Red Hat Enterprise Linux 8

Performing an advanced RHEL installation

Installing Red Hat Enterprise Linux 8 using Kickstart

Last Updated: 2019-05-07

Red Hat Enterprise Linux 8 Performing an advanced RHEL installation

Installing Red Hat Enterprise Linux 8 using Kickstart

Legal Notice

Abstract

This document is aimed at users who want to perform an advanced Red Hat Enterprise Linux installation using Kickstart and configure advanced installation options.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. INTRODUCTION

Red Hat Enterprise Linux 8 delivers a stable, secure, consistent foundation across hybrid cloud deployments with the tools needed to deliver workloads faster with less effort. It can be deployed as a guest on supported hypervisors and Cloud provider environments as well as deployed on physical infrastructure, so your applications can take advantage of innovations in the leading hardware architecture platforms.

1.1. SUPPORTED ARCHITECTURES

Red Hat Enterprise Linux supports the following architectures:

- AMD and Intel 64-bit architectures
- The 64-bit ARM architecture
- IBM Power Systems, Little Endian
- IBM Z

1.2. INSTALLATION TERMINOLOGY

This section describes Red Hat Enterprise Linux installation terminology. Different terminology can be used for the same concepts, depending on its upstream or downstream origin.

Anaconda: The operating system installer used in Fedora, Red Hat Enterprise Linux, and their derivatives. Anaconda is a set of Python modules and scripts with additional files like Gtk widgets (written in C), systemd units, and dracut libraries. Together, they form a tool that allows users to set parameters of the resulting (target) system. In this document, the term **installation program** refers to the installation aspect of **Anaconda**.

1.3. INSTALLATION METHODS

You can install Red Hat Enterprise Linux using one of the following methods:

Quick install

Install Red Hat Enterprise Linux on AMD64, Intel 64, and 64-bit ARM architectures using the graphical user interface. The quick installation assumes that you are familiar with Red Hat Enterprise Linux and your environment, and that you can accept the default settings provided by the installation program.

Graphical install

Install Red Hat Enterprise Linux using the graphical user interface and customize the graphical settings for your specific requirements.

Automated install

Install Red Hat Enterprise Linux using Kickstart. The automated installation allows you to perform unattended operating system installation tasks.

Additional resources

- To perform a quick install on AMD64, Intel 64, and 64-bit ARM architectures using the graphical user interface, see the *Performing a standard RHEL installation* document.

- To perform a graphical install using the graphical user interface, see the *Performing a standard RHEL installation* document.

PART I. PERFORMING AN AUTOMATED INSTALLATION USING KICKSTART

CHAPTER 2. KICKSTART INSTALLATION BASICS

Learn what Kickstart is and how to automate installations of Red Hat Enterprise Linux with it.

2.1. WHAT ARE KICKSTART INSTALLATIONS

Kickstart installations offer a means to automate the installation process, either partially or fully. Kickstart files contain answers to all questions normally asked by the installation program, such as what time zone you want the system to use, how the drives should be partitioned, or which packages should be installed. Providing a prepared Kickstart file therefore allows you to perform the installation automatically, without need for any manual intervention from the user. This is especially useful when deploying Red Hat Enterprise Linux on a large number of systems at once.

In addition to allowing you to automate the installation, Kickstart files also provide more options regarding software selection. When installing Red Hat Enterprise Linux manually using the graphical installation interface, your software selection is limited to pre-defined environments and add-ons. A Kickstart file allows you to install or remove individual packages as well.

Kickstart files can be kept on a single server system and read by individual computers during the installation. This installation method can support the use of a single Kickstart file to install Red Hat Enterprise Linux on multiple machines, making it ideal for network and system administrators.

All Kickstart scripts and the log files of their execution are stored in the **/tmp** directory to assist with debugging installation issues.

NOTE

In previous versions of Red Hat Enterprise Linux, Kickstart could be used for upgrading the system as well. Starting with Red Hat Enterprise Linux 7, this functionality has been removed and system upgrades are instead handled by specialized tools. For details on upgrading to Red Hat Enterprise Linux 8, see the documents Upgrading to RHEL 8 and Considerations in adopting RHEL 8 .

2.2. AUTOMATED INSTALLATION WORKFLOW

Kickstart installations can be performed using a local DVD, a local hard drive, NFS, FTP, HTTP, or HTTPS.

To use Kickstart:

1. Create a Kickstart file. Instead of writing it entirely by hand, you can copy a Kickstart file saved after a manual installation, or use an online generator tool to create the file, and edit it afterwards.
2. Make the Kickstart file available on removable media, a hard drive or a network location using an HTTP(S), FTP, or NFS server.
3. Create the boot media, which will be used to begin the installation.
4. Make the installation source available, similarly to the Kickstart file.
5. Start the Kickstart installation. Use the **inst.ks=** boot option either in the boot menu or in the boot loader configuration file to load the Kickstart file and use it during the installation.

6. Let the installation finish. This will happen automatically if the Kickstart file contains all mandatory commands and sections. If one or more of these mandatory parts are missing, or if an error occurs, the installation will require manual intervention to finish.

CHAPTER 3. CREATING KICKSTART FILES

Learn how to create a Kickstart file easier and faster than writing the file manually from scratch. The methods are:

- Copy a Kickstart file created as a result of a manual installation
- Use the online Kickstart configuration tool

You can manually edit and customize the file later, changing only the complicated parts.

3.1. CREATING A KICKSTART FILE BY PERFORMING A MANUAL INSTALLATION

The recommended approach to creating Kickstart files is to perform a manual installation on one system first. After the installation completes, all choices made during the installation are saved into a file named **anaconda-ks.cfg**, located in the **/root/** directory on the installed system. You can then use this file to reproduce the installation in exactly the same way as before. Alternatively, copy this file, make any changes you need, and use the resulting configuration file for further installations.

Procedure

1. Install RHEL using the graphical interface on a system.
   During the installation, create a user with administrator privileges.
2. Finish the installation and reboot into the installed system.
3. Log into the system with the administrator account.
4. Copy the file **/root/anaconda-ks.cfg** to a location of your choice.
   - To display the file contents in terminal:

     ```
     # cat /root/anaconda-ks.cfg
     ```

     You can copy the output and save to another file of your choice.
   - To copy the file to another location, use the file manager. Remember to change permissions on the copy, so that the file can be read by non-root users.

IMPORTANT

The file contains information about users and passwords.

Additional resources

- Performing a standard RHEL installation

3.2. CREATING A KICKSTART FILE WITH THE KICKSTART CONFIGURATION TOOL

Users with a Red Hat Customer Portal account can use the Kickstart Configuration Tool in the Customer Portal Labs to generate Kickstart files online. This tool will walk you through the basic configuration and

enables you to download the resulting Kickstart file. However, the tool currently does not support any advanced partitioning.

Procedure

1. Open the Kickstart generator lab information page at https://access.redhat.com/labsinfo/kickstartconfig
2. Click the **Go to Application** button to the left of heading and wait for the next page to load.
3. Select Red Hat Enterprise Linux 8 in the drop-down menu and wait for the page to update.
4. Complete the form details that describe the system to be installed.
   You can use the links on the left side of the form to quickly navigate between the sections of the form.
5. To download the generated Kickstart file, click the red **Download** button at the top of the page.
   Your web browser will save the file.

CHAPTER 4. MAKING KICKSTART FILES AVAILABLE TO THE INSTALLATION PROGRAM

Learn about the options you have for placing the Kickstart file so that the installation program on the target system can read this file.

4.1. PORTS FOR NETWORK-BASED INSTALLATION

The following table lists the ports that must be open on the server providing the files for each type of network-based installation.

Table 4.1. Ports for network-based installation

Protocol used	Ports to open
HTTP	80
HTTPS	443
NFS	2049, 111, 20048
TFTP	69

Additional resources

- See the *Securing networks* document for more information.

4.2. MAKING A KICKSTART FILE AVAILABLE ON AN NFS SERVER

This procedure describes how to place the Kickstart script file on an NFS server. This method enables you to install multiple systems from a single source, without having to use physical media. A network-based installation is convenient when used with a TFTP server, as it enables you to boot the installation from the network. This approach eliminates the need to create physical media and simultaneously deploys Red Hat Enterprise Linux on multiple systems.

Prerequisites

- You must have administrator level access to a server system reachable on the local network from the system to be installed.
- Your server firewall must allow the system you are installing to access the server. See Section 4.1, "Ports for network-based installation" for more information.

Procedure

NOTE

To perform an NFS-based installation, another system must act as the NFS host. This procedure is a basic outline of the process. The steps to set up an NFS server vary depending on the system's architecture, operating system, package manager, and service manager.

1. Install the **nfs-utils** package by running the following command as root:

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

2. Copy the Kickstart file to a directory on the NFS server.

3. Open the **/etc/exports** file using a text editor and add a line with the following syntax:

```
/exported_directory/ clients
```

4. Replace */exported_directory/* with the full path to the directory holding the Kickstart file. Instead of *clients*, use the host name or IP address of the computer that is to be installed from this NFS server, the subnetwork from which all computers are to have access the ISO image, or the asterisk sign (*****) if you want to allow any computer with network access to the NFS server to use the ISO image. See the *exports(5)* man page for detailed information about the format of this field.
A basic configuration that makes the **/rhel8-install/** directory available as read-only to all clients is:

```
/rhel8-install *
```

5. Save the **/etc/exports** file and exit the text editor.

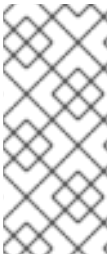
6. Start the nfs service:

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

If the service was running before you changed the **/etc/exports** file, enter the following command, in order for the running NFS server to reload its configuration:

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

The Kickstart file is now accessible over NFS and ready to be used for installation.



NOTE

When specifying the Kickstart source, use **nfs:** as the protocol, the server's host name or IP address, the colon sign (**:**), and the path inside directory holding the file. For example, if the server's host name is **myserver.example.com** and you have saved the file in **/rhel8-install/my-ks.cfg**, specify **inst.ks=nfs:myserver.example.com:/rhel8-install/my-ks.cfg** as the installation source boot option.

Additional resources

- For details on setting up TFTP server for PXE boot from network, see Chapter 11, *Preparing to install from the network using PXE*.

4.3. MAKING A KICKSTART FILE AVAILABLE ON AN HTTP OR HTTPS SERVER

Place the Kickstart file on an HTTP or HTTPS server for a network-based installation.

Prerequisites

- You must have administrator level access to a server system reachable on the local network from the system to be installed.
- Your server firewall must allow the system you are installing to access the server. See Section 4.1, "Ports for network-based installation" for more information.

Procedure

1. Install the **httpd** package by running the following command as root:

```
# yum install httpd
```

WARNING

If your Apache web server configuration enables SSL security, verify that you only enable the TLSv1 protocol, and disable SSLv2 and SSLv3. This is due to the POODLE SSL vulnerability (CVE-2014-3566). See https://access.redhat.com/solutions/1232413 for details.

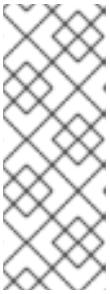
IMPORTANT

If you use an HTTPS server with a self-signed certificate, you must boot the installation program with the **inst.noverifyssl** option.

2. Copy the Kickstart file to the HTTP(S) server into a subdirectory of the **/var/www/html/** directory.
3. Start the httpd service:

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

The Kickstart file is now accessible and ready to be used for installation.



NOTE

When specifying the location of the Kickstart file, use http:// or https:// as the protocol, the server's host name or IP address, and the path of the Kickstart file, relative to the HTTP server root. For example, if you are using HTTP, the server's host name is **myserver.example.com**, and you have copied the Kickstart file as **/var/www/html/rhel8-install/my-ks.cfg**, specify **http://myserver.example.com/rhel8-install/my-ks.cfg** as the file location.

Additional resources

- For more information about HTTP and FTP servers, see the Deploying different types of servers document.

4.4. MAKING A KICKSTART FILE AVAILABLE ON A LOCAL VOLUME

A Kickstart file can be present directly on a volume on the system to be installed. This lets you bypass the need for another system.

Prerequisites

- You must have a drive that can be moved to the system to be installed, such as a USB stick.
- The drive must contain a partition that can be read by the installation program. The supported types are **ext2**, **ext3**, **ext4**, **xfs**, and **fat**.
- The drive must be already connected to the system and its volumes mounted.

Procedure

1. Find where the volumes from the drive are mounted, and what their UUIDs are:

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

2. Navigate to a suitable file system mounted from the drive.
3. Copy the Kickstart file into any place in this file system.
4. The string to use later with the **inst.ks=** option is in the form **hd:UUID=*UUID*:*path/to/kickstart-file.cfg***. Note that the path is relative to the file system root, not to the / root of file system hierarchy.
5. Unmount the drive volumes:

```
# umount /dev/xyz ...
```

   Add all the volumes to the command, separated by spaces.

4.5. MAKING A KICKSTART FILE AVAILABLE ON A LOCAL VOLUME FOR AUTOMATIC LOADING

A specially named Kickstart file can be present in the root of a specially named volume on the system to be installed. This lets you bypass the need for another system, and makes the installation program load the file automatically.

Prerequisites

- You must have a drive that can be moved to the system to be installed, such as a USB stick.
- The drive must contain a partition that can be read by the installation program. The supported types are **ext2**, **ext3**, **ext4**, **xfs**, and **fat**.
- The drive must be already connected to the system and its volumes mounted.

Procedure

1. Find where the volumes from the drive are mounted:

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

2. Navigate to a suitable file system mounted from the drive.
3. Copy the Kickstart file into root of this file system.
4. Rename the volume as **OEMDRV**:
   - For **ext2**, **ext3**, and **ext4** file systems:

     ```
     # e2label /dev/xyz OEMDRV
     ```

   - For the XFS file system:

     ```
     # xfs_admin -L OEMDRV /dev/xyz
     ```

   Replace */dev/xyz* with the path to the volume's block device.
5. Unmount the drive volumes:

   ```
   # umount /dev/xyz ...
   ```

   Add all the volumes to the command, separated by spaces.

CHAPTER 5. CREATING INSTALLATION SOURCES FOR KICKSTART INSTALLATIONS

This section describes how to create an installation source for the Boot ISO image using the Binary DVD ISO image that contains the required repositories and software packages.

5.1. TYPES OF INSTALLATION SOURCE

You can use one of the following installation sources for minimal boot images:

- **DVD:** Burn the Binary DVD ISO image to DVD and configure the installation program to install the software packages.
- **Hard drive or USB drive:** Copy the Binary DVD ISO image to the drive and configure the installation program to install the software packages from the drive. If you use a USB drive, verify that it is connected to the system before the installation begins. The installation program cannot detect media after the installation begins.
    - **Hard drive limitation**: The Binary DVD ISO image on the hard drive must be on a partition with a file system that the installation program can mount. The supported file systems are **xfs**, **ext2**, **ext3**, **ext4**, and **vfat (FAT32)**.

WARNING

On Microsoft Windows systems, the default file system used when formatting hard drives is NTFS; the exFAT file system is also available. However, neither of these file systems can be mounted during the installation. If you are creating a hard drive or a USB drive as an installation source on Microsoft Windows, verify that you formatted the drive as FAT32, however, the FAT32 file system cannot store files larger than 4 GiB.

In Red Hat Enterprise Linux 8 you can enable installation from a repository on a local hard drive. To do so, you need to copy the content of DVD ISO image to a directory on a hard drive and then specify the directory as installation source instead of the ISO image. For example: **inst.repo=hd:<device>:<path to the repository>**

- **Network location:** Copy the Binary DVD ISO image or the installation tree (extracted contents of the Binary DVD ISO image) to a network location and perform the installation over the network using the following protocols:
    - **NFS:** The Binary DVD ISO image is in a Network File System (NFS) share.
    - **HTTPS or HTTP:** The installation tree is on a network location that is accessible over HTTP or HTTPS.

5.2. PORTS FOR NETWORK-BASED INSTALLATION

The following table lists the ports that must be open on the server providing the files for each type of network-based installation.

Table 5.1. Ports for network-based installation

Protocol used	Ports to open
HTTP	80
HTTPS	443
NFS	2049, 111, 20048
TFTP	69

Additional resources

- See the *Securing networks* document for more information.

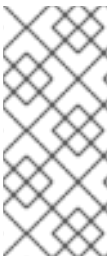
5.3. CREATING AN INSTALLATION SOURCE ON AN NFS SERVER

Follow the steps in this procedure to place the installation source on an NFS server. Use this installation method to install multiple systems from a single source, without having to connect to physical media. A network-based installation is convenient when used with a TFTP server, as it enables you to boot the installation from the network. This approach eliminates the need to create physical media and simultaneously deploys Red Hat Enterprise Linux on multiple systems.

Prerequisites

- You have downloaded a Binary DVD image. See Downloading the installation ISO image from the *Performing a standard RHEL installation* document for more information.
- You have created a bootable CD, DVD, or USB device from the image file. See Creating installation media from the *Performing a standard RHEL installation* document for more information.
- You have verified that your firewall allows the server you are installing to access the remote installation source. See Ports for network-based installation from the *Performing a standard RHEL installation* document for more information.

Procedure



NOTE

The NFS installation method uses the Binary DVD ISO image in a Network File System server's exported directory, which the system must be able to read. To perform an NFS-based installation, another system must act as the NFS host. The steps to configure an NFS server vary depending on the system architecture, operating system, package manager, and service manager. This procedure contains a basic outline of the process.

1. Install the **nfs-utils** package by running the following command as root:

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

2. Copy the Binary DVD ISO image to a directory on the NFS server.
3. Open the **/etc/exports** file using a text editor and add a line with the following syntax:

```
/exported_directory/ clients
```

4. Replace */exported_directory/* with the full path to the directory with the ISO image. Replace *clients* with the host name or IP address of the target system, the subnetwork that all target systems can use to access the ISO image, or the asterisk sign (*) if you want to allow any system with network access to the NFS server to use the ISO image. See the **exports(5)** man page for detailed information about the format of this field.
   A basic configuration that makes the **/rhel8-install/** directory available as read-only to all clients is:

```
/rhel8-install *
```

5. Save the **/etc/exports** file and exit the text editor.
6. Start the nfs service:

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

   If the service was running before you changed the **/etc/exports** file, run the following command for the running NFS server to reload its configuration:

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

   The ISO image is now accessible over NFS and ready to be used as an installation source.

NOTE

When configuring the installation source, use **nfs:** as the protocol, the server host name or IP address, the colon sign **(:)**, and the directory holding the ISO image. For example, if the server host name is **myserver.example.com** and you have saved the ISO image in **/rhel8-install/**, specify **nfs:myserver.example.com:/rhel8-install/** as the installation source.

5.4. CREATING AN INSTALLATION SOURCE USING HTTP OR HTTPS

Follow the steps in this procedure to create an installation source for a network-based installation using an installation tree, which is a directory containing extracted contents of the Binary DVD ISO image and a valid **.treeinfo** file. The installation source is accessed over HTTP or HTTPS.

Prerequisites

- You have downloaded a Binary DVD image. See Downloading the installation ISO image from the *Performing a standard RHEL installation* document for more information.
- You have created a bootable CD, DVD, or USB device from the image file. See Creating installation media from the *Performing a standard RHEL installation* document for more information.

- You have verified that your firewall allows the server you are installing to access the remote installation source. See Ports for network-based installation from the *Performing a standard RHEL installation* document for more information.

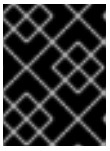
Procedure

1. Install the **httpd** package by running the following command as root:

```
# yum install httpd
```

WARNING

If your Apache web server configuration enables SSL security, verify that you enable only the TLSv1 protocol, and disable SSLv2 and SSLv3. This is due to the POODLE SSL vulnerability (CVE-2014-3566). See https://access.redhat.com/solutions/1232413 for details.



IMPORTANT

If you use an HTTPS server with a self-signed certificate, you must boot the installation program with the **noverifyssl** option.

2. Copy the Binary DVD ISO image to the HTTP(S) server.
3. Mount the Binary DVD ISO image, using the **mount** command, to a suitable directory:

```
# mount -o loop,ro -t iso9660 /image_directory/image.iso /mount_point/
```

   - Replace */image_directory/image.iso* with the path to the Binary DVD ISO image.
   - Replace */mount_point/* with the path to the directory where you want to locate the contents of the ISO image.

4. Copy the files from the mounted image to the HTTP(S) server root. This command creates the **/var/www/html/rhel8-install/** directory with the contents of the image.

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

5. Start the **httpd** service:

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

The installation tree is now accessible and ready to be used as the installation source.

NOTE

When configuring the installation source, use **http://** or **https://** as the protocol, the server host name or IP address, and the directory that contains the files from the ISO image, relative to the HTTP server root. For example, if you are using HTTP, the server host name is **myserver.example.com**, and you have copied the files from the image to **/var/www/html/rhel8-install/**, specify **http://myserver.example.com/rhel8-install/** as the installation source.

Additional resources

- For more information about HTTP servers, see the *Deploying different types of servers* document.

CHAPTER 6. STARTING KICKSTART INSTALLATIONS

You can start Kickstart installations in multiple ways:

- Manually by entering the installation program boot menu and specifying the options including Kickstart file there.
- Automatically by editing the boot options in PXE boot.
- Automatically by providing the file on a volume with specific name.

Learn how to perform each of these methods in the following sections.

6.1. STARTING A KICKSTART INSTALLATION MANUALLY

This section explains how to start a Kickstart installation manually, which means some user interaction is required (adding boot options at the **boot:** prompt). Use the boot option **inst.ks=*location*** when booting the installation system, replacing location with the location of your Kickstart file. The exact way to specify the boot option depends on your system's architecture.

Prerequisites

- You have a Kickstart file ready in a location accessible from the system to be installated

Procedure

1. Boot the system using a local media (a CD, DVD, or a USB flash drive).
2. At the boot prompt, specify the required boot options.
   a. If the Kickstart file or a required repository is in a network location, you may need to configure the network using the **ip=** option. The installer tries to configure all network devices using the DHCP protocol by default without this option.
   b. Add the **inst.ks=** boot option and the location of the Kickstart file.
   c. In order to access a software source from which necessary packages will be installed, you may need to add the **inst.repo=** option. If you do not specify this option, you must specify the installation source in the Kickstart file.
3. Start the installation by confirming your added boot options.
   The installation begins now, using the options specified in the Kickstart file. If the Kickstart file is valid and contains all required commands, the installation is completely automated from this point forward.

6.2. STARTING A KICKSTART INSTALLATION AUTOMATICALLY USING PXE

AMD64, Intel 64, and 64-bit ARM systems and IBM Power Systems servers have the ability to boot using a PXE server. When you configure the PXE server, you can add the boot option into the boot loader configuration file, which in turn lets you start the installation automatically. Using this approach, it is possible to automate the installation completely, including the boot process.

This procedure is intended as a general reference; detailed steps differ based on your system's architecture, and not all options are available on all architectures (for example, you cannot use PXE boot on IBM Z).

Prerequisites

- You must have a Kickstart file ready in a location accessible from the system to be installated.
- You must have a PXE server which can be used to boot the system and begin the installation.

Procedure

1. Open the boot loader configuration file on your PXE server, and add the **inst.ks=** boot option to the appropriate line. The name of the file and its syntax depends on your system's architecture and hardware:
   - On AMD64 and Intel 64 systems with BIOS, the file name can be either default or based on your system's IP address. In this case, add the **inst.ks=** option to the append line in the installation entry. A sample append line in the configuration file looks similar to the following:

     ```
     append initrd=initrd.img inst.ks=http://10.32.5.1/mnt/archive/RHEL-8/8.x/x86_64/kickstarts/ks.cfg
     ```

   - On systems using the GRUB2 boot loader (AMD64, Intel 64, and 64-bit ARM systems with UEFI firmware and IBM Power Systems servers), the file name will be **grub.cfg**. In this file, append the **inst.ks=** option to the kernel line in the installation entry. A sample kernel line in the configuration file will look similar to the following:

     ```
     kernel vmlinuz inst.ks=http://10.32.5.1/mnt/archive/RHEL-8/8.x/x86_64/kickstarts/ks.cfg
     ```

2. Boot the installation from the network server.
   The installation begins now, using the installation options specified in the Kickstart file. If the Kickstart file is valid and contains all required commands, the installation is completely automated.

Additional resources

- For information about setting up a PXE server, see Chapter 11, *Preparing to install from the network using PXE*.

6.3. STARTING A KICKSTART INSTALLATION AUTOMATICALLY USING A LOCAL VOLUME

You can start a Kickstart installation by putting a Kickstart file with a specific name on a specifically labelled storage volume.

Prerequisites

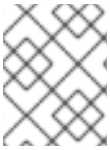
- You must have a volume prepared with label **OEMDRV** and the Kickstart file present in its root as **ks.cfg**.
- A drive containing this volume must be available on the system as the installation program boots.

Procedure

1. Boot the system using a local media (a CD, DVD, or a USB flash drive).
2. At the boot prompt, specify the required boot options.
   a. If a required repository is in a network location, you may need to configure the network using the **ip=** option. The installer tries to configure all network devices using the DHCP protocol by default without this option.
   b. In order to access a software source from which necessary packages will be installed, you may need to add the **inst.repo=** option. If you do not specify this option, you must specify the installation source in the Kickstart file.
3. Start the installation by confirming your added boot options.
   The installation begins now, and the Kickstart file is automatically detected and used to start an automated Kickstart installation.

CHAPTER 7. CONSOLES AND LOGGING DURING INSTALLATION

The Red Hat Enterprise Linux installer uses the **tmux** terminal multiplexer to display and control several windows you can use in addition to the main interface. Each of these windows serves a different purpose - they display several different logs, which can be used to troubleshoot any issues during the installation, and one of the windows provides an interactive shell prompt with **root** privileges, unless this prompt was specifically disabled using a boot option or a Kickstart command.



NOTE

In general, there is no reason to leave the default graphical installation environment unless you need to diagnose an installation problem.

The terminal multiplexer is running in virtual console 1. To switch from the actual installation environment to **tmux**, press **Ctrl**+**Alt**+**F1**. To go back to the main installation interface which runs in virtual console 6, press **Ctrl**+**Alt**+**F6**.

NOTE

If you choose text mode installation, you will start in virtual console 1 (**tmux**), and switching to console 6 will open a shell prompt instead of a graphical interface.

The console running **tmux** has 5 available windows; their contents are described in the table below, along with keyboard shortcuts used to access them. Note that the keyboard shortcuts are two-part: first press **Ctrl**+**b**, then release both keys, and press the number key for the window you want to use.

You can also use **Ctrl**+**b n** and **Ctrl**+**b p** to switch to the next or previous **tmux** window, respectively.

Table 7.1. Available tmux windows

Shortcut	Contents
Ctrl+**b 1**	Main installation program window. Contains text-based prompts (during text mode installation or if you use VNC direct mode), and also some debugging information.
Ctrl+**b 2**	Interactive shell prompt with **root** privileges.
Ctrl+**b 3**	Installation log; displays messages stored in **/tmp/anaconda.log**.
Ctrl+**b 4**	Storage log; displays messages related storage devices from kernel and system services, stored in **/tmp/storage.log**.
Ctrl+**b 5**	Program log; displays messages from other system utilities, stored in **/tmp/program.log**.

CHAPTER 8. MAINTAINING KICKSTART FILES

You can run automated checks on Kickstart files. Typically, you will want to verify that a new or problematic Kickstart file is valid.

8.1. INSTALLING KICKSTART MAINTENANCE TOOLS

To use the Kickstart maintenance tools, you must install the package that contains them.

Procedure

- Install the **pykickstart** package:

```
# yum install pykickstart
```

8.2. VERIFYING A KICKSTART FILE

Use the **ksvalidator** command line utility to verify that your Kickstart file is valid. This is useful when you make extensive changes to a Kickstart file.

Procedure

- Run **ksvalidator** on your Kickstart file:

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

  Replace */path/to/kickstart.ks* with the path to the Kickstart file you want to verify.

IMPORTANT

The validation tool cannot guarantee the installation will be successful. It ensures only that the syntax is correct and that the file does not include deprecated options. It does not attempt to validate the **%pre**, **%post** and **%packages** sections of the Kickstart file.

Additional resources

- The *ksvalidator(1)* manual page.

PART II. ADVANCED CONFIGURATION OPTIONS

CHAPTER 9. CONFIGURING SYSTEM PURPOSE

System administrators use System Purpose to record the intended use of a Red Hat Enterprise Linux 8 system by the organization. When you set a system's purpose, the entitlement server receives information that helps auto-attach a subscription that satisfies the intended use of the system. This section describes how to configure System Purpose using Kickstart.

9.1. OVERVIEW

You can enter System Purpose data in the following ways:

- During image creation.
- During installation using the graphical user interface.
- Using Kickstart automation scripts.
- Using the **syspurpose** command-line tool.

You can configure the following components:

- **Role**:
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Workstation
  - Red Hat Enterprise Linux Compute Node
- **Service Level Agreement**
  - Premium
  - Standard
  - Self-Support
- **Usage**:
  - Production
  - Disaster Recovery
  - Development/Test

Benefits include:

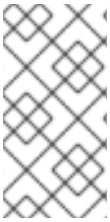
- In-depth system-level information for system administrators and business operations.
- Reduced overhead when determining why a system was procured and its intended purpose.
- Improved customer experience of Subscription Manager auto-attach as well as automated discovery and reconciliation of system usage.

Additional resources

- For more information about Image Builder, see the *Composing a customized RHEL system image* document.
- For more information about Kickstart, see the *Performing an advanced RHEL installation* document.
- For more information about Subscription Manager, see the *Using and Configuring Red Hat Subscription Manager* document.

9.2. CONFIGURING SYSTEM PURPOSE IN A KICKSTART FILE

Follow the steps in this procedure to use the **syspurpose** command to configure System Purpose in a Kickstart configuration file.



NOTE

While it is strongly recommended that you configure System Purpose, it is an optional feature of the Red Hat Enterprise Linux installation program. If you want to enable System Purpose after the installation completes, you can do so using the **syspurpose** command-line tool.

The following actions are available:

role

Set the intended role of the system. This action uses the following format:

```
syspurpose --role=
```

The role assigned can be:

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

sla

Set the intended sla of the system. This action uses the following format:

```
syspurpose --sla=
```

The sla assigned can be:

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

usage

Set the intended usage of the system. This action uses the following format:

```
syspurpose --usage=
```

The usage assigned can be:

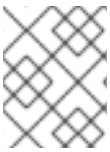
- **Production**
- **Disaster Recovery**
- **Development/Test**

9.3. RELATED INFORMATION

- For information about configuring System Purpose using the graphical user interface, or the **syspurpose** command-line tool, see the *Performing a standard RHEL installation* document.

CHAPTER 10. UPDATING DRIVERS DURING INSTALLATION

This section describes how to complete a driver update during the Red Hat Enterprise Linux installation process.



NOTE

This is an optional step of the installation process. Red Hat recommends that you do not perform a driver update unless it is necessary.

10.1. PREREQUISITE

You have been notified by Red Hat, your hardware vendor, or a trusted third-party vendor that a driver update is required during Red Hat Enterprise Linux installation.

10.2. OVERVIEW

Red Hat Enterprise Linux supports drivers for many hardware devices but some newly-released drivers may not be supported. A driver update should only be performed if an unsupported driver prevents the installation from completing. Updating drivers during installation is typically only required to support a particular configuration. For example, installing drivers for a storage adapter card that provides access to your system's storage devices.

WARNING

Driver update disks may disable conflicting kernel drivers. In rare cases, unloading a kernel module may cause installation errors.

10.3. TYPES OF DRIVER UPDATE

Red Hat, your hardware vendor, or a trusted third party provides the driver update as an ISO image file. Once you receive the ISO image file, choose the type of driver update.

Types of driver update

Automatic

The recommended driver update method; a storage device (including a CD, DVD, or USB flash drive) labeled **OEMDRV** is physically connected to the system. If the **OEMDRV** storage device is present when the installation starts, it is treated as a driver update disk, and the installation program automatically loads its drivers.

Assisted

The installation program prompts you to locate a driver update. You can use any local storage device with a label other than **OEMDRV**. The **inst.dd** boot option is specified when starting the installation. If you use this option without any parameters, the installation program displays all of the storage devices connected to the system, and prompts you to select a device that contains a driver update.

Manual

Manually specify a path to a driver update image or an RPM package. You can use any local storage

device with a label other than **OEMDRV**, or a network location accessible from the installation system. The **inst.dd=location** boot option is specified when starting the installation, where *location* is the path to a driver update disk or ISO image. When you specify this option, the installation program attempts to load any driver updates found at the specified location. With manual driver updates, you can specify local storage devices, or a network location (HTTP, HTTPS or FTP server).

NOTE

- You can use both **inst.dd=location** and **inst.dd** simultaneously, where *location* is the path to a driver update disk or ISO image. In this scenario, the installation program attempts to load any available driver updates from the location and also prompts you to select a device that contains the driver update.
- Initialize the network using the **ip= option** when loading a driver update from a network location.

Limitations

On UEFI systems with the Secure Boot technology enabled, all drivers must be signed with a valid certificate. Red Hat drivers are signed by one of Red Hat's private keys and authenticated by its corresponding public key in the kernel. If you load additional, separate drivers, verify that they are signed.

10.4. PREPARING A DRIVER UPDATE

This procedure describes how to prepare a driver update on a CD and DVD.

Prerequisites

- You received the driver update ISO image from Red Hat, your hardware vendor, or a trusted third-party vendor.
- You burned the driver update ISO image to a CD or DVD.

WARNING

If only a single ISO image file ending in **.iso** is available on the CD or DVD, the burn process has not been successful. See your system's burning software documentation for instructions on how to burn ISO images to a CD or DVD.

Procedure

1. Insert the driver update CD or DVD into your system's CD/DVD drive, and browse it using the system's file manager tool.
2. Verify that a single file **rhdd3** is available. **rhdd3** is a signature file that contains the driver description and a directory named **rpms**, which contains the RPM packages with the actual drivers for various architectures.

10.5. PERFORMING AN AUTOMATIC DRIVER UPDATE

This procedure describes how to perform an automatic driver update during installation.

Prerequisites

- You have placed the driver update image on a standard disk partition with an **OEMDRV** label or burnt the **OEMDRV** driver update image to a CD or DVD. Advanced storage, such as RAID or LVM volumes, may not be accessible during the driver update process.
- You have connected a block device with an **OEMDRV** volume label to your system, or inserted the prepared CD or DVD into your system's CD/DVD drive before starting the installation process.

Procedure

1. Once you have completed the prerequisite steps, the drivers are automatically loaded when the installation program starts, and installed on the system during the installation process.

10.6. PERFORMING AN ASSISTED DRIVER UPDATE

This procedure describes how to perform an assisted driver update during installation.

Prerequisite

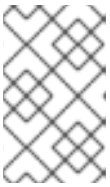
You have connected a block device without an **OEMDRV** volume label to your system and copied the driver disk image to this device, or you have prepared a driver update CD or DVD and inserted it into your system's CD/DVD drive before starting the installation process.

NOTE

If you burned an ISO image file to a CD or DVD but it does not have the **OEMDRV** volume label, you can use the **inst.dd** option with no arguments. The installation program provides an option to scan and select drivers from the CD or DVD. In this scenario, the installation program does not prompt you to select a driver update ISO image. Another scenario is to use the CD or DVD with the **inst.dd=location** boot option; this allows the installation program to automatically scan the CD or DVD for driver updates. For more information, see Section 10.7, "Performing a manual driver update" .

Procedure

1. From the boot menu window, press the **Tab** key on your keyboard to display the boot command line.
2. Append the **inst.dd** boot option to the command line and press **Enter** to execute the boot process.
3. From the menu, select a local disk partition or a CD or DVD device. The installation program scans for ISO files, or driver update RPM packages.
4. Optional: Select the driver update ISO file.



NOTE

This step is not required if the selected device or partition contains driver update RPM packages rather than an ISO image file, for example, an optical drive containing a driver update CD or DVD.

5. Select the required drivers.

   a. Use the number keys on your keyboard to toggle the driver selection.

   b. Press **c** to install the selected driver. The selected driver is loaded and the installation process starts.

10.7. PERFORMING A MANUAL DRIVER UPDATE

This procedure describes how to perform a manual driver update during installation.

Prerequisite

Place the driver update ISO image file on a USB flash drive or a web server, and connect it to your computer.

Procedure

1. From the boot menu window, press the **Tab** key on your keyboard to display the boot command line.

2. Append the **inst.dd=location** boot option to the command line, where location is a path to the driver update. Typically, the image file is located on a web server, for example, http://server.example.com/dd.iso, or on a USB flash drive, for example, **/dev/sdb1**. It is also possible to specify an RPM package containing the driver update, for example http://server.example.com/dd.rpm.

3. Press **Enter** to execute the boot process. The drivers available at the specified location are automatically loaded and the installation process starts.

Additional resources

- For more information about the **inst.dd** boot option, see the upstream inst.dd boot option content.

- For more information about all boot options, see the upstream Boot Options content.

10.8. DISABLING A DRIVER

This procedure describes how to disable a malfunctioning driver.

Prerequisites

- You have booted the installation program boot menu.

Procedure

1. From the boot menu, press the **Tab** key on your keyboard to display the boot command line.

2. Append the **modprobe.blacklist=driver_name** boot option to the command line.

3. Replace *driver_name* with the name of the driver or drivers you want to disable, for example:

   ```
   modprobe.blacklist=ahci
   ```

   Drivers disabled using the **modprobe.blacklist=** boot option remain disabled on the installed system and appear in the **/etc/modprobe.d/anaconda-blacklist.conf** file.

4. Press **Enter** to execute the boot process.

CHAPTER 11. PREPARING TO INSTALL FROM THE NETWORK USING PXE

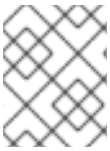
This section describes how to configure TFTP and DHCP on a PXE server to enable PXE boot and network installation.

11.1. NETWORK INSTALL OVERVIEW

A network installation allows you to install Red Hat Enterprise Linux to a system that has access to an installation server. At a minimum, two systems are required for a network installation:

PXE Server: A system running a DHCP server, a TFTP server, and an HTTP, HTTPS, FTP, or NFS server. While each server can run on a different physical system, the procedures in this section assume a single system is running all servers.

Client: The system to which you are installing Red Hat Enterprise Linux. Once installation starts, the client queries the DHCP server, receives the boot files from the TFTP server, and downloads the installation image from the HTTP, HTTPS, FTP or NFS server. Unlike other installation methods, the client does not require any physical boot media for the installation to start.



NOTE

To boot a client from the network, configure it in BIOS/UEFI or a quick boot menu. On some hardware, the option to boot from a network might be disabled, or not available.

The workflow steps to prepare to install Red Hat Enterprise Linux from a network using PXE are as follows:

Steps

1. Export the installation ISO image (or the installation tree) to an NFS, HTTPS, HTTP, or FTP server.
2. Configure the TFTP server and DHCP server, and start the TFTP service on the PXE server.
3. Boot the client, and start the installation.

IMPORTANT

The GRUB2 boot loader supports a network boot from HTTP in addition to a TFTP server. Sending the boot files (the kernel and initial RAM disk - vmlinuz and initrd) over this protocol might be slow and result in timeout failures. An HTTP server does not carry this risk, but it is recommended that you use a TFTP server when sending the boot files.

Additional resources

- To export the installation ISO image to a network location, see Chapter 5, *Creating installation sources for Kickstart installations* for information.
- To configure the TFTP server and DHCP server, and start the TFTP service, see Section 11.2, "Configuring a TFTP server for BIOS-based clients", Section 11.3, "Configuring a TFTP server for UEFI-based clients", and Section 11.4, "Configuring a network server for IBM Power systems" for information.

- Red Hat Satellite can automate the setup of a PXE server. For more information, see the Red Hat Satellite product documentation.

11.2. CONFIGURING A TFTP SERVER FOR BIOS-BASED CLIENTS

This procedure describes how to configure a TFTP server and DHCP server, and start the TFTP service on the PXE server for BIOS-based AMD and Intel 64-bit systems.

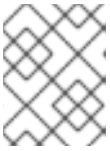
Procedure

1. As root, install the **tftp-server** package:

```
# yum install tftp-server
```

2. Allow incoming connections to the **tftp service** in the firewall:

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



NOTE

This command enables temporary access until the next server reboot. To enable permanent access, add the **--permanent** option to the command.

3. Configure your DHCP server to use the boot images packaged with **SYSLINUX**. A sample configuration in the **/etc/dhcp/dhcpd.conf** file might look like:

```
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. Access the **pxelinux.0** file from the **SYSLINUX** package in the Binary DVD ISO image file:

```
# 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. Extract the package:

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

6. Create a **pxelinux**/ directory within **tftpboot**/ and copy the required files, for example: **pxelinux.0 libcom.c32**, **ldlinux.c32**, **vesamenu.c32** into it:

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

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

7. Create the directory **pxelinux.cfg**/ in the **pxelinux**/ directory:

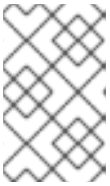
```
# mkdir /var/lib/tftpboot/pxelinux/pxelinux.cfg
```

8. Add a default configuration file to the **pxelinux.cfg**/ directory. A sample configuration file at **/var/lib/tftpboot/pxelinux/pxelinux.cfg/default** might look like:

```
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
```



NOTE

The **inst.repo=** option must be used to specify the installation program's image as well as the installation source. The installation program cannot boot without this option.

9. Create a subdirectory to store the boot image files in the **/var/lib/tftpboot/** directory, and copy the boot image files to the directory. In this example, we use the directory **/var/lib/tftpboot/pxelinux/images/RHEL-8.0/**:

```
# 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. Start and enable the **dhcpd** service:

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

11. Start and enable the **xinetd** service that manages the **tftp** service:

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

The PXE boot server is now ready to serve PXE clients. You can start the client (the system to which you are installing Red Hat Enterprise Linux), select **PXE Boot** when prompted to specify a boot source, and start the network installation.

11.3. CONFIGURING A TFTP SERVER FOR UEFI-BASED CLIENTS

This procedure describes how to configure a TFTP server and DHCP server, and start the TFTP service on the PXE server for UEFI-based AMD64, Intel 64, and 64-bit ARM systems.

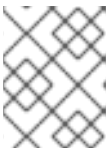
Procedure

1. As root, install the **tftp-server** package:

```
# yum install tftp-server
```

2. Allow incoming connections to the **tftp service** in the firewall:

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



NOTE

This command enables temporary access until the next server reboot. To enable permanent access, add the **--permanent** option to the command.

3. Configure your DHCP server to use the boot images packaged with **shim**. A sample configuration in the **/etc/dhcp/dhcpd.conf** file might look like:

```
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. Access the **shim.efi** file from the **shim** package, and the **grubx64.efi** file from the **grub2-efi** package in the Binary DVD ISO image file:

```
# 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. Extract the packages:

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

```
# rpm2cpio grub2-efi-version-architecture.rpm | cpio -dimv
```

6. Copy the EFI boot images from your boot directory.

```
# 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
```

IMPORTANT

You must rename the **shimx64.efi** file to **shim.efi after** it is copied.

7. Add a configuration file named **grub.cfg** to the **tftpboot/** directory. A sample configuration file at **/var/lib/tftpboot/uefi/grub.cfg** may look like:

```
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
}
```

NOTE

The **inst.repo=** option must be used to specify the installation program's image as well as the installation source. The installation program cannot boot without this option.

8. Create a subdirectory to store the boot image files in the **/var/lib/tftpboot/** directory, and copy the boot image files to the directory. In this example, we use the directory **/var/lib/tftpboot/images/RHEL-8.0/**:

```
# 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. Start and enable the **dhcpd** service:

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

10. Start and enable the **xinetd** service that manages the **tftp** service:

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

The PXE boot server is now ready to serve PXE clients. You can start the client (the system to which you are installing Red Hat Enterprise Linux), select **PXE Boot** when prompted to specify a boot source, and start the network installation.

Additional resources

- For more information about **shim**, see the upstream documentation: Using the Shim Program .

11.4. CONFIGURING A NETWORK SERVER FOR IBM POWER SYSTEMS

This procedure describes how to configure a network boot server for IBM Power systems using GRUB2.

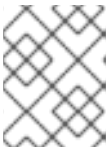
Procedure

1. As root, install the **tftp-server** package:

```
# yum install tftp-server
```

2. Allow incoming connections to the **tftp service** in the firewall:

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

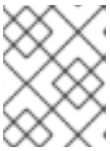


NOTE

This command enables temporary access until the next server reboot. To enable permanent access, add the **--permanent** option to the command.

3. Create a **GRUB2** network boot directory inside the tftp root.

```
# 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
```



NOTE

The command's output informs you of the file name that needs to be configured in your DHCP configuration, described in this procedure.

a. If the PXE server runs on an x86 machine, the **grub2-ppc64-modules** must be installed before creating a **GRUB2** network boot directory inside the tftp root:

```
# yum install grub2-ppc64-modules
```

4. Create a **GRUB2** configuration file: **/var/lib/tftpboot/boot/grub2/grub.cfg**. Below is an example configuration file:

```
set default=0
set timeout=5

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

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

NOTE

The **inst.repo=** option must be used to specify the installation program's image as well as the installation source. The installation program cannot boot without this option.

5. Mount the Binary DVD ISO image using the command:

```
# mount -t iso9660 /path_to_image/name_of_iso/ /mount_point -o loop,ro
```

6. Create a directory and copy the **initrd.img** and **vmlinuz** files from Binary DVD ISO image into it, for example:

```
# cp /mount_point/ppc/ppc64/{initrd.img,vmlinuz} /var/lib/tftpboot/grub2-ppc64/
```

7. Configure your DHCP server to use the boot images packaged with **GRUB2**. A sample configuration in the **/etc/dhcp/dhcpd.conf** file might look like:

```
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. Adjust the sample parameters (**subnet**, **netmask**, **routers**, **fixed-address** and **hardware ethernet**) to fit your network configuration. Note the **file name** parameter; this is the file name that was outputted by the **grub2-mknetdir** command earlier in this procedure.

9. Start and enable the **dhcpd** service:

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

10. Start and enable **xinetd** service that manages the **tftp** service:

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

The PXE boot server is now ready to serve PXE clients. You can start the client (the system to which you are installing Red Hat Enterprise Linux), select **PXE Boot** when prompted to specify a boot source, and start the network installation.

CHAPTER 12. BOOT OPTIONS

This section contains information about some of the boot options that you can use to modify the default behavior of the installation program. For a full list of boot options, see the upstream boot option content.

12.1. TYPES OF BOOT OPTIONS

There are two types of boot options; those with an equals "=" sign, and those without an equals "=" sign. Boot options are appended to the boot command line and multiple options must be separated by a single space. Boot options that are specific to the installation program always start with **inst**.

Options with an equals "=" sign

You must specify a value for boot options that use the **=** symbol. For example, the **inst.vncpassword=** option must contain a value, in this case, a password. The correct syntax for this example is **inst.vncpassword=password**.

Options without an equals "=" sign

This boot option does not accept any values or parameters. For example, the **rd.live.check** option forces the installation program to verify the installation media before starting the installation. If this boot option is present, the verification is performed; if the boot option is not present, the verification is skipped.

12.2. EDITING BOOT OPTIONS

This section contains information about the different ways that you can edit boot options from the boot menu. The boot menu opens after you boot the installation media.

Editing the boot: prompt

When using the **boot:** prompt, the first option must always specify the installation program image file that you want to load. In most cases, you can specify the image using the keyword. You can specify additional options according to your requirements.

Prerequisites

- You have created bootable installation media (USB, CD or DVD).
- You have booted the installation from the media, and the installation boot menu is open.

Procedure

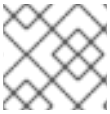
1. With the boot menu open, press the **Esc** key on your keyboard.
2. The **boot:** prompt is now accessible.
3. Press the **Tab** key on your keyboard to display the help commands.
4. Press the **Enter** key on your keyboard to start the installation with your options. To return from the **boot:** prompt to the boot menu, restart the system and boot from the installation media again.

NOTE

The **boot:** prompt also accepts **dracut** kernel options. A list of options is available in the **dracut.cmdline(7)** man page.

Editing the > prompt

You can use the **>** prompt to edit predefined boot options. For example, select **Test this media and install Red Hat Enterprise Linux 8.0.0** from the boot menu to display a full set of options.



NOTE

This procedure is for BIOS-based AMD64 and Intel 64 systems.

Prerequisites

- You have created bootable installation media (USB, CD or DVD).
- You have booted the installation from the media, and the installation boot menu is open.

Procedure

1. From the boot menu, select an option and press the **Tab** key on your keyboard. The **>** prompt is accessible and displays the available options.
2. Append the options that you require to the **>** prompt.
3. Press the **Enter** key on your keyboard to start the installation.
4. Press the **Esc** key on your keyboard to cancel editing and return to the boot menu.

Editing the GRUB2 menu

The GRUB2 menu is available on UEFI-based AMD64, Intel 64, and 64-bit ARM systems.

Prerequisites

- You have created bootable installation media (USB, CD or DVD).
- You have booted the installation from the media, and the installation boot menu is open.

Procedure

1. From the boot menu window, select an option and press the **e** key on your keyboard.
2. When you finish editing, press **F10** or **Ctrl+X** on your keyboard to start the installation using the specified options.

12.3. INSTALLATION SOURCE BOOT OPTIONS

This section contains information about the various installation source boot options.

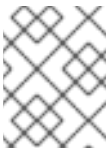
inst.repo=

The **inst.repo=** boot option specifies the installation source, that is, the location of images and packages. For example: **inst.repo=cdrom**. The target of the **inst.repo=** option must be one of the following installation media:

- an installable tree, which is a directory structure containing the installation program images, packages, and repository data as well as a valid **.treeinfo** file
- a DVD (a physical disk present in the system DVD drive)
- an ISO image of the full Red Hat Enterprise Linux installation DVD, placed on a hard drive or a network location accessible to the system.
  You can use the **inst.repo=** boot option to configure different installation methods using different formats. The following table contains details of the **inst.repo=** boot option syntax:

Table 12.1. Installation source boot options

Installation source	Boot option format
Any CD/DVD drive	**inst.repo=cdrom**
Specific CD/DVD drive	**inst.repo=cdrom:device**
Hard Drive	**inst.repo=hd:device:/path**
HMC	**inst.repo=hmc**
HTTP Server	**inst.repo=http://host/path**
HTTPS Server	**inst.repo=https://host/path**
NFS Server	**inst.repo=nfs:[options:]server:/path**



NOTE

The NFS Server option uses NFS protocol version 3 by default. To use a different version, add **+nfsvers=X** to the option.

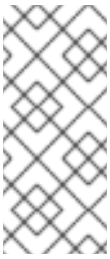
You can set disk device names with the following formats:

- Kernel device name, for example /**dev**/**sda1** or **sdb2**
- File system label, for example **LABEL=Flash** or **LABEL=RHEL8**
- File system UUID, for example **UUID=8176c7bf-04ff-403a-a832-9557f94e61db**
  Non-alphanumeric characters must be represented as **xNN**, where *NN* is the hexadecimal representation of the character. For example, **x20** is a white space **(" ")**.

inst.stage2=

The **inst.stage2=** boot option specifies the location of the installation program runtime image. This option expects a path to a directory containing a valid **.treeinfo** file. The location of the runtime image is read from the **.treeinfo** file. If the **.treeinfo** file is not available, the installation program attempts to load the image from **LiveOS**/**squashfs.img**.
You can use the **inst.stage2=** boot option multiple times to specify multiple HTTP or HTTPS sources. For example:

```
inst.stage2=host1/install.img inst.stage2=host2/install.img
inst.stage2=host3/install.img
```



NOTE

By default, the **inst.stage2=** boot option is used on the installation media and is set to a specific label, for example, **inst.stage2=hd:LABEL=RHEL-8-0-0-BaseOS-x86_64**. If you modify the default label of the file system containing the runtime image, or if you use a customized procedure to boot the installation system, you must verify that the **inst.stage2=** boot option is set to the correct value.

inst.dd=

The **inst.dd=** boot option is used to perform a driver update during the installation. See the *Performing an advanced RHEL installation* document for information on how to update drivers during installation.

inst.repo=hmc

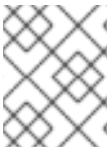
When booting from a Binary DVD, the installation program prompts you to enter additional kernel parameters. To set the DVD as an installation source, append **inst.repo=hmc** to the kernel parameters. The installation program then enables **SE** and **HMC** file access, fetches the images for stage2 from the DVD, and provides access to the packages on the DVD for software selection. This option eliminates the requirement of an external network setup and expands the installation options.

Additional resources

- For a full list of boot options, see the upstream boot option content.

12.4. NETWORK BOOT OPTIONS

This section contains information about commonly used network boot options.

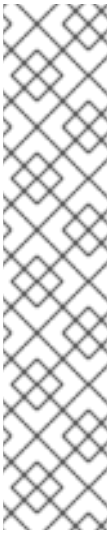


NOTE

Initial network initialization is handled by **dracut**. For a complete list, see the **dracut.cmdline(7)** man page.

ip=

Use the **ip=** boot option to configure one or more network interfaces. To configure multiple interfaces, you can use the **ip** option multiple times - once for each interface. If you configure multiple interfaces, you must specify a primary boot interface using the **bootdev** option. Alternatively, you can use the **ip** option once, and then use Kickstart to set up further interfaces. This option accepts several different formats. The following tables contain information about the most common options.



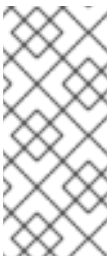
NOTE

In the following tables:

- The **ip** parameter specifies the client IP address. You can specify IPv6 addresses in square brackets, for example, [2001:DB8::1].
- The **gateway** parameter is the default gateway. IPv6 addresses are also accepted.
- The **netmask** parameter is the netmask to be used. This can be either a full netmask (for example, 255.255.255.0) or a prefix (for example, 64).
- The **hostname** parameter is the host name of the client system. This parameter is optional.

Table 12.2. Network interface configuration boot option formats

Configuration method	Boot option format
Automatic configuration of any interface	**ip=method**
Automatic configuration of a specific interface	**ip=interface:method**
Static configuration	**ip=ip::gateway:netmask:hostname:interface:none**
Automatic configuration of a specific interface with an override	**ip=ip::gateway:netmask:hostname:interface:method:mtu**



NOTE

The method **automatic configuration of a specific interface with an override** brings up the interface using the specified method of automatic configuration, such as **dhcp**, but overrides the automatically-obtained IP address, gateway, netmask, host name or other specified parameters. All parameters are optional, so specify only the parameters that you want to override.

The **method** parameter can be any of the following:

Table 12.3. Automatic interface configuration methods

Automatic configuration method	Value
DHCP	**dhcp**
IPv6 DHCP	**dhcp6**
IPv6 automatic configuration	**auto6**
iSCSI Boot Firmware Table (iBFT)	**ibft**

NOTE

- If you use a boot option that requires network access, such as **inst.ks=http://host:/path**, without specifying the ip option, the installation program uses **ip=dhcp**.
- To connect to an iSCSI target automatically, you must activate a network device for accessing the target. The recommended way to activate a network is to use the **ip=ibft** boot option.

nameserver=

The **nameserver=** option specifies the address of the name server. You can use this option multiple times.

rd.neednet=

Typically, **ip=** options are applied only if the network is required by the installation (based on any other boot options that are used). You can use **rd.neednet=1** to explicitly force the application of **ip=** options.

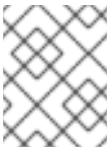
bootdev=

The **bootdev=** option specifies the boot interface. This option is mandatory if you use more than one **ip** option.

ifname=

The **ifname=** options assigns an interface name to a network device with a given MAC address. You can use this option multiple times. The syntax is **ifname=interface:MAC**. For example:

```
ifname=eth0:01:23:45:67:89:ab
```



NOTE

The **ifname=** option is the only supported way to set custom network interface names during installation.

inst.dhcpclass=

The **inst.dhcpclass=** option specifies the DHCP vendor class identifier. The **dhcpd** service sees this value as **vendor-class-identifier**. The default value is **anaconda-$(uname -srm)**.

inst.waitfornet=

Using the **inst.waitfornet=SECONDS** boot option causes the installation system to wait for network connectivity before installation. The value given in the **SECONDS** argument specifies the maximum amount of time to wait for network connectivity before timing out and continuing the installation process even if network connectivity is not present.

vlan=

Use the **vlan=** option to configure a Virtual LAN (VLAN) device on a specified interface with a given name. The syntax is **vlan=name:interface**. For example:

```
vlan=vlan5:em1
```

This configures a VLAN device named **vlan5** on the em1 interface. The name can take the following forms:

Table 12.4. VLAN device naming conventions

Naming scheme	Example
VLAN_PLUS_VID	**vlan0005**
VLAN_PLUS_VID_NO_PAD	**vlan5**
DEV_PLUS_VID	**em1.0005**
DEV_PLUS_VID_NO_PAD	**em1.5**

bond=

Use the **bond=** option to configure a bonding device with the following syntax: **bond=name[:slaves][:options]**. Replace *name* with the bonding device name, *slaves* with a comma-separated list of physical (ethernet) interfaces, and *options* with a comma-separated list of bonding options. For example:

```
bond=bond0:em1,em2:mode=active-backup,tx_queues=32,downdelay=5000
```

For a list of available options, execute the **modinfo** bonding command. Using this option without any parameters assumes **bond=bond0:eth0,eth1:mode=balance-rr**.

team=

Use the **team=** option to configure a team device with the following syntax: **team=master:slaves**. Replace *master* with the name of the master team device and *slaves* with a comma-separated list of physical (ethernet) devices to be used as slaves in the team device. For example:

```
team=team0:em1,em2
```

Additional resources

- For a full list of boot options, see the upstream boot option content.
- For more information about networking, see the *Configuring and managing networking* document.

12.5. CONSOLE BOOT OPTIONS

This section contains information about configuring boot options for your console, monitor display, and keyboard.

console=

Use the **console=** option to specify a device that you want to use as the primary console. For example, to use a console on the first serial port, use **console=ttyS0**. Use this option in conjunction with the **inst.text** option. You can use the **console=** option multiple times. If you do, the boot message is displayed on all specified consoles, but only the last one is used by the installation program. For example, if you specify **console=ttyS0 console=ttyS1**, the installation program uses **ttyS1**.

noshell

Use the **noshell** option to disable access to the root shell during installation. This is useful with Kickstart installations.

inst.lang=

Use the **inst.lang=** option to set the language that you want to use during the installation. **locale -a** or **localectl list-locales** returns a list of locales.

inst.geoloc=

Use the **inst.geoloc=** option to configure geolocation usage in the installation program. Geolocation is used to preset the language and time zone, and uses the following syntax: **inst.geoloc=value**. The **value** can be any of the following parameters:

Table 12.5. Values for the inst.geoloc boot option

Value	Boot option format
Disable geolocation	**inst.geoloc=0**
Use the Fedora GeoIP API	**inst.geoloc=provider_fedora_geoip**
Use the Hostip.info GeoIP API	**inst.geoloc=provider_hostip**

If you do not specify the **inst.geoloc=** option, the installation program uses **provider_fedora_geoip**.

inst.keymap=

Use the **inst.keymap=** option to specify the keyboard layout that you want to use for the installation.

inst.text

Use the **inst.text** option to force the installation program to run in text mode instead of graphical mode.

inst.cmdline

Use the **inst.cmdline** option to force the installation program to run in command-line mode. This mode does not allow any interaction, and you must specify all options in a Kickstart file or on the command line.

inst.graphical

Use the **inst.graphical** option to force the installation program to run in graphical mode. This mode is the default.

inst.resolution=

Use the **inst.resolution=** option to specify the screen resolution in graphical mode. The format is **NxM**, where *N* is the screen width and *M* is the screen height (in pixels). The lowest supported resolution is 800x600.

inst.xdriver=

Use the **inst.xdriver=** option to specify the name of the X driver that you want to use both during installation and on the installed system.

inst.usefbx

Use the **inst.usefbx** option to prompt the installation program to use the frame buffer X driver instead of a hardware-specific driver. This option is equivalent to **inst.xdriver=fbdev**.

modprobe.blacklist=

Use the **modprobe.blacklist=** option to blacklist or completely disable one or more drivers. Drivers (mods) that you disable using this option cannot load when the installation starts, and after the

installation finishes, the installed system retains these settings. You can find a list of the blacklisted drivers in the **/etc/modprobe.d/** directory. Use a comma-separated list to disable multiple drivers. For example:

```
modprobe.blacklist=ahci,firewire_ohci
```

inst.sshd=0

By default, the **sshd** option is automatically started only on the IBM Z architecture. On other architectures, **sshd** is not started unless you use the **inst.sshd** option. Use the **inst.sshd=0** option to prevent **sshd** from starting automatically on IBM Z.

inst.sshd

Use the **inst.sshd** option to start the **sshd** service during installation, so that you can connect to the system during the installation using SSH, and monitor the installation progress. For more information about SSH, see the **ssh(1)** man page. By default, the **sshd** option is automatically started only on the IBM Z architecture. On other architectures, **sshd** is not started unless you use the **inst.sshd** option.

NOTE

During installation, the root account has no password by default. You can set a root password during installation with the **sshpw** Kickstart command.

inst.kdump_addon=

Use the **inst.kdump_addon=** option to enable or disable the Kdump configuration screen (add-on) in the installation program. This screen is enabled by default; use **inst.kdump_addon=off** to disable it. Disabling the add-on disables the Kdump screens in both the graphical and text-based interface as well as the **%addon com_redhat_kdump** Kickstart command.

Additional resources

- For a full list of boot options, see the upstream boot option content.

12.6. DEBUG BOOT OPTIONS

This section contains information about the options that you can use when debugging issues.

inst.rescue=

Use the **inst.rescue=** option to run the rescue environment. The option is useful for trying to diagnose and fix systems.

inst.updates=

Use the **inst.updates=** option to specify the location of the **updates.img** file that you want to apply during installation. There are a number of sources for the updates.

Updates from a network

The easiest way to use **inst.updates=** is to specify the network location of **updates.img**. This does not require any modification to the installation tree. To use this method, edit the kernel command line to include **inst.updates**, for example:

```
inst.updates=http://some.website.com/path/to/updates.img
```

Updates from a disk image

You can also save an **updates.img** on a floppy drive or a USB key. This can be done only with an **ext2** filesystem type of **updates.img**. To save the contents of the image on your floppy drive, insert the floppy disc and run the following command:

```
dd if=updates.img of=/dev/fd0 bs=72k count=20
```

To use a USB key or flash media, replace **/dev/fd0** with the device name of your USB key.

Updates from an installation tree

If you are using a CD, hard drive, or HTTP install, you can save the **updates.img** in the installation tree so that all installations can detect the .img. Save the file in the **images/** directory. The file name must be **updates.img**.
For NFS installs, there are two options: You can either save the image in the **images/** directory, or in the **RHupdates/** directory in the installation tree.

inst.loglevel=

Use the **inst.loglevel=** option to specify the minimum level of messages logged on a terminal. This concerns only terminal logging; log files always contain messages of all levels. Possible values for this option from the lowest to highest level are: **debug**, **info**, **warning**, **error** and **critical**. The default value is **info**, which means that by default, the logging terminal displays messages ranging from **info** to **critical**.

inst.syslog=

When installation starts, the **inst.syslog=** option sends log messages to the **syslog** process on the specified host. The remote **syslog** process must be configured to accept incoming connections.

inst.virtiolog=

Use the **inst.virtiolog=** option to specify the virtio port (a character device at **/dev/virtio-ports/name**) that you want to use for forwarding logs. The default value is **org.fedoraproject.anaconda.log.0**; if this port is present, it is used.

inst.nokill

The **inst.nokill** option is a debugging option that prevents the installation program from rebooting when a fatal error occurs, or at the end of the installation process. Use the **inst.nokill** option to capture installation logs which would be lost upon reboot.

Additional resources

- For a full list of boot options, see the upstream boot option content.

12.7. KICKSTART BOOT OPTIONS

This section contains information about the Kickstart boot options.

inst.ks=

Use the **inst.ks=** boot option to define the location of a Kickstart file that you want to use to automate the installation. You can then specify locations using any of the **inst.repo** formats. You can use the option multiple times to specify multiple HTTP, HTTPS and FTP sources. If you specify multiple HTTP, HTTPS and FTP locations, the locations are run sequentially until one succeeds, for example:

```
inst.ks=host1/directory/ks.cfg inst.ks=host2/directory/ks.cfg inst.ks=host3/directory/ks.cfg
```

If you specify a device and not a path, the installation program looks for the Kickstart file in **/ks.cfg** on the device that you specify. If you use this option without specifying a device, the installation program uses the following option:

```
inst.ks=nfs:next-server:/filename
```

In the previous example, *next-server* is the DHCP next-server option or the IP address of the DHCP server itself, and *filename* is the DHCP filename option, or */kickstart/*. If the given file name ends with the / character, **ip-kickstart** is appended. The following table contains an example.

Table 12.6. Default Kickstart file location

DHCP server address	Client address	Kickstart file location
192.168.122.1	192.168.122.100	192.168.122.1:/kickstart/192.168.122.100-kickstart

If a volume with a label of **OEMDRV** is present, the installation program attempts to load a Kickstart file named **ks.cfg**. If your Kickstart file is in this location, you do not need to use the **inst.ks=** boot option.

inst.ks.all

All locations of type **http**, **https** or **ftp** that you specify with **inst.ks** are used sequentially until the Kickstart file is fetched. Other locations are ignored.

inst.ks.sendmac

Use the **inst.ks.sendmac** option to add headers to outgoing HTTP requests that contain the MAC addresses of all network interfaces. For example:

```
X-RHN-Provisioning-MAC-0: eth0 01:23:45:67:89:ab
```

This can be useful when using **inst.ks=http** to provision systems.

inst.ks.sendsn

Use the **inst.ks.sendsn** option to add a header to outgoing HTTP requests. This header contains the system serial number, read from **/sys/class/dmi/id/product_serial**. The header has the following syntax:

```
X-System-Serial-Number: R8VA23D
```

Additional resources

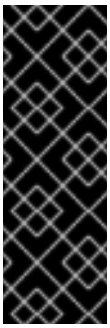
- For a full list of boot options, see the upstream boot option content.

12.8. ADVANCED INSTALLATION BOOT OPTIONS

This section contains information about advanced installation boot options.

inst.kexec

The **inst.kexec** option allows the installation program to use the **kexec** system call at the end of the installation, instead of performing a reboot. The **inst.kexec** option loads the new system immediately, and bypasses the hardware initialization normally performed by the BIOS or firmware.



IMPORTANT

This option is deprecated and available as a Technology Preview only. For information on Red Hat scope of support for Technology Preview features, see the Technology Preview Features Support Scope document.

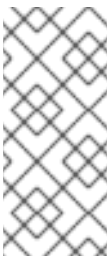
When **kexec** is used, device registers which would normally be cleared during a full system reboot, might stay filled with data, which could potentially create issues for some device drivers.

inst.multilib

Use the **inst.multilib** boot option to configure the system for multilib packages, that is, to allow installing 32-bit packages on a 64-bit AMD64 or Intel 64 system. Normally, on an AMD64 or Intel 64 system, only packages for this architecture (marked as x86_64) and packages for all architectures (marked as noarch) are installed. When you use the **inst.multilib** boot option, packages for 32-bit AMD or Intel systems (marked as i686) are automatically installed.
This applies only to packages directly specified in the **%packages** section. If a package is installed as a dependency, only the exact specified dependency is installed. For example, if you are installing the **bash** package which depends on the **glibc** package, the former is installed in multiple variants, while the latter is installed only in variants that the bash package requires.

selinux=0

By default, the **selinux=0** boot option operates in permissive mode in the installation program, and in enforcing mode in the installed system. The **selinux=0** boot option disables the use of SELinux in the installation program and the installed system.



NOTE

The **selinux=0** and **inst.selinux=0** options are not the same. The **selinux=0** option disables the use of SELinux in the installation program and the installed system. The **inst.selinux=0** option disables SELinux only in the installation program. By default, SELinux operates in permissive mode in the installation program, so disabling SELinux has little effect.

inst.zram

The **inst.zram** option controls the usage of zRAM swap during installation. The option creates a compressed block device inside the system RAM and uses it for swap space instead of the hard drive. This allows the installation program to run with less available memory than is possible without compression, and it might also make the installation faster. By default, swap on zRAM is enabled on systems with 2 GiB or less RAM, and disabled on systems with more than 2 GiB of memory. You can use this option to change this behavior; on a system with more than 2 GiB RAM, use **inst.zram=1** to enable the feature, and on systems with 2 GiB or less memory, use **inst.zram=0** to disable the feature.

Additional resources

- For a full list of boot options, see the upstream boot option content.

PART III. KICKSTART REFERENCES

APPENDIX A. KICKSTART SCRIPT FILE FORMAT REFERENCE

This reference describes in detail the kickstart file format.

A.1. KICKSTART FILE FORMAT

Kickstart scripts are plain text files that contain keywords recognized by the installation program, which serve as directions for the installation. Any text editor able to save files as ASCII text, such as **Gedit** or **vim** on Linux systems or **Notepad** on Windows systems, can be used to create and edit Kickstart files. The file name of your Kickstart configuration does not matter; however, it is recommended to use a simple name as you will need to specify this name later in other configuration files or dialogs.

Commands

Commands are keywords that serve as directions for installation. Each command must be on a single line. Commands can take options. Specifying commands and options is similar to using Linux commands in shell.

Sections

Certain special commands that begin with the percent **%** character start a section. Interpretation of commands in sections is different from commands placed outside sections. Every section must be finished with **%end** command.

Section types

The available sections are:

- **Add-on sections**. These sections use the **%addon *addon_name*** command.
- **Package selection sections**. Starts with **%packages**. Use it to list packages for installation, including indirect means such as package groups or modules.
- **Script sections**. These start with **%pre**, **%pre-install**, **%post**, and **%onerror**. These sections are not required.

Command section

The command section is a term used for the commands in the Kickstart file that are not part of any script section or `%packages ` section.

Script section count and ordering

All sections except the command section are optional and can be present multiple times. When a particular type of script section is to be ecvaluated, all sections of that type present in the Kickstart are evaluated in order of appearance: two **%post** sections are evaluated one after another, in the order as they appear. However, you do not have to specify the various types of script sections in any order: it does not matter if there are **%post** sections before **%pre** sections.

Comments

Kickstart comments are lines starting with the hash **#** character. These lines are ignored by the installation program.

Items that are not required can be omitted. Omitting any required item results in the installation program changing to the interactive mode so that the the user can provide an answer to the related item, just as during a regular interactive installation. It is also possible to declare the kickstart script as non-interactive with the **cmdline** command. In non-interactive mode, any missing answer aborts the installation process.

A.2. PACKAGE SELECTION IN KICKSTART

Kickstart uses sections started by the **%packages** command for selecting packages to install. You can install packages, groups, environments, module streams, and module profiles this way.

A.2.1. Package selection section

Use the **%packages** command to begin a Kickstart section which describes the software packages to be installed. The **%packages** section must end with the **%end** command.

You can specify packages by environment, group, module stream, module profile, or by their package names. Several environments and groups that contain related packages are defined. See the ***repository*/repodata/*-comps-*repository.architecture*.xml** file on the Red Hat Enterprise Linux 8 Installation DVD for a list of environments and groups.

The ***-comps-*repository.architecture*.xml** file contains a structure describing available environments (marked by the **<environment>** tag) and groups (the **<group>** tag). Each entry has an ID, user visibility value, name, description, and package list. If the group is selected for installation, the packages marked **mandatory** in the package list are always installed, the packages marked **default** are installed if they are not specifically excluded elsewhere, and the packages marked **optional** must be specifically included elsewhere even when the group is selected.

You can specify a package group or environment using either its ID (the **<id>** tag) or name (the **<name>** tag).

If you are not sure what package should be installed, Red Hat recommends you to select the **Minimal Install** environment. **Minimal Install** provides only the packages which are essential for running Red Hat Enterprise Linux 8. This will substantially reduce the chance of the system being affected by a vulnerability. If necessary, additional packages can be added later after the installation. For more details on **Minimal Install**, see the Installing the Minimum Amount of Packages Required section of the *Security Hardening* document. Note that **Initial Setup** can not run after a system is installed from a Kickstart file unless a desktop environment and the X Window System were included in the installation and graphical login was enabled.

IMPORTANT

To install a 32-bit package on a 64-bit system:

- specify the **--multilib** option for the **%packages** section
- append the package name with the 32-bit architecture for which the package was built; for example, **glibc.i686**

A.2.2. Package selection commands

These commands can be used within the **%packages** section of a Kickstart file.

Specifying an environment

Specify an entire environment to be installed as a line starting with the **@^** symbols:

```
%packages
@^Infrastructure Server
%end
```

This installs all packages which are part of the **Infrastructure Server** environment. All available environments are described in the ***repository*/repodata/*-comps-*repository.architecture*.xml** file on the Red Hat Enterprise Linux 8 Installation DVD.

Only a single environment should be specified in the Kickstart file. If more environments are specified, only the last specified environment is used.

Specifying groups

Specify groups, one entry to a line, starting with an **@** symbol, and then the full group name or group id as given in the ***-comps-*repository.architecture*.xml** file. For example:

```
%packages
@X Window System
@Desktop
@Sound and Video
%end
```

The **Core** group is always selected – it is not necessary to specify it in the **%packages** section.

Specifying individual packages

Specify individual packages by name, one entry to a line. You can use the asterisk character (*) as a wildcard in package names. For example:

```
%packages
sqlite
curl
aspell
docbook*
%end
```

The **docbook*** entry includes the packages **docbook-dtds** and **docbook-style** that match the pattern represented with the wildcard.

Specifying profiles of module streams

Specify profiles for module streams, one entry to a line, using the syntax for profiles:

```
%packages
@module:stream/profile
%end
```

This installs all packages listed in the specified profile of the module stream.

- When a module has a default stream specified, you can leave it out. When the default stream is not specified, you must specify it.
- When a module stream has a default profile specified, you can leave it out. When the default profile is not specified, you must specify it.
- Installing a module multiple times with different streams is not possible.
- Installing multiple profiles of the same module and stream is possible.

Modules and groups use the same syntax starting with the **@** symbol. When a module and a package group exist with the same name, the module takes precedence.

In Red Hat Enterprise Linux 8, modules are present only in the AppStream repository. To list available modules, use the **yum module list** command on an installed Red Hat Enterprise Linux 8 system.

It is also possible to enable module streams using the **module** Kickstart command and then install packages contained in the module stream by naming them directly.

Excluding environments, groups, or packages

Use a leading dash (**-**) to specify packages or groups to exclude from the installation. For example:

```
%packages
-@Graphical Administration Tools
-autofs
-ipa*compat
%end
```

IMPORTANT

Installing all available packages using only * in a Kickstart file is not supported.

You can change the default behavior of the **%packages** section by using several options. Some options work for the entire package selection, others are used with only specific groups.

Additional resources

- For more information about handling packages, see the Installing software with yum chapter of the *Configuring basic system settings* document.
- For more information about modules and streams, see the Installing, managing, and removing user space components document.

A.2.3. Common package selection options

The following options are available for the **%packages** sections. To use an option, append it to the start of the package selection section. For example:

```
%packages --multilib --ignoremissing
```

--default

Install the default set of packages. This corresponds to the package set which would be installed if no other selections were made in the **Package Selection** screen during an interactive installation.

--excludedocs

Do not install any documentation contained within packages. In most cases, this excludes any files normally installed in the **/usr/share/doc** directory, but the specific files to be excluded depend on individual packages.

--ignoremissing

Ignore any packages, groups, module streams, module profiles, and environments missing in the installation source, instead of halting the installation to ask if the installation should be aborted or continued.

--instLangs=

Specify a list of languages to install. Note that this is different from package group level selections. This option does not describe which package groups should be installed; instead, it sets RPM macros controlling which translation files from individual packages should be installed.

--multilib

Configure the installed system for multilib packages (that is, to allow installing 32-bit packages on a 64-bit system) and install packages specified in this section as such.
Normally, on an AMD64 and Intel 64 system, only packages for this architecture (marked as **x86_64**) and packages for all architectures (marked as **noarch**) would be installed. When you use this option, packages for 32-bit AMD and Intel systems (marked as **i686**) are automatically installed as well, if available.

This only applies to packages explicitly specified in the **%packages** section. Packages which are only being installed as dependencies without being specified in the Kickstart file are only installed in architecture versions in which they are needed, even if they are available for more architectures.

--nocore

Disables installation of the **@Core** package group which is otherwise always installed by default. Disabling the **@Core** package group should be only used for creating lightweight containers; installing a desktop or server system with **--nocore** will result in an unusable system.

NOTE

Using **-@Core** to exclude packages in the **@Core** package group does not work. The only way to exclude the **@Core** package group is with the **--nocore** option.

--excludeWeakdeps

Disables installation of packages from weak dependencies. These are packages linked to the selected package set by Recommends and Supplements flags. By default weak dependencies will be installed.

--retries=

Sets the number of times Yum will attempt to download packages (retries). The default value is 10. This option only applies during the installation, and will not affect Yum configuration on the installed system.

--timeout=

Sets the Yum timeout in seconds. The default value is 30. This option only applies during the installation, and will not affect Yum configuration on the installed system.

A.2.4. Options for specific package groups

The options in this list only apply to a single package group. Instead of using them at the **%packages** command in the Kickstart file, append them to the group name. For example:

```
%packages
@Graphical Administration Tools --optional
%end
```

--nodefaults

Only install the group's mandatory packages, not the default selections.

--optional

Install packages marked as optional in the group definition in the ***-comps-*repository*.*architecture*.xml** file, in addition to installing the default selections.
Note that some package groups, such as **Scientific Support**, do not have any mandatory or default packages specified - only optional packages. In this case the **--optional** option must always be used, otherwise no packages from this group will be installed.

A.3. PRE-INSTALLATION SCRIPTS IN KICKSTART

Pre-installation scripts are run immediately before installation begins.

A.3.1. Pre-installation script section

The **%pre** scripts are run on the system immediately after the Kickstart file has been loaded, but before it is completely parsed and installation begins. Each of these sections must start with **%pre** and end with **%end**.

The **%pre** script can be used for activation and configuration of networking and storage devices. It is also possible to run scripts, using interpreters available in the installation environment. Adding a **%pre** script can be useful if you have networking and storage that needs special configuration before proceeding with the installation, or have a script that, for example, sets up additional logging parameters or environment variables.

Debugging problems with **%pre** scripts can be difficult, so it is recommended only to use a **%pre** script when necessary.

Commands related to networking, storage, and file systems are available to use in the **%pre** script, in addition to most of the utilities in the installation environment **/sbin** and **/bin** directories.

You can access the network in the **%pre** section. However, the name service has not been configured at this point, so only IP addresses work, not URLs.

The **%pre** scripts ignore missing files for the **%include** commands. This is useful for generating the included files in the **%pre** section, and having them loaded later.

NOTE

Unlike the post-installation script, the pre-installation script is not run in the chroot environment.

A.3.2. Pre-installation Kickstart section options

The following options can be used to change the behavior of pre-installation scripts. To use an option, append it to the **%pre** line at the beginning of the script. For example:

```
%pre --interpreter=/usr/bin/python
-- Python script omitted --
%end
```

--interpreter=

Allows you to specify a different scripting language, such as Python. Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/bin/python**.

--erroronfail

Display an error and halt the installation if the script fails. The error message will direct you to where the cause of the failure is logged.

--log=

Logs the script's output into the specified log file. For example:

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

A.4. POST-INSTALLATION SCRIPTS IN KICKSTART

Post-installation scripts are run after the installation is complete, but before the system is rebooted for the first time. You can use this section to run tasks such as system subscription.

A.4.1. Post-installation script section

You have the option of adding commands to run on the system once the installation is complete, but before the system is rebooted for the first time. This section must start with **%post** and end with **%end**.

The **%post** section is useful for functions such as installing additional software or configuring an additional name server. The post-install script is run in a **chroot** environment, therefore, performing tasks such as copying scripts or RPM packages from the installation media do not work by default. You can change this behavior using the **--nochroot** option as described below. Then the **%post** script will run in the installation environment, not in **chroot** on the installed target system.

Because post-install script runs in a **chroot** environment, most **systemctl** commands will refuse to perform any action. For more information, see the Behavior of systemctl in a chroot Environment section of the *Configuring and managing system administration* document.

Note that during execution of the **%post** section, the installation media must be still inserted.

IMPORTANT

If you configured the network with static IP information, including a name server, you can access the network and resolve IP addresses in the **%post** section. If you configured the network for DHCP, the **/etc/resolv.conf** file has not been completed when the installation executes the **%post** section. You can access the network, but you cannot resolve IP addresses. Thus, if you are using DHCP, you must specify IP addresses in the **%post** section.

A.4.2. Post-installation Kickstart section options

The following options can be used to change the behavior of post-installation scripts. To use an option, append it to the **%post** line at the beginning of the script. For example:

```
%post --interpreter=/usr/bin/python3
-- Python script omitted --
%end
```

--interpreter=

Allows you to specify a different scripting language, such as Python. For example:

```
%post --interpreter=/usr/bin/python3
```

Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/bin/python3**.

--nochroot

Allows you to specify commands that you would like to run outside of the chroot environment. The following example copies the file /etc/resolv.conf to the file system that was just installed.

```
%post --nochroot
cp /etc/resolv.conf /mnt/sysimage/etc/resolv.conf
%end
```

--erroronfail

Display an error and halt the installation if the script fails. The error message will direct you to where the cause of the failure is logged.

--log=

Logs the script's output into the specified log file. Note that the path of the log file must take into account whether or not you use the **--nochroot** option. For example, without **--nochroot**:

```
%post --log=/root/ks-post.log
```

and with **--nochroot**:

```
%post --nochroot --log=/mnt/sysimage/root/ks-post.log
```

A.4.3. Example: Mounting NFS in a post-install script

This example of a **%post** section mounts an NFS share and executes a script named **runme** located at **/usr/new-machines/** on the share. Note that NFS file locking is not supported while in Kickstart mode, therefore the **-o nolock** option is required.

```
# 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. Example: Running subscription-manager as a post-install script

One of the most common uses of post-installation scripts in Kickstart installations is automatic registration of the installed system using Red Hat Subscription Manager. The following is an example of automatic subscription in a **%post** script:

```
%post --log=/root/ks-post.log
subscription-manager register --username=admin@example.com --password=secret --auto-attach
%end
```

The subscription-manager command-line script registers a system to a Red Hat Subscription Management server (Customer Portal Subscription Management, Satellite 6, or CloudForms System Engine). This script can also be used to assign or attach subscriptions automatically to the system that best-match that system. When registering to the Customer Portal, use the Red Hat Network login credentials. When registering to Satellite 6 or CloudForms System Engine, you may also need to specify more subscription-manager options like **--serverurl**, **--org**, **--environment** as well as credentials provided by your local administrator. Note that credentials in the form of an **--org --activationkey** combination is a good way to avoid exposing **--username --password** values in shared kickstart files.

Additional options can be used with the registration command to set a preferred service level for the system and to restrict updates and errata to a specific minor release version of RHEL for customers with Extended Update Support subscriptions that need to stay fixed on an older stream.

See also the How do I use subscription-manager in a kickstart file? article on the Red Hat Customer Portal for additional information about using **subscription-manager** in a Kickstart **%post** section.

A.5. ANACONDA CONFIGURATION SECTION

Additional installation options can be configured in the **%anaconda** section of your Kickstart file. This section controls the behavior of the user interface of the installation system.

This section must be placed towards the end of the Kickstart file, after Kickstart commands, and must start with **%anaconda** and end with **%end**.

Currently, the only command that can be used in the **%anaconda** section is **pwpolicy**.

Example A.1. Sample %anaconda script

The following is an example %anaconda section:

```
%anaconda
pwpolicy root --minlen=10 --strict
%end
```

This example **%anaconda** section sets a password policy which requires that the root password be at least 10 characters long, and strictly forbids passwords which do not match this requirement.

A.6. KICKSTART ERROR HANDLING SECTION

Starting with Red Hat Enterprise Linux 7, Kickstart installations can contain custom scripts which are run when the installation program encounters a fatal error. For example, an error in a package that has been requested for installation, failure to start VNC when specified, or an error when scanning storage devices. Installation cannot continue after such an error has occured. The installation program will run all **%onerror** scripts in the order they are provided in the Kickstart file. In addition, **%onerror** scripts will be run in the event of a traceback.

Each **%onerror** script is required to end with **%end**.

Error handling sections accept the following options:

--erroronfail

Display an error and halt the installation if the script fails. The error message will direct you to where the cause of the failure is logged.

--interpreter=

Allows you to specify a different scripting language, such as Python. For example:

```
%onerror --interpreter=/usr/bin/python
```

Any scripting language available on the system can be used; in most cases, these are **/usr/bin/sh**, **/usr/bin/bash**, and **/usr/bin/python**.

--log=

Logs the script's output into the specified log file.

A.7. KICKSTART ADD-ON SECTIONS

Starting with Red Hat Enterprise Linux 7, Kickstart installations support add-ons. These add-ons can expand the basic Kickstart (and Anaconda) functionality in many ways.

To use an add-on in your Kickstart file, use the **%addon *addon_name options*** command, and finish the command with an **%end** statement, similar to pre-installation and post-installation script sections. For example, if you want to use the Kdump add-on, which is distributed with Anaconda by default, use the following commands:

```
%addon com_redhat_kdump --enable --reserve-mb=auto
%end
```

The **%addon** command does not include any options of its own - all options are dependent on the actual add-on.

APPENDIX B. KICKSTART COMMANDS AND OPTIONS REFERENCE

This reference is a complete list of all Kickstart commands supported by the Red Hat Enterprise Linux installation program program. The commands are sorted alphabetically in a few broad categories. If a command can fall under multiple categories, it is listed in all of them.

B.1. KICKSTART CHANGES

The folowing sections describe the changes in Kickstart commands and options in Red Hat Enterprise Linux 8.

B.1.1. auth or authconfig is deprecated in RHEL 8

The **auth** or **authconfig** Kickstart command is deprecated in Red Hat Enterprise Linux 8 because the **authconfig** tool and package have been removed.

Similarly to **authconfig** commands issued on command line, **authconfig** commands in Kickstart scripts now use the **authselect-compat** tool to run the new **authselect** tool. For a description of this compatibility layer and its known issues, see the manual page **authselect-migration(7)**. The installation program will automatically detect use of the deprecated commands and install on the system the **authselect-compat** package to provide the compatibility layer.

B.1.2. Kickstart no longer supports Btrfs

The Btrfs file system is not supported in Red Hat Enterprise Linux 8. As a result, the Graphical User Interface (GUI) and the Kickstart commands no longer support Btrfs.

B.1.3. Using Kickstart files from previous RHEL releases

If you are using Kickstart files from previous RHEL releases, see the *Repositories* section of this document for more information about the Red Hat Enterprise Linux 8 BaseOS and AppStream repositories.

B.1.4. Deprecated Kickstart comands and options

The following Kickstart commands and options have been deprecated in Red Hat Enterprise Linux 8. Using them in Kickstart files will print a warning in the logs.

- **auth** or **authconfig** - use **authselect instead**
- **device**
- **deviceprobe**
- **dmraid**
- **install** - use the subcommands or methods directly as commands
- **lilo**
- **lilocheck**
- **mouse**

- **multipath**
- **bootloader --upgrade**
- **ignoredisk --interactive**
- **partition --active**
- **reboot --kexec**

Where only specific options are listed, the base command and its other options are still available and not deprecated.

Note also you can turn the deprecated command warnings into errors with the **inst.ksstrict** boot option.

B.1.5. Removed Kickstart comands and options

The following Kickstart commands and options have been completely removed in Red Hat Enterprise Linux 8. Using them in Kickstart files will cause an error.

- **upgrade** (This command had already previously been deprecated.)
- **btrfs**
- **part/partition btrfs**
- **part --fstype btrfs** or **partition --fstype btrfs**
- **logvol --fstype btrfs**
- **raid --fstype btrfs**
- **unsupported_hardware**

Where only specific options and values are listed, the base command and its other options are still available and not removed.

B.1.6. New Kickstart comands and options

The following commands and options have been added in Red Hat Enterprise Linux 8.

RHEL 8.0

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

B.2. KICKSTART COMMANDS FOR INSTALLATION PROGRAM CONFIGURATION AND FLOW CONTROL

The Kickstart commands in this list control the mode and course of installation, and what happens at its end.

B.2.1. autostep

The **autostep** Kickstart command is optional. This option makes the installation program step through every screen, displaying each briefly. Normally, Kickstart installations skip unnecessary screens.

Options

- **--autoscreenshot** - Take a screenshot at every step during installation. These screenshots are stored in **/tmp/anaconda-screenshots/** during the installation, and after the installation finishes you can find them in **/root/anaconda-screenshots**.
  Each screen is only captured right before the installation program switches to the next one. This is important, because if you do not use all required Kickstart options and the installation therefore does not begin automatically, you can go to the screens which were not automatically configured, perform any configuration you want. Then, when you click **Done** to continue, the screen is captured including the configuration you just provided.

Notes

- This option should not be used when deploying a system because it can disrupt package installation.

B.2.2. cdrom

The **cdrom** Kickstart command is optional. It performs the installation from the first optical drive on the system.

Syntax

```
cdrom
```

Notes

- Previously, the **cdrom** command had to be used together with the **install** command. The **install** command has been deprecated and **cdrom** can be used on its own, because it implies **install**.
- This command has no options.
- To actually run the installation, one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, or **url** must be specified.

B.2.3. cmdline

The **cmdline** Kickstart command is optional. It performs the installation in a completely non-interactive command line mode. Any prompt for interaction halts the installation.

Notes

- For a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option. If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from VNC and text mode.
- This mode is useful on IBM Z systems with the x3270 terminal.

B.2.4. driverdisk

The **driverdisk** Kickstart command is optional. Use it to provide additional drivers to the installation program.

Driver disks can be used during Kickstart installations to provide additional drivers not included by default. You must copy the driver disks's contents to the root directory of a partition on the system's hard drive. Then, you must use the **driverdisk** command to specify that the installation program should look for a driver disk and its location.

Syntax

```
driverdisk [partition|--source=url|--biospart=biospart]
```

Options

You must specify the location of driver disk in one way out of these:

- *partition* - Partition containing the driver disk. Note that the partition must be specified as a full path (for example, /**dev**/**sdb1**), *not* just the partition name (for example, **sdb1**).
- **--source=** - URL for the driver disk. Examples include:

```
driverdisk --source=ftp://path/to/dd.img
driverdisk --source=http://path/to/dd.img
driverdisk --source=nfs:host:/path/to/dd.img
```

- **--biospart=** - BIOS partition containing the driver disk (for example, **82p2**).

Notes

Driver disks can also be loaded from a hard disk drive or a similar device instead of being loaded over the network or from **initrd**. Follow this procedure:

1. Load the driver disk on a hard disk drive, a USB or any similar device.
2. Set the label, for example, *DD*, to this device.
3. Add the following line to your Kickstart file:

```
driverdisk LABEL=DD:/e1000.rpm
```

Replace *DD* with a specific label and replace *dd.rpm* with a specific name. Use anything supported by the **inst.repo** command instead of *LABEL* to specify your hard disk drive.

B.2.5. eula

The **eula** Kickstart command is optional. Use this option to accept the End User License Agreement (EULA) without user interaction. Specifying this option prevents Initial Setup from prompting you to accept the license agreement after you finish the installation and reboot the system for the first time. See the Completing initial setup section of the *Performing a standard RHEL installation* document for more information.

Options

- **--agreed** (required) - Accept the EULA. This option must always be used, otherwise the **eula** command is meaningless.

B.2.6. firstboot

The **firstboot** Kickstart command is optional. It determines whether the **Initial Setup** application starts the first time the system is booted. If enabled, the **initial-setup** package must be installed. If not specified, this option is disabled by default.

Options

- **--enable** or **--enabled** - Initial Setup is started the first time the system boots.
- **--disable** or **--disabled** - Initial Setup is not started the first time the system boots.
- **--reconfig** - Enable the Initial Setup to start at boot time in reconfiguration mode. This mode enables the language, mouse, keyboard, root password, security level, time zone and networking configuration options in addition to the default ones.

B.2.7. graphical

The **graphical** Kickstart command is optional. It performs the installation in graphical mode. This is the default.

Syntax

```
graphical options
```

Options

- **--non-interactive** - Performs the installation in a completely non-interactive mode. This mode will terminate the installation when user interaction is required.

Notes

- For a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option. If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from VNC and text mode.

B.2.8. halt

The **halt** Kickstart command is optional.

Halt the system after the installation has successfully completed. This is similar to a manual installation, where Anaconda displays a message and waits for the user to press a key before rebooting. During a Kickstart installation, if no completion method is specified, this option is used as the default.

Notes

- The **halt** command is equivalent to the **shutdown -H** command. For more details, see the *shutdown(8)* man page.
- For other completion methods, see the **poweroff**, **reboot**, and **shutdown** commands.

B.2.9. harddrive

The **harddrive** Kickstart command is optional. It performs the installation from a Red Hat installation tree or full installation ISO image on a local drive. The drive must contain a file system the installation program can mount: **ext2**, **ext3**, **ext4**, **vfat**, or **xfs**.

Syntax

```
harddrive
```

Options

- **--biospart=** - BIOS partition to install from (such as **82**).
- **--partition=** - Partition to install from (such as **sdb2**).
- **--dir=** - Directory containing the ***variant*** directory of the installation tree, or the ISO image of the full installation DVD.

Example

```
harddrive --partition=hdb2 --dir=/tmp/install-tree
```

Notes

- Previously, the **harddrive** command had to be used together with the **install** command. The **install** command has been deprecated and **harddrive** can be used on its own, because it implies **install**.
- To actually run the installation, one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, or **url** must be specified.

B.2.10. install (deprecated)

IMPORTANT

The **install** Kickstart command is deprecated in Red Hat Enterprise Linux 8. Use its methods as separate commands.

The **install** Kickstart command is optional. It specifies the default installation mode.

Syntax

```
install
installation_method
```

Notes

- The **install** command must be followed by an installation method command. The installation method command must be on a separate line.
- The methods include:
  - **cdrom**

- **harddrive**
- **hmc**
- **nfs**
- **liveimg**
- **url**

For details about the methods, see their separate reference pages.

B.2.11. liveimg

The **liveimg** Kickstart command is optional. It performs the installation from a disk image instead of packages.

Syntax

```
liveimg --url=SOURCE [OPTIONS]
```

Mandatory options

- **--url=** - The location to install from. Supported protocols are **HTTP**, **HTTPS**, **FTP**, and **file**.

Optional options

- **--url=** - The location to install from. Supported protocols are **HTTP**, **HTTPS**, **FTP**, and **file**.
- **--proxy=** - Specify an **HTTP**, **HTTPS** or **FTP** proxy to use while performing the installation.
- **--checksum=** - An optional argument with the **SHA256** checksum of the image file, used for verification.
- **--noverifyssl** - Disable SSL verification when connecting to an **HTTPS** server.

Example

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

Notes

- The image can be the **squashfs.img** file from a live ISO image, a compressed tar file (**.tar**, **.tbz**, **.tgz**, **.txz**, **.tar.bz2**, **.tar.gz**, or **.tar.xz**.), or any file system that the installation media can mount. Supported file systems are **ext2**, **ext3**, **ext4**, **vfat**, and **xfs**.
- When using the **liveimg** installation mode with a driver disk, drivers on the disk will not automatically be included in the installed system. If necessary, these drivers should be installed manually, or in the **%post** section of a kickstart script.
- Previously, the **liveimg** command had to be used together with the **install** command. The **install** command has been deprecated and **liveimg** can be used on its own, because it implies **install**.

- To actually run the installation, one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, or **url** must be specified.

B.2.12. logging

The **logging** Kickstart command is optional. It controls the error logging of Anaconda during installation. It has no effect on the installed system.

Syntax

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

Optional options

- **--host=** - Send logging information to the given remote host, which must be running a syslogd process configured to accept remote logging.
- **--port=** - If the remote syslogd process uses a port other than the default, set it using this option.
- **--level=** - Specify the minimum level of messages that appear on tty3. All messages are still sent to the log file regardless of this level, however. Possible values are **debug**, **info**, **warning**, **error**, or **critical**.

B.2.13. mediacheck

The **mediacheck** Kickstart command is optional. This command forces the installation program to perform a media check (**rd.live.check**) before starting the installation. This command requires that installations be attended, so it is disabled by default.

B.2.14. nfs

The **nfs** Kickstart command is optional. It performs the installation from a specified NFS server.

Syntax

```
nfs
```

Options

- **--server=** - Server from which to install (host name or IP).
- **--dir=** - Directory containing the ***variant*** directory of the installation tree.
- **--opts=** - Mount options to use for mounting the NFS export. (optional)

Example

```
nfs --server=nfsserver.example.com --dir=/tmp/install-tree
```

Notes

- Previously, the **nfs** command had to be used together with the **install** command. The **install** command has been deprecated and **nfs** can be used on its own, because it implies **install**.
- To actually run the installation, one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, or **url** must be specified.

B.2.15. ostreesetup

The **ostreesetup** Kickstart command is optional. It is used to set up OStree-based installations.

Syntax

```
ostreesetup --osname OSNAME [--remote REMOTE] --url URL --ref REF [--nogpg]
```

Mandatory options:

- **--osname *OSNAME*** - Management root for OS installation.
- **--url *URL*** - URL of the repository to install from.
- **--ref *REF*** - Name of the branch from the repository to be used for installation.

Optional options:

- **--remote *REMOTE*** - Management root for OS installation.
- **--nogpg** - Disable GPG key verification.

Notes

- For more information about the OStree tools, see the upstream documentation: https://ostree.readthedocs.io/en/latest/

B.2.16. poweroff

The **poweroff** Kickstart command is optional. It shuts down and powers off the system after the installation has successfully completed. Normally during a manual installation, Anaconda displays a message and waits for the user to press a key before rebooting.

Notes

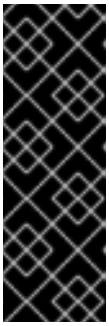
- The **poweroff** option is equivalent to the **shutdown -P** command. For more details, see the *shutdown(8)* man page.
- For other completion methods, see the **halt**, **reboot**, and **shutdown** Kickstart commands. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.
- The **poweroff** command is highly dependent on the system hardware in use. Specifically, certain hardware components such as the BIOS, APM (advanced power management), and ACPI (advanced configuration and power interface) must be able to interact with the system kernel. Consult your hardware documentation for more information on you system's APM/ACPI abilities.

B.2.17. reboot

The **reboot** Kickstart command is optional. It instructs the installation program to reboot after the installation is successfully completed (no arguments). Normally, Kickstart displays a message and waits for the user to press a key before rebooting.

Options

- **--eject** - Attempt to eject the bootable media (DVD, USB, or other media) before rebooting.
- **--kexec** - Uses the **kexec** system call instead of performing a full reboot, which immediately loads the installed system into memory, bypassing the hardware initialization normally performed by the BIOS or firmware.



IMPORTANT

This option is deprecated and available as a Technology Preview only. For information on Red Hat scope of support for Technology Preview features, see the Technology Preview Features Support Scope document.

When **kexec** is used, device registers (which would normally be cleared during a full system reboot) might stay filled with data, which could potentially create issues for some device drivers.

Notes

- Use of the **reboot** option *might* result in an endless installation loop, depending on the installation media and method.
- The **reboot** option is equivalent to the **shutdown -r** command. For more details, see the *shutdown(8)* man page.
- Specify **reboot** to automate installation fully when installing in command line mode on IBM Z.
- For other completion methods, see the **halt**, **poweroff**, and **shutdown** Kickstart options. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.

B.2.18. rescue

The **rescue** Kickstart command is optional. It automatically enters the installation program's rescue mode. This gives you a chance to repair the system in case of any problems.

Syntax

```
rescue [--nomount|--romount]
```

Options

- **--nomount** or **--romount** - Controls how the installed system is mounted in the rescue environment. By default, the installation program finds your system and mount it in read-write mode, telling you where it has performed this mount. You can optionally select to not mount anything (the **--nomount** option) or mount in read-only mode (the **--romount** option). Only one of these two options can be used.

B.2.19. shutdown

The **shutdown** Kickstart command is optional. It shuts down the system after the installation has successfully completed.

Notes

- The **shutdown** Kickstart option is equivalent to the **shutdown** command. For more details, see the *shutdown(8)* man page.
- For other completion methods, see the **halt**, **poweroff**, and **reboot** Kickstart options. The **halt** option is the default completion method if no other methods are explicitly specified in the Kickstart file.

B.2.20. sshpw

The **sshpw** Kickstart command is optional.

During the installation, you can interact with the installation program and monitor its progress over an **SSH** connection. Use the **sshpw** command to create temporary accounts through which to log on. Each instance of the command creates a separate account that exists only in the installation environment. These accounts are not transferred to the installed system.

Syntax

```
sshpw --username=name [options] password
```

Mandatory options

- **--username** - Provides the name of the user. This option is required.
- *password* - The password to use for the user. This option is required.

Optional options

- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use Python:

  ```
  $ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
  (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  This generates a sha512 crypt-compatible hash of your password using a random salt.
- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**
- **--lock** - If this option is present, this account is locked by default. This means that the user will not be able to log in from the console.
- **--sshkey** - If this is option is present, then the *<password>* string is interpreted as an ssh key value.

Notes

- By default, the **ssh** server is not started during the installation. To make **ssh** available during the installation, boot the system with the kernel boot option **inst.sshd**.
- If you want to disable root **ssh** access, while allowing another user **ssh** access, use the following:

```
sshpw --username=example_username example_password --plaintext
sshpw --username=root example_password --lock
```

- To simply disable root **ssh** access, use the following:

```
sshpw --username=root --lock
```

B.2.21. text

The **text** Kickstart command is optional. It performs the Kickstart installation in text mode. Kickstart installations are performed in graphical mode by default.

Syntax

```
text options
```

Options

- **--non-interactive** - Performs the installation in a completely non-interactive mode. This mode will terminate the installation when user interaction is required.

Notes

- Note that for a fully automatic installation, you must either specify one of the available modes (**graphical**, **text**, or **cmdline**) in the Kickstart file, or you must use the **console=** boot option. If no mode is specified, the system will use graphical mode if possible, or prompt you to choose from VNC and text mode.

B.2.22. url

The **url** Kickstart command is optional. It performs the installation from an installation tree image on a remote server using FTP, HTTP, or HTTPS.

Syntax

```
url --url=FROM [OPTIONS]
```

Mandatory options

- **--url=** - The location to install from. Supported protocols are **HTTP**, **HTTPS**, **FTP**, and **file**.

Optional options

- **--mirrorlist=** - The mirror URL to install from.
- **--proxy=** - Specify an **HTTP**, **HTTPS** or **FTP** proxy to use while performing the installation.
- **--noverifyssl** - Disable SSL verification when connecting to an **HTTPS** server.

- **--metalink=*URL*** - Specify the metalink URL to install from. Variable substitution is done for **$releasever** and **$basearch** in the *URL*.
- **--sslcacert=*path/to/SSLCACERT*** - Specify path to the file holding one or more SSL certificates to verify the repository host with.
- **--sslclientcert=*path/to/SSLCLIENTCERT*** - Specify path to the SSL client certificate (PEM file) which should be used to connect to the repository.
- **--sslclientkey=*path/to/SSLCLIENTKEY*** - Specify path to the private key file associated with the client certificate given with the **--sslclientcert** option.

Examples

- To install from a HTTP server:

```
url --url http://server/path
```

- To install from a FTP server:

```
url --url ftp://username:password@server/path
```

- To install from a local file:

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

Notes

- Previously, the **url** command had to be used together with the **install** command. The **install** command has been deprecated and **url** can be used on its own, because it implies **install**.
- To actually run the installation, one of **cdrom**, **harddrive**, **hmc**, **nfs**, **liveimg**, or **url** must be specified.

B.2.23. vnc

The **vnc** Kickstart command is optional. It allows the graphical installation to be viewed remotely through VNC.

This method is usually preferred over text mode, as there are some size and language limitations in text installations. With no additional options, this command starts a VNC server on the installation system with no password and displays the details required to connect to it.

Syntax

```
vnc [--host=host_name] [--port=port] [--password=password]
```

Options

- **--host=** - Connect to the VNC viewer process listening on the given host name.
- **--port=** - Provide a port that the remote VNC viewer process is listening on. If not provided, Anaconda uses the VNC default port of 5900.

- **--password=** - Set a password which must be provided to connect to the VNC session. This is optional, but recommended.

B.2.24. %include

The **%include** Kickstart command is optional.

Use the **%include** ***/path/to/file*** command to include the contents of another file in the Kickstart file as though the contents were at the location of the **%include** command in the Kickstart file.

This inclusion is evaluated only after the **%pre** script sections and can thus be used for files generated by scripts in the **%pre** sections. To include files before evaluation of **%pre** sections, use the **%ksappend** command.

B.2.25. %ksappend

The **%ksappend** Kickstart command is optional.

Use the **%ksappend** ***/path/to/file*** command to include the contents of another file in the Kickstart file as though the contents were at the location of the **%ksappend** command in the Kickstart file.

This inclusion is evaluated before the **%pre** script sections, unlike inclusion with the **%include** command.

B.3. KICKSTART COMMANDS FOR SYSTEM CONFIGURATION

The Kickstart commands in this list configure further details on the resulting system such as users, repositories, or services.

B.3.1. auth or authconfig (deprecated)

IMPORTANT

Use the new **authselect** command instead of the deprecated **auth** or **authconfig** Kickstart command. **auth** and **authconfig** are available only for limited backwards compatibility.

The **auth** or **authconfig** Kickstart command is optional. It sets up the authentication options for the system using the **authconfig** tool, which can also be run on the command line after the installation finishes.

Syntax

```
authconfig [options]
```

Notes

- Previously, the **auth** or **authconfig** Kickstart commands called the **authconfig** tool. This tool has been deprecated in Red Hat Enterprise Linux 8. These Kickstart commands now use the **authselect-compat** tool to call the new **authselect** tool. For a description of the compatibility layer and its known issues, see the manual page *authselect-migration(7)*. The installation program will automatically detect use of the deprecated commands and install on the system the **authselect-compat** package to provide the compatibility layer.

- Passwords are shadowed by default.
- When using OpenLDAP with the **SSL** protocol for security, make sure that the **SSLv2** and **SSLv3** protocols are disabled in the server configuration. This is due to the POODLE SSL vulnerability (CVE-2014-3566). See https://access.redhat.com/solutions/1234843 for details.

B.3.2. authselect

The **authselect** Kickstart command is optional. It sets up the authentication options for the system using the **authselect** command, which can also be run on the command line after the installation finishes.

Syntax

```
authselect [options]
```

Notes

- This command passes all options to the **authselect** command. Refer to the *authselect(8)* manual page and the **authselect --help** command for more details.
- This command replaces the deprecated **auth** or **authconfig** commands deprecated in Red Hat Enterprise Linux 8 together with the **authconfig** tool.
- Passwords are shadowed by default.
- When using OpenLDAP with the **SSL** protocol for security, make sure that the **SSLv2** and **SSLv3** protocols are disabled in the server configuration. This is due to the POODLE SSL vulnerability (CVE-2014-3566). See https://access.redhat.com/solutions/1234843 for details.

B.3.3. firewall

The **firewall** Kickstart command is optional. It specifies the firewall configuration for the installed system.

Syntax

```
firewall --enabled|--disabled device [incoming] [options]
```

Mandatory options

- **--enabled** or **--enable** - Reject incoming connections that are not in response to outbound requests, such as DNS replies or DHCP requests. If access to services running on this machine is needed, you can choose to allow specific services through the firewall.
- **--disabled** or **--disable** - Do not configure any iptables rules.
- *device* - the device for which to configure firewall.

Optional options

- **--trust** - Listing a device here, such as **em1**, allows all traffic coming to and from that device to go through the firewall. To list more than one device, use the option more times, such as **--trust em1 --trust em2**. Do not use a comma-separated format such as **--trust em1, em2**.

- *incoming* - Replace with one or more of the following to allow the specified services through the firewall.
  - **--ssh**
  - **--smtp**
  - **--http**
  - **--ftp**
- **--port=** - You can specify that ports be allowed through the firewall using the port:protocol format. For example, to allow IMAP access through your firewall, specify **imap:tcp**. Numeric ports can also be specified explicitly; for example, to allow UDP packets on port 1234 through, specify **1234:udp**. To specify multiple ports, separate them by commas.
- **--service=** - This option provides a higher-level way to allow services through the firewall. Some services (like **cups**, **avahi**, and so on.) require multiple ports to be open or other special configuration in order for the service to work. You can specify each individual port with the **--port** option, or specify **--service=** and open them all at once.
  Valid options are anything recognized by the firewall-offline-cmd program in the **firewalld** package. If **firewalld** is running, **firewall-cmd --get-services** provides a list of known service names.
- **--use-system-defaults** - Do nnt configure the firewall at all. This option instructs anaconda to do nothing and allows the system to rely on the defaults that were provided with the package or ostree. If this option is used with other options then all other options will be ignored.

B.3.4. group

The **group** Kickstart command is optional. It creates a new user group on the system.

```
group --name=name [--gid=gid]
```

Mandatory options

- **--name=** - Provides the name of the group.

Optional options

- **--gid=** - The group's GID. If not provided, defaults to the next available non-system GID.

Notes

- If a group with the given name or GID already exists, this command fails.
- The **user** command can be used to create a new group for the newly created user.

B.3.5. keyboard (required)

The **keyboard** Kickstart command is required. It sets one or more available keyboard layouts for the system.

Options

- **--vckeymap=** - Specify a **VConsole** keymap which should be used. Valid names correspond to the list of files in the **/usr/lib/kbd/keymaps/xkb/** directory, without the **.map.gz** extension.

- **--xlayouts=** - Specify a list of X layouts that should be used as a comma-separated list without spaces. Accepts values in the same format as **setxkbmap(1)**, either in the ***layout*** format (such as **cz**), or in the ***layout* (*variant*)** format (such as **cz (qwerty)**).
  All available layouts can be viewed on the **xkeyboard-config(7)** man page under **Layouts**.

- **--switch=** - Specify a list of layout-switching options (shortcuts for switching between multiple keyboard layouts). Multiple options must be separated by commas without spaces. Accepts values in the same format as **setxkbmap(1)**.
  Available switching options can be viewed on the **xkeyboard-config(7)** man page under **Options**.

Notes

- Either the **--vckeymap=** or the **--xlayouts=** option must be used.

Example

The following example sets up two keyboard layouts (**English (US)** and **Czech (qwerty)**) using the **--xlayouts=** option, and allows to switch between them using **Alt**+**Shift**:

```
keyboard --xlayouts=us,'cz (qwerty)' --switch=grp:alt_shift_toggle
```

B.3.6. lang (required)

The **lang** Kickstart command is required. It sets the language to use during installation and the default language to use on the installed system.

Options

- **--addsupport=** - Add support for additional languages. Takes the form of comma-separated list without spaces. For example:

  ```
  lang en_US --addsupport=cs_CZ,de_DE,en_UK
  ```

Notes

+

- Certain languages (for example, Chinese, Japanese, Korean, and Indic languages) are not supported during text-mode installation. If you specify one of these languages with the **lang** command, the installation process continues in English, but the installed system uses your selection as its default language.

Example

To set the language to English, the Kickstart file should contain the following line:

```
lang en_US
```

B.3.7. module

The **module** Kickstart command is optional. Use this command to enable a package module stream within kickstart script.

Syntax

```
module --name=NAME [--stream=STREAM]
```

Mandatory options

- **--name=** - Specifies the name of the module to enable. Replace *NAME* with the actual name.

Optional options

- **--stream=** - Specifies the name of the module stream to enable. Replace *STREAM* with the actual name.
  You do not need to specify this option for modules with a default stream defined. For modules without a default stream, this option is mandatory and leaving it out results in an error. Enabling a module multiple times with different streams is not possible.

Notes

- Using a combination of this command and the **%packages** section allows you to install packages provided by the enabled module and stream combination, without specifying the module and stream explicitly. Modules must be enabled before package installation. After enabling a module with the **module** command, you can install the packages enabled by this module by listing them in the **%packages** section.
- A single **module** command can enable only a single module and stream combination. To enable multiple modules, use multiple **module** commands. Enabling a module multiple times with different streams is not possible.
- In Red Hat Enterprise Linux 8, modules are present only in the AppStream repository. To list available modules, use the **yum module list** command on an installed Red Hat Enterprise Linux 8 system with a valid subscription.

Additional resources

- For more information about modules and streams, see the Installing, managing, and removing user space components document.

B.3.8. pwpolicy

The **pwpolicy** Kickstart command is optional. This command can be used to enforce a custom password policy, which specifies requirements for passwords created during installation, based on factors such as password length and strength.

Syntax

```
pwpolicy name [--minlen=length] [--minquality=quality] [--strict|--nostrict] [--emptyok|--noempty] [--changesok|--nochanges]
```

Mandatory options

- *name* - Replace with either **root**, **user** or **luks** to enforce the policy for the **root** password, user passwords, or LUKS passphrase, respectively.

Optional options

- **--minlen=** - Sets the minimum allowed password length, in characters. The default is **6**.
- **--minquality=** - Sets the minimum allowed password quality as defined by the **libpwquality** library. The default value is **1**.
- **--strict** - Enables strict password enforcement. Passwords which do not meet the requirements specified in **--minquality=** and **--minlen=** will not be accepted. This option is disabled by default.
- **--notstrict** - Passwords which do *not* meet the minimum quality requirements specified by the **--minquality=** and **-minlen=** options will be allowed, after **Done** is clicked twice in the GUI. For text mode interface, a similar mechanism is used.
- **--emptyok** - Allows the use of empty passwords. Enabled by default for user passwords.
- **--notempty** - Disallows the use of empty passwords. Enabled by default for the root password and the LUKS passphrase.
- **--changesok** - Allows changing the password in the user interface, even if the Kickstart file already specifies a password. Disabled by default.
- **--nochanges** - Disallows changing passwords which are already set in the Kickstart file. Enabled by default.

Notes

- This command can only be used inside the **%anaconda** section.
- The **libpwquality** library is used to check minimum password requirements (length and quality). You can use the **pwscore** and **pwmake** commands provided by the **libpwquality** package to check the quality score of a password, or to create a random password with a given score. See the **pwscore(1)** and **pwmake(1)** man page for details about these commands.

B.3.9. repo

The **repo** Kickstart command is optional. It configures additional yum repositories that can be used as sources for package installation. You can add multiple **repo** lines.

Syntax

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

Mandatory options

- **--name=** - The repository id. This option is required. If a repository has a name which conflicts with another previously added repository, it is ignored. Because the installation program uses a list of preset repositories, this means that you cannot add repositories with the same names as the preset ones.

URL options

These options are mutually exclusive and optional. The variables that can be used in yum repository configuration files are not supported here. You can use the strings **$releasever** and **$basearch** which are replaced by the respective values in the URL.

- **--baseurl=** - The URL to the repository.
- **--mirrorlist=** - The URL pointing at a list of mirrors for the repository.
- **--metalink=** - The URL with metalink for the repository.

Optional options

- **--install** - Save the provided repository configuration on the installed system in the **/etc/yum.repos.d/** directory. Without using this option, a repository configured in a Kickstart file will only be available during the installation process, not on the installed system.
- **--cost=** - An integer value to assign a cost to this repository. If multiple repositories provide the same packages, this number is used to prioritize which repository will be used before another. Repositories with a lower cost take priority over repositories with higher cost.
- **--excludepkgs=** - A comma-separated list of package names that must *not* be pulled from this repository. This is useful if multiple repositories provide the same package and you want to make sure it comes from a particular repository. Both full package names (such as **publican**) and globs (such as **gnome-***) are accepted.
- **--includepkgs=** - A comma-separated list of package names and globs that are allowed to be pulled from this repository. Any other packages provided by the repository will be ignored. This is useful if you want to install just a single package or set of packages from a repository while excluding all other packages the repository provides.
- **--proxy=[*protocol*://][*username*[:*password*]@]*host*[:*port*]** - Specify an HTTP/HTTPS/FTP proxy to use just for this repository. This setting does not affect any other repositories, nor how the **install.img** is fetched on HTTP installations.
- **--noverifyssl** - Disable SSL verification when connecting to an **HTTPS** server.
- **--sslcacert=*path/to/SSLCACERT*** - Specify path to the file holding one or more SSL certificates to verify the repository host with.
- **--sslclientcert=*path/to/SSLCLIENTCERT*** - Specify path to the SSL client certificate (PEM file) which should be used to connect to the repository.
- **--sslclientkey=*path/to/SSLCLIENTKEY*** - Specify path to the private key file associated with the client certificate given with the **--sslclientcert** option.

Notes

- Repositories used for installation must be stable. The installation can fail if a repository is modified before the installation concludes.

B.3.10. rootpw (required)

The **rootpw** Kickstart command is required. It sets the system's root password to the *password* argument.

Syntax

```
rootpw [--iscrypted|--plaintext] [--lock] password
```

Options

- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use python:

  ```
  $ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  This generates a sha512 crypt-compatible hash of your password using a random salt.

- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**.

- **--lock** - If this option is present, the root account is locked by default. This means that the root user will not be able to log in from the console. This option will also disable the **Root Password** screens in both the graphical and text-based manual installation.

B.3.11. selinux

The **selinux** Kickstart command is optional. It sets the state of SELinux on the installed system. The default SELinux policy is **enforcing**.

Syntax

```
selinux [--disabled|--enforcing|--permissive]
```

Options

- **--enforcing** - Enables SELinux with the default targeted policy being **enforcing**.

- **--permissive** - Outputs warnings based on the SELinux policy, but does not actually enforce the policy.

- **--disabled** - Disables SELinux completely on the system.

Additional resources

For more information regarding SELinux, see the Using SElinux document.

B.3.12. services

The **services** Kickstart command is optional. It modifies the default set of services that will run under the default systemd target. The list of disabled services is processed before the list of enabled services. Therefore, if a service appears on both lists, it will be enabled.

Syntax

```
services [--disabled=list] [--enabled=list]
```

Options

- **--disabled=** - Disable the services given in the comma separated list.
- **--enabled=** - Enable the services given in the comma separated list.

Notes

*Do not include spaces in the list of services. If you do, Kickstart will enable or disable only the services up to the first space. For example:

+

```
services --disabled=auditd, cups,smartd, nfslock
```

+ That disables only the **auditd** service. To disable all four services, this entry must include no spaces:

+

```
services --disabled=auditd,cups,smartd,nfslock
```

B.3.13. skipx

The **skipx** Kickstart command is optional. If present, X is not configured on the installed system.

If you install a display manager among your package selection options, this package creates an X configuration, and the installed system defaults to **graphical.target**. That overrides the effect of the **skipx** option.

B.3.14. sshkey

The **sshkey** Kickstart command is optional. It adds a SSH key to the **authorized_keys** file of the specified user on the installed system.

Syntax

```
sshkey --username=user KEY
```

Mandatory options

- **--username=** - The user for which the key will be installed.
- *KEY* - The SSH key.

B.3.15. syspurpose

The **syspurpose** Kickstart command is optional. Use it to set the system purpose which describes how the system will be used after installation. This information helps apply the correct subscription entitlement to the system.

Syntax

```
syspurpose [options]
```

Options

- **--role=** - Set the intended system role. Available values are:
  - Red Hat Enterprise Linux Server
  - Red Hat Enterprise Linux Workstation
  - Red Hat Enterprise Linux Compute Node
- **--sla=** - Set the Service Level Agreement. Available values are:
  - Premium
  - Standard
  - Self-Support
- **--usage=** - The intended usage of the system. Available values are:
  - Production
  - Disaster Recovery
  - Development/Test
- **--addon=** - Specifies additional layered products or features. You can use this option multiple times.

Notes

- Enter the values with spaces and enclose them in double quotes:

```
syspurpose --role="Red Hat Enterprise Linux Server"
```

- While it is strongly recommended that you configure System Purpose, it is an optional feature of the Red Hat Enterprise Linux installation program. If you want to enable System Purpose after the installation completes, you can do so using the **syspurpose** command-line tool.

B.3.16. timezone (required)

The **timezone** Kickstart command is required. It sets the system time zone.

Syntax

```
timezone timezone [options]
```

Mandatory options

- *timezone* - the time zone to set for the system.

Optional options

- **--utc** - If present, the system assumes the hardware clock is set to UTC (Greenwich Mean) time.
- **--nontp** - Disable the NTP service automatic starting.

- **--ntpservers=** - Specify a list of NTP servers to be used as a comma-separated list without spaces.

Notes

In Red Hat Enterprise Linux 8, time zone names are validated using the **pytz.all_timezones** list, provided by the **pytz** package. In previous releases, the names were validated against **pytz.common_timezones**, which is a subset of the currently used list. Note that the graphical and text mode interfaces still use the more restricted **pytz.common_timezones** list; you must use a Kickstart file to use additional time zone definitions.

B.3.17. user

The **user** Kickstart command is optional. It creates a new user on the system.

Syntax

```
user --name=username [options]
```

Mandatory options

- **--name=** - Provides the name of the user. This option is required.

Optional options

- **--gecos=** - Provides the GECOS information for the user. This is a string of various system-specific fields separated by a comma. It is frequently used to specify the user's full name, office number, and so on. See the **passwd(5)** man page for more details.
- **--groups=** - In addition to the default group, a comma separated list of group names the user should belong to. The groups must exist before the user account is created. See the **group** command.
- **--homedir=** - The home directory for the user. If not provided, this defaults to **/home/*username***.
- **--lock** - If this option is present, this account is locked by default. This means that the user will not be able to log in from the console. This option will also disable the **Create User** screens in both the graphical and text-based manual installation.
- **--password=** - The new user's password. If not provided, the account will be locked by default.
- **--iscrypted** - If this option is present, the password argument is assumed to already be encrypted. This option is mutually exclusive with **--plaintext**. To create an encrypted password, you can use python:

  ```
  $ python -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if (pw==getpass.getpass("Confirm: ")) else exit())'
  ```

  This generates a sha512 crypt-compatible hash of your password using a random salt.
- **--plaintext** - If this option is present, the password argument is assumed to be in plain text. This option is mutually exclusive with **--iscrypted**
- **--shell=** - The user's login shell. If not provided, the system default is used.

- **--uid=** - The user's UID (User ID). If not provided, this defaults to the next available non-system UID.
- **--gid=** - The GID (Group ID) to be used for the user's group. If not provided, this defaults to the next available non-system group ID.

Notes

- Consider using the **--uid** and **--gid** options to set IDs of regular users and their default groups at range starting at **5000** instead of **1000**. That is because the range reserved for system users and groups, **0**-**999**, might increase in the future and thus overlap with IDs of regular users.
  For changing the minimum UID and GID limits after the installation, which ensures that your chosen UID and GID ranges are applied automatically on user creation, see the Setting default permissions for new files using umask section of the *Configuring basic system settings* document.
- Files and directories are created with various permissions, dictated by the application used to create the file or directory. For example, the **mkdir** command creates directories with all permissions enabled. However, applications are prevented from granting certain permissions to newly created files, as specified by the **user file-creation mask** setting.
  The **user file-creation mask** can be controlled with the **umask** command. The default setting of the **user file-creation mask** for new users is defined by the **UMASK** variable in the **/etc/login.defs** configuration file on the installed system. If unset, it defaults to **022**. This means that by default when an application creates a file, it is prevented from granting write permission to users other than the owner of the file. However, this can be overridden by other settings or scripts. More information can be found in the Setting default permissions for new files using umask section of the *Configuring basic system settings* document.

B.3.18. xconfig

The **xconfig** Kickstart command is optional. It configures the X Window System.

Options

- **--startxonboot** - Use a graphical login on the installed system.

Notes

- Because Red Hat Enterprise Linux 8 does not include the KDE Desktop Environment, do not use the **--defaultdesktop=** documented in upstream.

B.4. KICKSTART COMMANDS FOR NETWORK CONFIGURATION

The Kickstart commands in this list let you configure networking on the system.

B.4.1. network

The **network** Kickstart command is optional. It configures network information for the target system and activates network devices in the installation environment.

The device specified in the first **network** command is activated automatically. Activation of the device can be also explicitly required by the **--activate** option.

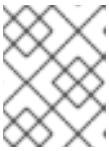
Options

- **--activate** - activate this device in the installation environment.
  If you use the **--activate** option on a device that has already been activated (for example, an interface you configured with boot options so that the system could retrieve the Kickstart file) the device is reactivated to use the details specified in the Kickstart file.

  Use the **--nodefroute** option to prevent the device from using the default route.

- **--no-activate** - do not activate this device in the installation environment.
  By default, Anaconda activates the first network device in the Kickstart file regardless of the **--activate** option. You can disable the default setting by using the **--no-activate** option.

- **--bootproto=** - One of **dhcp**, **bootp**, **ibft**, or **static**. The default option is **dhcp**; the **dhcp** and **bootp** options are treated the same. To disable **ipv4** configuration of the device, use **--noipv4** option.

  NOTE

  This option configures ipv4 configuration of the device. For ipv6 configuration use **--ipv6** and **--ipv6gateway** options.

  The DHCP method uses a DHCP server system to obtain its networking configuration. The BOOTP method is similar, requiring a BOOTP server to supply the networking configuration. To direct a system to use DHCP:

  ```
  network --bootproto=dhcp
  ```

  To direct a machine to use BOOTP to obtain its networking configuration, use the following line in the Kickstart file:

  ```
  network --bootproto=bootp
  ```

  To direct a machine to use the configuration specified in iBFT, use:

  ```
  network --bootproto=ibft
  ```

  The **static** method requires that you specify at least the IP address and netmask in the Kickstart file. This information is static and is used during and after the installation.

  All static networking configuration information must be specified on *one* line; you cannot wrap lines using a backslash (\) as you can on a command line.

  ```
  network --bootproto=static --ip=10.0.2.15 --netmask=255.255.255.0 --gateway=10.0.2.254 --nameserver=10.0.2.1
  ```

  You can also configure multiple nameservers at the same time. To do so, use the **--nameserver=** option once, and specify each of their IP addresses, separated by commas:

  ```
  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=** - specifies the device to be configured (and eventually activated in Anaconda) with the **network** command.

  If the **--device=** option is missing on the *first* use of the **network** command, the value of the

If the **--device=** option is missing on the *first* use of the **network** command, the value of the **ksdevice=** Anaconda boot option is used, if available. Note that this is considered deprecated behavior; in most cases, you should always specify a **--device=** for every **network** command.

The behavior of any subsequent **network** command in the same Kickstart file is unspecified if its **--device=** option is missing. Make sure you specify this option for any **network** command beyond the first.

You can specify a device to be activated in any of the following ways:

- the device name of the interface, for example, **em1**
- the MAC address of the interface, for example, **01:23:45:67:89:ab**
- the keyword **link**, which specifies the first interface with its link in the **up** state
- the keyword **bootif**, which uses the MAC address that pxelinux set in the **BOOTIF** variable. Set **IPAPPEND 2** in your **pxelinux.cfg** file to have pxelinux set the **BOOTIF** variable.

For example:

```
network --bootproto=dhcp --device=em1
```

- **--ip=** - IP address of the device.
- **--ipv6=** - IPv6 address of the device, in the form of *address*[/*prefix length*] - for example, **3ffe:ffff:0:1::1/128 `. If *prefix* is omitted, `64** is used. You can also use **auto** for automatic configuration, or **dhcp** for DHCPv6-only configuration (no router advertisements).
- **--gateway=** - Default gateway as a single IPv4 address.
- **--ipv6gateway=** - Default gateway as a single IPv6 address.
- **--nodefroute** - Prevents the interface being set as the default route. Use this option when you activate additional devices with the **--activate=** option, for example, a NIC on a separate subnet for an iSCSI target.
- **--nameserver=** - DNS name server, as an IP address. To specify more than one name server, use this option once, and separate each IP address with a comma.
- **--netmask=** - Network mask for the installed system.
- **--hostname=** - The host name for the installed system. The host name can either be a fully-qualified domain name (FQDN) in the format ***host_name.domainname***, or a short host name with no domain. Many networks have a Dynamic Host Configuration Protocol (DHCP) service which automatically supplies connected systems with a domain name; to allow DHCP to assign the domain name, only specify a short host name.

IMPORTANT

If your network does *not* provide a DHCP service, always use the FQDN as the system's host name.

- **--ethtool=** - Specifies additional low-level settings for the network device which will be passed to the ethtool program.

- **--onboot=** - Whether or not to enable the device at boot time.
- **--dhcpclass=** - The DHCP class.
- **--mtu=** - The MTU of the device.
- **--noipv4** - Disable IPv4 on this device.
- **--noipv6** - Disable IPv6 on this device.
- **--bondslaves=** - When this option is used, the bond device specified by the **--device=** option is created using slaves defined in the **--bondslaves=** option. For example:

```
network --device=bond0 --bondslaves=em1,em2
```

  The above command creates a bond device named **bond0** using the **em1** and **em2** interfaces as its slaves.

- **--bondopts=** - a list of optional parameters for a bonded interface, which is specified using the **--bondslaves=** and **--device=** options. Options in this list must be separated by commas (",") or semicolons (";"). If an option itself contains a comma, use a semicolon to separate the options. For example:

```
network --bondopts=mode=active-backup,balance-rr;primary=eth1
```

IMPORTANT

The **--bondopts=mode=** parameter only supports full mode names such as **balance-rr** or **broadcast**, not their numerical representations such as **0** or **3**.

- **--vlanid=** - Specifies virtual LAN (VLAN) ID number (802.1q tag) for the device created using the device specified in **--device=** as a parent. For example, **network --device=em1 --vlanid=171** creates a virtual LAN device **em1.171**.
- **--interfacename=** - Specify a custom interface name for a virtual LAN device. This option should be used when the default name generated by the **--vlanid=** option is not desirable. This option must be used along with **--vlanid=**. For example:

```
network --device=em1 --vlanid=171 --interfacename=vlan171
```

  The above command creates a virtual LAN interface named **vlan171** on the **em1** device with an ID of **171**.

  The interface name can be arbitrary (for example, **my-vlan**), but in specific cases, the following conventions must be followed:

  - If the name contains a dot (**.**), it must take the form of ***NAME.ID***. The *NAME* is arbitrary, but the *ID* must be the VLAN ID. For example: **em1.171** or **my-vlan.171**.
  - Names starting with **vlan** must take the form of **vlan*ID*** - for example, **vlan171**.

- **--teamslaves=** - Team device specified by the **--device=** option will be created using slaves specified in this option. Slaves are separated by commas. A slave can be followed by its configuration, which is a single-quoted JSON string with double quotes escaped by the \ character. For example:

```
network --teamslaves="p3p1'{\"prio\": -10, \"sticky\": true}',p3p2'{\"prio\": 100}'"
```

See also the **--teamconfig=** option.

- **--teamconfig=** - Double-quoted team device configuration which is a JSON string with double quotes escaped by the \ character. The device name is specified by **--device=** option and its slaves and their configuration by **--teamslaves=** option. For example:

```
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=** - When this option is used, the network bridge with device name specified using the **--device=** option will be created and devices defined in the **--bridgeslaves=** option will be added to the bridge. For example:

```
network --device=bridge0 --bridgeslaves=em1
```

- **--bridgeopts=** - An optional comma-separated list of parameters for the bridged interface. Available values are **stp**, **priority**, **forward-delay**, **hello-time**, **max-age**, and **ageing-time**. For information about these parameters, see the *bridge setting* table in the **nm-settings(5)** man page or at https://developer.gnome.org/NetworkManager/0.9/ref-settings.html.
Also see the Configuring and managing networking document for general information about network bridging.

- **--bindto=mac** - Bind the device configuration (**ifcfg**) file on the installed system to the device MAC address (**HWADDR**) instead of the default binding to the interface name (**DEVICE**). Note that this option is independent of the **--device=** option - **--bindto=mac** will be applied even if the same **network** command also specifies a device name, **link**, or **bootif**.

Notes

- The **ethN** device names such as **eth0** are no longer available in Red Hat Enterprise Linux 8 due to changes in the naming scheme. For more information about the device naming scheme, see the upstream document Predictable Network Interface Names .

- If you used a Kickstart option or a boot option to specify an installation repository on a network, but no network is available at the start of the installation, the installation program displays the **Network Configuration** window to set up a network connection prior to displaying the **Installation Summary** window. For more details, see the Configuring network and host name options section of the *Performing a standard RHEL installation* document.

B.4.2. realm

The **realm** Kickstart command is optional. Use it to join an Active Directory or IPA domain. For more information about this command, see the **join** section of the **realm(8)** man page.

Syntax

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

Options

- **--computer-ou=OU=** - Provide the distinguished name of an organizational unit in order to create the computer account. The exact format of the distinguished name depends on the client software and membership software. The root DSE portion of the distinguished name can usually be left out.
- **--no-password** - Join automatically without a password.
- **--one-time-password=** - Join using a one-time password. This is not possible with all types of realm.
- **--client-software=** - Only join realms which can run this client software. Valid values include **sssd** and **winbind**. Not all realms support all values. By default, the client software is chosen automatically.
- **--server-software=** - Only join realms which can run this server software. Possible values include **active-directory** or **freeipa**.
- **--membership-software=** - Use this software when joining the realm. Valid values include **samba** and **adcli**. Not all realms support all values. By default, the membership software is chosen automatically.

B.5. KICKSTART COMMANDS FOR HANDLING STORAGE

The Kickstart commands in this section configure aspects of storage such as devices, disks, partitions, LVM, and filesystems.

B.5.1. device (deprecated)

The **device** Kickstart command is optional. Use it to load additional kernel modules.

On most PCI systems, the installation program automatically detects Ethernet and SCSI cards. However, on older systems and some PCI systems, Kickstart requires a hint to find the proper devices. The **device** command, which tells the installation program to install extra modules, uses the following format:

Syntax

```
device moduleName --opts=options
```

Options

- *moduleName* - Replace with the name of the kernel module which should be installed.
- **--opts=** - Options to pass to the kernel module. For example:

```
device --opts="aic152x=0x340 io=11"
```

B.5.2. autopart

The **autopart** Kickstart command is optional. It automatically creates partitions.

The automatically created partitions are: a root (/) partition (1 GB or larger), a **swap** partition, and an appropriate /**boot** partition for the architecture. On large enough drives (50 GB and larger), this also creates a /**home** partition.

Options

- **--type=** - Selects one of the predefined automatic partitioning schemes you want to use. Accepts the following values:
  - **lvm**: The LVM partitioning scheme.
  - **plain**: Regular partitions with no LVM.
  - **thinp**: The LVM Thin Provisioning partitioning scheme.

  For a description of the available partition schemes, see Section C.1, "Supported device types".
- **--fstype=** - Selects one of the available file system types. The available values are **ext2**, **ext3**, **ext4**, **xfs**, and **vfat**. The default file system is **xfs**. For information about these file systems, see Section C.2, "Supported file systems".
- **--nohome** - Disables automatic creation of the **/home** partition.
- **--nolvm** - Do not use LVM for automatic partitioning. This option is equal to **--type=plain**.
- **--noboot** - Do not create a **/boot** partition.
- **--noswap** - Do not create a swap partition.
- **--encrypted** - Encrypts all partitions with Linux Unified Key Setup (LUKS). This is equivalent to checking the **Encrypt partitions** check box on the initial partitioning screen during a manual graphical installation.

NOTE

When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--passphrase=** - Provides a default system-wide passphrase for all encrypted devices.
- **--escrowcert=*URL_of_X.509_certificate*** - Stores data encryption keys of all encrypted volumes as files in **/root**, encrypted using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted volume. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Adds a randomly-generated passphrase to each encrypted volume. Store these passphrases in separate files in **/root**, encrypted using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default **aes-xts-plain64** is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has

no effect. Available types of encryption are listed in the Security hardening document, but Red Hat strongly recommends using either **aes-xts-plain64** or **aes-cbc-essiv:sha256**.

- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.

Notes

- The **autopart** option cannot be used together with the **part**/**partition**, **raid**, **logvol**, or **volgroup** options in the same Kickstart file.
- The **autopart** command is not mandatory, but you must include it if there are no **part** of **mount** commands in your Kickstart script.
- It is recommended to use the **autopart --nohome** Kickstart option when installing on a single FBA DASD of the CMS type. This ensures that the installation program does not create a separate /**home** partition. The installation then proceeds successfully.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

B.5.3. bootloader (required)

The **bootloader** Kickstart command is required. It specifies how the boot loader should be installed.

Syntax

```
bootloader [OPTIONS]
```

Options

- **--append=** - Specifies additional kernel parameters. To specify multiple parameters, separate them with spaces. For example:

  ```
  bootloader --location=mbr --append="hdd=ide-scsi ide=nodma"
  ```

  The **rhgb** and **quiet** parameters are automatically added when the **plymouth** package is installed, even if you do not specify them here or do not use the **--append=** command at all. To disable this behavior, explicitly disallow installation of **plymouth**:

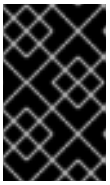
```
%packages
-plymouth
%end
```

This option is useful for disabling mechanisms which were implemented to mitigate the Meltdown and Spectre speculative execution vulnerabilities found in most modern processors (CVE-2017-5754, CVE-2017-5753, and CVE-2017-5715). In some cases, these mechanisms may be unnecessary, and keeping them enabled causes decreased performance with no improvement in security. To disable these mechanisms, add the options to do so into your Kickstart file - for example, **bootloader --append="nopti noibrs noibpb"** on AMD64/Intel 64 systems.

WARNING

Ensure your system is not at risk of attack before disabling any of the vulnerability mitigation mechanisms. See the Red Hat vulnerability response article for information about the Meltdown and Spectre vulnerabilities.

- **--boot-drive=** - Specifies which drive the boot loader should be written to, and therefore which drive the computer will boot from. If you use a multipath device as the boot drive, specify only one member of the device.



IMPORTANT

The **--boot-drive=** option is currently being ignored in Red Hat Enterprise Linux installations on IBM Z systems using the **zipl** boot loader. When **zipl** is installed, it determines the boot drive on its own.

- **--leavebootorder** - The installation program will add Red Hat Enterprise Linux 8 to the top of the list of installed systems in the boot loader, and preserve all existing entries as well as their order.
- **--driveorder=** - Specifies which drive is first in the BIOS boot order. For example:

```
bootloader --driveorder=sda,hda
```

- **--location=** - Specifies where the boot record is written. Valid values are the following:
  - **mbr** - The default option. Depends on whether the drive uses the Master Boot Record (MBR) or GUID Partition Table (GPT) scheme:
    On a GPT-formatted disk, this option installs stage 1.5 of the boot loader into the BIOS boot partition.

    On an MBR-formatted disk, stage 1.5 is installed into the empty space between the MBR and the first partition.
  - **partition** - Install the boot loader on the first sector of the partition containing the kernel.
  - **none** - Do not install the boot loader.

In most cases, this option does not need to be specified.

- **--nombr** - Do not install the boot loader to the MBR.
- **--password=** - If using GRUB2, sets the boot loader password to the one specified with this option. This should be used to restrict access to the GRUB2 shell, where arbitrary kernel options can be passed.
  If a password is specified, GRUB2 also asks for a user name. The user name is always **root**.
- **--iscrypted** - Normally, when you specify a boot loader password using the **--password=** option, it is stored in the Kickstart file in plain text. If you want to encrypt the password, use this option and an encrypted password.
  To generate an encrypted password, use the **grub2-mkpasswd-pbkdf2** command, enter the password you want to use, and copy the command's output (the hash starting with **grub.pbkdf2**) into the Kickstart file. An example **bootloader** Kickstart entry with an encrypted password looks similar to the following:

```
bootloader --iscrypted --
password=grub.pbkdf2.sha512.10000.5520C6C9832F3AC3D149AC0B24BE69E2D4FB0DBE
EDBD29CA1D30A044DE2645C4C7A291E585D4DC43F8A4D82479F8B95CA4BA4381F8550
510B75E8E0BB2938990.C688B6F0EF935701FF9BD1A8EC7FE5BD2333799C98F28420C5
CC8F1A2A233DE22C83705BB614EA17F3FDFDF4AC2161CEA3384E56EB38A2E39102F53
34C47405E
```

- **--timeout=** - Specifies the amount of time the boot loader waits before booting the default option (in seconds).
- **--default=** - Sets the default boot image in the boot loader configuration.
- **--extlinux** - Use the extlinux boot loader instead of GRUB2. This option only works on systems supported by extlinux.
- **--disabled** – This option is a stronger version of **--location=none**. While **--location=none** simply disables boot loader installation, **--disabled** disables boot loader installation and also disables installation of the package containing the boot loader, thus saving space.

Notes

- Red Hat recommends setting up a boot loader password on every system. An unprotected boot loader can allow a potential attacker to modify the system's boot options and gain unauthorized access to the system.
- In some cases, a special partition is required to install the boot loader on AMD64, Intel 64, and 64-bit ARM systems. The type and size of this partition depends on whether the disk you are installing the boot loader to uses the Master Boot Record (MBR) or a GUID Partition Table (GPT) schema. For more information, see the Configuring boot loader section of the *Performing a standard RHEL installation* document.
- Device names in the **sd*X*** (or /**dev**/**sd*X***) format are not guaranteed to be consistent across reboots, which can complicate usage of some Kickstart commands. When a command calls for a device node name, you can instead use any item from /**dev**/**disk**. For example, instead of:

```
part / --fstype=xfs --onpart=sda1
```

  You can use an entry similar to one of the following:

  -

```
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
```

This way the command will always target the same storage device. This is especially useful in large storage environments. See the chapter Overview of persistent naming attributes in the *Managing storage devices* document for more in-depth information about different ways to consistently refer to storage devices.

- The **--upgrade** option is deprecated in Red Hat Enterprise Linux 8.

B.5.4. clearpart

The **clearpart** Kickstart command is optional. It removes partitions from the system, prior to creation of new partitions. By default, no partitions are removed.

Options

- **--all** - Erases all partitions from the system.
  This option will erase all disks which can be reached by the installation program, including any attached network storage. Use this option with caution.

  You can prevent **clearpart** from wiping storage you want to preserve by using the **--drives=** option and specifying only the drives you want to clear, by attaching network storage later (for example, in the **%post** section of the Kickstart file), or by blacklisting the kernel modules used to access network storage.

- **--drives=** - Specifies which drives to clear partitions from. For example, the following clears all the partitions on the first two drives on the primary IDE controller:

  ```
  clearpart --drives=hda,hdb --all
  ```

  To clear a multipath device, use the format **disk/by-id/scsi-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to clear a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

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

  This format is preferable for all multipath devices, but if errors arise, multipath devices that do not use logical volume management (LVM) can also be cleared using the format **disk/by-id/dm-uuid-mpath-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to clear a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

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

  Never specify multipath devices by device names like **mpatha**. Device names such as this are not specific to a particular disk. The disk named **/dev/mpatha** during installation might not be the one that you expect it to be. Therefore, the **clearpart** command could target the wrong disk.

- **--initlabel** - Initializes a disk (or disks) by creating a default disk label for all disks in their respective architecture that have been designated for formatting (for example, msdos for x86). Because **--initlabel** can see all disks, it is important to ensure only those drives that are to be formatted are connected.

```
clearpart --initlabel --drives=names_of_disks
```

For example:

```
clearpart --initlabel --drives=dasda,dasdb,dasdc
```

- **--list=** - Specifies which partitions to clear. This option overrides the **--all** and **--linux** options if used. Can be used across different drives. For example:

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

- **--disklabel=*LABEL*** - Set the default disklabel to use. Only disklabels supported for the platform will be accepted. For example, on the 64-bit Intel and AMD architectures, the **msdos** and **gpt** disklabels are accepted, but **dasd** is not accepted.
- **--linux** - Erases all Linux partitions.
- **--none** (default) - Do not remove any partitions.
- **--cdl** - Reformat any LDL DASDs to CDL format.

Notes

- Device names in the **sd*X*** (or **/dev/sd*X***) format are not guaranteed to be consistent across reboots, which can complicate usage of some Kickstart commands. When a command calls for a device node name, you can instead use any item from **/dev/disk**. For example, instead of:

```
part / --fstype=xfs --onpart=sda1
```

You could use an entry similar to one of the following:

```
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
```

This way the command will always target the same storage device. This is especially useful in large storage environments. See the chapter Overview of persistent naming attributes in the *Managing storage devices* document for more in-depth information about different ways to consistently refer to storage devices.

- If the **clearpart** command is used, then the **part --onpart** command cannot be used on a logical partition.

B.5.5. fcoe

The **fcoe** Kickstart command is optional. It specifies which FCoE devices should be activated automatically in addition to those discovered by Enhanced Disk Drive Services (EDD).

Syntax

```
fcoe --nic=name [options]
```

Options

- **--nic=** (required) - The name of the device to be activated.
- **--dcb=** - Establish Data Center Bridging (DCB) settings.
- **--autovlan** - Discover VLANs automatically. This option is enabled by default.

B.5.6. ignoredisk

The **ignoredisk** Kickstart command is optional. It causes the installation program to ignore the specified disks.

This is useful if you use automatic partitioning and want to be sure that some disks are ignored. For example, without **ignoredisk**, attempting to deploy on a SAN-cluster the Kickstart would fail, as the installation program detects passive paths to the SAN that return no partition table.

Syntax

```
ignoredisk --drives=drive1,drive2,... | --only-use=drive
```

Options

- **--drives=*driveN,...*** - Replace *driveN* with one of **sda**, **sdb**,..., **hda**,... and so on.
- **--only-use** - Specifies a list of disks for the installation program to use. All other disks are ignored. For example, to use disk **sda** during installation and ignore all other disks:

  ```
  ignoredisk --only-use=sda
  ```

  To include a multipath device that does not use LVM:

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

  To include a multipath device that uses LVM:

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

You must specify one of the **--drives** and **--only-use**.

Notes

- The **--interactive** option is deprecated in Red Hat Enterprise Linux 8. This option allowed users to manually navigate the advanced storage screen.
- To ignore a multipath device that does not use logical volume management (LVM), use the format **disk/by-id/dm-uuid-mpath-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to ignore a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

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

- Multipath devices that use LVM are not assembled until after Anaconda has parsed the Kickstart file. Therefore, you cannot specify these devices in the format **dm-uuid-mpath**.

Instead, to ignore a multipath device that uses LVM, use the format **disk/by-id/scsi-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to ignore a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

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

- Never specify multipath devices by device names like **mpatha**. Device names such as this are not specific to a particular disk. The disk named **/dev/mpatha** during installation might not be the one that you expect it to be. Therefore, the **clearpart** command could target the wrong disk.
- Device names in the **sd*X*** (or **/dev/sd*X***) format are not guaranteed to be consistent across reboots, which can complicate usage of some Kickstart commands. When a command calls for a device node name, you can instead use any item from **/dev/disk**. For example, instead of:

```
part / --fstype=xfs --onpart=sda1
```

You can use an entry similar to one of the following:

```
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
```

This way the command will always target the same storage device. This is especially useful in large storage environments. See the chapter Overview of persistent naming attributes in the *Managing storage devices* document for more in-depth information about different ways to consistently refer to storage devices.

B.5.7. iscsi

The **iscsi** Kickstart command is optional. It specifies additional iSCSI storage to be attached during installation.

Syntax

```
iscsi --ipaddr=address [options]
```

Mandatory options

- **--ipaddr=** (required) - the IP address of the target to connect to.

Optional options

- **--port=** (required) - the port number. If not present, **--port=3260** is used automatically by default.
- **--target=** - the target IQN (iSCSI Qualified Name).
- **--iface=** - bind the connection to a specific network interface instead of using the default one determined by the network layer. Once used, it must be specified in all instances of the **iscsi** command in the entire Kickstart file.
- **--user=** - the user name required to authenticate with the target

- **--password=** - the password that corresponds with the user name specified for the target
- **--reverse-user=** - the user name required to authenticate with the initiator from a target that uses reverse CHAP authentication
- **--reverse-password=** - the password that corresponds with the user name specified for the initiator

Notes

- If you use the **iscsi** command, you must also assign a name to the iSCSI node, using the **iscsiname** command. The **iscsiname** command must appear before the **iscsi** command in the Kickstart file.
- Wherever possible, configure iSCSI storage in the system BIOS or firmware (iBFT for Intel systems) rather than use the **iscsi** command. Anaconda automatically detects and uses disks configured in BIOS or firmware and no special configuration is necessary in the Kickstart file.
- If you must use the **iscsi** command, ensure that networking is activated at the beginning of the installation, and that the **iscsi** command appears in the Kickstart file *before* you refer to iSCSI disks with commands such as **clearpart** or **ignoredisk**.

B.5.8. iscsiname

The **iscsiname** Kickstart command is optional. It assigns a name to an iSCSI node specified by the iscsi parameter. .Syntax

```
iscsiname iqn
```

Notes

- If you use the **iscsi** parameter in your Kickstart file, you must specify **iscsiname** *earlier* in the Kickstart file.

B.5.9. logvol

The **logvol** Kickstart command is optional. It creates a logical volume for Logical Volume Management (LVM).

Syntax

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

Mandatory options

- The *mntpoint* is where the partition is mounted and must be of one of the following forms:
  - /***path***
    For example, / or /**home**
  - **swap**
    The partition is used as swap space.

    To determine the size of the swap partition automatically, use the **--recommended** option:

```
swap --recommended
```

To determine the size of the swap partition automatically and also allow extra space for your system to hibernate, use the **--hibernation** option:

```
swap --hibernation
```

The size assigned will be equivalent to the swap space assigned by **--recommended** plus the amount of RAM on your system.

For the swap sizes assigned by these commands, see Section C.4, "Recommended partitioning scheme" for AMD64, Intel 64, and 64-bit ARM systems.

- **--vgname=*name*** - name of the volume group.
- **--name=*name*** - name of the logical volume.

Optional options

- **--noformat** - Use an existing logical volume and do not format it.
- **--useexisting** - Use an existing logical volume and reformat it.
- **--fstype=** - Sets the file system type for the logical volume. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, and **vfat**.
- **--fsoptions=** - Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the **/etc/fstab** file of the installed system and should be enclosed in quotes.
- **--mkfsoptions=** - Specifies additional parameters to be passed to the program that makes a filesystem on this partition. No processing is done on the list of arguments, so they must be supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem.
- **--fsprofile=** - Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For **ext2**, **ext3**, and **ext4**, this configuration file is **/etc/mke2fs.conf**.
- **--label=** - Sets a label for the logical volume.
- **--grow** - Tells the logical volume to grow to fill available space (if any), or up to the maximum size setting, if one is specified. A minimum size must be given, using either the **--percent=** option or the **--size=** option.
- **--size=** - The size of the logical volume in MiB. This option cannot be used together with the **--percent=** option.
- **--percent=** - The size of the logical volume, as a percentage of the free space in the volume group after any statically-sized logical volumes are taken into account. This option cannot be used together with the **--size=** option.

IMPORTANT

When creating a new logical volume, you must either specify its size statically using the **--size=** option, or as a percentage of remaining free space using the **--percent=** option. You cannot use both of these options on the same logical volume.

- **--maxsize=** - The maximum size in MiB when the logical volume is set to grow. Specify an integer value here such as **500** (do not include the unit).
- **--recommended** - Use this option when creating a logical volume to determine the size of this volume automatically, based on your system's hardware. For details about the recommended scheme, see Section C.4, "Recommended partitioning scheme" for AMD64, Intel 64, and 64-bit ARM systems.
- **--resize** - Resize a logical volume. If you use this option, you must also specify **--useexisting** and **--size**.
- **--encrypted** - Specifies that this logical volume should be encrypted with Linux Unified Key Setup (LUKS), using the passphrase provided in the **--passphrase=** option. If you do not specify a passphrase, the installation program uses the default, system-wide passphrase set with the **autopart --passphrase** command, or stops the installation and prompts you to provide a passphrase if no default is set.

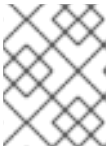
NOTE

When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--passphrase=** - Specifies the passphrase to use when encrypting this logical volume. You must use this option together with the **--encrypted** option; it has no effect by itself.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default **aes-xts-plain64** is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document, but Red Hat strongly recommends using either **aes-xts-plain64** or **aes-cbc-essiv:sha256**.
- **--escrowcert=*URL_of_X.509_certificate*** - Store data encryption keys of all encrypted volumes as files in **/root**, encrypted using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted volume. This option is only meaningful if **--encrypted** is specified.
- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Add a randomly-generated passphrase to each encrypted volume. Store these passphrases in separate files in **/root**, encrypted using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.

- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.
- **--thinpool** - Creates a thin pool logical volume. (Use a mount point of **none**)
- **--metadatasize=*size*** - Specify the metadata area size (in MiB) for a new thin pool device.
- **--chunksize=*size*** - Specify the chunk size (in KiB) for a new thin pool device.
- **--thin** - Create a thin logical volume. (Requires use of **--poolname**)
- **--poolname=*name*** - Specify the name of the thin pool in which to create a thin logical volume. Requires the **--thin** option.
- **--profile=*name*** - Specify the configuration profile name to use with thin logical volumes. If used, the name will also be included in the metadata for the given logical volume. By default, the available profiles are **default** and **thin-performance** and are defined in the **/etc/lvm/profile/** directory. See the **lvm(8)** man page for additional information.
- **--cachepvs=** - A comma-separated list of physical volumes which should be used as a cache for this volume.
- **--cachemode=** - Specify which mode should be used to cache this logical volume - either **writeback** or **writethrough**.



NOTE

For more information about cached logical volumes and their modes, see the **lvmcache(7)** man page.

- **--cachesize=** - Size of cache attached to the logical volume, specified in MiB. This option requires the **--cachepvs=** option.

Notes

- Do not use the dash (**-**) character in logical volume and volume group names when installing Red Hat Enterprise Linux using Kickstart. If this character is used, the installation finishes normally, but the **/dev/mapper/** directory will list these volumes and volume groups with every dash doubled. For example, a volume group named **volgrp-01** containing a logical volume named **logvol-01** will be listed as **/dev/mapper/volgrp—01-logvol—01**.
  This limitation only applies to newly created logical volume and volume group names. If you are reusing existing ones using the **--noformat** option, their names will not be changed.

Examples

- Create the partition first, create the logical volume group, and then create the logical volume:

```
part pv.01 --size 3000
volgroup myvg pv.01
logvol / --vgname=myvg --size=2000 --name=rootvol
```

- Create the partition first, create the logical volume group, and then create the logical volume to occupy 90% of the remaining space in the volume group:

```
part pv.01 --size 1 --grow
volgroup myvg pv.01
logvol / --vgname=myvg --name=rootvol --percent=90
```

Additional resources

- For more information regarding LVM, see the Configuring and managing logical volumes document.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

B.5.10. mount

The **mount** Kickstart command is optional. It assigns a mount point to an existing block device, and optionally reformats it to a given format.

Syntax

```
mount [--reformat [REFORMAT]] [--mkfsoptions MKFS_OPTS] [--mountoptions MOUNT_OPTS]
device mntpoint
```

Mandatory options:

- ***device*** - The block device to mount.
- ***mntpoint*** - Where to mount the ***device***. It must be a valid mount point, such as / or /**usr**, or **none** if the device is unmountable (for example **swap**).

Optional options:

- **--reformat=** - Specifies a new format (such as **ext4**) to which the device should be reformatted.
- **--mkfsoptions=** - Specifies additional options to be passed to the command which creates the new file system specified in **--reformat=**. The list of options provided here is not processed, so they must be specified in a format that can be passed directly to the **mkfs** program. The list of options should be either comma-separated or surrounded by double quotes, depending on the file system. See the **mkfs** man page for the file system you want to create (for example **mkfs.ext4(8)** or **mkfs.xfs(8)**) for specific details.
- **--mountoptions=** - Specifies a free form string that contains options to be used when mounting

the file system. The string will be copied to the **/etc/fstab** file on the installed system and should be enclosed in double quotes. See the **mount(8)** man page for a full list of mount options, and **fstab(5)** for basics.

Notes

- Unlike most other storage configuration commands in Kickstart, **mount** does not require you to describe the entire storage configuration in the Kickstart file. You only need to ensure that the described block device exists on the system. However, if you want to *create* the storage stack with all the devices mounted, you must use other commands such as **part** to do so.
- You can not use **mount** together with other storage-related commands such as **part**, **logvol**, or **autopart** in the same Kickstart file.

B.5.11. nvdimm

The **nvdimm** Kickstart command is optional. It performs an action on Non-Volatile Dual In-line Memory Module (NVDIMM) devices.

Syntax

```
nvdimm action [options]
```

Actions

- **reconfigure** - Reconfigure a specific NVDIMM device into a given mode. Additionally, the specified device is implicitly marked as to be used, so a subsequent **nvdimm use** command for the same device is redundant. This action uses the following format:

  ```
  nvdimm reconfigure [--namespace=NAMESPACE] [--mode=MODE] [--
  sectorsize=SECTORSIZE]
  ```

  - **--namespace=** - The device specification by namespace. For example:

    ```
    nvdimm reconfigure --namespace=namespace0.0 --mode=sector --sectorsize=512
    ```

  - **--mode=** - The mode specification. Currently, only the value **sector** is available.
  - **--sectorsize=** - Size of a sector for sector mode. For example:

    ```
    nvdimm reconfigure --namespace=namespace0.0 --mode=sector --sectorsize=512
    ```

    The supported sector sizes are 512 and 4096 bytes.

- **use** - Specify a NVDIMM device as a target for installation. The device must be already configured to the sector mode by the **nvdimm reconfigure** command. This action uses the following format:

  ```
  nvdimm use [--namespace=NAMESPACE|--blockdevs=DEVICES]
  ```

  - **--namespace=** - Specifies the device by namespace. For example:

    ```
    nvdimm use --namespace=namespace0.0
    ```

- **--blockdevs=** - Specifies a comma-separated list of block devices corresponding to the NVDIMM devices to be used. The asterisk * wildcard is supported. For example:

```
nvdimm use --blockdevs=pmem0s,pmem1s
nvdimm use --blockdevs=pmem*
```

Notes

- By default, all NVDIMM devices are ignored by the installation program. You must use the **nvdimm** command to enable installation on these devices.

B.5.12. part or partition

The **part** or **partition** Kickstart command is required. It creates a partition on the system.

Syntax

```
part|partition mntpoint --name=name --device=device --rule=rule [options]
```

Options

- *mntpoint* - Where the partition is mounted. The value must be of one of the following forms:
  - /***path***
    For example, /, /**usr**, /**home**
  - **swap**
    The partition is used as swap space.

    To determine the size of the swap partition automatically, use the **--recommended** option:

    ```
    swap --recommended
    ```

    The size assigned will be effective but not precisely calibrated for your system.

    To determine the size of the swap partition automatically but also allow extra space for your system to hibernate, use the **--hibernation** option:

    ```
    swap --hibernation
    ```

    The size assigned will be equivalent to the swap space assigned by **--recommended** plus the amount of RAM on your system.

    For the swap sizes assigned by these commands, see Section C.4, "Recommended partitioning scheme" for AMD64, Intel 64, and 64-bit ARM systems.
  - **raid.*id***
    The partition is used for software RAID (see **raid**).
  - **pv.*id***
    The partition is used for LVM (see **logvol**).
  - **biosboot**
    The partition will be used for a BIOS Boot partition. A 1 MiB BIOS boot partition is necessary

on BIOS-based AMD64 and Intel 64 systems using a GUID Partition Table (GPT); the boot loader will be installed into it. It is not necessary on UEFI systems. See also the **bootloader** command.

- **/boot/efi**
  An EFI System Partition. A 50 MiB EFI partition is necessary on UEFI-based AMD64, Intel 64, and 64-bit ARM; the recommended size is 200 MiB. It is not necessary on BIOS systems. See also the **bootloader** command.

- **--size=** - The minimum partition size in MiB. Specify an integer value here such as **500** (do not include the unit).

IMPORTANT

If the **--size** value is too small, the installation fails. Set the **--size** value as the minimum amount of space you require. For size recommendations, see Section C.4, "Recommended partitioning scheme".

- **--grow** - Tells the logical volume to grow to fill available space (if any), or up to the maximum size setting, if one is specified.

NOTE

If you use **--grow=** without setting **--maxsize=** on a swap partition, Anaconda limits the maximum size of the swap partition. For systems that have less than 2 GB of physical memory, the imposed limit is twice the amount of physical memory. For systems with more than 2 GB, the imposed limit is the size of physical memory plus 2GB.

- **--maxsize=** - The maximum partition size in MiB when the partition is set to grow. Specify an integer value here such as **500** (do not include the unit).

- **--noformat** - Specifies that the partition should not be formatted, for use with the **--onpart** command.

- **--onpart=** or **--usepart=** - Specifies the device on which to place the partition. For example:

```
partition /home --onpart=hda1
```

puts **/home** on **/dev/hda1**.

These options can also add a partition to a logical volume. For example:

```
partition pv.1 --onpart=hda2
```

The device must already exist on the system; the **--onpart** option will not create it.

It is also possible to specify an entire drive, rather than a partition, in which case Anaconda will format and use the drive without creating a partition table. Note, however, that installation of GRUB2 is not supported on a device formatted in this way, and must be placed on a drive with a partition table.

- **--ondisk=** or **--ondrive=** - Forces the partition to be created on a particular disk. For example, **--ondisk=sdb** puts the partition on the second SCSI disk on the system.

To specify a multipath device that does not use logical volume management (LVM), use the format **disk/by-id/dm-uuid-mpath-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to specify a disk with WWID **2416CD96995134CA5D787F00A5AA11017**, use:

```
part / --fstype=xfs --grow --asprimary --size=8192 --ondisk=disk/by-id/dm-uuid-mpath-
2416CD96995134CA5D787F00A5AA11017
```

Multipath devices that use LVM are not assembled until after Anaconda has parsed the Kickstart file. Therefore, you cannot specify these devices in the format **dm-uuid-mpath**. Instead, to specify a multipath device that uses LVM, use the format **disk/by-id/scsi-*WWID***, where *WWID* is the world-wide identifier for the device. For example, to specify a disk with WWID **58095BEC5510947BE8C0360F604351918**, use:

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

WARNING

Never specify multipath devices by device names like **mpatha**. Device names such as this are not specific to a particular disk. The disk named **/dev/mpatha** during installation might not be the one that you expect it to be. Therefore, the **clearpart** command could target the wrong disk.

- **--asprimary** - Forces the partition to be allocated as a *primary* partition. If the partition cannot be allocated as primary (usually due to too many primary partitions being already allocated), the partitioning process fails. This option only makes sense when the disk uses a Master Boot Record (MBR); for GUID Partition Table (GPT)-labeled disks this option has no meaning.

- **--fsprofile=** - Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For **ext2**, **ext3**, **ext4**, this configuration file is **/etc/mke2fs.conf**.

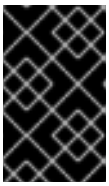
- **--mkfsoptions=** - Specifies additional parameters to be passed to the program that makes a filesystem on this partition. This is similar to **--fsprofile** but works for all filesystems, not just the ones that support the profile concept. No processing is done on the list of arguments, so they must be supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem.

- **--fstype=** - Sets the file system type for the partition. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, **vfat**, **efi** and **biosboot**.

- **--fsoptions** - Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the **/etc/fstab** file of the installed system and should be enclosed in quotes.

- **--label=** - assign a label to an individual partition.

- **--recommended** - Determine the size of the partition automatically. For details about the recommended scheme, see Section C.4, "Recommended partitioning scheme" for AMD64, Intel 64, and 64-bit ARM.



IMPORTANT

This option can only be used for partitions which result in a file system such as the **/boot** partition and **swap** space. It cannot be used to create LVM physical volumes or RAID members.

- **--onbiosdisk** - Forces the partition to be created on a particular disk as discovered by the BIOS.
- **--encrypted** - Specifies that this partition should be encrypted with Linux Unified Key Setup (LUKS), using the passphrase provided in the **--passphrase** option. If you do not specify a passphrase, Anaconda uses the default, system-wide passphrase set with the **autopart --passphrase** command, or stops the installation and prompts you to provide a passphrase if no default is set.

NOTE

When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--passphrase=** - Specifies the passphrase to use when encrypting this partition. You must use this option together with the **--encrypted** option; by itself it has no effect.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default **aes-xts-plain64** is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document, but Red Hat strongly recommends using either **aes-xts-plain64** or **aes-cbc-essiv:sha256**.
- **--escrowcert=*URL_of_X.509_certificate*** - Store data encryption keys of all encrypted partitions as files in **/root**, encrypted using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. The keys are stored as a separate file for each encrypted partition. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Add a randomly-generated passphrase to each encrypted partition. Store these passphrases in separate files in **/root**, encrypted using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.

- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.
- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.
- **--resize=** - Resize an existing partition. When using this option, specify the target size (in MiB) using the **--size=** option and the target partition using the **--onpart=** option.

Notes

- The **part** command is not mandatory, but you must include either **part**, **autopart** or **mount** in your Kickstart script.
- The **--active** option is deprecated in Red Hat Enterprise Linux 8.
- If partitioning fails for any reason, diagnostic messages appear on virtual console 3.
- All partitions created are formatted as part of the installation process unless **--noformat** and **--onpart** are used.
- Device names in the **sd*X*** (or **/dev/sd*X***) format are not guaranteed to be consistent across reboots, which can complicate usage of some Kickstart commands. When a command calls for a device node name, you can instead use any item from **/dev/disk**. For example, instead of:

```
part / --fstype=xfs --onpart=sda1
```

  You could use an entry similar to one of the following:

```
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
```

  This way the command will always target the same storage device. This is especially useful in large storage environments. See the chapter Overview of persistent naming attributes in the *Managing storage devices* document for more in-depth information about different ways to consistently refer to storage devices.
- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

B.5.13. raid

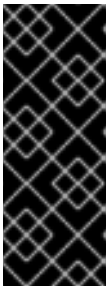
The **raid** Kickstart command is optional. It assembles a software RAID device.

Syntax

```
raid mntpoint --level=level --device=device-name partitions*
```

Options

- *mntpoint* - Location where the RAID file system is mounted. If it is /, the RAID level must be 1 unless a boot partition (/**boot**) is present. If a boot partition is present, the /**boot** partition must be level 1 and the root (/) partition can be any of the available types. The *partitions** (which denotes that multiple partitions can be listed) lists the RAID identifiers to add to the RAID array.

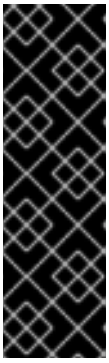


IMPORTANT

On IBM Power Systems, if a RAID device has been prepared and has not been reformatted during the installation, ensure that the RAID metadata version is **0.90** if you intend to put the /**boot** and **PReP** partitions on the RAID device.

The default Red Hat Enterprise Linux 7 **mdadm** metadata version is not supported for the boot device.

- **--level=** - RAID level to use (0, 1, 4, 5, 6, or 10). See Section C.3, "Supported RAID types" for information about various available RAID levels.
- **--device=** - Name of the RAID device to use - for example, **--device=root**.



IMPORTANT

Do not use **mdraid** names in the form of **md0** - these names are not guaranteed to be persistent. Instead, use meaningful names such as **root** or **swap**. Using meaningful names creates a symbolic link from /**dev**/**md**/***name*** to whichever /**dev**/**md*X*** node is assigned to the array.

If you have an old (v0.90 metadata) array that you cannot assign a name to, you can specify the array by a filesystem label or UUID (for example, **--device=rhel7-root --label=rhel7-root**).

- **--chunksize=** - Sets the chunk size of a RAID storage in KiB. In certain situations, using a different chunk size than the default (**512 Kib**) can improve the performance of the RAID.
- **--spares=** - Specifies the number of spare drives allocated for the RAID array. Spare drives are used to rebuild the array in case of drive failure.
- **--fsprofile=** - Specifies a usage type to be passed to the program that makes a filesystem on this partition. A usage type defines a variety of tuning parameters to be used when making a filesystem. For this option to work, the filesystem must support the concept of usage types and there must be a configuration file that lists valid types. For ext2, ext3, and ext4, this configuration file is /**etc**/**mke2fs.conf**.
- **--fstype=** - Sets the file system type for the RAID array. Valid values are **xfs**, **ext2**, **ext3**, **ext4**, **swap**, and **vfat**.
- **--fsoptions=** - Specifies a free form string of options to be used when mounting the filesystem. This string will be copied into the /**etc**/**fstab** file of the installed system and should be enclosed in quotes.
- **--mkfsoptions=** - Specifies additional parameters to be passed to the program that makes a

filesystem on this partition. No processing is done on the list of arguments, so they must be supplied in a format that can be passed directly to the mkfs program. This means multiple options should be comma-separated or surrounded by double quotes, depending on the filesystem.

- **--label=** - Specify the label to give to the filesystem to be made. If the given label is already in use by another filesystem, a new label will be created.
- **--noformat** - Use an existing RAID device and do not format the RAID array.
- **--useexisting** - Use an existing RAID device and reformat it.
- **--encrypted** - Specifies that this RAID device should be encrypted with Linux Unified Key Setup (LUKS), using the passphrase provided in the **--passphrase** option. If you do not specify a passphrase, Anaconda uses the default, system-wide passphrase set with the **autopart --passphrase** command, or stops the installation and prompts you to provide a passphrase if no default is set.

NOTE

When encrypting one or more partitions, Anaconda attempts to gather 256 bits of entropy to ensure the partitions are encrypted securely. Gathering entropy can take some time - the process will stop after a maximum of 10 minutes, regardless of whether sufficient entropy has been gathered.

The process can be sped up by interacting with the installation system (typing on the keyboard or moving the mouse). If you are installing in a virtual machine, you can also attach a **virtio-rng** device (a virtual random number generator) to the guest.

- **--luks-version=*LUKS_VERSION*** - Specifies which version of LUKS format should be used to encrypt the filesystem. This option is only meaningful if **--encrypted** is specified.
- **--cipher=** - Specifies the type of encryption to use if the Anaconda default **aes-xts-plain64** is not satisfactory. You must use this option together with the **--encrypted** option; by itself it has no effect. Available types of encryption are listed in the Security hardening document, but Red Hat strongly recommends using either **aes-xts-plain64** or **aes-cbc-essiv:sha256**.
- **--passphrase=** - Specifies the passphrase to use when encrypting this RAID device. You must use this option together with the **--encrypted** option; by itself it has no effect.
- **--escrowcert=*URL_of_X.509_certificate*** - Store the data encryption key for this device in a file in **/root**, encrypted using the X.509 certificate from the URL specified with *URL_of_X.509_certificate*. This option is only meaningful if **--encrypted** is specified.
- **--backuppassphrase** - Add a randomly-generated passphrase to this device. Store the passphrase in a file in **/root**, encrypted using the X.509 certificate specified with **--escrowcert**. This option is only meaningful if **--escrowcert** is specified.
- **--pbkdf=*PBKDF*** - Sets Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.
- **--pbkdf-memory=*PBKDF_MEMORY*** - Sets the memory cost for PBKDF. See also the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified.

- **--pbkdf-time=*PBKDF_TIME*** - Sets the number of milliseconds to spend with PBKDF passphrase processing. See also **--iter-time** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-iterations**.

- **--pbkdf-iterations=*PBKDF_ITERATIONS*** - Sets the number of iterations directly and avoids PBKDF benchmark. See also **--pbkdf-force-iterations** in the man page *cryptsetup(8)*. This option is only meaningful if **--encrypted** is specified, and is mutually exclusive with **--pbkdf-time**.

Example

The following example shows how to create a RAID level 1 partition for /, and a RAID level 5 for **/home**, assuming there are three SCSI disks on the system. It also creates three swap partitions, one on each drive.

```
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
```

Notes

- If you lose the LUKS passphrase, any encrypted partitions and their data is completely inaccessible. There is no way to recover a lost passphrase. However, you can save encryption passphrases with the **--escrowcert** and create backup encryption passphrases with the **--backuppassphrase** options.

B.5.14. reqpart

The **reqpart** Kickstart command is optional. It automatically creates partitions required by your hardware platform. These include a **/boot/efi** partition for systems with UEFI firmware, a **biosboot** partition for systems with BIOS firmware and GPT, and a **PRePBoot** partition for IBM Power Systems.

Syntax

```
reqpart [--add-boot]
```

Options

- **--add-boot** - Creates a separate **/boot** partition in addition to the platform-specific partition created by the base command.

Notes

- This command cannot be used toegether with **autopart**, because **autopart** does everything the **reqpart** command does and, in addition, creates other partitions or logical volumes such as / and **swap**. In contrast with **autopart**, this command only creates platform-specific partitions and

leaves the rest of the drive empty, allowing you to create a custom layout.

B.5.15. snapshot

The **snapshot** Kickstart command is optional. Use it to create LVM thin volume snapshots during the installation process. This enables you to back up a logical volume before or after the installation.

To create multiple snapshots, add the **snaphost** Kickstart command multiple times.

Syntax

```
snapshots vg_name/lv_name --name=snapshot_name --when=pre-install|post-install
```

Options

- ***vg_name*/*lv_name*** - Sets the name of the volume group and logical volume to create the snapshot from.
- **--name=*snapshot_name*** - Sets the name of the snapshot. This name must be unique within the volume group.
- **--when=*pre-install*|*post-install*** - Sets if the snapshot is created before the installation begins or after the installation is completed.

B.5.16. volgroup

The **volgroup** Kickstart command is optional. It creates a Logical Volume Management (LVM) group.

Syntax

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

Options

- **--noformat** - Use an existing volume group and do not format it.
- **--useexisting** - Use an existing volume group and reformat it. If you use this option, do not specify a *partition*. For example:

  ```
  volgroup rhel00 --useexisting --noformat
  ```

- **--pesize=** - Set the size of the volume group's physical extents in KiB. The default value is 4096 (4 MiB), and the minimum value is 1024 (1 MiB).
- **--reserved-space=** - Specify an amount of space to leave unused in a volume group in MiB. Applicable only to newly created volume groups.
- **--reserved-percent=** - Specify a percentage of total volume group space to leave unused. Applicable only to newly created volume groups.

Notes

- Create the partition first, then create the logical volume group, and then create the logical volume. For example:

```
part pv.01 --size 10000
volgroup volgrp pv.01 ` [command]`logvol / --vgname=volgrp --size=2000 --name=root
```

- Do not use the dash (**-**) character in logical volume and volume group names when installing Red Hat Enterprise Linux using Kickstart. If this character is used, the installation finishes normally, but the /**dev**/**mapper**/ directory will list these volumes and volume groups with every dash doubled. For example, a volume group named **volgrp-01** containing a logical volume named **logvol-01** will be listed as /**dev**/**mapper**/**volgrp—01-logvol—01**.
  This limitation only applies to newly created logical volume and volume group names. If you are reusing existing ones using the **--noformat** option, their names will not be changed.

B.5.17. zerombr

The **zerombr** Kickstart command is optional. If **zerombr** is specified, any invalid partition tables found on disks are initialized. This destroys all of the contents of disks with invalid partition tables.

Syntax

```
zerombr
```

Notes

- On IBM Z, if **zerombr** is specified, any Direct Access Storage Device (DASD) visible to the installation program which is not already low-level formatted is automatically low-level formatted with dasdfmt. The command also prevents user choice during interactive installations.
- If **zerombr** is not specified and there is at least one unformatted DASD visible to the installation program, a non-interactive Kickstart installation exits unsuccessfully.
- If **zerombr** is not specified and there is at least one unformatted DASD visible to the installation program, an interactive installation exits if the user does not agree to format all visible and unformatted DASDs. To circumvent this, only activate those DASDs that you will use during installation. You can always add more DASDs after installation is complete.

B.5.18. zfcp

The **zfcp** Kickstart command is optional. It defines a Fibre channel device.

This option only applies on IBM Z. All of the options described below must be specified.

Syntax

```
zfcp --devnum=devnum --wwpn=wwpn --fcplun=lun
```

Options

- **--devnum** - The device number (zFCP adapter device bus ID).
- **--wwpn** - The device's World Wide Port Name (WWPN). Takes the form of a 16-digit number, preceded by **0x**.

- **--fcplun** - The device's Logical Unit Number (LUN). Takes the form of a 16-digit number, preceded by **0x**.

Example

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

B.6. KICKSTART COMMANDS FOR ADDONS SUPPLIED WITH THE RHEL INSTALLATION PROGRAM

The Kickstart commands in this section are related to add-ons supplied by default with the Red Hat Enterprise Linux installation program: Kdump and OpenSCAP.

B.6.1. %addon com_redhat_kdump

The **%addon com_redhat_kdump** Kickstart command is optional. This command configures the kdump kernel crash dumping mechanism.

Syntax

```
%addon com_redhat_kdump [OPTIONS]
%end
```

NOTE

The syntax for this command is unusual because it is an add-on rather than a built-in Kickstart command.

Notes

Kdump is a kernel crash dumping mechanism that allows you to save the contents of the system's memory for later analysis. It relies on **kexec**, which can be used to boot a Linux kernel from the context of another kernel without rebooting the system, and preserve the contents of the first kernel's memory that would otherwise be lost.

In case of a system crash, **kexec** boots into a second kernel (a capture kernel). This capture kernel resides in a reserved part of the system memory. Kdump then captures the contents of the crashed kernel's memory (a crash dump) and saves it to a specified location. The location cannot be configured using this Kickstart command; it must be configured after the installation by editing the **/etc/kdump.conf** configuration file.

For more information about Kdump, see the Installing and configuring kdump chapter of the *Managing, monitoring and updating the kernel* document.

Options

- **--enable** - Enable kdump on the installed system.
- **--disable** - Disable kdump on the installed system.
- **--reserve-mb=** - The amount of memory you want to reserve for kdump, in MiB. For example:

```
%addon com_redhat_kdump --enable --reserve-mb=128
%end
```

You can also specify **auto** instead of a numeric value. In that case, the installation program will determine the amount of memory automatically based on the criteria described in the Memory requirements for kdump section of the *Managing, monitoring and updating the kernel* document.

If you enable kdump and do not specify a **--reserve-mb=** option, the value **auto** will be used.

- **--enablefadump** - Enable firmware-assisted dumping on systems which allow it (notably, IBM Power Systems servers).

B.6.2. %addon org_fedora_oscap

The **%addon org_fedora_oscap** Kickstart command is optional.

The OpenSCAP installation program add-on is used to apply SCAP (Security Content Automation Protocol) content - security policies - on the installed system. This add-on has been enabled by default since Red Hat Enterprise Linux 7.2. When enabled, the packages necessary to provide this functionality will automatically be installed. However, by default, no policies are enforced, meaning that no checks are performed during or after installation unless specifically configured.

IMPORTANT

Applying a security policy is not necessary on all systems. This screen should only be used when a specific policy is mandated by your organization rules or government regulations.

Unlike most other commands, this add-on does not accept regular options, but uses key-value pairs in the body of the **%addon** definition instead. These pairs are whitespace-agnostic. Values can be optionally enclosed in single quotes (') or double quotes (").

Keys

The following keys are recognized by the add-on:

- **content-type** - Type of the security content. Possible values are **datastream**, **archive**, **rpm**, and **scap-security-guide**.
  If the **content-type** is **scap-security-guide**, the add-on will use content provided by the **scap-security-guide** package, which is present on the boot media. This means that all other keys except **profile** will have no effect.
- **content-url** - Location of the security content. The content must be accessible using HTTP, HTTPS, or FTP; local storage is currently not supported. A network connection must be available to reach content definitions in a remote location.
- **datastream-id** - ID of the data stream referenced in the **content-url** value. Used only if **content-type** is **datastream**.
- **xccdf-id** - ID of the benchmark you want to use.
- **xccdf-path** - Path to the XCCDF file which should be used; given as a relative path in the archive.
- **profile** - ID of the profile to be applied. Use **default** to apply the default profile.
- **fingerprint** - A MD5, SHA1 or SHA2 checksum of the content referenced by **content-url**.

- **tailoring-path** - Path to a tailoring file which should be used, given as a relative path in the archive.

Examples

- The following is an example **%addon org_fedora_oscap** section which uses content from the **scap-security-guide** on the installation media:

Example B.1. Sample OpenSCAP Add-on Definition Using SCAP Security Guide

```
%addon org_fedora_oscap
content-type = scap-security-guide
profile = pci-dss
%end
```

- The following is a more complex example which loads a custom profile from a web server:

Example B.2. Sample OpenSCAP Add-on Definition Using a Datastream

```
%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
```

Additional resources

- Additional information about the OpenSCAP installation program add-on is available at https://www.open-scap.org/tools/oscap-anaconda-addon/.
- For more information about the profiles available in the SCAP Security Guide and what they do, see the OpenSCAP Portal.

APPENDIX C. PARTITIONING REFERENCE

C.1. SUPPORTED DEVICE TYPES

Standard partition

A standard partition can contain a file system or swap space. Standard partitions are most commonly used for /**boot** and the **BIOS Boot** and **EFI System partitions**. LVM logical volumes are recommended for most other uses.

LVM

Choosing **LVM** (or Logical Volume Management) as the device type creates an LVM logical volume. If no LVM volume group currently exists, one is automatically created to contain the new volume; if an LVM volume group already exists, the volume is assigned. LVM can improve performance when using physical disks, and it allows for advanced setups such as using multiple physical disks for one mount point, and setting up software RAID for increased performance, reliability, or both.

LVM Thin Provisioning

Using thin provisioning, you can manage a storage pool of free space, known as a thin pool, which can be allocated to an arbitrary number of devices when needed by applications. You can dynamically expand the pool when needed for cost-effective allocation of storage space.

C.2. SUPPORTED FILE SYSTEMS

This section describes the file systems available in Red Hat Enterprise Linux.

xfs

XFS is a highly scalable, high-performance file system that supports file systems up to 16 exabytes (approximately 16 million terabytes), files up to 8 exabytes (approximately 8 million terabytes), and directory structures containing tens of millions of entries. **XFS** also supports metadata journaling, which facilitates quicker crash recovery. The maximum supported size of a single XFS file system is 500 TB. **XFS** is the default and recommended file system on Red Hat Enterprise Linux.

ext4

The **ext4** file system is based on the **ext3** file system and features a number of improvements. These include support for larger file systems and larger files, faster and more efficient allocation of disk space, no limit on the number of subdirectories within a directory, faster file system checking, and more robust journaling. The maximum supported size of a single **ext4** file system is 50 TB.

ext3

The **ext3** file system is based on the **ext2** file system and has one main advantage - journaling. Using a journaling file system reduces the time spent recovering a file system after it terminates unexpectedly, as there is no need to check the file system for metadata consistency by running the fsck utility every time.

ext2

An **ext2** file system supports standard Unix file types, including regular files, directories, or symbolic links. It provides the ability to assign long file names, up to 255 characters.

swap

Swap partitions are used to support virtual memory. In other words, data is written to a swap partition when there is not enough RAM to store the data your system is processing.

vfat

The **VFAT** file system is a Linux file system that is compatible with Microsoft Windows long file names on the FAT file system.

BIOS Boot

A very small partition required for booting from a device with a GUID partition table (GPT) on BIOS systems and UEFI systems in BIOS compatibility mode.

EFI System Partition

A small partition required for booting a device with a GUID partition table (GPT) on a UEFI system.

PReP

This small boot partition is located on the first partition of the hard drive. The **PReP** boot partition contains the GRUB2 boot loader, which allows other IBM Power Systems servers to boot Red Hat Enterprise Linux.

C.3. SUPPORTED RAID TYPES

RAID stands for Redundant Array of Independent Disks, a technology which allows you to combine multiple physical disks into logical units. Some setups are designed to enhance performance at the cost of reliability, while others will improve reliability at the cost of requiring more disks for the same amount of available space.

This section describes supported software RAID types which you can use with LVM and LVM Thin Provisioning to set up storage on the installed system.

None

No RAID array will be set up.

RAID0

Performance: Distributes data across multiple disks. RAID 0 offers increased performance over standard partitions and can be used to pool the storage of multiple disks into one large virtual device. Note that RAID 0 offers no redundancy and that the failure of one device in the array destroys data in the entire array. RAID 0 requires at least two disks.

RAID1

Redundancy: Mirrors all data from one partition onto one or more other disks. Additional devices in the array provide increasing levels of redundancy. RAID 1 requires at least two disks.

RAID4

Error checking: Distributes data across multiple disks and uses one disk in the array to store parity information which safeguards the array in case any disk in the array fails. As all parity information is stored on one disk, access to this disk creates a "bottleneck" in the array's performance. RAID 4 requires at least three disks.

RAID5

Distributed error checking: Distributes data and parity information across multiple disks. RAID 5 offers the performance advantages of distributing data across multiple disks, but does not share the performance bottleneck of RAID 4 as the parity information is also distributed through the array. RAID 5 requires at least three disks.

RAID6

Redundant error checking: RAID 6 is similar to RAID 5, but instead of storing only one set of parity data, it stores two sets. RAID 6 requires at least four disks.

RAID10

Performance and redundancy: RAID 10 is nested or hybrid RAID. It is constructed by distributing data over mirrored sets of disks. For example, a RAID 10 array constructed from four RAID partitions consists of two mirrored pairs of striped partitions. RAID 10 requires at least four disks.

C.4. RECOMMENDED PARTITIONING SCHEME

Red Hat recommends that you create separate file systems at the following mount points:

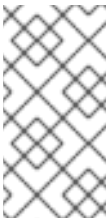
- **/boot**
- / (root)
- **/home**
- **swap**
- **/boot/efi**
- **PReP**

/boot partition - recommended size at least 1 GiB

The partition mounted on **/boot** contains the operating system kernel, which allows your system to boot Red Hat Enterprise Linux 8, along with files used during the bootstrap process. Due to the limitations of most firmwares, creating a small partition to hold these is recommended. In most scenarios, a 1 GiB boot partition is adequate. Unlike other mount points, using an LVM volume for **/boot** is not possible - **/boot** must be located on a separate disk partition.

WARNING

Normally, the **/boot** partition is created automatically by the installation program. However, if the / (root) partition is larger than 2 TiB and (U)EFI is used for booting, you need to create a separate **/boot** partition that is smaller than 2 TiB to boot the machine successfully.



NOTE

If you have a RAID card, be aware that some BIOS types do not support booting from the RAID card. In such a case, the **/boot** partition must be created on a partition outside of the RAID array, such as on a separate hard drive.

root - recommended size of 10 GiB

This is where "/", or the root directory, is located. The root directory is the top-level of the directory structure. By default, all files are written to this file system unless a different file system is mounted in the path being written to, for example, **/boot** or **/home**.
While a 5 GiB root file system allows you to install a minimal installation, it is recommended to allocate at least 10 GiB so that you can install as many package groups as you want.

IMPORTANT

Do not confuse the / directory with the **/root** directory. The **/root** directory is the home directory of the root user. The **/root** directory is sometimes referred to as *slash root* to distinguish it from the root directory.

/home - recommended size at least 1 GiB

To store user data separately from system data, create a dedicated file system for the **/home** directory. Base the file system size on the amount of data that is stored locally, number of users, and so on. You can upgrade or reinstall Red Hat Enterprise Linux 8 without erasing user data files. If you select automatic partitioning, it is recommended to have at least 55 GiB of disk space available for the installation, to ensure that the **/home** file system is created.

swap partition - recommended size at least 1 GB

Swap file systems support virtual memory; data is written to a swap file system when there is not enough RAM to store the data your system is processing. Swap size is a function of system memory workload, not total system memory and therefore is not equal to the total system memory size. It is important to analyze what applications a system will be running and the load those applications will serve in order to determine the system memory workload. Application providers and developers can provide guidance.
When the system runs out of swap space, the kernel terminates processes as the system RAM memory is exhausted. Configuring too much swap space results in storage devices being allocated but idle and is a poor use of resources. Too much swap space can also hide memory leaks. The maximum size for a swap partition and other additional information can be found in the **mkswap(8)** manual page.

The following table provides the recommended size of a swap partition depending on the amount of RAM in your system and if you want sufficient memory for your system to hibernate. If you let the installation program partition your system automatically, the swap partition size is established using these guidelines. Automatic partitioning setup assumes hibernation is not in use. The maximum size of the swap partition is limited to 10 percent of the total size of the hard drive, and the installation program cannot create swap partitions more than 128 GB in size. To set up enough swap space to allow for hibernation, or if you want to set the swap partition size to more than 10 percent of the system's storage space, or more than 128 GB, you must edit the partitioning layout manually.

/boot/efi partition - recommended size of 200 MiB

UEFI-based AMD64, Intel 64, and 64-bit ARM require a 200 MiB EFI system partition. The recommended minimum size is 200 MiB, the default size is 600 MiB, and the maximum size is 600 MiB. BIOS systems do not require an EFI system partition.

Table C.1. Recommended System Swap Space

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
Less than 2 GB	2 times the amount of RAM	3 times the amount of RAM
2 GB - 8 GB	Equal to the amount of RAM	2 times the amount of RAM
8 GB - 64 GB	4 GB to 0.5 times the amount of RAM	1.5 times the amount of RAM

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
More than 64 GB	Workload dependent (at least 4GB)	Hibernation not recommended

At the border between each range, for example, a system with 2 GB, 8 GB, or 64 GB of system RAM, discretion can be exercised with regard to chosen swap space and hibernation support. If your system resources allow for it, increasing the swap space can lead to better performance.

Distributing swap space over multiple storage devices - particularly on systems with fast drives, controllers and interfaces - also improves swap space performance.

Many systems have more partitions and volumes than the minimum required. Choose partitions based on your particular system needs.

NOTE

- Only assign storage capacity to those partitions you require immediately. You can allocate free space at any time, to meet needs as they occur.
- If you are unsure about how to configure partitions, accept the automatic default partition layout provided by the installation program.

PReP boot partition - recommended size of 4 to 8 MiB

When installing Red Hat Enterprise Linux on IBM Power System servers, the first partition of the hard drive should include a **PReP** boot partition. This contains the GRUB2 boot loader, which allows other IBM Power Systems servers to boot Red Hat Enterprise Linux.

C.5. ADVICE ON PARTITIONS

There is no best way to partition every system; the optimal setup depends on how you plan to use the system being installed. However, the following tips may help you find the optimal layout for your needs:

- Create partitions that have specific requirements first, for example, if a particular partition must be on a specific disk.
- Consider encrypting any partitions and volumes which might contain sensitive data. Encryption prevents unauthorized people from accessing the data on the partitions, even if they have access to the physical storage device. In most cases, you should at least encrypt the **/home** partition, which contains user data.
- In some cases, creating separate mount points for directories other than /, **/boot** and **/home** may be useful; for example, on a server running a **MySQL** database, having a separate mount point for **/var/lib/mysql** will allow you to preserve the database during a reinstallation without having to restore it from backup afterwards. However, having unnecessary separate mount points will make storage administration more difficult.
- Some special restrictions apply to certain directories with regards on which partitioning layouts can they be placed. Notably, the **/boot** directory must always be on a physical partition (not on an LVM volume).

- If you are new to Linux, consider reviewing the *Linux Filesystem Hierarchy Standard* at http://refspecs.linuxfoundation.org/FHS_2.3/fhs-2.3.html for information about various system directories and their contents.

- Each kernel installed on your system requires approximately 20 MB on the **/boot** partition. The default partition size of 1 GB for **/boot** should suffice for most common uses; increase the size of this partition if you plan to keep many kernels installed at the same time.

- The **/var** directory holds content for a number of applications, including the **Apache** web server, and is used by the **DNF** package manager to temporarily store downloaded package updates. Make sure that the partition or volume containing **/var** has at least 3 GB.

- The contents of the **/var** directory usually change very often. This may cause problems with older solid state drives (SSDs), as they can handle a lower number of read/write cycles before becoming unusable. If your system root is on an SSD, consider creating a separate mount point for **/var** on a classic (platter) HDD.

- The **/usr** directory holds the majority of software on a typical Red Hat Enterprise Linux installation. The partition or volume containing this directory should therefore be at least 5 GB for minimal installations, and at least 10 GB for installations with a graphical environment.

- If **/usr** or **/var** is partitioned separately from the rest of the root volume, the boot process becomes much more complex because these directories contain boot-critical components. In some situations, such as when these directories are placed on an iSCSI drive or an FCoE location, the system may either be unable to boot, or it may hang with a **Device is busy** error when powering off or rebooting.
  This limitation only applies to **/usr** or **/var**, not to directories below them. For example, a separate partition for **/var/www** will work without issues.

- Consider leaving a portion of the space in an LVM volume group unallocated. This unallocated space gives you flexibility if your space requirements change but you do not wish to remove data from other volumes. You can also select the **LVM Thin Provisioning** device type for the partition to have the unused space handled automatically by the volume.

- The size of an XFS file system can not be reduced - if you need to make a partition or volume with this file system smaller, you must back up your data, destroy the file system, and create a new, smaller one in its place. Therefore, if you expect needing to manipulate your partitioning layout later, you should use the ext4 file system instead.

- Use Logical Volume Management (LVM) if you anticipate expanding your storage by adding more hard drives or expanding virtual machine hard drives after the installation. With LVM, you can create physical volumes on the new drives, and then assign them to any volume group and logical volume as you see fit - for example, you can easily expand your system's **/home** (or any other directory residing on a logical volume).

- Creating a BIOS Boot partition or an EFI System Partition may be necessary, depending on your system's firmware, boot drive size, and boot drive disk label. See Section C.4, "Recommended partitioning scheme" for information about these partitions. Note that graphical installation will not let you create a BIOS Boot or EFI System Partition if your system does **not** require one - in that case, they will be hidden from the menu.

- If you need to make any changes to your storage configuration after the installation, Red Hat Enterprise Linux repositories offer several different tools which can help you do this. If you prefer a command line tool, try **system-storage-manager**.