DevOps TROUBLESHOOTING

Linux® Server Best Practices

KYLE RANKIN

DevOps Troubleshooting

If you don't get a GRUB prompt with any of the drives, then either GRUB is completely erased or your primary disk or disk controller has failed. First go through the steps in the next section to try to repair GRUB since they will help you use a rescue disk to determine whether the disk is available at all. If the disk isn't available at all, you'll want to turn to Chapter 10 which talks about resolving hardware failures.

Fix GRUB

The difficulty in identifying and fixing problems with GRUB is in the fact that without a functioning boot loader, you can't boot into your system and use the tools you would need to repair GRUB. There are a few different ways that GRUB might be broken on your system, but before we discuss those, you should understand that in the interest of booting quickly, some systems set GRUB with a short timeout of only a few seconds before they boot the default OS, even on servers. What's worse, some systems even hide the initial GRUB prompt from the user, so you have to press a special key (Esc for GRUB 1 releases, also known as GRUB legacy, and Shift for GRUB 2, also just known as GRUB) within a second or two after your BIOS has passed off control to GRUB.

If you don't know which version of GRUB you have installed, you may have to boot the system a few times and try out both Esc and Shift to see if you can get some sort of GRUB window to display. After that, you might still have to deal with a short timeout before GRUB boots the default OS, so you'll need to press a key (arrow keys are generally safe) to disable the timeout. The following sections discuss a few of the ways GRUB might be broken and then follow up with some general approaches to repair it.

No GRUB Prompt

The first way GRUB might be broken on your system is that it could have been completely removed from your MBR. Unfortunately, since GRUB is often hidden from the user even when it works correctly, you may not be able to tell whether GRUB is configured wrong or not installed at all. Test by pressing either the Esc or Shift keys during the boot process to confirm that no GRUB prompt appears.

It's rather rare for GRUB to disappear from the MBR completely, but it most often happens on dual-boot systems where you might load both Linux and Windows. The Windows install process has long been known to wipe out the boot code in the MBR, in which case you would get no GRUB prompt at all and instead would boot directly into Windows. Dual-boot setups are fairly rare on servers, however, so most likely if GRUB was completely removed from your MBR, your only clue would be some error from the BIOS stating that it couldn't find a suitable boot device. If you have already gone through the steps listed earlier to test your boot device order in your BIOS and still get this error, somehow GRUB was erased from the MBR.

This error might also occur on systems using Linux software RAID where the primary disk may have died. While some modern installs of GRUB can automatically install themselves to the MBR on all disks involved in a RAID, if your install doesn't default to that mode (or you are using an old version of GRUB and didn't manually install GRUB to the MBR of the other disks in your RAID array), when the primary disk dies there will be no other instance of GRUB on the remaining disks you can use.

Stage 1.5 GRUB Prompt

Another way GRUB can fail is that it can still be installed in the MBR, however, for some reason it can't locate the rest of the code it needs to boot the system. Remember that GRUB's first stage has to fit in only 446 bytes inside the MBR, so it contains the code it needs to locate and load the rest of the GRUB environment. GRUB normally loads what it calls stage 1.5 (GRUB 2 calls this core.img), which contains the code that can read Linux file systems and access the final GRUB stage, stage 2. Once stage 2 or core.img is loaded, GRUB can read its default configuration file from the file system, load any extra modules it needs, and display the normal GRUB menu. When GRUB can't find the file system that contains stage 2 or its configuration files, you might be left with a message that reads "loading stage 1.5" followed by either by an error or a simple grub> prompt.

If you get an error that loading stage 1.5 failed, move on to the section that talks about how to repair GRUB. If you get as far as a grub> prompt, that means that at least stage 1.5 did load, but it might be having trouble either loading stage 2 or reading your GRUB configuration file. This can happen if the GRUB configuration file or the stage 2 file gets corrupted, or if the file system that contains those files gets corrupted (in which case you'll want to read Chapter 4 on how to repair file systems). If you are particularly savvy with GRUB commands, or don't have access to a rescue disk, it might be possible to boot your system from the basic grub> prompt by typing the same GRUB boot commands that would be configured in your GRUB configuration file. In fact, if GRUB gets as far as the final stage and displays a prompt, you can use GRUB commands to attempt to read partitions and do some basic troubleshooting. That said, most of the time it's just simpler and faster to boot into a rescue disk and repair GRUB from there.

Misconfigured GRUB Prompt

Finally, you might find that you have a full GRUB menu loaded, but when you attempt to boot the default boot entry, GRUB fails and either returns you to the boot menu or displays an error. This usually means there are errors in your GRUB configuration file and either the disk or partition that is referenced in the file has changed (or the UUID changed, more on that in the upcoming section on how to fix a system that can't mount its root file system). If you get to this point and have an alternative older kernel or a rescue mode configured in your GRUB menu, try those and see if you can boot to the system with an older config. If so, you can follow the steps in the next section to repair GRUB from the system itself. Otherwise, if you are familiar with GRUB configuration, you can press E and attempt to tinker with the GRUB configuration from the GRUB prompt, or you can boot to a rescue disk.

Repair GRUB from the Live System

If you are fortunate enough to be able to boot into your live system (possibly with an older kernel or by tinkering with GRUB options), then you might have an easier time repairing GRUB. If you can boot into your system, GRUB was probably able to at least get to stage 2 and possibly even read its configuration file, so it's clearly installed in the MBR; the next section will go over the steps to reinstall GRUB to the MBR.

Once you are booted into the system, if the problem was with your GRUB configuration file, you can simply open up the configuration file (/boot/grub/menu.lst for GRUB 1, or /etc/default/grub for GRUB 2). In the case of GRUB 2, the real configuration file is in /boot/grub/grub.cfg, but that file is usually generated by a script and isn't intended to be edited by regular users, so once you edit /etc/default/grub, you will need to run the `/usr/sbin/update-grub` script to generate the new grub.cfg file. Even in the case of GRUB 1, the menu.lst file might be automatically generated by a script like `update-grub` depending on your distribution. If so, the distribution will usually say as much in a comment at the top of the file along with providing instructions on how to edit and update the configuration file.

Repair GRUB with a Rescue Disk

Most of the time when you have a problem with GRUB, it prevents you from booting into the system to repair it, so the quickest way to repair it is with a rescue disk. Most distributions make the process simpler for you by including a rescue disk as part of the install disk either on CD-ROM or a USB image. For instance, on a Red Hat or CentOS install disk you can type `linux rescue` at the boot prompt to enter the rescue mode. On an Ubuntu install disk, the rescue mode is listed as one of the options in the boot menu. For either rescue disk you should read the official documentation to find out all of the features of the rescue environment, but we will now discuss the basic steps to restore GRUB using either disk.

In the case of the Ubuntu rescue disk, after the disk boots it will present you with an option to reinstall the GRUB boot loader. You would select this option if you got no GRUB prompt at all when the system booted. Otherwise, if you suspect you just need to regenerate your GRUB configuration file, select the option to open a shell in the root environment, run `update-grub` to rebuild the configuration file, type `exit` to leave the shell, and then reboot the system.

In the case of the Red Hat or CentOS rescue disk, boot with the `linux rescue` boot option, then type `chroot /mnt/sysimage` to mount the root partition. Once the root partition is mounted and you have a shell prompt, if you need to re-install GRUB to the MBR, type `/sbin/grub-install /dev/sda`. Replace `/dev/sda` with your root partition device (if you are unsure what the device is, type `df` at this prompt and look to see what device it claims /mnt/sysimage is). From this prompt you can also view the /boot/grub/grub.conf file in case you need to make any custom changes to the options there.

Disable Splash Screens

Back in the earlier days of Linux, the boot process was a bit more exposed to the average user. When you booted a server, screens full of text scrolled by telling you exactly what the system was doing at any particular moment. Even on server installs, many systems default to hiding a lot of that valuable debug information.

If you’ve gotten past the GRUB prompt but the system gives errors further along in the boot process, you will want to disable any splash screen or other mode that suppresses output so you can see any errors. To do this, go to the GRUB menu that displays your different boot options (you may have to hit Esc or Shift at boot to display this menu), then press E to edit the boot arguments for that specific menu entry. Look for the line that contains all of your kernel boot arguments (it might start with the word `linux` or `kernel`) and edit it to remove arguments like `quiet` and `splash`. Optionally, you may add the word `nosplash` to make sure any splash screens are disabled. Now once you exit the editing mode and boot with these new options, you should be able to see debug output as your system boots.

Can’t Mount the Root File System

Apart from GRUB errors, one of the most common boot problems is from not being able to mount the root file system. After GRUB loads the kernel and initrd file into RAM, the initrd file is expanded into an initramfs temporary root file system in RAM. This file system contains kernel modules and programs the kernel needs to locate and mount the root file system and continue the boot process. To best troubleshoot any problems in which the kernel can’t mount the root file system, it’s important to understand how the kernel knows where the root file system is to begin with.

The Root Kernel Argument

The kernel knows where the root file system is because of the `root` option passed to it by GRUB. If you were to look in a GRUB configuration file for the line that contains kernel arguments, you might see something like `root=/dev/sda2`, `root=LABEL=/`, or `root=UUID=528c6527-24bf-42d1-b908-c175f7b06a0f`. In the first example, the kernel is given an explicit disk partition, /dev/sda2. This method of specifying the root device is most common in older systems and has been replaced with either disk labels or UUIDs, because any time a disk is added or repartitioned, it's possible that what was once /dev/sda2 is now /dev/sdb2 or /dev/sda3.

To get around the problem of device names changing around, distributions started labeling partitions with their mount point, so the root partition might be labeled / or `root` and the /home partition might be labeled `home` or `/home`. Then, instead of specifying the device at the `root=` line, in GRUB you would specify the device label such as `root=LABEL=/`. That way, if the actual device names changed around, the labels would still remain the same and the kernel would be able to find the root partition.

Labels seemed to solve the problem of device names changing but introduced a different problem—what happens when two partitions are labeled the same? What started happening is that someone would add a second disk to a server that used to be in a different system. This new disk might have its own / or `/home` label already, and when added to the new system, the kernel might not end up mounting the labels you thought it should. To get around this issue, some distributions started assigning partitions UUIDs (Universal Unique Identifiers). The UUIDs are long strings of characters that are guaranteed to be unique across all disk partitions in the world, so you could add any disk to your system and feel confident that you will never have the same UUID twice. Now instead of specifying a disk label at the boot prompt, you would specify a UUID like `root=UUID=528c6527-24bf-42d1-b908-c175f7b06a0f`.

The Root Device Changed

One of the most common reasons a kernel can't mount the root partition is because the root partition it was given has changed. When this happens