2

Virtual Machines and Automated Installations

CERTIFICATION OBJECTIVES

Even though installation is specified as a requirement in the RHCSA objectives, Red Hat has also stated that their exams are now given on pre-installed systems. In other words, you won't have to install RHEL 6 on a bare-metal system during the exams. However, the management of virtual machines (VMs) and Kickstart installations are also required RHCSA skills. In other words, you need to be prepared to install RHEL 6 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. Of course, this assumes you're working on a system with a physical 64-bit CPU.

Kickstart is the Red Hat system for automated installations. It works from a text file that provides answers to the RHEL 6 installation program. With those answers, the RHEL 6 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 practice 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, like 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 6 system with the packages required to set up VMs. If all else fails, that configuration can help you set up multiple installations of RHEL 6. 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. While in Chapter 1, you installed that solution during the installation process on a 64-bit system, you may also need to install the associated packages on a live system during an exam. In addition, there is a Virtual Machine Manager graphical console used by Red Hat to manage such VMs. Of course, that Virtual Machine Manager is a front end to command line tools that can also be used to install a system. Such tools can also be used to configure that system to be started automatically during the boot process.

While 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. So in this chapter, you'll learn how to set up an installation of RHEL 6 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, as the sample Kickstart file must be modified first, beyond unique settings for different systems. But once configured, you'll be able to set up as many installations of RHEL as you need using that baseline Kickstart file.

Access Remote Systems

The RHCSA objectives state that you need to know how to

- Access remote systems using SSH and VNC

If systems administrators had to be in physical contact with every system they have to administer, half of their lives would be spent en route from system to system. With tools such as the Secure Shell (SSH) and Virtual Network Computing (VNC), administrators have the ability to do their work remotely. While this chapter focuses on the uses of SSH, Chapter 9 focuses on the configuration of VNC. While SSH is automatically installed in a standard configuration in RHEL 6, custom configuration options such as passphrases are the province of the RHCE exam.

on the job

I probably should warn you that production systems should not be used for KVM-based virtual hosts. However, I'm writing this book on one such RHEL 6 system.

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, that's what you're going to do 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 ten 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 "overbook" the RAM and CPU on the local physical system. For example, on a system with 8GB of RAM, it's often reasonable to allocate 1GB each to ten VMs configured on that system.

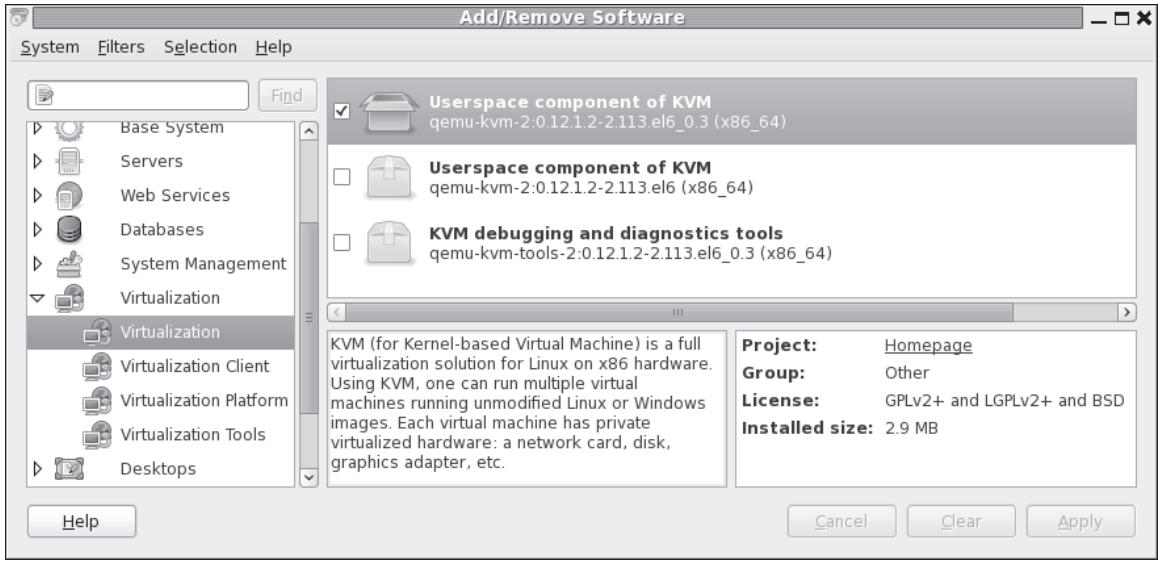
In practice, an administrator might replace ten physical machines on an older network with two physical systems. Each of the ten VMs would be installed twice, once on each physical system. Of course, those two physical systems require some powerful hardware. But the savings otherwise are immense, not 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 6, the Add/Remove Software tool can be a great help. Log in to the GUI as a regular user. To open it from the GUI, click System | Administration | Add/Remove 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-hand pane, click the arrow next to Virtualization. The four virtualization package groups should appear. Click the Virtualization package group, and the first package in that group to see a screen similar to that shown in Figure 2-1.

The list may be a bit too comprehensive; in Figure 2-1, you might note two different versions of the qemu-kvm package. Generally, only the latest version of the

FIGURE 2-1 Add/Remove Software Tool



package is required. All you need to do to install KVM packages is select appropriate packages from the Virtualization, 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.

TABLE 2-1 Packages Associated with Virtualization

Package	Description
qemu-kvm	The main KVM package
python-virtinst	Command line tools and libraries for creating VMs
virt-manager	GUI VM administration tool
virt-top	Command for VM statistics
virt-viewer	GUI connection to configured VMs
libvirt	C language toolkit with the libvirtd service
libvirt-client	C language toolkit for VM clients

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 6 system with a 64-bit CPU. While none of these packages are in the Virtualization Tools group, those packages may be helpful in real life. It includes tools that can help read and manage the VM disk images. If you choose to convert images from Xen or from some formats of VMware, the virt-v2v package is what you need.

Installation with the Add/Remove Software tool is fairly simple. Just select (or deselect) 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 with the yum install *packagename* command.

The Right KVM Modules

In most cases, installation of the right packages is good enough. Appropriate modules should be loaded automatically. Before KVM can work, the associated 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      45578  4
kvm            291875  1 kvm_intel
```

or

```
kvm_amd        35678  4
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 module, whether it be kvm_intel or kvm_amd:

```
# modprobe kvm
```

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.

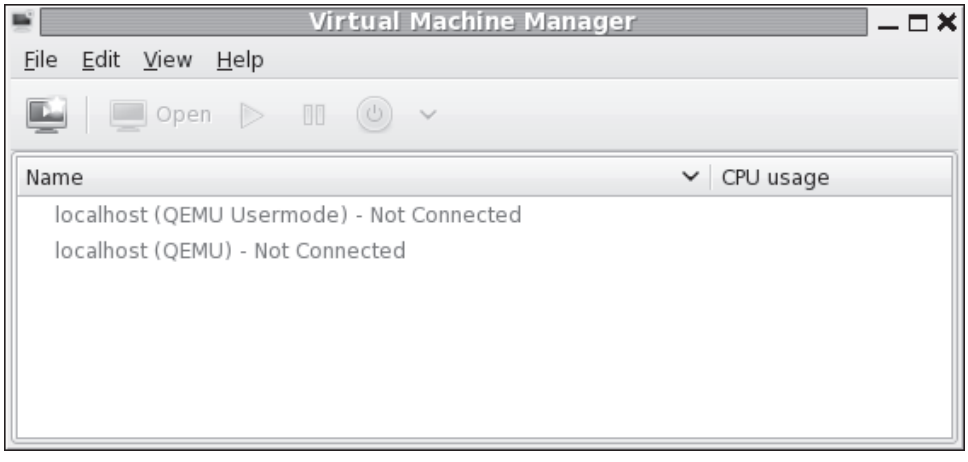
In some cases, two hypervisors, also known as virtual machine monitors, are shown on the localhost system. These hypervisors work with QEMU as processor emulators within the virtual machines. QEMU is also known by its former acronym, the quick emulator. If a usermode emulator appears, it is appropriate if you want to run 32-bit applications on a 64-bit system. But that's inefficient. In most cases, you'll want to create and manage VMs in the localhost (QEMU) regular mode.

on the job

As of the RHEL 6 release, the QEMU usermode emulator is subject to bug 634876 at https://bugzilla.redhat.com. The issue was addressed with an update to the virt-manager package.

FIGURE 2-2

Virtual Machine Manager



Connections to Hypervisors

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. It opens an Add Connection window that allows you to select:

- A hypervisor, normally KVM or Xen. (Xen was the default hypervisor on RHEL 5.)
- A connection, which may be local, or remote using a connection 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. It opens a details window named after the host of the local system, as shown in Figure 2-3.

As shown, the Overview tab lists the basics of the VM configuration, as discussed in Table 2-2.

FIGURE 2-3

VM Host Details.

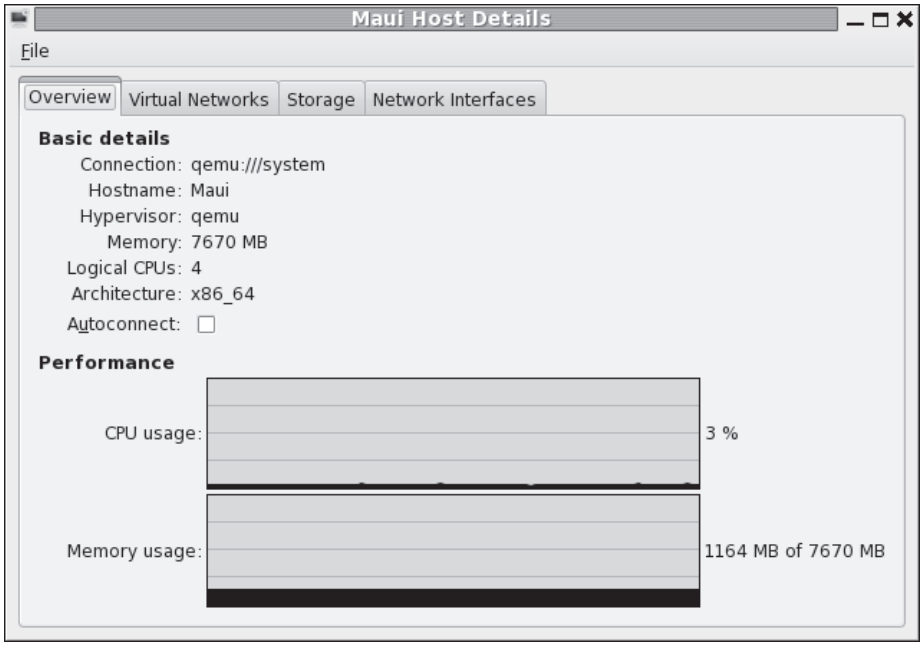


TABLE 2-2 VM Host Details

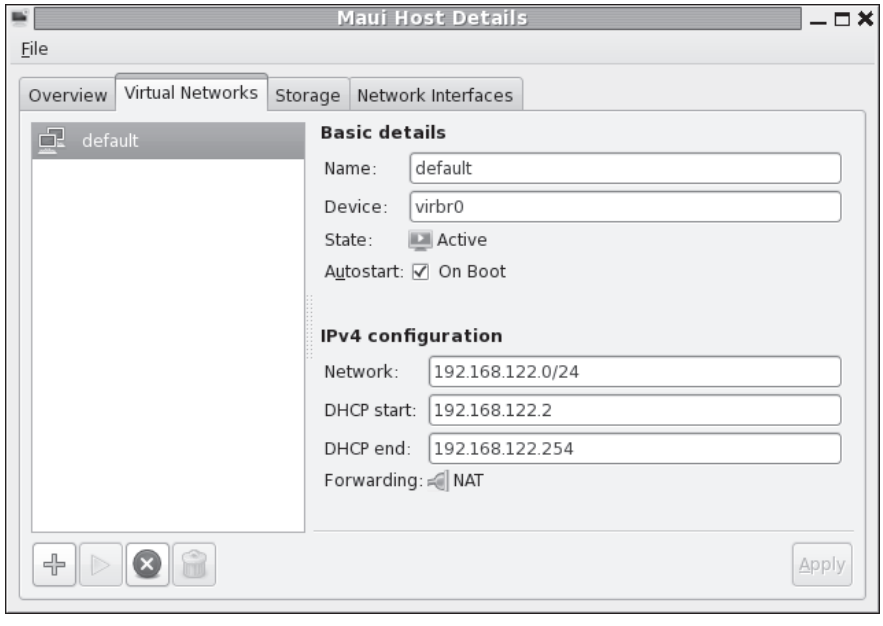
Setting	Description
Connection	Universal Resource Identifier (URI) for the hypervisor
Hostname	Hostname for the VM host
Hypervisor	QEMU is used by KVM
Memory	Available RAM from the physical system for VMs
Logical CPUs	Available CPU cores; "4" indicates four CPUs or a quad-core system
Architecture	CPU architecture
Autoconnect	Whether to automatically connect to the hypervisor during the boot process

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.

FIGURE 2-4 VM Host Details.



You'll note the given network is configured to start automatically when the VM is booted. So if there's an appropriate virtual network card 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 forward information using Network Address Translation (NAT).

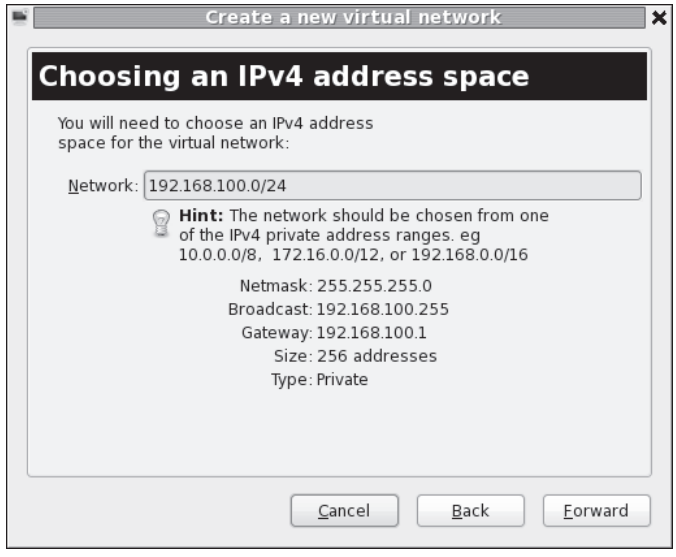
With the buttons in the lower-left part of the screen, you can add a new virtual network, start and stop an existing virtual network, and delete that network. In Exercise 2-1, you'll create a second virtual network. For the next section, stay in the host details window for the current hypervisor.

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 6 system configured, is based on an in-process installation of RHEL 6, and assumes the Virtual Machine Manager as discussed early in this chapter.

1. 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 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 appropriate entries for other network information, as shown in the illustration. Click Forward to continue.



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, at least one cable "modem" uses IP address 192.168.100.1 for maintenance. In that case, the noted 192.168.100.0/24 network would make that cable "modem" inaccessible. If you have such hardware, do 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 by a DHCP client. Per Chapter 1, Table 1-2, 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. Now you'll want a system that forwards network communication 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.

9. Review the summary of what's been configured. If satisfied, click Finish. The outsider network will now be available for use by new VM systems and network cards.

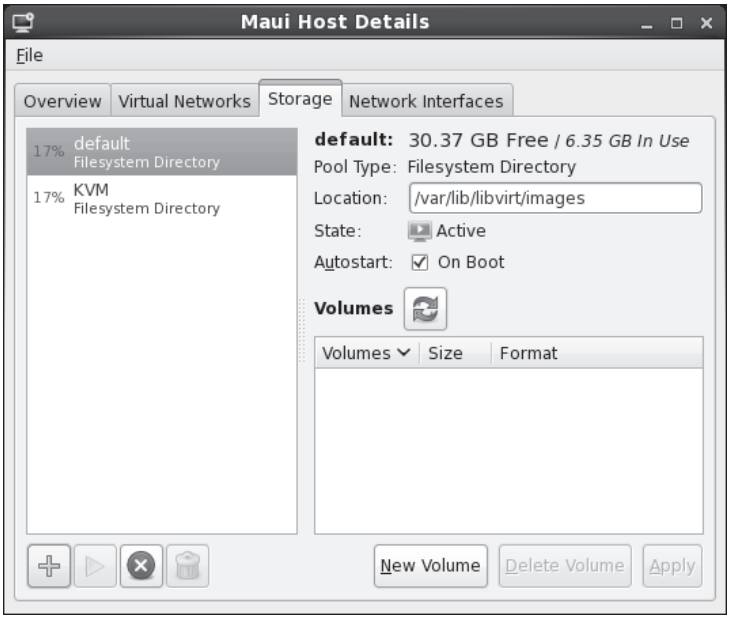
Virtual Storage on a Hypervisor

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.

FIGURE 2-5

VM Storage Details



As I had already dedicated the largest amount of free space to a partition dedicated to my /home directory, I chose to create dedicated storage in that area. To that end, I created a /home/michael/KVM directory to contain my VM files used for virtual hard drives.

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

```
$ mkdir /home/michael/KVM
# su - root
# chcon -R --reference /var/lib/libvirt/images /home/michael/KVM
# rmdir /var/lib/libvirt/images
# ln -s /home/michael/KVM /var/lib/libvirt/images
```

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 defined in Chapter 11.

Virtual Network Interfaces on a Hypervisor

Now you'll examine the virtual 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 devices shown in Figure 2-6 specify the physical devices to which KVM-based VMs can connect.

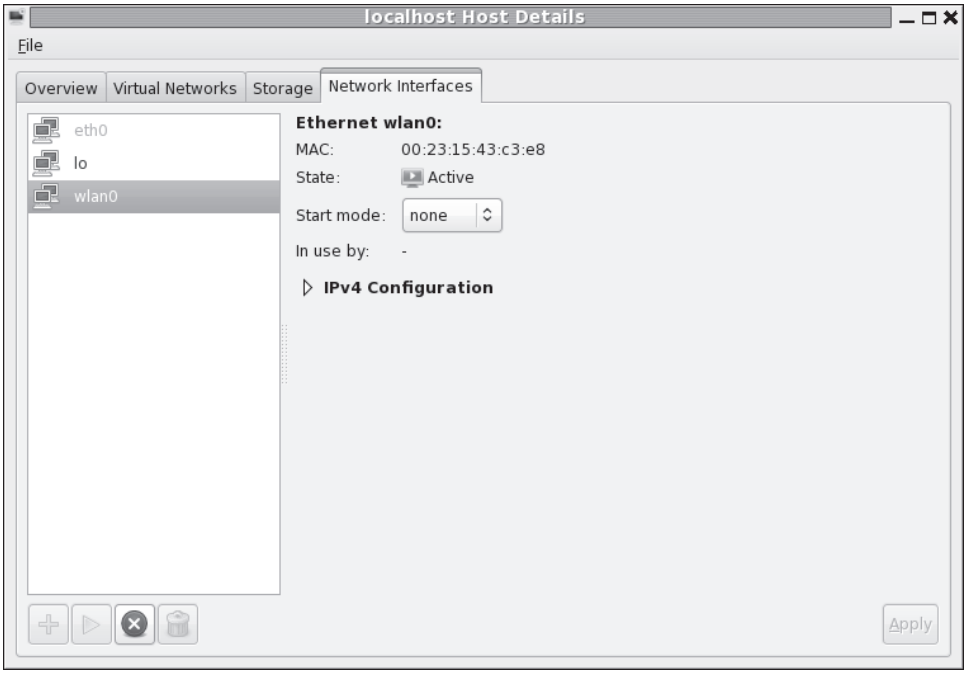
If the local system connects via the standard first Ethernet network card, the defaults with device eth0 should be sufficient. A properly configured VM should have access to external networks, given the firewall and IP forwarding configuration options described in Chapter 1. However, Figure 2-6 specifies one added interface, wlan0. That's a typical wireless network interface device file.

In the same fashion as with the Virtual Network and Storage tabs, you can add 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** Binds a physical and a virtual interface; typically associated with Xen.
- **Bond** Connects two or more network interfaces as if they were a single interface.

FIGURE 2-6

VM network cards



- **Ethernet** Sets up a virtual interface as a bridge.
- **VLAN** Connects a real or a virtual network interface to the VM system.

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 desired hypervisor, click New, and follow the prompts that appear. But as it's important to understand the process in detail, you'll read about the process, step-by-step. New VMs can be configured not only from the GUI, but also from the command line interface. As with other Linux services, the resulting VMs are configured in text files.

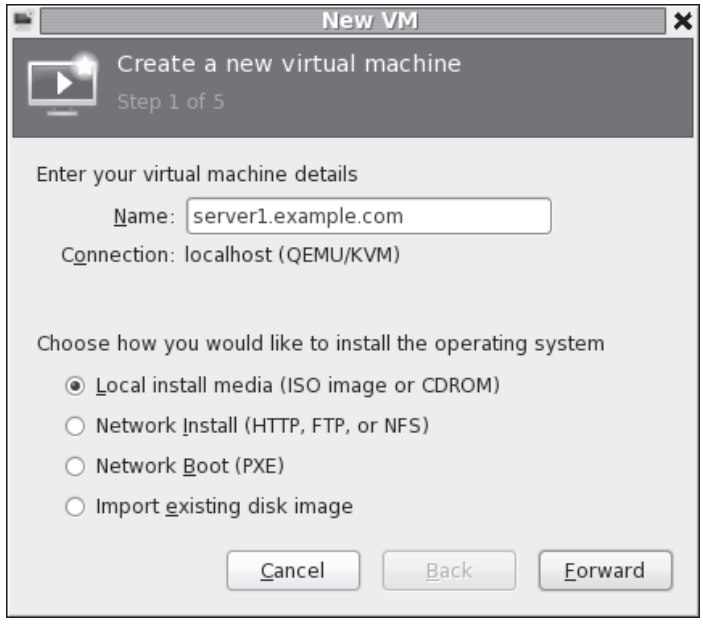
Configure a Virtual Machine on KVM

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. Run the **virt-manager** command. It should open the Virtual Machine Manager a touch more quickly than the GUI menu command. Right-click the localhost (QEMU) hypervisor and click Connect in the pop-up menu that appears. If prompted, enter the root administrative password. 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.
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 CDROM) or from a Network Installation server. That server may be associated with the HTTP, NFS, or FTP protocol. Select Local Install

FIGURE 2-7

Create a New VM.



Media 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 selectable, as shown in Figure 2-8. But in this case, select Use ISO Image and click Browse to navigate to the location of the RHEL 6 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 or 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 6. 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.
6. Now you'll set up the hard drives for the VM, in the screen shown in Figure 2-10. While 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

FIGURE 2-8

Virtual Machine Media Installation Options

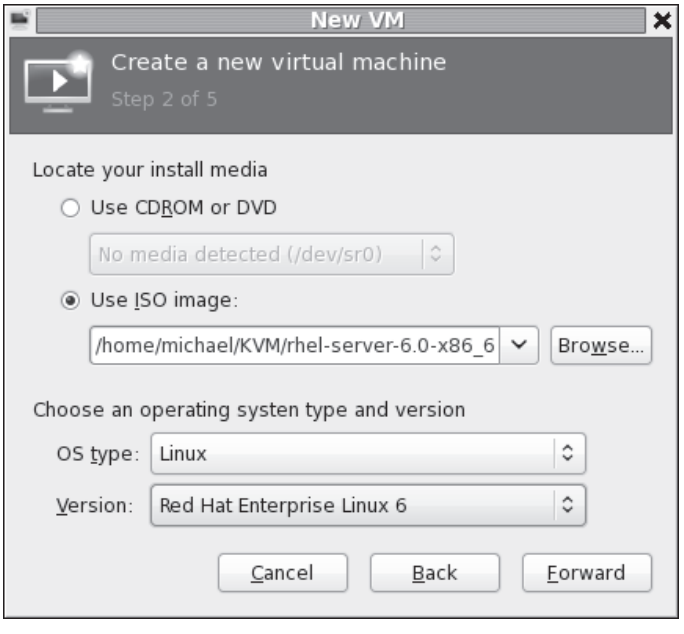


FIGURE 2-9

Virtual Machine RAM and CPU Selection.

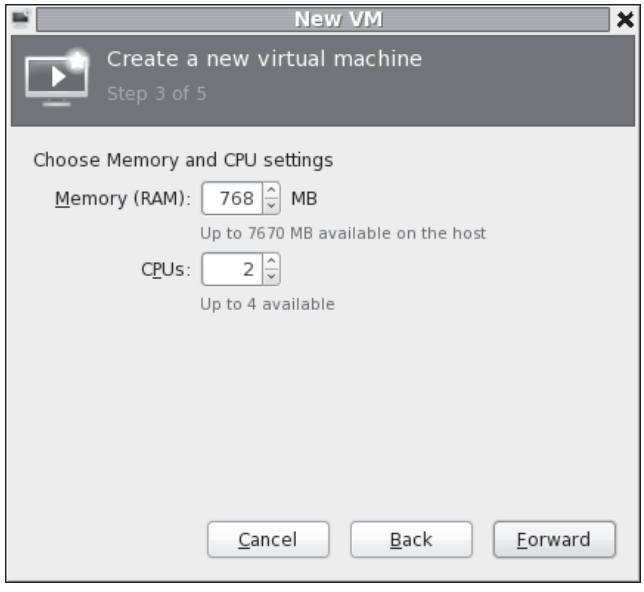
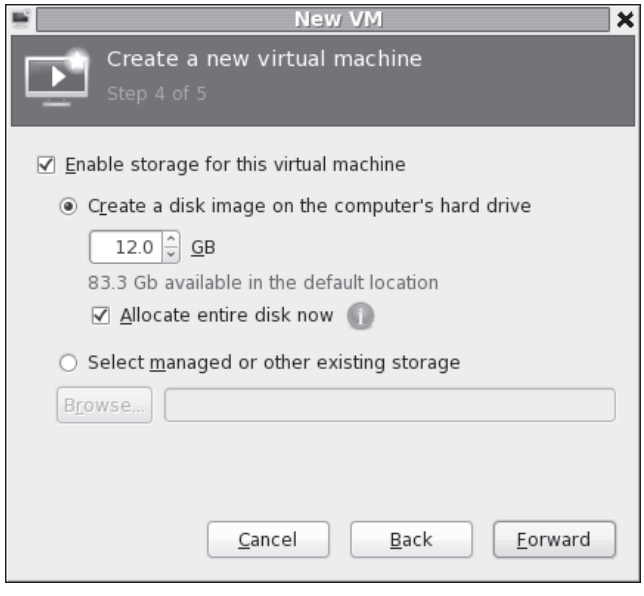


FIGURE 2-10

Create a Virtual Hard Drive



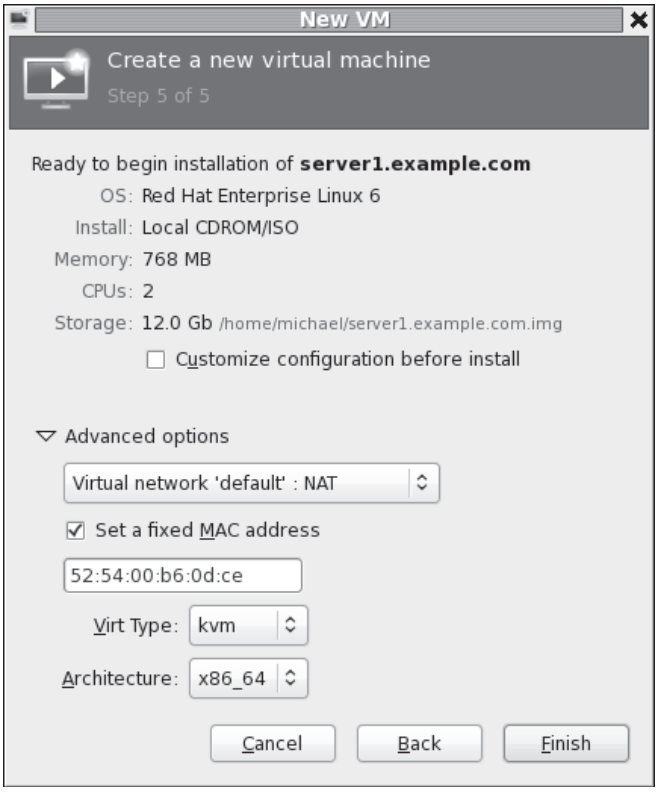
discussed earlier in this chapter. On an exam, it's likely that you'll have more than sufficient room in the /var/lib/libvirt/images directory. The Select Managed Or Other Existing Storage option supports the creation of a virtual hard drive in a different pre-configured storage pool.

7. Make sure the virtual drive is 12GB 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 from available virtual networks. If you performed Exercise 2-1, the 192.168.100.0/24 network address option should be available.

FIGURE 2-11

Create a Virtual Hard Drive

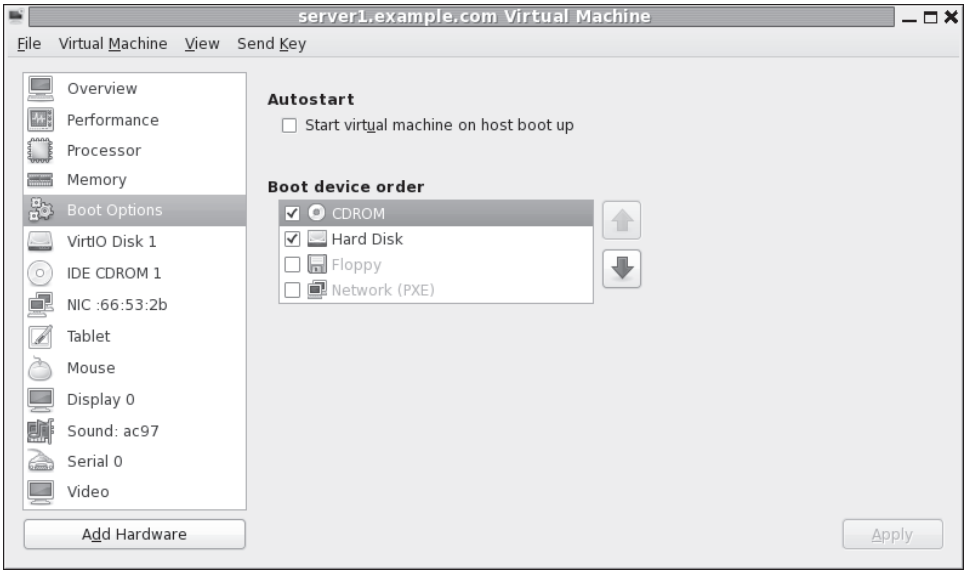


9. The system may take a little time to create the VM, including the huge file that will serve as the virtual hard drive. When complete, the Virtual Machine Manager should automatically start the system from the RHEL 6 installation DVD in a VNC window.
10. If the new system doesn't start automatically, 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 6 in the VM as discussed in Chapter 1.

    If you choose to check the integrity of the DVD during the installation process within the VM, the installation program "ejects" that DVD. KVM does not recognize that "eject." In that case, you'd have to click View | Details, select the IDE CDROM1 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.
12. Be aware, when selecting software, 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. A basic server for the host systems on a VM is sufficient, with the Desktop, Fonts, X Window System, and Internet Browser package groups added on.
13. 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!
14. If the system tries to reboot from the DVD, you would need to shut down the system. To do so, click Virtual Machine | Shut Down | Force Off. (The Shut Down | Shut Down option does not work at this time.)
15. If this is the first time you've run that command sequence, the Virtual Machine Manager prompts for confirmation. Click Yes.
16. Now click View | Details.
18. One way to change the boot order is to highlight CDROM, and then click the down arrow button. Click Apply, or the changes won't be recorded.
19. Now click View | Console and then Virtual Machine | Run. The system should now boot normally into the First Boot process discussed in Chapter 1.

FIGURE 2-12

Boot Options in the VM



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."

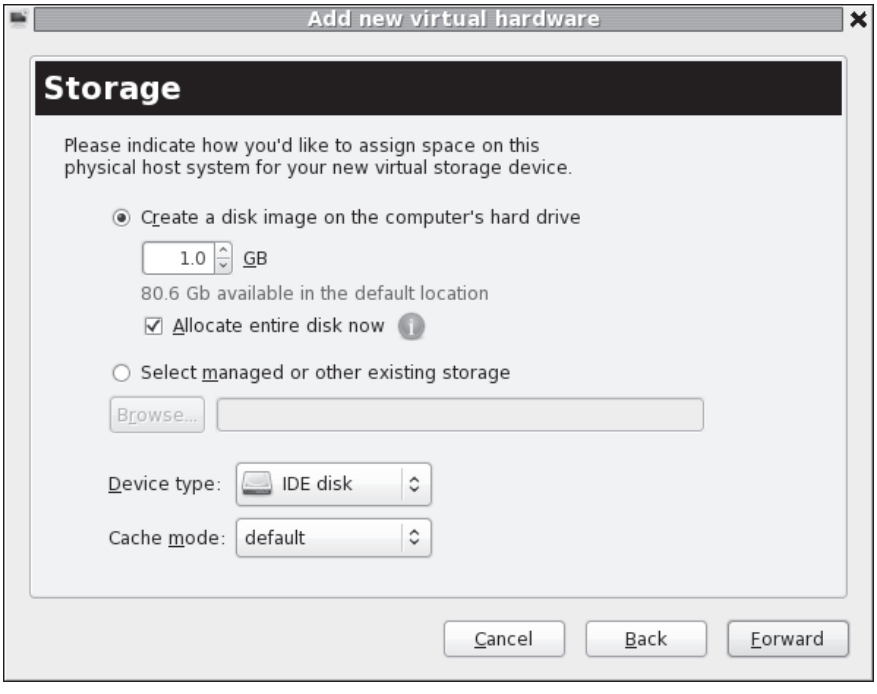
One more reason for the use of VMs is the ease with which additional virtual hard drives can be added. The process varies by VM solution. For the RHEL 6 default Virtual Machine Manager with KVM solution, you can do so from the machine window by clicking View | Details. You'll see an Add Hardware option in this screen.

EXERCISE 2-2

Add Virtual Hard Drives

In this exercise, you'll create an additional virtual hard drive on a KVM-based VM. It assumes there's an existing KVM VM for that purpose, along with the use of the GUI Virtual Machine Manager. Of course, given that it's KVM, it assumes that the local physical system supports hardware virtualization.

1. Open the Virtual Machine Manager. From the command line in a GUI, run the **virt-manager** command.
2. Highlight the regular 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.
3. If prompted, enter the root administrative password and click Authenticate.
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 drop-down menu, and click Forward to continue.
7. In the Storage window that appears, shown in the illustration, set up a 1.0GB drive, select Allocate Entire Disk Now, select IDE Device Type, in default Cache Mode. (You can also select a SCSI, USB, or Virtual (Virtio) Disk.) Make desired choices and click Forward to continue.



8. You'll 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. To learn more about KVM, it would be useful to set up something different, such as a SCSI disk. However, that's not required.
10. The next time you boot this system, run the **fdisk -l** command. It should confirm appropriate information about the configured hard drive devices.

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 I 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 KB, 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.

While you can edit this configuration file directly, changes aren't implemented until the **libvirtd** script in the /etc/init.d directory is restarted with a command like **/etc/init.d/libvirtd restart**.

Control Virtual Machines from the Command Line

Of course, command line tools can be used to create, clone, convert, and install VMs on RHEL 6. 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 --prompt** command. The command automatically prompts for the required information described earlier. Look at the command and prompts shown in Figure 2-14.

FIGURE 2-13

The Configuration file for a KVM Virtual Machine.

```
<domain type='kvm'>
  <name>server1.example.com</name>
  <uuid>d60a8c0a-d795-c0fc-893f-174ac0f37f6e</uuid>
  <memory>786432</memory>
  <currentMemory>786432</currentMemory>
  <vcpu>2</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.0.0'>hvm</type>
    <boot dev='hd'/>
    <boot dev='cdrom'/>
  </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='0x05' function='0x0'/>
    </disk>
    <disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <source file='/var/lib/libvirt/images/rhel-server-6.0-x86_64-boot.iso'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
      <address type='drive' controller='0' bus='1' unit='0'/>
"/etc/libvirt/qemu/server1.example.com.xml" 80L, 2844C
```

FIGURE 2-14

Configure a VM with the virt-install command.

```
[root@Maui ~]# virt-install --prompt
What is the name of your virtual machine? tester1.example.com
 How much RAM should be allocated (in megabytes)? 768
 What would you like to use as the disk (file path)? /home/michael/KVM/tester1.example
.com.img
 How large would you like the disk (/home/michael/KVM/tester1.example.com.img) to be (
in gigabytes)? 12
 What is the install CD-ROM/ISO or URL? ftp://192.168.122.1/pub/inst


Starting install...
Retrieving file .treeinfo...                              | 3.3 kB     00:00 ...
Retrieving file vmlinuz...                                | 7.2 MB     00:00 ...
Retrieving file initrd.img...                             |  57 MB     00:00 ...
Allocating 'tester1.example.com.img'                      |  12 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.
```

exam Watch

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 VNC window with a graphical view of the given installation program.

If you make a mistake with the **virt-install** command, you can abort the process by pressing CTRL-C. But be aware that 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. Thus, 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
   ```

2. Delete the associated XML configuration file in the /etc/libvirt/qemu directory. For the given system name, that file would be tester1.example.com.xml.
3. If desired, you may also want to delete the virtual disk file, normally created in the /var/lib/libvirt/images directory. However, this is not necessary if the file is of an acceptable size, as it is reusable. For the given system name, that file would be tester1.example.com.img.
4. With the following command, restart the VM daemon, to erase the tester1.example.com system from RAM:

   ```
   # /etc/init.d/libvirtd restart
   ```

5. 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 --prompt** command can't 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 would install a system named outsider1.example.org automatically from a Kickstart file named ks1.cfg from the

TABLE 2-3

Command Switches for virt-install

Switch	Description
-n (--name)	Sets the name for the VM
-r (--ram)	Configures the amount of RAM in MB
--disk	Defines the virtual disk; often used with **path=/var/lib/libvirt/images/*virt*.img**
-l (--location)	Specifies the directory or URL with the installation files (equivalent to **--location**)
-x (--extra-args=)	Includes extra data, such as the URL of a Kickstart file

FTP server on the noted IP address, with 768MB of RAM, and a outsider1.example.org.img virtual disk.

```
# virt-install -n outsider1.example.org -r 768 --disk \
path=/var/lib/libvirt/images/outsider1.example.org.img \
-l ftp://192.168.122.1/pub/inst \
-x "ks=ftp://192.168.122.1/pub/ks1.cfg"
```

If you run the **virt-install** command with a Kickstart file customized from a previous KVM-based installation, it may fail. A standard KVM-based installation configures the first virtual disk on device /dev/vda. In contrast, the **virt-install** command assumes that the first virtual disk in question is device /dev/sda. The Kickstart file described later in this chapter would need to be customized for this difference.

The noted file 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. The exception is the **-x**, another name for the **--extra-args=** switch. You may note additional switches, which are helpful but are not required for RHEL 6 installation. However, they are required to look for the given Kickstart file. So remember the format for the extra arguments, with the quotes, which may also be expressed as:

```
--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** command 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, the name of each VM is a domain, which is used by different **virsh** commands.

Take a look at the output to the **virsh --list all** command on my 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
```

Unfortunately, the **virsh shutdown** command does not work as of this writing. So to actually stop a VM from the command line, you'd have to run a somewhat more severe command:

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

The **virsh destroy** command switch 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.

***While the* shutdown *command works in Kickstart configuration files, to shut down a system from the command line, the functional command is* poweroff.**

TABLE 2-4 Commands at the virsh Prompt

virsh Command	Description
autostart <domain>	Start a domain during the host system boot process
capabilities	Lists abilities of the local hypervisor
edit <domain>	Edits the XML configuration file for the domain
list --all	List all domains
start <domain>	Boot the given domain
shutdown <domain>	Gracefully shut down the given domain

exam Watch

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

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.

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 boots the noted tester1.example.com system 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 like **ssh** to connect to that VM system normally. However, from the virtual host GUI, you'll still have to start the Virtual Machine Manager and connect to the associated hypervisor to actually connect to 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 following 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 --prompt** 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.

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.

FIGURE 2-15

Cloning a Virtual Machine

```
[root@Maui ~]# virt-clone --prompt
What is the name of the original virtual machine? server1.example.com
 What is the name for the cloned virtual machine? tester1.example.com
 What would you like to use as the cloned disk (file path) for '/var/lib/libvirt/image
s/server1.example.com.img'? /var/lib/libvirt/images/tester1.example.com.img
 What would you like to use as the cloned disk (file path) for '/var/lib/libvirt/image
s/server1.example.com-1.img'? /var/lib/libvirt/images/tester1.example.com-1.img
 What would you like to use as the cloned disk (file path) for '/var/lib/libvirt/image
s/server1.example.com-2.img'? /var/lib/libvirt/images/tester1.example.com-2.img
Allocating 'tester1.example.com.img'                          |  12 GB     01:23
Allocating 'tester1.example.com-1.img'                        | 1.0 GB     00:01
Allocating 'tester1.example.com-2.img'                        | 1.0 GB     00:01

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

The first time you boot a cloned machine, it may be best to boot it into runlevel 1. As described in Chapter 5, runlevel 1 does not start most services, including networking. In that case, you'll be able to modify any fixed 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 to change the hardware address for the related network card, to avoid conflicts with the original network card.

While 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 Red Hat. 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 6, not only 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.

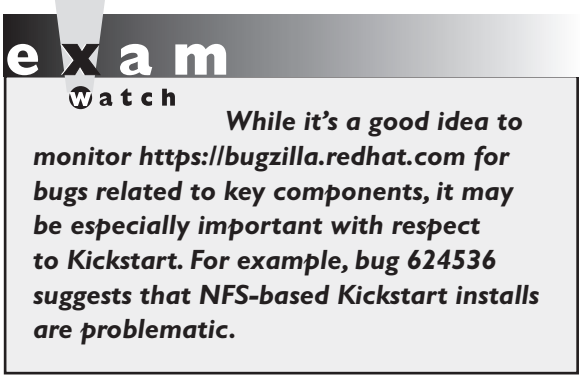
The steps described in this section assume a connection to the FTP server with RHEL 6 installation files created and configured in Chapter 1, Lab 2.

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. While 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.



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, host names, 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. I'll describe this process in more detail shortly.

2. Mount the desired local media. You may need to run a command like **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/sdb /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, a Kickstart configuration file on an FTP server may be a security risk. It's almost like the DNA of a system. If a cracker gets a hold of that file, he could use that to set up a copy of your systems, to see how to break into and compromise your data. As that file normally contains a root administrative password, you should change that password as soon as that system is booted for the first time.

on the Job

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 cracker with the right tools and a copy of that Kickstart configuration file can decrypt that password faster than you might expect.

5. 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.
6. Now try to access the Kickstart file on the local media. Boot the RHEL 6 installation CD/DVD. When the first menu appears, highlight Install Or Upgrade An Existing System and press TAB. Commands to Anaconda should appear, similar to the following. A cursor should appear at the end of that line.

   ```
   vmlinuz initrd=initrd.img
   ```

7. 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 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. Furthermore, the 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 initrd=initrd.img** line described earlier in Step 6:

```
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
```

However, you should not use NFS for sharing Kickstart files until the aforementioned bug 624536 has been addressed.

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.

on the job

Red Hat is working to ease the process of creating a Kickstart-based installation server. For more information, see the Cobbler project at https://fedorahosted.org/cobbler/.

Sample Kickstart File

I've based this section on the anaconda-ks.cfg file created when I installed RHEL 6 on a KVM-based VM. I've added a number of comments. While 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 on 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, available Kickstart documentation within an installed RHEL 6 system is somewhat sparse. In other words, you can't really rely on man pages or files in the /usr/share/doc directory for 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.

While most of the options are self-explanatory, I've interspersed my explanation of each command within the file. This file illustrates just a small portion of available commands. For more information on each command (and options) in this file, read the latest RHEL 6 Installation Guide, which is available online at http://docs.redhat.com/docs/en-US.

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.

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. But as some Kickstart options change the partitions on a hard drive, even tests can be dangerous. So 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 my anaconda-ks.cfg files. The first two lines are comments that tell me that this file was created during the installation process for RHEL 6:

```
# Kickstart file automatically generated by anaconda.
#version=RHEL6
```

The first command is simple; it starts the installation process. It defaults to the first available local media; in this case, the first RHEL installation DVD/CD or USB key.

```
install
```

The next step is to specify the source of the installation files. To use RHEL 6 DVDs, enter **cdrom**. To install from an NFS server, specify the URI as follows. If there's a reliable DNS server on 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 I specify 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 6 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=/home/michael/
```

The **lang** command specifies 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, as it specifies the keyboard to configure on this computer.

```
lang en_US.UTF-8
keyboard us
```

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 line configures static IP address information, with the noted network mask (**--netmask**), gateway address (**--gateway**), DNS server (**--nameserver**), and computer name (**--hostname**).

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

Please note that all options 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 6 installation process, the Kickstart configuration file can specify that password in encrypted format. While encryption is not required, it can at least delay a cracker who might break into a system after installation is complete. Since 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/
```

As for security, the **firewall** directive suggests that it's enabled. When coupled with **--service=ssh**, it specifies the service port number that's allowed through the firewall, based on how it's defined in the /etc/services file.

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

Next, the **authconfig** command sets up the Shadow Password Suite (**--enableshadow**), the SHA 512 bit encryption algorithm for password encryption (**--passalgo=sha512**), and authentication with any existing fingerprint reader.

An password encrypted to the SHA512 algorithm starts with a **$6**, like the root administrative password just shown.

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

The **selinux** directive can be set to **--enforcing**, **--permissive**, or **--disabled**.

```
selinux --enforcing
```

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 **--utc** 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
```

The default bootloader is GRUB. It should normally be installed on the Master Boot Record (MBR) of a hard drive. You can include a **--driveorder** switch to specify the drive with the bootloader and an **--append** switch to specify commands for the kernel. While the given **crashkernel=auto** option should automatically select available memory upon a crash, early reports on RHEL 6 suggest that you may need to replace it with a specific memory location such as **crashkernel=128M@16M**.

```
bootloader --location=mbr --driveorder=vda
--append="crashkernel=auto rhgb quiet"
```

As suggested by the comments that follow, it's first important to clear some existing set of partitions. First, the **clearpart --drives=vda --all --initlabel** directive clears all volumes on the vda virtual hard drive. If it hasn't been used before, **--initlabel** initializes that drive. Of course, before such a command takes effect, any existing comment character (#) must be removed. The **ignoredisk** directive that follows specifies volumes only on the noted vda drive. Of course, this works only if there is an specified virtual drive on the target VM. (It's possible to specify PATA or SCSI drives on such VMs, which would conflict with these directives.)

```
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
```

```
clearpart --drives=vda --all --initlabel
ignoredisk --only-use=vda
```

If you're planning to use this Kickstart file with the **virt-install** command described earlier, you'll need to substitute the sda device for vda, as the **virt-install** command does not normally use that virtual hard drive device file.

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=ext4 --size=500
part / --fstype=ext4 --size=8000
part swap --size=1000
part /home --fstype=ext4 --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 those **--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.

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/ --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**. I personally prefer the **shutdown** directive; if you want to avoid the First Boot process described earlier, you can also include the **firstboot --disabled** directive. As there's no way to set up a Kickstart file with answers to the First Boot process, that **--disabled** directive helps automate the Kickstart process.

```
shutdown
firstboot --disabled
```

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. So instead, you may prefer to substitute the following directive:

```
shutdown
```

While other options are available with respect to setting up RAID arrays and logical volumes, the focus of the Red Hat exams is to set up such volumes after

installation is complete. What follows is a list of package groups that are installed through this Kickstart configuration file. These names correspond to the names that you can find in the *-comps-rhel6-Server.xml file in the RHEL 6 DVD /repodata directory described in Chapter 1. Since the list is long, the following is just excerpts of package groups (which start with the @) and package names:

```
%packages
@ base
@ console-internet
...
nss-pam-ldapd
perl-DBD-SQLite
%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. But the %post directive and anything that follows is not required.

```
%post
```

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. This exercise even configures the SELinux context for that Kickstart file.

As 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 Chapter 1 Lab 2.

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. 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. 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 ks.cfg file are active, as they're commented out by default in the /root/anaconda-ks.cfg file. 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 on the system, the **--drives=vda** switch can focus on the first virtual drive on a KVM-based VM.
4. Review the location of the installation server, associated with the **url** or **nfs** directives. 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.
5. Make sure the following directives are included just before the **%post** directive at the end of the file:

```
shutdown
firstboot --disabled
```

6. 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 to all users. One method is with the following command:

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

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

```
# chcon --reference /var/ftp/pub /var/ftp/pub/ks.cfg
```

8. Make sure any existing firewalls do not block the communication port associated with the installation server. For detailed information, see Chapter 4. The simplest, though insecure way to do so is with the following command:

```
# iptables -F
```

9. Prepare the second computer so that it has sufficient hard drive space. That second computer can be a KVM-based VM on a local host. Boot that second computer into the RHEL 6 DVD.
10. 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 list, 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. When I installed it on my system, this one command found 27 dependencies, meaning that command installed a total of 28 packages.

on the job

Those of you sensitive to proper 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
```

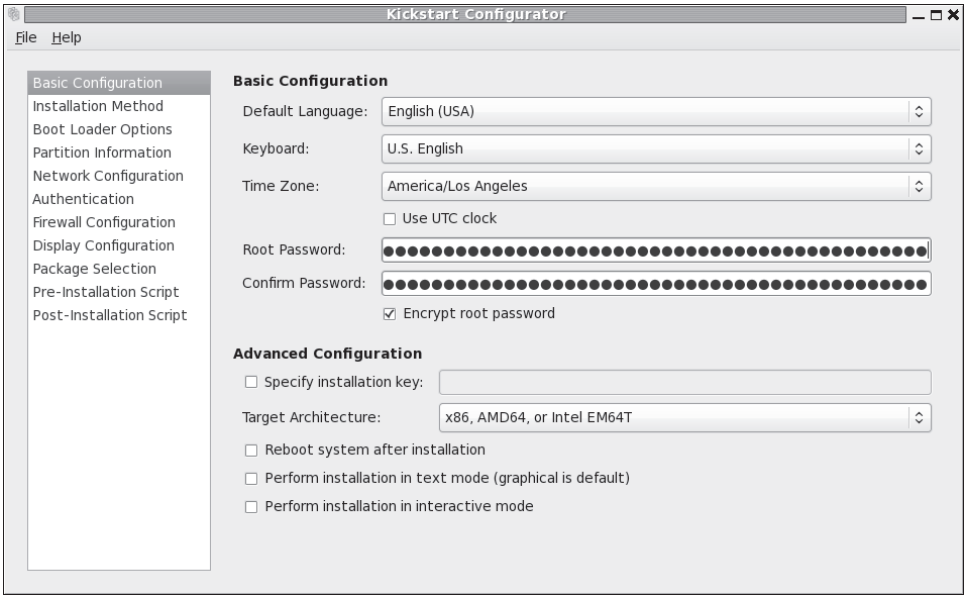
It should open the Kickstart Configurator shown in Figure 2-16. (Of course, it's probably a good idea to back up the anaconda-ks.cfg file first.)

on the job

Before starting the Kickstart Configurator, it's best to make sure there's an active connection to a remote RHEL 6 repository. If you're using a test copy of RHEL 6, that's also possible by connecting to the installation source created in Chapter 1, Lab 2 using the techniques described in Chapter 7.

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-hand 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-hand 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** Supports computers in which the hardware clock is set to the atomic realization of 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 you 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.
- **Perform Installation In Interactive Mode** Allows review of the steps associated with a Kickstart installation.

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 alternate boot loader such as Partition Magic or System Commander) to point to GRUB on the first sector of the Linux partition with the /boot directory.

Partition Information

The Partition Information options determine how this installation configures the hard disks on the affected computers. While 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 yes** command in the Kickstart file.

Don't use the* zerombr yes *option if you want to keep an alternate bootloader on the MBR such as Partition Magic or the Windows 7 Bootmgr.

You can remove partitions depending on whether they've been created to a Linux filesystem. If you're using a new hard drive, it's important to Initialize the Disk Label as well. Click the Add command; 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. This section also allows you to set up authentication information for various protocols:

- **NIS** Network Information Service to connect to a login authentication database on a network with Unix and Linux computers.
- **LDAP** In this context, the Lightweight Directory Assistance Protocol is an alternative login authentication database.
- **Kerberos 5** The MIT system for strong cryptography to authenticate users on a network.
- **Hesiod** Associated with Kerberos 5.
- **SMB** Samba (CIFS) connects to a Microsoft Windows–style network for login authentication.

- **Name Switch Cache** Associated with NIS for looking up passwords 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 like 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. While there is a lot of debate on the superiority of GUI- or 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 file, 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 First Boot 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. You should recognize it as the custom installation screens shown during the installation process.
As noted earlier, the associated screens are blank if there's no current connection to a remote repository such as updates from the RHN or the installation server described earlier.

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

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. Therefore, in this section, you'll examine how to use the **ssh** command to access remote systems as a client.

As suggested earlier, the stage is already set by the default installation of SSH on standard installations of RHEL 6. While firewalls are enabled by default, the standard RHEL 6 firewall leaves port 22 open for SSH access. Related configuration files are stored in the /etc/ssh directory. Detailed server configuration is part of the RHCE requirements. Related client commands are **scp** and **sftp**, also 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 messages that are sent between SSH clients and servers. And that's important on a public network like the Internet. RHEL incorporates SSH version 2, which includes a key exchange algorithm, which is the enhancement relative to SSH version 1.

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. There are four directives 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 for client/server authentication:

```
GSSAPIAuthentication yes
```

This next directive supports remote access to GUI tools. 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
```

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 accesses that account on the noted server1 system:

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

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 I were to run the following command from my user michael account:

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

The **ssh** command assumes that I'm 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.
RSA key fingerprint is b9:8a:c8:cd:c3:02:87:b3:1c:a9:a7:ed:d8:9c
:28:b8.
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'server1.example.com,192.168.122.50'
```

```
(RSA) to the list of known hosts.
michael@server1.example.com's password:
```

One of the flaws of this type of **ssh** command is how it sends the password over the network. More secure access is possible using passphrase-based access to 1024-bit (and higher) private/public keypairs. But that more secure option is the province of the RHCE exam, discussed in Chapter 11. When the connection is made, a copy of the remote key fingerprint is appended to the user's ~/.ssh/known_hosts file. The RSA is a reference to the public encryption key from the remote system; the acronym is based on the last names of its developers, Rivest, Shamir, and Adelman. The public key is given to remote systems; the private key is kept on the local system to match authorized remote requests.

Once connected via **ssh**, you can do anything on the remote system that's supported by your user privileges on that remote system. For example, you can even shut down the remote system gracefully with the **poweroff** command. After executing that command, you'll 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. While the **-l** switch doesn't work with that command, it still can be used to log in to the account of any user on the remote system. While 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 copy over an encrypted connection, the **scp** command can help. For example, I created some of the screenshots for this book on the test VMs configured in Chapters 1 and 2. To transmit that screenshot to my laptop system, I 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 a passphrase-based connection has been established (as discussed in Chapter 11), the command prompts for the password of the user michael on the

system named server1. Once the password is confirmed, the **scp** command copies the F02-20.tif file in encrypted format to the noted directory on the remote system named server1.

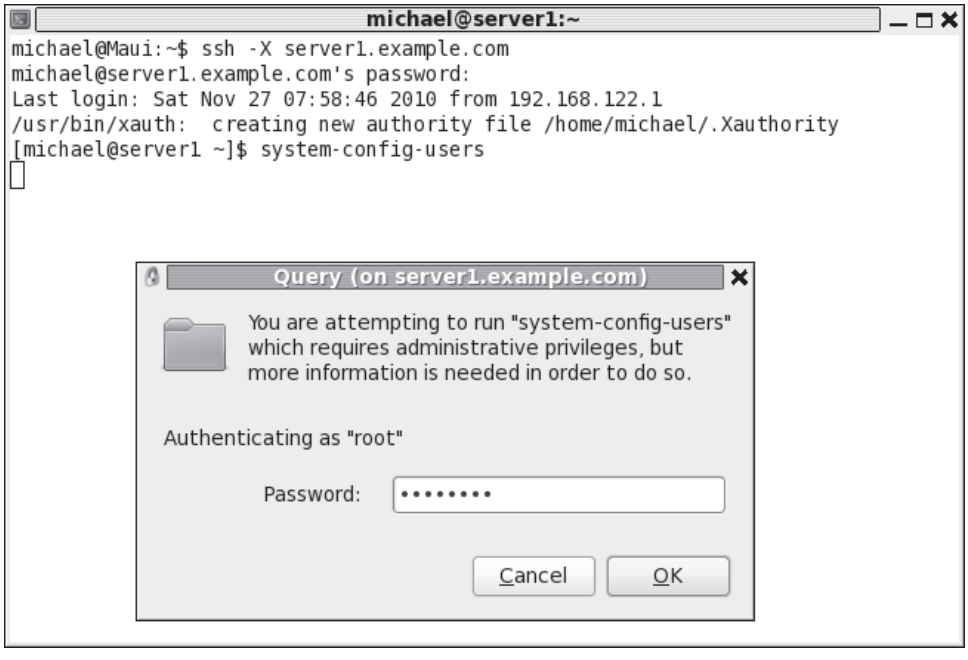
Graphical Secure Shell Access

The **ssh** command can be used to transmit GUI tools over a network. As strange as it sounds, it works if the local system works as a GUI server while you call remote GUI client applications from remote systems.

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. For example, you could use the command sequence shown in Figure 2-17 to administer users on the remote system.

FIGURE 2-17

Remote GUI Access via SSH.



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. While it's best to test services like Postfix with actual e-mail clients like 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 like Evolution. While 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:

- **telnet** and **nmap** to verify remote access to open ports.
- **mutt** as an e-mail client to verify the functionality of an e-mail server.
- **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 Ethereal can read that data fairly easily.

But **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 2.2.2)
```

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

```
^]
telnet> quit
```

You don't even need to execute the Escape character to quit; just type in the **QUIT** command.

If vsFTP were 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'd 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 web site 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 6 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 5.21 ( http://nmap.org ) at 2010-11-29 09:52 PST
Nmap scan report for server1.example.com (192.168.122.50)
Host is up (0.00027s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
```

FIGURE 2-18 Apply a port scanner locally

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

Starting Nmap 5.21 ( http://nmap.org ) at 2010-11-29 09:47 PST
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).
Hostname localhost resolves to 2 IPs. Only 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 0.07 seconds
[root@server1 ~]#
```

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 standard e-mail server services such as Postfix and sendmail. For example, once a server is configured for Post Office Protocol (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. For example, the **pine** utility would

create and store messages for user mj in the /home/mj/mail directory. Mail readers like **mutt**, **mail**, and **pine** 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 and then enter another addressee in the Cc: line, if desired. When you press ENTER, the message is sent and the **mail** utility stops and returns to the command line.

```
$ mail Michael
Subject: Test Message
Sent and received
Cc: mjang@example.com
$
```

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, with the Subject name of "hosts file":

```
$ mail -s 'hosts file' < /etc/hosts root@localhost
```

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 web sites 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 web site of your choice. For example. Figure 2-19 illustrates the result of the **elinks http://www.mheducation.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.

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, I could add the following text to home.html

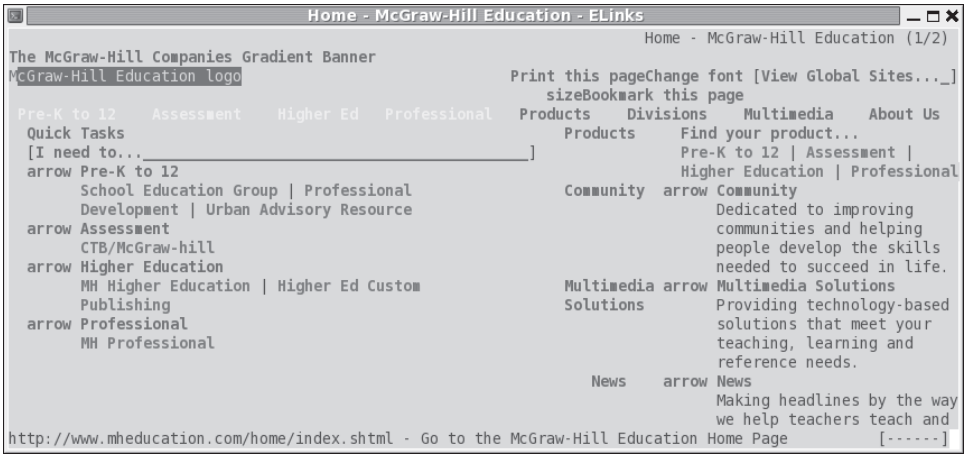
```
This is my home page
```

I 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
```

FIGURE 2-19

The elinks browser



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
```

But that client asks for a username and password. You could enter username anonymous and a random 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 6 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 have 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. Those of you studying for the RHCE will set up that kind of very secure FTP (vsFTP) server in. 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.

FIGURE 2-20

Commands in lftp.

```
[root@Maui ~]# lftp ftp.redhat.com
lftp ftp.redhat.com:~> help
    !<shell-command>                    (commands)
    alias [<name> [<value>]]            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]
    kill all|<job_no>                   lcd <ldir>
    lftp [OPTS] <site>                  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>           user <user|URL> [<pass>]
    version                             wait [<jobno>]
    zcat <files>                        zmore <files>
lftp ftp.redhat.com:~>
```

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 command itself.

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 like **modprobe kvm** to make sure appropriate modules are loaded. The Virtual Machine

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 local host computer

Manager can then be used to set up VMs using KVM on an RHEL 6 system. You can also use commands like **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. 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** copy command can copy files remotely over that encrypted connection.

When 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 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 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 named *vmname* to start automatically when the host system is booted.
- ❑ The **virsh start *vmname*** command starts the boot process for the VM named *vmname*.
- ❑ The **virsh destroy *vmname*** command in effect cuts power to the VM 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

- ❑ SSH is installed by default on RHEL 6. 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 benefits 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

11. What **ssh** command switch is used to specify a different user for remote logins?

_____

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

_____

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 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. However, even though it's required for the RHCSA exam, some readers may not have hardware that supports KVM. Options to KVM include VM solutions such as VMware, available from www.vmware.com or Virtualbox, open source edition, available from www.virtualbox.org.

Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter2/ subdirectory. In case you haven't yet set up RHEL 6 on a system, the first lab is presented here in the book.

Be aware, starting with Chapter 10, the labs for each of the chapters will be encrypted, using instructions described in each of those chapters.

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 12GB (with sufficient space for 11GB of data plus a swap partition, assuming at least 512MB of RAM). If you want to run the GUI installation program, you'll need at least 652MB of RAM. You'll also need room for an additional two virtual hard drives of 1GB each (14GB 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. That 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 this system 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 60GB of free space would be sufficient.

The steps described in this lab are general. By this time, you should have some experience with the installation of RHEL 6. In any case, exact steps vary with the type of installation and the boot media.

1. Start with the first RHEL 6 network boot CD. At the welcome screen, select Install Or Upgrade An Existing System. (If you use the RHEL 6 DVD, press TAB. Add **askmethod** after the initrd=initrd.img directive.)
2. Based on the steps discussed in Chapter 1, start the installation process for RHEL 6.
3. When prompted, set up the local system on a network configured on the KVM VM. 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.
4. At the appropriate step, 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.
5. When prompted to enter a hostname, enter **server1.example.com**. In the same screen, there will be a Configure Network button. Enter the IPv4 network information for the system in the window that appears.
6. Select custom partitioning at the appropriate step.

7. Create the first partition of about 500MB of disk space, formatted to the ext4 filesystem, and assign it to the /boot directory.
8. Create the next partition with 1GB of disk space (or more, if space is available), reserved for swap space.
9. Create a third partition with about 8GB disk space, formatted to the ext4 filesystem, and assign it to the top-level root directory, /.
10. Create another partition with about 1GB of disk space, and assign it to the /home directory.
11. Continue with the installation process, using your best judgment.
12. Choose to customize the package groups to be installed. Include a GUI, the X Window server. Installation of virtualization packages within a VM is not required.
13. Finish the installation normally.
14. Reboot when prompted and log in as the root user. Run the **poweroff** command when you're ready to finish this lab.

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**. Since menus are not always available, the Applications | System Tools | Virtual Machine Manager click sequence is not an acceptable answer.
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 that would shut down a system after installation is complete is **shutdown**.

Administration with the Secure Shell

11. The **ssh** command switch that is used to specify a different user for remote logins is **-l**.
12. The **ssh** command switch that enables access to remote GUI utilities is **-X**. The **-Y** switch is also an acceptable answer.

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

While there is nothing truly difficult about this lab, it should increase your confidence with VMs based on KVM. Once 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 directories, along with the space available. The following commands should confirm those directories 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 Network, 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 system as virtual guests."

Lab 2

One of the issues with system cloning is how it includes the hardware 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'll also need to assign a unique hardware address to the given network card. Because of such issues, KVM normally sets up a different network card for a cloned system. For example, if the original system had a eth0 network card with one hardware address, the cloned system would have an eth1 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 so is with the **virt-install** command and the **--prompt** switch. That command prompts for the following information:

- Allocated RAM, which should be at least 512MB (652MB for a GUI-based installation).
- The path to the virtual disk file, which should be the same as that virtual disk created in Lab 2.
- The size of the virtual disk file, if that file doesn't already exist.
- The URL for the FTP installation server created in Chapter 1, Lab 2. Alternatively, you could use the HTTP installation server also discussed in Chapter 1.

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 that 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, 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. Since 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 except the **-x** or **--extra-args=** needed to call the URL for the Kickstart file.

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 a 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

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

The lab also suggests one method for remotely accessing the console of 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.