Understanding the RHEL 7 Boot Procedure

To fix boot issues, it is essential to have a good understanding of the boot procedure. If issues occur during boot, you need to be able to judge in which phase of the boot procedure the issue occurs so that you can select the appropriate tool to fix the issue.The following steps summarize how the boot procedure happens on Linux.

1**. Performing POST**: The machine is powered on. From the system firmware, which can be the modern Universal Extended Firmware Interface (UEFI) or the classical Basic input Output System (BIOS), the Power-On Self-Test (POST) is executed, and the hardware that is required to start the system is initialized.

2**. Selecting the bootable** device: Either from the UEFI boot firmware or from the Master Boot Record, a bootable device is located.

3**. Loading the boot loader**: From the bootable device, a boot loader is located. On Red Hat, this is usually GRUB 2.

4**. Loading the kernel**: The boot loader may present a boot menu to the user,or can be configured to automatically start a default operating system. To load Linux, the kernel is loaded together with the initramfs. The initramfs contains kernel modules for all hardware that is required to boot, as well as the initial scripts required to proceed to the next stage of booting. On RHEL 7, the initramfs contains a complete operational system (which may be used for troubleshooting purposes).

5**. Starting /sbin/init**: Once the kernel is loaded into memory, the first of all processes is loaded, but still from the initramfs. This is the /sbin/init process,which on Red Hat is linked to systemd. The udev daemon is loaded as well to take care of further hardware initialization. All this is still happening from the initramfs image.

6**. Processing initrd.target**: The systemd process executes all units from the initrd.target, which prepares a minimal operating environment, where the root file system on disk is mounted on the /sysroot directory. At this point, enough is loaded to pass to the system installation that was written to the hard drive.

7. **Switching to the root file system**: The system switches to the root file system that is on disk and at this point can load the systemd process from disk as well.

8. **Running the default target**: Systemd looks for the default target to execute and runs all of its units. In this process, a login screen is presented, and the user can authenticate. Notice that the login prompt can be prompted before all systemd unit files have been loaded successfully. So, seeing a login prompt does not necessarily mean that your server is fully operational yet. In each of the phases listed, issues may occur because of misconfiguration or other problems.

Boot Phase Configuration and Troubleshooting Overview

Boot Phase	Configuring It	Fixing It
POST	Hardware configuration (F2, Esc, F10, or another key)	Replace hardware.
Selecting the bootable		

device
BIOS/UEFI configuration or
hardware boot menu
Replace hardware or use rescue
system.
Loading the boot loader **grub2-install** and edits to
/etc/defaults/grub
GRUB boot prompt and edits to
/etc/defaults/grub, followed by
grub2-mkconfig.
Loading the kernel Edits to the GRUB
configuration and /etc/
dracut.conf.
GRUB boot prompt and edits to
/etc/defaults/grub, followed by
grub2-mkconfig.
Starting /sbin/init Compiled into initramfs **init= kernel** boot argument,
rd.break kernel boot argument.
Processing initrd.target Compiled into initramfs Not typically required.
Switch to the root file
system
/etc/fstab /etc/fstab.
Running the default target /etc/systemd/system/
default.target
Start the rescue.target as a kernel
boot argument.

Passing Kernel Boot Arguments

If your server does not boot normally, the GRUB boot prompt offers a convenient way to stop the boot procedure and pass specific options to the kernel while booting. In this section, you learn how to access the boot prompt and how to pass specific boot arguments to the kernel while booting.

Accessing the Boot Prompt

When your server boots, you briefly see the GRUB 2 menu. Look fast because it will only last for a few seconds. From this boot menu you can type **e** to enter a mode where you can edit commands, or **c** to enter a full GRUB command prompt, as shown in Figure 19.1 . To pass boot options to a starting kernel, use **e** .

After passing an **e** to the GRUB boot menu, you'll see the interface that is in Figure 19.2 . From this interface, scroll down to locate the section that begins with linux16 /vmlinuz followed by a lot of arguments. This is the line that tells GRUB how to start a kernel, and by default it looks like this:
linux16 /vmlinuz-0-rescue-5dea58df1a3b4cb5947ddb6c78a6773f root=UUID=432d640e-3339-45fa-a66d-89da9c869550 ro rd.lvm.lv=centos/swap vconsole.font=latarcyrheb-sun16 rd.lvm.lv=centos/root crashkernel=auto vconsole.keymap=us rhgb quiet
To start, it is a good idea to remove the rhgb and quiet parts from this line;

After entering the boot options you want to use, press Ctrl+X to start the kernel with these options. Notice that these options are used one time only and are not persistent. To make them persistent you must modify the contents of the /etc/default/grub configuration

Starting a Troubleshooting Target

When you are in trouble, you have a few options that you can enter on the GRUB boot prompt:

■ **rd.break** This stops the boot procedure while still in the initramfs stage. This option is useful if you do not have the root password available. The complete procedure for recovering a missing root password follows later in this chapter.

■ **init=/bin/sh** or **init=/bin/bash** This specifies that a shell should be started immediately after loading the kernel and initrd. This is a useful option, but not the best option, because in some cases you'll lose console access or miss other functionality.

■ **systemd.unit=emergency.target** This enters in a bare minimal mode where a minimal number of systemd units is loaded. It requires a root password. To see that only a very limited number of unit files have been loaded, you can type the **systemctl list-units** command.

■ **systemd.unit=rescue.target** This starts some more systemd units to bring you in a more complete operational mode. It does require a root password. To see that only a very limited number of unit files have been loaded, you can type the **systemctl list-units** command.

Exploring troubleshooting targets.

1. (Re)start your computer. When the GRUB menu shows, select the first line in the menu and press **e** .

2. Scroll down to the line that starts with linux16 /vmlinuz. At the end of this line, type **systemd.unit=rescue.target** . Also remove the options **rhgb quit** from this line.

3. Enter the root password when you are prompted for it.

4. Type **systemctl list-units** . This shows all unit files that are currently loaded. You can see that a basic system environment has been loaded.

5. Type **systemctl show-environment** . This shows current shell environment variables.

6. Type systemctl reboot to reboot your machine.

7. When the GRUB menu shows, press e again to enter the editor mode. At the end of the line that loads the kernel, type systemd.unit=emergency.target .

8. When prompted for it, enter the root password to log in.

9. After successful login, type systemctl list-units . Notice that the number of unit files loaded is reduced to a bare minimum.

Normally, you should not ever need to use this option to troubleshoot a broken installation.

■ Rescue a Red Hat System: This is the most flexible rescue system. In Exercise 19.2 , you can explore it in detail. This should be the first option of choice when using a rescue disk.

■ Run a Memory Test: Run this option if you encounter memory errors. It allows you to mark bad memory chips so that your machine can boot normally.

■ Boot from Local Drive: If you cannot boot from GRUB on your hard disk, try this option first. It offers a boot loader that tries to install from your

machine's hard drive, and as such is the least intrusive option available.

After starting a rescue system, you usually need to enable full access to the on-disk installation. Typically, the rescue disk detects your installation and mounts it on the /mnt/sysimage directory. To fix access to the configuration files and their default locations as they should be available on disk, use the chroot /mnt/sysimage command to make the contents of this directory your actual working environment. If you do not use this chroot command, many utilities will not work, because if they write to a configuration file that would be the version of the configuration file that exists on the rescue disk (and for that reason is read-only). Using the chroot command ensures that all path references to configuration files are correct.

Using the Rescue Option

1. Restart your server from the installation disk. Select the Troubleshooting menu option.

2. From the Troubleshooting menu, select Rescue a Red Hat System . This prompts you to press Enter to start the installation. Do not worry: This option does not overwrite your current configuration; it just loads a rescue system.

3. The rescue system now prompts you that it will try to find an installed Linux system and mount on /mnt/sysimage. Press Continue to accept this option.

4. If a valid Red Hat installation was found, you are prompted that your system has been mounted under /mnt/sysimage. At this point, you can press Enter twice to access the rescue shell.

5. Your Linux installation at this point is accessible through the /mnt/sysimage directory. Type chroot /mnt/sysimage . At this point, you have access to your root file system and you can access all tools that you need to repair access to your system.

6. Type exit and reboot to restart your machine in a normal mode.

Reinstalling GRUB Using a Rescue Disk

One of the common reasons you need to start a rescue disk is because the GRUB 2 boot loader is broken. If that happens, you might need to install it again. After you have restored access to your server using a rescue disk, reinstalling GRUB 2 is not hard to do and consists of two steps:

- Make sure that you have made the contents of the /mnt/sysimage directory to your current working environment.
- Use the **grub2-install** command, followed by the name of the device on which you want to reinstall GRUB 2. So on a KVM virtual machine, the command to use is **grub2-install /dev/vda** , and on a physical server or a VMware or Virtual Box virtual machine, it is **grub2-install /dev/sda** .

Re-Creating the Initramfs Using a Rescue Disk

Occasionally, the initramfs image may get damaged as well. If this happens, you cannot boot your server into normal operational mode. To repair the initramfs image after booting into the rescue environment, you can use the **dracut** command. If used with no arguments, this command creates a new initramfs for the kernel currently loaded.
Alternatively, you can use the **dracut** command with several options to make an initramfs for specific kernel environments. There is also a configuration file with the name /etc/dracut.conf that you can use to include specific options while re-creating the initramfs. The **dracut** configuration is dispersed over different locations:
■ /usr/lib/dracut/dracut.conf.d/*.conf contains the system default configuration files.
■ /etc/dracut.conf.d contains custom dracut configuration files.
■ /etc/dracut.conf is used as the master configuration file.

Reinstalling GRUB 2

Boot loader code does not disappear just like that, but on occasion it can happen that the GRUB 2 boot code gets damaged. In that case, you better know how to reinstall GRUB 2. The exact approach depends on whether your server is still in a bootable state. If it is, it is fairly easy to reinstall GRUB 2. Just type **grub2-install** followed by the name of the device to which you want to install it. The command has many different options to fine-tune what exactly will be installed, but you probably will not need them because, by default, the command installs everything you need to make your system bootable again.
It becomes a little bit more complicated if your machine is in a nonbootable state. If that happens, you first need to start a rescue system and restore access to your server from the rescue system. (See Exercise 19.2 for the exact procedure how to do that.) After mounting your server's file systems on /mnt/sysimage and using **chroot /mnt/sysimage** to make the mounted system image your root image, it is as easy as described previously: Just run **grub2-install** to install GRUB 2 to the desired installation device. So if you are in a KVM virtual machine, run **grub2-install /dev/vda** , and if you are on a physical disk, run **grub2-install /dev/sda** .

Resetting the Root Password
A common scenario for a Linux administrator is that the root password has gone missing. If that happens, you need to reset it. The only way to do that is by booting into minimal mode, which allows you to log in without entering a password. To do so, follow these steps:
1. On system boot, press **e** when the GRUB 2 boot menu is shown.
2. Enter **rd.break** as boot argument to the line that loads the kernel and press **Ctrl+X** to boot with this option.
3. You'll now be dropped at the end of the boot stage where initramfs is loaded, just before a mount of the root file system on the directory /.
4. Type **mount -o remount,rw /sysroot** to get read/write access to the system image.
5. At this point, make the contents of the /sysimage directory your new root directory by typing **chroot /sysroot** .
6. Now you can enter **passwd** and set the new password for the user root.
7. Because at this very early boot stage SELinux has not been activated yet, the context type on /etc/shadow will be messed up. If you reboot at this point, no one will be able to log in. So you must make sure that the context type is set correctly. To do this, at this point you should load the SELinux policy by using **load_policy -i** .
8. Now you can manually set the correct context type to /etc/shadow. To do this, type **chcon -t shadow_t /etc/shadow** .
9. Reboot. You can now log in with the changed password for user root.