**Labs**

During the Red Hat exams, the tasks will be presented electronically. Therefore, this book presents most of the labs electronically as well. For more information, see the "Lab Questions" section toward the end of Chapter 2. Lab 1 is presented in the Study Guide, page 109.

**Lab 2**

In this lab, you'll clone the system created in Lab 1. Use the techniques discussed in this chapter to clone that system. The process can be completed either at the command line with the **virt-clone** command or with the Virtual Machine Manager.

In addition, when rebooting the system, you'll want to set this system up as the clone1.example.org system on the 192.168.100.50 IP address. You can substitute a different IP address as long as it's on a different network from the server1.example.com system.

When a system is cloned, it carries everything from the previous system. So the first time you boot a cloned system, it's best to boot it into the rescue target, which does not start networking. For more information on systemd targets and network configuration, see Chapters 3 and 5.

**Lab 3**

Use the **virt-install** command to create a new system. Use the model described in Chapter 1 for the tester1.example.com system, with an IP address of 192.168.122.150.

**Lab 4**

In this lab, you'll modify the Kickstart file created on the server1.example.com VM in Lab 1. It's the anaconda-ks.cfg file in the root user's home directory. That file will be used to automate the installation of a system on a virtual machine. Unless you've already created it, the system will be **tester1.example.com** on IP address **192.168.122.150**. If you've set up the server1.example.com system on a different network, make sure the tester1.example.com system is on the same network. Remember the lessons of the chapter when modifying that file. Here are a few hints:

1. Change the **network** directive to set the new system to an appropriate IP address and hostname.

2. Make sure the partition directives are active (uncommented). Change them as needed to make sure a size is specified.
3. Configure the system to shut down after installation is complete.
4. Don't be afraid of some trial and error. If the installation stops during the process, check the messages in different virtual consoles (see point 11).
5. Copy the Kickstart file to the installation directory created in Chapter 1. Use the techniques discussed in that chapter to make sure the Kickstart file matches the SELinux context of other files in that directory and ensure it is readable by all users.
6. Create a new KVM-based VM, using the techniques discussed in the chapter.
7. Boot the system from the RHEL 7 network boot CD.
8. At the boot screen, highlight the Install Red Hat Enterprise Linux 7.0 option and press TAB.
9. Add the **ks** directive along with the URL of the network installation source created in Chapter 1.
10. Press ENTER. The installation should now proceed to completion, without intervention.
11. If the installation stops during the process, make notes on where it stopped. The screens available during the installation process can help. To access those screens, press CTRL-ALT-F3, CTRL-ALT-F4, and CTRL-ALT-F5. To return to the main installation screen, press ALT-F6 (ALT-F1 if you're using the text-based installation program).
12. Match the stopping point with the appropriate entry in the Kickstart configuration file.
13. Modify the Kickstart configuration file, and rerun the Kickstart-based installation.

**Lab 5**

In this lab, you'll modify the Kickstart file created on the **server1.example.com** VM in Labs 1 and 4, for use by the **virt-install** command. Unless you've already created it, the system will be **outsider1.example.org** on IP address **192.168.100.100**. (If that IP address network is in use by another component such as a cable modem, another IP address such as **192.168.101.100** is acceptable.)

**Lab 6**

In this lab, you'll test a Secure Shell client on any and all virtual machines that you've created. Hopefully, you'll remember the root administrative password configured during the installation process.

1. To see how the system works, first run the following command on the local system (if desired, substitute 127.0.0.1 for localhost):

```
# ssh localhost
```

2. Proceed with the login to the localhost system. Log out with the **exit** command.
3. Review the current known hosts for the system with the **cat ~/.ssh/known_hosts** command. The file may appear incomprehensible, but you'll see a line such as the following near the end of the file, which refers to the public RSA key from the localhost system:

```
localhost ecdsa-sha2-nistp256 AAAAE2VjZHNhL...
```

4. Repeat the process with a remote system such as **server1.example.com**, or the equivalent IP address such as **192.168.122.50**. After returning to the local system, what do you see added to the known_hosts file? (Direct connections to the **server1.example.com** system may not work until you've set up the **/etc/hosts** file as discussed in Chapter 3.)
5. This step requires access to the GUI on a local system and GUI applications on the remote system. If you've followed the instructions so far in Chapters 1 and 2, you'll have a number of systems that meet these requirements. With that in mind, log in to the noted remote system from a local GUI-based terminal with the following command (substitute an appropriate hostname or IP address as needed):

```
# ssh -X root@192.168.122.50
```

6. Now try to open a GUI application on the remote system, perhaps even the Firefox web browser (if that's installed) with the **firefox** command. To confirm success, look at the title bar of the window that appears. It should display a message with the location of the remote application, such as the following:

```
Welcome to Firefox - Mozilla Firefox (on server1.example.com)
```

7. Exit from the remote application and then exit from the remote system.

**Lab 7**

In this lab, you'll perform three tasks associated with one of the VMs created in previous labs:

- Start a VM from the command line.
- Stop a VM from the command line.
- Configure that VM from the command line.

You may recognize these tasks from the RHCSA objectives. The process should be fairly simple. First, take account of currently configured VMs with the following command:

```
# virsh list --all
```

From the VMs that appear in the output, pick one that isn't currently running. If the server1.example.com system is not running, start it. Confirm that it is currently running. Use the **ssh** command to access a remote console on that VM system.

Now from the command line of the host physical system, stop that virtual console. What command actually performs the task? Confirm the result with the **virsh list --all** command.

If you want to set up the server1.example.com system to start automatically during the boot process, run the following command:

```
# virsh autostart server1.example.com
```

To confirm the change, review the contents of the **/etc/libvirt/qemu/autostart** directory. Next, enter the VM system and run the **ip addr show** command to confirm the IP address of that VM's network card. If you've configured this particular **server1.example.com** system per the instructions discussed in Chapter 1, that IP address should be **192.168.122.50**.

Now power down any currently running VMs, and reboot the physical host system. When the system boots again, log in to the local host system. Don't start Virtual Machine Manager. Run the **ssh** commands described in Lab 6. If it works, then you should be able to connect to the VM as if it were a remote system.

Exit from the remote system and run the **virsh list --all** command. You should see output similar to the following:

```
 Id    Name                           State
----------------------------------------------------
 2     server1.example.com            running
```

Now power down the remote system. You can **ssh** again into the remote system and run the **poweroff** command directly from there. How do you reverse the process, so this system does not start the next time you reboot the physical host system?

**Lab 8**

In this lab, you'll use the commands described at the end of Chapter 2 to test connections to available services. If you've created the network installation servers described in Chapter 1, there will be at least FTP and HTTP servers active on those systems. The default ports for these services are 21 and 80, respectively. Try the **telnet localhost 21** command on a local system, where the vsFTP service is active. Look at the following output:

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

Now exit from the telnet session. Confirm the IP address of the local system with the **ip addr show dev virbr0** command. It should be an address such as 192.168.122.1. Log in to a remote VM such as tester1.example.org with a command such as **ssh root@192.168.122.150**. Now try the same command again from that remote system; for example, for the server1.example.com system on IP address 192.168.122.50, run the following command:

```
# telnet 192.168.122.50 21
```

Do you get a "connection refused" or a "no route to host" message? What do each of those messages mean? It's acceptable if you're not certain about how to address this issue now, as firewalls are not covered until Chapter 4.

Now try **nmap** on the local system with the following command:

```
# nmap localhost
```

From the tester1.example.com system, review what other systems on the local network can see from the following command. Pay attention to the differences. That will give you hints

on what services are blocked by firewalls. Those firewalls may go beyond what's configured with the **firewall-cmd** command discussed in Chapter 4.

```
# nmap 192.168.122.50
```