Red Hat Enterprise Linux 8

高度な RHEL インストールの実行

Red Hat Enterprise Linux リリース版のインストールドキュメント

Last Updated: 2019-05-09

Red Hat Enterprise Linux 8 高度な RHEL インストールの実行

Red Hat Enterprise Linux リリース版のインストールドキュメント

法律上の通知

概要

本書は、キックスタートを使用して Red Hat Enterprise Linux の高度なインストールを実行し、高度なインストールオプションを設定するユーザーを対象としています。

目次

RED HAT ドキュメントへのフィードバック

ドキュメントの改善に関するご意見やご要望をお聞かせください。

- 特定の文章に簡単なコメントを記入する場合は、ドキュメントが Multi-page HTML 形式になっているのを確認してください。コメントを追加する部分を強調表示し、そのテキストの下に表示される **Add Feedback** ポップアップをクリックし、表示された手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
    1. Bugzilla の Web サイトにアクセスします。
    2. Component で **Documentation** を選択します。
    3. **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記入してください。
    4. **Submit Bug** をクリックします。

第1章 はじめに

Red Hat Enterprise Linux 8 は、より少ない労力でより迅速にワークロードを提供するのに必要なツールを使用して導入したハイブリッドクラウドで、安定し、安全で一貫した基盤を提供します。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. サポートされているアーキテクチャー

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

- AMD、Intel 64 ビット、および 64 ビット ARM
- IBM Power Systems
- IBM Z

1.2. インストールの用語

本セクションでは、Red Hat Enterprise Linux インストール用語を説明します。概念が同じでも、アップストリームかダウンストリームに使用されているかによって、別の用語が使用されている可能性もあります。

Anaconda - Fedora、Red Hat Enterprise Linux、およびその他の派生製品に使用されるオペレーティングシステムインストーラー。Anaconda は、Gtk ウィジェット (C で記述)、systemd ユニット、dracut ライブラリーなどの追加ファイルが含まれる、一連の Python モジュールおよびスクリプトです。これは、同時に、結果として得られる (ターゲット) システムのパラメーターを設定するツールを形成します。この文書では、**インストールプログラム** という用語は、**Anaconda** のインストールに関する機能を表しています。

1.3. インストール方法

以下のいずれの方法を使用して、Red Hat Enterprise Linux をインストールしてください。

1.3.1. クイックインストール

AMD、Intel 64 ビット、および 64 ビット ARM のアーキテクチャーに Red Hat Enterprise Linux をインストールします。このクイックインストールは、Red Hat Enterprise Linux とお使いの環境に精通し、インストールプログラムが指定するデフォルト設定を受け入れることを前提とします。

1.3.2. グラフィカルインストール

サポートされているすべてのアーキテクチャーに、グラフィカルユーザーインターフェースを使用して Red Hat Enterprise Linux をインストールし、特定の要件に合わせてグラフィカル設定をカスタマイズします。

1.3.3. 自動インストール

キックスタートを使用して、サポートされるすべてのアーキテクチャーに Red Hat Enterprise Linux をインストールします。自動インストールでは、オペレーティングシステムの無人インストール作業を実行できます。

関連資料

- グラフィカルユーザーインターフェースを使用して、AMD、Intel 64 ビット、および 64 ビット ARM でクイックインストールを実行するには、**『標準的な RHEL インストールの実行』**を参照してください。
- グラフィカルユーザーインターフェースを使用してサポートされるすべてのアーキテクチャーでグラフィカルインストールを実行するには、**『標準的な RHEL インストールの実行』**を参照してください。

パート I. キックスタートを使用した自動インストールの実行

第2章 キックスタートインストールの基礎

キックスタートの概要と、Red Hat Enterprise Linux のインストールを自動化するために使用する方法を学びます。

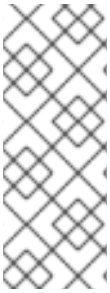
2.1. キックスタートを使ったインストールの概要

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がキックスタートを使用したインストールになります。キックスタートファイルには、システムで使用するタイムゾーン、ドライブのパーティション設定、インストールするパッケージなど、インストールプログラムで入力を求められる一般的な質問に対する答えがすべて格納されます。このため、準備されたキックスタートファイルをインストールの開始時に提供することで、ユーザーによる作業を必要としない自動インストールが実行できるようになります。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。

インストールを自動化する以外にも、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。グラフィカルインターフェースで 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、各コンピューターがインストール中に読み込むことができます。この方法は、1 つのキックスタートファイルを使用して複数のマシンに Red Hat Enterprise Linux をインストールできるので、ネットワークおよびシステム管理者にとって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、すべて **/tmp** ディレクトリーに保存されます。



注記

Red Hat Enterprise Linux の以前のバージョンでは、キックスタートは、同様のシステムをアップグレードすることを許可します。Red Hat Enterprise Linux 7 以降では、この機能は削除されていて、システムのアップグレードではなく、特殊なツールによって処理されます。Red Hat Enterprise Linux 8 へのアップグレードの詳細は『RHEL 8 へのアップグレード』 および 『Differences between RHEL 7 and RHEL 8』 を参照してください。

2.2. 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 またはローカルのハードドライブを使用するか、NFS、FTP、HTTP、HTTPS などにより実行できます。

キックスタートを使用する場合は、次の手順を行う必要があります。

1. キックスタートファイルを作成します。手動インストール、またはオンライン生成ツールを使用してファイルを作成し、後でそれを編集できます。
2. HTTP(S) サーバー、FTP サーバー、または NFS サーバーを使用して、リムーバブルメディア、ハードドライブ、ネットワークの場所のいずれかの場所でキックスタートファイルを利用可能にします。
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。
4. キックスタートファイルと同様に、インストールソースを使用できるようにします。

5. キックスタートインストールを開始します。インストール時に、起動メニューまたはブートローダー設定ファイルで **inst.ks=** を使用してキックスタートファイルを読み込んで使用します。

6. インストールを終了します。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自動的に行われます。この1つ以上の必須部分が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。

第3章 キックスタートファイルの作成

最初から手動でファイルを作成するよりも、簡単かつ迅速にキックスタートファイルを作成する方法を学びます。方法は以下のとおりです。

- 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します。
- オンラインのキックスタート設定ツールの使用

複雑な部分のみを変更できるように、後でファイルを手動で編集およびカスタマイズできます。

3.1. 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

まず任意のシステムに手動のインストールを行うことが、キックスタートファイル作成の推奨方法となります。インストールが完了すると、インストール中に選択された選択肢がすべて **anaconda-ks.cfg** という名前のファイルに保存されます。このファイルはインストールが完了したシステムの **/root/** ディレクトリーに置かれます。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、必要に応じて変更を加えると、この後のインストールでこの設定ファイルを使用できます。

手順

1. システムでグラフィカルインターフェースを使用して RHEL をインストールします。
   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. システムを再起動します。
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。
   - 端末内のファイルの内容を表示するには、以下を実行します。

     ```
     # 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、コピーにアクセス権を忘れずに変更してください。

重要

ファイルには、ユーザとパスワードの情報が含まれます。

関連資料

- 標準的な RHEL インストールの実行

3.2. 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の Labs の Kickstart Configuration Tool 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説明し、作成される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。ただし、

このツールは現在、高度なパーティション作成をサポートしていません。

手順

1. Lab で提供されているキックスタートジェネレーターの情報は https://access.redhat.com/labsinfo/kickstartconfig を参照してください。
2. 見出しの左にある青色の **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. インストールするシステムを説明するフォームの詳細を記入してください。
   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、ページの先頭に戻るまでスクロールし、赤色の **Download** ボタンをクリックします。
   Web ブラウザーはファイルを保存します。

第4章 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

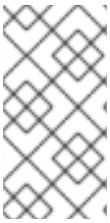
インストールするシステムでインストールプログラムがこのファイルを読み込むことができるように、キックスタートファイルを置くために必要なオプションを説明します。

4.1. 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を NFS サーバーに置く方法を説明します。この方法では、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。ネットワークベースのインストールでは、TFTP サーバーとともに使用すると、ネットワークからインストールを起動できるので便利です。このアプローチにより、物理メディアを作成し、同時に複数のシステムに Red Hat Enterprise Linux をデプロイする必要がなくなります。

前提条件

- インストールするシステムから、ローカルネットワークで到達可能なサーバーシステムへの管理者レベルのアクセスがある。
- ファイアウォールにより、インストールするシステムがサーバーにアクセスできるようになっていることを確認している。



手順

NFS ベースのインストールを実行するには、別のシステムが NFS ホストとして機能する必要があります。この手順は、プロセスの基本的な概要です。NFS サーバーをセットアップする手順は、システムのアーキテクチャー、オペレーティングシステム、パッケージマネージャー、およびサービスマネージャーによって異なります。

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、以下の構文で行を追加します。

```
/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へのフルパスに置き換えます。クライアントの代わりに、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構成は次のようになります。

```
/rhel8-install *
```

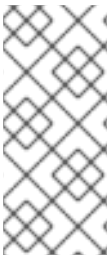
5. **/etc/exports** ファイルを保存して、テキストエディターを終了します。
6. nfs サービスを起動します。

```
# systemctl start nfs.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.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は **nfs:myserver.example.com:/rhel8-install/my-ks.cfg** となります。

4.2. HTTP サーバーまたは HTTPS サーバーでキックスタートファイルの準備

ネットワークベースのインストールの場合は、HTTP サーバーまたは HTTPS サーバーにキックスタートファイルを置きます。

前提条件

- インストールするシステムから、ローカルネットワークで到達可能なサーバーシステムへの管理者レベルのアクセスがある。
- ファイアウォールにより、インストールするシステムがサーバーにアクセスできるようになっていることを確認している。

手順

1. root で以下のコマンドを実行して、**httpd** パッケージをインストールします。

```
# yum install httpd
```

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。これは、POODLE SSL の脆弱性 (CVE-2014-3566) によるものです。詳細は「httpd における POODLE SSLv3.0 脆弱性問題の解決方法 (CVE-2014-3566)」をご覧ください。

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、**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
```

インストールツリーへのアクセスが可能になり、インストールソースとして使用できるようになります。

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。

関連資料

- HTTP サーバーおよび FTP サーバーの詳細は『Deploying different types of servers』を参照してください。

4.3. FTP サーバーでキックスタートファイルの準備

ネットワークベースのインストールの場合は、FTP サーバーにキックスタートファイルを置きます。

前提条件

- インストールするシステムから、ローカルネットワークで到達可能なサーバーシステムへの管理者レベルのアクセスがある。
- ファイアウォールにより、インストールするシステムがサーバーにアクセスできるようになっていることを確認している。

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。利用可能なオプションは、man ページの **vsftpd.conf(5)** 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 および SSLv3 は必ず無効にしてください。これは、POODLE SSL 脆弱性 (CVE-2014-3566) に対応するためです。詳細は「vsftpd における POODLE SSLv3.0 脆弱性問題の解決方法 (CVE-2014-3566)」を参照してください。

3. **/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。
4. **vsftpd** サービスを開始します。

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

5. **/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、再実行して必ず編集後のファイルを読み込ませてください。再実行する場合は、次のコマンドを使用します。

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

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/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 **ftp://myserver.example.com/rhel8-install/my-ks.cfg** となります。

4.4. ローカルボリュームでキックスタートファイルの準備

キックスタートファイルは、インストールされるシステムのボリュームに直接置くことができます。これにより、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、インストールするシステムに移動できるドライブが必要です。
- ドライブには、インストールプログラムで読み込むことができ、パーティションが含まれている必要があります。
  TODO パーティションの基礎となる許可されるタイプは何ですか?
- ドライブがすでにシステムに接続する必要があり、そのボリュームがマウントされている必要があります。

手順

手順

1. ドライブのボリュームがマウントされている場所と、UUID を確認します。

```
# lsblk -l -p -o name,rm,ro,hotplug,size,type,mountpoint,uuid
```

2. ドライブからマウントされた、適切な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任意の場所に、キックスタートファイルをコピーします。
4. **inst.ks=** オプションで後で使用する文字列の形式は **hd: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。
5. ドライブボリュームのマウントを解除します。

```
# 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。

4.5. 自動ロード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

特別に名前が付けられたキックスタートファイルが、インストールするシステムで特別に名前が付いたボリュームの 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ロードできるようになります。

前提条件

- USB スティックなど、インストールするシステムに移動できるドライブが必要です。
- ドライブには、インストールプログラムで読み込むことができ、パーティションが含まれている必要があります。
  TODO パーティションの基礎となる許可されるタイプは何ですか?
- ドライブがすでにシステムに接続する必要があり、そのボリュームがマウントされている必要があります。

手順

1. ドライブのボリュームがマウントされている場所を確認します。

```
# lsblk -l -p
```

2. ドライブからマウントされた、適切な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ルートにキックスタートファイルをコピーします。
4. 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。

5. ドライブボリュームのマウントを解除します。

```
# 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。

第5章 キックスタートインストール用のインストールソース作成

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が含まれる Binary DVD ISO イメージを使用して、Boot ISO イメージにインストールソースを作成する方法を説明します。

5.1. インストールソースの種類

最小ブートイメージのインストールソースは以下のいずれかになります。

- **DVD** - Binary DVD ISO イメージを DVD に書き込み、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に設定できます。
- **ハードドライブまたは USB ドライブ** - Binary 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。インストールが始まると、インストールプログラムがメディアを検出することはできません。
  - **ハードドライブの制限** - ハードドライブの Binary 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パーティションに置く必要があります。ファイルシステムは **xfs、ext2、ext3、ext4、および vfat (FAT32)** になります。

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、exFAT ファイルシステムも利用できますが、いずれもインストール時にマウントできません。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。ただし、FAT32 ファイルシステムは、4 GiB 以上のファイルには対応しません。

Red Hat Enterprise Linux 8 では、ローカルハードドライブのリポジトリーからインストールできます。その場合は、ISO イメージの代わりに、ディレクトリーを指定してください (**inst.repo=hd:<device>:<path to the repository>** など)。

- **ネットワークの場所** - Binary DVD ISO イメージ、またはインストールツリー (Binary DVD ISO イメージから抽出したコンテンツ) をネットワークの場所にコピーし、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できます。
  - **NFS** - Binary 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** - 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の場所にあります。

5.2. NFS サーバーへのインストールソースの作成

この手順では、インストールソースを NFS サーバーに置く方法を説明します。この方法では、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。ネットワークベースのインストールでは、TFTP サーバーとともに使用すると、ネットワークからインストールを起

動できるので便利です。このアプローチにより、物理メディアを作成し、同時に複数のシステムに Red Hat Enterprise Linux をデプロイする必要がなくなります。

前提条件

- Binary DVD イメージをダウンロードしている。
- 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。
- 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サーバー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。次の表は、ネットワークベースのインストールの種類に応じて開く必要があるポートの一覧です。詳細は『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。

表5.1 ネットワークインストール用ポート

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

手順

注記

NFS インストールでは、ネットワークファイルシステムサーバーでエクスポートされたディレクトリーにある Binary DVD ISO イメージを使用します。このディレクトリーは、システムによる読み取りが可能な状態である必要があります。NFS ベースのインストールを実行するには、別のシステムが NFS ホストとして機能する必要があります。NFS サーバーをセットアップする手順は、システムのアーキテクチャー、オペレーティングシステム、パッケージマネージャー、およびサービスマネージャーによって異なります。この手順は、プロセスの基本的な概要です。

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. Binary 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、以下の構文で行を追加します。

```
/exported_directory/ clients
```

4. **/exported_directory/** を、ISO イメージが含まれるディレクトリーへ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構成は次のようになります。

```
/rhel8-install *
```

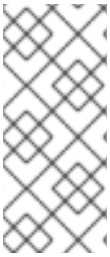
5. **/etc/exports** ファイルを保存して、テキストエディターを終了します。
6. nfs サービスを起動します。

```
# systemctl start nfs.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.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 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。

5.3. HTTP または HTTPS を使用したインストールソースの作成

この手順は、インストールツリー (Binary 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) を使用する、ネットワークベースのインストールを作成する手順を説明します。インストールソースには、HTTP、または HTTPS を使用してアクセスします。

前提条件

- Binary DVD イメージをダウンロードしている。
- 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。

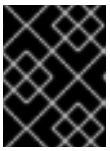
手順

1. root で以下のコマンドを実行して、**httpd** パッケージをインストールします。

```
# yum install httpd
```

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。これは、POODLE SSL の脆弱性 (CVE-2014-3566) によるものです。詳細は「httpd における POODLE SSLv3.0 脆弱性問題の解決方法 (CVE-2014-3566)」をご覧ください。



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、**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. HTTP(S) サーバーに Binary DVD ISO イメージをコピーします。
3. **mount** コマンドを使用して、Binary DVD ISO イメージを適切な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 /mount_point/
```

   - **/image_directory/image.iso** を、Binary DVD ISO イメージのパスに置き換えます。
   - **/mount_point/** を、ISO イメージに含まれるファイルが保存されるディレクトリーのパスに置き換えます。

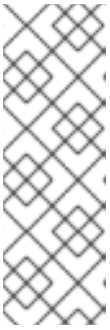
4. マウントされたイメージから、HTTP(S) サーバーの root にファイルをコピーします。このコマンドは、イメージに含まれるファイルが保存される **/var/www/html/rhel8-install/directory** を作成します。

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

5. **httpd** サービスを開始します。

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

インストールツリーへのアクセスが可能になり、インストールソースとして使用できるようになります。



注記

インストールソースとして設定する場合は、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、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 からの相対パス) 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の保存場所が **/var/www/html/rhel8-install/** の場合、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。

関連資料

- HTTP サーバーおよび FTP サーバーの詳細は『Deploying different types of servers』を参照してください。

5.4. FTP を使用したインストールソースの作成

この手順では、インストールツリー (Binary DVD ISO イメージから抽出したコンテンツと、有効な **.treeinfo** ファイルを含むディレクトリー) を使用する、ネットワークベースのインストールを作成する方法を説明します。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

- Binary DVD イメージをダウンロードしている。
- 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。

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。利用可能なオプションは、man ページの **vsftpd.conf(5)** 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 および SSLv3 は必ず無効にしてください。これは、POODLE SSL 脆弱性 (CVE-2014-3566) に対応するためです。詳細は「vsftpd における POODLE SSLv3.0 脆弱性問題の解決方法 (CVE-2014-3566)」を参照してください。

3. Binary DVD ISO イメージを FTP サーバーにコピーします。
4. mount コマンドを使用して、Binary DVD ISO イメージを適切な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 /mount_point
```

- **/image_directory/image.iso** を、Binary DVD ISO イメージのパスに置き換えます。
- **/mount_point** を、ISO イメージに含まれるファイルが保存されるディレクトリーのパスに置き換えます。

5. 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。

```
# 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。

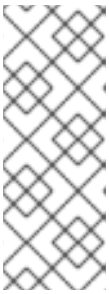
6. **vsftpd** サービスを開始します。

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

7. **/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、再実行して必ず編集後のファイルを読み込ませてください。再実行する場合は、次のコマンドを使用します。

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

インストールツリーへのアクセスが可能になり、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。

第6章 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。

それぞれを実行する方法を学びます。

6.1. 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** を使用してください。location は、キックスタートファイルの場所に置き換えます。起動オプションを指定する方法は、システムのアーキテクチャーによって異なります。

前提条件

- インストール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を用意している。

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) のいずれか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てネットワークを設定する必要があります。
   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ます。
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されます。

6.2. PXE を使用した自動キックスタートインストールの開始

64 ビット AMD、Intel、および 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で開始することが可能になります。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することが可能になります。

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、IBM Z で PXE ブートを使用することはできません)。

前提条件

- インストール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を用意している。
- システムを起動し、インストールを開始するのに使用できる PXE サーバーがある。

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 を持つ AMD64 システムおよび Intel 64 システムでは、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-7/7.x/Server/x86_64/kickstarts/ks.cfg
     ```

   - GRUB2 ブートローダーを使用しているシステム (64 ビット AMD、Intel、および ARM システムで UEFI ファームウェアを使用している場合、ならびに IBM Power Systems サーバー) では、ファイル名は **grub.cfg** になります。このファイル内のインストールエントリーにある 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-7/7.x/Server/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化されます。

関連資料

- PXE サーバーの設定方法は9章*PXE を使用したネットワークからのインストール準備*を参照してください。

6.3. 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** で用意されたボリュームと、**ks.cfg** として root に存在するキックスタートがある。
  TODO xref the setup module for this!

- インストールプログラムの起動時に、このボリュームを含むドライブが利用できるようになっている必要があります。

手順

- インストールプログラムの起動
  ファイルが自動的に検出され、自動化されているキックスタートインストールを開始するのに使用されます。

第7章 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。通常は、以下を行います。

- 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。
- 以前のバージョンの RHEL をインストールするのに使用したキックスタートファイルを適用するために必要な変更のレポートを生成します。

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
```

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。

手順

1. キックスタートファイルで **ksvalidator** を実行します。

```
$ ksvalidator /path/to/kickstart.ks
```

/path/to/kickstart.ks を、確認するキックスタートファイルへ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することはできません。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** セクションは検証しません。

関連資料

- man ページの **ksvalidator(1)**

7.3. RHEL のバージョン間のキックスタートファイルの構文の変更一覧

ksverdiff ユーティリティーは、Red Hat Enterprise Linux のメジャーリリース間で追加されたコマンドおよびオプションの変更一覧を表示するのに使用します。これは、Red Hat Enterprise Linux 8 で使用する既存のキックスタートファイルを更新する際に便利です。

手順

- **ksverdiff** コマンドを実行します。

```
$ ksverdiff -f RHEL7 -t RHEL8
```

これは、Red Hat Enterprise Linux のバージョン 7 と 8 の間の変更を表示します。また、**RHEL7** を、**RHEL6** に変更すると、バージョン 6 が使用できます。

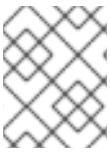
関連資料

- man ページの **ksverdiff(1)**

パート II. 高度な設定オプション

第8章 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お勧めします。

8.1. 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。

8.2. 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。

8.3. 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。

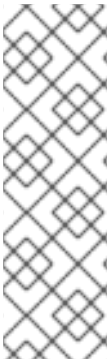
アシスト付き

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 パッケージへ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 イメージへのパスになります。このオプションを指定するすると、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 イメージへのパスになります。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 1 つにより署名され、カーネル内で対応する公開鍵により認証されます。追加で別の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。

8.4. 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、ドライバー更新用 ISO イメージを受け取っている。
- ドライバー更新用 ISO イメージを、CD または DVD に書き込んでいる。

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、**rpms** という名前のディレクトリーです。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含まれる RPM パッケージが含まれます。

8.5. 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、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

1. 前提条件の手順が完了して、インストールプログラムが起動すると、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時にシステムにインストールされます。

8.6. 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、ボリュームラベル **OEMDRV** が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ているか、ドライバー更新 CD または DVD を準備し、システムの CD/DVD ドライブに挿入している。

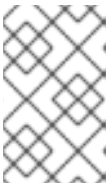
注記

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。詳細は 「手動によるドライバー更新の実行」 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. 必要に応じて、ドライバー更新 ISO ファイルを選択してください。



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。

8.7. 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
USB フラッシュドライブまたは Web サーバーに、ドライバー更新用 ISO イメージファイルを置き、それをコンピューターに接続します。

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。この場所は、ドライバー更新のパスです。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。

関連資料

- **inst.dd** 起動オプションの参照は、アップストリームの「inst.dd boot option」を参照してください。
- すべての起動オプションの詳細は、アップストリームの「Boot Options」を参照してください。

8.8. 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 **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
```

modprobe.blacklist= オプション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** キーを押して、起動プロセスを確認します。

第9章 PXE を使用したネットワークからのインストール準備

このセクションでは、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。

9.1. 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。

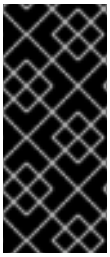
注記

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。

ステップ

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。このプロトコルを介してブートファイル (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。

関連資料

- ネットワークの場所にインストールの ISO イメージをエクスポートする方法は、5章 *キックスタートインストール用のインストールソース作成* を参照してください。

- TFTP サーバーおよび DHCP サーバーを設定し、TFTP サービスを開始する方法は、「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」、および「IBM Power システム向けにネットワークサーバーの設定」を参照してください。
- Red Hat Satellite は、PXE サーバーの設定を自動化できます。詳細は、Red Hat Satellite の製品ドキュメントを参照してください。

9.2. BIOS ベースのクライアント向けに TFTP サーバーの設定

この手順では、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 サービスを開始する方法を説明します。

手順

1. root で **tftp-server** パッケージをインストールします。

```
# yum install tftp-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。

3. **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。**/etc/dhcp/dhcpd.conf** ファイルの設定例は次のようになります。

```
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 "uefi/shim.efi";
     } else {
     filename "pxelinux/pxelinux.0";
   }
 }
}
```

4. バイナリーの DVD ISO イメージファイルの **SYSLINUX** パッケージから、**pxelinux.0** ファイルへアクセスします。

```
# 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 /publicly_available_directory
```

```
# umount /mount_point
```

5. パッケージを抽出します。

```
# rpm2cpio syslinux-tftpboot-version-architecture.rpm | cpio -dimv
```

6. **tftpboot**/ に **pxelinux**/ ディレクトリーを作成し、必要なファイル (**pxelinux.0** **libcom.c32**、**ldlinux.c32**、**vesamenu.c32** など) をコピーします。

```
# mkdir /var/lib/tftpboot/pxelinux
```

```
# cp publicly_available_directory/tftpboot/pxelinux.0 /var/lib/tftpboot/pxelinux
```

7. **pxelinux**/ ディレクトリーに **pxelinux.cfg**/ ディレクトリーを作成します。

```
# mkdir /var/lib/tftpboot/pxelinux/pxelinux.cfg
```

8. デフォルトの設定ファイルを **pxelinux.cfg**/ ディレクトリーに追加します。**/var/lib/tftpboot/pxelinux/pxelinux.cfg/default** にある設定ファイルは次のようになります。

```
default vesamenu.c32
prompt 1
timeout 600

display boot.msg

label linux
  menu label ^Install system
  menu default
  kernel images/RHEL-8.0/vmlinuz
  append initrd=images/RHEL-8.0/initrd.img ip=dhcp
inst.repo=http://10.32.5.1/mnt/archive/RHEL-8/8.x/Server/x86_64/os/
label vesa
  menu label Install system with ^basic video driver
  kernel images/RHEL-8.0/vmlinuz
  append initrd=images/RHEL-8.0/initrd.img ip=dhcp inst.xdriver=vesa nomodeset
inst.repo=http://10.32.5.1/mnt/archive/RHEL-8/8.x/Server/x86_64/os/
label rescue
  menu label ^Rescue installed system
  kernel images/RHEL-8.0/vmlinuz
  append initrd=images/RHEL-8.0/initrd.img rescue
label local
  menu label Boot from ^local drive
  localboot 0xffff
```

-

注記

インストールプログラムのイメージと、インストールソースを指定する場合は、**inst.repo=** オプションを使用する必要があります。このオプションがないと、インストールプログラムは起動できません。

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。この例では、**/var/lib/tftpboot/pxelinux/images/RHEL-8.0/** ディレクトリーを使用します。

```
# mkdir -p /var/lib/tftpboot/pxelinux/images/RHEL-8.0/
# cp /path_to_x86_64_images/pxeboot/{vmlinuz,initrd.img}
/var/lib/tftpboot/pxelinux/images/RHEL-8.0/
```

10. **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

11. **tftp** サービスを管理する **xinetd** サービスを開始して有効にします。

```
# systemctl start xinetd
# systemctl enable xinetd
```

これにより、PXE ブートサーバーでは、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. UEFI ベースのクライアント向けに TFTP サーバーの設定

この手順では、TFTP サーバーおよび DHCP サーバーを設定し、UEFI ベースの AMD および Intel の 64 ビットシステム、ならびに 64 ビットの 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。

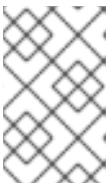
手順

1. root で **tftp-server** パッケージをインストールします。

```
# yum install tftp-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。

3. **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。**/etc/dhcp/dhcpd.conf** ファイルの設定例は次のようになります。

```
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 "shim.efi";
   } else {
     filename "pxelinux/pxelinux.0";
  }
  }
}
```

4. Binary DVD ISO イメージファイルにある **shim** パッケージの **shim.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。

```
# 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
/publicly_available_directory
```

```
# cp -pr /mount_point/BaseOS/Packages/grub2-efi-version-architecture.rpm
/publicly_available_directory
```

```
# umount /mount_point
```

5. パッケージを抽出します。

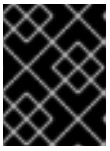
```
# rpm2cpio shim-version-architecture.rpm | cpio -dimv
```

```
# 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。

```
# cp publicly_available_directory/boot/efi/EFI/redhat/shimx64.efi /var/lib/tftpboot/uefi/
```

```
# cp publicly_available_directory/boot/efi/EFI/redhat/grubx64.efi /var/lib/tftpboot/uefi
```

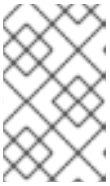


重要

shimx64.efi ファイルをコピーした **後** に、名前を **shim.efi** に変更する必要があります。

7. **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。**/var/lib/tftpboot/uefi/grub.cfg** にある設定ファイル例は、次のようになります。

```
set timeout=60
menuentry 'RHEL 8' {
  linuxefi images/RHEL-8.0/vmlinuz ip=dhcp inst.repo=http://10.32.5.1/mnt/archive/RHEL-8.0/Server/x86_64/os/
  initrdefi images/RHEL-8.0/initrd.img
}
```



注記

インストールプログラムのイメージと、インストールソースを指定する場合は、**inst.repo=** オプションを使用する必要があります。このオプションがないと、インストールプログラムは起動できません。

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。この例では、**/var/lib/tftpboot/images/RHEL-8.0/** ディレクトリーを使用します。

```
# mkdir -p /var/lib/tftpboot/images/RHEL-8.0/
# cp /path_to_x86_64_images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftpboot/images/RHEL-8.0/
```

9. **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

10. **tftp** サービスを管理する **xinetd** サービスを開始して有効にします。

```
# systemctl start xinetd
# systemctl enable xinetd
```

これにより、PXE ブートサーバーでは、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。

関連資料

- **shim** の詳細は、アップストリームのドキュメント「Using the Shim Program」 を参照してください。

9.4. IBM POWER システム向けにネットワークサーバーの設定

この手順では、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。

手順

1. root で **tftp-server** パッケージをインストールします。

```
# yum install tftp-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。

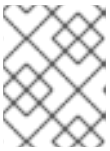
```
# firewall-cmd --add-service=tftp
```

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。

3. tftp のルートに、**GRUB2** ネットワークブート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 設定で構成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 サーバーが x86 マシンで実行している場合は、tftp root 内に **GRUB2** ネットワークブート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. **GRUB2** 設定ファイル (**/var/lib/tftpboot/boot/grub2/grub.cfg**) を作成します。以下は設定ファイルの例です。

```
set default=0
set timeout=5

echo -e "\
Welcome to the Red Hat Enterprise Linux 8 installer!\
\
"

menuentry 'Red Hat Enterprise Linux 8' {
  linux grub2-ppc64/vmlinuz ro ip=dhcp inst.repo=http://10.32.5.1/mnt/archive/RHEL-
8.0/Server/ppc64/os/
  initrd grub2-ppc64/initrd.img
}
```

注記

インストールプログラムのイメージと、インストールソースを指定する場合は、**inst.repo=** オプションを使用する必要があります。このオプションがないと、インストールプログラムは起動できません。

5. このコマンドを使用して Binary 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、Binary 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. **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。**/etc/dhcp/dhcpd.conf** ファイルの設定例は次のようになります。

```
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 (**subnet**、**netmask**、**routers**、**fixed-address**、および **hardware ethernet**) を調整します。**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. **dhcpd** サービスを開始して有効にします。

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

10. **tftp** サービスを管理する **xinetd** サービスを開始して有効にします。

```
# systemctl start xinetd
# systemctl enable xinetd
```

これにより、PXE ブートサーバーでは、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。

パート III. キックスタートの参照

付録A キックスタートスクリプト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。

A.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なテキストファイルです。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。コマンドおよびオプションは 1 行で記載される必要があります。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。
各セクションは決められた順序で指定してください。セクション内の項目は、特に指定がない限り順序は関係ありません。セクションの順序は次のようになります。

- **コマンドセクション**。このセクションは、開始コマンドおよび終了コマンドを使用しません。必要なコマンドおよびオプションをすべて追加する必要があります。
- **アドオンセクション**。このセクションは、**%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**、**%post**、および **%onerror** で開始します。これらのセクションは、任意の順序で指定でき、必須ではありません。

コマンドセクションを除くすべてのセクションは **%end** で終わります。それ以外の場合は、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。

必要のない項目が省略できます。必須項目が省略されている場合は、通常のインストールと同様、その関連項目の回答が求められます。回答を入力すると、インストールは自動的に続行します (他にも省略されている部分があればその部分まで)。

A.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、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。

A.2.1. パッケージの選択セクション

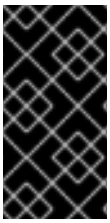
%packages コマンドを使用して、インストールするソフトウェアパッケージを説明するキックスタートセクションを開始します。**%packages** セクションは、**%end** コマンドで終了する必要があります。

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一覧は、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、パッケージ一覧があります。グループがインストールに選択されていると、パッケージ一覧で **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。

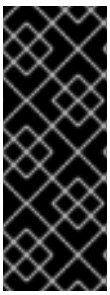
パッケージグループや環境は、その ID (**<id>** タグ) もしくは名前 (**<name>** タグ) を使用して指定できます。

どのパッケージをインストールすればよいか分からない場合は、**最小限のインストール** 環境を選択することを Red Hat では推奨しています。**最小限のインストール** は、Red Hat Enterprise Linux 8 の実行に必須のパッケージのみを提供します。これにより、システムが脆弱性に影響される可能性を大幅に減らします。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。**最小限のインストール**の詳細は、『Red Hat Enterprise Linux 8 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」を参照してください。



重要

32 ビットパッケージを 64 ビットシステムにインストールするには、**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。**--multilib** オプションもキックスタートファイルで指定する必要があります。利用可能なオプションは、下記を参照してください。



重要

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は実行しません。つまり、デフォルトでは、**root** 以外のユーザーは作成されません。追加のシステムをインストールする前に、キックスタートファイルの **user** オプションでユーザーを作成するか、**root** として仮想コンソールでインストール済みのシステムにログインして、**useradd** コマンドでユーザーを追加します。

A.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できるのは、1 つの環境だけです。

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** ファイルで指定されたとおり @ 記号で始め、その後に完全なグループ名、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ので、**%packages** セクションで指定する必要はありません。

***-comps-repository.architecture.xml** ファイルは、Red Hat Enterprise Linux 用の各バリアント用に、**Conflicts (variant)** と呼ばれるグループも定義します。このグループにはファイル競合を引き起こす既知のパッケージがすべて含まれ、除外されることを意図しています。

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 **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と、それは省略できます。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ため、それは省略できます。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、@ 起動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を一覧表示するには、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。

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。たとえば以下のようになります。

```
%packages
-@Graphical Internet
-autofs
-ipa*fonts
%end
```

重要

@Conflicts (variant) グループを除外した場合でも、キックスタートファイルで * のみを使用してすべての利用可能なパッケージを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、パッケージ選択全体で機能するものと、特定のグループのみで機能するものがあります。

関連資料

- パッケージの処理方法は、**『基本的なシステム設定の構成』**の「yum を使用したソフトウェアのインストール」を参照してください。
- モジュールおよびストリームの詳細は、『ユーザー領域コンポーネントのインストール、管理、および削除』 を参照してください。

A.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、**/usr/share/doc** ディレクトリーにインストールされるファイルは通常は除外されますが、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止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。

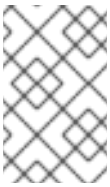
--multilib

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 (つまり、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し)、この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 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。このオプションを使用すると、32 ビットの AMD および Intel システム用のパッケージ (**i686** の印が付いている) がある場合は、それも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。

--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。**@Core** パッケージグループの無効化は、軽量コンテナの作成にのみ使用してください。**--nocore** を使用してデスクトップやサーバーシステムをインストールすると、使用できないシステムになってしまいます。



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。

--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。

A.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 Internet --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をインストールすることができません。

A.3. キックスタートにおけるインストール前スクリプト

インストールを開始する前に、インストール前のスクリプトがすぐに実行されます。

A.3.1. インストール前のスクリプトセクション

%preスクリプトは、キックスタートファイルの解析直後、インストールの開始前に、システムで実行されます。このセクションは、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末尾に配置し、**%pre** で開始し、**%end** で終了する必要があります。キックスタートファイルに **%post** セクションも含まれる場合は、**%pre** セクションと **%post** セクションの順番は重要ではありません。

%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と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** スクリプトを追加すると便利です。**%pre** スクリプトでの問題のデバッグは難しくなる可能性があるため、**%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。

インストール環境の **/sbin** ディレクトリー **/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が、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。キックスタートの、インストール前スクリプトのセクションは、複数のインストールツリーまたはソースメディアを管理できません。インストール前のスクリプトはインストールプロセスの第 2 段階で実行されるため、このような情報は作成された各キックスタートファイルに含める必要があります。

注記

インストール後のスクリプトとは違って、インストール前のスクリプトは chroot 環境では実行されません。

A.3.2. インストール前キックスター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/bin/python
-- Python script omitted --
%end
```

--interpreter=

Python などの異なる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/bin/python** になります。

--erroronfail

スクリプトが失敗した場合にエラーを表示し、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。

--log=

スクリプトの出力を、指定したログファイルにログします。以下に例を示します。

```
%pre --log=/mnt/sysimage/root/ks-pre.log
```

A.3.3. 例: インストール前スクリプトで動的にパーティションスキームを書き出し

このインストール前スクリプトは、システム内のハードドライブ数を判定して、ドライブ数に応じて異なるパーティション設定スキームでテキストファイルを書き込みます。

```
%pre
#!/bin/sh
hds=""
mymedia=""
for file in /proc/ide/h* do
mymedia=`cat $file/media`
if [ $mymedia == "disk" ] ; then
hds="$hds `basename $file`"
fi
done
set $hds
numhd=`echo $#`
drive1=`echo $hds | cut -d' ' -f1`
drive2=`echo $hds | cut -d' ' -f2`

#Write out partition scheme based on whether there are 1 or 2 hard drives
if [ $numhd == "2" ] ; then
#2 drives
echo "#partitioning scheme generated in %pre for 2 drives" > /tmp/part-include
echo "clearpart --all" >> /tmp/part-include
echo "part /boot --fstype xfs --size 75 --ondisk hda" >> /tmp/part-include
echo "part / --fstype xfs --size 1 --grow --ondisk hda" >> /tmp/part-include
echo "part swap --recommended --ondisk $drive1" >> /tmp/part-include
echo "part /home --fstype xfs --size 1 --grow --ondisk hdb" >> /tmp/part-include
else
#1 drive
echo "#partitioning scheme generated in %pre for 1 drive" > /tmp/part-include
echo "clearpart --all" >> /tmp/part-include
echo "part /boot --fstype xfs --size 75" >> /tmp/part-include
echo "part swap --recommended" >> /tmp/part-include
echo "part / --fstype xfs --size 2048" >> /tmp/part-include
```

```
echo "part /home --fstype xfs --size 2048 --grow" >> /tmp/part-include
fi
%end
```

キックスタートファイルに分割コマンドのセットを追加する代わりに、以下の行を追加します。

```
%include /tmp/part-include
```

スクリプト内で選択されたパーティション設定コマンドが使用されるようになります。

A.4. キックスタートにおけるインストール後のスクリプト

インストール後のスクリプトは、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します。このセクションを使用して、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。

A.4.1. インストール後のスクリプトセクション

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、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末尾に配置し、**%post** で開始し、**%end** で終了する必要があります。キックスタートファイルに **%pre** セクションも含まれる場合は、**%pre** セクションと **%post** セクションの順番は重要ではありません。

このセクションは、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より便利です。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、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、以下に記載のように **--nochroot** オプションを使用することで変更できます。

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。詳細は**『ファイルシステムの設定および管理』** の「chroot 環境における systemctl の挙動」を参照してください。

重要

ネームサーバーを含めて、ネットワークを静的 IP 情報で設定した場合は、ネットワークにアクセスして、**%post** セクション内で IP アドレスを解決できます。ネットワークを DHCP 用に設定した場合、インストールが **%post** セクションを実行する時点では、**/etc/resolv.conf** は完了していません。ネットワークにアクセスすることはできますが、IP アドレスは解決できません。このため、DHCP を使用する場合は、**%post** セクションに IP アドレスを指定する必要があります。

A.4.2. インストール後キックスタートセクションのオプション

以下のオプションを使ってインストール後のスクリプトの動作を変更することができます。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/bin/python
-- Python script omitted --
%end
```

--interpreter=

Python などの異なる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/bin/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/bin/python** になります。

--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/sysimage/etc/resolv.conf
%end
```

--erroronfail

スクリプトが失敗した場合にエラーを表示し、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。

--log=

スクリプトの出力を、指定したログファイルにログ記録します。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** なしの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/sysimage/root/ks-post.log
```

A.4.3. 例: インストール後スクリプトで NFS のマウント

上記の **%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** という名前の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
```

A.4.4. 例: 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
/usr/sbin/subscription-manager register --username=admin@example.com --password=secret --
serverurl=sam-server.example.com --org="Admin Group" --environment="Dev" --
servicelevel=standard --release="7.0"
%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、Subscription Asset Manager、CloudForms System Engine など) 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アタッチしたり割り当てたりする場合にも使用できます。Subscription Asset Manager または CloudForms System Engine に登録する場合は、ローカルの管理者が作成したユーザーアカウントを使用します。

登録コマンドで追加オプションを使用してシステムに適したサービスレベルを設定し、また特定のオペレーティングシステムのバージョンに対する更新やエラータを制限できます。

キックスタートファイルの **%post** セクションの **subscription-manager** を使用方法は、Red Hat カスタマーポータルの記事 How do I use subscription-manager in a kickstart file? を参照してください。

A.5. 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ー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A.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 文字以上である必要があり、この要件に一致しないものは厳密に禁止されます。

A.6. 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

スクリプトが失敗した場合にエラーを表示し、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。

--interpreter=

Python などの異なる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/bin/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/bin/python** になります。

--log=

スクリプトの出力を、指定したログファイルにログします。

A.7. 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** 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B キックスタートのコマンドとオプションの参照

これは、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です。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。

B.1. RHEL 8 で変更したキックスタートのコマンドとオプション

次の一覧は、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。

B.1.1. RHEL 8 で非推奨になったキックスタートのコマンドとオプション

以下のキックスタートコマンドとオプションは、Red Hat Enterprise Linux 8 で完全に削除されたため、キックスタートで使用するとログに警告が出力されます。

- **auth** または **authconfig** - **authselect instead** を使用します。
- **deviceprobe**
- **install** - サブコマンド、またはメソッドを直接コマンドをして使用します。
- **mouse**
- **lilo**
- **lilocheck**
- **bootloader --upgrade**
- **ignoredisk --interactive**
- **partition --active**

特定のオプションだけが表示されている場合は、基本コマンドと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。

B.1.2. RHEL 8 で削除されたキックスタートのコマンドとオプション

以下のキックスタートコマンドとオプションは、デフォルトでは、Red Hat Enterprise Linux 8 で完全に削除されたため、キックスタートで使用するとエラーが発生します。

- **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**

特定のオプションと値だけが表示されている場合は、基本コマンドとその他のオプションは引き続き利用でき、削除されていません。

B.1.3. RHEL 8 で追加されたキックスタートのコマンドとオプション

次のコマンドとオプションは、Red Hat Enterprise Linux 8 で追加されました。

RHEL 8.0

- **authselect**
- **module**

B.2. インストールプログラムの構成とフロー制御のためのキックスタートコマンド

この一覧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。

B.2.1. autostep

キックスタートコマンド **autostep** は任意です。このオプションを使用すると、すべてのウィンドウを省略せずに少しの間表示します。キックスタートインストールでは、通常、必要ない画面は表示されません。

オプション

- **--autoscreenshot** - インストール中のすべてのステップでスクリーンショットを撮り、インストール中はこれを **/tmp/anaconda-screenshots/** に保存します。インストールが完了すると、スクリーンショットは **/root/anaconda-screenshots** に保存されます。
  各スクリーンは、インストールプログラムが次のスクリーンに切り替える直前のショットを撮ります。必要なキックスタートオプションをすべて使用しておらず、インストールが自動的に開始しない場合は、自動的に設定されていないウィンドウに移動して、希望する設定を実行できるため、これは重要になります。**完了** をクリックして続行すると、指定した設定を含むウィンドウがキャプチャーされます。

備考

- このオプションは、パッケージのインストールを中断させることができるため、システムのデプロイ時には使用しないでください。

B.2.2. cdrom

キックスタートコマンド **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が、**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。

B.2.3. cmdline

キックスタートコマンド **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。このモードは、x3270 ターミナルと共に IBM Z システムで使用する場合に便利です。

備考

- 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場合は、いずれかを選択するように求められます。

B.2.4. driverdisk

キックスタートコマンド **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。

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上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してから、**driverdisk** コマンドを使用して検索すべきドライバーディスクとその場所を指定します。

構文

```
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)。

備考

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ができます。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のドライバーディスクを読み込みます。
2. このデバイスにラベルを設定します (**DD** など)。

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、dd.rpm は具体的な名前に置き換えます。LABEL には、**inst.repo** コマンドでサポートされている内容を使用して、ハードディスクドライブを指定します。

B.2.5. eula

キックスタートコマンド **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。詳細は、**『標準的な RHEL インストールの実行』**の「初期セットアップの完了」を参照してください。

オプション

- **--agreed** (必須) - EULA を受諾します。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。

B.2.6. firstboot

キックスタートコマンド **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。

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** - 起動時に、初期セットアップが再設定モードで開始されます。このモードでは、初期設定に加え、言語、マウス、キーボード、root パスワード、セキュリティーレベル、タイムゾーン、ネットワーク設定オプションなどを設定できます。

B.2.7. graphical

キックスタートコマンド **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。

備考

- 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場合は、いずれかを選択するように求められます。

B.2.8. halt

キックスタートコマンド **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

を使ったインストールで、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。

halt コマンドは **shutdown -h** コマンドと同じです。

他の完了方法は、**poweroff**、**reboot**、**shutdown** などのコマンドをご覧ください。

B.2.9. harddrive

キックスタートコマンド **harddrive** は任意で、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。ドライブには、インストールプログラムでマウント可能な以下のファイルシステムが含まれている必要があります (**ext2**、**ext3**、**ext4**、**vfat**、もしくは **xfs**)。

構文

```
harddrive
```

オプション

- **--biospart=** - BIOS パーティションを指定する場合に使用します (**82** など)。
- **--partition=** - 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
```

備考

- **harddrive** 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が、**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。

B.2.10. install

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。このメソッドは、別のコマンドで使用してください。

キックスタートコマンド **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** コマンドの次の行に、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。
- 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。

B.2.11. liveimg

キックスタートコマンド **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=** - **HTTP**、**HTTPS**、**FTP** など、インストール実行中に使用する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** を付けるオプションの引数です。
- **--noverifyssl** - **HTTPS** サーバーに接続の際に、SSL 確認を無効にします。

例

```
liveimg --url=file:///images/install/squashfs.img --
checksum=03825f567f17705100de3308a20354b4d81ac9d8bed4bb4692b2381045e56197 --
noverifyssl
```

備考

- 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は、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。

- **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が、**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。

- 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。

B.2.12. logging

キックスタートコマンド **logging** が任意です。インストール中に 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。

構文

```
logging [--host=host] [--port=port] [--level=debug|info|error|critical]
```

任意のオプション

- **--host=** - 指定リモートホストにログ情報を送信します。ログを受けとるには、リモートホストで syslogd プロセスの実行を設定しておく必要があります。

- **--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。

B.2.13. mediacheck

キックスタートコマンド **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 (**rd.live.check**)。インストール時の介入が必要とされるため、デフォルトでは無効になっています。

B.2.14. nfs

キックスタートコマンド **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
```

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が、**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。

B.2.15. poweroff

キックスタートコマンド **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。キックスタートを使ったインストールでは、完了の方法が指定されていないと、デフォルトで **halt** オプションが使用されます。

poweroff オプションは **shutdown -p** コマンドと同じです。

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。

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。

B.2.16. reboot

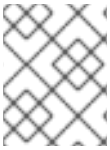
キックスタートコマンド **reboot** は任意です。インストールが正常に完了したらシステムを再起動するように、インストールプログラムに指示します (引数なし)。通常、キックスタートは、メッセージを表示し、ユーザーがキーを押してから再起動します。

reboot オプションは **shutdown -r** コマンドと同じです。

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。

その他の完了方法は、**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。



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**。

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せずに、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

kexec を使用したシステムブートでは、その複雑性のために明示的にテストすることができず、すべての状況で機能することが保証されるものではありません。

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。

B.2.17. rescue

キックスタートコマンド **rescue** は任意です。自動的にインストールプログラムのレスキューモードに入ります。問題が発生している場合は、これによりシステムを修復することができるようになります。

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。

B.2.18. shutdown

キックスタートコマンド **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。キックスタートを使ったインストールで、完了方法が指定されない場合は、**halt** コマンドが使用されます。

shutdown キックスタートオプションは、**shutdown** コマンドと同じです。

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。

B.2.19. sshpw

キックスタートコマンド **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、ログオンに使用する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へ転送されません。

構文

```
sshpw --username=name password [--iscrypted|--plaintext] [--lock]
```

必須オプション

- **--username** - 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** - このオプションを含めると、パスワード引数は既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** - このオプションを含めると、パスワード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** - このオプションを含めると、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。

備考

- デフォルトでは、インストール時に **ssh** サーバーは起動されません。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させます。
- 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 --lock
  ```

B.2.20. text

キックスタートコマンド **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。

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場合は、いずれかを選択するように求められます。

B.2.21. unsupported_hardware

キックスタートコマンド **unsupported_hardware** は任意です。インストールプログラムに、**Unsupported Hardware Detected** 警告を表示しないように指示します。

このコマンドが含まれておらず、サポートされていないハードウェアが検出された場合は、この警告によりインストールが止まります。

B.2.22. url

キックスタートコマンド **url** は任意です。FTP、HTTP、または HTTPS 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を実行します。

構文

```
url --url=FROM [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

- **--mirrorlist=** - インストール元となるミラー URL です。
- **--proxy=** - **HTTP**、**HTTPS**、**FTP** など、インストール実行中に使用するプロキシーを指定します。
- **--noverifyssl** - **HTTPS** サーバーに接続の際に、SSL 確認を無効にします。

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
```

備考

- **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が、**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。

B.2.23. vnc

キックスタートコマンド **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、この方法が通常はテキストモードよりも好まれます。追加のオプション指定がない場合は、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。

構文

```
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提供が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。

B.2.24. %include

キックスタートコマンド **%include** は任意です。

%include /path/to/file 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含めます。

B.3. システム設定用キックスタートコマンド

この一覧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。

B.3.1. auth または authconfig (非推奨)

重要

キックスタートコマンド **auth** または **authconfig** は、**authconfig** ツールとともに、Red Hat Enterprise Linux 8 では非推奨となりました。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** コマンド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、man ページの **authselect-migration(7)**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。

キックスタートコマンド **auth** または **authconfig** は任意です。**authconfig** 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。

構文

```
authconfig [options]
```

備考

- このコマンドは、すべてのオプションを **authconfig** コマンドに渡します。詳細は、man ページの authconfig、および **authconfig --help** コマンドを参照してください。
- 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 (CVE-2014-3566) 脆弱性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。

B.3.2. authselect

キックスタートコマンド **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。

構文

```
authselect [options]
```

備考

- 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、man ページの **authselect(8)**、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 (CVE-2014-3566) 脆弱性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。

B.3.3. firewall

キックスタートコマンド **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。

構文

```
firewall --enabled|--disabled device [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。

- device - ファイアウォールを設定するデバイス

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。複数のデバイスを指定する場合は、**--trust em1 --trust em2** を使用します。**--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。

- 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、高レベルでサービスのファイアウォール通過を許可する方法です。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に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** 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一覧が表示されます。

B.3.4. group

キックスタートコマンド **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=** - グループ名を与えます。

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。

備考

- 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ために使用できます。

B.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。

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/** ディレクトリー内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一覧で指定します。**setxkbmap(1)** と同じ形式 (**layout** 形式 (**cz** など)、または **layout (variant)** 形式 (**cz (qwerty)** など) のいずれかの形式) の値をとります。
  使用できるレイアウトは、man ページ **xkeyboard-config(7)** の **Layouts** を参照してください。
- **--switch=** - レイアウト切り替えのオプション一覧を指定します (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、man ページの **xkeyboard-config(7)** の **Options** をご覧ください。

備考

- **--vckeymap=** または **--xlayouts=** 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
```

B.3.6. lang (必須)

キックスタートコマンド **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。

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
  ```

備考

- **/usr/share/system-config-language/locale-list** は system-config-language パッケージの一部になります。使用できる言語コードはこのファイルの各行 1 番目のコラムを参照してください。

- 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** コマンドでこれら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
```

B.3.7. module

キックスタートコマンド **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モジュールストリームを有効にします。

構文

```
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。

備考

- 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一覧を追加することで、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。

- 単一の **module** コマンドは、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を一覧表示するには、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。

関連資料

- モジュールおよびストリームの詳細は、『ユーザー領域コンポーネントのインストール、管理、および削除』 を参照してください。

B.3.8. pwpolicy

キックスタートコマンド **pwpolicy** は任意です。このコマンドは、カスタムのパスワードポリシーを強制します。このポリシーは、パスワードの長さや強度などの要素に基づいて、インストール中に作成されるパスワードの要件を指定するものです。

構文

```
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** - **完了** を 2 回クリックすると、**--minquality=** と **-minlen=**で指定された要件を **満たさない**パスワードが許可されます。
- **--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ー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。

備考

- このコマンドは、**%anaconda** セクション内でのみ使用可能になります。
- **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** の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、man ページの **pwscore(1)** および **pwmake(1)** を参照してください。

B.3.9. repo

キックスタートコマンド **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。

構文

```
repo --name=repoid [--baseurl=<url>|--mirrorlist=url] [options]
```

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一覧が使用されるため、この一覧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。
- **--baseurl=** - リポジトリーの URL を入力します。yum のリポジトリー設定ファイル内で使用可能な変数には対応していません。同一リポジトリーの定義内では、このオプションは **--mirrorlist** オプションと一緒に使用することはできません。
- **--mirrorlist=** - リポジトリーのミラーの一覧を指す URL を入力します。ここでは、yum のリポジトリー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。このオプションと **--baseurl** オプションを同一リポジトリー定義内で使用することはできません。
- **--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。小さい数値の方が優先順位が高くなります。
- **--excludepkgs=** - このリポジトリーからは読み出しては**ならない**パッケージ名の一覧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したい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プルする必要のあるパッケージ名およびグロブの一覧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、このリポジトリーからプルしたい場合に便利なオプションです。
- **--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。
- **--ignoregroups=true** - インストールツリーの構成時に使用されるオプションです。インストールプロセス自体には影響しません。不要な大量のデータをミラーリングしないように、ツリーのミラーリングを行う際にパッケージグループの情報を検索しないように、構成用ツールに指示します。
- **--noverifyssl** - **HTTPS** サーバーに接続の際に、SSL 確認を無効にします。

備考

- 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。

B.3.10. rootpw (必須)

キックスタートコマンド **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
```

オプション

- **--iscrypted** - このオプションを含めると、パスワード引数は既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** - このオプションを含めると、パスワード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** - このオプションを含めると、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、グラフィカルとテキストベースの両方の手動インストールにおいて、**Root Password** ウィンドウが無効になります。

B.3.11. selinux

キックスタートコマンド **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 targeted 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発します。ただし、実際にはポリシーは実施されません。

- **--disabled** - SELinux を完全に無効にします。

関連資料

SELinux の詳細は『SELinux の使用』を参照してください。

B.3.12. services

キックスタートコマンド **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 (disabled) サービスの一覧が処理されてから、有効 (enabled) サービスの一覧が処理されます。したがって、両方の一覧に記載されているサービスは有効になります。

構文

```
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。

備考

*サービスを指定時には空白を入れないでください。空白があると、キックスタート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しません。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
```

B.3.13. skipx

キックスタートコマンド **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。

B.3.14. 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 **timezone** に設定します。

構文

```
timezone timezone [options]
```

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みなします。
- **--nontp** - NTP サービスの自動スタートを無効にします。
- **--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。

備考

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ースには、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。

B.3.15. user

キックスタートコマンド **user** は任意です。システムに新しいユーザーを作成します。

構文

```
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 情報を提供します。これは、コンマ区切りの様々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、man ページの **passwd(5)** 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** - このオプションを使用すると、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、グラフィカルとテキストベースの両方の手動インストールにおいて、**Create User** 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。提供されない場合は、そのアカウントがデフォルトでロックされます。
- **--iscrypted** - このオプションを含めると、パスワード引数は既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** - このオプションを含めると、パスワード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。提供されない場合は、システムデフォルトが使用されます。
- **--uid=** - ユーザーの UID (User ID) です。提供されない場合は、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。このオプションを使用しないと、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。

備考

- **--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** の範囲は将来広がり、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 の範囲は、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 と GID の最小制限を変更するには、**『基本的なシステム設定の構成』**の「umask を使用した、新規ファイルのデフォルト権限の設定」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、アプリケーションは、新規作成ファイルへ特定パーミッションを付与しないよう、**user file-creation mask** 設定で指定されます。
  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。詳細情報は**『基本的なシステム設定の構成』**の「umask を使用した、新規ファイルのデフォルト権限の設定」を参照してください。

B.3.16. xconfig

キックスタートコマンド **xconfig** は任意です。X Window System を設定します。

オプション

- **--defaultdesktop=** - **GNOME** または **KDE** を指定してデフォルトのデスクトップを設定します (GNOME Desktop Environment または KDE Desktop Environment のいずれかの選択された環境が **%packages** セクションにインストールされていることが前提)。

重要

現時点では、このオプションを使用して KDE をデフォルトのデスクトップ環境に指定することはできません。これは既知の問題です。回避策は 「How to configure default desktop session in RHEL7 ?」 を参照してください。この回避策は、キックスタートのインストール後のスクリプトに使用できます。

- **--startxonboot** - インストールされたシステムでグラフィカルログインを使用します。

備考

- **xconfig** コマンドを含まないキックスタートファイルで X Window System システムをインストールする場合は、インストール時に手動で X を設定する必要があります。X Window System をインストールしないキックスタートファイルでコマンドを使用しません。

B.4. ネットワーク設定用キックスタートコマンド

この一覧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。

B.4.1. auth または authconfig (非推奨)

重要

キックスタートコマンド **auth** または **authconfig** は、**authconfig** ツールとともに、Red Hat Enterprise Linux 8 では非推奨となりました。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** コマンド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、man ページの **authselect-migration(7)**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。

キックスタートコマンド **auth** または **authconfig** は任意です。**authconfig** 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。

構文

```
authconfig [options]
```

備考

- このコマンドは、すべてのオプションを **authconfig** コマンドに渡します。詳細は、man ページの authconfig、および **authconfig --help** コマンドを参照してください。
- 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 (CVE-2014-3566) 脆弱性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。

B.4.2. device

キックスタートコマンド **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 カードは自動検出されますが、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与える必要があります。追加モジュールのインストールをインストールプログラムに指示する **device** コマンドは、以下のような形式になります。

構文

```
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"
```

B.4.3. network

キックスタートコマンド **network** は任意です。インストールするシステムに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ベートします。

1つ目の **network** コマンドで指定しているデバイスが、自動的に起動します。また、デバイスの起動は、**--activate** オプションでの明示的な指定が必要な場合もあります。

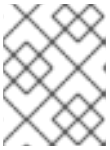
オプション

- **--activate** - インストール環境でこのデバイスを起動させます。
  起動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に起動オプションで設定したインターフェースなど)、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にデバイスが再起動します。

  デバイスにデフォルトのルートを使用させないようにする場合は **--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化します。**--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。**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構成を取得します。BOOTP メソッドも同様で、BOOTP サーバーがネットワーク構成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構成を取得する場合は、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で IP アドレスとネットマスクをキックスタートファイルで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、インストール中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構成情報はすべて**一行で**指定する必要があります。コマンドライン上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。
  1番目 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 の起動オプションを指定していればその値が使用されます。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内に記載される 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不明になってしまいます。2 番目およびそれ以降の **network** コマンドには、必ずこのオプションを指定してください。

  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。

  - インターフェースのデバイス名を使用して指定する (**em1** など)
  - インターフェースの MAC アドレスを使用して指定する (**01:23:45:67:89:ab** など)
  - **link** キーワードを使用して指定する (リンクが **up** 状態になっている 1 番目のインターフェース)
  - キーワード **bootif**。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** 変数を設定するには、**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 を省略した場合は 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** - インターフェー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 ターゲット用に用意した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起動させる場合は、このオプションを使用してください。

- **--nameserver=** - IP アドレスに DNS ネームサーバーを指定します。複数の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。

- **--nodns** - DNS サーバーを設定しません。

- **--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=** - インストールシステムのホスト名。ホスト名は、**host_name.domainname** の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は、接続システムにドメイン名を提供する Dynamic Host Configuration Protocol (DHCP) サービスが備わっています。DHCP によるドメインの割り当てを許可する場合は、短縮ホスト名のみを指定してください。

重要

ネットワークが DHCP サービスを提供**しない**場合は、システムのホスト名に FQDN を必ず使用してください。

- **--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。

- **--essid=** - ワイヤレスネットワークのネットワーク ID を指定します。

- **--wepkey=** - ワイヤレスネットワークの WEP 暗号化キーを指定します。

- **--wpakey=** - ワイヤレスネットワークの WPA 暗号化キーを指定します。

- **--onboot=** - 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** - このデバイス上で IPv4 を無効にします。

- **--noipv6** - このデバイス上で IPv6 を無効にします。

- **--bondslaves=** - このオプションを使用する場合、**--bondslaves=** オプションで定義されたスレーブを使用して **--device=** オプションで指定したネットワークデバイスが作成されます。以下に例を示します。

```
network --device=mynetwork --bondslaves=em1,em2
```

上記のコマンドは、**em1** および **em2** インターフェースをスレーブとして使用し、**mynetwork** という名前のボンドデバイスを作成します。

- **--bondopts=** - **--bondslaves=** と **--device=** のオプションを使用して指定される結合インターフェース用のオプションパラメーターの一覧です。この一覧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。

- **--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ース名を指定します。**--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** デバイス上に **171** という ID を持つ **vlan171** という名前の仮想 LAN インターフェースが作成されます。

インターフェースには任意の名前を付けることができますが (**my-vlan** など)、場合によって次の命名規則に従う必要があります。

  - 名前にドット (**.**) が含まれている場合は、**NAME.ID** の形にする必要があります。**NAME** は任意ですが、**ID** は VLAN ID である必要があります。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スレーブを使用して、**--device=** オプションで指定したチームデバイスを作成します。スレーブとスレーブの間はコンマで区切ってください。各スレーブの後ろにその設定を指定できます。二重引用符と \ 記号でエスケープした JSON 文字列を一重引用符で囲っている部分が設定になります。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。二重引用符と \ 記号でエスケープした JSON 文字列を一重引用符で囲っている部分が実際の設定になります。デバイス名は **--device=** オプションで、使用するスレーブとその設定は **--teamslaves=** オプションでそれぞれ指定します。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ース用パラメーターの一覧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。パラメーターの詳細は、man ページ **nm-settings(5)** の **bridge setting** テーブル、または https://developer.gnome.org/NetworkManager/0.9/ref-settings.html を参照してください。
また、ネットワークブリッジの一般情報は、『セキュリティーの設定および管理』を参照してください。

- **--bindto=mac** - インストールするシステムのデバイス設定 (**ifcfg**) ファイルはデバイスのインターフェース名 (**DEVICE**) にバインドされますが、これを 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。**--bindto=mac** は、同じ **network** コマンドでデバイス名、または **link**、**bootif** が指定されていても、**--bindto=mac** が適用されます。

備考

- **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標準にはいくつかのタイプがあります。このような標準仕様の詳細は『ネットワークの設定および管理』 を参照してください。

- 命名スキームへの変更により、**eth0** などの **ethN** デバイス名が Red Hat Enterprise Linux 8 で利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。

B.4.4. realm

キックスタートコマンド **realm** は任意です。Active Directory や IPA ドメインをジョインさせます。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。

構文

```
realm join [options] domain
```

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、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 の部分は省略しても構いません。

- **--no-password** - パスワードの入力なしで自動的にジョインします。

- **--one-time-password=** - ワンタイムパスワードを使用してジョインします。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ジョインしません。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ジョインしません。使用できる値は **active-directory** や **freeipa** などになります。
- **--membership-software=** - レルムにジョインする際に、ここに指定した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。

B.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。

B.5.1. device

キックスタートコマンド **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 カードは自動検出されますが、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与える必要があります。追加モジュールのインストールをインストールプログラムに指示する **device** コマンドは、以下のような形式になります。

構文

```
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"
  ```

B.5.2. autopart

キックスタートコマンド **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。十分な容量を持つドライブ (50 GB 以上) の場合は、**/home** パーティションも作成されます。

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

使用可能なパーティションスキームについての説明は、「対応デバイスの種類」を参照してください。

- **--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。これらのファイルシステムに関する詳細は「対応ファイルシステム」を参照してください。
- **--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。
- **--encrypted** - 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 つ以上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 (ランダムなデータ) 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 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相互作用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=** - 暗号化した全デバイスのシステムワイドなデフォルトの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キーを **/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。キー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** と併用しないと意味がありません。
- **--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与えます。パスフレーズは **/root** 配下に別々のファイルで格納されます。**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと意味がありません。
- **--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。このオプションは、**--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。

備考

- **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。

- CMS タイプの単一 FBA DASD 上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せず、インストールが問題なく進みます。

B.5.3. bootloader (必須)

キックスタートコマンド **bootloader** が必要です。ブートローダーがインストールされる方法を指定します。

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"
```

rhgb と **quiet** のパラメーターは、ここで特に指定していなくても、また **--append=** コマンド自体をまったく使用していない場合であっても、必ず使用されます。

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) これらのメカニズムは不要な場合があり、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します。これらのメカニズムを無効にするには、そのための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には、どのようなメカニズムであれ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 に起因する脆弱性は、「Meltdown & Spectre - Kernel Side-Channel Attacks - CVE-2017-5754 CVE-2017-5753 CVE-2017-5715」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、デバイスのメンバーを 1 つだけ指定します。

重要

現在、**zipl** ブートローダーを使用する 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、そこ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 Red Hat Enterprise Linux 8 をブートローダー内のインストール済みシステム一覧の最上位に追加し、その順番と既存の全エントリーを保持します。

- **--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) スキームを 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。
- **--password=** - GRUB2 を使用した場合、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 **bootloader** のエントリー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 ではなく、extlinux ブートローダーを使用します。このオプションが適切に動作するには、**extlinux** が対応しているシステムのみです。
- **--disabled** - これは **--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ので、スペースが節約できます。

備考

- Red Hat で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ています。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。

- 64 ビット AMD、Intel、および ARM 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る場合があります。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って異なります。詳細は、**『標準的な RHEL インストールの実行』**の「ブートローダーの設定」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的に参照する別の方法は、**『ストレージデバイスの管理』**の「永続的な命名属性の概要」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。

B.5.4. clearpart

キックスタートコマンド **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。

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はすべて消去されます。使用する場合は注意してください。

  clearpart に **--drives=** オプションを使用して消去したい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ブラック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 コントローラーにある 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 が **58095BEC5510947BE8C0360F604351918** のディスクを消去する場合は次のようにします。

```
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 が **2416CD96995134CA5D787F00A5AA11017** のディスクを消去する場合は次のようにします。

```
clearpart --drives=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
```

マルチパスのデバイスを消去する場合は、**mpatha** などのデバイス名は絶対に使用しないでください。このようなデバイス名は特定ディスクに固有な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が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** - フォーマット対象の全ディスクで、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 アーキテクチャーのデフォルトディスクラベルは msdos になります。**--initlabel** によりすべてのディスクが処理されてしまうので、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。

  ```
  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** のオプションは無効になります。異なるドライブをまたいで使用できます。以下に例を示します。

  ```
  clearpart --list=sda2,sda3,sdb1
  ```

- **--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。

備考

キックスタートファイルで **clearpart --all** コマンドを使用して、インストール時に既存の全パーティションを削除しようとすると、場合によっては Anaconda によりインストールが一時中断し、削除の確認が求められることがあります。人が介入せずに自動的にインストールを行う必要がある場合は **zerombr** コマンドをキックスタートファイルに追加します。

- **sdX** (または **/dev/sdX**) 形式でのデバイス名が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的に参照する別の方法は、**『ストレージデバイスの管理』**の「永続的な命名属性の概要」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。

B.5.5. fcoe

キックスタートコマンド **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。

構文

```
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。

B.5.6. ignoredisk

キックスタートコマンド **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。

構文

```
ignoredisk --drives=drive1,drive2,...
```

必須オプション

- **--drives=driveN,...** - driveN を、**sda**、**sdb**、**hda** などに置き換えます。

任意のオプション

- **--only-use** - インストールプログラムで使用するディスクの一覧を指定します。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、以下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、以下のコマンドを実行します。

```
ignoredisk --only-use=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
```

備考

- **--interactive** は Red Hat Enterprise Linux 8 で非推奨になります。このオプションにより、ユーザーは、高度なストレージ画面に手動で移動できました。
- 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 が **2416CD96995134CA5D787F00A5AA11017** のディスクを消去する場合は以下を使用します。

```
ignoredisk --drives=disk/by-id/dm-uuid-mpath-2416CD96995134CA5D787F00A5AA11017
```

- Anaconda がキックスタートファイルを解析するまで、LVM を使用するマルチパスのデバイスは構成されません。したがって、これらのデバイスは **dm-uuid-mpath** の形式では指定できません。LVM を使用するマルチパスのデバイスを無視する場合は、代わりに **disk/by-id/scsi-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 が **58095BEC5510947BE8C0360F604351918** のディスクを無視するには、以下のコマンドを使用します。

```
ignoredisk --drives=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
```

- マルチパスのデバイスを消去する場合は、**mpatha** などのデバイス名は絶対に使用しないでください。このようなデバイス名は特定ディスクに固有な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が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的に参照する別の方法は、**『ストレージデバイスの管理』**の「永続的な命名属性の概要」を参照してください。

B.5.7. iscsi

キックスタートコマンド **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。

構文

```
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。
- **--port=** (必須) - ポート番号を指定します (通常は**--port=3260**)。

任意のオプション

- **--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ースではなく、指定ネットワークインターフェースに接続を結合します。これを一度使用したら、キックスタートには、**iscsi** コマンドのインスタンスを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。

備考

- 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、**iscsi** コマンドより先に指定してください。
- iSCSI ストレージの設定は、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) 内で行うことが推奨されます。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、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、**clearpart** や **ignoredisk** などのコマンドによる参照より**先にまず iscsi** コマンドがキックスタート内で指定されていることを確認してください。

B.5.8. iscsiname

キックスタートコマンド **iscsiname** は任意です。これは、iscsi パラメーターが指定した iSCSI ノードへに名前を割り当てます。以下は構文になります。

```
iscsiname iqn
```

備考

- キックスタートファイルで **iscsi** パラメーターを使用する場合は、**その前に iscsiname** を指定する必要があります。

B.5.9. logvol

キックスタートコマンド **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。

構文

```
logvol mntpoint --vgname=name --name=name [options]
```

必須オプション

- **mntpoint** は、パーティションをマウントする場所で、次のいずれかの形式になります。
  - **/path**
    たとえば /、または /**home**。
  - **swap**
    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ながら、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
    ```

    --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。

    64 ビット AMD、Intel、および 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、そのボリュームの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、そのボリュームを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。
- **--mkfsoptions=** - このパーティションに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。引数のリストではなく、mkfs プログラムに直接渡すことが可能な形式で指定する必要があります。つまり、オプションが複数になった場合は、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。指定方法はファイルシステムによって異なります。

- **--label=** - 論理ボリュームのラベルを設定します。
- **--grow** - このオプションは、利用可能な最大サイズまでパーティションを拡張するか、指定したサイズまで拡張するように設定します。**--percent=** または **--size=** のいずれかのオプションを使用して、最小サイズを指定する必要があります。
- **--size=** - 論理ボリュームの最小サイズを 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=** - 静的にサイズ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

新規の論理ボリューム作成時には、**--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スペース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=** - grow が設定された場合の最大サイズを MiB 単位で指定します。**500** など整数値を入力してください (単位は不要)。
- **--recommended** - **swap** 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。64 ビット AMD、Intel、および ARM の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して暗号化します。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 つ以上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 (ランダムなデータ) 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 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相互作用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時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。単独では意味がありません。
- **--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。このオプションは、**--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キーを **/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。キー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** と併用しないと意味がありません。

- **--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与えます。パスフレーズは **/root** 配下に別々のファイルで格納されます。**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと意味がありません。

- **--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を使用)。

- **--metadatasize=size** - 新しいシンプールデバイスのメタデータ領域サイズを指定します (MiB 単位)。

- **--chunksize=size** - 新しいシンプールデバイスのチャンクサイズを指定します (KiB 単位)。

- **--thin** - シン論理ボリュームを作成します (**--poolname** と併用する必要があります)。

- **--poolname=name** - シン論理ボリュームを作成するシンプール名を指定します。**--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ファイル名を指定します。これを使用すると、この名前は当該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 man ページの **lvm(8)** 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れらのモードの詳細は、man ページの **lvmcache(7)** 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。

備考

- 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は、ダッシュ (「**-**」) 文字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。

例

- まず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から、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
```

- まず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から、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
```

関連資料

- LVM の詳細は『論理ボリュームの設定および管理』を参照してください。

B.5.10. mount

キックスタートコマンド **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 [--reformat [REFORMAT]] [--mkfsoptions MKFS_OPTS] [--mountoptions MOUNT_OPTS]
device mntpoint
```

必須オプション:

- **device** - マウントするブロックデバイス。
- **mntpoint** - **device** をマウントする場所。/、/**usr** 等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フォーマット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る)。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 man ページの **mount(8)** を、概要は **fstab(5)** を参照してください。

備考

- 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、すべてのストレージ設定をキックスタートファイルで記述する必要はありません。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** 等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。

B.5.11. nvdimm

キックスタートコマンド **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。

構文

```
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構成します。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。このアクションのフォーマットは以下のとおりです。

  ```
  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 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。このアクションのフォーマットは以下のとおりです。

  ```
  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*
```

備考

- 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。これら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。

B.5.12. part または partition (必須)

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。

構文

```
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。次のいずれかの形式になります。
  - **/path**
    /、**/usr**、**/home** など。
  - **swap**
    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には、**--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。

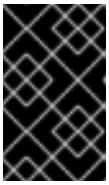
    64 ビット AMD、Intel、および ARM のシステムで、このコマンドが割り当てたスワップサイズは「推奨されるパーティション設定スキーム」を参照してください。
  - **raid.id**
    このパーティションはソフトウェア RAID に使用されます (**raid** を参照)。

- **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 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 AMD、Intel、および 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** 値は、必要となる領域の最小値として設定します。推奨は 「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** - これを指定すると、利用可能な最大サイズまでパーティションを拡張する、または指定限度サイズまで拡張するように指示します。

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 GB 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 GB 以上のシステムの場合は、物理メモリー量に 2GB 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。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、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してい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。

- **--ondisk=** または **--ondrive=** - 特定のディスク上にパーティションの作成を強制します。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) を使用してい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 が **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
```

  LVM を使用するマルチパスデバイスは、Anaconda がキックスタートファイルの解析を完了するまで構成されません。したがって、このようなデバイスは、**dm-uuid-mpath** の形式では指定しないため、代わりに LVM を使用するマルチパスデバイスを指定するには、**disk/by-id/scsi-WWID** の形式を使用します。**WWID** は、デバイスの world-wide identifier です。WWID が **58095BEC5510947BE8C0360F604351918** のディスクを指定するには、以下のコマンドを使用します。

```
part / --fstype=xfs --grow --asprimary --size=8192 --ondisk=disk/by-id/scsi-58095BEC5510947BE8C0360F604351918
```

警告

マルチパスのデバイスを消去する場合は、**mpatha** などのデバイス名は絶対に使用しないでください。このようなデバイス名は特定ディスクに固有な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が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。

- **--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意味があり、GUID Partition Table (GPT) ラベルが付いたディスクでは意味がありません。

- **--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使用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様々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上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、すべてのファイルシステムで機能するものです。ただし、プロファイルの概念には対応していません。引数のリストでは処理が行われないので、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、ファイルシステムによって、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。

- **--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。

- **--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。64 ビット AMD、Intel、および 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** スペース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上に強制的にパーティションを作成します。

- **--encrypted** - このパーティションを **--passphrase** オプションで入力したパスフレーズを使用して暗号化します。このパスフレーズを指定していない場合、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 つ以上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 (ランダムなデータ) 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 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相互作用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。単独では機能しません。

- **--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。このオプションは、**--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キーを **/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。キー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** と併用しないと意味がありません。

- **--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与えます。パスフレーズは **/root** 配下に別々のファイルで格納されます。**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと意味がありません。

- **--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) と **--onpart=** オプションで目的のパーティションを指定します。

備考

- **--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的に参照する別の方法は、**『ストレージデバイスの管理』**の「永続的な命名属性の概要」を参照してください。

B.5.13. raid

キックスタートコマンド **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。

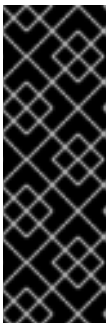
構文

```
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。

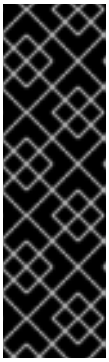


重要

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** になっているかを確認してください。

デフォルトの Red Hat Enterprise Linux 7 の **mdadm** メタデータバージョンは、ブートデバイスではサポートされていません。

- **--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**)。



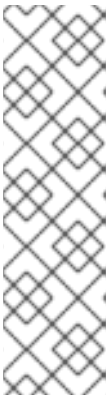
重要

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旧アレイ (v0.90 メタデータ) を所有している場合には、ファイルシステムのラベルまたは UUID でアレイを指定できます (**--device=rhel7-root --label=rhel7-root** など)。

- **--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、デフォルトサイズ (**512 Kib**) ではない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のドライブ数を指定します。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構成に使用されます。
- **--fsprofile=** - このパーティション上でファイルシステムを作成するプログラムに渡す使用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様々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。使用できる値は、**xfs**、**ext2**、**ext3**、**ext4**、**swap**、**vfat** 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。
- **--mkfsoptions=** - このパーティションに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。引数のリストではなく、mkfs プログラムに直接渡すことが可能な形式で指定する必要があります。つまり、オプションが複数になった場合は、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。指定方法は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既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化はしません。
- **--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** - この RAID デバイスを **--passphrase** オプションで入力したパスフレーズを使用して暗号化します。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 つ以上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 (ランダムなデータ) 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 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相互作用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。

- **--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。このオプションは、**--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。単独では機能しません。
- **--escrowcert=URL_of_X.509_certificate** - このデバイス用のデータ暗号化キーを **/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。**--encrypted** と併用しないと意味がありません。
- **--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与えます。パスフレーズは **/root** 配下にファイルとして格納されます。**--escrowcert** で指定した X.509 証明書を使用して暗号化されます。**--escrowcert** と併用しないと意味がありません。

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 つあり、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=rhel7-root --label=rhel7-root raid.01 raid.02 raid.03
raid /home --level=5 --device=rhel7-home --label=rhel7-home raid.11 raid.12 raid.13
```

B.5.14. reqpart

キックスタートコマンド **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

- **/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。

備考

- 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、/ および **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を作成するだけで、ドライブの残りは空のままにするので、カスタムレイアウトの作成が可能になります。

B.5.15. snapshot

キックスタートコマンド **snapshot** は任意です。このコマンドを使用すると、インストール中にシン論理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、**snaphost** キックスタートコマンドを複数回追加します。

構文

```
snapshots 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である必要があります。
- **--when=pre-install|post-install** - インストール前もしくは後にスナップショットを作成することを指定します。

B.5.16. volgroup

キックスタートコマンド **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。

構文

```
volgroup name partition [options]
```

オプション

- **--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** を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)、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。

備考

- 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から、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 volgrp pv.01 ` [command]`logvol / --vgname=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は、ダッシュ (「**-**」) 文字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。

B.5.17. zerombr

キックスタートコマンド **zerombr** は任意です。**zerombr** が指定されると、ディスク上で検出された無効なパーティションテーブルが初期化されます。これにより無効なパーティションテーブルのあるディスクのコンテンツがすべて破棄されます。このコマンドは、既に初期化されたディスクのシステム上で無人インストールを実行する際に必要となります。

構文

```
zerombr
```

-

備考

- IBM Z では、**zerombr** を指定すると、インストールプログラムが認識している Direct Access Storage Device (DASD) で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されます。このコマンドは、対話型インストール中のユーザー選択も阻止します。
- **zerombr** が指定されていない場合に、未フォーマットの DASD をインストールプログラムが 1 つ以上認識している場合は、非対話形式のキックスタートを使用したインストールが失敗に終わります。
- **zerombr** が指定されていない場合に、未フォーマットの DASD をインストールプログラムが 1 つ以上認識している場合は、認識されている未フォーマットの DASD のフォーマットにユーザーがすべて同意しなければ、対話形式のインストールは終了します。この状況を避けるには、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。

B.5.18. zfcp

キックスタートコマンド **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、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。

構文

```
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) になります。
- **--wwpn** - デバイスのワールドワイドポートネーム (WWPN)。**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。

例

```
zfcp --devnum=0.0.4000 --wwpn=0x5005076300C213e9 --fcplun=0x5022000000000000
```

B.6. RHEL インストールプログラムに付属のアドオンのための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にデフォルトで提供されるアドオンに関連しています。

B.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ダンピングメカニズムを設定します。

構文

```
%addon com_redhat_kdump [OPTIONS] [command]%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。

備考

Kdump とは、システムのメモリー内容を保存して後で分析できるよう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。**kexec** に依存します。これは、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、1 番目のカーネルからはアクセスできない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 and configuring kdump」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が、『**Managing, monitoring and updating the kernel**』の「Memory requirements for kdump」セクションに記載の基準に基づいて自動メモリー量を決定します。

  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。

B.6.2. %addon org_fedora_oscap

キックスタートコマンド **%addon org_fedora_oscap** は任意です。

OpenSCAP インストールプログラムアドオンは、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、このアドオンはデフォルトで有効になっています。有効時には、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、インストール中およびインストール後にチェックは実行されません。

重要

セキュリティーポリシーの適用は必ずしもすべてのシステムで必要なわけではありません。このウィンドウは、所定のポリシーの適用が業務規定や法令で義務付けられている場合にのみ使用してください。

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は受け付けず、**%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。

キー

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達するネットワーク接続が必要になります。

- **datastream-id** - **content-url** の値で言及され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。

- **xccdf-path** - 使用する XCCDF ファイルへのパスを、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。

- **fingerprint** - **content-url** で言及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へのパスをアーカイブ内の相対パスで指定します。

例

- インストールメディア上の scap-security-guide から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B.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 = 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B.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
```

関連資料

- OpenSCAP インストールプログラムのアドオンの追加情報は https://www.open-scap.org/tools/oscap-anaconda-addon/ を参照してください。
- SCAP セキュリティーガイドで使用可能なプロファイルの詳細は「OpenSCAP Portal」を参照してください。

付録C ディスクのパーティション設定

C.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スワップ領域を含めることができます。標準パーティションは、**/boot** と、**BIOS Boot** パーティション、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 論理ボリュームを使用することが推奨されます。

LVM

デバイスタイプで **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。現在 LVM ボリュームが存在しない場合は、新しいボリュームを含む LVM ボリュームが自動的に作成されます。LVM ボリュームが作成される場合は、ボリュームが割り当てられます。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して、1 つのマウントポイントに対して複数の物理ディスクを使用するなどの高度な設定が可能になります。

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ます。これは、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。

C.2. 対応ファイルシステム

本セクションでは、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、より迅速なクラッシュ復元が容易になります。1 つの XFS ファイルシステムでサポートされる最大サイズは 500 TB です。XFS は、Red Hat Enterprise Linux でデフォルトで推奨されるファイルシステムです。

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、改善がいくつか加えられています。より大きなファイルシステム、より大きなファイルに対応するようになり、また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。1 つディレクトリーに含まれるサブディレクトリー数には制限がなく、ファイルシステムのチェックが高速化、またジャーナリング機能もさらに堅牢になっています。1 つの ext4 ファイルシステムでサポートされる最大サイズは 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。ジャーナリング機能を使用すると、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、クラッシュ後の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 ファイルシステム上の Microsoft Windows の長いファイル名と互換性があります。

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。

C.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、利用可能な領域を同じサイズだけ必要とすることで、信頼性が向上します。

本セクションは、LVM および LVM シンプロビジョニングとともに使用して、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。

なし

RAID アレイは設定されません。

RAID0

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 レベル 0 は、標準パーティションのパフォーマンスを向上させます。複数のディスクを 1 つの大きな仮想デバイスにまとめることができます。RAID レベル 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。

RAID1

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。アレイ内のディスク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。

RAID4

エラーチェック - データを複数のディスクに分散させますが、アレイ内の 1 つのディスクにパリティー情報を格納します。これにより、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は 1 つのディスクに格納されるため、このディスクへのアクセスにより、アレイのパフォーマンスに「ボトルネック」が発生します。レベル 4 の RAID には少なくとも 3 つの RAID パーティションが必要です。

RAID5

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 レベル 5 は複数ディスクにデータを分散させ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で分散されるため、RAID レベル 4 のようにパフォーマンスにボトルネックが発生しません。RAID 5 には少なくとも 3 つのディスクが必要です。

RAID6

冗長エラーチェック - RAID レベル 6 は RAID レベル 5 と似ていますが、パリティーデータが 1 セットではなく 2 セット格納されます。RAID 6 には少なくとも 4 つのディスクが必要です。

RAID10

パフォーマンスおよび冗長性 - RAID レベル 10 は、ネスト化した RAID またはハイブリッド RAID に

なります。ミラーリングしているディスクセットに対して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 レベル 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 組のペアで構成されます。RAID 10 には少なくとも 4 つのディスクが必要です。

C.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。

- **/boot**
- / (root)
- **/home**
- **swap**
- **/boot/efi**

/boot パーティション (最小限 1 GiB のサイズを推奨)

/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収納されます。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、別のディスクパーティションに置く必要があります。

警告

通常、**/boot** パーティションは、インストールプログラムにより自動的に作成されます。ただし、/ (ルート) パーティションが 2 TiB を超え、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。

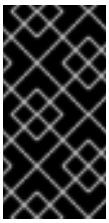
注記

RAID カードを実装している場合は、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、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)

「/ (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。

ルート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合しないように注意してください。**/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** 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)

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)

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作業負荷に依存するため、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いということになります。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、man ページの **mkswap(8)** 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、および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実行する場合、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るため、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、128GB 以上のサイズの swap パーティションは作成されません。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、またはサイズを 128GB 以上にしたい場合は、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。

/boot/efi パーティション (サイズは 200 MiB を推奨)

UEFI ベースの AMD、Intel、および 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。

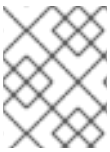
表C.1 システムの推奨 swap 領域

システム内の RAM の容量	推奨される swap 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 より小さい	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	4 GB から RAM 容量の半分まで	RAM 容量の 1.5 倍
64 GB を超える場合	ワークロードによる (最小 4GB)	ハイバネートは推奨されません

値が、範囲の境界線上にある場合 (システムの RAM が 2 GB、8 GB、または 64 GB などの場合)、swap 領域の決定やハイバネートへの対応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 領域を複数のディスクに分散させても、swap 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、インターフェー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。

注記

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設定のレイアウトをご利用ください。

C.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は、暗号化を検討してください。パーティションを暗号化すると、権限を持たない人が物理ストレージデバイスにはアクセスできても、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、少なくとも **/home** パーティションは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** に別のマウントポイントを持つことで、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、どのレイアウトを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) 物理デバイス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は「**Linux Filesystem Hierarchy Standard**」(http://refspecs.linuxfoundation.org/FHS_2.3/fhs-2.3.html) 確認してください。

- システムにインストールされるカーネルは、それぞれ **/boot** パーティションに約 20 MB の領域を必要とします。最も一般的な使用では、デフォルトの **/boot** パーティションサイズである 1 GB で十分なはずです。同時に多数のカーネルをインストールして維持しておく予定がある場合にはサイズを大きくしてください。

- **/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、**DNF** 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 3 GB となることを確認してください。

- 通常、**/var** ディレクトリーの内容は頻繁に変わります。古いソリッドステートドライブ (SSD) では、使用できなくなるまで読み取り/書き込みサイクル数が少なくて済むため、これが原因で問題が発生する場合があります。システムのルートが SSD にある場合は、従来の (platter) HDD の **/var** に別のマウントポイントを作成することを検討してください。

- **/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 GB、グラフィカル環境のインストールの場合は最低 10 GB 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** のみに適用され、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。

- 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。領域の必要性が変化した際に、ストレージの再割り当てを行うことで他のパーティションのデータを削除しなければならないという事態を避けたい場合など、未割り当ての領域を残すことで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、データをバックアップし、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のレイアウトの操作が必要になる可能性がある場合は、代わりに ext4 ファイルシステムを使用してください。

- インストール後にさらにハードドライブを追加して、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。このようなパーティションの詳細は 「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。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。