Chapter 2

Virtual Machines and Automated Installations

CERTIFICATION OBJECTIVES

The management of virtual machines (VMs) and Kickstart installations are required RHCSA skills. In other words, you need to be prepared to install RHEL 7 on a VM over a network, manually, and with the help of Kickstart.

Chapter 1 covered the basics of the installation process. It assumed that you could also set up virtualization during the installation process. But it's possible that you'll need to install and configure KVM after installation is complete.

Kickstart is the Red Hat system for automated installations. It works from a text file that provides answers to the RHEL 7 installation program. With those answers, the RHEL 7 installation program can work automatically, without further intervention.

Once installation is complete on the systems used for test, study, and service, you'll want to be able to administer them remotely. Not only is an understanding of SSH connections an RHCSA requirement, but also it's an excellent skill in the real world. The references to menu options in this book are based on the GNOME desktop environment. If you're using a different desktop environment, such as KDE, the steps are somewhat different.

CERTIFICATION OBJECTIVE 2.01

Configure KVM for Red Hat

In Chapter 1, you configured a physical 64-bit RHEL 7 system with the packages required to set up VMs. If all else fails, that configuration can help you set up multiple installations of RHEL 7. But if you're faced with a RHEL installation without the needed packages, what do you do?

With the right packages, you can set up KVM modules, get access to VM configuration commands, and set up detailed configuration for a group of VMs. Some of the commands described in this section are, in a way, previews of future chapters. For example, the tools associated with updates are covered in Chapter 7. But first, it's important to discuss why anyone would want to use a VM when physical hardware is so much more tangible.

INSIDE THE EXAM

Manage Virtual Machines

The RHCSA objectives suggest that you need to know how to

- Access a virtual machine's console
- Start and stop virtual machines
- Configure systems to launch virtual machines at boot
- Install Red Hat Enterprise Linux systems as virtual guests

It's reasonably safe to assume the VMs in question are based on Red Hat's default VM solution, KVM. In Chapter 1, you set up that solution during the installation process on a

64-bit system; however, you may also need to install the associated packages on a live system during an exam. In addition, there is the Virtual Machine Manager graphical console used by Red Hat to manage such VMs. Of course, that Virtual Machine Manager is a front end to the management API provided by the libvirt library. It can also be used to install and configure a system to be started automatically during the boot process.

Although the Red Hat exam blog noted in Chapter 1 suggests that you'll take an exam on a "pre-installed" system, that doesn't preclude installations on VMs. Therefore, in this chapter, you'll learn how to set up an installation of RHEL 7 on KVM.

Kickstart Installations

The RHCSA objectives state that you need to know how to

- Install Red Hat Enterprise Linux automatically using Kickstart

To that end, every RHEL installation includes a sample Kickstart file, based on the given installation. In this chapter, you'll learn how to use that file to automate the installation process. It's a bit trickier than it sounds, because the sample Kickstart file must be modified first, beyond unique settings for different systems. But once it's configured, you'll be able to set up as many installations of RHEL as you need using that baseline Kickstart file.

Access Remote Systems and Transfer Files Securely

The RHCSA objectives state that you need to know how to

- Access remote systems using SSH
- Securely transfer files between systems

If systems administrators had to be in physical contact with every system, half of their lives would be spent en route from system to system. With tools such as the Secure Shell (SSH), administrators have the ability to do their work remotely and transfer files securely. Although SSH is automatically installed in a standard configuration in RHEL 7, custom configuration options such as key-based authentication will be covered later in the book.

Why Virtual Machines

It seems like everyone wants to get into the VM game. And they should. Enterprises had once dedicated different physical systems for every service. Actually, to ensure reliability, they may have dedicated two or more systems for each of those services. Sure, it's possible to configure multiple services on a single system. In fact, you might do so on the Red Hat exams. But in enterprises that are concerned about security, systems are frequently dedicated to individual services, to reduce the risk if one system or service is compromised.

With appropriately configured systems, each service can be configured on its own dedicated VM. You might find 10 VMs all installed on a single physical host system. As different services typically use RAM and CPU cycles at different times, it's often reasonable to "overcommit" the RAM and CPU on the local physical system. For example, on a system with 256GB of RAM, it's often reasonable to allocate 16GB each to 20 VMs configured on that system.

In practice, an administrator might replace 20 physical machines on an older network with two physical systems. Each of the 20 VMs would be installed on a shared storage volume, formatted with a clustered filesystem such as GFS2, and mounted on each physical system. Of course, those two physical systems require some powerful hardware. But the savings otherwise are immense, not only in overall hardware costs, but also in facilities, energy consumption, and more.

If You Have to Install KVM

If you have to install any sort of software on RHEL 7, the GNOME Software tool can be a great help. Log in to the GUI as a regular user. To open it from the GUI, click Applications | System Tools | Software. As long as there's an appropriate connection to repositories such as the RHN or those associated with third parties, it'll take a few moments to search. In the left pane, click the arrow next to Virtualization. The four virtualization package groups should appear. Click the Virtualization Hypervisor package group and then the first package in that group to see a screen similar to that shown in Figure 2-1.

FIGURE 2-1

Add/Remove Software tool

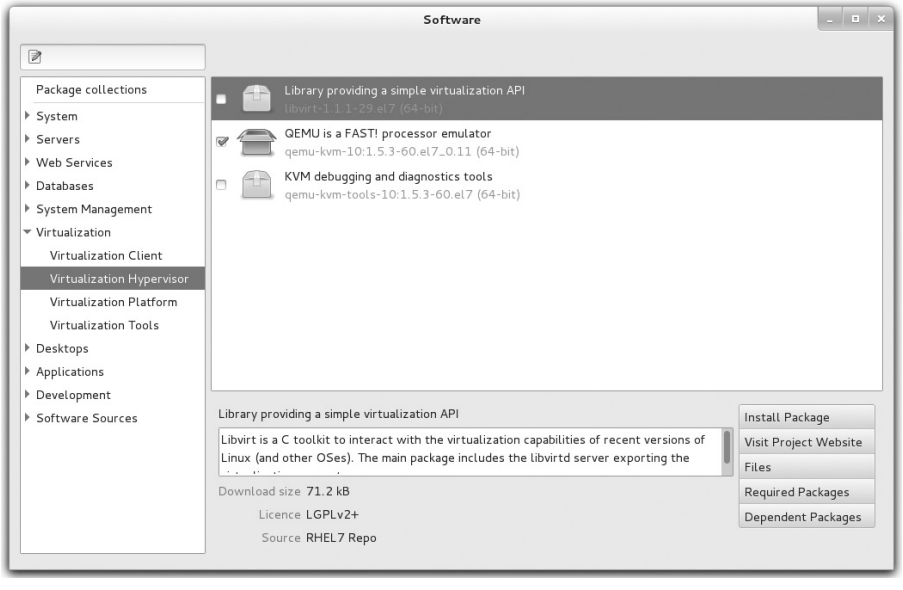


TABLE 2-1

Packages Associated with Virtualization

Package	Description
qemu-kvm	The main KVM package
libvirt	The libvirtd service to manage hypervisors
libvirt-client	The virsh command and clients API to manage virtual machines
virt-install	Command-line tools for creating VMs
virt-manager	GUI VM administration tool
virt-top	Command to display virtualization statistics
virt-viewer	Graphical console to connect to VMs

All you need to do to install KVM is select appropriate packages from the Virtualization Hypervisor, Virtualization Client, and Virtualization Platform package groups. If you don't remember the list shown in Table 2-1, just install the latest version of all virtualization packages.

That's just seven packages! Of course, in most configurations, they'll pull in other packages as dependencies. But that's all you really need to configure VMs on a physical RHEL 7 system. Although the Virtualization Tools group does not have any mandatory packages, it includes software that may be helpful in real life, such as tools that can help read and manage the VM disk images. If you want to display the content of a VM disk or manage VM partitions and filesystems from the hypervisor host, the libguestsfs-tools package is what you need.

Installation with the GNOME Software tool is fairly simple. Just select (or deselect) the desired packages and click Apply. If there are dependent packages that also require installation, you'll be prompted with the full list of those packages. Of course, from the command-line interface, you can install these packages one at a time with the **yum install *packagename*** command. As an alternative, install the Virtualization Host and Virtualization Client groups, as shown here:

```
# yum group install "Virtualization Host" "Virtualization Client"
```

You will learn more about yum and package groups in Chapter 7.

The Right KVM Modules

In most cases, installation of the right packages is good enough. Appropriate kernel modules should be loaded automatically. Before KVM can work, the associated kernel modules must be loaded. Run the following command:

```
# lsmod | grep kvm
```

If KVM modules are properly loaded, you'll see one of the following two sets of modules:

```
kvm_intel     138567  0
kvm           441119  1 kvm_intel
```

or

```
kvm_amd        59887   0
kvm            261575  1 kvm_amd
```

As the module names suggest, the output depends on the CPU manufacturer. If you don't get this output, first make sure the hardware is suitable. And as suggested in Chapter 1, make sure the **svm** or **vmx** flag is listed in the contents of the **/proc/cpuinfo** file. Otherwise, additional configuration may be required in the system BIOS or UEFI menu. Some menus include specific options for hardware virtualization, which should be enabled.

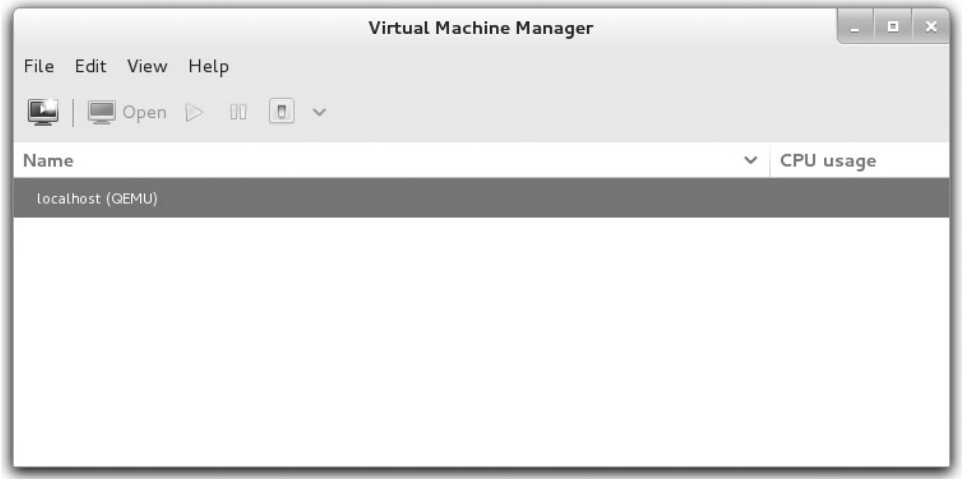
If one of the noted flags exists in the **/proc/cpuinfo** file, the next step is to try loading the applicable modules. The simplest method is with the **modprobe** command. The following command should also load the dependent KVM module. If the system has an AMD processor, replace **kvm_intel** with **kvm_amd**:

```
# modprobe kvm_intel
```

Configure the Virtual Machine Manager

The Virtual Machine Manager is part of the **virt-manager** package. And you can start it in a GUI with the command of the same name. Alternatively, in the GNOME desktop click Applications | System Tools | Virtual Machine Manager. It opens the Virtual Machine Manager window shown in Figure 2-2.

FIGURE 2-2 Virtual Machine Manager



If desired, the KVM-based VMs can be configured and administered remotely. All you need to do is connect to the remote hypervisor. To do so, click File | Add Connection. This opens an Add Connection window that allows you to select the following:

- A Linux container or a hypervisor, normally KVM or Xen. (Xen was the default hypervisor on RHEL 5, but has not been supported since RHEL 6.)
- A connection, which may be local or remote, using a connection method such as SSH.

Remote connections can be given with the hostname or IP address of the remote system.

Configuration by Hypervisor

Each hypervisor can be configured in some detail. Right-click the localhost (QEMU) hypervisor and select Details in the pop-up menu that appears. This opens a details window named after the host of the local system, as shown in Figure 2-3.

FIGURE 2-3 VM host details

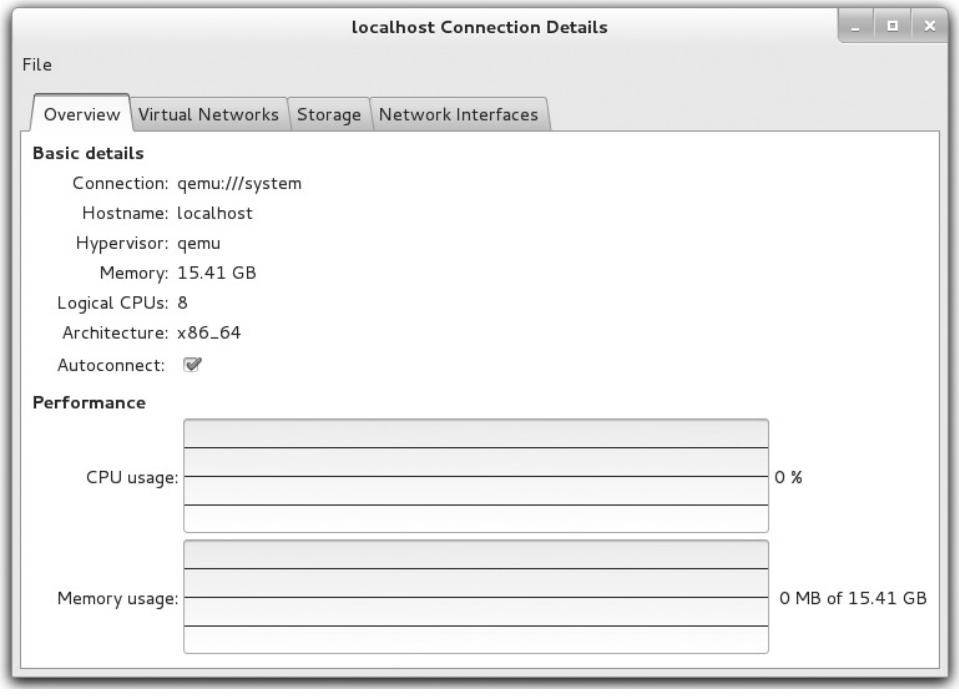


TABLE 2-2

VM Host Details

Setting	Description
Connection	Universal Resource Identifier (URI) for the hypervisor.
Hostname	Hostname for the VM host.
Hypervisor	QEMU is used by KVM.
Memory	Available RAM from the physical system for VMs.
Logical CPUs	Available logical CPUs; this is a quad-core system with hyper-threading enabled, which gives eight logical CPUs.
Architecture	CPU architecture.
Autoconnect	Indicates whether to automatically connect to the hypervisor during the boot process.

As shown in Table 2-2, the Overview tab lists the basics of the VM configuration. For the next section, stay in the host details window for the current hypervisor.

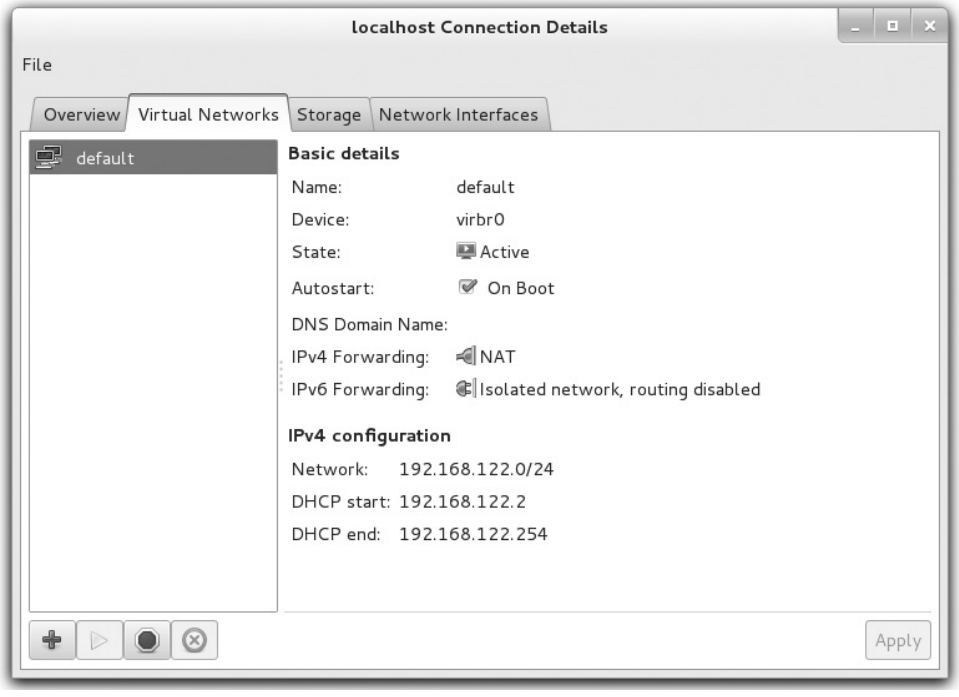
Virtual Networks on a Hypervisor

Now you'll examine the networks configured for VMs within the Virtual Machine Manager. In the host details window for the current hypervisor, click the Virtual Networks tab. The default virtual network shown in Figure 2-4 illustrates the standard network for VMs created with this hypervisor.

You'll note the given network is configured to start automatically when the VM is booted. So if there's an appropriate virtual network adapter configured on the VM, along with a client command associated with the Dynamic Host Configuration Protocol (DHCP), it's automatically given an IP address from the noted range. As noted in the figure, assigned addresses are configured to be translated using Network Address Translation (NAT) when traffic is forwarded to a physical network adapter.

With the buttons in the lower-left part of the screen, you can add a new virtual network, start and stop an existing virtual network, and delete that network. In Exercise 2-1, you'll create a second virtual network.

FIGURE 2-4 VM network details



EXERCISE 2-1

Create a Second Virtual Network

In this exercise, you'll create a second virtual network on the standard KVM hypervisor in the GUI Virtual Machine Manager. This exercise requires a RHEL 7 system configured as a Virtualization Host, as discussed early in this chapter.

1. If you do not have the details window open, right-click the standard localhost (QEMU) hypervisor. In the pop-up menu that appears, select Details.
2. In the Host Details window that appears with the name of the local system, select the Virtual Networks tab.
3. Click the plus sign in the lower-left corner of the Virtual Networks tab to open the Create a New Virtual Network Wizard.

4. Read the instructions, which you will follow in the coming steps. Click Forward to continue.
5. Assign a name for the new virtual network. For the purpose of this book, enter the name **outsider**. Click Forward to continue.
6. If not already input, type in the **192.168.100.0/24** network address in the Network text box. The system automatically calculates plausible entries for other network information, as shown in the illustration.

on the job

Take care to avoid IP address conflicts with existing hardware on the local network, such as with routers and wireless access points. For example, if a cable modem uses IP address 192.168.100.1 on its interface, the noted 192.168.100.0/24 network on the hypervisor would make that cable modem inaccessible from the Linux host. If you have such hardware, change the network address shown in the illustration.

7. Now you can select the range of IP addresses within the configured network that can be assigned to a DHCP client. Per Table 1-2 in Chapter 1, you'll configure a static IP address for the **outsider1.example.org** system on this network. As long as the noted 192.168.100.100 IP address is outside the range of DHCP-assignable IP addresses, no changes are required. Make any needed changes and click Forward to continue.
8. Optionally, you can define an IPv6 address range. IPv6 is part of the RHCE objectives and will be covered in Chapter 12. Click Forward to continue.
9. Now you'll want a system that forwards network packets to the physical network, if only because that's how systems on this network communicate with systems on different virtual networks, possibly on different virtual hosts. The destination can be Any Physical Device, in NAT mode, to help hide these systems from remote hosts. Unless you want to limit routing from VMs to a specific physical network card, the defaults under Forwarding to Physical Network should work. The options are covered later in this chapter, in the discussion of the Network Interfaces tab. Make appropriate selections and click Forward to continue.
10. Review the summary of what has been configured. If you are satisfied, click Finish. The outsider network will now be available for use by new VM systems and network cards.

Virtual Storage on a Hypervisor

Now you'll examine the virtual storage configured for VMs within the Virtual Machine Manager. In the host details window for the current hypervisor, click the Storage tab. The default filesystem directory shown in Figure 2-5 configures the /var/lib/libvirt/images directory for virtual images. Such images are essentially huge files of reserved space used as hard drives for VMs.

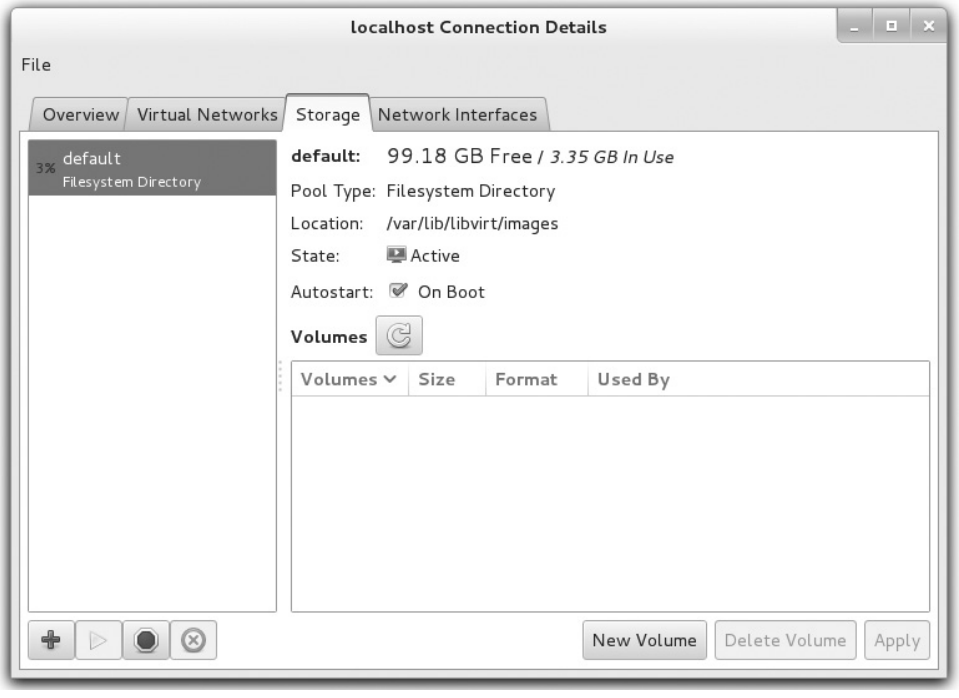
Those huge files can easily overwhelm many systems. One way to get control over such files is to dedicate a partition or logical volume to that /var/lib/libvirt/images directory.

Because you may have already dedicated the largest amount of free space to a partition for your /home directory, you can create dedicated storage in that area. As an example, the user "michael" can have a /home/michael/KVM directory to contain his VM files used for virtual hard drives.

The following commands will create the appropriate directory as a regular user, log in as the root user, set the appropriate SELinux contexts, remove the /var/lib/libvirt/images directory, and re-create that directory as a link to the appropriate user directory:

```
$ mkdir /home/michael/KVM
$ su - root
# semanage fcontext -a -t virt_image_t '/home/michael/KVM(/.*)?'
# restorecon /home/michael/KVM
# rmdir /var/lib/libvirt/images
# ln -s /home/michael/KVM /var/lib/libvirt/images
```

FIGURE 2-5 VM storage details



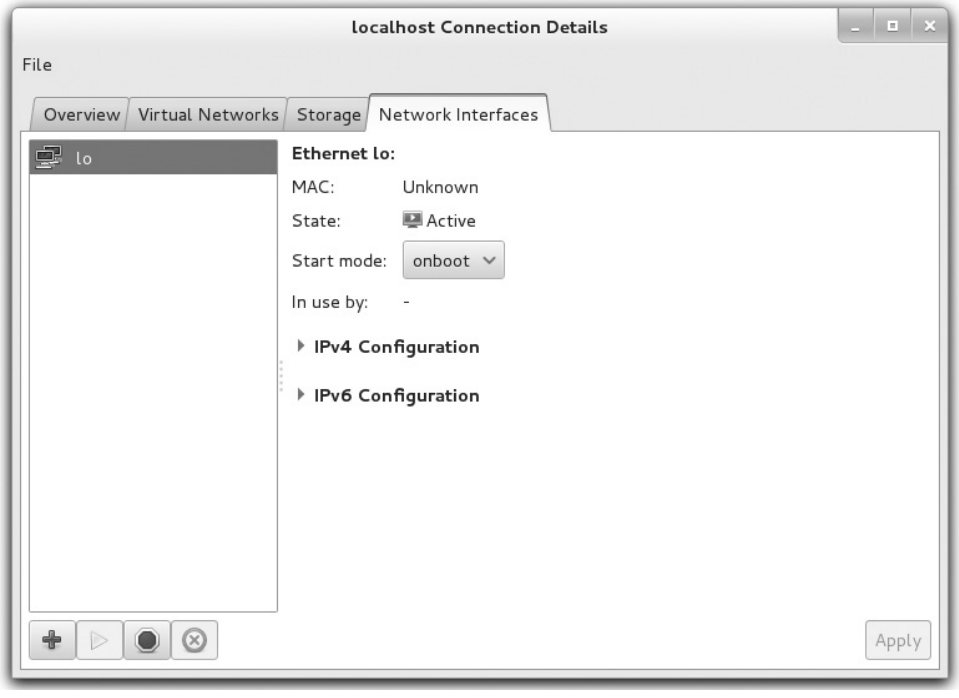
One advantage of this setup is that it retains the default SELinux settings for the /var /lib/libvirt/images directory, as defined in the file_contexts file in the /etc/selinux/targeted /contexts/files directory. In other words, this configuration survives a relabel of SELinux, as explained in Chapter 4.

Network Interfaces on a Hypervisor

Now you'll examine the network interfaces configured for VMs within the Virtual Machine Manager. In the host details window for the current hypervisor, click the Network Interfaces tab. The network interface device shown in Figure 2-6 specifies only the loopback interface. You may see other interfaces, such as an Ethernet adapter, if this is installed on your system.

If the local system connects via an Ethernet network card or wireless adapter, the default configuration should be sufficient. A properly configured VM should have access to

FIGURE 2-6 VM network cards



external networks, given the firewall and IP forwarding configuration options described in Chapter 1. In RHEL 7, each virtual network is associated to a virtual switch, such as virbr0. Virtual switches operate in NAT mode by default when traffic is forwarded outside the physical host.

In the same fashion as with the Virtual Network and Storage tabs, you can configure another network interface by clicking the plus sign in the lower-left corner of the Network Interfaces tab. It opens a Configure Network Interfaces window that can help you configure one of four different types of network interfaces:

- **Bridge** Bridges a physical and a virtual interface
- **Bond** Combines two or more interfaces in a single logical interface for redundancy
- **Ethernet** Configures an interface
- **VLAN** Configures an interface with IEEE 802.1Q VLAN tagging

CERTIFICATION OBJECTIVE 2.02

Configure a Virtual Machine on KVM

The process for configuring a VM on KVM is straightforward, especially from the Virtual Machine Manager. In essence, all you have to do is right-click the QEMU hypervisor, click New, and follow the prompts that appear. However, because it's important to understand the process in detail, you'll read about it, step by step. New VMs can be configured not only from the GUI, but also from the command-line interface. As is common for Linux services, the resulting VM configuration is stored as a text file.

Configure a Virtual Machine on KVM

To follow along with this section, open the Virtual Machine Manager in the GUI. Another way to do so is from a GUI-based command line (by running the **virt-manager** command). If prompted, enter the root administrative password. If the localhost (QEMU) hypervisor is shown as not connected, right-click it and select Connect in the pop-up menu that appears. With the following steps, you'll set up the VM associated with the server1.example.com system discussed in Chapter 1. Now to set up a new VM, take the following steps:

1. Right-click the localhost (QEMU) hypervisor. In the pop-up menu that appears, click New to open the New VM window shown in Figure 2-7.

FIGURE 2-7

Create a new VM

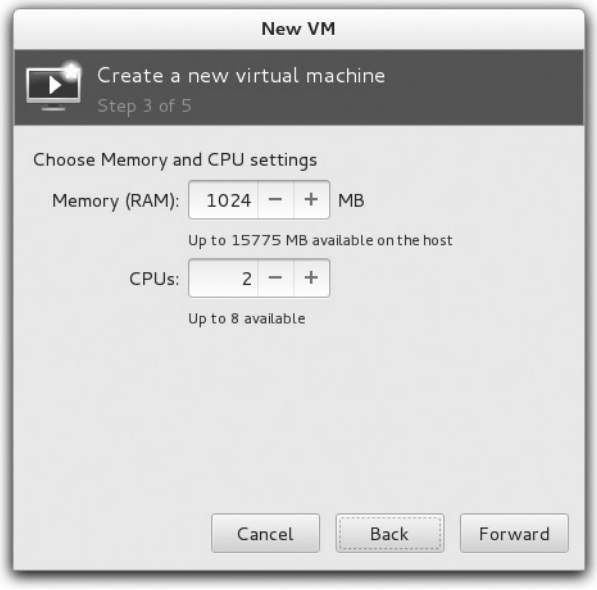
2. Type in a name for the new VM; to match the discussion in the remainder of this book, you should name this VM **server1.example.com**.
3. Now select whether the installation media is available on local install media (ISO image or CD-ROM) or from a network installation server. That server may be associated with the HTTP, NFS, or FTP protocol. Select the Local Install Media option and click Forward to continue. (In Lab 1, you'll rerun this process with the Network Install option.)
4. If the media is available in a local CD/DVD drive, an option for such will be available, as shown in Figure 2-8. But for the purpose of these steps, select Use ISO Image and click Browse to navigate to the location of the RHEL 7 DVD or Network Boot ISO image. In addition, you'll need to use the OS Type and Version drop-down text boxes to select an operating system type and distribution, as shown.
5. Choose the amount of RAM memory and number of CPUs to allocate to the VM. Be aware of the minimums described earlier in this chapter and Chapter 1 for RHEL 7. As shown in Figure 2-9, in smaller print, you'll see information about available RAM and CPUs. Make appropriate selections and click Forward to continue.

FIGURE 2-8

Virtual machine media installation options

FIGURE 2-9

Virtual machine RAM and CPU selection



6. Now you'll set up the hard drive for the VM, in the screen shown in Figure 2-10. Although it's possible to set it up in dedicated physical volumes, the standard is to set up big files as virtual hard drives. While the default location for such files is the /var /lib/libvirt/images directory, it can be changed, as discussed earlier in this chapter. On an exam, it's likely that you'll have more than sufficient room in the /var/lib/libvirt /images directory. The Select Managed or Other Existing Storage option supports the creation of a virtual hard drive in a different preconfigured storage pool.
7. Make sure the virtual drive is 16GB and the Allocate Entire Disk Now option is selected and click Forward to continue.
8. In the next window, confirm the options selected so far. Click Advanced Options to open the selections shown in Figure 2-11.

   You may have options to select one of the available virtual networks. If you performed Exercise 2-1, the "outsider" virtual network, associated to the IP subnet 192.168.100.0/24, should also be available.
9. The system may take a little time to create the VM, including the large file that will serve as the virtual hard drive. When complete, the Virtual Machine Manager should automatically start the system from the RHEL 7 installation DVD in a console window.

FIGURE 2-10

Create a virtual hard drive

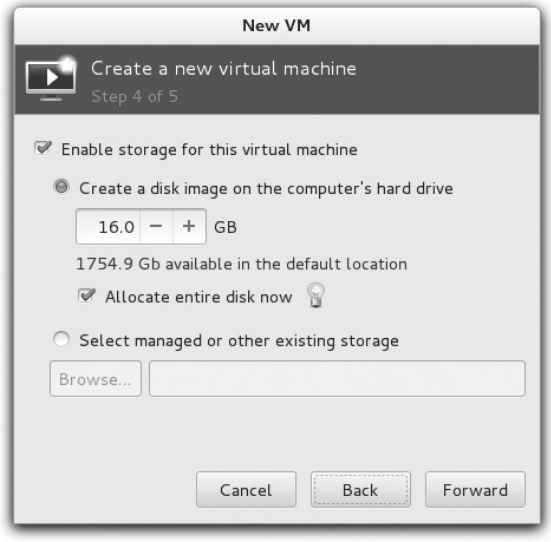
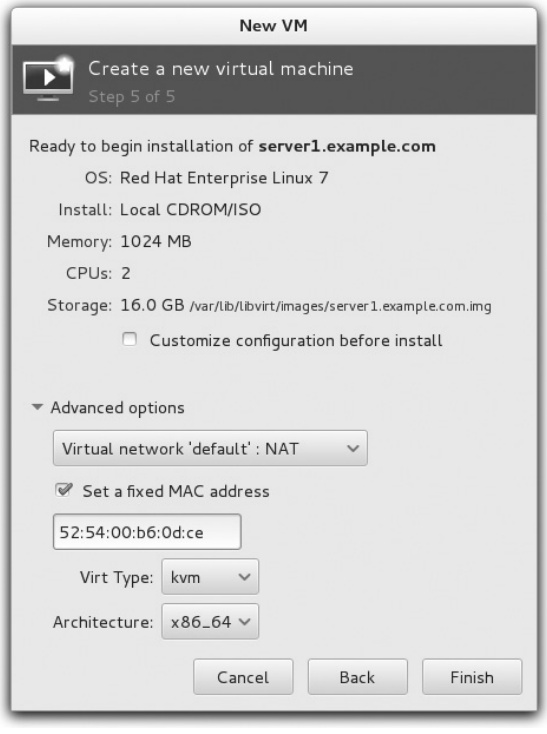


FIGURE 2-11

Review the configuration options



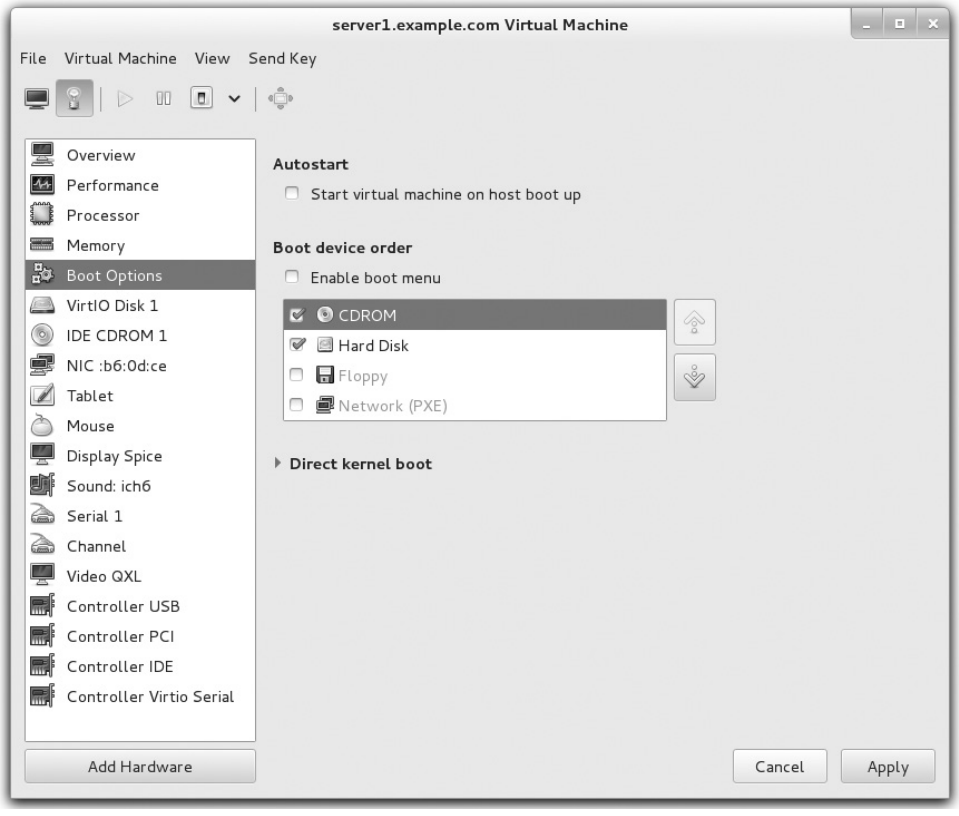
10. If the new system doesn't start automatically, that VM should be listed in the Virtual Machine Manager shown back in Figure 2-2. You should then be able to highlight the new VM (in this case, named server1.example.org) and click Open.
11. You should now be able to proceed with the installation of RHEL 7 in the VM as discussed in Chapter 1.
12. If you reboot the VM, the installation program "ejects" the DVD. If you want to reconnect the DVD later, you have to click View | Details, select the IDE CDROM 1 option, click Disconnect, and then click Connect. In the Choose Media window that appears, select the appropriate file with the DVD ISO image or CD-ROM for physical media.
13. Be aware that when you select software to install, this system is a virtual guest, not a virtual host configured in Chapter 1. There is no need to add any virtualization packages to the installation. Select Server with GUI without specifying any of the optional add-ons, and click Done.
14. When the installation is complete, click Reboot. If the system tries to boot from the DVD drive again, you'll need to change the boot order between the DVD and the hard drive. If the system boots directly from the hard drive, you're done!
15. If the system tries to boot from the DVD, you need to shut down the system. To do so, click Virtual Machine | Shut Down | Shut Down.
16. If this is the first time you've run that command sequence, the Virtual Machine Manager prompts for confirmation. Click Yes.
17. Now click View | Details.
18. In the left pane, select Boot Options, as shown in Figure 2-12.
19. One way to change the boot order is to highlight CDROM and then click the down-arrow button. Click Apply; otherwise, the changes won't be recorded.
20. Now click View | Console and then Virtual Machine | Run. The system should now boot normally into the Initial Setup screen discussed in Chapter 1.

exam Watch

The steps discussed in this section describe how to meet the RHCSA objective to "access a virtual machine's console." It also suggests one method that you can use to "start and stop virtual machines."

One more reason for the use of VMs is the ease with which additional virtual hard drives can be added. The process varies by VM solution. For the RHEL 7 default Virtual Machine Manager with KVM solution, you can do so from the machine window by clicking View | Details. You'll see an Add Hardware option in this screen.

FIGURE 2-12 Boot options in the VM



EXERCISE 2-2

Add Virtual Hard Drives

In this exercise, you'll create an additional virtual hard drive on a KVM-based VM. We assume there's an existing KVM VM for that purpose, along with the use of the GUI Virtual Machine Manager.

1. Open the Virtual Machine Manager. From the command line in a GUI, run the **virt-manager** command.
2. If prompted, enter the root administrative password and click Authenticate.

3. Highlight the localhost (QEMU) hypervisor. If it isn't already connected, right-click it and select Connect from the pop-up menu that appears. This step may happen automatically.
4. Right-click an existing VM, and click Open in the pop-up menu that appears.
5. Click View | Details. In the bottom-left corner of the window that opens, click Add Hardware.
6. In the Add New Virtual Hardware window that appears, select Storage from the menu on the left.
7. In the Storage window (shown next), set up a 1.0GB drive, select Allocate Entire Disk Now, and select the Virtio Disk device type in default cache mode. (You can also select a SATA or IDE disk.) Make desired choices and click Forward to continue.

8. You may see a confirmation of selected settings. If satisfied, click Finish to create the new virtual hard drive.
9. Repeat previous steps to create a second 1GB hard drive.
10. The next time you boot this system, run the **fdisk -l** command from the root account. It should confirm appropriate information about the configured hard drive devices.

FIGURE 2-13 The configuration file for a KVM virtual machine

```
<domain type='kvm'>
  <name>server1.example.com</name>
  <uuid>7782a007-60eb-4292-b731-8b2b60594933</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-rhel7.0.0'>hvm</type>
    <boot dev='hd'/>
    <bootmenu enable='no'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source file='/var/lib/libvirt/images/server1.example.com.img'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
    </disk>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source file='/var/lib/libvirt/images/server1.example.com-1.img'/>
      <target dev='hda' bus='ide'/>
      <address type='drive' controller='0' bus='0' target='0' unit='0'/>
    </disk>
```

KVM Configuration Files

KVM-based VMs are normally configured in two different directories: /etc/libvirt and /var/lib/libvirt. When a KVM VM is configured, it is set up in files in XML format in the /etc/libvirt/qemu directory. For example, Figure 2-13 shows a relevant excerpt of the configuration file for the main VM we used to help prepare this book (server1.example.com.xml).

Important parameters for the VM are labeled. For example, the amount of memory is shown in KiB (1 KiB = 1024 bytes), two virtual CPUs are allocated, KVM is the emulator, the disk can be found in the server1.example.com.img file in the /var/lib/libvirt/images directory, and so on.

Although you can edit this configuration file directly, changes aren't implemented until the **libvirtd** service is restarted with a command such as **systemctl restart libvirtd**.

Control Virtual Machines from the Command Line

Of course, command-line tools can be used to create, clone, convert, and install VMs on RHEL 7. The key commands to that end are **virt-install**, **virsh**, and **virt-clone**. The **virsh** command is an especially useful way to address two different RHCSA objectives.

The virt-install Command

You can perform the same steps as was done earlier in this chapter using the Virtual Machine Manager. All you need is the **virt-install** command. The command with the ***--help*** switch shows all the options for the required information described earlier. Look at the command help screen and compare with the example shown in Figure 2-14.

exam Watch

The virt-install **command is one method to address the RHCSA objective "Install Red Hat Enterprise Linux systems as virtual guests."**

For many, that's simpler than configuring the GUI Virtual Machine Manager. The Creating Domain message at the end of Figure 2-14 starts a console window with a graphical view of the given installation program. If you get a "cannot open display" error, ensure that you have opened a GNOME Desktop session as root.

If you make a mistake with the **virt-install** command, you can abort the process by pressing CTRL-C. But be aware that the newly created VM is still running. And there's now a configuration file and virtual disk for that VM. If you try to rerun the **virt-install** command with the same name for the VM, an error message will appear. Therefore, if you want to use the same name for the VM, take the following steps:

1. Stop the VM just created. If it's the tester1.example.com system shown in Figure 2-14, you can do so with the following command:

```
# virsh destroy tester1.example.com
```

FIGURE 2-14

Configure a VM with the virt-install command

```
[root@Maui ~]# virt-install --name=tester1.example.com \
> --ram=1024 --vcpus=2 \
> --disk path=/var/lib/libvirt/images/tester1.example.com.img,size=16 \
> --graphics=spice \
> --location=ftp://192.168.122.1/pub/inst \
> --os-type=linux \
> --os-variant=rhel7

Starting install...
Retrieving file .treeinfo...                          | 4.2 kB     00:00 !!!
Retrieving file vmlinuz...                            | 9.3 MB     00:00 !!!
Retrieving file initrd.img...                         |  68 MB     00:00 !!!
Allocating 'tester1.example.com.img'                  |  16 GB     00:00
Creating domain...                                    |    0 B     00:00
Domain installation still in progress. You can reconnect to
the console to complete the installation process.
```

2. Delete the associated XML configuration file in the /etc/libvirt/qemu directory and the virtual disk file, normally created in the /var/lib/libvirt/images directory. However, this is not necessary if you want to reuse the file.

```
# virsh undefine tester1.example.com --remove-all-storage
```

3. Now you'll be able to run the **virt-install** command again with the same name for the VM.

The virt-install Command and Kickstart

For Kickstart installations described later in this chapter, the **virt-install** command can be used to cite a Kickstart configuration file. For that purpose, you'll need to understand some of the key switches associated with the **virt-install** command, as shown in Table 2-3.

For example, the following **virt-install** command will install a system named outsider1.example.org automatically using a Kickstart file named ks1.cfg from the FTP server on the noted IP address, with 1GB of RAM, and an outsider1.example.org.img virtual disk.

```
# virt-install -n outsider1.example.org -r 1024 --disk \
path=/var/lib/libvirt/images/outsider1.example.org.img,size=16 \
-l ftp://192.168.122.1/pub/inst \
-x ks=ftp://192.168.122.1/pub/ks1.cfg
```

This command contains a number of switches. Most of the switches shown are described in the examples listed in the man page for the **virt-install** command. You may note additional command options, which are helpful but are not required for RHEL 7 installation. However,

TABLE 2-3

Command Switches for virt-install

Switch	Description
-n (--name)	Sets the name for the VM.
--vcpus	Configures the number of virtual CPUs.
-r (--ram)	Configures the amount of RAM in MB.
--disk	Defines the virtual disk; often used with **path=/var/lib/libvirt/images/*virt*.img,size=*size_in_GB***.
-l (--location)	Specifies the directory or URL with the installation files (equivalent to **--location**).
--graphics	Specifies the graphical display settings of the guest; valid options are **vnc**, **spice**, and **none**.
-x (--extra-args=)	Includes extra data, such as the URL of a Kickstart file.

they are required to look for the given Kickstart file. Remember the format for the extra arguments, with the quotes, which may also be expressed as follows:

```
--extra-args="ks=ftp://192.168.122.1/pub/ks1.cfg"
```

The virsh Command

The **virsh** command starts a front end to existing KVM VMs. When run alone, it moves from a regular command line to the following prompt:

```
virsh #
```

From that prompt, run the **help** command. It includes access to a number of commands, some of which are listed in Table 2-4. Not all of those commands shown in the output to the **help** screen are active for KVM. Those **virsh** commands that are usable can also be run directly from the bash shell prompt; for example, the **virsh list --all** command lists all configured VMs, whether or not they're currently running. In the context of KVM, an instance of an operating system running on a VM is a *domain*. Domain names are referenced by different **virsh** commands.

Take a look at the output to the **virsh list --all** command on our system:

```
 Id Name                 State
----------------------------------
  - server1.example.com  shut off
  - tester1.example.com  shut off
```

With the right **virsh** commands, you can meet two RHCSA objectives. First, the following command starts the noted server1.example.com system:

```
# virsh start server1.example.com
```

TABLE 2-4

Selected Commands at the virsh Prompt

virsh Command	Description
autostart <domain>	Configures a domain to be started during the host system boot process
capabilities	Lists the abilities of the local hypervisor
edit <domain>	Edits the XML configuration file for the domain
list --all	Lists all domains
start <domain>	Boots the given domain
shutdown <domain>	Gracefully shuts down the given domain

The **virsh shutdown** command gracefully shut downs the operating system and powers off a VM:

```
# virsh shutdown server1.example.com
```

To immediately power off a virtual machine, you'd have to run a somewhat more severe command:

```
# virsh destroy server1.example.com
```

The **virsh destroy** command is functionally equivalent to disconnecting the power cord on a physical system. As that can lead to different problems, it's best to stop a VM by running the **poweroff** command from within the VM.

Even on the most protected systems, power failures happen. Kernel updates still require a system reboot. In those cases, it's helpful to automate the start of VMs on a virtual host during the boot process.

exam Watch

To start and stop a VM, you can run the virsh start *vmname* **and** virsh destroy *vmname* **commands, where** *vmname* **is the domain name of the VM, as shown in the output to the** virsh list --all **command.**

In addition, the **virsh** command is the most straightforward way to make sure a VM is started the next time a system is booted. For example, the following command configures the noted tester1.example.com system to start during the boot process of the host system:

```
# virsh autostart tester1.example.com
```

Once the boot process is complete for both the host and the VM, you'll be able to use commands such as **ssh** to connect to that VM system normally. However, from the physical host GUI, you'll still have to start the Virtual Machine Manager and connect to the associated hypervisor to actually access the virtual console for that tester1.example.com system.

The command creates a soft linked file in the /etc/libvirt/qemu/autostart directory. To reverse the process, either run the command

```
# virsh autostart --disable tester1.example.com
```

or delete the soft linked file named after the target VM from that directory.

exam Watch

To configure a VM to start automatically when a system is booted, you can run the virsh autostart *vmname* **command, where** *vmname* **is the name of the VM, as shown in the output to the** virsh list --all **command.**

The virt-clone Command

The **virt-clone** command can be used to clone an existing VM. Before starting the process, make sure the system to be cloned is shut down. It's straightforward; one example where a **tester1.example.com** system is created from a **server1.example.com** system is shown in Figure 2-15.

FIGURE 2-15

Cloning a virtual machine

```
[root@Maui ~]# virt-clone --original=server1.example.com \
> --name=tester1.example.com \
> --file=/var/lib/libvirt/images/tester1.example.com.img \
> --file=/var/lib/libvirt/images/tester1.example.com-1.img \
> --file=/var/lib/libvirt/images/tester1.example.com-2.img
Allocating 'tester1.example.com.img'                    |  16 GB     00:46
Allocating 'tester1.example.com-1.img'                  | 1.0 GB     00:00
Allocating 'tester1.example.com-2.img'                  | 1.0 GB     00:00

Clone 'tester1.example.com' created successfully.
[root@Maui ~]#
```

Note that you must specify a path to a virtual disk using the ***--file*** switch for every disk of the original virtual machine that you want to clone. In this case, server1.example.com has three virtual drives because we added two new drives in Exercise 2-2.

Once the process is complete, not only will you find the noted hard drive images in the specified directories, but also you'll find a new XML configuration file for that VM in the /etc/libvirt/qemu directory.

The first time you boot a cloned machine, it may be best to boot it into the rescue target. The rescue target does not start most services, including networking (for more information, see Chapter 5). In that case, you'll be able to modify any networking settings, such as the hostname and IP address, before starting that cloned machine on a production network. In addition, you'll want to make sure that the hardware (MAC) address for the related network card is different from that of the source guest to avoid conflicts with the original network card.

Although that process may not be difficult for one or two VMs, imagine setting up a few dozen VMs, each later configured for different services. That situation would be helped by more automation. To that end, Red Hat provides a system known as Kickstart.

CERTIFICATION OBJECTIVE 2.03

Automated Installation Options

Kickstart is Red Hat's solution for an automated installation of RHEL. Think of each of the steps performed during the installation process as questions. With Kickstart, each of those questions can be answered automatically with one text file. With Kickstart, you can set up identical systems very quickly. To that end, Kickstart files are useful for quick deployment and distribution of Linux systems.

In addition, the installation process is an opportunity to learn more about RHEL 7—not only the boot media, but the partitions and logical volumes that can be configured after installation is complete. With the advent of VMs, it isn't difficult to set up an automated installation on a new VM with the help of Kickstart.

The steps described in this section assume a connection to the FTP server with RHEL 7 installation files created and configured in Lab 2 of Chapter 1.

Kickstart Concepts

One of the problems with a Kickstart-based installation is that it does not include the custom settings created after the basic installation was complete. Although it's possible to include those settings based on post-installation scripts, that's beyond the scope of the RHCSA exam.

There are two methods for creating the required Kickstart configuration file:

- Start with the anaconda-ks.cfg file from the root user's home directory, /root.
- Use the graphical Kickstart Configurator, accessible via the **system-config-kickstart** command.

exam Watch

While it's a good idea to monitor https://bugzilla.redhat.com for bugs related to key components, it may be especially important with respect to Kickstart. For example, bug 1121008 suggests that prior to Anaconda version 19.31.83-1, NFS-based Kickstart installs with custom mount options were problematic.

The first option lets you use the Kickstart template file created for the local system by Anaconda: anaconda-ks.cfg in the /root directory. The second option, the Kickstart Configurator, is discussed in detail later in this chapter.

It's relatively easy to customize the anaconda-ks.cfg file for different systems. Shortly, you'll see how to customize that file as needed for different hard disk sizes, hostnames, IP addresses, and more.

Set Up Local Access to Kickstart

Once the Kickstart file is configured, you can set it up on local media such as a USB key, a CD, a spare partition, or even a floppy drive. (Don't laugh; many VM systems, including KVM, make it easy to use virtual floppy drives.) To do so, follow these basic steps:

1. Configure and edit the anaconda-ks.cfg file as desired. We'll describe this process in more detail shortly.

2. Mount the desired local media. You may need to run a command such as **fdisk -l** as the root user to identify the appropriate device file. If the drive doesn't mount automatically, you can then mount the drive with a command such as **mount /dev/sdb1 /mnt**.
3. Copy the Kickstart file to ks.cfg on the mounted local media. (Other names are okay; ks.cfg is just the most common filename for this purpose in Red Hat documentation.)
4. Make sure the ks.cfg file has at least read permissions for all users. If SELinux is active on the local system, the contexts should normally match that of other files in the same directory. For more information, see Chapter 4.

   Be aware that a Kickstart configuration file on an FTP server may be a security risk. It's almost like the DNA of a system. If a black hat hacker gets hold of that file, he could use it to set up a copy of your systems and see how to break into and compromise your data. Because this file normally contains a root administrative password, you should change the password as soon as the system is booted for the first time.

Be careful with the Kickstart configuration file. Unless direct root logins are disabled, the file includes the root administrative password. Even if that password is encrypted, a black hat hacker with the right tools and a copy of that Kickstart configuration file can run a dictionary attack and decrypt that password if it is not secure enough.

   You should now be ready to use the Kickstart media on a different system. You'll get to try this again shortly in an exercise.
5. Now try to access the Kickstart file on the local media. Boot the RHEL 7 installation CD/DVD. When the first menu appears, highlight Install Red Hat Enterprise Linux 7.0 and press TAB. Commands to Anaconda should appear, similar to the following, and a cursor should appear at the end of that line:

   ```
   > vmlinuz initrd=initrd.img inst.stage2=hd:LABEL=RHEL-7.0↵
   \x20Server.x86_64 quiet
   ```

6. Add information for the location of the Kickstart file to the end of the line. For example, the following addition locates that file on the first partition of the second hard drive, which may be a USB drive:

   ```
   ks=hd:sdb1:/ks.cfg
   ```

   Alternatively, if the kickstart file is on the boot CD, try adding the following command:

   ```
   ks=cdrom:/ks.cfg
   ```

Alternatively, if the kickstart file is on the first floppy drive, enter the following:

```
ks=hd:fd0:/ks.cfg
```

There may be some trial and error with this method. Yes, device files are generally assigned in sequence (sda, sdb, sdc, and so on). However, unless you boot Linux with the given storage media, there is no certainty about which device file is assigned to a specific drive.

Set Up Network Access to Kickstart

The process of setting up a Kickstart file from local media can be time consuming, especially if you have to go from system to system to load that file. In many cases, it's more efficient to set up the Kickstart file on a network server. One logical location is the same network server used for the installation files. For example, based on the FTP server created in Chapter 1, Lab 2, assume there's a ks.cfg file in the FTP server's /var/ftp/pub directory. SELinux contexts should match that of that directory, which can be confirmed with the following commands:

```
# ls -Zd /var/ftp/pub
# ls -Z /var/ftp/pub
```

Once an appropriate ks.cfg file is in the /var/ftp/pub directory, you can access it by adding the following directive to the end of the **vmlinuz** line described earlier in Step 5:

```
ks=ftp://192.168.122.1/pub/ks.cfg
```

Similar options are possible for a Kickstart file on an NFS and an HTTP server, as follows:

```
ks=nfs:192.168.122.1:/ks.cfg
ks=http://192.168.122.1/ks.cfg
```

If there's an operational DNS server on the local network, you can substitute the hostname or fully qualified domain name of the target server for the IP address.

To ease the process of creating a Kickstart-based installation server, see the Cobbler project at http://cobbler.github.com. Cobbler uses profiles and small blocks of code (called "snippets") to dynamically generate Kickstart files and automate network installations.

Sample Kickstart File

We've based this section on the anaconda-ks.cfg file created when we installed RHEL 7 inside a KVM-based VM with a number of added comments. Although you're welcome to use it as a sample file, be sure to customize it for your hardware and network. This section just scratches the surface of what you can do with a Kickstart file; your version of this file may vary.

exam Watch

Unlike what's available for many other Red Hat packages, the Kickstart documentation available within an installed RHEL 7 system is somewhat sparse. In other words, you can't really rely on man pages or files in the /usr/share/doc directory for Kickstart help during an exam. If you're uncertain about specific commands to include in the Kickstart file, the Kickstart Configurator described later in this chapter can help.

Although most of the options are self-explanatory, we've interspersed our explanation of each command within the file. This file illustrates just a small portion of the available commands. For more information on each command (and options) in this file, read the latest RHEL 7 Installation Guide, which is available online at https://access.redhat.com/documentation.

Follow these ground rules and guidelines when setting up a Kickstart file:

- In general, retain the order of the directives. However, some variation is allowable depending on whether the installation is from local media or over a network.
- You do not need to use all the options.
- If you leave out a required option, the user will be prompted for the answer.
- Don't be afraid to make a change; for example, partition-related directives are commented out by default.
- Line wrapping in the file is acceptable.

on the Job

If you leave out an option, the installation process will stop at that point. This is an easy way to see if a Kickstart file is properly configured. However, because some Kickstart options change the partitions on a hard drive, even tests can be dangerous. Therefore, it's best to test a Kickstart file on a test system—or even better, an experimental VM.

The following is the code from one of our anaconda-ks.cfg files. The first line tells us that this file was created for RHEL 7:

```
#version=RHEL7
```

Next, the **auth** command sets up the Shadow Password Suite (**--enableshadow**) and the SHA 512-bit encryption algorithm for password encryption (**--passalgo=sha512**). A password encrypted to the SHA512 algorithm starts with a **$6**:

```
authconfig --enableshadow --passalgo=sha512
```

The next command is simple; it starts the installation process from the first DVD/CD drive on the system:

```
cdrom
```

The next step is to specify the source of the installation files. To use RHEL 7 DVDs, leave the existing **cdrom** entry. To install from an NFS server, specify the URI as follows. If there's a reliable DNS server for the local network, you can substitute the hostname for the IP address.

```
nfs --server=192.168.122.1 --dir=/inst
```

You can also configure a connection to an FTP or HTTP server by substituting one of the commands shown here. The directories specified are based on the FTP and HTTP installation servers created in Chapter 1:

```
url --url http://192.168.122.1/inst
```

or

```
url --url ftp://192.168.122.1/pub/inst
```

If the ISO file that represents the RHEL 7 DVD exists on a local hard drive partition, you can specify that as well. For example, the following directive points to ISO CDs or DVDs on the /dev/sda10 partition:

```
harddrive --partition=/dev/sda10 --dir=/tmp/michael/
```

The **firstboot --enable** runs the setup agent during the first installation. If you want to avoid the Firstboot process, you can also replace this line with the **firstboot --disabled** directive. As there's no way to set up a Kickstart file with answers to the Firstboot prompts, that **--disabled** directive helps automate the Kickstart process.

```
firstboot --disabled
```

The **ignoredisk** directive that follows specifies volumes only on the noted vda drive. Of course, this works only if there is a specified virtual drive on the target VM. (It's possible to specify SAS or SCSI drives on such VMs, which would conflict with these directives.)

```
# ignoredisk --only-use=vda
```

The **lang** command sets the language to use during the installation process. It matters if the installation stops due to a missing command in this file. The **keyboard** command is self-explanatory—it specifies the keyboard layout to configure on this computer.

```
keyboard --vckeymap=us --xlayouts='us'
lang en_US.UTF-8
```

The required **network** command is simplest if there's a DHCP server for the local network: **network --device eth0 --bootproto dhcp**. In contrast, the following two lines configure static IP address information, with the noted network mask (**--netmask**), gateway address (**--gateway**), DNS server (**--nameserver**), and computer name (**--hostname**).

```
network --bootproto static --device=eth0 --gateway=192.168.122.1↵
--ip=192.168.122.150 --netmask=255.255.255.0 --noipv6↵
--nameserver==192.168.122.1 --activate
network --hostname tester1.example.com
```

Please note that all static networking information for the **network** command *must* be on *one* line. Line wrapping, if the options exceed the space in a text editor, is acceptable. If you're setting up this file for a different system, don't forget to change the IP address and hostname information accordingly. Be aware, if you did not configure networking during the installation process, it won't be written to the subject anaconda-ks.cfg file. Given the complexity of the **network** directive, you could either use the Kickstart Configurator to help set up that directive or configure networking after installation is complete.

As the password for the root user is part of the RHEL 7 installation process, the Kickstart configuration file can specify that password in encrypted format. Although encryption is not required, it can at least delay a black hat hacker who might break into a system after installation is complete. Because the associated cryptographic hash function is the same as is used for the /etc/shadow file, you can copy the desired password from that file.

```
rootpw --iscrypted $6$5UrLfXTk$CsCW0nQytrUuvycuLT317/
```

The **timezone** command is associated with a long list of time zones. They're documented in the tzdata package. For a full list, run the **rpm -ql tzdata** command. By default, Red Hat sets the hardware clock to the equivalent of Greenwich Mean Time with the **--isUtc** switch. That setting supports automated changes for daylight saving time. The following setting can be found as a subdirectory and file in the /usr/share/zoneinfo directory:

```
timezone America/Los_Angeles --isUtc
```

The **user** directive can be included to create a user during the boot process. It requires a username, an encrypted password, and optionally a list of groups the user should belong to and the GECOS information for the user (typically his full name). In the following example, the encrypted password is omitted for brevity:

```
user --groups=wheel name=michael --password=... --iscrypted --gecos="MJ"
```

As for security, the **firewall** directive can optionally be added. When coupled with **--service=ssh**, it specifies the services that are allowed through the firewall:

```
firewall --service=ssh
```

The **selinux** directive is also optional and can be set to ***--enforcing***, ***--permissive***, or ***--disabled***. The default is ***--enforcing***:

```
selinux --enforcing
```

The default bootloader is GRUB 2. It should normally be installed on the space between the Master Boot Record (MBR) of a hard drive and the first partition. You can include a ***--boot-drive*** switch to specify the drive with the bootloader and an ***--append*** switch to specify parameters for the kernel:

```
bootloader --location=mbr --boot-drive=vda
```

As suggested by the comments that follow, it's first important to clear some existing sets of partitions. First, the **clearpart *--all --initlabel --drives=vda*** directive clears all partitions on the vda virtual hard drive. If it hasn't been used before, ***--initlabel*** initializes that drive:

```
clearpart --all --initlabel --drives=vda
```

Changes are required in the partition (**part**) directives that follow. They should specify the directory, filesystem format (***--fstype***), and ***--size*** in MB:

```
part /boot --fstype="xfs" --size=500
part swap --fstype="swap" --size=1000
part / --fstype="xfs" --size=10000
part /home --fstype="xfs" --size=1000
```

Be aware, your version of an anaconda-ks.cfg file may include an ***--onpart*** directive that specifies partition device files such as /dev/vda1. That would lead to an error unless the noted partitions already exist. So if you see any ***--onpart*** directives, it's simplest to delete them. Otherwise, you'd have to create those partitions before starting the installation process, and that can be tricky.

Although other partition options may be used for RAID arrays and logical volumes, the implicit focus of the Red Hat exams is to set up such volumes after installation is complete. If you want to try out other options such as logical volumes, create your own Kickstart file. It's best if you set it up from a different VM installation. Just be aware, the Kickstart file can configure physical volumes (PVs), volume groups (VGs), and logical volumes (LVs), in that order (and the order is important), similar to what's shown here:

```
part pv.01 --fstype="lvmpv" --ondisk=vda --size=11008
part /boot --fstype="xfs" --ondisk=vda --size=500
part swap --fstype="swap" --ondisk=vda --size=1000
volgroup rhel --pesize=4096 pv.01
logvol / --fstype="xfs" --size=10000 --name=root --vgname=rhel
logvol /home --fstype="xfs" --size=1000 -name=home --vgname=rhel
```

For more information on how LVs are configured, see Chapter 8.

The default version of the Kickstart file may contain a **repo** directive. It would point to the FTP network installation source from Chapter 1, Lab 2, and should be deleted from or commented out of the Kickstart file as follows:

```
#repo --name="Red Hat Enterprise Linux" ↵
--baseurl=ftp://192.168.122.1/pub/inst --cost=100
```

To make sure the system actually completes the installation process, this is the place to include a directive such as **reboot**, **shutdown**, **halt**, or **poweroff**. If you're reusing an existing KVM-based VM, it may be necessary to shut off the system to change the boot media from the CD/DVD to the hard drive. Therefore, you may prefer to use the following directive:

```
shutdown
```

What follows is a list of package groups that are installed through this Kickstart configuration file. These names correspond to the names you can find in the *-comps-Server.x86_64.xml file in the RHEL 7 DVD /repodata directory described in Chapter 1. Because the list is long, the following is just an excerpt of package groups (which start with **@**) and package names:

```
%packages
@base
@core
...
@print-client
@x11
%end
```

After the package groups are installed, you can specify post-installation commands after the following directive. For example, you could set up custom configuration files. However, the **%post** directive and anything that follows is not required.

```
%post
```

Finally, use the **ksvalidator** utility to verify the syntax of the Kickstart file. An example is shown here:

```
# ksvalidator ks.cfg
The following problem occurred on line 32 of the kickstart file:

Unknown command: vogroup
```

EXERCISE 2-3

Create and Use a Sample Kickstart File

In this exercise, you will use the anaconda-ks.cfg file to duplicate the installation from one computer to another with identical hardware. This exercise installs all the exact same packages with the same partition configuration on the second computer. Additionally, this exercise even configures the SELinux context for that Kickstart file.

Because the objective is to install the same packages as the current installation, no changes are required to packages or package groups from the default anaconda-ks.cfg file in the /root directory. This assumes access to a network installation source such as that created in Lab 2 of Chapter 1.

The steps in this exercise assume sufficient space and resources for at least two different KVM-based VMs, as discussed in Chapter 1:

1. Review the /root/anaconda-ks.cfg file on server1.example.com. Copy it to ks.cfg.
2. If there's an existing **network** directive in the file, modify it to point to an IP address of 192.168.122.150, with a hostname of tester1.example.com. If a system with that hostname and IP address already exists, use a different hostname and IP address on the same network. It is okay if such a directive doesn't already exist; networking can be configured after installation is complete, using the techniques discussed in Chapter 3.
3. Make sure the directives associated with drives and partitions in the ks.cfg file are active and they are not commented out. Pay attention to the **clearpart** directive; it should normally be set to **--all** to erase all partitions and **--initlabel** to initialize newly created disks. If there's more than one hard drive attached to the VM, the **--drives=vda** switch can focus on the first virtual drive on a KVM-based VM.
4. Remove the **cdrom** directive if present. Review the location of the installation server, associated with the **url** or **nfs** directive. This lab assumes it's an FTP server accessible on IP address 192.168.122.1, in the pub/inst/ subdirectory. If it's a different IP address and directory, substitute accordingly.

   ```
   url --url ftp://192.168.122.1/pub/inst
   ```

5. Make sure the following directive is included just before the **%packages** directive at the end of the file:

   ```
   shutdown
   ```

6. Use the **ksvalidator** utility to check the syntax of the Kickstart file. If no errors are reported, proceed to the next step:

   ```
   ksvalidator ks.cfg
   ```

7. Copy the ks.cfg file to the base directory of the installation server; if it's the vsFTP server, that directory is /var/ftp/pub. Make sure that file is readable for all users (by default it is accessible only by root with permissions 600). For example, you could use the following command:

   ```
   # chmod +r /var/ftp/pub/ks.cfg
   ```

8. Assuming that the base directory is /var/ftp/pub, modify the SELinux context of that file with the following command:

   ```
   # restorecon /var/ftp/pub/ks.cfg
   ```

9. Make sure any existing firewalls do not block the communication port associated with the installation server. For detailed information, see Chapter 4. The simplest way to do so is to open the ftp service with the **firewall-cmd** command:

   ```
   # firewall-cmd --permanent --add-service=ftp
   # firewall-cmd --reload
   ```

10. Create a KVM-based virtual machine on the local host so that it has sufficient hard drive space. Boot that VM using the RHEL 7 DVD.
11. At the Red Hat Installation menu, highlight the first option and press TAB. It will display the startup directives toward the bottom of the screen. At the end of that line, add the following directive:

    ```
    ks=ftp://192.168.122.1/pub/ks.cfg
    ```

If the Kickstart file is on a different server or on local media, substitute accordingly.

You should now see the system installation creating the same basic setup as the first system. If the installation process stops before rebooting, then there's some problem with the Kickstart file, most likely a case of insufficient information.

The Kickstart Configurator

Even users who prefer to work at the command line can learn from the Red Hat GUI tool known as the Kickstart Configurator. It includes most (but not all) of the basic options associated with setting up a Kickstart configuration file. You can install it with the following command:

```
# yum install system-config-kickstart
```

As a GUI tool associated with the installation process, this command typically includes a number of dependencies.

on the job

Those of you sensitive to properly written English may object to the term "Kickstart Configurator," but it is the name given by Red Hat to the noted GUI configuration tool.

Now that you understand the basics of what goes into a Kickstart file, it's time to solidify your understanding through the graphical Kickstart Configurator. It can help you learn more about how to configure the Kickstart file. Once the right packages are installed, it can be opened from a GUI command line with the **system-config-kickstart** command. To start it with the default configuration for the local system, cite the anaconda-ks.cfg file as follows:

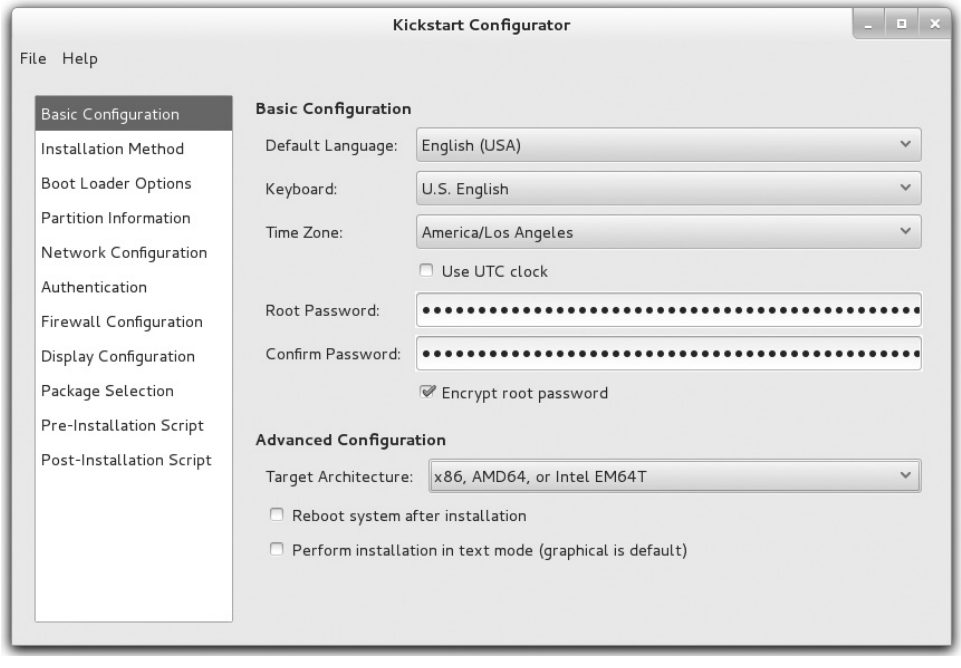
```
# system-config-kickstart /root/anaconda-ks.cfg
```

This should open the Kickstart Configurator shown in Figure 2-16. (Of course, it's probably a good idea to back up the anaconda-ks.cfg file first.)

on the job

Before starting the Kickstart Configurator, it's best to make sure there's an active connection to a remote RHEL 7 repository through the RHN.

FIGURE 2-16 The Kickstart Configurator



The screen shown in Figure 2-16 illustrates a number of basic installation steps. If you've already installed RHEL, all of these steps should look familiar.

A number of other options appear in the left pane, each associated with different Kickstart commands. To learn more about Kickstart, experiment with some of these settings. Use the File | Save command to save these settings with the filename of your choice, which you can then review in a text editor. Alternatively, you can choose File | Preview to see the effect of different settings on the Kickstart file.

The following sections provide a brief overview of each option shown in the left pane. A detailed understanding of the Kickstart Configurator can also help you understand the installation process.

Basic Configuration

In the Basic Configuration screen, you can assign settings for the following components:

- **Default Language** Specifies the default language for the installation and operating system.
- **Keyboard** Sets the default keyboard; normally associated with language.
- **Time Zone** Customizes the local time zone and specifies whether the hardware clock is set to UTC, which is essentially the same as Greenwich Mean Time.
- **Root Password** Specifies the password for the root administrative user; may be encrypted.
- **Target Architecture** Can help customize a Kickstart file for different systems.
- **Reboot System After Installation** Adds the **reboot** command to the end of the kickstart file.
- **Perform System Installation in Text Mode** Supports automated installation in text mode. Once automated, the installation mode should not matter.

Installation Method

The Installation Method options are straightforward. You're either installing Linux for the first time or upgrading a previous installation. The installation method, and your entries, are based on the location of the installation files. For example, if you select an NFS installation method, the Kickstart Configurator prompts you for the name or IP address of the NFS server and the shared directory with the RHEL installation files.

You can set up the Kickstart file to install RHEL from a CD/DVD, a local hard drive partition, or one of the standard network servers: NFS, HTTP, or FTP.

Boot Loader Options

The next section lists boot loader options. The default boot loader is GRUB, which supports encrypted passwords for an additional level of security during the boot process.

Linux boot loaders are normally installed on the MBR. If you're dual-booting Linux and Microsoft Windows with GRUB, you *can* set up the Windows boot loader (or an alternative third-party boot loader) to point to GRUB on the first sector of the Linux partition with the /boot directory.

Partition Information

The Partition Information section determines how this installation configures the hard disks on the affected computers. Although it supports the configuration of standard and RAID partitions, it does not yet support the configuration of LVM groups. The Clear Master Boot Record option allows you to wipe the MBR from an older hard disk that might have a problem there; it includes the **zerombr** command in the Kickstart file.

Don't use the zerombr **option if you want to keep an alternative bootloader on the MBR such as the Microsoft Windows Bootmgr.**

You can remove partitions depending on whether they've been created on a Linux filesystem. If you're using a new hard drive, it's important to initialize the disk label as well. Click the Add button; it opens the Partition Options dialog box.

Network Configuration

The Network Configuration section enables you to set up IP addressing on the network cards on a target computer. You can customize static IP addressing for a specific computer, or configure the use of a DHCP server. Just click Add Network Device and explore the Network Device Information window.

Authentication

The Authentication section lets you set up two forms of security for user passwords: Shadow Passwords, which encrypts user passwords in the /etc/shadow file, and the encryption hash for those passwords. If you have a fingerprint scanner installed, you can select the corresponding check box to enable a fingerprint reader. This enables two-factor authentication by requesting users to provide their credentials at logon and scan on the fingerprint reader.

This section also allows you to set up authentication information for various protocols:

- **NIS** Network Information Service is used to connect to a login authentication database on a network with Unix and Linux computers.
- **LDAP** In this context, the Lightweight Directory Access Protocol is a directory service that can be used as an alternative login authentication database.

- **Kerberos 5** The MIT system for strong cryptography is used to authenticate users on a network.
- **Hesiod** Hesiod is a network database that can be used to store user account and password information.
- **SMB** Samba connects to a Microsoft Windows–style network for login authentication.
- **Name Switch Cache** Associated with NIS for looking up user accounts and groups.

Firewall Configuration

The Firewall Configuration section allows you to configure a default firewall for the subject computer. On most systems, you'll want to keep the number of trusted services to a minimum. However, in a situation such as the Red Hat exams, you may be asked to set up a multitude of services on a single system, which would require the configuration of a multitude of trusted services on a firewall.

In this section, you can also configure basic SELinux settings. The Active and Disabled options are straightforward; the Warn option corresponds to a permissive implementation of SELinux. For more information, see Chapter 4.

Display Configuration

The Display Configuration section supports the installation of a basic Linux GUI. The actual installation depends on those packages and package groups selected in the next section. Although there is a lot of debate on the superiority of GUI- versus text-based administrative tools, text-based tools are more stable. For this reason (and more), many Linux administrators don't even install a GUI. However, if you're installing Linux on a series of workstations, as might be done with a series of Kickstart files, it's likely that most of the users won't be administrators.

In addition, you can disable or enable the Setup Agent, also known as the Firstboot process. For a completely automated installation, the Setup Agent should be disabled.

Package Selection

The Package Selection section allows you to choose the package groups that are installed through this Kickstart file. As noted earlier, the associated screens are blank if there's no current connection to a remote repository such as updates from the RHN. At the time of writing, you would get the same problem if you used a local installation source. In that case, you need to manually edit the file generated by Kickstart Configurator and add the required package selection.

Installation Scripts

You can add pre-installation and post-installation scripts to the Kickstart file. Post-installation scripts are more common, and they can help configure other parts of a Linux operating system in a common way. For example, if you wanted to install a directory with employee benefits information, you could add a post-installation script that adds the appropriate **cp** commands to copy files from a network server.

CERTIFICATION OBJECTIVE 2.04

Administration with the Secure Shell and Secure Copy

Red Hat Enterprise Linux installs the Secure Shell (SSH) packages by default. The RHCSA requirement with respect to SSH is simple; you need to know how to use it to access remote systems. In addition, you also need to know how to securely transfer files between systems. Therefore, in this section, you'll examine how to use the **ssh** and **scp** commands to access remote systems and transfer files.

As suggested earlier, the stage is already set by the default installation of SSH on standard installations of RHEL 7. Although firewalls are enabled by default, the standard RHEL 7 firewall leaves TCP port 22 open for SSH access. Related configuration files are stored in the /etc/ssh directory. SSH server configuration is part of the RHCE requirements. Related client commands such as **ssh**, **scp**, and **sftp** are covered in this section.

The Secure Shell daemon is secure because it encrypts messages. In other words, users who are listening on a network can't read the data sent between SSH clients and servers. And that's important on a public network such as the Internet. RHEL incorporates SSH version 2, which supports multiple key-exchange methods and is incompatible with the older SSH version 1. Key-based authentication for SSH is covered in Chapter 4. If you're studying for the RHCE objectives on SSH, read Chapter 11.

Configure an SSH Client

The main SSH client configuration file is /etc/ssh/ssh_config. Individual users can have custom SSH client configurations in their ~/.ssh/config files. Four directives are included by default. First, the **Host** * directive applies the other directives to all connections:

```
Host *
```

This is followed by a directive that supports authentication using the Generic Security Services Application Programming Interface (GSSAPI) for client/server authentication. This provides support for Kerberos authentication:

```
GSSAPIAuthentication yes
```

This next directive supports remote access to GUI applications. X11 is a legacy reference to the X Window System server used on Linux.

```
ForwardX11Trusted yes
```

The next directives allow the client to set several environmental variables. The details are normally trivial between two Red Hat Enterprise Linux systems.

```
SendEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
SendEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
SendEnv LC_IDENTIFICATION LC_ALL LC_LANGUAGE
SendEnv XMODIFIERS
```

This sets the stage for command-line access of remote systems.

Command-Line Access

This section is based on standard access with the **ssh** command. To access a remote system, you need the username and password on that remote system. By default, direct ssh-based access to the root account is enabled. For example, the following command opens a shell using that account on the noted server1 system:

```
$ ssh root@server1.example.com
```

If you get an error such as "Name or service not known" when you attempt to access a remote host via SSH, that indicates that the system cannot resolve the hostname to an IP address. We will configure name resolution in Chapter 3. In the meantime, to log in to server1.example.com via SSH, use its IP address 192.168.122.50.

The following command works in the same way:

```
$ ssh -l root server1.example.com
```

Without the username, the **ssh** command assumes that you're logging in remotely as the username on the local system. For example, if you were to run the command

```
$ ssh server1.example.com
```

from the user michael account, the **ssh** command assumes that you're trying to log in to the server1.example.com system as user michael. The first time the command is run between systems, it presents something similar to the following message:

```
$ ssh server1.example.com
The authenticity of host 'server1.example.com (192.168.122.50)'
can't be established.
ECDSA key fingerprint is b6:80:5d:8c:1d:ab:18:ab:46:15:c5:c8:e3:ea:9f:1c.
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'server1.example.com,192.168.122.50'
(ECDSA) to the list of known hosts.
michael@server1.example.com's password:
```

Once connected via **ssh**, you can do anything on the remote system that's supported by your user privileges on that machine. For example, you can even shut down the remote system gracefully with the **poweroff** command. After executing that command, you'll typically have a couple of seconds to exit out of the remote system with the **exit** command.

More SSH Command-Line Tools

If you prefer to access the remote system with an FTP-like client, the **sftp** command is for you. Although the **-l** switch doesn't have the same meaning of the **ssh** command, it still can be used to log in to the account of any user on the remote system. Whereas regular FTP communication proceeds in clear text, communication with the **sftp** command can be used to transfer files in encrypted format.

Alternatively, if you just want to transfer files over an encrypted connection, the **scp** command can help. For example, we created some of the screenshots for this book on the test VMs configured in Chapters 1 and 2. To transmit one of those screenshots to one of our systems, we used a command similar to the following, which copied the F02-20.tif file from the local directory to the remote system with the noted hostname, in the /home/michael /RHbook/Chapter2 directory:

```
# scp F02-20.tif michael@server1:/home/michael/RHbook/Chapter2/
```

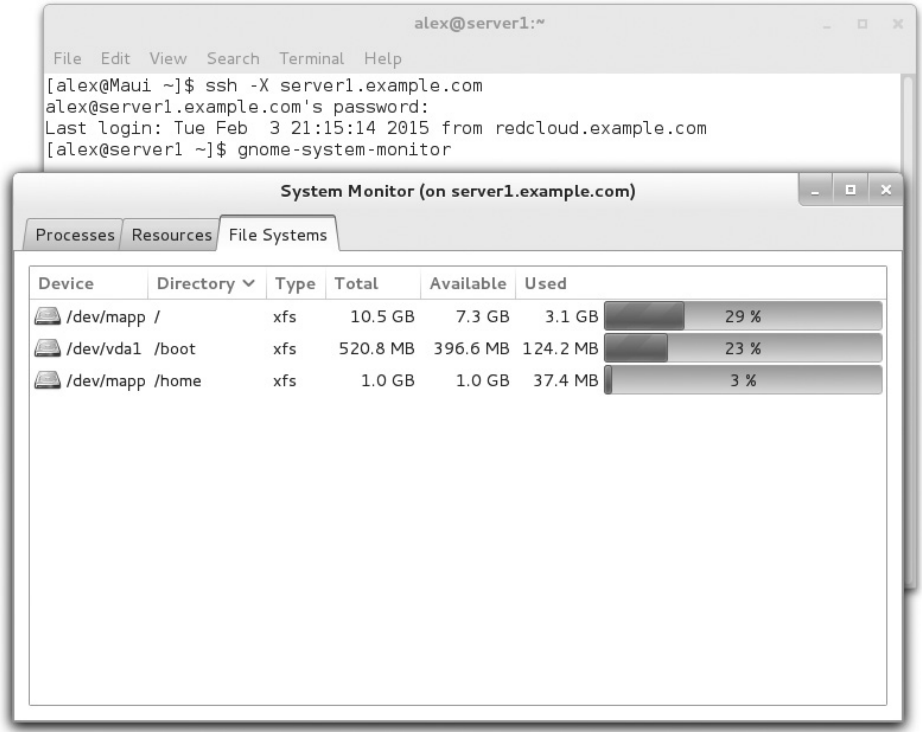
Unless key-based authentication has been configured (as discussed in Chapter 4), the command prompts for the password of the user michael on the system named server1. Once the password is confirmed, the **scp** command transfers the F02-20.tif file in encrypted format to the noted directory on the remote system named server1.

Graphical Secure Shell Access

The **ssh** command can be used to forward the output of a GUI application over a network. As strange as it sounds, it works if the local system runs an X server while you call remote GUI client applications from remote systems.

FIGURE 2-17

Remote GUI access via SSH



By default, both the SSH server and client configuration files are set up to support X11 communication over a network. All you need to do is connect to the remote system with the **-X** switch (or **-Y**, to use trusted X11 forwarding, which bypasses some security extension controls). For example, you could use the command sequence shown in Figure 2-17 to monitor the remote system.

CERTIFICATION OBJECTIVE 2.05

Consider Adding These Command-Line Tools

You may want to consider adding several command-line tools to help administer various Linux systems. These tools will be used later in this book to make sure various servers are actually operational. Although it's best to test services such as Postfix with actual e-mail clients (for instance, Evolution and Thunderbird), command tools like **telnet**, **nmap**, and **mutt** can be used to check these services remotely from a command-line interface. For exam

purposes, you can use these tools to test, diagnose, and solve system issues in the time that it would take to download a complex tool such as Evolution. Although the **ssh** command can help access GUI tools remotely, communication with such tools can be time consuming.

For administrative purposes, tools of interest include the following:

- Use **telnet** and **nmap** to verify remote access to open ports.
- Use **mutt** as an e-mail client to verify the functionality of an e-mail server.
- Use **elinks** as a web browser to make sure web services are accessible.
- Use **lftp** to access FTP servers with command completion.

Checking Ports with telnet

The **telnet** command is a surprisingly powerful tool. Anyone who is aware of the security implications of clear-text clients may hesitate to use **telnet**. People who use **telnet** to log in to remote servers do transmit their usernames, passwords, and other commands in clear text. Anyone with a protocol analyzer such as Wireshark can read that data fairly easily.

However, **telnet** can do more. When run locally, it can verify the operation of a service. For example, the following command verifies the operation of vsFTP on the local system:

```
$ telnet localhost 21
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 (vsFTPd 3.0.2)
```

The "Escape character" is CTRL-] (the CTRL key and the right square bracket pressed simultaneously). Pressing this key combination from the noted screen brings up the **telnet>** prompt. From there, you can exit with the **quit** command.

```
^]
telnet> quit
```

In most cases, you don't even need to execute the Escape character to quit; just type in the **quit** command.

If vsFTP is not running or had been configured to communicate on a port other than 21, you'd get the following response:

```
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
```

If there's no firewall, you'd get the same result from a remote system. If a firewall is blocking communications over port 21, however, you may get a message similar to the following:

```
telnet: connect to address 192.168.122.50: No route to host
```

Some services such as the Postfix e-mail server are by default configured to accept connections only from the local system. In that case, with or without a firewall, you'd get the "connection refused" message when trying to connect from a remote system.

Checking Ports with nmap

The **nmap** command is a powerful port-scanning tool. As such, the website of the nmap developers states that "when used improperly, nmap can (in rare cases) get you sued, fired, expelled, jailed, or banned by your ISP." Nevertheless, it is included in the standard RHEL 7 repositories. As such, it is supported by Red Hat for legal use. It's a quick way to get a view of the services that are open locally and remotely. For example, the **nmap localhost** command shown in Figure 2-18 detects and reveals those services that are running on the local system.

But in contrast, when the port scanner is run from a remote system, it looks like only one port is open. That shows the effect of the firewall on the server.

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-02-02 09:52 PST
Nmap scan report for server1.example.com (192.168.122.50)
Host is up (0.027s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
```

Configure an E-mail Client

The configuration process for a GUI e-mail client should be trivial for any candidate for Red Hat certification. However, the same may not necessarily be true for command-line clients, and they're useful for testing the functionality of standard e-mail server services such as Postfix and Sendmail. For example, once a server is configured for Post Office Protocol

FIGURE 2-18

Applying a port scanner locally

```
[root@server1 ~]# nmap localhost

Starting Nmap 6.40 ( http://nmap.org ) at 2015-02-03 21:36 GMT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
111/tcp open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 2.40 seconds
[root@server1 ~]#
```

(POP) e-mail—even e-mail that is delivered using the near-ubiquitous version 3 (POP3)—it can be checked with the following command:

```
# mutt -f pop://username@host
```

Since GUI e-mail clients should be trivial for readers, the remainder of this section is focused on the use of command-line e-mail clients.

Command-Line Mail

One way to test a local mail system is with the built-in command line **mail** utility. It provides a simple text-based interface. The system keeps each user's mail in /var/mail directory files associated with each username. Users who read messages with the **mail** utility can also reply, forward, or delete associated messages.

You can certainly use any of the other mail readers, such as **mutt**, or the e-mail managers associated with different GUI web browsers to test your system. Other mail readers store messages in different directories. Mail readers such as **mutt** and **mail** can be used to send messages if a Simple Mail Transfer Protocol (SMTP) server is active for the local system.

There are two basic methods for using **mail**. First, you can enter the subject and then the text of the message. When done, press CTRL-D. The message is sent, and the **mail** utility returns to the command line. Here's an example:

```
$ mail michael
Subject: Test Message
Text of the message
EOT
$
```

Alternatively, you can redirect a file as the text of an e-mail to another user. For example, the following command sends a copy of /etc/hosts to the root user on server1, with the Subject name of "hosts file":

```
$ mail -s 'hosts file' < /etc/hosts root@server1.example.com
```

Reading Mail Messages

By default, the **mail** system doesn't open for a user unless there is actual e-mail in the appropriate file. Once the mail system is open, the user will see a list of new and already read messages. If you've opened the **mail** system for an account, you can enter the number of the message and press ENTER. If you press ENTER with no argument, the mail utility assumes you want to read the next unread message. To delete a mail message, use the **d** command after reading the message, or use **d#** to delete the message numbered #.

Alternatively, mail messages can be read from the user-specified file in the local /var/mail directory. Files in this directory are named for the associated username.

The Use of Text and Graphical Browsers

Linux includes a variety of graphical browsers. Access of regular and secure websites is available through their associated protocols, the Hypertext Transfer Protocol (HTTP), and its secure cousin, Hypertext Transfer Protocol, Secure (HTTPS). The use of graphical browsers should be trivial for any serious user of Linux.

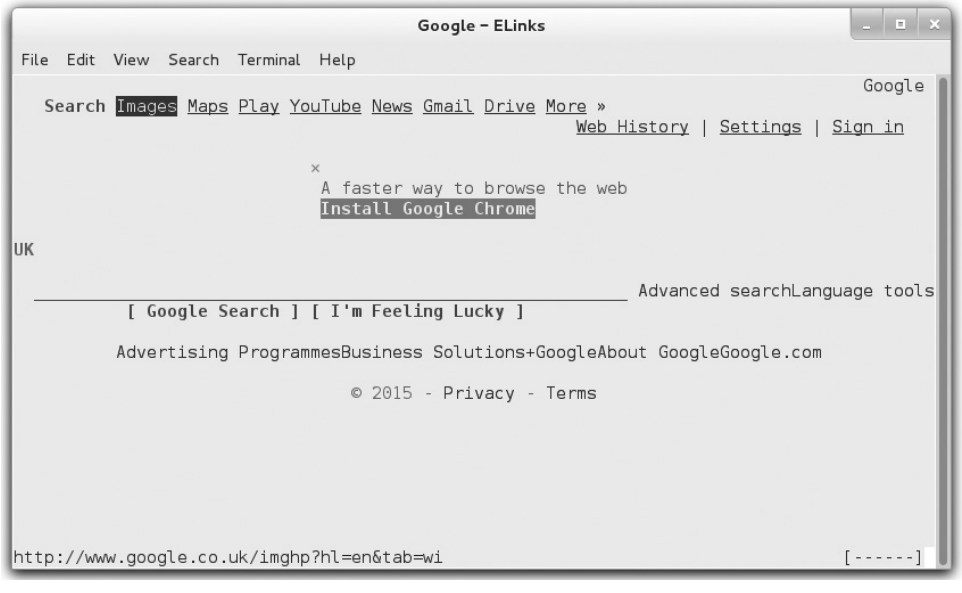
You may not always have access to the GUI, especially when working from a remote system. In any case, text-based browsers work more quickly. The standard text-based browser for Red Hat is ELinks. Once the package is installed, you can use it from the command line to open the website of your choice. For example, Figure 2-19 illustrates the result of the **elinks http://www.google.com** command.

To exit from ELinks, press the ESC key to access the menu bar, and then press F | X and accept the prompt to exit from the browser. As an alternative, the Q key can be used as a shortcut.

If you configure a web server, the easiest way to make sure it works is with a simple-text home page. No HTML coding is required. For example, we could add the following text to home.html:

```
This is my home page
```

FIGURE 2-19 The ELinks browser



We could then run the **elinks home.html** command to view this text in the ELinks browser. If you've set up an Apache file server on the /var/www/html/inst directory as discussed in Chapter 1, you can also use **elinks** to review the files copied to that server with the following command:

```
$ elinks http://192.168.122.1/inst
```

Using lftp to Access URLs

The original FTP client software was a basic command-line, text-oriented client application that offered a simple but efficient interface. Most web browsers offer a graphical interface and can also be used as an FTP client.

Any FTP client allows you to view the directory tree and files. Using **ftp** as a client is easy. You could use the **ftp** command to connect to a server such as ftp.redhat.com with the following command:

```
# ftp ftp.redhat.com
```

However, that client asks for a username and password. You could enter username **anonymous** and your e-mail address as a password to access the Red Hat FTP server. But if you accidentally enter a real username and password, that data is sent in clear text, available to anyone who happens to be using the right network analyzer applications on the network. Strangely enough, the **ftp** command client is not installed on standard RHEL 7 installations.

That's one reason why **lftp** is better. It automatically attempts an anonymous login, without prompting for a username or password. It also supports command completion, which can especially help you access files and directories with longer names.

Sure, there are risks with most FTP clients as they transmit data in clear text, but as long as usage of this command is limited to public servers with anonymous access, the risk is minimal. After all, if you use **lftp** to download Linux packages from public servers, it's not like you're putting any private information at risk. To be sure, there are other security risks with such clients, but Red Hat developers are constantly working to keep that client up to date.

If the risks are acceptable, the **lftp** command can be used to log in to an FTP server where usernames and passwords are enabled. User michael could log in to such a server with the following command:

```
$ lftp ftp.example.org -u michael
```

The **lftp** client can handle a number of different commands, as shown in Figure 2-20. Some of these commands are described in Table 2-5.

Almost all commands from the FTP prompt are run at the remote host, similar to a Telnet session. You can run regular shell commands from that prompt; just start the command with an exclamation point (!).

This is only a subset of the commands available through **lftp**. If you don't remember something, the command **help *cmd*** yields a brief description of the specified command.

FIGURE 2-20

Commands in lftp

```
[root@Maui ~]# lftp ftp.redhat.com
lftp ftp.redhat.com:~> help
    !<shell-command>                    (commands)
    alias [<name> [<value>]]            attach [PID]
    bookmark [SUBCMD]                   cache [SUBCMD]
    cat [-b] <files>                    cd <rdir>
    chmod [OPTS] mode file...           close [-a]
    [re]cls [opts] [path/][pattern]     debug [<level>|off] [-o <file>]
    du [options] <dirs>                 exit [<code>|bg]
    get [OPTS] <rfile> [-o <lfile>]     glob [OPTS] <cmd> <args>
    help [<cmd>]                        history -w file|-r file|-c|-l [cnt]
    jobs [-v] [<job_no...>]             kill all|<job_no>
    lcd <ldir>                          lftp [OPTS] <site>
    ln [-s] <file1> <file2>             ls [<args>]
    mget [OPTS] <files>                 mirror [OPTS] [remote [local]]
    mkdir [-p] <dirs>                   module name [args]
    more <files>                        mput [OPTS] <files>
    mrm <files>                         mv <file1> <file2>
    [re]nlist [<args>]                  open [OPTS] <site>
    pget [OPTS] <rfile> [-o <lfile>]    put [OPTS] <lfile> [-o <rfile>]
    pwd [-p]                            queue [OPTS] [<cmd>]
    quote <cmd>                         repeat [OPTS] [delay] [command]
    rm [-r] [-f] <files>                rmdir [-f] <dirs>
    scache [<session_no>]               set [OPT] [<var> [<val>]]
    site <site-cmd>                     source <file>
    torrent [-0 <dir>] <file|URL>...    user <user|URL> [<pass>]
    wait [<jobno>]                      zcat <files>
    zmore <files>
lftp ftp.redhat.com:~>
```

TABLE 2-5

Standard lftp Client Commands

Command	Description
cd	Changes the current working directory at the remote host
ls	Lists files at the remote host
get	Retrieves one file from the remote host
mget	Retrieves many files from the remote host with wildcards or full filenames
put	Uploads one file from your computer to the remote host
mput	Uploads a group of files to the remote host
pwd	Lists the current working directory on the remote host
quit	Ends the FTP session
!ls	Lists files on your host computer in the current directory
lcd	Changes the local host directory for upload/download
!pwd	Lists the current working directory on the local host computer

CERTIFICATION SUMMARY

Given the importance of virtualization in today's computing environment, it's no surprise that Red Hat has made KVM part of the requirements associated with the RHCSA. Assuming a valid connection to appropriate repositories, installation of KVM-related packages is fairly easy. You may need to use a command such as **modprobe kvm** to make sure appropriate modules are loaded. The Virtual Machine Manager can then be used to set up VMs using KVM on an RHEL 7 system. You can also use commands such as **virt-install**, **virt-clone**, and **virsh** to install, clone, and manage those VMs.

You can automate your entire installation with Kickstart. Every RHEL system has a Kickstart template file in the /root directory, which you can modify and use to install RHEL on other systems automatically. Alternatively, you can use the GUI Kickstart Configurator to create an appropriate Kickstart file.

With all of these systems, remote access is a must. The SSH command can help set up remote encrypted communications between Linux systems. The RHCSA requires that you know how to use an SSH client and to securely transfer files between systems. The **ssh** command can be used to log in to remote systems; the **ssh -X** command can even be used to access remote GUI applications. The **scp** command can copy files remotely over an encrypted connection.

When you are reviewing and troubleshooting RHEL services, it can be helpful to have some command-line tools at your disposal. The **telnet** command can connect to remote services on selected ports. The **nmap** command can be used as a port scanner. The **mutt** command can check the functionality of an e-mail server. The **elinks** command can be used as a command-line browser. Finally, the **lftp** command is an excellent FTP client that supports command completion.

TWO-MINUTE DRILL

The following are some of the key points from the certification objectives in Chapter 2.

Configure KVM for Red Hat

- ❑ Packages required for KVM are part of the Virtualization package groups.
- ❑ KVM-based VMs can be configured with the Virtual Machine Manager.
- ❑ The kernel modules required for KVM include **kvm** and **kvm_intel** or **kvm_amd**.

Configure a Virtual Machine on KVM

- The default storage directory for KVM-based VMs is /var/lib/libvirt/images.
- VM configuration files are stored in various /etc/libvirt subdirectories.
- VM consoles are accessible with the Virtual Machine Manager, which you can start in the GUI with the **virt-manager** command.
- VMs can be installed, cloned, and configured with the **virt-install**, **virt-clone**, and **virsh** commands.
- The **virsh list --all** command lists all configured VMs.
- The **virsh autostart *vmname*** command configures the VM domain named *vmname* to start automatically when the host system is booted.
- The **virsh start *vmname*** command starts the boot process for the VM domain named *vmname.*
- The **virsh destroy *vmname*** command in effect cuts power to the VM domain named *vmname.*

Automated Installation Options

- Installation of a system is documented in the /root/anaconda-ks.cfg Kickstart text file.
- The Kickstart file can be modified directly or with the Kickstart Configurator tool.
- Kickstart files can be called from local media or network servers.

Administration with the Secure Shell and Secure Copy

- SSH is installed by default on RHEL 7. It's even accessible through default firewalls.
- The **ssh** command can be used to access remote systems securely. It can even enable access to remote GUI utilities.
- Related commands include **sftp** and **scp**.

Consider Adding These Command-Line Tools

- Administrators may sometimes only have the command line to verify access to servers.
- The **telnet** and **nmap** commands can be used to verify remote access to open ports.
- The **mutt** e-mail client can be used to verify the functionality of an e-mail server.
- The **elinks** console web browser can verify the working of a web server.
- The **lftp** client can be used to verify access to FTP servers with the benefit of command completion.

SELF TEST

The following questions will help measure your understanding of the material presented in this chapter. As no multiple choice questions appear on the Red Hat exams, no multiple choice questions appear in this book. These questions exclusively test your understanding of the chapter. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of these questions.

Configure KVM for Red Hat

1. Name one kernel module associated with KVM.

   _____

2. What is the name of the tool that can configure KVM-based VMs in the GUI?

   _____

Configure a Virtual Machine on KVM

3. What command starts the Virtual Machine Manager in the GUI?

   _____

4. In what directory are default virtual disks stored by the Virtual Machine Manager?

   _____

5. What command can be used to create a new VM?

   _____

Automated Installation Options

6. What command starts the GUI-based Kickstart configuration tool?

   _____

7. What's the name of the file in the /root directory that documents how RHEL was installed?

   _____

8. What directive in the Kickstart configuration file is related to networking?

   _____

9. If the installation FTP server is located at ftp://server1.example.com/pub/inst, what directive in the Kickstart configuration file points to that server?

_____

10. What directive in the Kickstart configuration file would shut down a system after installation is complete?

_____

Administration with the Secure Shell and Secure Copy

11. What **ssh** command switch enables access to remote GUI utilities?

_____

12. What command initiates a secure copy of the file /etc/hosts from the system server1.example.com to the /tmp directory on the local host?

_____

Consider Adding These Command-Line Tools

13. What command would you use to see if a server is running on port 25 on a system with IP address 192.168.122.1?

_____

14. What command can be used to verify active and available (from your client) services on a remote system with IP address 192.168.122.1?

_____

LAB QUESTIONS

Several of these labs involve installation exercises. You should do these exercises on test machines only. The instructions in some of these labs delete all of the data on a system.

Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the DVD that accompanies the book in the Chapter2/ subdirectory. In case you haven't yet set up RHEL 7 on a system, the first lab is presented here in the book.

Lab 1

In this lab, you will install RHEL to create a basic server on a KVM-based VM. You will need sufficient room for one hard disk of at least 16GB (with sufficient space for 11GB of data plus a swap partition,

assuming at least 512MB of spare RAM for the VM). You'll also need room for an additional two virtual hard drives of 1GB each (18GB total).

The steps in this lab assume an installation on a KVM-based VM. To start the process, open a GUI and run the **virt-manager** command. If it doesn't happen automatically, right-click the Localhost (QEMU) option and click Connect in the pop-up menu that appears. Enter the root administrative password if prompted to do so. Once connected, you can then right-click the same option and then click New. This starts the wizard that helps configure a VM.

If you're configuring the actual VMs to be used in future chapters, this will be the server1.example .com system discussed in Chapter 1.

Ideally, there will be sufficient space on your main machine for at least four different virtual systems of the given size. That includes the three systems specified in Chapter 1, plus one spare. In other words, a logical volume or partition with 75GB of free space would be (barely) sufficient.

The steps described in this lab are general. By this time, you should have some experience with the installation of RHEL 7. In any case, the exact steps vary with the type of installation and the boot media:

1. Start with the RHEL 7 network boot CD or the installation DVD.
2. Based on the steps discussed in Chapter 1, start the installation process for RHEL 7.
3. From the Installation Summary screen, select Installation Source and point the system to the FTP-based installation server created in Chapter 1. If you followed the directions in that chapter, the server will be on ftp://192.168.122.1/pub/inst.
4. From the Installation Summary screen, click Installation Destination and select custom partitioning.
5. Create the first partition of about 500MB of disk space, formatted to the xfs filesystem, and assign it to the /boot directory.
6. Create the next partition with 1GB of disk space (or more, if space is available), reserved for swap space.
7. Create a third partition with about 10GB disk space, formatted to the xfs filesystem, and assign it to the top-level root directory, /.
8. Create another partition with about 1GB of disk space and assign it to the /home directory.
9. From the Installation Summary screen, set up the local system on a network configured on the KVM hypervisor. The default is the 192.168.122.0/24 network; for the server1.example.com system, this will be on IP address 192.168.122.50 and gateway 192.168.122.1. Configure the hostname **server1.example.com**.
10. From the Installation Summary screen, click Software Selection and then select Server with GUI. Installation of virtualization packages within a VM is not required.
11. Continue with the installation process, using your best judgment.
12. Reboot when prompted and log in as the root user. Run the **poweroff** command when you're ready to finish this lab.

A SELF TEST ANSWERS

Configure KVM for Red Hat

1. Three kernel modules are associated with KVM: kvm, kvm_intel, and kvm_amd.
2. The tool that can configure KVM-based VMs in the GUI is the Virtual Machine Manager.

Configure a Virtual Machine on KVM

3. The command that starts the Virtual Machine Manager in the GUI is **virt-manager**.
4. The directory with default virtual disks for the Virtual Machine Manager is /var/lib/libvirt/images.
5. The command that can be used to create a new VM is **virt-install**.

Automated Installation Options

6. The command that starts the GUI-based Kickstart configuration tool is **system-config-kickstart**.
7. The name of the Kickstart file in the /root directory that documents how RHEL was installed is anaconda-ks.cfg.
8. The directive in the Kickstart configuration file related to networking is **network**.
9. The directive that points to the given FTP installation server is **url --url ftp://server1.example.com/pub/inst**.
10. The directive in the Kickstart configuration file that would shut down a system after installation is complete is **shutdown**.

Administration with the Secure Shell and Secure Copy

11. The **ssh** command switch that enables access to remote GUI utilities is **-X**. The **-Y** switch is also an acceptable answer.
12. The required command to securely copy /etc/hosts from server1 to /tmp on the local host is **scp server1.example.com:/etc/hosts /tmp/**.

Consider Adding These Command-Line Tools

13. The command that you would use to see if a server is running on port 25 on a system with IP address 192.168.122.1 is **telnet 192.168.122.1 25**.
14. The command that can be used to verify active and available services on a remote system with IP address 192.168.122.1 is **nmap 192.168.122.1**.

LAB ANSWERS

Lab 1

Although there is nothing truly difficult about this lab, it should increase your confidence with VMs based on KVM. Once it's complete, you should be able to log in to the VM as the root administrative user and run the following checks on the system:

1. Check mounted filesystems, along with the space available. The following commands should confirm those filesystems that are mounted, along with the free space available on the associated volumes:

```
# mount
# df -m
```

2. Assuming you have a good connection to the Internet and a subscription to the Red Hat Portal, make sure the system is up to date. If you're using a rebuild distribution, access to their public repositories is acceptable. In either case, run the following command to make sure the local system is up to date:

```
# yum update
```

This lab confirms your ability to "install Red Hat Enterprise Linux systems as virtual guests."

Lab 2

Remember, this and all future labs in this book can be found on the DVD that comes with this book. Labs 2 through 8 can be found in the Chapter2/ subdirectory of that DVD.

One of the issues with system cloning is how it includes the hardware MAC address of any network cards. Such conflicts can lead to problems on a network. So not only would you have to change the IP address, but you may also need to ensure that a unique hardware address is assigned to the given virtual network adapter. Because of such issues, KVM normally sets up a different hardware MAC address for a cloned system. For example, if the original system had an eth0 network card with one hardware address, the cloned system would have a network card with a different hardware address.

If this seems like too much trouble, feel free to delete the cloned system. After all, there is no reference to VM cloning in the RHCSA requirements. However, it may be helpful to have a different backup system. And that's an excellent opportunity to practice the skills gained in Lab 4 with Kickstart installations.

Lab 3

The purpose of this lab is to show you the command-line method for configuring a KVM-based VM. If you haven't yet set up the four different VMs suggested in Chapter 1 (three VMs and a backup), this is an excellent opportunity to do so. One way to do this is with the **virt-install** command. Specify the following information:

- Allocated RAM (**--ram**) in megabytes, which should be at least 512.
- The path to the virtual disk file (**--disk**), which could be the same as that virtual disk created in Lab 2, and its size in gigabytes, if that file doesn't already exist.
- The URL (**--location**) for the FTP installation server created in Chapter 1, Lab 2. Alternatively, you could use the HTTP installation server also discussed in Chapter 1.
- The OS type **(--os-type=*linux*)** and variant **(--os-variant=*rhel7*)**.

You can now complete this installation normally or run a variation of that installation in Lab 5.

Lab 4

If you're not experienced with Kickstart configuration, some trial and error may be required. But it's best to run into problems now and not during a Red Hat exam or on the job. If you're able to set up a Kickstart file that can be used to install a system without intervention, you're ready to address this challenge on the RHCSA exam.

One common problem relates to virtual disks that have just been created. They must be initialized first; that's the purpose of the **--initlabel** switch to the **clearpart** directive.

Lab 5

If you've recently run a Kickstart installation for the first time, it's best to do it again. If you practice now, it means you'll be able to set up a Kickstart installation faster during an exam. And that's just the beginning. Imagine the confidence you'll have if your boss needs a couple of dozen VMs with the same software and volumes. Assuming the only differences are hostname and network settings, you'll be able to accomplish this task fairly quickly.

If you can set up a Kickstart installation from the command line with the **virt-install** command, it'll be a lot easier to set it up on a remote virtual host. You'll be able to configure new systems from remote locations, thus increasing your value in the workplace.

If you haven't yet set up the four VMs suggested in Chapter 1 (three as test systems, one as a backup), this is your opportunity to do so.

To use a Kickstart file with **virt-install**, you'll need to use regular command switches. As you're not allowed to bring this book into an exam, try to perform this lab without referring to the main body of this chapter. You'll be able to refer to the man page for the **virt-install** command for all of the important switches.

Be sure to put the **ks=** directive along with the URL of the Kickstart file within quotes. Success is the installation of a new system.

Lab 6

This lab is designed to increase your understanding of the use of the **ssh** command as a client. The encryption performed should be transparent, and will not affect any commands used through an SSH connection to administer remote systems.

Lab 7

This lab is somewhat critical with respect to several different RHCSA objectives. Once you understand the process, the actual tasks are deceptively simple. After completing this lab, you should have confidence in your abilities to do the following:

- Start and stop virtual machines.
- Configure systems to launch virtual machines at boot.

The lab also suggests one method for remotely accessing a VM.

Lab 8

This lab is designed to increase your familiarity with two important network troubleshooting tools, **telnet** and **nmap**. Network administrators with some Linux experience may prefer other tools. If you're familiar with other tools such as **nc**, great. It's the results that matter.