs stonith level add 2 rh7-2 my_apc
# pcs stonith level
 Node: rh7-2
  Level 1 - my_ilo
  Level 2 - my_apc
```

次のコマンドは、指定したノードおよびデバイスのフェンスレベルを削除します。ノードやデバイスを指定しないと、指定したフェンスレベルがすべてのノードから削除されます。

```
pcs stonith level remove level  [node_id] [stonith_id] ... [stonith_id]
```

以下のコマンドを使用すると、指定したノードや stonith id のフェンスレベルが削除されます。ノードや stonith id を指定しないと、すべてのフェンスレベルが削除されます。

```
pcs stonith level clear [node]|stonith_id(s)]
```

複数の stonith ID を指定する場合はコンマで区切って指定します。空白は入力しないでください。以下に例を示します。

```
# pcs stonith level clear dev_a,dev_b
```

次のコマンドは、フェンスレベルで指定されたフェンスデバイスとノードがすべて存在することを確認します。

```
pcs stonith level verify
```

フェンストポロジーのノードは、ノード名に適用する正規表現と、ノードの属性 (およびその値) で指定できます。たとえば、次のコマンドでは、ノード **node1**、**node2**、および **node3** がフェンスデバイス **apc1** および **apc2** を使用するように設定し、ノード **node4**、**node5**、および **node6** がフェンスデバイス **apc3** および **apc4** を使用するように設定します。

```
# pcs stonith level add 1 "regexp%node[1-3]" apc1,apc2
# pcs stonith level add 1 "regexp%node[4-6]" apc3,apc4
```

次のコマンドでは、ノード属性のマッチングを使用して、同じように設定します。

```
# pcs node attribute node1 rack=1
# pcs node attribute node2 rack=1
# pcs node attribute node3 rack=1
# pcs node attribute node4 rack=2
# pcs node attribute node5 rack=2
# pcs node attribute node6 rack=2
# pcs stonith level add 1 attrib%rack=1 apc1,apc2
# pcs stonith level add 1 attrib%rack=2 apc3,apc4
```

53.6. 冗長電源のフェンシング設定

冗長電源にフェンシングを設定する場合は、ホストを再起動するときに、クラスターが、最初に両方の電源をオフにしてから、いずれかの電源をオンにするようにする必要があります。

ノードの電源が完全にオフにならないと、ノードがリソースを解放しない場合があります。このとき、解放できなかったリソースに複数のノードが同時にアクセスして、リソースが破損する可能性があります。

以下の例にあるように、各デバイスを一度だけ定義し、両方のデバイスがノードのフェンスに必要であると指定する必要があります。

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"
```

```
# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

53.7. 設定済みのフェンスデバイスの表示

以下のコマンドは、現在設定されているフェンスデバイスをすべて表示します。**stonith_id** を指定すると、設定されているその stonith デバイスのオプションだけが表示されます。**--full** オプションが指定されていると、設定された stonith のオプションがすべて表示されます。

```
pcs stonith config [stonith_id] [--full]
```

53.8. PCS コマンドとしてのフェンスデバイスのエクスポート

Red Hat Enterprise Linux 8.7 では、**pcs stonith config** コマンドの **--output-format=cmd** オプションを使用して、別のシステムに設定済みのフェンスデバイスを再作成するのに使用できる **pcs** コマンドを表示できます。

次のコマンドは、**fence_apc_snmp** フェンスデバイスを作成し、デバイスを再作成するために使用できる **pcs** コマンドを表示します。

```
# pcs stonith create myapc fence_apc_snmp ip="zapc.example.com"
pcmk_host_map="z1.example.com:1;z2.example.com:2" username="apc" password="apc"
# pcs stonith config --output-format=cmd
Warning: Only 'text' output format is supported for stonith levels
pcs stonith create --no-default-ops --force -- myapc fence_apc_snmp \
  ip=zapc.example.com password=apc 'pcmk_host_map=z1.example.com:1;z2.example.com:2'
username=apc \
  op \
    monitor interval=60s id=myapc-monitor-interval-60s
```

53.9. フェンスデバイスの修正と削除

次のコマンドを使用して、現在設定されているフェンシングデバイスのオプションを変更または追加します。

```
pcs stonith update stonith_id [stonith_device_options]
```

pcs stonith update コマンドを使用して SCSI フェンスデバイスを更新すると、stonith リソースが実行されているのと同じノードで実行中の全リソースを再起動することになります。RHEL 8.5 では、以下のコマンドのいずれかのバージョンを使用して、他のクラスターリソースを再起動しなくても SCSI デバイスを更新できます。RHEL 8.7 では、SCSI フェンスデバイスをマルチパスデバイスとして設定できます。

```
pcs stonith update-scsi-devices stonith_id set device-path1 device-path2
pcs stonith update-scsi-devices stonith_id add device-path1 remove device-path2
```

現在の設定からフェンスデバイスを削除する場合は次のコマンドを使用します。

```
pcs stonith delete stonith_id
```

53.10. 手動によるクラスターノードのフェンシング

次のコマンドで、ノードを手動でフェンスできます。**--off** を指定すると、stonith に API コールの **off** を使用し、ノードをオフにします (再起動はしません)。

```
pcs stonith fence node [--off]
```

ノードがアクティブでない場合でも、フェンスデバイスがそのノードをフェンスできない状況では、ノードのリソースをクラスターが復旧できない可能性があります。この場合は、ノードの電源が切れたことを手動で確認した後、次のコマンドを入力して、ノードの電源が切れたことをクラスターに確認し、そのリソースを回復のために解放できます。

警告

指定したノードが実際にオフになっていない状態で、クラスターソフトウェア、または通常クラスターが制御するサービスを実行すると、データ破損またはクラスター障害が発生します。

```
pcs stonith confirm node
```

53.11. フェンスデバイスの無効化

フェンスデバイス/リソースを無効にする場合は、**pcs stonith disable** コマンドを実行します。

以下のコマンドは、フェンスデバイス **myapc** を無効にします。

```
# pcs stonith disable myapc
```

53.12. ノードがフェンスデバイスを使用しないように設定する手順

特定のノードがフェンスデバイスを使用できないようにするには、フェンスリソースの場所の制約を設定します。

以下の例では、フェンスデバイスの **node1-ipmi** が、**node1** で実行されないようにします。

```
# pcs constraint location node1-ipmi avoids node1
```

53.13. 統合フェンスデバイスで使用する ACPI の設定

クラスターが統合フェンスデバイスを使用する場合は、即時かつ完全なフェンシングを実行できるように、ACPI (Advanced Configuration and Power Interface) を設定する必要があります。

クラスターノードが統合フェンスデバイスでフェンシングされるように設定されている場合は、そのノードの ACPI Soft-Off を無効にします。ACPI Soft-Off を無効にすることにより、統合フェンスデバイスは、クリーンシャットダウンを試行する代わりに、ノードを即時に、かつ完全にオフにできます (例: **shutdown -h now**)。それ以外の場合は、ACPI Soft-Off が有効になっていると、統合フェンスデバイスがノードをオフにするのに 4 秒以上かかることがあります (以下の注記部分を参照してください)。

さらに、ACPI Soft-Off が有効になっていて、ノードがシャットダウン時にパニック状態になるか、フリーズすると、統合フェンスデバイスがノードをオフにできない場合があります。このような状況では、フェンシングが遅延するか、失敗します。したがって、ノードが統合フェンスデバイスでフェンシングされ、ACPI Soft-Off が有効になっている場合は、クラスターが徐々に復元します。または管理者の介入による復旧が必要になります。

注記

ノードのフェンシングにかかる時間は、使用している統合フェンスデバイスによって異なります。統合フェンスデバイスの中には、電源ボタンを押し続けるのと同じ動作を実行するものもあります。この場合は、ノードがオフになるのに 4 秒から 5 秒かかります。また、電源ボタンを押してすぐ離すのと同等の動作を行い、ノードの電源をオフにする行為をオペレーティングシステムに依存する統合フェンスデバイスもあります。この場合は、ノードがオフになるのにかかる時間は 4~5 秒よりも長くなります。

- ACPI Soft-Off を無効にする場合は、BIOS 設定を instant-off、またはこれに類似する設定に変更することが推奨されます。これにより、BIOS で ACPI Soft-Off を無効化で説明しているように、ノードは遅延なくオフになります。

システムによっては、BIOS で ACPI Soft-Off を無効にできません。お使いのクラスターでは、BIOS で ACPI Soft-Off を無効にできない場合に、以下のいずれかの方法で ACPI Soft-Off を無効にできます。

- **/etc/systemd/logind.conf** ファイルに **HandlePowerKey=ignore** を設定し、以下のように logind.conf ファイルで ACPI Soft-Off の無効化に記載されているように、ノードがフェンシングされるとすぐにオフになることを確認します。これが、ACPI Soft-Off を無効にする 1 つ目の代替方法です。

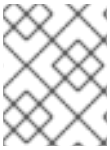
- GRUB 2 ファイルを使用した ACPI の完全な無効化で説明されているように、カーネル起動コマンドラインに **acpi=off** を追加します。これは、ACPI Soft-Off を無効にする 2 つ目の代替方法です。この方法の使用が推奨される場合、または 1 つ目の代替方法が利用できない場合に使用してください。

重要

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全が無効になってるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に **限り**、この方法を使用してください。

53.13.1. BIOS で ACPI Soft-Off を無効化

以下の手順で、各クラスターノードの BIOS を設定して、ACPI Soft-Off を無効にできます。



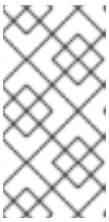
注記

BIOS で ACPI Soft-Off を無効にする手順は、サーバーシステムにより異なる場合があります。この手順は、お使いのハードウェアのドキュメントで確認する必要があります。

手順

1. ノードを再起動して **BIOS CMOS Setup Utility** プログラムを起動します。
2. 電源メニュー (または同等の電源管理メニュー) に移動します。
3. 電源メニューで、**Soft-Off by PWR-BTTN** 機能 (または同等) を **Instant-Off** (または、遅延なく

電源ボタンでノードをオフにする同等の設定) に設定します。**BIOS CMOS 設定ユーティリティー** は、**ACPI Function** が **Enabled** に設定され、**Soft-Off by PWR-BTTN** が **Instant-Off** に設定されていることを示しています。



注記

ACPI Function、**Soft-Off by PWR-BTTN**、および **Instant-Off** に相当するものは、コンピューターによって異なります。ただし、この手順の目的は、電源ボタンを使用して遅延なしにコンピューターをオフにするように BIOS を設定することです。

4. **BIOS CMOS Setup Utility** プログラムを終了します。BIOS 設定が保存されます。

5. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスをテストする方法は フェンスデバイスのテスト を参照してください。

BIOS CMOS Setup Utility

```
`Soft-Off by PWR-BTTN` set to
`Instant-Off`
```

```
+---------------------------------------------|-------------------+
|   ACPI Function             [Enabled]       |   Item Help       |
|   ACPI Suspend Type         [S1(POS)]       |-------------------|
| x Run VGABIOS if S3 Resume   Auto           |   Menu Level   *  |
|   Suspend Mode              [Disabled]      |                   |
|   HDD Power Down            [Disabled]      |                   |
|   Soft-Off by PWR-BTTN      [Instant-Off    |                   |
|   CPU THRM-Throttling       [50.0%]         |                   |
|   Wake-Up by PCI card       [Enabled]       |                   |
|   Power On by Ring          [Enabled]       |                   |
|   Wake Up On LAN            [Enabled]       |                   |
| x USB KB Wake-Up From S3     Disabled       |                   |
|   Resume by Alarm           [Disabled]      |                   |
| x  Date(of Month) Alarm       0             |                   |
| x  Time(hh:mm:ss) Alarm       0 :  0 :      |                   |
|   POWER ON Function         [BUTTON ONLY    |                   |
| x KB Power ON Password       Enter          |                   |
| x Hot Key Power ON           Ctrl-F1        |                   |
|                                             |                   |
|                                             |                   |
+---------------------------------------------|-------------------+
```

この例では、**ACPI Function** が **Enabled** に設定され、**Soft-Off by PWR-BTTN** が **Instant-Off** に設定されていることを示しています。

53.13.2. logind.conf ファイルで ACPI Soft-Off の無効化

/etc/systemd/logind.conf ファイルで電源キーの処理を無効にする場合は、以下の手順を行います。

手順

1. **/etc/systemd/logind.conf** ファイルに、以下の設定を定義します。

```
HandlePowerKey=ignore
```

2. **systemd-logind** サービスを再起動します。

```
# systemctl restart systemd-logind.service
```

3. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスをテストする方法は フェンスデバイスのテスト を参照してください。

53.13.3. GRUB 2 ファイルでの ACPI の完全な無効化

ACPI Soft-Off は、カーネルの GRUB メニューエントリーに **acpi=off** を追加して無効にできます。

重要

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全が無効になってるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に **限り**、この方法を使用してください。

手順

以下の手順で、GRUB 2 ファイルで ACPI を無効にします。

1. 以下のように、**grubby** ツールで、**--args** オプションと **--update-kernel** オプションを使用して、各クラスターノードの **grub.cfg** ファイルを変更します。

```
# grubby --args=acpi=off --update-kernel=ALL
```

2. ノードを再起動します。

3. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスをテストする方法は フェンスデバイスのテスト を参照してください。

第54章 クラスターリソースの設定

次のコマンドを使用して、クラスターリソースを作成および削除します。

クラスターリソースを作成するコマンドの形式は、以下のとおりです。

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op
operation_action operation_options [operation_action operation options]...] [meta
meta_options...] [clone [clone_options] | master [master_options] [--wait[=n]]
```

主なクラスターリソースの作成オプションには、以下が含まれます。

- **--before** および **--after** オプションは、リソースグループに含まれるリソースを基準にして、追加するリソースの位置を指定します。
- **--disabled** オプションは、リソースが自動的に起動しないことを示しています。

クラスター内に作成できるリソースの数に制限はありません。

リソースの制約を設定して、クラスター内のそのリソースの動作を指定できます。

リソース作成の例

以下のコマンドは、仕様 **ocf**、プロバイダー **heartbeat**、およびタイプ **IPaddr2** で、リソース **VirtualIP** を作成します。このリソースのフローティングアドレスは 192.168.0.120 であり、システムは、30 秒間隔で、リソースが実行しているかどうかを確認します。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 op
monitor interval=30s
```

または、以下のように、standard フィールドおよび provider フィールドを省略できます。規格とプロバイダーはそれぞれ **ocf** と **heartbeat** にデフォルト設定されます。

```
# pcs resource create VirtualIP IPaddr2 ip=192.168.0.120 cidr_netmask=24 op monitor
interval=30s
```

設定済みリソースの削除

次のコマンドを使用して、設定済みのリソースを削除します。

```
pcs resource delete resource_id
```

たとえば、次のコマンドは、リソース ID が **VirtualIP** の既存リソースを削除します。

```
# pcs resource delete VirtualIP
```

54.1. リソースエージェント識別子

リソースに定義する識別子は、リソースに使用するエージェント、そのエージェントを検索する場所、およびそれが準拠する仕様をクラスターに指示します。

以下の表は、リソースエージェントのこれらのプロパティーについて説明しています。

表54.1 リソースエージェント識別子

フィールド	説明
standard	エージェントが準拠する規格。使用できる値とその意味は以下のとおりです。 * **ocf** - 指定した **type** は、Open Cluster Framework Resource Agent API に準拠した実行可能ファイルの名前で、**/usr/lib/ocf/resource.d/provider** の下に置かれます。 * **lsb** - 指定した **type** は、Linux Standard Base Init Script Actions に準拠する実行ファイルの名前です。type で完全パスを指定しないと、システムは **/etc/init.d** ディレクトリーを探します。 * **systemd** - 指定した **type** は、インストール済み **systemd** ユニットの名前です。 * **service** - Pacemaker は、指定した **type** を選択します (まずは **lsb** エージェントとして、次に **systemd** エージェントとして)。 * **nagios** - 指定した **type** は、Nagios Plugin API に準拠する実行可能ファイルの名前で、**/usr/libexec/nagios/plugins** ディレクトリーに置かれます。OCF スタイルのメタデータは、**/usr/share/nagios/plugins-metadata** ディレクトリーに置かれます (一般的なプラグインは **nagios-agents-metadata** パッケージで入手可能)。
type	使用するリソースエージェントの名前 (例: **IPaddr** または **Filesystem**)
provider	OCF 仕様により、複数のベンダーで同じリソースエージェントを指定できます。Red Hat が提供するエージェントのほとんどは、プロバイダーに **heartbeat** を使用しています。

以下の表には、利用可能なリソースプロパティーを表示するコマンドをまとめています。

表54.2 リソースプロパティーを表示させるコマンド

pcs 表示コマンド	出力
pcs resource list	利用できる全リソースのリストを表示
pcs resource standards	利用できるリソースエージェントのリストを表示
pcs resource providers	利用できるリソースエージェントプロバイダーのリストを表示
pcs resource list string	利用できるリソースを指定文字列でフィルターしたリストを表示。仕様名、プロバイダー名、タイプ名などでフィルターを指定して、リソースを表示できます。

54.2. リソース固有のパラメーターの表示

各リソースで以下のコマンドを使用すると、リソースの説明、そのリソースに設定できるパラメーター、およびそのリソースに設定されるデフォルト値が表示されます。

```
pcs resource describe [standard:[provider:]]type
```

たとえば、以下のコマンドは、**apache** タイプのリソース情報を表示します。

```
# pcs resource describe ocf:heartbeat:apache
This is the resource agent for the Apache Web server.
This resource agent operates both version 1.x and version 2.x Apache
servers.

...
```

54.3. リソースのメタオプションの設定

リソースには、リソース固有のパラメーターの他に、リソースオプションを設定できます。このような追加オプションは、クラスターがリソースの動作を決定する際に使用されます。

以下の表は、リソースのメタオプションを示しています。

表54.3 リソースのメタオプション

フィールド	デフォルト	説明
priority	**0**	すべてのリソースをアクティブにできない場合に、クラスターは優先度の低いリソースを停止して、優先度が高いリソースを実行し続けます。
target-role	**Started**	クラスターがこのリソースを維持しようとする状態を示します。設定できる値は以下のとおりです。 * **Stopped** - リソースの強制停止 * **Started** - リソースの起動を許可 (昇格可能なクローンの場合、適切であればマスターロールに昇格される) * **Master** - リソースの起動を許可し、適切であれば昇格を許可 * **Slave** - リソースの起動を許可、ただしリソースが昇格可能な場合、スレーブモードに限る RHEL 8.5 では、Pacemaker 設定でロールが指定される場合、**pcs** コマンドラインインターフェイスが **Promoted** および **Unpromoted** を受け入れます。これらのロール名は、**Master** および **Slave** Pacemaker ロールと機能的に同等です。

フィールド	デフォルト	説明
is-managed	**true**	クラスターによるリソースの起動および停止を許可するかどうかを示します。使用できる値は **true** または **false** です。
resource-stickiness	0	リソースを同じ場所に残すための優先度の値です。この属性の詳細は、現在のノードを優先するようにリソースを設定 を参照してください。
requires	Calculated	リソースを起動できる条件を示します。 以下の条件を除き、**fencing** がデフォルトに設定されます。以下の値が使用できます。 * **nothing** - クラスターは常にリソースを開始できます。 * **quorum** - クラスターは、設定されているノードの過半数がアクティブな場合に限りこのリソースを起動できます。**stonith-enabled** が **false** に設定されている場合、またはリソースの **standard** が **stonith** の場合は、このオプションがデフォルト値となります。 * **fencing** -設定されているノードの過半数がアクティブ、**かつ** 障害が発生しているノードや不明なノードがフェンスになっている場合に限り、クラスターはこのリソースを起動できます。 * **unfencing** - 設定されているノードの過半数がアクティブで、**かつ** 障害が発生しているノードや不明なノードがすべてフェンスされており、**フェンシング** されていないノードに **限り**、クラスターはこのリソースを起動できます。**provides=unfencing stonith** メタオプションがフェンスデバイスに設定されている場合のデフォルト値です。

フィールド	デフォルト	説明
migration-threshold	**INFINITY**	指定したリソースが任意のノードで失敗した回数です。この回数を超えると、そのノードには、このリソースのホストとして不適格とするマークが付けられます。値を 0 にするとこの機能は無効になり、ノードに不適格マークが付けられることはありません。**INFINITY** (デフォルト) に設定すると、クラスターは、これを非常に大きい有限数として扱います。このオプションは、失敗した操作に **on-fail=restart** (デフォルト) が設定されていて、かつ失敗した起動操作のクラスタープロパティー **start-failure-is-fatal** が **false** に設定されている場合に限り有効です。
failure-timeout	**0** (無効)	**migration-threshold** オプションと併用されます。障害が発生していないかのように動作し、障害が発生したノードにリソースを戻せるようになるまで待機する秒数を示します。
multiple-active	**stop_start**	リソースが複数のノードでアクティブであることが検出された場合に、クラスターが実行すべき動作を示します。設定できる値は以下のとおりです。 * **block** - リソースに unmanaged のマークを付けます。 * **stop_only** - 実行中のインスタンスをすべて停止し、以降の動作は行いません。 * **stop_start** - 実行中のインスタンスをすべて停止してから、リソースを 1 カ所でのみ起動します。 * **stop_unexpected** - (RHEL 8.7 以降) リソースの予期しないインスタンスのみを停止し、完全な再起動は必要ありません。ユーザーは、サービスとそのリソースエージェントが、完全に再起動しなくても追加のアクティブインスタンスで機能することを確認する必要があります。

フィールド	デフォルト	説明
critical	**true**	(RHEL 8.4 以降) リソースがリソースグループに含まれる場合に作成された暗黙的なコロケーションの成約など、従属リソース (target_resource) としてリソース関連のコロケーション成約すべてに、**influence** オプションのデフォルト値を設定します。**influence** コロケーション制約オプションは、従属リソースが移行のしきい値に達して失敗した場合に、クラスターで別のノードにプライマリーリソースと従属リソースを移行するか、クラスターでサーバーの切り替えなしに従属リソースをオフラインのままにするかを決定します。**critical** リソースのメタオプションには、**true** または **false** の値を指定できます。デフォルト値は **true** です。
allow-unhealthy-nodes	**false**	(RHEL 8.7 以降) **true** に設定されている場合、ノードの正常性が低下したことでリソースがノードから強制的に切り離されることはありません。正常性リソースにこの属性が設定されている場合、クラスターはノードの正常性が回復したかどうかを自動的に検出し、リソースをノードに戻すことができます。ノードの正常性は、ローカルの状態に基づいて正常性リソースエージェントによって設定された正常性属性と、クラスターがそれらの状態にどのように反応するかを決定する戦略関連のオプションの組み合わせによって決定されます。

54.3.1. リソースオプションのデフォルト値の変更

Red Hat Enterprise Linux 8.3 では、**pcs resource defaults update** コマンドを使用して、全リソースのリソースオプションのデフォルト値を変更できます。たとえば、次のコマンドは、**resource-stickiness** のデフォルト値を 100 にリセットします。

```
# pcs resource defaults update resource-stickiness=100
```

以前のリリースのすべてのリソースのデフォルトを設定する元の **pcs resource defaults name=value** コマンドは、複数のデフォルトが設定されない限りサポートされます。ただし、**pcs resource defaults update** が、コマンドの推奨されるバージョンになりました。

54.3.2. リソースセットのリソースオプションのデフォルト値の変更

Red Hat Enterprise Linux 8.3 では、**pcs resource defaults set create** コマンドを使用して、複数のリソースのデフォルトセットを作成できます。これにより、**resource** 式を含むルールを指定できます。RHEL 8.3 では、このコマンドで指定したルールは、**and**、**or** および括弧を含め、**resource** 式のみを使用できます。RHEL 8.4 では、このコマンドで指定したルールは、 **and**、**or** および括弧など、**resource** と **date** 式のみを使用できます。

pcs resource defaults set create コマンドを使用して、特定タイプの全リソースにデフォルトのリソース値を設定できます。たとえば、停止に時間がかかるデータベースを実行している場合は、データベースタイプの全リソースで **resource-stickiness** のデフォルト値を増やすことで、想定している頻度よりも多く、このようなリソースが他のノードに移動されるのを回避できます。

以下のコマンドは、**pqsql** タイプの全リソースに、**resource-stickiness** のデフォルト値を 100 に設定します。

- リソースのデフォルトセット名を指定する **id** オプションは必須ではありません。このオプションを設定すると、**pcs** が自動的に ID を生成します。この値を設定すると、より分かりやすい名前に設定できます。
- この例では、**::pgsql** は、クラスやプロバイダーは任意でタイプが **pgsql** を指定します。
  - **ocf:heartbeat:pgsql** を指定すると、クラスが **ocf**、プロバイダーが **heartbeat**、タイプが **pgsql** に指定されます。
  - **ocf:pacemaker:** を指定すると、タイプは任意でクラスが **ocf**、プロバイダーが **pacemaker** に指定されます。

```
# pcs resource defaults set create id=pgsql-stickiness meta resource-stickiness=100 rule
resource ::pgsql
```

既存セットのデフォルト値を変更する場合は、**pcs resource defaults set update** コマンドを使用します。

54.3.3. 現在設定されているリソースのデフォルトの表示

pcs resource defaults コマンドは、指定したルールなど、現在設定されているリソースオプションのデフォルト値のリストを表示します。

次の例では **resource-stickiness** のデフォルト値を 100 にリセットした後のコマンド出力を示しています。

```
# pcs resource defaults
Meta Attrs: rsc_defaults-meta_attributes
  resource-stickiness=100
```

以下の例では、タイプが **pqsql** の全リソースの **resource-stickiness** のデフォルト値を 100 にリセットし、**id** オプションを **id=pgsql-stickiness** に設定します。

```
# pcs resource defaults
Meta Attrs: pgsql-stickiness
  resource-stickiness=100
  Rule: boolean-op=and score=INFINITY
    Expression: resource ::pgsql
```

54.3.4. リソース作成でメタオプションの設定

リソースのメタオプションにおけるデフォルト値のリセットの有無に関わらず、リソースを作成する際に、特定リソースのリソースオプションをデフォルト以外の値に設定できます。以下は、リソースのメタオプションの値を指定する際に使用する **pcs resource create** コマンドの形式です。

pcs resource create **resource_id** [**standard**:[**provider**:]]**type** [**resource options**] [meta **meta_options**...]

たとえば、以下のコマンドでは **resource-stickiness** の値を 50 に設定したリソースを作成します。

 # pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 meta resource-stickiness=50

また、次のコマンドを使用すると既存のリソース、グループ、クローン作成したリソースなどのリソースメタオプションの値を作成することもできます。

pcs resource meta **resource_id** | **group_id** | **clone_id meta_options**

以下の例では、**dummy_resource** という名前の既存リソースがあります。このコマンドは、**failure-timeout** メタオプションの値を 20 秒に設定します。これにより 20 秒でリソースが同じノード上で再起動を試行できるようになります。

 # pcs resource meta dummy_resource failure-timeout=20s

上記のコマンドを実行した後、リソースの値を表示して、**failure-timeout=20s** が設定されているかどうかを確認できます。

```
# pcs resource config dummy_resource
 Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
  Meta Attrs: failure-timeout=20s
  ...
```

54.4. リソースグループの設定

クラスターの最も一般的な設定要素の 1 つがリソースセットです。リソースセットはまとめて配置し、順番に起動し、その逆順で停止する必要があります。この設定を簡単にするため、Pacemaker はリソースグループの概念をサポートします。

54.4.1. リソースグループの作成

以下のコマンドを使用してリソースグループを作成し、グループに追加するリソースを指定します。グループが存在しない場合は、このコマンドによりグループが作成されます。グループが存在する場合は、このコマンドにより別のリソースがグループに追加されます。リソースは、このコマンドで指定された順序で起動し、その逆順で停止します。

pcs resource group add **group_name resource_id** [**resource_id**] ... [**resource_id**] [--before **resource_id** | --after **resource_id**]

このコマンドの **--before** オプションおよび **--after** オプションを使用して、追加するリソースの位置を、そのグループにすでに含まれるリソースを基準にして指定できます。

以下のコマンドを使用して、リソースを作成するときに、既存のグループに新しいリソースを追加することもできます。以下のコマンドでは、作成するリソースが **group_name** グループに追加されます。**group_name** グループが存在しない場合は作成されます。

pcs resource create **resource_id** [**standard**:[**provider**:]]**type** [resource_options] [op **operation_action operation_options**] --group **group_name**

グループに含まれるリソースの数に制限はありません。グループの基本的なプロパティーは以下のとおりです。

- グループ内に、複数のリソースが置かれています。
- リソースは、指定した順序で起動します。グループ内に実行できないリソースがあると、そのリソースの後に指定されたリソースは実行できません。
- リソースは、指定した順序と逆の順序で停止します。

以下の例では、既存リソースの **IPaddr** と **Email** が含まれるリソースグループ **shortcut** が作成されます。

```
# pcs resource group add shortcut IPaddr Email
```

この例では、以下のように設定されています。

- **IPaddr** が起動してから、**Email** が起動します。
- **Email** リソースが停止してから、**IPAddr** が停止します。
- **IPaddr** を実行できない場合は、**Email** も実行できません。
- **Email** を実行できなくても、**IPaddr** には影響ありません。

54.4.2. リソースグループの削除

以下のコマンドを使用して、グループからリソースを削除します。グループにリソースが残っていないと、このコマンドによりグループ自体が削除されます。

```
pcs resource group remove group_name resource_id...
```

54.4.3. リソースグループの表示

以下のコマンドは、現在設定されているリソースグループをリスト表示します。

```
pcs resource group list
```

54.4.4. グループオプション

リソースグループには、**priority** オプション、**target-role** オプション、または **is-managed** オプションを設定できます。このオプションは、1 つのリソースに設定されている場合と同じ意味を維持します。リソースのメタオプションの詳細は、リソースのメタオプションの設定 を参照してください。

54.4.5. グループの粘着性

粘着性は、リソースを現在の場所に留ませる優先度の度合いを示し、グループで加算されます。グループのアクティブなリソースが持つ stickness 値の合計が、グループの合計になります。そのため、**resource-stickiness** のデフォルト値が 100 で、グループに 7 つのメンバーがあり、そのメンバーの 5 つがアクティブな場合は、グループ全体でスコアが 500 の現在の場所が優先されます。

54.5. リソース動作の決定

リソースの制約を設定して、クラスター内のそのリソースの動作を指定できます。以下の制約のカテゴリーを設定できます。

- **場所** の制約 - この制約は、リソースを実行するノードを指定します。場所の制約を設定する方法は、リソースを実行するノードの決定 を参照してください。
- **順序** の制約 - この制約は、リソースが実行する順序を決定します。順序の制約を設定する方法は、クラスターリソースの実行順序の決定 を参照してください。
- **コロケーション** の制約 - この制約は、他のリソースとの対比でリソースの配置先を決定します。コロケーションの制約の詳細は、クラスターリソースのコロケーション を参照してください。

複数リソースをまとめて配置して、順番に起動するまたは逆順で停止する一連の制約を簡単に設定する方法として、Pacemaker ではリソースグループという概念に対応しています。リソースグループの作成後に、個別のリソースの制約を設定するようにグループ自体に制約を設定できます。

第55章 リソースを実行するノードの決定

場所の制約は、リソースを実行するノードを指定します。場所の制約を設定することで、たとえば特定のノードで優先してリソースを実行する、または特定のノードではリソースを実行しないことを決定できます。

リソースを実行しているノードは、場所の制約の他に、そのリソースの **resource-stickiness** 値からの影響を受けます。この値で、現在リソースを実行中のノードにどの程度リソースを残す設定にするかが決まります。**resource-stickiness** 値の設定に関する詳細は、現在のノードを優先するようにリソースを設定 を参照してください。

55.1. 場所の制約の設定

基本的な場所の制約を設定し、オプションの **score** 値で制約の相対的な優先度を指定することで、リソースの実行を特定のノードで優先するか、回避するかを指定できます。

以下のコマンドは、リソースの実行を、指定した 1 つまたは複数のノードで優先するように、場所の制約を作成します。1 回のコマンドで、特定のリソースの制約を複数のノードに対して作成できます。

```
pcs constraint location rsc prefers node[=score] [node[=score]] ...
```

次のコマンドは、リソースが指定ノードを回避する場所の制約を作成します。

```
pcs constraint location rsc avoids node[=score] [node[=score]] ...
```

次の表は、場所の制約を設定する基本的なオプションを説明します。

表55.1 場所の制約オプション

フィールド	説明
rsc	リソース名
node	ノード名

フィールド	説明
score	指定のリソースが指定のノードを優先するべきか回避するべきかを示す優先度を示す正の整数値。**INFINITY** は、リソースの場所制約のデフォルト **score** 値です。 **pcs constraint location rsc prefers** コマンドで **score** の値を **INFINITY** にすると、そのノードが利用可能な場合は、リソースがそのノードで優先的に実行します。ただし、そのノードが利用できない場合に、別のノードでそのリソースを実行しないようにする訳ではありません。 **pcs constraint location rsc avoids** コマンドで **score** に **INFINITY** を指定すると、その他のノードが利用できない場合でも、そのリソースはそのノードでは実行されないことを示します。これは、**-INFINITY** のスコアで **pcs constraint location add** コマンドを設定するのと同じです。 数値スコア (**INFINITY** 以外) は、その制約が任意で、それを上回る要因が他にない限り有効となることを意味します。たとえば、リソースが別のノードに置かれ、その **resource-stickiness** スコアが、場所制約のスコアよりも **優先** される場合は、リソースがその場所に残されます。

以下のコマンドは、**Webserver** リソースが、**node1** ノードで優先的に実行するように指定する場所の制約を作成します。

```
# pcs constraint location Webserver prefers node1
```

pcs では、コマンドラインの場所の制約に関する正規表現に対応しています。この制約は、リソース名に一致する正規表現に基づいて、複数のリソースに適用されます。これにより、1 つのコマンドラインで複数の場所の制約を設定できます。

次のコマンドは、**dummy0** から **dummy9** までのリソースの実行が **node1** に優先されるように指定する場所の制約を作成します。

```
# pcs constraint location 'regexp%dummy[0-9]' prefers node1
```

Pacemaker は、http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04 で説明しているように、POSIX 拡張正規表現を使用するため、以下のコマンドを実行しても同じ制約を指定できます。

```
# pcs constraint location 'regexp%dummy[[:digit:]]' prefers node1
```

55.2. ノードのサブセットへのリソース検出を制限

Pacemaker がどこでリソースを開始しても、開始する前にそのリソースがすでに実行しているかどうかを確認するために、すべてのノードでワンタイム監視操作 (プローブとも呼ばれています) を実行します。このリソース検出のプロセスは、監視を実行できないノードではエラーになる場合があります。

ノードに場所の制約を設定する際に、**pcs constraint location** コマンドの **resource-discovery** オプションを指定して、指定したリソースに対して、Pacemaker がこのノードでリソース検出を実行するか

どうかの優先度を指定できます。物理的にリソースが稼働可能なノードのサブセットでリソース検出を制限すると、ノードが大量に存在する場合にパフォーマンスを大幅に改善できます。**pacemaker_remote** を使用して、ノード数を 100 単位で拡大する場合は、このオプションの使用を検討してください。

以下のコマンドは、**pcs constraint location** コマンドで **resource-discovery** オプションを指定する場合の形式を示しています。このコマンドでは、基本的な場所の制約に対応します。score を正の値にすると、リソースが特定のノードで優先的に実行するように設定されます。score を負の値にすると、リソースがノードを回避するように設定されます。基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

```
pcs constraint location add id rsc node score [resource-discovery=option]
```

以下の表は、リソース検出の制約を設定する基本パラメーターを説明します。

表55.2 リソース検出制約パラメーター

フィールド	説明
id	制約自体にユーザーが選択した名前。
rsc	リソース名
node	ノード名
score	指定のリソースが指定のノードを優先するべきか回避するべきかを示す優先度を示す整数値。スコアが正の値の場合は、ノードを優先するようにリソースを設定する基本的な場所の制約となり、負の場合は、ノードを回避するようにリソースを設定する基本的な場所の制約となります。 **score** の値を **INFINITY** に設定すると、そのノードが利用可能な場合は、リソースがそのノードで優先的に実行します。ただし、そのノードが利用できない場合に、別のノードでそのリソースを実行しないようにする訳ではありません。**score** の値を **-INFINITY** に設定すると、他のノードが利用できない場合でも、リソースはそのノードでは実行されません。 数値スコア (**INFINITY** または **-INFINITY** 以外) は、その制約が任意で、それを上回る要因が他にない限り有効となることを意味します。たとえば、リソースが別のノードに置かれ、その **resource-stickiness** スコアが、場所制約のスコアよりも **優先** される場合は、リソースがその場所に残されます。

resource-discovery オプション	* **always** - このノードに指定したリソースで、リソース検出を常に実行します。これは、リソースの場所の制約の **resource-discovery** のデフォルト値です。 * **never** - このノードで、指定したリソースに対してリソース検出を行いません。 * **exclusive** - このノード (および同様に **exclusive** マークが付いているその他のノード) で指定したリソースに対してのみ、リソースの検出を行います。複数のノードで同じリソースの **exclusive** 検出を使用する複数の場所制約により、**resource-discovery** が排他的であるノードのサブセットが作成されます。1つまたは複数のノードで、リソースが **exclusive** 検出用にマーク付けされている場合、そのリソースは、ノードのサブセット内にのみ配置できます。

警告

resource-discovery を **never** または **exclusive** に設定すると、Pacemaker が、想定されていない場所で実行している不要なサービスのインスタンスを検出して停止する機能がなくなります。関連するソフトウェアをアンインストールしたままにするなどして、リソース検出なしでサービスがノードでアクティブにならないようにすることは、システム管理者の責任です。

55.3. 場所の制約方法の設定

場所の制約を使用する場合は、リソースをどのノードで実行できるかを指定する一般的な方法を設定できます。

- オプトインクラスター - デフォルトでは、すべてのリソースを、どのノードでも実行できません。そして、特定のリソースに対してノードを選択的に許可できるようにクラスターを設定します。
- オプトアウトクラスター - デフォルトでは、すべてのリソースをどのノードでも実行できるクラスターを設定してから、リソースを特定のノードで実行しないように、場所の制約を作成します。

クラスターでオプトインまたはオプトアウトのどちらを選択するかは、優先する設定やクラスターの設定により異なります。ほとんどのリソースをほとんどのノードで実行できるようにする場合は、オプトアウトを使用した方が設定しやすくなる可能性があります。ほとんどのリソースを、一部のノードでのみ実行する場合は、オプトインを使用した方が設定しやすくなる可能性があります。

55.3.1. オプトインクラスターの設定

オプトインクラスターを作成する場合は、クラスタープロパティー **symmetric-cluster** を **false** に設定し、デフォルトでは、いずれのノードでもリソースの実行を許可しないようにします。

```
# pcs property set symmetric-cluster=false
```

個々のリソースでノードを有効にします。以下のコマンドは、場所の制約を設定し、**Webserver** リソースでは **example-1** ノードが優先され、**Database** リソースでは **example-2** ノードが優先されるようにし、いずれのリソースも優先ノードに障害が発生した場合は **example-3** ノードにフェイルオーバーできるようにします。オプトインクラスターに場所の制約を設定する場合は、スコアをゼロに設定すると、リソースに対してノードの優先や回避を指定せずに、リソースをノードで実行できます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver prefers example-3=0
# pcs constraint location Database prefers example-2=200
# pcs constraint location Database prefers example-3=0
```

55.3.2. オプトアウトクラスターの設定

オプトアウトクラスターを作成するには、クラスタープロパティー **symmetric-cluster** を **true** に設定し、デフォルトで、すべてのノードでリソースの実行を許可します。これは、**symmetric-cluster** が明示的に設定されていない場合のデフォルト設定です。

```
# pcs property set symmetric-cluster=true
```

以下のコマンドを実行すると、オプトインクラスターの設定の例と同じ設定になります。全ノードのスコアは暗黙で 0 になるため、優先ノードに障害が発生した場合はいずれのリソースも **example-3** ノードにフェイルオーバーできます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

INFINITY は、スコアのデフォルト値であるため、上記コマンドでは、スコアに INFINITY を指定する必要はないことに注意してください。

55.4. 現在のノードを優先するリソースの設定

リソースには、リソースのメタオプションの設定 で説明されているように、リソースの作成時にメタ属性として設定できる **resource-stickiness** 値があります。**resource-stickiness** 値は、現在実行しているノード上にリソースが残す量を決定します。Pacemaker は、他の設定 (場所の制約の score 値など) とともに **resource-stickiness** 値を考慮して、リソースを別のノードに移動するか、そのまま残すかを決定します。

resource-stickiness の値を 0 にすると、クラスターは、必要に応じてリソースを移動して、ノード間でリソースのバランスを調整できます。これにより、関連のないリソースが起動または停止したときにリソースが移動する可能性があります。stickiness が高くなると、リソースは現在の場所に留まり、その他の状況が stickiness を上回る場合に限り移動するようになります。これにより、新しく追加したノードに割り当てられたリソースは、管理者の介入なしには利用できなくなる可能性があります。

デフォルトでは、**resource-stickiness** の値が 0 の状態でリソースが作成されます。**resource-stickiness** が 0 に設定され、場所の制約がない Pacemaker のデフォルト動作では、クラスターノード間で均等に分散されるようにリソースを移動します。この設定では、正常なリソースの移動頻度が想定よりも増える可能性があります。この動作を防ぐには、デフォルトの **resource-stickiness** の値を 1 に設定します。このデフォルトはクラスター内のすべてのリソースに適用されます。この値を小さく指定すると、作成する他の制約で簡単に上書きできますが、Pacemaker がクラスター全体で正常なリソースを不必要に移動するのを防ぐには十分です。

以下のコマンドは、デフォルトの **resource-stickiness** 値を 1 に設定します。

```
# pcs resource defaults update resource-stickiness=1
```

resource-stickiness に正の値が設定されている場合、リソースは新たに追加されたノードに移動しません。この時点でリソースバランスが必要な場合は、**resource-stickiness** の値を一時的に 0 に設定できます。

場所の制約スコアが **resource-stickiness** の値よりも大きい場合には、クラスターは場所の制約が指定するノードに、正常なリソースを依然として移動する可能性があります。

Pacemaker がリソースを配置する場所を決定する方法の詳細は、ノード配置ストラテジーの設定 を参照してください。

第56章 クラスターリソースの実行順序の決定

リソースが実行する順序を指定する、順序の制約を設定できます。

次は、順序の制約を設定するコマンドの形式です。

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

以下の表では、順序の制約を設定する場合のプロパティーとオプションをまとめています。

表56.1 順序の制約のプロパティー

フィールド	説明
resource_id	動作を行うリソースの名前。
action	リソースに対して指示されるアクション。**action** プロパティーでは、以下の値が使用できます。 * **start** - リソースの開始アクションを指示します。 * **stop** - リソースの停止アクションを指示します。 * **promote** - スレーブ (昇格されていない) リソースからマスター (昇格された) リソースにリソースを昇格します。 * **demote** - マスター (昇格された) リソースからスレーブ (昇格されていない) リソースにリソースを降格します。 動作を指定しない場合のデフォルトの動作は **start** です。
kind オプション	制約の実施方法。**kind** オプションでは、以下の値が使用できます。 * **Optional** - 両方のリソースが指定の動作を実行する場合のみ適用されます。オプションの順序の詳細は 勧告的な順序付けの設定 を参照してください。 * **Mandatory** - 常に制約を有効にします (デフォルト値)。1 番目に指定したリソースが停止している場合や起動できない場合は、2 番目に指定したリソースが停止します。この強制的な順序付けの詳細は 強制的な順序付けの設定 を参照してください。 * **Serialize** - 指定するリソースに対して、複数の停止または起動のアクションが同時に発生しないようにします。指定した 1 番目および 2 番目のリソースは、いずれの順序でも起動できますが、片方が完了しないともう片方が開始しません。典型的なユースケースは、リソースの起動によりホストの負荷が高くなる場合です。

フィールド	説明
symmetrical オプション	true の場合、逆の制約は逆のアクションに適用されます (たとえば、A の開始後に B が開始する場合は、B が A が停止前に停止する場合など)。**kind** が **Serialize** の順序制約を対称にすることはできません。デフォルト値は、**Mandatory** および **Optional** の場合は **true** で、**Serialize** の場合は **false** です。

次のコマンドを使用すると、すべての順序の制約からリソースが削除されます。

```
pcs constraint order remove resource1 [resourceN]...
```

56.1. 強制的な順序付けの設定

必須順序制約は、最初のリソースに対する最初のアクションが正常に完了しない限り、2 番目のリソースの 2 番目のアクションが開始しないことを示しています。命令できるアクションが **stop** または **start** で、昇格可能なクローンが **demote** および **promote** とします。たとえば、A then B(start A then start B と同等) は、A が適切に開始しない限り、B が開始しないことを示しています。順序の制約は、この制約の **kind** オプションが **Mandatory** に設定されているか、デフォルトのままに設定されている場合は必須になります。

symmetrical オプションが **true** に設定されているか、デフォルトのままにすると、逆のアクションの命令は逆順になります。**start** と **stop** のアクションは対称になり、**demote** と **promote** は対称になります。たとえば、対称的に、promote A then start B 順序は stop B then demote A(B が正常に停止するまで A が降格しない) ことを示しています。対称順序は、A の状態を変更すると、B に予定されているアクションが発生すること示しています。たとえば、A then B と設定した場合は、失敗により A が再起動すると、B が最初に停止してから、A が停止し、それにより A が開始し、それにより B が開始することを示します。

クラスターは、それぞれの状態変化に対応することに注意してください。2 番目のリソースで停止操作を開始する前に 1 番目のリソースが再起動し、起動状態にあると、2 番目のリソースを再起動する必要がありません。

56.2. 勧告的な順序付けの設定

順序の制約に **kind=Optional** オプションを指定すると、制約はオプションと見なされ、両方のリソースが指定の動作を実行する場合にのみ適用されます。1 番目に指定しているリソースの状態を変更しても、2 番目に指定しているリソースには影響しません。

次のコマンドは、**VirtualIP** と **dummy_resource** という名前のリソースに、勧告的な順序の制約を設定します。

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

56.3. リソースセットへの順序の設定

一般的に、管理者は、複数のリソースの連鎖を作成する場合に順序を設定します (例: リソース A が開始してからリソース B を開始し、その後にリソース C を開始)。複数のリソースを作成して同じ場所に配置し (コロケーションを指定)、起動の順序を設定する必要がある場合は、このようなリソースが含まれるリソースグループを設定できます。

ただし、特定の順序で起動する必要があるリソースをリソースグループとして設定することが適切ではない場合があります。

- リソースを順番に起動するように設定する必要があるものの、リソースは必ずしも同じ場所に配置しない場合
- リソース C の前にリソース A または B のいずれかが起動する必要があるものの、A と B の間には関係が設定されていない場合
- リソース C およびリソース D の前にリソース A およびリソース B の両方が起動している必要があるものの、A と B、または C と D の間には関係が設定されていない場合

このような状況では、**pcs constraint order set** コマンドを使用して、1 つまたは複数のリソースセットに対して順序の制約を作成できます。

pcs constraint order set コマンドを使用して、リソースセットに以下のオプションを設定できます。

- **sequential** - リソースセットに順序を付ける必要があるかどうかを指定します。**true** または **false** に設定できます。デフォルト値は **true** です。
  sequential を **false** に設定すると、セットのメンバーに順序を設定せず、順序の制約にあるセット間で順序付けできます。そのため、このオプションは、制約に複数のセットが登録されている場合に限り有効です。それ以外の場合は、制約を設定しても効果がありません。
- **require-all** - 続行する前にセットの全リソースがアクティブである必要があるかどうかを指定します。**true** または **false** に設定できます。**require-all** を **false** に設定すると、次のセットに進む前に、セットの 1 つのリソースのみを開始する必要があります。**require-all** を **false** に設定しても、**sequential** が **false** に設定されている順序なしセットと併用しない限り、効果はありません。デフォルト値は **true** です。
- **action** - クラスターリソースの実行順序の決定 の表順序の制約のプロパティーで説明されているように、**start**、**promote**、**demote**、または **stop** に設定できます。
- **role** - **Stopped**、**Started**、**Master**、または **Slave** に設定できます。RHEL 8.5 では、**pcs** コマンドラインインターフェイスは **role** の値に **Promoted** および **Unpromoted** を受け入れます。**Promoted** および **Unpromoted** ロールは、**Master** および **Slave** ロールと機能的に等価です。

pcs constraint order set コマンドの **setoptions** パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- **id** - 定義する制約の名前を指定します。
- **kind** - クラスターリソースの実行順序の決定 の表順序の制約のプロパティーで説明されているように、制約を有効にする方法を指定します。
- **symmetrical** - クラスターリソースの実行順序の決定 の表順序の制約のプロパティーで説明しているように、逆の作用に逆の制約を適用するかどうかを設定します。

> pcs constraint order set **resource1 resource2** [**resourceN**]... [**options**] [set **resourceX resourceY** ... [**options**]] [setoptions [**constraint_options**]]

D1、**D2**、**D3** という 3 つのリソースがある場合は、次のコマンドを実行すると、この 3 つのリソースを、順序を指定したリソースセットとして設定します。

> **# pcs constraint order set D1 D2 D3**

この例では、**A**、**B**、**C**、**D**、**E**、および **F** という名前の 6 つのリソースがある場合に、以下のように、起動するリソースセットに順序制約を設定します。

- **A** と **B** は、互いに独立して起動します。
- **A** または **B** のいずれかが開始すると、**C** が開始します。
- **C** が開始すると、**D** が開始します。
- **D** が開始したら、**E** と **F** が互いに独立して起動します。

symmetrical=false が設定されているため、リソースの停止は、この制約の影響を受けません。

```
# pcs constraint order set A B sequential=false require-all=false set C D set E F
sequential=false setoptions symmetrical=false
```

56.4. PACEMAKER で管理されないリソース依存関係の起動順序の設定

クラスターは、クラスターが管理していない依存関係を持つリソースを含めることができます。この場合は、Pacemaker を起動する前にその依存関係を起動し、Pacemaker が停止した後に停止する必要があります。

systemd resource-agents-deps ターゲットを使用してこの条件を設定するために、スタートアップ順序を設定できます。このターゲットに対して **systemd** ドロップインユニットを作成すると、Pacemaker はこのターゲットに対して相対的な順序を適切に設定できます。

たとえば、クラスターが管理していない外部サービス **foo** に依存するリソースがクラスターに含まれている場合は、以下の手順を実行します。

1. 以下を含むドロップインユニット **/etc/systemd/system/resource-agents-deps.target.d/foo.conf** を作成します。

```
[Unit]
Requires=foo.service
After=foo.service
```

2. **systemctl daemon-reload** コマンドを実行します。

この方法で指定するクラスターの依存関係はサービス以外のものとなります。たとえば、**/srv** にファイルシステムをマウントする依存関係がある場合は、以下の手順を実行してください。

1. **/etc/fstab** ファイルに **/srv** が記載されていることを確認します。これは、システムマネージャーの設定が再読み込みされる際に、システムの起動時に **systemd** ファイルの **srv.mount** に自動的に変換されます。詳細は、man ページの **systemd.mount**(5) および **systemd-fstab-generator**(8) を参照してください。
2. ディスクのマウント後に Pacemaker が起動するようにするには、以下を含むドロップインユニット **/etc/systemd/system/resource-agents-deps.target.d/srv.conf** を作成します。

```
[Unit]
Requires=srv.mount
After=srv.mount
```

3. **systemctl daemon-reload** コマンドを実行します。

Pacemaker クラスターが使用する LVM ボリュームグループに、iSCSI ターゲットなど、リモートブロックストレージに存在する 1 つ以上の物理ボリュームが含まれている場合は、Pacemaker が起動する前にサービスが開始されるように、ターゲット用に **systemd resource-agents-deps** ターゲットと **systemd** ドロップインユニットを設定することができます。

以下の手順では、**blk-availability.service** を依存関係として設定します。**blk-availability.service** サービスは、**iscsi.service** などのサービスが含まれるラッパーです。お使いのデプロイメントでこれが必要な場合は、**blk-availability** の代わりに **iscsi.service**(iSCSI のみ) または **remote-fs.target** を依存関係として設定できます。

1. 以下を含むドロップインユニット **/etc/systemd/system/resource-agents-deps.target.d/blk-availability.conf** を作成します。

```
[Unit]
Requires=blk-availability.service
After=blk-availability.service
```

2. **systemctl daemon-reload** コマンドを実行します。

第57章 クラスターリソースのコロケーション

あるリソースの場所を、別のリソースの場所に依存させるように指定する場合は、コロケーションの制約を設定します。

2 つのリソース間にコロケーション制約を作成すると、リソースがノードに割り当てられる割り当て順序に重要な影響を及ぼす点に注意してください。リソース B の場所を把握していない場合は、リソース B に相対的となるようにリソース A を配置することができません。このため、コロケーションの制約を作成する場合は、リソース A をリソース B に対してコロケーションを設定するのか、もしくはリソース B をリソース A に対してコロケーションを設定するのかを考慮する必要があります。

また、コロケーションの制約を作成する際に注意しておきたいもう 1 つの点として、リソース A をリソース B に対してコロケーションを設定すると仮定した場合は、クラスターがリソース B に選択するノードを決定する際に、リソース A の優先度も考慮に入れます。

次のコマンドはコロケーションの制約を作成します。

> pcs constraint colocation add [master|slave] **source_resource** with [master|slave] **target_resource** [**score**] [**options**]

以下の表は、コロケーション制約を設定するのに使用するプロパティーおよびオプションをまとめています。

表57.1 コロケーション制約のパラメーター

パラメーター	説明
source_resource	コロケーションソース。制約の条件を満たさない場合、クラスターではこのリソースの実行を許可しないことを決定する可能性があります。
target_resource	コロケーションターゲット。クラスターは、このリソースの配置先を決定してから、ソースリソースの配置先を決定します。
score	正の値を指定するとリソースが同じノードで実行します。負の値を指定すると同じノードで実行しなくなります。デフォルト値である +**INFINITY** を指定すると、**source_resource** は必ず **target_resource** と同じノードで実行します。-**INFINITY** は、**source_resource** を **target_resource** と同じノードで実行してはならないことを示しています。

パラメーター	説明
influence オプション	(RHEL 8.4 以降) 従属リソースが移行のしきい値に達して失敗した場合に、クラスターで別のノードにプライマリーリソース (**source_resource**) と従属リソース (**target_resource)** を移行するか、クラスターでサーバーの切り替えなしに従属リソースをオフラインのままにするかを決定します。 **influence** コロケーション制約オプションには、**true** または **false** の値を指定できます。このオプションのデフォルト値は、従属リソースの **critical** リソースメタオプションの値で決定します。デフォルト値は **true** です。 このオプションの値が **true** の場合、Pacemaker は、プライマリーリソースと依存するリソース両方をアクティブに維持しようとします。依存するリソースが障害の移行しきい値に達すると、可能な場合は両方のリソースが別のノードに移行します。 このオプションの値が **false** の場合、Pacemaker は、依存するリソースのステータスが原因でプライマリーリソースを移行しないようにします。この場合、依存するリソースが障害の移行しきい値に達すると、プライマリーリソースがアクティブで、現在のノードに留まると停止します。

57.1. リソースの強制的な配置の指定

制約スコアが **+INFINITY** または **-INFINITY** の場合は常に強制的な配置が発生します。制約条件が満たされないと source_resource の実行が許可されません。**score=INFINITY** の場合、target_resource がアクティブではないケースが含まれます。

myresource1 を、常に **myresource2** と同じマシンで実行する必要がある場合は、次のような制約を追加します。

```
# pcs constraint colocation add myresource1 with myresource2 score=INFINITY
```

INFINITY を使用しているため、(何らかの理由で) **myresource2** がクラスターのいずれのノードでも実行できない場合は、**myresource1** の実行が許可されません。

または、逆の設定、つまり **myresource1** が **myresource2** と同じマシンでは実行されないようにクラスターを設定することもできます。この場合は **score=-INFINITY** を使用します。

```
# pcs constraint colocation add myresource1 with myresource2 score=-INFINITY
```

ここでも、**-INFINITY** を指定することで、制約は結合しています。このため、実行できる場所として残っているノードで **myresource2** がすでに実行されている場合は、いずれのノードでも **myresource1** を実行できなくなります。

57.2. リソースの勧告的な配置の指定

リソースの勧告的な配置は、リソースの配置は推奨ではあるものの、必須ではありません。制約のスコ

アが **-INFINITY** より大きく、**INFINITY** より小さい場合、クラスターは希望の設定に対応しようとしますが、クラスターリソースの一部を停止するという選択肢がある場合には、その設定が無視される場合があります。

57.3. 複数リソースのコロケーション

お使いの設定で、コロケーションと起動の順番を指定してリソースを作成する必要がある場合には、このようなリソースを含むリソースグループを設定できます。ただし、コロケーションを設定する必要があるリソースをリソースグループとして設定することが適切ではない場合もあります。

- リソースのセットにコロケーションを設定する必要があるものの、リソースが必ずしも順番に起動する必要がない場合
- リソース C を、リソース A またはリソース B のいずれかに対してコロケーションを設定する必要があるものの、リソース A とリソース B との間に関係が設定されていない場合
- リソース C およびリソース D を、リソース A およびリソース B の両方に対してコロケーションを設定する必要があるものの、A と B の間、または C と D の間に関係が設定されていない場合

このような状況では、**pcs constraint colocation set** コマンドを使用して、リソースの 1 つまたは複数のセットでコロケーションの制約を作成できます。

pcs constraint colocation set コマンドを使用すると、以下のオプションをリソースのセットに設定できます。

- **sequential** - セットのメンバーで相互のコロケーションが必要であるかどうかを指定します。**true** または **false** に設定できます。
  sequential を **false** に設定すると、このセットのメンバーがアクティブであるかどうかに関係なく、このセットのメンバーを、制約の中で、このセットの後にリストされている他のセットに対してコロケーションを設定できます。そのため、このオプションは制約でこのセットの後に他のセットが指定されている場合に限り有効です。他のセットが指定されていない場合は、制約の効果がありません。
- **role** - **Stopped**、**Started**、**Master**、または **Slave** に設定できます。

pcs constraint colocation set コマンドの **setoptions** パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- **id** - 定義する制約の名前を指定します。
- **score** - 制約の優先度を示します。このオプションの詳細は、場所の制約の設定 の表の場所の制約オプションを参照してください。

セットのメンバーをリストすると、各メンバーは、自身の前のメンバーに対してコロケーションが設定されます。たとえば、set A B は B が A の場所に配置されることを意味します。しかし、複数のセットをリストする場合、各セットはその後のメンバーと同じ場所に配置されます。たとえば、set C D sequential=false set A B は、C と D のセットが、A と B のセットと同じ場所に配置されることを意味します (ただし、C と D には関係がなく、B は A にはコロケーションが設定されています)。

以下のコマンドは、リソースのセットにコロケーションの制約を作成します。

```
pcs constraint colocation set resource1 resource2] [resourceN]... [options] [set resourceX resourceY] ... [options]] [setoptions [constraint_options]]
```

コロケーション制約を削除する場合はコマンドに source_resource を付けて使用します。

pcs constraint colocation remove **source_resource target_resource**

第58章 リソース制約とリソース依存関係の表示

設定した制約を表示させるコマンドがいくつかあります。設定されたリソース制約をすべて表示するか、特定のタイプのリソース制約だけに表示を制限することもできます。また、設定したリソース依存関係を表示できます。

設定済みの全制約の表示

以下のコマンドは、現在の場所、順序、ロケーションの制約をすべて表示します。**--full** オプションを指定すると、制約の内部 ID が表示されます。

```
pcs constraint [list|show] [--full]
```

RHEL 8.2 では、リソース制約をデフォルトでリスト表示すると、期限切れの制約が表示されなくなりました。リストに期限切れの制約を含めるには、**pcs constraint** コマンドに **--all** オプションを使用します。これにより、期限切れの制約のリストが表示され、制約とそれに関連するルールが **(expired)** として表示されます。

場所の制約の表示

以下のコマンドは、現在の場所の制約をリスト表示します。

- **resources** を指定すると、リソース別に場所の制約が表示されます。これはデフォルトの動作です。
- **nodes** を指定すると、ノード別に場所の制約が表示されます。
- 特定のリソースまたはノードを指定すると、そのリソースまたはノードの情報のみが表示されます。

```
pcs constraint location [show [resources [resource...]] | [nodes [node...]]] [--full]
```

順序の制約の表示

以下のコマンドは、現在の順序の制約をすべて表示します。

```
pcs constraint order [show]
```

コロケーション制約の表示

次のコマンドは、現在のコロケーション制約をリスト表示します。

```
pcs constraint colocation [show]
```

リソース固有の制約の表示

以下のコマンドは、特定リソースを参照する制約をリスト表示します。

```
pcs constraint ref resource ...
```

リソース依存関係の表示 (Red Hat Enterprise Linux 8.2 以降)

次のコマンドは、クラスターリソース間の関係をツリー構造で表示します。

```
pcs resource relations resource [--full]
```

--full オプションを指定すると、制約 ID およびリソースタイプを含む追加情報が表示されます。

以下の例では、3 つのリソースが設定されています。C, D、および E。

```
# pcs constraint order start C then start D
Adding C D (kind: Mandatory) (Options: first-action=start then-action=start)
# pcs constraint order start D then start E
Adding D E (kind: Mandatory) (Options: first-action=start then-action=start)

# pcs resource relations C
C
`- order
  |  start C then start D
  `- D
    `- order
      |  start D then start E
      `- E
# pcs resource relations D
D
|- order
|  |  start C then start D
|  `- C
`- order
  |  start D then start E
  `- E
# pcs resource relations E
E
`- order
  |  start D then start E
  `- D
    `- order
      |  start C then start D
      `- C
```

以下の例では、2 つのリソースが設定されています。A および B。リソース A および B はリソースグループ G の一部です。

```
# pcs resource relations A
A
`- outer resource
  `- G
    `- inner resource(s)
      |  members: A B
      `- B
# pcs resource relations B
B
`- outer resource
  `- G
    `- inner resource(s)
      |  members: A B
      `- A
# pcs resource relations G
G
`- inner resource(s)
```

```
|  members: A B
|- A
`- B
```

第59章 ルールによるリソースの場所の決定

さらに複雑な場所の制約には、Pacemaker のルールを使用してリソースの場所を決定できます。

59.1. PACEMAKER ルール

Pacemaker ルールを使用すると、設定をより動的に作成できます。ルールには、(ノード属性を使用して) 時間ベースで異なる処理グループにマシンを割り当て、場所の制約の作成時にその属性を使用する方法があります。

各ルールには、日付などの様々な式だけでなく、その他のルールも含めることができます。ルールの **boolean-op** フィールドに応じて各種の式の結果が組み合わされ、最終的にそのルールが **true** または **false** のどちらに評価されるかどうかが決まります。次の動作は、ルールが使用される状況に応じて異なります。

表59.1 ルールのプロパティー

フィールド	説明
role	リソースが指定のロールにある場合にのみ適用するルールを制限します。使用できる値は以下のようになります。**Started**、**Slave**、**Master**備考:**role="Master"** が指定されたルールは、クローンインスタンスの最初の場所を判断できません。どのアクティブインスタンスが昇格されるかにのみ影響します。
score	ルールが **true** に評価される場合に適用されるスコア。場所の制約として、ルールでの使用に制限されます。
score-attribute	ルールが **true** に評価されると検索し、スコアとして使用するノード属性。場所の制約として、ルールでの使用に制限されます。
boolean-op	複数の式オブジェクトからの結果を組み合わせる方法。使用できる値は **and** および **or** です。デフォルト値は **and** です。

59.1.1. ノード属性の式

ノードで定義される属性に応じてリソースを制御する場合に使用されるノード属性の式です。

表59.2 式のプロパティー

フィールド	説明
attribute	テストするノード属性。

フィールド	説明
type	値をテストする方法を指定します。使用できる値は、**string**、**integer**、**number** (RHEL 8.4 以降)、**version** です。デフォルト値は **string** です。
operation	実行する比較動作。設定できる値は以下のとおりです。 * **lt** - ノード属性の値が **value** 未満の場合は True * **gt** - ノード属性の値が **value** を超える場合は True * **lte** - ノード属性の値が **value** 未満または同等になる場合は True * **gte** - ノード属性の値が **value** を超えるか、同等になる場合は True * **eq** - ノード属性の値が **value** と同等になる場合に True * **ne** - ノード属性の値が **value** と同等にならない場合は True * **defined** - ノードに指定属性がある場合は True * **not_defined** - ノードに指定属性がない場合は True
value	比較のためにユーザーが提供した値 (**operation** が **defined** または **not_defined** に設定されていない場合に限り必要)

管理者が追加する属性のほかに、以下の表で説明されているように、クラスターは、使用可能な各ノードに特殊な組み込みノード属性を定義します。

表59.3 組み込みノード属性

名前	説明
#uname	ノード名
#id	ノード ID
#kind	ノードタイプ。使用できる値は、**cluster**、**remote**、および **container** です。**kind** の値は、**ocf:pacemaker:remote** リソースで作成された Pacemaker リモートノードの **remote**、および Pacemaker リモートゲストノードおよびバンドルノードの **container** です。

名前	説明
#is_dc	このノードが指定コントローラー (DC) の場合は **true**、それ以外の場合は **false**
#cluster_name	**cluster-name** クラスタープロパティーの値 (設定されている場合)。
#site_name	**site-name** ノード属性の値 (設定されている場合)。それ以外は **#cluster-name** と同じ。
#role	関連する昇格可能なクローンがこのノードで果たすロール。昇格可能なクローンに対する場所の制約のルール内でのみ有効です。

59.1.2. 時刻と日付ベースの式

日付の式は、現在の日付と時刻に応じてリソースまたはクラスターオプションを制御する場合に使用します。オプションで日付の詳細を含めることができます。

表59.4 日付の式のプロパティー

フィールド	説明
start	ISO 8601 仕様に準じた日付と時刻。
end	ISO 8601 仕様に準じた日付と時刻。
operation	状況に応じて、現在の日付と時刻を start と end のいずれかの日付、または両方の日付と比較します。設定できる値は以下のとおりです。 * **gt** - 現在の日付と時刻が **start** 以降の場合は True * **lt** - 現在の日付と時刻が **end** 以前の場合は True * **in_range** - 現在の日付と時刻が **start** 以降、かつ **end** 以前の場合は True * **date-spec** - 現在の日付/時刻に対して cron のような比較を実行します。

59.1.3. 日付の詳細

日付の詳細は、時間に関係する cron のような式を作成するのに使用されます。各フィールドには 1 つの数字または範囲が含まれます。指定のないフィールドは、デフォルトを 0 に設定するのではなく、無視されます。

たとえば、**monthdays="1"** は各月の最初の日と一致し、**hours="09-17"** は午前 9 時から午後 5 時まで (両時間を含む) の時間と一致します。ただし、**weekdays="1,2"** または **weekdays="1-2,5-6"** には複数の範囲が含まれるため、指定することはできません。

表59.5 日付詳細のプロパティー

フィールド	説明
id	日付の一意の名前
hours	設定できる値は、0-23
monthdays	設定できる値は、0-31 (月および年により異なる)
weekdays	設定できる値は、1-7 (1=月曜、7=日曜)
yeardays	設定できる値は、1-366 (年により異なる)
months	設定できる値は、1-12
weeks	設定できる値は、1-53 (**weekyear** により異なる)
years	グレゴリオ暦 (新暦) に準じる年
weekyears	グレゴリオ暦の年とは異なる場合がある (例: **2005-001 Ordinal** は **2005-01-01 Gregorian** であり **2004-W53-6 Weekly** でもある)
moon	設定できる値は、0-7 (0 は新月。4 満月)

59.2. ルールを使用した PACEMAKER の場所の制約の設定

以下のコマンドを使用して、ルールを使用する Pacemaker 制約を使用します。**score** を省略すると、デフォルトの INFINITY に設定されます。**resource-discovery** を省略すると、デフォルトの **always** に設定されます。

resource-discovery オプションの詳細は、ノードのサブセットへのリソース検出を制限 を参照してください。

基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

ルールを使用して場所の制約を設定する場合は、**score** を正または負の値にすることができます。正の値は prefers を示し、負の値は avoids を示します。

```
pcs constraint location rsc rule [resource-discovery=option] [role=master|slave] [score=score |
score-attribute=attribute] expression
```

expression オプションは、以下のいずれかに設定できます。ここで、**duration_options** および **date_spec_options** は、日付の詳細 の表日付詳細のプロパティーで説明されているように、hours、monthdays、weekdays、yeardays、months、weeks、years、weekyears、moon になります。

- **defined|not_defined attribute**
- **attribute lt|gt|lte|gte|eq|ne [string|integer|number**(RHEL 8.4 以降)**|version] value**

- **date gt|lt date**
- **date in_range date to date**
- **date in_range date to duration duration_options …**
- **date-spec date_spec_options**
- **expression and|or expression**
- **(expression)**

持続時間は、計算により **in_range** 操作の終了を指定する代替方法です。たとえば、19 カ月間を期間として指定できます。

以下の場所の制約は、現在が 2018 年の任意の時点である場合に true の式を設定します。

```
# pcs constraint location Webserver rule score=INFINITY date-spec years=2018
```

以下のコマンドは、月曜日から金曜日までの 9 am から 5 pm までが true となる式を設定します。hours の値 16 には、時間 (hour) の値が一致する 16:59:59 までが含まれます。

```
# pcs constraint location Webserver rule score=INFINITY date-spec hours="9-16"
weekdays="1-5"
```

以下のコマンドは、13 日の金曜日が満月になると true になる式を設定します。

```
# pcs constraint location Webserver rule date-spec weekdays=5 monthdays=13 moon=4
```

ルールを削除するには、以下のコマンドを使用します。削除しているルールがその制約内で最後のルールになる場合は、その制約も削除されます。

```
pcs constraint rule remove rule_id
```

第60章 クラスターリソースの管理

クラスターリソースの表示、変更、および管理に使用できるコマンドは複数あります。

60.1. 設定されているリソースの表示

設定されているリソースのリストを表示する場合は、次のコマンドを使用します。

```
pcs resource status
```

例えば、**VirtualIP** と言う名前のリソースと **WebSite** という名前のリソースでシステムを設定していた場合、**pcs resource status** コマンドを実行すると次のような出力が得られます。

```
# pcs resource status
 VirtualIP (ocf::heartbeat:IPaddr2): Started
 WebSite (ocf::heartbeat:apache): Started
```

リソースに設定されているパラメーターを表示する場合は、次のコマンドを使用します。

```
pcs resource config resource_id
```

たとえば、次のコマンドは、現在設定されているリソース **VirtualIP** のパラメーターを表示します。

```
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

RHEL 8.5 では、個々のリソースのステータスを表示するのに、次のコマンドを使用します。

```
pcs resource status resource_id
```

たとえば、システムが **VirtualIP** という名前のリソースで設定されていると、**pcs resource status VirtualIP** コマンドは以下の出力を表示します。

```
# pcs resource status VirtualIP
 VirtualIP      (ocf::heartbeat:IPaddr2):       Started
```

RHEL 8.5 では、特定のノードで実行されているリソースのステータスを表示するのに、以下のコマンドを使用します。このコマンドを使用すると、クラスターとリモートノードの両方でリソースのステータスを表示できます。

```
pcs resource status node=node_id
```

たとえば、**node-01** が **VirtualIP** および **WebSite** という名前のリソースを実行している場合、**pcs resource status node=node-01** コマンドは以下の出力を表示します。

```
# pcs resource status node=node-01
 VirtualIP      (ocf::heartbeat:IPaddr2):       Started
 WebSite        (ocf::heartbeat:apache):        Started
```

60.2. PCS コマンドとしてのクラスターリソースのエクスポート

Red Hat Enterprise Linux 8.7 では、**pcs resource config** コマンドの **--output-format=cmd** オプションを使用して、別のシステムに設定済みのクラスターデバイスを再作成するのに使用できる **pcs** コマンドを表示できます。

以下のコマンドは、Red Hat 高可用性クラスター内のアクティブ/パッシブ Apache HTTP サーバー用に作成された 4 つのリソース (**LVM-activate** リソース、**Filesystem** リソース、**IPaddr2** リソース、および **Apache** リソース) を作成します。

```
# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group apachegroup
# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv" directory="/var/www"
fstype="xfs" --group apachegroup
# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 cidr_netmask=24 --group apachegroup
# pcs resource create Website apache configfile="/etc/httpd/conf/httpd.conf"
statusurl="http://127.0.0.1/server-status" --group apachegroup
```

リソースを作成した後、次のコマンドを実行すると、別のシステムでそれらのリソースを再作成するために使用できる **pcs** コマンドが表示されます。

```
# pcs resource config --output-format=cmd
pcs resource create --no-default-ops --force -- my_lvm ocf:heartbeat:LVM-activate \
  vg_access_mode=system_id vgname=my_vg \
  op \
    monitor interval=30s id=my_lvm-monitor-interval-30s timeout=90s \
    start interval=0s id=my_lvm-start-interval-0s timeout=90s \
    stop interval=0s id=my_lvm-stop-interval-0s timeout=90s;
pcs resource create --no-default-ops --force -- my_fs ocf:heartbeat:Filesystem \
  device=/dev/my_vg/my_lv directory=/var/www fstype=xfs \
  op \
    monitor interval=20s id=my_fs-monitor-interval-20s timeout=40s \
    start interval=0s id=my_fs-start-interval-0s timeout=60s \
    stop interval=0s id=my_fs-stop-interval-0s timeout=60s;
pcs resource create --no-default-ops --force -- VirtualIP ocf:heartbeat:IPaddr2 \
  cidr_netmask=24 ip=198.51.100.3 \
  op \
    monitor interval=10s id=VirtualIP-monitor-interval-10s timeout=20s \
    start interval=0s id=VirtualIP-start-interval-0s timeout=20s \
    stop interval=0s id=VirtualIP-stop-interval-0s timeout=20s;
pcs resource create --no-default-ops --force -- Website ocf:heartbeat:apache \
  configfile=/etc/httpd/conf/httpd.conf statusurl=http://127.0.0.1/server-status \
  op \
    monitor interval=10s id=Website-monitor-interval-10s timeout=20s \
    start interval=0s id=Website-start-interval-0s timeout=40s \
    stop interval=0s id=Website-stop-interval-0s timeout=60s;
pcs resource group add apachegroup \
  my_lvm my_fs VirtualIP Website
```

pcs コマンドまたは 1 つの設定済みリソースのみ再作成するために使用できるコマンドを表示するには、そのリソースのリソース ID を指定します。

```
# pcs resource config VirtualIP --output-format=cmd
pcs resource create --no-default-ops --force -- VirtualIP ocf:heartbeat:IPaddr2 \
  cidr_netmask=24 ip=198.51.100.3 \
```

```
  op \
    monitor interval=10s id=VirtualIP-monitor-interval-10s timeout=20s \
    start interval=0s id=VirtualIP-start-interval-0s timeout=20s \
    stop interval=0s id=VirtualIP-stop-interval-0s timeout=20s
```

60.3. リソースパラメーターの修正

設定されているリソースのパラメーターを変更する場合は、次のコマンドを使用します。

```
pcs resource update resource_id [resource_options]
```

以下のコマンドシーケンスでは、**VirtualIP** リソースに設定したパラメーターの初期値、**ip** パラメーターの値を変更するコマンド、変更されたパラメーター値を示しています。

```
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
# pcs resource update VirtualIP ip=192.169.0.120
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.169.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

注記

pcs resource update コマンドを使用してリソースの動作を更新すると、特に呼び出しのないオプションはデフォルト値にリセットされます。

60.4. クラスターリソースの障害ステータスの解除

リソースに障害が発生すると、クラスターの状態を表示するときに障害メッセージが表示されます。このリソースを解決する場合、**pcs resource cleanup** コマンドで障害状態を消去できます。このコマンドはリソースの状態と **failcount** をリセットし、リソースの動作履歴を消去して現在の状態を再検出するようにクラスターに指示します。

以下のコマンドは、**resource_id** によって指定されたリソースをクリーンアップします。

```
pcs resource cleanup resource_id
```

resource_id を指定しないと、このコマンドは、全リソースのリソース状態と **failcount** をリセットします。

pcs resource cleanup コマンドは、失敗したアクションとして表示されるリソースのみを検証します。全ノードの全リソースを調査するには、次のコマンドを入力します。

```
pcs resource refresh
```

デフォルトでは、**pcs resource refresh** コマンドは、リソースのステータスが分かっているノードだけを検証します。ステータスが分からないすべてのリソースを検証するには、以下のコマンドを実行します。

```
pcs resource refresh --full
```

60.5. クラスター内のリソースの移動

Pacemaker は、リソースを別のノードに移動するように設定し、必要に応じて手動でリソースを移動するように設定する様々なメカニズムを提供します。

クラスターリソースの手動による移行 に従って、**pcs resource move** コマンドと **pcs resource relocate** コマンドで、クラスターのリソースを手動で移動します。このコマンドの他にも、クラスターリソースの無効化、有効化、および禁止 に従ってリソースを有効、無効、および禁止にしてクラスターリソースの挙動を制御することもできます。

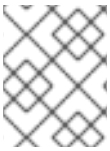
失敗した回数が、定義した値を超えると、新しいノードに移動し、外部接続が失われた時にリソースを移動するようにクラスターを設定できます。

60.5.1. 障害発生によるリソースの移動

リソースの作成時に、リソースに **migration-threshold** オプションを設定し、定義した回数だけ障害が発生した場合にリソースが新しいノードに移動されるように設定できます。このしきい値に一度到達すると、このノードでは、以下が行われるまで、障害が発生したリソースを実行できなくなります。

- 管理者が **pcs resource cleanup** コマンドを使用して、リソースの **failcount** を手動でリセットします。
- リソースの **failure-timeout** 値に到達します。

デフォルトで、**migration-threshold** の値が **INFINITY** に設定されています。**INFINITY** は、内部的に非常に大きな有限数として定義されます。0 にすると、**migration-threshold** 機能が無効になります。



注記

リソースの **migration-threshold** を設定するのと、リソースの状態を維持しながら別の場所に移動させるようにリソースの移動を設定するのは同じではありません。

次の例では、**dummy_resource** リソースに、移行しきい値 10 を追加します。この場合は、障害が 10 回発生すると、そのリソースが新しいノードに移動します。

```
# pcs resource meta dummy_resource migration-threshold=10
```

次のコマンドを使用すると、クラスター全体にデフォルトの移行しきい値を追加できます。

```
# pcs resource defaults update migration-threshold=10
```

リソースの現在の障害ステータスと制限を確認するには、**pcs resource failcount show** コマンドを使用します。

移行しきい値の概念には、リソース起動の失敗とリソース停止の失敗の 2 つの例外があります。クラスタープロパティー **start-failure-is-fatal** が **true** に設定された場合 (デフォルト) は、起動の失敗により **failcount** が **INFINITY** に設定され、リソースが常に即座に移動するようになります。

停止時の失敗は、起動時とは若干異なり、極めて重大となります。リソースの停止に失敗し STONITH が有効になっていると、リソースを別のノードで起動できるように、クラスターによるノードのフェンスが行われます。STONITH を有効にしていない場合はクラスターに続行する手段がないため、別の

ノードでのリソース起動は試行されません。ただし、障害のタイムアウト後に再度停止が試行されます。

60.5.2. 接続状態の変更によるリソースの移動

以下の 2 つのステップに従って、外部の接続が失われた場合にリソースが移動するようにクラスターを設定します。

1. **ping** リソースをクラスターに追加します。**ping** リソースは、同じ名前のシステムユーティリティーを使用して、マシン (DNS ホスト名または IPv4/IPv6 アドレスによって指定されるリスト) にアクセス可能であるかをテストし、その結果を使用して **pingd** と呼ばれるノード属性を維持します。
2. 接続が失われたときに別のノードにリソースを移動させる、リソース場所制約を設定します。

以下の表には、**ping** リソースに設定できるプロパティーを紹介しています。

表60.1 ping リソースのプロパティー

フィールド	説明
dampen	今後の変更が発生するまでに待機する (弱める) 時間。これにより、クラスターノードが、わずかに異なる時間に接続が失われたことに気が付いたときに、クラスターでリソースがバウンスするのを防ぎます。
multiplier	接続された ping ノードの数は、ノードの数にこの値を掛けて、スコアを取得します。複数の ping ノードが設定された場合に便利です。
host_list	現在の接続状態を判断するために接続するマシン。使用できる値には、解決可能な DNS ホスト名、IPv4 アドレス、および IPv6 アドレスが含まれます。ホストリストのエントリーはスペースで区切られます。

次のコマンド例は、**gateway.example.com** への接続を検証する **ping** リソースを作成します。実際には、ネットワークゲートウェイやルーターへの接続を検証します。リソースがすべてのクラスターノードで実行されるように、**ping** リソースをクローンとして設定します。

```
# pcs resource create ping ocf:pacemaker:ping dampen=5s multiplier=1000
host_list=gateway.example.com clone
```

以下の例は、既存のリソース **Webserver** に場所制約ルールを設定します。これにより、**Webserver** リソースが現在実行しているホストが **gateway.example.com** へ ping できない場合に、Webserver リソースを **gateway.example.com** へ ping できるホストに移動します。

```
# pcs constraint location Webserver rule score=-INFINITY pingd lt 1 or not_defined pingd
```

60.6. 監視操作の無効化

定期的な監視を停止する最も簡単な方法は、監視を削除することです。ただし、一時的に無効にしたい場合もあります。このような場合は、操作の定義に **enabled="false"** を追加します。監視操作を再度有効にするには、操作の定義に **enabled="true"** を設定します。

pcs resource update コマンドを使用してリソースの動作を更新すると、特に呼び出しのないオプションはデフォルト値にリセットされます。たとえば、カスタムのタイムアウト値 600 を使用して監視操作を設定している場合に以下のコマンドを実行すると、タイムアウト値がデフォルト値の 20 にリセットされます (**pcs resource op default** コマンドを使用しても、デフォルト値を設定できます)。

```
# pcs resource update resourceXZY op monitor enabled=false
# pcs resource update resourceXZY op monitor enabled=true
```

このオプションの元の値 600 を維持するために、監視操作に戻す場合は、以下の例のように、その値を指定する必要があります。

```
# pcs resource update resourceXZY op monitor timeout=600 enabled=true
```

60.7. クラスターリソースタグの設定および管理

Red Hat Enterprise Linux 8.3 では、**pcs** コマンドを使用してクラスターリソースをタグ付けできます。これにより、1 つのコマンドで、指定したリソースセットを有効化、無効化、マネージド化、または管理非対象化することができます。

60.7.1. カテゴリー別に管理するためのクラスターリソースのタグ付け

以下の手順では、リソースタグを使用して 2 つのリソースをタグ付けし、タグ付けされたリソースを無効にします。この例では、タグ付けする既存のリソースの名前は **d-01** および **d-02** です。

手順

1. **special-resources** という名前のタグ (**d-01** および **d-02**) を作成します。

```
[root@node-01]# pcs tag create special-resources d-01 d-02
```

2. リソースタグ設定を表示します。

```
[root@node-01]# pcs tag config
special-resources
  d-01
  d-02
```

3. **special-resources** タグが付けられた全リソース を無効にします。

```
[root@node-01]# pcs resource disable special-resources
```

4. リソースのステータスを表示して、**d-01** および **d-02** リソースが無効になっていることを確認します。

```
[root@node-01]# pcs resource
  * d-01        (ocf::pacemaker:Dummy): Stopped (disabled)
  * d-02        (ocf::pacemaker:Dummy): Stopped (disabled)
```

pcs resource disable コマンドに加え、**pcs resource enable**、**pcs resource manage** および **pcs resource unmanage** コマンドはタグ付けされたリソースの管理をサポートします。

リソースタグを作成したら、以下を実行します。

- **pcs tag delete** コマンドを使用して、リソースタグを削除できます。
- **pcs tag update** コマンドを使用して、既存のリソースタグのリソースタグ設定を変更できます。

60.7.2. タグ付けされたクラスターリソースの削除

pcs コマンドでは、タグ付けされたクラスターリソースを削除できません。タグ付けられたリソースを削除するには、以下の手順に従います。

手順

1. リソースタグを削除します。
   a. 以下のコマンドは、**special-resources** タグの付いたすべてのリソースからこのリソースタグを削除します。

```
[root@node-01]# pcs tag remove special-resources
[root@node-01]# pcs tag
 No tags defined
```

   b. 以下のコマンドは、リソース **d-01** からのみリソースタグ **special-resources** を削除します。

```
[root@node-01]# pcs tag update special-resources remove d-01
```

2. リソースを削除します。

```
[root@node-01]# pcs resource delete d-01
Attempting to stop: d-01... Stopped
```

第61章 複数のノードでアクティブなクラスターリソース (クローンリソース) の作成

クラスターリソースが複数のノードでアクティブになるように、リソースのクローンを作成できます。たとえば、ノードの分散のために、クローンとなるリソースを使用して、クラスター全体に分散させる IP リソースのインスタンスを複数設定できます。リソースエージェントが対応していれば、任意のリソースのクローンを作成できます。クローンは、1つのリソースまたは1つのリソースグループで構成されます。

注記

同時に複数のノードでアクティブにできるリソースのみがクローンに適しています。たとえば、共有メモリーデバイスから **ext4** などの非クラスター化ファイルシステムをマウントする **Filesystem** リソースのクローンは作成しないでください。**ext4** パーティションはクラスターを認識しないため、同時に複数のノードから繰り返し行われる読み取りや書き込みの操作には適していません。

61.1. クローンリソースの作成および削除

リソースと、そのリソースのクローンを同時に作成できます。

以下のコマンドを実行して、リソースと、そのリソースのクローンを作成します。

RHEL 8.4 以降:

> pcs resource create **resource_id** [**standard**:[**provider**:]]**type** [**resource options**] [meta **resource meta options**] clone [**clone_id**] [**clone options**]

RHEL 8.3 以前:

> pcs resource create **resource_id** [**standard**:[**provider**:]]**type** [**resource options**] [meta **resource meta options**] clone [**clone options**]

デフォルトでは、クローンの名前は **resource_id-clone** となります。RHEL 8.4 では、 **clone_id** オプションの値を指定して、クローンのカスタム名を設定できます。

1つのコマンドで、リソースグループの作成と、リソースグループのクローン作成の両方を行うことはできません。

作成済みリソースまたはリソースグループのクローンを作成する場合は、次のコマンドを実行します。

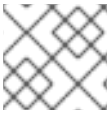
RHEL 8.4 以降:

> pcs resource clone **resource_id** | **group_id** [**clone_id**][**clone options**]...

RHEL 8.3 以前:

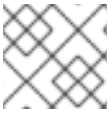
> pcs resource clone **resource_id** | **group_id** [**clone options**]...

デフォルトでは、クローンの名前は **resource_id-clone** または **group_name-clone** です。RHEL 8.4 では、**clone_id** オプションの値を指定して、クローンのカスタム名を設定できます。



注記

リソース設定の変更が必要なのは、1 つのノードのみです。



注記

制約を設定する場合は、グループ名またはクローン名を必ず使用します。

リソースのクローンを作成すると、クローンのデフォルト名は、リソース名に **-clone** を付けた名前になります。次のコマンドは、タイプが **apache** のリソース **webfarm** と、そのクローンとなるリソース **webfarm-clone** を作成します。

```
# pcs resource create webfarm apache clone
```

注記

あるリソースまたはリソースグループのクローンを、別のクローンの後にくるように作成する場合は、多くの場合 **interleave=true** オプションを設定する必要があります。これにより、依存されているクローンが同じノードで停止または開始した時に、依存しているクローンのコピーを停止または開始できるようになります。このオプションを設定しない場合は、次のようになります。クローンリソース B がクローンリソース A に依存していると、ノードがクラスターから離れてから戻ってきてから、そのノードでリソース A が起動すると、リソース B の全コピーが、全ノードで再起動します。これは、依存しているクローンリソースに **interleave** オプションが設定されていない場合は、そのリソースの全インスタンスが、そのリソースが依存しているリソースの実行インスタンスに依存するためです。

リソースまたはリソースグループのクローンを削除する場合は、次のコマンドを使用します。リソースやリソースグループ自体は削除されません。

```
pcs resource unclone resource_id | clone_id | group_name
```

以下の表には、クローンのリソースに指定できるオプションを示しています。

表61.1 リソースのクローンオプション

フィールド	説明
priority, target-role, is-managed	リソースのメタオプションの設定 の表リソースのメタオプションで説明されているように、クローンされたリソースから継承されるオプション。
clone-max	起動するリソースのコピーの数。デフォルトは、クラスター内のノード数です。
clone-node-max	1 つのノードで起動できるリソースのコピー数。デフォルト値は **1** です。

フィールド	説明
notify	クローンのコピーを停止したり起動する時に、前もって、およびアクションが成功した時に、他のコピーに通知します。使用できる値は **false** および **true** です。デフォルト値は **false** です。
globally-unique	クローンの各コピーで異なる機能を実行させるかどうか。使用できる値は **false** および **true** です。 このオプションの値を **false** にすると、リソースが実行しているすべてのノードで同じ動作を行うため、1台のマシンごとに実行できるクローンのコピーは1つです。 このオプションの値を **true** にすると、任意のマシンで実行中のクローンのコピーが、別のインスタンスが別のノードまたは同じノードで実行されているかに関係なく、そのインスタンスと同等ではありません。**clone-node-max** の値が1より大きい場合にはデフォルト値が **true** になり、小さい場合は **false** がデフォルト値になります。
ordered	コピーを、(並列ではなく) 連続して開始する必要があります。使用できる値は **false** および **true** です。デフォルト値は **false** です。
interleave	(クローン間の) 順序制約の動作を変更して、(2番目のクローンの全インスタンスが終了するまで待機する代わりに) 2番目のクローンと同じノードにあるコピーが起動または停止するとすぐに、最初のクローンのコピーが起動または停止できるようにします。使用できる値は **false** および **true** です。デフォルト値は **false** です。
clone-min	このフィールドに値を指定した場合は、**interleave** オプションが **true** に設定されていても、元のクローンの後に順序付けされたクローンは、元のクローンに指定された数だけインスタンスが実行するまで、起動できません。

安定した割り当てパターンを実現するために、クローンは、デフォルトでわずかに固定 (sticky) されています。これは、クローンが実行しているノードにとどまることをわずかに優先することを示します。**resource-stickiness** の値を指定しないと、クローンが使用する値は1となります。値を小さくすることで他のリソースのスコア計算への阻害を最小限に抑えながら、Pacemaker によるクラスター内の不要なコピーの移動を阻止することができます。**resource-stickiness** リソースのメタオプションを設定する方法は、リソースのメタオプションの設定 を参照してください。

61.2. クローンリソース制約の表示

ほとんどの場合、アクティブなクラスターノードに対するクローンのコピーは1つです。ただし、リソースクローンの **clone-max** には、そのクラスター内のノード合計より小さい数を設定できます。こ

の場合は、リソースの場所の制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。これらの制約は、クローンの ID を使用する必要があることを除いて、通常のリソースの制約と同じように記述されます。

次のコマンドは、クラスターがリソースのクローン **webfarm-clone** を **node1** に優先的に割り当てる場所の制約を作成します。

```
# pcs constraint location webfarm-clone prefers node1
```

順序制約の動作はクローンでは若干異なります。以下の例では、**interleave** クローンオプションをデフォルトの **false** のままにしているため、起動する必要がある **webfarm-clone** のすべてのインスタンスが起動するまで、**webfarm-stats** のインスタンスは起動しません。**webfarm-clone** のコピーを 1 つも起動できない場合にのみ、**webfarm-stats** がアクティブになりません。さらに、**webfarm-stats** が停止するまで待機してから、**webfarm-clone** が停止します。

```
# pcs constraint order start webfarm-clone then webfarm-stats
```

通常のリソース (またはリソースグループ) とクローンのコロケーションは、リソースを、クローンのアクティブコピーを持つ任意のマシンで実行できることを意味します。クラスターは、クローンが実行している場所と、リソース自体の場所の優先度に基づいてコピーを選択します。

クローン間のコロケーションも可能です。この場合、クローンに対して許可できる場所は、そのクローンが実行中のノード (または実行するノード) に限定されます。割り当ては通常通り行われます。

以下のコマンドは、コロケーション制約を作成し、**webfarm-stats** リソースが **webfarm-clone** のアクティブなコピーと同じノードで実行するようにします。

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

61.3. 昇格可能なクローンリソース

昇格可能なクローンリソースは、**promotable** メタ属性が **true** に設定されているクローンリソースです。昇格可能なクローンリソースにより、インスタンスの操作モードは、**master** および **slave** のいずれかにできます。モードの名前には特別な意味はありませんが、インスタンスの起動時に、**Slave** 状態で起動する必要があるという制限があります。

61.3.1. 昇格可能なクローンリソースの作成

次のコマンドを実行すると、リソースを昇格可能なクローンとして作成できます。

RHEL 8.4 以降:

```
pcs resource create resource_id [standard:[provider:]]type [resource options] promotable
[clone_id] [clone options]
```

RHEL 8.3 以前:

```
pcs resource create resource_id [standard:[provider:]]type [resource options] promotable [clone
options]
```

デフォルトでは、昇格可能なクローンの名前は **resource_id-clone** となります。

RHEL 8.4 では、**clone_id** オプションの値を指定して、クローンのカスタム名を設定できます。

また、次のコマンドを使用して、作成済みのリソースまたはリソースグループから、昇格可能なリソースを作成することもできます。

RHEL 8.4 以降:

> pcs resource promotable **resource_id** [**clone_id**] [**clone options**]

RHEL 8.3 以前:

> pcs resource promotable **resource_id** [**clone options**]

デフォルトでは、昇格可能なクローンの名前は **resource_id-clone** または **group_name-clone** になります。

RHEL 8.4 では、**clone_id** オプションの値を指定して、クローンのカスタム名を設定できます。

以下の表には、昇格可能なリソースに指定できる追加クローンオプションを示しています。

表61.2 昇格可能なクローンに利用できる追加のクローンオプション

フィールド	説明
promoted-max	昇格できるリソースのコピー数。デフォルト値は1です。
promoted-node-max	1つのノードで昇格できるリソースのコピー数。デフォルト値は1です。

61.3.2. 昇格可能なリソース制約の表示

ほとんどの場合、昇格可能なリソースには、アクティブなクラスターノードごとに1つのコピーがあります。そうではない場合は、リソースの場所制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。これらの制約は、通常のリソースと同様に記述されます。

リソースのロールをマスターにするかスレーブにするかを指定するコロケーション制約を作成できます。次のコマンドは、リソースのコロケーション制約を作成します。

> pcs constraint colocation add [master|slave] **source_resource** with [master|slave] **target_resource** [**score**] [**options**]

コロケーションの制約の詳細は、クラスターリソースのコロケーション を参照してください。

昇格可能なリソースが含まれる順序制約を設定する場合に、リソースに指定できるアクションに、リソースのスレーブロールからマスターへのロールの昇格を指定する **promote** があります。また、**demote** を指定すると、マスターからスレーブにリソースを降格できます。

順序制約を設定するコマンドは次のようになります。

> pcs constraint order [**action**] **resource_id** then [**action**] **resource_id** [**options**]

リソースの順序制約の詳細は、クラスターリソースの実行順序の決定 を参照してください。

61.4. 障害時の昇格リソースの降格

RHEL 8.3 では、昇格可能なリソースを設定できます。そのため、そのリソースの **昇格** または **監視** アクションが失敗した場合、またはリソースのクォーラムが失われると、リソースは降格されますが、完全に停止されることはありません。これにより、リソースを完全に停止したときに、手動で介入する必要がなくなります。

- 昇格可能なリソースを **promote** アクションが失敗したときに降格するように設定するには、以下の例のように **on-fail** 操作メタオプションを **demote** に設定します。

```
# pcs resource op add my-rsc promote on-fail="demote"
```

- **monitor** アクションが失敗したときに昇格可能なリソースを降格するように設定するには、**interval** をゼロ以外の値に設定し、**on-fail** 操作メタオプションを **demote** に設定して、**ロール** を **Master** に設定します。

```
# pcs resource op add my-rsc monitor interval="10s" on-fail="demote" role="Master"
```

- クラスターパーティションでクォーラムが失われると、昇格されたリソースが降格されますが、実行され続け、他のすべてのリソースが停止されるようにクラスターを設定するには、**no-quorum-policy** クラスタープロパティーを **demote** に設定します。

操作の **on-fail** メタ属性を **demote** に設定しても、リソースの昇格を決定する方法には影響しません。影響を受けるノードのプロモーションスコアが引き続き最高となっている場合は、再度昇格するように選択されます。

第62章 クラスターノードの管理

クラスターサービスの起動や停止、クラスターノードの追加や削除など、クラスターノードの管理に使用できる、さまざまな **pcs** コマンドがあります。

62.1. クラスターサービスの停止

次のコマンドで、指定ノード (複数指定可) のクラスターサービスを停止します。**pcs cluster start** と同様に **--all** オプションを使うと全ノードのクラスターサービスが停止されます。ノードを指定しない場合はローカルノードのクラスターサービスのみが停止されます。

```
pcs cluster stop [--all | node] [...]
```

次のコマンドで、ローカルノードのクラスターサービスを強制的に停止できます。このコマンドは、**kill -9** コマンドを実行します。

```
pcs cluster kill
```

62.2. クラスターサービスの有効化および無効化

次のコマンドを使用して、クラスターサービスを有効にします。これにより、指定した 1 つ以上のノードで起動時にクラスターサービスが実行されるように設定されます。

ノードがフェンスされた後にクラスターに自動的に再参加するようになり、クラスターが最大強度を下回る時間が最小限に抑えられます。クラスターサービスを有効にしていないと、クラスターサービスを手動で開始する前に、管理者が問題を調査できます。これにより、たとえばハードウェアに問題があるノードで再度問題が発生する可能性がある場合は、クラスターに戻さないようにできます。

- **--all** オプションを使用すると、全ノードでクラスターサービスが有効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが有効になります。

```
pcs cluster enable [--all | node] [...]
```

指定した 1 つまたは複数のノードの起動時に、クラスターサービスが実行されないよう設定する場合は、次のコマンドを使用します。

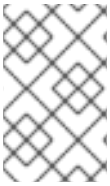
- **--all** オプションを指定すると、全ノードでクラスターサービスが無効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが無効になります。

```
pcs cluster disable [--all | node] [...]
```

62.3. クラスターノードの追加

次の手順で既存のクラスターに新しいノードを追加します。

この手順は、**corosync** を実行している標準クラスターを追加します。corosync 以外のノードをクラスターに統合する方法は corosync 以外のノードのクラスターへの統合: pacemaker_remote サービス を参照してください。



注記

運用保守期間中に、既存のクラスターにノードを追加することが推奨されます。これにより、新しいノードとそのフェンシング設定に対して、適切なリソースとデプロイメントのテストを実行できます。

この例では、**clusternode-01.example.com**、**clusternode-02.example.com**、および **clusternode-03.example.com** が既存のクラスターノードになります。新たに追加するノードは **newnode.example.com** になります。

手順

クラスターに追加する新しいノードで、以下の作業を行います。

1. クラスターパッケージをインストールします。クラスターで SBD、Booth チケットマネージャー、またはクォーラムデバイスを使用する場合は、対応するパッケージ (**sbd**、**booth-site**、**corosync-qdevice**) を、新しいノードにも手動でインストールする必要があります。

```
[root@newnode ~]# yum install -y pcs fence-agents-all
```

   クラスターパッケージに加えて、既存のクラスターノードにインストールしたクラスターで実行しているすべてのサービスをインストールおよび設定する必要があります。たとえば、Red Hat の高可用性クラスターで Apache HTTP サーバーを実行している場合は、追加するノードにサーバーをインストールする必要があります。また、サーバーのステータスを確認する **wget** ツールも必要です。

2. **firewalld** デーモンを実行している場合は、次のコマンドを実行して、Red Hat High Availability Add-On で必要なポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

3. ユーザー ID **hacluster** のパスワードを設定します。クラスターの各ノードで、同じパスワードを使用することが推奨されます。

```
[root@newnode ~]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. 次のコマンドを実行して **pcsd** サービスを開始し、システムの起動時に **pcsd** が有効になるようにします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

既存クラスターのノードの 1 つで、以下の作業を行います。

1. 新しいクラスターノードで **hacluster** ユーザーを認証します。

```
[root@clusternode-01 ~]# pcs host auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

–

2. 新しいノードを既存のクラスターに追加します。さらに、このコマンドは **corosync.conf** クラスター設定ファイルをクラスターのすべてのノード (追加する新しいノードを含む) に対して同期します。

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
```

クラスターに追加する新しいノードで、以下の作業を行います。

1. 新しいノードで、クラスターサービスを開始して有効にします。

```
[root@newnode ~]# pcs cluster start
Starting Cluster...
[root@newnode ~]# pcs cluster enable
```

2. 新しいクラスターノードに対して、フェンシングデバイスを設定してテストします。

62.4. クラスターノードの削除

次のコマンドは、指定したノードをシャットダウンして、クラスター内のその他のすべてのノードで、クラスターの設定ファイル **corosync.conf** からそのノードを削除します。

```
pcs cluster node remove node
```

62.5. リンクが複数あるクラスターへのノードの追加

複数のリンクを持つクラスターにノードを追加する場合は、すべてのリンクにアドレスを指定する必要があります。

以下の例では、ノード **rh80-node3** をクラスターに追加し、1 番目のリンクに IP アドレス 192.168.122.203 を、2 番目のリンクに 192.168.123.203 を指定します。

```
# pcs cluster node add rh80-node3 addr=192.168.122.203 addr=192.168.123.203
```

62.6. 既存のクラスターへのリンクの追加および修正

RHEL 8.1 では、ほとんどの場合は、クラスターを再起動することなく、既存のクラスターのリンクを追加または変更できます。

62.6.1. 既存クラスターへのリンクの追加および削除

実行中のクラスターに新しいリンクを追加するには、**pcs cluster link add** コマンドを使用します。

- リンクの追加時に、各ノードのアドレスを指定する必要があります。
- リンクの追加および削除は、**knet** トランスポートプロトコルを使用している場合に限り可能です。
- クラスター内で常に 1 つはリンクを定義する必要があります。
- クラスター内のリンクの最大数は 8 で、指定番号は 0-7 です。3、6、7 のみを指定するなど、リンクはどれでも定義できます。

- リンク番号を指定せずにリンクを追加すると、**pcs** は利用可能なリンクで番号が一番小さいものを使用します。
- 現在設定されているリンクのリンク番号は、**corosync.conf** ファイルに含まれます。**corosync.conf** ファイルを表示するには、**pcs cluster corosync** コマンドまたは **pcs cluster config show** コマンド (RHEL 8.4 以降の場合) を実行します。

以下のコマンドは、リンク番号 5 を 3 つのノードクラスターに追加します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12 node3=10.0.5.31
options linknumber=5
```

既存のリンクを削除するには、**pcs cluster link delete** コマンドまたは **pcs cluster link remove** コマンドを使用します。以下のコマンドのいずれかを実行すると、クラスターからリンク番号 5 が削除されます。

```
[root@node1 ~] # pcs cluster link delete 5

[root@node1 ~] # pcs cluster link remove 5
```

62.6.2. リンクが複数あるクラスター内のリンクの変更

クラスターに複数のリンクがあり、そのいずれかを変更する場合は、以下の手順を実行します。

手順

1. 変更するリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 2
```

2. アドレスとオプションを更新して、クラスターにリンクを追加し直します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12
node3=10.0.5.31 options linknumber=2
```

62.6.3. 単一リンクを使用したクラスターのリンクアドレスの変更

クラスターで 1 つのみリンクを使用し、別のアドレスを使用するようにリンクを変更する必要がある場合は、以下の手順を実行します。この例では、元のリンクはリンク 1 です。

1. 新しいアドレスおよびオプションを指定して新規リンクを追加します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12
node3=10.0.5.31 options linknumber=2
```

2. 元のリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 1
```

クラスターへのリンクの追加時に、現在使用中のアドレスは指定できないことに注意してください。たとえば、リンクが 1 つある 2 ノードクラスターがあり、ノード 1 つだけでアドレスを変更する場合に上記の手順を使用して、新規アドレスと既存のアドレスを指定するリンクを新たに追加できません。代わ

りに、以下の例のように、既存のリンクを削除し、アドレスを更新したリンクを追加しなおすことができます。

この例では、以下のように設定されています。

- 既存クラスターのリンクはリンク 1 で、ノード 1 に 10.0.5.11 のアドレスを使用し、ノード 2 に 10.0.5.12 アドレスを使用します。
- ノード 2 のアドレスを 10.0.5.31 に変更します。

手順

リンクが 1 つである 2 ノードクラスターのアドレスのいずれかのみを更新するには、以下の手順に従います。

1. 現在使用されていないアドレスを使用して、既存のクラスターに新しい一時的なリンクを追加します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.13 node2=10.0.5.14 options linknumber=2
```

2. 元のリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 1
```

3. 変更後の新しいリンクを追加します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.31 options linknumber=1
```

4. 作成した一時的なリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 2
```

62.6.4. リンクが 1 つのクラスター内のリンクオプションの変更

クラスターで使用されているリンクが 1 つのみで、そのリンクのオプションを変更しつつも、使用するアドレスを変更しない場合には、一時的なリンクを追加してからリンクを削除し、リンクを別のものに更新できます。

この例では、以下のように設定されています。

- 既存クラスターのリンクはリンク 1 で、ノード 1 に 10.0.5.11 のアドレスを使用し、ノード 2 に 10.0.5.12 アドレスを使用します。
- リンクオプション **link_priority** を 11 に変更します。

手順

次の手順で、1 つのリンクを持つクラスターでリンクオプションを変更します。

1. 現在使用されていないアドレスを使用して、既存のクラスターに新しい一時的なリンクを追加します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.13 node2=10.0.5.14 options
linknumber=2
```

2. 元のリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 1
```

3. 元のリンクのオプションを更新して追加し直します。

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12 options
linknumber=1 link_priority=11
```

4. 一時的なリンクを削除します。

```
[root@node1 ~] # pcs cluster link remove 2
```

62.6.5. 新しいリンクの追加時にリンクの変更はできません。

設定で新しいリンクを追加することができない場合や、既存のリンクを 1 つ変更することが唯一のオプションである場合は、以下の手順を使用します。これにより、クラスターをシャットダウンする必要があります。

手順

以下の例では、クラスター内のリンク番号 1 を更新し、リンクの **link_priority** オプションを 11 に設定します。

1. クラスターのクラスターサービスを停止します。

```
[root@node1 ~] # pcs cluster stop --all
```

2. リンクアドレスとオプションを更新します。
pcs cluster link update コマンドでは、すべてのノードアドレスとオプションを指定する必要はありません。代わりに、変更するアドレスのみを指定できます。この例では、**node1** および **node3** のアドレスを変更し、**link_priority** オプションのみを変更します。

```
[root@node1 ~] # pcs cluster link update 1 node1=10.0.5.11 node3=10.0.5.31 options
link_priority=11
```

オプションを削除するには、 **option=** 形式で Null 値にオプションを設定します。

3. クラスターを再起動します。

```
[root@node1 ~] # pcs cluster start --all
```

62.7. ノードのヘルスストラテジーの設定

ノードは、そのクラスターメンバーシップを維持するためには十分に機能していても、別の側面では正常に機能しておらず、リソースにとって適切ではないロケーションになることがあります。たとえば、ディスクドライブが SMART エラーを報告していたり、CPU の負荷が高くなっている場合などがそうです。RHEL 8.7 では、Pacemaker のノードヘルスストラテジーを使用して、自動的にリソースを正常でないノードから移動できます。

次のヘルスノードリソースエージェントを使用して、ノードのヘルスを監視できます。このエージェントは、CPU とディスクのステータスに基づいてノードの属性を設定します。

- **ocf:pacemaker:HealthCPU**: CPU のアイドリングを監視
- **ocf:pacemaker:HealthIOWait**: CPU I/O 待機を監視
- **ocf:pacemaker:HealthSMART**: ディスクドライブの SMART ステータスを監視
- **ocf:pacemaker:SysInfo**: ローカルシステム情報を使用してさまざまなノード属性を設定し、ディスク領域の使用状況を監視するヘルスエージェントとしても機能

さらに、すべてのリソースエージェントがヘルスノードストラテジーの定義に使用できるノード属性を提供する可能性があります。

手順

次の手順では、CPU I/O 待機が 15% を超えるノードからリソースを移動するクラスターのヘルスノードストラテジーを設定します。

1. **health-node-strategy** クラスタープロパティーを設定して、Pacemaker がノードヘルスの変化に応答する方法を定義します。

```
# pcs property set node-health-strategy=migrate-on-red
```

2. ヘルスノードリソースエージェントを使用するクラスターリソースのクローンを作成し、**allow-unhealthy-nodes** リソースメタオプションを設定して、ノードのヘルスが回復したかどうかをクラスターが検出してリソースをノードに戻すかどうかを定義します。すべてのノードのヘルスを継続的にチェックするには、定期的な監視アクションを使用してこのリソースを設定します。
この例では、**HealthIOWait** リソースエージェントを作成して CPU I/O 待機を監視し、ノードからリソースを移動するための制限を 15% に設定します。このコマンドは、**allow-unhealthy-nodes** リソースメタオプションを **true** に設定し、繰り返しの監視間隔を 10 秒に設定します。

```
# pcs resource create io-monitor ocf:pacemaker:HealthIOWait red_limit=15 op monitor interval=10s meta allow-unhealthy-nodes=true clone
```

62.8. 多数のリソースを使用した大規模なクラスターの設定

デプロイするクラスターにノードとリソースが多数含まれる場合に、クラスターの以下のパラメーターのデフォルト値を変更する必要がある場合があります。

cluster-ipc-limit クラスタープロパティー

cluster-ipc-limit クラスタープロパティーは、あるクラスターデーモンが別のクラスターデーモンを切断するまでに対応できる IPC メッセージバックログの最大数になります。多数のリソースを消去するか、大規模なクラスターで同時にリソースを変更すると、多数の CIB 更新が一度に行われます。これが原因で、CIB イベントキューのしきい値に到達するまでに Pacemaker サービスで全設定の更新を処理する時間がない場合には、低速なクライアントがエビクトされる可能性がありました。
大規模なクラスターで使用するための **cluster-ipc-limit** の推奨値は、クラスターのリソース数にノード数を乗算した値です。この値は、クラスターデーモン PID の Evicting client メッセージがログに表示されると増える可能性があります。

pcs property set コマンドを使用して、**cluster-ipc-limit** の値をデフォルト値 500 から増やすことができます。たとえば、リソースが 200 個ある 10 ノードクラスターの場合には、以下のコマンドを使用して **cluster-ipc-limit** の値を 2000 に設定できます。

```
# pcs property set cluster-ipc-limit=2000
```

PCMK_ipc_buffer Pacemaker パラメーター

非常に大規模なデプロイメントでは、内部 Pacemaker メッセージがメッセージバッファーのサイズを超える可能性があります。バッファーサイズを超えると、以下の形式のシステムログにメッセージが表示されます。

```
Compressed message exceeds X% of configured IPC limit (X bytes); consider setting
PCMK_ipc_buffer to X or higher
```

このメッセージが表示されると、各ノードの **/etc/sysconfig/pacemaker** 設定ファイルで **PCMK_ipc_buffer** の値を増やしてください。たとえば、**PCMK_ipc_buffer** の値をデフォルト値から 13396332 バイトに増やすには、以下のようにクラスター内の各ノードの **/etc/sysconfig/pacemaker** ファイルのアンコメントされている **PCMK_ipc_buffer** フィールドを変更します。

```
PCMK_ipc_buffer=13396332
```

この変更を適用するには、以下のコマンドを実行します。

```
# systemctl restart pacemaker
```

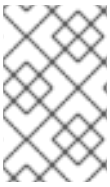
第63章 PACEMAKER クラスターのプロパティー

クラスターのプロパティーは、クラスター動作中に発生する可能性がある状況に直面した場合に、クラスターの動作を制御します。

63.1. クラスタープロパティーおよびオプションの要約

以下の表には、Pacemaker クラスターのプロパティーのデフォルト値や、設定可能な値などをまとめています。

フェンス動作を決定するクラスタープロパティーがあります。これらのプロパティーの詳細は、フェンスデバイスの一般的なプロパティー の表フェンスの動作を決定するクラスタープロパティーを参照してください。



注記

この表に記載しているプロパティー以外にも、クラスターソフトウェアで公開されるクラスタープロパティーがあります。このようなプロパティーでは、デフォルト値を別の値には変更しないことが推奨されます。

表63.1 クラスターのプロパティー

オプション	デフォルト	説明
batch-limit	0	クラスターを並列に実行できるリソースアクションの数。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。デフォルト値の 0 は、任意のノードで CPU の負荷が高い場合に動的に制限を課すことを意味します。
migration-limit	-1 (無制限)	クラスターが、ノードで並行に実行することが許可されている移行ジョブの数。

オプション	デフォルト	説明
no-quorum-policy	stop	クラスターにクォーラムがない場合のアクション。設定できる値は以下のとおりです。 * ignore - 全リソースの管理を続行する * freeze - リソース管理は継続するが、影響を受けるパーティションに含まれないノードのリソースは復帰させない * stop - 影響を受けるクラスターパーティション内の全リソースを停止する * suicide - 影響を受けるクラスターパーティション内の全ノードをフェンスする * demote - クラスターパーティションがクォーラムを失うと、プロモートされたリソースを降格し、その他のリソースを停止する
symmetric-cluster	true	リソースを、デフォルトで任意のノードで実行できるかどうかを示します。
cluster-delay	60s	(アクションの実行を除く) ネットワーク上のラウンドトリップ遅延です。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。
dc-deadtime	20s	起動時に他のノードからの応答を待つ時間。正しい値は、ネットワークの速度と負荷、および使用するスイッチの種類によって異なります。
stop-orphan-resources	true	削除されたリソースを停止すべきかどうかを示します。
stop-orphan-actions	true	削除されたアクションをキャンセルするかどうかを示します。

オプション	デフォルト	説明
start-failure-is-fatal	true	特定のノードでリソースの起動に失敗した場合に、そのノードで開始試行を行わないようにするかを示します。**false** に設定すると、クラスターは、同じノードで開始するかどうかを、リソースが失敗した回数と、移行しきい値により設定します。リソースに **migration-threshold** オプションを設定する方法は、リソースのメタオプションの設定 を参照してください。 **start-failure-is-fatal** を **false** に設定すると、リソースを起動できない障害があるノードが、すべての依存アクションを遅らせる可能性があるというリスクが発生します。この理由により、**start-failure-is-fatal** のデフォルトは true となっています。**start-failure-is-fatal=false** を設定するリスクは、移行しきい値を低く設定することで軽減できます。これにより、何度も失敗してもその他のアクションを続行できます。
pe-error-series-max	-1 (すべて)	ERROR となるスケジューラー入力を保存する数。問題を報告する場合に使用されます。
pe-warn-series-max	-1 (すべて)	WARNING となるスケジューラー入力を保存する数。問題を報告する場合に使用されます。
pe-input-series-max	-1 (すべて)	normal となるスケジューラー入力を保存する数。問題を報告する場合に使用されます。
cluster-infrastructure		Pacemaker が現在実行しているメッセージングスタック。情報提供および診断目的に使用されます。ユーザーは設定できません。
dc-version		クラスターの DC (Designated Controller) で Pacemaker のバージョン。診断目的に使用され、ユーザーは設定できません。

オプション	デフォルト	説明
cluster-recheck-interval	15 分	Pacemaker は基本的にイベント駆動型で、失敗によるタイムアウトやほとんどの時間ベースのルールについてクラスターを再確認するタイミングを予め把握します。Pacemaker は、このプロパティーで指定された非アクティブ期間の後にもクラスターを再確認します。このクラスターの再確認には 2 つの目的があります。**date-spec** のルールがこの頻繁で必ず確認されるようにすることと、一部のタイプのスケジューラーバグについてフェイルセーフとして機能することです。0 を値として指定すると、このポーリングが無効になります。正の値は時間間隔を示します。
maintenance-mode	false	メンテナンスモードでは、クラスターが干渉されないモードになり、指示されない限り、サービスを起動したり、停止したりしません。メンテナンスモードが完了すると、クラスターは、サービスの現在の状態のサニティーチェックを実行してから、これを必要とするサービスを停止するか、開始します。
shutdown-escalation	20min	正常にシャットダウンして終了を試みるのをやめる時間。高度な使用のみ。
stop-all-resources	false	クラスターがすべてのリソースを停止します。
enable-acl	false	クラスターが、**pcs acl** コマンドで設定したアクセス制御リストを使用できるかどうかを示します。
placement-strategy	default	クラスターノードでリソースの配置を決定する際に、クラスターが使用率属性を考慮にいれるかどうかと、どのように考慮するかを示します。
priority-fencing-delay	0 (無効)	(RHEL 8.3 以降) スプリットブレインが発生した場合に、リソースの実行数が最も少ないノードがフェンスされるように、2 ノードクラスターを設定できます。 **priority-fencing-delay** プロパティーでは、期間を設定できます。このプロパティーのデフォルト値は 0 (無効) です。このプロパティーがゼロ以外の値に設定されている場合や、**priority** メタ属性が 1 つ以上のリソースに対して設定されている場合は、スプリットブレインが発生すると、実

オプション	デフォルト	説明
		行されているすべてのリソースの中で、最も優先順位が高い組み合わせのノードが存続する可能性が高くなります。 たとえば、 **pcs resource defaults priority=1** と **pcs property set priority-fencing-delay=15s** を設定し、他の優先度が設定されていない場合には、他のノードはフェンシングを開始するまで 15 秒間待機するため、最も多くのリソースを実行するノードが存続する可能性が高くなります。特定のリソースが他のリソースよりも重要である場合は、優先度を高く設定できます。 プロモート可能なクローンに優先順位が設定されている場合には、そのクローンのマスターロールを実行しているノードの優先度が 1 ポイント追加されます。 **priority-fencing-delay** プロパティーで遅延を設定すると、**pcmk_delay_base** と **pcmk_delay_max** のフェンスデバイスプロパティーから遅延に追加されます。この動作により、両方のノードの優先度が同等の場合、またはノードの損失以外の理由で両方ノノードをフェンシングする必要がある場合 (例: **on-fail=fencing** がリソースモニター操作用に設定されている)、ある程度の遅延を許容します。組み合わせて使用する場合には、優先ノードが優先されるよう、**priority-fencing-delay** プロパティーを、**pcmk_delay_base** および **pcmk_delay_max** の最大遅延よりもはるかに大きい値に設定することを推奨します (値を 2 倍すると完全に安全となります)。 Pacemaker 自体がスケジュールしたフェンシングのみが **priority-fencing-delay** を監視します。**dlm_controld** などの外部コードでスケジュールされるフェンシングは、フェンスデバイスに必要な情報を提供しません。

オプション	デフォルト	説明
node-health-strategy	none	ヘルスリソースエージェントと組み合わせて使用し、Pacemaker がノードヘルスの変化にどのように応答するかを制御します。設定できる値は以下のとおりです。 * **none** - ノードヘルスを追跡しません。 * **migrate-on-red** - リソースは、エージェントが監視するローカルの状態に基づき、ノードのステータスが **red** であるとヘルスエージェントが判断したノードから移動されます。 * **only-green** - リソースは、エージェントが監視するローカルの状態に基づき、ノードのステータスが **yellow** または **red** であるとヘルスエージェントが判断したノードから移動されます。 * **progressive**, **custom** - ヘルス属性の内部数値に従って、ヘルス状態に対するクラスターの応答をよりきめ細かく制御できる高度なノードヘルスストラテジー。

63.2. クラスターのプロパティーの設定と削除

クラスタープロパティーの値を設定するには、次の pcs コマンドを使用します。

```
pcs property set property=value
```

たとえば、**symmetric-cluster** の値を **false** に設定する場合は、次のコマンドを使用します。

```
# pcs property set symmetric-cluster=false
```

設定からクラスタープロパティーを削除する場合は、次のコマンドを使用します。

```
pcs property unset property
```

代わりに **pcs property set** コマンドの値フィールドを空白にしてもクラスタープロパティーを削除することができます。これにより、そのプロパティーの値がデフォルト値に戻されます。たとえば、**symmetric-cluster** プロパティーを **false** に設定したことがある場合は、設定した値が次のコマンドにより削除され、**symmetric-cluster** の値がデフォルト値の **true** に戻されます。

```
# pcs property set symmetic-cluster=
```

63.3. クラスタープロパティー設定のクエリー

ほとんどの場合は、各種のクラスターコンポーネントの値を表示するのに **pcs** コマンドを使用する場合に、**pcs list** または **pcs show** を同じように使用できます。次の例では、**pcs list** は、複数のプロパ

ティーの設定を完全に表示するのに使用される形式で、**pcs show** は、特定のプロパティーの値を表示する場合に使用される形式です。

クラスターに設定されたプロパティー設定の値を表示する場合は、次の pcs コマンドを使用します。

```
pcs property list
```

明示的に設定されていないプロパティー設定のデフォルト値など、クラスターのプロパティー設定の値をすべて表示する場合は、次のコマンドを使用します。

```
pcs property list --all
```

特定のクラスタープロパティーの現在の値を表示する場合は、次のコマンドを使用します。

```
pcs property show property
```

たとえば、**cluster-infrastructure** プロパティーの現在の値を表示する場合は、次のコマンドを実行します。

```
# pcs property show cluster-infrastructure
Cluster Properties:
 cluster-infrastructure: cman
```

情報提供の目的で、次のコマンドを使用して、プロパティーがデフォルト以外の値に設定されているかどうかに関わらず、プロパティーのデフォルト値のリストを表示できます。

```
pcs property [list|show] --defaults
```

第64章 仮想ドメインをリソースとして設定

pcs resource create コマンドを使用して、**VirtualDomain** をリソースタイプとして指定すると、**libvirt** 仮想化フレームワークが管理する仮想ドメインを、クラスターリソースとして設定できます。

仮想ドメインをリソースとして設定する場合は、以下の点を考慮してください。

- 仮想ドメインは、クラスターリソースとして設定する前に停止する必要があります。
- 仮想ドメインをクラスターリソースにすると、クラスターツールを使用しない限り、起動、停止、または移行を行うことができません。
- クラスターリソースとして設定した仮想ドメインを、ホストの起動時に起動するように設定することはできません。
- 仮想ドメインの実行を許可するすべてのノードは、その仮想ドメインに必要な設定ファイルおよびストレージデバイスにアクセスできるようにする必要があります。

クラスターが仮想ドメイン内のサービスを管理できるようにする場合は、仮想ドメインをゲストノードとして設定できます。

64.1. 仮想ドメインリソースのオプション

以下の表は、**VirtualDomain** リソースに設定できるリソースオプションを説明します。

表64.1 仮想ドメインリソースのリソースオプション

フィールド	デフォルト	説明
config		(必須) この仮想ドメインの **libvirt** 設定ファイルへの絶対パス。
hypervisor	システムに依存	接続先のハイパーバイザーの URI。**virsh --quiet uri** コマンドを実行して、システムのデフォルト URI を確認できます。
force_stop	**0**	停止時にドメインを常に強制的にシャットダウン (破棄) します。デフォルト動作では、正常なシャットダウンの試行に失敗した後でのみ、強制シャットダウンを実行します。仮想ドメイン (または仮想化バックエンド) が正常なシャットダウンに対応していない場合に限り、これを **true** に設定する必要があります。

フィールド	デフォルト	説明
migration_transport	システムに依存	移行中にリモートハイパーバイザーに接続するのに使用されるトランスポート。このパラメーターを省略すると、リソースは **libvirt** のデフォルトトランスポートを使用して、リモートハイパーバイザーに接続します。
migration_network_suffix		専用の移行ネットワークを使用します。移行 URI は、このパラメーターの値をノード名の末尾に追加することで設定されます。ノード名が完全修飾ドメイン名 (FQDN) の場合は、FQDN の最初のピリオド (.) の直前に接尾辞を挿入します。この設定されたホスト名がローカルで解決可能であり、関連する IP アドレスが優先ネットワークを介して到達可能であることを確認してください。
monitor_scripts		仮想ドメイン内でサービスの監視を追加で実行する場合は、監視するスクリプトのリストとともに、このパラメーターを追加します。**備考**:監視スクリプトを使用する場合、**start** および **migrate_from** の操作は、すべての監視スクリプトが正常に完了した場合にのみ完了します。この遅延に対応できるように、この操作のタイムアウトを必ず設定してください。
autoset_utilization_cpu	**true**	**true** に設定すると、監視の実行時に、エージェントが **virsh** から **domainU** の **vCPU** 数を検出し、これをリソースの CPU 使用率に組み込みます。
autoset_utilization_hv_memory	**true**	true に設定すると、監視の実行時に、エージェントが **virsh** から **Max memor** 数を検出し、これをリソースの **hv_memory** の使用率に組み込みます。
migrateport	ランダムハイポート	このポートは、**qemu** 移行 URI で使用されます。これを設定しないと、ポートにはランダムハイポートが使用されます。

フィールド	デフォルト	説明
snapshot		仮想マシンイメージが保存されるスナップショットディレクトリーへのパス。このパラメーターが設定されていると、仮想マシンの RAM 状態は、停止時にスナップショットディレクトリーのファイルに保存されます。起動時にドメインのステータスファイルが存在すると、ドメインは、最後に停止する直前の状態に復元されます。このオプションは、**force_stop** オプションと互換性がありません。

VirtualDomain リソースオプションに加えて、**allow-migrate** メタデータオプションを設定して、リソースの別のノードへのライブ移行を許可できます。このオプションを **true** に設定すると、状態を失うことなくリソースを移行できます。このオプションがデフォルトの状態である **false** に設定されていると、仮想ドメインは、ノード間で移行される際に、最初のノードでシャットダウンしてから、2 番目のノードで再起動します。

64.2. 仮想ドメインリソースの作成

以下の手順では、以前に作成した仮想マシンのクラスターに **VirtualDomain** リソースを作成します。

手順

1. 仮想マシンを管理するために **VirtualDomain** リソースエージェントを作成する場合、Pacemaker では、ディスクのファイルに、仮想マシンの **xml** 設定ファイルをダンプする必要があります。たとえば、**guest1** という名前の仮想マシンを作成した場合は、ゲストを実行できるクラスターノードのいずれかにファイルに **xml** ファイルをダンプします。任意のファイル名を使用できますが、この例では **/etc/pacemaker/guest1.xml** を使用します。

   ```
   # virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
   ```

2. 仮想マシンの **xml** 設定ファイルを、各ノードの同じ場所にあるゲストを実行できるその他のすべてのクラスターノードにコピーします。

3. 仮想ドメインの実行が許可されているすべてのノードが、その仮想ドメインに必要なストレージデバイスにアクセスできるようにします。

4. 仮想ドメインが、仮想ドメインを実行する各ノードで起動および停止できることを別途テストします。

5. ゲストノードが実行している場合はシャットダウンします。Pacemaker は、クラスターで設定されているノードを起動します。仮想マシンは、ホストの起動時に自動的に起動するように設定することはできません。

6. **pcs resource create** コマンドを使用して、**VirtualDoman** リソースを設定します。たとえば、次のコマンドは、**VM** という名前の **VirtualDomain** リソースを設定します。**allow-migrate** オプションは **true** に設定されるため、**pcs resource move VM nodeX** コマンドはライブ移行として実行されます。
   この例では、**migration_transport** が **ssh** に設定されます。SSH 移行が適切に機能するには、鍵を使用しないロギングがノード間で機能する必要があります。

```
# pcs resource create VM VirtualDomain config=/etc/pacemaker/guest1.xml
migration_transport=ssh meta allow-migrate=true
```

第65章 クラスタークォーラムの設定

Red Hat High Availability Add-On クラスターは、スプリットブレインの状況を回避するために、**votequorum** サービスをフェンシングと併用します。クラスターの各システムには多くの投票数が割り当てられ、過半数の票を取得しているものだけがクラスターの操作を継続できます。サービスは、すべてのノードに読み込むか、いずれのノードにも読み込まないようにする必要があります。サービスをクラスターノードのサブセットに読み込むと、結果が予想できなくなります。**votequorum** サービスの設定および操作の詳細は、man ページの **votequorum**(5) を参照してください。

65.1. クォーラムオプションの設定

pcs cluster setup コマンドを使用してクラスターを作成する場合は、クォーラム設定の特殊な機能を使用できます。以下の表には、これらのオプションをまとめています。

表65.1 クォーラムオプション

オプション	説明
auto_tie_breaker	これを有効にすると、クラスターは、決定論的に最大 50% のノードが同時に失敗しても存続されます。クラスターパーティションや、**auto_tie_breaker_node** に設定された **nodeid** (設定されていない場合は最小の **nodeid**) と通信したままのノードのセットは、クォーラムに達した状態を維持します。その他のノードはクォーラムに達しません。 **auto_tie_breaker** オプションを指定すると、均等の分割でクラスターが動作を継続できるようになるため、主に偶数個のノードがあるクラスターで使用されます。複数で不均等の分割など、より複雑な障害は、クォーラムデバイスを使用することが推奨されます。 **auto_tie_breaker** オプションは、クォーラムデバイスと互換性がありません。
wait_for_all	有効にすると、最低 1 回、同時にすべてのノードが現れた後に、初回だけ、クラスターがクォーラムに達します。 **wait_for_all** オプションは、主にクォーラムデバイス **lms** (last man standing) アルゴリズムを使用する 2 ノードクラスター、および偶数のノードで設定されるクラスターに使用されます。 **wait_for_all** オプションは、クラスターに 2 つのノードがあり、クォーラムデバイスを使用せず、**auto_tie_breaker** が無効になっている場合に自動的に有効になります。**wait_for_all** を明示的に 0 に設定すると、このオプションをオーバーライドできます。
last_man_standing	有効にすると、クラスターは特定の状況で **expected_votes** とクォーラムを動的に再計算します。このオプションを有効にする場合は、**wait_for_all** を有効にする必要があります。**last_man_standing** オプションには、クォーラムデバイスとの互換性がありません。

オプション	説明
last_man_standing_window	クラスターのノードが失われた後の、**expected_votes** およびクォーラムを再計算するまでの待ち時間 (ミリ秒単位) です。

このオプションの設定および使用の詳細は、man ページの **votequorum**(5) を参照してください。

65.2. クォーラムオプションの変更

pcs quorum update コマンドを使用して、クラスターの一般的なクォーラムオプションを変更できます。稼働中のシステムでは、**quorum.two_node** および **quorum.expected_votes** オプションを変更できます。その他すべてのクォーラムオプションについては、このコマンドを実行するには、クラスターを停止する必要があります。クォーラムオプションの詳細は **votequorum**(5) の man ページを参照してください。

pcs quorum update コマンドの形式は次のとおりです。

```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=
[time-in-ms] [wait_for_all=[0|1]]
```

以下の一連のコマンドは、**wait_for_all** クォーラムオプションを変更し、このオプションの更新された状態を表示します。クラスターの稼働中はこのコマンドを実行できないことに注意してください。

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
Error: node1: corosync is running
Error: node2: corosync is running

[root@node1:~]# pcs cluster stop --all
node2: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
node2: Stopping Cluster (corosync)...

[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
node2: corosync is not running
node1: corosync is not running
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded

[root@node1:~]# pcs quorum config
Options:
  wait_for_all: 1
```

65.3. クォーラム設定およびステータスの表示

クラスターを実行したら、以下のクラスタークォーラムコマンドを実行して、クォーラムの設定やステータスを表示できます。

次のコマンドは、クォーラムの設定を表示します。

```
pcs quorum [config]
```

次のコマンドは、クォーラムのランタイム状態を表示します。

```
pcs quorum status
```

65.4. クォーラムに達しないクラスターの実行

長時間クラスターでノードを使用しなかったためにクォーラムが失われた場合に、**pcs quorum expected-votes** コマンドを使用して、ライブクラスターの **expected_votes** パラメーターの値を変更できます。これにより、クォーラムがない場合でも、クラスターは操作を継続できます。

警告

ライブクラスターで期待される票数 (vote) を変更する場合は、細心の注意を払って行ってください。期待される票数を手動で変更したために、実行しているクラスターが 50% 未満となる場合は、クラスターの他のノードを個別に起動してクラスターサービスを開始できるため、データの破損や予期せぬ結果が発生することがあります。この値を変更する場合は、**wait_for_all** パラメーターが有効になっていることを確認してください。

次のコマンドは、ライブクラスターで期待される票数を、指定の値に設定します。これはライブクラスターにのみ影響し、設定ファイルは変更されません。リロードが行われると、**expected_votes** の値は、設定ファイルの値にリセットされます。

```
pcs quorum expected-votes votes
```

クォーラムに達していない状態でクラスターにリソース管理を続行させたい場合は、**pcs quorum unblock** コマンドを使用して、クォーラムの確立時にクラスターがすべてのノードを待機することのないようにします。

注記

このコマンドは細心の注意を払って使用する必要があります。このコマンドを実行する前に、現在クラスターにないノードの電源を切り、共有リソースにアクセスできない状態であることを確認する必要があります。

```
# pcs quorum unblock
```

第66章 COROSYNC 以外のノードのクラスターへの統合: PACEMAKER_REMOTE サービス

pacemaker_remote サービスを使用すると、**corosync** を実行していないノードをクラスターに統合し、そのリソースが実際のクラスターノードであるかのように、クラスターがリソースを管理できます。

pacemaker_remote サービスが提供する機能には以下が含まれます。

- **pacemaker_remote** サービスを使用すると、RHEL 8.1 の 32 ノードのサポート制限を超えた拡張が可能になります。
- **pacemaker_remote** サービスを使用すると、仮想環境をクラスターリソースとして管理でき、さらに仮想環境内の個別のサービスをクラスターリソースとして管理できます。

pacemaker_remote サービスは、以下の用語を使用して記述されます。

- **クラスターノード** - 高可用性サービスを実行しているノード (**pacemaker** および **corosync**)。
- **リモートノード** - **pacemaker_remote** を実行して、**corosync** クラスターメンバーシップを必要としないクラスターにリモートで統合するノード。リモートノードは、**ocf:pacemaker:remote** リソースエージェントを使用するクラスターリソースとして設定されます。
- **ゲストノード** - **pacemaker_remote** サービスを実行する仮想ゲストノード。仮想ゲストリソースはクラスターにより管理されます。クラスターにより起動し、リモートノードとしてクラスターに統合されます。
- pacemaker_remote - Pacemaker クラスター環境のリモートノードおよび KVM ゲストノードでリモートアプリケーション管理を実行できるサービスデーモン。このサービスは、corosync を実行していないノードでリソースをリモートで管理できる Pacemaker のローカル実行プログラムデーモン (**pacemaker-execd**) の拡張バージョンです。

pacemaker_remote サービスを実行している Pacemaker クラスターには次のような特徴があります。

- リモートノードおよびゲストノードは、**pacemaker_remote** サービスを実行します (仮想マシン側で必要な設定はほとんどありません)。
- クラスターノードで実行しているクラススタック (**pacemaker** および **corosync**) はリモートノードで **pacemaker_remote** サービスに接続するため、クラスターに統合できます。
- クラスターノードで実行しているクラススタック (**pacemaker** および **corosync**) はゲストノードを開始し、ゲストノードで **pacemaker_remote** サービスに即座に接続するため、クラスターに統合できます。

クラスターノードと、クラスターノードが管理するリモートおよびゲストノードの主な違いは、リモートおよびゲストノードはクラススタックを実行しないことです。そのため、リモートおよびゲストノードには以下の制限があります。

- クォーラムでは実行されない
- フェンスデバイスの動作を実行しない
- クラスターの指定コントローラー (DC) として機能できない
- **pcs** コマンドは一部しか実行できない

その一方で、リモートノードおよびゲストノードは、クラススタックに関連するスケーラビリティーの制限に拘束されません。

このような制限事項以外に、リモートノードとゲストノードは、リソース管理に関してクラスターノードと同様に動作し、リモートノードとゲストノード自体をフェンスすることができます。クラスターは、各リモートノードおよびゲストノードでリソースを管理および監視する機能を完全に備えています。制約を構築したり、スタンバイ状態にしたり、**pcs** コマンドを使用してクラスターノードで実行するその他の操作を実行したりできます。リモートノードおよびゲストノードは、クラスターノードと同様にクラスターステータスの出力に表示されます。

66.1. PACEMAKER_REMOTE ノードのホストおよびゲストの認証

クラスターノードと pacemaker_remote の間の接続には、TLS (Transport Layer Security) が使用され、PSK (Pre-Shared Key) の暗号化と TCP 上の認証 (デフォルトで 3121 ポートを使用) でセキュア化されます。そのため、クラスターノードと、**pacemaker_remote** を実行しているノードは、同じ秘密鍵を共有する必要があります。デフォルトでは、クラスターノードとリモートノードの両方でこのキーを **/etc/pacemaker/authkey** に配置する必要があります。

pcs cluster node add-guest コマンドは、ゲストノードに **authkey** を設定し、**pcs cluster node add-remote** コマンドは、リモートノードに **authkey** を設定します。

66.2. KVM ゲストノードの設定

Pacemaker ゲストノードは、**pacemaker_remote** サービスを実行する仮想ゲストノードです。仮想ゲストノードはクラスターにより管理されます。

66.2.1. ゲストノードリソースのオプション

ゲストノードとして動作するように仮想マシンを設定する場合は、仮想マシンを管理する **VirtualDomain** リソースを作成します。**VirtualDomain** リソースに設定できるオプションの説明は、仮想ドメインリソースのオプション の表仮想ドメインリソースのリソースオプションを参照してください。

VirtualDomain リソースオプションのほかにも、メタデータオプションはリソースをゲストノードとして定義し、接続パラメーターを定義します。**pcs cluster node add-guest** コマンドを使用して、これらのリソースオプションを設定します。以下の表は、これらのメタデータオプションについて説明しています。

表66.1 KVM リソースをリモートノードとして設定するためのメタデータオプション

フィールド	デフォルト	説明
remote-node	<none>	このリソースが定義するゲストノードの名前。リソースをゲストノードとして有効にし、ゲストノードの識別に使用される一意名を定義します。**警告**:この値を、リソースやノードの ID と重複させることはできません。
remote-port	3121	**pacemaker_remote** へのゲスト接続に使用するカスタムのポートを設定します。

フィールド	デフォルト	説明
remote-addr	**pcs host auth** コマンドで指定されるアドレス	接続先の IP アドレスまたはホスト名
remote-connect-timeout	60s	保留中のゲスト接続がタイムアウトするまでの時間

66.2.2. 仮想マシンのゲストノードとしての統合

以下の手順では、**libvirt** と KVM 仮想ゲストを使用して、Pacemaker で仮想マシンを起動し、そのマシンをゲストノードとして統合する手順の概要を説明します。

手順

1. **VirtualDomain** リソースを設定します。
2. すべての仮想マシンで次のコマンドを実行し、**pacemaker_remote** パッケージをインストールし、**pcsd** サービスを起動し、これを起動時に実行できるようにし、ファイアウォールを介して、TCP の 3121 ポートを許可します。

```
# yum install pacemaker-remote resource-agents pcs
# systemctl start pcsd.service
# systemctl enable pcsd.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

3. 各仮想マシンに、すべてのノードが認識できる静的ネットワークアドレスと一意なホスト名を割り当てます。
4. ゲストノードとして統合しようとしているノードに **pcs** を認証していない場合は認証します。

```
# pcs host auth nodename
```

5. 次のコマンドを使用して、既存の **VirtualDomain** リソースをゲストノードに変換します。このコマンドは、追加するゲストノードではなく、クラスターノードで実行する必要があります。リソースを変換する以外にも、このコマンドは **/etc/pacemaker/authkey** をゲストノードにコピーし、ゲストノードで **pacemaker_remote** デーモンを起動して有効にします。任意に定義できるゲストノードのノード名は、ノードのホスト名とは異なる場合があります。

```
# pcs cluster node add-guest nodename resource_id [options]
```

6. **VirtualDomain** リソースの作成後は、クラスターの他のノードと同じように、ゲストノードを扱うことができます。たとえば、クラスターノードから実行される次のコマンドのように、リソースを作成し、リソースにリソース制約を設定してゲストノードで実行できます。ゲストノードはグループに追加できます。これにより、ストレージデバイス、ファイルシステム、および仮想マシンをグループ化できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers nodename
```

66.3. PACEMAKER リモートノードの設定

リモートノードは、**ocf:pacemaker:remote** がリソースエージェントとして指定された状態で、クラスターリソースとして定義されます。**pcs cluster node add-remote** コマンドを使用してこのリソースを作成します。

66.3.1. リモートノードリソースのオプション

以下の表は、**remote** リソースに設定できるリソースオプションを示しています。

表66.2 リモートノードのリソースオプション

フィールド	デフォルト	説明
reconnect_interval	0	リモートノードへのアクティブな接続が切断された後、リモートノードへの再接続を試みる前に待機する時間 (秒単位)。この待機期間は繰り返し発生します。待機期間の後に再接続に失敗した場合、待機期間の後に、新しい再接続が試行されます。このオプションが使用されると、Pacemaker は待機期間の後に無限にリモートノードへ接続を試みます。
server	**pcs host auth** コマンドで指定されるアドレス	接続するサーバーの場所。IP アドレスまたはホスト名を指定できます。
port		接続する TCP ポート。

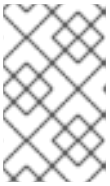
66.3.2. リモートノードの設定の概要

以下のセクションでは、Pacemaker リモートノードを設定し、そのノードを既存の Pacemaker クラスター環境に統合する手順の概要を説明します。

手順

1. リモートノードを設定するノードで、ローカルファイアウォールを介してクラスター関連のサービスを許可します。

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```



注記

iptables を直接使用する場合や、**firewalld** 以外のファイアウォールソリューションを使用する場合は、以下のポートを開きます。TCP ポート 2224 および 3121

2. リモートノードに、**pacemaker_remote** デーモンをインストールします。

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. リモートノードで、**pcsd** を開始し、有効にします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

4. リモートノードとして追加するノードに **pcs** を認証していない場合は、認証します。

```
# pcs host auth remote1
```

5. 以下のコマンドを使用して、リモートノードリソースをクラスターに追加します。このコマンドは、関連するすべての設定ファイルを新規ノードに追加し、ノードを起動し、これをシステムの起動時に **pacemaker_remote** を開始するように設定することもできます。このコマンドは、追加するリモートノードではなく、クラスターノードで実行する必要があります。

```
# pcs cluster node add-remote remote1
```

6. **remote** リソースをクラスターに追加した後、リモートノードを、クラスター内の他のノードを処理するのと同じように処理できます。たとえば、以下のコマンドをクラスターノードから実行すると、リソースを作成し、そのリソースにリソース制約を配置して、リモートノードで実行できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers remote1
```

警告

リソースグループ、コロケーション制約、または順序制約でノード接続リソースを利用しないでください。

7. リモートノードのフェンスリソースを設定します。リモートノードは、クラスターノードと同じ方法でフェンスされます。クラスターノードと同様に、リモートノードで使用するフェンスリソースを設定します。リモートノードはフェンスアクションを開始できないことに注意してください。クラスターノードのみが、実際に別のノードに対してフェンシング操作を実行できます。

66.4. ポートのデフォルトの場所の変更

Pacemaker または **pacemaker_remote** のいずれかのポートのデフォルトの場所を変更する必要がある場合は、これらのデーモンのどちらにも影響を与える **PCMK_remote_port** 環境変数を設定できます。この環境変数は、以下のように **/etc/sysconfig/pacemaker** に配置して有効にできます。

```
\#==#==# Pacemaker Remote
...
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

特定のゲストノードまたはリモートノードで使用されるデフォルトのポートを変更する場合は、**PCMK_remote_port** 変数を、そのノードの **/etc/sysconfig/pacemaker** ファイルに設定する必要があります。また、ゲストノードまたはリモートノードの接続を作成するクラスターリソースを、同じポート番号で設定する必要もあります (ゲストノードの場合は **remote-port** メタデータオプション、リモートノードの場合は **port** オプションを使用します)。

66.5. PACEMAKER_REMOTE ノードを含むシステムのアップグレード

アクティブな Pacemaker リモートノードで **pacemaker_remote** サービスが停止すると、クラスターは、ノードの停止前に、リソースをノードから正常に移行します。これにより、クラスターからノードを削除せずに、ソフトウェアのアップグレードやその他の定期的なメンテナンスを実行できるようになりました。ただし、**pacemaker_remote** がシャットダウンすると、クラスターは即座に再接続を試みます。リソースの監視タイムアウトが発生する前に **pacemaker_remote** が再起動しないと、クラスターは監視操作が失敗したと判断します。

アクティブな Pacemaker リモートノードで、**pacemaker_remote** サービスが停止したときに監視が失敗しないようにするには、以下の手順に従って、**pacemaker_remote** を停止する可能性があるシステム管理を実行する前に、ノードをクラスターから削除します。

手順

1. ノードからすべてのサービスを除去する **pcs resource disable resourcename** コマンドを使用して、ノードの接続リソースを停止します。接続リソースは、リモートノードの場合は **ocf:pacemaker:remote** リソース、通常はゲストノードの場合は **ocf:heartbeat:VirtualDomain** リソースになります。ゲストノードの場合、このコマンドは VM も停止するため、メンテナンスを実行するには、クラスターの外部で (たとえば、**virsh** を使用して) VM を起動する必要があります。

   ```
   pcs resource disable resourcename
   ```

2. 必要なメンテナンスを実行します。

3. ノードをクラスターに戻す準備ができたら、**pcs resource enable** コマンドでリソースを再度有効にします。

   ```
   pcs resource enable resourcename
   ```

第67章 クラスターメンテナンスの実行

クラスターのノードでメンテナンスを実行するには、そのクラスターで実行しているリソースおよびサービスを停止するか、移行する必要がある場合があります。または、サービスを変更しない状態で、クラスターソフトウェアの停止が必要になる場合があります。Pacemaker は、システムメンテナンスを実行するための様々な方法を提供します。

- クラスターの別のノードでサービスが継続的に実行している状態で、クラスター内のノードを停止する必要がある場合は、そのクラスターノードをスタンバイモードにすることができます。スタンバイノードのノードは、リソースをホストすることができなくなります。ノードで現在アクティブなリソースは、別のノードに移行するか、(他のノードがそのリソースを実行できない場合は) 停止します。スタンバイモードの詳細は、ノードをスタンバイモードに を参照してください。

- リソースを停止せずに、現在実行しているノードから個別のリソースを移行する必要がある場合は、**pcs resource move** コマンドを使用してリソースを別のノードに移行できます。**pcs resource move** コマンドを実行すると、現在実行しているノードでそれが実行されないように、制約がリソースに追加されます。リソースを戻す準備ができたら、**pcs resource clear** または **pcs constraint delete** コマンドを実行すると、制約を削除できます。ただし、このコマンドを実行しても、リソースが必ずしも元のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なるためです。**pcs resource relocate run** コマンドを使用すると、リソースを優先ノードに移動できます。

- 完全にリソースの実行を停止して、クラスターが再び起動しないようにするには、**pcs resource disable** コマンドを使用します。**pcs resource disable** コマンドの詳細は、クラスターリソースの無効化、有効化、および禁止 を参照してください。

- Pacemaker が、リソースに対して何らかのアクションを実行しないようにする場合 (たとえば、リソースのメンテナンス中に復元アクションを無効にする場合や、**/etc/sysconfig/pacemaker** 設定をリロードする必要がある場合) は、リソースの非管理モードへの設定 で説明されているように **pcs resource unmanage** コマンドを使用します。Pacemaker Remote 接続リソースは、非管理モードにしないでください。

- クラスターを、サービスの開始や停止が行われない状態にする必要がある場合は、**maintenance-mode** クラスタープロパティーを設定できます。クラスターをメンテナンスモードにすると、すべてのリソースが自動的に非管理モードになります。メンテナンスモードのクラスターの詳細は クラスターをメンテナンスモードに を参照してください。

- RHEL High Availability Add-On および Resilient Storage Add-On に含まれるパッケージを更新する必要がある場合は、RHEL 高可用性クラスターの更新 で説明されているように、一度に 1 つのパッケージを更新するか、全体のクラスターに対して更新を行うことができます。

- Pacemaker リモートノードでメンテナンスを実行する必要がある場合は、リモートノードおよびゲストノードのアップグレード で説明されているように、リモートノードリソースを無効にすることで、ノードをクラスターから削除できます。

- RHEL クラスターで仮想マシンを移行する必要がある場合は、RHEL クラスターでの仮想マシンの移行 で説明するように、まず仮想マシンでクラスターサービスを停止してクラスターからノードを削除し、移行後にクラスターのバックアップを開始する必要があります。

67.1. ノードをスタンバイモードに

クラスターノードがスタンバイモードになると、ノードがリソースをホストできなくなります。ノードで現在アクティブなリソースは、すべて別のノードに移行されます。

以下のコマンドは、指定ノードをスタンバイモードにします。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードにします。

このコマンドは、リソースのパッケージを更新する場合に使用できます。また、設定をテストして、ノードを実際にシャットダウンせずに復元のシュミレーションを行う場合にも、このコマンドを使用できます。

```
pcs node standby node | --all
```

次のコマンドは、指定したノードをスタンバイモードから外します。このコマンドを実行すると、指定ノードはリソースをホストできるようになります。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードから外します。

```
pcs node unstandby node | --all
```

pcs resource ban コマンドを実行すると、指定されたノードでリソースが実行されないことに注意してください。**pcs node unstandby** コマンドを実行すると、指定されたノードでリソースを実行できます。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。

67.2. クラスターリソースの手動による移行

クラスターの設定を無視して、強制的にリソースを現在の場所から移行させることができます。次のような 2 つの状況が考えられます。

- ノードがメンテナンスで、そのノードで実行中の全リソースを別のノードに移行する必要がある
- 個別に指定したリソースを移行する必要がある

ノードで実行中の全リソースを別のノードに移行する場合は、そのノードをスタンバイモードにします。

個別に指定したリソースは、以下のいずれかの方法で移行できます。

- **pcs resource move** コマンドを使用して、現在実行しているノードからリソースを移行できます。
- **pcs resource relocate run** コマンドを使用して、現在のクラスターのステータス、制約、リソースの場所、およびその他の設定により決定される優先ノードへ、リソースを移行します。

67.2.1. 現在のノードからのリソースの移動

現在実行しているノードからリソースを移動するには、以下のコマンドを使用して、リソースの **resource_id** を定義どおりに指定します。移行するリソースを実行する移行先のノードを指定する場合は、**destination_node** を使用します。

```
pcs resource move resource_id [destination_node] [--master] [lifetime=lifetime]
```

注記

pcs resource move コマンドを実行すると、現在実行しているノードでリソースが実行されないように、制約がリソースに追加されます。RHEL 8.6 以降、このコマンドに **--autodelete** オプションを指定できるようになり、リソースが移動されると、このコマンドが作成する場所の制約が自動的に削除されます。以前のリリースでは、**pcs resource clear** または **pcs constraint delete** コマンドを実行して、制約を手動で削除できます。制約を削除しても、リソースが必ずしも元のノードに戻る訳ではありません。この時点でリソースが実行できる場所は、リソースの最初の設定方法によって異なります。

pcs resource move コマンドで **--master** パラメーターを指定すると、制約はリソースの昇格されたインスタンスにのみ適用されます。

任意で **pcs resource move** コマンドの **lifetime** パラメーターを設定すると、制限が維持される期間を指定できます。**lifetime** パラメーターの単位は、ISO 8601 に定義されている形式に従って指定します。ISO 8601 では、Y (年)、M (月)、W (週)、D (日)、H (時)、M (分)、S (秒) のように、単位を大文字で指定する必要があります。

分単位の M と、月単位の M を区別するには、分単位の値の前に PT を指定する必要があります。たとえば、**lifetime** パラメーターが 5M の場合は 5 カ月の間隔を示し、 **lifetime** パラメーターが PT5M の場合は 5 分の間隔を示します。

以下のコマンドは、**resource1** リソースを **example-node2** ノードへ移動し、1 時間 30 分は元のノードへ戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

以下のコマンドは、**resource1** リソースを **example-node2** ノードへ移行し、30 分は元のノードへ戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

67.2.2. リソースを優先ノードへ移行

フェイルオーバーや管理者の手作業によるノードの移行により、リソースが移行した後、フェイルオーバーの原因となった状況が改善されたとしても、そのリソースが必ずしも元のノードに戻るとは限りません。リソースを優先ノードへ移行するには、以下のコマンドを実行します。優先ノードは、現在のクラスター状態、制約、リソースの場所、およびその他の設定により決定され、時間とともに変更する可能性があります。

```
pcs resource relocate run [resource1] [resource2] ...
```

リソースを指定しないと、すべてのリソースが優先ノードに移行します。

このコマンドは、リソースのスティッキネスを無視し、各リソースの優先ノードを算出します。優先ノードの算出後、リソースを優先ノードに移行する場所の制約を作成します。リソースが移行すると、制約が自動的に削除されます。**pcs resource relocate run** コマンドによって作成された制約をすべて削除するには、**pcs resource relocate clear** コマンドを入力します。リソースの現在の状態と、リソースのスティッキネスを無視した最適なノードを表示する場合は、**pcs resource relocate show** コマンドを実行します。

67.3. クラスターリソースの無効化、有効化、および禁止

pcs resource move コマンドや **pcs resource relocate** コマンドのほかにも、クラスターリソースの動作を制御するのに使用できる様々なコマンドがあります。

クラスターリソースの無効化

実行中のリソースを手動で停止し、クラスターが再起動しないようにする場合は、以下のコマンドを使用します。その他の設定 (制約、オプション、失敗など) によっては、リソースが起動した状態のままになる可能性があります。**--wait** オプションを指定すると、pcs はリソースが停止するまで n 秒間待機します。その後、リソースが停止した場合は 0 を返し、リソースが停止しなかった場合は 1 を返します。n を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource disable resource_id [--wait[=n]]
```

RHEL 8.2 では、リソースを無効にしても、他のリソースに影響が及ばない場合に限り、リソースを無効にできます。これを確認することは、複雑なリソース関係が設定されている場合は手作業では不可能です。

- **pcs resource disable --simulate** コマンドは、クラスター設定を変更せずに、リソースを無効にする効果を表示します。
- **pcs resource disable --safe** コマンドは、あるノードから別のノードに移行されるなど、その他のリソースが何らかの影響を受けない場合にのみリソースを無効にします。**pcs resource safe-disable** コマンドは、**pcs resource disable --safe** コマンドのエイリアスです。
- **pcs resource disable --safe --no-strict** コマンドは、他のリソースが停止または降格されない場合に限りリソースを無効にします。

RHEL 8.5 では、**pcs resource disable --safe** コマンドに **--brief** オプションを指定して、エラーのみを出力できます。RHEL 8.5 では、安全な無効化操作に失敗した場合に **pcs resource disable --safe** コマンドが生成するエラーレポートには、影響を受けるリソース ID も含まれます。リソースの無効化によって影響を受けるリソースのリソース ID のみを把握する必要がある場合は、**--brief** オプションを使用します。これにより、詳細なシミュレーション結果は提供されません。

クラスターリソースの有効化

クラスターがリソースを起動できるようにするには、次のコマンドを使用します。他の設定によっては、リソースが停止したままになることがあります。**--wait** オプションを指定すると、pcs はリソースが開始するまで最長で n 秒間待機します。その後、リソースが開始した場合には 0、リソースが開始しなかった場合には 1 を返します。n を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource enable resource_id [--wait[=n]]
```

特定のノードでリソースが実行されないようにする

指定したノードでリソースが実行されないようにする場合は、次のコマンドを使用します。ノードを指定しないと、現在実行中のノードでリソースが実行されないようになります。

```
pcs resource ban resource_id [node] [--master] [lifetime=lifetime] [--wait[=n]]
```

pcs resource ban コマンドを実行すると、場所制約である -INFINITY がリソースに追加され、リソースが指定のノードで実行されないようにします。**pcs resource clear** または **pcs constraint delete** コマンドを実行すると制約を削除できます。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。

pcs resource ban コマンドの **--master** パラメーターを指定すると、制約の範囲がマスターロールに限定され、resource_id の代わりに master_id を指定する必要があります。

任意で **pcs resource ban** コマンドに **lifetime** パラメーターを設定し、制約が持続する期間を指定できます。

任意で、**pcs resource ban** コマンドに **--wait[=n]** パラメーターを設定し、移行先のノードでリソースが起動するまでの待機時間 (秒単位) できます。待機時間がこの値を超えると、リソースが起動した場合に 0 が返され、リソースが起動しなかった場合は 1 が返されます。n の値を指定しないと、デフォルトのリソースのタイムアウト値が使用されます。

現在のノードでリソースを強制的に起動
指定したリソースを現在のノードで強制的に起動する場合は、**pcs resource** コマンドの **debug-start** パラメーターを使用します。この場合、クラスターの推奨は無視され、起動しているリソースからの出力が表示されます。これは、主にデバッグリソースに使用されます。クラスターでのリソースの起動は (ほぼ) 毎回 Pacemaker で行われるため、直接 **pcs** コマンドを使用した起動は行われません。リソースが起動しない原因は、大抵、リソースが誤って設定されているか (システムログでデバッグします)、リソースが起動しないように制約が設定されているか、リソースが無効になっているかのいずれかになります。この場合は、次のコマンドを使用してリソースの設定をテストできます。ただし、通常は、クラスター内でリソースを起動するのに使用しないでください。

debug-start コマンドの形式は以下のようになります。

```
pcs resource debug-start resource_id
```

67.4. リソースの非管理モードへの設定

リソースが **非管理** モードの場合、リソースは引き続き設定に含まれますが、Pacemaker はこのリソースを管理しません。

以下のコマンドは、指定のリソースを **非管理** モードに設定します。

```
pcs resource unmanage resource1 [resource2] ...
```

以下のコマンドは、リソースをデフォルトの **管理** モードに設定します。

```
pcs resource manage resource1 [resource2] ...
```

pcs resource manage または **pcs resource unmanage** コマンドを使用してリソースグループの名前を指定できます。このコマンドは、グループのすべてのリソースに対して実行されるため、1 つのコマンドでグループ内の全リソースすべて **管理** または **非管理** モードに設定し、グループに含まれるリソースを個別に管理できます。

67.5. クラスターをメンテナンスモードに

クラスターがメンテナンスモードの場合、クラスターは指示されない限り、サービスを開始したり、停止したりしません。メンテナンスモードが完了すると、クラスターは、サービスの現在の状態のサニティーチェックを実行してから、これを必要とするサービスを停止するか、開始します。

クラスターをメンテナンスモードにするには、以下のコマンドを使用して、**maintenance-mode** クラスタープロパティーを **true** に設定します。

```
# pcs property set maintenance-mode=true
```

クラスターをメンテナンスモードから外すには、次のコマンドを使用して、**maintenance-mode** クラスタープロパティーを **false** に設定します。

```
# pcs property set maintenance-mode=false
```

設定からクラスタープロパティーを削除する場合は、次のコマンドを使用します。

```
pcs property unset property
```

代わりに **pcs property set** コマンドの値フィールドを空白にしてもクラスタープロパティーを削除することができます。これにより、そのプロパティーの値がデフォルト値に戻されます。たとえば、**symmetric-cluster** プロパティーを **false** に設定したことがある場合は、設定した値が次のコマンドにより削除され、**symmetric-cluster** の値がデフォルト値の **true** に戻されます。

```
# pcs property set symmetric-cluster=
```

67.6. RHEL 高可用性クラスターの更新

RHEL High Availability Add-On および Resilient Storage Add-On を設定するパッケージを、個別または一括で更新するには、以下に示す一般的な方法のいずれかを使用できます。

- **ローリング更新**:サービスからノードを、一度に 1 つずつ削除し、そのソフトウェアを更新してから、そのノードをクラスターに戻します。これにより、各ノードの更新中も、クラスターがサービスの提供とリソースの管理を継続できます。
- **クラスター全体の更新**:クラスター全体を停止し、更新をすべてのノードに適用してから、クラスターのバックアップを開始します。

警告

Red Hat Enterprise LInux の High Availability クラスターおよび Resilient Storage クラスターのソフトウェア更新手順を実行する場合は、更新を開始する前に、更新を行うノードがクラスターのアクティブなメンバーではないことを確認する必要があります。

これらの各方法の詳細な説明および更新手順は RHEL 高可用性またはレジリエントストレージクラスターにソフトウェア更新を適用するのに推奨されるプラクティス を参照してください。

67.7. リモートノードおよびゲストノードのアップグレード

アクティブなリモートノードまたはゲストノードで **pacemaker_remote** サービスが停止すると、クラスターは、ノードを停止する前に、ノードからリソースを適切に移行します。これにより、クラスターからノードを削除せずに、ソフトウェアのアップグレードやその他の定期的なメンテナンスを実行できるようになりました。ただし、**pacemaker_remote** がシャットダウンすると、クラスターは即座に再接続を試みます。リソースの監視タイムアウトが発生する前に **pacemaker_remote** が再起動しないと、クラスターは監視操作が失敗したと判断します。

アクティブな Pacemaker リモートノードで、**pacemaker_remote** サービスが停止したときに監視が失敗しないようにするには、以下の手順に従って、**pacemaker_remote** を停止する可能性があるシステム管理を実行する前に、ノードをクラスターから削除します。

手順

手順

1. ノードからすべてのサービスを除去する **pcs resource disable resourcename** コマンドを使用して、ノードの接続リソースを停止します。接続リソースは、リモートノードの場合は **ocf:pacemaker:remote** リソース、通常はゲストノードの場合は **ocf:heartbeat:VirtualDomain** リソースになります。ゲストノードの場合、このコマンドは VM も停止するため、メンテナンスを実行するには、クラスターの外部で (たとえば、**virsh** を使用して) VM を起動する必要があります。

```
pcs resource disable resourcename
```

2. 必要なメンテナンスを実行します。

3. ノードをクラスターに戻す準備ができたら、**pcs resource enable** コマンドでリソースを再度有効にします。

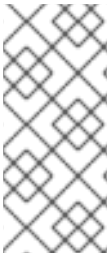
```
pcs resource enable resourcename
```

67.8. RHEL クラスターでの仮想マシンの移行

Support Policies for RHEL High Availability Clusters - General Conditions with Virtualized Cluster Members の説明にあるように、Red Hat ではハイパーバイザーまたはホスト全体でのアクティブなクラスターノードのライブマイグレーションはサポートしていません。ライブマイグレーションを実行する必要がある場合は、まず仮想マシンでクラスターサービスを停止してクラスターからノードを削除し、移行後にクラスターのバックアップを開始する必要があります。以下の手順では、クラスターから仮想マシンを削除し、仮想マシンを移行し、クラスターに仮想マシンを復元する手順の概要を説明します。

以下の手順では、クラスターから仮想マシンを削除し、仮想マシンを移行し、クラスターに仮想マシンを復元する手順の概要を説明します。

以下の手順では、全クラスターノードとして使用する仮想マシンが対象で、特別な配慮なしでライブマイグレーションが可能なクラスターリソースとして管理される仮想マシン (例: ゲストノードとして使用する仮想マシン) は対象外です。RHEL High Availability および Resilient Storage Add-On を設定するパッケージを更新するのに必要な一般的な手順は、RHEL 高可用性またはレジリエントストレージクラスターにソフトウェア更新を適用するのに推奨されるプラクティス を参照してください。



注記

この手順を実行する前に、クラスターノードを削除するクラスターのクォーラムへの影響を考慮してください。たとえば、3 ノードクラスターがあり、ノードを 1 つ削除すると、クラスターが対応できる障害数はあと 1 つだけになります。3 ノードクラスターの 1 つのノードがすでにダウンしている場合は、2 番目のノードを削除するとクォーラムが失われます。

手順

1. 移行する仮想マシンで実行しているリソースやソフトウェアの停止または移動を行う前に準備を行う必要がある場合は、以下の手順を実行します。

2. 仮想マシンで以下のコマンドを実行し、仮想マシン上のクラスターソフトウェアを停止します。

```
# pcs cluster stop
```

3. 仮想マシンのライブマイグレーションを実行します。

4. 仮想マシンでクラスターサービスを起動します。

```
# pcs cluster start
```

67.9. UUID によるクラスターの識別

Red Hat Enterprise Linux 8.7 では、作成されたクラスターには関連する UUID があります。クラスター名は一意のクラスター識別子ではないため、同じ名前の複数のクラスターを管理する設定管理データベースなどのサードパーティーツールは、その UUID によってクラスターを一意に識別できます。現在のクラスター UUID は、**pcs cluster config [show]** コマンドで表示できます。このコマンドの出力には、クラスター UUID が含まれています。

UUID を既存のクラスターに追加するには、次のコマンドを実行します。

```
# pcs cluster config uuid generate
```

既存の UUID でクラスターの UUID を再生成するには、次のコマンドを実行します。

```
# pcs cluster config uuid generate --force
```

第68章 論理ボリュームの設定および管理

68.1. 論理ボリューム管理の概要

論理ボリューム管理 (LVM) により、物理ストレージに抽象化レイヤーが作成され、論理ストレージボリュームを作成するのに役立ちます。様々な面で、物理ストレージを直接使用するよりも柔軟性が高くなります。

また、ハードウェアストレージ設定がソフトウェアから見えなくなるため、アプリケーションを停止したりファイルシステムをアンマウントしたりせずに、サイズ変更や移動が可能になります。したがって、運用コストが削減できます。

68.1.1. LVM のアーキテクチャー

以下は、LVM のコンポーネントです。

物理ボリューム

物理ボリューム (PV) は、LVM 使用用に指定されたパーティションまたはディスク全体です。詳細は、LVM 物理ボリュームの管理 を参照してください。

ボリュームグループ

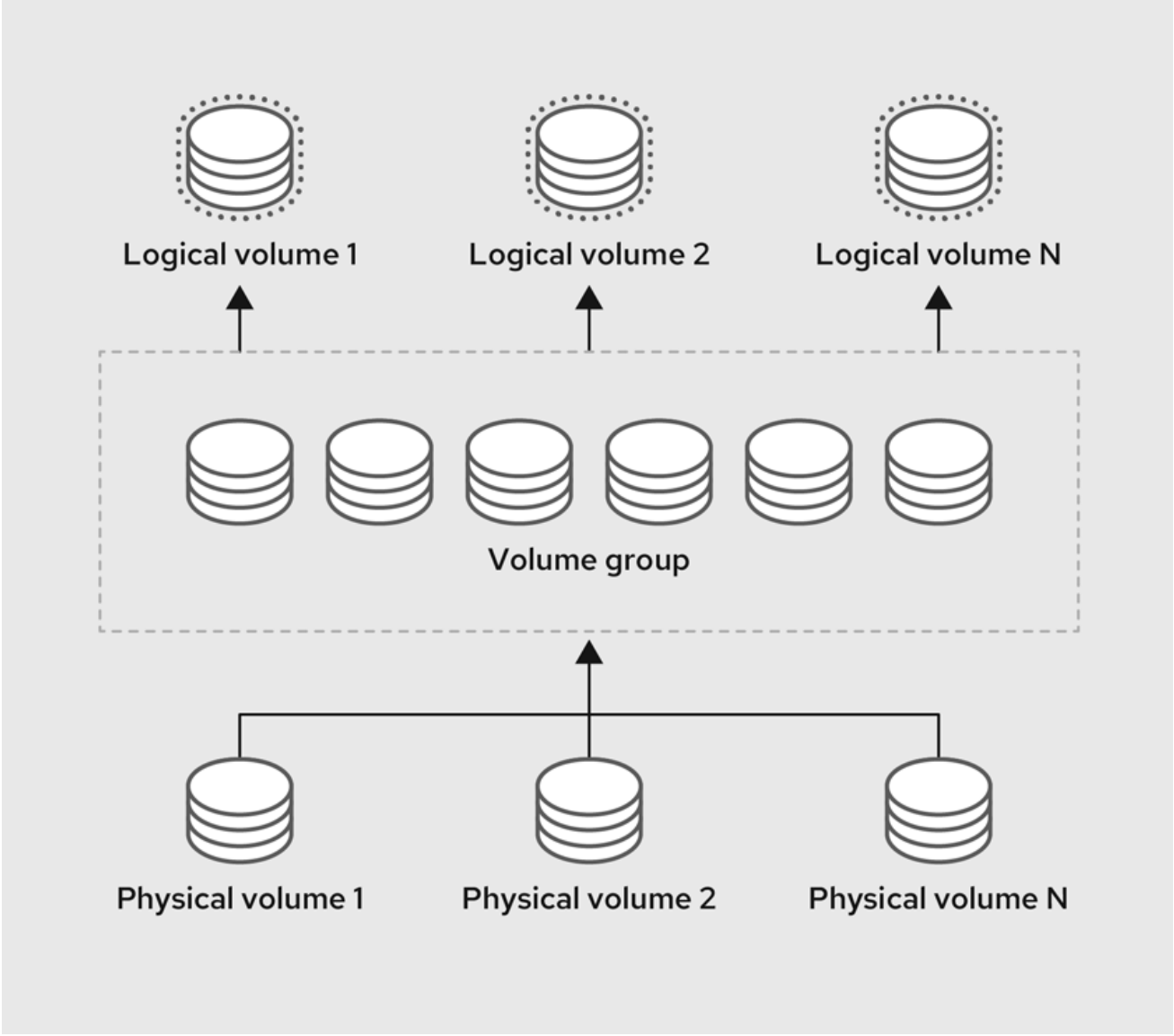
ボリュームグループ (VG) は、物理ボリューム (PV) の集合です。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。詳細は、LVM ボリュームグループの管理 を参照してください。

論理ボリューム

論理ボリュームは、マウント可能なストレージデバイスを表します。詳細は、LVM 論理ボリュームの管理 を参照してください。

以下の図は、LVM のコンポーネントを示しています。

図68.1 LVM 論理ボリュームのコンポーネント



68.1.2. LVM の利点

物理ストレージを直接使用する場合と比較して、論理ボリュームには、以下のような利点があります。

容量の柔軟性

論理ボリュームを使用すると、ディスクとパーティションを 1 つの論理ボリュームに集約できます。この機能を使用すると、ファイルシステムを複数のデバイスにまたがって拡張でき、1 つの大きなファイルシステムとして扱うことができます。

サイズ変更可能なストレージボリューム

基になるデバイスを再フォーマットしたり、パーティションを再作成したりせずに、簡単なソフトウェアコマンドを使用して論理ボリュームのサイズを拡大または縮小できます。

オンラインデータ移動

より新しく、迅速で、障害耐性の高いストレージサブシステムを導入するために、システムがアクティブな状態でもデータを移動できます。データは、ディスクが使用中の場合でもディスクに再配置できます。たとえば、ホットスワップ可能なディスクを削除する前に空にできます。

便利なデバイスの命名

論理ストレージボリュームは、ユーザー定義のカスタマイズした名前で管理できます。

ストライプ化ボリューム

2 つ以上のデバイスにまたがってデータをストライプ化する論理ボリュームを作成できます。これにより、スループットが大幅に向上します。

RAID ボリューム

論理ボリュームは、データの RAID を設定する際に便利な方法を提供します。これにより、デバイス障害に対する保護が可能になり、パフォーマンスが向上します。

ボリュームスナップショット

論理ボリュームの特定の時点のコピーであるスナップショットを作成して、一貫性のあるバックアップを作成したり、実際のデータに影響を与えずに変更の影響をテストしたりすることができます。

シンボリューム

論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能な物理容量よりも大きな論理ボリュームを作成できます。

キャッシュボリューム

キャッシュ論理ボリュームは、SSD ドライブなどの高速なブロックデバイスを使用して、大規模で低速なブロックデバイスのパフォーマンスを向上させます。

68.2. LVM 物理ボリュームの管理

物理ボリューム (PV) は、LVM 使用用に指定されたパーティションまたはディスク全体です。LVM 論理ボリューム用にデバイスを使用する場合は、デバイスを物理ボリュームとして初期化する必要があります。

ディスクデバイス全体を物理ボリュームに使用している場合は、そのディスクにはパーティションテーブルを含めないでください。ディスクパーティションが DOS の場合は、**fdisk**、**cfdisk** などのコマンドを使用して、パーティション ID を 0x8e に設定している必要があります。ディスクデバイス全体を物理ボリュームに使用している場合は、そのディスクにはパーティションテーブルを含めないでください。既存のパーティションテーブルはすべて消去する必要があります。これにより、そのディスク上のすべてのデータが効果的に破壊されます。root として、**wipefs -a <PhysicalVolume>** コマンドを使用して、既存のパーティションテーブルを削除できます。

68.2.1. 物理ボリュームの概要

ブロックデバイスを物理ボリュームとして初期化すると、デバイスの先頭位置にラベルが付けられます。以下は、LVM ラベルについて説明しています。

- LVM ラベルにより物理デバイスの正しい識別とデバイスの順序付けが行われます。ラベルが付けられていない、LVM 以外のデバイスは、起動時にシステムが検出した順序に応じて、再起動後に名前が変更される場合があります。LVM ラベルは、再起動してもクラスター全体で維持されます。

- LVM ラベルは、デバイスを LVM 物理ボリュームとして識別するものです。これには、物理ボリューム用のランダムな一意識別子 (UUID) が含まれます。また、ブロックデバイスのサイズもバイト単位で保存し、LVM メタデータがデバイスのどこに保存されているかも記録します。

- LVM ラベルは、デフォルトでは 2 番目の 512 バイトセクターに配置されます。物理ボリュームを作成する場合は、先頭の 4 つのセクターのいずれかにラベルを配置することにより、このデフォルト設定を書き換えることができます。これにより、必要に応じて LVM ボリュームを、このセクターを利用する他のユーザーと併用できるようになります。

以下は、LVM メタデータについて説明しています。

- LVM メタデータには、システムにある LVM ボリュームグループの設定詳細が含まれています。デフォルトでは、メタデータの複製コピーが、ボリュームグループ内で、すべての物理ボ

リュームの、すべてのメタデータ領域に保存されています。LVM メタデータのサイズは小さく、ASCII 形式が使用されます。

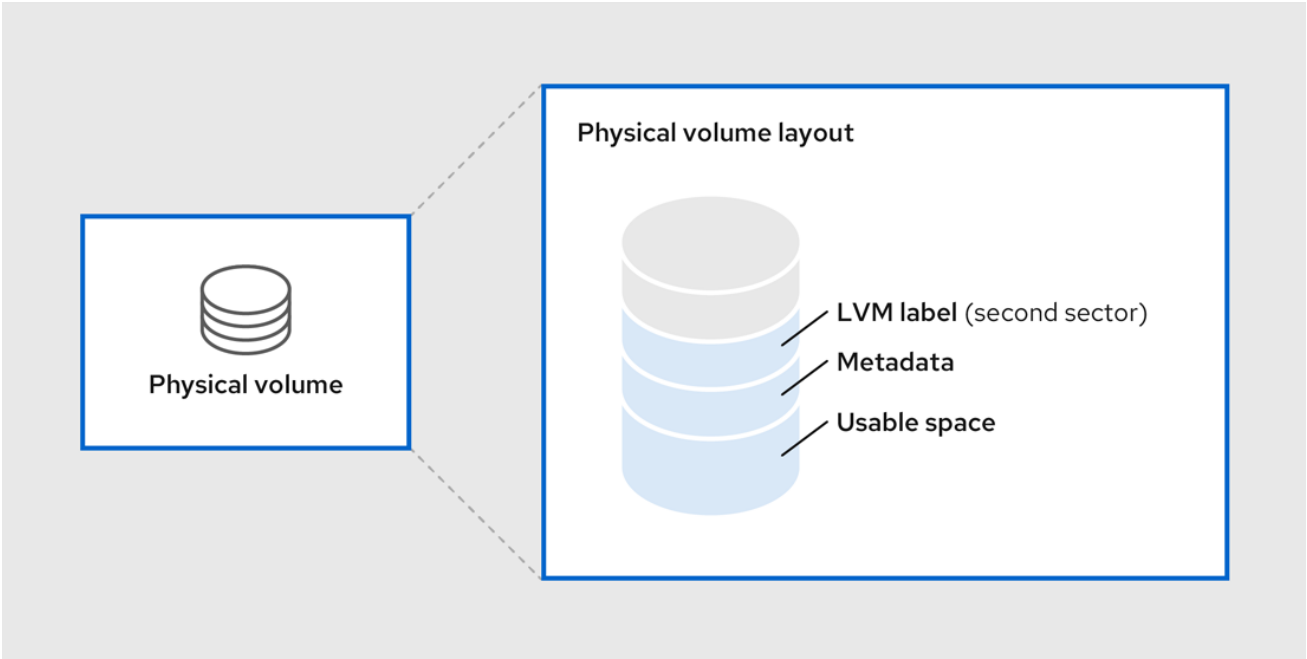
- 現在、LVM では、各物理ボリュームにメタデータのコピーを 1 つまたは 2 つ保存できます。コピーをゼロにすることもできます。デフォルトでは 1 つ保存されます。物理ボリューム上に保存するメタデータのコピー数を一度設定したら、その数を後で変更することはできません。最初のコピーはデバイスの先頭にあるラベルの後に保存されます。2 つ目のコピーがある場合は、デバイスの最後に配置されます。意図したものとは別のディスクに誤って書き込みを行い、ディスクの先頭領域を上書きしてしまった場合でも、デバイス後部にある 2 つ目のコピーでメタデータを復元できます。

次の図は、LVM 物理ボリュームのレイアウトを示しています。LVM ラベルが 2 番目のセクターにあり、その後にメタデータ領域、使用可能なデバイス領域と続きます。

注記

Linux カーネルおよび本書では、セクターのサイズを 512 バイトとしています。

図68.2 物理ボリュームのレイアウト



関連情報

- ディスク上の複数パーティション

68.2.2. ディスク上の複数パーティション

LVM を使用して、ディスクパーティションから物理ボリューム (PV) を作成できます。

Red Hat では、以下のような理由により、ディスク全体に対応するパーティションを 1 つ作成し、1 つの LVM 物理ボリュームとしてラベルを付けることを推奨しています。

管理上の利便性

各ディスクが一度だけ表示されると、システムのハードウェアの追跡が簡単になります。これは、特にディスクに障害が発生した場合に役に立ちます。

ストライピングのパフォーマンス

LVM は、2 つの物理ボリュームが同じ物理ディスクにあるかどうかを認識しません。2 つの物理ボリュームが同じ物理ディスクにあるときに、ストライプ化された論理ボリュームを作成すると、作成されたボリュームのディスクは同じでも、パーティションは異なる可能性があります。このとき、パフォーマンスは、改善ではなく低下します。

RAID の冗長性

LVM は、2 つの物理ボリュームが同じデバイスにあるかどうかを判断できません。2 つの物理ボリュームが同じデバイスにある場合に RAID 論理ボリュームを作成すると、パフォーマンスとフォールトトレランスが失われる可能性があります。

1 つのディスクを、複数の LVM 物理ボリュームに分割しないといけない場合があります (推奨はされません)。たとえば、ディスクがほとんどないシステムで、既存システムを LVM ボリュームに移行する場合に、パーティション間でデータを移動しなければならない場合があります。さらに、大容量のディスクが存在し、管理目的で複数のボリュームグループを必要とする場合は、そのディスクにパーティションを設定する必要があります。ディスクに複数のパーティションがあり、そのパーティションがいずれも同じボリュームグループにある場合に、ボリュームを作成するときは、論理ボリュームに追加するパーティションを注意して指定してください。

LVM は、パーティション化していないディスクを物理ボリュームとして使用することをサポートしますが、パーティションなしで PV を作成すると、混合オペレーティングシステム環境で問題が発生する可能性があるため、ディスク全体を単一のパーティションとして作成することが推奨されます。他のオペレーティングシステムはデバイスを空き状態として解釈し、ドライブの先頭にある PV ラベルを上書きする可能性があります。

68.2.3. LVM 物理ボリュームの作成

この手順では、LVM 物理ボリューム (PV) を作成し、ラベルを付ける方法を説明します。

この手順では、**/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** を、システムで利用可能なストレージデバイスに置き換えます。

前提条件

- **lvm2** パッケージがインストールされている。

手順

1. **pvcreate** コマンドに、スペースで区切られたデバイス名を引数として使用して、複数の物理ボリュームを作成します。

```
# pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Physical volume "/dev/vdb3" successfully created.
```

   これにより、ラベルが **/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** に配置され、それらが LVM に属する物理ボリュームとしてマークされます。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成した物理ボリュームを表示します。

   a. **pvdisplay** コマンド: 各物理ボリュームの詳細をそれぞれ複数行出力します。物理プロパティー (サイズ、エクステント、ボリュームグループなど) および他のオプションが、決められた形式で表示されます。

```
# pvdisplay
```

```
--- NEW Physical volume ---
  PV Name               /dev/vdb1
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb2
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb3
  VG Name
  PV Size               1.00 GiB
[..]
```

b. **pvs** コマンド: 物理ボリュームの情報を設定可能な形式で出力します。

```
# pvs
  PV         VG  Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
/dev/vdb3        lvm2           1020.00m   0
```

c. **pvscan** コマンド: システムにある物理ボリュームで対応している LVM ブロックデバイスをすべてスキャンします。このコマンドで、特定の物理ボリュームがスキャンされないように、**lvm.conf** ファイルでフィルターを定義することができます。

```
# pvscan
  PV  /dev/vdb1                    lvm2 [1.00 GiB]
  PV  /dev/vdb2                    lvm2 [1.00 GiB]
  PV  /dev/vdb3                    lvm2 [1.00 GiB]
```

関連情報

- **pvcreate (8)**、**pvdisplay (8)**、**pvs (8)**、**pvscan (8)**、および **lvm (8)** の man ページ

68.2.4. LVM 物理ボリュームの削除

デバイスを LVM で使用する必要がなくなった場合、**pvremove** コマンドを使用して LVM ラベルを削除できます。**pvremove** コマンドを実行すると、空の物理ボリュームにある LVM メタデータをゼロにします。

手順

1. 物理ボリュームを削除します。

```
# pvremove /dev/vdb3
Labels on physical volume "/dev/vdb3" successfully wiped.
```

2. 既存の物理ボリュームを表示し、必要なボリュームが削除されているかどうかを確認します。

```
# pvs
  PV         VG   Fmt    Attr    PSize      PFree
```

```
/dev/vdb1      lvm2        1020.00m   0
/dev/vdb2      lvm2        1020.00m   0
```

削除する物理ボリュームがボリュームグループの一部になっている場合は、**vgreduce** コマンドで、ボリュームグループから物理ボリュームを取り除く必要があります。詳細は、ボリュームグループからの物理ボリュームの削除 を参照してください。

関連情報

- **pvremove(8)** の man ページ

68.2.5. 関連情報

- parted でディスクにパーティションテーブルを作成
- **parted(8)** man ページ

68.3. LVM ボリュームグループの管理

ボリュームグループ (VG) は、物理ボリューム (PV) の集合です。これにより、論理ボリューム (LV) に割り当て可能なディスク領域のプールが作成されます。

ボリュームグループ内で、割り当て可能なディスク領域は、エクステントと呼ばれる固定サイズの単位に分割されます。割り当て可能な領域の最小単位は、1 エクステントです。エクステントは、物理ボリュームでは物理エクステントと呼ばれます。

論理ボリュームには、物理エクステントと同じサイズの論理エクステントが割り当てられます。そのため、エクステントのサイズは、ボリュームグループ内のすべての論理ボリュームで同じになります。ボリュームグループは、論理エクステントを物理エクステントにマッピングします。

68.3.1. LVM ボリュームグループの作成

この手順では、物理ボリューム**/dev/vdb1** および **/dev/vdb2**を使用して、LVM ボリュームグループ (VG) **myvg**を作成する方法を説明します。

前提条件

- **lvm2** パッケージがインストールされている。
- 物理ボリュームが作成されます。物理ボリュームの作成方法は、LVM 物理ボリュームの作成 を参照してください。

手順

1. ボリュームグループを作成します。

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
 Volume group "myvg" successfully created.
```

これにより、**myvg** という名前の VG が作成されます。物理ボリュームの **/dev/vdb1** および **/dev/vdb2** は、ボリュームグループ **myvg** のベーストレージレベルです。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成したボリュームグループを表示します。

a. **vgs** コマンド: ボリュームグループの情報を設定可能な形式で提供し、1 ボリュームグループにつき 1 行ずつ表示します。

```
# vgs
  VG    #PV #LV #SN  Attr  VSize   VFree
 myvg   2    0   0   wz-n  159.99g 159.99g
```

b. **vgdisplay** コマンド: 決められた形式でボリュームグループのプロパティー (サイズ、エクステント、物理ボリュームの数など) およびその他のオプションを表示します。以下の例は、ボリュームグループ **myvg** に関する **vgdisplay** コマンドの出力を示しています。既存のすべてのボリュームグループを表示するには、ボリュームグループを指定しないでください。

```
# vgdisplay myvg
  --- Volume group ---
  VG Name               myvg
  System ID
  Format                lvm2
  Metadata Areas        4
  Metadata Sequence No  6
  VG Access             read/write
[..]
```

c. **vgscan** コマンド: ボリュームグループ用に、システムにあるサポートされるすべての LVM ブロックデバイスをスキャンします。

```
# vgscan
  Found volume group "myvg" using metadata type lvm2
```

3. オプション:オプション: 空き物理ボリュームを 1 つまたは複数追加して、ボリュームグループの容量を増やします。

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

4. オプション:既存のボリュームグループの名前を変更します。

```
# vgrename myvg myvg1
Volume group "myvg" successfully renamed to "myvg1"
```

関連情報

- **vgcreate (8)**、**vgextend (8)**、**vgdisplay (8)**、**vgs (8)**、**vgscan (8)**、**vgrename (8)**、および **lvm (8)** の man ページ

68.3.2. LVM ボリュームグループの統合

2 つのボリュームグループを統合して 1 つのボリュームグループにするには、**vgmerge** コマンドを使用します。ボリュームの物理エクステントサイズが同じで、かつ両ボリュームグループの物理ボリュームおよび論理ボリュームのサマリーがマージ先ボリュームグループの制限内に収まる場合は、非アクティブなマージ元のボリュームを、アクティブまたは非アクティブのマージ先ボリュームにマージができます。

手順

- 非アクティブなボリュームグループ **databases** をアクティブまたは非アクティブなボリュームグループ **myvg** にマージして、詳細なランタイム情報を提供します。

```
# vgmerge -v myvg databases
```

関連情報

- **vgmerge(8)** の man ページ

68.3.3. ボリュームグループからの物理ボリュームの削除

ボリュームグループから未使用の物理ボリュームを削除するには、**vgreduce** コマンドを使用します。**vgreduce** コマンドは、空の物理ボリュームを 1 つまたは複数削除して、ボリュームグループの容量を縮小します。これにより、物理ボリュームが解放され、異なるボリュームグループで使用したり、システムから削除できるようになります。

手順

1. 物理ボリュームがまだ使用中の場合は、データを同じボリュームグループから別の物理ボリュームに移行します。

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

2. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にない場合は、以下を行います。

   a. **/dev/vdb4** から、物理ボリュームを新規作成します。

```
# pvcreate /dev/vdb4
  Physical volume "/dev/vdb4" successfully created
```

   b. 新規作成した物理ボリュームを **myvg** ボリュームグループに追加します。

```
# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended
```

   c. データを **/dev/vdb3** から **/dev/vdb4** に移動します。

```
# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. ボリュームグループから物理ボリューム **/dev/vdb3** を削除します。

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

検証

- /dev/vdb3 物理ボリュームが myvg ボリュームグループから削除されているかどうかを確認します。

```
# pvs
  PV         VG    Fmt   Attr   PSize       PFree      Used
  /dev/vdb1 myvg  lvm2   a--    1020.00m    0         1020.00m
  /dev/vdb2 myvg  lvm2   a--    1020.00m    0         1020.00m
  /dev/vdb3       lvm2   a--    1020.00m   1008.00m    12.00m
```

関連情報

- **vgreduce(8)**、**pvmove(8)**、および **pvs(8)** の man ページ

68.3.4. LVM ボリュームグループの分割

この手順では、既存のボリュームグループを分割する方法を説明します。この物理ボリュームに未使用領域が十分にあれば、新たにディスクを追加しなくてもボリュームグループを作成できます。

初期設定では、ボリュームグループ myvg は/dev/vdb1、/dev/vdb2、および /dev/vdb3 で設定されます。この手順を完了すると、ボリュームグループ myvg は /dev/vdb1 および /dev/vdb2 で設定され、2 番目のボリュームグループ yourvg は /dev/vdb3 で設定されます。

前提条件

- ボリュームグループに十分な空き領域がある。**vgscan** コマンドを使用すると、現在ボリュームグループで利用可能な空き領域の容量を確認できます。
- 既存の物理ボリュームの空き容量に応じて、**pvmove** コマンドを使用して、使用されている物理エクステントをすべて他の物理ボリュームに移動します。詳細は、ボリュームグループからの物理ボリュームの削除 を参照してください。

手順

1. 既存のボリュームグループ myvg を新しいボリュームグループ yourvg に分割します。

```
# vgsplit myvg yourvg /dev/vdb3
  Volume group "yourvg" successfully split from "myvg"
```

注記

既存のボリュームグループを使用して論理ボリュームを作成した場合は、次のコマンドを実行して論理ボリュームを非アクティブにします。

```
# lvchange -a n /dev/myvg/mylv
```

論理ボリュームを作成する方法は、LVM 論理ボリュームの管理 を参照してください。

2. 2 つのボリュームグループの属性を表示します。

```
# vgs
```

```
  VG     #PV #LV #SN Attr   VSize  VFree
  myvg     2   1   0 wz--n- 34.30G 10.80G
  yourvg   1   0   0 wz--n- 17.15G 17.15G
```

検証

- 新規作成したボリュームグループ **yourvg** が、**/dev/vdb3** 物理ボリュームで設定されているかどうかを確認します。

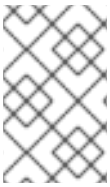
```
# pvs
  PV         VG     Fmt   Attr   PSize       PFree      Used
  /dev/vdb1 myvg   lvm2   a--    1020.00m    0          1020.00m
  /dev/vdb2 myvg   lvm2   a--    1020.00m    0          1020.00m
  /dev/vdb3 yourvg lvm2   a--     1020.00m   1008.00m    12.00m
```

関連情報

- **vgsplit(8)**、**vgs(8)**、および **pvs(8)** の man ページ

68.3.5. ボリュームグループを別のシステムへ移動

LVM ボリュームグループ全体を、別のシステムに移動できます。これを実行するには、**vgexport** と **vgimport** のコマンドの使用が推奨されます。



注記

vgimport コマンドの **--force** 引数を使用できます。この引数を使用すると、物理ボリュームがないボリュームグループをインポートし、その後に **vgreduce --removemissing** コマンドを実行することが可能になります。

vgexport コマンドは、非アクティブのボリュームグループにシステムがアクセスできないようにするため、物理ボリュームの割り当て解除が可能になります。**vgimport** コマンドは、**vgexport** コマンドで非アクティブにしていたボリュームグループに、マシンが再度アクセスできるようにします。

ボリュームグループを 2 つのシステム間で移行するには、以下の手順に従います。

1. ボリュームグループ内のアクティブなボリュームのファイルにアクセスしているユーザーがいないことを確認してから、論理ボリュームをアンマウントします。

2. **vgchange** コマンドで **-a n** 引数を使用して、そのボリュームグループを非アクティブとしてマークします。これによりこのボリュームグループでこれ以上の動作が発生しないようにします。

3. **vgexport** コマンドを使用してボリュームグループをエクスポートします。これにより、削除するシステムからボリュームグループへアクセスできなくなります。
   ボリュームグループをエクスポートして **pvscan** コマンドを実行すると、以下の例のように、エクスポート先のボリュームグループに物理ボリュームが表示されます。

```
# pvscan
  PV /dev/sda1    is in exported VG myvg [17.15 GB / 7.15 GB free]
  PV /dev/sdc1    is in exported VG myvg [17.15 GB / 15.15 GB free]
  PV /dev/sdd1   is in exported VG myvg [17.15 GB / 15.15 GB free]
  ...
```

次にシステムがシャットダウンする時に、ボリュームグループを設定していたディスクを外して、新しいシステムに接続できます。

4. ディスクが新しいシステムに接続したら、**vgimport** コマンドを使用してボリュームグループをインポートし、新しいシステムからアクセスできるようにします。

5. **vgchange** コマンドで **-a y** 引数を使用して、ボリュームグループをアクティブにします。

6. ファイルシステムをマウントして使用できるようにします。

68.3.6. LVM ボリュームグループの削除

この手順では、既存のボリュームグループを削除する方法を説明します。

前提条件

- ボリュームグループには論理ボリュームがありません。ボリュームグループから論理ボリュームを削除するには、LVM 論理ボリュームの削除 を参照してください。

手順

1. クラスター環境にボリュームグループが存在する場合は、その他のすべてのノードで、ボリュームグループの **lockspace** を停止します。削除するノード以外の全ノードで、次のコマンドを実行します。

```
# vgchange --lockstop vg-name
```

ロックが停止するのを待ちます。

2. ボリュームグループを削除します。

```
# vgremove vg-name
  Volume group "vg-name" successfully removed
```

関連情報

- **vgremove(8)** の man ページ

68.4. LVM 論理ボリュームの管理

論理ボリュームは、ファイルシステム、データベース、またはアプリケーションが使用できる仮想のブロックストレージデバイスです。LVM 論理ボリュームを作成する場合は、物理ボリューム (PV) をボリュームグループ (Volume Group: VG) に統合します。これによりディスク領域のプールが作成され、そこから LVM 論理ボリューム (Logical Volume: LV) を割り当てます。

68.4.1. 論理ボリュームの概要

管理者は、標準のディスクパーティションとは異なり、データを破棄せずに論理ボリュームを拡大または縮小できます。ボリュームグループの物理ボリュームが別のドライブまたは RAID アレイにある場合は、ストレージデバイスに論理ボリュームを分散することもできます。

論理ボリュームを、ボリュームに必要なデータよりも小さい容量に縮小すると、データが失われる可能性があります。さらに、ファイルシステムの中には縮小できないものもあります。柔軟性を最大限にするために、現在のニーズに合わせて論理ボリュームを作成し、過剰なストレージ容量を未割り当ての状

態にします。必要に応じて、未割り当ての領域を使用するように、論理ボリュームを安全に拡張できます。

重要

AMD システム、Intel システム、ARM システム、および IBM Power Systems サーバーで、ブートローダーは LVM ボリュームを読み取ることができません。このため、**/boot** パーティションは、LVM ではなく標準のパーティションで作成してください。IBM Z の場合は、**zipl** ブートローダーによりリニアマッピングを使用した LVM 論理ボリューム上の **/boot** に対応しています。デフォルトのインストールプロセスでは、/ パーティションと swap パーティションは常に LVM ボリューム内に、**/boot** パーティションは別途、物理ボリューム上に作成されます。

以下は、論理ボリュームの種類になります。

リニアボリューム

リニアボリュームは、複数の物理ボリュームの領域を 1 つの論理ボリュームに統合します。たとえば、60GB ディスクが 2 つある場合は、120GB の論理ボリュームを作成できます。物理ストレージは連結されます。

ストライプ化論理ボリューム

LVM 論理ボリュームにデータを書き込む際に、ファイルシステムは、基になる物理ボリューム全体にデータを分配します。このとき、ストライプ化論理ボリュームを作成すると、データを物理ボリュームに書き込む方法を制御できます。順次の読み取りおよび書き込みが大量に行われる場合には、これによりデータ I/O の効率を向上できます。
ストライピングは、ラウンドロビン式で、指定した数の物理ボリュームにデータを書き込んでいくことで、パフォーマンスを向上させます。I/O は、ストライピングでは並行して実行されます。これにより、ストライプで追加される各物理ボリュームでは、ほぼ直線的なパフォーマンスの向上が期待できます。

RAID 論理ボリューム

LVM は、RAID レベル 0、1、4、5、6、10 に対応します。RAID 論理ボリュームはクラスターには対応していません。RAID 論理ボリュームを作成するとき、LVM は、データまたはアレイ内のパリティーサブボリュームごとに、サイズが 1 エクステントのメタデータサブボリュームを作成します。

シンプロビジョニングされた論理ボリューム (シンボリューム)

シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。シンプロビジョニングされたボリュームセットを作成すると、システムは要求されるストレージの全量を割り当てる代わりに、実際に使用する容量を割り当てることができます。

スナップショットボリューム

LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。スナップショットの取得後に作成元のデバイスに変更が加えられると、データが変更する前に、これから変更する部分のコピーがスナップショット機能により作成されるため、このコピーを使用して、デバイスの状態を再構築できます。

シンプロビジョニングされたスナップショットボリューム

シンプロビジョニングされたスナップショットボリュームを使用すると、同じデータボリュームにより多くの仮想デバイスを格納できます。シンプロビジョニングされたスナップショットは、特定のタイミングでキャプチャーする際にすべてのデータをコピーしていないため、便利です。

キャッシュボリューム

LVM は、高速ブロックデバイス (SSD ドライブなど) を、大規模で低速なブロックデバイスのライトバックまたはライトスルーのキャッシュとして使用することに対応します。既存の論理ボリュームのパフォーマンスを改善するためにキャッシュ論理ボリュームを作成したり、大規模で低速なデ

バイスと共に小規模で高速なデバイスで設定される新規のキャッシュ論理ボリュームを作成したりできます。

68.4.2. CLI コマンドの使用

以下のセクションでは、LVM CLI コマンドの一般的な操作機能を説明します。

コマンドラインの引数で単位の指定
コマンドラインの引数でサイズが必要な場合は、常に単位を明示的に指定できます。単位を指定しないと、デフォルトで KB または MB が指定されます。LVM CLI コマンドでは、分数を使用できません。

コマンドライン引数で単位を指定する場合は、LVM が大文字と小文字を区別しません。たとえば、M と m は同じで、2 の累乗 (1024 の倍数) が使用されます。ただし、コマンドで **--units** 引数を指定すると、小文字は、単位が 1024 の倍数であることを示し、その単位は 1000 の倍数であることを示します。

ボリュームグループおよび論理ボリュームの指定
LVM CLI コマンドでボリュームグループまたは論理ボリュームを指定する場合は、以下の点に留意してください。

- ここで、コマンドではボリュームグループまたは論理ボリューム名を引数として取り、完全パス名はオプションになります。たとえば、ボリュームグループ **vg0** 内の論理ボリューム **lvol0** は、**vg0/lvol0** と指定できます。
- ボリュームグループのリストは必須ですが、空の場合は、ボリュームグループのリストが置き換えられます。
- 論理ボリュームのリストが必要ですが、ボリュームグループを指定すると、そのボリュームグループにある論理ボリュームがすべて置き換えられます。たとえば、**lvdisplay vg0** コマンドは、ボリュームグループ **vg0** の論理ボリュームをすべて表示します。

出力の詳細レベルを上げる
すべての LVM コマンドは、出力の詳細レベルを上げるために複数回入力できる **-v** 引数を受け入れます。以下の例は、**lvcreate** コマンドのデフォルト出力を示しています。

```
# lvcreate -L 50MB new_vg
  Rounding up size to full physical extent 52.00 MB
  Logical volume "lvol0" created
```

以下のコマンドは、**lvcreate** コマンドの出力と、**-v** 引数を表示します。

```
# lvcreate -v -L 50MB new_vg
  Rounding up size to full physical extent 52.00 MB
    Archiving volume group "new_vg" metadata (seqno 1).
    Creating logical volume lvol0
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 2).
    Activating logical volume new_vg/lvol0.
    activation/volume_list configuration setting not defined: Checking only host tags for new_vg/lvol0.
    Creating new_vg-lvol0
    Loading table for new_vg-lvol0 (253:0).
    Resuming new_vg-lvol0 (253:0).
    Wiping known signatures on logical volume "new_vg/lvol0"
    Initializing 4.00 KiB of logical volume "new_vg/lvol0" with value 0.
  Logical volume "lvol0" created
```

引数の **-vv**、**-vvv**、および **-vvvv** を使用すると、表示されるコマンド実行がより詳細になります。**-vvvv** 引数は、この時点で情報の最大数を提供します。以下の例は、**lvcreate** コマンドで **-vvvv** 引数を指定して、出力の最初の行を示しています。

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913         Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:916         O_DIRECT will be used
#config/config.c:864       Setting global/locking_type to 1
#locking/locking.c:138       File-based locking selected.
#config/config.c:841       Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358       Getting target version for linear
#ioctl/libdm-iface.c:1569         dm version   OF   [16384]
#ioctl/libdm-iface.c:1569         dm versions   OF   [16384]
#activate/activate.c:358       Getting target version for striped
#ioctl/libdm-iface.c:1569         dm versions   OF   [16384]
#config/config.c:864       Setting activation/mirror_region_size to 512
...
```

LVM CLI コマンドのヘルプの表示
コマンドの **--help** 引数を使用して、LVM CLI コマンドのヘルプを表示できます。

```
# commandname --help
```

コマンドの man ページを表示するには、**man** コマンドを実行します。

```
# man commandname
```

man lvm コマンドは、LVM に関する一般的なオンライン情報を提供します。

68.4.3. LVM 論理ボリュームの作成

この手順では、**/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** 物理ボリュームを使用して作成された **myvg** ボリュームグループから **mylv** LVM 論理ボリューム (LV) を作成する方法を説明します。

前提条件

- **lvm2** パッケージがインストールされている。
- ボリュームグループが作成されます。詳細は、LVM ボリュームグループの作成 を参照してください。

手順

1. 論理ボリュームを作成します。

```
# lvcreate -n mylv -L 500M myvg
```

-n オプションを使用して LV 名を **mylv** に設定し、**-L** オプションを使用して、Mb 単位で LV のサイズを設定しますが、他の単位を使用することもできます。デフォルトでは、論理ボリュームのタイプはリニアですが、**--type** オプションを使用して必要なタイプを指定できます。

重要

ボリュームグループに要求されるサイズとタイプの空き物理エクステントが十分にない場合、このコマンドは失敗します。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成した論理ボリュームを表示します。

   a. **lvs** コマンド: 論理ボリューム情報を設定可能な形式で提供して、1つの論理ボリュームにつき1行ずつ表示します。

   ```
   # lvs
     LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
    mylv myvg -wi-ao----   500.00m
   ```

   b. **lvdisplay** コマンド: 決められた形式で、論理ボリュームのプロパティー (サイズ、レイアウト、マッピングなど) を表示します。

   ```
   # lvdisplay -v /dev/myvg/mylv
     --- Logical volume ---
     LV Path                /dev/myvg/mylv
     LV Name                mylv
     VG Name                myvg
     LV UUID                YTnAk6-kMlT-c4pG-HBFZ-Bx7t-ePMk-7YjhaM
     LV Write Access        read/write
   [..]
   ```

   c. **lvscan** コマンド: システム内のすべての論理ボリュームをスキャンし、それらをリスト表示します。

   ```
   # lvscan
    ACTIVE                 '/dev/myvg/mylv' [500.00 MiB] inherit
   ```

3. 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **xfs** ファイルシステムが作成されます。

   ```
   # mkfs.xfs /dev/myvg/mylv
   meta-data=/dev/myvg/mylv       isize=512    agcount=4, agsize=32000 blks
            =                     sectsz=512   attr=2, projid32bit=1
            =                     crc=1        finobt=1, sparse=1, rmapbt=0
            =                     reflink=1
   data     =                       bsize=4096   blocks=128000, imaxpct=25
            =                     sunit=0      swidth=0 blks
   naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
   log      =internal log           bsize=4096   blocks=1368, version=2
            =                     sectsz=512   sunit=0 blks, lazy-count=1
   realtime =none                   extsz=4096   blocks=0, rtextents=0
   Discarding blocks...Done.
   ```

4. 論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

   ```
   # mount /dev/myvg/mylv /mnt

   # df -h
   ```

```
Filesystem              1K-blocks  Used   Available Use% Mounted on

/dev/mapper/myvg-mylv   506528    29388  477140     6%   /mnt
```

関連情報

- **lvcreate(8)**、**lvdisplay(8)**、**lvs(8)**、**lvscan(8)**、**lvm(8)**、および **mkfs.xfs(8)** の man ページ

68.4.4. RAID0 ストライピング 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のデータサブボリューム全体に分散します。以下の手順では、ディスク間でデータをストライピングする **mylv** という LVM RAID0 論理ボリュームを作成します。

前提条件

1. 3 つ以上の物理ボリュームを作成している。物理ボリュームの作成方法は、LVM 物理ボリュームの作成 を参照してください。
2. ボリュームグループを作成している。詳細は、LVM ボリュームグループの作成 を参照してください。

手順

1. 既存のボリュームグループから RAID0 論理ボリュームを作成します。次のコマンドは、ボリュームグループ **myvg** から RAID0 ボリューム **mylv** を作成します。これは、サイズが **2G** で、ストライプが 3 つ、ストライプ サイズが **4kB** です。

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv my_vg
  Rounding size 2.00 GiB (512 extents) up to stripe boundary size 2.00 GiB(513 extents).
  Logical volume "mylv" created.
```

2. RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに ext4 ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/my_vg/mylv
```

3. 論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

```
# mount /dev/my_vg/mylv /mnt

# df
Filesystem              1K-blocks     Used  Available  Use% Mounted on
/dev/mapper/my_vg-mylv   2002684     6168  1875072    1%   /mnt
```

検証

- 作成された RAID0 ストライピング論理ボリュームを表示します。

```
# lvs -a -o +devices,segtype my_vg
  LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert Devices Type
  mylv my_vg rwi-a-r--- 2.00g mylv_rimage_0(0),mylv_rimage_1(0),mylv_rimage_2(0) raid0
```

```
[mylv_rimage_0] my_vg iwi-aor--- 684.00m /dev/sdf1(0) linear
[mylv_rimage_1] my_vg iwi-aor--- 684.00m /dev/sdg1(0) linear
[mylv_rimage_2] my_vg iwi-aor--- 684.00m /dev/sdh1(0) linear
```

68.4.5. LVM 論理ボリュームの名前の変更

この手順では、既存の論理ボリューム **mylv** の名前を **mylv1** に変更する方法を説明します。

手順

1. 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。

```
# umount /mnt
```

/mnt は、マウントポイントに置き換えます。

2. 既存の論理ボリュームの名前を変更します。

```
# lvrename myvg mylv mylv1
Renamed "mylv" to "mylv1" in volume group "myvg"
```

また、デバイスへの完全パスを指定して、論理ボリュームの名前を変更することもできます。

```
# lvrename /dev/myvg/mylv /dev/myvg/mylv1
```

関連情報

- **lvrename(8)** の man ページ

68.4.6. 論理ボリュームからのディスクの削除

この手順では、ディスクを交換するか、別のボリュームで使用するために、既存の論理ボリュームからディスクを削除する方法を説明します。

ディスクを削除する前に、LVM 物理ボリュームのエクステントを、別のディスクまたはディスクセットに移動する必要があります。

手順

1. LV を使用する際に、物理ボリュームの使用済み容量と空き容量を表示します。

```
# pvs -o+pv_used
  PV         VG    Fmt    Attr   PSize      PFree     Used
 /dev/vdb1 myvg  lvm2   a--    1020.00m    0         1020.00m
 /dev/vdb2 myvg  lvm2   a--    1020.00m    0         1020.00m
 /dev/vdb3 myvg  lvm2   a--    1020.00m   1008.00m   12.00m
```

2. データを他の物理ボリュームに移動します。

   a. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にある場合は、以下のコマンドを使用してデータを移動します。

```
# pvmove /dev/vdb3
```

```
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

b. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にない場合は、以下のコマンドを使用して新しい物理ボリュームを追加し、新たに作成した物理ボリュームを使用してボリュームグループを拡張し、この物理ボリュームにデータを移動します。

```
# pvcreate /dev/vdb4
  Physical volume "/dev/vdb4" successfully created

# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended

# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. 物理ボリュームを削除します。

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

論理ボリュームに、障害のある物理ボリュームが含まれる場合は、その論理ボリュームを使用することはできません。見つからない物理ボリュームをボリュームグループから削除します。その物理ボリュームに論理ボリュームが割り当てられていない場合は、**vgreduce** コマンドの **--removemissing** パラメーターを使用できます。

```
# vgreduce --removemissing myvg
```

関連情報

- **pvmove(8)**、**vgextend(8)**、**vereduce(8)**、および **pvs(8)** の man ページ

68.4.7. LVM 論理ボリュームの削除

この手順では、既存の論理ボリューム **/dev/myvg/mylv1** をボリュームグループ **myvg** から削除する方法を説明します。

手順

1. 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。

```
# umount /mnt
```

2. クラスター環境に論理ボリュームが存在する場合は、アクティブになっているすべてのノードで、論理ボリュームを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
# lvchange --activate n vg-name/lv-name
```

3. **lvremove** ユーティリティーを使用して、論理ボリュームを削除します。

```
# lvremove /dev/myvg/mylv1

Do you really want to remove active logical volume "mylv1"? [y/n]: y
Logical volume "mylv1" successfully removed
```

注記

この場合は、論理ボリュームが非アクティブになっていません。削除する前に論理ボリュームを明示的に非アクティブにした場合は、アクティブな論理ボリュームを削除するかどうかを確認するプロンプトが表示されません。

関連情報

- **lvremove(8)** の man ページ

68.4.8. 永続的なデバイス番号の設定

メジャーデバイス番号とマイナーデバイス番号は、モジュールのロード時に動的に割り当てられます。一部のアプリケーションは、ブロックデバイスが常に同じデバイス (メジャーとマイナー) 番号でアクティブにされている場合に、最も効果的に機能します。これらは **lvcreate** と **lvchange** コマンドで、以下の引数を使用することによって指定できます。

```
--persistent y --major major --minor minor
```

別のデバイスにすでに動的に割り当てられている番号を使用しないように、マイナー番号は大きくします。

NFS を使用してファイルシステムをエクスポートする場合は、そのエクスポートファイルで **fsid** パラメーターを指定すると、LVM 内で永続的なデバイス番号を設定する必要がなくなります。

68.4.9. LVM エクステントサイズの指定

ボリュームグループの作成に物理ボリュームが使用されると、ディスク領域はデフォルトで 4MB のエクステントに分割されます。このエクステントは、論理ボリュームのサイズを拡張/縮小する最小単位です。エクステントの数が多くても、論理ボリュームの I/O パフォーマンスに影響を与えることはありません。

エクステントサイズのデフォルト設定が適切でない場合は、**vgcreate** コマンドに **-s** オプションを使用して、エクステントのサイズを指定できます。**vgcreate** コマンドに **-p** 引数と **-l** 引数を使用すると、ボリュームグループに追加可能な物理ボリュームまたは論理ボリュームの数に制限をかけることができます。

68.4.10. RHEL システムロールを使用した LVM 論理ボリュームの管理

storage ロールを使用して、次のタスクを実行します。

- 複数のディスクで設定されるボリュームグループに LVM 論理ボリュームを作成します。
- 論理ボリューム上に特定のラベルを付けて ext4 ファイルシステムを作成します。
- ext4 ファイルシステムを永続的にマウントします。

前提条件

- **storage** ロールを含む Ansible Playbook がある。

68.4.10.1. 論理ボリュームを管理する Ansible Playbook の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、**storage** ロールを適用して、ボリュームグループに LVM 論理ボリュームを作成します。

例68.1 myvg ボリュームグループに mylv 論理ボリュームを作成する Playbook

```
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - sda
          - sdb
          - sdc
        volumes:
          - name: mylv
            size: 2G
            fs_type: ext4
            mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- **myvg** ボリュームグループは、次のディスクで設定されます。
  - **/dev/sda**
  - **/dev/sdb**
  - **/dev/sdc**
- **myvg** ボリュームグループがすでに存在する場合は、Playbook により論理ボリュームがボリュームグループに追加されます。
- **myvg** ボリュームグループが存在しない場合は、Playbook により作成されます。
- Playbook は、**mylv** 論理ボリューム上に Ext4 ファイルシステムを作成し、**/mnt** ファイルシステムを永続的にマウントします。

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** ファイル

68.4.10.2. 関連情報

- 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

68.4.11. LVM ボリュームグループの削除

この手順では、既存のボリュームグループを削除する方法を説明します。

前提条件

- ボリュームグループには論理ボリュームがありません。ボリュームグループから論理ボリュームを削除するには、LVM 論理ボリュームの削除 を参照してください。

手順

1. クラスター環境にボリュームグループが存在する場合は、その他のすべてのノードで、ボリュームグループの **lockspace** を停止します。削除するノード以外の全ノードで、次のコマンドを実行します。

```
# vgchange --lockstop vg-name
```

   ロックが停止するのを待ちます。

2. ボリュームグループを削除します。

```
# vgremove vg-name
  Volume group "vg-name" successfully removed
```

関連情報

- **vgremove(8)** の man ページ

68.5. 論理ボリュームのサイズ変更

論理ボリュームを作成したら、ボリュームのサイズを変更できます。

68.5.1. 論理ボリュームとファイルシステムの拡張

この手順では、論理ボリュームを拡張し、同じ論理ボリューム上のファイルシステムを拡張する方法を説明します。

論理ボリュームのサイズを拡張するには、**lvextend** コマンドを使用します。論理ボリュームを拡張する場合は、追加するボリュームの容量、または拡張後のボリュームのサイズを指定できます。

前提条件

1. ファイルシステムを持つ既存の論理ボリューム (LV) がある。**df -Th** コマンドを使用して、ファイルシステムのタイプを確認します。
   LV およびファイルシステムの作成に関する詳細は、LVM 論理ボリュームの作成 を参照してください。

2. LV およびファイルシステムを拡張するのに十分な領域がボリュームグループにある。**vgs -o name,vgfree** コマンドを使用して、利用可能な領域を確認します。

手順

1. オプション:オプション: ボリュームグループに LV を拡張するのに十分な領域がない場合は、以下のコマンドを使用してボリュームグループに新しい物理ボリュームを追加します。

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

詳細は、LVM ボリュームグループの作成 を参照してください。

2. ボリュームグループが十分に大きくなったので、要件に応じて以下のいずれかの手順を実行します。

   a. 指定されたサイズで LV を拡張するには、次のコマンドを使用します。

```
# lvextend -L 3G /dev/myvg/mylv
Size of logical volume myvg/mylv changed from 2.00 GiB (512 extents) to 3.00 GiB (768 extents).
Logical volume myvg/mylv successfully resized.
```

注記

lvextend コマンドで **-r** オプションを使用すれば、1 つのコマンドで、論理ボリュームを拡張し、基礎となるファイルシステムのサイズを変更できます。

```
# lvextend -r -L 3G /dev/myvg/mylv
```

警告

lvresize コマンドを使用して、同じ引数で論理ボリュームを拡張することもできますが、このコマンドでは、誤って縮小されない保証はありません。

   b. **mylv** 論理ボリュームを拡張して、**myvg** ボリュームグループの未割り当て領域をすべて埋めるには、次のコマンドを使用します。

```
# lvextend -l +100%FREE /dev/myvg/mylv
 Size of logical volume myvg/mylv changed from 10.00 GiB (2560 extents) to 6.35 TiB (1665465 extents).
 Logical volume myvg/mylv successfully resized.
```

   lvcreate コマンドと同様に、**lvextend** コマンドの **-l** 引数を使用して、論理ボリュームの拡張サイズをエクステント数で指定できます。また、この引数を使用してボリュームグループのパーセンテージ、またはボリュームグループに残ってる空き領域をパーセンテージで指定することもできます。

3. **lvextend** コマンドで **r** オプションを使用して 1 つのコマンドで LV を拡張してファイルシステムのサイズを変更しない場合は、以下のコマンドを使用して論理ボリューム上のファイルシステムのサイズを変更します。

```
xfs_growfs /mnt/mnt1/
meta-data=/dev/mapper/myvg-mylv  isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=262144, imaxpct=25
```

```
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 262144 to 524288
```

注記

xfs_growfs は、**-D** オプションを指定しないと、基となるデバイスがサポートする最大サイズまでファイルシステムを拡張します。詳細は Increasing the size of an XFS file system を参照してください。

ext4 ファイルシステムのサイズを変更する場合は、Resizing an ext4 file system を参照してください。

検証

- 以下のコマンドを使用して、ファイルシステムが拡大しているかどうかを確認します。

```
# df -Th
Filesystem           Type      Size  Used Avail Use% Mounted on
devtmpfs             devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                tmpfs     1.9G  8.6M  1.9G   1% /run
tmpfs                tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/rhel-root xfs        45G  3.7G   42G   9% /
/dev/vda1            xfs      1014M  369M  646M  37% /boot
tmpfs                tmpfs     374M     0  374M   0% /run/user/0
/dev/mapper/myvg-mylv xfs       2.0G   47M  2.0G   3% /mnt/mnt1
```

関連情報

- **vgextend(8)**、**lvextend(8)**、および **xfs_growfs(8)** の man ページ

68.5.2. 論理ボリュームの縮小

論理ボリュームのサイズを縮小するには、**lvreduce** コマンドを使用します。

注記

GFS2 および XFS のファイルシステムでは縮小に対応していないため、GFS2 または XFS のファイルシステムが含まれる論理ボリュームのサイズは縮小できません。

縮小する論理ボリュームにファイルシステムが含まれている場合は、データの損失を防ぐため、ファイルシステムが、縮小する論理ボリュームにある領域を使用しないようにしてください。そのため、論理ボリュームにファイルシステムが含まれている場合は **lvreduce** コマンドの **--resizefs** オプションを使用することが推奨されます。

このオプションを使用すると、**lvreduce** コマンドは論理ボリュームを縮小する前にファイルシステムの縮小を試みます。ファイルシステムの縮小に失敗した場合 (ファイルシステムが満杯であったり、ファイルシステムが縮小をサポートしない場合に失敗します)、**lvreduce** コマンドの実行に失敗し、論理ボリュームを縮小しません。

警告

ほとんどの場合、**lvreduce** コマンドはデータ損失の可能性を警告し、確認を要求します。しかし、論理ボリュームが非アクティブな状態であったり、**--resizefs** オプションが使用されなかった場合など、警告が表示されない場合があるため、データの損失を防ぐのに確認プロンプトのみを信頼しないようにしてください。

lvreduce コマンドの **--test** オプションは、ファイルシステムの確認やファイルシステムのサイズ変更のテストを行わないため、操作が安全な場所を示しません。

手順

- **myvg** ボリュームグループの **mylv** 論理ボリュームを 64 メガバイトに縮小するには、次のコマンドを使用します。

```
# lvreduce --resizefs -L 64M myvg/mylv
fsck from util-linux 2.37.2
/dev/mapper/myvg-mylv: clean, 11/25688 files, 4800/102400 blocks
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mapper/myvg-mylv to 65536 (1k) blocks.
The filesystem on /dev/mapper/myvg-mylv is now 65536 (1k) blocks long.

Size of logical volume myvg/mylv changed from 100.00 MiB (25 extents) to 64.00 MiB (16 extents).
 Logical volume myvg/mylv successfully resized.
```

  この例では、**mylv** にはファイルシステムが含まれ、このコマンドによって論理ボリュームとともにサイズが変更されます。

- サイズ変更値の前に **-** 記号を指定すると、その値が論理ボリュームの実際のサイズから減算されます。論理ボリュームを絶対サイズ 64 メガバイトに縮小するには、次のコマンドを使用します。

```
# lvreduce --resizefs -L -64M myvg/mylv
```

関連情報

- **lvreduce(8)** の man ページ

68.5.3. ストライプ化論理ボリュームの拡張

ストライプ化論理ボリュームのサイズを拡大するには、ボリュームグループを設定している物理ボリュームに、ストライプをサポートする十分な空き領域が必要です。たとえば、ボリュームグループ全域を使用する 2 way ストライプがある場合は、ボリュームグループに物理ボリュームを 1 つ追加しただけでは、ストライプを拡張することはできません。ボリュームグループには物理ボリュームを 2 つ以上追加する必要があります。

たとえば、以下の **vgs** コマンドで表示された、2 つの物理ボリュームで設定されるボリュームグループ **vg** について考えてみましょう。

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   0   0 wz--n- 271.31G 271.31G
```

ボリュームグループの全領域を使用して、ストライプを作成できます。

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
  Using default stripesize 64.00 KB
  Rounding up size to full physical extent 271.31 GB
  Logical volume "stripe1" created
# lvs -a -o +devices
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe1 vg   -wi-a- 271.31G                               /dev/sda1(0),/dev/sdb1(0)
```

ボリュームグループの空き領域がなくなっていることに注意してください。

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   1   0 wz--n- 271.31G    0
```

以下のコマンドで、ボリュームグループに物理ボリュームをもう 1 つ追加します。 これで、135 ギガバイトの領域が追加されます。

```
# vgextend vg /dev/sdc1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     3   1   0 wz--n- 406.97G 135.66G
```

この時点では、ストライプ化論理ボリュームを、ボリュームグループの最大サイズまで拡大することはできません。データをストライプ化するには、基になる物理デバイスが 2 つ必要です。

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
```

ストライプ化論理ボリュームを拡張するには、もう 1 つの物理ボリュームを追加してから、論理ボリュームを拡張します。この例では、ボリュームグループに物理ボリュームを 2 つ追加することにより、ボリュームグループの最大サイズまで、論理ボリュームを拡張できるようになっています。

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 542.00 GB
  Logical volume stripe1 successfully resized
```

ストライプ化論理ボリュームを拡張するのに十分な物理デバイスがない場合でも、その拡張部分がストライプ化されなくても問題がないならば、ボリュームの拡張は可能です。ただし、これによりパフォー

マンスが一定ではなくなる可能性があります。論理ボリュームに領域を追加する場合、デフォルトの動作では、既存の論理ボリュームの最後のセグメントと同じストライピングパラメーターを使用するようになっていますが、このパラメーターはオーバーライドできます。以下の例では、初回の **lvextend** コマンドが失敗した後に、既存のストライプ化論理ボリュームを拡張して残りの空き領域を使用するようにしています。

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

68.6. LVM 用のカスタム報告

LVM では、カスタマイズされたレポートを生成したり、レポートの出力をフィルタリングしたりするための様々な設定およびコマンドラインオプションが提供されます。LVM レポート機能の完全な説明は、man ページの **lvmreport**(7) を参照してください。

pvs、**lvs**、および **vgs** コマンドを使用して、LVM オブジェクトについての簡潔でカスタマイズ可能なレポートを作成することができます。このコマンドが生成するレポートには、オブジェクトごとに 1 行の出力が含まれます。各行には、オブジェクトに関連するプロパティーのフィールドについて、順序付けられたリストが含まれます。レポートするオブジェクトを選択する方法には、物理ボリューム別、ボリュームグループ別、論理ボリューム別、物理ボリュームセグメント別、および論理ボリュームセグメント別の 5 つの方法があります。

lvm fullreport コマンドを使用して、物理ボリューム、ボリュームグループ、論理ボリューム、物理ボリュームセグメント、および論理ボリュームセグメントに関する情報を一度に報告できます。このコマンドとその機能については、**lvm-fullreport**(8) の man ページを参照してください。

LVM は、LVM コマンドの実行中に収集された操作、メッセージ、および各オブジェクトのステータス (完全なオブジェクト ID 付き) のログが含まれるログレポートをサポートします。LVM ログレポートの詳細は、man ページの **lvmreport**(7) を参照してください。

68.6.1. LVM 表示の形式の制御

コマンドの **pvs**、**lvs**、または **vgs** のどれを使用するかによって、表示されるデフォルトのフィールドセットとソート順序が決定します。このコマンドの出力は、以下の引数を使用して制御できます。

- **-o** 引数を使用すると、表示するフィールドをデフォルト以外に変更できます。たとえば、以下のコマンドは、物理ボリュームの名前とサイズのみを表示します。

  ```
  # pvs -o pv_name,pv_size
  PV PSize
  /dev/sdb1 17.14G
  /dev/sdc1 17.14G
  /dev/sdd1 17.14G
  ```

- -o 引数との組み合わせで使用するプラス記号 (+) を使用して、出力にフィールドを追加できます。
  以下の例は、デフォルトフィールドに加えて、物理ボリュームの UUID を表示しています。

  ```
  # pvs -o +pv_uuid
  PV VG Fmt Attr PSize PFree PV UUID
  ```

```
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-UqkCS
```

- コマンドに **-v** 引数を追加すると、追加のフィールドが含まれます。たとえば、**pvs -v** コマンドは、デフォルトフィールドに加えて、**DevSize** と **PV UUID** のフィールドも表示します。

```
# pvs -v
Scanning for physical volume names
PV VG Fmt Attr PSize PFree DevSize PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G 17.14G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-
mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-
tUqkCS
```

- **--noheadings** 引数は、見出し行を表示しません。これはスクリプトを作成する際に便利です。以下の例は、**pv_name** 引数と共に **--noheadings** 引数を使用して、すべての物理ボリュームのリストを生成しています。

```
# pvs --noheadings -o pv_name
/dev/sdb1
/dev/sdc1
/dev/sdd1
```

- **--separator separator** 引数は、**区切り文字** を使用して、各フィールドを区切ります。次の例は、**pvs** コマンドのデフォルト出力フィールドを等号 (=) で分割しています。

```
# pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
/dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
/dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

  separator 引数の使用時にフィールドを配置するには、**--aligned** 引数とともに **separator** 引数を使用します。

```
# pvs --separator = --aligned
PV =VG =Fmt =Attr=PSize =PFree
/dev/sdb1 =new_vg=lvm2=a- =17.14G=17.14G
/dev/sdc1 =new_vg=lvm2=a- =17.14G=17.09G
/dev/sdd1 =new_vg=lvm2=a- =17.14G=17.14G
```

lvs コマンドまたは **vgs** コマンドの **-P** 引数を使用して、通常の出力では表示されない、障害が発生したボリュームの情報を表示します

表示引数のリストは、man ページの **pvs**(8)、**vgs**(8)、および **lvs**(8) を参照してください。

ボリュームグループフィールドは、物理ボリューム (および物理ボリュームセグメント) フィールド、または論理ボリューム (および論理ボリュームセグメント) フィールドと混在させることができますが、物理ボリュームフィールドと論理ボリュームフィールドは混在させることができません。たとえば、以下のコマンドは、1つの物理ボリュームつき1行の出力を表示します。

```
# vgs -o +pv_name
  VG     #PV #LV #SN Attr   VSize  VFree  PV
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdc1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdd1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdb1
```

68.6.2. LVM オブジェクト表示フィールド

pvs、**vgs**、および **lvs** コマンドを使用して、LVM オブジェクトに関する追加情報を表示できます。

フィールド名の接頭辞は、コマンドのデフォルトと一致する場合は省略できます。たとえば、**pvs** コマンドでは、**name** は **pv_name**、**vgs** コマンドでは、**name** は **vg_name** と解釈されます。

以下のコマンドの実行は、**pvs -o pv_free** の実行に相当します。

```
# pvs -o free
  PFree
  17.14G
  17.09G
  17.14G
```

注記

pvs、**vgs**、および **lvs** 出力の属性フィールドにある文字数は、以降のリリースで増える可能性があります。既存の文字フィールドの位置は変更しませんが、新しいフィールドが末尾に追加される可能性があります。相対的な位置を使用して特定の属性文字を検索するスクリプトを作成する場合は、このことを考慮して、フィールドの終点ではなく、フィールドの始点を基点として文字検索を行います。たとえば、**lv_attr** フィールドの 9 番目のビットの文字 **p** を検索する場合は、文字列^/.....p/で指定できます。/*p$/は使用しないでください。

表68.1「pvs コマンド表示フィールド」 は、**pvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、フィールドの説明を一覧にまとめています。

表68.1 pvs コマンド表示フィールド

引数	ヘッダー	説明
dev_size	DevSize	物理ボリュームを作成する基となるデバイスのサイズ
pe_start	1st PE	基となるデバイス内の最初の物理エクステントの開始点までのオフセット
pv_attr	Attr	物理ボリュームのステータス - (a)llocatable または e(x)ported
pv_fmt	Fmt	物理ボリュームのメタデータ形式 (**lvm2** または **lvm1**)
pv_free	PFree	物理ボリュームにある残りの空き領域
pv_name	PV	物理ボリュームの名前

引数	ヘッダー	説明
pv_pe_alloc_count	Alloc	使用される物理エクステントの数
pv_pe_count	PE	物理エクステントの数
pvseg_size	SSize	物理ボリュームのセグメントサイズ
pvseg_start	Start	物理ボリュームセグメントの最初の物理エクステント
pv_size	PSize	物理ボリュームのサイズ
pv_tags	PV Tags	物理ボリュームに割り当てられた LVM タグ
pv_used	Used	物理ボリュームで現在使用中の領域の量
pv_uuid	PV UUID	物理ボリュームの UUID

デフォルトでは、**pvs** コマンドは **pv_name**、**vg_name**、**pv_fmt**、**pv_attr**、**pv_size**、および **pv_free** フィールドを表示します。この表示は、**pv_name** でソートされています。

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G
```

pvs コマンドに **-v** 引数を使用すると、デフォルトの表示に、**dev_size** フィールドおよび **pv_uuid** フィールドが追加されます。

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

pvs コマンドに **--segments** 引数を使用すると、各物理ボリュームセグメントの情報を表示します。セグメントはエクステントの集合です。セグメントの表示は、論理ボリュームがフラグメント化 (断片化) しているかどうかを確認するのに役立ちます。

デフォルトで **pvs --segments** コマンドが表示するフィールドは、**pv_name**、**vg_name**、**pv_fmt**、**pv_attr**、**pv_size**、**pv_free**、**pvseg_start**、および **pvseg_size** です。この表示は、物理ボリューム内では **pv_name** および **pvseg_size** でソートされています。

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  Start SSize
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M     0  1172
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1172    16
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1188     1
```

```
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G    0   26
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G   26   24
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G   50   26
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G   76   24
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G  100   26
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G  126   24
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G  150   22
  /dev/sda1  vg       lvm2 a-   17.14G 16.75G  172  4217
  /dev/sdb1  vg       lvm2 a-   17.14G 17.14G    0  4389
  /dev/sdc1  vg       lvm2 a-   17.14G 17.14G    0  4389
  /dev/sdd1  vg       lvm2 a-   17.14G 17.14G    0  4389
  /dev/sde1  vg       lvm2 a-   17.14G 17.14G    0  4389
  /dev/sdf1  vg       lvm2 a-   17.14G 17.14G    0  4389
  /dev/sdg1  vg       lvm2 a-   17.14G 17.14G    0  4389
```

pvs -a コマンドを使用して、LVM が検出した、LVM 物理ボリュームとして初期化していないデバイスを確認できます。

```
# pvs -a
  PV                         VG     Fmt  Attr PSize  PFree
  /dev/VolGroup00/LogVol01                --       0      0
  /dev/new_vg/lvol0                       --       0      0
  /dev/ram                                --       0      0
  /dev/ram0                               --       0      0
  /dev/ram2                               --       0      0
  /dev/ram3                               --       0      0
  /dev/ram4                               --       0      0
  /dev/ram5                               --       0      0
  /dev/ram6                               --       0      0
  /dev/root                               --       0      0
  /dev/sda                                --       0      0
  /dev/sdb                                --       0      0
  /dev/sdb1                  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc                                --       0      0
  /dev/sdc1                  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd                                --       0      0
  /dev/sdd1                  new_vg lvm2 a-   17.14G 17.14G
```

表68.2「vgs 表示フィールド」 は、**vgs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。

表68.2 vgs 表示フィールド

引数	ヘッダー	説明
lv_count	#LV	ボリュームグループに含まれる論理ボリュームの数
max_lv	MaxLV	ボリュームグループで許容される論理ボリュームの最大数 (無制限の場合は 0)
max_pv	MaxPV	ボリュームグループで許容される物理ボリュームの最大数 (無制限の場合は 0)

引数	ヘッダー	説明
pv_count	#PV	ボリュームグループを定義する物理ボリューム数
snap_count	#SN	ボリュームグループに含まれるスナップショット数
vg_attr	Attr	ボリュームグループのステータス - (w)riteable (書き込み可能)、(r)eadonly (読み取りのみ)、resi(z)eable (サイズ変更可能)、e(x)ported (エクスポート済)、(p)artial (部分的)、および (c)lustered (クラスター化)
vg_extent_count	#Ext	ボリュームグループの物理エクステントの数
vg_extent_size	Ext	ボリュームグループの物理エクステントのサイズ
vg_fmt	Fmt	ボリュームグループのメタデータ形式 (**lvm2** または **lvm1**)
vg_free	VFree	ボリュームグループの残りの空き領域のサイズ
vg_free_count	Free	ボリュームグループの空き物理エクステントの数
vg_name	VG	ボリュームグループ名
vg_seqno	Seq	ボリュームグループの改訂を示す番号
vg_size	VSize	ボリュームグループのサイズ
vg_sysid	SYS ID	LVM1 システム ID
vg_tags	VG Tags	ボリュームグループに割り当てられた LVM タグ
vg_uuid	VG UUID	ボリュームグループの UUID

デフォルトで **vgs** コマンドが表示するフィールドは、**vg_name**、**pv_count**、**lv_count**、**snap_count**、**vg_attr**、**vg_size**、および **vg_free** です。この表示は、**vg_name** でソートされています。

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  new_vg   3   1   1 wz--n- 51.42G 51.36G
```

vgs コマンドに **-v** 引数を使用すると、デフォルトの表示に **vg_extent_size** および **vg_uuid** フィールドが追加されます。

```
# vgs -v
    Finding all volume groups
    Finding volume group "new_vg"
  VG     Attr   Ext   #PV #LV #SN VSize  VFree  VG UUID
  new_vg wz--n- 4.00M   3   1   1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

表68.3「lvs 表示フィールド」 は、**lvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。

注記

Red Hat Enterprise Linux の最近のリリースでは、**lvs** コマンドの出力にフィールドが追加されている場合があります。ただし、フィールドの順序は同じで、追加のフィールドは出力の最後に表示されます。

表68.3 lvs 表示フィールド

引数	ヘッダー	説明
* **chunksize** * **chunk_size**	Chunk	スナップショットボリュームのユニットサイズ
copy_percent	Copy%	ミラー化論理ボリュームの同期のパーセンテージ。さらに **pv_move** コマンドで物理エクステントを移動する時にも使用されます。
devices	Devices	論理ボリュームを設定するデバイス - 物理ボリューム、論理ボリューム、および物理エクステントと論理エクステントの開始点
lv_ancestors	Ancestors	シンプールスナップショットにおける、論理ボリュームの先祖 (ancestor)
lv_descendants	Descendants	シンプールスナップショットにおける、論理ボリュームの子孫 (descendant)
lv_attr	Attr	論理ボリュームのステータス。論理ボリュームの属性ビットは以下のようになります。 * ビット 1:ボリュームタイプ - (m)irrored (ミラー化)、(M)irrored without initial sync (初期同期なしのミラー化)、(o)rigin (作成元)、(O)rigin with merging snapshot (マージするスナップショットがある作成元)、(r)aid (RAID)、(R)aid without initial sync (初期同期なしの RAID)、(s)napshot (スナップショット)、merging (S)napshot (マージするスナップショット)、(p)vmove (物理ボリュームの移動)、(v)irtual (仮想)、mirror or raid (i)mage (ミラーまたは RAID イメージ)、mirror or raid (I)mage out-of-sync (ミラーまたは RAID イメージの非同期)、mirror (l)og device (ミラーログデバイス)、under (c)onversion (変換中)、thin (V)olume (シンボリューム)、(t)hin pool (シンプール)、(T)hin pool data (シンプールデータ)、raid or thin pool m(e)tadata or pool metadata spare (RAID またはシンプールメタデータもしくはプールメタデータのスペア) * ビット 2:パーミッション - (w)riteable (書き込み可能)、(r)ead-only (読み取り専用)、(R)ead-only activation of non-read-only volume (読み取り専用でないボリュームを読み取り専用にアクティブ化)

引数	ヘッダー	説明
		* ビット 3:割り当てポリシー - (a)nywhere (どこでも)、(c)ontiguous (連続的)、(i)nherited (継承)、c(l)ing (膠着)、(n)ormal (通常)。これは、たとえば **pvmove** コマンドの実行時など、割り当ての変更に対してボリュームが現在ロックされている場合に大文字になります。 * ビット 4 - 固定されたマイナー番号 * ビット 5:ステータス - (a)ctive (アクティブ)、(s)uspended (サスペンド)、(I)nvalid snapshot (無効なスナップショット)、invalid (S)uspended snapshot (無効なサスペンドされたスナップショット)、snapshot (m)erge failed (スナップショットのマージが失敗)、suspended snapshot (M)erge failed (サスペンドされたスナップショットのマージが失敗)、mapped (d)evice present without tables (テーブルのないマッピングされたデバイス)、mapped device present with (i)nactive table (非アクティブのテーブルを持つマッピングされたデバイス) * ビット 6 - デバイス開放 (o) * ビット 7:ターゲットタイプ - (m)irror (ミラー)、(r)aid (RAID)、(s)napshot (スナップショット)、(t)hin (シン)、(u)nknown (不明)、(v)irtual (仮想)。これは、同じカーネルターゲットに関連する論理ボリュームをまとめます。たとえば、ミラーイメージ、ミラーログ、ミラー自体が、元のデバイスマッパーのミラーカーネルドライバーを使用する場合は、(m) と表示されます。md raid カーネルドライバーを使用する同等の RAID はすべて (r) と表示されます。元のデバイスマッパードライバーを使用するスナップショットは (s) と表示され、シンプロビジョニングドライバーを使用するシンボリュームのスナップショットは (t) と表示されます。 * ビット 8:新しく割り当てられたデータブロックは使用前に、ゼロ (z) のブロックで上書きされます。 * ビット 9:ボリュームの正常性 - (p)artial (部分的)、(r)efresh needed (更新が必要)、(m)ismatches exist (不一致が存在)、(w)ritemostly (書き込み多発)。部分的 (p) は、この論理ボリュームが使用する 1 つ以上の物理ボリュームがシステムから欠落していることを表します。更新 (r) は、この RAID 論理ボリュームが使用する 1 つ以上の物理ボリュームで書き込みエラーが発生したことを表します。書き込みエラーは、その物理ボリュームの一時的な障害により引き起こされたか、物理ボリュームに障害があることを示すかのいずれかの可能性があります。デバイスは更新するか、置き換える必要があります。不一致 (m) は、RAID 論理ボリュームのアレイに一貫していない部分があることを表します。不整合は、RAID 論理ボリュームで **check** 操作を開始すると検出されます。(スクラビング操作 **check** および **repair** は、**lvchange** コマンドにより RAID 論理ボリューム上で実行できます。) 書き込み多発 (w) は、write-mostly とマークが付けられた RAID 1 論理ボリュームのデバイスを表します。 * ビット 10 - s(k)ip activation (アクティブ化のスキップ - このボリュームには、アクティブ化の実行時にスキップされるようにフラグが設定されます。

引数	ヘッダー	説明
lv_kernel_major	KMaj	論理ボリュームの実際のメジャーデバイス番号 (非アクティブの場合は -1)
lv_kernel_minor	KMIN	論理ボリュームの実際のマイナーデバイス番号 (非アクティブの場合は -1)
lv_major	Maj	論理ボリュームの永続的なメジャーデバイス番号 (未指定の場合は -1)
lv_minor	Min	論理ボリュームの永続的なマイナーデバイス番号 (未指定の場合は -1)
lv_name	LV	論理ボリュームの名前
lv_size	LSize	論理ボリュームのサイズ
lv_tags	LV Tags	論理ボリュームに割り当てられた LVM タグ
lv_uuid	LV UUID	論理ボリュームの UUID
mirror_log	Log	ミラーログが存在するデバイス
modules	Modules	この論理ボリュームを使用するのに必要な、対応するカーネルデバイスマッパーターゲット
move_pv	Move	**pvmove** コマンドで作成された一時的な論理ボリュームの元となる物理ボリューム
origin	Origin	スナップショットボリュームの作成元のデバイス
* **regionsize** * **region_size**	Region	ミラー化論理ボリュームのユニットサイズ
seg_count	#Seg	論理ボリュームのセグメント数
seg_size	SSize	論理ボリュームのセグメントサイズ
seg_start	Start	論理ボリュームのセグメントのオフセット
seg_tags	Seg Tags	論理ボリュームのセグメントに割り当てられた LVM タグ
segtype	タイプ	論理ボリュームのセグメントタイプ (例: ミラー、ストライプ、リニア)
snap_percent	Snap%	使用中スナップショットボリュームの現在のパーセンテージ

引数	ヘッダー	説明
stripes	#Str	論理ボリュームのストライプ、またはミラーの数
* **stripesize** * **stripe_size**	Stripe	ストライプ化論理ボリュームのストライプのユニットサイズ

デフォルトで **lvs** コマンドが表示するのは以下になります。デフォルトの表示は、ボリュームグループ内では **vg_name** および **lv_name** でソートされます。

```
# lvs
  LV     VG            Attr       LSize    Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  origin VG            owi-a-s---     1.00g
  snap   VG            swi-a-s---  100.00m      origin 0.00
```

lvs コマンドの一般的な用途は、論理ボリュームを設定するデバイスを表示するコマンドに、**devices** を追加することです。また、この例では、**-a** オプションを指定して、RAID ミラーなどの論理ボリュームのコンポーネントである内部ボリュームを、括弧で囲んで表示します。この例には、RAID ボリューム、ストライプのボリューム、シンプールのボリュームが含まれます。

```
# lvs -a -o +devices
  LV              VG           Attr       LSize   Pool   Origin Data%  Meta%  Move Log Cpy%Sync Convert
Devices
  raid1           VG           rwi-a-r---   1.00g                                    100.00
raid1_rimage_0(0),raid1_rimage_1(0)
  [raid1_rimage_0] VG          iwi-aor---   1.00g                                                 /dev/sde1(7041)
  [raid1_rimage_1] VG          iwi-aor---   1.00g                                                 /dev/sdf1(7041)
  [raid1_rmeta_0]  VG          ewi-aor---   4.00m                                                  /dev/sde1(7040)
  [raid1_rmeta_1]  VG          ewi-aor---   4.00m                                                  /dev/sdf1(7040)
  stripe1         VG           -wi-a-----  99.95g                                                /dev/sde1(0),/dev/sdf1(0)
  stripe1         VG           -wi-a-----  99.95g                                                /dev/sdd1(0)
  stripe1         VG           -wi-a-----  99.95g                                                /dev/sdc1(0)
  [lvol0_pmspare]  rhel_host-083 ewi-------   4.00m                                                      /dev/vda2(0)
  pool00          rhel_host-083 twi-aotz--  <4.79g              72.90  54.69
pool00_tdata(0)
  [pool00_tdata]   rhel_host-083 Twi-ao----  <4.79g                                                     /dev/vda2(1)
  [pool00_tmeta]   rhel_host-083 ewi-ao----   4.00m                                                     /dev/vda2(1226)
  root            rhel_host-083 Vwi-aotz--  <4.79g pool00        72.90
  swap            rhel_host-083 -wi-ao---- 820.00m                                                    /dev/vda2(1227)
```

lvs コマンドで **-v** 引数を使用して、デフォルトの表示に、**seg_count**、**lv_major**、**lv_minor**、**lv_kernel_major**、**lv_kernel_minor**、**lv_uuid** のフィールドを追加します。

```
# lvs -v
    Finding all logical volumes
  LV        VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%  Move Copy%  Log Convert LV
UUID
  lvol0     new_vg    1 owi-a- 52.00M  -1  -1 253  3                                       LBy1Tz-sr23-OjsI-LT03-
nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20                   1ye1OU-1clu-
o79k-20h2-ZGF0-qCJm-CfbsIx
```

lvs コマンドの **--segments** 引数を使用して、セグメント情報を強調するデフォルトの列で情報を表示できます。**segments** 引数を使用する場合、**seg** 接頭辞は必要に応じて使用します。デフォルトで **lvs --segments** コマンドが表示するフィールドは、**lv_name**、**vg_name**、**lv_attr**、**stripes**、**segtype**、**seg_size** です。デフォルトの表示は、ボリュームグループ内では **vg_name**、**lv_name** でソートされ、論理ボリュームでは **seg_start** でソートされます。論理ボリュームが断片化されると、このコマンドの出力が表示されます。

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    1 linear 512.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear  88.00M
```

lvs --segments コマンドで **-v** 引数を使用すると、デフォルトの表示に **seg_start**、**stripsize**、および **chunksize** フィールドが追加されます。

```
# lvs -v --segments
    Finding all logical volumes
  LV         VG     Attr   Start SSize  #Str Type   Stripe Chunk
  lvol0      new_vg owi-a-    0  52.00M    1 linear     0     0
  newvgsnap1 new_vg swi-a-    0   8.00M    1 linear     0  8.00K
```

以下の1つ目の例は、設定された論理ボリュームが1つあるシステムで実行した **lvs** コマンドのデフォルト出力を示しています。その次の例は、**segments** 引数を指定した **lvs** コマンドのデフォルト出力を表示しています。

```
# lvs
  LV    VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 new_vg -wi-a- 52.00M
# lvs --segments
  LV    VG     Attr   #Str Type   SSize
  lvol0 new_vg -wi-a-    1 linear 52.00M
```

68.6.3. LVM 報告のソート

通常、**lvs** コマンド、**vgs** コマンド、または **pvs** コマンドの出力全体をソートして、コラムを正しく配置する場合は、出力を生成して内部に保管する必要があります。**--unbuffered** 引数を指定すると、生成直後にソートされていないままの出力で表示できます。

別の順列のコラムリストのソートを指定するには、報告コマンドのいずれかと一緒に **-O** 引数を使用します。出力自体に、このフィールドを含める必要はありません。

以下の例は、物理ボリュームの名前、サイズ、および空き領域を表示する **pvs** コマンドの出力を示しています。

```
# pvs -o pv_name,pv_size,pv_free
  PV         PSize  PFree
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
```

以下の例では、空き領域のフィールドでソートされた同じ出力を示しています。

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
  PV         PSize  PFree
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
```

以下の例では、ソートするフィールドを表示する必要がないことを示しています。

```
# pvs -o pv_name,pv_size -O pv_free
  PV         PSize
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
  /dev/sdb1  17.14G
```

逆順でソートするには、**-O** 引数の後で指定するフィールドの先頭に **-** 印を付けます。

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
  PV         PSize  PFree
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
```

68.6.4. LVM レポート表示への単位の指定

LVM 報告表示用の単位を指定するには、報告コマンドに **--units** 引数を使用します。

ベース 2 ユニット

2 の累乗 (1024 の倍数) で表示されるデフォルトの単位。以下を指定できます。

- 人間が判読できる (**r**) < 丸めインジケータ付き
- バイト (**b**)
- セクター (**秒**)
- キロバイト (**k**)
- メガバイト (**m**)
- ギガバイト (**g**)
- テラバイト (**t**)
- ペタバイト (**p**)
- エクサバイト (**e**)
- デフォルトの単位である人間が読める形式 (**h**)

デフォルトの表示は **r** で、人間が判読できます。このデフォルト設定を上書きするには、**/etc/lvm/lvm.conf** ファイルの global セクションに units パラメーターを設定します。

基本 10 単位

単位指定 (**R**、**B**、**S**、**K**、**M**、**G**、**T**、**P**、**E**、**H**) を大文字にすることで、表示する単位を 1000 の倍数で指定できます。

次の例では、**pvs**、**vgs**、および **lvs** コマンドの出力を基数 2 のギガバイト単位で指定します。

```
# pvs --units g /dev/sdb
  PV         VG    Fmt  Attr PSize   PFree
  /dev/sdb  test  lvm2 a--  931.00g 930.00g
```

```
# vgs --units g test
  VG   #PV #LV #SN Attr VSize   VFree
  test   1   1   0 wz-n 931.00g 931.00g
```

```
# lvs --units g test
  LV    VG   Attr     LSize Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lvol0 test wi-a---- 1.OOg
```

次の例では、**pvs**、**vgs**、および **lvs** コマンドの出力をベース 10 ギガバイト単位で指定します。

```
# pvs --units G /dev/sdb
  PV         VG   Fmt  Attr  PSize   PFree
  /dev/sdb  test lvm2 a--   999.65G 998.58G
```

```
# vgs --units G test
  VG   #PV #LV #SN Attr VSize   VFree
  test   1   1   0 wz-n 999.65G 998.58G
```

```
# lvs --units G test
  LV    VG   Attr     LSize Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lvol0 test wi-a---- 1.07G
```

512 バイトとして定義されたセクター (**s**) またはカスタム単位を指定できます。次の例は、**pvs** コマンドの出力を複数のセクターとして表示します。

```
# pvs --units s
  PV         VG     Fmt  Attr PSize       PFree
  /dev/sdb   test   lvm2 a--  1952440320S 1950343168S
```

以下の例は、**pvs** コマンドの出力を 4 MB 単位で表示しています。

```
# pvs --units 4m
  PV         VG     Fmt  Attr PSize      PFree
  /dev/sdb   test   lvm2 a--  238335.00U 238079.00U
```

r 単位の目的は、**h** (人間が読める形式) と同様に機能することですが、さらに、報告される値に **<** または **>** の接頭辞を付けて、実際のサイズが表示サイズよりわずかに大きいまたは小さいことを示します。**r** 設定は、LVM コマンドのデフォルトです。LVM は 10 進数値を四捨五入するため、正確でないサイズが報告されます。次の点に注意してください。

```
# vgs --units g test
  VG   #PV #LV #SN Attr VSize   VFree
  test   1   1   0 wz-n 931.00g 930.00g
```

```
# vgs --units r test
  VG   #PV #LV #SN Attr VSize    VFree
  test   1   1   0 wz-n <931.00g <930.00
```

```
# vgs test
  VG   #PV #LV #SN Attr VSize    VFree
  test   1   1   0 wz-n <931.00g <930.00g
```

--units が指定されていない場合は、**r** がデフォルトの単位であることに注意してください。また、**--units g** (または他の **--units**) が常に正確なサイズを表示するとは限らないことも示しています。また、表示されたサイズが正確でないことを示す **<** である **r** の主な目的も示しています。この例では、VG サイズがギガバイトの正確な倍数ではなく、.01 も分数の正確な表現ではないため、値は正確ではありません。

68.6.5. JSON 形式で LVM コマンド結果の表示

LVM 表示コマンドで **--reportformat** オプションを使用して、JSON 形式で出力を表示できます。

以下の例では、標準的なデフォルト形式の **lvs** の出力を示しています。

```
# lvs
  LV      VG            Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  my_raid my_vg         Rwi-a-r---  12.00m                                    100.00
  root    rhel_host-075 -wi-ao----   6.67g
  swap    rhel_host-075 -wi-ao---- 820.00m
```

以下のコマンドは、JSON 形式を指定する場合と同じ LVM 設定の出力を表示します。

```
# lvs --reportformat json
  {
      "report": [
          {
              "lv": [
                  {"lv_name":"my_raid", "vg_name":"my_vg", "lv_attr":"Rwi-a-r---", "lv_size":"12.00m",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"100.00", "convert_lv":""},
                  {"lv_name":"root", "vg_name":"rhel_host-075", "lv_attr":"-wi-ao----", "lv_size":"6.67g",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"", "convert_lv":""},
                  {"lv_name":"swap", "vg_name":"rhel_host-075", "lv_attr":"-wi-ao----", "lv_size":"820.00m",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"", "convert_lv":""}
              ]
          }
      ]
  }
```

また、**/etc/lvm/lvm.conf** ファイルで **output_format** 設定を使用して、レポート形式を設定オプションとして設定することもできます。ただし、コマンドラインの **--reportformat** 設定は、この設定よりも優先されます。

68.6.6. LVM コマンドログの表示

レポート指向および処理指向の LVM コマンドを使用して、コマンドログを報告できます (これが **log/report_command_log** 設定で有効になっている場合)。このレポートで表示およびソートするフィールドセットを決定できます。

以下の例では、LVM コマンド向けの完全なログレポートを生成するように LVM を設定します。この例では、論理ボリューム **lvol0** と **lvol1** の両方が、それらの論理ボリュームを含むボリュームグループ **VG** とともに正常に処理されたことを確認できます。

```
# lvmconfig --type full log/command_log_selection
command_log_selection="all"

# lvs
  Logical Volume
  ==============
  LV    LSize Cpy%Sync
  lvol1 4.00m 100.00
  lvol0 4.00m

  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1
    3 status  processing vg      vg              success     0       1

# lvchange -an vg/lvol1
  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol1   vg      success     0       1
    2 status  processing vg      vg              success     0       1
```

LVM レポートおよびコマンドログの設定の詳細は、man ページの **lvmreport** を参照してください。

68.7. RAID 論理ボリュームの設定

VM Redundant Array of Independent Disks (RAID) ボリュームの作成、有効化、変更、削除、表示、および使用が可能です。

68.7.1. RAID 論理ボリューム

論理ボリュームマネージャー (LVM) は、Redundant Array of Independent Disks (RAID) レベル 0、1、4、5、6、10 をサポートします。LVM RAID ボリュームには以下の特徴があります。

- LVM は、Multiple Devices (MD) カーネルドライバーを活用した RAID 論理ボリュームを作成して管理する
- アレイから RAID1 イメージを一時的に分割し、後でアレイにマージし直すことが可能
- LVM RAID ボリュームはスナップショットに対応

その他にも、以下のような特徴があります。

クラスター

RAID 論理ボリュームはクラスターには対応していません。
RAID 論理ボリュームは 1 台のマシンに排他的に作成およびアクティブ化できますが、複数のマシンで同時にアクティブにすることはできません。

Subvolumes

RAID 論理ボリューム (LV) を作成するとき、LVM は、データまたはアレイ内のパリティーサブボリュームごとに、サイズが 1 エクステントのメタデータサブボリュームを作成します。
たとえば、2 方向の RAID1 アレイを作成すると、メタデータサブボリュームが 2 つ (**lv_rmeta_0** および **lv_rmeta_1**) と、データサブボリュームが 2 つ (**lv_rimage_0** および **lv_rimage_1**) 作成されます。同様に、3 方向ストライプ (および暗黙的なパリティーデバイスが 1 つ) の RAID4 を作成すると、メタデータサブボリュームが 4 つ (**lv_rmeta_0**、**lv_rmeta_1**、**lv_rmeta_2**、**lv_rmeta_3**)、データサブボリュームが 4 つ (**lv_rimage_0**、**lv_rimage_1**、**lv_rimage_2**、**lv_rimage_3**) 作成されます。

インテグリティー

RAID デバイスに障害が発生したり、ソフト破損が発生したときにデータが失われる場合があります。データストレージにおけるソフト破損は、ストレージデバイスから取得したデータが、そのデバイスに書き込まれるデータとは異なることを意味します。RAID LV に整合性を追加すると、ソフト破損が軽減または防止します。詳しくは、DM 整合性を備えた RAID LV の作成 を参照してください。

68.7.2. RAID レベルとリニアサポート

レベル 0、1、4、5、6、10、リニアなど、RAID 別の対応設定は以下のとおりです。

レベル 0

ストライピングとも呼ばれる RAID レベル 0 は、パフォーマンス指向のストライピングデータマッピング技術です。これは、アレイに書き込まれるデータがストライプに分割され、アレイのメンバーディスク全体に書き込まれることを意味します。これにより低い固有コストで高い I/O パフォーマンスを実現できますが、冗長性は提供されません。
RAID レベル 0 実装は、アレイ内の最小デバイスのサイズまで、メンバーデバイス全体にだけデータをストライピングします。つまり、複数のデバイスのサイズが少し異なる場合、それぞれのデバイスは最小ドライブと同じサイズであるかのように処理されます。したがって、レベル 0 アレイの共通ストレージ容量は、すべてのディスクの合計容量です。メンバー ディスクのサイズが異なる場合、RAID0 は使用可能なゾーンを使用して、それらのディスクのすべての領域を使用します。

レベル 1

RAID レベル 1 (ミラーリング) は、アレイの各メンバーディスクに同一のデータを書き込み、ミラー化されたコピーを各ディスクに残すことによって冗長性を提供します。ミラーリングは、データの可用性の単純化と高レベルにより、いまでも人気があります。レベル 1 は 2 つ以上のディスクと連携して、非常に優れたデータ信頼性を提供し、読み取り集中型のアプリケーションに対してパフォーマンスが向上しますが、比較的コストが高くなります。
RAID レベル 1 は、アレイ内のすべてのディスクに同じ情報を書き込むためコストがかかります。これにより、データの信頼性が提供されますが、レベル 5 などのパリティーベースの RAID レベルよりもスペース効率が大幅に低下します。ただし、この領域の非効率性にはパフォーマンス上の利点があります。パリティーベースの RAID レベルは、パリティーを生成するためにかなり多くの CPU 電力を消費しますが、RAID レベル 1 は単に同じデータを、CPU オーバーヘッドが非常に少ない複数の RAID メンバーに複数回書き込むだけです。そのため、RAID レベル 1 は、ソフトウェア RAID が使用されているマシンや、マシンの CPU リソースが一貫して RAID アクティビティー以外の操作でアレイ化されます。

レベル 1 アレイのストレージ容量は、ハードウェア RAID 内でミラーリングされている最小サイズの

ハードディスクの容量と同じか、ソフトウェア RAID 内でミラーリングされている最小のパーティションと同じ容量になります。レベル 1 の冗長性は、すべての RAID タイプの中で最も高いレベルであり、アレイは 1 つのディスクのみで動作できます。

レベル 4

レベル 4 は、1 つのディスクドライブでパリティー連結を使用して、データを保護します。パリティー情報は、アレイ内の残りのメンバーディスクのコンテンツに基づいて計算されます。この情報は、アレイ内のいずれかのディスクに障害が発生した場合にデータの再構築に使用できます。その後、再構築されたデータを使用して、交換前に失敗したディスクに I/O 要求に対応でき、交換後に失敗したディスクを接続します。
パリティー専用ディスクは、RAID アレイへのすべての書き込みトランザクションにおいて固有のボトルネックとなるため、ライトバックキャッシングなどの付随する技術なしにレベル 4 が使用されることはほとんどありません。または、システム管理者が意図的にこのボトルネックを考慮してソフトウェア RAID デバイスを設計している特定の状況下で使用されます。たとえば、アレイにデータが格納されると書き込みトランザクションがほとんどないようなアレイです。RAID レベル 4 にはほとんど使用されないため、Anaconda ではこのオプションとしては使用できません。ただし、実際には必要な場合は、ユーザーが手動で作成できます。

ハードウェア RAID レベル 4 のストレージ容量は、最小メンバーパーティションの容量にパーティションの数を掛けて 1 を引いた値に等しくなります。RAID レベル 4 アレイのパフォーマンスは常に非対称です。つまり、読み込みは書き込みを上回ります。これは、パリティーを生成するときに書き込み操作が余分な CPU リソースとメインメモリー帯域幅を消費し、実際のデータをディスクに書き込むときに余分なバス帯域幅も消費するためです。これは、データだけでなくパリティーも書き込むためです。読み取り操作は、アレイが劣化状態にない限り、データを読み取るだけでパリティーを読み取る必要はありません。その結果、読み取り操作では、通常の操作条件下で同じ量のデータ転送を行う場合でも、ドライブおよびコンピューターのバス全体に生成されるトラフィックが少なくなります。

レベル 5

これは RAID の最も一般的なタイプです。RAID レベル 5 は、アレイのすべてのメンバーディスクドライブにパリティーを分散することにより、レベル 4 に固有の書き込みボトルネックを排除します。パリティー計算プロセス自体のみがパフォーマンスのボトルネックです。最近の CPU はパリティーを非常に高速に計算できます。しかし、RAID 5 アレイに多数のディスクを使用していて、すべてのデバイスの合計データ転送速度が十分に高い場合、パリティー計算がボトルネックになる可能性があります。
レベル 5 のパフォーマンスは非対称であり、読み取りは書き込みよりも大幅に優れています。RAID レベル 5 のストレージ容量は、レベル 4 と同じです。

レベル 6

パフォーマンスではなくデータの冗長性と保存が最重要事項であるが、レベル 1 の領域の非効率性が許容できない場合は、これが RAID の一般的なレベルです。レベル 6 では、複雑なパリティースキームを使用して、アレイ内の 2 つのドライブから失われたドライブから復旧できます。複雑なパリティースキームにより、ソフトウェア RAID デバイスで CPU 幅が大幅に高くなり、書き込みトランザクションの際に増大度が高まります。したがって、レベル 6 はレベル 4 や 5 よりもパフォーマンスにおいて、非常に非対称です。
RAID レベル 6 アレイの合計容量は、RAID レベル 5 および 4 と同様に計算されますが、デバイス数から追加パリティーストレージ領域用に 2 つのデバイス (1 ではなく) を引きます。

レベル 10

この RAID レベルでは、レベル 0 のパフォーマンスとレベル 1 の冗長性を組み合わせます。また、2 台以上のデバイスを使用するレベル 1 アレイの無駄なスペースをある程度削減することができます。レベル 10 では、たとえば、データごとに 2 つのコピーのみを格納するように設定された 3 ドライブアレイを作成することができます。これにより、全体用のアレイサイズを最小デバイスのみと同じ

サイズ (3 つのデバイス、レベル 1 アレイなど) ではなく、最小デバイスのサイズの 1.5 倍にすることができます。これにより、CPU プロセスの使用量が RAID レベル 6 のようにパリティーを計算するのを防ぎますが、これは領域効率が悪くなります。
RAID レベル 10 の作成は、インストール時には対応していません。インストール後に手動で作成できます。

リニア RAID

リニア RAID は、より大きな仮想ドライブを作成するドライブのグループ化です。
リニア RAID では、あるメンバードライブからチャンクが順次割り当てられます。最初のドライブが完全に満杯になったときにのみ次のドライブに移動します。これにより、メンバードライブ間の I/O 操作が分割される可能性はないため、パフォーマンスの向上は見られません。リニア RAID は冗長性がなく、信頼性は低下します。メンバードライブが 1 台でも故障すると、アレイ全体が使用できなくなり、データが失われる可能性があります。容量はすべてのメンバーディスクの合計になります。

68.7.3. LVM RAID のセグメントタイプ

RAID 論理ボリュームを作成するには、RAID タイプを **lvcreate** コマンドの **--type** 引数として指定します。ほとんどのユーザーの場合、**raid1**、**raid4**、**raid5**、**raid6**、**raid10** の 5 つの使用可能なプライマリータイプのいずれかを指定するだけで十分です。

以下の表は、考えられる RAID セグメントタイプを示しています。

表68.4 LVM RAID のセグメントタイプ

セグメントタイプ	説明
raid1	RAID1 ミラーリング。**-m** 引数を指定し、ストライピングを指定しない場合は、これが **lvcreate** コマンドの **--type** 引数のデフォルト値になります。
raid4	RAID4 専用パリティーディスク
raid5_la	• RAID5 left asymmetric • ローテートパリティー 0 + データ継続
raid5_ra	• RAID5 right asymmetric • ローテートパリティー N + データ継続
raid5_ls	• RAID5 left symmetric • **raid5** と同じです。 • ローテートパリティー 0 +データ再起動

セグメントタイプ	説明
raid5_rs	• RAID5 right symmetric • ローテートパリティー N + データ再起動
raid6_zr	• RAID6 zero restart • **raid6** と同じです。 • ローテートパリティーゼロ (左から右) + データ再起動
raid6_nr	• RAID6 N restart • ローテートパリティー N (左から右) + データ再起動
raid6_nc	• RAID6 N continue • ローテートパリティー N (左から右) + データを継続
raid10	• ストライピング + ミラーリング。**-m** 引数を指定し、1 よりも大きい数をストライプの数として指定すると、これが **lvcreate** コマンドの **--type** 引数のデフォルト値になります。 • ミラーセットのストライピング
raid0/raid0_meta	ストライピング。RAID0 では、ストライプサイズの単位で、複数のデータサブボリュームに論理ボリュームデータが分散されます。これは、パフォーマンスを向上させるために使用します。論理ボリュームのデータは、いずれかのデータサブボリュームで障害が発生すると失われます。

68.7.4. RAID 論理ボリュームの作成

-m 引数に指定する値に応じて、複数のコピーを持つ RAID1 アレイを作成できます。同様に、 **-i** 引数を使用して、RAID 0、4、5、6、10 論理ボリュームのストライピング数を指定できます。**-I** 引数で、ストライプのサイズを指定することもできます。以下の手順では、異なるタイプの RAID 論理ボリュームを作成するさまざまな方法を説明します。

手順

- 2 方向 RAID を作成します。以下のコマンドは、ボリュームグループ **my_vg** 内にサイズが **1G** の 2 方向 RAID1 アレイ **my_lv** を作成します。

```
# lvcreate --type raid1 -m 1 -L 1G -n my_lv my_vg
Logical volume "my_lv" created.
```

- ストライピングで RAID5 アレイを作成します。次のコマンドは、ボリュームグループ **my_vg**

に、3 つのストライプと 1 つの暗黙のパリティードライブ (**my_lv**) を持つ、サイズが **1G** の RAID5 アレイを作成します。LVM ストライピングボリュームと同様にストライピングの数を指定できることに注意してください。正しい数のパリティードライブが自動的に追加されます。

```
# lvcreate --type raid5 -i 3 -L 1G -n my_lv my_vg
```

- ストライピングで RAID6 アレイを作成します。次のコマンドは、ボリュームグループ **my_vg** に 3 つの 3 ストライプと 2 つの暗黙的なパリティードライブ (**my_lv** という名前) を持つ RAID6 アレイを作成します。これは、**1G** 1 ギガバイトのサイズです。

```
# lvcreate --type raid6 -i 3 -L 1G -n my_lv my_vg
```

検証

- 2 ウェイ RAID1 アレイである LVM デバイス **my_vg/my_lv** を表示します。

```
# lvs -a -o name,copy_percent,devices _my_vg_
  LV              Copy%  Devices
  my_lv           6.25   my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(256)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

関連情報

- **lvcreate(8)** と **lvmraid(7)** の man ページ

68.7.5. RAID0 ストライピング 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のデータサブボリューム全体に分散します。以下の手順では、ディスク間でデータをストライピングする **mylv** という LVM RAID0 論理ボリュームを作成します。

前提条件

1. 3 つ以上の物理ボリュームを作成している。物理ボリュームの作成方法は、LVM 物理ボリュームの作成 を参照してください。
2. ボリュームグループを作成している。詳細は、LVM ボリュームグループの作成 を参照してください。

手順

1. 既存のボリュームグループから RAID0 論理ボリュームを作成します。次のコマンドは、ボリュームグループ **myvg** から RAID0 ボリューム **mylv** を作成します。これは、サイズが **2G** で、ストライプが 3 つ、ストライプ サイズが **4kB** です。

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv my_vg
  Rounding size 2.00 GiB (512 extents) up to stripe boundary size 2.00 GiB(513 extents).
  Logical volume "mylv" created.
```

2. RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに ext4 ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/my_vg/mylv
```

3. 論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

```
# mount /dev/my_vg/mylv /mnt

# df
Filesystem            1K-blocks     Used  Available  Use% Mounted on
/dev/mapper/my_vg-mylv   2002684     6168  1875072    1%   /mnt
```

検証

- 作成された RAID0 ストライピング論理ボリュームを表示します。

```
# lvs -a -o +devices,segtype my_vg
  LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert Devices Type
  mylv my_vg rwi-a-r--- 2.00g mylv_rimage_0(0),mylv_rimage_1(0),mylv_rimage_2(0) raid0
  [mylv_rimage_0] my_vg iwi-aor--- 684.00m /dev/sdf1(0) linear
  [mylv_rimage_1] my_vg iwi-aor--- 684.00m /dev/sdg1(0) linear
  [mylv_rimage_2] my_vg iwi-aor--- 684.00m /dev/sdh1(0) linear
```

68.7.6. RAID0 を作成するためのパラメーター

RAID0 ストライピング論理ボリュームは、**lvcreate --type raid0[meta] --stripes _Stripes --stripesize StripeSize VolumeGroup [PhysicalVolumePath]** コマンドを使用して作成することができます。

次の表は、RAID0 ストライピング論理ボリュームを作成するときに使用できるさまざまなパラメーターを説明しています。

表68.5 RAID0 ストライピング論理ボリュームを作成するためのパラメーター

パラメーター	説明
--type raid0[_meta]	**raid0** を指定すると、メタデータボリュームなしで RAID0 ボリュームが作成されます。**raid0_meta** を指定すると、メタデータボリュームとともに RAID0 ボリュームが作成されます。RAID0 には耐障害性がないため、RAID1/10 の場合のようにミラーリングされたすべてのデータブロックを格納したり、RAID4/5/6 の場合のようにすべてのパリティーブロックを計算して格納したりしません。したがって、ミラーリングされたブロックまたはパリティーブロックの再同期の進行状態を把握するメタデータボリュームは必要ありません。RAID0 から RAID4/5/6/10 への変換では、メタデータボリュームが必須となります。**raid0_meta** を指定すると、これらのメタデータボリュームが事前に割り当てられ、それぞれの割り当ての失敗を防ぐことができます。
--stripes Stripes	論理ボリュームを分散するデバイスの数を指定します。
--stripesize StripeSize	各ストライプのサイズをキロバイト単位で指定します。これは、次のデバイスに移動する前にデバイスに書き込まれるデータの量です。

パラメーター	説明
VolumeGroup	使用するボリュームグループを指定します。
PhysicalVolumePath	使用するデバイスを指定します。指定しない場合は、LVM によって、**Stripes** オプションに指定されているデバイスの数が、各ストライプに 1 つずつ選択されます。

68.7.7. ソフトデータの破損

データストレージにおけるソフト破損は、ストレージデバイスから取得したデータが、そのデバイスに書き込まれるデータとは異なることを意味します。破損したデータは、ストレージデバイスで無期限に存在する可能性があります。破損したデータは、このデータを取得および使用するまで、検出されない可能性があります。

設定のタイプに応じて、Redundant Array of Independent Disks (RAID) 論理ボリューム (LV) は、デバイスに障害が発生した場合のデータ損失を防ぎます。RAID アレイで構成されているデバイスに障害が発生した場合、その RAID LV の一部である他のデバイスからデータを回復できます。ただし、RAID 設定により、データの一貫性は確保されません。ソフト破損、無兆候破損、ソフトエラー、およびサイレントエラーでは、システム設計やソフトウェアが想定どおりに機能し続けている場合でも、破損するデータを示す用語です。

デバイスマッパー (DM) 整合性は、RAID レベル 1、4、5、6、10 で使用され、ソフト破損によるデータ損失を軽減または防止します。RAID レイヤーでは、データの結合のないコピーが、ソフト破損エラーを修正できるようになります。整合性層は、各 RAID イメージの上にありますが、追加のサブ LV が、各 RAID イメージの整合性メタデータまたはデータチェックサムを格納します。整合性のある RAID LV からデータを取得すると、整合性データのチェックサムが破損のデータを分析します。破損が検出されると、整合性レイヤーはエラーメッセージを返し、RAID 層は、別の RAID イメージからデータの破損していないコピーを取得します。RAID レイヤーは、ソフト破損を修復するために、破損したデータを、破損していないデータで書き換えます。

DM 整合性で新しい RAID LV を作成したり、既存の RAID LV に整合性を追加する場合は、以下の点を考慮してください。

- 整合性メタデータには、追加のストレージ領域が必要です。各 RAID イメージには、データに追加されるチェックサムがあるため、500MB の全データに 4 MB のストレージ領域が必要になります。
- 一部の RAID 設定には、より多くの影響がありますが、データにアクセスする際のレイテンシーにより、DM 整合性を追加するとパフォーマンスに影響が及びます。RAID1 設定は通常、RAID5 またはそのバリアントよりも優れたパフォーマンスを提供します。
- RAID 整合性ブロックサイズは、パフォーマンスにも影響を及ぼします。RAID 整合性ブロックサイズが大きいと、パフォーマンスが向上します。ただし、RAID 整合性ブロックのサイズが小さくなると、後方互換性がより高くなります。
- 利用可能な整合性モードには、**bitmap** または **journal** の 2 つがあります。通常、**bitmap** 整合性モードは、**journal** モードよりも優れたパフォーマンスを提供します。

ヒント

パフォーマンスの問題が発生した場合は、整合性で RAID1 を使用するか、特定の RAID 設定のパフォーマンスをテストして、要件を満たすことを確認してください。

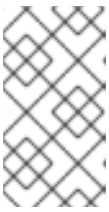
68.7.8. DM 整合性での RAID LV の作成

デバイスマッパー (DM) 整合性を持つ RAID LV を作成したり、既存の RAID LV に整合性を追加したりすると、ソフト破損によるデータ損失のリスクが軽減されます。LV を使用する前に、整合性の同期と RAID メタデータが完了するのを待ちます。そうしないと、バックグラウンドの初期化が LV のパフォーマンスに影響する可能性があります。

手順

1. DM 整合性のある RAID LV を作成します。次の例では、**my_vg** ボリュームグループに **test-lv** という名前の整合性を持つ新しい RAID LV を作成します。使用可能なサイズは **256M** で、RAID レベルは **1** です。

```
# lvcreate --type raid1 --raidintegrity y -L 256M -n test-lv my_vg
Creating integrity metadata LV test-lv_rimage_0_imeta with size 8.00 MiB.
Logical volume "test-lv_rimage_0_imeta" created.
Creating integrity metadata LV test-lv_rimage_1_imeta with size 8.00 MiB.
Logical volume "test-lv_rimage_1_imeta" created.
Logical volume "test-lv" created.
```



注記

既存の RAID LV に DM 整合性を追加するには、次のコマンドを実行します。

```
# lvconvert --raidintegrity y my_vg/test-lv
```

RAID LV に整合性を追加すると、その RAID LV で実行可能な操作の数が制限されます。

2. オプション:特定の操作を実行する前に整合性を削除します。

```
# lvconvert --raidintegrity n my_vg/test-lv
Logical volume my_vg/test-lv has removed integrity.
```

検証

- 追加された DM 整合性に関する情報を表示します。
  - **my_vg** ボリュームグループ内に作成された test-lv RAID LV の情報を表示します。

```
# lvs -a my_vg
  LV                  VG      Attr       LSize   Origin                Cpy%Sync
  test-lv             my_vg rwi-a-r--- 256.00m                          2.10
  [test-lv_rimage_0]      my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig] 93.75
  [test-lv_rimage_0_imeta]  my_vg ewi-ao----   8.00m
  [test-lv_rimage_0_iorig]  my_vg -wi-ao---- 256.00m
  [test-lv_rimage_1]      my_vg gwi-aor--- 256.00m [test-lv_rimage_1_iorig] 85.94
 [...]
```

以下は、この出力から得られるさまざまなオプションについて説明したものです。

g 属性

これは、Attr 列の下にある属性のリストで、RAID イメージが整合性を使用していることを示します。整合性は、チェックサムを **_imeta** RAID LV に保存します。

Cpy%Sync 列

最上位の RAID LV と各 RAID イメージの両方の同期の進行状況を示します。

RAID イメージ

LV 列に **raid_image_N** で表示されます。

LV 列

これにより、最上位の RAID LV と各 RAID イメージの同期の進行状況が 100% と表示されるようになります。

- 各 RAID LV のタイプを表示します。

```
# lvs -a my-vg -o+segtype
  LV                  VG      Attr       LSize   Origin               Cpy%Sync Type
  test-lv             my_vg rwi-a-r--- 256.00m                        87.96    raid1
  [test-lv_rimage_0]      my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig] 100.00
integrity
  [test-lv_rimage_0_imeta] my_vg ewi-ao----   8.00m                            linear
  [test-lv_rimage_0_iorig] my_vg -wi-ao---- 256.00m                            linear
  [test-lv_rimage_1]      my_vg gwi-aor--- 256.00m [test-lv_rimage_1_iorig] 100.00
integrity
  [...]
```

- 各 RAID イメージで検出された不一致の数をカウントする増分カウンターがあります。**my_vg/test-lv** の下の **rimage_0** から整合性で検出されたデータの不一致を表示します。

```
# lvs -o+integritymismatches my_vg/test-lv_rimage_0
  LV                  VG      Attr       LSize   Origin               Cpy%Sync IntegMismatches
  [test-lv_rimage_0] my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig]    100.00
0
```

この例では、整合性はデータの不一致を検出していないため、**IntegMismatches** カウンターはゼロ (0) を示しています。

- 以下の例に示すように、**/var/log/messages** ログファイル内のデータ整合性情報を表示します。

例68.2 カーネルメッセージログから dm-integrity の不一致の例

```
device-mapper: integrity: dm-12:セクター 0x24e7 でチェックサムが失敗しました。
```

例68.3 カーネルメッセージログからの dm-integrity データ修正の例

```
md/raid1:mdX: 読み込みエラーが修正されました (dm-16 の 9448 の 8 セクター)
```

関連情報

- **lvcreate(8)** と **lvmraid(7)** の man ページ

68.7.9. 最小/最大 I/O レートオプション

RAID10 論理ボリュームを作成する際に、sync 操作で論理ボリュームを初期化するのに必要なバックグ

ラウンド I/O は、特に RAID 論理ボリュームを多数作成している場合に、他の I/O 操作 (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。

RAID 論理ボリュームが初期化される速度は、復旧スロットルを実装することで制御できます。**sync** 操作が実行される速度を制御するには、**lvcreate** コマンドの **--minrecoveryrate** および **--maxrecoveryrate** オプションを使用して、これらの操作の最小および最大 I/O 速度を設定します。

これらのオプションは次のように指定できます。

--maxrecoveryrate Rate[bBsSkKmMgG]

RAID 論理ボリュームの最大復旧速度を設定し、通常の I/O 操作が押し出されないようにします。Rate は、アレイ内の各デバイスに対する 1 秒あたりの量を指定します。接尾辞を指定しない場合は、kiB/sec/device とみなします。復旧速度を 0 に設定すると無制限になります。

--minrecoveryrate Rate[bBsSkKmMgG]

RAID 論理ボリュームの最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、同期操作の I/O が最小スループットを達成できるようにします。Rate は、アレイ内の各デバイスに対する 1 秒あたりの量を指定します。接尾辞を指定しない場合は、kiB/sec/device とみなします。

たとえば、**lvcreate --type raid10 -i 2 -m 1 -L 10G --maxrecoveryrate 128 -n my_lv my_vg** コマンドを使用して、ボリュームグループ **my_vg** 内に 3 ストライプでサイズ 10G、最大回復速度 128 kiB/sec/device の 2 方向の RAID10 アレイ **my_lv** を作成します。RAID のスクラブ操作の最小および最大復旧速度を指定することもできます。

68.7.10. リニアデバイスの RAID 論理ボリュームへの変換

既存のリニア論理ボリュームを RAID 論理ボリュームに変換することができます。この操作を行うには、**lvconvert** コマンドの **--type** 引数を使用します。

RAID 論理ボリュームは、メタデータとデータのサブボリュームのペアで構成されています。リニアデバイスを RAID1 アレイに変換すると、新しいメタデータサブボリュームが作成され、リニアボリュームと同じ物理ボリューム上の元の論理ボリュームと関連付けられます。追加のイメージは、メタデータ/データ サブボリュームのペアに追加されます。複製元の論理ボリュームとペアのメタデータイメージを同じ物理ボリュームに配置できないと、**lvconvert** は失敗します。

手順

1. 変換が必要な論理ボリュームデバイスを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV     Copy%  Devices
  my_lv         /dev/sde1(0)
```

2. リニア論理ボリュームを RAID デバイスに変換します。以下のコマンドは、ボリュームグループ __my_vg のリニア論理ボリューム **my_lv** を、2 方向の RAID1 アレイに変換します。

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
  Are you sure you want to convert linear LV my_vg/my_lv to raid1 with 2 images enhancing resilience? [y/n]: y
  Logical volume my_vg/my_lv successfully converted.
```

検証

- 論理ボリュームが RAID デバイスに変換されているかどうかを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           6.25   my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(256)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.11. LVM RAID1 論理ボリュームを LVM リニア論理ボリュームに変換

既存の RAID1 LVM 論理ボリュームを LVM リニア論理ボリュームに変換することができます。この操作を行うには、**lvconvert** コマンドを使用し、**-m0** 引数を指定します。これにより、RAID アレイを構成する全 RAID データサブボリュームおよび全 RAID メタデータサブボリュームが削除され、最高レベルの RAID1 イメージがリニア論理ボリュームとして残されます。

手順

1. 既存の LVM RAID1 論理ボリュームを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

2. 既存の RAID1 LVM 論理ボリュームを LVM リニア論理ボリュームに変換します。以下のコマンドは、LVM RAID1 論理ボリューム **my_vg/my_lv** を、LVM リニアデバイスに変換します。

```
# lvconvert -m0 my_vg/my_lv
  Are you sure you want to convert raid1 LV my_vg/my_lv to type linear losing all resilience?
[y/n]: y
  Logical volume my_vg/my_lv successfully converted.
```

LVM RAID1 論理ボリューム を LVM リニアボリュームに変換する場合は、削除する物理ボリュームを指定することもできます。以下の例では、**lvconvert** コマンドは **/dev/sde1** を削除して、**/dev/sdf1** をリニアデバイスを構成する物理ボリュームとして残すように指定します。

```
# lvconvert -m0 my_vg/my_lv /dev/sde1
```

検証

- RAID1 論理ボリュームが LVM リニアデバイスに変換されたかどうかを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV    Copy%  Devices
  my_lv        /dev/sdf1(1)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.12. ミラーリングされた LVM デバイスの RAID1 論理ボリュームへの変換

セグメントタイプのミラーを持つ既存のミラーリングされた LVM デバイスを RAID1 LVM デバイスに変換できます。この操作を行うには、**--type** raid1 引数を指定して、**lvconvert** コマンドを使用します。これにより、**mimage** という名前のミラーサブボリュームの名前が、**rimage** という名前の RAID サブボリュームに変更されます。

さらに、ミラーログも削除し、対応するデータサブボリュームと同じ物理ボリューム上にデータサブボリューム用の **rmeta** という名前のメタデータサブボリュームを作成します。

手順

1. ミラーリングされた論理ボリューム **my_vg/my_lv の** のレイアウトを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           15.20 my_lv_mimage_0(0),my_lv_mimage_1(0)
  [my_lv_mimage_0]       /dev/sde1(0)
  [my_lv_mimage_1]       /dev/sdf1(0)
  [my_lv_mlog]           /dev/sdd1(0)
```

2. ミラーリングされた論理ボリューム **my_vg/my_lv** を RAID1 論理ボリュームに変換します。

```
# lvconvert --type raid1 my_vg/my_lv
Are you sure you want to convert mirror LV my_vg/my_lv to raid1 type? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

検証

- ミラーリングされた論理ボリュームが RAID1 論理ボリュームに変換されているかどうかを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(0)
  [my_lv_rmeta_0]        /dev/sde1(125)
  [my_lv_rmeta_1]        /dev/sdf1(125)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.13. RAID 論理ボリュームのサイズ変更

RAID 論理ボリュームのサイズは、以下の方法で変更できます。

- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** コマンドまたは **lvextend** コマンドで増やすことができます。これは、RAID イメージの数を変更するものではありません。スト

ライプ化 RAID 論理ボリュームでは、ストライプ化 RAID 論理ボリュームの作成時と同じ、ストライプを丸める制約が適用されます。

- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** コマンドまたは **lvreduce** コマンドで減らすことができます。これは、RAID イメージの数を変更するものではありません。**lvextend** コマンドでは、ストライプ化 RAID 論理ボリュームの作成時と同じストライプを丸める制約が適用されます。

- **lvconvert** コマンドで **--stripes N** パラメーターを使用すると、ストライプ化 RAID 論理ボリューム (**raid4**/**5**/**6**/**10**) のストライプの数を変更できます。このように、ストライプを追加または削除することで、RAID 論理ボリュームのサイズを増減できます。**raid10** ボリュームにはストライプを追加することしかできないため注意してください。この機能は、同じ RAID レベルを維持しながら、RAID 論理ボリュームの属性を変更することができる、RAID の **再成形** 機能になります。RAID 再成形と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再生成する例は、man ページの **lvmraid**(7) を参照してください。

68.7.14. 既存の RAID1 デバイスのイメージ数を変更

LVM ミラーリングの実装でイメージの数を変更できる方法と同様に、既存の RAID1 アレイのイメージの数を変更できます。

lvconvert コマンドを使用して RAID1 論理ボリュームにイメージを追加すると、次の操作を実行できます。

- 結果として作成されるデバイス用イメージの総数を指定する
- デバイスに追加するイメージの数
- オプションで、新しいメタデータ/データイメージのペアが存在する物理ボリュームを指定する

手順

1. 2 ウェイ RAID1 アレイである LVM デバイス **my_vg/my_lv** を表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             6.25 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(256)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

   メタデータサブボリューム (**rmeta** と呼ばれる) は、対応するデータサブボリューム (**rimage**) と同じ物理デバイスに常に存在します。メタデータ/データのサブボリュームのペアは、**--alloc** をどこかに指定しない限り、RAID アレイにある別のメタデータ/データサブボリュームのペアと同じ物理ボリュームには作成されません。

2. 2 ウェイ RAID1 論理ボリューム **my_vg/my_lv** を 3 ウェイ RAID1 論理ボリュームに変換します。

```
# lvconvert -m 2 my_vg/my_lv
Are you sure you want to convert raid1 LV my_vg/my_lv to 3 images enhancing resilience?
[y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

既存の RAID1 デバイスのイメージ数を変更する場合の例を以下に示します。

- また、RAID にイメージを追加する際に、使用する物理ボリュームを指定することもできます。次のコマンドは、2 方向 RAID1 論理ボリューム **my_vg/my_lv** を 3 方向 RAID1 論理ボリュームに変換し、物理ボリューム **/dev/sdd1** がアレイに使用されるように指定します。

```
# lvconvert -m 2 my_vg/my_lv /dev/sdd1
```

- 3 方向 RAID1 論理ボリュームを 2 方向 RAID1 論理ボリュームに変換します。

```
# lvconvert -m1 my_vg/my_lv
Are you sure you want to convert raid1 LV my_vg/my_lv to 2 images reducing resilience? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

- 削除するイメージを含む物理ボリューム **/dev/sde1** を指定して、3 方向 RAID1 論理ボリュームを 2 方向 RAID1 論理ボリュームに変換します。

```
# lvconvert -m1 my_vg/my_lv /dev/sde1
```

また、イメージとその関連付けられたメタデータのサブボリュームを削除すると、それよりも大きな番号のイメージが下に移動してそのスロットを引き継ぎます。**lv_rimage_0**、**lv_rimage_1**、および **lv_rimage_2** で構成される 3 方向 RAID1 アレイから**lv_rimage_1** を削除すると、**lv_rimage_0** と **lv_rimage_1** で構成される RAID1 アレイになります。サブボリューム **lv_rimage_2** の名前が、空のスロットを引き継いで **lv_rimage_1** になります。

検証

- 既存の RAID1 デバイスのイメージ数を変更した後に、RAID1 デバイスを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV Cpy%Sync Devices
  my_lv 100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0] /dev/sdd1(1)
  [my_lv_rimage_1] /dev/sde1(1)
  [my_lv_rimage_2] /dev/sdf1(1)
  [my_lv_rmeta_0] /dev/sdd1(0)
  [my_lv_rmeta_1] /dev/sde1(0)
  [my_lv_rmeta_2] /dev/sdf1(0)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.15. RAID イメージを複数の論理ボリュームに分割

RAID 論理ボリュームのイメージを分割して新しい論理ボリュームを形成できます。既存の RAID1 論理ボリュームから RAID イメージを削除する場合と同様に、RAID データのサブボリューム (およびその関連付けられたメタデータのサブボリューム) をデバイスから削除する場合、それより大きい番号のイメージは、そのスロットを埋めるために番号が変更になります。そのため、RAID アレイを構成する論理ボリューム上のインデックス番号は連続する整数となります。

注記

RAID1 アレイがまだ同期していない場合は、RAID イメージを分割できません。

手順

1. 2 ウェイ RAID1 アレイである LVM デバイス **my_vg/my_lv** を表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             12.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

2. RAID イメージを別の論理ボリュームに分割します。以下の例は、2 方向の RAID1 論理ボリューム **my_lv** を、**my_lv** と **new** の 2 つのリニア論理ボリュームに分割します。

```
# lvconvert --splitmirror 1 -n new my_vg/my_lv
Are you sure you want to split raid1 LV my_vg/my_lv losing all resilience? [y/n]: y
```

3 方向 RAID1 論理ボリューム **my_lv** を 2 方向 RAID1 論理ボリューム **my_lv** とリニア論理ボリューム **new** に分割します。

```
# lvconvert --splitmirror 1 -n new my_vg/my_lv
```

検証

- RAID 論理ボリュームのイメージを分割した後に、論理ボリュームを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV      Copy%  Devices
  my_lv          /dev/sde1(1)
  new            /dev/sdf1(1)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.16. RAID イメージの分割とマージ

lvconvert コマンドで、**--splitmirrors** 引数とともに **--trackchanges** 引数を使用すると、すべての変更を追跡しながら、RAID1 アレイのイメージを一時的に読み取り専用に分割できます。この機能を使えば、イメージを分割した後に変更した部分のみを再同期しながら、後でイメージをアレイに統合することができます。

--trackchanges 引数を使用して RAID イメージを分割する場合、分割するイメージを指定することはできますが、分割されるボリューム名を変更することはできません。また、作成されたボリュームには以下の制約があります。

- 作成された新規ボリュームは読み取り専用です。
- 新規ボリュームのサイズは変更できません。

- 残りのアレイの名前は変更できません。
- 残りのアレイのサイズは変更できません。
- 新規のボリュームと、残りのアレイを個別にアクティブにすることはできません。

分割されたイメージを結合することができます。イメージをマージすると、イメージが分割されてから変更したアレイの部分のみが再同期されます。

手順

1. RAID 論理ボリュームを作成します。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
  Logical volume "my_lv" created
```

2. オプション:作成した RAID 論理ボリュームを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]        /dev/sdb1(1)
  [my_lv_rimage_1]        /dev/sdc1(1)
  [my_lv_rimage_2]        /dev/sdd1(1)
  [my_lv_rmeta_0]         /dev/sdb1(0)
  [my_lv_rmeta_1]         /dev/sdc1(0)
  [my_lv_rmeta_2]         /dev/sdd1(0)
```

3. 作成した RAID 論理ボリュームからイメージを分割し、残りのアレイへの変更を追跡します。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  my_lv_rimage_2 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_2' to merge back into my_lv
```

4. オプション:イメージを分割した後の論理ボリュームを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]        /dev/sdc1(1)
  [my_lv_rimage_1]          /dev/sdd1(1)
  [my_lv_rmeta_0]         /dev/sdc1(0)
  [my_lv_rmeta_1]         /dev/sdd1(0)
```

5. ボリュームをアレイにマージして戻します。

```
# lvconvert --merge my_vg/my_lv_rimage_1
  my_vg/my_lv_rimage_1 successfully merged back into my_vg/my_lv
```

検証

- マージされた論理ボリュームを表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
```

```
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sdc1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdc1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
```

関連情報

- **lvconvert(8)** の man ページ

68.7.17. RAID 障害ポリシーの設定

LVM RAID は、**lvm.conf** ファイルの **raid_fault_policy** フィールドで定義されている詳細設定に基づいて、デバイス障害を自動で処理します。

- **raid_fault_policy** フィールドが **allocate** に設定されている場合、システムは障害が発生したデバイスをボリュームグループの予備のデバイスに置き換えようとします。予備のデバイスがないと、システムログにレポートが送信されます。
- **raid_fault_policy** フィールドが **warn** に設定されている場合、システムは警告を生成して、デバイスが失敗したことがログに示されます。これにより、ユーザーは実施する一連の動作を確認できます。

残りのデバイスで該当するポリシーに対応できる限り、RAID 論理ボリュームは操作を続行します。

68.7.17.1. allocateRAID 障害ポリシー

以下の例では、**raid_fault_policy** フィールドは **lvm.conf** ファイルで **allocate** に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
```

/dev/sde デバイスが失敗すると、システムログはエラーメッセージを表示します。

```
# grep lvm /var/log/messages
Jan 17 15:57:18 bp-01 lvm[8599]: Device #0 of raid1 array, my_vg-my_lv, has failed.
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994294784: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994376704: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at 0:
Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
4096: Input/output error
Jan 17 15:57:19 bp-01 lvm[8599]: Couldn't find device with uuid
```

```
3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
Jan 17 15:57:27 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is not in-sync.
Jan 17 15:57:36 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is now in-sync.
```

raid_fault_policy フィールドが **allocate** に設定されているため、障害が発生したデバイスは、ボリュームグループの新しいデバイスに置き換わります。

```
# lvs -a -o name,copy_percent,devices vg
  Couldn't find device with uuid 3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
  LV            Copy%  Devices
  lv            100.00 lv_rimage_0(0),lv_rimage_1(0),lv_rimage_2(0)
  [lv_rimage_0]        /dev/sdh1(1)
  [lv_rimage_1]        /dev/sdf1(1)
  [lv_rimage_2]        /dev/sdg1(1)
  [lv_rmeta_0]         /dev/sdh1(0)
  [lv_rmeta_1]         /dev/sdf1(0)
  [lv_rmeta_2]         /dev/sdg1(0)
```

障害が発生したデバイスを交換しても、LVM は、障害が発生したデバイスが見つけられないと示すことに注意してください。これは、障害が発生したデバイスが、RAID 論理ボリュームからは削除されても、ボリュームグループからは削除されていないためです。障害が発生したデバイスをボリュームグループから削除するには、**vgreduce --removemissing VG** を実行できます。

raid_fault_policy を **allocate** に設定したにもかかわらず、予備のデバイスがない場合は、割り当てが失敗し、論理ボリュームがそのままの状態になります。割り当てに失敗すると、ドライブが修復され、**lvchange** コマンドの **--refresh** オプションで、障害が発生したデバイスの復元を開始できます。もしくは、障害が発生したデバイスを交換することもできます。

68.7.17.2. warnRAID 障害ポリシー

以下の例では、**raid_fault_policy** フィールドは **lvm.conf** ファイルで **warn** に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV               Copy%  Devices
  my_lv            100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]        /dev/sdh1(1)
  [my_lv_rimage_1]        /dev/sdf1(1)
  [my_lv_rimage_2]        /dev/sdg1(1)
  [my_lv_rmeta_0]         /dev/sdh1(0)
  [my_lv_rmeta_1]         /dev/sdf1(0)
  [my_lv_rmeta_2]         /dev/sdg1(0)
```

/dev/sdh デバイスに障害が発生すると、システムログはエラーメッセージを表示します。ただし、この場合、LVM はイメージの 1 つを置き換えて、RAID デバイスを自動的に修復しようとはしません。したがって、デバイスに障害が発生したら、**lvconvert** コマンドの **--repair** 引数を使用してデバイスを置き換えることができます。

68.7.18. 論理ボリュームで RAID デバイスの交換

論理ボリュームの RAID デバイスは置き換えることができます。

- RAID デバイスに障害がない場合は、「障害のない RAID デバイスの交換」 を参照してください。

- RAID デバイスに障害が発生した場合は、「論理ボリュームに障害が発生した RAID デバイスの交換」 を参照してください。

68.7.18.1. 障害のない RAID デバイスの交換

論理ボリュームの RAID デバイスを交換するには、**lvconvert** コマンドの **--replace** 引数を使用します。

前提条件

- RAID デバイスに障害が発生していません。以下のコマンドを使用すると、RAID デバイスが失敗しても動作しません。

手順

- RAID デバイスを置き換えます。

```
# lvconvert --replace dev_to_remove vg/lv possible_replacements
```

  - **dev_to_remove** を、置き換える物理ボリュームへのパスに置き換えます。
  - **vg/lv** を、RAID アレイのボリュームグループおよび論理ボリューム名に置き換えます。
  - **possible_replacements** を、交換する物理ボリュームへのパスに置き換えます。

例68.4 RAID1デバイスの置き換え

以下の例では、RAID1 論理ボリュームを作成した後に、そのボリューム内のデバイスを交換しています。

1. RAID1 アレイを作成します。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg

  Logical volume "my_lv" created
```

2. RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdb2(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdb2(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
```

3. **/dev/sdb2** 物理ボリュームを置き換えます。

```
# lvconvert --replace /dev/sdb2 my_vg/my_lv
```

4. 代替の RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
  my_lv           37.50 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdc2(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdc2(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
```

例68.5 代替の物理ボリュームの指定

以下の例では、RAID1 論理ボリュームを作成した後に、そのボリュームのデバイスを交換し、交換したデバイスに使用する物理ボリュームを指定しています。

1. RAID1 アレイを作成します。

```
# lvcreate --type raid1 -m 1 -L 100 -n my_lv my_vg

  Logical volume "my_lv" created
```

2. RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
```

3. 物理ボリュームを確認します。

```
# pvs

  PV         VG       Fmt  Attr PSize    PFree
  /dev/sda1  my_vg    lvm2 a--  1020.00m  916.00m
  /dev/sdb1  my_vg    lvm2 a--  1020.00m  916.00m
  /dev/sdc1  my_vg    lvm2 a--  1020.00m 1020.00m
  /dev/sdd1  my_vg    lvm2 a--  1020.00m 1020.00m
```

4. **/dev/sdb1** 物理ボリュームを **/dev/sdd1** に置き換えます。

```
# lvconvert --replace /dev/sdb1 my_vg/my_lv /dev/sdd1
```

5. 代替の RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
```

```
  my_lv            28.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
```

例68.6 複数の RAID デバイスの置き換え

一度に 2 つ以上の RAID デバイスを交換するには、以下の例のように複数の **replace** 引数を指定します。

1. RAID1 アレイを作成します。

```
# lvcreate --type raid1 -m 2 -L 100 -n my_lv my_vg

  Logical volume "my_lv" created
```

2. RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
```

3. **/dev/sdb1** および **/dev/sdc1** の物理ボリュームを置き換えます。

```
# lvconvert --replace /dev/sdb1 --replace /dev/sdc1 my_vg/my_lv
```

4. 代替の RAID1 アレイを確認します。

```
# lvs -a -o name,copy_percent,devices my_vg

  LV              Copy%  Devices
  my_lv            60.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rimage_2]       /dev/sde1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
  [my_lv_rmeta_2]        /dev/sde1(0)
```

68.7.18.2. LVM RAID のデバイスに障害が発生しました。

RAID は従来の LVM ミラーリングとは異なります。LVM ミラーリングでは、障害が発生したデバイスを削除する必要がありました。削除しないと、ミラー化論理ボリュームがハングします。RAID アレイ

は、障害があるデバイスがあっても稼働し続けることができます。RAID1 以外の RAID タイプでデバイスを削除すると、レベルが低い RAID に変換されます (たとえば、RAID6 から RAID5、もしくは RAID4 または RAID5 から RAID0)。

そのため、障害のあるデバイスを無条件に削除してから交換するのではなく、**lvconvert** コマンドで **--repair** 引数を使用して、RAID ボリュームのデバイスを 1 回で置き換えることができます。

68.7.18.3. 論理ボリュームの障害が発生した RAID デバイスの交換

LVM RAID デバイス障害が一時的な障害であったり、障害が発生したデバイスの修復が可能な場合は、障害が発生したデバイスの復旧を開始できます。

前提条件

- 以前に不具合を起こしたデバイスが機能するようになりました。

手順

- RAID デバイスが含まれる論理ボリュームを更新します。

```
# lvchange --refresh my_vg/my_lv
```

検証手順

- 復元されたデバイスで論理ボリュームを調べます。

```
# lvs --all --options name,devices,lv_attr,lv_health_status my_vg
```

68.7.18.4. 論理ボリュームに障害が発生した RAID デバイスの交換

この手順では、LVM RAID 論理ボリュームで物理ボリュームとして機能する障害のあるデバイスを置き換えます。

前提条件

- ボリュームグループには、障害が発生したデバイスを置き換えるのに十分な空き容量を提供する物理ボリュームが含まれています。
  ボリュームグループに十分な空きエクステントがある物理ボリュームがない場合は、**vgextend** ユーティリティーを使用して、十分なサイズの物理ボリュームを新たに追加します。

手順

1. 以下の例では、RAID 論理ボリュームが次のように配置されます。

```
# lvs --all --options name,copy_percent,devices my_vg

  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdc1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
```

```
  [my_lv_rmeta_0]         /dev/sde1(0)
  [my_lv_rmeta_1]         /dev/sdc1(0)
  [my_lv_rmeta_2]         /dev/sdd1(0)
```

2. **/dev/sdc** デバイスに障害が発生した場合、**lvs** コマンドの出力は以下のようになります。

```
# lvs --all --options name,copy_percent,devices my_vg

  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         [unknown](1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          [unknown](0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

3. 障害が発生したデバイスを交換して、論理ボリュームを表示します。

```
# lvconvert --repair my_vg/my_lv

  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
  Faulty devices in my_vg/my_lv successfully replaced.
```

オプション:障害が発生したデバイスを交換する物理ボリュームを手動で指定するには、コマンドの最後に物理ボリュームを追加します。

```
# lvconvert --repair my_vg/my_lv replacement_pv
```

4. 代替の論理ボリュームを調べます。

```
# lvs --all --options name,copy_percent,devices my_vg

  /dev/sdc: open failed: No such device or address
  /dev/sdc1: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  LV              Cpy%Sync Devices
  my_lv           43.79    my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdb1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
```

```
[my_lv_rmeta_0]       /dev/sde1(0)
[my_lv_rmeta_1]       /dev/sdb1(0)
[my_lv_rmeta_2]       /dev/sdd1(0)
```

障害が発生したデバイスをボリュームグループから削除するまで、LVM ユーティリティーは、障害が発生したデバイスが見つけられないことを示しています。

5. 障害が発生したデバイスをボリュームグループから削除します。

```
# vgreduce --removemissing VG
```

68.7.19. RAID 論理ボリュームでのデータ整合性の確認 (RAID スクラビング)

LVM は、RAID 論理ボリュームのスクラビングに対応します。RAID スクラビングは、アレイ内のデータおよびパリティーブロックをすべて読み込み、それが一貫しているかどうかを確認するプロセスです。

手順

1. オプション:スクラビングプロセスが使用する I/O 帯域幅を制限します。
   RAID スクラビング操作を実行する際に、**sync** 操作で必要になるバックグラウンド I/O は、その他の I/O (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。リカバリースロットルを実装してスクラビング操作のレートを制御できます。

   次の手順で、**lvchange --syncaction** コマンドに以下のオプションを追加します。

   --maxrecoveryrate Rate[bBsSkKmMgG]

   操作が通常の I/O 操作に押し出すように、最大復旧速度を設定します。復旧速度を 0 に設定すると、操作がバインド解除されることを意味します。

   --minrecoveryrate Rate[bBsSkKmMgG]

   最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。

   Rate 値は、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しないと、オプションはデバイスごとの 1 秒あたらりの kiB を想定します。

2. アレイ内の不一致数を修復せずに、アレイ内の不一致の数を表示します。

```
# lvchange --syncaction check vg/raid_lv
```

3. アレイ内の不一致を修正します。

```
# lvchange --syncaction repair vg/raid_lv
```

注記

lvchange --syncaction repair 操作は、**lvconvert --repair** 操作と同じ機能を実行しません。

- **lvchange --syncaction repair** 操作は、アレイでバックグラウンドの同期操作を開始します。
- **lvconvert --repair** 操作は、ミラーまたは RAID 論理ボリュームの障害が発生したデバイスを修復するか、置き換えます。

4. オプション:スクラビング操作に関する情報を表示します。

```
# lvs -o +raid_sync_action,raid_mismatch_count vg/lv
```

- **raid_sync_action** フィールドは、RAID ボリュームが現在実行している同期操作を表示します。これには、以下のいずれかの値を使用できます。

  idle
  すべての同期操作が完了している (何も実行していません)。

  resync
  アレイを初期化、またはマシン障害後の復旧を実行する。

  recover
  アレイ内のデバイスを置き換える。

  check
  アレイの不一致を検索する。

  repair
  不一致を検索し、修復する。

- **raid_mismatch_count** フィールドは、**check** 操作時に検出された不一致の数を表示します。
- **Cpy%Sync** フィールドは、**sync** 操作の進捗を表示します。
- **lv_attr** フィールドは、追加のインジケーターを提供します。このフィールドのビット 9 は、論理ボリュームの正常性を示し、以下のインジケーターに対応しています。
  - (**m**) (不一致) は、RAID 論理ボリュームに不一致があることを示します。この文字は、スクラビング操作で RAID に一貫性がない部分があることを検出した後に表示されます。
  - (**r**) (更新) は、LVM がデバイスラベルを読み取り、デバイスを稼働できると認識した場合でも、RAID アレイのデバイスに障害が発生し、カーネルがこれを障害と認識していることを示します。デバイスが利用可能になったことをカーネルに通知するように論理ボリュームを更新するか、デバイスに障害が発生したと思われる場合はデバイスを交換します。

関連情報

- 詳細は、**lvchange(8)** および **lvmraid(7)** の man ページを参照してください。

68.7.20. RAID レベルの変更 (RAID テイクオーバー)

LVM は、Raid **テイクオーバー** をサポートします。これは、RAID 論理ボリュームの RAID レベルを別のレベル (たとえば RAID 5 から RAID 6) へ変えることを意味します。RAID レベルの変更は、通常、デバイスの耐障害性を増減したり、論理ボリュームのストライプ化をやり直すために行われます。RAID テイクオーバーには **lvconvert** を使用します。RAID テイクオーバーの詳細と、**lvconvert** を使用して RAID 論理ボリュームを変換する例は、man ページの **lvmraid**(7) を参照してください。

68.7.21. RAID ボリュームの属性の変更 (RAID 再成形)

RAID **再成形** とは、同じ RAID レベルを維持しつつ、RAID 論理ボリュームの属性を変更することを指します。変更できる属性には、RAID レイアウト、ストライプのサイズ、ストライプの数などがあります。RAID 再成形と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再生成する例は、man ページの **lvmraid**(7) を参照してください。

68.7.22. RAID1 論理ボリュームでの I/O 操作の制御

lvchange コマンドの **--writemostly** パラメーターおよび **--writebehind** パラメーターを使用して、RAID1 論理ボリュームのデバイスに対する I/O 操作を制御できます。これらのパラメーターを使用する書式は以下のとおりです。

- **--[raid]writemostly PhysicalVolume[:{t|y|n}]**
  RAID1 論理ボリュームのデバイスを **write-mostly** とマークします。これらのドライブのすべての読み取りは、必要でない限り回避されます。このパラメーターを設定することにより、ドライブに対する I/O 操作の回数を最小限に抑えることができます。デフォルトでは、論理ボリュームに指定した物理ボリュームの **write-mostly** 属性を yes に設定します。 **:n** を物理ボリュームに追加して **write-mostly** フラグを削除したり、**:t** を指定して値を切り替えたりできます。**--writemostly** 引数は、1 つのコマンドで複数回指定できるため、1 回で論理ボリュームのすべての物理ボリュームで、write-mostly 属性を切り替えることができます。

- **--[raid]writebehind IOCount**
  write-mostly というマークが付いている RAID1 論理ボリュームのデバイスに許可される、未処理の書き込みの最大数を指定します。この値を上回ると書き込みは同期され、設定要素になっているデバイスへの書き込みがすべて、アレイが書き込みの完了を知らせる前に完了してしまいます。この値をゼロに設定すると、設定はクリアになり、システムが値を任意に選択できるようになります。

68.7.23. RAID 論理ボリュームのリージョンサイズの変更

RAID 論理ボリュームを作成すると、論理ボリュームのリージョンサイズは、**/etc/lvm/lvm.conf** ファイルの **raid_region_size** パラメーターの値になります。このデフォルト値は、**lvcreate** コマンドの **-R** オプションで上書きできます。

RAID 論理ボリュームを作成したら、**lvconvert** コマンドの **-R** オプションで、ボリュームのリージョンサイズを変更できます。以下の例では、論理ボリューム **vg/raidlv** のリージョンサイズを 4096K に変更します。リージョンサイズを変更する場合は、RAID ボリュームを同期する必要があります。

```
# lvconvert -R 4096K vg/raid1
Do you really want to change the region_size 512.00 KiB of LV vg/raid1 to 4.00 MiB? [y/n]: y
  Changed region size on RAID LV vg/raid1 to 4.00 MiB.
```

68.8. 論理ボリュームのスナップショット

LVM スナップショット機能を使用すると、サービスを中断することなく、ある時点でのボリュームの仮想イメージ (**/dev/sda** など) を作成できます。

68.8.1. スナップショットボリュームの概要

スナップショットを作成した後で元のボリューム (スナップショットの元になるボリューム) を変更すると、スナップショット機能は、ボリュームの状態を再構築できるように、変更前の変更されたデータ領域のコピーを作成します。スナップショットを作成しても、作成元への完全な読み取り/書き込みのアクセスは引き続き可能です。

スナップショットは、スナップショットの作成後に変更したデータ部分のみをコピーするため、スナップショット機能に必要なストレージは最小限になります。たとえば、コピー元がほとんど更新されない場合は、作成元の 3 ~ 5 % の容量があれば十分にスナップショットを維持できます。バックアップ手順に代わるものではありません。スナップショットコピーは仮想コピーであり、実際のメディアバックアップではありません。

作成元のボリュームへの変更を保管するために確保する領域は、スナップショットのサイズによって異なります。たとえば、スナップショットを作成してから作成元を完全に上書きする場合に、その変更の保管に必要なスナップショットのサイズは、作成元のボリュームと同等か、それ以上になります。スナップショットのサイズは定期的に監視する必要があります。たとえば、**/usr** など、その大部分が読み取り用に使用されるボリュームの短期的なスナップショットに必要な領域は、**/home** のように大量の書き込みが行われるボリュームの長期的なスナップショットに必要な領域よりも小さくなります。

スナップショットが満杯になると、作成元のボリュームの変更を追跡できなくなるため、そのスナップショットは無効になります。ただし、スナップショットが無効になるのを防ぐために、使用量が **snapshot_autoextend_threshold** 値を超えるたびにスナップショットを自動的に拡張するように LVM を設定できます。スナップショットは完全にサイズ変更可能で、次の操作を実行できます。

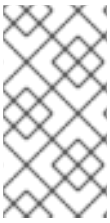
- ストレージ容量に余裕がある場合は、スナップショットボリュームのサイズを大きくして、削除されないようにすることができます。
- スナップショットのボリュームサイズが必要以上に大きければ、そのボリュームのサイズを縮小して、他の論理ボリュームで必要となる領域を確保できます。

スナップショットボリュームには、次の利点があります。

- 最も一般的な用途は、継続的にデータを更新している稼動中のシステムを停止せずに、論理ボリューム上でバックアップを実行する必要がある場合にスナップショットを撮ることです。
- スナップショットファイルシステムで **fsck** コマンドを実行し、ファイルシステムの整合性をチェックすれば、複製元のファイルシステムを修復する必要があるかどうかを判断できます。
- スナップショットは読み取りおよび書き込み用であるため、スナップショットを撮ってそのスナップショットにテストを実行することにより、実際のデータに触れることなく、実稼働データにアプリケーションのテストを実行できます。
- LVM ボリュームを作成して、Red Hat の仮想化と併用することが可能です。LVM スナップショットを使用して、仮想ゲストイメージのスナップショットを作成できます。このスナップショットは、最小限のストレージを使用して、既存のゲストの変更や新規ゲストの作成を行う上で利便性の高い方法を提供します。

68.8.2. 元のボリュームのスナップショット作成

-s または **--size** 引数の後に必要なサイズを指定して **lvcreate** コマンドを使用し、元のボリューム (作成元) のスナップショットを作成します。ボリュームのスナップショットは書き込み可能です。デフォルトでは、シンプロビジョニングされたスナップショットと比較すると、通常のアクティベーションコマンドを実行中に、作成元のボリュームを使用して、スナップショットのボリュームを有効にします。LVM は、元のボリュームのサイズとボリュームに必要なメタデータサイズの合計よりも大きいスナップショットボリュームの作成をサポートしていません。これより大きいスナップショットボリュームを指定すると、LVM は作成元のサイズに必要なスナップショットボリュームを作成します。



注記

クラスター内のノードは LVM スナップショットをサポートしていません。共有ボリュームグループ内にスナップショットボリュームは作成できません。ただし、共有論理ボリューム上でデータの一貫したバックアップ作成が必要な場合は、ボリュームを排他的にアクティブにした上で、スナップショットを作成できます。

次の手順は、**origin** という名前の論理ボリュームを作成し、その論理ボリュームから、**snap** という名前のスナップショットボリュームを作成します。

前提条件

- ボリュームグループ **vg001** を作成している。詳細は、LVM ボリュームグループの作成 を参照してください。

手順

1. ボリュームグループ **vg001** から、論理ボリューム **origin** を作成します。

```
# lvcreate -L 1G -n origin vg001
Logical volume "origin" created.
```

2. 名前が **snap** で、サイズが 100 MB のスナップショット論理ボリューム **/dev/vg001/origin** を作成します。

```
# lvcreate --size 100M --name snap --snapshot /dev/vg001/origin
  Logical volume "snap" created.
```

元の論理ボリュームにファイルシステムが含まれている場合は、任意のディレクトリー上でスナップショット論理ボリュームをマウントしてから、そのファイルシステムのコンテンツにアクセスし、元のファイルシステムが更新を継続している間にバックアップを実行できます。

3. 元のボリュームと、使用されているスナップショットボリュームの現在の割合を表示します。

```
# lvs -a -o +devices
  LV      VG    Attr       LSize   Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---   1.00g                                             /dev/sde1(0)
  snap vg001  swi-a-s--- 100.00m     origin 0.00                                    /dev/sde1(256)
```

lvdisplay /dev/vg001/origin コマンドを使用して、論理ボリューム **/dev/vg001/origin** のステータスと、すべてのスナップショット論理ボリュームおよびそれらのステータス (アクティブまたは非アクティブなど) を表示することもできます。

警告

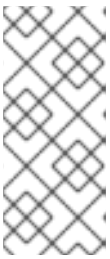
複製元ボリュームが変更されると、スナップショットのサイズが拡大されるため、**lvs** コマンドを使用して、スナップショットボリュームのパーセンテージを定期的に監視して、満杯にならないように確認することが重要です。スナップショットは、100% になると完全に消失します。これは、作成元ボリュームの変更されていない部分に書き込みが行われるため、スナップショットが必ず破損するためです。

4. ただし、使用量が 100% の場合にスナップショットが無効になるのを回避するために、使用量が **snapshot_autoextend_threshold** 値を超えるたびにスナップショットを自動的に拡張するように LVM を設定できます。**/etc/lvm.conf** ファイルから **snapshot_autoextend_threshold** および **snapshot_autoextend_percent** オプションの既存の値を表示し、要件に従ってそれらを編集します。
次の例では、**snapshot_autoextend_threshold** オプションを 100 未満の値に設定し、**snapshot_autoextend_percent** オプションをスナップショットボリュームを拡張するための要件に応じた値に設定します。

```
# vi /etc/lvm.conf
snapshot_autoextend_threshold = 70
snapshot_autoextend_percent = 20
```

次のコマンドを実行して、このスナップショットを手動で拡張することもできます。

```
# lvextend -L+100M /dev/vg001/snap
```



注記

この機能には、ボリュームグループに未割り当てのスペースが必要です。同様に、スナップショットの自動拡張を実行しても、スナップショットに必要なサイズとして計算される最大サイズを超えて拡張されることはありません。スナップショットのサイズが複製元のボリュームを包含できるまで拡大されると、スナップショットの自動拡張はモニターされなくなります。

関連情報

- **lvcreate (8)**、**lvextend (8)**、および **lvs (8)** の man ページ
- **/etc/lvm/lvm.conf** ファイル

68.8.3. スナップショットと元のボリュームのマージ

--merge オプションを指定して **lvconvert** コマンドを使用し、スナップショットを元のボリューム (作成元) にマージします。データやファイルを失った場合や、システムを以前の状態に復元する必要がある場合に、システムのロールバックを実行できます。スナップショットボリュームをマージすると、作成された論理ボリュームには、元のボリュームの名前、マイナー番号、および UUID が含まれます。
マージの進行中、作成元に対する読み取りまたは書き込みは、マージ中のスナップショットに対して実行されているかのように見えます。マージが完了すると、マージされたスナップショットは削除されます。

作成元のボリュームとスナップショットボリュームの両方が起動されておらず、アクティブでない場合、マージはすぐに開始されます。それ以外の場合は、作成元またはスナップショットのいずれかがアクティブ化され、両方が閉じられた後にマージが開始されます。スナップショットを閉じることができない作成元 (**root** ファイルシステムなど) にマージできるのは、作成元のボリュームがアクティブ化された後です。

手順

1. スナップショットボリュームをマージします。以下のコマンドは、スナップショットボリューム **vg001/snap** をその **作成元** にマージします。

```
# lvconvert --merge vg001/snap
Merging of volume vg001/snap started.
  vg001/origin: Merged: 100.00%
```

2. 元のボリュームを表示します。

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---  1.00g                                       /dev/sde1(0)
```

関連情報

- **lvconvert(8)** の man ページ

68.9. シンプロビジョニングされたボリューム (シンボリューム) の作成および管理

Red Hat Enterprise Linux は、シンプロビジョニングされたスナップショットボリュームと論理ボリュームをサポートします。

論理ボリュームとスナップショットボリュームは、シンプロビジョニングできます。

- シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。
- シンプロビジョニングされたスナップショットボリュームを使用すると、同じデータボリュームにより多くの仮想デバイスを格納できます。

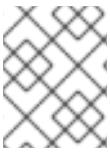
68.9.1. シンプロビジョニングの概要

最新のストレージスタックの多くは、シックプロビジョニングとシンプロビジョニングのどちらかを選択できるようになりました。

- シックプロビジョニングは、ブロックストレージの従来の動作を実行でき、実際の使用状況に関係なくブロックが割り当てられます。
- シンプロビジョニングは、ブロックストレージのプールよりも大きいサイズ (データを保存する物理デバイスよりもサイズが大きくなる可能性のあるブロックストレージ) をプロビジョニングできるので、過剰にプロビジョニングされる可能性があります。個々のブロックは実際に使用されるまで割り当てられないため、オーバープロビジョニングが発生する可能性があります。同じプールを共有する複数のシンプロビジョニングされたデバイスがある場合に、これらのデバイスは過剰にプロビジョニングされる可能性があります。

シンプロビジョニングを使用すると、物理ストレージをオーバーコミットでき、代わりにシンプールと呼ばれる空き領域のプールを管理できます。アプリケーションで必要な場合は、このシンプールを任意の数のデバイスに割り当てることができます。シンプールは、ストレージ領域をコスト効率よく割り当てる必要がある場合に、動的に拡張できます。

たとえば、10 人のユーザーから、各自のアプリケーションに使用するファイルシステムをそれぞれ 100GB 要求された場合には、各ユーザーに 100GB のファイルシステムを作成します (ただし、実際には 100GB 未満のストレージが、必要に応じて使用されます)。



注記

シンプロビジョニングを使用する場合は、ストレージプールを監視して、使用可能な物理スペースが不足したときに容量を追加することが重要です。

以下は、シンプロビジョニングされたデバイスを使用する利点です。

- 使用可能な物理ストレージよりも大きい論理ボリュームを作成できます。
- 同じデータボリュームに保存する仮想デバイスを増やすことができます。
- データ要件をサポートするために論理的かつ自動的に拡張できるファイルシステムを作成できます。未使用のブロックはプールに戻され、プール内の任意のファイルシステムで使用できます。

シンプロビジョニングされたデバイスを使用する場合に発生する可能性のある欠点は次のとおりです。

- シンプロビジョニングされたボリュームには、使用可能な物理ストレージが不足するという固有のリスクがあります。基盤となるストレージを過剰にプロビジョニングした場合に、使用可能な物理ストレージが不足しているために停止する可能性があります。たとえば、バッキング用に 1T の物理ストレージのみを使用して、10T のシンプロビジョニングストレージを作成する場合に、この 1T が使い果たされると、ボリュームは使用不可または書き込み不能になります。
- ボリュームが、シンプロビジョニングデバイスの後の階層に破棄するように送信していない場合には、使用量の計算が正確でなくなります。たとえば、**-o discard mount** オプションを指定せずにファイルシステムを配置し、シンプロビジョニングされたデバイス上で **fstrim** を定期的に実行しないと、以前に使用されたストレージの割り当てが解除されることはありません。このような場合に、実際に使用していなくても、時間の経過とともにプロビジョニングされた量をすべて使用することになります。
- 使用可能な物理スペースが不足しないように、論理的および物理的な使用状況を監視する必要があります。
- スナップショットのあるファイルシステムでは、コピーオンライト (CoW) 操作が遅くなる可能性があります。
- データブロックは複数のファイルシステム間で混在する可能性があり、エンドユーザーにそのように表示されない場合でも、基盤となるストレージのランダムアクセス制限につながります。

68.9.2. シンプロビジョニングされた論理ボリュームの作成

シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。シンプロビジョニングされたボリュームセットを作成すると、システムは要求されるストレージの全量を割り当てる代わりに、実際に使用する容量を割り当てることができます。

lvcreate コマンドに **-T** (または **--thin**) オプションを付けて、シンプールまたはシンボリュームを作成できます。また、**lvcreate** の **-T** オプションを使用して、1つのコマンドで同時にシンプールとシンプロビジョニングされたボリュームの両方を作成することもできます。この手順では、シンプロビジョニングされた論理ボリュームを作成および拡張する方法について説明します。

前提条件

- ボリュームグループを作成している。詳細は、LVM ボリュームグループの作成 を参照してください。

手順

1. シンプールを作成します。

```
# lvcreate -L 100M -T vg001/mythinpool
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

物理領域のプールを作成しているため、プールのサイズを指定する必要があります。**lvcreate** コマンドの **-T** オプションでは引数を使用できません。コマンドで指定する他のオプションをもとに、作成するデバイスのタイプを決定します。次の例に示すように、追加のパラメーターを使用してシンプールを作成することもできます。

- また、**lvcreate** コマンドの --**--thinpool** パラメーターを指定して、シンプールを作成することもできます。**-T** オプションとは異なり、**-thinpool** パラメーターでは、作成するシンプール論理ボリュームの名前を指定する必要があります。次の例では、**-thinpool** パラメーターを使用して、サイズが 100M のボリュームグループ vg001 に mythinpool という名前のシンプールを作成します。

```
# lvcreate -L 100M --thinpool mythinpool vg001
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

- プールの作成ではストライピングがサポートされているため、**-i** および **-I** オプションを使用してストライプを作成できます。次のコマンドは、ボリュームグループ vg001 に、thinpool という名前の 2 つの 64 KB ストライプと 256 KB のチャンクサイズを持つ 100 M シンプールを作成します。また、vg001/thinvolume という名前でサイズが 1T のシンボリュームを作成します。

注記

ボリュームグループに十分な空き領域がある 2 つの物理ボリュームがあることを確認してください。そうでない場合にはシンプールを作成できません。

```
# lvcreate -i 2 -I 64 -c 256 -L 100M -T vg001/thinpool -V 1T --name thinvolume
```

2. シンボリュームを作成します。

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
```

```
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

このような場合には、ボリュームを含むプールのサイズよりも大きい、ボリュームの仮想サイズを指定しています。次の例に示すように、追加のパラメーターを使用してシンボリュームを作成することもできます。

- シンボリュームとシンプールの両方を作成するには、**lvcreate** コマンドの **-T** オプションを使用して、サイズと仮想サイズの両方の引数を指定します。

```
# lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger
automatic extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

- 残りの空き領域を使用してシンボリュームとシンプールを作成するには、**100%FREE** オプションを使用します。

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool -n thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of data.
  Logical volume "thinvolume" created.
```

- 既存の論理ボリュームをシンプールボリュームに変換するには、**lvconvert** コマンドの **--thinpool** パラメーターを使用します。また、**-poolmetadata** パラメーターを **--thinpool** パラメーターと組み合わせて使用して、既存の論理ボリュームをシンプールボリュームのメタデータボリュームに変換する必要があります。
  以下の例は、ボリュームグループ vg001 の既存の論理ボリューム lv1 を、シンプールボリュームに変換します。また、ボリュームグループ vg001 の既存の論理ボリューム lv2 を、そのシンプールボリュームのメタデータボリュームに変換します。

```
# lvconvert --thinpool vg001/lv1 --poolmetadata vg001/lv2
  Converted vg001/lv1 to thin pool.
```

注記

論理ボリュームをシンプールボリュームまたはシンプールメタデータボリュームに変換すると、論理ボリュームのコンテンツが破棄されます。**lvconvert** はデバイスのコンテンツを保存するのではなく、コンテンツを上書きするためです。

- デフォルトでは、**lvcreate** コマンドは、次の式を使用してシンプールメタデータ論理ボリュームのおおよそのサイズを設定します。

```
Pool_LV_size / Pool_LV_chunk_size * 64
```

スナップショットが大量にある場合や、シンプールのサイズが小さく、後で急激に大きくなることが予測される場合は、**lvcreate** コマンドの **--poolmetadatasize** パラメーターで、シンプールのメタデータボリュームのデフォルト値を大きくしないといけない場合があり

ます。シンプールのメタデータ論理ボリュームで対応している値は 2MiB ~ 16GiB です。

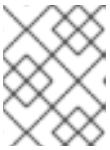
次の例は、シンプールのメタデータボリュームのデフォルト値を増やす方法を示しています。

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool --poolmetadatasize 16M -n
thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "thinvolume" created.
```

3. 作成されたシンプールとシンボリュームを表示します。

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 100.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 100.00m
/dev/sda(1)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

4. オプション:**lvextend** コマンドを使用して、シンプールのサイズを拡張します。ただし、シンプールのサイズを縮小することはできません。



注記

シンプールとシンボリュームの作成中に **-l100%FREE** 引数を使用すると、このコマンドは失敗します。

以下のコマンドは、既存のシンプールのサイズ (100M) を変更し、さらに 100M 分を拡張します。

```
# lvextend -L+100M vg001/mythinpool
  Size of logical volume vg001/mythinpool_tdata changed from 100.00 MiB (25 extents) to
200.00 MiB (50 extents).
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (200.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.

  Logical volume vg001/mythinpool successfully resized
```

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 200.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(1)
```

```
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(27)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

5. オプション:シンプールとシンボリュームの名前を変更するには、次のコマンドを使用します。

```
# lvrename vg001/mythinpool vg001/mythinpool1
  Renamed "mythinpool" to "mythinpool1" in volume group "vg001"

# lvrename vg001/thinvolume vg001/thinvolume1
  Renamed "thinvolume" to "thinvolume1" in volume group "vg001"
```

名前を変更した後に、シンプールとシンボリュームを表示します。

```
# lvs
  LV          VG       Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
mythinpool1 vg001    twi-a-tz 100.00m                    0.00
thinvolume1 vg001    Vwi-a-tz   1.00g mythinpool1        0.00
```

6. オプション:シンプールを削除するには、次のコマンドを使用します。

```
# lvremove -f vg001/mythinpool1
  Logical volume "thinvolume1" successfully removed.
  Logical volume "mythinpool1" successfully removed.
```

関連情報

- **lvcreate (8)**、**lvrename (8)**、**lvs (8)**、および **lvconvert (8)** の man ページ

68.9.3. チャンクサイズの概要

チャンクは、スナップショットストレージ専用の物理ディスクの最大単位です。

チャンクサイズを使用するには、以下の基準を使用します。

- チャンクサイズが小さいほどメタデータが増え、パフォーマンスも低下しますが、スナップショットで領域の使用率が向上します。
- チャンクサイズが大きいほどメタデータ操作は少なくなりますが、スナップショットの領域効率が低下します。

デフォルトでは、**lvm2** は 64 KiB のチャンクサイズから開始し、そのチャンクサイズに対して適切なメタデータサイズを推定します。**lvm2** が作成および使用できるメタデータの最小サイズは 2 MiB です。メタデータのサイズを 128 MiB より大きくする必要がある場合、lvm2 はチャンクサイズを増やすため、メタデータのサイズはコンパクトなまま保たれます。しかし、これによりチャンクサイズの値が大きくなり、スナップショットの使用におけるスペース効率が低下する可能性があります。このような場合、チャンクサイズを小さくし、メタデータサイズを大きくすることを推奨します。

要件に従ってチャンクサイズを指定するには、**-c** または **--chunksize** パラメーターを使用して、**lvm2** の推定チャンクサイズを無効にします。シンプールの作成後はチャンクサイズを変更できないことに注意してください。

ボリュームデータサイズが TiB の範囲にある場合は、サポートされる最大サイズである約 15.8 GiB をメタデータサイズとして使用し、要件に従ってチャンクサイズを設定します。ただし、ボリュームのデータサイズを拡張し、チャンクサイズを小さくする必要がある場合には、メタデータサイズを拡大できないことに注意してください。

注記

不適切なチャンクサイズとメタデータサイズの組み合わせを使用すると、ユーザーが **metadata** スペースを使い果たしたり、アドレス指定可能な最大シンプールデータサイズが制限されているためにシンプールサイズをそれ以上拡張できなくなったりして、問題が発生する可能性があります。

関連情報

- **lvmthin (7)** man ページ

68.9.4. シンプロビジョニングのスナップショットボリューム

Red Hat Enterprise Linux は、シンプロビジョニングされたスナップショットボリュームをサポートします。シン論理ボリュームのスナップショットにより、シン論理ボリューム (LV) を作成することもできます。シンプロビジョニングのスナップショットボリュームには、他のシンボリュームと同じ特性があります。ボリュームのアクティブ化、拡張、名前変更、削除、さらにはスナップショット作成も個別に行うことができます。

注記

すべてのシンボリュームや、LVM スナップショットボリュームと同様に、シンプロビジョニングのスナップショットボリュームは、クラスターのノード間では対応していません。スナップショットボリュームは、1 つのクラスターノードで排他的にアクティブにする必要があります。

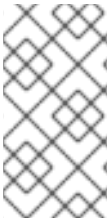
従来のスナップショットでは、作成されたスナップショットごとに新しい領域を割り当てる必要があります。この領域では、作成元に変更が加えられてもデータが保持されます。ただし、シンプロビジョニングスナップショットは、作成元と同じ領域を共有します。シン LV のスナップショットは、シン LV とそのスナップショットのいずれかに共通のデータブロックが共有されるので効率的です。シン LV のスナップショットを作成することも、他のシンスナップショットから作成することもできます。再帰スナップショットに共通のブロックもシンプールで共有されます。

シンプロビジョニングのスナップショットボリュームの利点は以下のとおりです。

- オリジンのスナップショットの数を増やしても、パフォーマンスへの影響はほとんどありません。
- シンスナップショットボリュームは、新しいデータのみが書き込まれ、各スナップショットにコピーされないため、ディスク使用量を減らすことができます。
- 従来のスナップショットの要件でしたが、シンスナップショットボリュームを作成元と同時にアクティブにする必要はありません。
- スナップショットからオリジンを復元する場合、シンスナップショットをマージする必要はありません。オリジンを削除して、代わりにスナップショットを使用できます。従来のスナップショットには、コピーバックする必要がある変更を保存する別のボリュームがあります。つまり、元のスナップショットにマージしてリセットする必要があります。

- 従来のスナップショットと比較して、許可されるスナップショットの上限数がはるかに増えています。

シンプロビジョニングのスナップショットボリュームを使用する利点は数多くありますが、従来の LVM スナップショットボリューム機能の方がニーズに適している場合もあります。すべてのタイプのボリュームで従来のスナップショットを使用できます。ただし、シンスナップショットを使用するには、シンプロビジョニングを使用する必要があります。



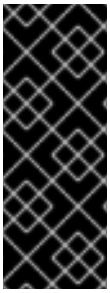
注記

シンプロビジョニングのスナップショットボリュームのサイズを制限することはできません。スナップショットは、必要な場合はシンプール内の全領域を使用します。一般的には、使用するスナップショットの形式を決定する際に、使用しているサイトの特定要件を考慮するようにしてください。

デフォルトで、シンスナップショットボリュームは、通常のアクティブ化コマンドの実行時に省略されます。

68.9.5. シンプロビジョニングのスナップショットボリュームの作成

シンプロビジョニングされたスナップショットボリュームを使用すると、同じデータボリュームにより多くの仮想デバイスを格納できます。



重要

シンプロビジョニングのスナップショットボリュームを作成する場合、ボリュームのサイズは指定しません。サイズパラメーターを指定すると、作成されるスナップショットはシンプロビジョニングのスナップショットボリュームにはならず、データを保管するためにシンプールを使用することもありません。たとえば、**lvcreate -s vg/thinvolume -L10M** コマンドは、作成元ボリュームがシンボリュームであっても、シンプロビジョニングのスナップショットを作成しません。

シンプロビジョニングのスナップショットは、シンプロビジョニングされた作成元ボリューム用に作成するか、シンプロビジョニングされていない作成元ボリューム用にも作成できます。次の手順では、シンプロビジョニングされたスナップショットボリュームを作成するさまざまな方法について説明します。

前提条件

- シンプロビジョニングされた論理ボリュームを作成ている。詳細は、シンプロビジョニングの概要 を参照してください。

手順

- シンプロビジョニングされたスナップショットボリュームを作成します。以下のコマンドは、シンプロビジョニングされた論理ボリューム **vg001/thinvolume** で、シンプロビジョニングのスナップショットボリューム (名前: **mysnapshot1**) を作成します。

```
# lvcreate -s --name mysnapshot1 vg001/thinvolume
  Logical volume "mysnapshot1" created
```

```
# lvs
  LV        VG      Attr     LSize   Pool      Origin     Data%  Move Log Copy%  Convert
```

```
  mysnapshot1 vg001    Vwi-a-tz   1.00g mythinpool thinvolume   0.00
  mythinpool  vg001    twi-a-tz 100.00m                         0.00
  thinvolume  vg001    Vwi-a-tz   1.00g mythinpool              0.00
```

注記

シンプロビジョニングを使用する場合は、ストレージ管理者がストレージプールを監視し、容量が満杯になり始めたら容量を追加することが重要です。シンボリュームのサイズを拡張する方法は、シンプロビジョニングされた論理ボリュームの作成 を参照してください。

- シンプロビジョニングされていない論理ボリュームの、シンプロビジョニングされたスナップショットを作成することもできます。シンプロビジョニングされていない論理ボリュームはシンプール内に含まれていないため、外部の複製元と呼ばれます。外部の作成元ボリュームは、複数の異なるシンプールからであっても、多くのシンプロビジョニングのスナップショットボリュームで使用でき、共有できます。外部の作成元は、シンプロビジョニングのスナップショットが作成される際に非アクティブであり、かつ読み取り専用である必要があります。次の例では、**origin_volume** という名前の読み取り専用の非アクティブな論理ボリュームのシンスナップショットボリュームを作成します。このシンプロビジョニングのスナップショットボリュームの名前は **mythinsnap** です。論理ボリューム **origin_volume** は、既存のシンプール **vg001/pool** を使用する、ボリュームグループ **vg001** 内のシンプロビジョニングのスナップショットボリューム **mythinsnap** に対する外部の作成元になります。作成元のボリュームは、スナップショットボリュームと同じボリュームグループに属している必要があります。作成元の論理ボリュームを指定するときは、ボリュームグループを指定しないでください。

```
# lvcreate -s --thinpool vg001/pool origin_volume --name mythinsnap
```

- 以下のコマンドを実行して、最初のスナップショットボリュームの 2 番目のシンプロビジョニングのスナップショットボリュームを作成できます。

```
# lvcreate -s vg001/mysnapshot1 --name mysnapshot2
Logical volume "mysnapshot2" created.
```

3 番目のシンプロビジョニングされたスナップショットボリュームを作成するには、次のコマンドを使用します。

```
# lvcreate -s vg001/mysnapshot2 --name mysnapshot3
Logical volume "mysnapshot3" created.
```

検証

- シンスナップショット論理ボリュームのすべての祖先と子孫のリストを表示します。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV          Ancestors                       Descendants
  mysnapshot2 mysnapshot1,thinvolume          mysnapshot3
  mysnapshot1 thinvolume                      mysnapshot2,mysnapshot3
  mysnapshot3 mysnapshot2,mysnapshot1,thinvolume
  mythinpool
  thinvolume                                  mysnapshot1,mysnapshot2,mysnapshot3
```

ここでは、以下のようになります。

- thinvolume は、ボリュームグループ vg001 で元となるボリュームです。
- mysnapshot1 は thinvolume のスナップショットです。
- mysnapshot2 は mysnapshot1 のスナップショットです。
- mysnapshot3 は mysnapshot2 のスナップショットです、

注記

lv_ancestors フィールドと **lv_descendants** フィールドには、既存の依存関係が表示されます。ただし、削除されたエントリーは追跡しません。このチェーンの最中にエントリーが削除されると、依存関係チェーンが壊れるためです。

関連情報

- **lvcreate(8)** の man ページ

68.10. キャッシュを有効にして論理ボリュームのパフォーマンスを改善

LVM 論理ボリュームにキャッシュを追加して、パフォーマンスを向上できます。LVM は、SSD などの高速なデバイスを使用して、論理ボリュームに I/O 操作をキャッシュします。

以下の手順では、高速デバイスから特別な論理ボリュームを作成し、この特別な論理ボリュームを元の論理ボリュームに接続して、パフォーマンスを向上させます。

68.10.1. LVM でのキャッシュの取得方法

LVM は、以下のようなキャッシュの取得方法を提供します。論理ボリューム上のさまざまなタイプの I/O パターンに適しています。

dm-cache

このメソッドは、高速なボリユームで頻繁に使用されるデータをキャッシュして、このようなデータへのアクセス時間を短縮します。このメソッドは、読み取りおよび書き込みの両方の操作をキャッシュします。
dm-cache メソッドは、**cache** タイプの論理ボリュームを作成します。

dm-writecache

このメソッドは、書き込み操作のみをキャッシュします。高速なボリュームは書き込み操作を保存し、それらをバックグラウンドで低速なディスクに移行します。高速ボリュームは通常 SSD または永続メモリー (PMEM) ディスクです。
dm-writecache メソッドは、**writecache** タイプの論理ボリュームを作成します。

関連情報

- **lvmcache(7)** man ページ

68.10.2. LVM キャッシュコンポーネント

LVM は、キャッシュを LVM 論理ボリュームに追加するためのサポートを提供します。LVM キャッシュは、LVM 論理ボリュームタイプを使用します。

Main LV
: より大きく、より遅い、元のボリューム。

キャッシュプール LV
: メイン LV からデータをキャッシュするために使用できる複合 LV。キャッシュデータを保持するためのデータと、キャッシュデータを管理するためのメタデータの 2 つのサブ LV があります。データおよびメタデータ用に特定のディスクを設定できます。キャッシュプールは **dm-cache** でのみ使用できます。

Cachevol LV
: メイン LV からデータをキャッシュするために使用できる線形 LV。データとメタデータ用に個別のディスクを設定することはできません。**cachevol** は、**dm-cache** または **dm-writecache** でのみ使用できます。

これらの関連付けられた LV はすべて、同じボリュームグループにある必要があります。

メインの論理ボリューム (LV) を、キャッシュされたデータを保持する高速で通常は小さい LV と組み合わせることができます。高速 LV は、SSD ドライブなどの高速ブロックデバイスから作成されます。論理ボリュームのキャッシュを有効にすると、LVM は元のボリュームの名前を変更および非表示にし、元の論理ボリュームで設定される新しい論理ボリュームを表示します。新しい論理ボリュームの設定は、キャッシュ方法と、**cachevol** オプションまたは **cachepool** オプションを使用しているかどうかによって異なります。

cachevol オプションおよび **cachepool** オプションは、キャッシングコンポーネントの配置に対するさまざまなレベルの制御を公開します。

- **cachevol** オプションを使用すると、高速なデバイスは、データブロックのキャッシュされたコピーとキャッシュ管理用のメタデータの両方を保存します。
- **cachepool** オプションを使用すると、別のデバイスはデータブロックのキャッシュコピーとキャッシュ管理用のメタデータを保存できます。
  dm-writecache メソッドは、**cachepool** と互換性がありません。

すべての設定において、LVM は、結果として作成される 1 つのデバイスを公開し、すべてのキャッシングコンポーネントをグループ化します。作成されるデバイスは、元の低速な論理ボリュームと同じ名前になります。

関連情報

- **lvmcache(7)** man ページ
- シンプロビジョニングされたボリューム (シンボリューム) の作成および管理

68.10.3. 論理ボリュームの dm-cache キャッシュの有効化

この手順では、**dm-cache** メソッドを使用して、論理ボリュームで一般的に使用されるデータのキャッシュを有効にします。

前提条件

- システムに、**dm-cache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

手順

1. 高速デバイスに **cachevol** ボリュームを作成します。

```
# lvcreate --size cachevol-size --name <fastvol> <vg> </dev/fast-pv>
```

以下の値を置き換えます。

cachevol-size

5G などの **cachevol** ボリュームのサイズ

fastvol

cachevol ボリュームの名前

vg

ボリュームグループ名

/dev/fast-pv

高速ブロックデバイスへのパス (例: **/dev/sdf**)

例68.7 cachevol ボリュームの作成

```
# lvcreate --size 5G --name fastvol vg /dev/sdf
Logical volume "fastvol" created.
```

2. **cachevol** ボリュームをメインの論理ボリュームに接続して、キャッシュを開始します。

```
# lvconvert --type cache --cachevol <fastvol> <vg/main-lv>
```

以下の値を置き換えます。

fastvol

cachevol ボリュームの名前

vg

ボリュームグループ名

main-lv

低速な論理ボリュームの名前

例68.8 メイン LV へのcachevol ボリュームの接続

```
# lvconvert --type cache --cachevol fastvol vg/main-lv
Erase all existing data on vg/fastvol? [y/n]: y
Logical volume vg/main-lv is now cached.
```

検証手順

- 新しく作成した論理ボリュームで **dm-cache** が有効になっているかどうかを確認します。

```
# lvs --all --options +devices <vg>

LV              Pool          Type   Devices
```

```
main-lv          [fastvol_cvol] cache  main-lv_corig(0)
[fastvol_cvol]                 linear /dev/fast-pv
[main-lv_corig]                 linear /dev/slow-pv
```

関連情報

- **lvmcache(7)** man ページ

68.10.4. 論理ボリュームに cachepool を使用した dm-cache キャッシュの有効化

この手順では、キャッシュデータとキャッシュメタデータ論理ボリュームを個別に作成し、ボリュームをキャッシュプールに統合することができます。

前提条件

- システムに、**dm-cache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

手順

1. 高速デバイスに **cachepool** ボリュームを作成します。

```
# lvcreate --type cache-pool --size <cachepool-size> --name <fastpool> <vg /dev/fast>
```

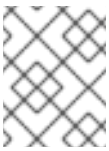
以下の値を置き換えます。

cachepool-size
: **cachepool** のサイズ (例: **5G**)

fastpool
: **cachepool** ボリュームの名前

vg
: ボリュームグループ名

/dev/fast
: 高速ブロックデバイスへのパス (例: **/dev/sdf1**)



注記

--poolmetadata オプションを使用して、cache-pool の作成時にプールメタデータの場所を指定できます。

例68.9 cachevol ボリュームの作成

```
# lvcreate --type cache-pool --size 5G --name fastpool vg /dev/sde
Logical volume "fastpool" created.
```

2. キャッシュを開始するために、メイン論理ボリュームに **cachepool** をアタッチします。

```
# lvconvert --type cache --cachepool <fastpool> <vg/main>
```

以下の値を置き換えます。

fastpool

cachepool ボリュームの名前

vg

ボリュームグループ名

main

低速な論理ボリュームの名前

例68.10 メイン LV への cachepool の接続

```
# lvconvert --type cache --cachepool fastpool vg/main
Do you want wipe existing metadata of cache pool vg/fastpool? [y/n]: y
Logical volume vg/main is now cached.
```

検証手順

- **cache-pool** タイプで新しく作成したデバイスボリュームを調べます。

```
# lvs --all --options +devices <vg>

LV                  Pool              Type        Devices
[fastpool_cpool]                        cache-pool  fastpool_pool_cdata(0)
[fastpool_cpool_cdata]                      linear      /dev/sdf1(4)
[fastpool_cpool_cmeta]                      linear      /dev/sdf1(2)
[lvol0_pmspare]                         linear      /dev/sdf1(0)
main                [fastpoool_cpool]  cache       main_corig(0)
[main_corig]                            linear      /dev/sdf1(O)
```

関連情報

- **lvcreate(8)** の man ページ
- **lvmcache(7)** man ページ
- **lvconvert(8)** の man ページ

68.10.5. 論理ボリュームの dm-writecache キャッシュの有効化

この手順では、**dm-writecache** メソッドを使用して、論理ボリュームへの書き込み I/O 操作のキャッシュを有効にします。

前提条件

- システムに、**dm-writecache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

- 低速な論理ボリュームがアクティブな場合は、非アクティブ化する。

手順

1. 低速な論理ボリュームがアクティブな場合は、非アクティブにします。

```
# lvchange --activate n <vg>/<main-lv>
```

以下の値を置き換えます。

vg
ボリュームグループ名

main-lv
低速な論理ボリュームの名前

2. 高速なデバイス上に非アクティブな **cachevol** ボリュームを作成します。

```
# lvcreate --activate n --size <cachevol-size> --name <fastvol> <vg> </dev/fast-pv>
```

以下の値を置き換えます。

cachevol-size
5G などの **cachevol** ボリュームのサイズ

fastvol
cachevol ボリュームの名前

vg
ボリュームグループ名

/dev/fast-pv
高速ブロックデバイスへのパス (例: **/dev/sdf**)

例68.11 非アクティブ化された cachevol ボリュームの作成

```
# lvcreate --activate n --size 5G --name fastvol vg /dev/sdf
WARNING: Logical volume vg/fastvol not zeroed.
Logical volume "fastvol" created.
```

3. **cachevol** ボリュームをメインの論理ボリュームに接続して、キャッシュを開始します。

```
# lvconvert --type writecache --cachevol <fastvol> <vg/main-lv>
```

以下の値を置き換えます。

fastvol
cachevol ボリュームの名前

vg
ボリュームグループ名

main-lv
低速な論理ボリュームの名前

例68.12 メイン LV への cachevol ボリュームの接続

```
# lvconvert --type writecache --cachevol fastvol vg/main-lv
Erase all existing data on vg/fastvol? [y/n]?: y
Using writecache block size 4096 for unknown file system block size, logical block
size 512, physical block size 512.
WARNING: unable to detect a file system block size on vg/main-lv
WARNING: using a writecache block size larger than the file system block size may
corrupt the file system.
Use writecache block size 4096? [y/n]: y
Logical volume vg/main-lv now has writecache.
```

4. 作成された論理ボリュームをアクティベートします。

```
# lvchange --activate y <vg/main-lv>
```

以下の値を置き換えます。

vg
ボリュームグループ名

main-lv
低速な論理ボリュームの名前

検証手順

- 新たに作成されたデバイスを確認します。

```
# lvs --all --options +devices vg

LV              VG Attr       LSize   Pool          Origin          Data%  Meta%  Move Log
Cpy%Sync Convert Devices
 main-lv         vg Cwi-a-C--- 500.00m [fastvol_cvol] [main-lv_wcorig] 0.00
main-lv_wcorig(0)
 [fastvol_cvol]   vg Cwi-aoC--- 252.00m
/dev/sdc1(0)
 [main-lv_wcorig] vg owi-aoC--- 500.00m
/dev/sdb1(0)
```

関連情報

- **lvmcache(7)** man ページ

68.10.6. 論理ボリュームのキャッシュの無効化

この手順では、論理ボリュームで現在有効な **dm-cache** キャッシュまたは **dm-writecache** キャッシュを無効にします。

前提条件

- キャッシュは、論理ボリュームで有効になります。

手順

1. 論理ボリュームを非アクティブにします。

```
# lvchange --activate n <vg>/<main-lv>
```

vg はボリュームグループ名に置き換え、**main-lv** はキャッシュが有効になっている論理ボリュームの名前に置き換えます。

2. **cachevol** ボリュームまたは **cachepool** ボリュームの割り当てを解除します。

```
# lvconvert --splitcache <vg>/<main-lv>
```

以下の値を置き換えます。

vg はボリュームグループ名に置き換え、**main-lv** はキャッシュが有効になっている論理ボリュームの名前に置き換えます。

例68.13 cachevol または cachepool ボリュームの接続解除

```
# lvconvert --splitcache vg/main-lv
Detaching writecache already clean.
Logical volume vg/main-lv writecache has been detached.
```

検証手順

- 論理ボリュームが接続されていないことを確認します。

```
# lvs --all --options +devices <vg>

LV      Attr       Type   Devices
fastvol -wi------- linear /dev/fast-pv
main-lv -wi------- linear /dev/slow-pv
```

関連情報

- **lvmcache(7)** man ページ

68.11. 論理ボリュームのアクティブ化

アクティブ状態の論理ボリュームは、ブロックデバイスを介して使用できます。アクティブになっている論理ボリュームにはアクセスにでき、変更できます。論理ボリュームを作成すると、デフォルトでアクティブになります。

個々の論理ボリュームを非アクティブにしてカーネルが認識しないようにする必要がある状況はさまざまです。個々の論理ボリュームは、**lvchange** コマンドの **-a** オプションを使用してアクティブまたは非アクティブにできます。

個々の論理ボリュームを非アクティブにするには、以下のコマンドを実行します。

```
lvchange -an vg/lv
```

個々の論理ボリュームをアクティブにするには、以下のコマンドを実行します。

```
lvchange -ay vg/lv
```

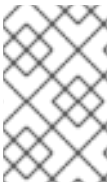
vgchange コマンドの **-a** オプションを使用して、ボリュームグループの論理ボリュームをすべてアクティブまたは非アクティブにできます。これは、ボリュームグループの個々の論理ボリュームに **lvchange -a** コマンドを実行するのと同じです。

ボリュームグループの論理ボリュームをすべて非アクティブにするには、以下のコマンドを実行します。

```
vgchange -an vg
```

ボリュームグループの論理ボリュームをすべてアクティブにするには、以下のコマンドを実行します。

```
vgchange -ay vg
```



注記

systemd-mount ユニットがマスクされていない限り、手動アクティベーション中に、**systemd** は **/etc/fstab** ファイルからの対応するマウントポイントで LVM ボリュームを自動的にマウントします。

68.11.1. 論理ボリュームの自動アクティブ化の制御

論理ボリュームの自動アクティブ化は、システム起動時に論理ボリュームをイベントベースで自動的にアクティブにすることを指します。システムでデバイスが利用可能になると (デバイスのオンラインイベント)、**systemd/udev** は、各デバイスに **lvm2-pvscan** サービスを実行します。このサービスは、named デバイスを読み込む **pvscan --cache -aay device** コマンドを実行します。デバイスがボリュームグループに属している場合、**pvscan** コマンドは、そのボリュームグループに対する物理ボリュームがすべて、そのシステムに存在するかどうかを確認します。存在する場合は、このコマンドが、そのボリュームグループにある論理ボリュームをアクティブにします。

/etc/lvm/lvm.conf 設定ファイルで以下の設定オプションを使用して、論理ボリュームの自動アクティブ化を制御できます。

- **global/event_activation**
  event_activation が無効になっている場合、**systemd/udev** は、システムの起動時に存在する物理ボリュームでのみ、論理ボリュームを自動アクティブにします。すべての物理ボリュームが表示されていないと、一部の論理ボリュームが自動的にアクティブにならない場合もあります。

- **activation/auto_activation_volume_list**
  auto_activation_volume_list を空のリストに設定すると、自動アクティベーションは完全に無効になります。特定の論理ボリュームとボリュームグループに **auto_activation_volume_list** を設定すると、自動アクティベーションは、設定した論理ボリュームに制限されます。

このオプションの設定は、**/etc/lvm/lvm.conf** 設定ファイルを参照してください。

68.11.2. 論理ボリュームのアクティブ化の制御

以下の方法で、論理ボリュームのアクティブ化を制御できます。

- **/etc/lvm/conf** ファイルの **activation/volume_list** 設定で行います。これにより、どの論理ボリュームをアクティブにするかを指定できます。このオプションの使用方法は **/etc/lvm/lvm.conf** 設定ファイルを参照してください。

- 論理ボリュームのアクティブ化スキップフラグで行います。このフラグが論理ボリュームに設定されていると、通常のアクティベーションコマンド時にそのボリュームがスキップされます。

以下の方法で、論理ボリュームのアクティブ化スキップフラグを設定できます。

- **lvcreate** コマンドの **-kn** オプションまたは **--setactivationskip n** オプションを指定すると、論理ボリュームの作成時にアクティブ化スキップフラグをオフにできます。

- **lvchange** コマンドの **-kn** オプションまたは **--setactivationskip n** オプションを指定すると、既存の論理ボリュームのアクティブ化スキップフラグをオフにできます。

- **lvchange** コマンドの **-ky** オプションまたは **--setactivationskip y** オプションを指定すると、オフになっているボリュームに対して、アクティベーションスキップフラグを再度オンにできます。

このアクティブ化スキップフラグが論理ボリュームに設定されているかを確認するには、**lvs** コマンドを実行します。以下のような **k** 属性が表示されます。

```
# lvs vg/thin1s1
LV       VG  Attr       LSize Pool  Origin
thin1s1  vg  Vwi---tz-k 1.00t pool0 thin1
```

標準オプション **-ay** または **--activate y** の他に、**-K** オプションまたは **--ignoreactivationskip** オプションを使用して、**k** 属性セットで論理ボリュームをアクティブにできます。

デフォルトでは、シンプロビジョニングのスナップショットボリュームに、作成時にアクティブ化スキップのフラグが付いています。**/etc/lvm/lvm.conf** ファイルの **auto_set_activation_skip** 設定で、新たに作成した、シンプロビジョニングのスナップショットボリュームの、デフォルトのアクティブ化スキップ設定を制御できます。

以下のコマンドは、アクティブ化スキップフラグが設定されているシンプロビジョニングされたスナップショット論理ボリュームをアクティベートします。

```
# lvchange -ay -K VG/SnapLV
```

以下のコマンドは、アクティブ化スキップフラグがない、シンプロビジョニングされたスナップショットを作成します。

```
# lvcreate --type thin -n SnapLV -kn -s ThinLV --thinpool VG/ThinPoolLV
```

以下のコマンドは、スナップショット論理ボリュームから、アクティブ化スキップフラグを削除します。

```
# lvchange -kn VG/SnapLV
```

68.11.3. 共有論理ボリュームのアクティベーション

以下のように、**lvchange** コマンドおよび **vgchange** コマンドの **-a** オプションを使用して、共有論理ボリュームの論理ボリュームのアクティブ化を制御できます。

コマンド	アクティベーション
lvchange -ay|e	共有論理ボリュームを排他モードでアクティベートし、1台のホストのみが論理ボリュームをアクティベートできるようにします。アクティベーションが失敗 (論理ボリュームが別のホストでアクティブの場合に発生する可能性あり) すると、エラーが報告されます。
lvchange -asy	共有モードで共有論理ボリュームをアクティブにし、複数のホストが論理ボリュームを同時にアクティブにできるようにします。アクティブできない場合は (その論理ボリュームは別のホストで排他的にアクティブになっている場合に発生する可能性あり)、エラーが報告されます。スナップショットなど、論理タイプが共有アクセスを禁止している場合は、コマンドがエラーを報告して失敗します。複数のホストから同時に使用できない論理ボリュームタイプには、シン、キャッシュ、RAID、およびスナップショットがあります。
lvchange -an	論理ボリュームを非アクティブにします。

68.11.4. 欠落しているデバイスを含む論理ボリュームのアクティブ化

lvchange コマンドで、**activation_mode** パラメーターを以下のいずれかの値に設定することで、デバイスが見つからない論理ボリュームをアクティブにするように設定できます。

アクティベーションモード	意味
complete	物理ボリュームが見つからない論理ボリュームのみをアクティベートすることを許可します。これが最も制限的なモードです。
degraded	物理ボリュームが見つからない RAID 論理ボリュームをアクティベートすることを許可します。
partial	物理ボリュームが見つからない論理ボリュームをすべてアクティベートすることを許可します。このオプションは、回復または修復にのみ使用してください。

activation_mode のデフォルト値は、**/etc/lvm/lvm.conf** ファイルの **activation_mode** 設定により決まります。詳細情報は、man ページの **lvmraid**(7) を参照してください。

68.12. LVM デバイスの可視性および使用を制限する

論理ボリュームマネージャー (LVM) がスキャンできるデバイスを制御することにより、LVM で表示および使用できるデバイスを制限できます。

LVM デバイススキャンの設定を調整するには、**/etc/lvm/lvm.conf** ファイルで LVM デバイスフィルター設定を編集します。**lvm.conf** ファイル内のフィルターは、一連の単純な正規表現で設定されています。システムは、これらの式を **/dev** ディレクトリー内の各デバイス名に適用して、検出された各ブロックデバイスを受け入れるか拒否するかを決定します。

68.12.1. LVM デバイスフィルター

論理ボリュームマネージャー (LVM) デバイスフィルターは、デバイス名パターンのリストです。

パターンは、任意の文字で区切られた正規表現と、**a** (許可) または **r** (拒否) が先頭に付けられた正規表現です。デバイスに一致する最初の正規表現は、LVM が特定のデバイスを許可するか、拒否 (無視) するかを判断します。デバイスは、シンボリックリンクを介して複数の名前を持つことができます。フィルターがこれらのデバイス名のいずれかを受け入れる場合、LVM はそのデバイスを使用します。LVM は、パターンに一致しないデバイスも受け付けます。

デフォルトのデバイスフィルターは、システム上のすべてのデバイスを受け入れます。理想的なユーザー設定のデバイスフィルターは、1 つ以上のパターンを受け入れ、それ以外はすべて拒否します。たとえば、このような場合、パターンリストは **r|.*|** で終わることができます。.

lvm.conf ファイルの **devices/filter** フィールドと **devices/global_filter** フィールドで、LVM デバイスのフィルター設定を見つけることができます。

68.12.1.1. 関連情報

- **lvm.conf(5)** man ページ

68.12.1.2. LVM デバイスフィルター設定の例

以下のリストは、LVM がスキャンして後で使用できるデバイスを制御するフィルター設定を示しています。**lvm.conf** ファイルでデバイスフィルターを設定します。

- 以下は、すべてのデバイスをスキャンするデフォルトのフィルター設定です。

```
filter = [ "|a.*|" ]
```

- 以下のフィルターは、ドライブにメディアが入っていない場合の遅延を回避するために **cdrom** デバイスを削除します。

```
filter = [ "r|^/dev/cdrom$|" ]
```

- 以下のフィルターはすべてのループデバイスを追加して、その他のすべてのブロックデバイスを削除します。

```
filter = [ "a|loop|", "r|.*|" ]
```

- 次のフィルターは、すべてのループおよび統合開発環境 (IDE) デバイスを追加し、他のすべてのブロックデバイスを削除します。

```
filter = [ "a|loop|", "a|/dev/hd.*|", "r|.*|" ]
```

- 以下のフィルターは 1 番目の IDE ドライブにパーティション 8 のみを追加して、他のすべてのブロックデバイスを削除します。

```
filter = [ "a|^/dev/hda8$|", "r|.*|" ]
```

関連情報

- **lvm.conf(5)** man ページ

68.13. LVM の割り当ての制御

デフォルトでは、ボリュームグループは、同じ物理ボリューム上に並行ストライプを配置しないなど、常識的な規則に従って物理エクステントを割り当てます。これが **normal** の割り当てポリシーです。**vgcreate** コマンドで **--alloc** 引数を使用して、**contiguous**、**anywhere**、または **cling** の割り当てポリシーを指定できます。一般的に、**normal** 以外の割り当てポリシーが必要となるのは、通常とは異なる、標準外のエクステント割り当てを必要とする特別なケースのみです。

68.13.1. LVM の割り当てポリシー

LVM の操作で物理エクステントを 1 つまたは複数の論理ボリュームに割り当てる必要がある場合、割り当ては以下のように行われます。

- ボリュームグループで割り当てられていない物理エクステントのセットが、割り当てのために生成されます。コマンドラインの末尾に物理エクステントの範囲を指定すると、指定した物理ボリュームの中で、その範囲内で割り当てられていない物理エクステントだけが、割り当て用エクステントとして考慮されます。

- 割り当てポリシーは順番に試行されます。最も厳格なポリシー (**contiguous**) から始まり、最後は **--alloc** オプションで指定した割り当てポリシーか、特定の論理ボリュームやボリュームグループにデフォルトとして設定されている割り当てポリシーが試行されます。割り当てポリシーでは、埋める必要がある空の論理ボリューム領域の最小番号の論理エクステントから始まり、割り当てポリシーによる制限に従って、できるだけ多くの領域の割り当てを行います。領域が足りなくなると、LVM は次のポリシーに移動します。

割り当てポリシーの制限は以下のとおりです。

- **contiguous** の割り当てポリシーでは、論理ボリュームの最初の論理エクステントを除いたすべての論理エクステントは、その直前の論理エクステントに物理的に隣接している必要があります。
  論理ボリュームがストライプ化またはミラー化されている場合は、**contiguous** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レッグ) に個別に適用されます。

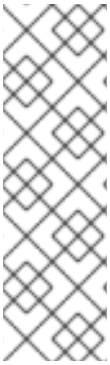
- **cling** の割り当てポリシーでは、既存の論理ボリュームに追加する論理エクステントに使用される物理ボリュームが、その論理ボリュームにある別の (1 つ以上の) 論理エクステントですでに使用されている必要があります。設定パラメーター **allocation/cling_tag_list** が定義されており、リスト表示されているいずれかのタグが 2 つの物理ボリュームにある場合、この 2 つの物理ボリュームは一致すると見なされます。これにより、割り当てのために、同様のプロパティー (物理的な場所など) が設定されている物理ボリュームのグループにタグを付け、その物理ボリュームを同等なものとして処理できます。
  論理ボリュームがストライプ化またはミラー化されると、**cling** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レッグ) に個別に適用されます。

- **normal** の割り当てポリシーは、並列の論理ボリューム (異なるストライプまたはミラーイメージ/レッグ) 内の同じオフセットで、その並列の論理ボリュームにすでに割り当てられている論理エクステントと同じ物理ボリュームを共有する物理エクステントは選択しません。
  ミラーデータを保持するために、論理ボリュームと同時にミラーログを割り当てる場合、**normal** の割り当てポリシーでは、最初にログとデータに対して、それぞれ別の物理ボリュームを選択しようとします。異なる物理ボリュームを選択できず、かつ **allocation/mirror_logs_require_separate_pvs** 設定パラメーターが 0 に設定されている場合は、データの一部とログが物理ボリュームを共有できるようになります。

  また、シンプールメタデータを割り当てる場合も、**normal** の割り当てポリシーはミラーログを割り当てる場合と同じようになりますが、設定パラメーターは **allocation/thin_pool_metadata_require_separate_pvs** の値が適用されます。

- 割り当て要求を満たすのに十分な空きエクステントがあっても、**normal** の割り当てポリシーによって使用されない場合は、たとえ同じ物理ボリュームに 2 つのストライプを配置することによってパフォーマンスが低下しても、**anywhere** 割り当てポリシーがその空きエクステントを使用します。

割り当てポリシーは、**vgchange** コマンドで変更できます。



注記

今後の更新で、定義された割り当てポリシーに基づくレイアウト操作のコードが変更される可能性があることに注意してください。たとえば、割り当て可能な空き物理エクステントの数が同じ 2 つの空の物理ボリュームをコマンドラインで指定する場合、現行バージョンの LVM では、それが表示されている順番に使用が検討されます。ただし、今後のリリースで、そのプロパティーが変更されない保証はありません。特定の論理ボリューム用に特定のレイアウトを取得することが重要な場合は、各手順に適用される割り当てポリシーに基づいて LVM がレイアウトを決定することがないように、**lvcreate** と **lvconvert** を順に使用してレイアウトを構築してください。

特定のケースで、割り当てプロセスがどのように行われているかを確認するには、コマンドに **-vvvv** オプションを追加するなどして、デバッグロギングの出力を表示します。

68.13.2. 物理ボリュームでの割り当て防止

pvchange コマンドを使用すると、1 つまたは複数の物理ボリュームの空き領域で物理エクステントが割り当てられないように設定できます。これは、ディスクエラーが発生した場合や、物理ボリュームを取り除く場合に必要となる可能性があります。

以下のコマンドは、**/dev/sdk1** での物理エクステントの割り当てを無効にします。

```
# pvchange -x n /dev/sdk1
```

pvchange コマンドで **-xy** 引数を使用すると、無効にされていた割り当てを許可できます。

68.13.3. cling 割り当てポリシーを使用した論理ボリュームの拡張

LVM ボリュームを拡張する際には、**lvextend** コマンドの **--alloc cling** オプションを使用して、**cling** 割り当てポリシーを指定できます。このポリシーにより、既存の論理ボリュームの最終セグメントと同じ物理ボリュームの領域が選択されます。物理ボリューム上に十分な領域がなく、タグのリストが **/etc/lvm/lvm.conf** ファイルで定義されている場合には、LVM が、その物理ボリュームにいずれかのタグが付けられているかを確認し、既存エクステントと新規エクステント間で、物理ボリュームのタグを適合させようとします。

たとえば、使用している論理ボリュームが、1 つのボリュームグループ内の 2 サイト間でミラー化されている場合は、**@site1** タグと **@site2** タグを使用し、サイトの場所に応じて物理ボリュームにタグを付けることができます。この場合は、**lvm.conf** ファイル内に以下の行を指定します。

```
cling_tag_list = [ "@site1", "@site2" ]
```

以下の例では、**lvm.conf** ファイルが変更されて、次のような行が追加されています。

```
cling_tag_list = [ "@A", "@B" ]
```

また、この例では、**/dev/sdb1**、**/dev/sdc1**、**/dev/sdd1**、**/dev/sde1**、**/dev/sdf1**、**/dev/sdg1**、および

/dev/sdh1 の物理ボリュームで設定されるボリュームグループ **taft** が作成されています。この物理ボリュームには、**A**、**B**、および **C** のタグが付けられています。この例では、**C** のタグは使用されていませんが、LVM がタグを使用して、ミラーレッグに使用する物理ボリュームを選択することを示しています。

```
# pvs -a -o +pv_tags /dev/sd[bcdefgh]
  PV         VG   Fmt  Attr PSize  PFree  PV Tags
  /dev/sdb1  taft lvm2 a--  15.00g 15.00g A
  /dev/sdc1  taft lvm2 a--  15.00g 15.00g B
  /dev/sdd1  taft lvm2 a--  15.00g 15.00g B
  /dev/sde1  taft lvm2 a--  15.00g 15.00g C
  /dev/sdf1  taft lvm2 a--  15.00g 15.00g C
  /dev/sdg1  taft lvm2 a--  15.00g 15.00g A
  /dev/sdh1  taft lvm2 a--  15.00g 15.00g A
```

以下のコマンドは、ボリュームグループ **taft** から 10 ギガバイトのミラー化ボリュームを作成します。

```
# lvcreate --type raid1 -m 1 -n mirror --nosync -L 10G taft
  WARNING: New raid1 won't be synchronised. Don't read what you didn't write!
  Logical volume "mirror" created
```

以下のコマンドは、ミラーレッグおよび RAID メタデータのサブボリュームに使用されるデバイスを表示します。

```
# lvs -a -o +devices
  LV              VG   Attr       LSize  Log Cpy%Sync Devices
  mirror          taft Rwi-a-r--- 10.00g       100.00 mirror_rimage_0(0),mirror_rimage_1(0)
  [mirror_rimage_0] taft iwi-aor--- 10.00g             /dev/sdb1(1)
  [mirror_rimage_1] taft iwi-aor--- 10.00g             /dev/sdc1(1)
  [mirror_rmeta_0]  taft ewi-aor---  4.00m             /dev/sdb1(0)
  [mirror_rmeta_1]  taft ewi-aor---  4.00m             /dev/sdc1(0)
```

以下のコマンドは、ミラー化ボリュームのサイズを拡張します。**cling** 割り当てポリシーで、同じタグが付いた物理ボリュームを使用して、ミラーレッグが拡張される必要があることを示します。

```
# lvextend --alloc cling -L +10G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 20.00 GiB
  Logical volume mirror successfully resized
```

以下に表示したコマンドは、レッグとして同一のタグが付いた物理ボリュームを使用してミラーレッグが拡張されているのを示しています。**C** のタグが付いた物理ボリュームは無視される点に注意してください。

```
# lvs -a -o +devices
  LV              VG   Attr       LSize  Log Cpy%Sync Devices
  mirror          taft Rwi-a-r--- 20.00g       100.00 mirror_rimage_0(0),mirror_rimage_1(0)
  [mirror_rimage_0] taft iwi-aor--- 20.00g             /dev/sdb1(1)
  [mirror_rimage_0] taft iwi-aor--- 20.00g             /dev/sdg1(0)
  [mirror_rimage_1] taft iwi-aor--- 20.00g             /dev/sdc1(1)
  [mirror_rimage_1] taft iwi-aor--- 20.00g             /dev/sdd1(0)
  [mirror_rmeta_0]  taft ewi-aor---  4.00m             /dev/sdb1(0)
  [mirror_rmeta_1]  taft ewi-aor---  4.00m             /dev/sdc1(0)
```

68.13.4. タグを使用した LVM RAID オブジェクト間の区別

LVM RAID オブジェクトにタグを割り当て、グループごとにアクティブ化などの LVM RAID の動作の制御を自動化できます。

lvm の割り当ては、割り当てポリシーに基づいて PV レベルで発生するため、物理ボリューム (PV) タグは、論理ボリューム (LV) またはボリュームグループ (VG) タグではなく、LVM RAID の割り当て制御を行います。ストレージタイプを異なるプロパティーで区別するには、適切にタグを付けます (NVMe、SSD、HDD など)。Red Hat は、VG に追加した後に新規 PV を適切にタグ付けすることを推奨します。

この手順では、**/dev/sda** が SSD で、 **/dev/sd[b-f]** が 1 つのパーティションを持つ HDD であることを前提とします。

前提条件

- **lvm2** パッケージがインストールされている。
- PV として使用するストレージデバイスが利用できます。

手順

1. ボリュームグループを作成します。

```
# vgcreate MyVG /dev/sd[a-f]1
```

2. 物理ボリュームにタグを追加します。

```
# pvchange --addtag ssds /dev/sda1

# pvchange --addtag hdds /dev/sd[b-f]1
```

3. RAID6 論理ボリュームを作成します。

```
# lvcreate --type raid6 --stripes 3 -L1G -nr6 MyVG @hdds
```

4. リニアキャッシュプールボリュームを作成します。

```
# lvcreate -nr6pool -L512m MyVG @ssds
```

5. RAID6 ボリュームをキャッシュに変換します。

```
# lvconvert --type cache --cachevol MyVG/r6pool MyVG/r6
```

関連情報

- man ページの **lvcreate(8)**、**lvconvert(8)**、**lvmraid(7)**、および **lvmcache(7)**

68.14. LVM のトラブルシューティング

LVM ツールを使用して、LVM ボリュームおよびグループのさまざまな問題のトラブルシューティングを行うことができます。

68.14.1. LVM での診断データの収集

LVM コマンドが想定どおりに機能しない場合は、以下の方法で診断情報を収集できます。

手順

- 以下の方法を使用して、さまざまな診断データを収集します。
  - **-v** 引数を LVM コマンドに追加して、コマンドの出力の詳細レベルを増やします。**v** を追加すると、詳細度をさらに増やすことができます。**v** は最大 4 つ許可されます (例: **-vvvv**)。
  - **/etc/lvm/lvm.conf** 設定ファイルの **log** セクションで、**level** オプションの値を増やします。これにより、LVM がシステムログにより多くの情報を提供します。
  - 問題が論理ボリュームのアクティブ化に関連する場合は、アクティブ化中に LVM がログメッセージをログに記録できるようにします。
    i. **/etc/lvm/lvm.conf** 設定ファイルの **log** セクションで **activation = 1** オプションを設定します。
    ii. LVM コマンドに **-vvvv** オプションを付けて実行します。
    iii. コマンドの出力を確認します。
    iv. **activation** オプションを **0** にリセットします。
    オプションを **0** にリセットしないと、メモリー不足の状況でシステムが応答しなくなる可能性があります。
  - 診断目的で情報ダンプを表示します。

```
# lvmdump
```

  - 追加のシステム情報を表示します。

```
# lvs -v
```

```
# pvs --all
```

```
# dmsetup info --columns
```

  - **/etc/lvm/backup/** ディレクトリーの最後の LVM メタデータのバックアップと、**/etc/lvm/archive/** ディレクトリー内のアーカイブバージョンを確認します。
  - 現在の設定情報を確認します。

```
# lvmconfig
```

  - **/run/lvm/hints** キャッシュファイルで、物理ボリュームを持つデバイスを記録します。

関連情報

- **lvmdump(8)** の man ページ

68.14.2. 障害の発生した LVM デバイスに関する情報の表示

ボリュームが失敗した理由を特定するのに役立つ、障害の発生した LVM ボリュームに関する情報を表示できます。

手順

- **vgs** ユーティリティーまたは **lvs** ユーティリティーを使用して、障害が発生したボリュームを表示します。

例68.14 障害が発生したボリュームグループ

この例では、ボリュームグループ **myvg** を設定するデバイスのいずれかが失敗しています。ボリュームグループは使用できませんが、障害が発生したデバイスに関する情報を表示できます。

```
# vgs --options +devices
 /dev/vdb1: open failed: No such device or address
 /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

VG    #PV #LV #SN Attr   VSize  VFree  Devices
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](0)
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/vdb1(0)
```

例68.15 障害が発生した論理ボリューム

この例では、ボリュームグループの論理ボリュームが失敗したためにデバイスのいずれかが失敗しています。コマンドの出力には、障害が発生した論理ボリュームが表示されます。

```
# lvs --all --options +devices

  /dev/vdb1: open failed: No such device or address
  /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

  LV    VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert Devices
  mylv myvg -wi-a---p- 20.00g                                                     [unknown](0)
[unknown](5120),/dev/sdc1(0)
```

例68.16 ミラー化論理ボリュームのレッグに障害が発生しました。

以下の例は、ミラー化論理ボリュームのレッグが失敗した場合の **vgs** ユーティリティーおよび **lvs** ユーティリティーからのコマンドの出力を示しています。

```
# vgs --all --options +devices

  VG    #PV #LV #SN Attr   VSize VFree Devices
  corey 4 4 0 rz-pnc 1.58T 1.34T my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T unknown device(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdb1(0)
```

```
# lvs --all --options +devices

  LV                 VG    Attr   LSize   Origin Snap%  Move Log          Copy%  Devices
  my_mirror corey mwi-a- 120.00G my_mirror_mlog 1.95
my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  [my_mirror_mimage_0] corey iwi-ao 120.00G unknown device(0)
  [my_mirror_mimage_1] corey iwi-ao 120.00G /dev/sdb1(0)
  [my_mirror_mlog] corey lwi-ao 4.00M /dev/sdd1(0)
```

68.14.3. ボリュームグループから見つからない LVM 物理ボリュームの削除

物理ボリュームに障害が発生した場合は、ボリュームグループ内の残りの物理ボリュームをアクティブにし、その物理ボリュームを使用していたすべての論理ボリュームをボリュームグループから削除できます。

手順

1. ボリュームグループ内の残りの物理ボリュームをアクティベートします。

```
# vgchange --activate y --partial myvg
```

2. 削除する論理ボリュームを確認します。

```
# vgreduce --removemissing --test myvg
```

3. ボリュームグループから、失われた物理ボリュームを使用していた論理ボリュームをすべて削除します。

```
# vgreduce --removemissing --force myvg
```

4. オプション:保持する論理ボリュームを誤って削除した場合には、**vgreduce** 操作を元に戻すことができます。

```
# vgcfgrestore myvg
```

警告

シンプールを削除すると、LVM は操作を元に戻すことができません。

68.14.4. 見つからない LVM 物理ボリュームのメタデータの検索

物理ボリュームのボリュームグループメタデータ領域が誤って上書きされたり、破棄されたりする場合は、メタデータ領域が正しくないことを示すエラーメッセージか、システムが特定の UUID を持つ物理ボリュームを見つけることができないことを示すエラーメッセージが表示されます。

この手順では、物理ボリュームが見つからないか、破損している、アーカイブされた最新のメタデータを見つけます。

手順

1. 物理ボリュームを含むボリュームグループのアーカイブされたメタデータファイルを検索します。アーカイブされたメタデータファイルは、**/etc/lvm/archive/volume-group-name_backup-number.vg** パスにあります。

   ```
   # cat /etc/lvm/archive/myvg_00000-1248998876.vg
   ```

   00000-1248998876 を backup-number に置き換えます。ボリュームグループの番号が最も高い、既知の有効なメタデータファイルの最後のものを選択します。

2. 物理ボリュームの UUID を検索します。以下の方法のいずれかを使用します。

   - 論理ボリュームをリスト表示します。

     ```
     # lvs --all --options +devices

       Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
     ```

   - アーカイブされたメタデータファイルを確認します。ボリュームグループ設定の **physical_volumes** セクションで、**id =** のラベルが付いた値として UUID を検索します。

   - **--partial** オプションを使用してボリュームグループを非アクティブにします。

     ```
     # vgchange --activate n --partial myvg

       PARTIAL MODE. Incomplete logical volumes will be processed.
       WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
       WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/vdb1).
       0 logical volume(s) in volume group "myvg" now active
     ```

68.14.5. LVM 物理ボリュームでのメタデータの復元

この手順では、破損したり、新しいデバイスに置き換えたりする物理ボリュームのメタデータを復元します。物理ボリュームのメタデータ領域を書き換えて、物理ボリュームからデータを復旧できる場合があります。

警告

作業用の LVM 論理ボリュームでこの手順を実行しないでください。誤った UUID を指定すると、データが失われることになります。

前提条件

- 見つからない物理ボリュームのメタデータを特定している。詳細は、見つからない LVM 物理ボリュームのメタデータの検索 を参照してください。

手順

1. 物理ボリュームでメタデータを復元します。

```
# pvcreate --uuid physical-volume-uuid \
        --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \
        block-device
```

注記

コマンドは、LVM メタデータ領域のみを上書きし、既存のデータ領域には影響を与えません。

例68.17 /dev/vdb1 での物理ボリュームの復元

以下の例では、以下のプロパティーで **/dev/vdb1** デバイスを物理ボリュームとしてラベル付けします。

- **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk** の UUID
- **VG_00050.vg** に含まれるメタデータ情報 (ボリュームグループの最新のアーカイブメタデータ)

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \
        --restorefile /etc/lvm/archive/VG_00050.vg \
        /dev/vdb1


 ...
 Physical volume "/dev/vdb1" successfully created
```

2. ボリュームグループのメタデータを復元します。

```
# vgcfgrestore myvg

  Restored volume group myvg
```

3. ボリュームグループの論理ボリュームを表示します。

```
# lvs --all --options +devices myvg
```

現在、論理ボリュームは非アクティブです。以下に例を示します。

```
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  mylv myvg   -wi--- 300.00G                             /dev/vdb1 (0),/dev/vdb1(0)
  mylv myvg   -wi--- 300.00G                             /dev/vdb1 (34728),/dev/vdb1(0)
```

4. 論理ボリュームのセグメントタイプが RAID の場合は、論理ボリュームを再同期します。

```
# lvchange --resync myvg/mylv
```

5. 論理ボリュームを非アクティブにします。

```
# lvchange --activate y myvg/mylv
```

6. ディスク上の LVM メタデータが、それを上書きしたものと同じかそれ以上のスペースを使用する場合は、この手順で物理ボリュームを回復できます。メタデータを上書きしたものがメタデータ領域を超えると、ボリューム上のデータが影響を受ける可能性があります。そのデータを復元するには、**fsck** コマンドを使用することができます。

検証手順

- アクティブな論理ボリュームを表示します。

```
# lvs --all --options +devices

  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  mylv myvg   -wi--- 300.00G                             /dev/vdb1 (0),/dev/vdb1(0)
  mylv myvg   -wi--- 300.00G                             /dev/vdb1 (34728),/dev/vdb1(0)
```

68.14.6. LVM 出力の丸めエラー

ボリュームグループの領域使用量を報告する LVM コマンドは、報告された数を **2** 進法に切り上げ、人間が判読できる出力を提供します。これには、**vgdisplay** ユーティリティーおよび **vgs** ユーティリティーが含まれます。

丸めの結果、報告された空き領域の値は、ボリュームグループが提供する物理エクステントよりも大きくなる可能性があります。報告された空き領域のサイズの論理ボリュームを作成しようとすると、以下のエラーが発生する可能性があります。

```
Insufficient free extents
```

エラーを回避するには、ボリュームグループの空き物理エクステントの数を調べる必要があります。これは、空き領域の正確な値です。次に、エクステントの数を使用して、論理ボリュームを正常に作成できます。

68.14.7. LVM ボリューム作成時の丸めエラーの防止

LVM 論理ボリュームを作成する場合は、丸めエラーを防ぐために論理ボリュームの論理エクステントの数を指定できます。

手順

1. ボリュームグループの空き物理エクステントの数を検索します。

```
# vgdisplay myvg
```

例68.18 ボリュームグループの空きエクステント

たとえば、以下のボリュームグループには 8780 個のの空き物理エクステントがあります。

```
--- Volume group ---
 VG Name               myvg
 System ID
 Format                lvm2
 Metadata Areas        4
 Metadata Sequence No  6
 VG Access             read/write
[...]
Free  PE / Size       8780 / 34.30 GB
```

2. 論理ボリュームを作成します。ボリュームサイズをバイトではなくエクステントに入力します。

例68.19 エクステントの数を指定して論理ボリュームを作成

```
# lvcreate --extents 8780 --name mylv myvg
```

例68.20 残りの領域をすべて使用する論理ボリュームの作成

または、論理ボリュームを拡張して、ボリュームグループ内の残りの空き領域の割合を使用できます。以下に例を示します。

```
# lvcreate --extents 100%FREE --name mylv myvg
```

検証手順

- ボリュームグループが使用するエクステントの数を確認します。

```
# vgs --options +vg_free_count,vg_extent_count

  VG     #PV #LV #SN  Attr   VSize   VFree  Free  #Ext
  myvg   2   1   0   wz--n- 34.30G    0    0     8780
```

68.14.8. LVM RAID のトラブルシューティング

LVM RAID デバイスのさまざまな問題のトラブルシューティングを実行して、データエラーの修正、デバイスの復旧、障害が発生したデバイスの置き換えを行うことができます。

68.14.8.1. RAID 論理ボリュームでのデータ整合性の確認 (RAID スクラビング)

LVM は、RAID 論理ボリュームのスクラビングに対応します。RAID スクラビングは、アレイ内のデータおよびパリティーブロックをすべて読み込み、それが一貫しているかどうかを確認するプロセスです。

手順

1. オプション:スクラビングプロセスが使用する I/O 帯域幅を制限します。
RAID スクラビング操作を実行する際に、**sync** 操作で必要になるバックグラウンド I/O は、その他の I/O (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。リカバリースロットルを実装してスクラビング操作のレートを制御できます。

   次の手順で、**lvchange --syncaction** コマンドに以下のオプションを追加します。

   --maxrecoveryrate Rate[bBsSkKmMgG]

   操作が通常の I/O 操作に押し出すように、最大復旧速度を設定します。復旧速度を 0 に設定すると、操作がバインド解除されることを意味します。

   --minrecoveryrate Rate[bBsSkKmMgG]

   最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。

   Rate 値は、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しないと、オプションはデバイスごとの 1 秒あたらりの kiB を想定します。

2. アレイ内の不一致数を修復せずに、アレイ内の不一致の数を表示します。

```
# lvchange --syncaction check vg/raid_lv
```

3. アレイ内の不一致を修正します。

```
# lvchange --syncaction repair vg/raid_lv
```

注記

lvchange --syncaction repair 操作は、**lvconvert --repair** 操作と同じ機能を実行しません。

- **lvchange --syncaction repair** 操作は、アレイでバックグラウンドの同期操作を開始します。
- **lvconvert --repair** 操作は、ミラーまたは RAID 論理ボリュームの障害が発生したデバイスを修復するか、置き換えます。

4. オプション:スクラビング操作に関する情報を表示します。

```
# lvs -o +raid_sync_action,raid_mismatch_count vg/lv
```

- **raid_sync_action** フィールドは、RAID ボリュームが現在実行している同期操作を表示します。これには、以下のいずれかの値を使用できます。

  idle

  すべての同期操作が完了している (何も実行していません)。

  resync

アレイを初期化、またはマシン障害後の復旧を実行する。

recover

アレイ内のデバイスを置き換える。

check

アレイの不一致を検索する。

repair

不一致を検索し、修復する。

- **raid_mismatch_count** フィールドは、**check** 操作時に検出された不一致の数を表示します。
- **Cpy%Sync** フィールドは、**sync** 操作の進捗を表示します。
- **lv_attr** フィールドは、追加のインジケーターを提供します。このフィールドのビット 9 は、論理ボリュームの正常性を示し、以下のインジケーターに対応しています。
  - (**m**) (不一致) は、RAID 論理ボリュームに不一致があることを示します。この文字は、スクラビング操作で RAID に一貫性がない部分があることを検出した後に表示されます。
  - (**r**) (更新) は、LVM がデバイスラベルを読み取り、デバイスを稼働できると認識した場合でも、RAID アレイのデバイスに障害が発生し、カーネルがこれを障害と認識していることを示します。デバイスが利用可能になったことをカーネルに通知するように論理ボリュームを更新するか、デバイスに障害が発生したと思われる場合はデバイスを交換します。

関連情報

- 詳細は、**lvchange(8)** および **lvmraid(7)** の man ページを参照してください。

68.14.8.2. LVM RAID のデバイスに障害が発生しました。

RAID は従来の LVM ミラーリングとは異なります。LVM ミラーリングでは、障害が発生したデバイスを削除する必要がありました。削除しないと、ミラー化論理ボリュームがハングします。RAID アレイは、障害があるデバイスがあっても稼働し続けることができます。RAID1 以外の RAID タイプでデバイスを削除すると、レベルが低い RAID に変換されます (たとえば、RAID6 から RAID5、もしくは RAID4 または RAID5 から RAID0)。

そのため、障害のあるデバイスを無条件に削除してから交換するのではなく、**lvconvert** コマンドで **--repair** 引数を使用して、RAID ボリュームのデバイスを 1 回で置き換えることができます。

68.14.8.3. 論理ボリュームの障害が発生した RAID デバイスの交換

LVM RAID デバイス障害が一時的な障害であったり、障害が発生したデバイスの修復が可能な場合は、障害が発生したデバイスの復旧を開始できます。

前提条件

- 以前に不具合を起こしたデバイスが機能するようになりました。

手順

- RAID デバイスが含まれる論理ボリュームを更新します。

```
# lvchange --refresh my_vg/my_lv
```

検証手順

- 復元されたデバイスで論理ボリュームを調べます。

```
# lvs --all --options name,devices,lv_attr,lv_health_status my_vg
```

68.14.8.4. 論理ボリュームに障害が発生した RAID デバイスの交換

この手順では、LVM RAID 論理ボリュームで物理ボリュームとして機能する障害のあるデバイスを置き換えます。

前提条件

- ボリュームグループには、障害が発生したデバイスを置き換えるのに十分な空き容量を提供する物理ボリュームが含まれています。
  ボリュームグループに十分な空きエクステントがある物理ボリュームがない場合は、**vgextend** ユーティリティーを使用して、十分なサイズの物理ボリュームを新たに追加します。

手順

1. 以下の例では、RAID 論理ボリュームが次のように配置されます。

```
# lvs --all --options name,copy_percent,devices my_vg

  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdc1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          /dev/sdc1(0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

2. **/dev/sdc** デバイスに障害が発生した場合、**lvs** コマンドの出力は以下のようになります。

```
# lvs --all --options name,copy_percent,devices my_vg

  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and assumed devices.
  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         [unknown](1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          [unknown](0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

3. 障害が発生したデバイスを交換して、論理ボリュームを表示します。

```
# lvconvert --repair my_vg/my_lv

  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and assumed devices.
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
  Faulty devices in my_vg/my_lv successfully replaced.
```

オプション:障害が発生したデバイスを交換する物理ボリュームを手動で指定するには、コマンドの最後に物理ボリュームを追加します。

```
# lvconvert --repair my_vg/my_lv replacement_pv
```

4. 代替の論理ボリュームを調べます。

```
# lvs --all --options name,copy_percent,devices my_vg

  /dev/sdc: open failed: No such device or address
  /dev/sdc1: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  LV              Cpy%Sync Devices
  my_lv           43.79    my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdb1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          /dev/sdb1(0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

障害が発生したデバイスをボリュームグループから削除するまで、LVM ユーティリティーは、障害が発生したデバイスが見つけられないことを示しています。

5. 障害が発生したデバイスをボリュームグループから削除します。

```
# vgreduce --removemissing VG
```

68.14.9. マルチパス化された LVM デバイスに対する重複した物理ボリューム警告のトラブルシューティング

マルチパスストレージで LVM を使用する場合は、ボリュームグループまたは論理ボリュームのリストを表示する LVM コマンドを実行すると、以下のようなメッセージが表示される場合があります。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/dm-5 not /dev/sdd
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowerb not /dev/sde
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sddlmab not /dev/sdf
```

これらの警告のトラブルシューティングにより、LVM が警告を表示する理由を理解し、または警告を非表示にできます。

68.14.9.1. 重複した PV 警告の原因

Device Mapper Multipath (DM Multipath)、EMC PowerPath、または Hitachi Dynamic Link Manager (HDLM) などのマルチパスソフトウェアがシステム上のストレージデバイスを管理すると、特定の論理ユニット (LUN) への各パスが異なる SCSI デバイスとして登録されます。

マルチパスソフトウェアは、各パスにマップする新しいデバイスを作成します。各 LUN には、同じ基礎となるデータを参照する **/dev** ディレクトリーに複数のデバイスノードがあるため、すべてのデバイスノードには同じ LVM メタデータが含まれます。

表68.6 異なるマルチパスソフトウェアでのデバイスマッピングの例

マルチパスソフトウェア	LUN への SCSI パス	マルチパスデバイスパスへのマッピング
DM Multipath	**/dev/sdb** および **/dev/sdc**	**/dev/mapper/mpath1** または **/dev/mapper/mpatha**
EMC PowerPath		**/dev/emcpowera**
HDLM		**/dev/sddlmab**

複数のデバイスノードが原因で、LVM ツールは同じメタデータを複数回検出し、複製として報告します。

68.14.9.2. PV の重複警告が発生した場合

LVM は、以下のいずれかのケースで重複した PV 警告を表示します。

同じデバイスへの単一パス

出力に表示される 2 つデバイスは、両方とも同じデバイスへの単一パスです。
以下の例は、重複デバイスが、同じデバイスへの両方の単一パスである、重複した PV の警告を示しています。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sdd not /dev/sdf
```

multipath -ll コマンドを使用して現在の DM Multipath トポロジーをリスト表示すると、同じマルチパスマップの下に **/dev/sdd** と **/dev/sdf** の両方を確認できます。

これらの重複メッセージは警告のみで、LVM 操作が失敗しているわけではありません。代わりに、LVM が物理ボリュームとしてデバイスのいずれかのみを使用して他を無視していることを警告します。

メッセージは、LVM が誤ったデバイスを選択するか、ユーザーが警告を中断していることを示す場合は、フィルターを適用できます。フィルターは、物理ボリュームに必要なデバイスのみを検索し、マルチパスデバイスへの基礎となるパスを省略するように LVM を設定します。その結果、警告が表示されなくなりました。

マルチパスマップ

出力に表示される 2 つのデバイスは、両方ともマルチパスマップです。
以下の例は、両方のマルチパスマップである 2 つのデバイスに対する重複した物理ボリューム警告を示しています。重複した物理ボリュームは、同じデバイスへの異なるパスではなく、2 つのデバイスに置かれます。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/mapper/mpatha not
/dev/mapper/mpathc

Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowera not
/dev/emcpowerh
```

この状況は、同じデバイスへの両方の単一パスであるデバイスに対する重複する警告よりも複雑です。これらの警告は、多くの場合、マシンがアクセスできないデバイス (LUN クローンやミラーなど) にアクセスしていることを意味します。

マシンから削除するデバイスが分からないと、この状況は復旧できない可能性があります。Red Hat は、この問題に対処するために Red Hat テクニカルサポートにお問い合わせください。

68.14.9.3. PV の重複警告を防ぐ LVM デバイスフィルターの例

以下の例は、1 つの論理ユニット (LUN) への複数のストレージパスによって引き起こされる、重複した物理ボリュームの警告を回避する LVM デバイスフィルターを示しています。

設定するフィルターには、LVM がメタデータをチェックする必要があるすべてのデバイスが含まれている必要があります。たとえば、root ボリュームグループのあるローカルのハードドライブや、マルチパスを設定したデバイスなどです。マルチパスデバイスへの基礎となるパス (**/dev/sdb**、**/dev/sdd** など) を拒否すると、マルチパスデバイス自体で一意の各メタデータ領域が一度検出されるため、重複した物理ボリュームの警告を回避できます。

- このフィルターは、最初のハードドライブと DM Multipath デバイスの次のパーティションを受け入れますが、その他のパーティションはすべて拒否します。

```
filter = [ "a|/dev/sda2$|", "a|/dev/mapper/mpath.*|", "r|.*|" ]
```

- このフィルターは、すべての HP SmartArray 全コントローラーと、EMC PowerPath デバイスを許可します。

```
filter = [ "a|/dev/cciss/.*|", "a|/dev/emcpower.*|", "r|.*|" ]
```

- このフィルターは、最初の IDE ドライブとマルチパスデバイス上のパーティションをすべて受け入れます。

```
filter = [ "a|/dev/hda.*|", "a|/dev/mapper/mpath.*|", "r|.*|" ]
```

68.14.9.4. LVM デバイスフィルター設定の適用

この手順では、LVM スキャンするデバイスを制御する LVM デバイスフィルターの設定を変更します。

前提条件

- 使用するデバイスフィルターパターンを準備します。

手順

1. **/etc/lvm/lvm.conf** ファイルを変更せずに、デバイスフィルターパターンをテストします。LVM コマンドに、**--config 'devices{ filter = [your device filter pattern] }'** オプションを指定して使用します。以下に例を示します。

```
-
```

```
# lvs --config 'devices{ filter = [ "a|/dev/emcpower.*|", "r|.*|" ] }'
```

2. **/etc/lvm/lvm.conf** 設定ファイルで **filter** オプションを編集して、新しいデバイスフィルターパターンを使用します。
3. 新しい設定で、使用する物理ボリュームまたはボリュームグループがないことを確認します。

```
# pvscan
```

```
# vgscan
```

4. 再起動時に LVM が必要なデバイスのみをスキャンするように **initramfs** ファイルシステムを再構築します。

```
# dracut --force --verbose
```

68.14.9.5. 関連情報

- LVM デバイスの可視性および使用を制限する
- LVM デバイスフィルター