Red Hat Enterprise Linux 5

Virtualization Guide

The definitive guide for virtualization on Red Hat Enterprise Linux

Christopher Curran

Jan Mark Holzer

Red Hat Enterprise Linux 5 Virtualization Guide
The definitive guide for virtualization on Red Hat Enterprise Linux
Edition 4

Author	Christopher Curran	ccurran@redhat.com
Author	Jan Mark Holzer	

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588 Research Triangle Park, NC 27709 USA

The Red Hat Enterprise Linux Virtualization Guide contains information on installation, configuring, administering, tips, tricks and troubleshooting virtualization technologies used in Red Hat Enterprise Linux.

Preface

This book is the Red Hat Enterprise Linux Virtualization Guide. The Guide covers all aspects of using and managing virtualization products included with Red Hat Enterprise Linux.

1. About this book

This book is divided into 7 parts:

- System Requirements
- Installation
- Configuration
- Administration
- Reference
- Tips and Tricks
- Troubleshooting

2. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

2.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

[1] https://fedorahosted.org/liberation-fonts/

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**-**`Alt`**-**`F1`** to switch to the first virtual terminal. Press **`Ctrl`**-**`Alt`**-**`F7`** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`Mono-spaced Bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.

> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

2.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

2.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A Note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

3. We need feedback

If you find a typographical error in the *Virtualization Guide*, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *http://bugzilla.redhat.com/bugzilla/* against the component *Virtualization_Guide*.

If you have a suggestion for improving the documentation, try and be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

4. How should CIO's think about virtualization

by Lee Congdon, Chief Information Officer, Red Hat, Inc.

You may already be heavily invested in the rapidly emerging technology of virtualization. If so, consider some of the ideas below for further exploiting the technology. If not, now is the right time to get started.

Virtualization provides a set of tools for increasing flexibility and lowering costs, things that are important in every enterprise and Information Technology organization. Virtualization solutions are becoming increasingly available and rich in features.

Since virtualization can provide significant benefits to your organization in multiple areas, you should be establishing pilots, developing expertise and putting virtualization technology to work now.

Virtualization for Innovation

In essence, virtualization increases flexibility by decoupling an operating system and the services and applications supported by that system from a specific physical hardware platform. It allows the establishment of multiple virtual environments on a shared hardware platform.

Organizations looking to innovate find that the ability to create new systems and services without installing additional hardware (and to quickly tear down those systems and services when they are no longer needed) can be a significant boost to innovation.

Among possible approaches are the rapid establishment of development systems for the creation of custom software, the ability to quickly set up test environments, the capability to provision alternate software solutions and compare them without extensive hardware investments, support for rapid prototyping and agile development environments, and the ability to quickly establish new production services on demand.

These environments can be created in house or provisioned externally, as with Amazon's EC2 offering. Since the cost to create a new virtual environment can be very low, and can take advantage of existing hardware, innovation can be facilitated and accelerated with minimal investment.

Virtualization can also excel at supporting innovation through the use of virtual environments for training and learning. These services are ideal applications for virtualization technology. A student can start course work with a known, standard system environment. Class work can be isolated from the production network. Learners can establish unique software environments without demanding exclusive use of hardware resources.

As the capabilities of virtual environments continue to grow, we're likely to see increasing use of virtualization to enable portable environments tailored to the needs of a specific user. These environments can be moved dynamically to an accessible or local processing environment, regardless of where the user is located. The user's virtual environments can be stored on the network or carried on a portable memory device.

A related concept is the Appliance Operating System, an application package oriented operating system designed to run in a virtual environment. The package approach can yield lower development and support costs as well as insuring the application runs in a known, secure environment. An Appliance Operating System solution provides benefits to both application developers and the consumers of those applications.

How these applications of virtualization technology apply in your enterprise will vary. If you are already using the technology in more than one of the areas noted above, consider an additional investment in a solution requiring rapid development. If you haven't started with virtualization, start with a training and learning implementation to develop skills, then move on to application development and testing. Enterprises with broader experience in virtualization should consider implementing portable virtual environments or application appliances.

Virtualization for Cost Savings

Virtualization can also be used to lower costs. One obvious benefit comes from the consolidation of servers into a smaller set of more powerful hardware platforms running a collection of virtual environments. Not only can costs be reduced by reducing the amount of hardware and reducing the amount of unused capacity, but application performance can actually be improved since the virtual guests execute on more powerful hardware.

Further benefits include the ability to add hardware capacity in a non-disruptive manner and to dynamically migrate workloads to available resources.

Depending on the needs of your organization, it may be possible to create a virtual environment for disaster recovery. Introducing virtualization can significantly reduce the need to replicate identical hardware environments and can also enable testing of disaster scenarios at lower cost.

Virtualization provides an excellent solution for addressing peak or seasonal workloads. If you have complementary workloads in your organization, you can dynamically allocate resources to the applications which are currently experiencing the greatest demand. If you have peak workloads that you are currently provisioning inside your organization, you may be able to buy capacity on demand externally and implement it efficiently using virtual technology.

Cost savings from server consolidation can be compelling. If you are not exploiting virtualization for this purpose, you should start a program now. As you gain experience with virtualization, explore the benefits of workload balancing and virtualized disaster recovery environments.

Virtualization as a Standard Solution

Regardless of the specific needs of your enterprise, you should be investigating virtualization as part of your system and application portfolio as the technology is likely to become pervasive. We expect operating system vendors to include virtualization as a standard component, hardware vendors to build virtual capabilities into their platforms, and virtualization vendors to expand the scope of their offerings.

If you don't have plans to incorporate virtualization in your solution architecture, now is a very good time to identify a pilot project, allocate some underutilized hardware platforms, and develop expertise with this flexible and cost-effective technology. Then, extend your target architectures to incorporate virtual solutions. Although substantial benefits are available from virtualizing existing services, building new applications with an integrated virtualization strategy can yield further benefits in both manageability and availability.

You can learn more about Red Hat's virtualization solutions at *http://www.redhat.com/products/*

Part I. Requirements and Limitations for Virtualization with Red Hat Enterprise Linux

System requirements, support restrictions and limitations

These chapters outline the system requirements, support restrictions, and limitations of virtualization on Red Hat Enterprise Linux.

System requirements

This chapter lists system requirements for successfully running virtualization on Red Hat Enterprise Linux. You require a system running Red Hat Enterprise Linux 5 Server with the virtualization packages. The host a configured *hypervisor* (either Xen or KVM). For information on installing the hypervisor, read *Chapter 5, Installing the virtualization packages*.

Minimum system requirements

- 6GB free disk space
- 2GB of RAM.

Recommended system requirements

- 6GB plus the required disk space recommended by the guest operating system per guest. For most operating systems more than 6GB of disk space is recommended.
- One processor core or hyper-thread for each virtualized CPU and one for the hypervisor.
- 2GB of RAM plus additional RAM virtualized guests.

> **KVM overcommit**
>
> The KVM hypervisor supports guests using more than the available RAM and processor cores for virtualized guests. For information on safely overcommitting resources with KVM refer to *Section 28.4, “Overcommitting with KVM”*.

Xen para-virtualization requirements

Para-virtualized guests require a Red Hat Enterprise Linux 5 installation tree available over the network using the `NFS`, `FTP` or `HTTP` protocols.

Xen full virtualization requirements

Full virtualization with the Xen Hypervisor requires:

- an Intel processor with the Intel VT extensions,
- an AMD processor with the AMD-V extensions, or
- an Intel Itanium processor.

Refer to *Section 28.6, “Verifying virtualization extensions”* to determine if your processor has the virtualization extensions.

KVM requirements

The KVM hypervisor requires:

- an Intel processor with the Intel VT and the Intel 64 extensions, or
- an AMD processor with the AMD-V and the AMD64 extensions.

Refer to *Section 28.6, “Verifying virtualization extensions”* to determine if your processor has the virtualization extensions.

Storage support

The supported guest storage methods are:

- files on local storage,
- physical disk partitions,
- locally connected physical LUNs,
- LVM partitions, and
- iSCSI and Fibre Channel based LUNs.

File based guest storage

File based guest images should be stored in the **`/var/lib/xen/images/`** folder. If you use a different directory you must add the directory to the SELinux policy. Refer to *Section 11.1, “SELinux and virtualization”* for details.

Xen restrictions and support

Red Hat Enterprise Linux 5 supports various architecture combinations for hosts and virtualized guests. This lists tested compatible guests for Red Hat Enterprise Linux 5 hosts. Other combinations may be possible but are not tested and are unsupported by Red Hat.

The x86 architecture

The 32 bit Red Hat Enterprise Linux *kernel-xen* package on x86-compatible systems is limited to 16 processor cores.

Supported fully virtualized guests

Operating system	Support level
Red Hat Enterprise Linux 3 x86	Optimized
Red Hat Enterprise Linux 4 x86	Optimized
Red Hat Enterprise Linux 5 x86	Optimized
Windows Server 2000 32-Bit	Supported
Windows Server 2003 32-Bit	Supported
Windows XP 32-Bit	Supported
Windows Vista 32-Bit	Supported

Note, other 32 bit operating systems may work but are not tested and unsupported.

To utilize full virtualization on Red Hat Enterprise Linux 5 your processor must have an Intel-VT or AMD-V enabled processor.

Supported para-virtualized guest

Operating system	Support level
Red Hat Enterprise Linux 4 x86 Update 5 and higher	Optimized
Red Hat Enterprise Linux 5 x86	Optimized

To utilize para-virtualization on Red Hat Enterprise Linux 5 your processor must have the Physical Address Extension (PAE) instruction set.

AMD64 and Intel 64 architecture

The *kernel-xen* package on AMD64 and Intel 64 machines has the following processor limitations:

- Red Hat Enterprise Linux 5.0 supports up to 32 CPU processor cores.
- Red Hat Enterprise Linux 5.1 supports up to 32 CPU processor cores.
- Red Hat Enterprise Linux 5.2 supports up to 64 CPU processor cores.
- Red Hat Enterprise Linux 5.3 supports up to 126 CPU processor cores.
- Red Hat Enterprise Linux 5.4 supports up to 256 CPU processor cores.

Supported fully virtualized guests

Operating system	Support level
Red Hat Enterprise Linux 3 x86-64	Optimized
Red Hat Enterprise Linux 3 x86	Optimized
Red Hat Enterprise Linux 4 AMD64/Intel 64	Optimized
Red Hat Enterprise Linux 4 x86	Optimized
Red Hat Enterprise Linux 5 AMD64/Intel 64	Optimized
Red Hat Enterprise Linux 5 x86	Optimized
Windows Server 2000 32-Bit	Supported
Windows Server 2003 32-Bit	Supported
Windows XP 32-Bit	Supported
Windows Vista 32-Bit	Supported
Windows Vista 64-Bit	Supported
Windows Server 2008 32-Bit	Supported
Windows Server 2008 64-Bit	Supported
Solaris 32 bit	Supported

Other x86 and AMD64 or Intel 64 based operating systems may work but are not and unsupported.

Supported para-virtualized guest

Operating system	Support level
Red Hat Enterprise Linux 4 AMD64/Intel 64 Update 5 and higher	Optimized
Red Hat Enterprise Linux 4 x86 Update 5 and higher	Technology preview in 5.2 and 5.3. Fully supported in 5.4 and newer.
Red Hat Enterprise Linux 5 AMD64/Intel 64	Optimized
Red Hat Enterprise Linux 5 x86	Technology preview in 5.2 and 5.3. Fully supported in 5.4 and newer.

Intel Itanium architecture

The *kernel-xen* package on Itanium systems is limited to 32 processor cores.

Supported fully virtualized guests

Operating system	Support level
Red Hat Enterprise Linux 3 Itanium	Supported
Red Hat Enterprise Linux 4 Itanium	Optimized

Operating system	Support level
Red Hat Enterprise Linux 5 Itanium	Optimized
Windows Server 2003 for Itanium-based Systems	Supported

Supported para-virtualized guest

Operating system	Support level
Red Hat Enterprise Linux 5 Itanium	Optimized

Itanium® support

Virtualization with the Xen hypervisor on the Intel Itanium architecture requires the guest firmware image package, refer to *Installing the Xen hypervisor with* ***yum*** for more information.

KVM restrictions and support

Important

KVM only supports AMD64 and Intel 64 versions of Red Hat Enterprise Linux. Other architectures are not supported at this time.

The Red Hat Enterprise Linux *kvm* package is limited to 64 processor cores. This limit applies except on certain platforms where the limit is 96 processor cores.

Supported fully virtualized guests

Operating system	Support level
Red Hat Enterprise Linux 3 x86	Optimized with para-virtualized drivers
Red Hat Enterprise Linux 4 x86	Optimized with para-virtualized drivers
Red Hat Enterprise Linux 4 AMD 64 and Intel 64	Optimized with para-virtualized drivers
Red Hat Enterprise Linux 5 x86	Optimized with para-virtualized drivers
Red Hat Enterprise Linux 5 AMD 64 and Intel 64	Optimized with para-virtualized drivers
Windows Server 2003 R2 32-Bit	Optimized with para-virtualized drivers
Windows Server 2003 R2 64-Bit	Optimized with para-virtualized drivers
Windows Server 2003 Service Pack 2 32-Bit	Optimized with para-virtualized drivers
Windows Server 2003 Service Pack 2 64-Bit	Optimized with para-virtualized drivers
Windows XP 32-Bit	Optimized with para-virtualized drivers (network driver only)
Windows Vista 32-Bit	Supported
Windows Vista 64-Bit	Supported
Windows Server 2008 32-Bit	Optimized with para-virtualized drivers
Windows Server 2008 64-Bit	Supported

Virtualization limitations

This chapter covers the limitations of the virtualization packages.

This chapter covers additional limitations of virtualization technology in Red Hat Enterprise Linux.

4.1. General limitations for virtualization

Switching hypervisors

Presently, there are no applications for switching Xen-based guests to KVM or KVM-based guests to Xen. Guests can only be used on the hypervisor type that they were created on. There is an application in development at time of writing which may be released with future version of Red Hat Enterprise Linux

Other limitations

For the list of all other limitations and issues affecting virtualization read the *Red Hat Enterprise Linux Release Notes* for your version. *The Release Notes* cover the present new features, known issues and limitations as they are updated or discovered.

Test before deployment

You should test for the maximum anticipated load and virtualized network stress before deploying heavy I/O applications. Stress testing is important as there are performance drops caused by virtualization with increased I/O usage.

4.2. KVM limitations

The following limitations apply to the KVM hypervisor:

Constant TSC bit

Systems without a Constant Time Stamp Counter require additional configuration. Refer to *Chapter 17, KVM guest timing management* for details on determining whether you have a Constant Time Stamp Counter and configuration steps for fixing any related issues.

Memory overcommit

KVM supports memory overcommit and can store guests' memory in swap. A guest will run slower if it is swapped frequently. When KSM is used, make sure that the swap size is the size of the overcommit ratio.

CPU overcommit

It is not supported to have more than 10 virtual CPUs per physical processor core. Any number of overcommitted virtual CPUs above the number of physical processor cores may cause problems with certain virtualized guests.

Overcommitting CPUs has some risk and can lead to instability. Refer to *Section 28.4, “Overcommitting with KVM”* for tips and recommendations on overcommitting CPUs.

Virtualized SCSI devices

SCSI emulation is presently not supported. Virtualized SCSI devices are completely disabled in KVM.

Virtualized IDE devices

KVM is limited to a maximum of four virtualized (emulated) IDE devices per guest.

Para-virtualized devices

Para-virtualized devices, which use the **`virtio`** drivers, are PCI devices. Presently, guests are limited to a maximum of 32 PCI devices. Some PCI devices are critical for the guest to run and these devices cannot be removed. The default, required devices are:

- the host bridge,
- the ISA bridge and usb bridge (The usb and isa bridges are the same device),
- the graphics card (using either the Cirrus or qxl driver), and
- the memory balloon device.

Out of the 32 available PCI devices for a guest 4 are not removable. This means there are only 28 PCI slots available for additional devices per guest. Every para-virtualized network or block device uses one slot. Each guest can use up to 28 additional devices made up of any combination of para-virtualized network, para-virtualized disk devices, or other PCI devices using VTd.

Migration limitations

Live migration is only possible with CPUs from the same vendor (that is, Intel to Intel or AMD to AMD only).

The No eXecution (NX) bit in Must be set on or off for both CPUs for live migration.

4.3. Xen limitations

Note

All limitations in this chapter are limitations for Red Hat Enterprise Linux 5.4 except where noted. Older versions may have smaller limitations.

Xen host (dom0) limitations

- A limit of 100 block devices using the **`tap:aio`** driver and file-based devices per host. The total number of block devices attached to para-virtualized guests cannot exceed 100 devices per host.

Working around the para-virtualized device limit

There are two methods for working around the para-virtualized device limit: using phy devices or using LVM on the guest.

A host has no limit to the number of phy devices it can have if it has sufficient resources.

LVM, or similar logical partitioning tool, can be used on a block device to create additional logical partitions on a single para-virtualized block device.

Xen Para-virtualization limitations

- A maximum total of 256 devices per virtualized guest.
- A maximum of 15 network devices per virtualized guest.

Xen full virtualization limitations

- A maximum of four virtualized (emulated) IDE devices per guest.

  Devices using the para-virtualized drivers for fully-virtualized guests do not have this limitation.

- Virtualized (emulated) IDE devices are limited by the total number of loopback devices supported by the system. The default number of available loopback devices on Red Hat Enterprise Linux 5.4 is 8. That is all virtualized guests on the system can have no more than 8 virtualized (emulated) IDE devices.

  For more information on loopback devices, refer to *Red Hat KnowledgeBase*[1].

  Using more than 8 loopback devices

  By default, Red Hat Enterprise Linux limits the number of loopback devices available. This number can be raised by modifying the kernel limit.

  In the **/etc/modprobe.conf** add the following line:

  ```
  options loop max_loop=64
  ```

  Reboot the machine or run the following commands to update the kernel:

  ```
  # rmmod loop
        # modprobe loop
  ```

- A limit of 100 para-virtualized block devices per host. The total number of block devices (using the **tap:aio** driver) attached to para-virtualized guests cannot exceed 100 devices.
- A maximum of 256 block devices using the para-virtualized drivers per virtualized guest.
- A maximum of 15 network devices per virtualized guest.
- A maximum of 15 virtualized SCSI devices per virtualized guest.

4.4. Application limitations

There are aspects of virtualization which make virtualization unsuitable for certain types of applications.

Applications with high I/O throughput requirements should use the para-virtualized drivers for fully virtualized guests. Without the para-virtualized drivers certain applications may be unstable under heavy I/O loads.

The following applications should be avoided for their high I/O requirement reasons:

- **kdump** server

- **netdump** server

You should carefully evaluate databasing applications before running them on a virtualized guest. Databases generally use network and storage I/O devices intensively. These applications may not be suitable for a fully virtualized environment. Consider para-virtualization or para-virtualized drivers for increased I/O performance. Refer to *Chapter 14, Xen Para-virtualized Drivers* for more information on the para-virtualized drivers for fully virtualized guests.

Other applications and tools which heavily utilize I/O or require real-time performance should be evaluated carefully. Using full virtualization with the para-virtualized drivers (refer to *Chapter 14, Xen Para-virtualized Drivers*) or para-virtualization results in better performance with I/O intensive applications. Applications still suffer a small performance loss from running in virtualized environments. The performance benefits of virtualization through consolidating to newer and faster hardware should be evaluated against the potential application performance issues associated with using fully virtualized hardware.

Part II. Installation

Virtualization installation topics

These chapters describe setting up the host and installing virtualized guests with Red Hat Enterprise Linux. It is recommended to read these chapters carefully to ensure successful installation of virtualized guest operating systems.

Installing the virtualization packages

Before you can use virtualization, the virtualization packages must be installed on Red Hat Enterprise Linux. Virtualization packages can be installed either during the installation sequence or after installation using the **`yum`** command and the Red Hat Network (RHN).

You can install both the KVM and Xen hypervisors on a single system. The Xen hypervisor uses the *kernel-xen* package and the KVM hypervisor uses the default Red Hat Enterprise Linux kernel with the *kvm* kernel module. Xen and KVM each use a different kernel and only one hypervisor can be active at any given time. Red Hat recommends to only install one hypervisor, the hypervisor you want to use for virtualization.

To change hypervisor from Xen to KVM or KVM to Xen refer to *Section 28.2, “Changing between the KVM and Xen hypervisors”*.

5.1. Installing Xen with a new Red Hat Enterprise Linux installation

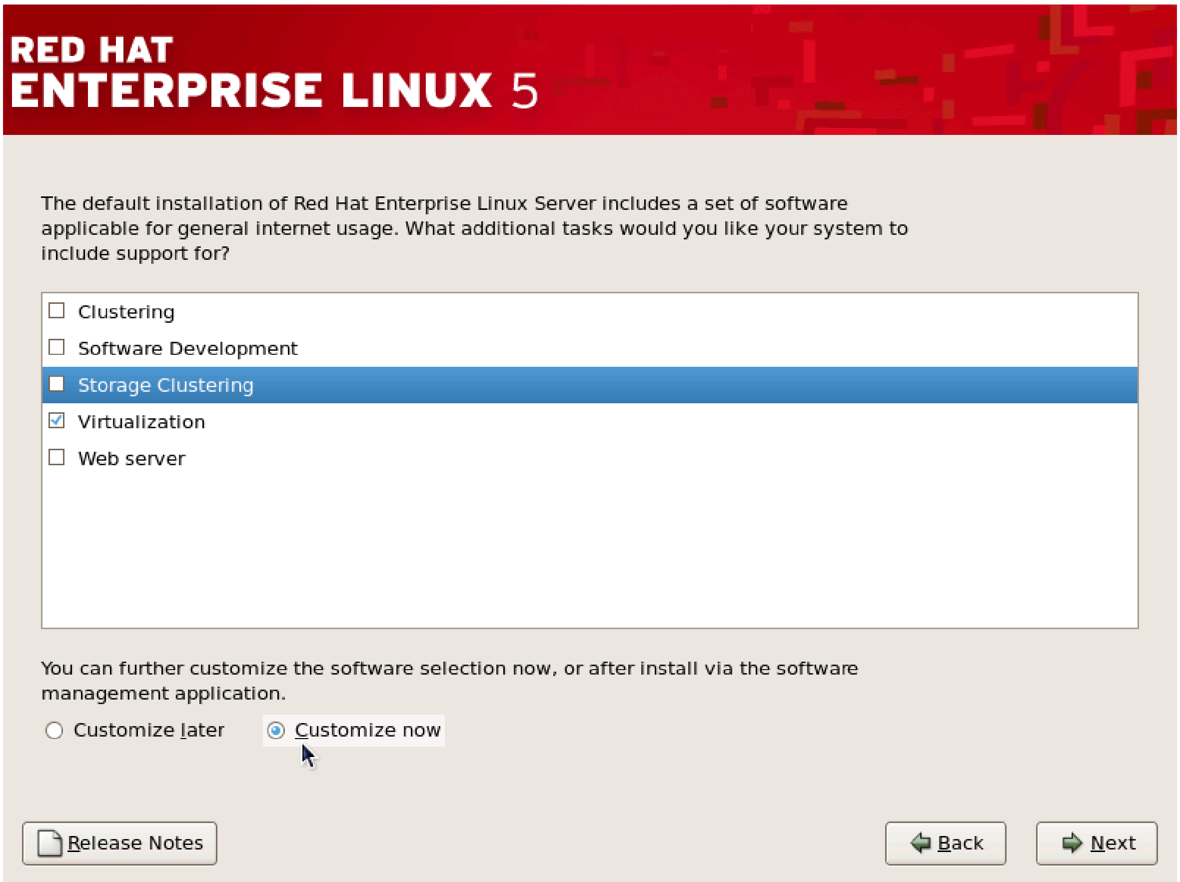
This section covers installing virtualization tools and Xen packages as part of a fresh Red Hat Enterprise Linux installation.

> **Need help installing?**
>
> The *Installation Guide* (available from *redhat.com*[1]) covers installing Red Hat Enterprise Linux in detail.

1. Start an interactive Red Hat Enterprise Linux installation from the Red Hat Enterprise Linux Installation CD-ROM, DVD or PXE.

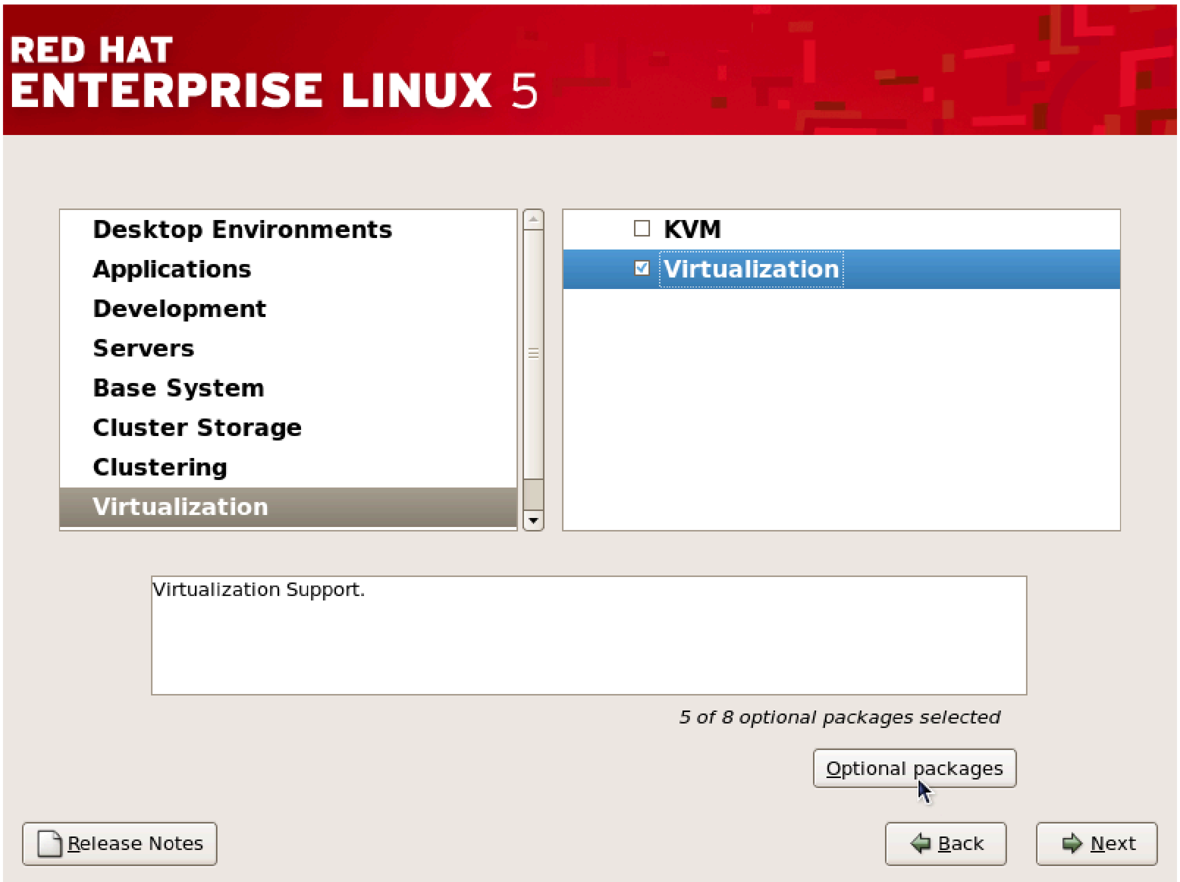
2. You must enter a valid installation number when prompted to receive access to the virtualization and other Advanced Platform packages.

3. Complete the other steps up to the package selection step.



Select the **Virtualization** package group and the **Customize Now** radio button.

4. Select the **Virtualization** package group. This selects the Xen hypervisor, **`virt-manager`**, **`libvirt`** and **`virt-viewer`** for installation.



5. **Customize the packages (if required)**
   Customize the **Virtualization** group if you require other virtualization packages.

   Press **Close** followed by **Next** to continue the installation.

Note

You require a valid RHN virtualization entitlement to receive updates for the virtualization packages.

Installing Xen packages with Kickstart files

This section describes how to use a Kickstart file to install Red Hat Enterprise Linux with the Xen hypervisor packages. Kickstart files allow for large, automated installations without a user manually installing each individual system. The steps in this section will assist you in creating and using a Kickstart file to install Red Hat Enterprise Linux with the virtualization packages.

In the `%packages` section of your Kickstart file, append the following package group:

```
%packages
@virtualization
```

For Intel Itanium systems

Fully virtualized guests on the Itanium® architecture require the guest firmware image package (*xen-ia64-guest-firmware*). Append this package to your kickstart file.

```
xen-ia64-guest-firmware
```

More information on Kickstart files can be found on Red Hat's website, *redhat.com*[2], in the *Installation Guide*.

5.2. Installing Xen packages on an existing Red Hat Enterprise Linux system

The section describes the steps necessary to install the virtualization packages on a working Red Hat Enterprise Linux system.

Adding packages to your list of Red Hat Network entitlements

This section describes how to enable Red Hat Network (RHN) entitlements for the virtualization packages. You need these entitlements enabled to install and update the virtualization packages on Red Hat Enterprise Linux. You require a valid Red Hat Network account in order to install virtualization packages on Red Hat Enterprise Linux.

In addition, your machines must be registered with RHN. To register an unregistered installation of Red Hat Enterprise Linux, run the **`rhn_register`** command and follow the prompts.

If you do not have a valid Red Hat subscription, visit the *Red Hat online store*[3].

Procedure 5.1. Adding the Virtualization entitlement with RHN

1. Log in to *RHN*[4] using your RHN username and password.

2. Select the systems you want to install virtualization on.

3. In the **System Properties** section the present systems entitlements are listed next to the **Entitlements** header. Use the **(Edit These Properties)** link to change your entitlements.

4. Select the **Virtualization** checkbox.

Your system is now entitled to receive the virtualization packages. The next section covers installing these packages.

Installing the Xen hypervisor with `yum`

To use virtualization on Red Hat Enterprise Linux you need the **`xen`** and **`kernel-xen`** packages. The **`xen`** package contains the hypervisor and basic virtualization tools. The **`kernel-xen`** package contains a modified Linux kernel which runs as a virtual machine guest on the hypervisor.

To install the **`xen`** and **`kernel-xen`** packages, run:

[2] http://www.redhat.com/docs/manuals/enterprise/
[3] https://www.redhat.com/wapps/store/catalog.html

```
# yum install xen kernel-xen
```

Fully virtualized guests on the Itanium® architecture require the guest firmware image package (**xen-ia64-guest-firmware**) from the supplementary installation DVD. This package can also be can be installed from RHN with the **yum** command:

```
# yum install xen-ia64-guest-firmware
```

It is advised to install additional virtualization packages for management and configuration. *Recommended virtualization packages:* lists the recommended packages.

Recommended virtualization packages:

python-virtinst
: Provides the **virt-install** command for creating virtual machines.

libvirt
: **libvirt** is an API library for interacting with hypervisors. **libvirt** uses the **xm** virtualization framework and the **virsh** command line tool to manage and control virtual machines.

libvirt-python
: The libvirt-python package contains a module that permits applications written in the Python programming language to use the interface supplied by the **libvirt** API.

virt-manager
: **virt-manager**, also known as **Virtual Machine Manager**, provides a graphical tool for administering virtual machines. It uses **libvirt** library as the management API.

Install the other recommended virtualization packages:

```
# yum install virt-manager libvirt libvirt-python python-virtinst
```

5.3. Installing KVM with a new Red Hat Enterprise Linux installation

This section covers installing virtualization tools and KVM package as part of a fresh Red Hat Enterprise Linux installation.

Need help installing?

The *Installation Guide* (available from *redhat.com*[5]) covers installing Red Hat Enterprise Linux in detail.

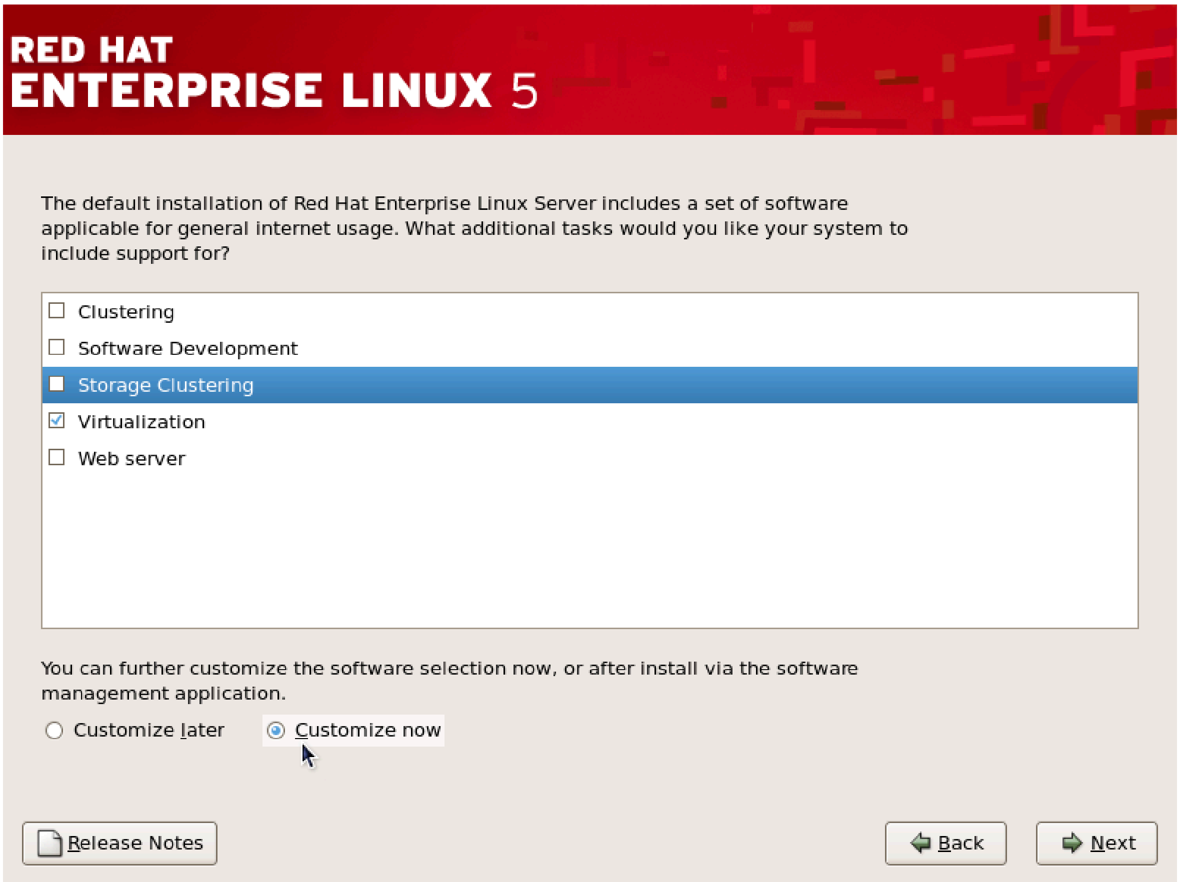
You need a valid installation number

You cannot select the virtualization packages during the installation without a valid installation number.

1. Start an interactive Red Hat Enterprise Linux installation from the Red Hat Enterprise Linux Installation CD-ROM, DVD or PXE.

2. You must enter a valid installation number when prompted to receive access to the virtualization and other Advanced Platform packages.

3. Complete the other steps up to the package selection step.



Select the **Virtualization** package group and the **Customize Now** radio button.

4. Select the **KVM** package group. Deselect the **Virtualization** package group. This selects the KVM hypervisor, **`virt-manager`**, **`libvirt`** and **`virt-viewer`** for installation.

5. **Customize the packages (if required)**
   Customize the **Virtualization** group if you require other virtualization packages.

Press **Close** followed by **Next** to continue the installation.

> **Note**
> You require a valid RHN virtualization entitlement to receive updates for the virtualization packages.

Installing KVM packages with Kickstart files

This section describes how to use a Kickstart file to install Red Hat Enterprise Linux with the KVM hypervisor packages. Kickstart files allow for large, automated installations without a user manually installing each individual system. The steps in this section will assist you in creating and using a Kickstart file to install Red Hat Enterprise Linux with the virtualization packages.

In the `%packages` section of your Kickstart file, append the following package group:

```
%packages
@kvm
```

More information on Kickstart files can be found on Red Hat's website, *redhat.com*[6], in the *Installation Guide*.

5.4. Installing KVM packages on an existing Red Hat Enterprise Linux system

The section describes the steps for installing the KVM hypervisor on a working Red Hat Enterprise Linux 5.4 or newer.

Adding packages to your list of Red Hat Network entitlements

This section describes how to enable Red Hat Network (RHN) entitlements for the virtualization packages. You need these entitlements enabled to install and update the virtualization packages on Red Hat Enterprise Linux. You require a valid Red Hat Network account in order to install virtualization packages on Red Hat Enterprise Linux.

In addition, your machines must be registered with RHN. To register an unregistered installation of Red Hat Enterprise Linux, run the **rhn_register** command and follow the prompts.

If you do not have a valid Red Hat subscription, visit the *Red Hat online store*[7].

Procedure 5.2. Adding the Virtualization entitlement with RHN

1. Log in to *RHN*[8] using your RHN username and password.
2. Select the systems you want to install virtualization on.
3. In the **System Properties** section the present systems entitlements are listed next to the **Entitlements** header. Use the **(Edit These Properties)** link to change your entitlements.
4. Select the **Virtualization** checkbox.

Your system is now entitled to receive the virtualization packages. The next section covers installing these packages.

Installing the KVM hypervisor with yum

To use virtualization on Red Hat Enterprise Linux you require the **kvm** package. The **kvm** package contains the KVM kernel module providing the KVM hypervisor on the default Red Hat Enterprise Linux kernel.

To install the **kvm** package, run:

```
# yum install kvm
```

Now, install additional virtualization management packages.

Recommended virtualization packages:

python-virtinst
: Provides the **virt-install** command for creating virtual machines.

[6] http://www.redhat.com/docs/manuals/enterprise/
[7] https://www.redhat.com/wapps/store/catalog.html

`libvirt`
: **`libvirt`** is an API library for interacting with hypervisors. **`libvirt`** uses the **`xm`** virtualization framework and the **`virsh`** command line tool to manage and control virtual machines.

`libvirt-python`
: The libvirt-python package contains a module that permits applications written in the Python programming language to use the interface supplied by the **`libvirt`** API.

`virt-manager`
: **`virt-manager`**, also known as **Virtual Machine Manager**, provides a graphical tool for administering virtual machines. It uses **`libvirt`** library as the management API.

Install the other recommended virtualization packages:

```
# yum install virt-manager libvirt libvirt-python python-virtinst
```

Virtualized guest installation overview

After you have installed the virtualization packages on the host system you can create guest operating systems. This chapter describes the general processes for installing guest operating systems on virtual machines. You can create guests using the **New** button in **virt-manager** or use the command line interface **`virt-install`**. Both methods are covered by this chapter.

Detailed installation instructions are available for specific versions of Red Hat Enterprise Linux, other Linux distributions, Solaris and Windows. Refer to *Chapter 7, Guest operating system installation procedures* for those procedures.

6.1. Creating guests with virt-install

You can use the **`virt-install`** command to create virtualized guests from the command line. **`virt-install`** is used either interactively or as part of a script to automate the creation of virtual machines. Using **`virt-install`** with Kickstart files allows for unattended installation of virtual machines.

The **`virt-install`** tool provides a number of options one can pass on the command line. To see a complete list of options run:

```
$ virt-install --help
```

The **`virt-install`** man page also documents each command option and important variables.

`qemu-img` is a related command which may be used before **`virt-install`** to configure storage options.

An important option is the `--vnc` option which opens a graphical window for the guest's installation.

This example creates a Red Hat Enterprise Linux 3 guest, named *`rhel3support`*, from a CD-ROM, with virtual networking and with a 5 GB file-based block device image. This example uses the KVM hypervisor.

```
# virt-install --accelerate --hvm --connect qemu:///system \
 --network network:default \
 --name rhel3support --ram=756\
 --file=/var/lib/libvirt/images/rhel3support.img \
 --file-size=6 --vnc --cdrom=/dev/sr0
```

Example 6.1. Using virt-install with KVM to create a Red Hat Enterprise Linux 3 guest

```
# virt-install --name fedora11 --ram 512 --file=/var/lib/libvirt/images/
fedora11.img \
 --file-size=3 --vnc --cdrom=/var/lib/libvirt/images/fedora11.iso
```

Example 6.2. Using virt-install to create a fedora 11 guest

6.2. Creating guests with virt-manager

`virt-manager`, also known as Virtual Machine Manager, is a graphical tool for creating and managing virtualized guests.

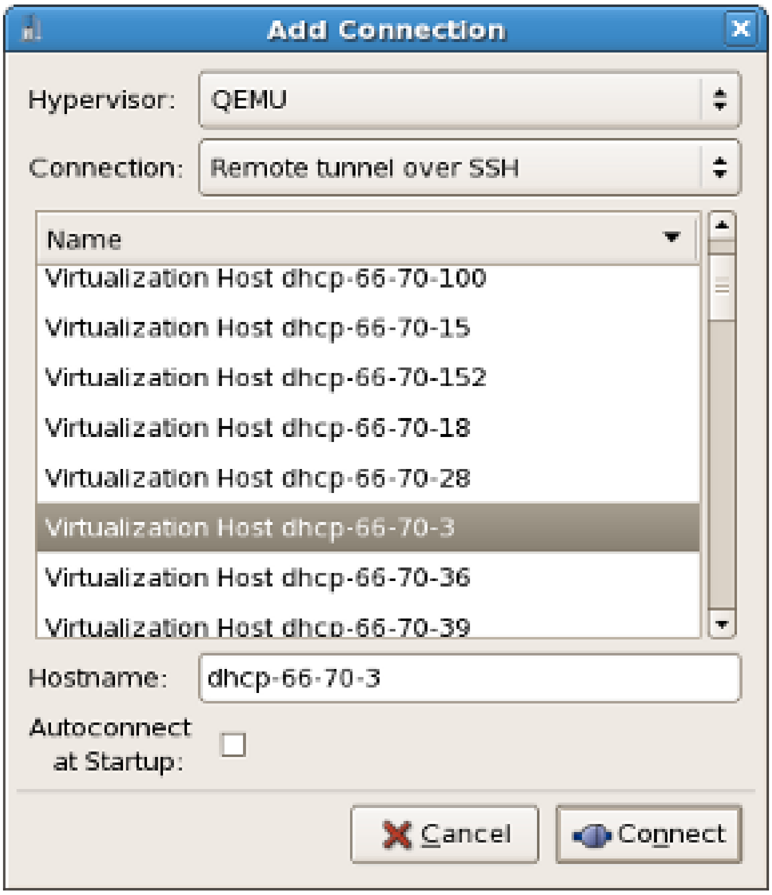
Procedure 6.1. Creating a virtualized guest with **virt-manager**

1. To start **virt-manager** run the following command as root:

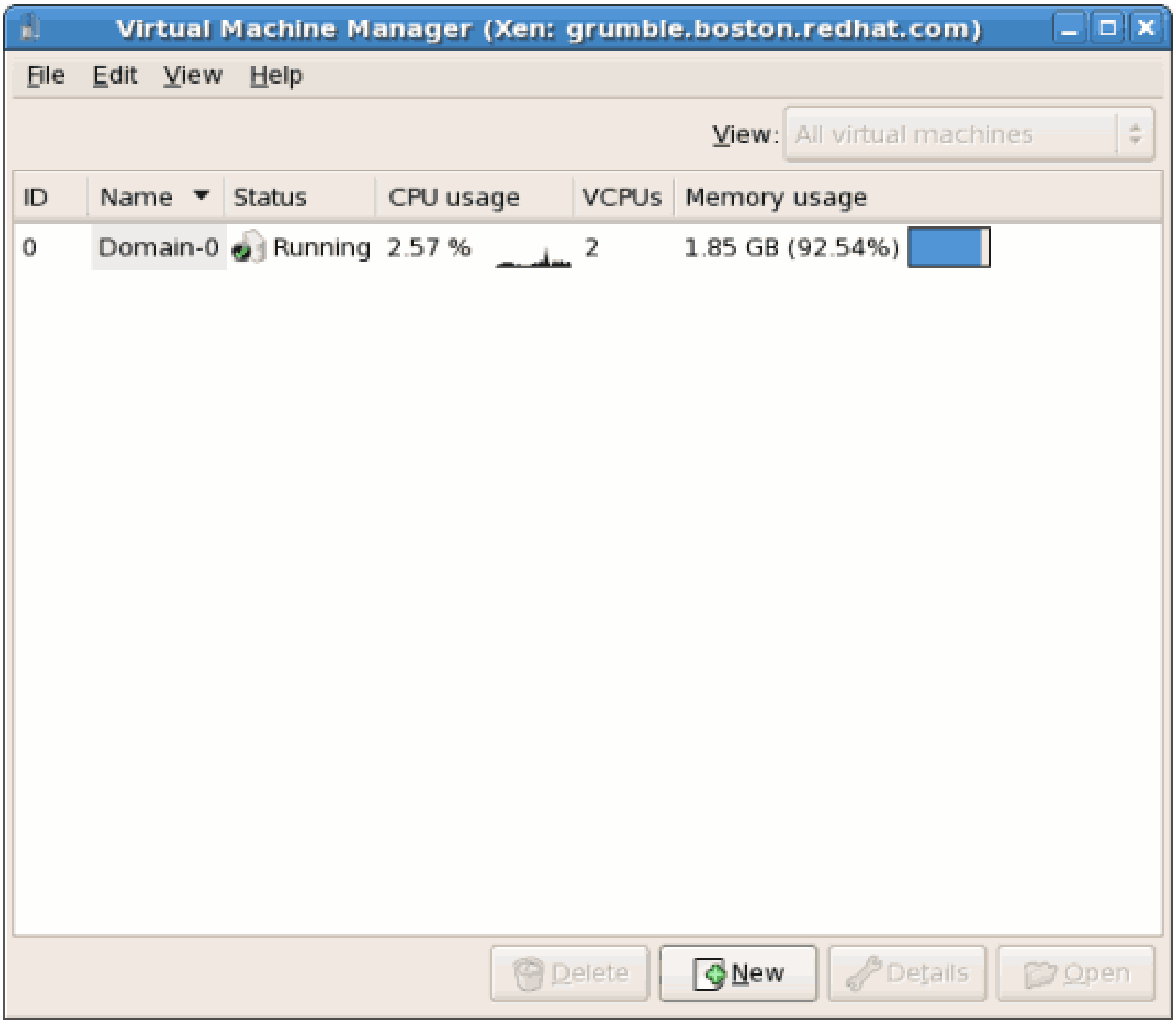
```
# virt-manager &
```

   The **`virt-manager`** command opens a graphical user interface window. Various functions are not available to users without root privileges or **sudo** configured, including the **New** button and you will not be able to create a new virtualized guest.

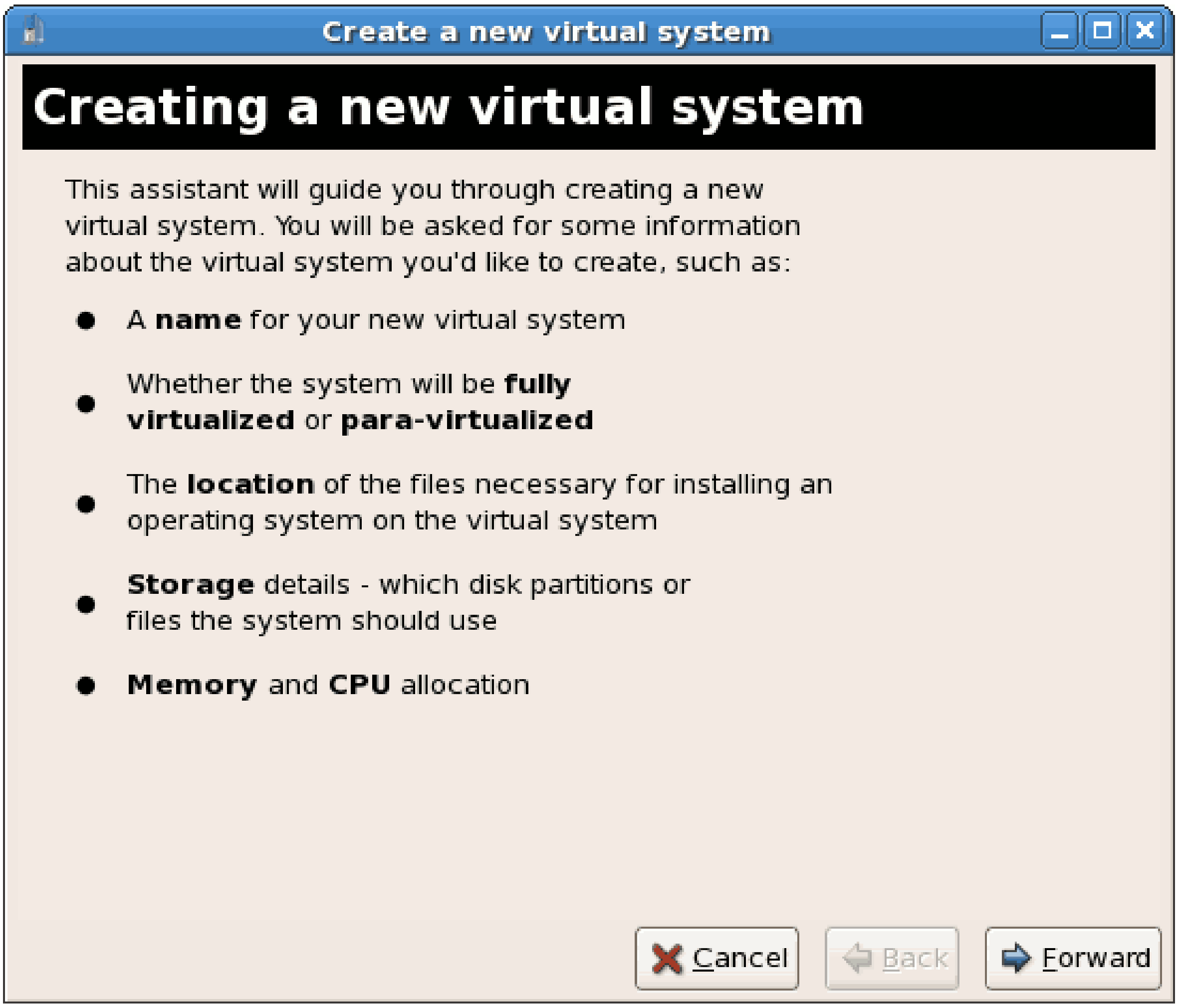
2. Open the **File -> Open Connection**. The dialog box below appears. . Select a hypervisor and click the **Connect** button:



3. The **virt-manager** window allows you to create a new virtual machine. Click the **New** button to create a new guest. This opens the wizard shown in the screenshot.



4. The **Create a new virtual system** window provides a summary of the information you must provide in order to create a virtual machine:

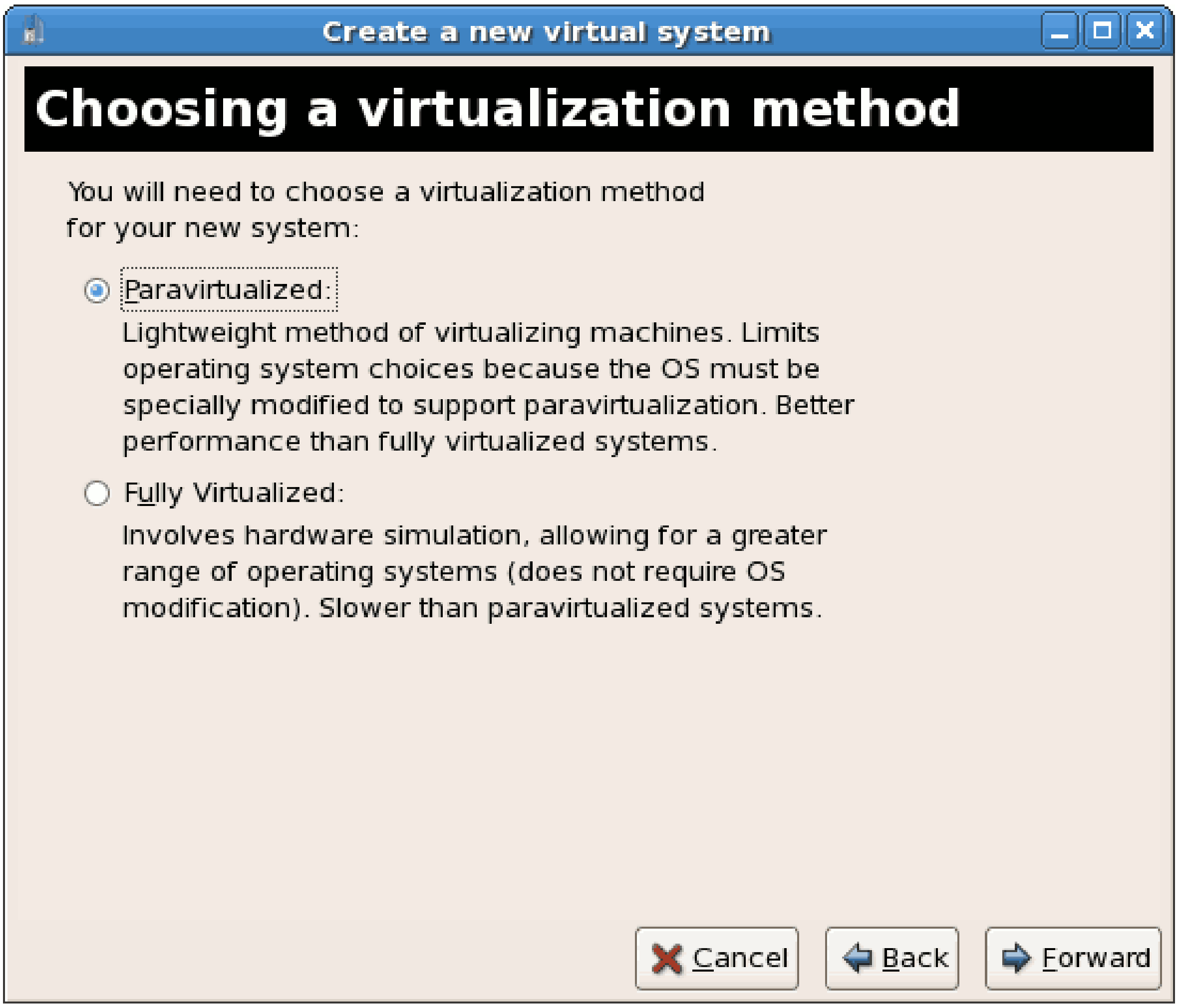


Review the information for your installation and click the **Forward** button.

5. The **Choosing a virtualization method** window appears. Choose between **Para-virtualized** or **Fully virtualized**.

   Full virtualization requires a system with Intel® VT or AMD-V processor. If the virtualization extensions are not present the **fully virtualized** radio button or the **Enable kernel/hardware acceleration** will not be selectable. The **Para-virtualized** option will be grayed out if **`kernel-xen`** is not the kernel running presently.

   If you connected to a KVM hypervisor only full virtualization is available.



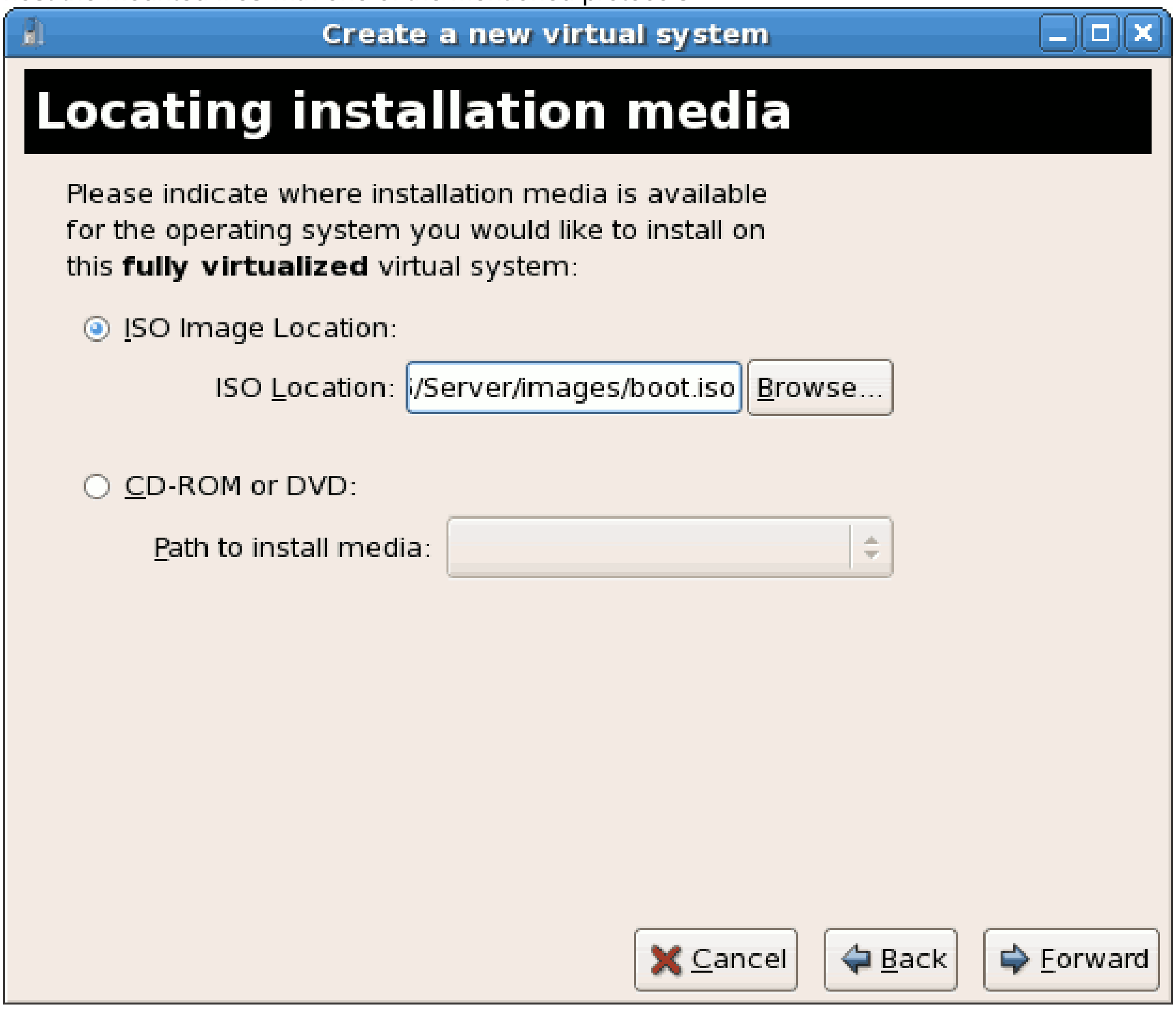
   Choose the virtualization type and click the **Next** button.

6. The **Locating installation media** prompt asks for the installation media for the type of installation you selected. This screen is dependent on what was selected in the previous step.

   a. The para-virtualized installation requires an installation tree accessible using one of the following network protocols: HTTP, FTP or NFS. The installation media URL must contain a Red Hat Enterprise Linux installation tree. This tree is hosted using NFS, FTP or HTTP. The network services and files can be hosted using network services on the host or another mirror.

      Using a CD-ROM or DVD image (tagged as an `.iso` file), mount the CD-ROM image and host the mounted files with one of the mentioned protocols.

   b. A fully virtualized guest installation require bootable installation DVDs, CD-ROMs or images of bootable installation DVDs or CD-ROMs (with the .iso or .img file type) locally. Windows installations use DVD, CD-ROM or .iso file. Many Linux and UNIX-like operating systems use an .iso file to install a base system before finishing the installation with a network based installation tree.



After selecting the appropriate installation media, click the **Forward** button.

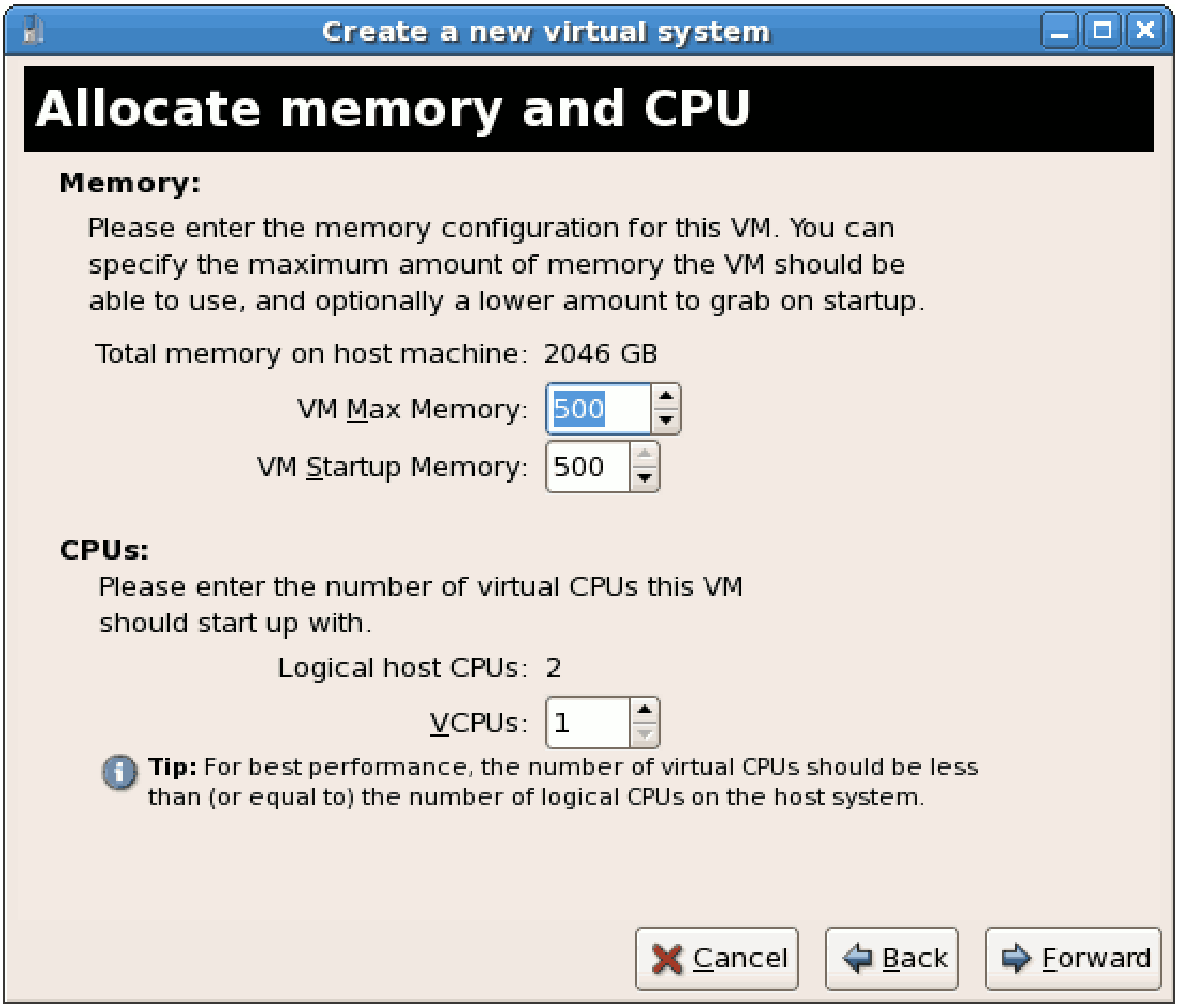
7. The **Assigning storage space** window displays. Choose a disk partition, LUN or create a file based image for the guest storage.

   The convention for file based images in Red Hat Enterprise Linux 5 all file based guest images are in the **`/var/lib/xen/images/`** directory. Other directory locations for file based images are prohibited by SELinux. If you run SELinux in enforcing mode, refer to *Section 11.1, “SELinux and virtualization”* for more information on installing guests.

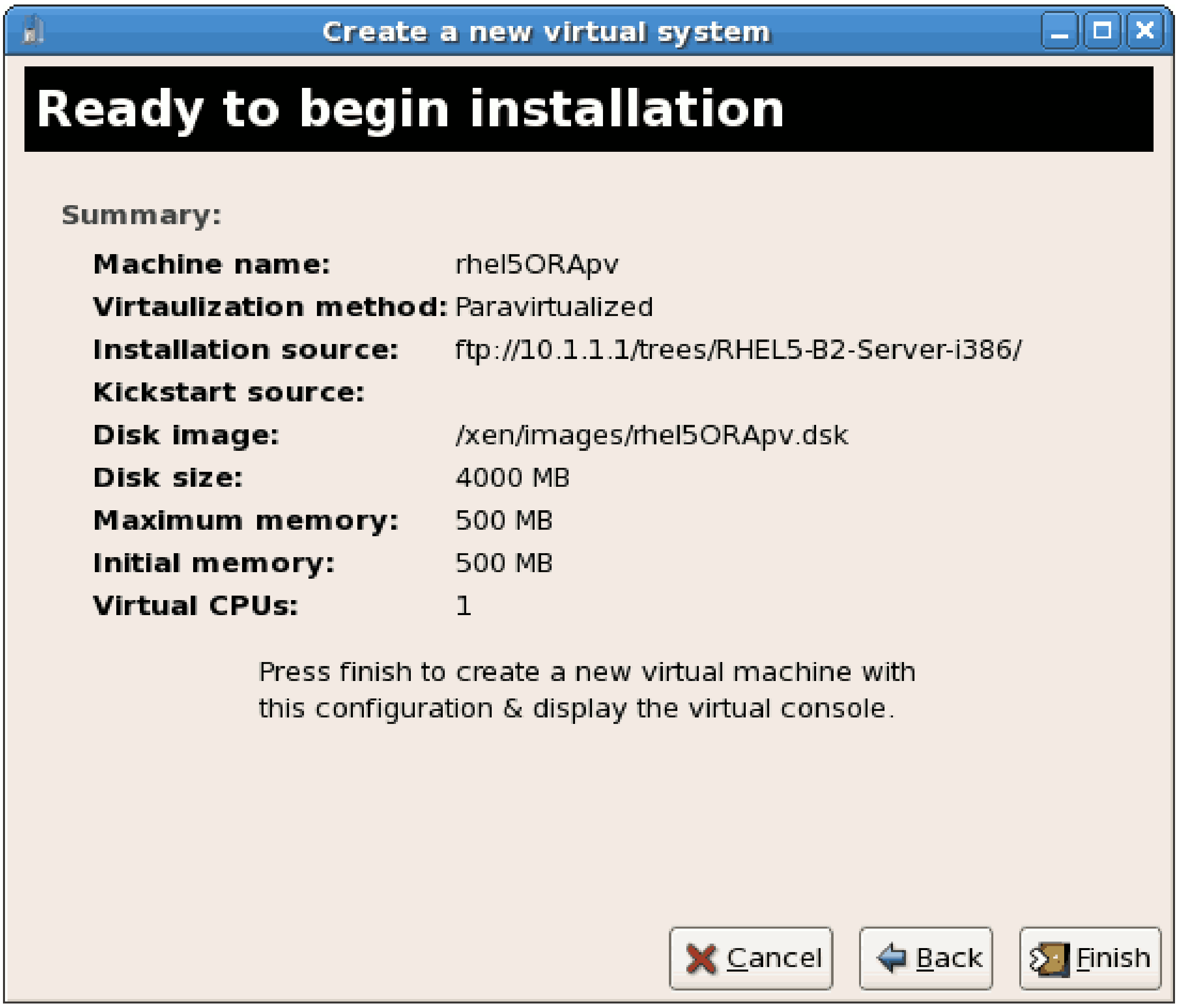
8. The Allocate memory and CPU window displays. Choose appropriate values for the virtualized CPUs and RAM allocation. These values affect the host's and guest's performance.

   Guests require sufficient physical memory (RAM) to run efficiently and effectively. Choose a memory value which suits your guest operating system and application requirements. Most operating system require at least 512MB of RAM to work responsively. Remember, guests use physical RAM. Running too many guests or leaving insufficient memory for the host system results in significant usage of virtual memory. Virtual memory is significantly slower causing degraded system performance and responsiveness. Ensure to allocate sufficient memory for all guests and the host to operate effectively.

   Assign sufficient virtual CPUs for the virtualized guest. If the guest runs a multithreaded application assign the number of virtualized CPUs it requires to run most efficiently. Do not assign more virtual CPUs than there are physical processors (or hyper-threads) available on the host system. It is possible to over allocate virtual processors, however, over allocating has a significant, negative affect on guest and host performance due to processor context switching overheads.



9. The ready to begin installation window presents a summary of all configuration information you entered. Review the information presented and use the **Back** button to make changes, if necessary. Once you are satisfied click the **Finish** button and to start the installation process.



A VNC window opens showing the start of the guest operating system installation process.

This concludes the general process for creating guests with `virt-manager`. *Chapter 7, Guest operating system installation procedures* contains step-by-step instructions to installing a variety of common operating systems.

6.3. Installing guests with PXE

This section covers the steps required to install guests with PXE. PXE guest installation requires a shared network device, also known as a network bridge. The procedures below cover creating a bridge and the steps required to utilize it the bridge for a PXE installation.

1. **Create a new bridge**
   a. Create a new network script file in the **/etc/sysconfig/network-scripts/** directory. This example creates a file named **ifcfg-installation** which makes a bridge named *installation*

   ```
   # cd /etc/sysconfig/network-scripts/
   # vim ifcfg-installation
   DEVICE=installation
   TYPE=Bridge
   BOOTPROTO=dhcp
   ONBOOT=yes
   ```

   > **Warning**
   > The line, *TYPE=Bridge*, is case-sensitive. It must have uppercase 'B' and lower case 'ridge'.

   b.
   ```
   Start the new bridge.
   # ifup installation
   ```

   c. There are no interfaces added to the new bridge yet. Use the **brctl show** command to view details about network bridges on the system.

   ```
   # brctl show
   bridge name     bridge id               STP enabled     interfaces
   installation    8000.000000000000       no
   virbr0          8000.000000000000       yes
   ```

   The **virbr0** bridge is the default bridge used by **libvirt** for Network Address Translation (NAT) on the default Ethernet device.

2. **Add an interface to the new bridge**
   Edit the configuration file for the interface. Add the **BRIDGE** parameter to the configuration file with the name of the bridge created in the previous steps.

```
# Intel Corporation Gigabit Network Connection
DEVICE=eth1
BRIDGE=installation
BOOTPROTO=dhcp
HWADDR=00:13:20:F7:6E:8E
ONBOOT=yes
```

   After editing the configuration file, restart networking or reboot.

```
# service network restart
```

   Verify the interface is attached with the **brctl show** command:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
installation    8000.001320f76e8e       no              eth1
virbr0          8000.000000000000       yes
```

3. **Security configuration**
   Configure **iptables** to allow all traffic to be forwarded across the bridge.

```
# iptables -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
# service iptables save
# service iptables restart
```

   Disable iptables on bridges

   Alternatively, prevent bridged traffic from being processed by **iptables** rules. In **/etc/sysctl.conf** append the following lines:

```
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
```

   Reload the kernel parameters configured with **sysctl**

```
# sysctl -p /etc/sysctl.conf
```

4. **Restart libvirt before the installation**
   Restart the **libvirt** daemon.

```
# service libvirtd reload
```

The bridge is configured, you can now begin an installation.

PXE installation with virt-install

For **`virt-install`** append the **`--network=bridge:BRIDGENAME`** installation parameter where installation is the name of your bridge. For PXE installations use the *`--pxe`* parameter.

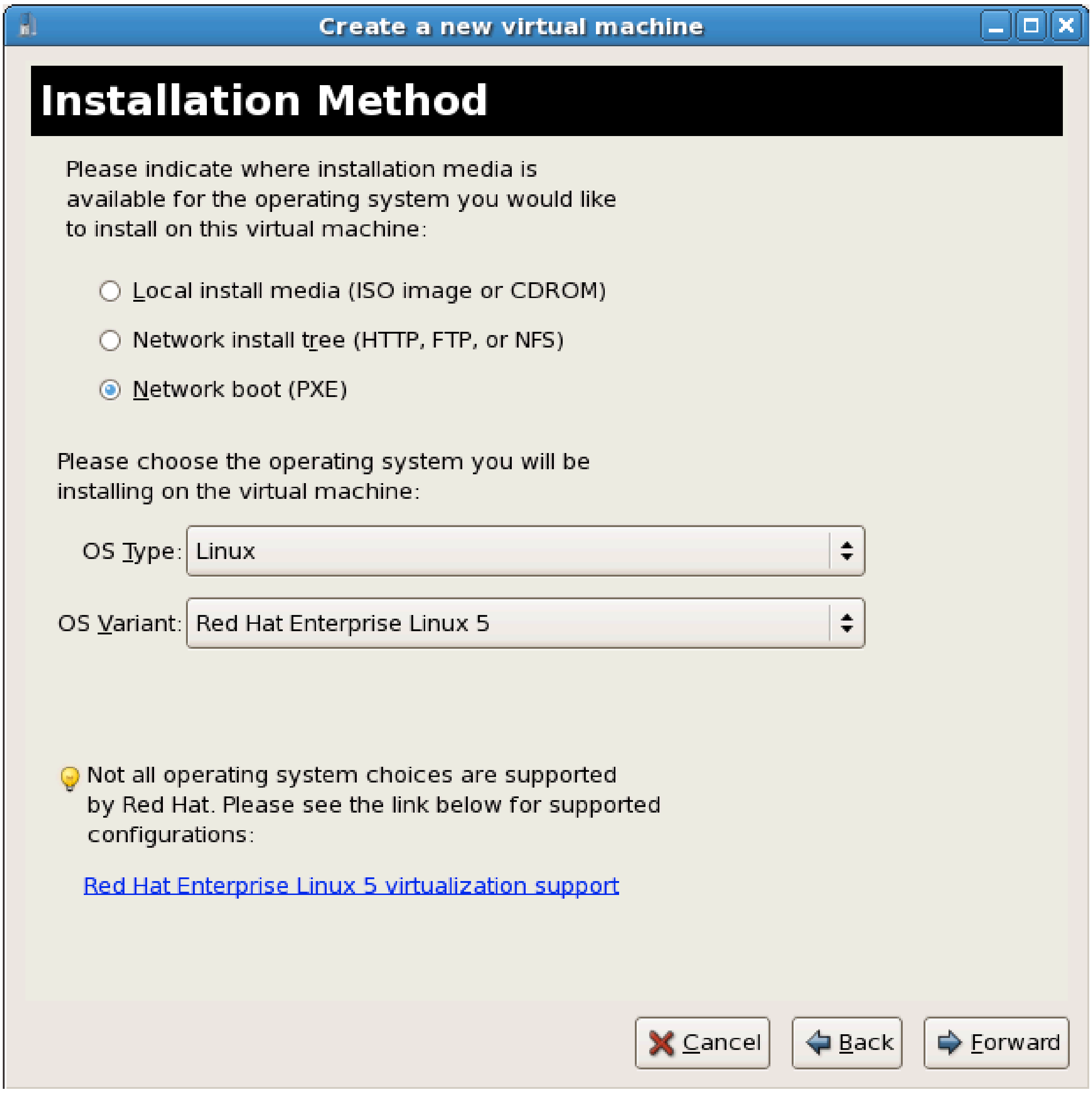
```
# virt-install --accelerate --hvm --connect qemu:///system \
    --network=bridge:installation --pxe\
    --name EL10 --ram=756 \
    --vcpus=4
    --os-type=linux --os-variant=rhel5
    --file=/var/lib/libvirt/images/EL10.img \
```

Example 6.3. PXE installation with virt-install

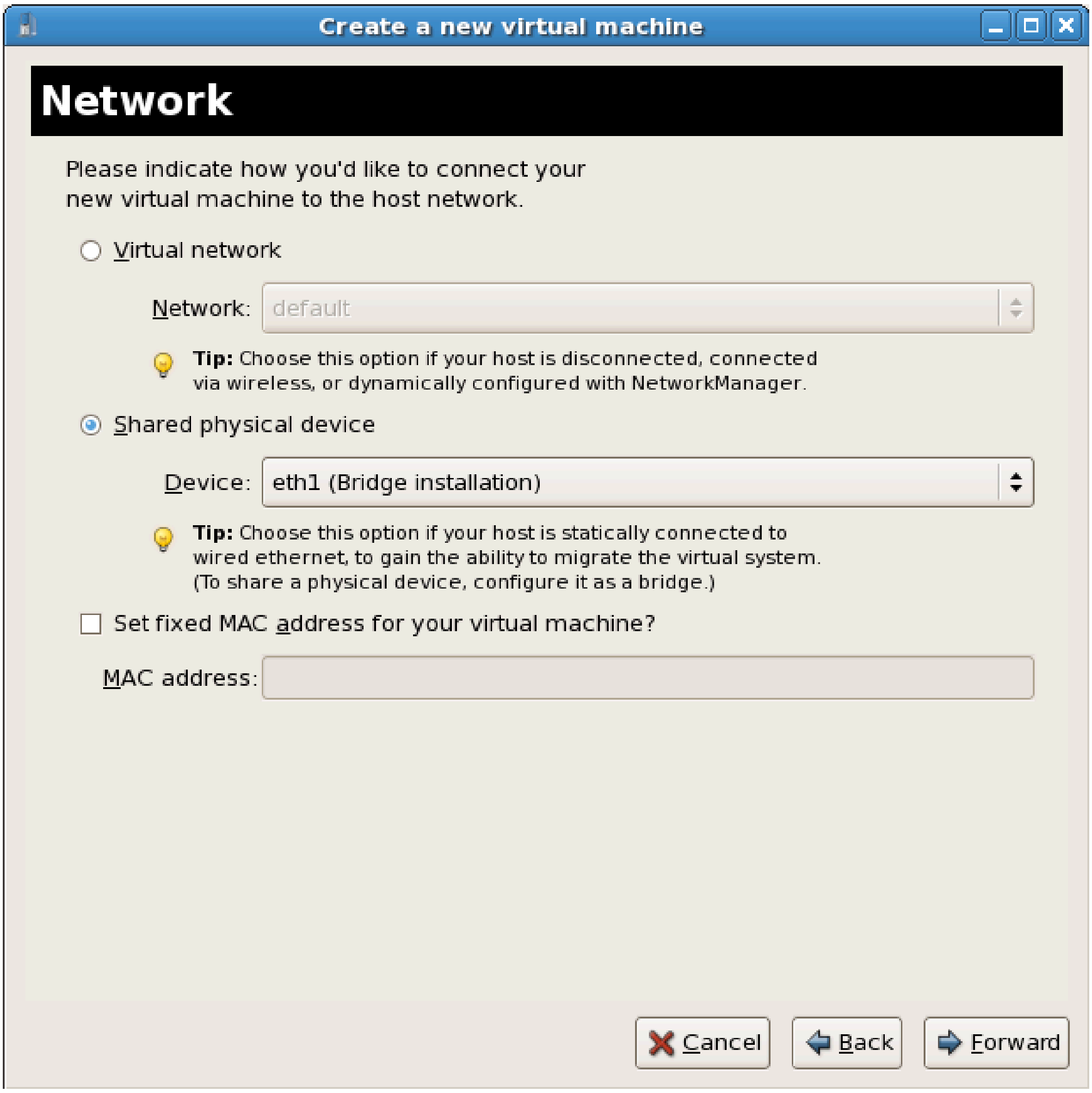
PXE installation with virt-manager

The steps below are the steps that vary from the standard virt-manager installation procedures. For the standard installations refer to *Chapter 7, Guest operating system installation procedures*.

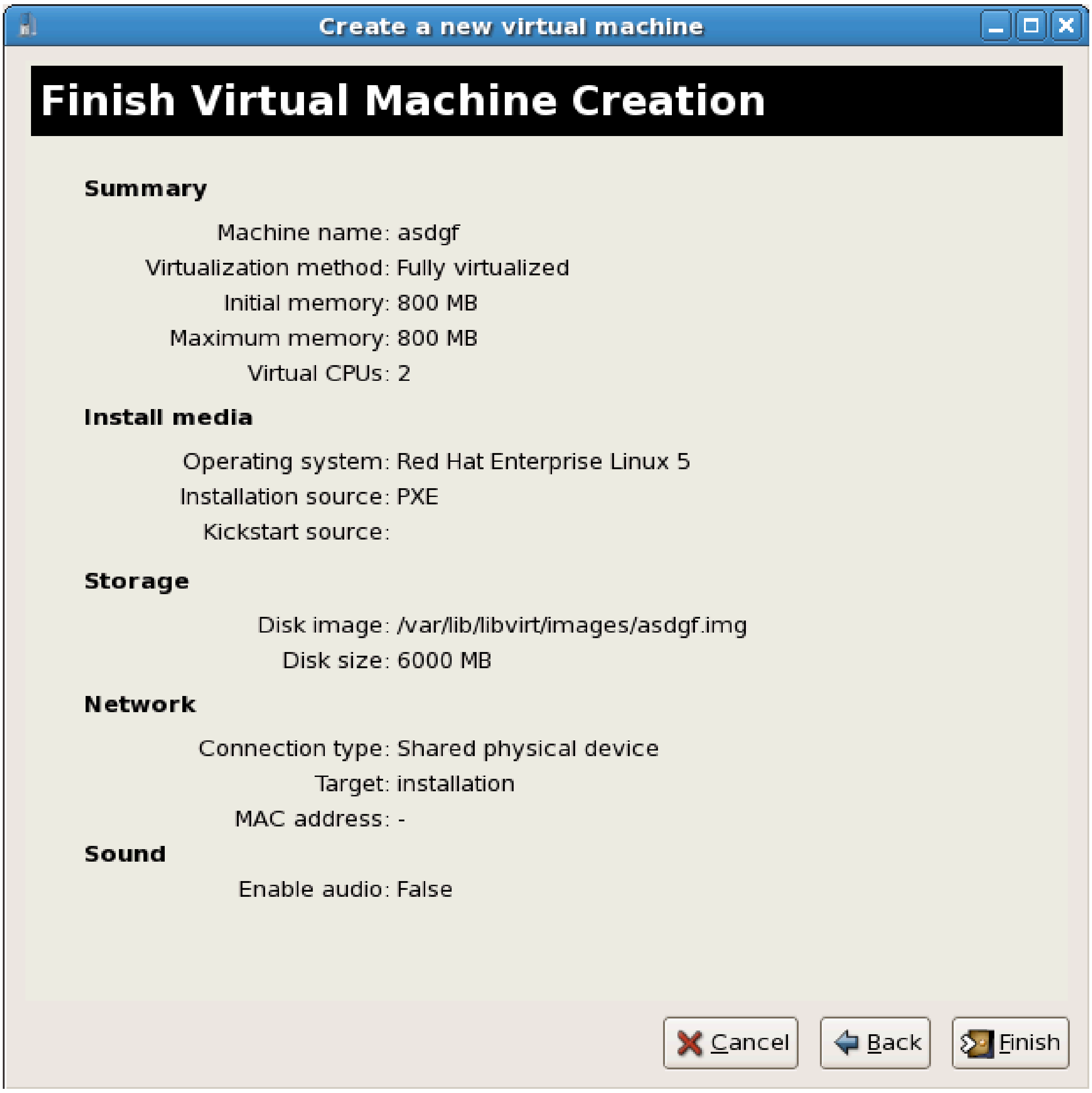
1. **Select PXE**
   Select PXE as the installation method.



2. **Select the bridge**
   Select **Shared physical device** and select the bridge created in the previous procedure.



3. **Start the installation**
   The installation is ready to start.



A DHCP request is sent and if a valid PXE server is found the guest installation processes will start.

Guest operating system installation procedures

This chapter covers how to install various guest operating systems in a virtualized environment on Red Hat Enterprise Linux. To understand the basic processes, refer to *Chapter 6, Virtualized guest installation overview*.

7.1. Installing Red Hat Enterprise Linux 5 as a para-virtualized guest

This section describes how to install Red Hat Enterprise Linux 5 as a para-virtualized guest. Para-virtualization is a faster than full virtualization and supports all of the advantages of full virtualization. Para-virtualization requires a special, supported kernel, the **kernel-xen** kernel.

> **Important note on para-virtualization**
>
> Para-virtualization only works with the Xen hypervisor. Para-virtualization does not work with the KVM hypervisor.

Ensure you have root access before starting the installation.

This method installs Red Hat Enterprise Linux from a remote server. The installation instructions presented in this section are similar to installing from the minimal installation live CD-ROM.

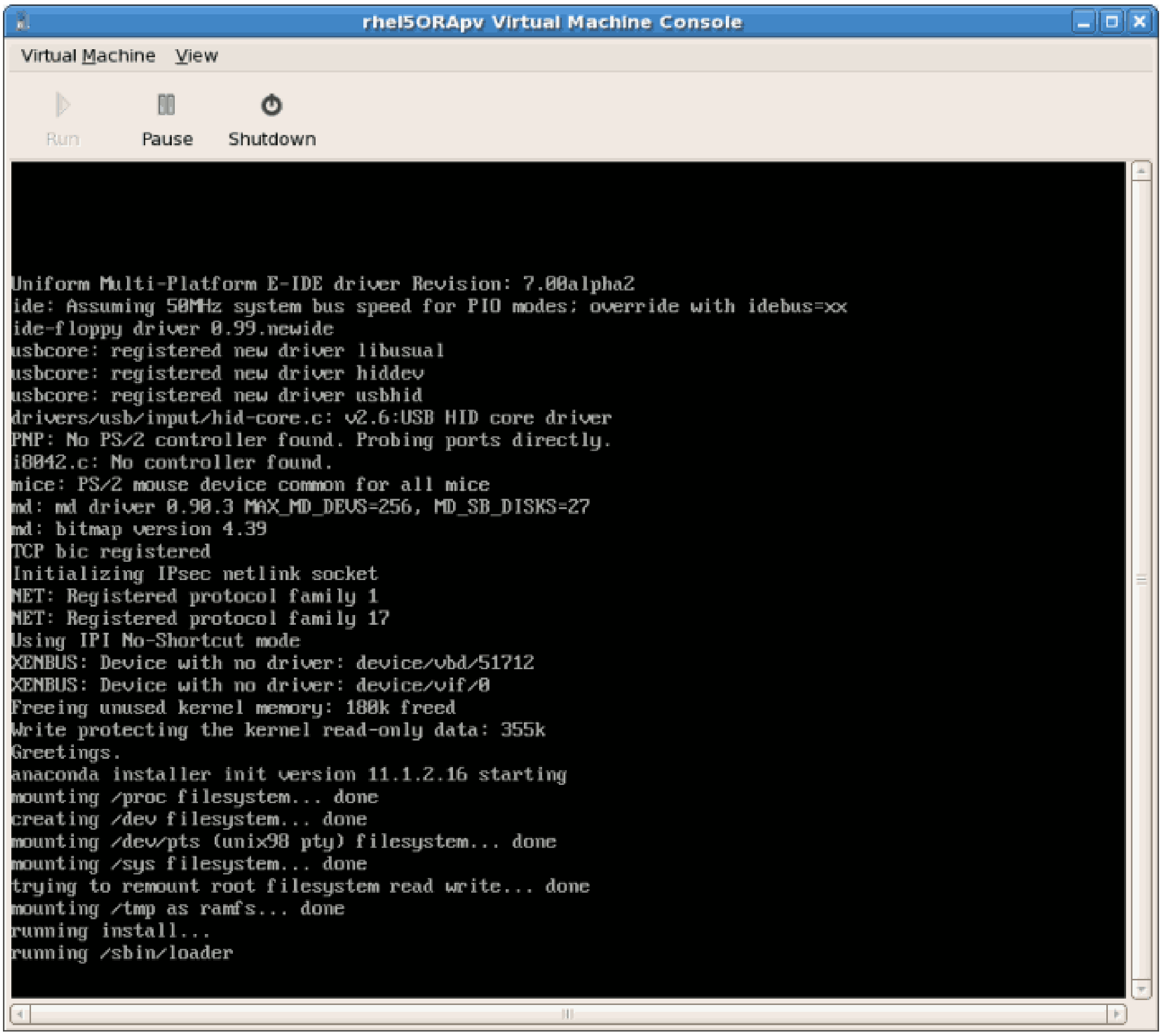
Create para-virtualized Red Hat Enterprise Linux 5 guests using virt-manager or virt-install. For instructions on **virt-manager**, refer to the procedure in *Section 6.2, “Creating guests with virt-manager”*.

Create a para-virtualized guest with the command line based **virt-install** tool. The **--vnc** option shows the graphical installation. The name of the guest in the example is *rhel5PV*, the disk image file is *rhel5PV.dsk* and a local mirror of the Red Hat Enterprise Linux 5 installation tree is *ftp://10.1.1.1/trees/RHEL5-B2-Server-i386/*. Replace those values with values accurate for your system and network.

```
# virt-install -n rhel5PV -r 500 \
-f /var/lib/libvirt/images/rhel5PV.dsk -s 3 --vnc -p \
-l ftp://10.1.1.1/trees/RHEL5-B2-Server-i386/
```

> **Automating installation**
>
> Red Hat Enterprise Linux can be installed without a graphical interface or manual input. Use Kickstart files to automate the installation process.

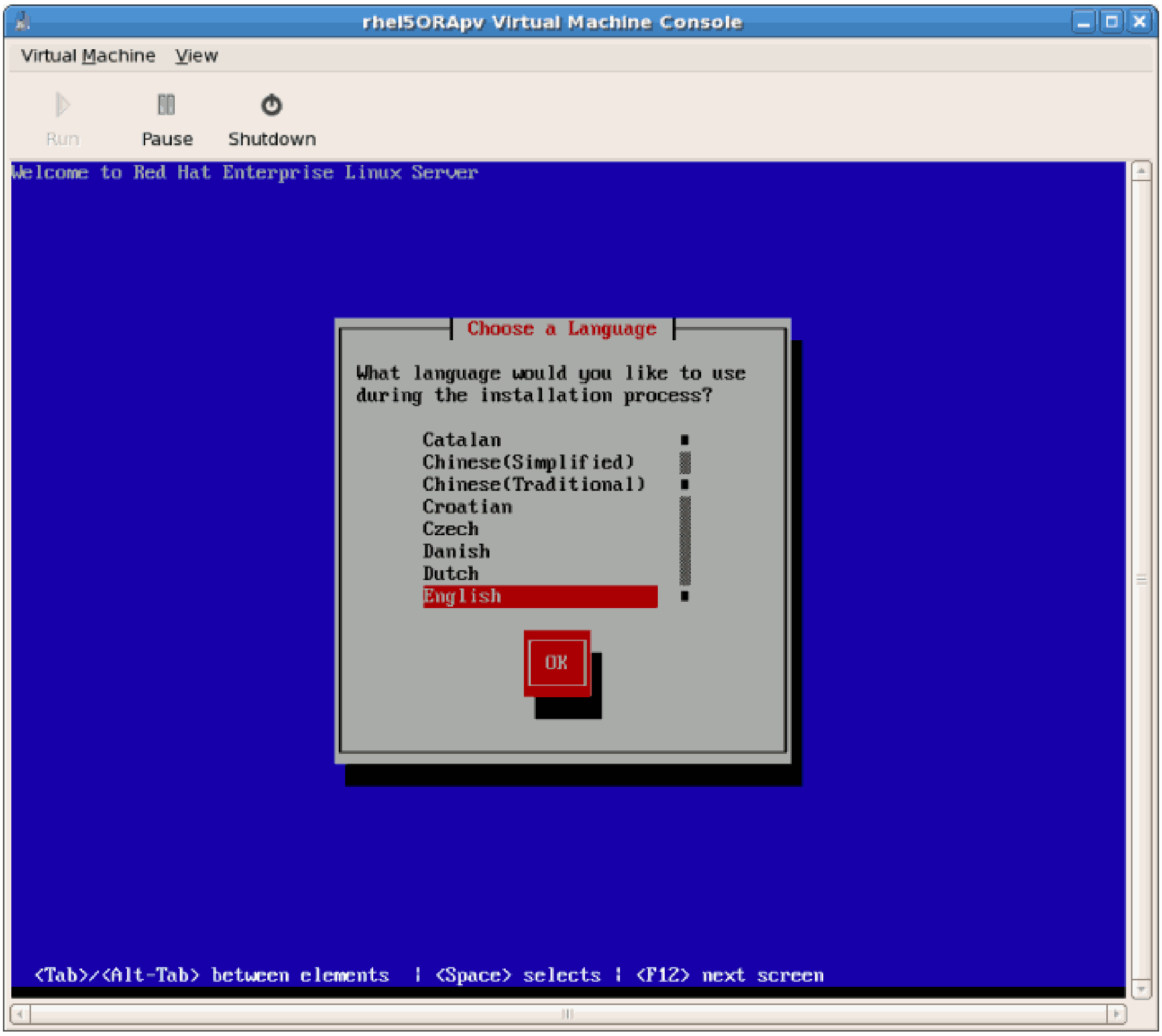
Using either method opens this window, displaying the initial boot phases of your guest:



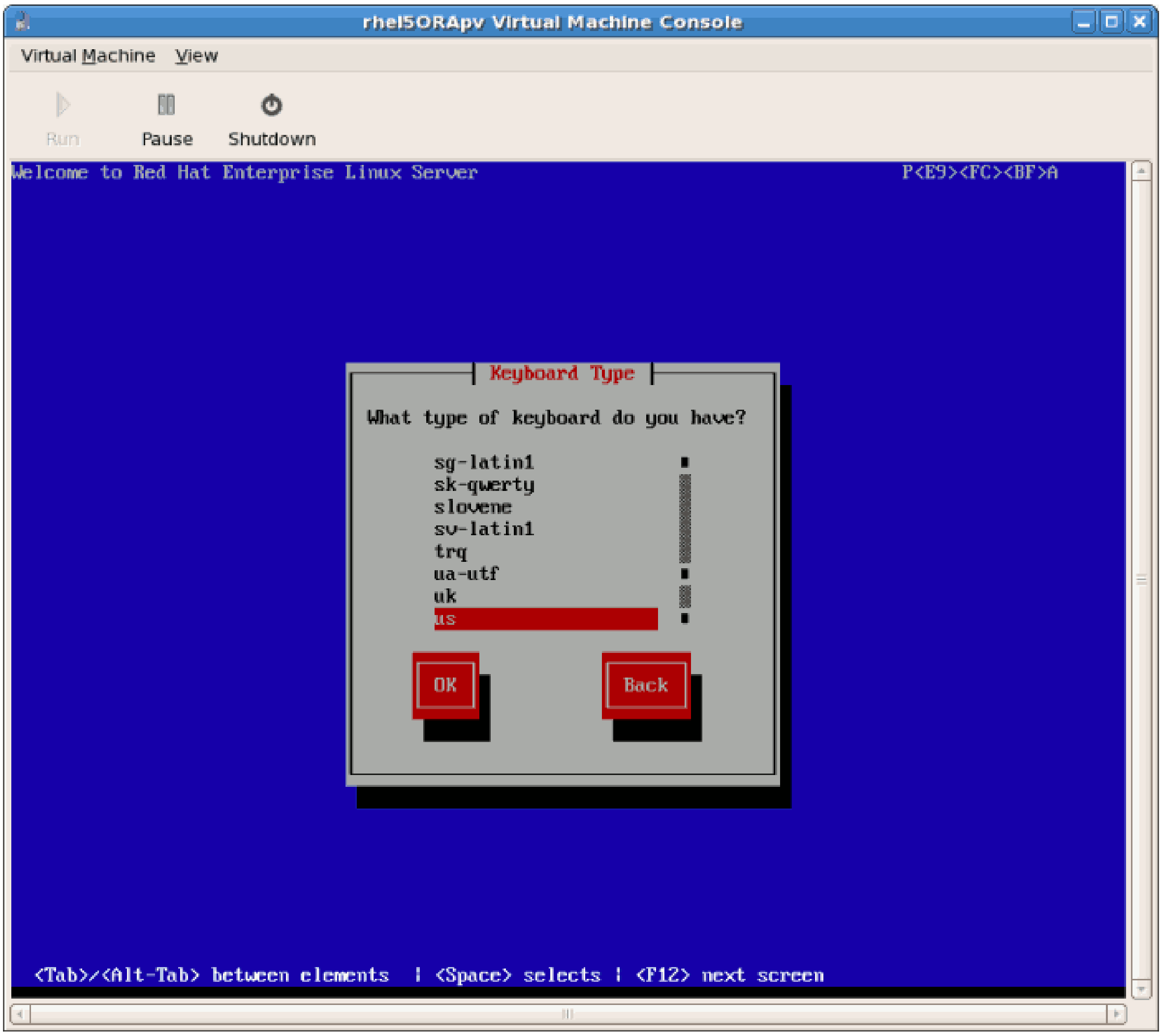
After your guest has completed its initial boot, the standard installation process for Red Hat Enterprise Linux starts. For most systems the default answers are acceptable.

Procedure 7.1. Para-virtualized Red Hat Enterprise Linux guest installation procedure

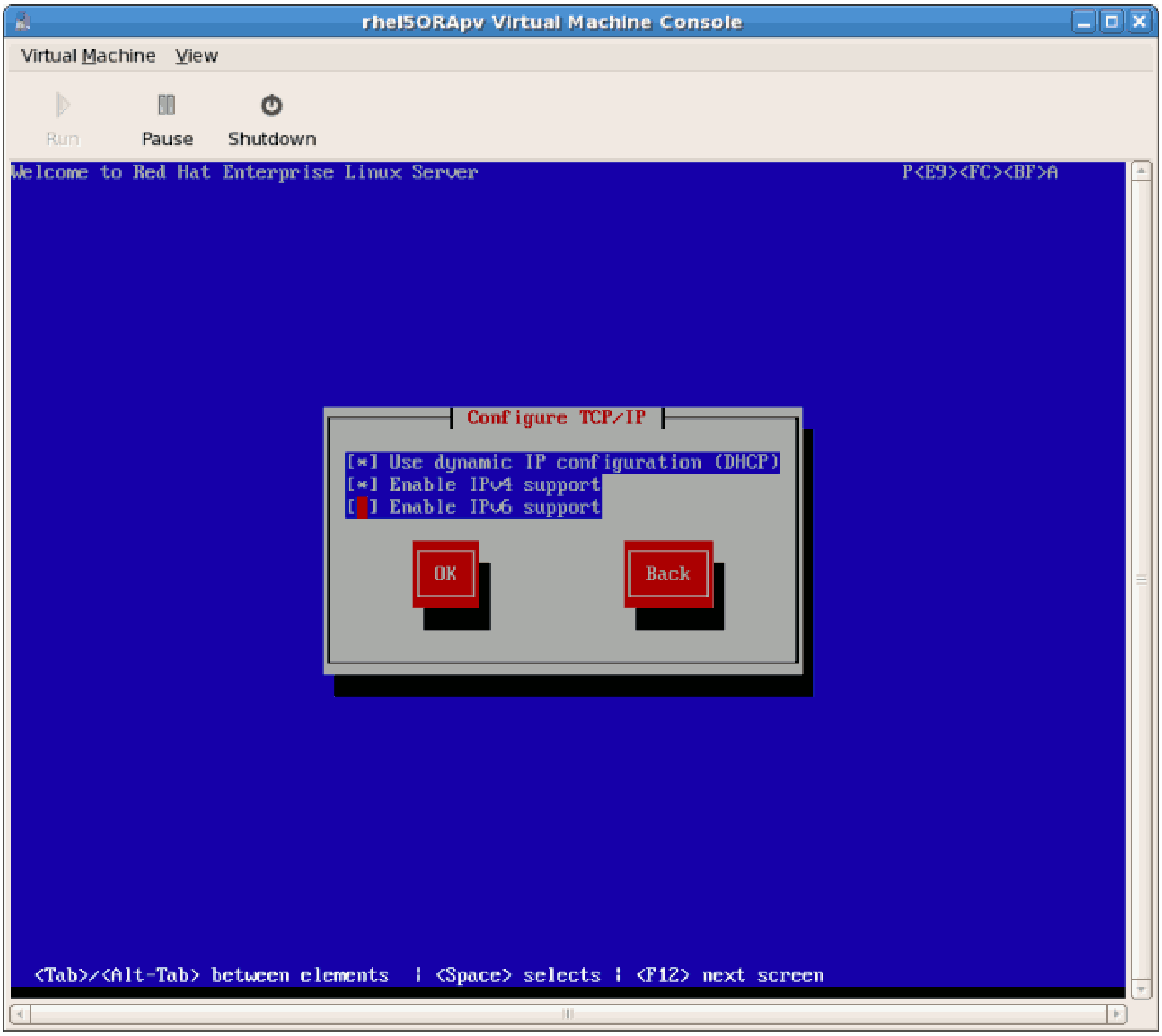
1. Select the language and click **OK**.



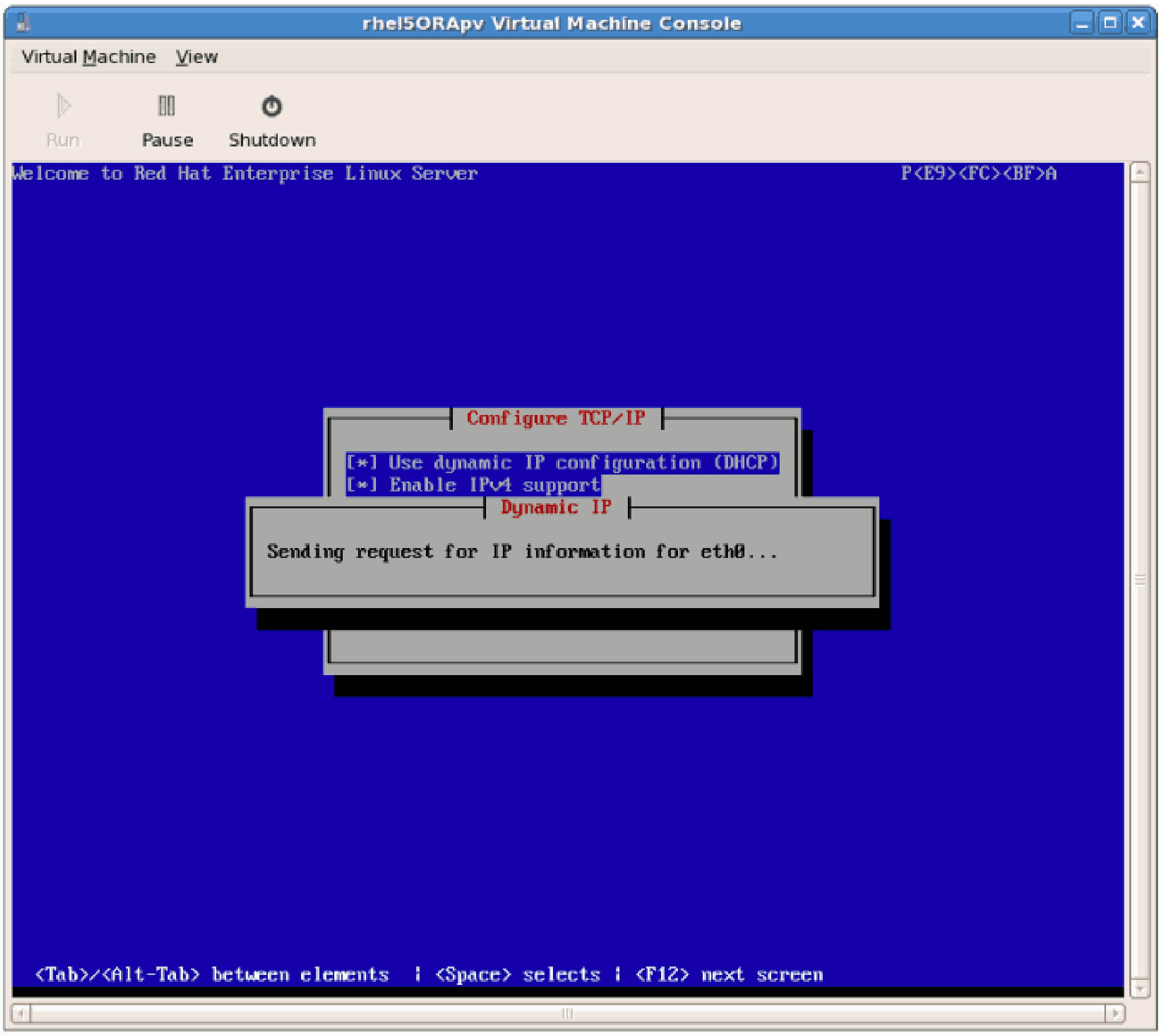
2. Select the keyboard layout and click **OK**.



3. Assign the guest's network address. Choose to use `DHCP` (as shown below) or a static IP address:



4. If you select DHCP the installation process will now attempt to acquire an IP address:

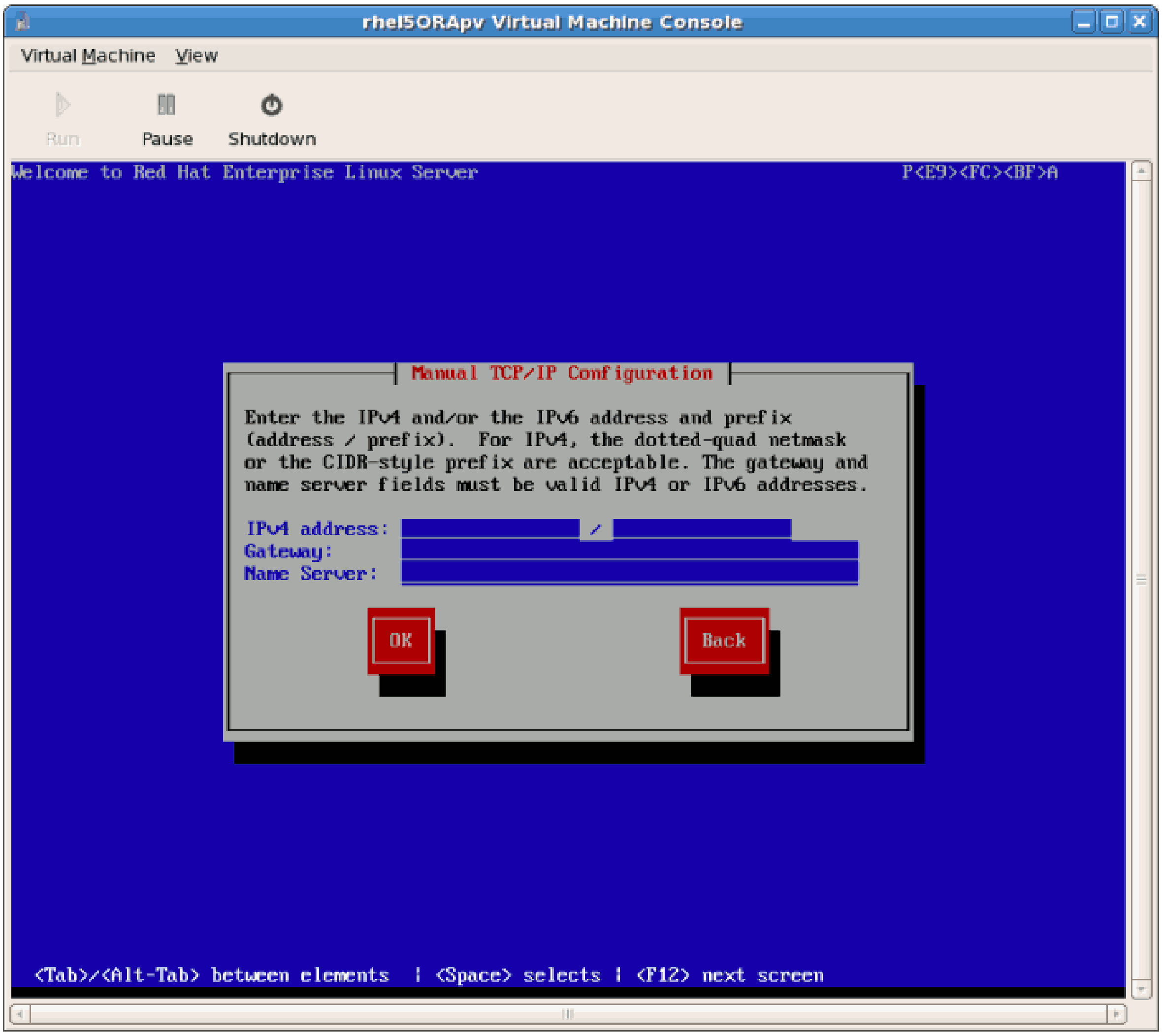


5. If you chose a static IP address for your guest this prompt appears. Enter the details on the guest's networking configuration:

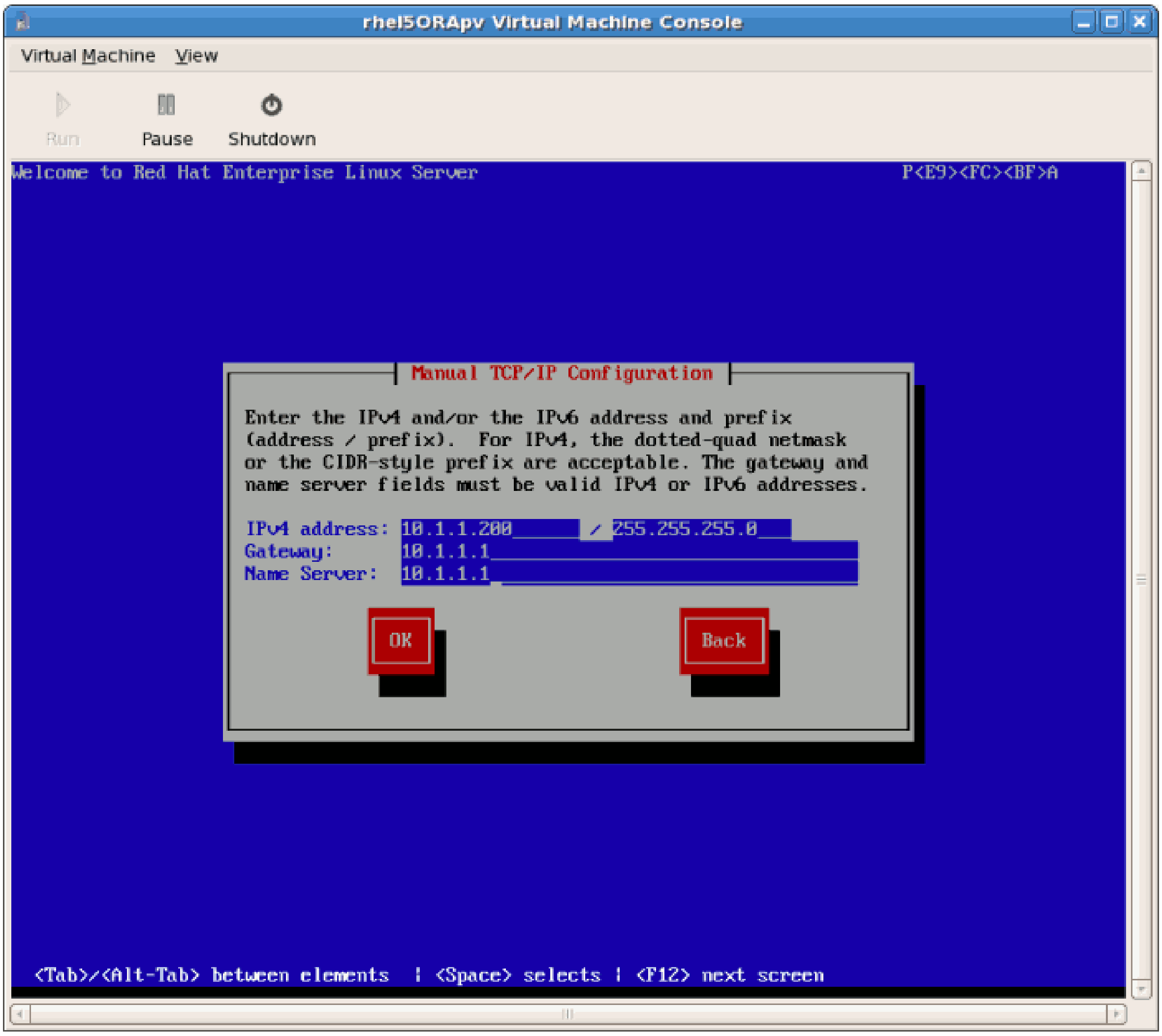
   a. Enter a valid IP address. Ensure the IP address you enter can reach the server with the installation tree.

   b. Enter a valid Subnet mask, default gateway and name server address.

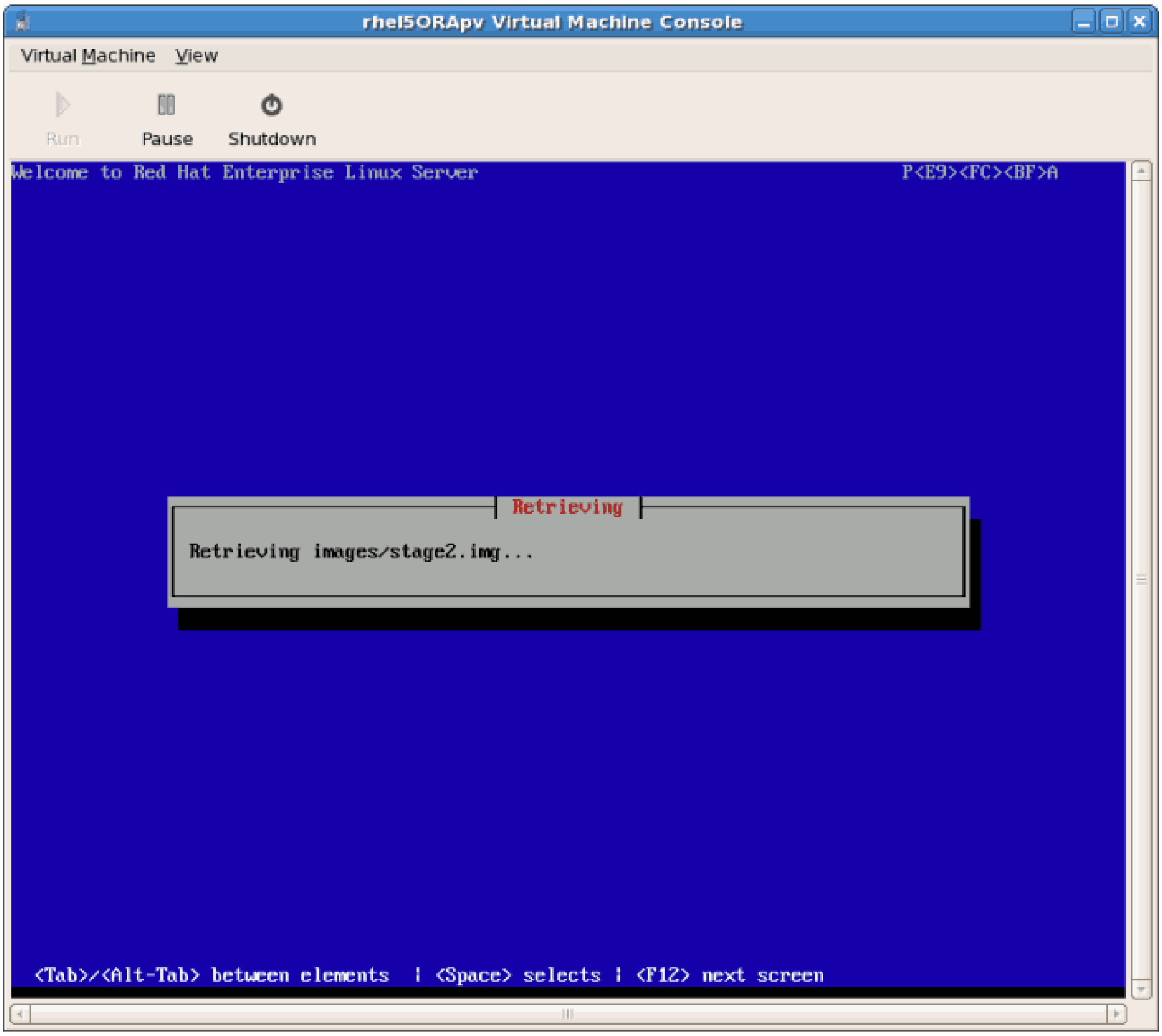
   Select the language and click **OK**.



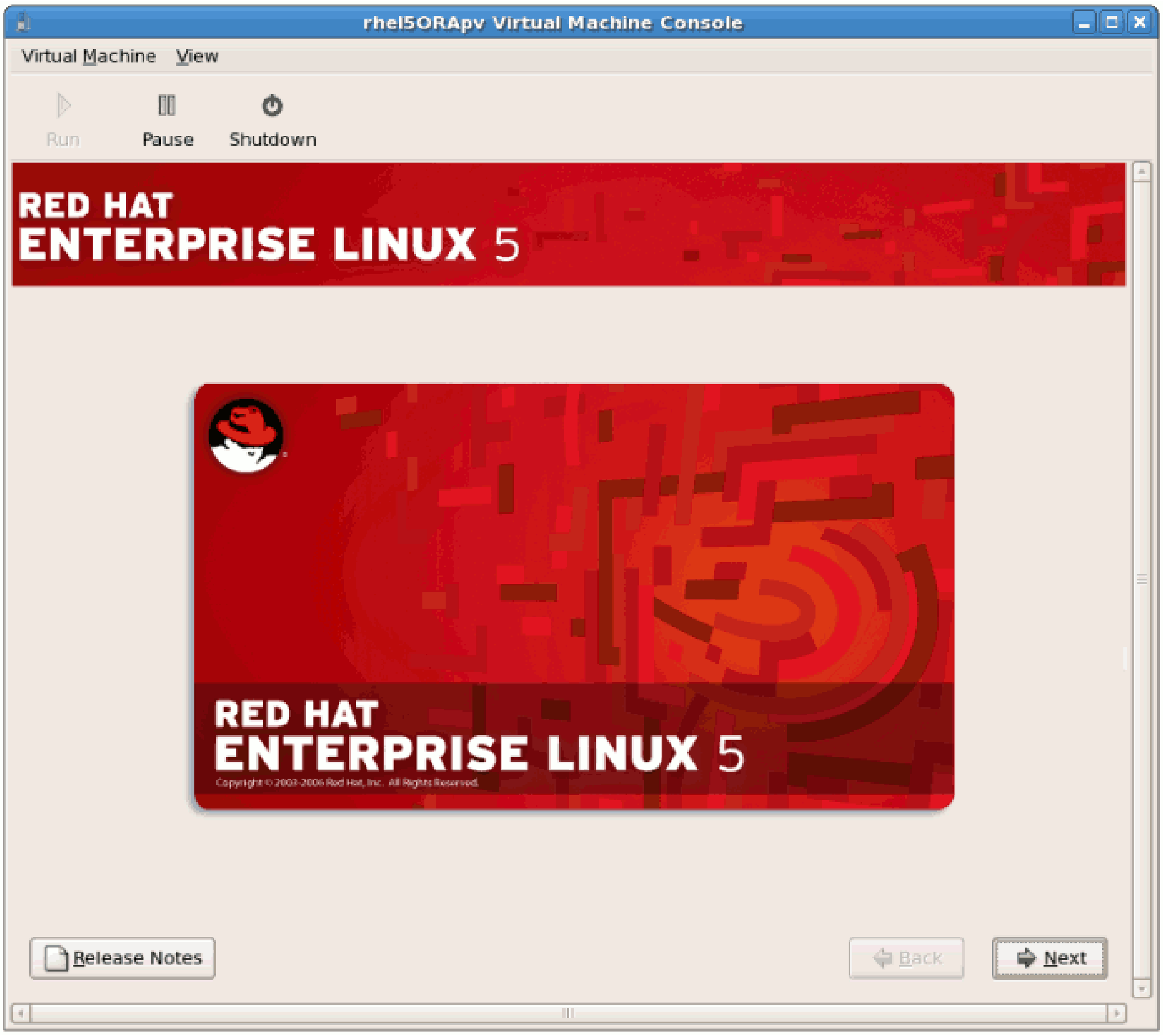
6. This is an example of a static IP address configuration:



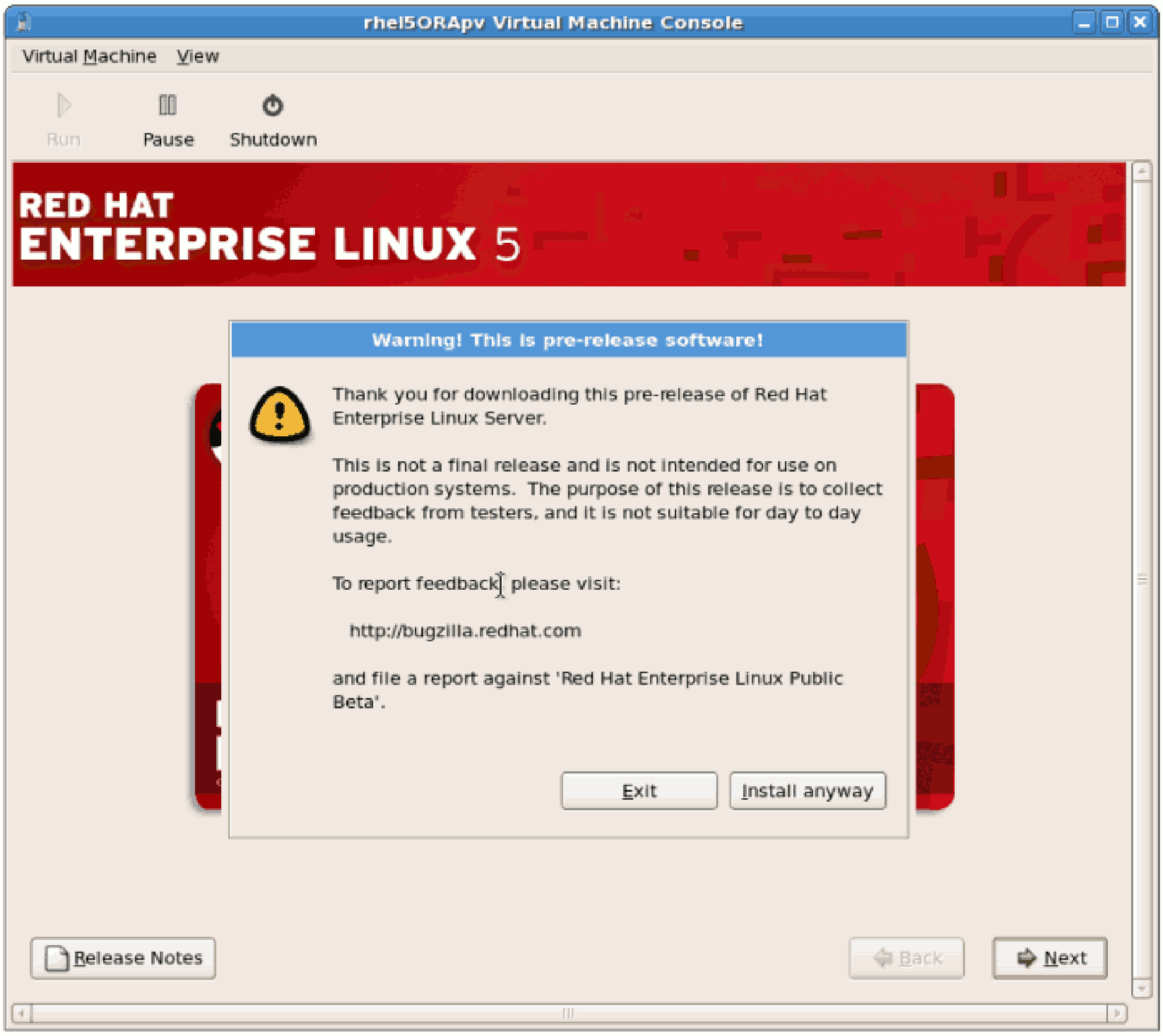
7. The installation process now retrieves the files it needs from the server:



Once the initial steps are complete the graphical installation process starts.

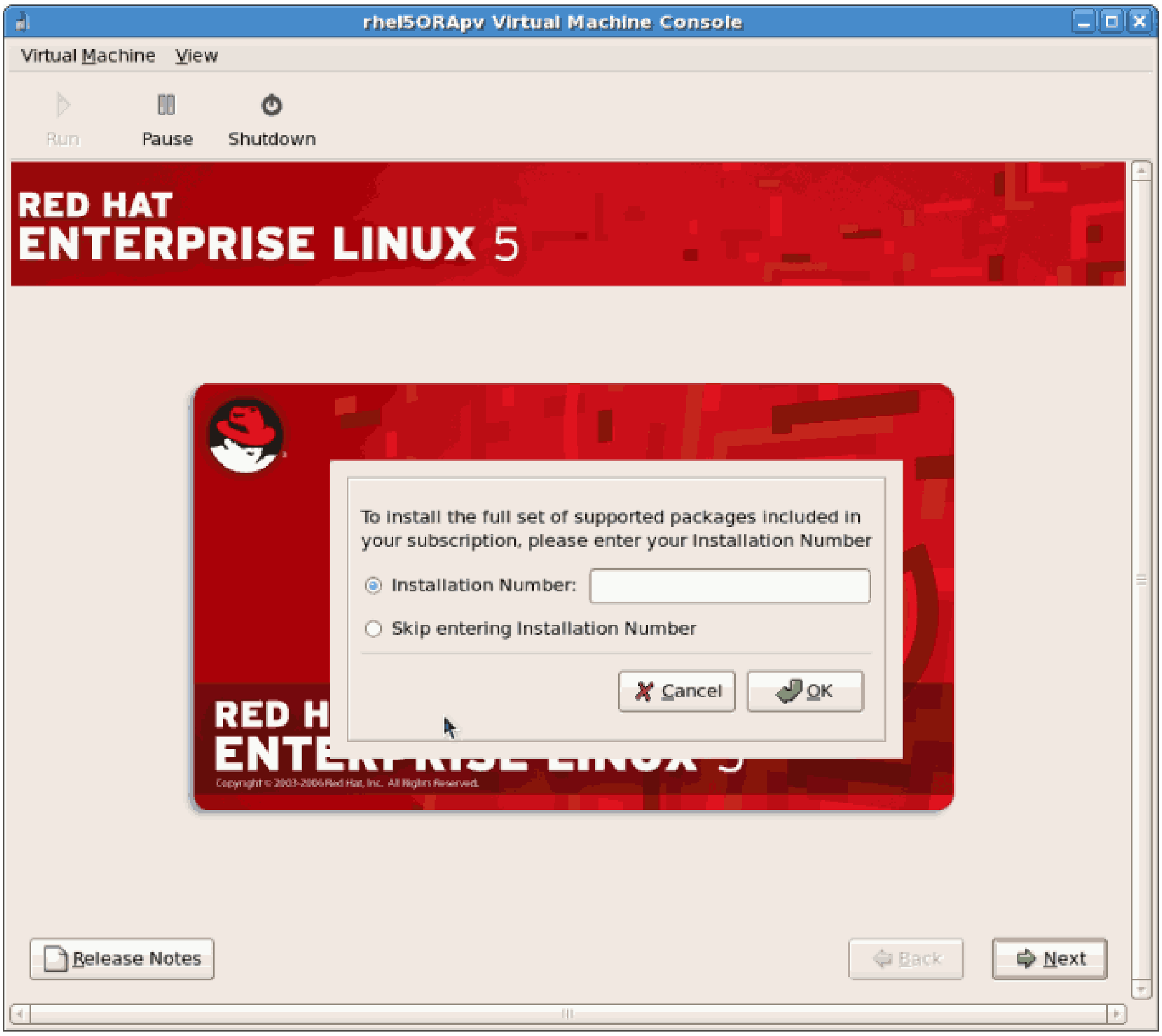


If you are installing a Beta or early release distribution confirm that you want to install the operating system. Click **Install Anyway**, and then click **OK**:


rhel5ORApv Virtual Machine Console
Virtual Machine
View
Run
Pause
Shutdown
RED HAT
ENTERPRISE LINUX 5
Warning! This is pre-release software!
Thank you for downloading this pre-release of Red Hat Enterprise Linux Server.
This is not a final release and is not intended for use on production systems. The purpose of this release is to collect feedback from testers, and it is not suitable for day to day usage.
To report feedback, please visit:
http://bugzilla.redhat.com
and file a report against 'Red Hat Enterprise Linux Public Beta'.
Exit
Install anyway
Release Notes
Back
Next

Procedure 7.2. The graphical installation process

1. Enter a valid registration code. If you have a valid RHN subscription key please enter in the `Installation Number` field:

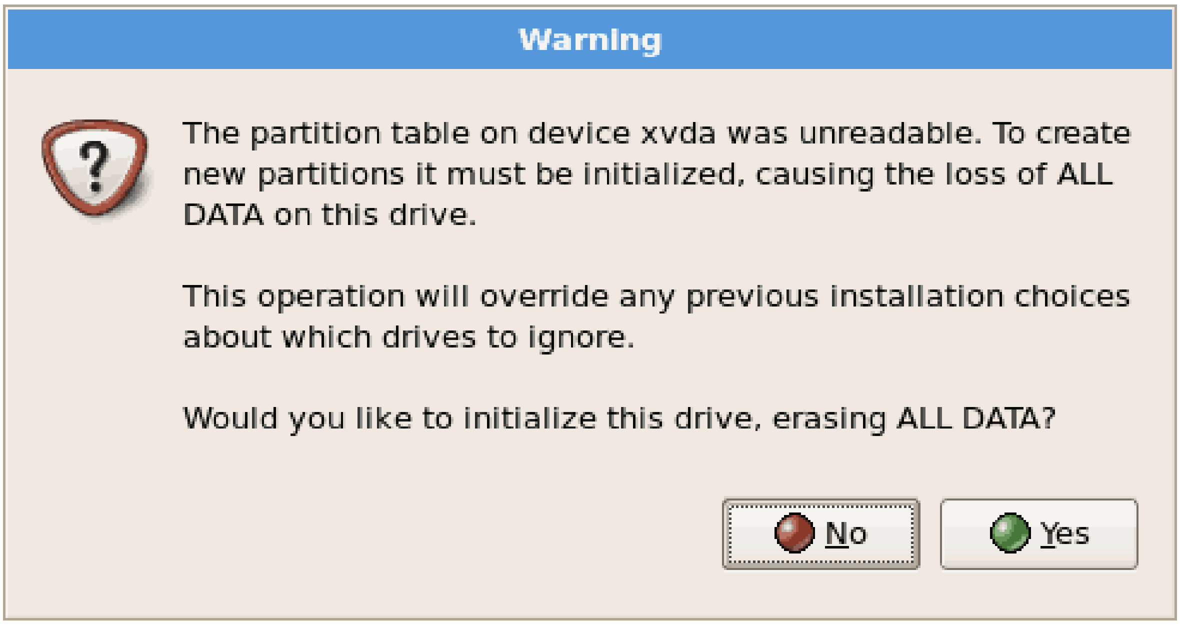


Note

If you skip the registration step the you can confirm your Red Hat Network account details after the installation with the **`rhn_register`** command. The **`rhn_register`** command requires root access.

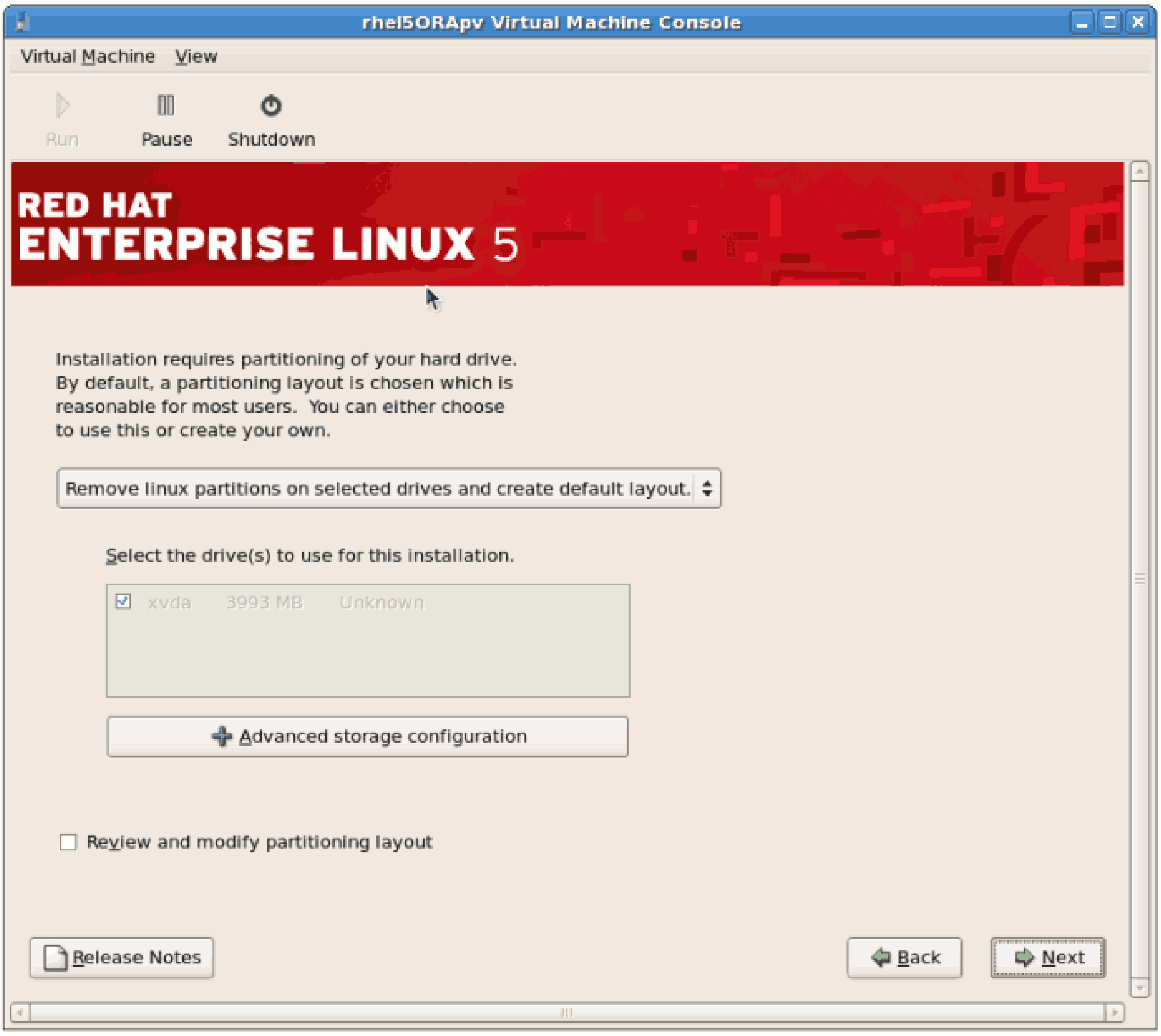
```
# rhn_register
```

2. The installation prompts you to confirm erasure of all data on the storage you selected for the installation:



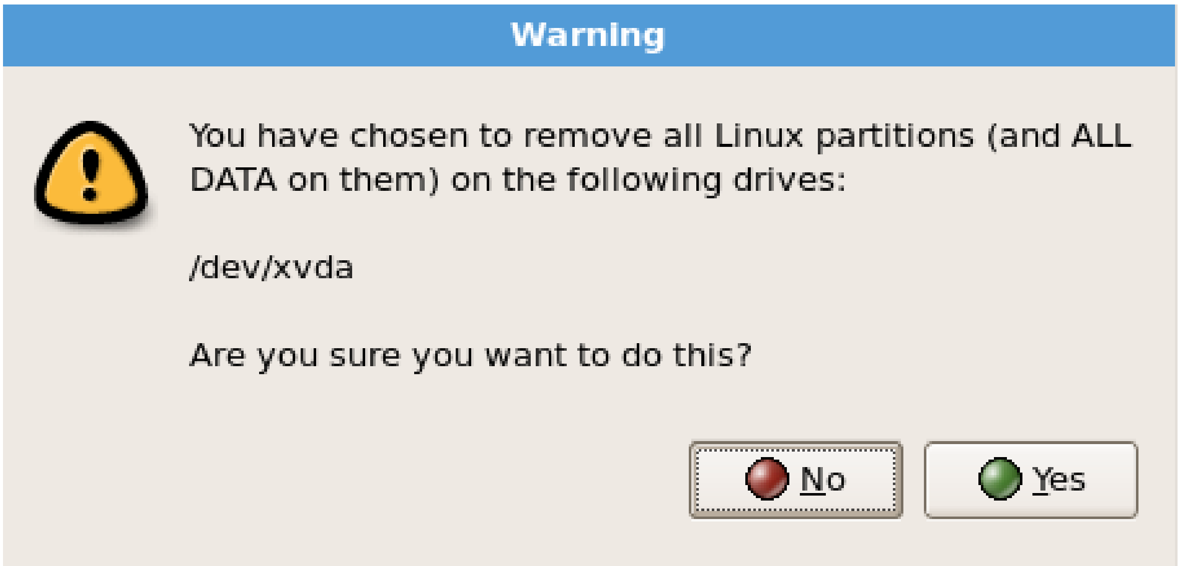
Click **Yes** to continue.

3. Review the storage configuration and partition layout. You can chose to select the advanced storage configuration if you want to use iSCSI for the guest's storage.



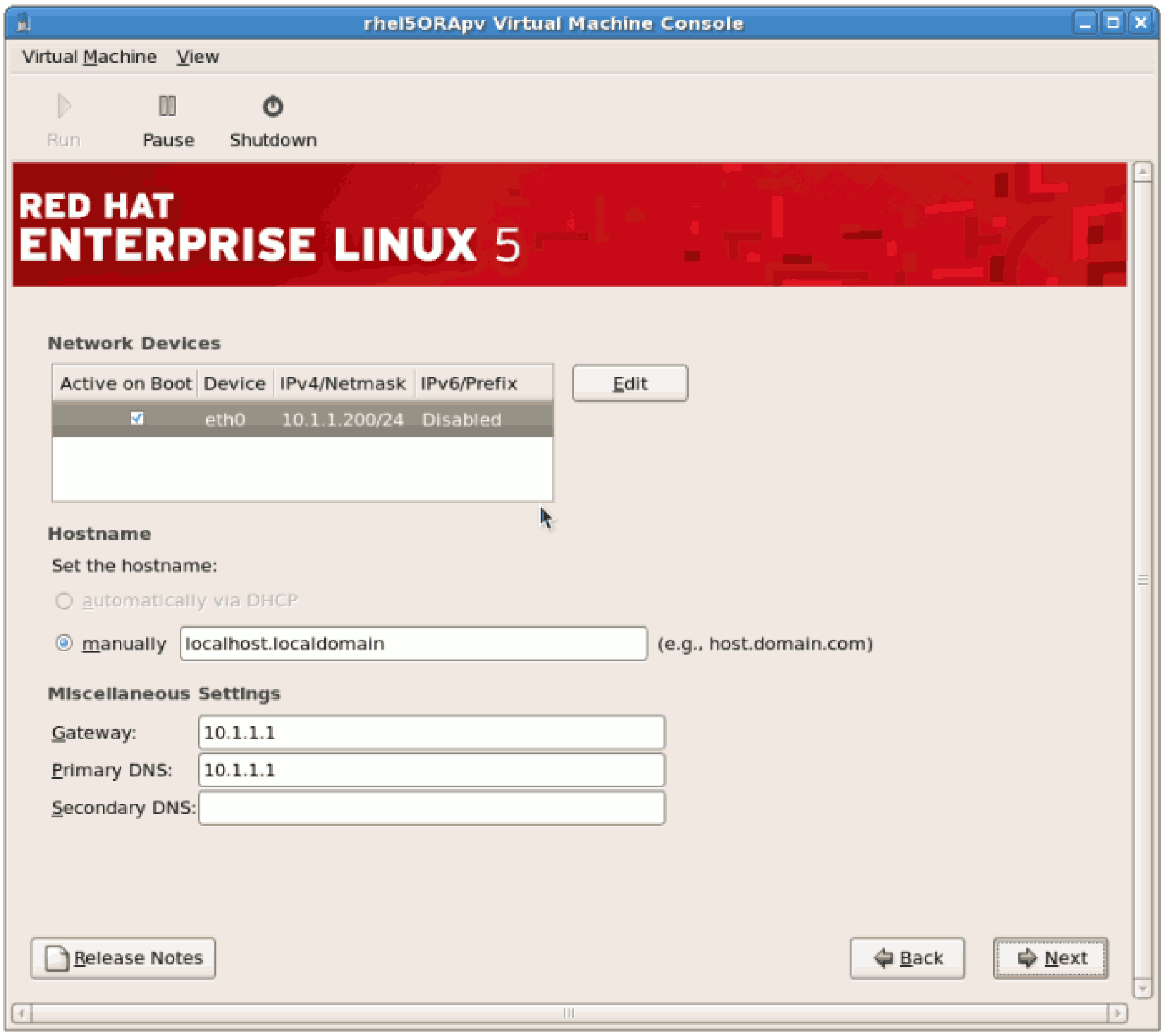
Make your selections then click **Next**.

4. Confirm the selected storage for the installation.



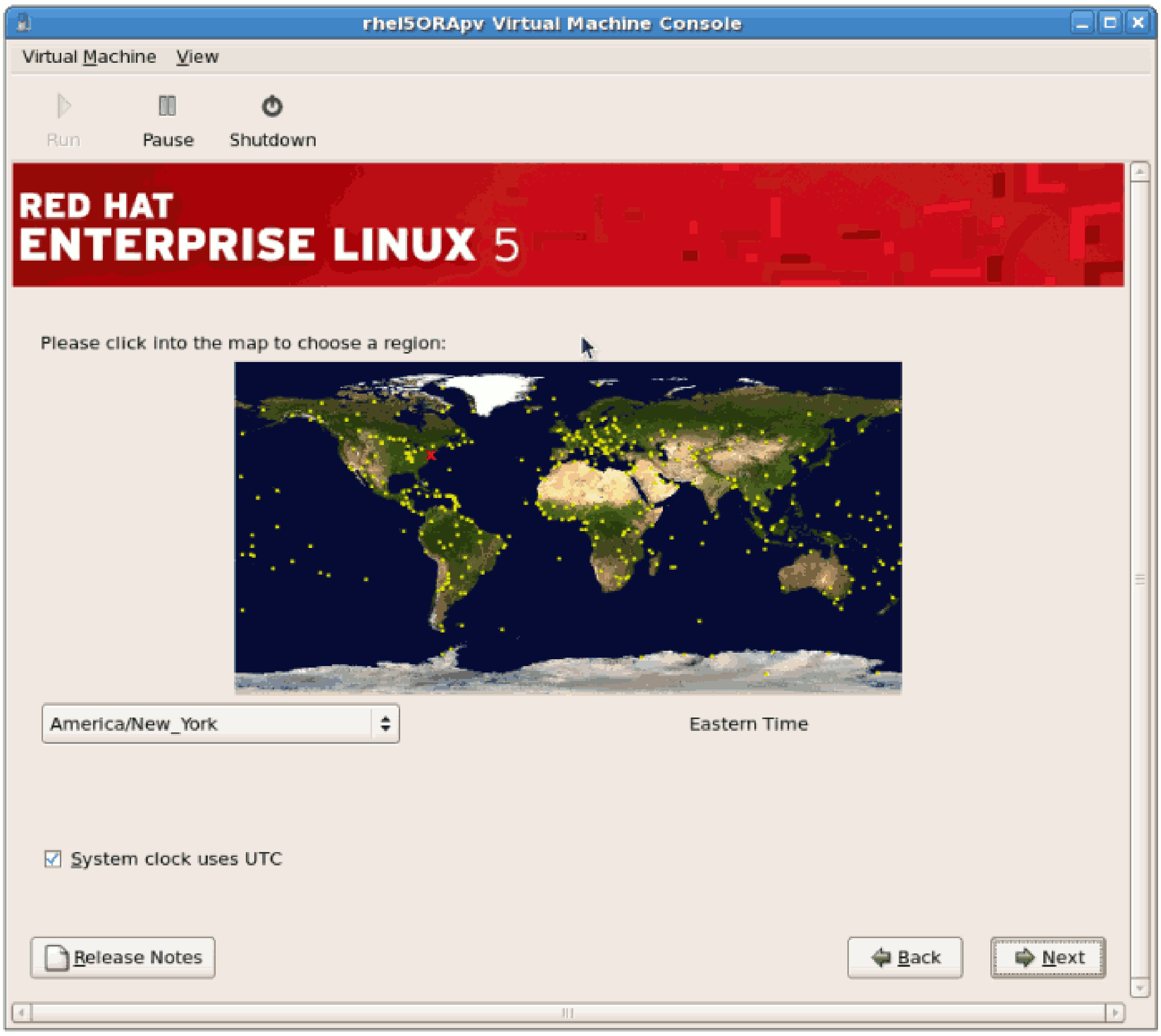
Click **Yes** to continue.

5. Configure networking and hostname settings. These settings are populated with the data entered earlier in the installation process. Change these settings if necessary.

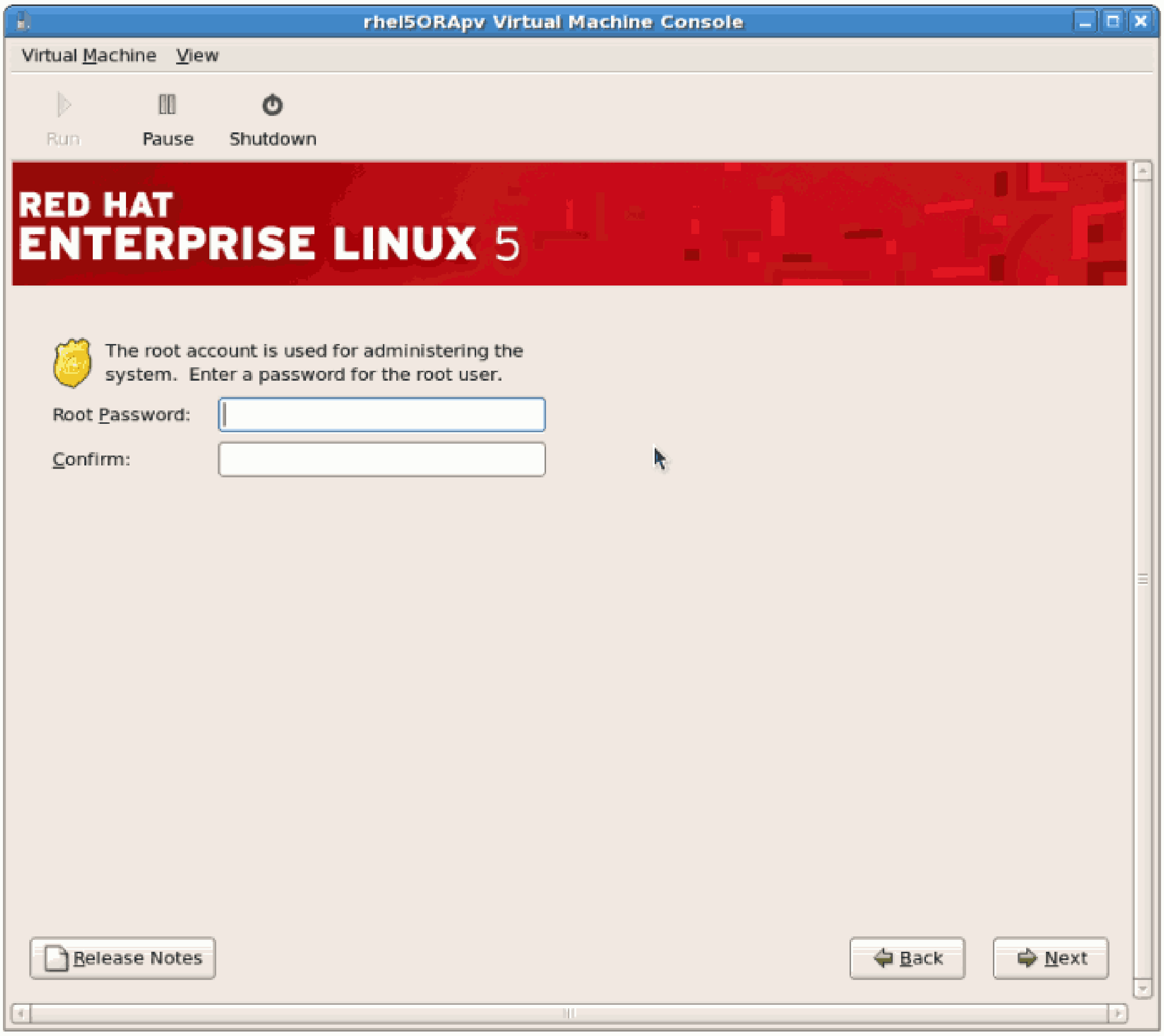


Click **OK** to continue.

6. Select the appropriate time zone for your environment.

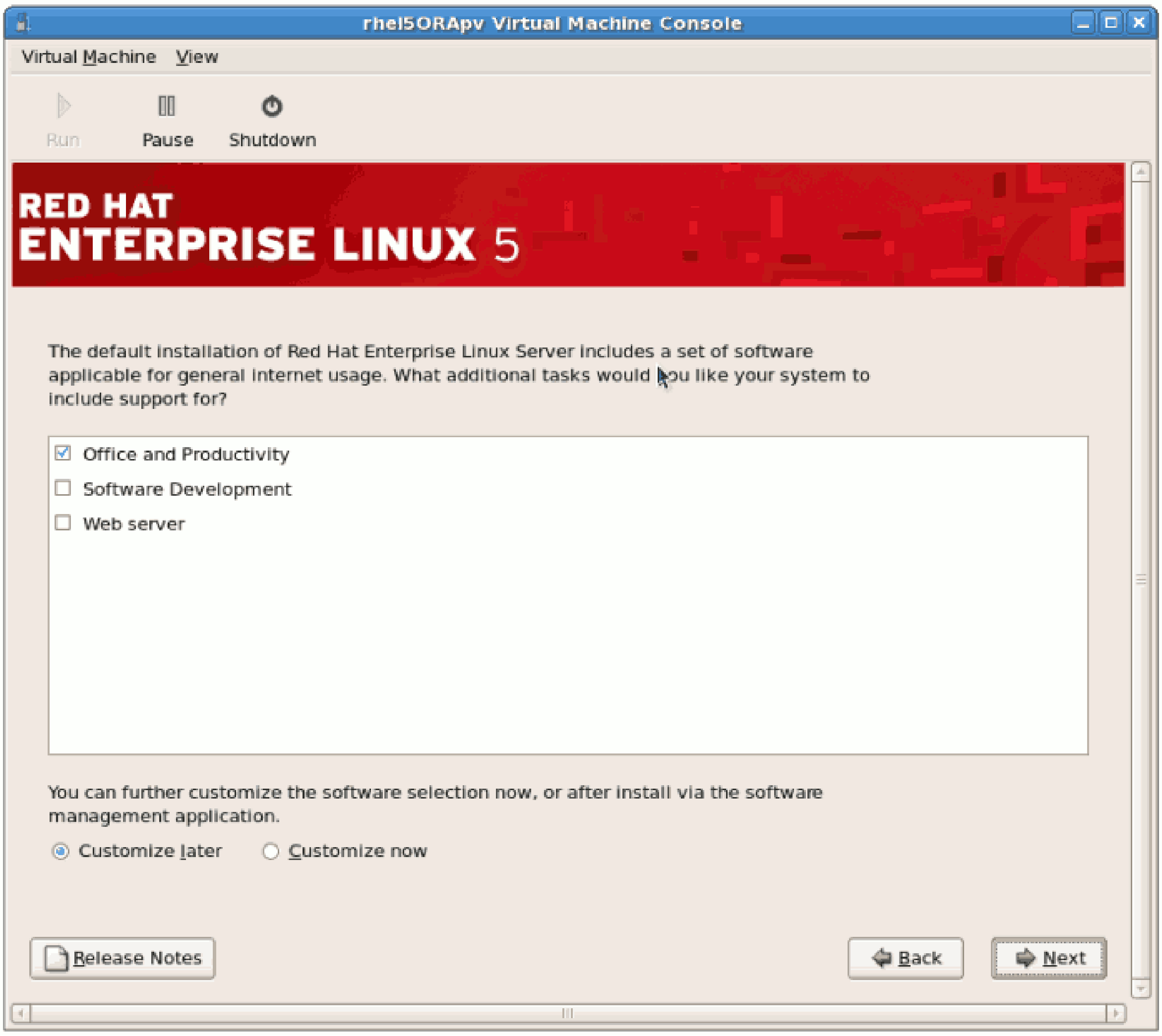


7. Enter the root password for the guest.



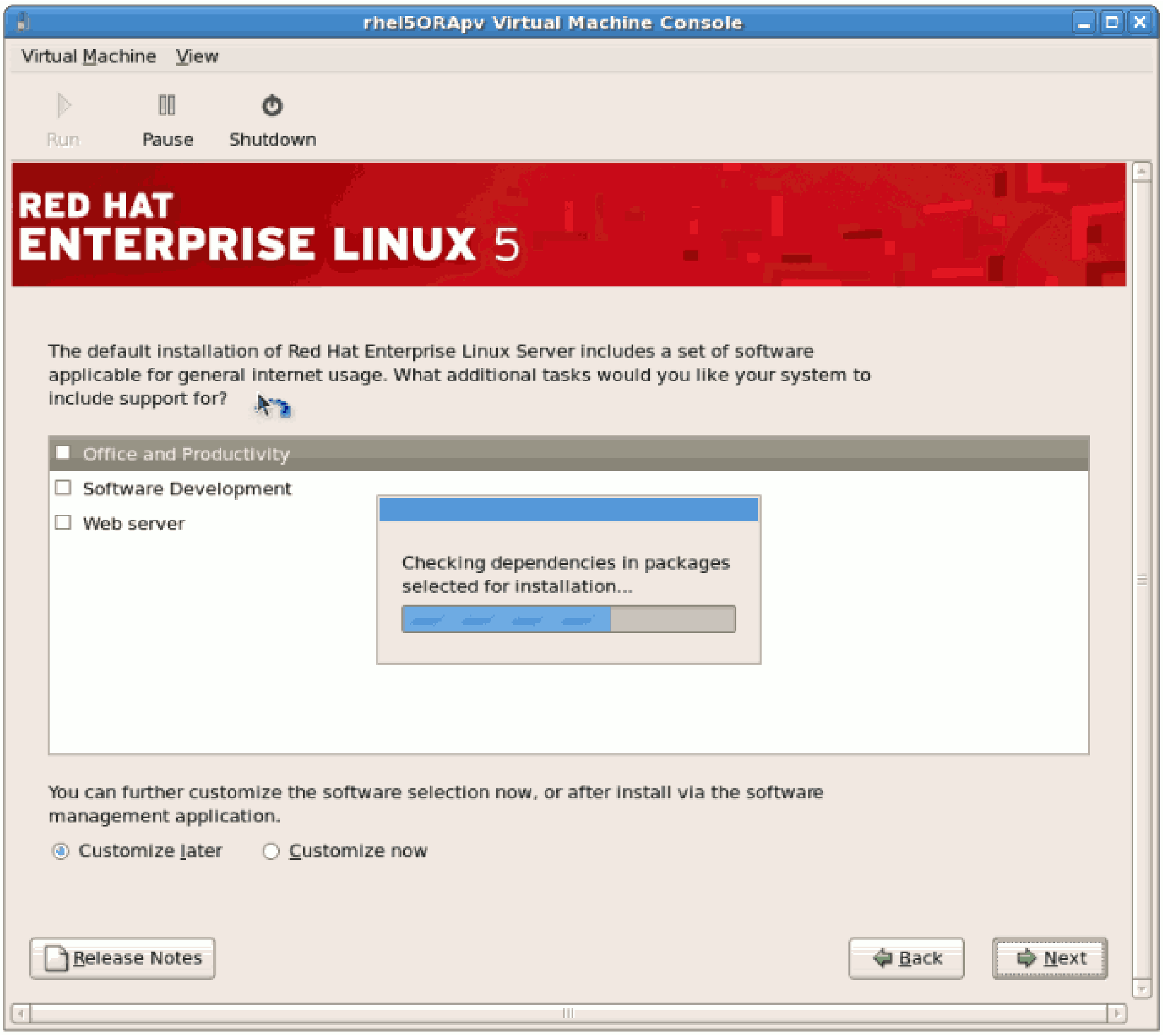
Click **Next** to continue.

8. Select the software packages to install. Select the **Customize Now** button. You must install the **kernel-xen** package in the **System** directory. The **kernel-xen** package is required for para-virtualization.



Click **Next**.

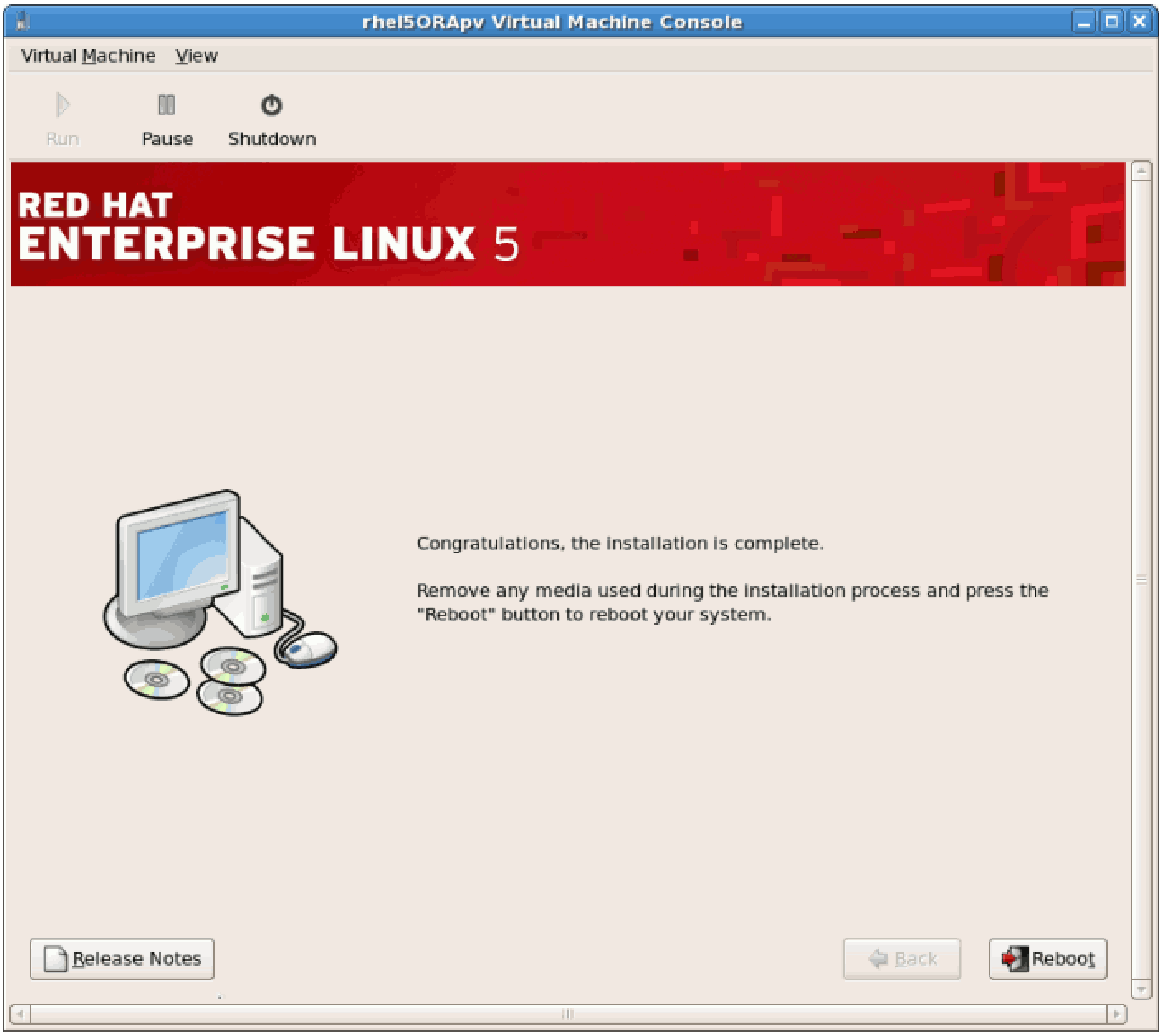
9. Dependencies and space requirements are calculated.



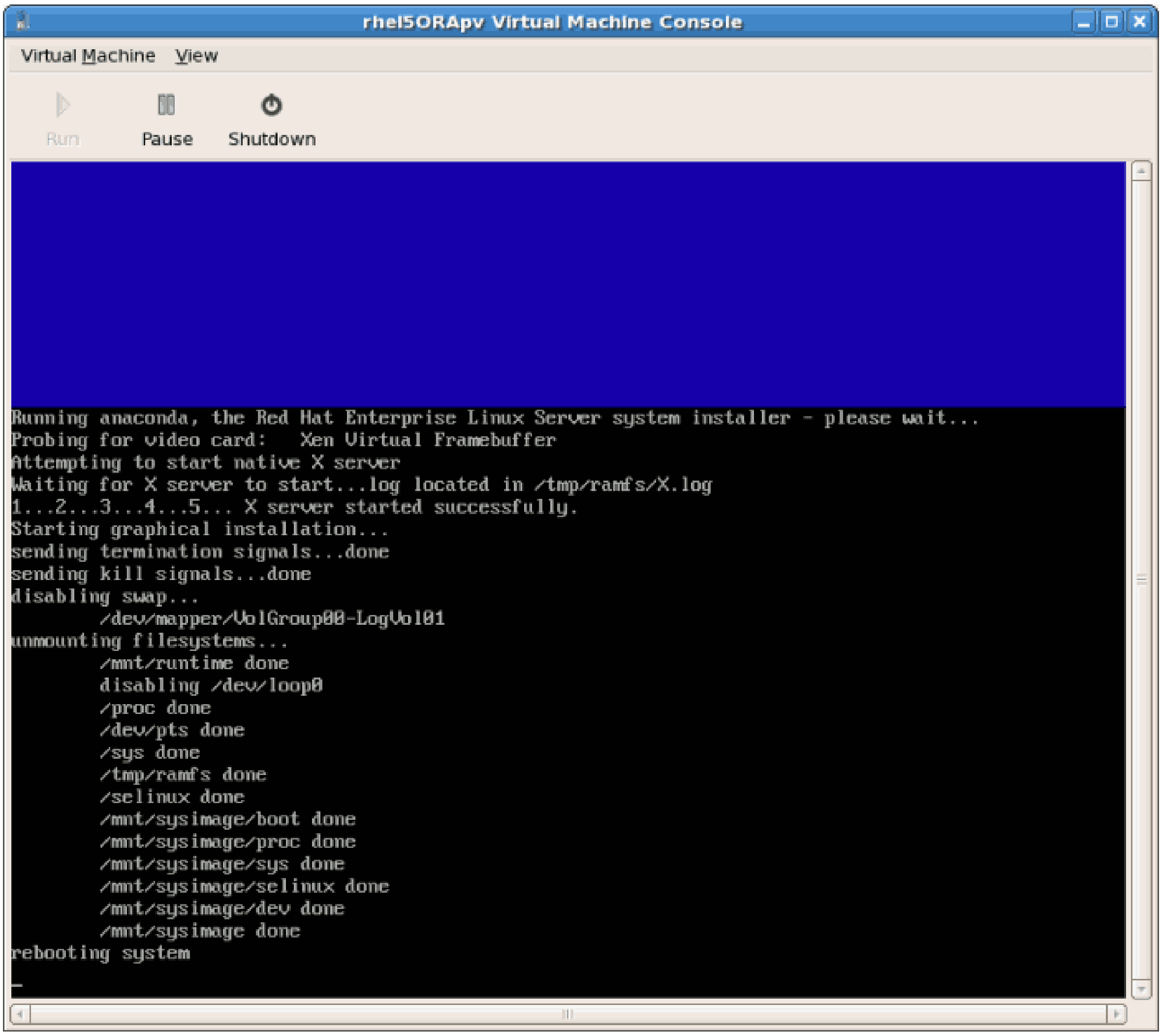
10. After the installation dependencies and space requirements have been verified click **Next** to start the actual installation.

11. All of the selected software packages are installed automatically.

12. After the installation has finished reboot your guest:



13. The guest will not reboot, instead it will shutdown..



14. Boot the guest. The guest's name was chosen when you used the **`virt-install`** in *Section 7.1, “Installing Red Hat Enterprise Linux 5 as a para-virtualized guest”*. If you used the default example the name is *`rhel5PV`*.

    Run:

15. Booting the guest starts the *First Boot* configuration screen. This wizard prompts you for some basic configuration choices for your guest.

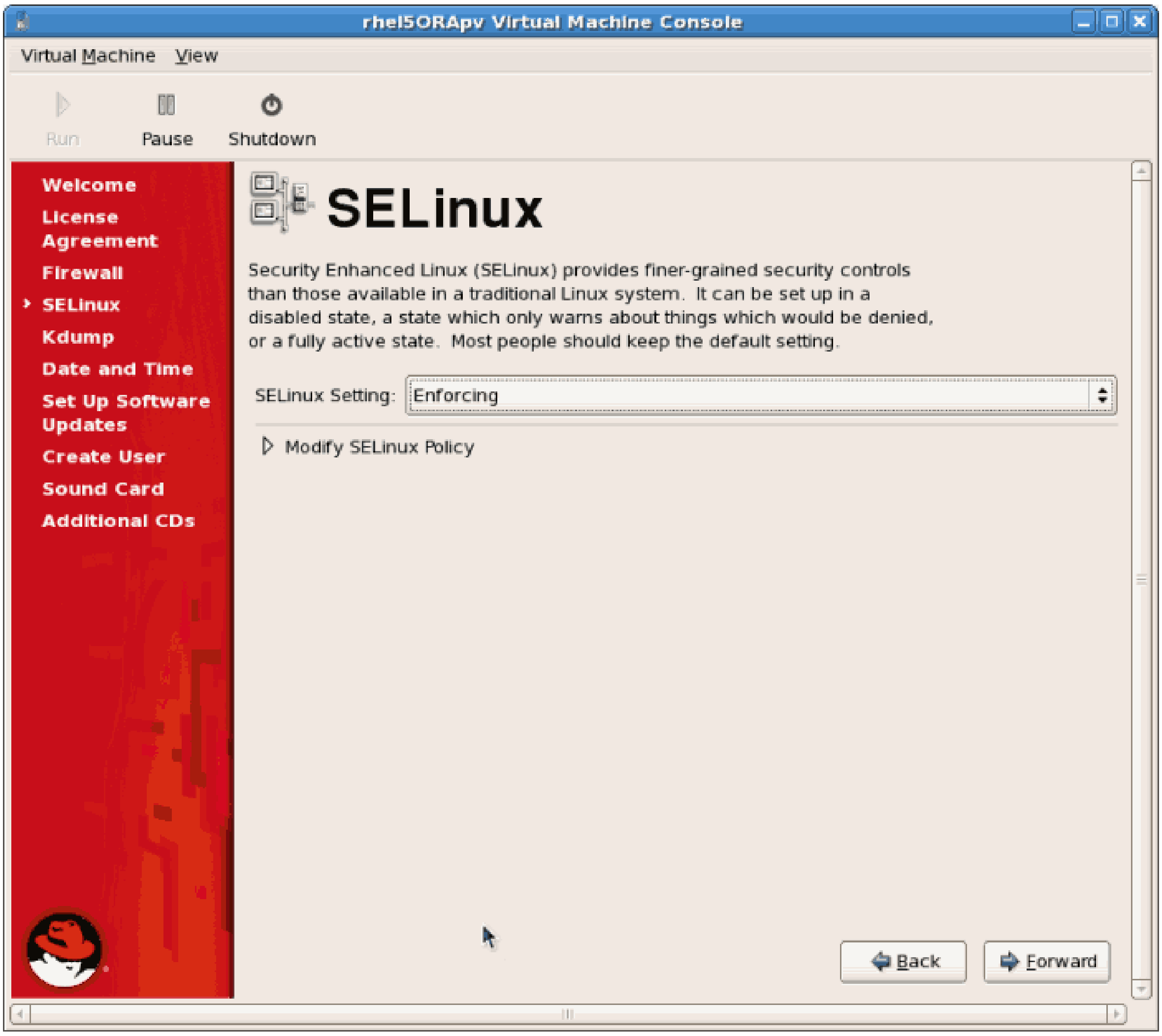
16. Read and agree to the license agreement.

Click **Forward** on the license agreement windows.

17. Configure the firewall.

Click **Forward** to continue.

18. Configure SELinux. It is strongly recommended you run SELinux in **enforcing mode**. You can choose to either run SELinux in permissive mode or completely disable it.

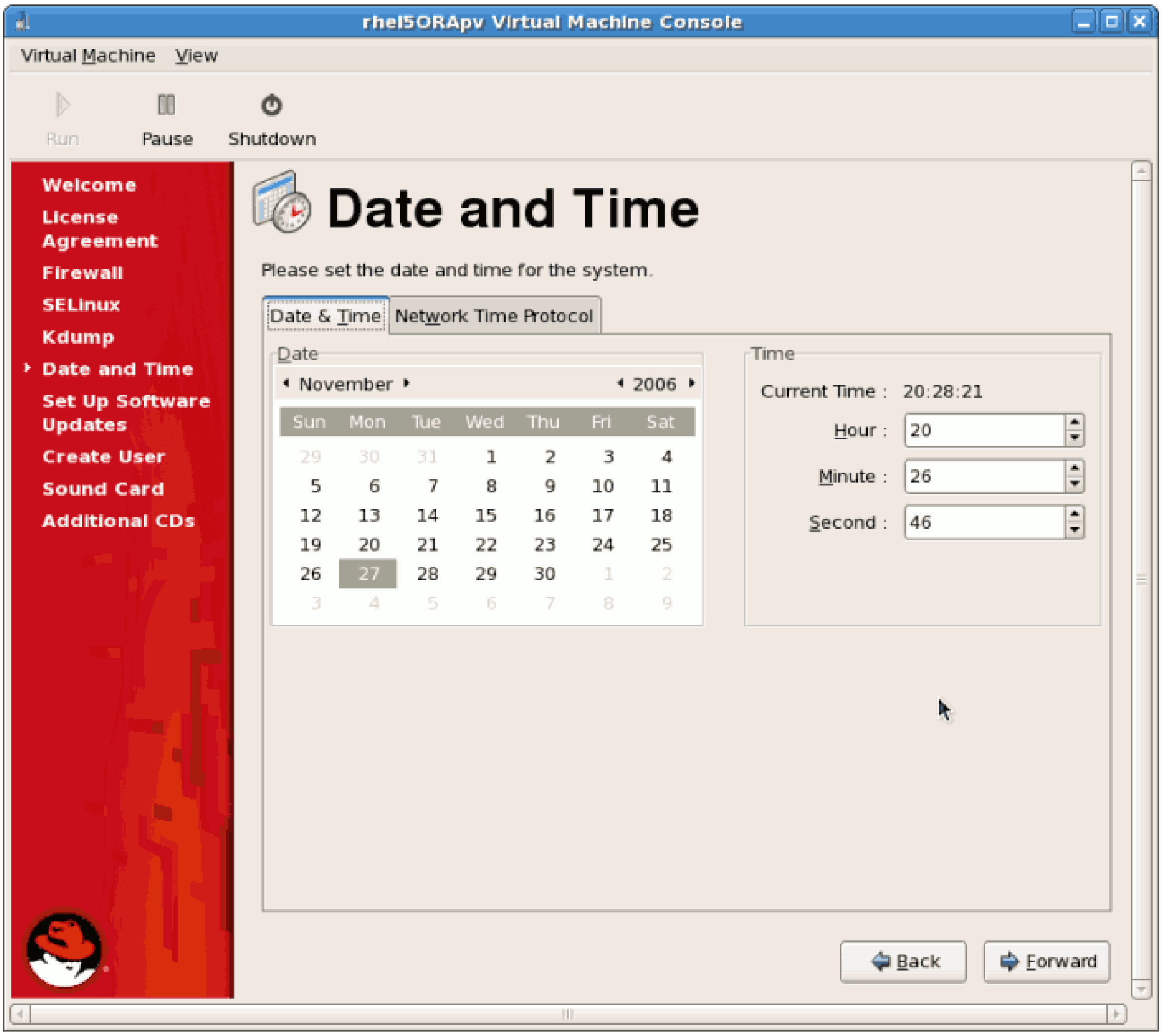


Click **Forward** to continue.

19. Enable **kdump** if necessary.

Click **Forward** to continue.

20. Confirm time and date are set correctly for your guest. If you install a para-virtualized guest time and date should sync with the hypervisor.

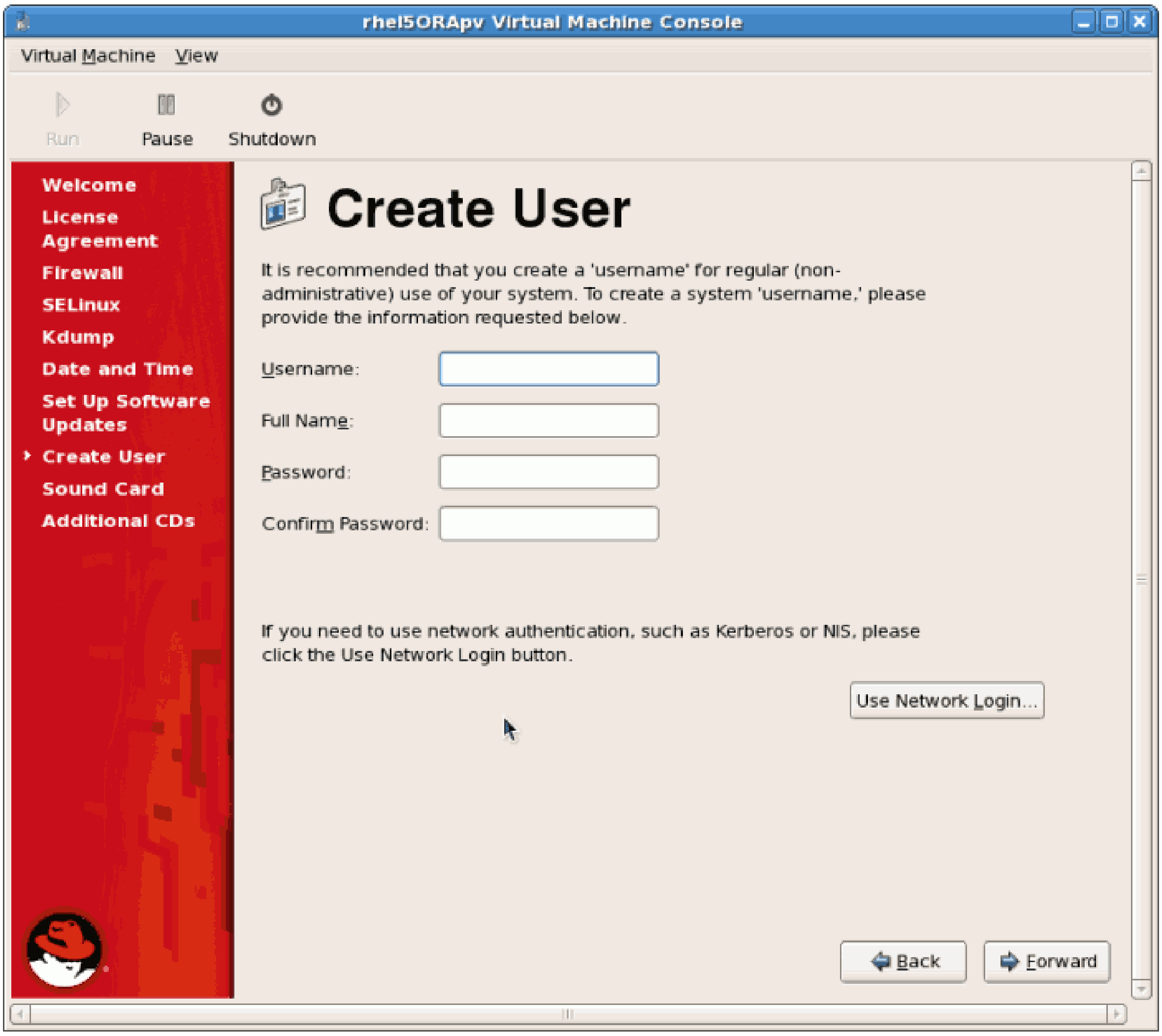


Click **Forward** to continue.

21. Set up software updates. If you have a Red Hat Network subscription or want to trial one use the screen below to register your newly installed guest in RHN.

Click **Forward** to continue.

22. Create a non root user account. It is advised to create a non root user for normal usage and enhanced security. Enter the Username, Name and password.

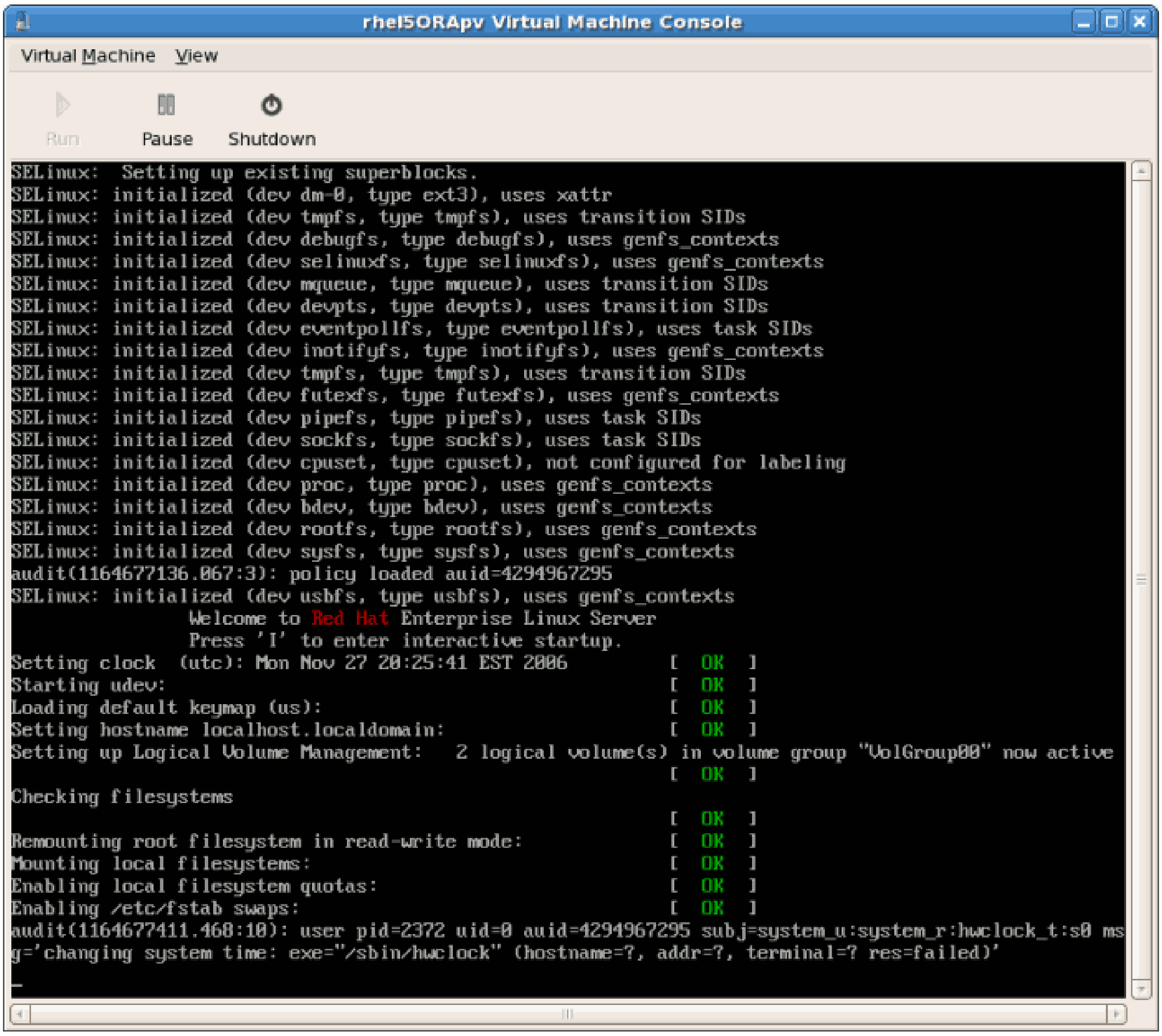


Click the **Forward** button.

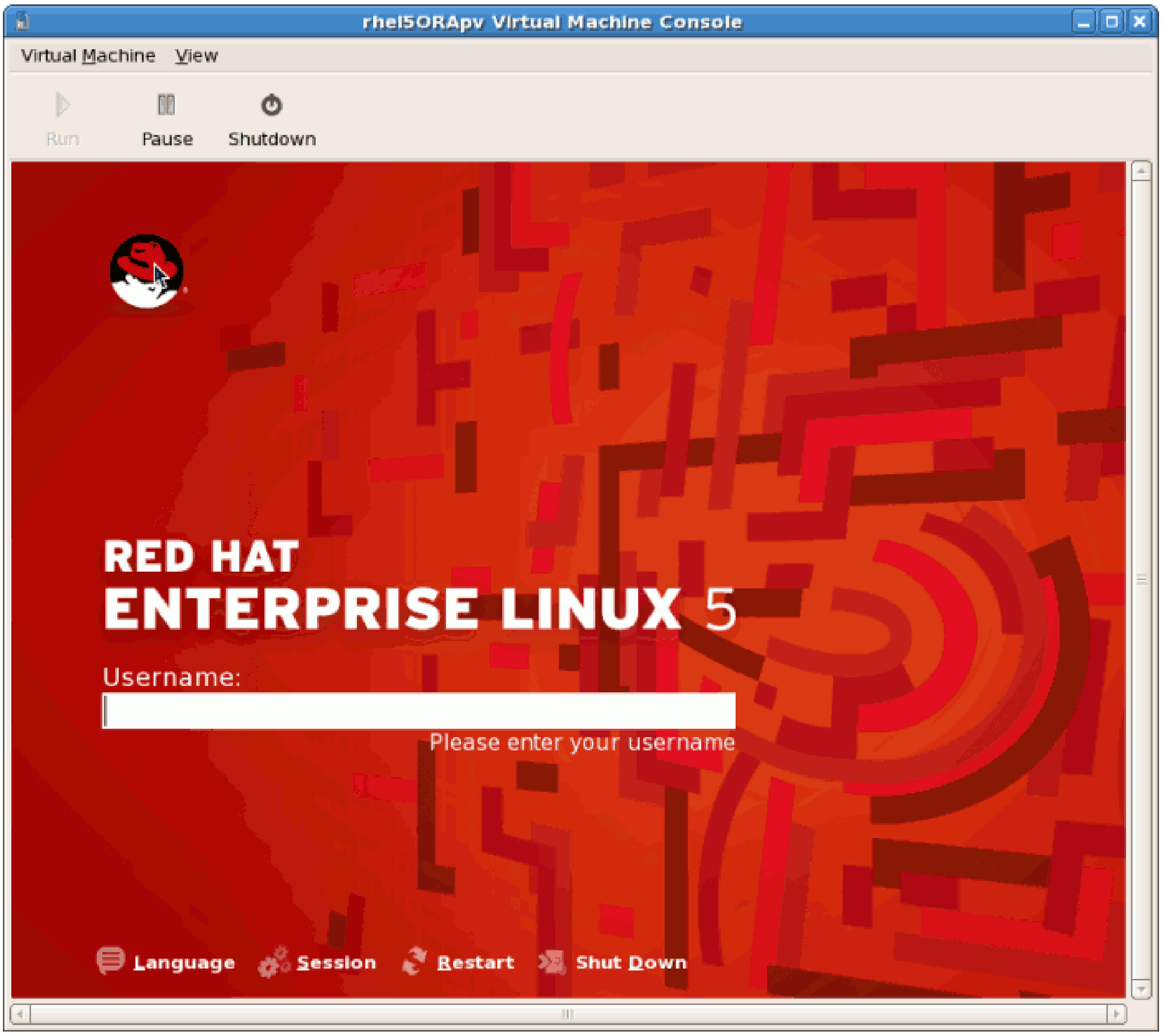
23. If a sound device is detected and you require sound, calibrate it. Complete the process and click **Forward**.

24. You can to install any additional software packages from CD you could do so on this screen. It it often more efficient to not install any additional software at this point but add it later using yum. Click **Finish**.

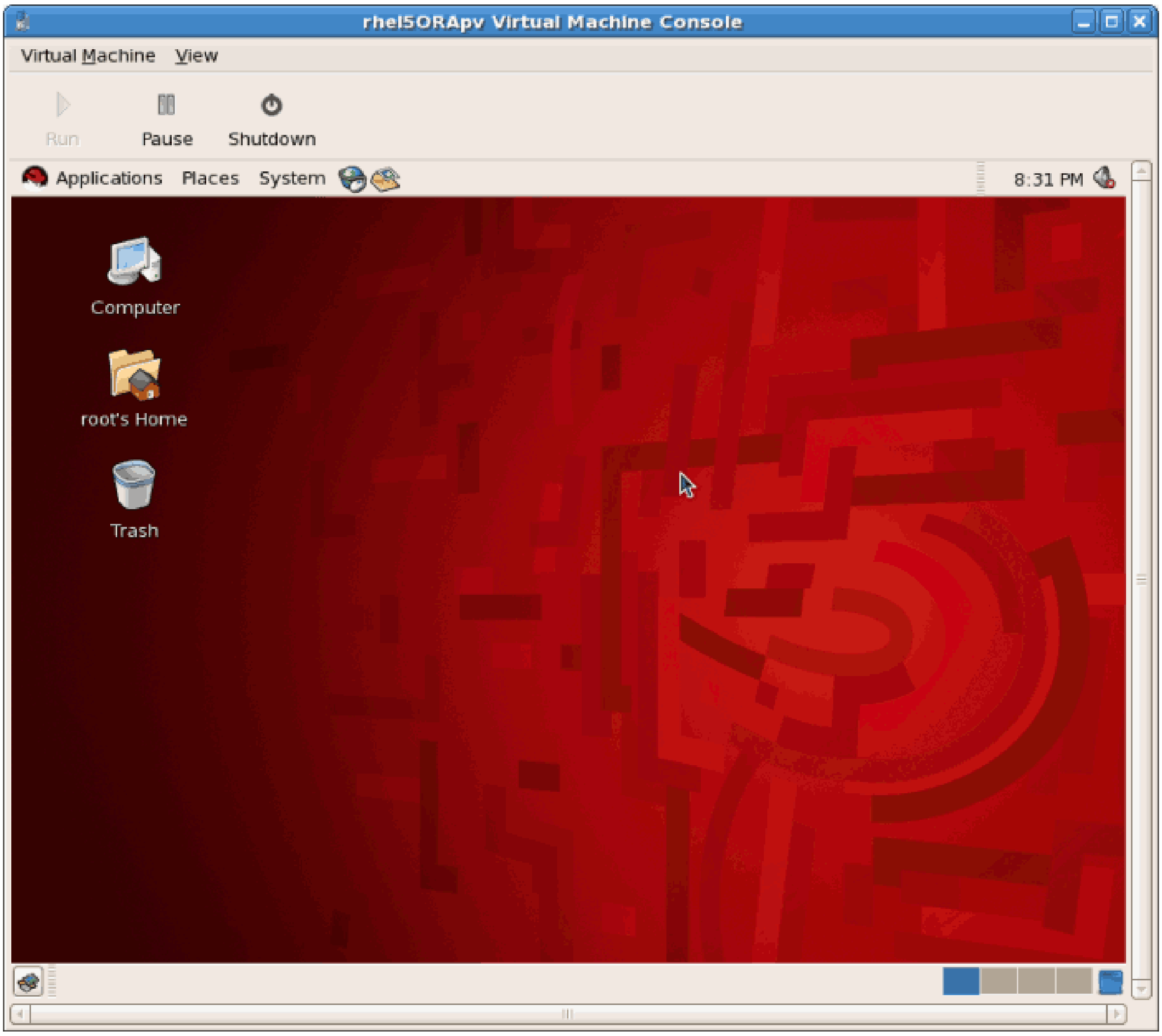
25. The guest now configure any settings you changed and continues the boot process.



26. The Red Hat Enterprise Linux 5 login screen displays. Log in using the username created in the previous steps.



27. You have now successfully installed a para-virtualized Red Hat Enterprise Linux guest.



7.2. Installing Red Hat Enterprise Linux as a fully virtualized guest

This section covers installing a fully virtualized Red Hat Enterprise Linux 5 guest. The steps that uses the KVM hypervisor requires Red Hat Enterprise Linux 5.4 or newer.

Procedure 7.3. Creating a fully virtualized Red Hat Enterprise Linux 5 guest with virt-manager

1. **Open virt-manager**

   Start **`virt-manager`**. Launch the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu. Alternatively, run the **`virt-manager`** command as root.

2. **Select the hypervisor**
   Select the hypervisor. If installed, select Xen or KVM. For this example, select KVM. Note that presently KVM is named `qemu`.

   Connect to a hypervisor if you have not already done so. Open the **File** menu and select the **Add Connection...** option. Refer to *Section 23.1, “The open connection window”*.

   Once a hypervisor connection is selected the **New** button becomes available. Press the **New** button.

3. **Start the new virtual machine wizard**
   Pressing the **New** button starts the virtual machine creation wizard.

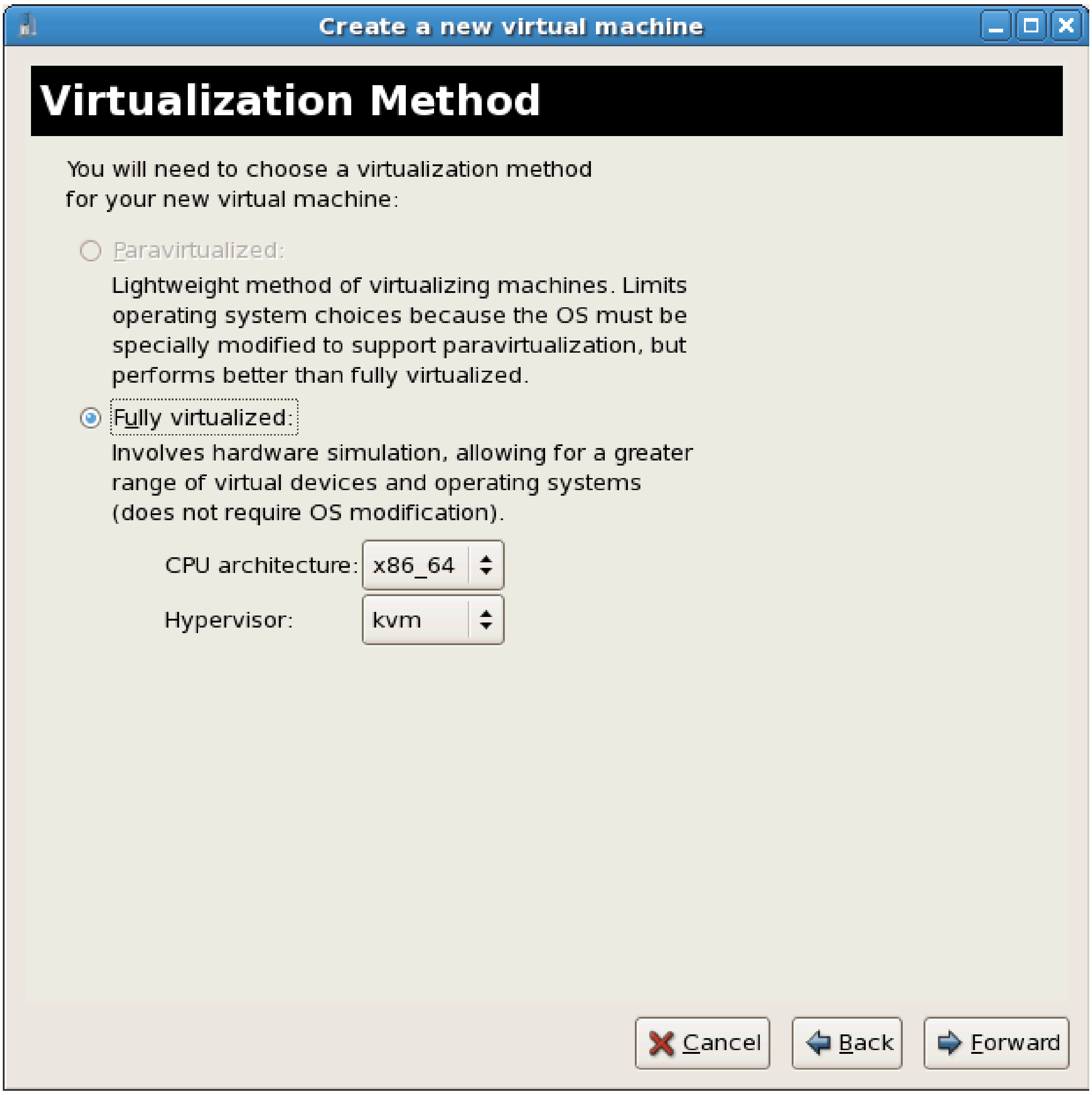
   Press **Forward** to continue.

4. **Name the virtual machine**

   Provide a name for your virtualized guest. Punctuation and whitespace characters are not permitted.

   Press **Forward** to continue.

5. **Choose a virtualization method**
   Choose the virtualization method for the virtualized guest. Note you can only select an installed virtualization method. If you selected KVM or Xen earlier (*Step 4*) you must use the hypervisor you selected. This example uses the KVM hypervisor.



   Press **Forward** to continue.

6. **Select the installation method**

   For all versions of Windows you must use **local install media**, either an ISO image or physical optical media.

   PXE may be used if you have a PXE server configured for Windows network installation. PXE Windows installation is not covered by this guide.

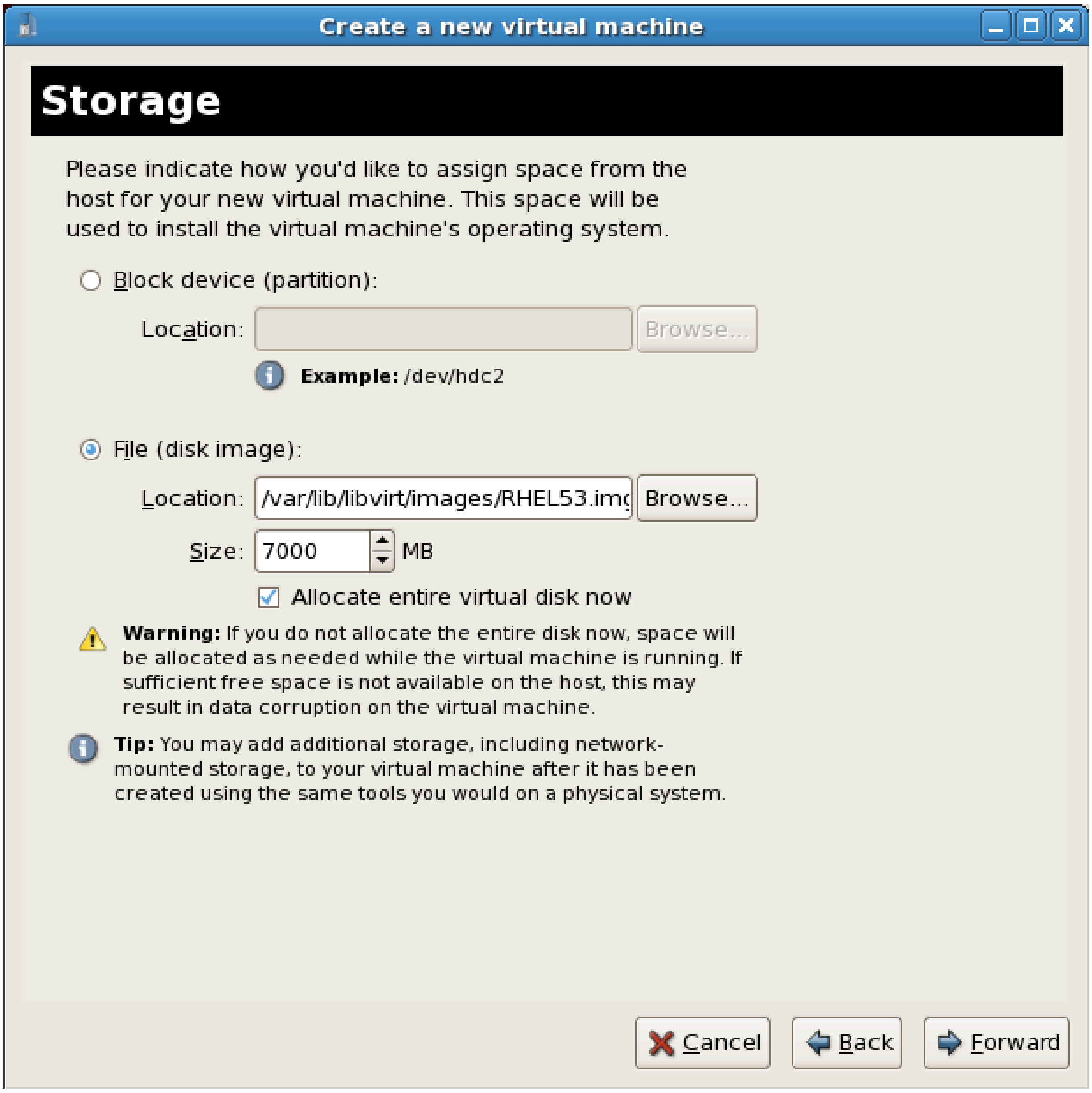
   Set **OS Type** to **Windows** and **OS Variant** to **Microsoft Windows 2008** as shown in the screenshot.

   Press **Forward** to continue.

7. **Locate installation media**
   Select ISO image location or CD-ROM or DVD device. This example uses an ISO file image of the Red Hat Enterprise Linux installation DVD.

8. **Storage setup**

   Assign a physical storage device (**Block device**) or a file-based image (**File**). File-based images must be stored in the `/var/lib/libvirt/images/` directory. Assign sufficient storage for your virtualized guest. Assign sufficient space for your virtualized guest and any application it requires.



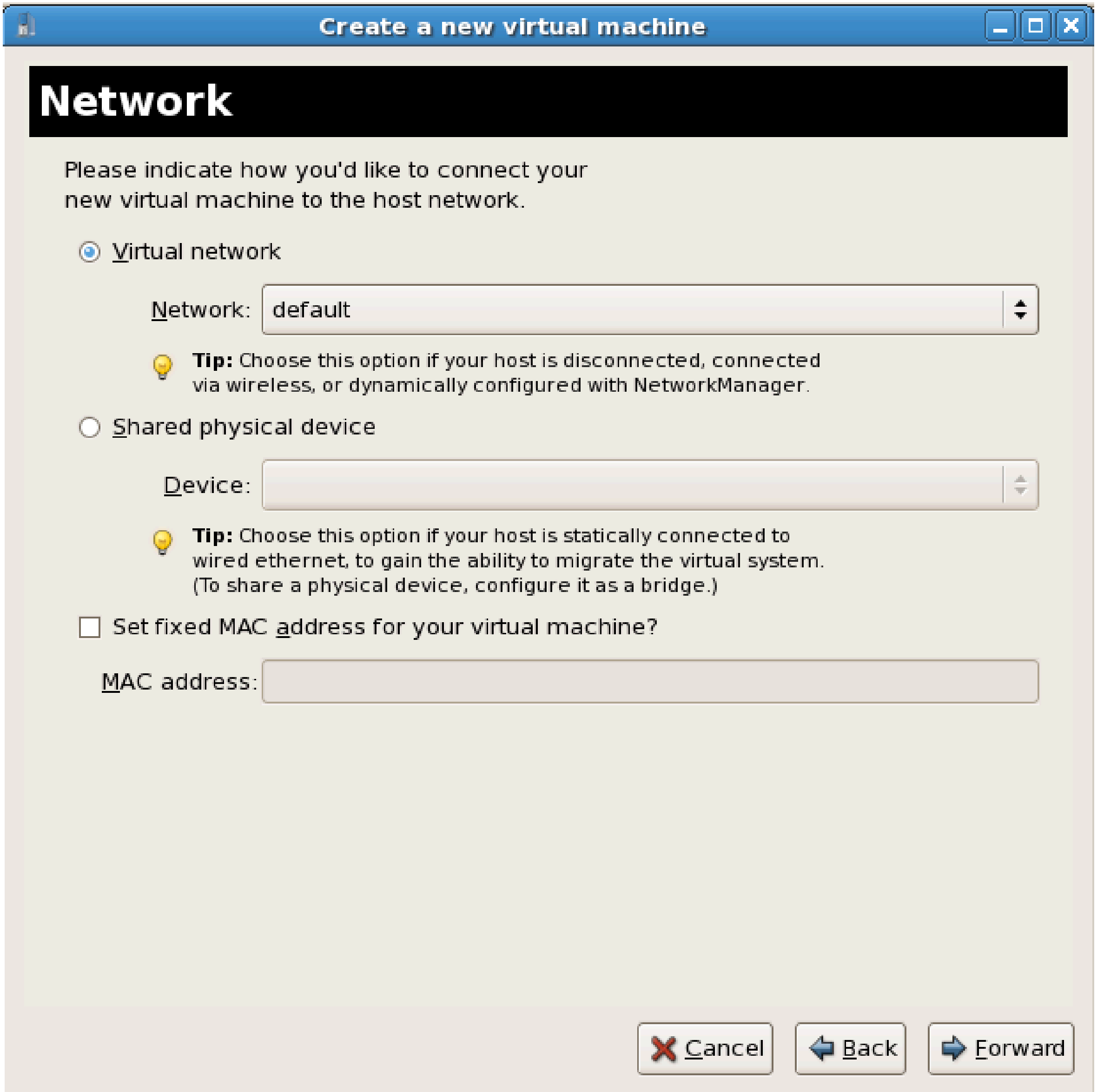
   Press **Forward** to continue.

> **To migrating this guest**
>
> Live and offline migrations require guests to be installed on shared network storage. For information on setting up shared storage for guests refer to *Chapter 9, Shared storage and virtualization*.

9. **Network setup**
   Select either **Virtual network** or **Shared physical device**.

   The virtual network option uses Network Address Translation (NAT) to share the default network device with the virtualized guest. Use the virtual network option for wireless networks.

   The shared physical device option uses a network bond to give the virtualized guest full access to a network device.

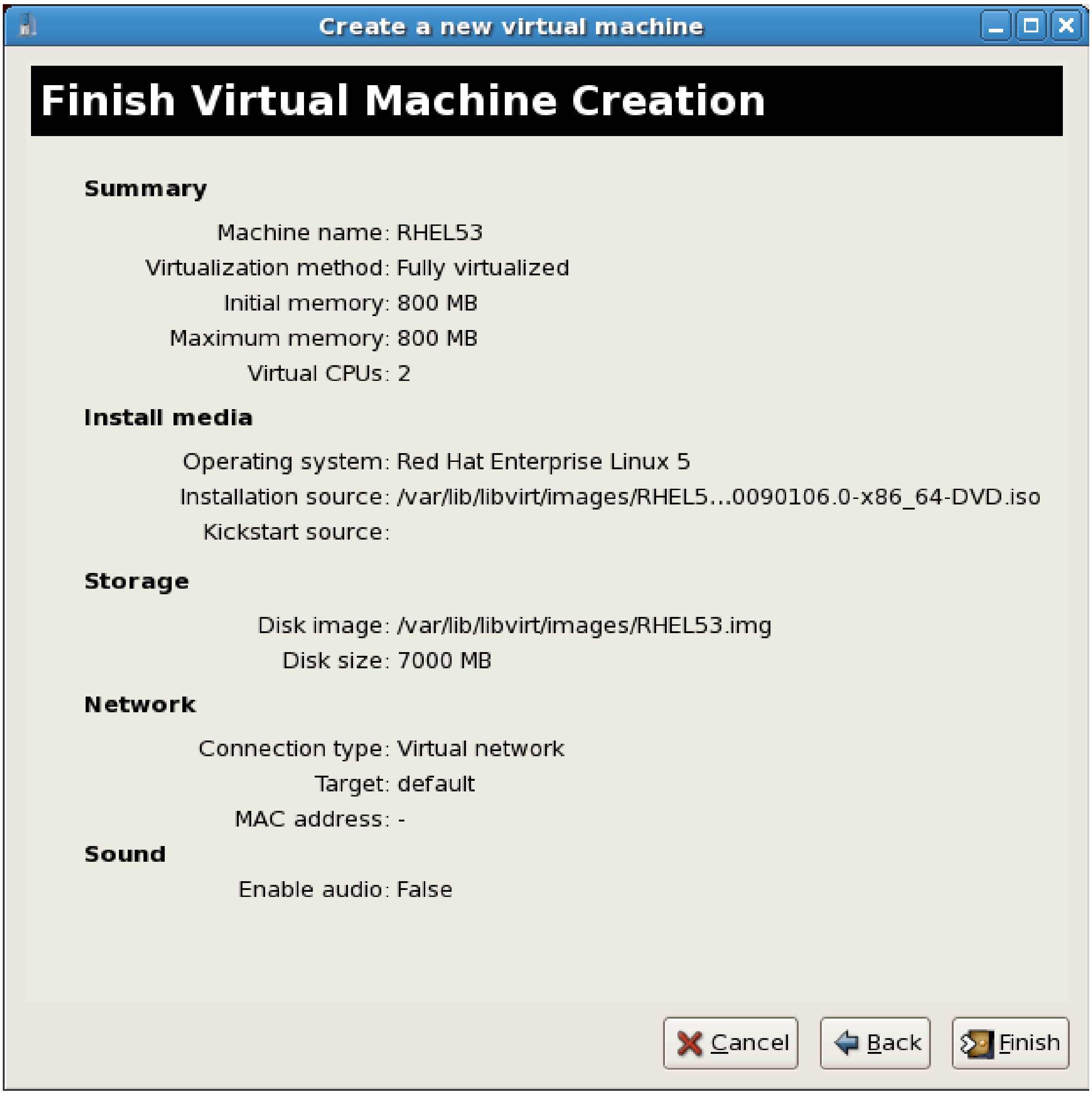


   Press **Forward** to continue.

10. **Memory and CPU allocation**

    The Allocate memory and CPU window displays. Choose appropriate values for the virtualized CPUs and RAM allocation. These values affect the host's and guest's performance.

11. **Verify and start guest installation**
    Verify the configuration.



    Press **Finish** to start the guest installation procedure.

12. **Installing Red Hat Enterprise Linux**
    Complete the Red Hat Enterprise Linux 5 installation sequence. The installation sequence is covered by the *Installation Guide*, refer to *Red Hat Documentation*[1] for the Red Hat Enterprise Linux *Installation Guide*.

A fully virtualized Red Hat Enterprise Linux 5 Guest is now installed.

7.3. Installing Windows XP as a fully virtualized guest

Windows XP can be installed as a fully virtualized guest. This section describes how to install Windows XP as a fully virtualized guest on Red Hat Enterprise Linux.

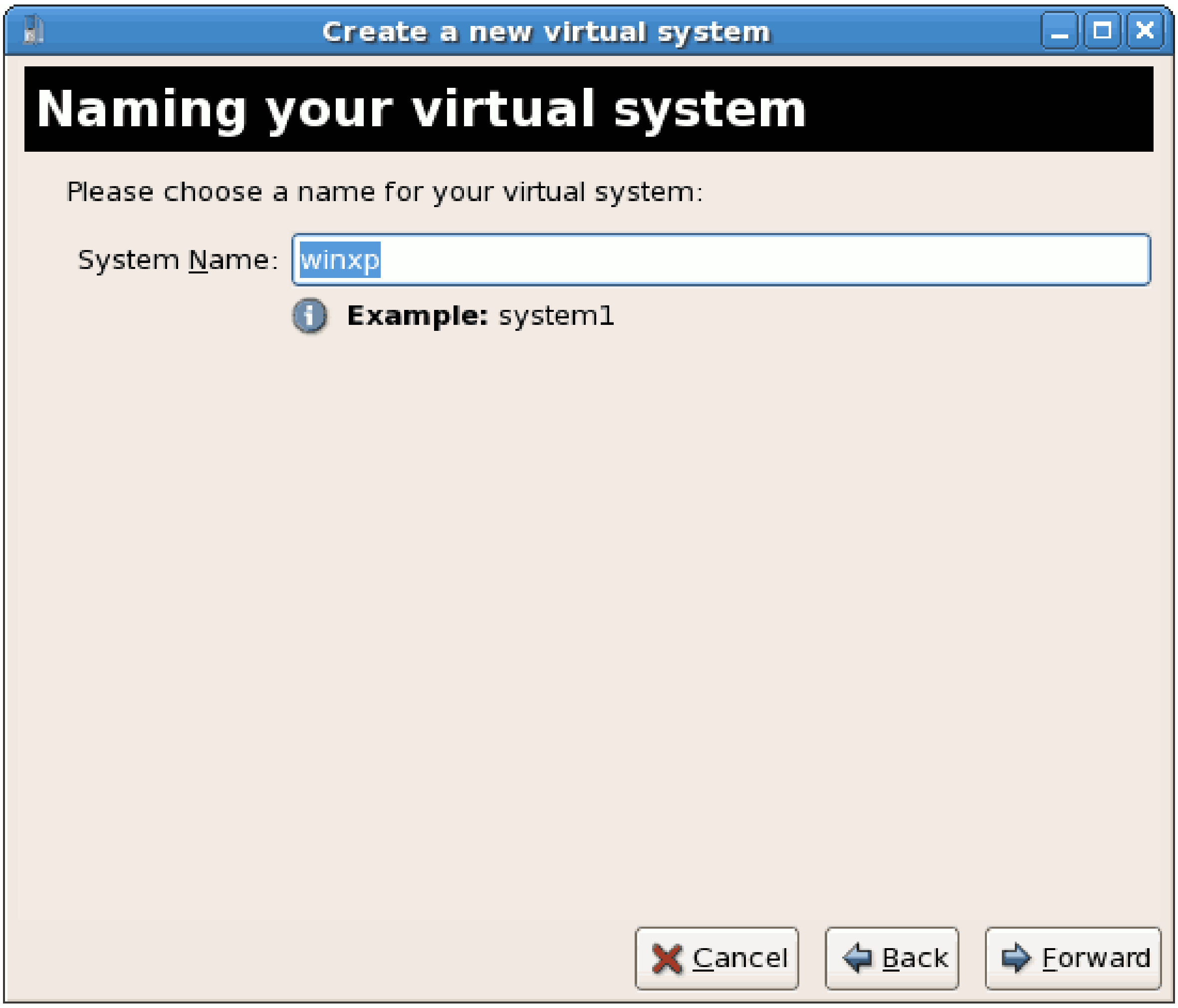
Before commencing this procedure ensure you must have root access.

Itanium® support

Presently, Red Hat Enterprise Linux hosts on the Itanium® architecture does not support fully virtualized Windows XP guests. Only Windows Server 2003 for Itanium-based Systems is supported for Itanium systems.

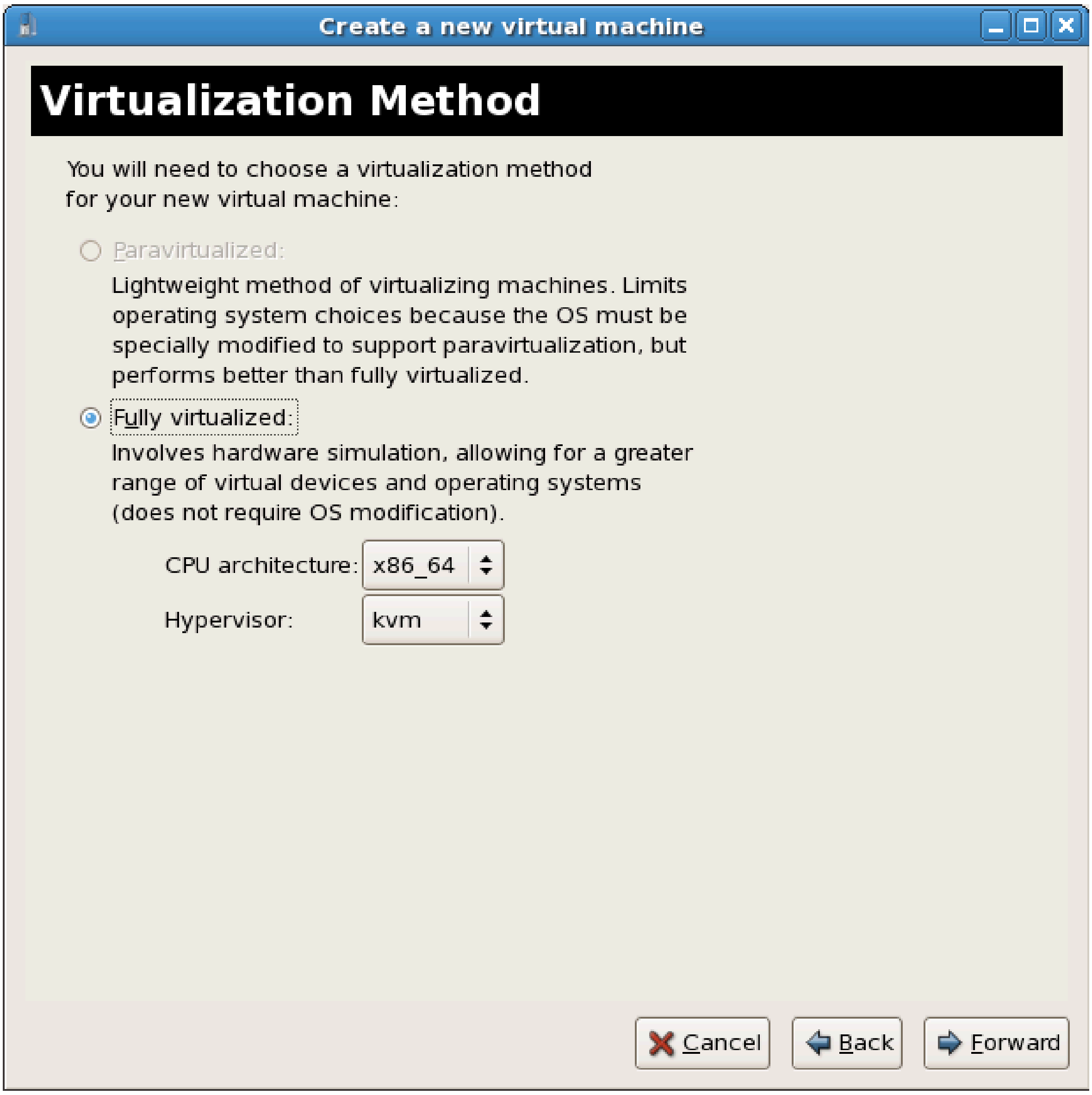
1. **Starting virt-manager**
   Open **Applications > System Tools > Virtual Machine Manager**. Open a connection to the host (click **File > Open Connection**). Click the **New** button to create a new virtual machine.

2. **Naming your virtual system**
   Enter the **System Name** and click the **Forward** button.



3. **Choosing a virtualization method**
   If you selected KVM or Xen earlier (step *Step 1*) you must use the hypervisor you selected. This example uses the KVM hypervisor.

   Windows can only be installed using full virtualization.



4. **Choosing an installation method**
   This screen enables you to specify the installation method and the type of operating system.

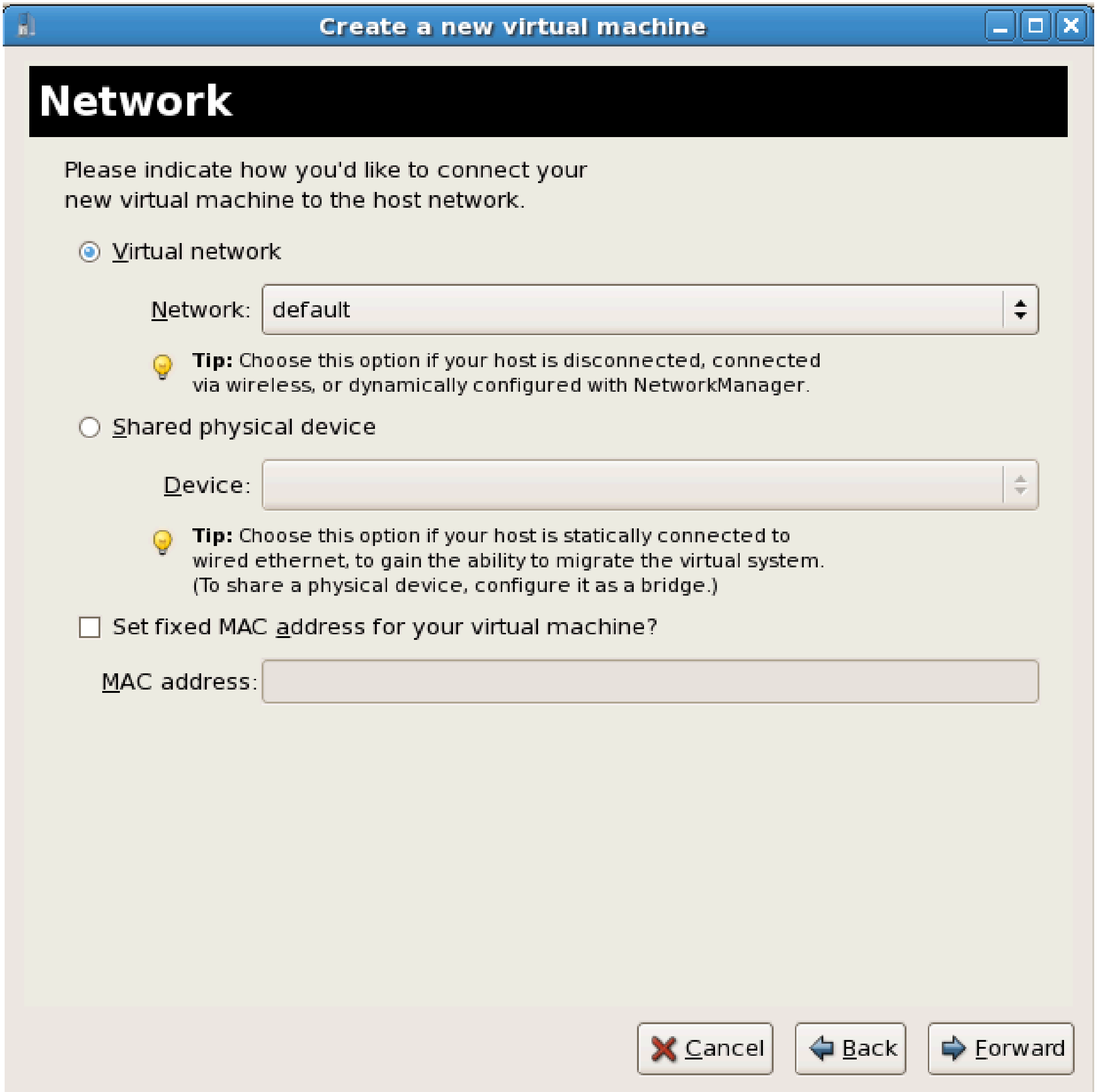
5. The **Assigning storage space** window displays. Choose a disk partition, LUN or create a file based image for the guest storage.

   The convention for file based images in Red Hat Enterprise Linux 5 all file based guest images are in the **`/var/lib/libvirt/images/`** directory. Other directory locations for file based images are prohibited by SELinux. If you run SELinux in enforcing mode, refer to *Section 11.1, "SELinux and virtualization"* for more information on installing guests.

6. **Network setup**
   Select either **Virtual network** or **Shared physical device**.

   The virtual network option uses Network Address Translation (NAT) to share the default network device with the virtualized guest. Use the virtual network option for wireless networks.

   The shared physical device option uses a network bond to give the virtualized guest full access to a network device.

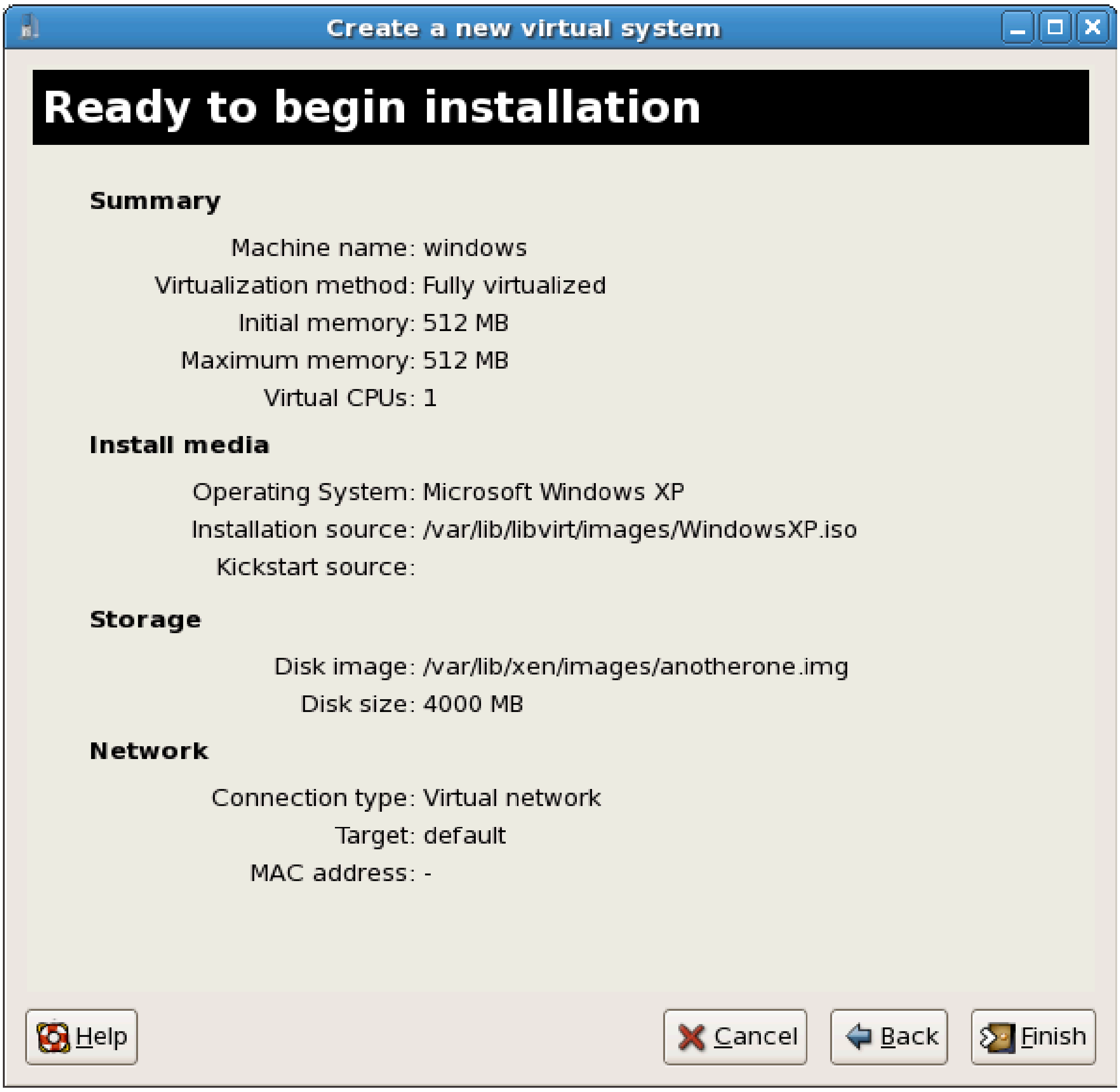


   Press **Forward** to continue.

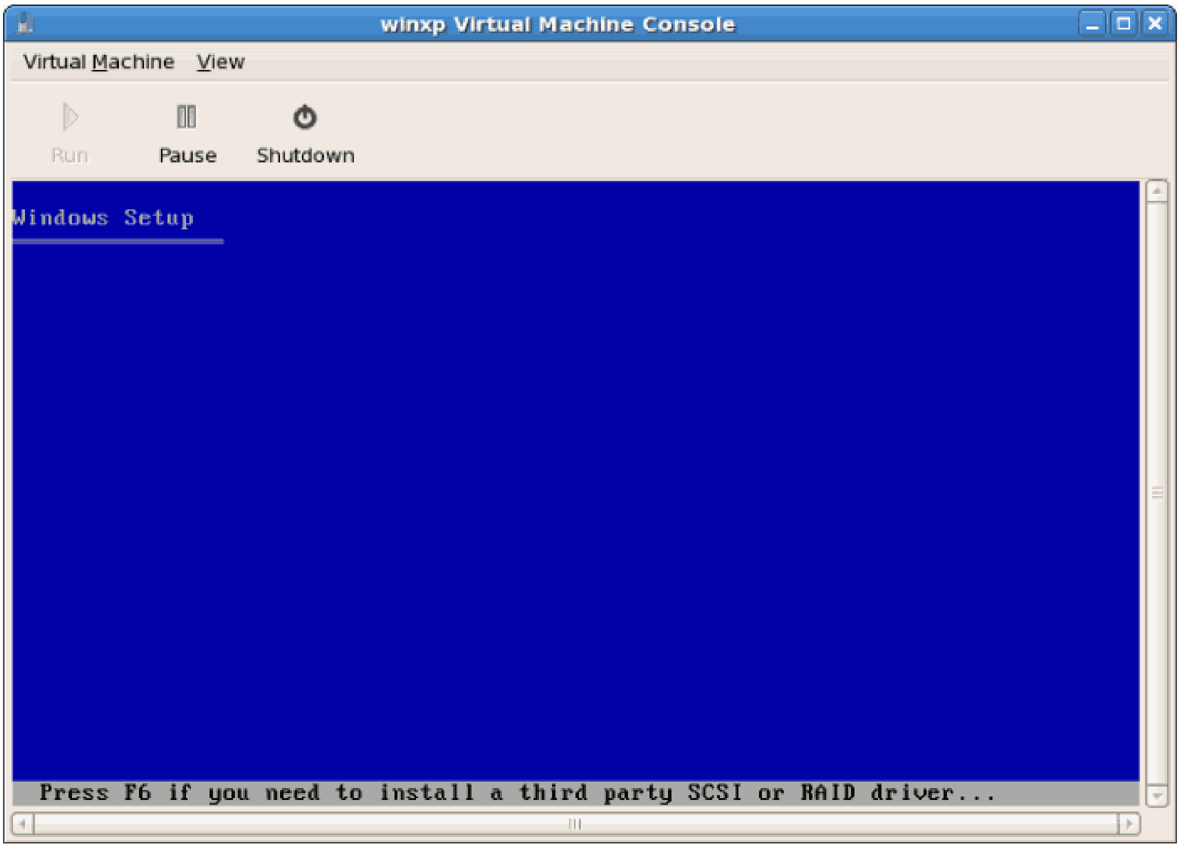
7. The Allocate memory and CPU window displays. Choose appropriate values for the virtualized CPUs and RAM allocation. These values affect the host's and guest's performance.

   Virtualized guests require sufficient physical memory (RAM) to run efficiently and effectively. Choose a memory value which suits your guest operating system and application requirements. Most operating system require at least 512MB of RAM to work responsively. Remember, guests use physical RAM. Running too many guests or leaving insufficient memory for the host system results in significant usage of virtual memory and swapping. Virtual memory is significantly slower causing degraded system performance and responsiveness. Ensure to allocate sufficient memory for all guests and the host to operate effectively.

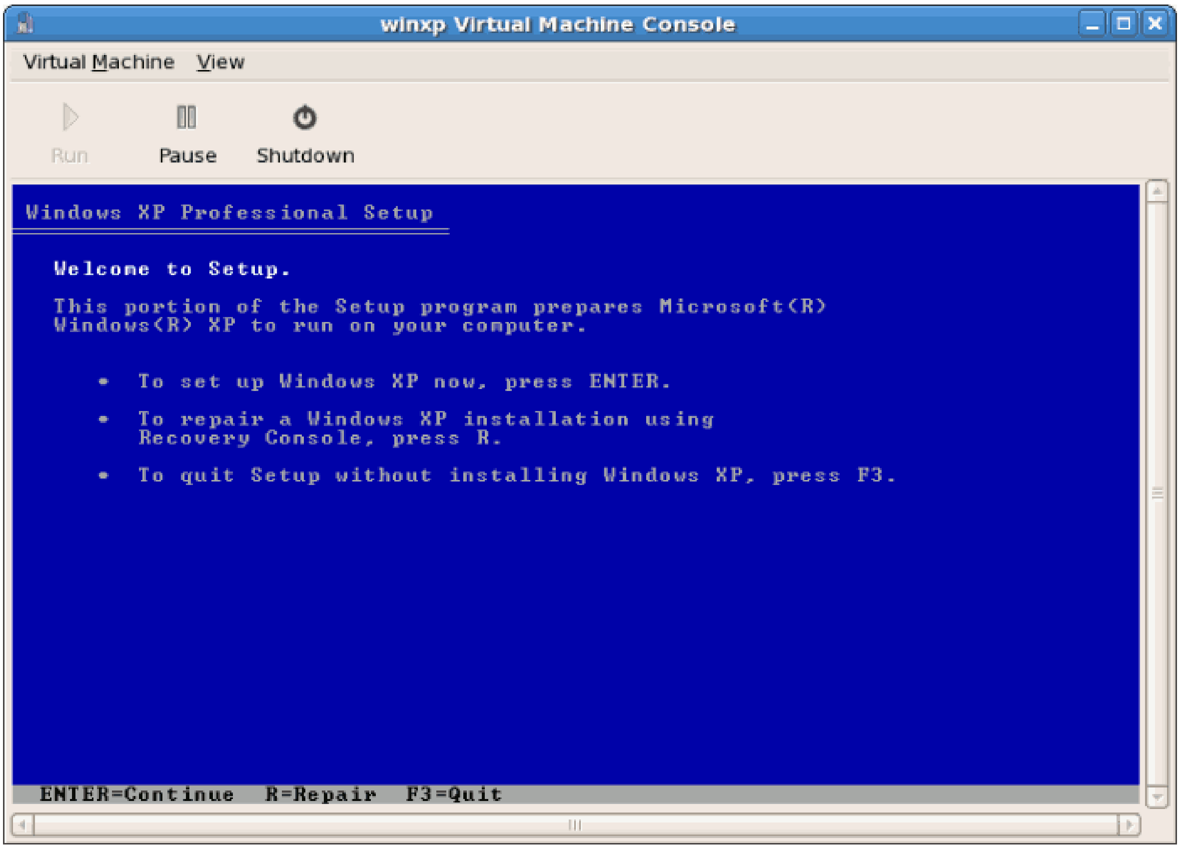
8. Before the installation continues you will see the summary screen. Press **Finish** to proceed to the guest installation:

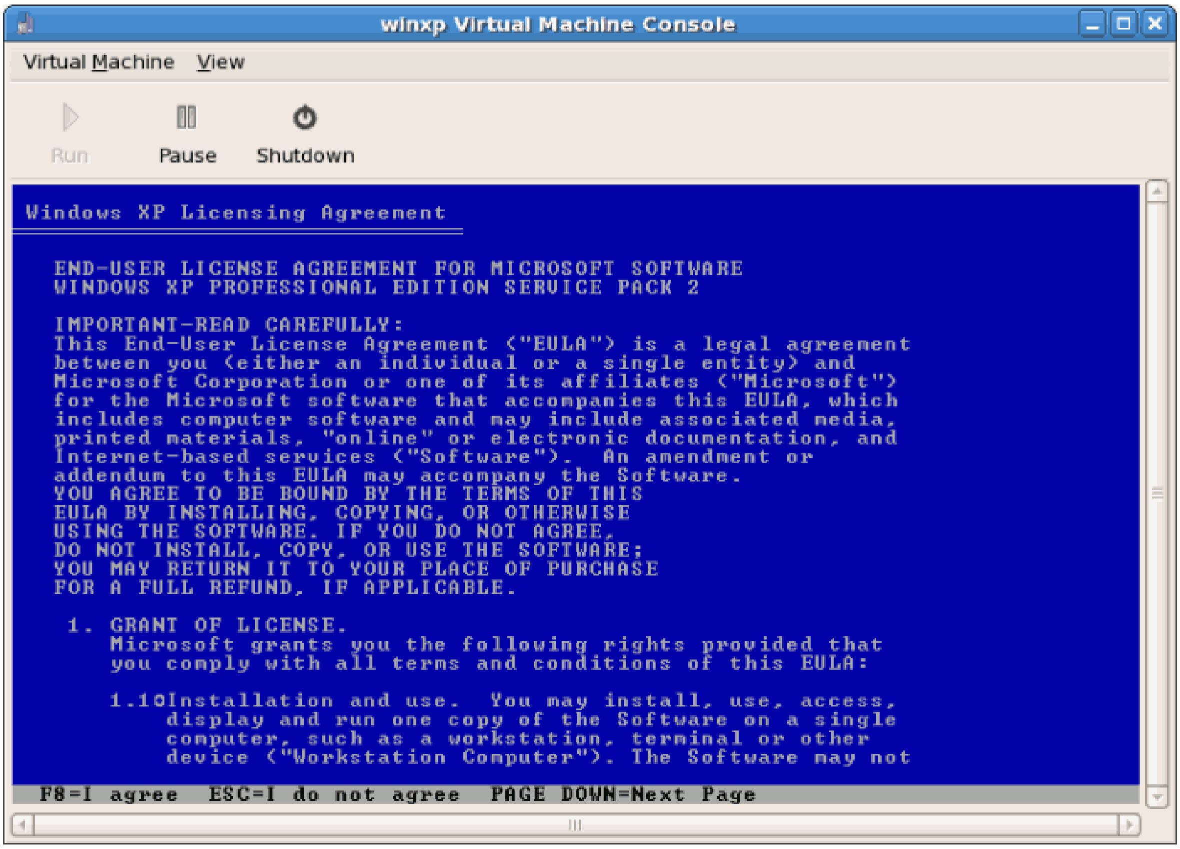


9. You must make a hardware selection so open a console window quickly after the installation starts. Click **Finish** then switch to the **virt-manager** summary window and select your newly started Windows guest. Double click on the system name and the console window opens. Quickly and repeatedly press **F5** to select a new `HAL`, once you get the dialog box in the Windows install select the '`Generic i486 Platform`' tab (scroll through selections with the **`Up`** and **`Down`** arrows.

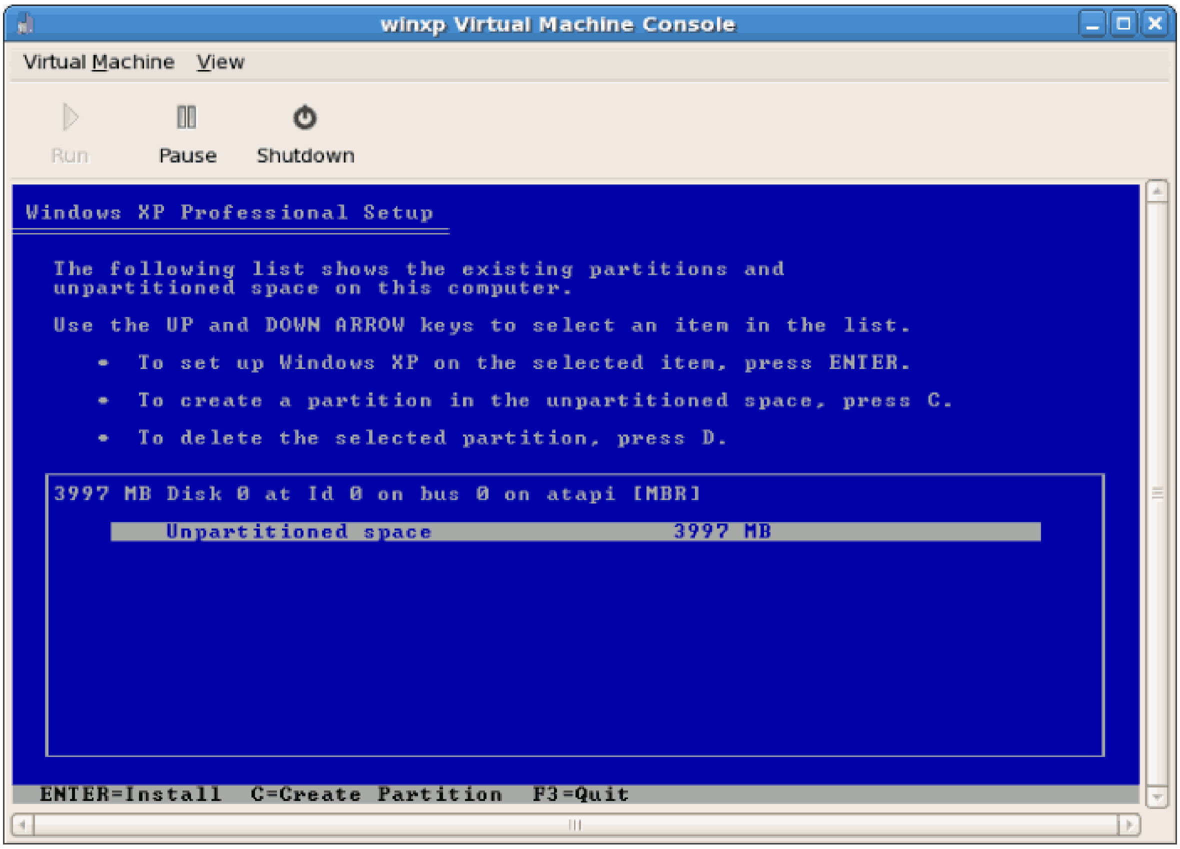


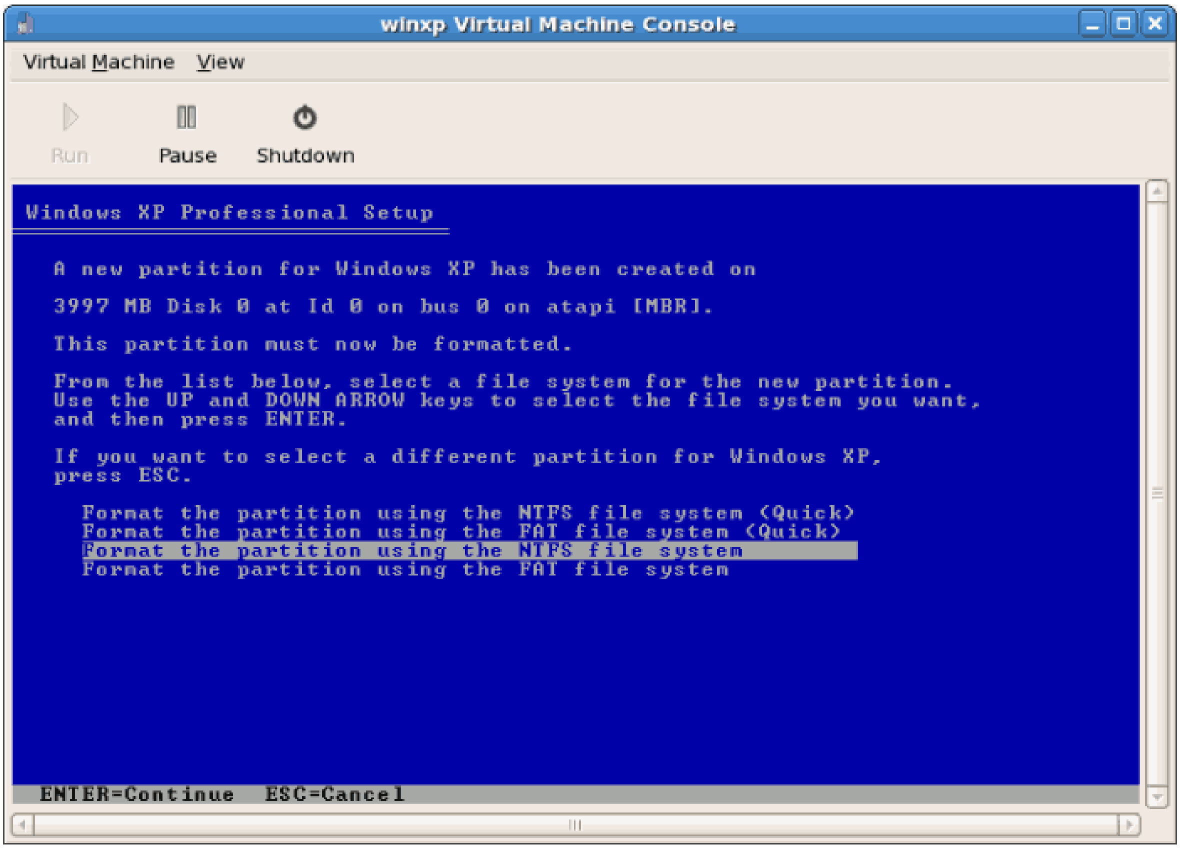
10. The installation continues with the standard Windows installation.



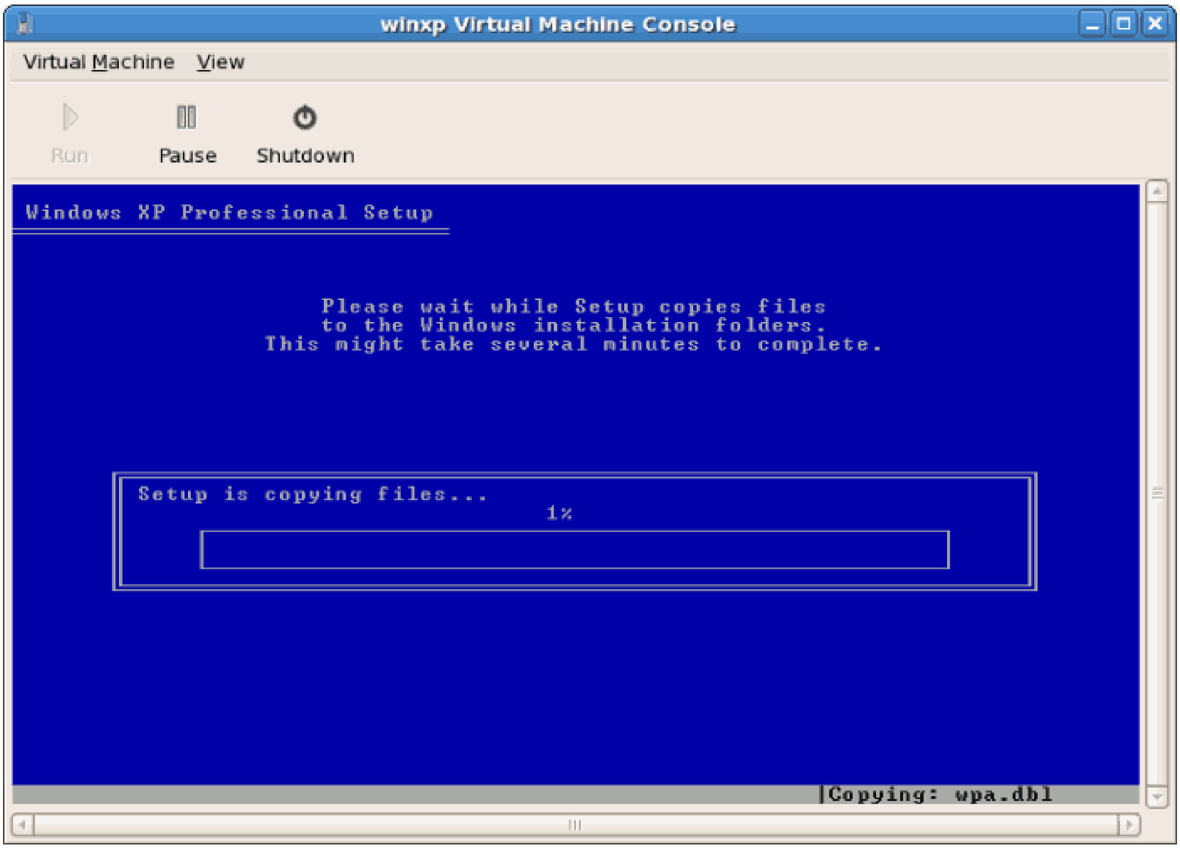


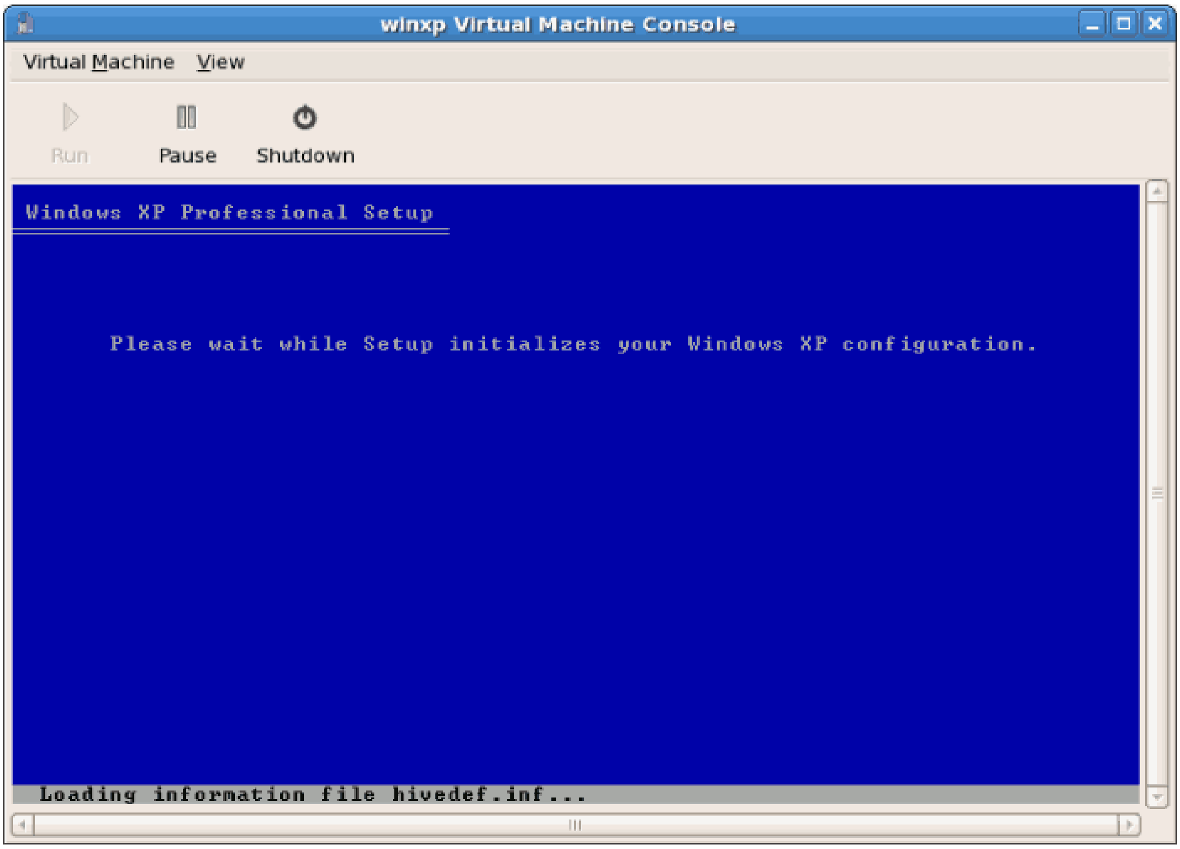
11. Partition the hard drive when prompted.



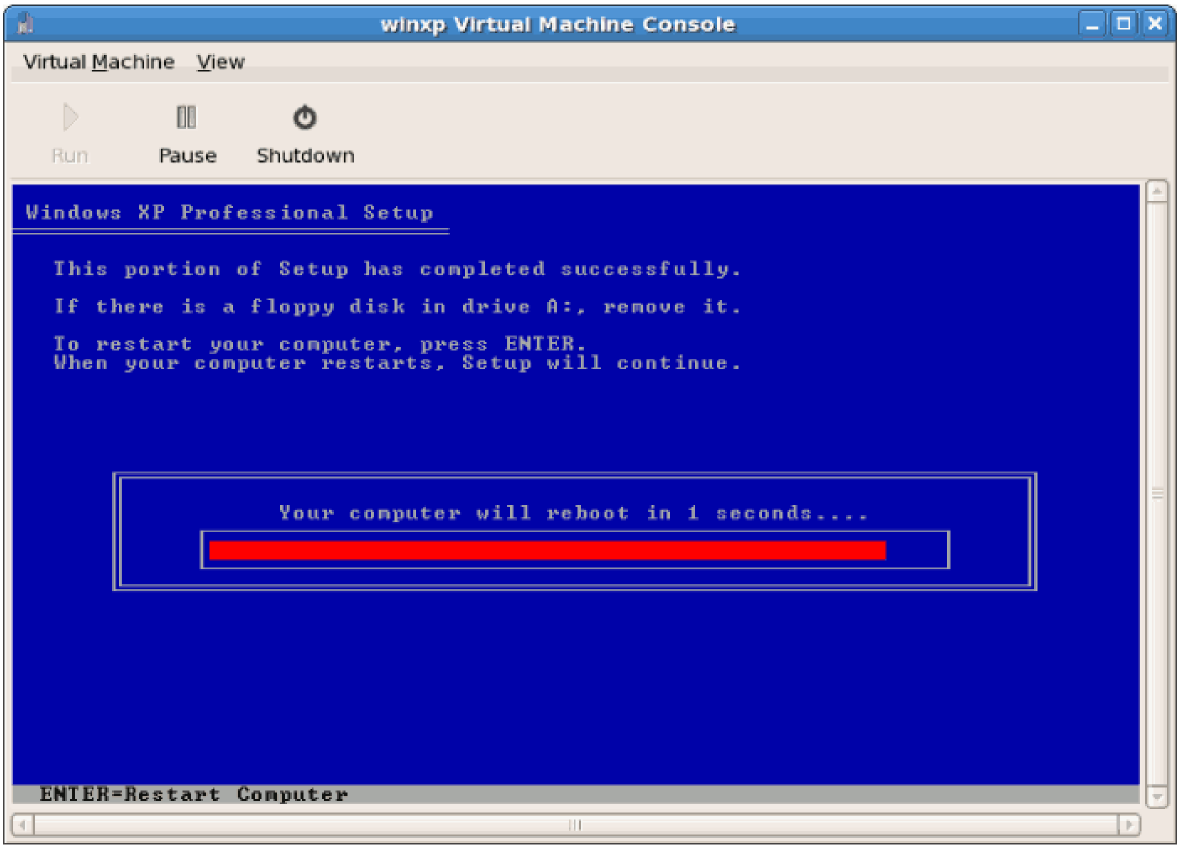


12. After the drive is formatted Windows starts copying the files to the hard drive.





13. The files are copied to the storage device, Windows now reboots.

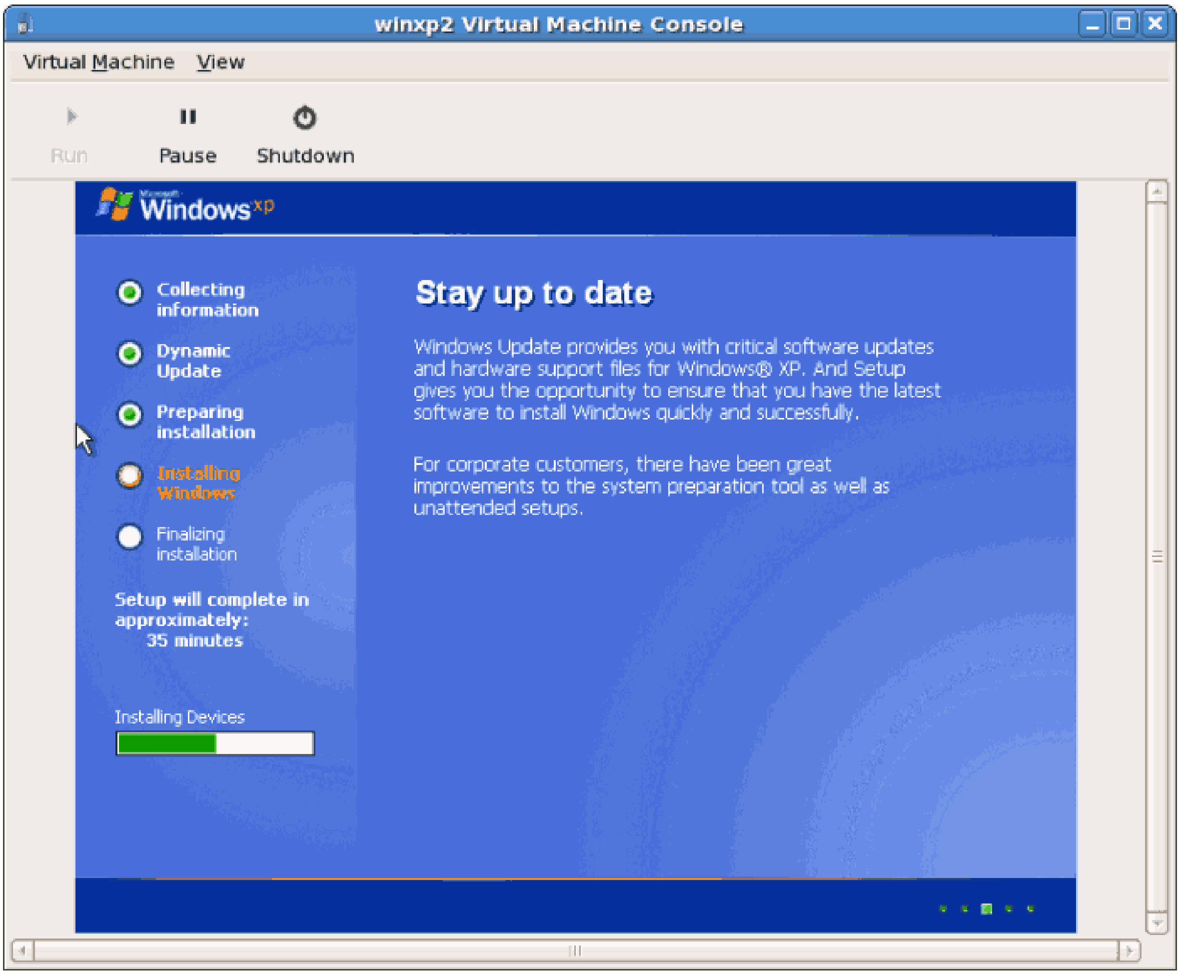


14. Restart your Windows guest:

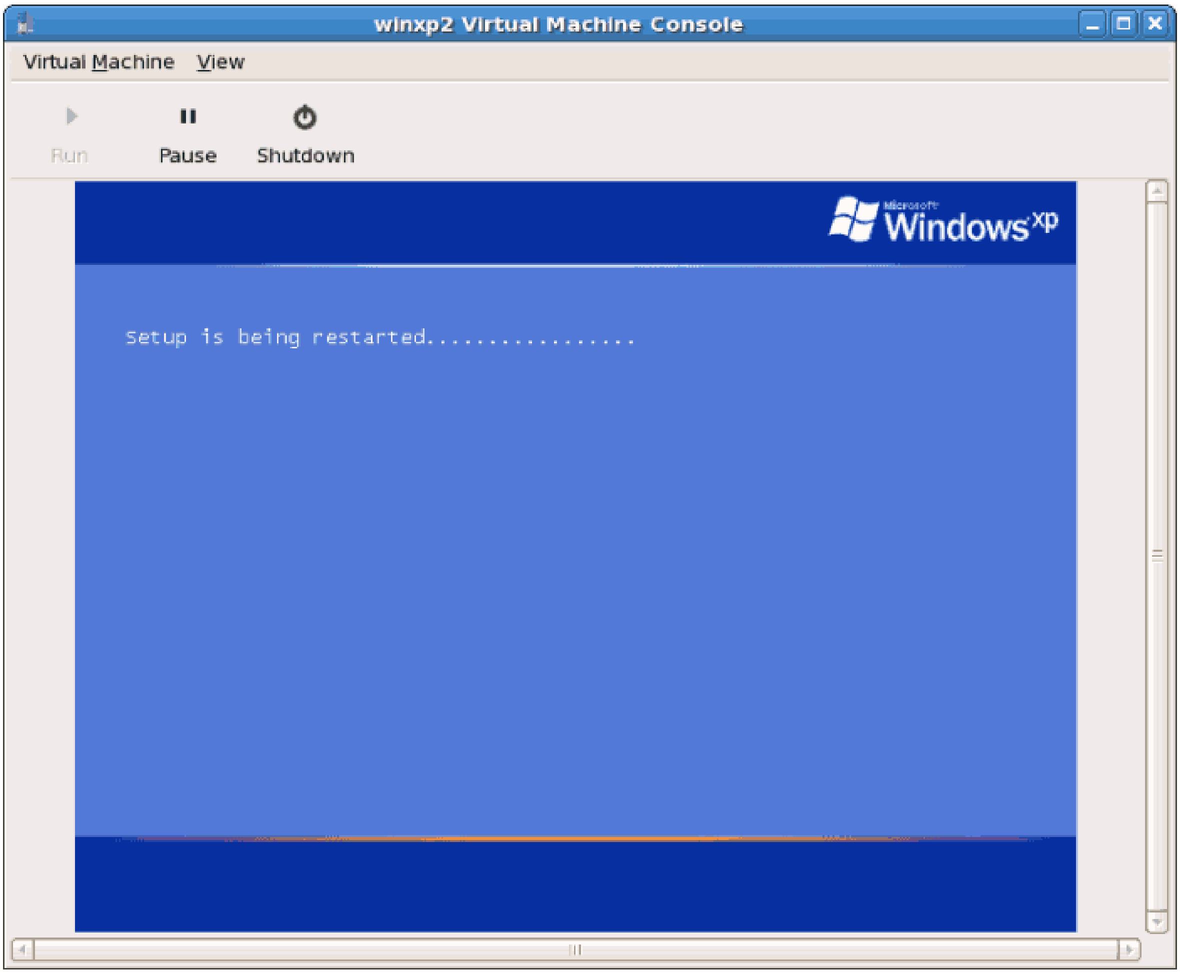
```
# virsh start WindowsGuest
```

Where *WindowsGuest* is the name of your virtual machine.

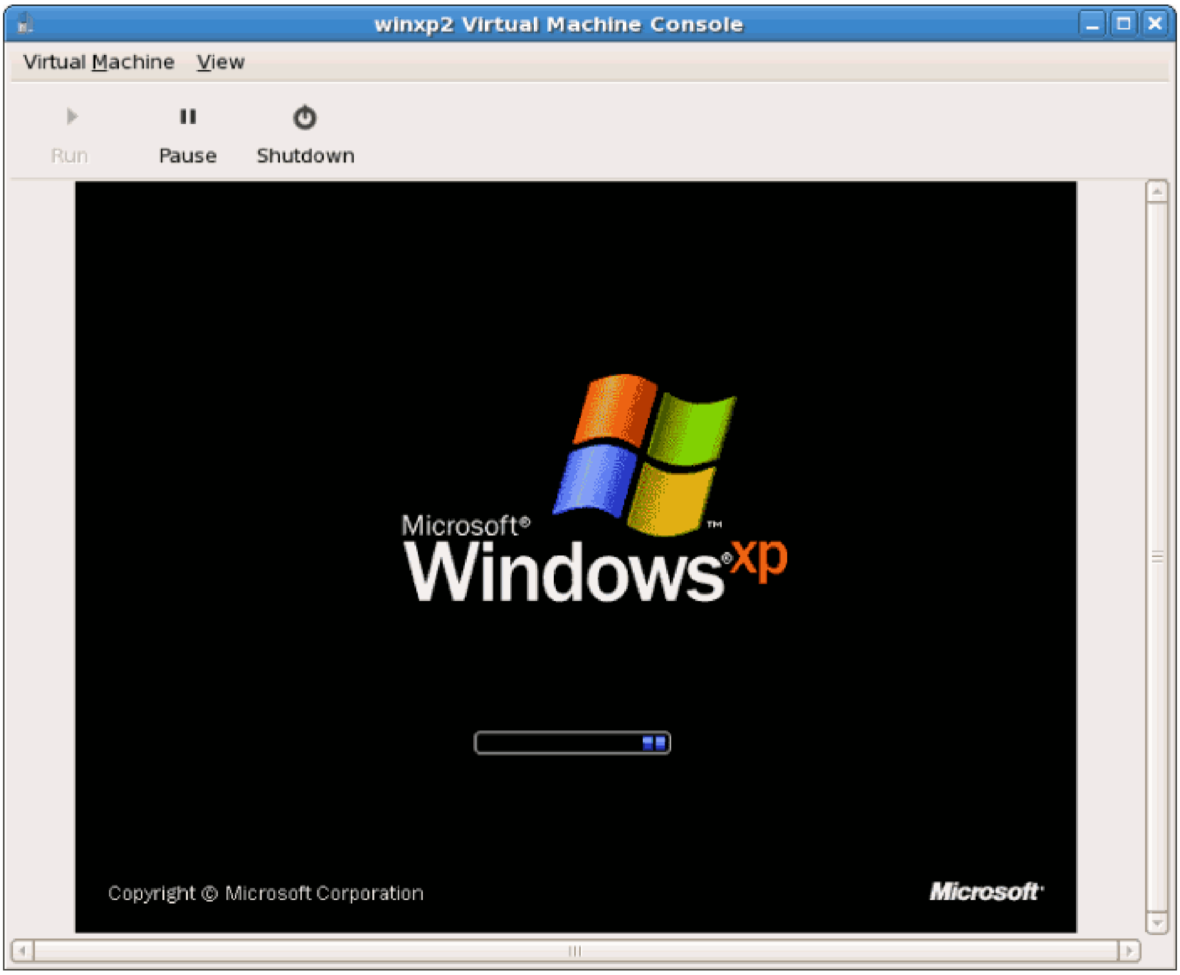
15. When the console window opens, you will see the setup phase of the Windows installation.



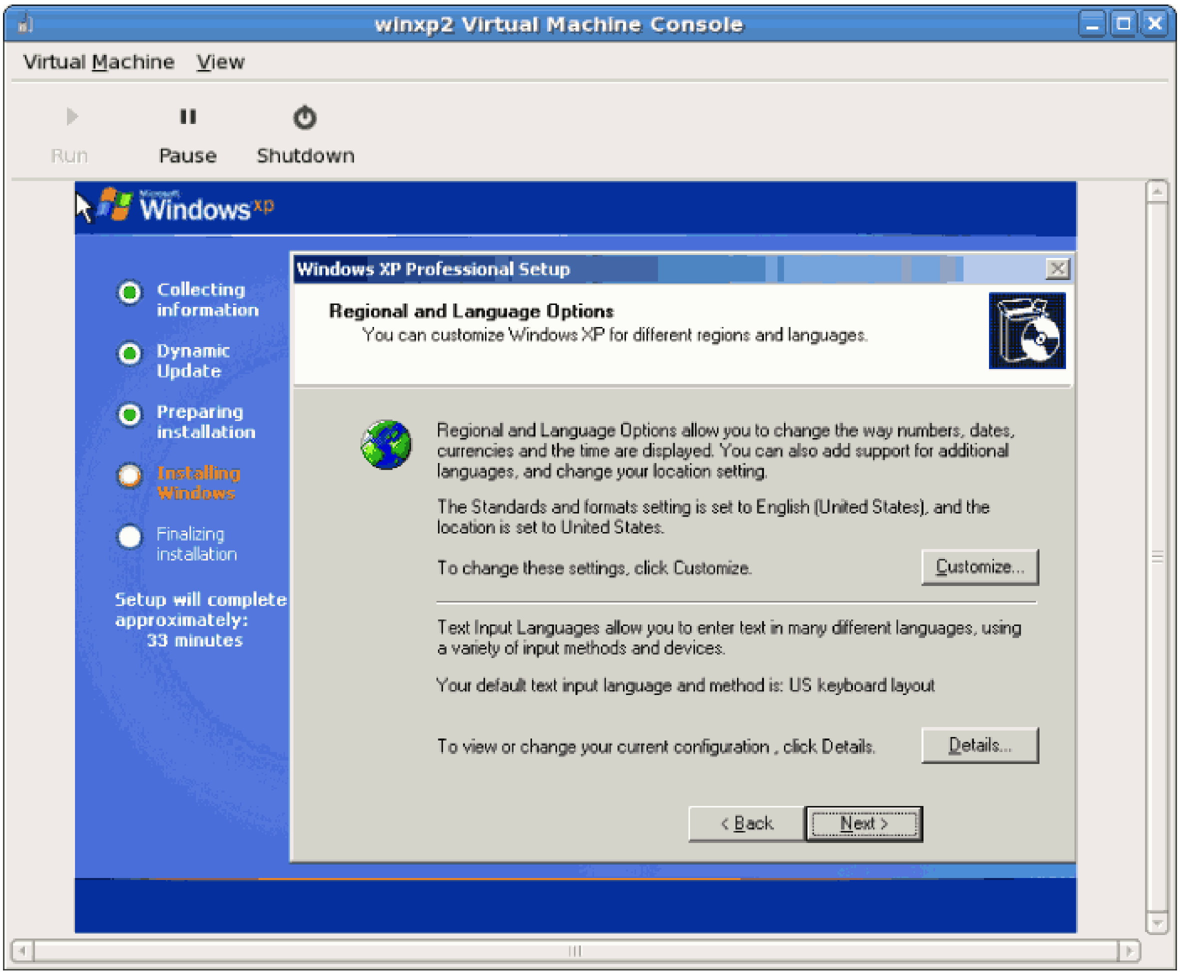
16. If your installation seems to get stuck during the setup phase, restart the guest with **`virsh reboot`** ***`WindowsGuestName`***. The will usually get the installation to continue. As you restart the virtual machine you will see a `Setup is being restarted` message:



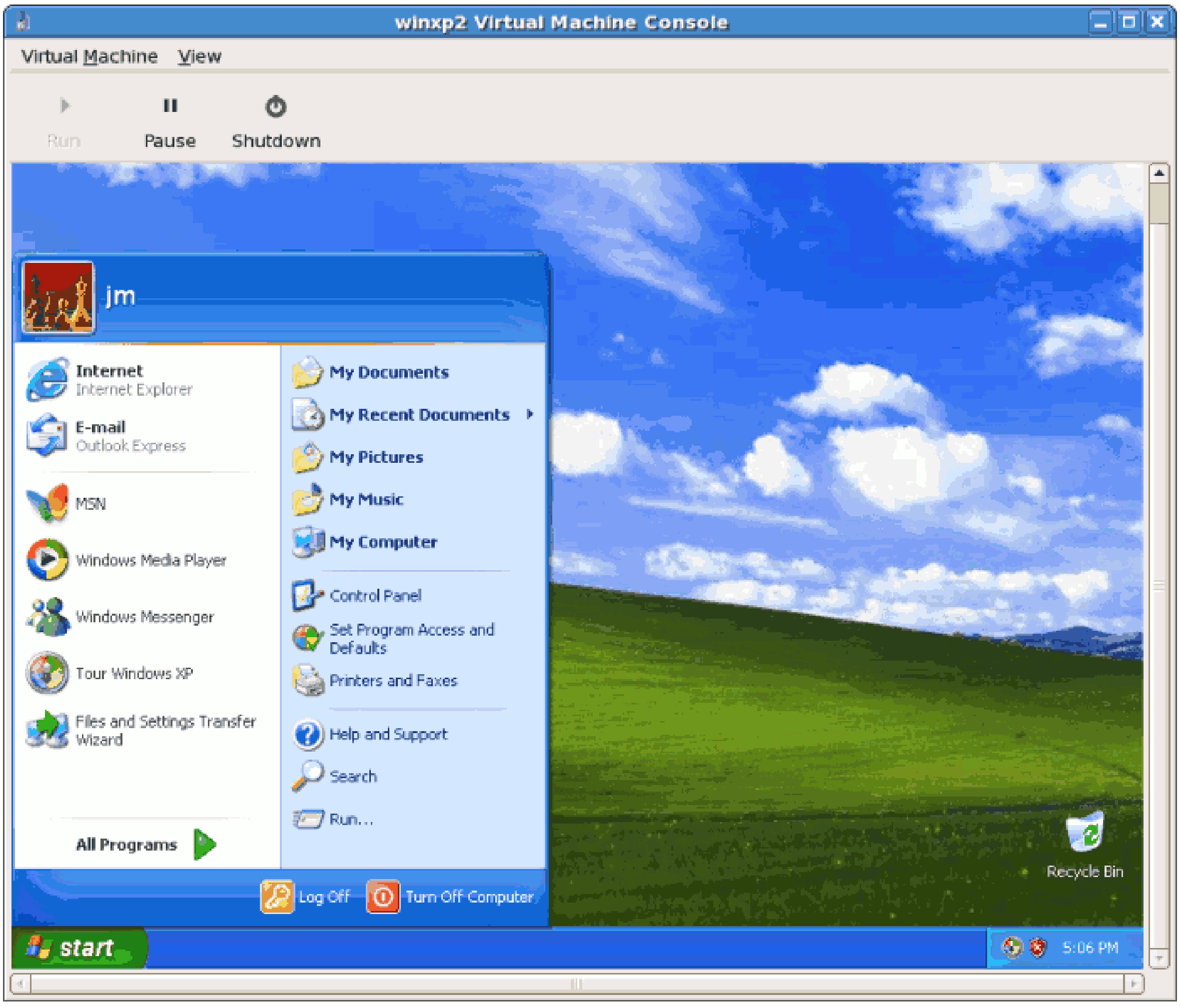
17. After setup has finished you will see the Windows boot screen:



18. Now you can continue with the standard setup of your Windows installation:



19. The setup process is complete, a Windows desktop displays.



7.4. Installing Windows Server 2003 as a fully virtualized guest

This chapter describes installing a fully virtualized Windows Server 2003 guest with the **`virt-install`** command. **`virt-install`** can be used instead of virt-manager This process is similar to the Windows XP installation covered in *Section 7.3, "Installing Windows XP as a fully virtualized guest"*.

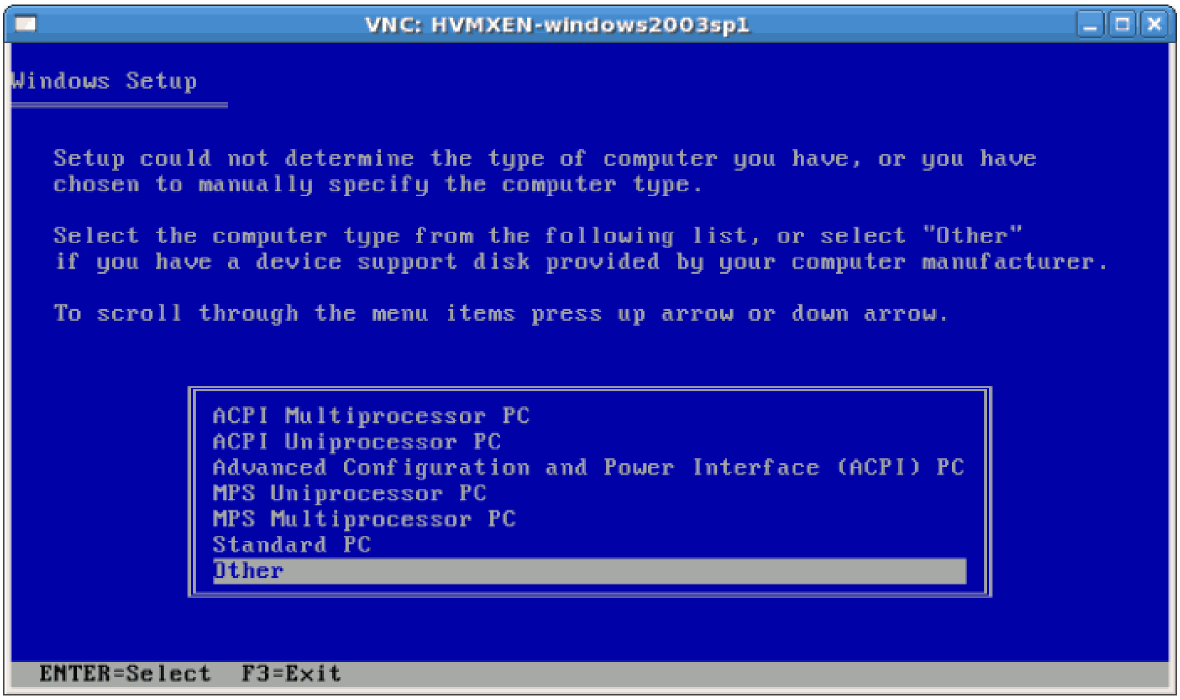
> **Itanium® support**
>
> Presently, Red Hat Enterprise Linux hosts on the Itanium® architecture do not support fully virtualized windows guests. This section only applies to x86 and x86-64 hosts.

1. Using **`virt-install`** for installing Windows Server 2003 as the console for the Windows guest opens the virt-viewer window promptly. An example of using the **`virt-install`** for installing a Windows Server 2003 guest:

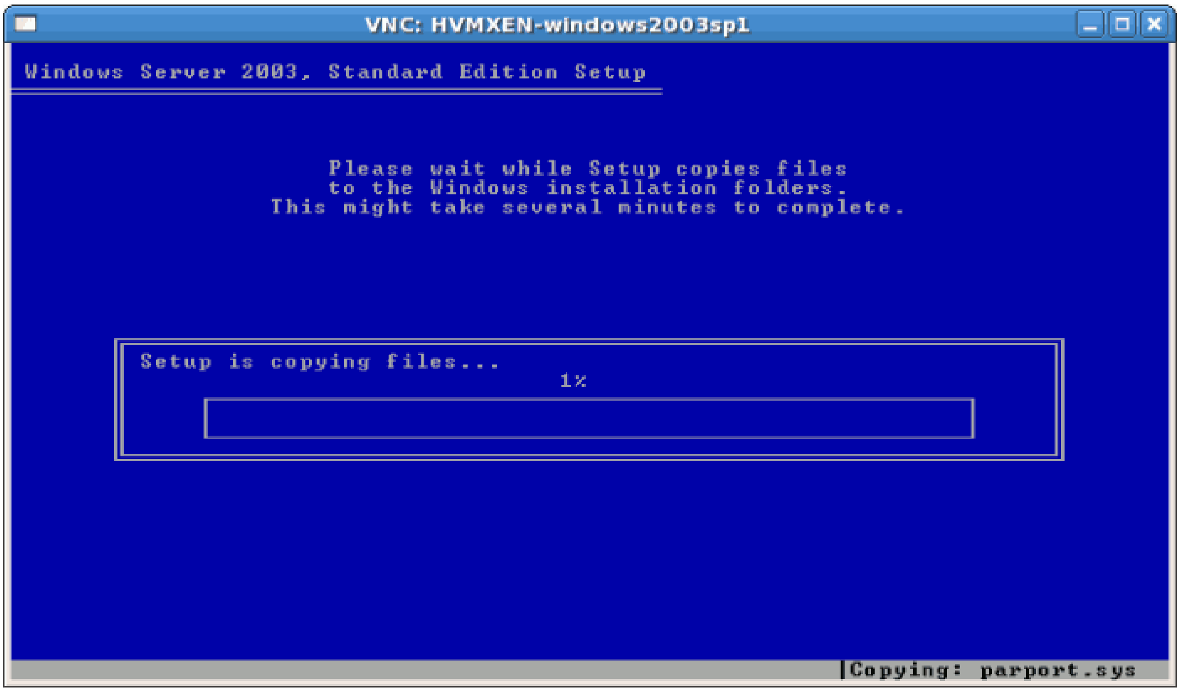
   Start the installation with the **`virt-install`** command.

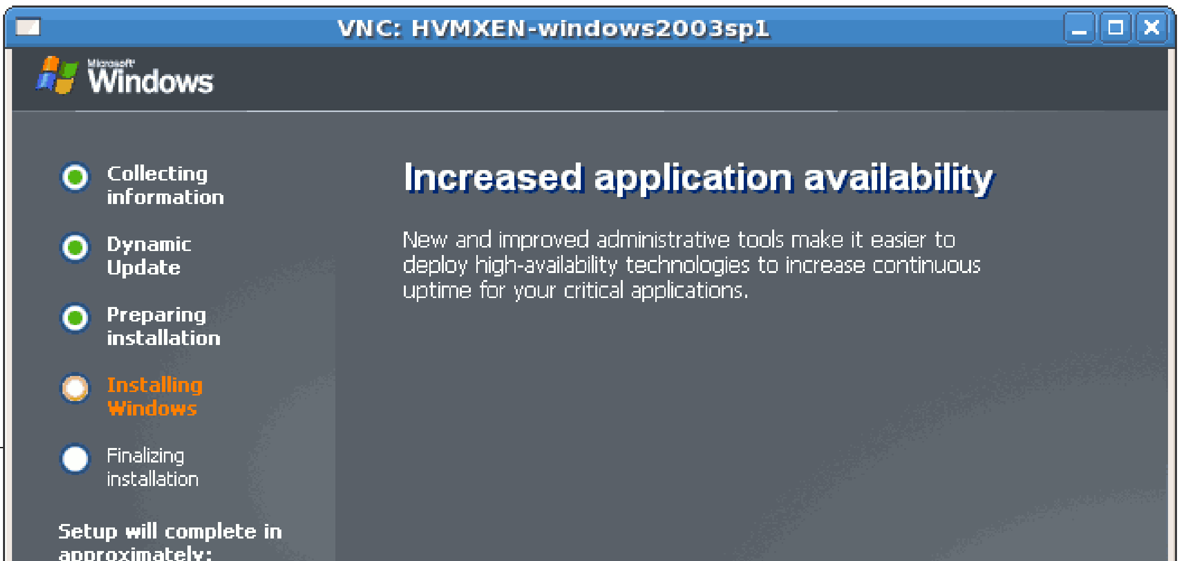
   ```
   # virt-install -hvm -s 5 -f /var/lib/libvirt/images/windows2003spi1.dsk
    \
   -n windows2003sp1 -cdrom=/ISOs/WIN/en_windows_server_2003_sp1.iso  \
   -vnc -r 1024
   ```

2. Once the guest boots into the installation you must quickly press **F5**. If you do not press **F5** at the right time you will need to restart the installation. Pressing **F5** allows you to select different **HAL** or **Computer Type**. Choose `Standard PC` as the Computer Type. This is the only non standard step required.



3. Complete the rest of the installation.





4. Windows Server 2003 is now installed as a fully virtualized guest.

7.5. Installing Windows Server 2008 as a fully virtualized guest

This section covers installing a fully virtualized Windows Server 2008 guest. The steps that uses the KVM hypervisor requires Red Hat Enterprise Linux 5.4 or newer.

Procedure 7.4. Installing Windows Server 2008 with virt-manager

1. **Open virt-manager**

   Start **`virt-manager`**. Launch the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu. Alternatively, run the **`virt-manager`** command as root.

2. **Select the hypervisor**

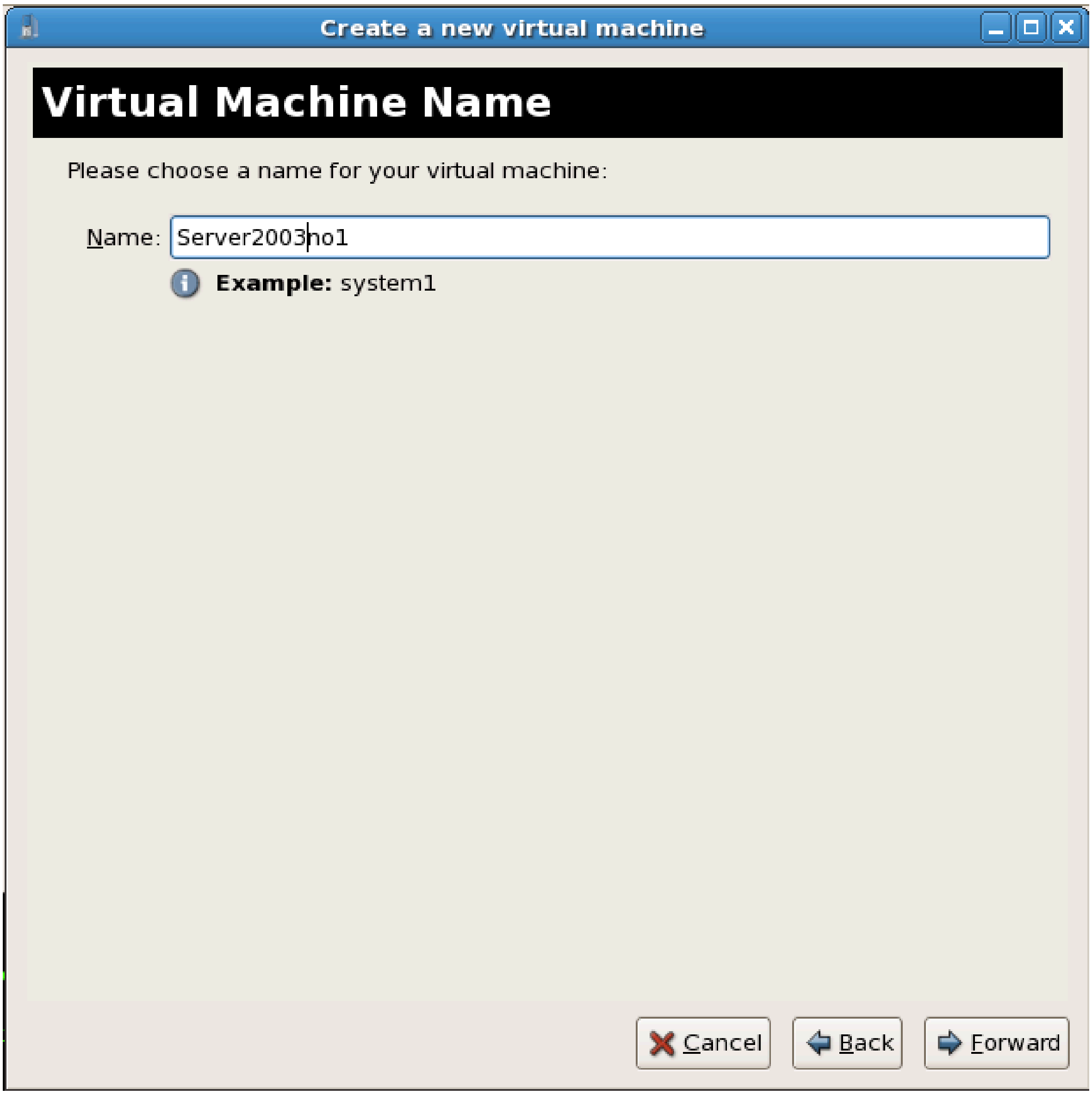
   Select the hypervisor. If installed, select Xen or KVM. For this example, select KVM. Note that presently KVM is named `qemu`.

   Once the option is selected the **New** button becomes available. Press the **New** button.

3. **Start the new virtual machine wizard**
   Pressing the **New** button starts the virtual machine creation wizard.

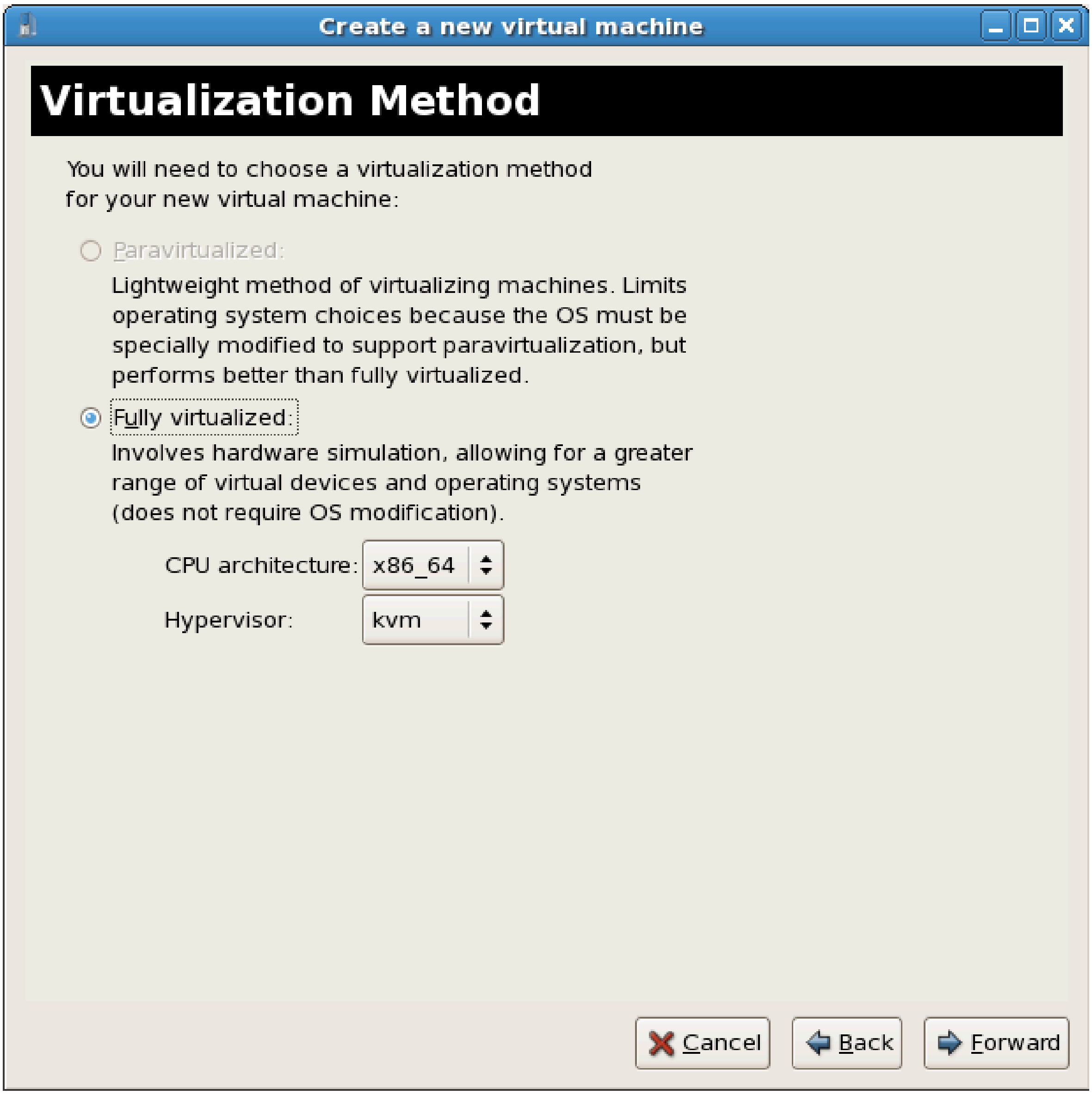
   Press **Forward** to continue.

4. **Name the virtual machine**
   Provide a name for your virtualized guest. Punctuation and whitespace characters are not permitted.



   Press **Forward** to continue.

5. **Choose a virtualization method**
Choose the virtualization method for the virtualized guest. Note you can only select an installed virtualization method. If you selected KVM or Xen earlier (step 2) you must use the hypervisor you selected. This example uses the KVM hypervisor.



Press **Forward** to continue.

6. **Select the installation method**

   For all versions of Windows you must use **local install media**, either an ISO image or physical optical media.

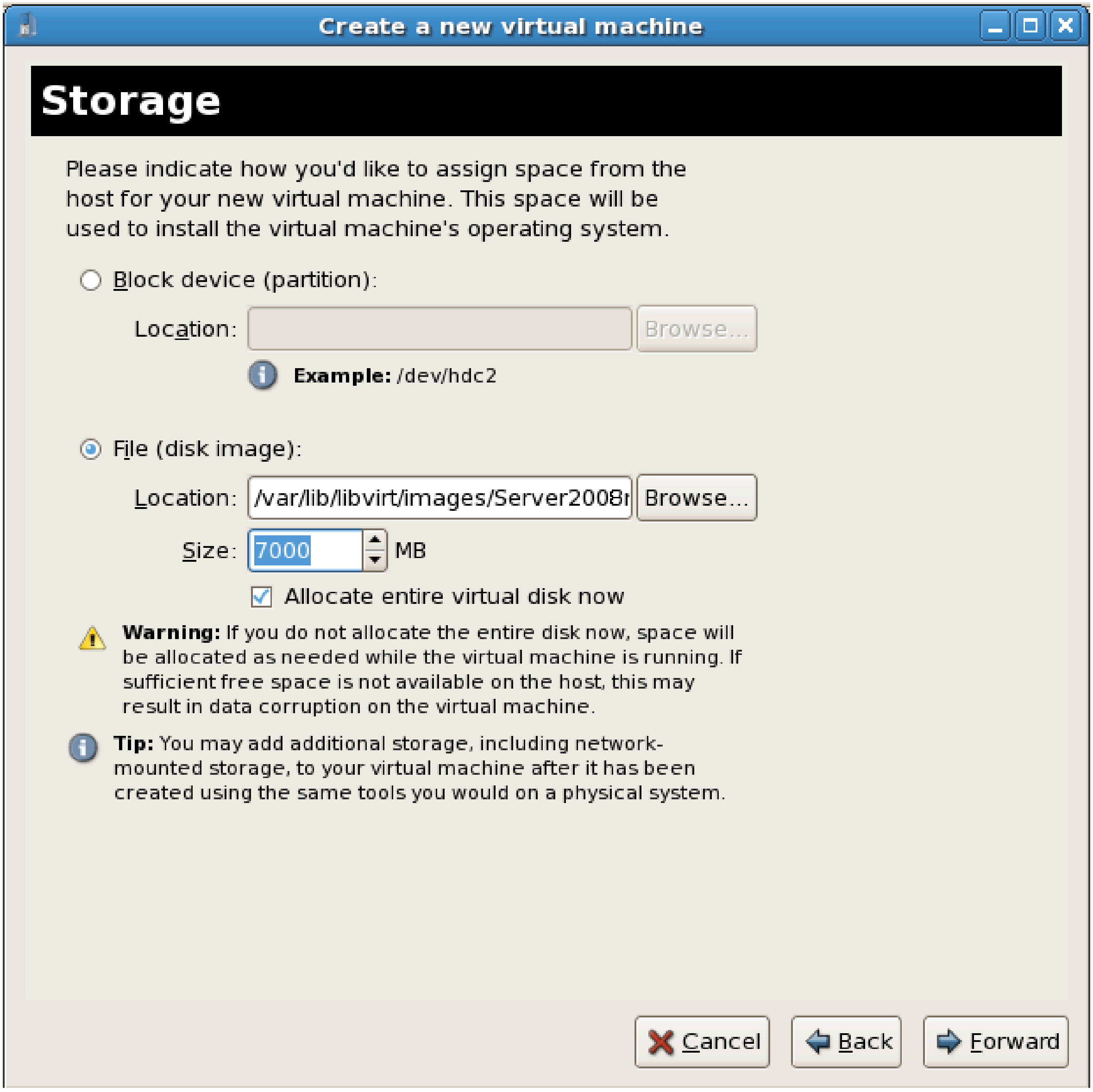
   PXE may be used if you have a PXE server configured for Windows network installation. PXE Windows installation is not covered by this guide.

   Set **OS Type** to **Windows** and **OS Variant** to **Microsoft Windows 2008** as shown in the screenshot.

   Press **Forward** to continue.

7. **Locate installation media**
   Select ISO image location or CD-ROM or DVD device. This example uses an ISO file image of the Windows Server 2008 installation CD.

8. **Storage setup**
   Assign a physical storage device (**Block device**) or a file-based image (**File**). File-based images must be stored in the **`/var/lib/libvirt/images/`** directory. Assign sufficient storage for your virtualized guest. Assign sufficient space for your virtualized guest and any application it requires.

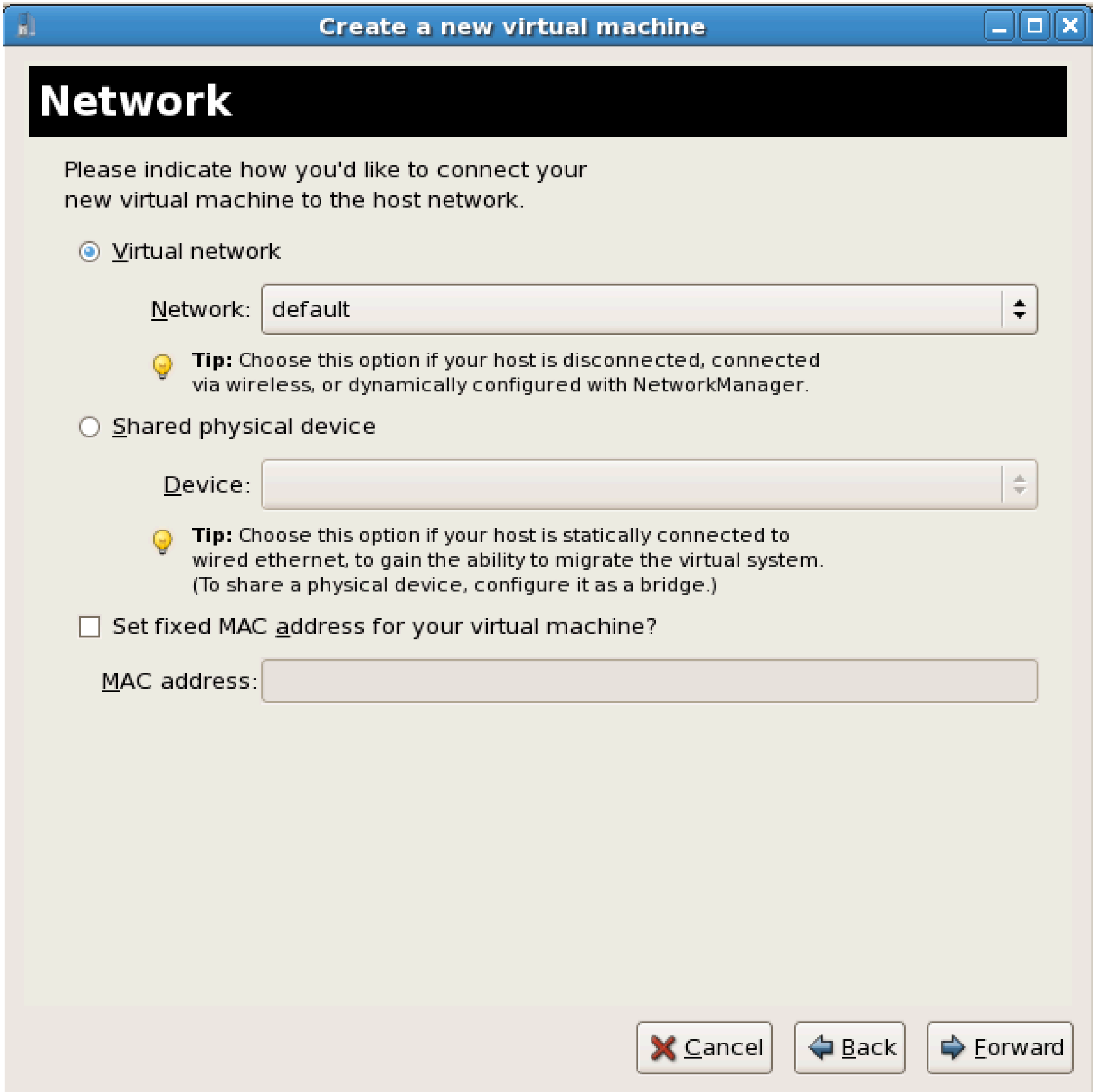


   Press **Forward** to continue.

9. **Network setup**
   Select either **Virtual network** or **Shared physical device**.

   The virtual network option uses Network Address Translation (NAT) to share the default network device with the virtualized guest. Use the virtual network option for wireless networks.

   The shared physical device option uses a network bond to give the virtualized guest full access to a network device.



   Press **Forward** to continue.

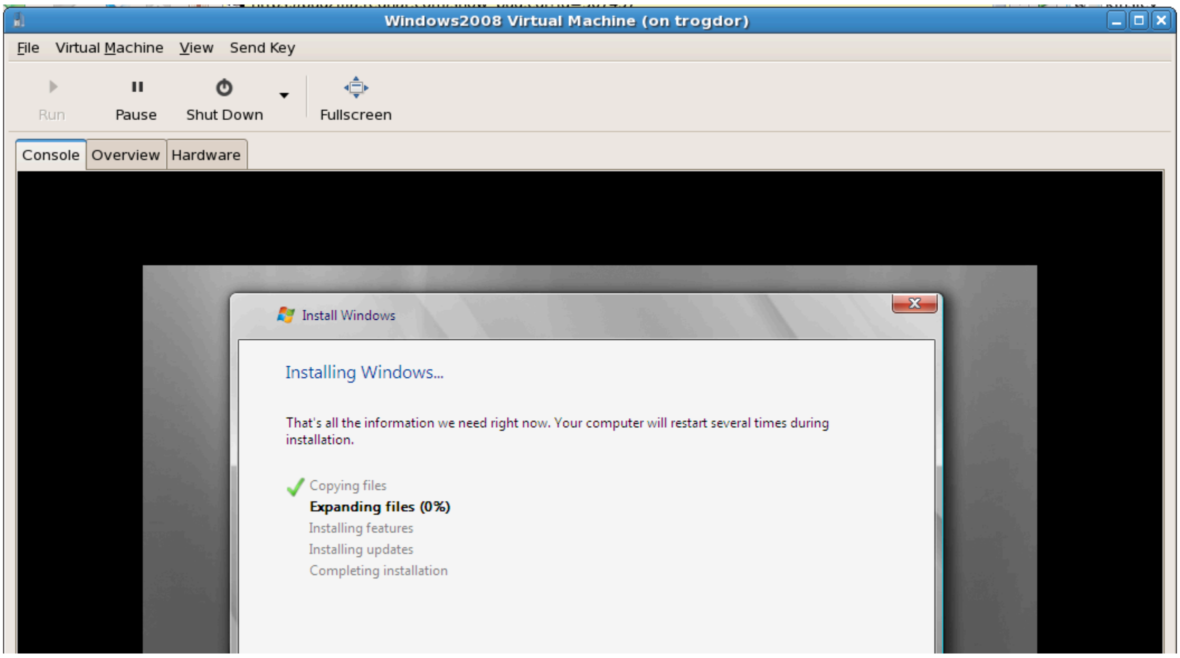
10. **Memory and CPU allocation**

    The Allocate memory and CPU window displays. Choose appropriate values for the virtualized CPUs and RAM allocation. These values affect the host's and guest's performance.

11. **Verify and start guest installation**
    Verify the configuration.

    Press **Finish** to start the guest installation procedure.

12. **Installing Windows**



Complete the Windows Server 2008 installation sequence. The installation sequence is not covered by this guide, refer to Microsoft's *documentation*[2] for information on installing Windows.

Part III. Configuration

Configuring Virtualization in Red Hat Enterprise Linux

These chapters cover configuration procedures for various advanced virtualization tasks. These tasks include adding network and storage devices, enhancing security, improving performance, and using the para-virtualized drivers on fully virtualized guests.

Virtualized block devices

This chapter covers installing and configuring block devices in virtualized guests. The term block devices refers to various forms of storage devices.

8.1. Creating a virtualized floppy disk controller

Floppy disk controllers are required for a number of older operating systems, especially for installing drivers. Presently, physical floppy disk devices cannot be accessed from virtualized guests. However, creating and accessing floppy disk images from virtualized floppy drives is supported. This section covers creating a virtualized floppy device.

An image file of a floppy disk is required. Create floppy disk image files with the **dd** command. Replace /dev/fd0 with the name of a floppy device and name the disk appropriately.

```
# dd if=/dev/fd0 of=~/legacydrivers.img
```

Para-virtualized drivers note

The para-virtualized drivers can map physical floppy devices to fully virtualized guests. For more information on using para-virtualized drivers read *Chapter 14, Xen Para-virtualized Drivers*.

This example uses a guest created with **`virt-manager`** running a fully virtualized Red Hat Enterprise Linux installation with an image located in **`/var/lib/libvirt/images/rhel5FV.img`**. The Xen hypervisor is used in the example.

1. Create the XML configuration file for your guest image using the **`virsh`** command on a running guest.

   ```
   # virsh dumpxml rhel5FV > rhel5FV.xml
   ```

   This saves the configuration settings as an XML file which can be edited to customize the operations and devices used by the guest. For more information on using the virsh XML configuration files, refer to *Chapter 29, Creating custom libvirt scripts*.

2. Create a floppy disk image for the guest.

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/rhel5FV-floppy.img bs=512
    count=2880
   ```

3. Add the content below, changing where appropriate, to your guest's configuration XML file. This example creates a guest with a floppy device as a file based virtual device.

   ```
   <disk type='file' device='floppy'>
    <source file='/var/lib/libvirt/images/rhel5FV-floppy.img'/>
    <target dev='fda'/>
   </disk>
   ```

4. Stop the guest.

```
# virsh stop rhel5FV
```

5. Restart the guest using the XML configuration file.

```
# virsh create rhel5FV.xml
```

The floppy device is now available in the guest and stored as an image file on the host.

8.2. Adding storage devices to guests

This section covers adding storage devices to virtualized guest. Additional storage can only be added after guests are created. The supported storage devices and protocol include:

- local hard drive partitions,
- logical volumes,
- Fibre Channel or iSCSI directly connected to the host.
- File containers residing in a file system on the host.
- **NFS** file systems mounted directly by the virtual machine.
- iSCSI storage directly accessed by the guest.
- Cluster File Systems (**GFS**).

Adding file based storage to a guest

File-based storage or file-based containers are files on the hosts file system which act as virtualized hard drives for virtualized guests. To add a file-based container perform the following steps:

1. Create an empty container file or using an existing file container (such as an ISO file).

   a. Create a sparse file using the **dd** command. Sparse files are not recommended due to data integrity and performance issues. Sparse files are created much faster and can used for testing but should not be used in production environments.

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M
    seek=4096 count=0
   ```

   b. Non-sparse, pre-allocated files are recommended for file based storage containers. Create a non-sparse file, execute:

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M
    count=4096
   ```

   Both commands create a 400MB file which can be used as additional storage for a virtualized guest.

2. Dump the configuration for the guest. In this example the guest is called *Guest1* and the file is saved in the users home directory.

```
# virsh dumpxml Guest1 > ~/Guest1.xml
```

3. Open the configuration file (*Guest1.xml* in this example) in a text editor. Find the entries starting with "disk=". This entry resembles:

```
>disk type='file' device='disk'<
        >driver name='tap' type='aio'/<
        >source file='/var/lib/libvirt/images/Guest1.img'/<
        >target dev='xvda'/<
>/disk<
```

4. Add the additional storage by modifying the end of disk= entry. Ensure you specify a device name for the virtual block device which is not used already in the configuration file. The following example entry adds file, named **FileName.img**, as a file based storage container:

```
>disk type='file' device='disk'<
        >driver name='tap' type='aio'/<
        >source file='/var/lib/libvirt/images/Guest1.img'/<
        >target dev='xvda'/<
>/disk<
>disk type='file' device='disk'<
        >driver name='tap' type='aio'/<
 >source file='/var/lib/libvirt/images/FileName.img'/<
        >target dev='hda'/<
>/disk<
```

5. Restart the guest from the updated configuration file.

```
# virsh create Guest1.xml
```

6. The following steps are Linux guest specific. Other operating systems handle new storage devices in different ways. For other systems, refer to that operating system's documentation

   The guest now uses the file **`FileName.img`** as the device called **`/dev/hdb`**. This device requires formatting from the guest. On the guest, partition the device into one primary partition for the entire device then format the device.

7. Mount the disk on the guest.

```
# mount /dev/hdb1 /myfiles
```

The guest now has an additional virtualized file-based storage device.

Adding hard drives and other block devices to a guest

System administrators use additional hard drives for to provide more storage space or to separate system data from user data. This procedure, *Adding physical block devices to virtualized guests*, describes how to add a hard drive on the host to a virtualized guest.

The procedure works for all physical block devices, this includes CD-ROM, DVD and floppy devices.

Procedure 8.1. Adding physical block devices to virtualized guests

1. Physically attach the hard disk device to the host. Configure the host if the drive is not accessible by default.

2. Configure the device with **`multipath`** and persistence on the host if required.

3. Use the **`virsh attach`** command. Replace: *`myguest`* with your guest's name, *`/dev/hdb1`* with the device to add, and *`hdc`* with the location for the device on the guest. The *`hdc`* must be an unused device name. Use the *`hd*`* notation for Windows guests as well, the guest will recognize the device correctly.

   Append the `--type hdd` parameter to the command for CD-ROM or DVD devices.

   Append the `--type floppy` parameter to the command for floppy devices.

```
# virsh attach-disk myguest /dev/hdb1 hdc --driver tap --mode readonly
```

4. The guest now has a new hard disk device called **`/dev/hdb`** on Linux or **`D: drive`**, or similar, on Windows. This device may require formatting.

8.3. Configuring persistent storage in Red Hat Enterprise Linux 5

This section is for systems with external or networked storage; that is, Fibre Channel or iSCSI based storage devices. It is recommended that those systems have persistent device names configured for your hosts. This assists live migration as well as providing consistent device names and storage for multiple virtualized systems.

Universally Unique Identifiers(UUIDs) are a standardized method for identifying computers and devices in distributed computing environments. This sections uses UUIDs to identify iSCSI or Fibre Channel LUNs. UUIDs persist after restarts, disconnection and device swaps. The UUID is similar to a label on the device.

Systems which are not running **`multipath`** must use *Single path configuration*. Systems running **`multipath`** can use *Multiple path configuration*.

Single path configuration

This procedure implements *LUN* device persistence using **udev**. Only use this procedure for hosts which are not using **multipath**.

1. Edit the **/etc/scsi_id.config** file.

   a. Ensure the **options=-b** is line commented out.

   ```
   # options=-b
   ```

   b. Add the following line:

   ```
   options=-g
   ```

   This option configures **udev** to assume all attached SCSI devices return a UUID.

2. To display the UUID for a given device run the **scsi_id -g -s /block/sd*** command. For example:

   ```
   # scsi_id -g -s /block/sd*
   3600a0b800013275100000015427b625e
   ```

   The output may vary from the example above. The output displays the UUID of the device `/dev/sdc`.

3. Verify the UUID output by the **scsi_id -g -s /block/sd*** command is identical from computer which accesses the device.

4. Create a rule to name the device. Create a file named **20-names.rules** in the **/etc/udev/rules.d** directory. Add new rules to this file. All rules are added to the same file using the same format. Rules follow this format:

   ```
   KERNEL="sd*", BUS="scsi", PROGRAM="/sbin/scsi_id -g -s", RESULT=UUID,
    NAME=devicename
   ```

   Replace *UUID* and *devicename* with the UUID retrieved above, and a name for the device. This is a rule for the example above:

   ```
   KERNEL="sd*", BUS="scsi", PROGRAM="/sbin/scsi_id -g -s",
    RESULT="3600a0b800013275100000015427b625e", NAME="rack4row16"
   ```

   The **udev** daemon now searches all devices named **/dev/sd*** for the UUID in the rule. Once a matching device is connected to the system the device is assigned the name from the rule. In the a device with a UUID of 3600a0b800013275100000015427b625e would appear as **/dev/rack4row16**.

5. Append this line to **/etc/rc.local**:

   ```
   /sbin/start_udev
   ```

6. Copy the changes in the **/etc/scsi_id.config**, **/etc/udev/rules.d/20-names.rules**, and **/etc/rc.local** files to all relevant hosts.

```
/sbin/start_udev
```

Networked storage devices with configured rules now have persistent names on all hosts where the files were updated This means you can migrate guests between hosts using the shared storage and the guests can access the storage devices in their configuration files.

Multiple path configuration

The **multipath** package is used for systems with more than one physical path from the computer to storage devices. **multipath** provides fault tolerance, fail-over and enhanced performance for network storage devices attached to Red Hat Enterprise Linux systems.

Implementing LUN persistence in a **multipath** environment requires defined alias names for your multipath devices. Each storage device has a UUID which acts as a key for the aliased names. Identify a device's UUID using the **scsi_id** command.

```
# scsi_id -g -s /block/sdc
```

The multipath devices will be created in the **/dev/mpath** directory. In the example below 4 devices are defined in **/etc/multipath.conf**:

```
multipaths {
 multipath {
 wwid  3600805f30015987000000000768a0019
 alias  oramp1
 }
 multipath {
 wwid  3600805f30015987000000000d643001a
 alias  oramp2
 }
 mulitpath {
 wwid  3600805f3001598700000000086fc001b
 alias  oramp3
 }
 mulitpath {
 wwid  3600805f300159870000000000984001c
 alias  oramp4
 }
}
```

This configuration will create 4 LUNs named **/dev/mpath/oramp1**, **/dev/mpath/oramp2**, **/dev/mpath/oramp3** and **/dev/mpath/oramp4**. Once entered, the mapping of the devices' WWID to their new names are now persistent after rebooting.

8.4. Add a virtualized CD-ROM or DVD device to a guest

To attach an ISO file to a guest while the guest is online use **virsh** with the *attach-disk* parameter.

```
# virsh attach-disk [domain-id] [source] [target] --driver file --type
 cdrom --mode readonly
```

The *source* and *target* parameters are paths for the files and devices, on the host and guest respectively. The *source* parameter can be a path to an ISO file or the device from the **/dev** directory.

Shared storage and virtualization

This chapter covers using shared, networked storage with virtualization on Red Hat Enterprise Linux.

The following methods are supported for virtualization:

- Fibre Channel
- iSCSI
- NFS
- GFS2

Networked storage is essential for live and offline guest migrations. You cannot migrate guests without shared storage.

9.1. Using iSCSI for storing guests

This section covers using iSCSI-based devices to store virtualized guests.

9.2. Using NFS for storing guests

This section covers using NFS to store virtualized guests.

9.3. Using GFS2 for storing guests

This section covers using the Red Hat Global File System 2 (GFS2) to store virtualized guests.

Server best practices

The following tasks and tips can assist you with securing and ensuring reliability of your Red Hat Enterprise Linux 5 server host (dom0).

- Run SELinux in enforcing mode. You can do this by executing the command below.

  ```
  # setenforce 1
  ```

- Remove or disable any unnecessary services such as **AutoFS**, **NFS**, **FTP**, **HTTP**, **NIS**, **telnetd**, **sendmail** and so on.
- Only add the minimum number of user accounts needed for platform management on the server and remove unnecessary user accounts.
- Avoid running any unessential applications on your host. Running applications on the host may impact virtual machine performance and can affect server stability. Any application which may crash the server will also cause all virtual machines on the server to go down.
- Use a central location for virtual machine installations and images. Virtual machine images should be stored under **/var/lib/libvirt/images/**. If you are using a different directory for your virtual machine images make sure you add the directory to your SELinux policy and relabel it before starting the installation.
- Installation sources, trees, and images should be stored in a central location, usually the location of your `vsftpd` server.

Security for virtualization

When deploying virtualization technologies on your corporate infrastructure, you must ensure that the host cannot be compromised. The host, in the Xen hypervisor, is a privileged domain that handles system management and manages all virtual machines. If the host is insecure, all other domains in the system are vulnerable. There are several ways to enhance security on systems using virtualization. You or your organization should create a *Deployment Plan* containing the operating specifications and specifies which services are needed on your virtualized guests and host servers as well as what support is required for these services. Here are a few security issues to consider while developing a deployment plan:

- Run only necessary services on hosts. The fewer processes and services running on the host, the higher the level of security and performance.
- Enable *SELinux* on the hypervisor. Read *Section 11.1, "SELinux and virtualization"* for more information on using SELinux and virtualization.
- Use a firewall to restrict traffic to dom0. You can setup a firewall with default-reject rules that will help secure attacks on dom0. It is also important to limit network facing services.
- Do not allow normal users to access dom0. If you do permit normal users dom0 access, you run the risk of rendering dom0 vulnerable. Remember, dom0 is privileged, and granting unprivileged accounts may compromise the level of security.

11.1. SELinux and virtualization

Security Enhanced Linux was developed by the NSA with assistance from the Linux community to provide stronger security for Linux. SELinux limits an attackers abilities and works to prevent many common security exploits such as buffer overflow attacks and privilege escalation. It is because of these benefits that Red Hat recommends all Red Hat Enterprise Linux systems should run with SELinux enabled and in enforcing mode.

SELinux prevents guest images from loading if SELinux is enabled and the images are not in the correct directory. SELinux requires that all guest images are stored in **`/var/lib/libvirt/images`**.

Adding LVM based storage with SELinux in enforcing mode

The following section is an example of adding a logical volume to a virtualized guest with SELinux enabled. These instructions also work for hard drive partitions.

Procedure 11.1. Creating and mounting a logical volume on a virtualized guest with SELinux enabled

1. Create a logical volume. This example creates a 5 gigabyte logical volume named *`NewVolumeName`* on the volume group named *`volumegroup`*.

```
# lvcreate -n NewVolumeName -L 5G volumegroup
```

2. Format the *`NewVolumeName`* logical volume with a file system that supports extended attributes, such as ext3.

```
# mke2fs -j /dev/volumegroup/NewVolumeName
```

3. Create a new directory for mounting the new logical volume. This directory can be anywhere on your file system. It is advised not to put it in important system directories (**/etc**, **/var**, **/sys**) or in home directories (**/home** or **/root**). This example uses a directory called **/virtstorage**

   ```
   # mkdir /virtstorage
   ```

4. Mount the logical volume.

   ```
   # mount /dev/volumegroup/NewVolumeName /virtstorage
   ```

5. Set the correct SELinux type for the Xen folder.

   ```
   semanage fcontext -a -t xen_image_t "/virtualization(/.*)?"
   ```

   Alternatively, set the correct SELinux type for a KVM folder.

   ```
   semanage fcontext -a -t virt_image_t "/virtualization(/.*)?"
   ```

   If the targeted policy is used (targeted is the default policy) the command appends a line to the **/etc/selinux/targeted/contexts/files/file_contexts.local** file which makes the change persistent. The appended line may resemble this:

   ```
   /virtstorage(/.*)?    system_u:object_r:xen_image_t:s0
   ```

6. Run the command to change the type of the mount point (**/virtstorage**) and all files under it to **xen_image_t** (**restorecon** and **setfiles** read the files in **/etc/selinux/targeted/contexts/files/**).

   ```
   # restorecon -R -v /virtualization
   ```

11.2. SELinux considerations

This sections contains things to you must consider when you implement SELinux into your virtualization deployment. When you deploy system changes or add devices, you must update your SELinux policy accordingly. To configure an LVM volume for a guest, you must modify the SELinux context for the respective underlying block device and volume group.

```
# semanage fcontext -a -t xen_image _t -f -b /dev/sda2
# restorecon /dev/sda2
```

The Boolean parameter **xend_disable_t** can set the **xend** to unconfined mode after restarting the daemon. It is better to disable protection for a single daemon than the whole system. It is advisable that you should not re-label directories as **xen_image_t** that you will use elsewhere.

Network Configuration

This page provides an introduction to the common networking configurations used by libvirt based applications. This information applies to all hypervisors, whether Xen, KVM or another. For additional information consult the libvirt network architecture docs.

The two common setups are "virtual network" or "shared physical device". The former is identical across all distributions and available out-of-the-box. The latter needs distribution specific manual configuration.

12.1. Network address translation (NAT) with libvirt

One of the most common methods for sharing network connections is to use Network address translation (NAT) forwarding (also know as virtual networks).

Host configuration

Every standard `libvirt` installation provides NAT based connectivity to virtual machines out of the box. This is the so called 'default virtual network'. Verify that it is available with the **`virsh net-list --all`** command.

```
# virsh net-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
```

If it is missing, the example XML configuration file can be reloaded and activated:

```
# virsh net-define /usr/share/libvirt/networks/default.xml
```

The default network is defined from **`/usr/share/libvirt/networks/default.xml`**

Mark the default network to automatically start:

```
# virsh net-autostart default
Network default marked as autostarted
```

Start the default network:

```
# virsh net-start default
Network default started
```

Once the `libvirt` default network is running, you will see an isolated bridge device. This device does *not* have any physical interfaces added, since it uses NAT and IP forwarding to connect to outside world. Do not add new interfaces.

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
```

libvirt adds **iptables** rules which allow traffic to and from guests attached to the virbr0 device in the **INPUT**, **FORWARD**, **OUTPUT** and **POSTROUTING** chains. **libvirt** then attempts to enable the **ip_forward** parameter. Some other applications may disable **ip_forward**, so the best option is to add the following to **/etc/sysctl.conf**.

```
net.ipv4.ip_forward = 1
```

Guest configuration

Once the host configuration is complete, a guest can be connected to the virtual network based on its name. To connect a guest to the 'default' virtual network, the following XML can be used in the guest:

```
<interface type='network'>
  <source network='default'/>
</interface>
```

> **Note**
>
> Defining a MAC address is optional. A MAC address is automatically generated if omitted. Manually setting the MAC address is useful in certain situations.
>
> ```
> <interface type='network'>
>   <source network='default'/>
>   <mac address='00:16:3e:1a:b3:4a'/>
>   </interface>
> ```

12.2. Bridged networking with libvirt

Bridged networking (also known as physical device sharing) is used for dedicating a physical device to a virtual machine. Bridging is often used for more advanced setups and on servers with multiple network interfaces.

Disable Xen network scripts

If your system was using a Xen bridge, it is recommended to disable the default Xen network bridge by editing **/etc/xen/xend-config.sxp** and changing the line:

```
(network-script network-bridge)
```

To:

```
(network-script /bin/true)
```

Disable NetworkManager

NetworkManager does not support bridging. NetworkManager must be disabled to use the older network scripts networking.

```
# chkconfig NetworkManager off
# chkconfig network on
# service NetworkManager stop
# service network start
```

Note

Instead of turning off NetworkManager, you can add "*NM_CONTROLLED=no*" to the **ifcfg-*** scripts used in the examples.

Creating network initscripts

Create or edit the following two network configuration files. This step can be repeated (with different names) for additional network bridges.

Change to the **/etc/sysconfig/network-scripts** directory:

```
# cd /etc/sysconfig/network-scripts
```

Open the network script for the device you are adding to the bridge. In this example, **ifcfg-eth0** defines the physical network interface which is set as part of a bridge:

```
DEVICE=eth0
# change the hardware address to match the hardware address your NIC uses
HWADDR=00:16:76:D6:C9:45
ONBOOT=yes
BRIDGE=br0
```

Tip

You can configure the device's Maximum Transfer Unit (MTU) by appending an *MTU* variable to the end of the configuration file.

```
MTU=9000
```

Create a new network script in the **/etc/sysconfig/network-scripts** directory called **ifcfg-br0** or similar. The *br0* is the name of the bridge, this can be anything as long as the name of the file is the same as the DEVICE parameter.

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
DELAY=0
```

> **Warning**
>
> The line, *`TYPE=Bridge`*, is case-sensitive. It must have uppercase 'B' and lower case 'ridge'.

After configuring, restart networking or reboot.

```
# service network restart
```

Configure **`iptables`** to allow all traffic to be forwarded across the bridge.

```
# iptables -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
# service iptables save
# service iptables restart
```

> **Disable iptables on bridges**
>
> Alternatively, prevent bridged traffic from being processed by **`iptables`** rules. In **`/etc/sysctl.conf`** append the following lines:
>
> ```
> net.bridge.bridge-nf-call-ip6tables = 0
> net.bridge.bridge-nf-call-iptables = 0
> net.bridge.bridge-nf-call-arptables = 0
> ```
>
> Reload the kernel parameters configured with **`sysctl`**

```
# sysctl -p /etc/sysctl.conf
```

Restart the **`libvirt`** daemon.

```
# service libvirtd reload
```

You should now have a "shared physical device", which guests can be attached and have full LAN access. Verify your new bridge:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
br0             8000.000e0cb30550       no              eth0
```

Note, the bridge is completely independent of the **`virbr0`** bridge. Do *not* attempt to attach a physical device to **`virbr0`**. The **`virbr0`** bridge is only for Network Address Translation (NAT) connectivity.

Pre-Red Hat Enterprise Linux 5.4 Xen networking

This chapter covers special topics for networking and network configuration with the Xen hypervisor.

Most guest network configuration occurs during the guest initialization and installation process. To learn about configuring networking during the guest installation process, read the relevant sections of the installation process, *Chapter 6, Virtualized guest installation overview*.

Network configuration is also covered in the tool specific reference chapters for **`virsh`** (*Chapter 22, Managing guests with virsh*) and **`virt-manager`** (*Chapter 23, Managing guests with the Virtual Machine Manager (virt-manager)*). Those chapters provide a detailed description of the networking configuration tasks using both tools.

> **Tip**
> Using para-virtualized network drivers improves performance on fully virtualized Linux guests. *Chapter 14, Xen Para-virtualized Drivers* explains how to utilize para-virtualized network drivers.

13.1. Configuring multiple guest network bridges to use multiple Ethernet cards

Process to setup network bridges (with the Xen hypervisor):

1. Configure another network interface using either the **`system-config-network`** application. Alternatively, create a new configuration file named **`ifcfg-ethX`** in the **`/etc/sysconfig/network-scripts/`** directory where *X* is any number not already in use. Below is an example configuration file for a second network interface called `eth1`

```
$ cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
TYPE=Ethernet
NETMASK=255.255.255.0
IPADDR=10.1.1.1
GATEWAY=10.1.1.254
ARP=yes
```

2. Copy the file, **`/etc/xen/scripts/network-bridge`**, to **`/etc/xen/scripts/network-bridge.xen`**.

3. Comment out any existing network scripts in **`/etc/xen/xend-config.sxp`** and add the line *`(network-xen-multi-bridge)`*.

4. Create a custom script to create multiple network bridges. A sample scripts is below, this example script will create two Xen network bridges (xenbr0 and xenbr1) one will be attached to eth1 and the other one to eth0. If you want to create additional bridges just follow the example in the script and copy/paste the lines accordingly:

```
#!/bin/sh
# network-xen-multi-bridge
# Exit if anything goes wrong.
set -e
# First arg is the operation.
OP=$1
shift
script=/etc/xen/scripts/network-bridge.xen
case ${OP} in
start)
 $script start vifnum=1 bridge=xenbr1 netdev=eth1
 $script start vifnum=0 bridge=xenbr0 netdev=eth0
 ;;
stop)
 $script stop vifnum=1 bridge=xenbr1 netdev=eth1
 $script stop vifnum=0 bridge=xenbr0 netdev=eth0
 ;;
status)
 $script status vifnum=1 bridge=xenbr1 netdev=eth1
 $script status vifnum=0 bridge=xenbr0 netdev=eth0
 ;;
*)
 echo 'Unknown command: ' ${OP}
 echo 'Valid commands are: start, stop, status'
 exit 1
esac
```

13.2. Red Hat Enterprise Linux 5.0 Laptop network configuration

For Red Hat Enterprise Linux 5.1 or newer

This section describes manually adding network bridges. This procedure is not required or recommended for all versions of Red Hat Enterprise Linux newer than version 5.0. For newer versions use "**Virtual Network**" adapters when creating guests in **`virt-manager`**. NetworkManager works with virtual network devices by default in Red Hat Enterprise Linux 5.1 and newer.

An example of a virsh XML configuration file virtual network device:

```
<interface type='network'>
 <mac address='AA:AA:AA:AA:AA:AA'/>
 <source network='default'/>
 <target dev='vnet0'/>
```

```
 <model type='virtio'/>
</interface>
```

In **xm** configuration files, virtual network devices are labeled "vif".

The challenge in running the Xen hypervisor on a laptop is that most laptops will connected to the network via wireless network or wired connections. Often these connections are switched multiple times a day. In such an environment, the system assumes it has access to the same interface all the time and it also can perform **ifup** or **ifdown** calls to the network interface it is using. In addition wireless network cards do not work well in a virtualization environment due to Xen's (default) bridged network usage.

This setup will also enable you to run Xen in offline mode when you have no active network connection on your laptop. The easiest solution to run Xen on a laptop is to follow the procedure outlined below:

- You will be configuring a 'dummy' network interface which will be used by Xen. In this example the interface is called **dummy0**. This will also allow you to use a hidden IP address space for your guests.
- You will need to use static IP address as DHCP will not listen on the dummy interface for DHCP requests. You can compile your own version of DHCP to listen on dummy interfaces, however you may want to look into using dnsmasq for DNS, DHCP and tftpboot services in a Xen environment. Setup and configuration are explained further down in this section/chapter.
- You can also configure NAT and IP masquerading in order to enable access to the network from your guests.

Configuring a dummy network interface

Perform the following configuration steps on your host:

1. create a dummy0 network interface and assign it a static IP address. In our example I selected 10.1.1.1 to avoid routing problems in our environment. To enable dummy device support add the following lines to **/etc/modprobe.conf**

   ```
   alias dummy0 dummy
   options dummy numdummies=1
   ```

2. To configure networking for dummy0 edit/create **/etc/sysconfig/network-scripts/ifcfg-dummy0**:

```
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
TYPE=Ethernet
NETMASK=255.255.255.0
IPADDR=10.1.1.1
ARP=yes
```

3. Bind `xenbr0` to `dummy0`, so you can use networking even when not connected to a physical network. Edit **/etc/xen/xend-config.sxp** to include the `netdev=dummy0` entry:

```
(network-script 'network-bridge bridge=xenbr0 netdev=dummy0')
```

4. Open **/etc/sysconfig/network** in the guest and modify the default gateway to point to `dummy0`. If you are using a static IP, set the guest's IP address to exist on the same subnet as `dummy0`.

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
GATEWAY=10.1.1.1
IPADDR=10.1.1.10
NETMASK=255.255.255.0
```

5. Setting up NAT in the host will allow the guests access Internet, including with wireless, solving the Xen and wireless card issues. The script below will enable NAT based on the interface currently used for your network connection.

Configuring NAT for virtualized guests

Network address translation (NAT) allows multiple network address to connect through a single IP address by intercepting packets and passing them to the private IP addresses. You can copy the following script to **/etc/init.d/xenLaptopNAT** and create a soft link to **/etc/rc3.d/S99xenLaptopNAT**. this automatically starts NAT at boot time.

NetworkManager and wireless NAT

The script below may not work well with wireless network or **NetworkManager** due to start up delays. In this case run the script manually once the machine has booted.

```
#!/bin/bash
PATH=/usr/bin:/sbin:/bin:/usr/sbin
export PATH
GATEWAYDEV=`ip route | grep default | awk {'print $5'}`
```

```
iptables -F
case "$1" in
start)
 if test -z "$GATEWAYDEV"; then
 echo "No gateway device found"
    else
 echo  "Masquerading using $GATEWAYDEV"
 /sbin/iptables -t nat -A POSTROUTING -o $GATEWAYDEV -j MASQUERADE
fi
 echo "Enabling IP forwarding"
 echo 1 > /proc/sys/net/ipv4/ip_forward
 echo "IP forwarding set to `cat /proc/sys/net/ipv4/ip_forward`"
 echo "done."
 ;;
*)
echo "Usage: $0 {start|restart|status}"
;;
esac
```

Configuring dnsmasq for the DNS, DHCP and tftpboot services

One of the challenges in running virtualization on a laptop (or any other computer which is not connected by a single or stable network connection) is the change in network interfaces and availability. Using a dummy network interface helps to build a more stable environment but it also brings up new challenges in providing DHCP, DNS and tftpboot services to your virtual machines/ guests. The default DHCP daemon shipped with Red Hat Enterprise Linux and Fedora Core will not listen on dummy interfaces, your DNS forwarded information may change as you connect to different networks and VPNs.

One solution to the above challenges is to use dnsmasq which can provide all of the above service in a single package and will also allow you to control its service only being available to requests from your dummy interface. Below is a short write up on how to configure **dnsmasq** on a laptop running virtualization:

- Get the latest version of dnsmasq from *here*[1].
- Document for dnsmasq can be found *here*[2].
- Copy the other files referenced below from *http://et.redhat.com/~jmh/tools/xen/* and grab the file **dnsmasq.tgz**. The tar archive includes the following files:
  - **nm-dnsmasq** can be used as a dispatcher script for NetworkManager. It will be run every time NetworkManager detects a change in connectivity and force a restart/reload of dnsmasq. It should be copied to **/etc/NetworkManager/dispatcher.d/nm-dnsmasq**
  - **xenDNSmasq** can be used as the main start up or shut down script for **/etc/init.d/xenDNSmasq**
  - **dnsmasq.conf** is a sample configuration file for **/etc/dnsmasq.conf**
  - **dnsmasq** is the binary image for **/usr/local/sbin/dnsmasq**

- Once you have unpacked and build dnsmasq (the default installation will be the binary into **`/usr/local/sbin/dnsmasq`**) you need to edit your dnsmasq configuration file. The file is located in **`/etc/dnsmaqs.conf`**
- Edit the configuration to suit your local needs and requirements. The following parameters are likely the ones you want to modify:
  - The `interface` parameter allows `dnsmasq` to listen for `DHCP` and `DNS` requests only on specified interfaces. This could be dummy interfaces but not your public interfaces as well as the local loopback interface. Add another `interface` line for more than one interface. `interface=dummy0` is an example which listens on the `dummy0` interface.
  - `dhcp-range` to enable the integrated `DHCP` server, you need to supply the range of addresses available for lease and optionally a lease time. If you have more than one network, you will need to repeat this for each network on which you want to supply `DHCP` service. An example would be (for network 10.1.1.* and a lease time of 12hrs): `dhcp-range=10.1.1.10,10.1.1.50,255.255.255.0,12h`
  - `dhcp-option` to override the default route supplied by dnsmasq, which assumes the router is the same machine as the one running dnsmasq. An example would be `dhcp-option=3,10.1.1.1`
- After configuring dnsmasq you can copy the script below as **`xenDNSmasq`** to **`/etc/init.d`**
- If you want to automatically start dnsmasq during system boot you should register it using chkconfig(8):

  ```
  chkconfig --add xenDNSmasq
  ```

  Enable it for automatic start up:

  ```
  chkconfig --levels 345 xenDNSmasq on
  ```

- To configure **`dnsmasq`** to restart every time **NetworkManager** detects a change in connectivity you can use the supplied script **`nm-dnsmasq`**.
  - Copy the **`nm-dnsmasq`** script to **`/etc/NetworkManager/dispatcher.d/`**
  - The **NetworkManager** dispatcher will execute the script (in alphabetical order if you have other scripts in the same directory) every time there is a change in connectivity
- **`dnsmasq`** will also detect changes in your **`/etc/resolv.conf`** and automatically reload them (that is, if you start up a VPN session for example).
- Both the **`nm-dnsmasq`** and **`xenDNSmasq`** script will also set up NAT if you have your virtualized guests on a hidden network to allow them access to the public network.

Xen Para-virtualized Drivers

Para-virtualized drivers provide increased performance for fully virtualized Red Hat Enterprise Linux guests. Use these drivers if you are using fully virtualized Red Hat Enterprise Linux guests and require better performance.

Other para-virtualized drivers

There are other para-virtualized drivers for Windows for both Xen and KVM hypervisors.

For Windows guests on Xen hosts, refer to the *Windows Para-virtualized Drivers Guide* which covers installation and administration.

For Windows guests on KVM hosts, refer to *Chapter 15, KVM Para-virtualized Drivers*.

The RPM packages for the para-virtualized drivers include the modules for storage and networking para-virtualized drivers for the supported Red Hat Enterprise Linux guest operating systems. These drivers enable high performance throughput of I/O operations in unmodified Red Hat Enterprise Linux guest operating systems on top of a Red Hat Enterprise Linux 5.1 (or greater) host.

The supported guest operating systems are:

- Red Hat Enterprise Linux 3
- Red Hat Enterprise Linux 4
- Red Hat Enterprise Linux 5

Architecture support for para-virtualized drivers

The minimum guest operating system requirements are architecture dependent. Only x86 and x86-64 guests are supported.

The drivers are not supported on Red Hat Enterprise Linux guest operating systems prior to Red Hat Enterprise Linux 3 .

Using Red Hat Enterprise Linux 5 as the virtualization platform allows System Administrators to consolidate Linux and Windows workloads onto newer, more powerful hardware with increased power and cooling efficiency. Red Hat Enterprise Linux 4 (as of update 6) and Red Hat Enterprise Linux 5 guest operating systems are aware of the underlying virtualization technology and can interact with it efficiently using specific interfaces and capabilities. This approach can achieve similar throughput and performance characteristics compared to running on the bare metal system.

As this approach requires modifications in the guest operating system not all operating systems and use models can use para-virtualized virtualization. For operating systems which can not be modified the underlying virtualization infrastructure has to emulate the server hardware (CPU, Memory as well as IO devices for storage and network). Emulation for IO devices can be very slow and will be especially troubling for high-throughput disk and network subsystems. The majority of the performance loss occurs in this area.

The para-virtualized device drivers part of the distributed RPM packages bring many of the performance advantages of para-virtualized guest operating systems to unmodified operating systems because only the para-virtualized device driver (but not the rest of the operating system) is aware of the underlying virtualization platform.

After installing the para-virtualized device drivers, a disk device or network card will continue to appear as a normal, physical disk or network card to the operating system. However, now the device driver interacts directly with the virtualization platform (with no emulation) to efficiently deliver disk and network access, allowing the disk and network subsystems to operate at near native speeds even in a virtualized environment, without requiring changes to existing guest operating systems.

The para-virtualized drivers have certain host requirements. 64 bit hosts can run:

- 32 bit guests.
- 64 bit guests.
- a mixture of 32 bit and 64 bit guests.

14.1. System requirements

This section provides the requirements for para-virtualized drivers with Red Hat Enterprise Linux.

Installation

Before you install the para-virtualized drivers the following requirements (listed below) must be met.

> **Red Hat Enterprise Linux 4.7 and 5.3 and newer**
>
> All version of Red Hat Enterprise Linux from 4.7 and 5.3 have the kernel module for the para-virtualized drivers, the `pv-on-hvm` module, in the default kernel package. That means the para-virtualized drivers are available for Red Hat Enterprise Linux 4.7 and newer or 5.3 and newer guests.

You will need the following RPM packages for para-virtualized drivers for each guest operating system installation.

Minimum host operating system version:

- Red Hat Enterprise Linux 5.1 or newer.

Minimum guest operating system version:

- Red Hat Enterprise Linux 5.1 or newer.
- Red Hat Enterprise Linux 4 Update 6 or newer.
- Red Hat Enterprise Linux 3 Update 9 or newer.

Red Hat Enterprise Linux 5 requires:

- **`kmod-xenpv`**.

Red Hat Enterprise Linux 4 requires:

- **`kmod-xenpv`**,
- **`modules-init-tools`** (for versions prior to Red Hat Enterprise Linux 4.6z you require **`modules-init-tools-3.1-0.pre5.3.4.el4_6.1`** or greater), and
- **`modversions`**.

Red Hat Enterprise Linux 3 requires:
- **kmod-xenpv**.

You require at least 50MB of free disk space in the **/lib** file system.

14.2. Para-virtualization Restrictions and Support

This section outlines support restrictions and requirements for using para-virtualized drivers on Red Hat Enterprise Linux. What we support and the restrictions put upon support can be found in the sections below.

Supported Guest Operating Systems

Support for para-virtualized drivers is available for the following operating systems and versions:

- Red Hat Enterprise Linux 5.1 and newer.
- Red Hat Enterprise Linux 4 Update 6 and newer.
- Red Hat Enterprise Linux 3 Update 9 and newer.

You are supported for running a 32 bit guest operating system with para-virtualized drivers on 64 bit Red Hat Enterprise Linux 5 Virtualization.

The table below indicates the kernel variants supported with the para-virtualized drivers. You can use the command shown below to identify the exact kernel revision currently installed on your host. Compare the output against the table to determine if it is supported.

```
# rpm -q --queryformat '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n' kernel
```

The Red Hat Enterprise Linux 5 i686 and x86_64 kernel variants include Symmetric Multiprocessing(SMP), no separate SMP kernel RPM is required.

Take note of processor specific kernel requirements for Red Hat Enterprise Linux 3 Guests in the table below.

Kernel Architecture	Red Hat Enterprise Linux 3	Red Hat Enterprise Linux 4	Red Hat Enterprise Linux 5
athlon	Supported (AMD)		
athlon-SMP	Supported (AMD)		
i32e	Supported (Intel)		
i686	Supported (Intel)	Supported	Supported
i686-PAE			Supported
i686-SMP	Supported (Intel)	Supported	
i686-HUGEMEM	Supported (Intel)	Supported	
x86_64	Supported (AMD)	Supported	Supported
x86_64-SMP	Supported (AMD)	Supported	
x86_64-LARGESMP		Supported	
Itanium (IA64)			Supported

Table 14.1. Supported guest kernel architectures for para-virtualized drivers

> **Important**
>
> The host system requires Red Hat Enterprise Linux 5.1 or newer.

> **Finding which kernel you are using**
>
> Write the output of the command below down or remember it. This is the value that determines which packages and modules you need to download.
>
> ```
> # rpm -q --queryformat '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n' kernel
> ```
>
> Your output should appear similar to this:
>
> ```
> kernel-PAE-2.6.18-53.1.4.el5.i686
> ```
>
> The name of the kernel is PAE (an abbreviation of Physical Address Extensions), kernel version is 2.6.18, the release is 53.1.4.el5 and the architecture is i686. The kernel rpm should always be in the format **kernel-name-version-release.arch.rpm**.

Important Restrictions

Para-virtualized device drivers can be installed after successfully installing a guest operating system. You will need a functioning host and guest before you can install these drivers.

> **Para-virtualized block devices and `GRUB`**
>
> **GRUB** can not presently, access para-virtualized block devices. Therefore, a guest can not be booted from a device that uses the para-virtualized block device drivers. Specifically, the disk that contains the Master Boot Record(MBR), a disk containing a boot loader (`GRUB`), or a disk that contains the kernel `initrd` images. That is, any disk which contains the `/boot` directory or partition can not use the para-virtualized block device drivers.

Red Hat Enterprise Linux 3 kernel variant architecture dependencies

For Red Hat Enterprise Linux 3 based guest operating systems you must use the processor specific kernel and para-virtualized driver RPMs as seen in the tables below. If you fail to install the matching para-virtualized driver package loading of the `xen-pci-platform` module will fail.

The table below shows which host kernel is required to run a Red Hat Enterprise Linux 3 guest on if the guest was compiled for an Intel processor.

Guest kernel type	Required host kernel type
ia32e (UP and SMP)	x86_64
i686	i686
i686-SMP	i686

Guest kernel type	Required host kernel type
i686-HUGEMEM	i686

Table 14.2. Required host kernel architecture for guests using para-virtualized drivers on Red Hat Enterprise Linux 3 for Intel processors

The table below shows which host kernel is required to run a Red Hat Enterprise Linux 3 guest on if the guest was compiled for an AMD processor.

Guest kernel type	Required host kernel type
athlon	i686
athlon-SMP	i686
x86_64	x86_64
x86_64-SMP	x86_64

Table 14.3. Required host kernel architectures for guests using para-virtualized drivers on Red Hat Enterprise Linux 3 for AMD processors

14.3. Installing the Para-virtualized Drivers

The following three chapters describe how to install and configure your fully virtualized guests to run on Red Hat Enterprise Linux 5.1 or above with para-virtualized drivers.

Verify your architecture is supported before proceeding

Para-virtualized drivers are only supported on certain hardware and version combinations. Verify your hardware and operating system requirements are met before proceeding to install para-virtualized drivers.

Maximizing the benefit of the para-virtualized drivers for new installations

If you are installing a new guest system, in order to gain maximal benefit from the para-virtualized block device drivers, you should create the guest with at least two disks.

Using the para-virtualized drivers for the disk that contains the **`MBR`** and the boot loader (**`GRUB`**), and for the **`/boot`** partition. This partition can be very small, as it only needs to have enough capacity to hold the **`/boot`** partition.

Use the second disk and any additional disks for all other partitions (for example, **`/`**, **`/usr`**) or logical volumes.

Using this installation method, when the para-virtualized block device drivers are later installed after completing the install of the guest, only booting the guest and accessing the **`/boot`** partition will use the virtualized block device drivers.

14.3.1. Common installation steps

The list below covers the high level steps common across all guest operating system versions.

1. Copy the RPMs for your hardware architecture to a suitable location in your guest operating system. Your home directory is sufficient. If you do not know which RPM you require verify against the table at *Section 14.2, “Para-virtualization Restrictions and Support”*.

2. Use the **rpm** utility to install the RPM packages. The **rpm** utility will install the following four new kernel modules into **/lib/modules/[%kversion][%kvariant]/extra/xenpv/%release**:
   - the PCI infrastructure module, **xen-platform-pci.ko**,
   - the ballooning module, **xen-balloon.ko**,
   - the virtual block device module, **xen-vbd.ko**,
   - and the virtual network device module, **xen.vnif.ko**.

3. If the guest operating does not support automatically loading the para-virtualized drivers (for example, Red Hat Enterprise Linux 3) perform the required post-install steps to copy the drivers into the operating system specific locations.

4. Shut down your guest operating system.

5. Reconfigure the guest operating system's configuration file on the host to use the installed para-virtualized drivers.

6. Remove the “type=ioemu” entry for the network device.

7. Add any additional storage entities you want to use for the para-virtualized block device driver.

8. Restart your guest:

   ```
   # virsh start YourGuestName
   ```

   Where *YourGuestName* is the name of the guest operating system.

9. Reconfigure the guest network

14.3.2. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 3

This section contains detailed instructions for the para-virtualized drivers in a Red Hat Enterprise 3 guest operating system.

> **Please note**
>
> These packages do not support booting from a para-virtualized disk. Booting the guest operating system kernel still requires the use of the emulated IDE driver, while any other (non-system) user-space applications and data can use the para-virtualized block device drivers.

Driver Installation

The list below covers the steps to install a Red Hat Enterprise Linux 3 guest with para-virtualized drivers.

1. Copy the **kmod-xenpv** rpm for your hardware architecture and kernel variant to your guest operating system.

2. Use the **rpm** utility to install the RPM packages. Ensure you have correctly identified which package you need for your guest operating system variant and architecture.

```
[root@rhel3]# rpm -ivh kmod-xenpv*
```

3. You need to perform the commands below to enable the correct and automated loading of the para-virtualized drivers. *%kvariant* is the kernel variant the para-virtualized drivers have been build against and *%release* corresponds to the release version of the para-virtualized drivers.

```
[root@rhel3]# mkdir -p /lib/modules/'uname -r'/extra/xenpv
[root@rhel3]# cp -R /lib/modules/2.4.21-52.EL[%kvariant]/extra/xenpv/
%release \
/lib/modules/'uname -r'/extra/xenpv
[root@rhel3]# cd /lib/modules/'uname -r'/extra/xenpv/%release
[root@rhel3]# insmod xen-platform-pci.o
[root@rhel3]# insmod xen-balloon.o`
[root@rhel3]# insmod xen-vbd.o
[root@rhel3]# insmod xen-vnif.o
```

> **Note**
> Warnings will be generated by **insmod** when installing the binary driver modules due to Red Hat Enterprise Linux 3 having **MODVERSIONS** enabled. These warnings can be ignored.

4. Verify **/etc/modules.conf** and make sure you have an alias for `eth0` like the one below. If you are planning to configure multiple interfaces add an additional line for each interface.

```
alias eth0 xen-vnif
```

Edit **/etc/rc.local** and add the line:

```
insmod /lib/modules/'uname -r'/extra/xenpv/%release/xen-vbd.o
```

> **Note**
> Substitute “%release” with the actual release version (for example 0.1-5.el) for the para-virtualized drivers. If you update the para-virtualized driver RPM package make sure you update the release version to the appropriate version.

5. Shutdown the virtual machine (use “**#shutdown -h now**” inside the guest).

6. Edit the guest configuration file in **/etc/xen/YourGuestsName** in the following ways:

   - Remove the “**type=ioemu**” entry from the “**vif=**” entry.

- Add any additional disk partitions, volumes or LUNs to the guest so that they can be accessed via the para-virtualized (**`xen-vbd`**) disk driver.
- For each additional physical device, LUN, partition or volume add an entry similar to the one below to the “**`disk=`**” section in the guest configuration file. The original “**`disk=`**” entry might also look like the entry below.

  ```
  disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w"]
  ```

- Once you have added additional physical devices, LUNs, partitions or volumes; the para-virtualized driver entry in your XML configuration file should resemble the entry shown below.

  ```
  disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
  "tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
  ```

  Note

  Use “**`tap:aio`**” for the para-virtualized device if a file based image is used.

7. Boot the virtual machine using the **`virsh`** command:

   ```
   # virsh start YourGuestName
   ```

Be aware

The para-virtualized drivers are not automatically added and loaded to the system because **`weak-modules`** and **`modversions`** support is not provided in Red Hat Enterprise Linux 3. To insert the module execute the command below.

`insmod xen-vbd.ko`

Red Hat Enterprise Linux 3 requires the manual creation of the special files for the block devices which use **`xen-vbd`**. The steps below will cover how to create and register para-virtualized block devices.

Use the following script to create the special files after the para-virtualized block device driver is loaded.

```
#!/bin/sh
module="xvd"
mode="664"
major=`awk "\\$2==\"$module\" {print \\$1}" /proc/devices`
# < mknod for as many or few partitions on xvd disk attached to FV guest >
# change/add xvda to xvdb, xvbd, etc. for 2nd, 3rd, etc., disk added in
# in xen config file, respectively.
mknod /dev/xvdb b $major 0
mknod /dev/xvdb1 b $major 1
```

```
mknod /dev/xvdb2 b $major 2
chgrp disk /dev/xvd*
chmod $mode /dev/xvd*
```

For each additional virtual disk, increment the minor number by 16. In the example below an additional device, minor number 16, is created.

```
mknod /dev/xvdc b $major 16
mknod /dev/xvdc1 b $major 17
```

This would make the next device 32 which can be created by:

```
mknod /dev/xvdd b $major 32
mknod /dev/xvdd1 b $major 33
```

Now you should verify the partitions which you have created are available.

```
[root@rhel3]# cat /proc/partitions
major   minor     #blocks   name

  3        0     10485760   hda
  3        1       104391   hda1
  3        2     10377990   hda2
202        0        64000   xvdb
202        1        32000   xvdb1
202        2        32000   xvdb2
253        0      8257536   dm-0
253        1      2031616   dm-1
```

In the above output, you can observe that the partitioned device “**xvdb**” is available to the system.

The commands below mount the new block devices to local mount points and updates the **/etc/fstab** inside the guest to mount the devices/partitions during boot.

```
[root@rhel3]# mkdir /mnt/pvdisk_p1
[root@rhel3]# mkdir /mnt/pvdisk_p2
[root@rhel3]# mount /dev/xvdb1 /mnt/pvdisk_p1
[root@rhel3]# mount /dev/xvdb2 /mnt/pvdisk_p2
[root@rhel3]# df /mnt/pvdisk_p1
Filesystem           1K-blocks      Used   Available Use%  Mounted on
/dev/xvdb1               32000        15       31985   1%  /mnt/pvdisk_p1
```

Performance tip

Using a Red Hat Enterprise Linux 5.1 host (dom0), the "**noapic**" parameter should be added to the kernel boot line in your virtual guest's **/boot/grub/grub.conf** entry as seen below. Keep in mind your architecture and kernel version may be different.

kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64 rhgb noapic

A Red Hat Enterprise Linux 5.2 dom0 will not need this kernel parameter for the guest.

Important

The Itanium (ia64) binary RPM packages and builds are not presently available.

14.3.3. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 4

This section contains detailed instructions for the para-virtualized drivers in a Red Hat Enterprise 4 guest operating system.

Please note

These packages do not support booting from a para-virtualized disk. Booting the guest operating system kernel still requires the use of the emulated IDE driver, while any other (non-system) user-space applications and data can use the para-virtualized block device drivers.

Driver Installation

The list below covers the steps to install a Red Hat Enterprise Linux 4 guest with para-virtualized drivers.

1. Copy the **`kmod-xenpv`**, **`modules-init-tools`** and **`modversions`** RPMs for your hardware architecture and kernel variant to your guest operating system.

2. Use the **`rpm`** utility to install the RPM packages. Make sure you have correctly identified which package you need for your guest operating system variant and architecture. An updated module-init-tools is required for this package, it is available with the Red Hat Enterprise Linux 4-6-z kernel or newer.

```
[root@rhel4]# rpm -ivh modversions
[root@rhel4]# rpm -Uvh module-init-tools
[root@rhel4]# rpm -ivh kmod-xenpv*
```

Note

There are different packages for UP, SMP, Hugemem and architectures so make sure you have the right RPMs for your kernel.

3. Execute **`cat /etc/modules.conf`** to verify you have an alias for `eth0` like the one below. If you are planning to configure multiple interfaces add an additional line for each interface. It it does not look like the entry below change it.

```
alias eth0 xen-vnif
```

4. Shutdown the virtual machine (use "**#shutdown -h now**" inside the guest).

5. Edit the guest configuration file in **/etc/xen/YourGuestsName** in the following ways:

   - Remove the "**type=ioemu**" entry from the "**vif=**" entry.

   - Add any additional disk partitions, volumes or LUNs to the guest so that they can be accessed via the para-virtualized (**xen-vbd**) disk driver.

   - For each additional physical device, LUN, partition or volume add an entry similar to the one shown below to the "**disk=**" section in the guest configuration file. The original "**disk=**" entry might also look like the entry below.

   ```
   disk = [ "file:/var/lib/libvirt/images/rhel4_64_fv.dsk,hda,w"]
   ```

   - Once you have added additional physical devices, LUNs, partitions or volumes; the para-virtualized driver entry in your XML configuration file should resemble the entry shown below.

   ```
   disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
   "tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
   ```

   ### Note

   Use "**tap:aio**" for the para-virtualized device if a file based image is used.

6. Boot the virtual machine using the **virsh** command:

```
# virsh start YourGuestName
```

On the first reboot of the virtual guest, **kudzu** will ask you to "*Keep or Delete the Realtek Network device*" and "*Configure the xen-bridge device*". You should configure the **xen-bridge** and delete the Realtek network device.

Performance tip

Using a Red Hat Enterprise Linux 5.1 host (dom0), the "**noapic**" parameter should be added to the kernel boot line in your virtual guest's **/boot/grub/grub.conf** entry as seen below. Keep in mind your architecture and kernel version may be different.

```
kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64
rhgb noapic
```

A Red Hat Enterprise Linux 5.2 dom0 will not need this kernel parameter for the guest.

Now, verify the partitions which you have created are available.

```
[root@rhel4]# cat /proc/partitions
major    minor   #blocks   name

   3        0    10485760  hda
   3        1      104391  hda1
   3        2    10377990  hda2
 202        0       64000  xvdb
 202        1       32000  xvdb1
 202        2       32000  xvdb2
 253        0     8257536  dm-0
 253        1     2031616  dm-1
```

In the above output, you can see the partitioned device “**xvdb**” is available to the system.

The commands below mount the new block devices to local mount points and updates the **/etc/fstab** inside the guest to mount the devices/partitions during boot.

```
[root@rhel4]# mkdir /mnt/pvdisk_p1
[root@rhel4]# mkdir /mnt/pvdisk_p2
[root@rhel4]# mount /dev/xvdb1 /mnt/pvdisk_p1
[root@rhel4]# mount /dev/xvdb2 /mnt/pvdisk_p2
[root@rhel4]# df /mnt/pvdisk_p1
Filesystem           1K-blocks      Used   Available Use%  Mounted on
/dev/xvdb1               32000        15       31985   1%  /mnt/pvdisk_p1
```

Note

This package is not supported for Red Hat Enterprise Linux 4-GA through Red Hat Enterprise Linux 4 update 2 systems and kernels.

Important note

IA64 binary RPM packages and builds are not presently available.

Automatic module loading

The **xen-vbd** driver may not automatically load. Execute the following command on the guest, substituting *%release* with the correct release version for the para-virtualized drivers.

```
# insmod /lib/modules/'uname -r'/weak-updates/xenpv/%release/xen-vbd.ko
```

14.3.4. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 5

This section contains detailed instructions for the para-virtualized drivers in a Red Hat Enterprise 5 guest operating system.

> **Please note**
>
> These packages do not support booting from a para-virtualized disk. Booting the guest operating system kernel still requires the use of the emulated IDE driver, while any other (non-system) user-space applications and data can use the para-virtualized block device drivers.

Driver Installation

The list below covers the steps to install a Red Hat Enterprise Linux 5 guest with para-virtualized drivers.

1. Copy the **`kmod-xenpv`** rpm for your hardware architecture and kernel variant to your guest operating system.

2. Use the **`rpm`** utility to install the RPM packages. Make sure you correctly identify which package you need for your guest operating system variant and architecture.

```
[root@rhel5]# rpm -ivh kmod-xenpv*
```

3. Issue the command below to disable automatic hardware detection inside the guest operating system

```
[root@rhel5]# chkconfig kudzu off
```

4. Execute **`cat /etc/modules.conf`** to verify you have an alias for `eth0` like the one below. If you are planning to configure multiple interfaces add an additional line for each interface. It it does not look like the entry below change it.

```
alias eth0 xen-vnif
```

5. Shutdown the virtual machine (use "**`#shutdown -h now`**" inside the guest).

6. Edit the guest configuration file in **`/etc/xen/<Your GuestsName>`** in the following ways:

   - Remove the "**`type=ioemu`**" entry from the "**`vif=`**" entry.

   - Add any additional disk partitions, volumes or LUNs to the guest so that they can be accessed via the para-virtualized (**`xen-vbd`**) disk driver.

   - For each additional physical device, LUN, partition or volume add an entry similar to the one shown below to the "**`disk=`**" section in the guest configuration file. The original "**`disk=`**" entry might also look like the entry below.

```
disk = [ "file:/var/lib/libvirt/images/rhel4_64_fv.dsk,hda,w"]
```

- Once you have added additional physical devices, LUNs, partitions or volumes; the para-virtualized driver entry in your XML configuration file should resemble the entry shown below.

```
disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
"tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
```

> **Note**
> Use "**tap:aio**" for the para-virtualized device if a file based image is used.

7. Boot the virtual machine using the **virsh** command:

```
# virsh start YourGuestName
```

To verify the network interface has come up after installing the para-virtualized drivers issue the following command on the guest. It should display the interface information including an assigned IP address

```
[root@rhel5]# ifconfig eth0
```

Now, verify the partitions which you have created are available.

```
[root@rhel5]# cat /proc/partitions
major minor  #blocks    name
   3     0   10485760   hda
   3     1     104391   hda1
   3     2   10377990   hda2
 202     0      64000   xvdb
 202     1      32000   xvdb1
 202     2      32000   xvdb2
 253     0    8257536   dm-0
 253     1    2031616   dm-1
```

In the above output, you can see the partitioned device "**xvdb**" is available to the system.

The commands below mount the new block devices to local mount points and updates the **/etc/fstab** inside the guest to mount the devices/partitions during boot.

```
[root@rhel5]# mkdir /mnt/pvdisk_p1
[root@rhel5]# mkdir /mnt/pvdisk_p2
[root@rhel5]# mount /dev/xvdb1 /mnt/pvdisk_p1
[root@rhel5]# mount /dev/xvdb2 /mnt/pvdisk_p2
[root@rhel5]# df /mnt/pvdisk_p1
Filesystem           1K-blocks      Used   Available Use%  Mounted on
```

```
/dev/xvdb1                32000          15        31985    1%   /mnt/pvdisk_p1
```

Performance tip

Using a Red Hat Enterprise Linux 5.1 host (`dom0`), the "**`noapic`**" parameter should be added to the kernel boot line in your virtual guest's **`/boot/grub/grub.conf`** entry as seen below. Keep in mind your architecture and kernel version may be different.

```
kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64
rhgb noapic
```

A Red Hat Enterprise Linux 5.2 dom0 will not need this kernel parameter for the guest.

14.4. Para-virtualized Network Driver Configuration

Once the para-virtualized network driver is loaded you may need to reconfigure the guest's network interface to reflect the driver and virtual Ethernet card change.

Perform the following steps to reconfigure the network interface inside the guest.

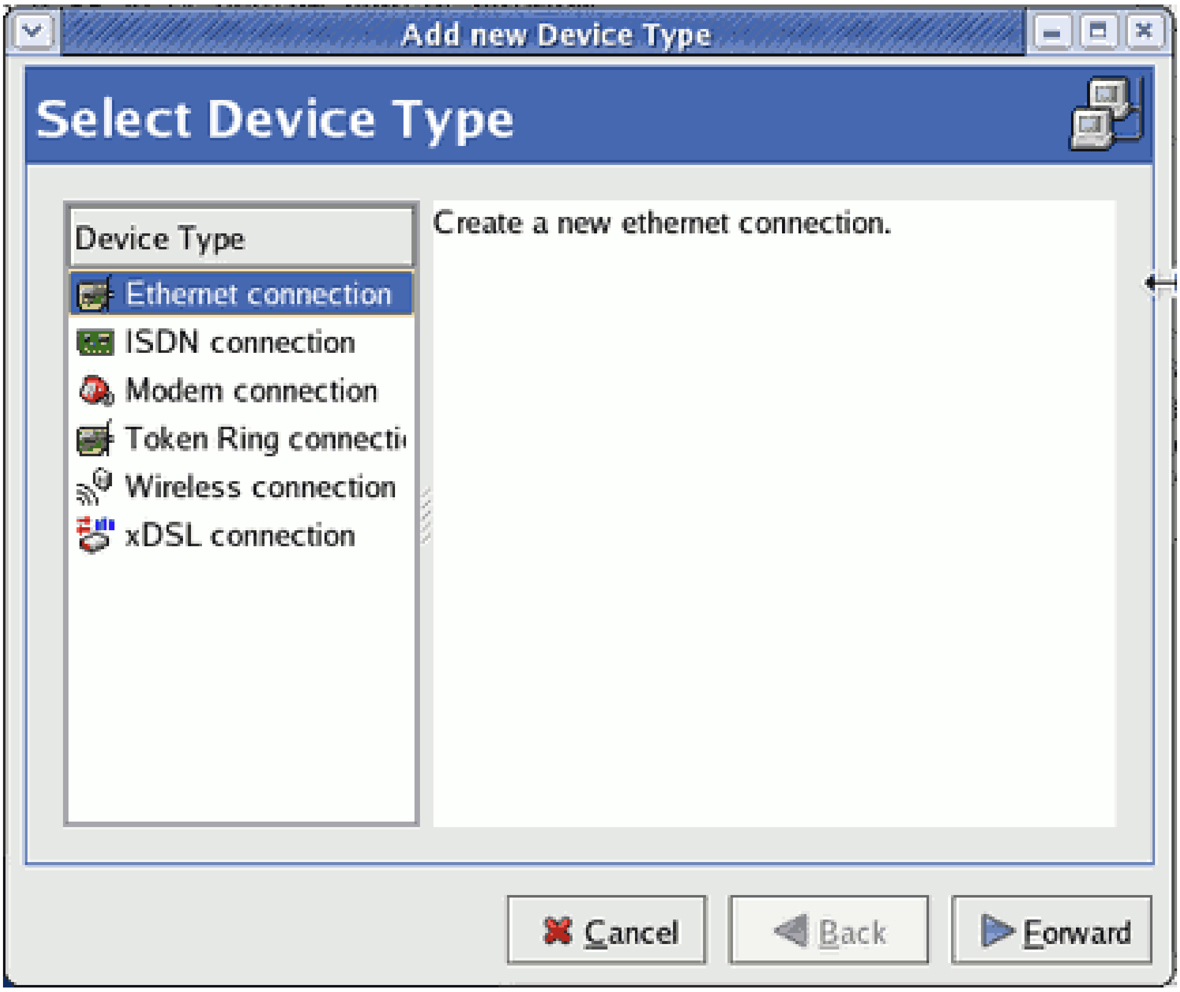
1. In **`virt-manager`** open the console window for the guest and log in as `root`.
2. On Red Hat Enterprise Linux 4 verify the file **`/etc/modprobe.conf`** contains the line "**`alias eth0 xen-vnif`**".

   ```
   # cat /etc/modprobe.conf
   alias eth0 xen-vnif
   ```

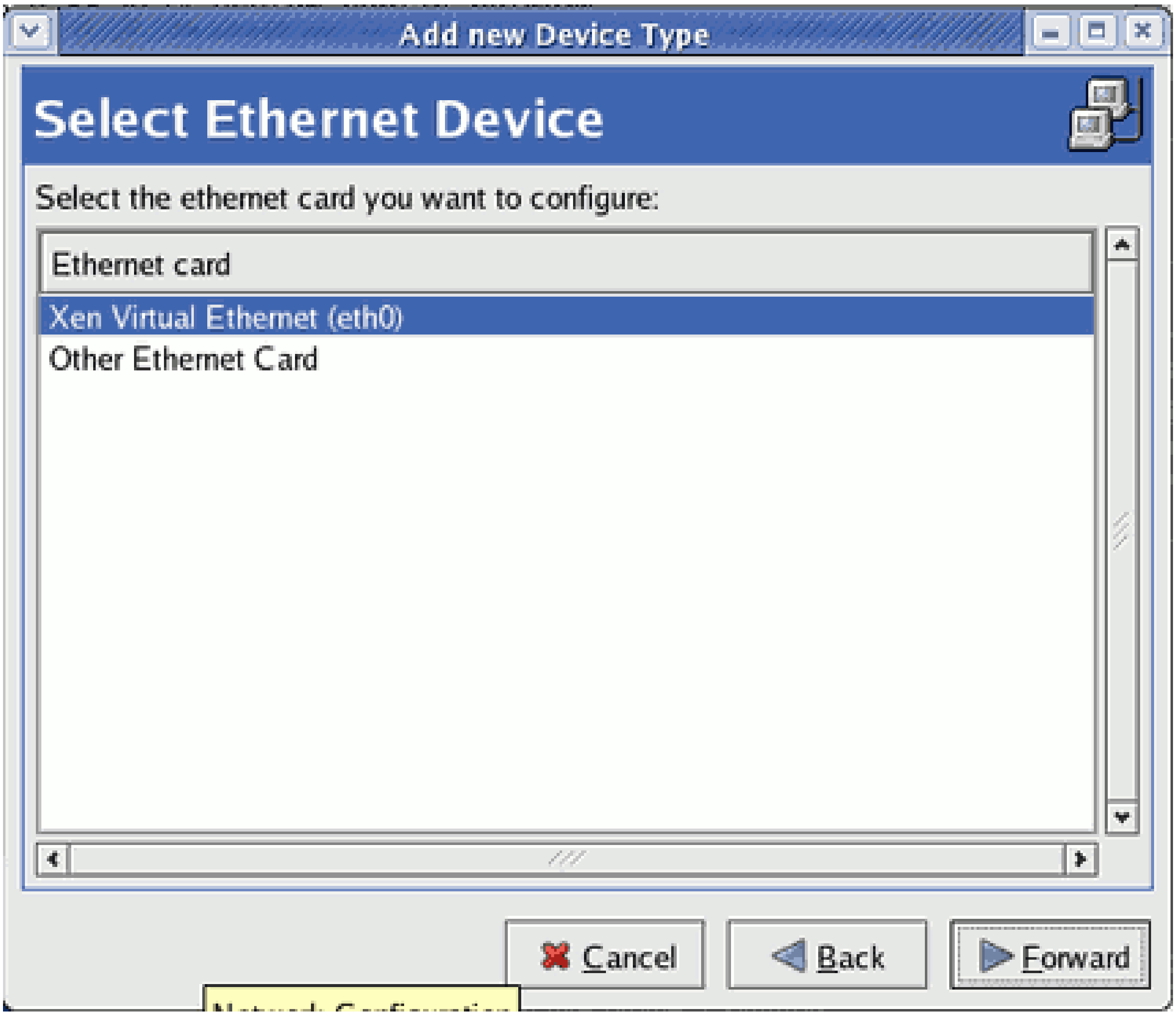
3. To display the present settings for **`eth0`** execute "**`# ifconfig eth0`**". If you receive an error about the device not existing you should load the modules manually as outlined in *Section 32.4, "Manually loading the para-virtualized drivers"*.

   ```
   ifconfig eth0
   eth0      Link encap:Ethernet  HWaddr 00:00:00:6A:27:3A
             BROADCAST MULTICAST  MTU:1500  Metric:1
             RX packets:630150 errors:0 dropped:0 overruns:0 frame:0
             TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:109336431 (104.2 MiB)  TX bytes:846 (846.0 b)
   ```

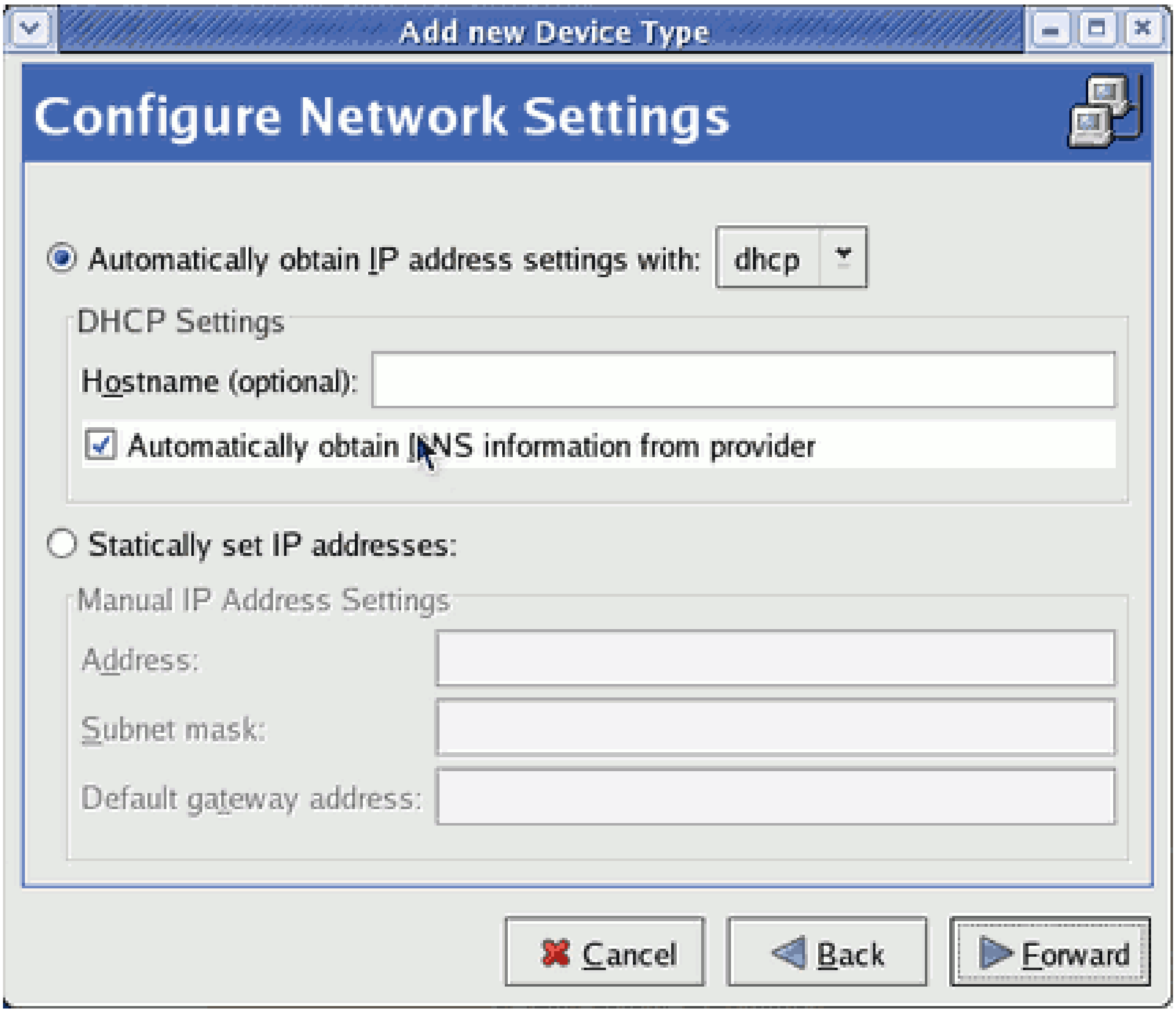
4. Start the network configuration utility(NetworkManager) with the command "**`# system-config-network`**". Click on the "**Forward**" button to start the network card configuration.



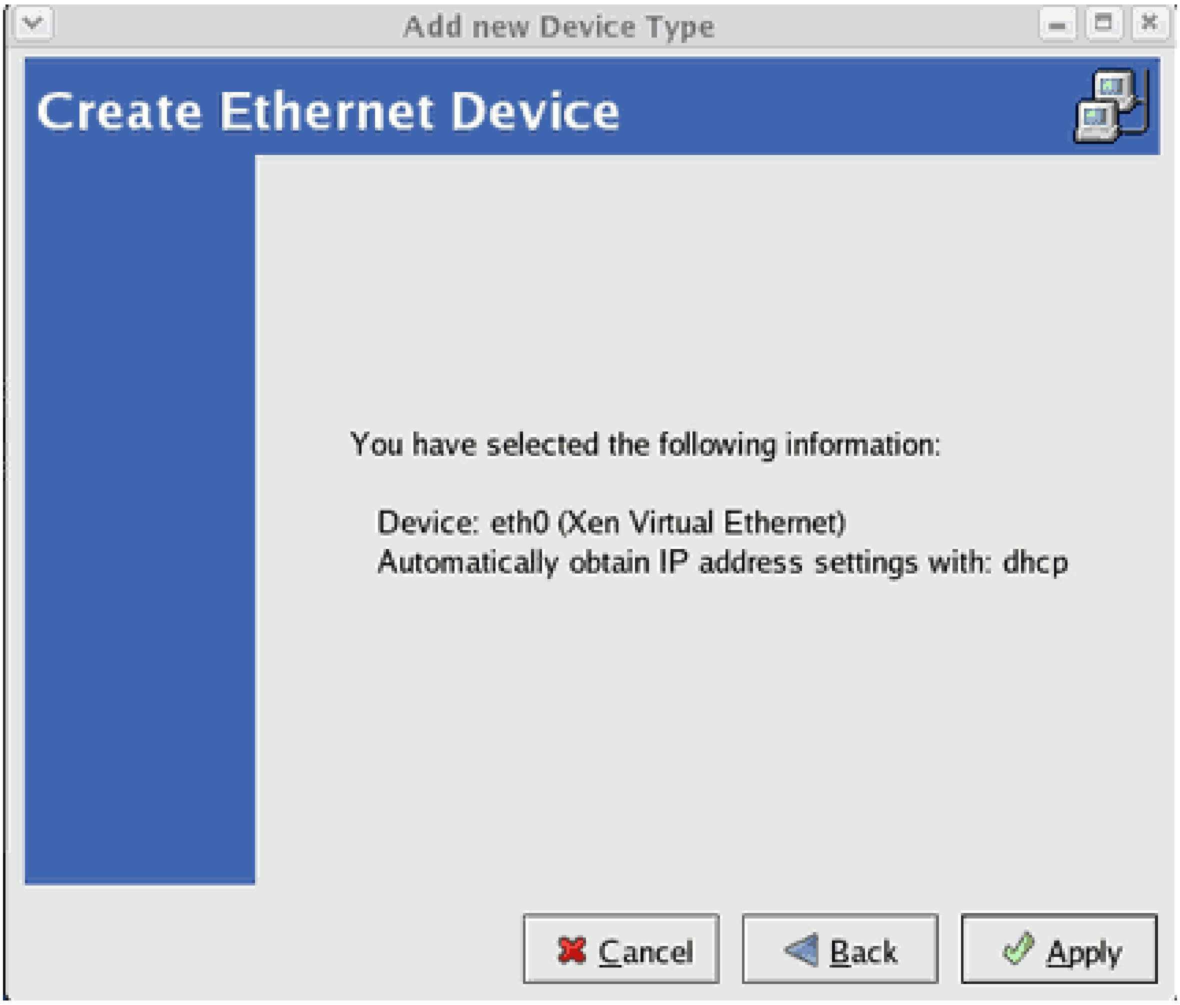
5. Select the '**Xen Virtual Ethernet Card (eth0)**' entry and click '**Forward**'.



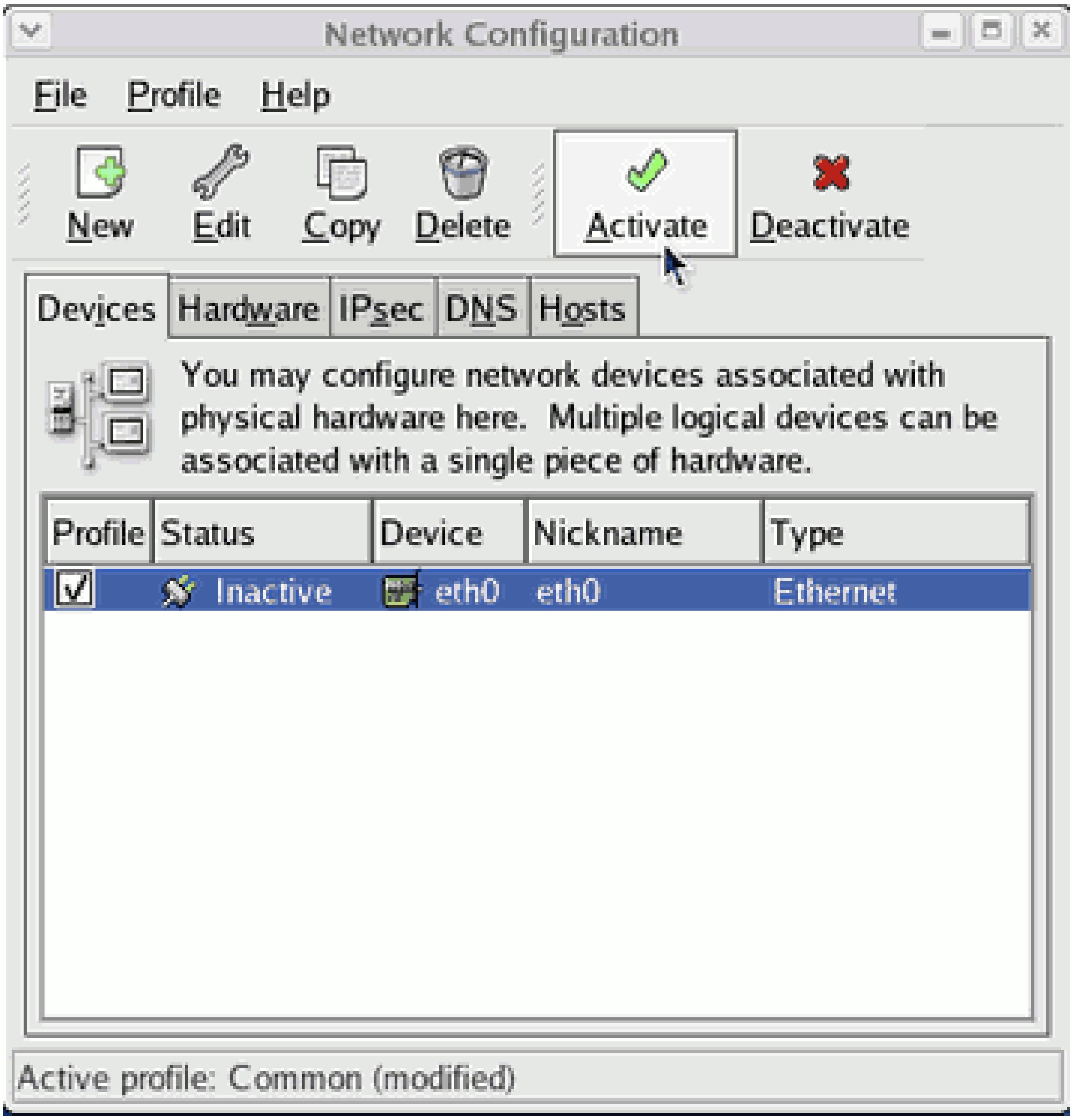
Configure the network settings as required.



6. Complete the configuration by pressing the '**Apply**' button.



7. Press the '**Activate**' button to apply the new settings and restart the network.



8. You should now see the new network interface with an IP address assigned.

```
ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:16:3E:49:E4:E0
          inet addr:192.168.78.180  Bcast:192.168.79.255
 Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:630150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:501209 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109336431 (104.2 MiB)  TX bytes:46265452 (44.1 MiB)
```

14.5. Additional Para-virtualized Hardware Configuration

This section will explain how to add additional virtual network or storage to a guest operating system. For more details on configuring network and storage resources on Red Hat Enterprise Linux 5 Virtualization read the document available on *Emerging Technologies, Red Hat.com*[1]

14.5.1. Virtualized Network Interfaces

Perform the following steps to configure additional network devices for your guest.

Edit your guest configuration file in **/etc/xen/YourGuestName** replacing **YourGuestName** with the name of your guest.

The original entry may look like the one below.

```
vif = [ "mac=00:16:3e:2e:c5:a9,bridge=xenbr0" ]
```

Add an additional entry to the “**vif=**” section of the configuration file similar to the one seen below.

```
vif = [ "mac=00:16:3e:2e:c5:a9,bridge=xenbr0",
    "mac=00:16:3e:2f:d5:a9,bridge=xenbr0" ]
```

Make sure you generate a unique MAC address for the new interface. You can use the command below.

```
# echo 'import virtinst.util ; print virtinst.util.randomMAC()' | python
```

After the guest has been rebooted perform the following step in the guest operating system. Verify the update has been added to your **/etc/modules.conf** in Red Hat Enterprise Linux 3 or **/etc/modprobe.conf** in Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5. Add a new alias for each new interface you added.

```
alias eth1 xen-vnif
```

Now test that each new interface you added make sure it is available inside the guest.

```
# ifconfig eth1
```

The command above should display the properties of **eth1**, repeat the command for **eth2** if you added a third interface, and so on.

Now you can configure the new network interfaces using **redhat-config-network** or Red Hat Enterprise Linux3 or **system-config-network** on Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5.

14.5.2. Virtual Storage Devices

Perform the following steps to configure additional virtual storage devices for your guest.

[1] http://et.redhat.com/~jmh/docs/Installing_RHEL5_Virt.pdf

Edit your guest configuration file in **/etc/xen/YourGuestName** replacing **YourGuestName** with the name of your guest. The original entry may look like the one below.

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w"]
```

Now, add an additional entry for your new physical device, LUN, partition or volume to the "**disk=**" parameter in the configuration file. Storage entities which use the para-virtualized driver resemble the entry below. The "**tap:aio**" parameter instructs the hypervisor to use the para-virtualized driver.

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage1.dsk,xvda,w" ]
```

If you want to add more entries just add them to the "**disk=**" section as a comma separated list.

Note

You need to increment the letter for the '**xvd**' device, that is for your second storage entity it would be '**xvdb**' instead of '**xvda**'.

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage1.dsk,xvda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage2.dsk,xvdb,w" ]
```

Verify the partitions have been created and are available.

```
# cat /proc/partitions
major minor  #blocks    name
   3     0  10485760    hda
   3     1    104391    hda1
   3     2  10377990    hda2
 202     0     64000    xvda
 202     1     64000    xvdb
 253     0   8257536    dm-0
 253     1   2031616    dm-1
```

In the above output you can see the partition or device "**xvdb**" is available to the system.

Mount the new devices and disks to local mount points and update the **/etc/fstab** inside the guest to mount the devices and partitions at boot time.

```
# mkdir /mnt/pvdisk_xvda
# mkdir /mnt/pvdisk_xvdb
# mount /dev/xvda /mnt/pvdisk_xvda
# mount /dev/xvdb /mnt/pvdisk_xvdb
# df /mnt
Filesystem           1K-blocks      Used   Available Use%  Mounted on
/dev/xvda                64000        15       63985   1%  /mnt/pvdisk_xvda
/dev/xvdb                64000        15       63985   1%  /mnt/pvdisk_xvdb
```

KVM Para-virtualized Drivers

Para-virtualized drivers are available for virtualized Windows guests running on KVM hosts. These para-virtualized drivers are included in the virtio package. The virtio package supports block (storage) devices and network interface controllers.

Para-virtualized drivers enhance the performance of fully virtualized guests. With the para-virtualized drivers guest I/O latency decreases and throughput increases to near bare-metal levels. It is recommended to use the para-virtualized drivers for fully virtualized guests running I/O heavy tasks and applications.

The KVM para-virtualized drivers are automatically loaded and installed on Red Hat Enterprise Linux 4.8 and newer and Red Hat Enterprise Linux 5.3 and newer. Those Red Hat Enterprise Linux versions detect and install the drivers so additional installation steps are not required.

As with the KVM module, the virtio drivers are only available on hosts running Red Hat Enterprise Linux 5.4 and newer.

Note

There are only 28 PCI slots available for additional devices per guest. Every para-virtualized network or block device uses one slot. Each guest can use up to 28 additional devices made up of any combination of para-virtualized network, para-virtualized disk devices, or other PCI devices using VTd.

The following Microsoft Windows versions have supported KVM para-virtualized drivers:

- Windows XP,
- Windows Server 2003,
- Windows Vista, and
- Windows Server 2008.

Windows XP is not supported running the virtio para-virtualized block driver. Only the virtio network driver is supported on Windows XP.

15.1. Installing the KVM Windows para-virtualized drivers

This section covers the installation process for the KVM Windows para-virtualized drivers. The KVM para-virtualized drivers can be loaded during the Windows installation or installed after the guest is installed.

You can install the para-virtualized drivers on your guest by one of the following methods:

- hosting the installation files on a network accessible to the guest,
- using a virtualized CD-ROM device of the driver installation disk .iso file, or
- using a virtualized floppy device to install the drivers during boot time (for Windows guests).

This guide describes installation from the para-virtualized installer disk as a virtualized CD-ROM device.

1. **Download the drivers**
   Download the *virtio-win* package (in the Red Hat Enterprise Linux Supplementary channel) with the **yum** command.

   ```
   # yum install virtio-win
   ```

   The drivers are also available on the Red Hat Enterprise Linux Supplementary disc or from Microsoft (*windowsservercatalog.com*[1]). Note that, Red Hat Enterprise Virtualization Hypervisor 5.4 and Red Hat Enterprise Linux 5.4 are based on the same code base.

   The *virtio-win* package installs a CD-ROM image, **virtio-win.iso**, in the **/usr/share/virtio-win/** directory.

2. **Install the para-virtualized drivers**
   It is recommended to install the drivers on the guest before attaching or modifying a device to use the para-virtualized drivers.

   For block devices storing root file systems or other block devices required for booting the guest, the drivers must be installed before the device is modified. If the drivers are not installed on the guest and the driver is set to the virtio driver the guest will not boot.

Mounting the image with virt-manager

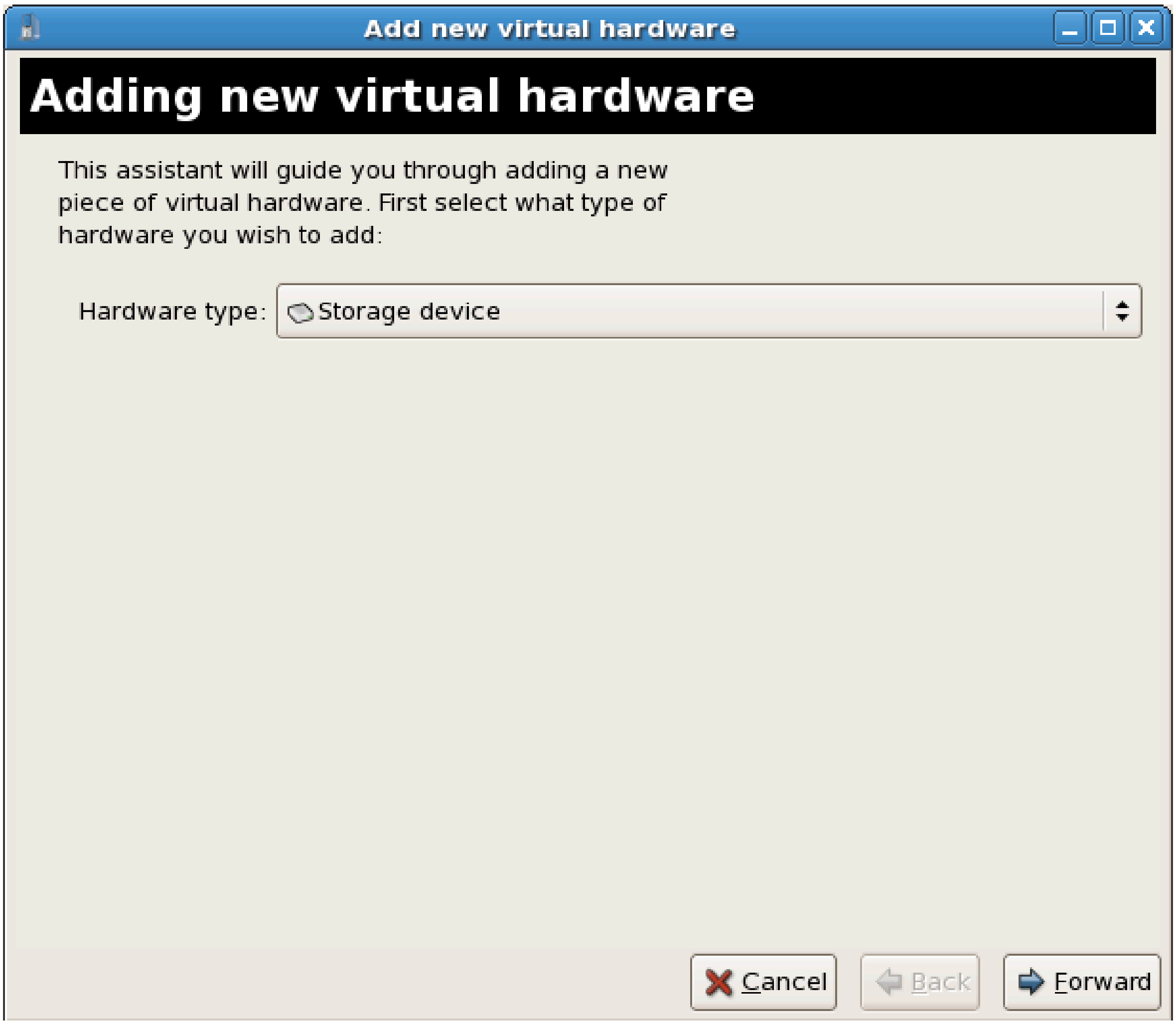
Follow *Using **virt-manager** to mount a CD-ROM image for a Windows guest* to add a CD-ROM image with **virt-manager**.

Procedure 15.1. Using **virt-manager** to mount a CD-ROM image for a Windows guest

1. Open **virt-manager**, select your virtualized guest from the list of virtual machines and press the **Details** button.

2. Click the **Add** button in the **Details** panel.

3. This opens a wizard for adding the new device. Select **Storage device** from the drop down menu, then click **Forward**.

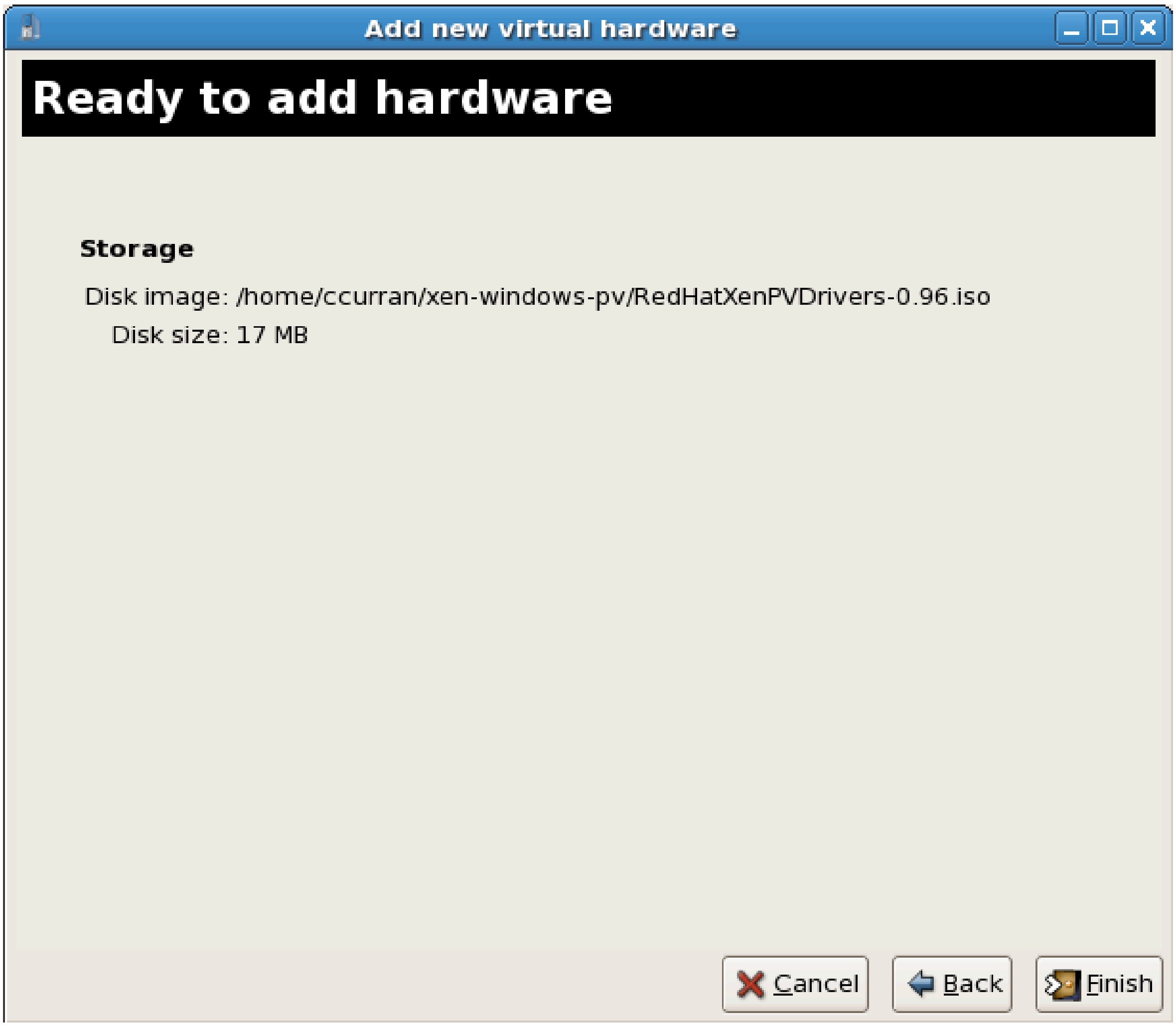


4. Choose the **File (disk image)** option and set the file location of the para-virtualized drivers .iso file. The location of the .iso files is **`/usr/share/xenpv-win`** if you used **`yum`** to install the para-virtualized driver packages.

   If the drivers are stored on a physical CD, use the **Normal Disk Partition** option.

   Set the **Device type** to **IDE cdrom** and click **Forward** to proceed.

5. The disk has been assigned and is available for the guest once the guest is started. Click **Finish** to close the wizard or back if you made a mistake.



Installing with a virtualized floppy disk

This procedure covers installing the para-virtualized drivers during a Windows installation.

- Upon installing the Windows VM for the first time using the run-once menu attach **`viostor.vfd`** as a floppy

  a. **Windows Server 2003**
     When windows prompts to press F6 for third party drivers, do so and follow the onscreen instructions.

  b. **Windows Server 2008**
     When the installer prompts you for the driver, click on **Load Driver**, point the installer to drive A: and pick the driver that suits your guest operating system and architecture.

Using KVM para-virtualized drivers for existing devices

Modify an existing hard disk device attached to the guest to use the **`virtio`** driver instead of virtualized IDE driver. This example edits libvirt configuration files. Alternatively, **`virt-manager`**,

`virsh attach-disk` or **`virsh attach-interface`** can add a new device using the para-virtualized drivers *Using KVM para-virtualized drivers for new devices*.

1. Below is a file-based block device using the virtualized IDE driver. This is a typical entry for a virtualized guest not using the para-virtualized drivers.

```
<disk type='file' device='disk'>
   <source file='/var/lib/libvirt/images/disk1.img'/>
   <target dev='hda' bus='ide'/>
</disk>
```

2. Change the entry to use the para-virtualized device by modifying the **bus=** entry to **`virtio`**.

```
<disk type='file' device='disk'>
   <source file='/var/lib/libvirt/images/disk1.img'/>
   <target dev='hda' bus='virtio'/>
</disk>
```

Using KVM para-virtualized drivers for new devices

This procedure covers creating new devices using the KVM para-virtualized drivers with **`virt-manager`**.

Alternatively, the **`virsh attach-disk`** or **`virsh attach-interface`** commands can be used to attach devices using the para-virtualized drivers.

> **Install the drivers first**
>
> Ensure the drivers have been installed on the Windows guest before proceeding to install new devices. If the drivers are unavailable the device will not be recognized and will not work.

1. Open the virtualized guest by double clicking on the name of the guest in **`virt-manager`**.

2. Open the **Hardware** tab.

3. Press the **Add Hardware** button.

4. In the Adding Virtual Hardware tab select **Storage** or **Network** for the type of device.

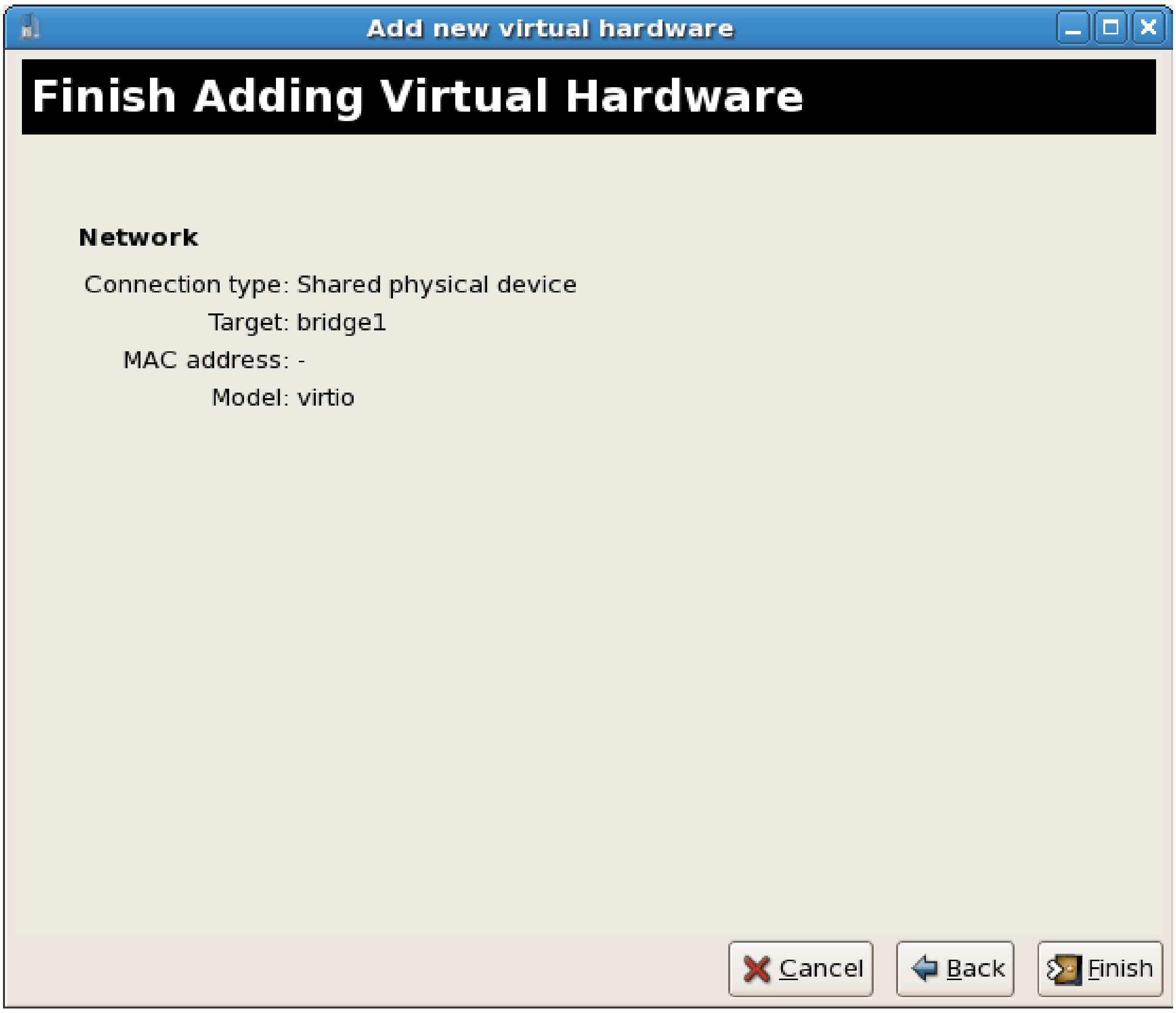
   1. **New disk devices**

      Select the storage device or file based image. Select **Virtio Disk** as the **Device type** and press **Forward**.

   2. **New network devices**

      Select **Virtual network** or **Shared physical device**. Select **virtio** as the **Device type** and press **Forward**.

5. Press **Finish** to save the device.



6. Reboot the guest. The device may to be recognized by the Windows guest until it restarts.

Part IV. Administration

Administering virtualized systems

These chapters contain information for administering host and virtualized guests using tools included in Red Hat Enterprise Linux.

Managing guests with xend

The **xend** node control daemon performs certain system management functions that relate to virtual machines. This daemon controls the virtualized resources, and **xend** must be running to interact with virtual machines. Before you start **xend**, you must specify the operating parameters by editing the **xend** configuration file **`/etc/xen/xend-config.sxp`**. Here are the parameters you can enable or disable in the **`xend-config.sxp`** configuration file:

Item	Description
(console-limit)	Determines the console server's memory buffer limit xend_unix_server and assigns values on a per domain basis.
(min-mem)	Determines the minimum number of megabytes that is reserved for domain0 (if you enter 0, the value does not change).
(dom0-cpus)	Determines the number of CPUs in use by domain0 (at least 1 CPU is assigned by default).
(enable-dump)	Determines that a crash occurs then enables a dump (the default is 0).
(external-migration-tool)	Determines the script or application that handles external device migration. Scripts must reside in **`etc/xen/scripts/external-device-migrate`**.
(logfile)	Determines the location of the log file (default is **`/var/log/xend.log`**).
(loglevel)	Filters out the log mode values: DEBUG, INFO, WARNING, ERROR, or CRITICAL (default is DEBUG).
(network-script)	Determines the script that enables the networking environment (scripts must reside in **`etc/xen/scripts`** directory).
(xend-http-server)	Enables the http stream packet management server (the default is no).
(xend-unix-server)	Enables the unix domain socket server, which is a socket server is a communications endpoint that handles low level network connections and accepts or rejects incoming connections. The default value is set to yes.
(xend-relocation-server)	Enables the relocation server for cross-machine migrations (the default is no).
(xend-unix-path)	Determines the location where the **`xend-unix-server`** command outputs data (default is **`var/lib/xend/xend-socket`**)
(xend-port)	Determines the port that the http management server uses (the default is 8000).

Item	Description
(xend-relocation-port)	Determines the port that the relocation server uses (the default is 8002).
(xend-relocation-address)	Determines the host addresses allowed for migration. The default value is the value of `xend-address`.
(xend-address)	Determines the address that the domain socket server binds to. The default value allows all connections.

Table 16.1. **xend** configuration parameters

After setting these operating parameters, you should verify that xend is running and if not, initialize the daemon. At the command prompt, you can start the **xend** daemon by entering the following:

```
service xend start
```

You can use **xend** to stop the daemon:

```
service xend stop
```

This stops the daemon from running.

You can use **xend** to restart the daemon:

```
service xend restart
```

The daemon starts once again.

You check the status of the **xend** daemon.

```
service xend status
```

The output displays the daemon's status.

Enabling xend at boot time

Use the **`chkconfig`** command to add the `xend` to the **`initscript`**.

`chkconfig --level 345 xend`

The the `xend` will now start at runlevels 3, 4 and 5.

KVM guest timing management

KVM uses the constant Time Stamp Counter (TSC) feature of many modern CPUs. Some CPUs do not have a constant Time Stamp Counter which will affect the way guests running on KVM keep time. Guest's running without accurate timekeeping can have serious affects on some networked applications as your guest will run faster or slower than the actual time.

Guests can have several problems caused by inaccurate clocks and counters:

- Clocks can fall out of synchronization with the actual time which invalidates sessions and affects networks.
- Guests with slower clocks may have issues migrating.
- Guests may stop or crash.

These problems exist on other virtualization platforms and timing should always be tested.

NTP

The Network Time Protocol (NTP) daemon should be running on the host and the guests. Enable the `ntpd` service:

```
# service ntpd start
```

Add the ntpd service to the default startup sequence:

```
# chkconfig ntpd on
```

Using the `ntpd` service should minimize the affects of clock skew in all cases.

Determining if your CPU has the constant Time Stamp Counter

Your CPU has a constant Time Stamp Counter if the `constant_tsc` flag is present. To determine if your CPU has the `constant_tsc` flag run the following command:

```
$ cat /proc/cpuinfo | grep constant_tsc
```

If any output is given your CPU has the `constant_tsc` bit. If no output is given follow the instructions below.

Configuring hosts without a constant Time Stamp Counter

Systems without constant time stamp counters require additional configuration. Power management features interfere with accurate time keeping and must be disabled for guests to accurately keep time with KVM.

> **Note**
>
> These instructions are for AMD revision F cpus only.

If the CPU lacks the `constant_tsc` bit, disable all power management features (*BZ#513138*[1]). Each system has several timers it uses to keep time. The TSC is not stable on the host, which is sometimes caused by **cpufreq** changes, deep C state, or migration to a host with a faster TSC. To stop deep C states, which cam stop the TSC, add "**processor.max_cstate=1**" to the kernel boot options in grub on the host:

```
term Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
 kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/LogVol00 rhgb
 quiet processor.max_cstate=1
```

Disable **cpufreq** (only necessary on hosts without the **constant_tsc**) by editing the **/etc/sysconfig/cpuspeed** configuration file and change the **MIN_SPEED** and **MAX_SPEED** variables to the highest frequency available. Valid limits can be found in the **/sys/devices/system/cpu/cpu*/cpufreq/scaling_available_frequencies** files.

Using the para-virtualized clock with Red Hat Enterprise Linux guests

For certain Red Hat Enterprise Linux guests, additional kernel parameters are required. These parameters can be set by appending them to the end of the /kernel line in the /boot/grub/grub.conf file of the guest.

The table below lists versions of Red Hat Enterprise Linux and the parameters required for guests on systems without a constant Time Stamp Counter.

Red Hat Enterprise Linux	Additional guest kernel parameters
5.4 AMD64/Intel 64 with the para-virtualized clock	Additional parameters are not required
5.4 AMD64/Intel 64 without the para-virtualized clock	divider=10 notsc lpj=n
5.4 x86 with the para-virtualized clock	Additional parameters are not required
5.4 x86 without the para-virtualized clock	divider=10 clocksource=acpi_pm lpj=n
5.3 AMD64/Intel 64	divider=10 notsc
5.3 x86	divider=10 clocksource=acpi_pm
4.8 AMD64/Intel 64	notsc divider=10
4.8 x86	clock=pmtmr divider=10
3.9 AMD64/Intel 64	Additional parameters are not required
3.9 x86	Additional parameters are not required

[1] https://bugzilla.redhat.com/show_bug.cgi?id=513138

Using the para-virtualized clock with Windows guests

Enable the para-virtualized clock on Window guests by editing the boot parameters. Windows boot settings are stored in the boot.ini file. To enable the para-virtualized clock add the following line:

```
/use pmtimer
```

For more information on Windows boot settings and the pmtimer option, refer to *Available switch options for the Windows XP and the Windows Server 2003 Boot.ini files*[2].

[2] http://support.microsoft.com/kb/833721

Xen live migration

The Xen hypervisor supports *migration* for para-virtualized guests and fully virtualized guests. Migration is only supported on Red Hat Enterprise Linux 5.1 and newer systems. Migration can be performed offline or live.

- Offline migration suspends the virtualized guest on the original host, transfers it to the destination host and then resumes it once the guest is fully transferred. Offline migration uses the **`virsh migrate`** command.

```
# virsh migrate GuestName libvirtURI
```

- Live migration keeps the guest running on the source host and begins moving the memory without stopping the guest. All modified memory pages are tracked and sent to the destination after the image is sent. The memory is updated with the changed pages. The process continues until it reaches some heuristic; either it successfully copied all the pages over, or the source is changing too fast and the destination host cannot make progress. If the heuristic is met the guest is briefly paused on the source host and the registers and buffers are sent. The registers are loaded on the new host and the guest is then resumed on the destination host. If the guest is cannot be merged (which happens when guests are under extreme loads) the guest is paused and then an offline migration is started instead.

  Live migration uses the **`--live`** option for the **`virsh migrate`** command.

```
# virsh migrate--live GuestName libvirtURI
```

Itanium® support note

Migration is presently unsupported on the Itanium® architecture.

To enable migration with Xen a few changes must be made to the **`/etc/xen/xend-config.sxp`** configuration file. By default, migration is disabled as migration can be a potential security hazard if incorrectly configured. Opening the migration port can allow an unauthorized host to initiate a migration or connect to the migration ports. Authentication and authorization are not configured for migration requests and the only control mechanism is based on hostnames and IP addresses. Special care should be taken to ensure the migration port is not accessible to unauthorized hosts.

Virtualization migration security

IP address and hostname filters only offer minimal security. Both of these attributes can be forged if the attacker knows the address or hostname of the migration client. The best method for securing migration is to isolate the network from external and unauthorized internal connections.

Enabling migration

Modify the following entries in **`/etc/xen/xend-config.sxp`** to enable migration. Modify the values, when necessary, and remove the comments (the **#** symbol) preceding the following parameters:

`(xend-relocation-server yes)`
: The default value, which disables migration, is no. Change the value of **`xend-relocation-server`** to yes to enable migration.

`(xend-relocation-port 8002)`
: The parameter, `(xend-relocation-port)`, specifies the port `xend` should use for the relocation interface, if `xend-relocation-server` is set to `yes`

  The default value of this variable should work for most installations. If you change the value make sure you are using an unused port on the relocation server.

  The port set by the **`xend-relocation-port`** parameter must be open on both systems.

`(xend-relocation-address '')`
: `(xend-relocation-address)`is the address the `xend` listens for migration commands on the **`relocation-socket`** connection if `xend-relocation-server` is set.

  The default is listen on all active interfaces, the parameter can restrict the relocation server to only listen to a specific interface. The default value in **`/etc/xen/xend-config.sxp`** is an empty string(*''*). This value should be replaced with a single, valid IP address. For example:

```
(xend-relocation-address '10.0.0.1')
```

`(xend-relocation-hosts-allow '')`
: The `(xend-relocation-hosts-allow 'hosts')` parameter controls which hostnames can communicate on the relocation port.

  Unless you are using SSH or TLS, the guest's virtual memory is transferred in raw form without encryption of the communication. Modify the `xend-relocation-hosts-allow` option to restrict access to the migration server.

  If the value is empty, as denoted in the example above by an empty string surrounded by single quotes, then all connections are allowed. This assumes the connection arrives on a port and interface which the relocation server listens on, see also `xend-relocation-port` and `xend-relocation-address` above).

  Otherwise, the `(xend-relocation-hosts-allow)` parameter should be a sequence of regular expressions separated by spaces. Any host with a fully-qualified domain name or an IP address which matches one of these regular expressions will be accepted.

  An example of a `(xend-relocation-hosts-allow)` attribute:

```
(xend-relocation-hosts-allow '^localhost$ ^localhost\\.localdomain$')
```

After you have configured the parameters in your configuration file, restart the Xen service.

```
# service xend restart
```

18.1. A live migration example

Below is an example of how to setup a simple environment for live migration. This configuration is using **NFS** for the shared storage. **NFS** is suitable for demonstration environments but for a production environment a more robust shared storage configuration using Fibre Channel or iSCSI and **GFS** is recommended.

The configuration below consists of two servers (`et-virt07` and `et-virt08`), both of them are using **`eth1`** as their default network interface hence they are using **`xenbr1`** as their Xen networking bridge. We are using a locally attached SCSI disk (**`/dev/sdb`**) on `et-virt07` for shared storage using **NFS**.

Setup for live migration

Create and mount the directory used for the migration:

```
# mkdir /var/lib/libvirt/images
# mount /dev/sdb /var/lib/libvirt/images
```

Important

Ensure the directory is exported with the correct options. If you are exporting the default directory **`/var/lib/libvirt/images/`** make sure you *only* export **`/var/lib/libvirt/images/`** and *not* **`/var/lib/xen/`** as this directory is used by the `xend` daemon and other tools. Sharing **`/var/lib/xen/`** will cause unpredictable behavior.

```
# cat /etc/exports
/var/lib/libvirt/images  *(rw,async,no_root_squash)
```

Verify it is exported via **NFS**:

```
# showmount -e et-virt07
Export list for et-virt07:
/var/lib/libvirt/images *
```

Install the guest

The install command in the example used for installing the guest:

```
# virt-install -p -f /var/lib/libvirt/images/testvm1.dsk -s 5 -n\
testvm1 --vnc -r 1024 -l http://porkchop.devel.redhat.com/rel-eng/RHEL5-\
Server-20070105.0/4.92/x86_64/os/ -b xenbr1
```

For step by step installation instructions, refer to *Chapter 7, Guest operating system installation procedures*

Verify environment for migration

Make sure the virtualized network bridges are configured correctly and have the same name on both hosts:

```
[et-virt08 ~]# brctl show
```

```
bridge name     bridge id               STP enabled     interfaces
xenbr1          8000.feffffffffff       no              peth1
vif0.1
```

```
[et-virt07 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
xenbr1          8000.feffffffffff       no              peth1
vif0.1
```

Verify the relocation parameters are configured on both hosts:

```
[et-virt07 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

```
[et-virt08 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

Make sure the relocation server has started and is listening on the dedicated port for Xen migrations (8002):

```
[et-virt07 ~]# lsof -i :8002
COMMAND  PID  USER   FD   TYPE  DEVICE SIZE NODE NAME
python 3445 root 14u IPv4 10223 TCP *:teradataordbms (LISTEN)
```

```
[et-virt08 ~]# lsof -i :8002
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
python 3252 root 14u IPv4 10901 TCP *:teradataordbms (LISTEN)
```

Verify the NFS directory has been mounted on the other host and you can see and access the virtual machine image and file system:

```
[et-virt08 ~]# df /var/lib/libvirt/images
Filesystem           1K-blocks      Used Available Use% Mounted on
et-virt07:/var/lib/libvirt/images    70562400   2379712  64598336   4% /
var/lib/libvirt/images
```

```
[et-virt08 ~]# file /var/lib/libvirt/images/testvm1.dsk
/var/lib/libvirt/images/testvm1.dsk: x86 boot sector; partition 1: ID=0x83,
active, starthead 1, startsector 63, 208782 sectors; partition 2: ID=0x8e,
starthead 0, startsector 208845, 10265535 sectors, code offset 0x48
```

```
[et-virt08 ~]# touch /var/lib/libvirt/images/foo
[et-virt08 ~]# rm -f /var/lib/libvirt/images/foo
```

Verification of save and restore on local host

Start up the virtual machine (if it has not yet):

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

```
[et-virt07 ~]# virsh start testvm1
Domain testvm1 started
```

Verify the virtual machine is running:

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

Save the virtual machine on the local host:

```
[et-virt07 images]# time virsh save testvm1 testvm1.sav
real    0m15.744s
user    0m0.188s
sys     0m0.044s
```

```
[et-virt07 images]# ls -lrt testvm1.sav
-rwxr-xr-x 1 root root 1075657716 Jan 12 06:46 testvm1.sav
```

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

Restore the virtual machine on the local host:

```
[et-virt07 images]# virsh restore testvm1.sav
```

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

```
testvm1        blocked
```

Initiate the live migration to `et-virt08`. in the example below `et-virt07` is the hostname you are migrating to and <domain-id> must be replaced with a guest domain available to the host system.

```
[et-virt08 ~]# xm migrate --live <domain-id> et-virt07
```

Verify the virtual machine has been shut down on `et-virt07`

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

Verify the virtual machine has been migrated to `et-virt08`:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

Testing the progress and initiating the live migration

Create the following script inside the virtual machine to log date and hostname during the migration. This script performs I/O tasks on the guest's file system.

```
#!/bin/bash

while true
do
touch /var/tmp/$$.log
echo `hostname` >>  /var/tmp/$$.log
 echo `date`     >>  /var/tmp/$$.log
 cat  /var/tmp/$$.log
 df /var/tmp
 ls -l  /var/tmp/$$.log
 sleep 3
 done
```

Remember, that script is only for testing purposes and unnecessary for a live migration in a production environment.

Verify the virtual machine is running on et-virt08 before we try to migrate it to `et-virt07`:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

Initiate a live migration to `et-virt07`. You can add the **`time`** command to see how long the migration takes:

```
[et-virt08 ~]# xm migrate --live testvm1 et-virt07
```

run the script inside the guest:

```
# ./doit
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 62 Jan 12 02:26 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:30 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 124 Jan 12 02:26 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:30 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:33 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 186 Jan 12 02:26 /var/tmp/2279.log
Fri Jan 12 02:26:45 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:48 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:51 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:54:57 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:55:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:55:03 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 744 Jan 12 06:55 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
```

Verify the virtual machine has been shut down on `et-virt08`:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

Verify the virtual machine has started up on `et-virt07`:

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

Run through another cycle migrating from `et-virt07` to `et-virt08`. Initiate a migration from `et-virt07` to `et-virt08`:

```
[et-virt07 images]# xm migrate --live testvm1 et-virt08
```

Verify the virtual machine has been shut down:

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

Before initiating the migration start the simple script in the guest and note the change in time when migrating the guest:

```
# ./doit
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 62 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 124 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
```

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 186 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 248 Jan 12 02:30 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:03 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 310 Jan 12 02:30 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:03 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:06 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 372 Jan 12 02:30 /var/tmp/2418.log
```

After the migration command completes on `et-virt07` verify on `et-virt08` that the virtual machine has started:

```
[et-virt08 ~]# virsh list
```

```
 Id Name                    State
----------------------------------
Domain-0                    running
testvm1        blocked
```

and run another cycle:

```
[et-virt08 ~]# time virsh migrate --live testvm1 et-virt07
real    0m10.378s
user    0m0.068s
sys     0m0.052s
```

At this point you have successfully performed an offline and a live migration test.

18.2. Configuring guest live migration

This section covers offline migration of Xen guests to other servers running Red Hat Enterprise Linux. Further, migration is performed in an offline method (using the **`xm migrate`** command). Live migration can be done from the same command. However there are some additional modifications that you must do to the **`xend-config`** configuration file. This example identifies the entries that you must modify to ensure a successful migration:

`(xend-relocation-server yes)`
: The default for this parameter is 'no', which keeps the relocation/migration server deactivated (unless on a trusted network) and the domain virtual memory is exchanged in raw form without encryption.

`(xend-relocation-port 8002)`
: This parameter sets the port that **`xend`** uses for migration. Use this value unless your network environment requires a custom value. Remove the comment symbol to enable it.

`(xend-relocation-address )`
: This parameter is the address that listens for relocation socket connections, after you enable the **`xend-relocation-server`**. The Xen hypervisor only listens for migration network traffic on the specified interface.

`(xend-relocation-hosts-allow )`
: This parameter controls the host that communicates with the relocation port. If the value is empty, then all incoming connections are allowed. You must change this to a space-separated sequences of regular expressions (for example, **`xend-relocation-hosts-allow- '^localhost\\.localdomain$'`**). A host with a fully qualified domain name or IP address that matches these expressions are accepted.

After you configure these parameters, you must reboot the host for the Xen hypervisor load the new parameters.

KVM live migration

This chapter covers migrating guests running on a KVM hypervisor to another KVM host.

Migration is name for the process of moving a virtualized guest from one host to another. Migration is a key feature of virtualization as software is completely separated from hardware. Migration is useful for:

- Load balancing - guests can be moved to hosts with lower usage when a host becomes overloaded.
- Hardware failover - when hardware devices on the host start to fail, guests can be safely relocated so the host can be powered down and repaired.
- Energy saving - guests can be redistributed to other hosts and host systems powered off to save energy and cut costs in low usage periods.
- Geographic migration - guests can be moved to another location for lower latency or in serious circumstances.

Migrations can be performed live or offline. To migrate guests the storage must be shared. Migration works by sending the guests memory to the destination host. The shared storage stores the guest's default file system. The file system image is not sent over the network from the source host to the destination host.

An offline migration suspends the guest then moves an image of the guests memory to the destination host. The guest is resumed on the destination host and the memory the guest used on the source host is freed.

The time an offline migration takes depends network bandwidth and latency. A guest with 2GB of memory should take an average of ten or so seconds on a 1 Gbit Ethernet link.

A live migration keeps the guest running on the source host and begins moving the memory without stopping the guest. All modified memory pages are monitored for changes and sent to the destination while the image is sent. The memory is updated with the changed pages. The process continues until the amount of pause time allowed for the guest equals the predicted time for the final few pages to be transfer. KVM estimates the time remaining and attempts to transfer the maximum amount of page files from the source to the destination until KVM predicts the amount of remaining pages can be transferred during a very brief time while the virtualized guest is paused. The registers are loaded on the new host and the guest is then resumed on the destination host. If the guest is cannot be merged (which happens when guests are under extreme loads) the guest is paused and then an offline migration is started instead.

The time an offline migration takes depends network bandwidth and latency. If the network is in heavy use or a low bandwidth the migration will take much longer.

19.1. Live migration requirements

Migrating guests requires the following:

Migration requirements

- A virtualized guest installed on shared networked storage using one of the following protocols:
  - Fibre Channel

   - iSCSI
   - NFS
   - GFS2
- Two or more Red Hat Enterprise Linux systems of the same version with the same updates.
- Both system must have the appropriate ports open.
- Both systems must have identical network configurations. All bridging and network configurations must be exactly the same on both hosts.
- Shared storage must mount at the same location on source and destination systems. The mounted directory name must be identical.

Configuring network storage

Configure shared storage and install a guest on the shared storage. For shared storage instructions, refer to *Chapter 9, Shared storage and virtualization*.

Alternatively, use the NFS example in *Section 19.2, "Share storage example: NFS for a simple migration"*.

19.2. Share storage example: NFS for a simple migration

This example uses NFS to share guest images with other KVM hosts. This example is not practical for large installations, this example is only for demonstrating migration techniques and small deployments. Do not use this example for migrating or running more than a few virtualized guests.

For advanced and more robust shared storage instructions, refer to *Chapter 9, Shared storage and virtualization*

1. **Export your libvirt image directory**
   Add the default image directory to the **`/etc/exports`** file:

   ```
   /var/lib/libvirt/images *.bne.redhat.com(rw,no_root_squash,async)
   ```

   Change the hosts parameter as required for your environment.

2. **Start NFS**
   a. Install the NFS packages if they are not yet installed:

   ```
   # yum install nfs
   ```

   b. Open the ports for NFS in **`iptables`** and add NFS to the **`/etc/hosts.allow`** file.

   c. Start the NFS service:

   ```
   # service nfs start
   ```

3. **Mount the shared storage on the destination**
   On the destination system, mount the **/var/lib/libvirt/images** directory:

   ```
   # mount sourceURL:/var/lib/libvirt/images /var/lib/libvirt/images
   ```

   > **Locations must be the same on source and destination**
   > Whichever directory is chosen for the guests must exactly the same on host and guest. This applies to all types of shared storage. The directory must be the same or the migration will fail.

19.3. Live KVM migration with virsh

A guest can be migrated to another host with the **virsh** command. The **migrate** command accepts parameters in the following format:

```
# virsh migrate --live GuestName DestinationURL
```

The *GuestName* parameter represents the name of the guest which you want to migrate.

The *DestinationURL* parameter is the URL or hostname of the destination system. The destination system must run the same version of Red Hat Enterprise Linux, be using the same hypervisor and have **libvirt** running.

Once the command is entered you will be prompted for the root password of the destination system.

Example: live migration with virsh

This example migrates from test1.bne.redhat.com to test2.bne.redhat.com. Change the host names for your environment. This example migrates a virtual machine named **RHEL4test**.

This example assumes you have fully configured shared storage and meet all the prerequisites (listed here: *Migration requirements*).

1. **Verify the guest is running**
   From the source system, test1.bne.redhat.com, verify RHEL4test is running:

   ```
   [root@test1 ~]# virsh list
   Id Name                 State
   ----------------------------------
    10 RHEL4                running
   ```

2. **Migrate the guest**
   Execute the following command to live migrate the guest to the destination, `test2.bne.redhat.com`. Append **`/system`** to the end of the destination URL to tell libvirt that you need full access.

   ```
   # virsh migrate --live RHEL4test qemu+ssh://test2.bne.redhat.com/system
   ```

   Once the command is entered you will be prompted for the root password of the destination system.

3. **Wait**
   The migration may take some time depending on load and the size of the guest. **`virsh`** only reports errors. The guest continues to run on the source host until fully migrated.

4. **Verify the guest has arrived at the destination host**
   From the destination system, `test2.bne.redhat.com`, verify `RHEL4test` is running:

   ```
   [root@test2 ~]# virsh list
   Id Name                 State
   ----------------------------------
    10 RHEL4                running
   ```

The live migration is now complete.

> **Other networking methods**
> libvirt supports a variety of networking methods including TLS/SSL, unix sockets, SSH, and unecrypted TCP. Refer to *Chapter 20, Remote management of virtualized guests* for more information on using other methods.

19.4. Migrating with virt-manager

This section covers migrating KVM based guests with **`virt-manager`**.

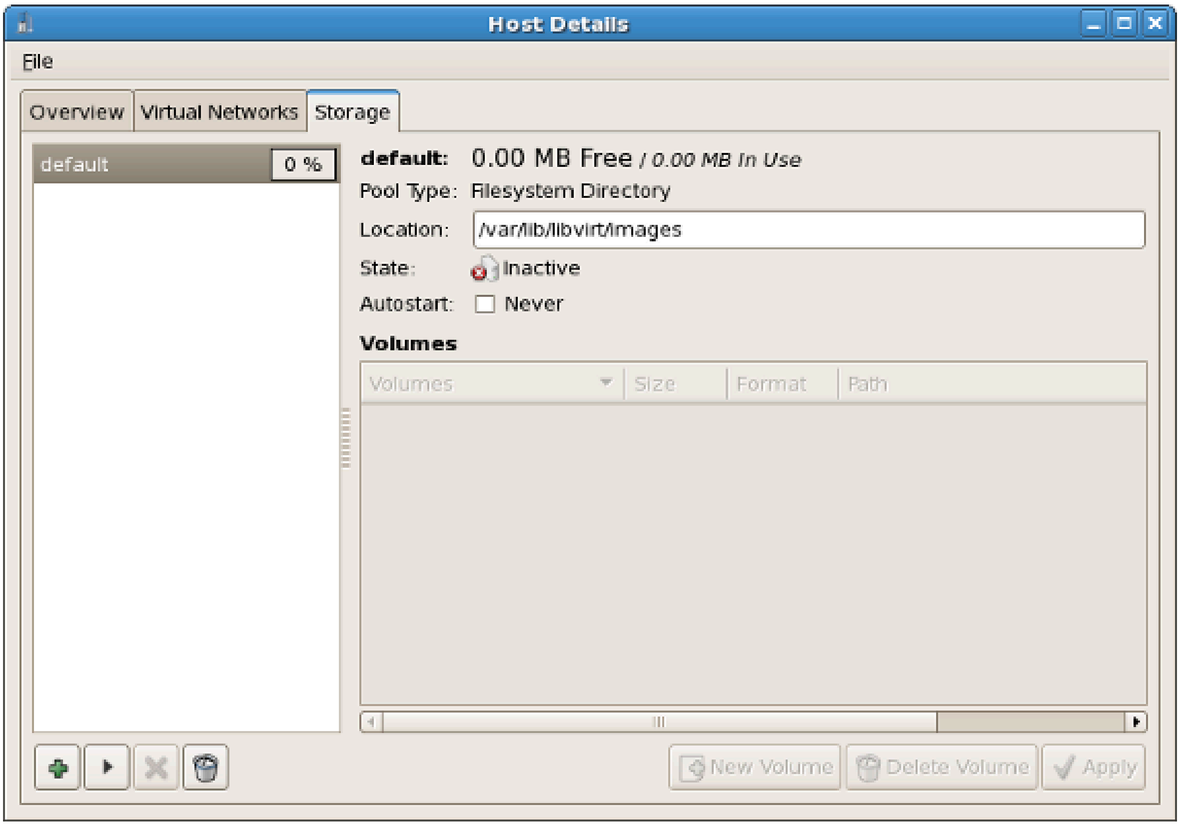
1. Connect to the source and target hosts. On the **File** menu, click **Add Connection**, the **Add Connection** window appears.

   Enter the following details:

2. Add a storage pool with the same NFS to the source and target hosts.

   On the **Edit** menu, click **Host Details**, the Host Details window appears.

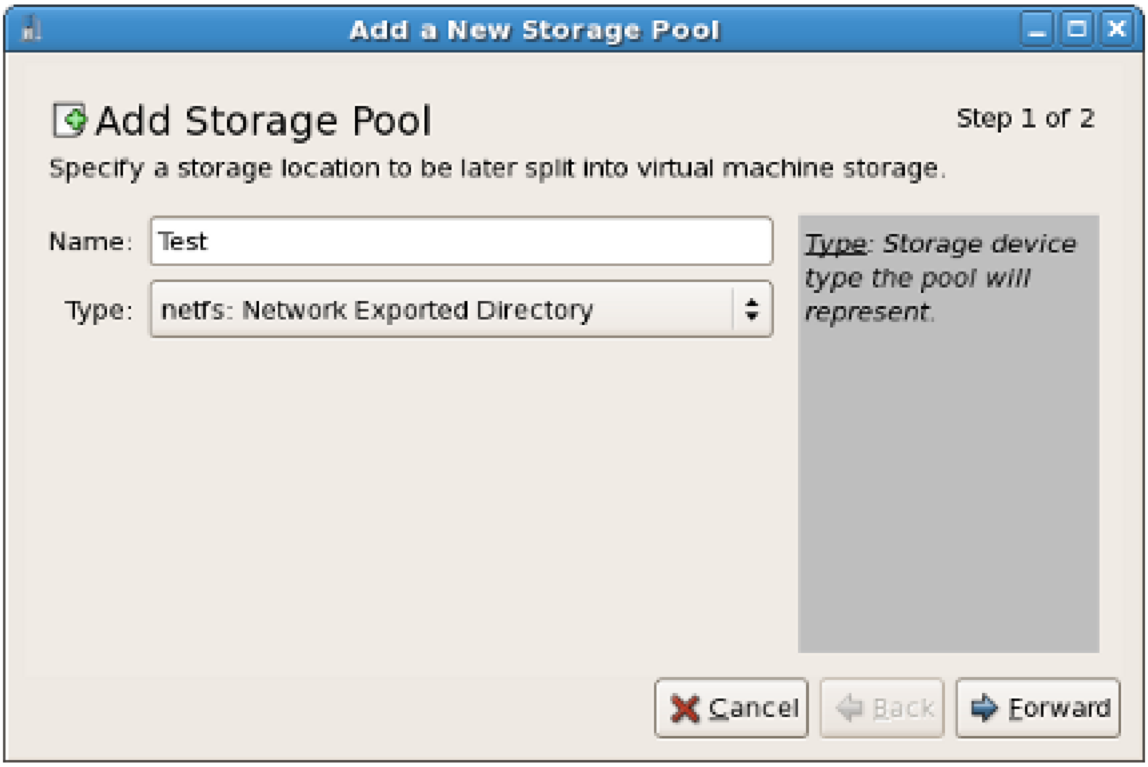
   Click the **Storage** tab.



3. Add a new storage pool. In the lower left corner of the window, click the **+** button. The Add a New Storage Pool window appears.

   Enter the following details:

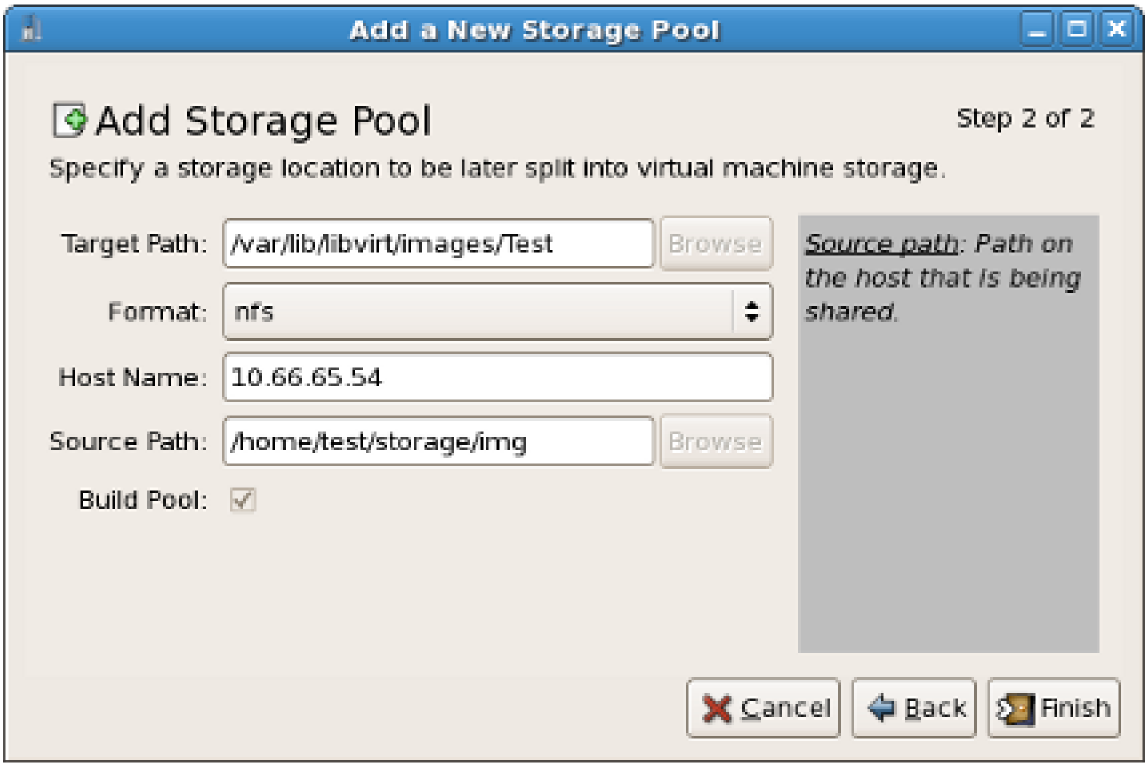
   - **Name**: Enter the name of the storage pool.
   - **Type**: Select **netfs: Network Exported Directory**.



   Click **Forward**.

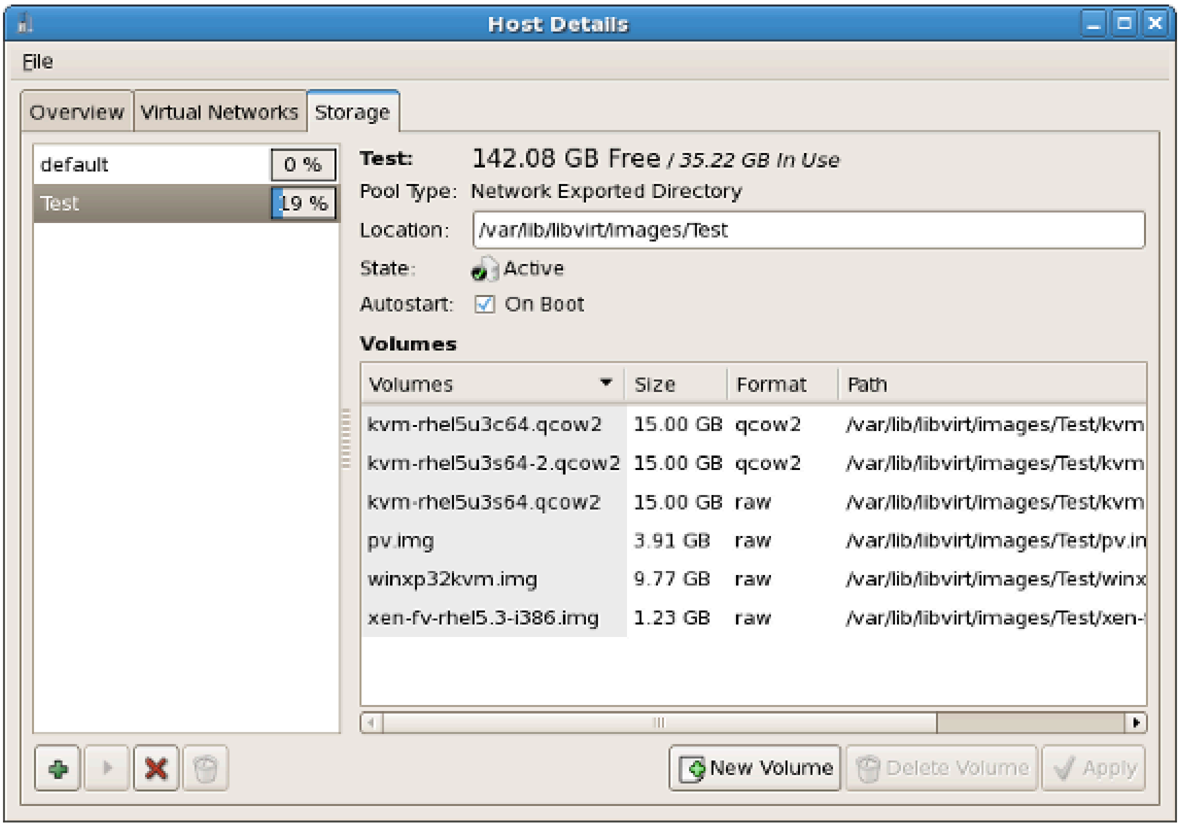
4. Enter the following details:

   - **Format**: Select the storage type. This must be NFS or iSCSI for live migrations.
   - **Host Name**: Enter the IP address or fully-qualified domain name of the storage server.

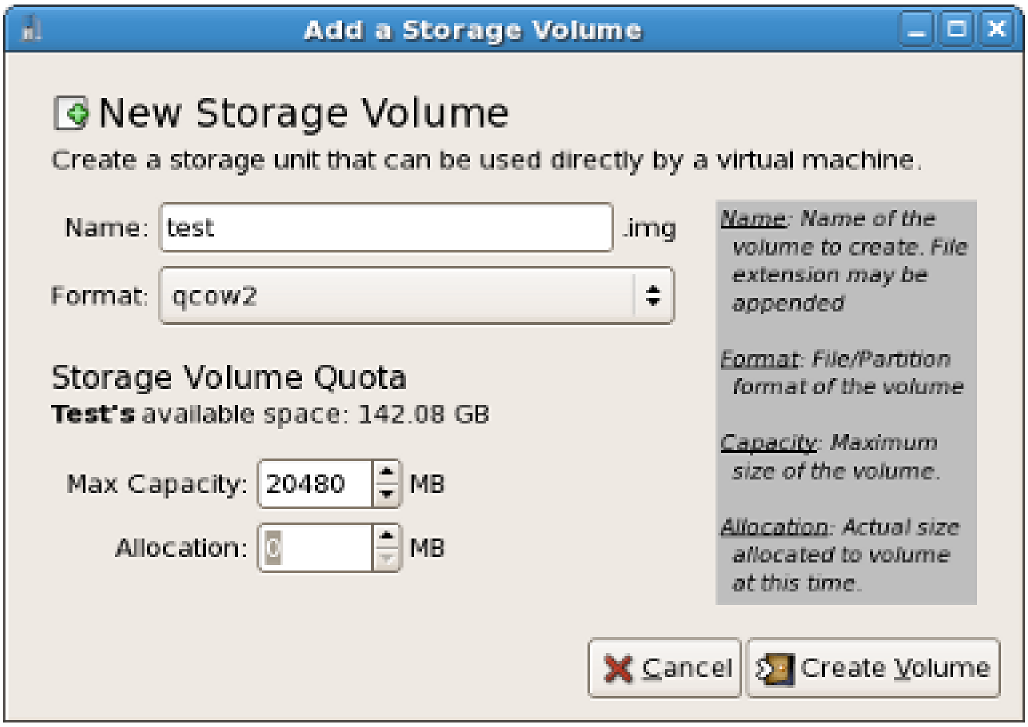


Click **Finish**.

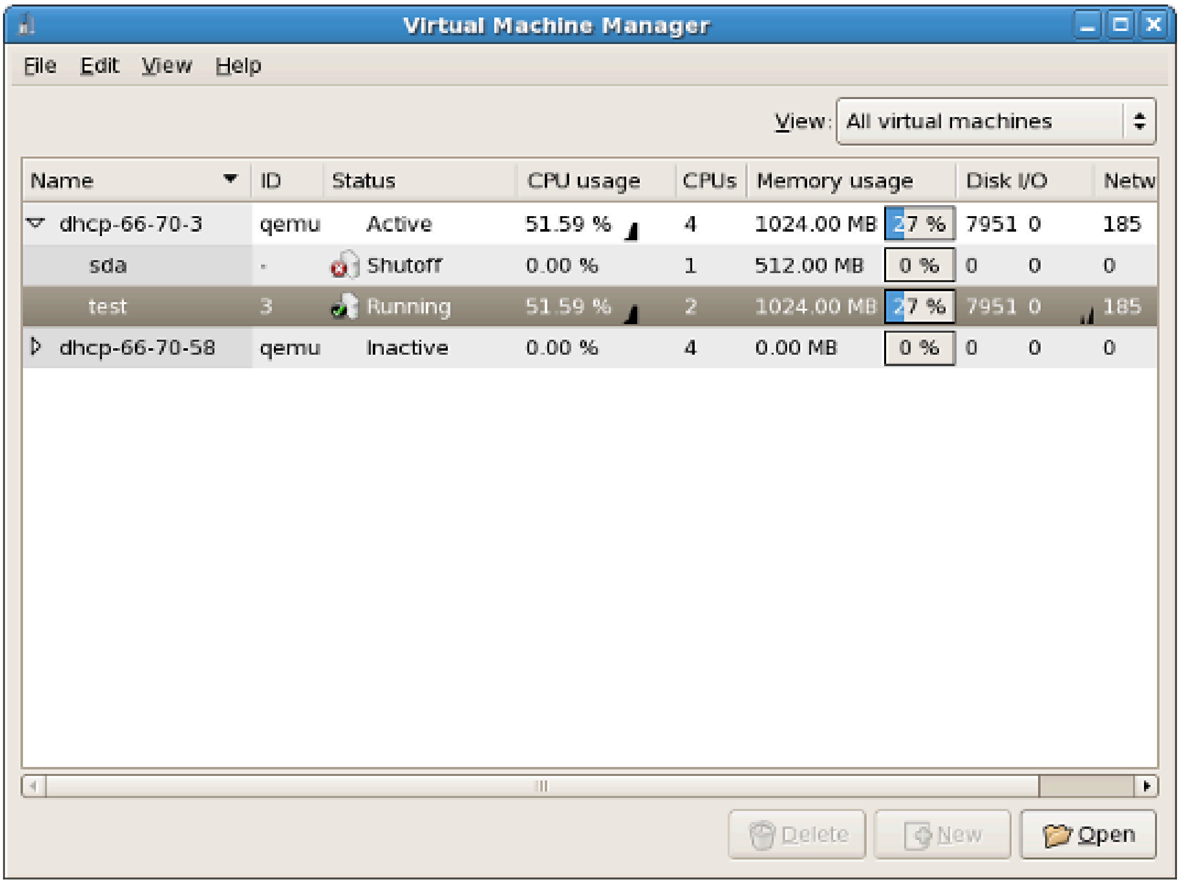
5. Create a new volume in the shared storage pool, click **New Volume**.



6. Enter the details, then click **Create Volume**.

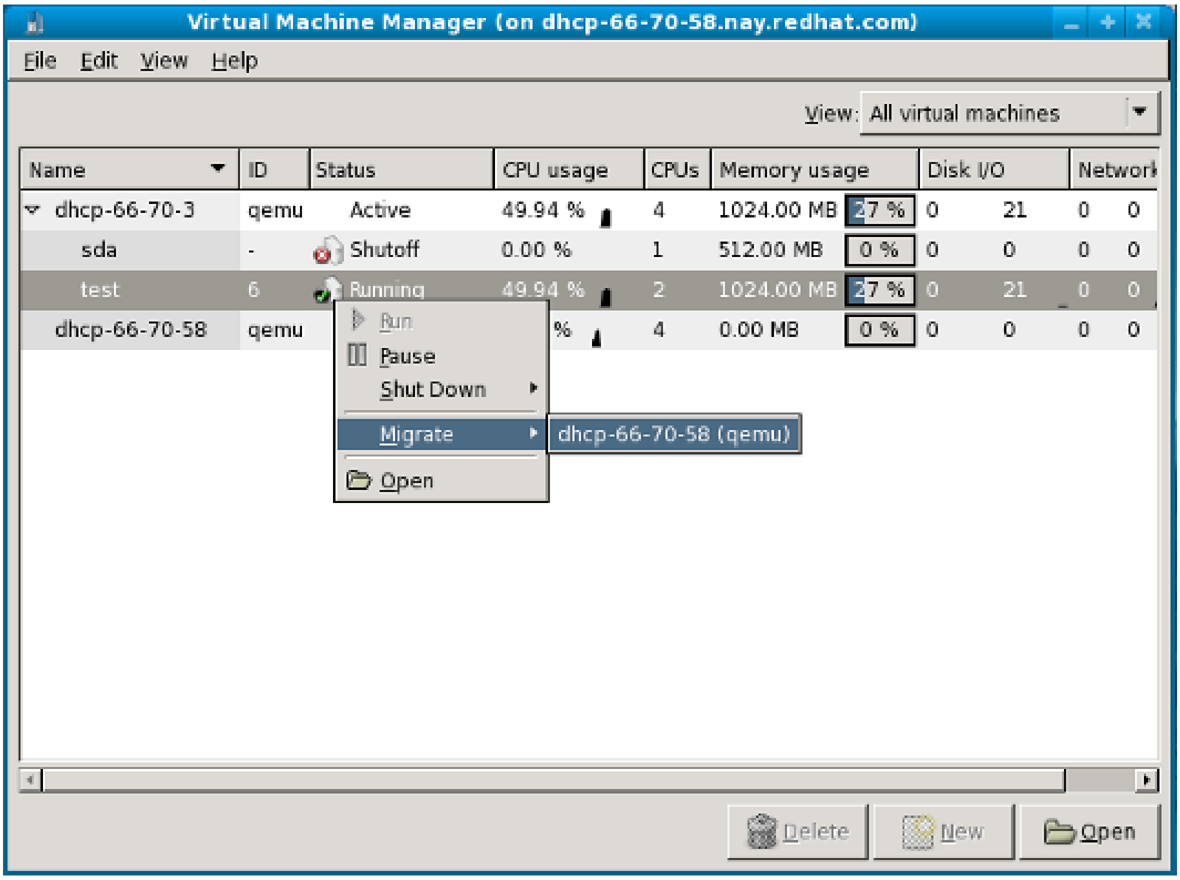


7. Create a virtual machine with the new volume, then run the virtual machine.

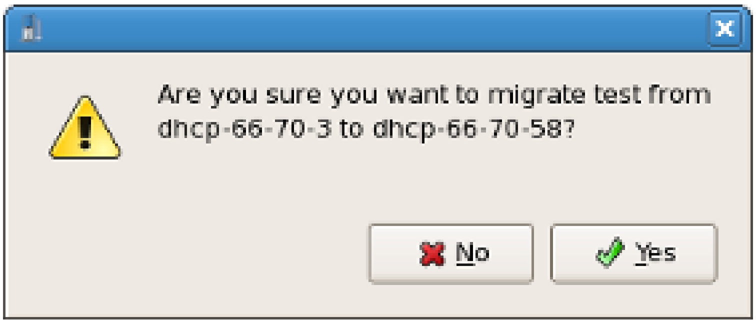


The Virtual Machine window appears.

8. In the Virtual Machine Manager window, right-click on the virtual machine, select **Migrate**, then click the migration location.



9. Click **Yes** to confirm migration.



The Virtual Machine Manger displays the virtual machine in its new location.

Remote management of virtualized guests

This section explains how to remotely manage your virtualized guests using **ssh** or TLS and SSL.

20.1. Remote management with SSH

The *ssh* package provides an encrypted network protocol which can securely send management functions to remote virtualization servers. The method described uses the **libvirt** management connection securely tunneled over an **SSH** connection to manage the remote machines. All the authentication is done using **SSH** public key cryptography and passwords or passphrases gathered by your local **SSH** agent. In addition the **VNC** console for each guest virtual machine is tunneled over **SSH**.

SSH is usually configured by default so you probably already have SSH keys setup and no extra firewall rules needed to access the management service or **VNC** console.

Be aware of the issues with using **SSH** for remotely managing your virtual machines, including:

- you require root log in access to the remote machine for managing virtual machines,
- the initial connection setup process may be slow,
- there is no standard or trivial way to revoke a user's key on all hosts or guests, and
- ssh does not scale well with larger numbers of remote machines.

Configuring SSH access for virt-manager

The following instructions assume you are starting from scratch and do not already have **SSH** keys set up.

1. You need a public key pair on the machine **virt-manager** is used. If **ssh** is already configured you can skip this command.

   ```
   $ ssh-keygen -t rsa
   ```

2. To permit remote log in, **virt-manager** needs a copy of the public key on each remote machine running **libvirt**. Copy the file **$HOME/.ssh/id_rsa.pub** from the machine you want to use for remote management using the **scp** command:

   ```
   $ scp $HOME/.ssh/id_rsa.pub root@somehost:/root/key-dan.pub
   ```

3. After the file has copied, use **ssh** to connect to the remote machines as root and add the file that you copied to the list of authorized keys. If the root user on the remote host does not already have an list of authorized keys, make sure the file permissions are correctly set

   ```
   $ ssh root@somehost
   # mkdir /root/.ssh
   # chmod go-rwx /root/.ssh
   # cat /root/key-dan.pub >> /root/.ssh/authorized_keys
   ```

```
# chmod go-rw /root/.ssh/authorized_keys
```

The `libvirt` daemon (`libvirtd`)

The `libvirt` daemon provide an interface for managing virtual machines. You must have the `libvirtd` daemon installed and running on every remote host that you need to manage. Using the Red Hat **`kernel-xen`** package requires a special kernel and CPU hardware support, see *Chapter 1, System requirements* for details.

```
$ ssh root@somehost
# chkconfig libvirtd on
# service libvirtd start
```

After `libvirtd` and **SSH** are configured you should be able to remotely access and manage your virtual machines. You should also be able to access your guests with **`VNC`** at this point.

20.2. Remote management over TLS and SSL

You can manage virtual machines using TLS and SSL. TLS and SSL provides greater scalability but is more complicated than ssh (refer to *Section 20.1, "Remote management with SSH"*). TLS and SSL is the same technology used by web browsers for secure connections. The **`libvirt`** management connection opens a TCP port for incoming connections, which is securely encrypted and authenticated based on x509 certificates. In addition the VNC console for each guest virtual machine will be setup to use TLS with x509 certificate authentication.

This method does not require shell accounts on the remote machines being managed. However, extra firewall rules are needed to access the management service or VNC console. Certificate revocation lists can revoke users' access.

Steps to setup TLS/SSL access for virt-manager

The following short guide assuming you are starting from scratch and you do not have any TLS/SSL certificate knowledge. If you are lucky enough to have a certificate management server you can probably skip the first steps.

`libvirt` server setup

For more information on creating certificates, refer to the **libvirt** website, *http://libvirt.org/remote.html*.

Xen VNC Server

The Xen VNC server can have TLS enabled by editing the configuration file, **`/etc/xen/xend-config.sxp`**. Remove the commenting on the **`(vnc-tls 1)`** configuration parameter in the configuration file.

The **`/etc/xen/vnc`** directory needs the following 3 files:

- **`ca-cert.pem`** - The CA certificate
- **`server-cert.pem`** - The Server certificate signed by the CA
- **`server-key.pem`** - The server private key

This provides encryption of the data channel. It might be appropriate to require that clients present their own x509 certificate as a form of authentication. To enable this remove the commenting on the **(vnc-x509-verify 1)** parameter.

virt-manager and **virsh** client setup

The setup for clients is slightly inconsistent at this time. To enable the **libvirt** management API over TLS, the CA and client certificates must be placed in **/etc/pki**. For details on this consult *http://libvirt.org/remote.html*

In the **virt-manager** user interface, use the '**SSL/TLS**' transport mechanism option when connecting to a host.

For **virsh**, the URI has the following format:

- **qemu://hostname.guestname/system** for KVM.
- **xen://hostname.guestname/** for Xen.

To enable SSL and TLS for VNC, it is necessary to put the certificate authority and client certificates into **$HOME/.pki**, that is the following three files:

- CA or **ca-cert.pem** - The CA certificate.
- **libvirt-vnc** or **clientcert.pem** - The client certificate signed by the CA.
- **libvirt-vnc** or **clientkey.pem** - The client private key.

20.3. Transport modes

For remote management, **libvirt** supports the following transport modes:

Transport Layer Security (TLS)

Transport Layer Security TLS 1.0 (SSL 3.1) authenticated and encrypted TCP/IP socket, usually listening on a public port number. To use this you will need to generate client and server certificates. The standard port is 16514.

UNIX sockets

Unix domain sockets are only accessible on the local machine. Sockets are not encrypted, and use UNIX permissions or SELinux for authentication. The standard socket names are **/var/run/libvirt/libvirt-sock** and **/var/run/libvirt/libvirt-sock-ro** (for read-only connections).

SSH

Transported over an Secure Shell protocol (SSH) connection. Requires Netcat (the *nc* package) installed. The libvirt daemon (**libvirtd**) must be running on the remote machine. Port 22 must be open for SSH access. You should use some sort of ssh key management (for example, the **ssh-agent** utility) or you will be prompted for a password.

ext

The ext parameter is used for any external program which can make a connection to the remote machine by means outside the scope of libvirt. This usually covers third-party, unsupported security applications.

tcp

Unencrypted TCP/IP socket. Not recommended for production use, this is normally disabled, but an administrator can enable it for testing or use over a trusted network. The default port is 16509.

The default transport, if no other is specified, is tls.

Remote URIs

A Uniform Resource Identifier (URI) is used by virsh and libvirt to connect to a remote host. URIs can also be used with the --connect parameter for the virsh command to execute single commands or migrations on remote hosts.

libvirt URIs take the general form (content in square brackets, "[]", represents optional functions):

```
driver[+transport]://[username@][hostname][:port]/[path][?extraparameters]
```

Either the transport method or the hostname must be provided in order to distinguish this from a local URI.

Examples of remote management parameters

- Connect to a remote Xen hypervisor on the host named `towada`, using SSH transport and the SSH username `ccurran`.

  ```
  xen+ssh://ccurran@towada/
  ```

- Connect to a remote Xen hypervisor on the host named `towada` using TLS.

  ```
  xen://towada/
  ```

- Connect to a remote Xen hypervisor on host `towada` using TLS. The *`no_verify=1`* tells libvirt not to verify the server's certificate.

  ```
  xen://towada/?no_verify=1
  ```

- Connect to a remote KVM hypervisor on host `towada` using SSH.

  ```
  qemu+ssh://towada/system
  ```

Testing examples

- Connect to the local KVM hypervisor with a non-standard UNIX socket. The full path to the Unix socket is supplied explicitly in this case.

```
qemu+unix:///system?socket=/opt/libvirt/run/libvirt/libvirt-sock
```

- Connect to the libvirt daemon with an unencrypted TCP/IP connection to the server with the IP address 10.1.1.10 on port 5000. This uses the test driver with default settings.

```
test+tcp://10.1.1.10:5000/default
```

Extra URI parameters

Extra parameters can be appended to remote URIs. The table below *Table 20.1, "Extra URI parameters"* covers the recognized parameters. All other parameters are ignored. Note that parameter values must be URI-escaped (that is, a question mark (?) is appended before the parameter and special characters are converted into the URI format).

Name	Transport mode	Description	Example usage
name	all modes	The name passed to the remote virConnectOpen function. The name is normally formed by removing transport, hostname, port number, username and extra parameters from the remote URI, but in certain very complex cases it may be better to supply the name explicitly.	name=qemu:///system
command	ssh and ext	The external command. For ext transport this is required. For ssh the default is ssh. The PATH is searched for the command.	command=/opt/openssh/bin/ssh
socket	unix and ssh	The path to the UNIX domain socket, which overrides the default. For ssh transport, this is passed to the remote netcat command (see netcat).	socket=/opt/libvirt/run/libvirt/libvirt-sock
netcat	ssh	The name of the netcat command on the remote machine. The default is nc. For ssh transport, libvirt constructs an	netcat=/opt/netcat/bin/nc

Name	Transport mode	Description	Example usage
		ssh command which looks like: command -p port [-l username] hostname netcat -U socket where port, username, hostname can be specified as part of the remote URI, and command, netcat and socket come from extra parameters (or sensible defaults).	
no_verify	tls	If set to a non-zero value, this disables client checks of the server's certificate. Note that to disable server checks of the client's certificate or IP address you must change the libvirtd configuration.	no_verify=1
no_tty	ssh	If set to a non-zero value, this stops ssh from asking for a password if it cannot log in to the remote machine automatically (for using ssh-agent or similar). Use this when you do not have access to a terminal - for example in graphical programs which use libvirt.	no_tty=1

Table 20.1. Extra URI parameters

Part V. Virtualization Reference Guide

Virtualization commands, system tools, applications and additional systems reference

These chapters provide detailed descriptions of virtualization commands, system tools, and applications included in Red Hat Enterprise Linux. These chapters are designed for users requiring information on advanced functionality and other features.

Virtualization tools

The following is a list of tools for virtualization administration, debugging and networking tools that are useful for systems running Xen.

System Administration Tools

- **vmstat**
- **iostat**
- **lsof**

```
# lsof -i :5900
xen-vncfb 10635  root   5u  IPv4 218738  TCP
 grumble.boston.redhat.com:5900 (LISTEN)
```

- **qemu-img**

Advanced Debugging Tools

- **systemTap**
- **crash**
- **xen-gdbserver**
- **sysrq**
- **sysrq t**
- **sysrq w**
- **sysrq c**

Networking

brtcl

- ```
  # brctl show
  bridge name   bridge id            STP enabled    interfaces
  xenbr0        8000.feffffffffff    no             vif13.0
         pdummy0
                                                    vif0.0
  ```

- ```
  # brctl showmacs xenbr0
  port no  mac addr                is local?       aging timer
    1      fe:ff:ff:ff:ff:ff       yes             0.00
  ```

- ```
  # brctl showstp xenbr0
  xenbr0
  bridge id               8000.fefffffffff
  designated root         8000.fefffffffff
  ```
  ```

```
root port                 0                     path cost
 0
max age                   20.00                 bridge max age
 20.00
hello time                2.00                  bridge hello time
 2.00
forward delay             0.00                  bridge forward delay
 0.00
aging time              300.01
hello timer               1.43                  tcn timer
 0.00
topology change timer     0.00                  gc timer
 0.02
flags

vif13.0 (3)
port id                   8003                  state
 forwarding
designated root           8000.fefffffffff     path cost
 100
designated bridge         8000.fefffffffff     message age timer
 0.00
designated port           8003                  forward delay timer
 0.00
designated cost           0                     hold timer
 0.43
flags

pdummy0 (2)
port id                   8002                  state
 forwarding
designated root           8000.fefffffffff     path cost
 100
designated bridge         8000.fefffffffff     message age timer
 0.00
designated port           8002                  forward delay timer
 0.00
designated cost           0                     hold timer
 0.43
flags

vif0.0 (1)
port id                   8001                  state
 forwarding
designated root           8000.fefffffffff     path cost
 100
designated bridge         8000.fefffffffff     message age timer
 0.00
designated port           8001                  forward delay timer
 0.00
```

```
designated cost         0                    hold timer
 0.43
flags
```

- **ifconfig**
- **tcpdump**

KVM tools

- **ps**
- **pstree**
- **top**
- **kvmtrace**
- **kvm_stat**

Xen tools

- **xentop**
- **xm dmesg**
- **xm log**

Managing guests with virsh

virsh is a command line interface tool for managing guests and the hypervisor.

The **virsh** tool is built on the **libvirt** management API and operates as an alternative to the **xm** command and the graphical guest Manager (**virt-manager**). **virsh** can be used in read-only mode by unprivileged users. You can use **virsh** to execute scripts for the guest machines.

virsh command quick reference

The following tables provide a quick reference for all virsh command line options.

Command	Description
help	Prints basic help information.
list	Lists all guests.
dumpxml	Outputs the XML configuration file for the guest.
create	Creates a guest from an XML configuration file and starts the new guest.
start	Starts an inactive guest.
destroy	Forces a guest to stop.
define	Outputs an XML configuration file for a guest.
domid	Displays the guest's ID.
domuuid	Displays the guest's UUID.
dominfo	Displays guest information.
domname	Displays the guest's name.
domstate	Displays the state of a guest.
quit	Quits the interactive terminal.
reboot	Reboots a guest.
restore	Restores a previously saved guest stored in a file.
resume	Resumes a paused guest.
save	Save the present state of a guest to a file.
shutdown	Gracefully shuts down a guest.
suspend	Pauses a guest.
undefine	Deletes all files associated with a guest.
migrate	Migrates a guest to another host.

Table 22.1. Guest management commands

The following **virsh** command options to manage guest and hypervisor resources:

Command	Description
setmem	Sets the allocated memory for a guest.
setmaxmem	Sets maximum memory limit for the hypervisor.

Command	Description
setvcpus	Changes number of virtual CPUs assigned to a guest.
vcpuinfo	Displays virtual CPU information about a guest.
vcpupin	Controls the virtual CPU affinity of a guest.
domblkstat	Displays block device statistics for a running guest.
domifstat	Displays network interface statistics for a running guest.
attach-device	Attach a device to a guest, using a device definition in an XML file.
attach-disk	Attaches a new disk device to a guest.
attach-interface	Attaches a new network interface to a guest.
detach-device	Detach a device from a guest, takes the same kind of XML descriptions as command **attach-device**.
detach-disk	Detach a disk device from a guest.
detach-interface	Detach a network interface from a guest.

Table 22.2. Resource management options

These are miscellaneous **virsh** options:

Command	Description
version	Displays the version of **virsh**
nodeinfo	Outputs information about the hypervisor

Table 22.3. Miscellaneous options

Connecting to the hypervisor

Connect to a hypervisor session with **virsh**:

```
# virsh connect {hostname OR URL}
```

Where **<name>** is the machine name of the hypervisor. To initiate a read-only connection, append the above command with **-readonly**.

Creating a virtual machine XML dump (configuration file)

Output a guest's XML configuration file with **virsh**:

```
# virsh dumpxml {domain-id, domain-name or domain-uuid}
```

This command outputs the guest's XML configuration file to standard out (**stdout**). You can save the data by piping the output to a file. An example of piping the output to a file called *guest.xml*:

```
# virsh dumpxml GuestID > guest.xml
```

This file **guest.xml** can recreate the guest (refer to *Editing a guest's configuration file*. You can edit this XML configuration file to configure additional devices or to deploy additional guests. Refer to *Section 29.1, “Using XML configuration files with virsh”* for more information on modifying files created with **virsh dumpxml**.

An example of **virsh dumpxml** output:

```
# virsh dumpxml r5b2-mySQL01
<domain type='xen' id='13'>
    <name>r5b2-mySQL01</name>
    <uuid>4a4c59a7ee3fc78196e4288f2862f011</uuid>
    <bootloader>/usr/bin/pygrub</bootloader>
    <os>
        <type>linux</type>
        <kernel>/var/lib/libvirt/vmlinuz.2dgnU_</kernel>
 <initrd>/var/lib/libvirt/initrd.UQafMw</initrd>
        <cmdline>ro root=/dev/VolGroup00/LogVol00 rhgb quiet</cmdline>
    </os>
    <memory>512000</memory>
    <vcpu>1</vcpu>
    <on_poweroff>destroy</on_poweroff>
    <on_reboot>restart</on_reboot>
    <on_crash>restart</on_crash>
    <devices>
        <interface type='bridge'>
            <source bridge='xenbr0'/>
            <mac address='00:16:3e:49:1d:11'/>
            <script path='vif-bridge'/>
        </interface>
        <graphics type='vnc' port='5900'/>
        <console tty='/dev/pts/4'/>
    </devices>
</domain>
```

Creating a guest from a configuration file

Guests can be created from XML configuration files. You can copy existing XML from previously created guests or use the **dumpxml** option (refer to *Creating a virtual machine XML dump (configuration file)*). To create a guest with **virsh** from an XML file:

```
# virsh create configuration_file.xml
```

Editing a guest's configuration file

Instead of using the **dumpxml** option (refer to *Creating a virtual machine XML dump (configuration file)*) guests can be edited either while they run or while they are offline. The **virsh edit** command provides this functionality. For example, to edit the guest named *softwaretesting*:

```
# virsh edit softwaretesting
```

This opens a text editor. The default text editor is the **$EDITOR** shell parameter (set to **vi** by default).

Suspending a guest

Suspend a guest with **virsh**:

```
# virsh suspend {domain-id, domain-name or domain-uuid}
```

When a guest is in a suspended state, it consumes system RAM but not processor resources. Disk and network I/O does not occur while the guest is suspended. This operation is immediate and the guest can be restarted with the **resume** (*Resuming a guest*) option.

Resuming a guest

Restore a suspended guest with **virsh** using the **resume** option:

```
# virsh resume {domain-id, domain-name or domain-uuid}
```

This operation is immediate and the guest parameters are preserved for **suspend** and **resume** operations.

Save a guest

Save the current state of a guest to a file using the **virsh** command:

```
# virsh save {domain-name, domain-id or domain-uuid} filename
```

This stops the guest you specify and saves the data to a file, which may take some time given the amount of memory in use by your guest. You can restore the state of the guest with the **restore** (*Restore a guest*) option. Save is similar to pause, instead of just pausing a guest the present state of the guest is saved.

Restore a guest

Restore a guest previously saved with the **virsh save** command (*Save a guest*) using **virsh**:

```
# virsh restore filename
```

This restarts the saved guest, which may take some time. The guest's name and UUID are preserved but are allocated for a new id.

Shut down a guest

Shut down a guest using the **virsh** command:

```
# virsh shutdown {domain-id, domain-name or domain-uuid}
```

You can control the behavior of the rebooting guest by modifying the **on_shutdown** parameter in the guest's configuration file file.

Rebooting a guest

Reboot a guest using **virsh** command:

```
#virsh reboot {domain-id, domain-name or domain-uuid}
```

You can control the behavior of the rebooting guest by modifying the **on_reboot** parameter in the guest's configuration file file.

Forcing a guest to stop

Force a guest to stop with the **virsh** command:

```
# virsh destroy {domain-id, domain-name or domain-uuid}
```

This command does an immediate ungraceful shutdown and stops the specified guest. Using **virsh destroy** can corrupt guest file systems . Use the **destroy** option only when the guest is unresponsive. For para-virtualized guests, use the **shutdown** option(*Shut down a guest*) instead.

Getting the domain ID of a guest

To get the domain ID of a guest:

```
# virsh domid {domain-name or domain-uuid}
```

Getting the domain name of a guest

To get the domain name of a guest:

```
# virsh domname {domain-id or domain-uuid}
```

Getting the UUID of a guest

To get the Universally Unique Identifier (UUID) for a guest:

```
# virsh domuuid {domain-id or domain-name}
```

An example of **virsh domuuid** output:

```
# virsh domuuid r5b2-mySQL01
4a4c59a7-ee3f-c781-96e4-288f2862f011
```

Displaying guest Information

Using **virsh** with the guest's domain ID, domain name or UUID you can display information on the specified guest:

```
# virsh dominfo {domain-id, domain-name or domain-uuid}
```

This is an example of **virsh dominfo** output:

```
# virsh dominfo r5b2-mySQL01
id:             13
```

```
name:            r5b2-mysql01
uuid:            4a4c59a7-ee3f-c781-96e4-288f2862f011
os type:        linux
state:           blocked
cpu(s):          1
cpu time:       11.0s
max memory:      512000 kb
used memory:     512000 kb
```

Displaying host information

To display information about the host:

```
# virsh nodeinfo
```

An example of **virsh nodeinfo** output:

```
# virsh nodeinfo
CPU model                    x86_64
CPU (s)                      8
CPU frequency                2895 Mhz
CPU socket(s)                2
Core(s) per socket           2
Threads per core:            2
Numa cell(s)                 1
Memory size:                 1046528 kb
```

This displays the node information and the machines that support the virtualization process.

Displaying the guests

To display the guest list and their current states with **virsh**:

```
# virsh list
```

Other options available include:

the **--inactive** option to list inactive guests (that is, guests that have been defined but are not currently active), and

the **--all** option lists all guests. For example:

```
# virsh list --all
 Id Name                 State
----------------------------------
  0 Domain-0             running
  1 Domain202            paused
  2 Domain010            inactive
  3 Domain9600           crashed
```

The output from **virsh list** is categorized as one of the six states (listed below).

- The `running` state refers to guests which are currently active on a CPU.
- Guests listed as `blocked` are blocked, and are not running or runnable. This is caused by a guest waiting on I/O (a traditional wait state) or guests in a sleep mode.
- The `paused` state lists domains that are paused. This occurs if an administrator uses the **pause** button in **`virt-manager`**, **`xm pause`** or **`virsh suspend`**. When a guest is paused it consumes memory and other resources but it is ineligible for scheduling and CPU resources from the hypervisor.
- The `shutdown` state is for guests in the process of shutting down. The guest is sent a shutdown signal and should be in the process of stopping its operations gracefully. This may not work with all guest operating systems; some operating systems do not respond to these signals.
- Domains in the `dying` state are in is in process of dying, which is a state where the domain has not completely shut-down or crashed.
- `crashed` guests have failed while running and are no longer running. This state can only occur if the guest has been configured not to restart on crash.

Displaying virtual CPU information

To display virtual CPU information from a guest with **`virsh`**:

```
# virsh vcpuinfo {domain-id, domain-name or domain-uuid}
```

An example of **`virsh vcpuinfo`** output:

```
# virsh vcpuinfo r5b2-mySQL01
VCPU:           0
CPU:            0
State:          blocked
CPU time:       0.0s
CPU Affinity:   yy
```

Configuring virtual CPU affinity

To configure the affinity of virtual CPUs with physical CPUs:

```
# virsh vcpupin domain-id vcpu cpulist
```

The **`domain-id`** parameter is the guest's ID number or name.

The **`vcpu`** parameter denotes the number of virtualized CPUs allocated to the guest.The **`vcpu`** parameter must be provided.

The **`cpulist`** parameter is a list of physical CPU identifier numbers separated by commas. The **`cpulist`** parameter determines which physical CPUs the VCPUs can run on.

Configuring virtual CPU count

To modify the number of CPUs assigned to a guest with **`virsh`**:

```
# virsh setvcpus {domain-name, domain-id or domain-uuid} count
```

The new *count* value cannot exceed the count above the amount specified when the guest was created.

Configuring memory allocation

To modify a guest's memory allocation with **virsh** :

```
# virsh setmem {domain-id or domain-name} count
```

You must specify the *count* in kilobytes. The new count value cannot exceed the amount you specified when you created the guest. Values lower than 64 MB are unlikely to work with most guest operating systems. A higher maximum memory value will not affect the an active guest unless the new value is lower which will shrink the available memory usage.

Displaying guest block device information

Use **virsh domblkstat** to display block device statistics for a running guest.

```
# virsh domblkstat GuestName block-device
```

Displaying guest network device information

Use **virsh domifstat** to display network interface statistics for a running guest.

```
# virsh domifstat GuestName interface-device
```

Migrating guests with virsh

A guest can be migrated to another host with **virsh**. Migrate domain to another host. Add --live for live migration. The **migrate** command accepts parameters in the following format:

```
# virsh migrate --live GuestName DestinationURL
```

The *--live* parameter is optional. Add the *--live* parameter for live migrations.

The *GuestName* parameter represents the name of the guest which you want to migrate.

The *DestinationURL* parameter is the URL or hostname of the destination system. The destination system must run the same version of Red Hat Enterprise Linux, be using the same hypervisor and have **libvirt** running.

Once the command is entered you will be prompted for the root password of the destination system.

Managing virtual networks

This section covers managing virtual networks with the **virsh** command. To list virtual networks:

```
# virsh net-list
```

This command generates output similar to:

```
# virsh net-list
Name                 State      Autostart
-----------------------------------------
default              active     yes
vnet1              active     yes
vnet2              active     yes
```

To view network information for a specific virtual network:

```
# virsh net-dumpxml NetworkName
```

This displays information about a specified virtual network in XML format:

```
# virsh net-dumpxml vnet1
<network>
  <name>vnet1</name>
  <uuid>98361b46-1581-acb7-1643-85a412626e70</uuid>
  <forward dev='eth0'/>
  <bridge name='vnet0' stp='on' forwardDelay='0' />
  <ip address='192.168.100.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.100.128' end='192.168.100.254' />
    </dhcp>
  </ip>
</network>
```

Other **virsh** commands used in managing virtual networks are:

- **virsh net-autostart *network-name*** — Autostart a network specified as *network-name*.
- **virsh net-create *XMLfile*** — generates and starts a new network using an existing XML file.
- **virsh net-define *XMLfile*** — generates a new network device from an existing XML file without starting it.
- **virsh net-destroy *network-name*** — destroy a network specified as *network-name*.
- **virsh net-name *networkUUID*** — convert a specified *networkUUID* to a network name.
- **virsh net-uuid *network-name*** — convert a specified *network-name* to a network UUID.
- **virsh net-start *nameOfInactiveNetwork*** — starts an inactive network.
- **virsh net-undefine *nameOfInactiveNetwork*** — removes the definition of an inactive network.

Managing guests with the Virtual Machine Manager (virt-manager)

This section describes the Virtual Machine Manager (**`virt-manager`**) windows, dialog boxes, and various GUI controls.

`virt-manager` provides a graphical view of hypervisors and guest on your system and on remote machines. You can use **`virt-manager`** to define both para-virtualized and fully virtualized guests. **`virt-manager`** can perform virtualization management tasks, including:

- assigning memory,
- assigning virtual CPUs,
- monitoring operational performance,
- saving and restoring, pausing and resuming, and shutting down and starting virtualized guests,
- links to the textual and graphical consoles, and
- live and offline migrations.

23.1. The open connection window

This window appears first and prompts the user to choose a hypervisor session. Non-privileged users can initiate a read-only session. Root users can start a session with full blown read-write status. For normal use, select the **Local Xen host** option or QEMU (for KVM).

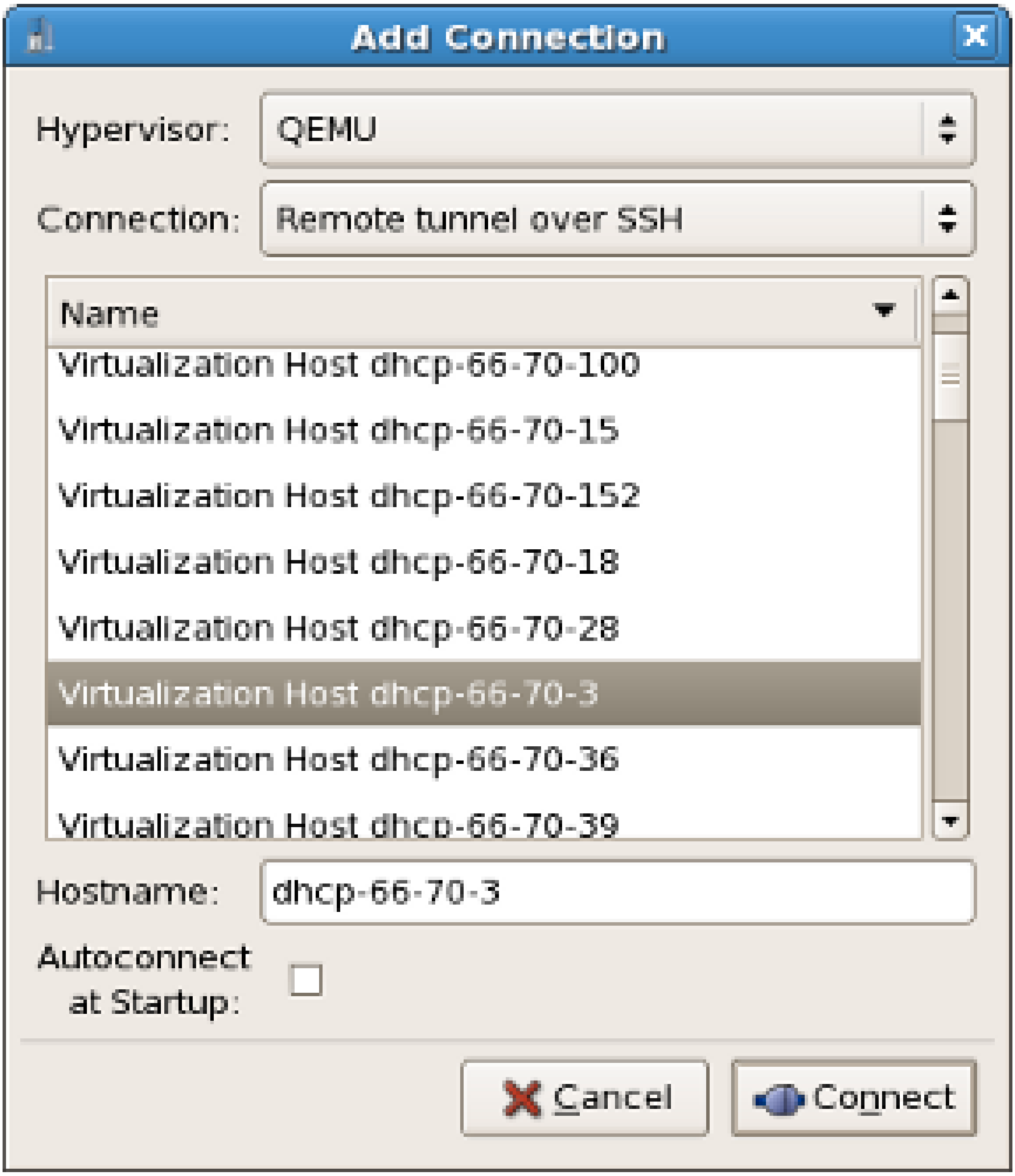


Figure 23.1. Virtual Machine Manager connection window

23.2. The Virtual Machine Manager main window

This main window displays all the running virtual machines and resources currently allocated to them (including domain0). You can decide which fields to display. Double-clicking on the desired virtual machine brings up the respective console for that particular machine. Selecting a virtual machine and double-click the **Details** button to display the Details window for that machine. You can also access the **File** menu to create a new virtual machine.

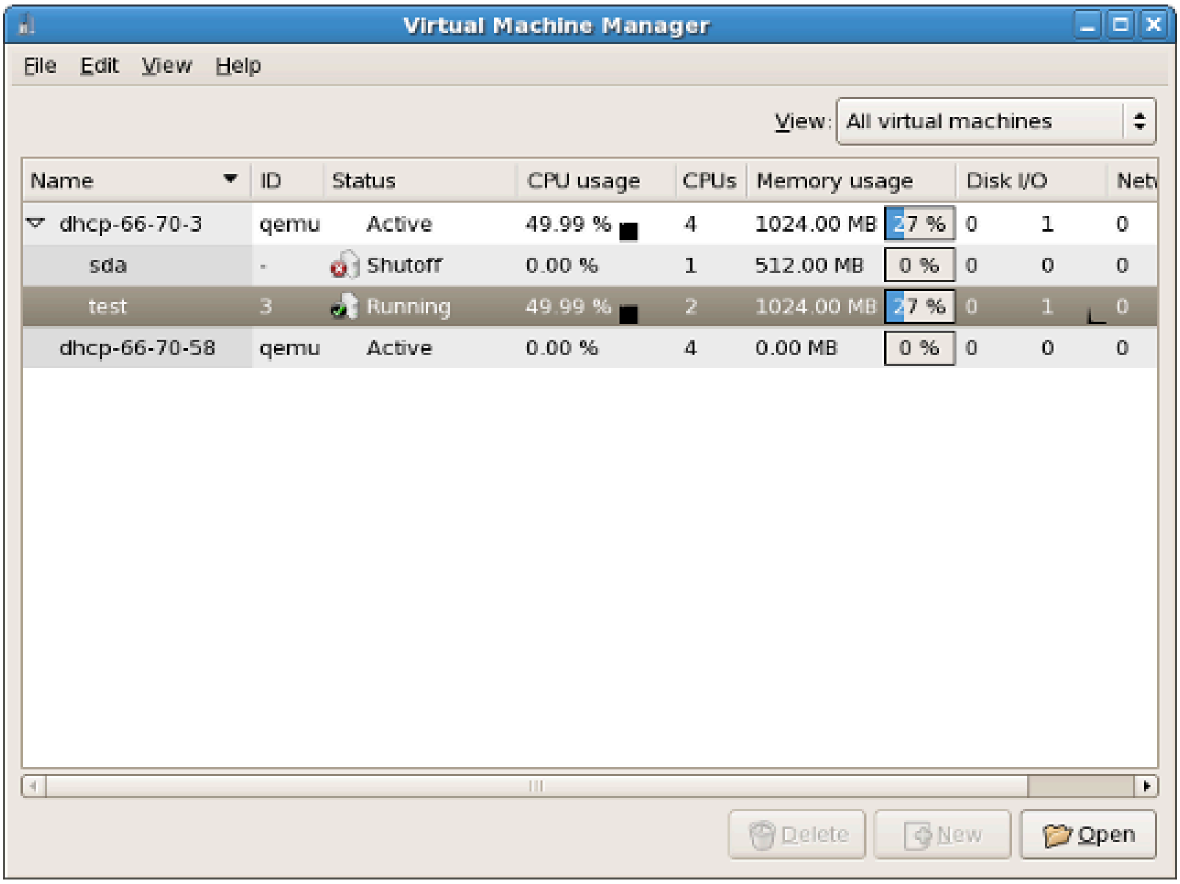


Figure 23.2. Virtual Machine Manager main window

23.3. The Virtual Machine Manager details window

This window displays graphs and statistics of a guest's live resource utilization data available from **`virt-manager`**. The UUID field displays the globally unique identifier for the virtual machines.

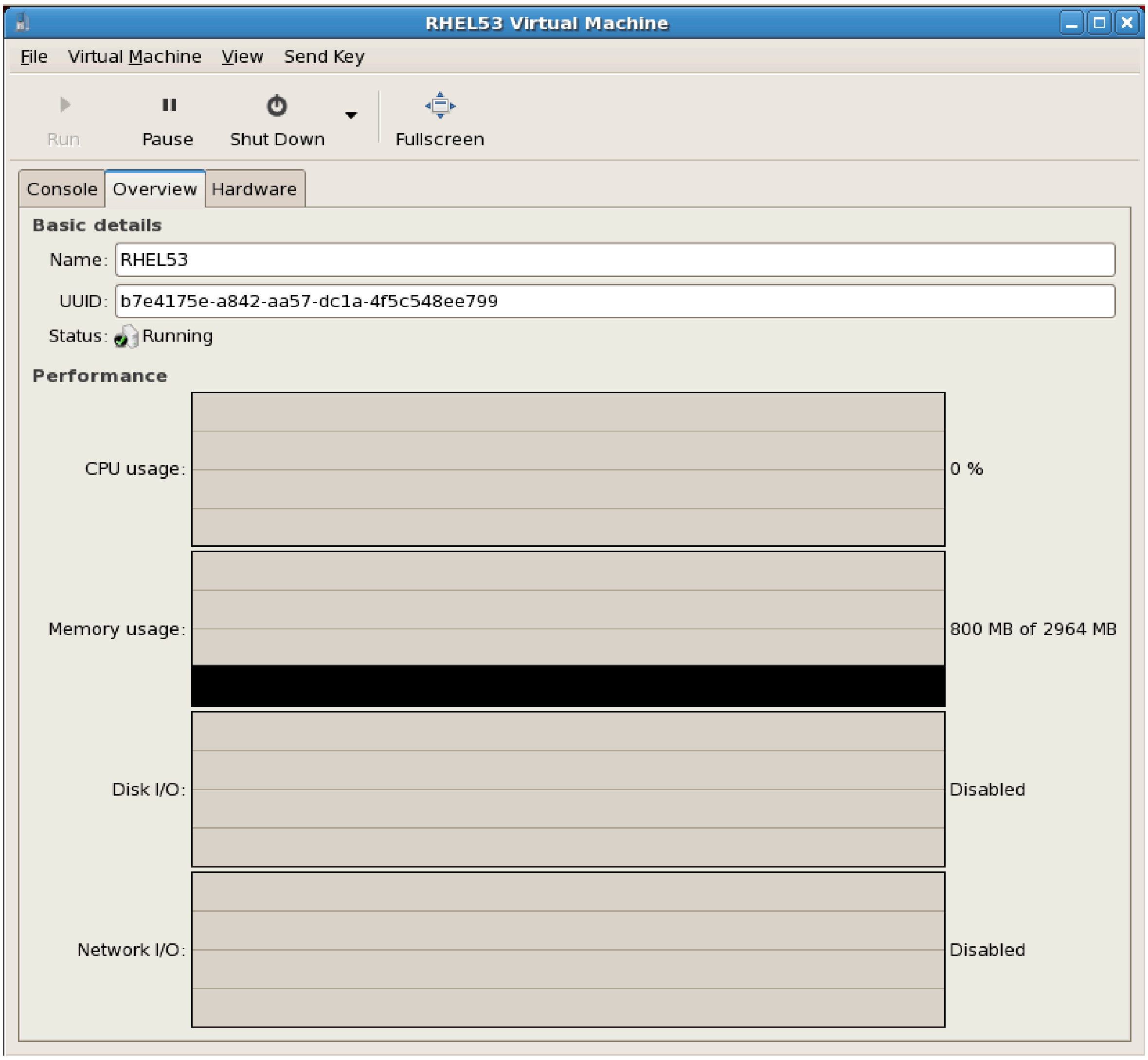


Figure 23.3. `virt-manager` details window

23.4. Virtual Machine graphical console

This window displays a virtual machine's graphical console. Para-virtualized and fully virtualized guests use different techniques to export their local virtual framebuffers, but both technologies use **VNC** to make them available to the Virtual Machine Manager's console window. If your virtual machine is set to require authentication, the Virtual Machine Graphical console prompts you for a password before the display appears.

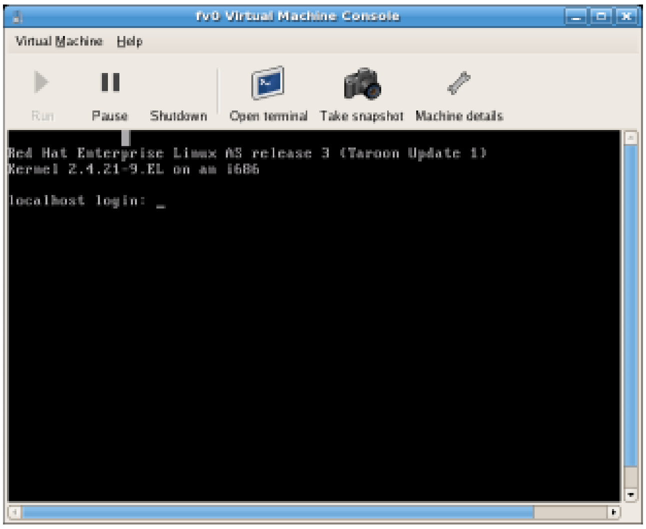


Figure 23.4. Graphical console window

A note on security and VNC

VNC is considered insecure by many security experts, however, several changes have been made to enable the secure usage of VNC for virtualization on Red Hat enterprise Linux. The guest machines only listen to the local host (`dom0`)'s loopback address (`127.0.0.1`). This ensures only those with shell privileges on the host can access virt-manager and the virtual machine through VNC.

Remote administration can be performed following the instructions in *Chapter 20, Remote management of virtualized guests*. TLS can provide enterprise level security for managing guest and host systems.

Your local desktop can intercept key combinations (for example, Ctrl+Alt+F11) to prevent them from being sent to the guest machine. You can use **`virt-manager`**sticky key' capability to send these sequences. You must press any modifier key (Ctrl or Alt) 3 times and the key you specify gets treated as active until the next non-modifier key is pressed. Then you can send Ctrl-Alt-F11 to the guest by entering the key sequence 'Ctrl Ctrl Ctrl Alt+F1'.

23.5. Starting virt-manager

To start **`virt-manager`** session open the **Applications** menu, then the **System Tools** menu and select **Virtual Machine Manager** (**`virt-manager`**).

The **`virt-manager`** main window appears.

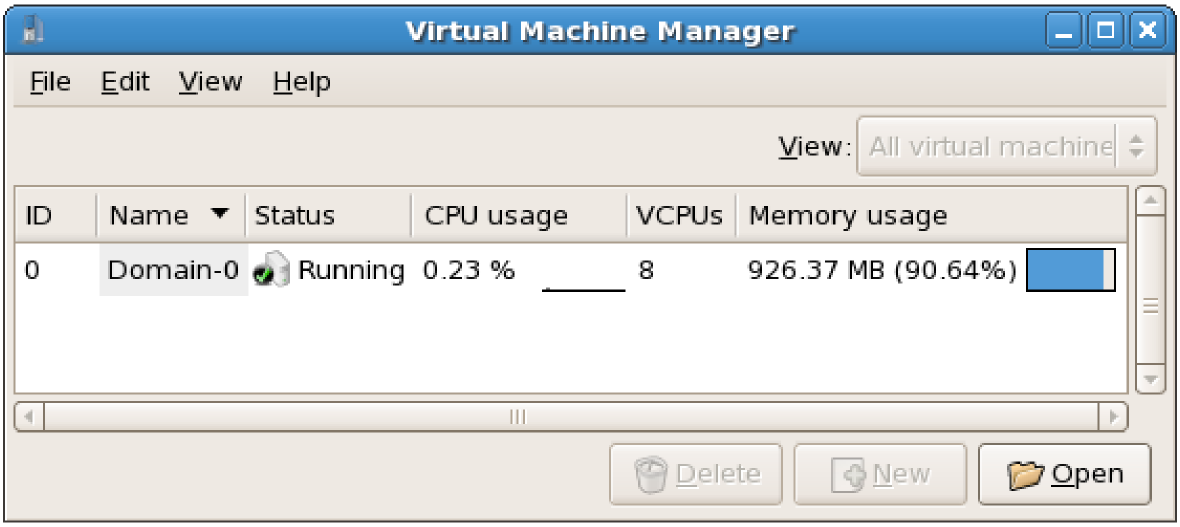


Figure 23.5. Starting **`virt-manager`**

Alternatively, **`virt-manager`** can be started remotely using ssh as demonstrated in the following command:

```
ssh -X host's address[remotehost]# virt-manager
```

Using **ssh** to manage virtual machines and hosts is discussed further in *Section 20.1, “Remote management with SSH”*.

23.6. Restoring a saved machine

After you start the Virtual Machine Manager, all virtual machines on your system are displayed in the main window. Domain0 is your host system. If there are no machines present, this means that currently there are no machines running on the system.

To restore a previously saved session:

1. From the **File** menu, select **Restore a saved machine**.

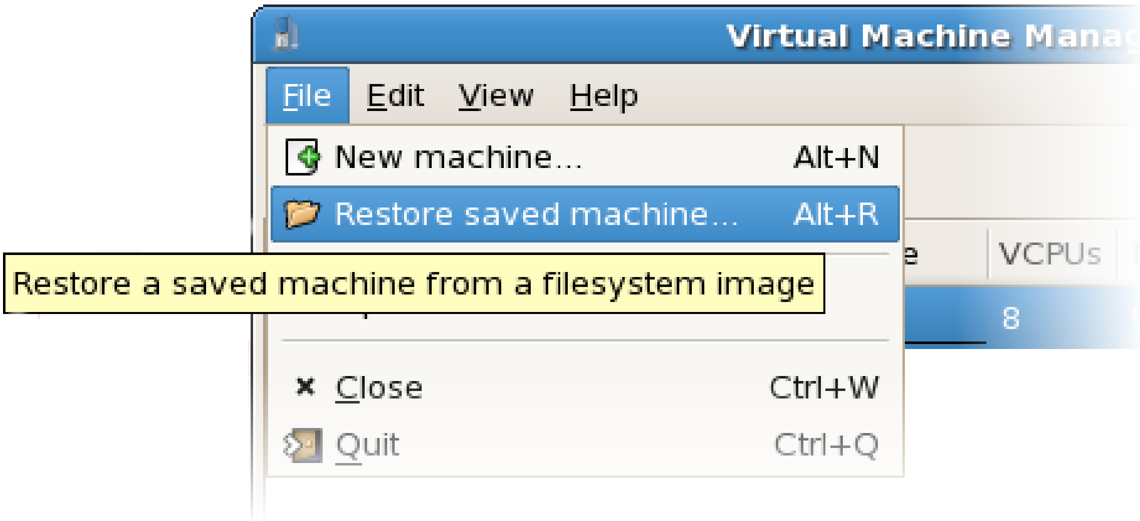


Figure 23.6. Restoring a virtual machine

2. The **Restore Virtual Machine** main window appears.

3. Navigate to correct directory and select the saved session file.

4. Click **Open**.

The saved virtual system appears in the Virtual Machine Manager main window.

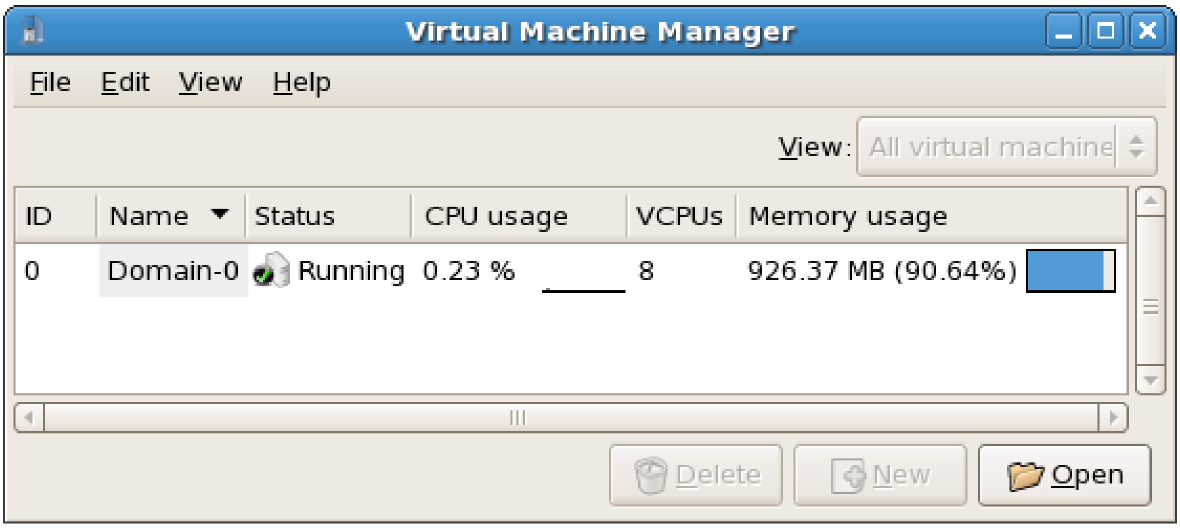


Figure 23.7. A restored virtual machine manager session

23.7. Displaying guest details

You can use the Virtual Machine Monitor to view activity data information for any virtual machines on your system.

To view a virtual system's details:

1. In the Virtual Machine Manager main window, highlight the virtual machine that you want to view.

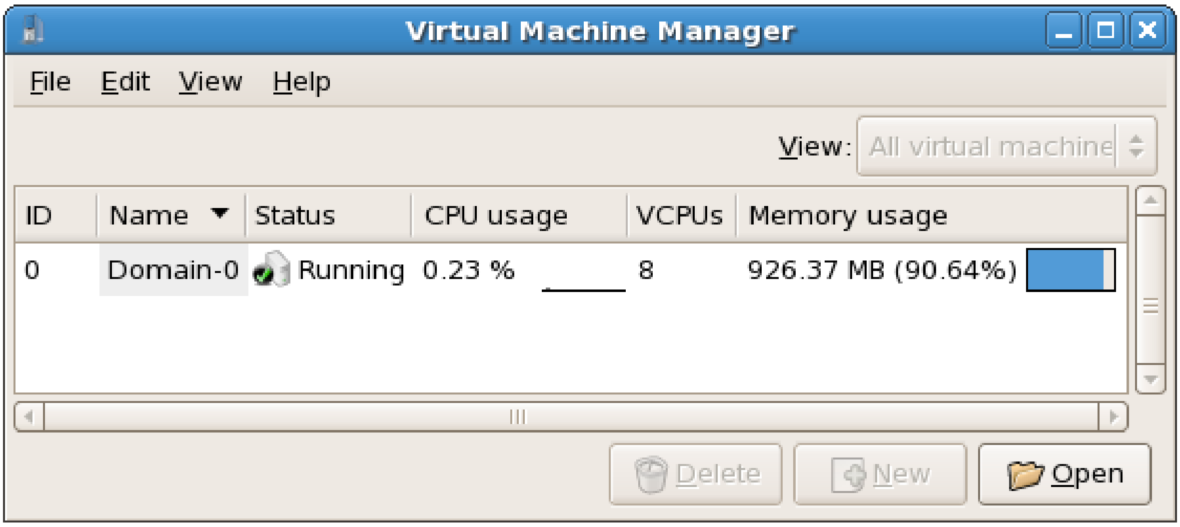


Figure 23.8. Selecting a virtual machine to display

2. From the Virtual Machine Manager **Edit** menu, select **Machine Details** (or click the **Details** button on the bottom of the Virtual Machine Manager main window).

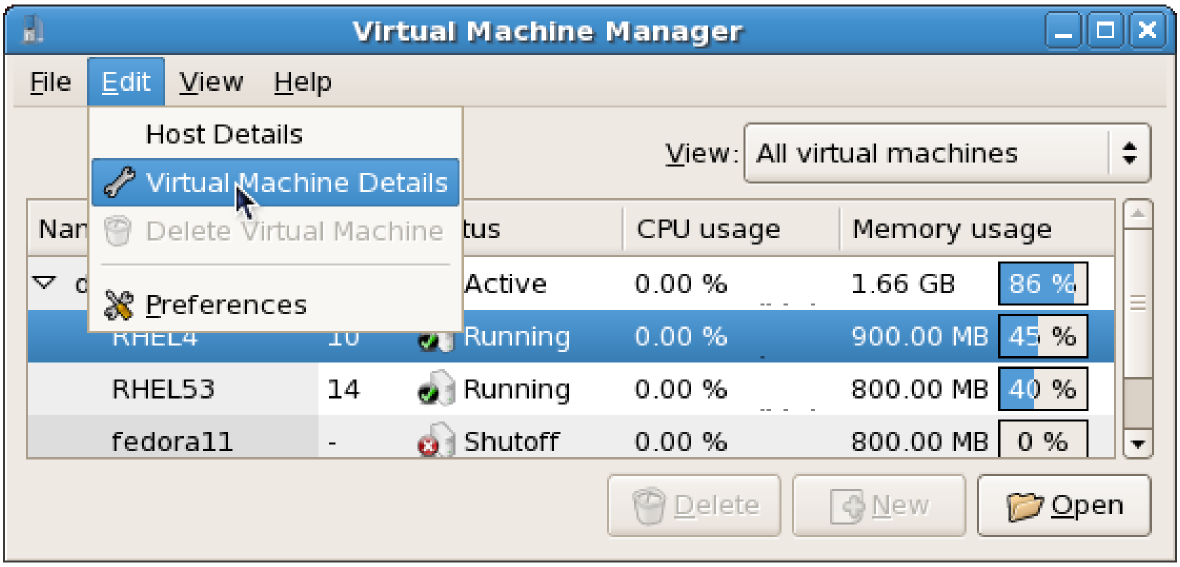


Figure 23.9. Displaying virtual machine details menu

3. On the **Virtual Machine Details** window, click the **Hardware**tab.

   The **Virtual Machine Details Hardware** window appears.

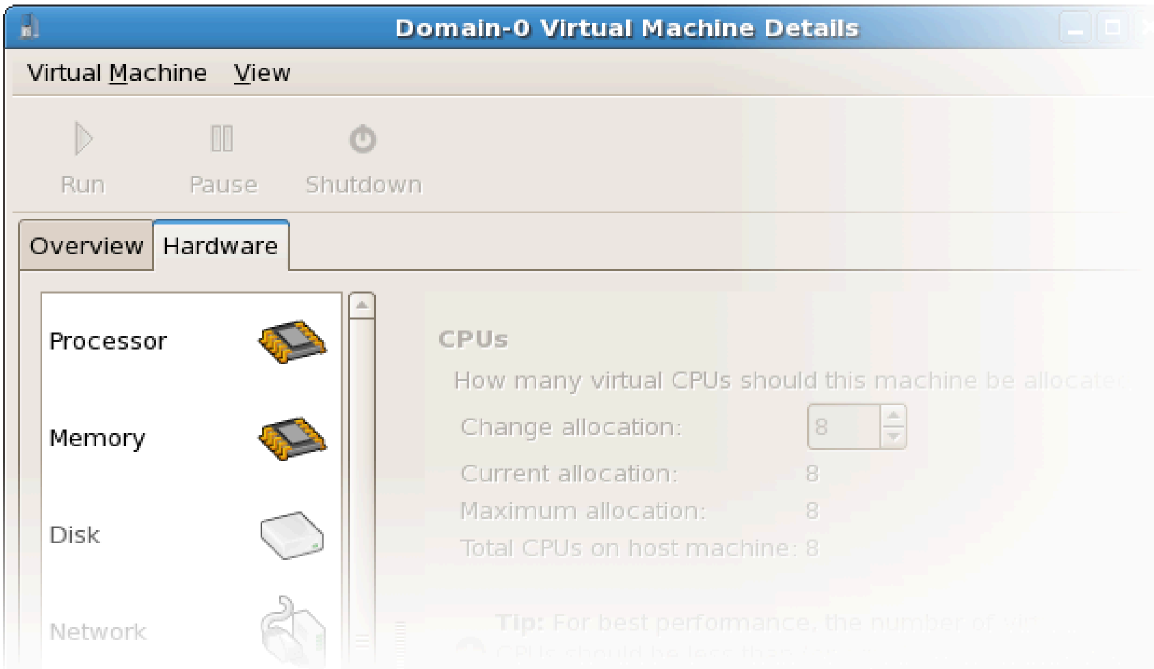


Figure 23.11. Displaying guest hardware details

4. On the **Hardware** tab, click on **Processor** to view or change the current processor allocation.

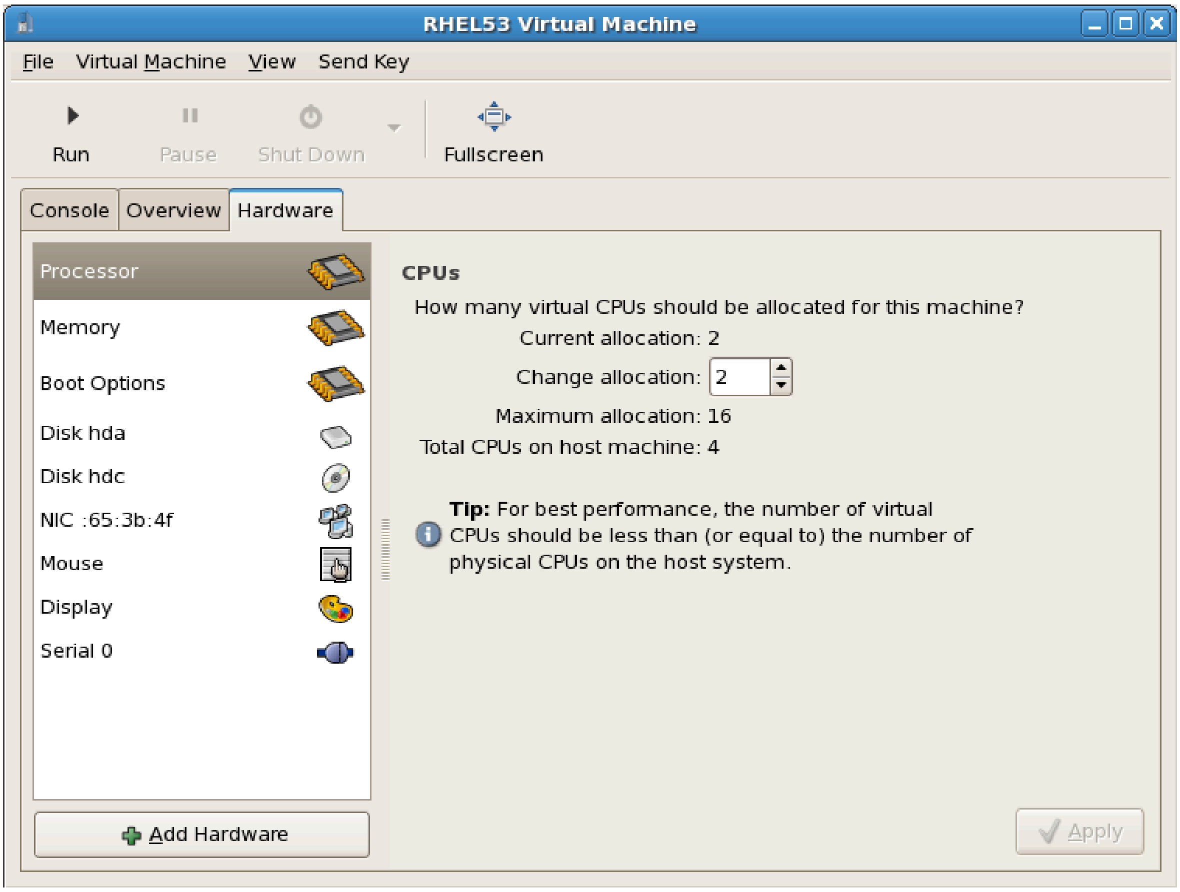


Figure 23.12. Processor allocation panel

5. On the **Hardware** tab, click on **Memory** to view or change the current RAM memory allocation.

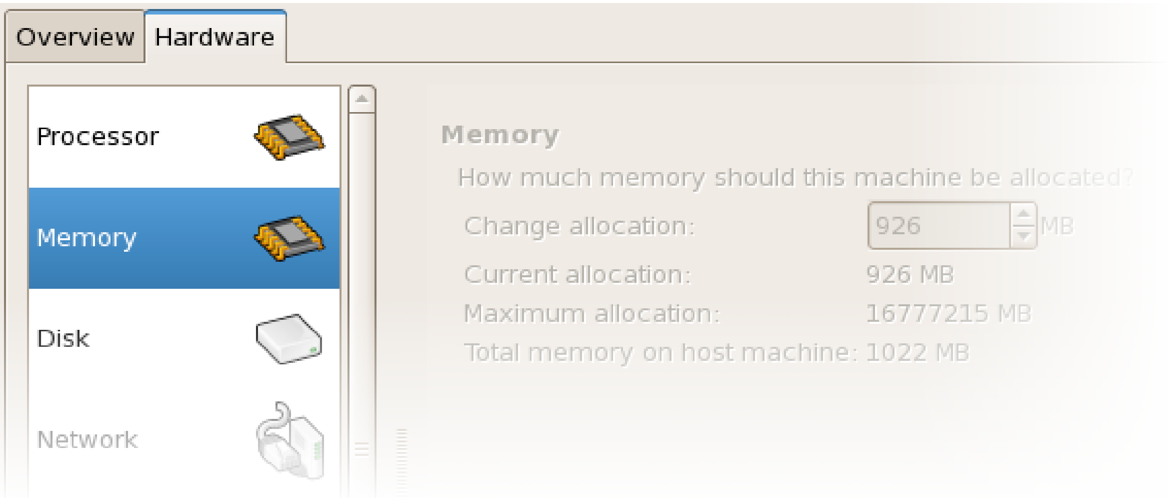


Figure 23.13. Displaying memory allocation

6. On the **Hardware** tab, click on **Disk** to view or change the current hard disk configuration.

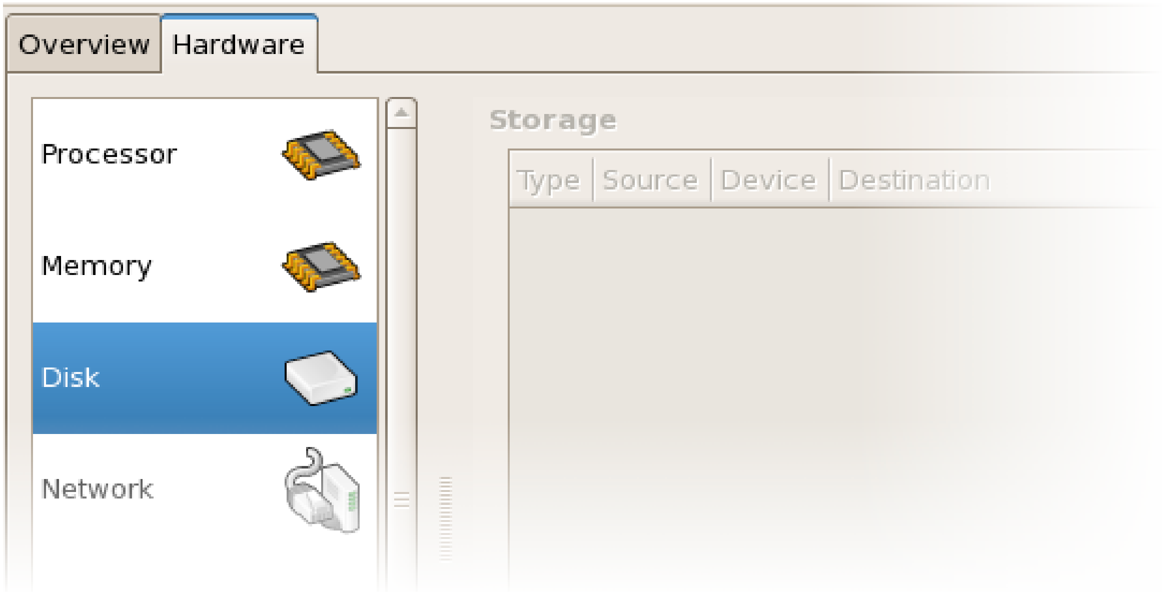


Figure 23.14. Displaying disk configuration

7. On the **Hardware** tab, click on **Network** to view or change the current network configuration.

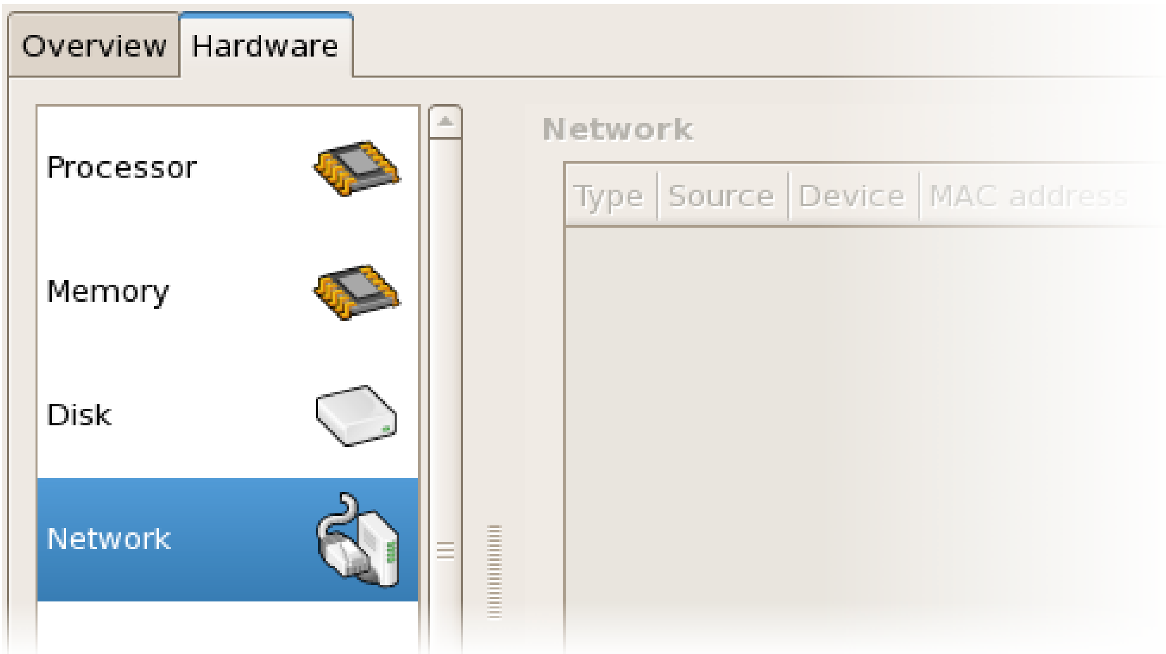


Figure 23.15. Displaying network configuration

23.8. Status monitoring

You can use the Virtual Machine Manager to modify the virtual system Status monitoring.

To configure Status monitoring, and enable Consoles:

1. From the **Edit** menu, select **Preferences**.

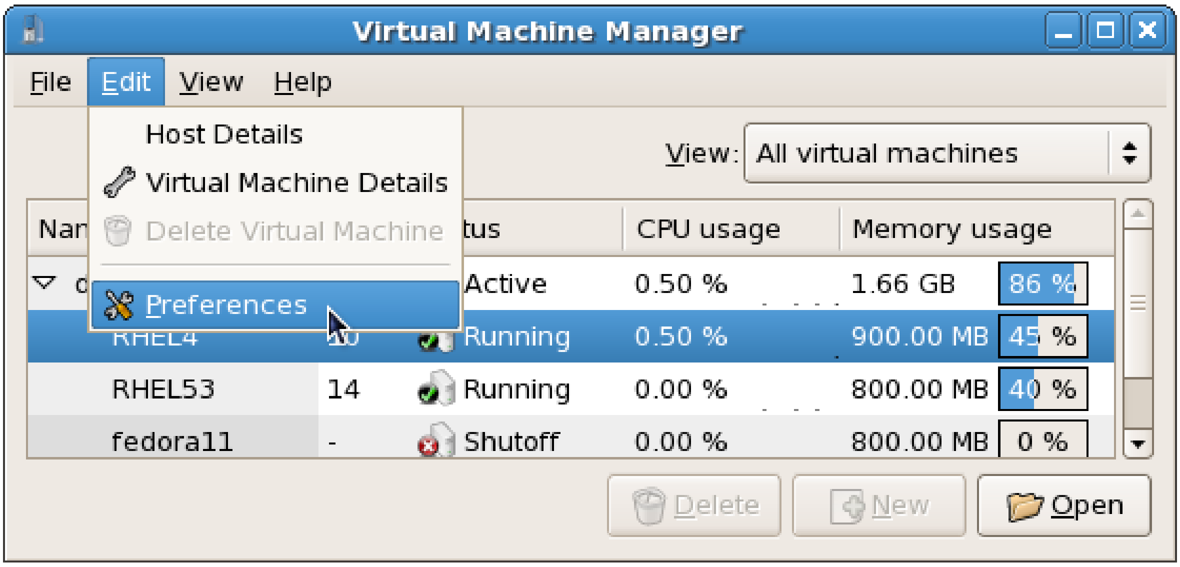


Figure 23.16. Modifying guest preferences

The Virtual Machine Manager Preferences window appears.

2. From the Status monitoring area selection box, specify the time (in seconds) that you want the system to update.

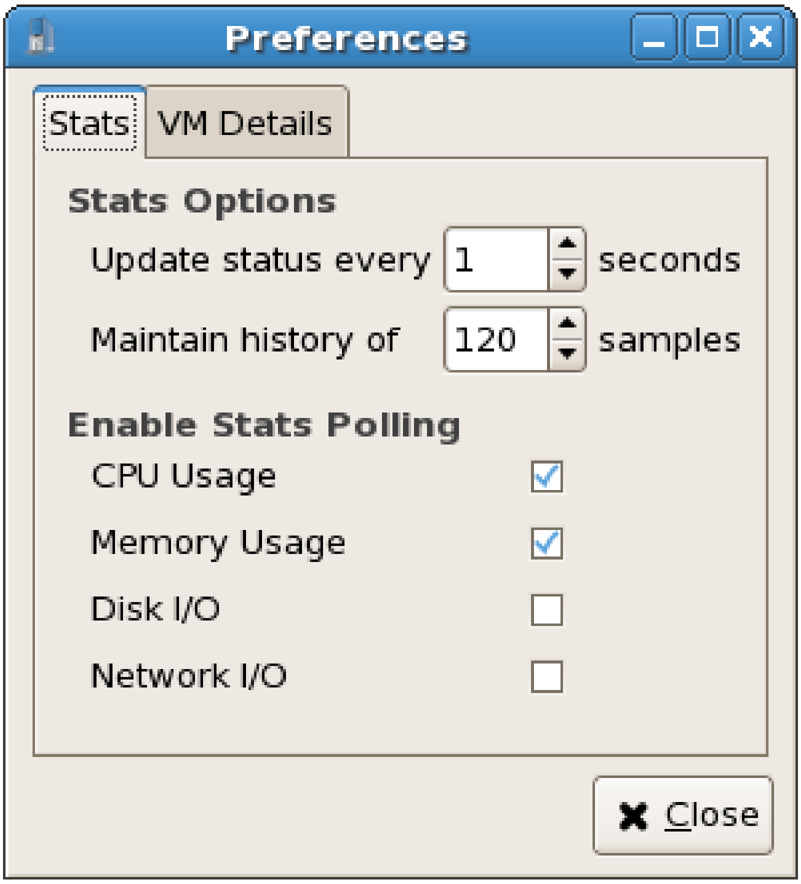


Figure 23.17. Configuring status monitoring

3. From the Consoles area, specify how to open a console and specify an input device.

23.9. Displaying guest identifiers

To view the guest IDs for all virtual machines on your system:

1. From the **View** menu, select the **Domain ID** check box.

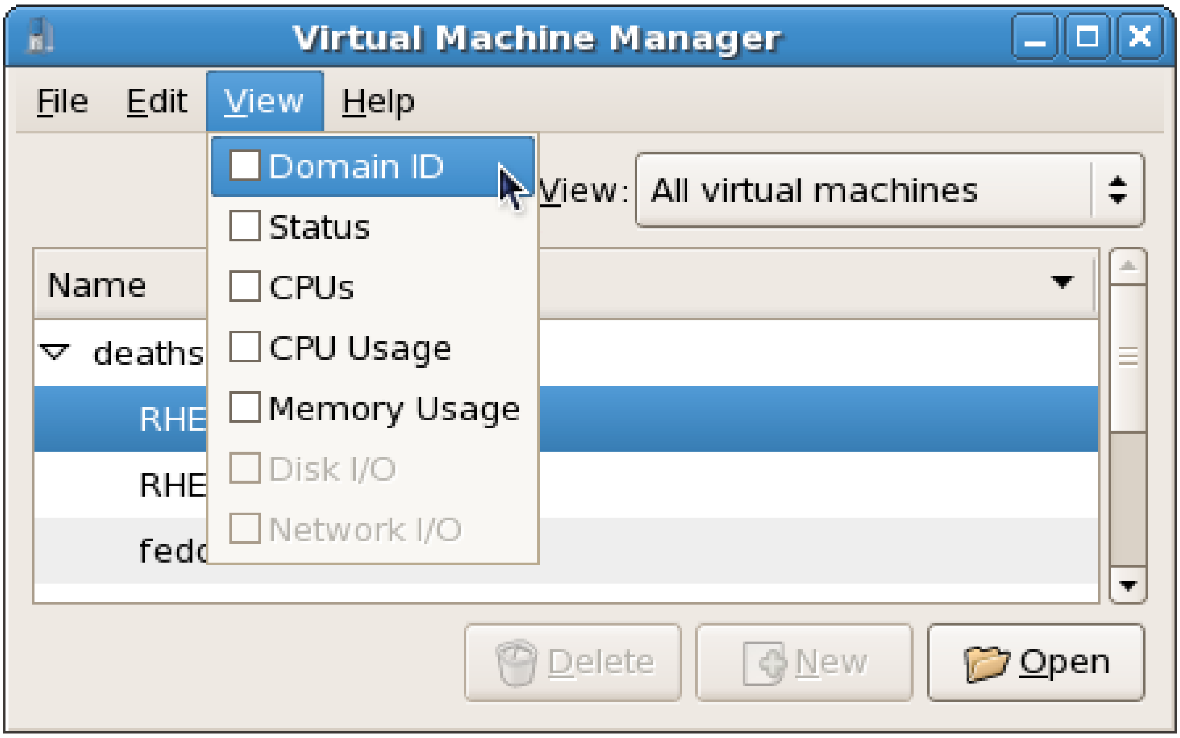


Figure 23.18. Viewing guest IDs

2. The Virtual Machine Manager lists the Domain IDs for all domains on your system.

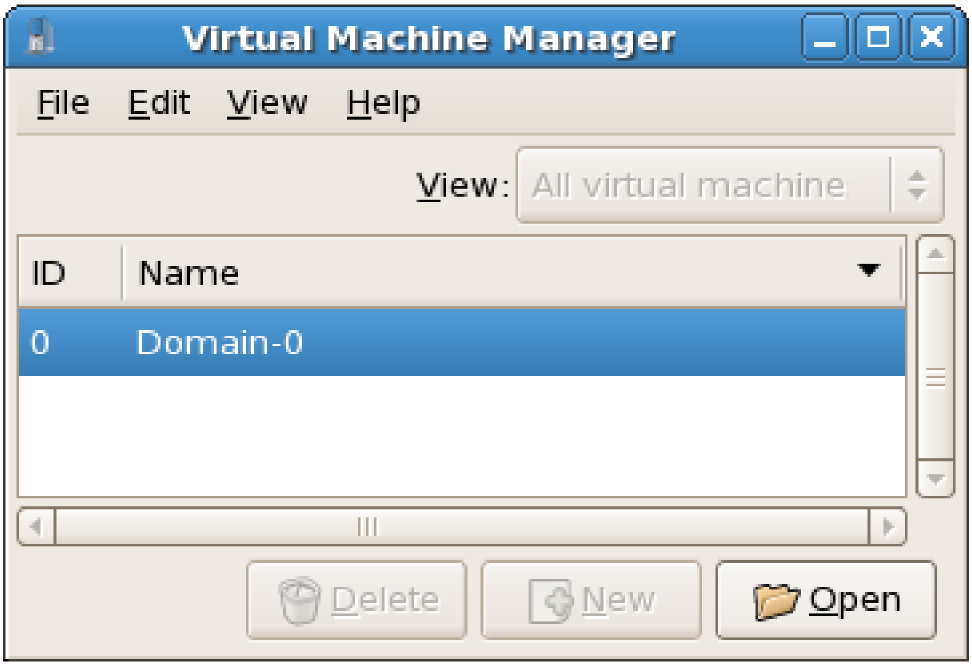


Figure 23.19. Displaying domain IDs

23.10. Displaying a guest's status

To view the status of all virtual machines on your system:

1. From the **View** menu, select the **Status** check box.

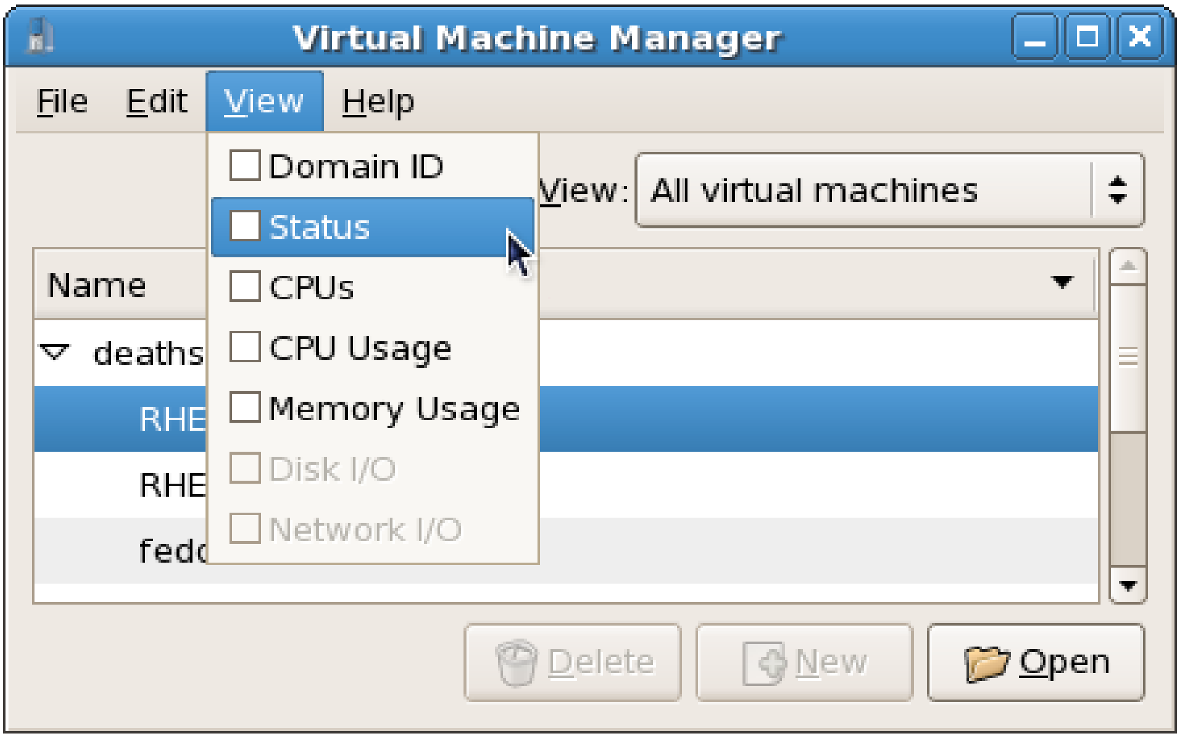


Figure 23.20. Selecting a virtual machine's status

2. The Virtual Machine Manager lists the status of all virtual machines on your system.

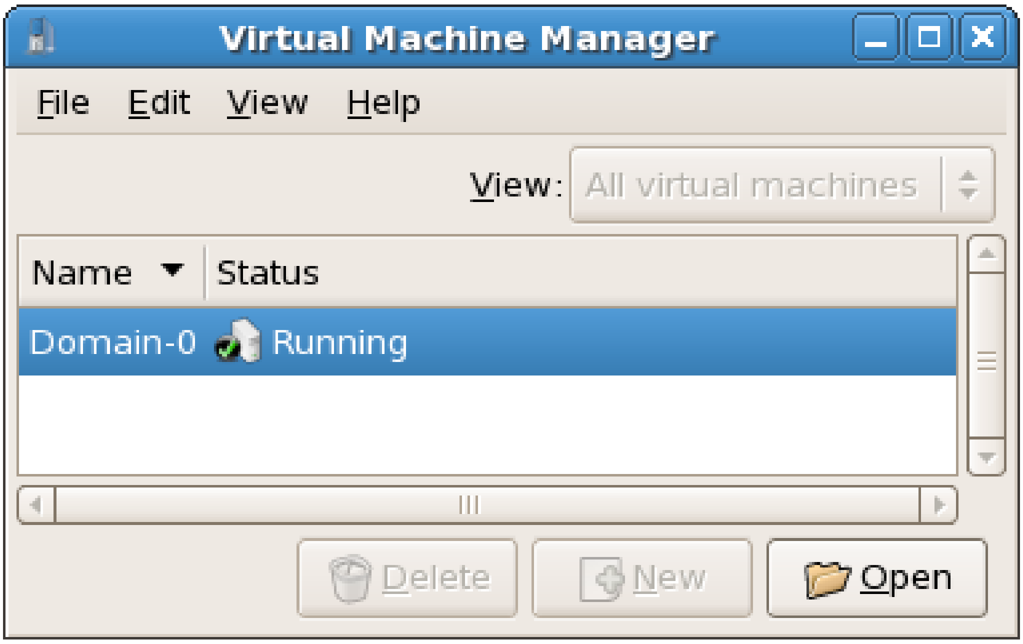


Figure 23.21. Displaying a virtual machine's status

23.11. Displaying virtual CPUs

To view the amount of virtual CPUs for all virtual machines on your system:

1. From the **View** menu, select the **Virtual CPUs** check box.

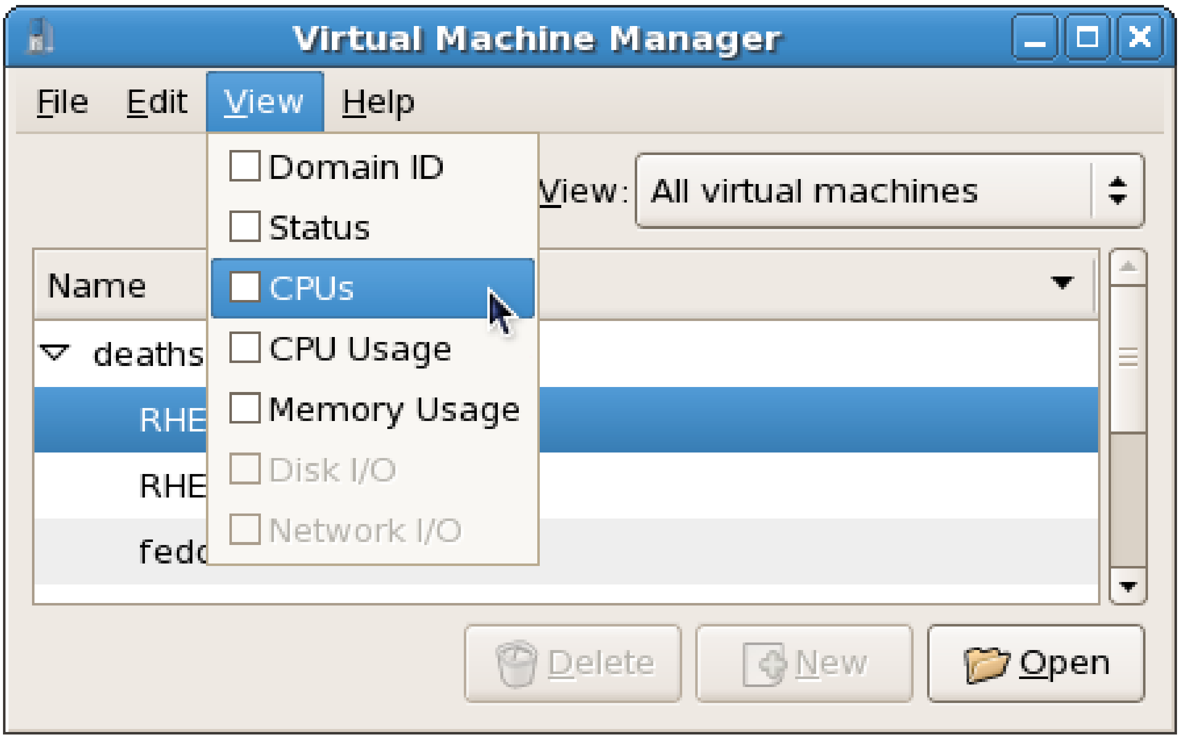


Figure 23.22. Selecting the virtual CPUs option

2. The Virtual Machine Manager lists the Virtual CPUs for all virtual machines on your system.

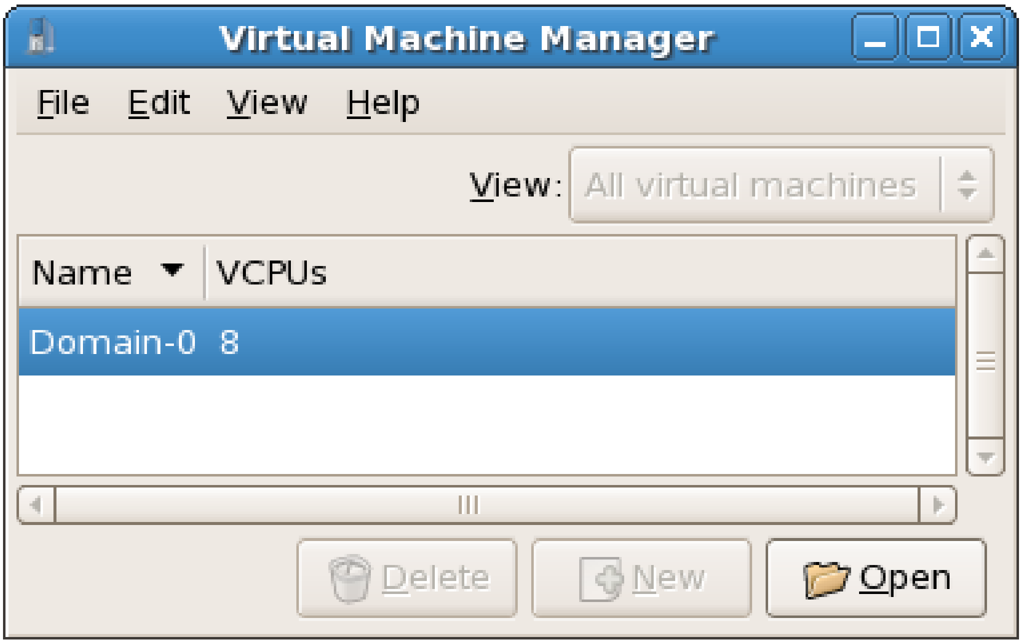


Figure 23.23. Displaying Virtual CPUs

23.12. Displaying CPU usage

To view the CPU usage for all virtual machines on your system:

1. From the **View** menu, select the **CPU Usage** check box.

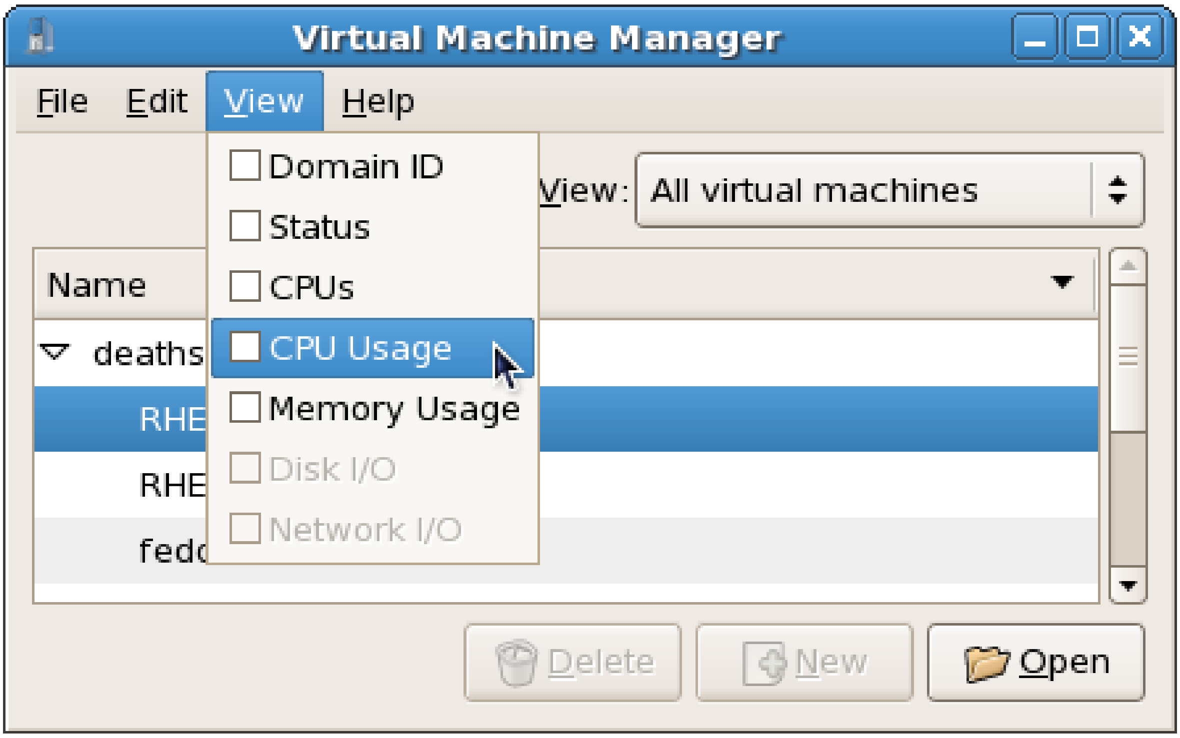


Figure 23.24. Selecting CPU usage

2. The Virtual Machine Manager lists the percentage of CPU in use for all virtual machines on your system.

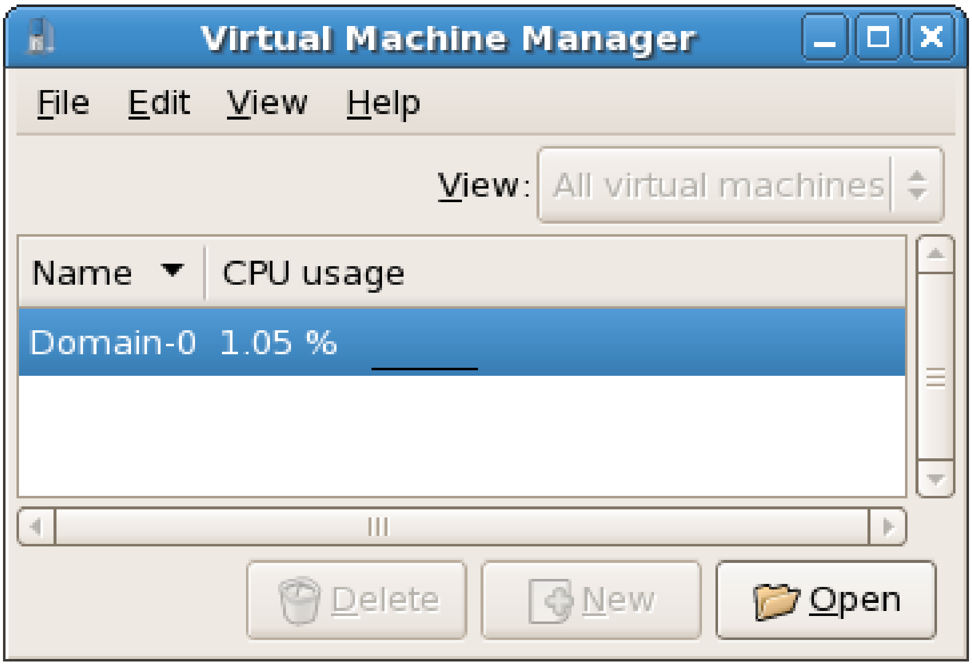


Figure 23.25. Displaying CPU usage

23.13. Displaying memory usage

To view the memory usage for all virtual machines on your system:

1. From the **View** menu, select the **Memory Usage** check box.

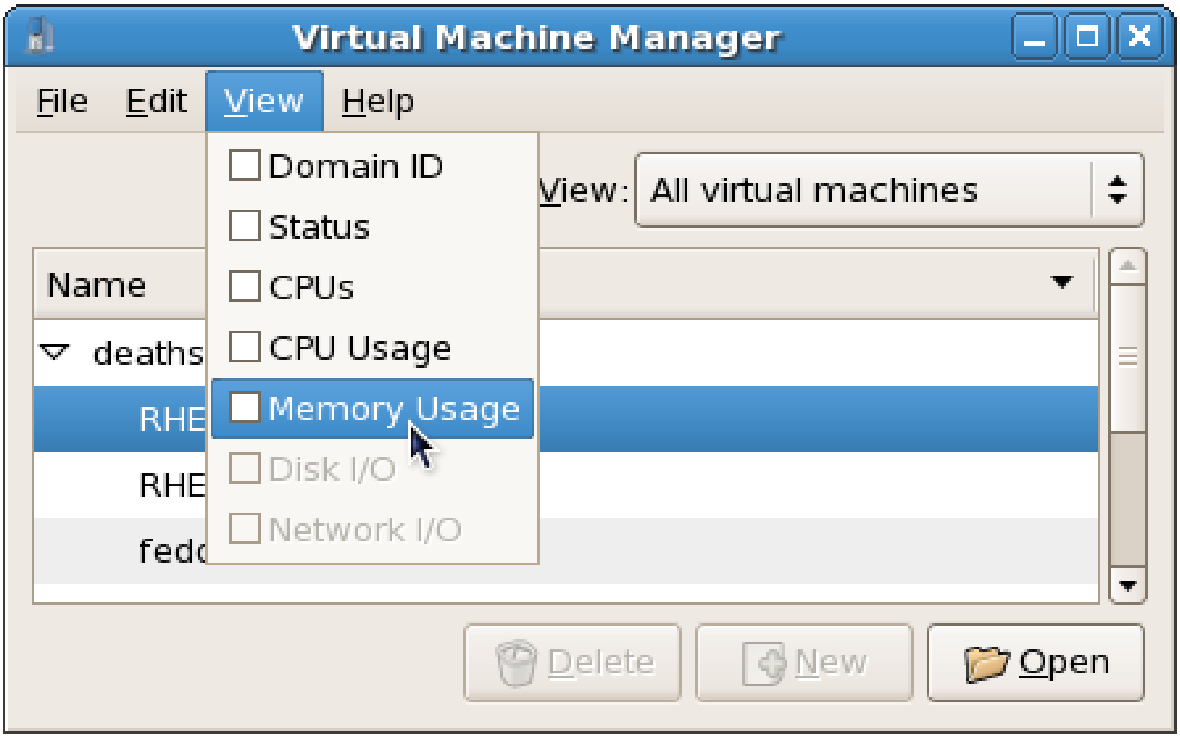


Figure 23.26. Selecting Memory Usage

2. The Virtual Machine Manager lists the percentage of memory in use (in megabytes) for all virtual machines on your system.

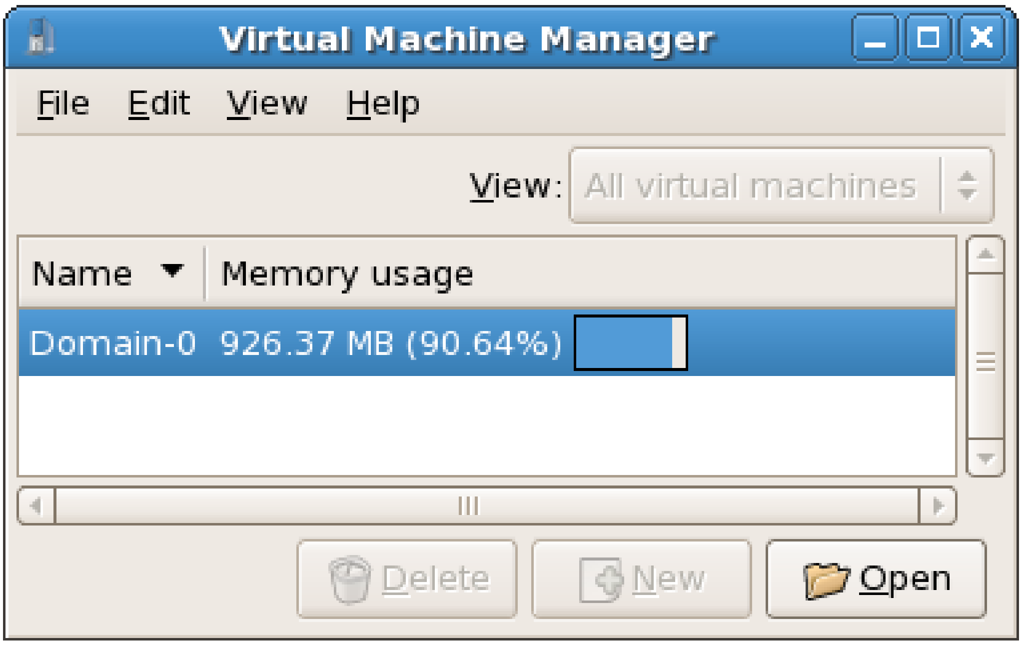


Figure 23.27. Displaying memory usage

23.14. Managing a virtual network

To configure a virtual network on your system:

1. From the **Edit** menu, select **Host Details**.

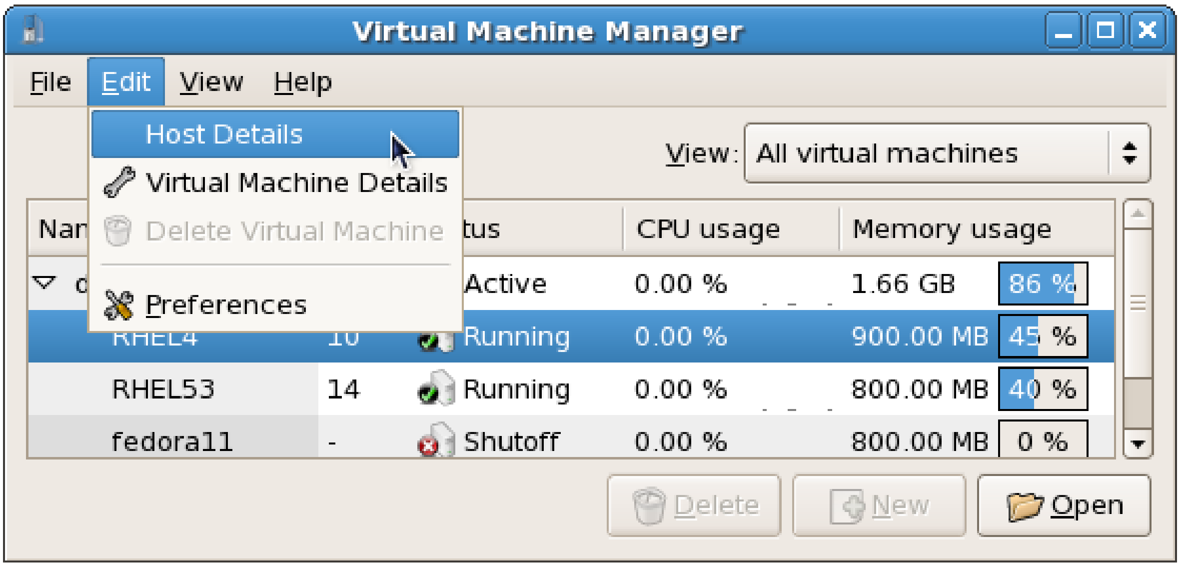


Figure 23.28. Selecting a host's details

2. This will open the **Host Details** menu. Click the **Virtual Networks** tab.

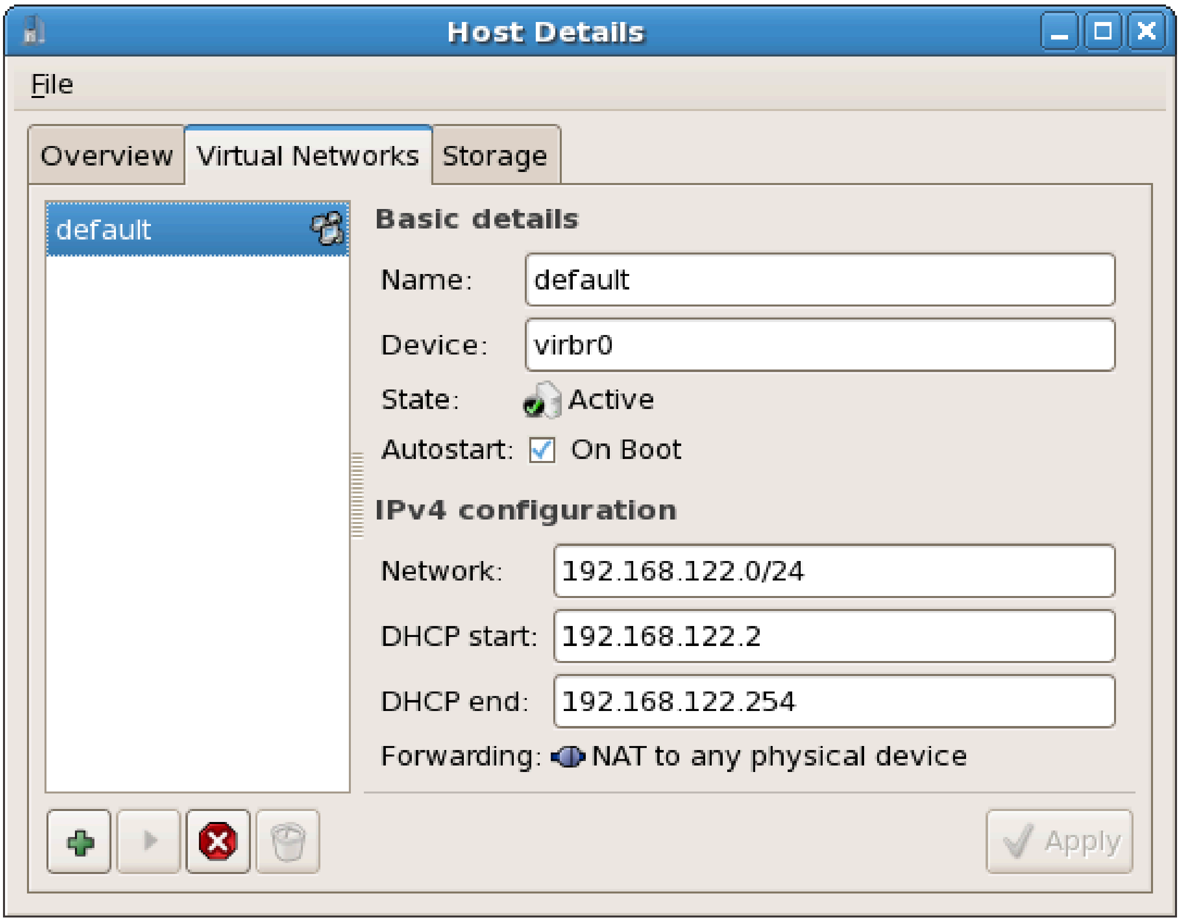


Figure 23.29. Virtual network configuration

3. All available virtual networks are listed on the left-hand box of the menu. You can edit the configuration of a virtual network by selecting it from this box and editing as you see fit.

23.15. Creating a virtual network

To create a virtual network on your system:

1. Open the **Host Details** menu (refer to *Section 23.14, "Managing a virtual network"*) and click the **Add** button.

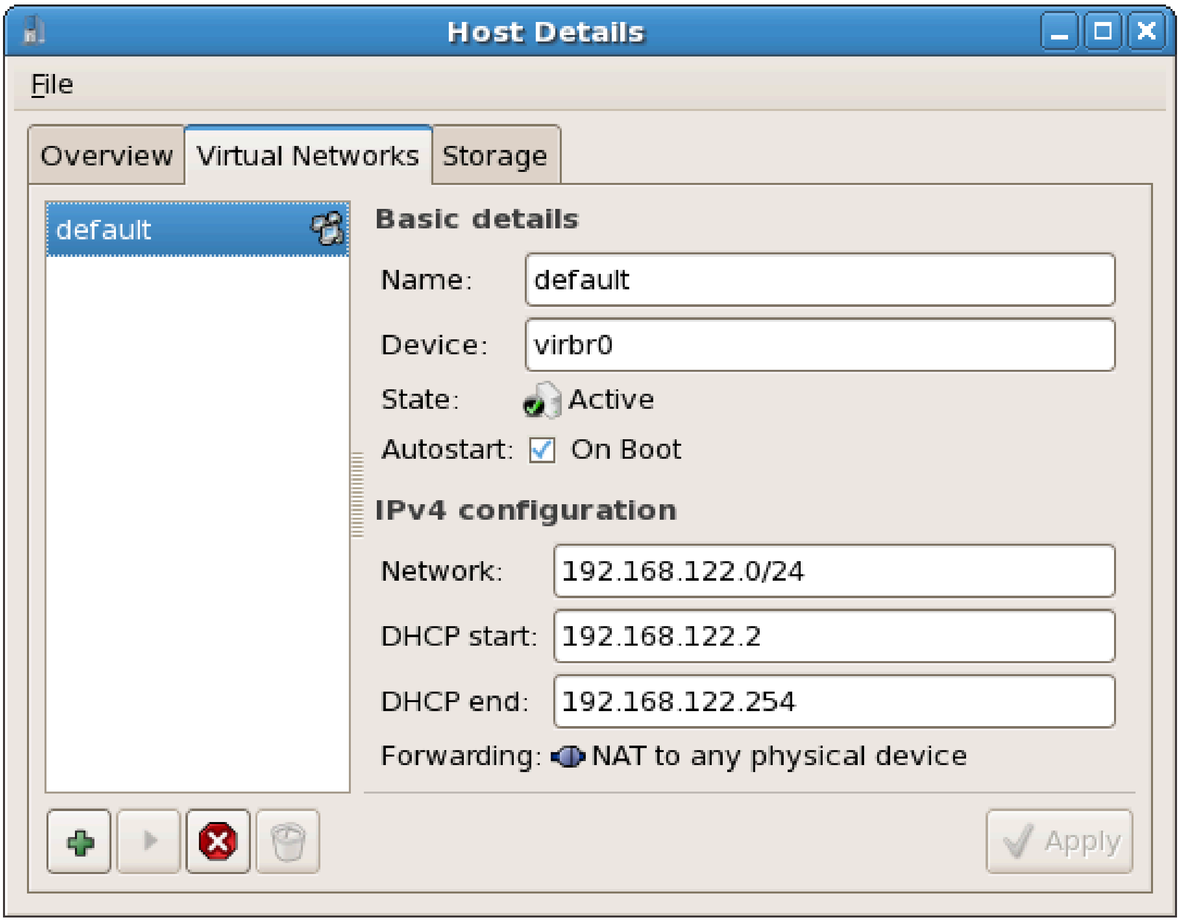


Figure 23.30. Virtual network configuration

2. Enter an appropriate name for your virtual network and click **Forward**.

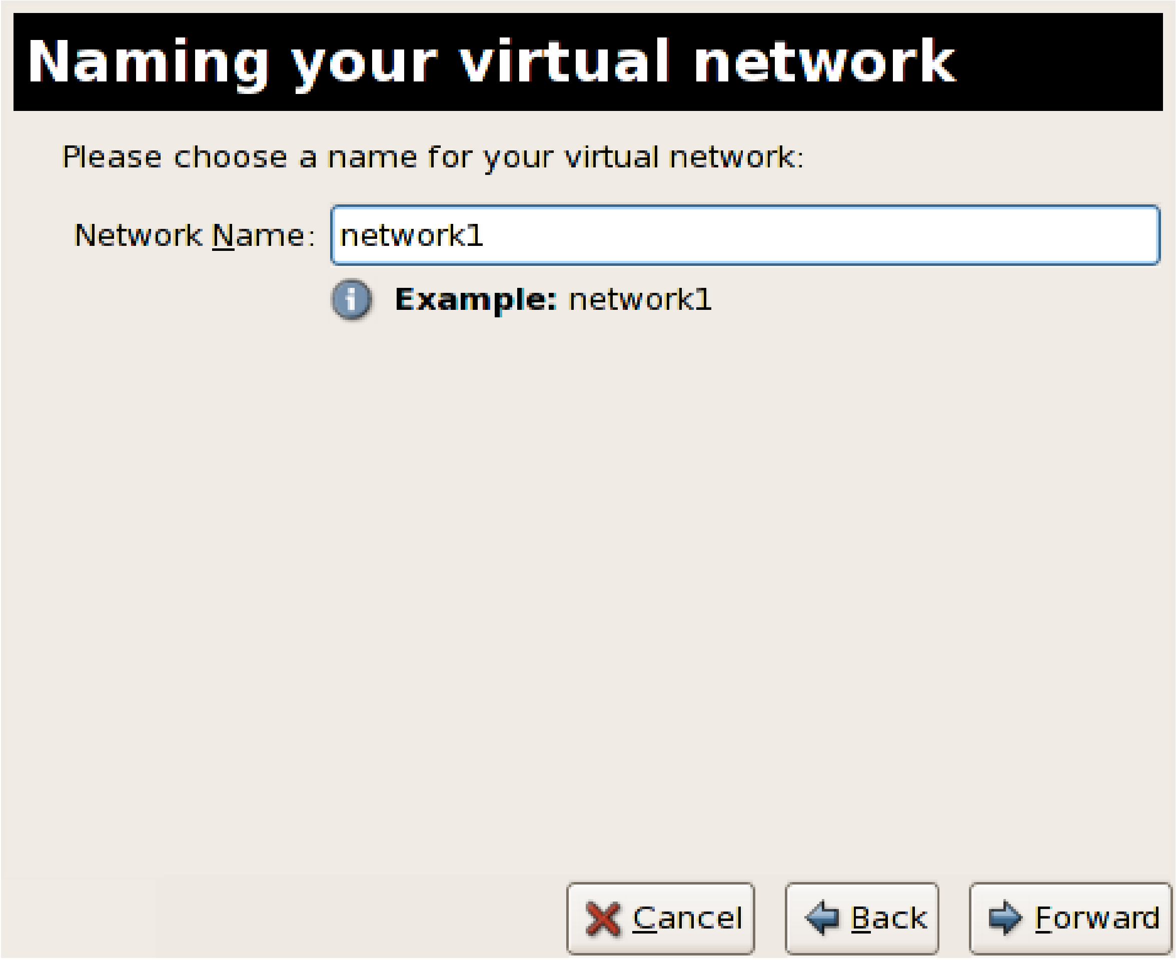


Figure 23.32. Naming your virtual network

3. Enter an IPv4 address space for your virtual network and click **Forward**.

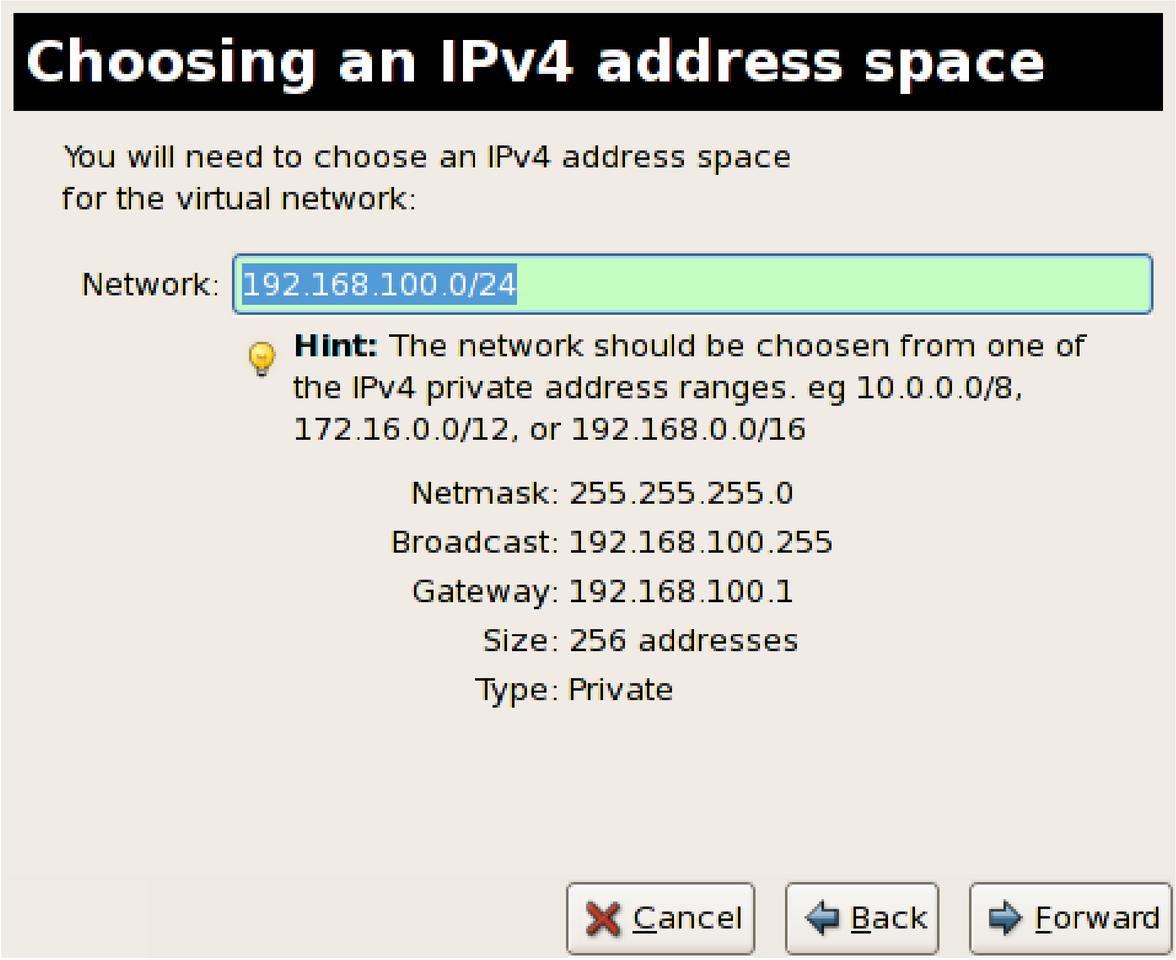


Figure 23.33. Choosing an IPv4 address space

4. Define the DHCP range for your virtual network by specifying a **Start** and **End** range of IP addresses. Click **Forward** to continue.

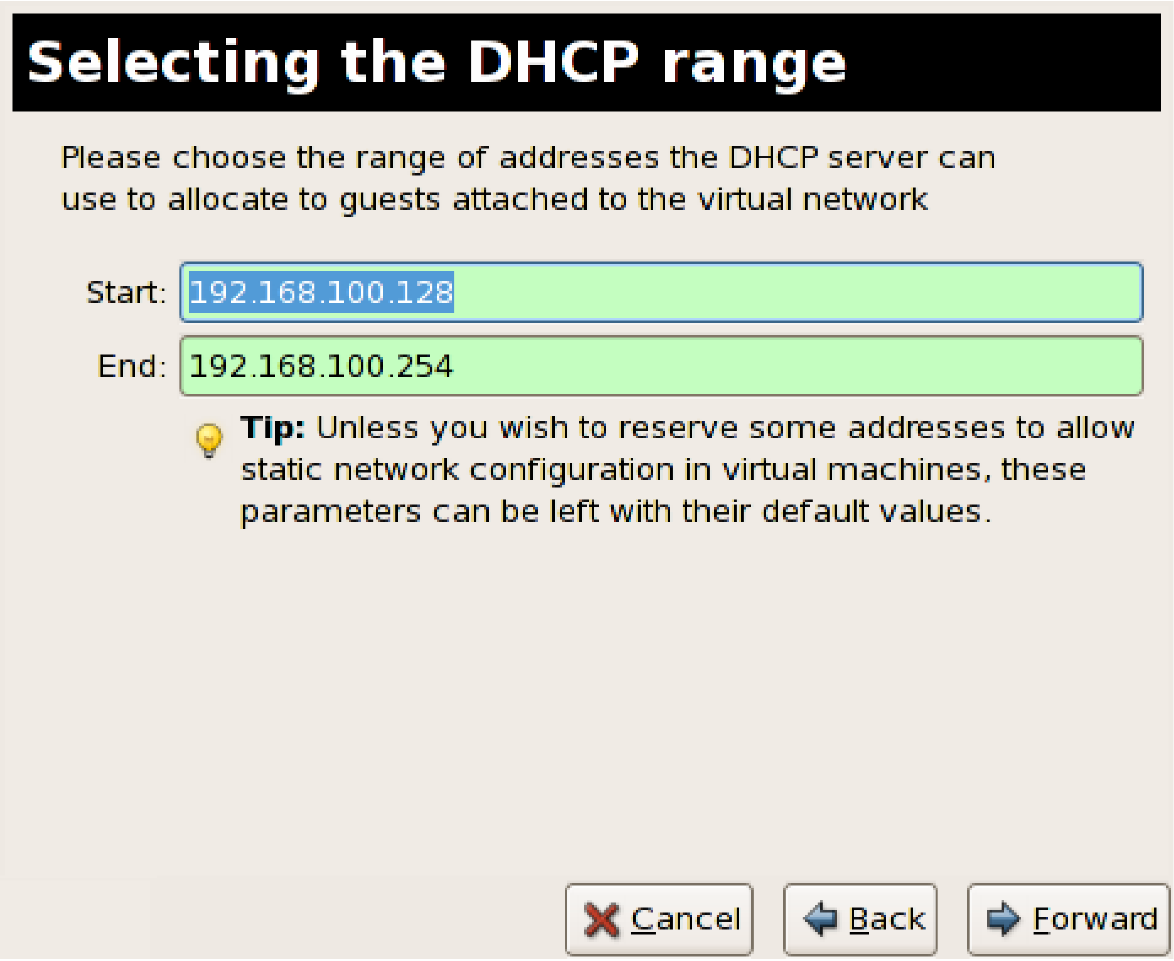


Figure 23.34. Selecting the DHCP range

5. Select how the virtual network should connect to the physical network.

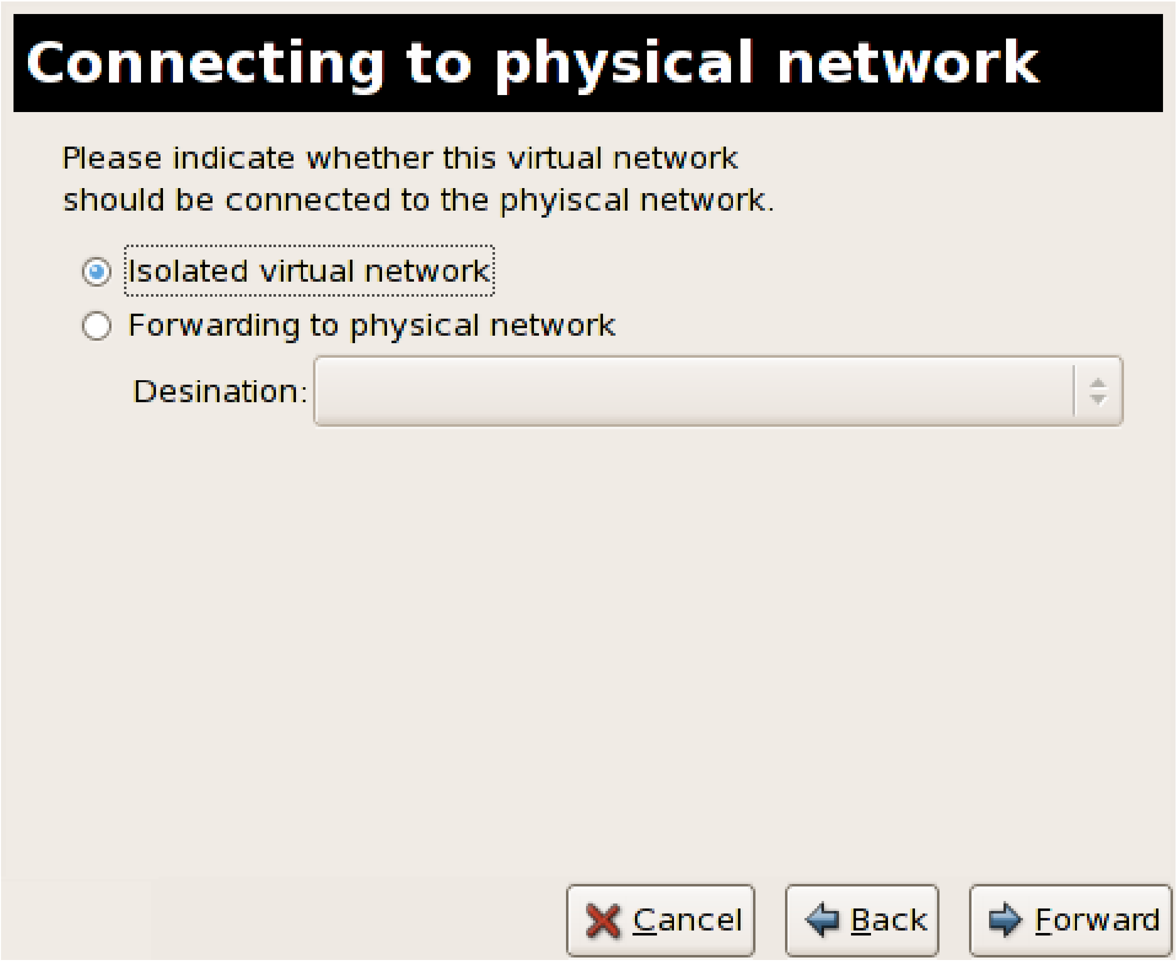


Figure 23.35. Connecting to physical network

If you select **Forwarding to physical network**, choose whether the **Destination** should be **NAT to any physical device** or **NAT to physical device eth0**.

Click **Forward** to continue.

6. You are now ready to create the network. Check the configuration of your network and click **Finish**.

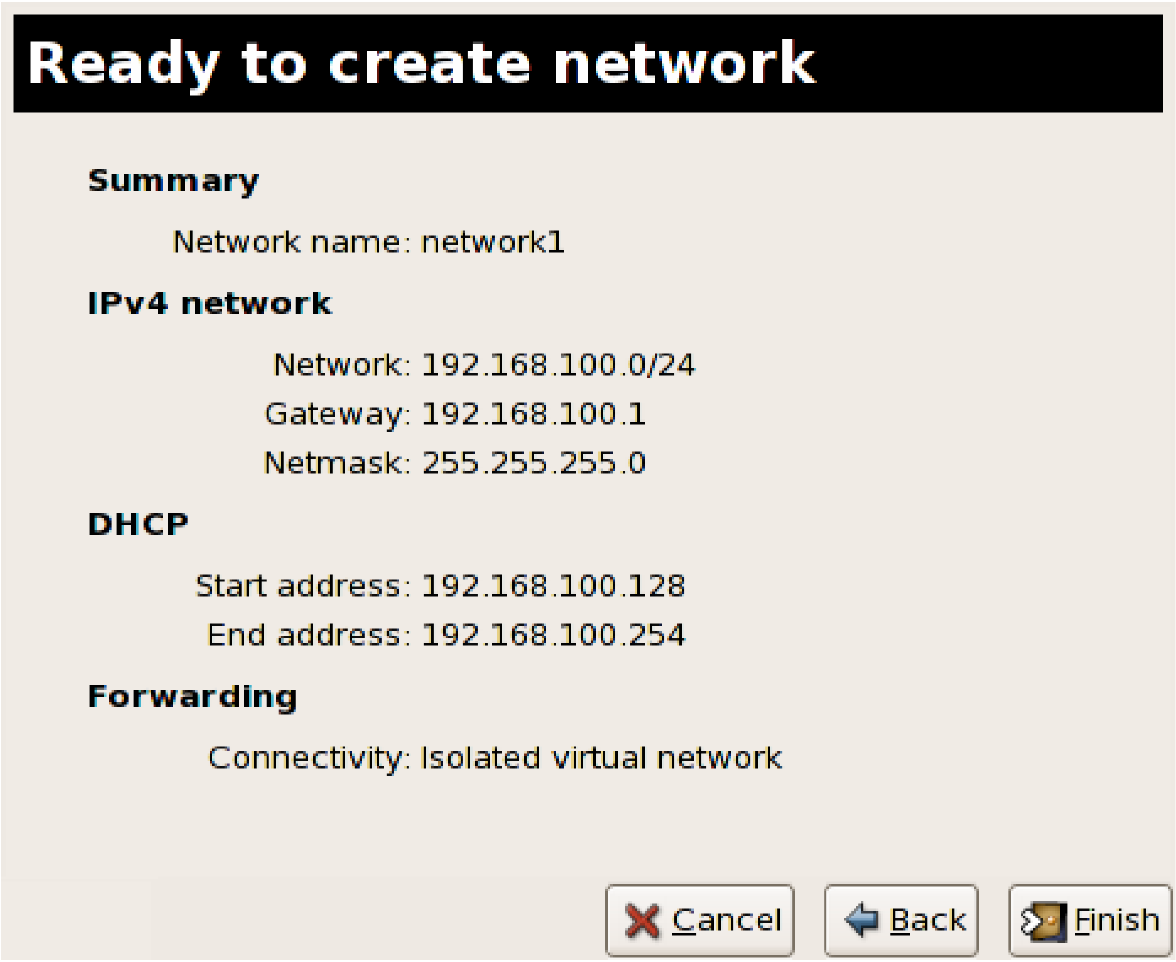


Figure 23.36. Ready to create network

7. The new virtual network is now available in the **Virtual Network** tab of the **Host Details** menu.

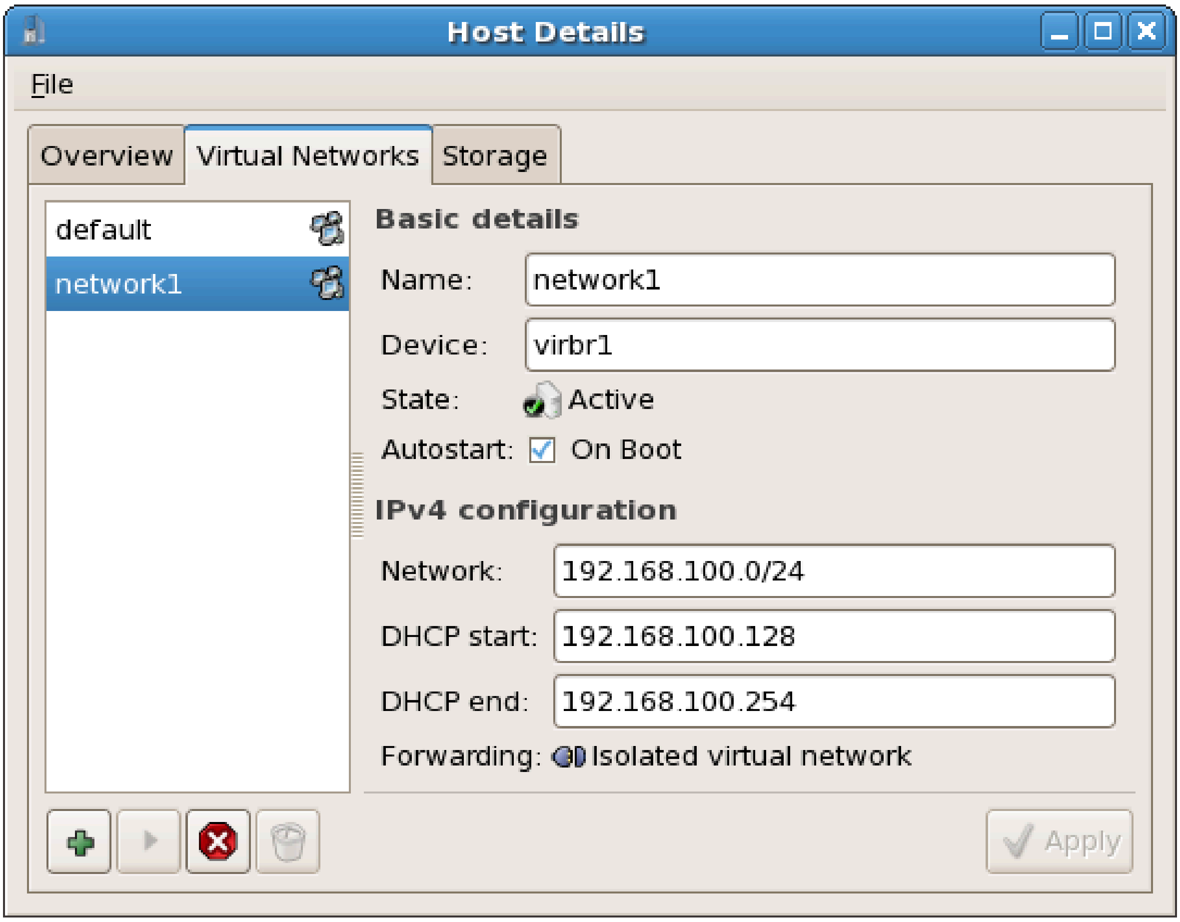


Figure 23.37. New virtual network is now available

The xm command quick reference

The **xm** command can manage the Xen hypervisor. Most operations can be performed with the libvirt tools, **virt-manager** application or the **virsh** command. The **xm** command does not have the error checking capacity of the libvirt tools and should not be used for tasks the libvirt tools support.

There are a few operations which currently can not be performed using **virt-manager**. Some options for other Xen implementations of the **xm** command do not work in Red Hat Enterprise Linux 5. The list below provides an overview of command options available and unavailable.

> **Warning**
> It is advised to use **virsh** or **virt-manager** instead of **xm**. The **xm** command does not handle error checking or configuration file errors very well and mistakes can lead to system instability or errors in virtual machines. Editing Xen configuration files manually is dangerous and should be avoided. Use this chapter at your own risk.

Basic management options

The following are basic and commonly used **xm** commands:

- **xm help [--long]**: view available options and help text.
- use the **xm list** command to list active domains:

```
$ xm list
Name                                ID  Mem(MiB)   VCPUs      State
 Time(s)
Domain-0                            0     520        2        r-----
 1275.5
r5b2-mySQL01                       13     500        1        -b----
  16.1
```

- **xm create [-c]** ***DomainName/ID***: start a virtual machine. If the -c option is used, the start up process will attach to the guest's console.
- **xm console** ***DomainName/ID***: attach to a virtual machine's console.
- **xm destroy** ***DomainName/ID***: terminates a virtual machine , similar to a power off.
- **xm reboot** ***DomainName/ID***: reboot a virtual machine, runs through the normal system shut down and start up process.
- **xm shutdown** ***DomainName/ID***: shut down a virtual machine, runs a normal system shut down procedure.
- **xm pause**
- **xm unpause**
- **xm save**
- **xm restore**

* **`xm migrate`**

Resource management options

Use the following **`xm`** commands to manage resources:

* **`xm mem-set`**
* use the **`xm vcpu-list`** to list virtualized CPU affinities:

```
$ xm vcpu-list
Name                          ID  VCPUs   CPU State  Time(s)  CPU
 Affinity
Domain-0                       0    0      0    r--    708.9    any cpu
Domain-0                       0    1      1   -b-     572.1    any cpu
r5b2-mySQL01                  13    0      1   -b-      16.1    any cpu
```

* **`xm vcpu-pin`**
* **`xm vcpu-set`**
* use the **`xm sched-credit`** command to display scheduler parameters for a given domain:

```
$ xm sched-credit -d 0
{'cap': 0, 'weight': 256}
$ xm sched-credit -d 13
{'cap': 25, 'weight': 256}
```

Monitoring and troubleshooting options

Use the following **`xm`** commands for monitoring and troubleshooting:

* **`xm top`**
* **`xm dmesg`**
* **`xm info`**
* **`xm log`**
* use the **`xm uptime`** to display the uptime of guests and hosts:

```
$ xm uptime
Name                       ID  Uptime
Domain-0                    0  3:42:18
r5b2-mySQL01               13  0:06:27
```

* **`xm sysrq`**
* **`xm dump-core`**
* **`xm rename`**

- **`xm domid`**
- **`xm domname`**

Currently unsupported options

The **`xm vnet-list`** is currently unsupported.

Configuring the Xen kernel boot parameters

The GNU Grand Unified Boot Loader (GRUB) is a program for booting various installed operating systems or kernels. GRUB also allows the user to pass arguments to the kernel. The GRUB configuration file (located in **/boot/grub/grub.conf**) creates the list of operating systems the GRUB boot menu interface. When you install the **kernel-xen** RPM, a script adds the **kernel-xen** entry to the GRUB configuration file which boots **kernel-xen** by default. Edit the **grub.conf** file to modify the default kernel or to add additional kernel parameters.

```
title Red Hat Enterprise Linux Server (2.6.18-3.el5xen)
root (hd0,0)
 kernel  /xen.gz.-2.6.18-3.el5
 module  /vmlinuz-2.6..18-3.el5xen ro root=/dev/VolGroup00/LogVol00  rhgb
 quiet
 module  /initrd-2.6.18-3. el5xenxen.img
```

If you set your Linux grub entries to reflect this example, the boot loader loads the hypervisor, **initrd** image, and Linux kernel. Since the kernel entry is on top of the other entries, the kernel loads into memory first. The boot loader sends, and receives, command line arguments to and from the hypervisor and Linux kernel. This example entry shows how you would restrict the Dom0 linux kernel memory to 800 MB.

```
title Red Hat Enterprise Linux Server (2.6.18-3.el5xen)
root (hd0,0)
 kernel  /xen.gz.-2.6.18-3.el5 dom0_mem=800M
 module  /vmlinuz-2.6..18-3.el5xen ro root=/dev/VolGroup00/LogVol00  rhgb
 quiet
 module  /initrd-2.6.18-3. el5xenxen.img
```

You can use these GRUB parameters to configure the Virtualization hypervisor:

```
mem
```

This limits the amount of memory that is available to the hypervisor kernel.

```
com1=115200, 8n1
```

This enables the first serial port in the system to act as serial console (com2 is assigned for the next port, and so on).

```
dom0_mem
```

This limits the memory available for the hypervisor.

```
dom0_max_vcpus
```

This limits the amount of CPUs visible to the Xen domain0.

```
acpi
```

This switches the ACPI hypervisor to the hypervisor and domain0. The ACPI parameter options include:

```
/*   ****  Linux config options: propagated to domain0  ****/
/*   "acpi=off":      Disables both ACPI table parsing and interpreter.
 */
/*   "acpi=force":    Overrides the disable blacklist.
 */
/*   "acpi=strict":   Disables out-of-spec workarounds.
 */
/*   "acpi=ht":       Limits ACPI from boot-time to enable HT.
 */
/*   "acpi=noirq":    Disables ACPI interrupt routing.
 */
```

```
noacpi
```

This disables ACPI for interrupt delivery.

Configuring ELILO

ELILO is the boot loader used on EFI-based systems, notably Itanium®. Similar to the GRUB, the boot loader on x86 and x86-64 systems, ELILO allows the user to select which installed kernel to load during the system boot sequence. ELILO also allows the user to pass arguments to the kernel. The ELILO configuration file, which is located in the EFI boot partition and symbolically linked to **`/etc/elilo.conf`**, contains a list of global options and image stanzas. When you install the **`kernel-xen`** RPM, a post install script adds the appropriate image stanza to the **`elilo.conf`**.

> **ELILO**
>
> This section on ELILO is for systems running the Xen kernel on the intel Itanium architecture.

The ELILO configuration file has two sections:

- Global options that affect the behavior of ELILO and all the entries. Typically there is no need to change these from the default values.
- Image stanzas that define a boot selection along with associated options.

Here is a sample image stanza in elilo.conf:

```
image=vmlinuz-2.6.18-92.el5xen
        vmm=xen.gz-2.6.18-92.el5
        label=linux
        initrd=initrd-2.6.18-92.el5xen.img
        read-only
        root=/dev/VolGroup00/rhel5_2
        append="-- rhgb quiet"
```

The *`image`* parameter indicates the following lines apply to a single boot selection. This stanza defines a hypervisor (*vmm*), **`initrd`**, and command line arguments (*`read-only`*, *`root`* and *append*) to the hypervisor and kernel. When ELILO is loaded during the boot sequence, the image is labeled label *`linux`*.

ELILO translates *`read-only`* to the kernel command line option *`ro`* which causes the root file system to be mounted read-only until the **`initscripts`** mount the root drive as read-write. ELILO copies the "*`root`*" line to the kernel command line. These are merged with the "*append*" line to build a complete command line:

```
"-- root=/dev/VolGroup00/rhel5_2 ro rhgb quiet"
```

The **`--`** symbols delimit hypervisor and kernel arguments. The hypervisor arguments come first, then the **`--`** delimiter, followed by the kernel arguments. The hypervisor does not usually have any arguments.

Technical note

ELILO passes the entire command line to the hypervisor. The hypervisor divides the content and passes the kernel options to the kernel.

To customize the hypervisor, insert parameters before the **--**. An example of the hypervisor memory (*mem*) parameter and the *quiet* parameter for the kernel:

```
append="dom0_mem=2G -- quiet"
```

ELILO hypervisor parameters

Parameter	Description
mem=	The *mem* parameter defines the hypervisor maximum RAM usage. Any additional RAM in the system is ignored. The parameter may be specified with a B, K, M or G suffix; representing bytes, kilobytes, megabytes and gigabytes respectively. If no suffix is specified the default unit is kilobytes.
dom0_mem=	*dom0_mem=* sets the amount of RAM to allocate to dom0. The same suffixes are respected as for the mem parameter above. The default in Red Hat Enterprise Linux 5.2 on Itanium® is 4G.
dom0_max_vcpus=	*dom0_max_vcpus=* sets the number of CPUs to allocate to the hypervisor. The default in Red Hat Enterprise Linux 5.2 on Itanium® is 4.
com1=<*baud*>,DPS,<*io_base*>,<*irq*>	*com1* sets the parameters for the first serial line. For example, **com1=9600,8n1,0x408,5**. The *io_base* and *irq* options can be omitted to leave them as the standard defaults. The *baud* parameter can be set as *auto* to indicate the boot loader setting should be preserved. The *com1* parameter can be omitted if serial parameters are set as global options in ELILO or in the EFI configuration.
com2=<*baud*>,DPS,<*io_base*>,<*irq*>	Set the parameters for the second serial line. Refer the description of the *com1* parameter above.
console=<*specifier_list*>	The *console* is a comma delimited preference list for the console options. Options include vga, com1 and com2. This setting should be omitted because the hypervisor attempts to inherit EFI console settings.

For more information on ELILO parameters

A complete list of ELILO parameters are available from *XenSource*[1].

A modified example of the configuration above, showing syntax for appending memory and cpu allocation parameters to the hypervisor:

```
image=vmlinuz-2.6.18-92.el5xen
        vmm=xen.gz-2.6.18-92.el5
```

```
        label=linux
        initrd=initrd-2.6.18-92.el5xen.img
        read-only
        root=/dev/VolGroup00/rhel5_2
        append="dom0_mem=2G dom0_max_vcpus=2 --"
```

Additionally this example removes the kernel parameters "*`rhgb quiet`*" so that kernel and **`initscript`** output are generated on the console. Note the double-dash remains so that the append line is correctly interpreted as hypervisor arguments.

Xen configuration files

Red Hat Enterprise Linux uses **libvirt** configuration files for most tasks. Some users may need Xen configuration files which contain the following standard variables. Configuration items within these files must be enclosed in single quotes('). These configuration files reside in the **/etc/xen** directory.

Item	Description
pae	Specifies the physical address extension configuration data.
apic	Specifies the advanced programmable interrupt controller configuration data.
memory	Specifies the memory size in megabytes.
vcpus	Specifies the numbers of virtual CPUs.
console	Specifies the port numbers to export the domain consoles to.
nic	Specifies the number of virtual network interfaces.
vif	Lists the randomly-assigned MAC addresses and bridges assigned to use for the domain's network addresses.
disk	Lists the block devices to export to the domain and exports physical devices to domain with read only access.
dhcp	Enables networking using DHCP.
netmask	Specifies the configured IP netmasks.
gateway	Specifies the configured IP gateways.
acpi	Specifies the advanced configuration power interface configuration data.

Table 27.1. libvirt configuration files

The table below, *Table 27.2, "Xen configuration file reference"*, is formatted output from **xm create --help_config**.

Parameter	Description
vncpasswd=NAME	Password for VNC console on HVM domain.
vncviewer=**no** | **yes**	Spawn a vncviewer listening for a vnc server in the domain. The address of the vncviewer is passed to the domain on the kernel command line using *VNC_SERVER=<host>:<port>*. The port used by vnc is 5500 + DISPLAY. A display value with a free port is chosen if possible. Only valid when vnc=1.
vncconsole=**no** | **yes**	Spawn a vncviewer process for the domain's graphical console. Only valid when vnc=1.
name=NAME	Domain name. Must be unique.
bootloader=FILE	Path to boot loader.

Parameter	Description
bootargs=NAME	Arguments to pass to boot loader.
bootentry=NAME	DEPRECATED. Entry to boot via boot loader. Use *bootargs*.
kernel=FILE	Path to kernel image.
ramdisk=FILE	Path to ramdisk.
features=FEATURES	Features to enable in guest kernel
builder=FUNCTION	Function to use to build the domain.
memory=MEMORY	Domain memory in MB.
maxmem=MEMORY	Maximum domain memory in MB.
shadow_memory=MEMORY	Domain shadow memory in MB.
cpu=CPU	CPU which hosts VCPU0.
cpus=CPUS	CPUS to run the domain on.
pae=PAE	Disable or enable PAE of HVM domain.
acpi=ACPI	Disable or enable ACPI of HVM domain.
apic=APIC	Disable or enable APIC of HVM domain.
vcpus=VCPUs	The number of Virtual CPUS in domain.
cpu_weight=WEIGHT	Set the new domain's cpu weight. *WEIGHT* is a float that controls the domain's share of the cpu.
restart=onreboot | always | never	Deprecated. Use *on_poweroff, on_reboot*, and *on_crash* instead. Whether the domain should be restarted on exit. - *onreboot*: restart on exit with shutdown code reboot - always: always restart on exit, ignore exit code - never: never restart on exit, ignore exit code
on_poweroff=destroy | restart | preserve | destroy	Behavior when a domain exits with reason 'poweroff'. - destroy: the domain is cleaned up as normal; - restart: a new domain is started in place of the old one; - preserve: no clean-up is done until the domain is manually destroyed (using **xm** destroy, for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
on_reboot=destroy | restart | preserve | destroy	Behavior when a domain exits with reason 'reboot'. - destroy: the domain is cleaned up as normal; - restart: a new domain is started in place of the old one; - preserve: no clean-up is done until the domain is manually destroyed (using xm destroy, for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
on_crash=destroy | restart | preserve | destroy	Behavior when a domain exits with reason 'crash'. - destroy: the domain is cleaned up as normal; - restart: a new domain is started in place of the old one; - preserve: no clean-up

Parameter	Description
	is done until the domain is manually destroyed (using xm destroy, for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
blkif=**no | yes**	Make the domain a block device backend.
netif=**no | yes**	Make the domain a network interface backend.
tpmif=**no | yes**	Make the domain a TPM interface backend.
disk=phy:DEV,VDEV,MODE[,DOM]	Add a disk device to a domain. The physical device is *DEV*, which is exported to the domain as *VDEV*. The disk is read-only if *MODE* is **r**, read-write if *MODE* is **w**. If *DOM* is specified it defines the backend driver domain to use for the disk. The option may be repeated to add more than one disk.
pci=BUS:DEV.FUNC	Add a PCI device to a domain, using given parameters (in hex). For example *pci=c0:02.1a*. The option may be repeated to add more than one pci device.
ioports=FROM[-TO]	Add a legacy I/O range to a domain, using given params (in hex). For example *ioports=02f8-02ff*. The option may be repeated to add more than one i/o range.
irq=IRQ	Add an IRQ (interrupt line) to a domain. For example *irq=7*. This option may be repeated to add more than one IRQ.
usbport=PATH	Add a physical USB port to a domain, as specified by the path to that port. This option may be repeated to add more than one port.
vfb=type={vnc,sdl}, vncunused=1, vncdisplay=N, *vnclisten=ADDR, display=DISPLAY,* *xauthority=XAUTHORITY, vncpasswd=PASSWORD,* *keymap=KEYMAP*	Make the domain a framebuffer backend. The backend type should be either *sdl* or *vnc*. For *type=vnc*, connect an external vncviewer. The server will listen on *ADDR* (default 127.0.0.1) on port N+5900. *N* defaults to the domain id. If *vncunused=1*, the server will try to find an arbitrary unused port above 5900. For *type=sdl*, a viewer will be started automatically using the given *DISPLAY* and *XAUTHORITY*, which default to the current user's ones.
vif=type=TYPE, mac=MAC, bridge=BRIDGE, ip=IPADDR, *script=SCRIPT, backend=DOM, vifname=NAME*	Add a network interface with the given *MAC* address and bridge. The *vif* is configured by calling the given configuration script. If type is not specified, default is netfront not ioemu device. If mac is not specified a random *MAC* address is used. If not specified then the network backend chooses it's own *MAC* address. If bridge is not specified the first bridge found is used. If script is not specified the default script is used.

Parameter	Description
	If backend is not specified the default backend driver domain is used. If vifname is not specified the backend virtual interface will have name vifD.N where D is the domain id and N is the interface id. This option may be repeated to add more than one vif. Specifying vifs will increase the number of interfaces as needed.
vtpm=instance=INSTANCE,backend=*DOM*	Add a TPM interface. On the backend side use the given instance as virtual TPM instance. The given number is merely the preferred instance number. The hotplug script will determine which instance number will actually be assigned to the domain. The association between virtual machine and the TPM instance number can be found in **/etc/xen/vtpm.db**. Use the backend in the given domain.
access_control=policy=POLICY,label=*LABEL*	Add a security label and the security policy reference that defines it. The local ssid reference is calculated when starting or resuming the domain. At this time, the policy is checked against the active policy as well. This way, migrating through the save or restore functions are covered and local labels are automatically created correctly on the system where a domain is started or resumed.
nics=NUM	DEPRECATED. Use empty vif entries instead. Set the number of network interfaces. Use the vif option to define interface parameters, otherwise defaults are used. Specifying vifs will increase the number of interfaces as needed.
root=DEVICE	Set the *root=* parameter on the kernel command line. Use a device, e.g. **/dev/sda1**, or **/dev/nfs** for NFS root.
extra=ARGS	Set extra arguments to append to the kernel command line.
ip=IPADDR	Set the kernel IP interface address.
gateway=IPADDR	Set the kernel IP gateway.
netmask=MASK	Set the kernel IP netmask.
hostname=NAME	Set the kernel IP hostname.
interface=INTF	Set the kernel IP interface name.
dhcp=off|dhcp	Set the kernel dhcp option.
nfs_server=IPADDR	Set the address of the NFS server for NFS root.
nfs_root=PATH	Set the path of the root NFS directory.
device_model=FILE	Path to device model program.
fda=FILE	Path to fda

Parameter	Description
fdb=FILE	Path to fdb
serial=FILE	Path to serial or pty or vc
localtime=**no | yes**	Is RTC set to localtime
keymap=FILE	Set keyboard layout used
usb=**no | yes**	Emulate USB devices
usbdevice=NAME	Name of a USB device to add
stdvga=**no | yes**	Use *std vga* or Cirrus Logic graphics
isa=**no | yes**	Simulate an ISA only system
boot=a|b|c|d	Default boot device
nographic=**no | yes**	Should device models use graphics?
soundhw=audiodev	Should device models enable audio device?
vnc	Should the device model use VNC?
vncdisplay	VNC display to use
vnclisten	Address for VNC server to listen on.
vncunused	Try to find an unused port for the VNC server. Only valid when vnc=1.
sdl	Should the device model use SDL?
display=DISPLAY	X11 display to use
xauthority=XAUTHORITY	X11 Authority to use
uuid	xenstore UUID (universally unique identifier) to use. One will be randomly generated if this option is not set, just like MAC addresses for virtual network interfaces. This must be a unique value across the entire cluster.

Table 27.2. Xen configuration file reference

Table 27.4, “Configuration parameter default values” lists all configuration parameters available, the Python parser function which sets the value and default values. The setter function gives an idea of what the parser does with the values you specify. It reads them as Python values, then feeds them to a setter function to store them. If the value is not valid Python, you get an obscure error message. If the setter rejects your value, you should get a reasonable error message, except it appears to get lost somehow, along with your bogus setting. If the setter accepts, but the value makes no sense, the program proceeds, and you can expect it to fall flat on its face somewhere down the road.

Parser function	Valid arguments
set_bool	Accepted values: • **yes** • **y** • **no** • **yes**

Parser function	Valid arguments
set_float	Accepts a floating point number with Python's float(). For example: • **3.14** • **10.** • **.001** • **1e100** • **3.14e-10**
set_int	Accepts an integer with Python's int().
set_value	accepts any Python value.
append_value	accepts any Python value, and appends it to the previous value which is stored in an array.

Table 27.3. Python functions which set parameter values

Parameter	Parser function	Default value
name	*setter*	*default value*
vncpasswd	*set_value*	*None*
vncviewer	*set_bool*	*None*
vncconsole	*set_bool*	*None*
name	*set_value*	*None*
bootloader	*set_value*	*None*
bootargs	*set_value*	*None*
bootentry	*set_value*	*None*
kernel	*set_value*	*None*
ramdisk	*set_value*	*''*
features	*set_value*	*''*
builder	*set_value*	*'linux'*
memory	*set_int*	*128*
maxmem	*set_int*	*None*
shadow_memory	*set_int*	*0*
cpu	*set_int*	*None*
cpus	*set_value*	*None*
pae	*set_int*	*0*
acpi	*set_int*	*0*
apic	*set_int*	*0*
vcpus	*set_int*	*1*
cpu_weight	*set_float*	*None*
restart	*set_value*	*None*

Parameter	Parser function	Default value
on_poweroff	*set_value*	*None*
on_reboot	*set_value*	*None*
on_crash	*set_value*	*None*
blkif	*set_bool*	*0*
netif	*set_bool*	*0*
tpmif	*append_value*	*0*
disk	*append_value*	*[]*
pci	*append_value*	*[]*
ioports	*append_value*	*[]*
irq	*append_value*	*[]*
usbport	*append_value*	*[]*
vfb	*append_value*	*[]*
vif	*append_value*	*[]*
vtpm	*append_value*	*[]*
access_control	*append_value*	*[]*
nics	*set_int*	*-1*
root	*set_value*	*''*
extra	*set_value*	*''*
ip	*set_value*	*''*
gateway	*set_value*	*''*
netmask	*set_value*	*''*
hostname	*set_value*	*''*
interface	*set_value*	*"eth0"*
dhcp	*set_value*	*'off'*
nfs_server	*set_value*	*None*
nfs_root	*set_value*	*None*
device_model	*set_value*	*''*
fda	*set_value*	*''*
fdb	*set_value*	*''*
serial	*set_value*	*''*
localtime	*set_bool*	*0*
keymap	*set_value*	*''*
usb	*set_bool*	*0*
usbdevice	*set_value*	*''*
stdvga	*set_bool*	*0*
isa	*set_bool*	*0*
boot	*set_value*	*'c'*
nographic	*set_bool*	*0*

Parameter	Parser function	Default value
soundhw	*set_value*	*''*
vnc	*set_value*	*None*
vncdisplay	*set_value*	*None*
vnclisten	*set_value*	*None*
vncunused	*set_bool*	*1*
sdl	*set_value*	*None*
display	*set_value*	*None*
xauthority	*set_value*	*None*
uuid	*set_value*	*None*

Table 27.4. Configuration parameter default values

Part VI. Tips and Tricks

Tips and Tricks to Enhance Productivity

These chapters contain useful hints and tips to improve virtualization performance, scale and stability.

Tips and tricks

This chapter contain useful hints and tips to improve virtualization performance, scale and stability.

28.1. Automatically starting guests

This section covers how to make virtualized guests start automatically during the host system's boot phase.

This example uses **virsh** to set a guest, *TestServer*, to automatically start when the host boots.

```
# virsh autostart TestServer
Domain TestServer marked as autostarted
```

The guest now automatically starts with the host.

To stop a guest automatically booting use the `--disable` parameter

```
# virsh autostart --disable TestServer
Domain TestServer unmarked as autostarted
```

The guest no longer automatically starts with the host.

28.2. Changing between the KVM and Xen hypervisors

This section covers changing between the KVM and Xen hypervisors.

Red Hat only supports one active hypervisor at a time.

Migrating virtualized guests between hypervisors

Presently, there is no application for switching Xen-based guests to KVM or KVM-based guests to Xen. Guests can only be used on the hypervisor type that they were created on.

Warning

This procedure is only available for the Intel 64 or AMD64 version of Red Hat Enterprise Linux 5.4 or newer. No other configurations or Red Hat Enterprise Linux versions are supported. KVM is not available in versions earlier than Red Hat Enterprise Linux 5.4.

28.2.1. Xen to KVM

The following procedure covers changing from the Xen hypervisor to the KVM hypervisor. This procedure assumes the *kernel-xen* package is installed and enabled.

1. **Install the KVM package**
   Install the *kvm* package if you have not already done so.

   ```
   # yum install kvm
   ```

2. **Verify which kernel is in use**
   The *kernel-xen* package may be installed. Use the **uname** command to determine which kernel is running:

```
$ uname -r
2.6.18-159.el5xen
```

The present kernel, "**2.6.18-159.el5xen**", is running on the system. If the default kernel, "**2.6.18-159.el5**", is running you can skip the substep.

- **Changing the Xen kernel to the default kernel**
  The **grub.conf** file determines which kernel is booted. To change the default kernel edit the **/boot/grub/grub.conf** file as shown below.

```
default=1
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/
LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro root=/dev/VolGroup00/
LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

Notice the **default=1** parameter. This is instructing the GRUB boot loader to boot the second entry, the Xen kernel. Change the default to *0* (or the number for the default kernel):

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/
LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro root=/dev/VolGroup00/
LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

3. **Reboot to load the new kernel**
   Reboot the system. The computer will restart with the default kernel. The KVM module should be automatically loaded with the kernel. Verify KVM is running:

```
$ lsmod | grep kvm
kvm_intel              85992  1
kvm                   222368  2 ksm,kvm_intel
```

   The **kvm** module and either the **kvm_intel** module or the **kvm_amd** module are present if everything worked.

28.2.2. KVM to Xen

The following procedure covers changing from the KVM hypervisor to the Xen hypervisor. This procedure assumes the *kvm* package is installed and enabled.

1. **Install the Xen packages**
   Install the *kernel-xen* and *xen* package if you have not already done so.

```
# yum install kernel-xen xen
```

   The *kernel-xen* package may be installed but disabled.

2. **Verify which kernel is in use**
   Use the **uname** command to determine which kernel is running.

   ```
   $ uname -r
   2.6.18-159.el5
   ```

   The present kernel, "**2.6.18-159.el5**", is running on the system. This is the default kernel. If the kernel has **xen** on the end (for example, **2.6.18-159.el5xen**) then the Xen kernel is running and you can skip the substep.

   - **Changing the default kernel to the Xen kernel**
     The **grub.conf** file determines which kernel is booted. To change the default kernel edit the **/boot/grub/grub.conf** file as shown below.

     ```
     default=0
     timeout=5
     splashimage=(hd0,0)/grub/splash.xpm.gz
     hiddenmenu
     title Red Hat Enterprise Linux Server (2.6.18-159.el5)
             root (hd0,0)
             kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/
     LogVol00 rhgb quiet
             initrd /initrd-2.6.18-159.el5.img
     title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
             root (hd0,0)
             kernel /xen.gz-2.6.18-159.el5
             module /vmlinuz-2.6.18-159.el5xen ro root=/dev/VolGroup00/
     LogVol00 rhgb quiet
             module /initrd-2.6.18-159.el5xen.img
     ```

     Notice the **default=0** parameter. This is instructing the GRUB boot loader to boot the first entry, the default kernel. Change the default to *1* (or the number for the Xen kernel):

     ```
     default=1
     timeout=5
     splashimage=(hd0,0)/grub/splash.xpm.gz
     hiddenmenu
     title Red Hat Enterprise Linux Server (2.6.18-159.el5)
             root (hd0,0)
             kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/
     LogVol00 rhgb quiet
             initrd /initrd-2.6.18-159.el5.img
     title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
             root (hd0,0)
             kernel /xen.gz-2.6.18-159.el5
             module /vmlinuz-2.6.18-159.el5xen ro root=/dev/VolGroup00/
     LogVol00 rhgb quiet
             module /initrd-2.6.18-159.el5xen.img
     ```

3. **Reboot to load the new kernel**
   Reboot the system. The computer will restart with the Xen kernel. Verify with the **uname** command:

   ```
   $ uname -r
   2.6.18-159.el5xen
   ```

   If the output has `xen` on the end the Xen kernel is running.

28.3. Using qemu-img

The **qemu-img** command line tool is used for formatting various file systems used by Xen and KVM. **qemu-img** should be used for formatting virtualized guest images, additional storage devices and network storage. **qemu-img** options and usages are listed below.

Formatting and creating new images or devices

Create the new disk image filename of size size and format format.

```
# qemu-img create [-6] [-e] [-b base_image] [-f format] filename [size]
```

If base_image is specified, then the image will record only the differences from base_image. No size needs to be specified in this case. base_image will never be modified unless you use the "commit" monitor command.

Convert an existing image to another format

The convert option is used for converting a recognized format to another image format.

Command format:

```
# qemu-img convert [-c] [-e] [-f format] filename [-O output_format]
 output_filename
```

convert the disk image filename to disk image output_filename using format output_format. it can be optionally encrypted ("-e" option) or compressed ("-c" option).

only the format "qcow" supports encryption or compression. the compression is read-only. it means that if a compressed sector is rewritten, then it is rewritten as uncompressed data.

The encryption uses the AES format with very secure 128 bit keys. use a long password (16 characters) to get maximum protection.

image conversion is also useful to get smaller image when using a format which can grow, such as **qcow** or **cow**. The empty sectors are detected and suppressed from the destination image.

getting image information

the **info** parameter displays information about a disk image. the format for the **info** option is as follows:

```
# qemu-img info [-f format] filename
```

give information about the disk image filename. use it in particular to know the size reserved on disk which can be different from the displayed size. if vm snapshots are stored in the disk image, they are displayed too.

Supported formats

The format of an image is usually guessed automatically. The following formats are supported:

`raw`
: Raw disk image format (default). This format has the advantage of being simple and easily exportable to all other emulators. If your file system supports holes (for example in ext2 or ext3 on Linux or NTFS on Windows), then only the written sectors will reserve space. Use **`qemu-img info`** to know the real size used by the image or **`ls -ls`** on Unix/Linux.

`qcow2`
: QEMU image format, the most versatile format. Use it to have smaller images (useful if your file system does not supports holes, for example: on Windows), optional AES encryption, zlib based compression and support of multiple VM snapshots.

`qcow`
: Old QEMU image format. Only included for compatibility with older versions.

`cow`
: User Mode Linux Copy On Write image format. The **`cow`** format is included only for compatibility with previous versions. It does not work with Windows.

`vmdk`
: VMware 3 and 4 compatible image format.

`cloop`
: Linux Compressed Loop image, useful only to reuse directly compressed CD-ROM images present for example in the Knoppix CD-ROMs.

28.4. Overcommitting with KVM

The KVM hypervisor supports overcommitting CPUs and overcommitting memory. Overcommitting is allocating more virtualized CPUs or memory than there are physical resources on the system. With CPU overcommit, under-utilized virtualized servers or desktops can run on fewer servers which saves power and money.

> **Xen support**
>
> CPU overcommitting is not supported for the Xen hypervisor. Overcommitting CPUs with the Xen hypervisor may cause system instability and crashes of the host and virtualized guests.

Overcommitting memory

Most operating systems and applications do not use 100% of the available RAM all the time. This behavior can be exploited with KVM to use more memory for virtualized guests than what is physically available.

With KVM, virtual machines are Linux processes. Guests on the KVM hypervisor do not have blocks of physical RAM assigned to them instead they function as processes. Each process is allocated memory when it requests more memory. KVM uses this to allocate memory for guests when the guest operating system requests more or less memory. The guest only uses slightly more physical memory than the virtualized operating system appears to use.

When physical memory is nearly completely used or a process is inactive for some time, Linux moves the process's memory to swap. Swap is usually a partition on a hard disk drive or solid state drive which Linux uses to extend virtual memory. Swap is significantly slower than RAM.

As KVM virtual machines are Linux processes, memory used by virtualized guests can be put into swap if the guest is idle or not in heavy use. Memory can be committed over the total size of the swap and physical RAM. This can cause issues if virtualized guests use their total RAM. Without sufficient swap space for the virtual machine processes to be swapped to the **`pdflush`** process starts. **`pdflush`** kills processes to free memory so the system does not crash. **`pdflush`** may destroy virtualized guests or other system processes which may cause file system errors and may leave virtualized guests unbootable.

> **Warning**
> If sufficient swap is not available guest operating systems will be forcibly shut down. This may leave guests inoperable. Avoid this by never overcommitting more memory than there is swap available.

The swap partition is used for swapping underused memory to the hard drive to speed up memory performance. The default size of the swap partition is calculated from amount of RAM and overcommit ratio. It is recommended to make your swap partition larger if you intend to overcommit memory with KVM. A recommended overcommit ratio is 50% (0.5). The formula used is:

```
(0.5 * RAM) + (overcommit ratio * RAM) = Recommended swap size
```

Red Hat *Knowledgebase*[1] has an article on safely and efficiently determining the size of the swap partition.

It is possible to run with an overcommit ratio of ten times the number of virtualized guests over the amount of physical RAM in the system. This only works with certain application loads (for example desktop virtualization with under 100% usage). Setting overcommit ratios is not a hard formula, you must test and customize the ratio for your environment.

Overcommitting virtualized CPUs

The KVM hypervisor supports overcommitting virtualized CPUs. Virtualized CPUs can be overcommitted as far as load limits of virtualized guests allow. Use caution when overcommitting VCPUs as loads near 100% may cause dropped requests or unusable response times.

Virtualized CPUs are overcommitted best when each virtualized guest only has a single VCPU. The Linux scheduler is very efficient with this type of load. KVM should safely support guests with loads under 100% at a ratio of 5 VCPUs Overcommitting single VCPU virtualized guests is not an issue.

You cannot overcommit symmetric multiprocessing guests on more than the physical number of processing cores. For example a guest with four VCPUs should not be run on a host with a dual

[1] http://kbase.redhat.com/faq/docs/DOC-15252

core processor. Overcommitting symmetric multiprocessing guests in over the physical number of processing cores will cause significant performance degradation.

Assigning guests VCPUs up to the number of physical cores is appropriate and works as expected. For example, running virtualized guests with four VCPUs on a quad core host. Guests with less than 100% loads should function effectively in this setup.

> **Always test first**
> Do not overcommit memory or CPUs in a production environment without extensive testing. Applications which use 100% of memory or processing resources may become unstable in overcommitted environments. Test before deploying.

28.5. Modifying /etc/grub.conf

This section describes how to safely and correctly change your **`/etc/grub.conf`** file to use the virtualization kernel. You must use the `xen` kernel to use the Xen hypervisor. Copy your existing `xen` kernel entry make sure you copy all of the important lines or your system will panic upon boot (**`initrd`** will have a length of '**`0`**'). If you require `xen` hypervisor specific values you must append them to the `xen` line of your grub entry.

The output below is an example of a **`grub.conf`** entry from a system running the *kernel-xen* package. The **`grub.conf`** on your system may vary. The important part in the example below is the section from the **`title`** line to the next new line.

```
#boot=/dev/sda
default=0
timeout=15
#splashimage=(hd0,0)/grub/splash.xpm.gz hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console

title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21.el5xen)
 root (hd0,0)
 kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1
 module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro root=/dev/VolGroup00/LogVol00
 module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

> **An important point regarding editing `grub.conf`...**
> Your **`grub.conf`** could look very different if it has been manually edited before or copied from an example. Read *Chapter 25, Configuring the Xen kernel boot parameters* for more information on using virtualization and grub.

To set the amount of memory assigned to your host system at boot time to 256MB you need to append `dom0_mem=256M` to the `xen` line in your **`grub.conf`**. A modified version of the grub configuration file in the previous example:

```
#boot=/dev/sda
default=0
```

```
timeout=15
#splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console

title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21.el5xen)
 root (hd0,0)
 kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1 dom0_mem=256MB
 module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro
 root=/dev/VolGroup00/LogVol00
 module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

28.6. Verifying virtualization extensions

Use this section to determine whether your system has the hardware virtualization extensions. Virtualization extensions (Intel VT or AMD-V) are required for full virtualization.

> **Can I use virtualization without the virtualization extensions?**
> If hardware virtualization extensions are not present you can use Xen para-virtualization with the Red Hat *kernel-xen* package.

Run the following command to verify the CPU virtualization extensions are available:

```
$ grep -E 'svm|vmx' /proc/cpuinfo
```

The following output contains a `vmx` entry indicating an Intel processor with the Intel VT extensions:

```
flags   : fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
 dts acpi mmx fxsr sse sse2 ss ht  tm syscall lm constant_tsc pni monitor
 ds_cpl
 vmx est tm2 cx16 xtpr lahf_lm
```

The following output contains an `svm` entry indicating an AMD processor with the AMD-V extensions:

```
flags   :  fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
 mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt lm 3dnowext 3dnow pni cx16
 lahf_lm cmp_legacy svm cr8legacy ts fid vid ttp tm stc
```

The "`flags:`" content may appear multiple times for each hyperthread, core or CPU on in the system.

The virtualization extensions may be disabled in the BIOS. If the extensions do not appear or full virtualization does not work refer to *Enabling virtualization extensions in BIOS*.

28.7. Identifying guest type and implementation

The script below can identify if the environment an application or script is running in is a para-virtualized, a fully virtualized guest or on the hypervisor.

```
#!/bin/bash
declare -i IS_HVM=0
declare -i IS_PARA=0
check_hvm()
{
 IS_X86HVM="$(strings /proc/acpi/dsdt | grep int-xen)"
   if [ x"${IS_X86HVM}" != x ]; then
    echo "Guest type is full-virt x86hvm"
    IS_HVM=1
 fi
}
check_para()
{
 if $(grep -q control_d /proc/xen/capabilities); then
   echo "Host is dom0"
   IS_PARA=1
 else
   echo "Guest is para-virt domU"
   IS_PARA=1
 fi
}
if [ -f /proc/acpi/dsdt ]; then
 check_hvm
fi

if [ ${IS_HVM} -eq 0 ]; then
 if [ -f /proc/xen/capabilities ] ; then
  check_para
 fi
     fi
if [ ${IS_HVM} -eq 0 -a ${IS_PARA} -eq 0 ]; then
 echo "Baremetal platform"
fi
```

Examining hosts

For examining hosts, use the **`virsh capabilites`** command.

28.8. Generating a new unique MAC address

In some case you will need to generate a new and unique *MAC address* for a guest. There is no command line tool available to generate a new MAC address at the time of writing. The script provided below can generate a new MAC address for your guests. Save the script to your guest as **`macgen.py`**. Now from that directory you can run the script using **`./macgen.py`** . and it will generate a new MAC address. A sample output would look like the following:

```
$ ./macgen.py
00:16:3e:20:b0:11
```

```
#!/usr/bin/python
# macgen.py script to generate a MAC address for virtualized guests on Xen
#
import random
#
def randomMAC():
 mac = [ 0x00, 0x16, 0x3e,
  random.randint(0x00, 0x7f),
  random.randint(0x00, 0xff),
  random.randint(0x00, 0xff) ]
 return ':'.join(map(lambda x: "%02x" % x, mac))
#
print randomMAC()
```

Another method to generate a new MAC for your guest

You can also use the built-in modules of **`python-virtinst`** to generate a new MAC address and **`UUID`** for use in a guest configuration file:

```
# echo  'import virtinst.util ; print\
 virtinst.util.uuidToString(virtinst.util.randomUUID())' | python
# echo  'import virtinst.util ; print virtinst.util.randomMAC()' | python
```

The script above can also be implemented as a script file as seen below.

```
#!/usr/bin/env python
#  -*- mode: python; -*-
print ""
print "New UUID:"
import virtinst.util ; print
 virtinst.util.uuidToString(virtinst.util.randomUUID())
print "New MAC:"
import virtinst.util ; print virtinst.util.randomMAC()
print ""
```

28.9. Limit network bandwidth for a Xen guest

In some environments it may be required to limit the network bandwidth available to certain guests. This can be used to implement basic Quality of Service on a host running multiple virtual machines. By default, the guest can use any bandwidth setting available which your physical network card supports. The physical network card must be mapped to one of virtual machine's virtual network interfaces. In Xen the “**`rate`**” parameter part of the **`VIF`** entries can throttle virtualized guests.

This list covers the variables

`rate`

The `rate=` option can be added to the `VIF=` entry in a virtual machine configuration file to limit a virtual machine's network bandwidth or specify a specific time interval for a time window.

time window
: The time window is optional to the `rate=` option:

  The default time window is 50ms.

  A smaller time window will provide less burst transmission, however, the replenishment rate and latency will increase.

  The default 50ms time window is a good balance between latency and throughput and in most cases will not require changing.

Examples of **`rate`** parameter values and uses.

`rate=10Mb/s`
: Limit the outgoing network traffic from the guest to 10MB/s.

`rate=250KB/s`
: Limit the outgoing network traffic from the guest to 250KB/s.

`rate=10MB/s@50ms`
: Limit bandwidth to 10MB/s and provide the guest with a 50KB chunk every 50ms.

In the virtual machine configuration a sample **`VIF`** entry would look like the following:

```
vif = [ 'rate=10MB/s , mac=00:16:3e:7a:55:1c, bridge=xenbr1']
```

This `rate` entry would limit the virtual machine's interface to 10MB/s for outgoing traffic

28.10. Configuring Xen processor affinities

Xen can allocate virtual CPUs to associate with one or more host CPUs. This allocates real processing resources to virtualized guests. This approach allows Red Hat Enterprise Linux optimize processor resources when employing dual-core, hyper-threading, or other CPU concurrency technologies. The Xen credit scheduler automatically balances virtual CPUs between physical ones, to maximize system use. Red Hat Enterprise Linux allows the credit scheduler to move CPUs around as necessary, as long as the virtual CPU is pinned to a physical CPU.

If you are running I/O intensive tasks, it is recommended to dedicate either a hyperthread or an entire processor core to run domain0.

Note that this is unnecessary for KVM as KVM uses the default Linux kernel scheduler.

CPU affinities can be set with **`virsh`** or **`virt-manager`**:

To set CPU affinities using **`virsh`** refer to *Configuring virtual CPU affinity* for more information.

To configure and view CPU information with **`virt-manager`** refer to *Section 23.11, “Displaying virtual CPUs ”* for more information.

28.11. Modifying the Xen hypervisor

Managing host systems often involves changing the boot configuration file **`/boot/grub/grub.conf`**. Managing several or more hosts configuration files quickly becomes difficult. System administrators often prefer to use the 'cut and paste' method for editing multiple **grub.conf** files. If you do this, ensure you include all five lines in the Virtualization entry (or this will create system errors). Hypervisor specific values are all found on the 'xen' line. This example represents a correct **grub.conf** virtualization entry:

```
# boot=/dev/sda/
default=0
timeout=15
#splashimage=(hd0, 0)/grub/splash.xpm.gz

hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21. el5xen)
root (hd0, 0)
kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1
module /vmlinuz-2.6.17-1.2519.4.21el5xen ro root=/dev/VolGroup00/LogVol00
module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

For example, to change the memory entry on your hypervisor (dom0) to 256MB at boot time, edit the 'xen' line and append it with this entry: **'dom0_mem=256M'**. This example a modified **grub.conf** with the hypervisor's memory entry modified.

```
# boot=/dev/sda
default=0
timeout=15
#splashimage=(hd0,0)/grubs/splash.xpm.gz
hiddenmenu
serial --unit=0 --speed =115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21. el5xen)
root (hd0,0)
kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1 dom0_mem=256MB
module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro
root=/dev/VolGroup00/LogVol00
module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

28.12. Very Secure ftpd

vsftpd can provide access to installation trees for para-virtualized guests (for example, the Red Hat Enterprise Linux 5 repositories) or other data. If you have not installed vsftpd during the server installation you can grab the RPM package from your **Server** directory of your installation media and install it using the **rpm -ivh vsftpd*.rpm** (note that the RPM package must be in your current directory).

1. To configure vsftpd, edit **/etc/passwd** using **vipw** and change the ftp user's home directory to the directory where you are going to keep the installation trees for your para-virtualized guests. An example entry for the FTP user would look like the following:

   ```
   ftp:x:14:50:FTP User:/xen/pub:/sbin/nologin
   ```

2. to have `vsftpd` start automatically during system boot use the chkconfig utility to enable the automatic start up of `vsftpd`.

3. verify that `vsftpd` is not enabled using the **chkconfig --list vsftpd**:

```
$ chkconfig --list vsftpd
vsftpd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

4. run the **chkconfig --levels 345 vsftpd on** to start vsftpd automatically for run levels 3, 4 and 5.

5. use the **chkconfig --list vsftpd** command to verify vsftdp has been enabled to start during system boot:

```
$ chkconfig --list vsftpd
vsftpd          0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

6. use the **service vsftpd start vsftpd** to start the vsftpd service:

```
$service vsftpd start vsftpd
Starting vsftpd for vsftpd:                  [  OK  ]
```

28.13. Configuring LUN Persistence

This section covers how to implement *LUN* persistence in guests and on the host machine with and without multipath.

Implementing LUN persistence without multipath

If your system is not using multipath, you can use **udev** to implement LUN persistence. Before implementing LUN persistence in your system, ensure that you acquire the proper UUIDs. Once you acquire these, you can configure LUN persistence by editing the **scsi_id** file that resides in the **/etc** directory. Once you have this file open in a text editor, you must comment out this line:

```
# options=-b
```

Then replace it with this parameter:

```
# options=-g
```

This tells udev to monitor all system SCSI devices for returning UUIDs. To determine the system UUIDs, use the **scsi_id** command:

```
# scsi_id -g -s /block/sdc
*3600a0b80001327510000015427b625e*
```

The long string of characters in the output is the UUID. The UUID does not change when you add a new device to your system. Acquire the UUID for each the device in order to create rules for the devices. To create new device rules, edit the **20-names.rules** file in the **/etc/udev/rules.d** directory. The device naming rules follow this format:

```
# KERNEL="sd*",  BUS="scsi",  PROGRAM="sbin/scsi_id", RESULT="UUID",
 NAME="devicename"
```

Replace your existing *UUID* and *devicename* with the above UUID retrieved entry. The rule should resemble the following:

```
KERNEL="sd*",  BUS="scsi",  PROGRAM="sbin/scsi_id",
 RESULT="3600a0b80001327510000015427b625e", NAME="mydevicename"
```

This enables all devices that match the **/dev/sd*** pattern to inspect the given UUID. When it finds a matching device, it creates a device node called *dev/devicename*. For this example, the device node is */dev/mydevice* . Finally, append the **/etc/rc.local** file with this line:

```
/sbin/start_udev
```

Implementing LUN persistence with multipath

To implement LUN persistence in a multipath environment, you must define the alias names for the multipath devices. For this example, you must define four device aliases by editing the **multipath.conf** file that resides in the **/etc/** directory:

```
multipath  {
             wwid       3600a0b80001327510000015427b625e
             alias      oramp1
}
multipath  {
             wwid       3600a0b80001327510000015427b6
             alias      oramp2
}
multipath  {
             wwid       3600a0b80001327510000015427b625e
             alias      oramp3
}
multipath  {
             wwid       3600a0b80001327510000015427b625e
             alias      oramp4
}
```

This defines 4 LUNs: **/dev/mpath/oramp1**, **/dev/mpath/oramp2**, **/dev/mpath/oramp3**, and **dev/mpath/oramp4**. The devices will reside in the **/dev/mpath** directory. These LUN names are persistent over reboots as it creates the alias names on the wwid of the LUNs.

28.14. Disable SMART disk monitoring for guests

SMART disk monitoring can be disabled as we are running on virtual disks and the physical storage is managed by the host.

```
/sbin/service smartd stop
/sbin/chkconfig --del smartd
```

28.15. Cleaning up old Xen configuration files

Over time you will see a number of files accumulate in **`/var/lib/xen`**, the usually named **`vmlinuz.******`** and **`initrd.******`**. These files are the initrd and vmlinuz files from virtual machines which either failed to boot or failed for some other reason. These files are temporary files extracted from virtual machine's boot disk during the start up sequence. These files should be automatically removed after the virtual machine is shut down cleanly. Then you can safely delete old and stale copies from this directory.

28.16. Configuring a VNC Server

To configure a VNC server use the **Remote Desktop** application in **System** > **Preferences**. Alternatively, you can run the **`vino-preferences`** command.

The following steps set up a dedicated VNC server session:

1. Edit the **`~/.vnc/xstartup`** file to start a GNOME session whenever **vncserver** is started. The first time you run the **vncserver** script it will ask you for a password you want to use for your VNC session.

2. A sample **`xstartup`** file:

```
#!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
#xsetroot -solid grey
#vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
if test -z "$DBUS_SESSION_BUS_ADDRESS" ; then
 eval `dbus-launch --sh-syntax –exit-with-session`
 echo "D-BUS per-session daemon address is: \
 $DBUS_SESSION_BUS_ADDRESS"
fi
exec  gnome-session
```

28.17. Cloning guest configuration files

You can copy an existing configuration file to create an all new guest. You must modify the name parameter of the guests' configuration file. The new, unique name then appears in the hypervisor and is viewable by the management utilities. You must generate an all new UUID as well by using the **`uuidgen`** command. Then for the **`vif`** entries you must define a unique MAC address for each guest (if you are copying a guest configuration from an existing guest, you can create a script to handle it). For the xen bridge information, if you move an existing guest configuration file to a new host, you must update the **`xenbr`** entry to match your local networking configuration. For the Device entries, you must modify the entries in the **`'disk='`** section to point to the correct guest image.

You must also modify these system configuration settings on your guest. You must modify the HOSTNAME entry of the **`/etc/sysconfig/network`** file to match the new guest's hostname.

You must modify the **HWADDR** address of the **/etc/sysconfig/network-scripts/ifcfg-eth0** file to match the output from **ifconfig eth0** file and if you use static IP addresses, you must modify the **IPADDR** entry.

28.18. Duplicating an existing guest and its configuration file

This section outlines copying an existing configuration file to create a new guest. There are key parameters in your guest's configuration file you must be aware of, and modify, to successfully duplicate a guest.

`name`
: The name of your guest as it is known to the hypervisor and displayed in the management utilities. This entry should be unique on your system.

`uuid`
: A unique handle for the guest, a new UUID can be regenerated using the **uuidgen** command. A sample UUID output:

```
$ uuidgen
a984a14f-4191-4d14-868e-329906b211e5
```

`vif`

- The *MAC address* must define a unique MAC address for each guest. This is automatically done if the standard tools are used. If you are copying a guest configuration from an existing guest you can use the script *Section 28.8, "Generating a new unique MAC address"*.
- If you are moving or duplicating an existing guest configuration file to a new host you have to make sure you adjust the `xenbr` entry to correspond with your local networking configuration (you can obtain the bridge information using the command **brctl show** command).
- Device entries, make sure you adjust the entries in the **disk=** section to point to the correct guest image.

Now, adjust the system configuration settings on your guest:

/etc/sysconfig/network
: Modify the `HOSTNAME` entry to the guest's new **hostname**.

/etc/sysconfig/network-scripts/ifcfg-eth0

- Modify the `HWADDR` address to the output from **ifconfig eth0**
- Modify the `IPADDR` entry if a static IP address is used.

Creating custom libvirt scripts

This section provides some information which may be useful to programmers and system administrators intending to write custom scripts to make their lives easier by using **`libvirt`**.

Chapter 28, Tips and tricks is recommended reading for programmers thinking of writing new applications which use **`libvirt`**.

29.1. Using XML configuration files with virsh

`virsh` can handle XML configuration files. You may want to use this to your advantage for scripting large deployments with special options. You can add devices defined in an XML file to a running para-virtualized guest. For example, to add a ISO file as **hdc** to a running guest create an XML file:

```
# cat satelliteiso.xml
<disk type="file" device="disk">
 <driver name="file"/>
 <source file="/var/lib/libvirt/images/rhn-satellite-5.0.1-11-redhat-linux-
as-i386-4-embedded-oracle.iso"/>
 <target dev="hdc"/>
 <readonly/>
</disk>
```

Run **`virsh attach-device`** to attach the ISO as **hdc** to a guest called "satellite" :

```
# virsh attach-device satellite satelliteiso.xml
```

Part VII. Troubleshooting

Introduction to Troubleshooting and Problem Solving

The following chapters provide information to assist you in troubleshooting issues you may encounter using virtualization.

Important note on virtualization issues

Your particular problem may not appear in this book due to ongoing development which creates and fixes bugs. For the most up to date list of known bugs, issues and bug fixes read the Red Hat Enterprise Linux *Release Notes* for your version and hardware architecture. The *Release Notes* can be found in the documentation section of the Red Hat website, *www.redhat.com/docs/manuals/enterprise/*[1].

If all else fails...

Contact Red Hat Global Support Services (*https://www.redhat.com/apps/support/*). Our staff will happily assist you in resolving your issues.

Troubleshooting Xen

This chapter covers essential concepts to assist you in troubleshooting problems in Xen. Troubleshooting topics covered in this chapter include:

- troubleshooting tools for Linux and virtualization.
- troubleshooting techniques for identifying problems.
- The location of log files and explanations of the information in logs.

This chapter is to give you, the reader, a background to identify where problems with virtualization technologies are. Troubleshooting takes practice and experience which are difficult to learn from a book. It is recommended that you experiment and test virtualization on Red Hat Enterprise Linux to develop your troubleshooting skills.

If you cannot find the answer in this document there may be an answer online from the virtualization community. Refer to *Section B.1, "Online resources"* for a list of Linux virtualization websites.

30.1. Debugging and troubleshooting Xen

This section summarizes the System Administrator applications, the networking utilities, and debugging tools. You can employ these standard System administration tools and logs to assist with troubleshooting:

Useful commands and applications for troubleshooting

xentop
: **xentop** displays real-time information about a host system and the guest domains.

xm
: Using the **dmesg** and **log**

- **vmstat**
- **iostat**
- **lsof**

The **iostat**, **mpstat** and **sar** commands are all provided by the **sysstat** package.

You can employ these Advanced Debugging Tools and logs to assist with troubleshooting:

- **XenOprofile**
- **systemtap**
- **crash**
- **sysrq**
- **sysrq t**
- **sysrq w**

These networking tools can assist with troubleshooting virtualization networking problems:

- **`ifconfig`**
- **`tcpdump`**

  The **`tcpdump`** command 'sniffs' network packets. **`tcpdump`** is useful for finding network abnormalities and problems with network authentication. There is a graphical version of **`tcpdump`** named **`wireshark`**.

- **`brctl`**

  `brctl` is a networking tool that inspects and configures the Ethernet bridge configuration in the Virtualization linux kernel. You must have root access before performing these example commands:

```
# brctl show

bridge-name    bridge-id          STP  enabled  interfaces
-----------------------------------------------------------------------------
xenbr0             8000.feffffff       no        vif13.0
xenbr1             8000.ffffefff       yes       pddummy0
xenbr2             8000.ffffffef       no        vif0.0

# brctl showmacs xenbr0

port-no           mac-addr                  local?       aging timer

1                 fe:ff:ff:ff:ff:           yes            0.00
2                 fe:ff:ff:fe:ff:           yes            0.00


# brctl showstp xenbr0

xenbr0

bridge-id              8000.fefffffffff

designated-root        8000.fefffffffff

root-port              0                   path-cost             0

max-age                20.00               bridge-max-age        20.00

hello-time             2.00                bridge-hello-time     2.00

forward-delay          0.00                bridge-forward-delay  0.00

aging-time            300.01

hello-timer            1.43                tcn-timer             0.00

topology-change-timer  0.00                gc-timer              0.02
```

Listed below are some other useful commands for troubleshooting virtualization on Red Hat Enterprise Linux 5. All utilities mentioned can be found in the **Server** repositories Red Hat Enterprise Linux 5.

- **strace** is a command which traces system calls and events received and used by another process.
- **vncviewer**: connect to a VNC server running on your server or a virtual machine. Install **vncviwer** using the **yum install vnc** command.
- **vncserver**: start a remote desktop on your server. Gives you the ability to run graphical user interfaces such as virt-manager via a remote session. Install **vncserver** using the **yum install vnc-server** command.

30.2. Log files overview

When deploying Red Hat Enterprise Linux 5 with Virtualization into your network infrastructure, the host's Virtualization software uses many specific directories for important configuration, log files, and other utilities. All the Xen logs files are standard ASCII files, and accessible with a text editor:

- The Xen configuration directory is **/etc/xen/**. This directory contains the **xend** daemon and other virtual machine configuration files. The networking script files are found in the **scripts** directory).
- All of actual log files themselves that you will consult for troubleshooting purposes reside in the **/var/log/xen** directory.
- You should also know that the default directory for all virtual machine file based disk images resides in the **/var/lib/xen** directory.
- Xen kernel information is stored in the **/proc/xen/** directory.

30.3. Log file descriptions

Xen features the **xend** daemon and **qemu-dm** process, two utilities that write the multiple log files to the **/var/log/xen/** directory:

- **xend.log** is the log file that contains all the data collected by the **xend** daemon, whether it is a normal system event, or an operator initiated action. All virtual machine operations (such as create, shutdown, destroy and so on) appear in this log. The **xend.log** is usually the first place to look when you track down event or performance problems. It contains detailed entries and conditions of the error messages.
- **xend-debug.log** is the log file that contains records of event errors from **xend** and the Virtualization subsystems (such as framebuffer, Python scripts, and so on).
- **xen-hotplug-log** is the log file that contains data from hotplug events. If a device or a network script does not come online, the event appears here.
- **qemu-dm.[PID].log** is the log file created by the **qemu-dm** process for each fully virtualized guest. When using this log file, you must retrieve the given **qemu-dm** process PID, by using the **ps** command to examine process arguments to isolate the **qemu-dm** process on the virtual machine. Note that you must replace the [PID] symbol with the actual PID **qemu-dm** process.

If you encounter any errors with the Virtual Machine Manager, you can review the generated data in the **virt-manager.log** file that resides in the **/.virt-manager** directory. Note that every time you start the Virtual Machine Manager, it overwrites the existing log file contents. Make sure to backup the **virt-manager.log** file, before you restart the Virtual Machine manager after a system error.

30.4. Important directory locations

There are other utilities and log files you should be aware of for tracking errors and troubleshooting problems with Xen:

- Virtualized guest images reside in the **/var/lib/libvirt/images** directory.
- When you restart the **xend** daemon, it updates the **xend-database** that resides in the **/var/lib/xen/xend-db** directory.
- Virtual machine dumps (that you perform with the **xm dump-core** command) resides in the **/var/lib/xen/dumps** directory.
- The **/etc/xen** directory contains the configuration files that you use to manage system resources. The **xend** daemon configuration file is **/etc/xen/xend-config.sxp**. This file can be edited to implement system-wide changes and configure the networking. However, manually editing files in the **/etc/xen/** folder is not advised.
- The **proc** folders are another resource that allows you to gather system information. These proc entries reside in the **/proc/xen** directory:

/proc/xen/capabilities

/proc/xen/balloon

/proc/xen/xenbus/

30.5. Troubleshooting with the logs

When encountering installation issues with Xen, refer to the host system's two logs to assist with troubleshooting. The **xend.log** file contains the same basic information as when you run the **xm log** command. This log is found in the **/var/log/** directory. Here is an example log entry for when you create a domain running a kernel:

```
[2006-12-27 02:23:02 xend] ERROR (SrvBase: 163) op=create: Error creating
 domain: (0, 'Error')
Traceback (most recent call list)
File "/usr/lib/python2.4/site-packages/xen/xend/server/SrvBase.py" line 107
 in_perform val = op_method (op,req)
File
"/usr/lib/python2.4/site-packages/xen/xend/server/SrvDomainDir.py line 71
 in op_create
raise XendError ("Error creating domain: " + str(ex))
XendError: Error creating domain: (0, 'Error')
```

The other log file, **xend-debug.log**, is very useful to system administrators since it contains even more detailed information than **xend.log** . Here is the same error data for the same kernel domain creation problem:

```
ERROR: Will only load images built for Xen v3.0
```

```
ERROR: Actually saw: GUEST_OS=netbsd, GUEST_VER=2.0, XEN_VER=2.0;
 LOADER=generic, BSD_SYMTAB'
ERROR: Error constructing guest OS
```

When calling customer support, always include a copy of both these log files when contacting the technical support staff.

30.6. Troubleshooting with the serial console

The serial console is helpful in troubleshooting difficult problems. If the Virtualization kernel crashes and the hypervisor generates an error, there is no way to track the error on a local host. However, the serial console allows you to capture it on a remote host. You must configure the host to output data to the serial console. Then you must configure the remote host to capture the data. To do this, you must modify these options in the **grub.conf** file to enable a 38400-bps serial console on **com1/dev/ttyS0**:

```
title Red Hat Enterprise Linux (2.6.18-8.2080_RHEL5xen0)
  root (hd0,2)
  kernel /xen.gz-2.6.18-8.el5 com1=38400,8n1
  module /vmlinuz-2.618-8.el5xen ro root=LABEL=/rhgb quiet console=xvc
 console=tty xencons=xvc
        module /initrd-2.6.18-8.el5xen.img
```

The **sync_console** can help determine a problem that causes hangs with asynchronous hypervisor console output, and the **"pnpacpi=off"** works around a problem that breaks input on the serial console. The parameters **"console=ttyS0"** and **"console=tty"** means that kernel errors get logged with on both the normal VGA console and on the serial console. Then you can install and set up **ttywatch** to capture the data on a remote host connected by a standard null-modem cable. For example, on the remote host you could type:

Itanium serial console troubleshooting

To access the hypervisor via a serial console on the Itanium® architecture you must enable the console in ELILO. For more information on configuring ELILO, refer to *Chapter 26, Configuring ELILO*.

```
ttywatch --name myhost --port /dev/ttyS0
```

This pipes the output from **/dev/ttyS0** into the file **/var/log/ttywatch/myhost.log** .

30.7. Para-virtualized guest console access

Para-virtualized guest operating systems automatically has a virtual text console configured to transmit data to the host operating system. Connect to a guest's virtual console with the following command:

```
# virsh console [guest name, ID or UUID]
```

You can also use **virt-manager** to display the virtual text console. In the Virtual Machine Details window, select **Serial Console** from the **View** menu.

30.8. Fully virtualized guest console access

Fully virtualized guest operating systems automatically has a text console configured for use, but the difference is the kernel guest is not configured. To enable the guest virtual serial console to work with the Full Virtualized guest, you must modify the guest's **`grub.conf`** file, and include the **`'console =ttyS0 console=tty0'`** parameter. This ensures that the kernel messages are sent to the virtual serial console (and the normal graphical console). If you plan to use the virtual serial console in a full virtualized guest, you must edit the configuration file in the **`/etc/xen/`** directory. On the host domain, access the serial console with the following command:

```
# virsh console
```

You can also use **`virt-manager`** to display the virtual text console. In the Virtual Machine Details window, select **Serial Console** from the **View** menu.

30.9. Accessing data on guest disk image

You can use two separate applications that assist you in accessing data from within a guest disk image. Before using these tools, you must shut down the guests. Accessing the file system from the guest and dom0 could potentially harm your system.

You can use the kpartx application to handle partitioned disks or LVM volume groups:

```
yum install kpartx
```

```
kpartx -av /dev/xen/guest1
add map guest1p1 : 0 208782 linear /dev/xen/guest1 63
add map guest1p2: 0 16563015 linear /dev/xen/guest1 208845
```

To access LVM volumes on a second partition, you must rescan LVM with vgscan and activate the volume group on the partition (called **`VolGroup00`** by default) by using the **`vgchange -ay`** command:

```
# kpartx -a /dev/xen/guest1
#vgscan
Reading all physical volumes . This may take a while...
Found volume group "VolGroup00" using metadata type lvm2
# vgchange -ay VolGroup00
2 logical volumes in volume group VolGroup00 now active.
# lvs
LV VG Attr Lsize Origin Snap% Move Log Copy%
LogVol00 VolGroup00 -wi-a- 5.06G
LogVol01 VolGroup00 -wi-a- 800.00M
# mount /dev/VolGroup00/LogVol00 /mnt/
....
#umount /mnt/
#vgchange -an VolGroup00
#kpartx -d /dev/xen/guest1
```

You must remember to deactivate the logical volumes with **`vgchange -an`**, remove the partitions with **`kpartx-d`** , and delete the loop device with **`losetup -d`** when you finish.

30.10. Common Xen problems

When attempting to start the **xend** service, nothing happens. Type **virsh list** and receive the following:

```
Error: Error connecting to xend: Connection refused. Is xend running?
```

Try to run **xend start** manually and receive more errors:

```
Error: Could not obtain handle on privileged command interfaces (2 = No
 such file or directory)
Traceback (most recent call last:)

File "/usr/sbin/xend/", line 33 in ?

from xen.xend.server. import SrvDaemon

File "/usr/lib/python2.4/site-packages/xen/xend/server/SrvDaemon.py" , line
 26 in ?

from xen.xend import XendDomain

File "/usr//lib/python2.4/site-packages/xen/xend/XendDomain.py" , line 33,
 in ?


from xen.xend import XendDomainInfo

File "/usr/lib/python2.4/site-packages/xen/xend/image.py" , line37, in ?

import images

File "/usr/lib/python2.4/site-packages/xen/xend/image.py" , line30, in ?

xc = xen.lowlevel.xc.xc ()

RuntimeError: (2, 'No such file or directory' )
```

What has most likely happened here is that you rebooted your host into a kernel that is not a **kernel-xen** kernel. To correct this, you must select the **kernel-xen** kernel at boot time (or set the **kernel-xen** kernel to the default in the **grub.conf** file.

30.11. Guest creation errors

When you attempt to create a guest, you receive an **"Invalid argument"** error message. This usually means that the kernel image you are trying to boot is incompatible with the hypervisor. An example of this would be if you were attempting to run a non-PAE FC5 kernel on a PAE only FC6 hypervisor.

You do a yum update and receive a new kernel, the **grub.conf** default kernel switches right back to a bare-metal kernel instead of the Virtualization kernel.

To correct this problem you must modify the default kernel RPM that resides in the **/etc/sysconfig/kernel/** directory. You must ensure that **kernel-xen** parameter is set as the default option in your **gb.conf** file.

30.12. Troubleshooting with serial consoles

Linux kernels can output information to serial ports. This is useful for debugging kernel panics and hardware issues with video devices or headless servers. The subsections in this section cover setting up serial console output for machines running Red Hat Enterprise Linux virtualization kernels and their virtualized guests.

30.12.1. Serial console output for Xen

By default, Xen's serial console is disabled and no data is output from serial ports.

To receive kernel information on a serial port modify the **/boot/grub/grub.conf** file by setting the appropriate serial device parameters.

If your serial console is on `com1`, modify **/boot/grub/grub.conf** by inserting the lines *com1=115200,8n1*, *console=tty0* and *console=ttyS0,115200* where shown.

```
title Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
 root (hd0, 8)
 kernel /boot/xen.gz-2.6.18-92.el5 com1=115200,8n1
 module /boot/vmlinuz-2.6.18-92.el5xen ro
 root=LABEL=RHEL5_i386 console=tty0
 console=ttyS0,115200
 module /boot/initrd-2.6.18-92.el5xen.img
```

If your serial console is on `com2`, modify **/boot/grub/grub.conf** by inserting the lines *com2=115200,8n1 console=com2L*, *console=tty0* and *console=ttyS0,115200* where shown.

```
title Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
 root (hd0, 8)
 kernel /boot/xen.gz-2.6.18-92.el5 com2=115200,8n1 console=com2L
 module /boot/vmlinuz-2.6.18-92.el5xen ro root=LABEL=RHEL5_i386
 console=tty0
 console=ttyS0,115200
 module /boot/initrd-2.6.18-92.el5xen.img
```

Save the changes and reboot the host. The hypervisor outputs serial data on the serial (`com1`, `com2` and so on) you selected in the previous step.

Note the example using the `com2` port, the parameter *console=ttyS0* on the *vmlinuz* line us used. The behavior of every port being used as *console=ttyS0* is not standard Linux behavior and is specific to the Xen environment.

30.12.2. Xen serial console output from para-virtualized guests

This section describes how to configure a virtualized serial console for Red Hat Enterprise Linux para-virtualized guests.

Serial console output from para-virtualized guests can be received using the "**`virsh console`**" or in the "**Serial Console**" window of **`virt-manager`**. Set up the virtual serial console using this procedure:

1. Log in to your para-virtualized guest.

2. Edit **`/boot/grub/grub.conf`** as follows:

```
Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
 root (hd0, 0) kernel /boot/vmlinuz-2.6.18-92.el5xen ro
 root=LABEL=RHEL5_i386 console=xvc0
 initrd /boot/initrd-2.6.18-92.el5xen.img
```

3. Reboot the para-virtualized guest.

You should now get kernel messages on the virt-manager "Serial Console" and "virsh console".

Logging the para-virtualized domain serial console output

The Xen daemon(`xend`) can be configured to log the output from serial consoles of para-virtualized guests.

To configure `xend` edit **`/etc/sysconfig/xend`**. Change the entry:

```
# Log all guest console output (cf xm console)
#XENCONSOLED_LOG_GUESTS=no
```

to:

```
# Log all guest console output (cf xm console)
XENCONSOLED_LOG_GUESTS=yes
```

Reboot the host to activate logging the guest serial console output.

Logs from the guest serial consoles are stored in the **`/var/log/xen/console`** file.

30.12.3. Serial console output from fully virtualized guests

This section covers how to enable serial console output for fully virtualized guests.

Fully virtualized guest serial console output can be viewed with the "**`virsh console`**" command.

Be aware fully virtualized guest serial consoles have some limitations. Present limitations include:

- logging output with xend is unavailable.

- output data may be dropped or scrambled.

The serial port is called `ttyS0` on Linux or `COM1` on Windows.

You must configure the virtualized operating system to output information to the virtual serial port.

To output kernel information from a fully virtualized Linux guest into the domain modify the **`/boot/grub/grub.conf`** file by inserting the line "*`console=tty0 console=ttys0,115200`*".

```
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
 root (hd0,0)
 kernel /vmlinuz-2.6.18-92.el5 ro root=/dev/volgroup00/logvol00
 console=tty0 console=ttys0,115200
 initrd /initrd-2.6.18-92.el5.img
```

Reboot the guest.

View the serial console messages using the "**virsh console**" command.

> **Note**
> Serial console messages from fully virtualized domains are not logged in **/var/log/xen/console** as they are for para-virtualized guests.

30.13. Guest configuration files

When you create guests with the virt-manager or virt-install tools on Red Hat Enterprise Linux 5, the guests configuration files are created automatically in the **/etc/xen** directory. The example below is a typical a para-virtualized guest configuration file:

```
name = "rhel5vm01"
memory = "2048"
disk = ['tap:aio:/var/lib/libvirt/images/rhel5vm01.dsk,xvda,w',]
vif = ["type=ieomu, mac=00:16:3e:09:f0:12 bridge=xenbr0',
"type=ieomu, mac=00:16:3e:09:f0:13 ]
vnc = 1
vncunused = 1
uuid = "302bd9ce-4f60-fc67-9e40-7a77d9b4e1ed"
bootloader = "/usr/bin/pygrub"
vcpus=2
on_reboot = "restart"
on_crash = "restart"
```

Note that the **serial="pty"** is the default for the configuration file. This configuration file example is for a fully-virtualized guest:

```
name = "rhel5u5-86_64"
builder = "hvm"
memory = 500
disk = ['/var/lib/libvirt/images/rhel5u5-x86_64.dsk.hda,w']
vif = [ 'type=ioemu, mac=00:16:3e:09:f0:12, bridge=xenbr0', 'type=ieomu,
 mac=00:16:3e:09:f0:13, bridge=xenbr1']
uuid = "b10372f9-91d7-ao5f-12ff-372100c99af5'
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader/"
vnc = 1
vncunused = 1
apic = 1
```

```
acpi = 1
pae = 1
vcpus =1
serial ="pty" # enable serial console
on_boot = 'restart'
```

Xen configuration files

Editing Xen configuration files is unsupported. Use **`virsh dumpxml`** and **`virsh create`** (or **`virsh edit`**) to edit the **`libvirt`** configuration files (xml based) which have error checking and safety checks.

30.14. Interpreting Xen error messages

You receive the following error:

```
failed domain creation due to memory shortage, unable to balloon domain0
```

A domain can fail if there is not enough RAM available. Domain0 does not balloon down enough to provide space for the newly created guest. You can check the **`xend.log`** file for this error:

```
[2006-12-21] 20:33:31 xend 3198] DEBUG (balloon:133) Balloon: 558432 Kib
 free; 0 to scrub; need 1048576; retries: 20
[2006-12-21] 20:33:31 xend. XendDomainInfo 3198] ERROR (XendDomainInfo: 202
Domain construction failed
```

You can check the amount of memory in use by domain0 by using the **`xm list domain0`** command. If dom0 is not ballooned down, you can use the command **`virsh setmem dom0 NewMemSize`** to check memory.

You receive the following error:

```
wrong kernel image: non-PAE kernel on a PAE
```

This message indicates that you are trying to run an unsupported guest kernel image on your hypervisor. This happens when you try to boot a non-PAE, para-virtualized guest kernel on a Red Hat Enterprise Linux 5 host. The Red Hat *kernel-xen* package only supports guest kernels with PAE and 64 bit architectures.

Type this command:

```
# xm create -c va-base

Using config file "va-base"
Error: (22, 'invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] ERRORs
(XendDomainInfo:202) Domain construction failed

Traceback (most recent call last)
```

```
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py", line
 195 in create vm.initDomain()
File " /usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py", line
 1363 in initDomain raise VmError(str(exn))
VmError: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XenDomainInfo: 1449]
XendDlomainInfo.destroy: domain=1
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XenDomainInfo: 1457]
XendDlomainInfo.destroy:Domain(1)
```

If you need to run a 32 bit non-PAE kernel you will need to run your guest as a fully virtualized virtual machine. For para-virtualized guests, if you need to run a 32 bit PAE guest, then you must have a 32 bit PAE hypervisor. For para-virtualized guests, to run a 64 bit PAE guest, then you must have a 64 bit PAE hypervisor. For full virtualization guests you must run a 64 bit guest with a 64 bit hypervisor. The 32 bit PAE hypervisor that comes with Red Hat Enterprise Linux 5 i686 only supports running 32 bit PAE para virtualized and 32 bit fully virtualized guest OSes. The 64 bit hypervisor only supports 64 bit para-virtualized guests.

This happens when you move the full virtualized HVM guest onto a Red Hat Enterprise Linux 5 system. Your guest may fail to boot and you will see an error in the console screen. Check the PAE entry in your configuration file and ensure that pae=1.You should use a 32 bit distribution.

You receive the following error:

```
Unable to open a connection to the Xen hypervisor or daemon
```

This happens when the virt-manager application fails to launch. This error occurs when there is no localhost entry in the **/etc/hosts** configuration file. Check the file and verify if the localhost entry is enabled. Here is an example of an incorrect localhost entry:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
localhost.localdomain localhost
```

Here is an example of a correct localhost entry:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 localhost.localdomain localhost
localhost.localdomain. localhost
```

You receive the following error (in the **xen-xend.logfile**):

```
Bridge xenbr1 does not exist!
```

This happens when the guest's bridge is incorrectly configured and this forces the Xen hotplug scripts to timeout. If you move configuration files between hosts, you must ensure that you update the guest configuration files to reflect network topology and configuration modifications. When you attempt to start a guest that has an incorrect or non-existent Xen bridge configuration, you will receive the following errors:

```
# xm create mySQL01

Using config file " mySQL01"
Going to boot Red Hat Enterprise Linux Server (2.6.18.-1.2747 .el5xen)
kernel: /vmlinuz-2.6.18-12747.el5xen
initrd: /initrd-2.6.18-1.2747.el5xen.img
Error: Device 0 (vif) could not be connected. Hotplug scripts not working.
```

In addition, the **xend.log** displays the following errors:

```
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:143) Waiting for
 devices vif
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:149) Waiting for 0
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:464)
 hotplugStatusCallback

/local/domain/0/backend/vif/2/0/hotplug-status

[2006-11-14 15:08:09 xend.XendDomainInfo 3875] DEBUG (XendDomainInfo:1449)
 XendDomainInfo.destroy: domid=2
[2006-11-14 15:08:09 xend.XendDomainInfo 3875] DEBUG (XendDomainInfo:1457)
 XendDomainInfo.destroyDomain(2)
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:464)
 hotplugStatusCallback

/local/domain/0/backend/vif/2/0/hotplug-status
```

To resolve this problem, open the guest's configuration file found in the **/etc/xen** directory. For example, editing the guest *mySQL01*

```
# vim /etc/xen/mySQL01
```

Locate the **vif** entry. Assuming you are using **xenbr0** as the default bridge, the proper entry should resemble the following:

```
# vif = ['mac=00:16:3e:49:1d:11, bridge=xenbr0',]
```

You receive these python deprecation errors:

```
# xm shutdown win2k3xen12
# xm create win2k3xen12

Using config file "win2k3xen12".

/usr/lib64/python2.4/site-packages/xenxm/opts.py:520: Deprecation Warning:
Non ASCII character '\xc0' in file win2k3xen12 on line 1, but no encoding
declared; see http://www.python.org/peps/pep-0263.html for details

execfile (defconfig, globs, locs,)
```

```
Error: invalid syntax 9win2k3xen12, line1)
```

Python generates these messages when an invalid (or incorrect) configuration file. To resolve this problem, you must modify the incorrect configuration file, or you can generate a new one.

30.15. The layout of the log directories

The basic directory structure in a Red Hat Enterprise Linux 5 Virtualization environment is as follows:

`/etc/xen/` directory contains

- configuration files used by the `xend` daemon.
- the **`scripts`** directory which contains the scripts for Virtualization networking.

> **Tip**
>
> Before moving virtual machine configuration files to a different location, ensure you are not working off old or stale configuration files.

`/var/log/xen/`

- directory holding all Xen related log files.

`/var/lib/libvirt/images/`

- The default directory for virtual machine image files.
- If you are using a different directory for your virtual machine images make sure you add the directory to your SELinux policy and relabel it before starting the installation.

`/proc/xen/`

- The xen related information in the /proc file system.

Troubleshooting

This chapter covers common problems and solutions with Red Hat Enterprise Linux virtualization.

31.1. Identifying available storage and partitions

Verify the block driver is loaded and the devices and partitions are available to the guest. This can be done by executing "**cat /proc/partitions**" as seen below.

```
# cat /proc/partitions
major minor  #blocks   name
 202    16  104857600  xvdb
   3     0    8175688  hda
```

31.2. After rebooting Xen-based guests the console freezes

Occasionally, a Xen guest's console freezes when the guest boots. The console still displays messages but the user cannot log in.

To fix this issue add the following line to the **/etc/inittab** file:

```
1:12345:respawn:/sbin/mingetty xvc0
```

After saving the file, reboot. The console session should now be interactive as expected.

31.3. Virtualized Ethernet devices are not found by networking tools

The networking tools cannot identify the `Xen Virtual Ethernet` networking card inside the guest operation system you should execute **cat /etc/modprobe.conf** (in Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5) or **cat /etc/modules.conf** (in Red Hat Enterprise Linux 3). The output should contain the line "**alias eth0 xen-vnif**" and a similar line for each additional interface. To fix this problem you will need to add the aliasing lines (for example, **alias eth0 xen-vnif**) for every para-virtualized interface for the guest.

31.4. Loop device errors

If file based guest images are used you may have to increase the number of configured loop devices. The default configuration allows up to 8 active loop devices. If more than 8 file based guests or loop devices are needed the number of loop devices configured can be adjusted in **/etc/modprobe.conf**. Edit **/etc/modprobe.conf** and add the following line to it:

```
                options loop max_loop=64
```

This example uses 64 but you can specify another number to set the maximum loop value. You may also have to implement loop device backed guests on your system. To employ loop device backed guests for a para-virtualized guest, use the **phy: block device** or **tap:aio** commands. To employ loop device backed guests for a full virtualized system, use the **phy: device** or **file: file** commands.

31.5. Failed domain creation caused by a memory shortage

This may cause a domain to fail to start. The reason for this is there is not enough memory available or **dom0** has not ballooned down enough to provide space for a recently created or started guest. In your **/var/log/xen/xend.log**, an example error message indicating this has occurred:

```
[2006-11-21 20:33:31 xend 3198] DEBUG (balloon:133) Balloon: 558432 KiB
 free;
 0 to scrub; need 1048576; retries: 20.
[2006-11-21 20:33:52 xend.XendDomainInfo 3198] ERROR (XendDomainInfo:202)
 Domain construction failed
```

You can verify the amount of memory currently used by **dom0** with the command "**xm list Domain-0**". If **dom0** is not ballooned down you can use the command "**xm mem-set Domain-0 NewMemSize**" where `NewMemSize` should be a smaller value.

31.6. Wrong kernel image error

Para-virtualized guests cannot use the kernel-xen kernel. Use only the standard kernel for para-virtualized guests.

If you try to boot a non kernel-xen kernel as a para-virtualized guest the following error message appears:

```
# xm create testVM
Using config file "./testVM".
Going to boot Red Hat Enterprise Linux Server (2.6.18-1.2839.el5)
kernel: /vmlinuz-2.6.18-1.2839.el5
initrd: /initrd-2.6.18-1.2839.el5.img
Error: (22, 'Invalid argument')
```

In the above error you can see that the kernel line shows that it's trying to boot a non-xen kernel. The correct entry in the example is "**kernel: /vmlinuz-2.6.18-1.2839.el5xen**".

The solution is to verify you have indeed installed a kernel-xen in your guest and it is the default kernel to boot in your **/etc/grub.conf** configuration file.

If you have **kernel-xen** installed in your guest you can start your guest:

```
xm create -c GuestName
```

Where **GuestName** is the name of the guest. The previous command will present you with the **GRUB** boot loader screen and allow you to select the kernel to boot. You will have to choose the kernel-xen kernel to boot. Once the guest has completed the boot process you can log into the guest and edit **/etc/grub.conf** to change the default boot kernel to your kernel-xen. Simply change the line "**default=X**" (where X is a number starting at '**0**') to correspond to the entry with your kernel-xen line.

The numbering starts at '**0**' so if your kernel-xen entry is the second entry you would enter '**1**' as the default,for example "**default=1**".

31.7. Wrong kernel image error - non-PAE kernel on a PAE platform

If you to boot a non-PAE para-virtualized guest you will see the error message below. It basically indicates you are trying to run a guest kernel on your Hypervisor which at this time is not supported. The Xen hypervisor presently only supports PAE and 64 bit para-virtualized guest kernels.

```
# xm create -c va-base
Using config file "va-base".
Error: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] ERROR (XendDomainInfo:202)
 Domain construction failed
Traceback (most recent call last):
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py",
 line 195, in  create vm.initDomain()
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py",
 line 1363, in initDomain raise VmError(str(exn))
VmError: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1449)
 XendDomainInfo.destroy: domid=1
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1457)
 XendDomainInfo.destroyDomain(1)
```

If you need to run a 32 bit or non-PAE kernel you will need to run your guest as a fully-virtualized virtual machine. The rules for hypervisor compatibility are:

- para-virtualized guests must match the architecture type of your hypervisor. To run a 32 bit PAE guest you must have a 32 bit PAE hypervisor.
- to run a 64 bit para-virtualized guest your Hypervisor must be a 64 bit version too.
- fully virtualized guests your hypervisor must be 32 bit or 64 bit for 32 bit guests. You can run a 32 bit (PAE and non-PAE) guest on a 32 bit or 64 bit hypervisor.
- to run a 64 bit fully virtualized guest your hypervisor must be 64 bit too.

31.8. Fully-virtualized 64 bit guest fails to boot

If you have moved the configuration file to a Red Hat Enterprise Linux 5 causing your fully-virtualized guest fails to boot and present the error, "`Your CPU does not support long mode. Use a 32 bit distribution`". The problem is a missing or incorrect **pae** setting. Make sure you have an entry "**pae=1**" in your guest's configuration file.

31.9. A missing localhost entry causes virt-manager to fail

The **virt-manager** application may fail to launch and display an error such as "`Unable to open a connection to the Xen hypervisor/daemon`". This is usually caused by a missing `localhost` entry in the **/etc/hosts** file. Verify that you indeed have a `localhost` entry and if it is missing from **/etc/hosts** and insert a new entry for `localhost` if it is not present. An incorrect **/etc/hosts** may resemble the following:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
localhost.localdomain localhost
```

The correct entry should look similar to the following:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1               localhost.localdomain localhost
localhost.localdomain localhost
```

31.10. Microcode error during guest boot

During the boot phase of your virtual machine you may see an error message similar to:

```
Applying Intel CPU microcode update: FATAL: Module microcode not found.
ERROR: Module microcode does not exist in /proc/modules
```

As the virtual machine is running on virtual CPUs there is no point updating the microcode. Disabling the microcode update for your virtual machines will stop this error:

```
/sbin/service microcode_ctl stop
/sbin/chkconfig --del microcode_ctl
```

31.11. Wrong bridge configured on the guest causing hot plug script timeouts

If you have moved configuration files between different hosts you may to make sure your guest configuration files have been updated to reflect any change in your network topology, such as Xen bridge numbering.

If you try to start a guest which has an incorrect or non-existent virtual network bridge configured you will see the following error after starting the guest

```
# xm create r5b2-mySQL01
Using config file "r5b2-mySQL01".
Going to boot Red Hat Enterprise Linux Server (2.6.18-1.2747.el5xen)
kernel: /vmlinuz-2.6.18-1.2747.el5xen
initrd: /initrd-2.6.18-1.2747.el5xen.img
Error: Device 0 (vif) could not be connected. Hotplug scripts not working
```

In **/var/log/xen/xen-hotplug.log** you will see the following error being logged

```
bridge xenbr1 does not exist!
```

and in **/var/log/xen/xend.log** you will see the following messages (or similar messages) being logged

```
[2006-12-14 15:07:08 xend 3874] DEBUG (DevController:143) Waiting for
 devices vif.
[2006-12-14 15:07:08 xend 3874] DEBUG (DevController:149) Waiting for 0.
[2006-12-14 15:07:08 xend 3874] DEBUG (DevController:464)
 hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
[2006-12-14 15:07:08 xend 3874] DEBUG (DevController:464)
 hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
[2006-12-14 15:08:48 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1449)
 XendDomainInfo.destroy: domid=2
[2006-12-14 15:08:48 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1457)
 XendDomainInfo.destroyDomain(2)
[2006-12-14 15:08:48 xend 3874] DEBUG (DevController:464)
 hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
[2006-12-14 15:08:48 xend 3874] DEBUG (DevController:464)
 hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
[2006-12-14 15:08:48 xend 3874] DEBUG (DevController:464)
 hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
[2006-12-14 15:08:48 xend 3874] DEBUG (DevController:464)
hotplugStatusCallback /local/domain/0/backend/vif/2/0/hotplug-status.
```

To resolve this issue edit your guest's configuration file and modify the **vif** entry to reflect your local configuration. For example if your local configuration is using **xenbr0** as its default bridge you should modify your **vif** entry in your configuration file from

```
vif = [ 'mac=00:16:3e:49:1d:11, bridge=xenbr1', ]
```

to

```
vif = [ 'mac=00:16:3e:49:1d:11, bridge=xenbr0', ]
```

31.12. Python depreciation warning messages when starting a virtual machine

Sometimes Python will generate a message like the one below, these are often caused by either an invalid or incorrect configuration file. A configuration file containing non-ascii characters will cause these errors.The solution is to correct the configuration file or generate a new one.

Another cause is an incorrect configuration file in your current working directory. “**xm create**” will look in the current directory for a configuration file and then in **/etc/xen**

```
# xm shutdown win2k3xen12
# xm create win2k3xen12
Using config file "win2k3xen12".
/usr/lib64/python2.4/site-packages/xen/xm/opts.py:520: DeprecationWarning:
Non-ASCII character '\xc0' in file win2k3xen12 on line 1, but no encoding
declared; see http://www.python.org/peps/pep-0263.html for details
```

```
execfile(defconfig, globs, locs)
Error: invalid syntax (win2k3xen12, line 1)
```

31.13. Enabling Intel VT and AMD-V virtualization hardware extensions in BIOS

This section describes how to identify hardware virtualization extensions and enable them in your BIOS if they are disabled.

The Intel VT extensions can be disabled in the BIOS. Certain laptop vendors have disabled the Intel VT extensions by default in their CPUs.

The virtualization extensions can not be disabled in the BIOS for AMD-V (capable processors installed in a Rev 2 socket.

The virtualization extensions are sometimes disabled in BIOS, usually by laptop manufacturers. Refer to *Section 31.13, “Enabling Intel VT and AMD-V virtualization hardware extensions in BIOS”* for instructions on enabling disabled virtualization extensions.

Verify the virtualization extensions are enabled in BIOS. The BIOS settings for Intel® VT or AMD-V are usually in the **Chipset** or **Processor** menus. The menu names may vary from this guide, the virtualization extension settings may be found in **Security Settings** or other non standard menu names.

Procedure 31.1. Enabling virtualization extensions in BIOS

1. Reboot the computer and open the system's BIOS menu. This can usually be done by pressing **delete** or **`Alt`** + **`F4`**.

2. Select **Restore Defaults**, and then select **Save & Exit**.

3. Power off the machine and disconnect the power supply.

4. Power on the machine and open the **BIOS Setup Utility**. Open the **Processor** section and enable **Intel®Virtualization Technology** or **AMD-V**. The values may also be called **Virtualization Extensions** on some machines. Select **Save & Exit**.

5. Power off the machine and disconnect the power supply.

6. Run **`cat /proc/cpuinfo | grep vmx svm`**. If the command outputs, the virtualization extensions are now enabled. If there is no output your system may not have the virtualization extensions or the correct BIOS setting enabled.

Troubleshooting the Xen para-virtualized drivers

This chapter deals with issues you may encounter with Xen hosts and fully virtualized Red Hat Enterprise Linux guests using the para-virtualized drivers.

32.1. Red Hat Enterprise Linux 5 Virtualization log file and directories

Red Hat Enterprise Linux 5 Virtualization related log file

In Red Hat Enterprise Linux 5, the log file written by the **xend** daemon and the **qemu-dm** process are all kept in the following directories:

`/var/log/xen/`
: directory holding all log file generated by the **xend** daemon and qemu-dm process.

`xend.log`

- This logfile is used by xend to log any events generate by either normal system events or operator initiated events.
- virtual machine operations such as create, shutdown, destroy etc are all logged in this logfile.
- Usually this logfile will be the first place to look at in the event of a problem. In many cases you will be able to identify the root cause by scanning the logfile and review the entries logged just prior to the actual error message.

`xend-debug.log`

- Records error events from xend and its subsystems (from the framebuffer and Python scripts)

`xen-hotplug.log`

- Logs events from hotplug events.
- Event notifications from devices not coming online or network bridges not online are logged in this file.

`qemu-dm.`***`PID`***`.log`

- This file is created by the **qemu-dm** process which is started for each fully virtualized guest.
- The `PID` is replaced with the `PID` of the process of the related qemu-dm process
- You can retrieve the `PID` for a given **qemu-dm** process using the **ps** command and in looking at the process arguments you can identify the virtual machine the **qemu-dm** process belongs to.

If you are troubleshooting a problem with the **virt-manager** application you can also review the logfile generated by it. The logfile for **virt-manager** will be in a directory called **`.virt-manager`** in the user's **home** directory whom ran **virt-manager**. This directory will usually be **`~/.virt-manager/virt-manager`**.

> **Note**
>
> The logfile is overwritten every time you start **virt-manager**. If you are troubleshooting a problem with **virt-manager** make sure you save the logfile before you restart **virt-manager** after an error has occurred.

Red Hat Enterprise Linux 5 Virtualization related directories

There are a few other directories and files which may be of interest when troubleshooting a Red Hat Enterprise Linux 5 Virtualization environment:

`/var/lib/libvirt/images/`
: the standard directory for file-based guest images.

`/var/lib/xen/xend-db/`
: directory that hold the xend database which is generated every time the daemon is restarted.

`/etc/xen/`
: Stores a number of configuration files for the Xen hypervisor.

- **`/etc/xen/xend-config.sxp`** is the main configuration for the xend daemon. The **`xend-config.sxp`** file enables or disables migration and other features not configured by **`libvirt`**. Use the **`libvirt`** tools for all other features.

`/var/xen/dump/`
: Holds dumps generate by virtual machines or when using the **`xm dump-core`** command.

`/proc/xen/`
: Stores **`xen-kernel`** information in the following files:

- **`/proc/xen/capabilities`**
- **`/proc/xen/privcmd`**
- **`/proc/xen/balloon`**
- **`/proc/xen/xenbus`**
- **`/proc/xen/xsd_port`**
- **`/proc/xen/xsd_kva`**

32.2. Para-virtualized guest fail to load on a Red Hat Enterprise Linux 3 guest operating system

Red Hat Enterprise Linux 3 is uses processor architecture specific kernel RPMs and because of this the para-virtualized drivers may fail to load if the para-virtualized driver RPM does not match the installed kernel architecture.

When the para-virtualized driver modules are inserted, a long list of unresolved modules will be displayed. A shortened excerpt of the error can be seen below.

```
# insmod xen-platform-pci.o
Warning: kernel-module version mismatch
xen-platform-pci.o was compiled for kernel version 2.4.21-52.EL
while this kernel is version 2.4.21-50.EL
xen-platform-pci.o: unresolved symbol __ioremap_R9eac042a
xen-platform-pci.o: unresolved symbol flush_signals_R50973be2
xen-platform-pci.o: unresolved symbol pci_read_config_byte_R0e425a9e
xen-platform-pci.o: unresolved symbol __get_free_pages_R9016dd82
[...]
```

The solution is to use the correct RPM package for your hardware architecture for the para-virtualized drivers.

32.3. A warning message is displayed while installing the para-virtualized drivers on Red Hat Enterprise Linux 3

Installing the para-virtualized drivers on a Red Hat Enterprise Linux 3 kernel prior to 2.4.21-52 may result in a warning message being displayed stating the modules have been compiled with a newer version than the running kernel.

This message, as seen below, can be safely ignored.

```
Warning: kernel-module version mismatch
xen-platform-pci.o was compiled for kernel version 2.4.21-52.EL
while this kernel is version 2.4.21-50.EL
Warning: loading xen-platform-pci.o will taint the kernel: forced load
See http://www.tux.org/lkml/#export-tainted for information about tainted
 modules
Module xen-platform-pci loaded, with warnings
```

The important part of the message above is the last line which should state the module has been loaded with warnings.

32.4. Manually loading the para-virtualized drivers

If for some reason the para-virtualized drivers failed to load automatically during the boot process you can attempt to load them manually.

This will allow you to reconfigure network or storage entities or identify why they failed to load in the first place. The steps below should load the para-virtualized driver modules.

First, locate the para-virtualized driver modules on your system.

```
# cd /lib/modules/`uname -r`/
# find . -name 'xen-*.ko' -print
```

Take note of the location and load the modules manually. Substitute {LocationofPV-drivers} with the correct location you noted from the output of the commands above.

```
# insmod \
```

```
/lib/modules/'uname -r'/{LocationofPV-drivers}/xen-platform-pci.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-balloon.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-vnif.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-vbd.ko
```

32.5. Verifying the para-virtualized drivers have successfully loaded

One of the first tasks you will want to do is to verify that the drivers have actually been loaded into your system.

After the para-virtualized drivers have been installed and the guest has been rebooted you can verify that the drivers have loaded. First you should confirm the drivers have logged their loading into **`/var/log/messages`**

```
# grep -E "vif|vbd|xen" /var/log/messages
                    xen_mem: Initialising balloon driver
                    vif vif-0: 2 parsing device/vif/0/mac
                    vbd vbd-768: 19 xlvbd_add at /local/domain/0/backend/
vbd/21/76
                    vbd vbd-768: 19 xlvbd_add at /local/domain/0/backend/
vbd/21/76
                    xen-vbd: registered block device major 202
```

You can also use the **`lsmod`** command to list the loaded para-virtualized drivers. It should output a list containing the `xen_vnif`, `xen_vbd`, `xen_platform_pci` and `xen_balloon` modules.

```
# lsmod|grep xen
xen_vbd                19168  1
xen_vnif               28416  0
xen_balloon            15256  1 xen_vnif
xen_platform_pci       98520  3 xen_vbd,xen_vnif,xen_balloon,[permanent]
```

32.6. The system has limited throughput with para-virtualized drivers

If network throughput is still limited even after installing the para-virtualized drivers and you have confirmed they are loaded correctly (refer to *Section 32.5, “Verifying the para-virtualized drivers have successfully loaded”*). To fix this problem, remove the '**`type=ioemu`**' part of '**`vif=`**' line in your guest's configuration file.

Appendix A. Xen system architecture

A functional Red Hat Enterprise Linux system with virtualization is multi-layered and is driven by the privileged Xen kernel component. Xen can host multiple guest operating systems. Each guest operating system runs in its own domain. Xen schedules virtual CPUs within the virtual machines to make the best use of the available physical CPUs. Each guest operating systems handles its own applications. These guest operating systems schedule each application accordingly.

You can deploy Xen in one of two choices: *full virtualization* or *para-virtualization*. Full virtualization provides total abstraction of the underlying physical system and creates a new virtual system in which the guest operating systems can run. No modifications are needed in the guest OS or application (the guest OS or application is not aware of the virtualized environment and runs normally). Para-virtualization requires user modification of the guest operating systems that run on the virtual machines (these guest operating systems are aware that they are running on a virtual machine) and provide near-native performance. You can deploy both para-virtualization and full virtualization across your virtualization infrastructure.

The first domain, known as **domain0** (dom0), is automatically created when you boot the system. Domain0 is the privileged guest and it possesses management capabilities which can create new domains and manage their virtual devices. Domain0 handles the physical hardware, such as network cards and hard disk controllers. Domain0 also handles administrative tasks such as suspending, resuming, or migrating guest domains to other virtual machines.

The **hypervisor** (Red Hat's Virtual Machine Monitor) is a virtualization platform that allows multiple operating systems to run on a single host simultaneously within a full virtualization environment. A guest is an operating system (OS) that runs on a virtual machine in addition to the host or main OS.

With Xen, each guests **memory** comes from a slice of the host's physical memory. For para-virtualized guests, you can set both the initial memory and the maximum size of the virtual machine. You can add (or remove) physical memory to the virtual machine at runtime without exceeding the maximum size you specify. This process is called ballooning.

You can configure each guest with a number of virtual **cpus** (called VCPUs). VCPUs are scheduled according to the workload on the physical CPUs.

You can grant a guest any number of **virtual disks**. The guest sees these as either hard disks or (for full virtual guests) as CD-ROM drives. Each virtual disk is served to the guest from a block device or from a regular file on the host. The device on the host contains the entire full disk image for the guest, and usually includes partition tables, multiple partitions, and potentially LVM physical volumes.

Virtual networking interfaces runs on the guest. Other interfaces can run on the guest like virtual Ethernet Internet cards (VNICs). These network interfaces are configured with a persistent virtual media access control (MAC) address. The default installation of a new guest installs the VNIC with a MAC address selected at random from a reserved pool of over 16 million addresses, so it is unlikely that any two guests will receive the same MAC address. Complex sites with a large number of guests can allocate MAC addresses manually to ensure that they remain unique on the network.

Each guest has a virtual **text console** that connects to the host. You can redirect guest log in and console output to the text console.

You can configure any guest to use a virtual **graphical console** that corresponds to the normal video console on the physical host. You can do this for full virtual and para-virtualized guests. It employs the features of the standard graphic adapter like boot messaging, graphical booting, multiple virtual terminals, and can launch the X window system. You can also use the graphical keyboard to configure the virtual keyboard and mouse.

Guests can be identified in any of three **identities**: the domain name (**`domain-name`**), the identity number (**`domain-id`**), or the UUID. The **`domain-name`** is a text string that corresponds to a guest configuration file. Theguests name can be used to start guests, and when the guest runs that same name can identify and control it. The domain-id is a unique, non-persistent number that gets assigned to an active domain can identify and control that guest. The UUID is a persistent, unique identifier that is controlled from the guest's configuration file and ensures that the guest is identified over time by system management tools. It is visible to the guest when it runs. A new UUID is automatically assigned to each guest by the system tools when the guest first installs.

Appendix B. Additional resources

To learn more about virtualization and Red Hat Enterprise Linux, refer to the following resources.

B.1. Online resources

- *http://www.cl.cam.ac.uk/research/srg/netos/xen/* The project website of the Xen™ para-virtualization machine manager from which the Red Hat *kernel-xen* package is derived. The site maintains the upstream xen project binaries and source code and also contains information, architecture overviews, documentation, and related links regarding xen and its associated technologies.

- The Xen Community website

  http://www.xen.org/

- *http://www.libvirt.org/* is the official website for the **`libvirt`** virtualization API.

- *http://virt-manager.et.redhat.com/* is the project website for the **Virtual Machine Manager** (virt-manager), the graphical application for managing virtual machines.

- Open Virtualization Center

  http://www.openvirtualization.com[1]

- Red Hat Documentation

  http://www.redhat.com/docs/

- Virtualization technologies overview

  http://virt.kernelnewbies.org[2]

- Red Hat Emerging Technologies group

  http://et.redhat.com[3]

B.2. Installed documentation

- **`/usr/share/doc/xen-<version-number>/`** is the directory which contains information about the Xen para-virtualization hypervisor and associated management tools, including various example configurations, hardware-specific information, and the current Xen upstream user documentation.

- **`man virsh`** and **`/usr/share/doc/libvirt-<version-number>`** — Contains sub commands and options for the **`virsh`** virtual machine management utility as well as comprehensive information about the **`libvirt`** virtualization library API.

- **`/usr/share/doc/gnome-applet-vm-<version-number>`** — Documentation for the GNOME graphical panel applet that monitors and manages locally-running virtual machines.

- **`/usr/share/doc/libvirt-python-<version-number>`** — Provides details on the Python bindings for the **`libvirt`** library. The **`libvirt-python`** package allows python developers to create programs that interface with the **`libvirt`** virtualization management library.

- **/usr/share/doc/python-virtinst-<*version-number*>** — Provides documentation on the **virt-install** command that helps in starting installations of Fedora and Red Hat Enterprise Linux related distributions inside of virtual machines.
- **/usr/share/doc/virt-manager-<*version-number*>** — Provides documentation on the Virtual Machine Manager, which provides a graphical tool for administering virtual machines.

Glossary

This glossary is intended to define the terms used in this Installation Guide.

Bare-metal
: The term bare-metal refers to the underlying physical architecture of a computer. Running an operating system on bare-metal is another way of referring to running an unmodified version of the operating system on the physical hardware. Examples of operating systems running on bare metal are *dom0* or a normally installed operating system.

dom0
: Also known as the *Host* or host operating system.

  dom0 refers to the host instance of Red Hat Enterprise Linux running the *Hypervisor* which facilitates virtualization of guest operating systems. Dom0 runs on and manages the physical hardware and resource allocation for itself and the guest operating systems.

Domains
: *domU* and *Domains* are both domains. Domains run on the *Hypervisor*. The term domains has a similar meaning to *Virtual machines* and the two are technically interchangeable. A domain is a Virtual Machine.

domU
: **domU** refers to the guest operating system which run on the host system (*Domains*).

Full virtualization
: Xen and KVM can use full virtualization. Full virtualization uses hardware features of the processor to provide total abstraction of the underlying physical system (*Bare-metal*) and create a new virtual system in which the guest operating systems can run. No modifications are needed in the guest operating system. The guest operating system and any applications on the guest are not aware of the virtualized environment and run normally. Para-virtualization requires a modified version of the Linux operating system.

Fully virtualized
: See *Full virtualization*.

Guest system
: Also known as guests, virtual machines or *domU*.

Hardware Virtual Machine
: See *Full virtualization*

Hypervisor
: The hypervisor is the software layer that abstracts the hardware from the operating system permitting multiple operating systems to run on the same hardware. The hypervisor runs on a host operating system allowing other virtualized operating systems to run on the host's hardware.

Host
: The host operating system, also known as *dom0*.

  The host operating system environment runs the virtualization software for *Fully virtualized* and *Para-virtualized* guest systems.

I/O
: Short for input/output (pronounced "eye-oh"). The term I/O describes any program, operation or device that transfers data to or from a computer and to or from a peripheral device. Every transfer is an output from one device and an input into another. Devices such as

keyboards and mouses are input-only devices while devices such as printers are output-only. A writable CD-ROM is both an input and an output device.

Itanium®
: The Intel Itanium® processor architecture.

Kernel-based Virtual Machine
: KVM (Kernel-based Virtual Machine) is a *Full virtualization* solution for Linux on AMD64 and Intel 64 hardware. VM is a Linux kernel module built for the standard Red Hat Enterprise Linux kernel. KVM can run multiple, unmodified virtualized guest Windows and Linux operating systems. KVM is a hypervisor which uses the libvirt virtualization tools (virt-manager and virsh).

KVM is a set of Linux kernel modules which manage devices, memory and management APIs for the Hypervisor module itself. Virtualized guests are run as Linux processes and threads which are controlled by these modules.

LUN
: A Logical Unit Number (LUN) is a number assigned to a logical unit (a SCSI protocol entity).

Migration
: Migration is name for the process of moving a virtualized guest from one host to another. Migration can be conducted offline (where the guest is suspended and then moved) or live (where a guest is moved without suspending). Xen fully virtualized guests, Xen para-virtualized guest and KVM fully virtualized guests can all be migrated.

Migration is a key feature of virtualization as software is completely separated from hardware. Migration is useful for:

- Load balancing - guests can be moved to hosts with lower usage when a host becomes overloaded.
- Hardware failover - when hardware devices on the host start to fail, guests can be safely relocated so the host can be powered down and repaired.
- Energy saving - guests can be redistributed to other hosts and host systems powered off to save energy and cut costs in low usage periods.
- Geographic migration - guests can be moved to another location for lower latency or in serious circumstances.

Shared, networked storage is used for storing guest images. Without shared storage migration is not possible.

An offline migration suspends the guest then moves an image of the guests memory to the destination host. The guest is resumed on the destination host and the memory the guest used on the source host is freed.

The time an offline migration takes depends network bandwidth and latency. A guest with 2GB of memory should take several seconds on a 1 Gbit Ethernet link.

A live migration keeps the guest running on the source host and begins moving the memory without stopping the guest. All modified memory pages are tracked and sent to the destination after the image is sent. The memory is updated with the changed pages. The process continues until it reaches some heuristic; either it successfully copied all the pages over, or the source is changing too fast and the destination host cannot make progress. If the heuristic is met the guest is briefly paused on the source host and the registers and buffers are sent. The registers are loaded on the new host and the guest is then resumed on the destination host. If the guest cannot be merged (which happens when guests are under extreme loads) the guest is paused and then an offline migration is started instead.

The time an offline migration takes depends network bandwidth and latency as well as activity on the guest. If the guest is using significant I/O or CPU the migration will take much longer.

MAC Addresses
: The Media Access Control Address is the hardware address for a Network Interface Controller. In the context of virtualization MAC addresses must be generated for virtual network interfaces with each MAC on your local domain being unique.

Para-virtualization
: Para-virtualization uses a special kernel, sometimes referred to as the Xen kernel or the *kernel-xen* package. Para-virtualized guest kernels are run concurrently on the host while using the host's libraries and devices. A para-virtualized installation can have complete access to all devices on the system which can be limited with security settings (SELinux and file controls). Para-virtualization is faster than full virtualization. Para-virtualization can effectively be used for load balancing, provisioning, security and consolidation advantages.

As of Fedora 9 a special kernel will no longer be needed. Once this patch is accepted into the main Linux tree all Linux kernels after that version will have para-virtualization enabled or available.

Para-virtualized
: See *Para-virtualization*,

Para-virtualized drivers
: Para-virtualized drivers are device drivers that operate on fully virtualized Linux guests. These drivers greatly increase performance of network and block device I/O for fully virtualized guests.

Security Enhanced Linux
: Short for Security Enhanced Linux, SELinux uses Linux Security Modules (LSM) in the Linux kernel to provide a range of minimum privilege required security policies.

Universally Unique Identifier
: A Universally Unique Identifier (UUID) is a standardized numbering method for devices, systems and certain software objects in distributed computing environments. Types of UUIDs in virtualization include: `ext2` and `ext3` file system identifiers, RAID device identifiers, iSCSI and LUN device identifiers, MAC addresses and virtual machine identifiers.

Virtualization
: Virtualization is a board computing term for running software, usually operating systems, concurrently and isolated from other programs

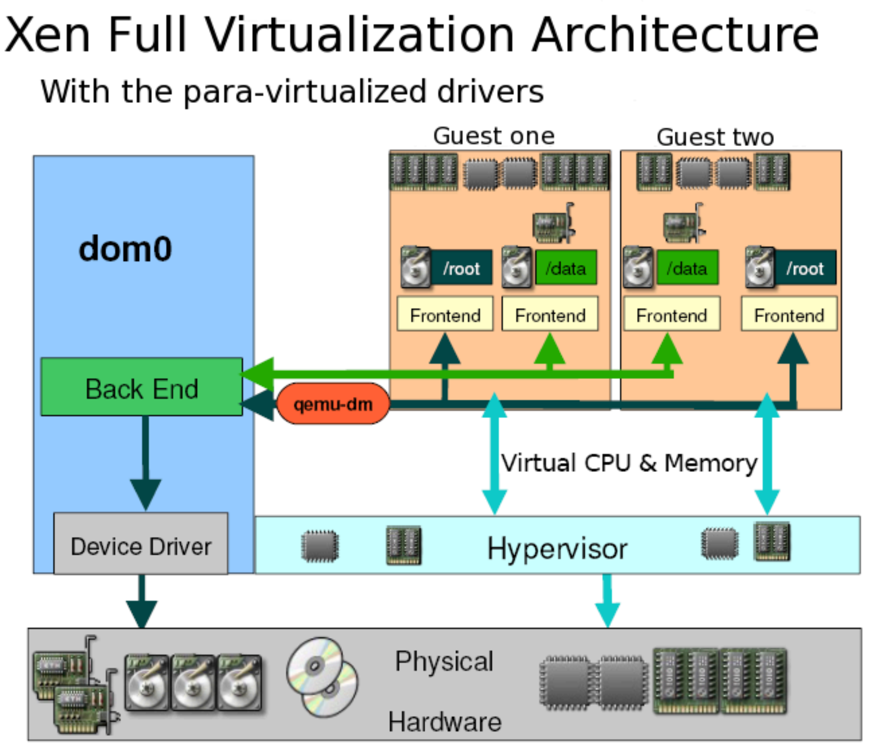
on one system. Most existing implementations of virtualization use a hypervisor, a software layer on top of an operating system, to abstract hardware. The hypervisor allows multiple operating systems to run on the same physical system by giving the guest operating system virtualized hardware. There are various methods for virtualizing operating systems:

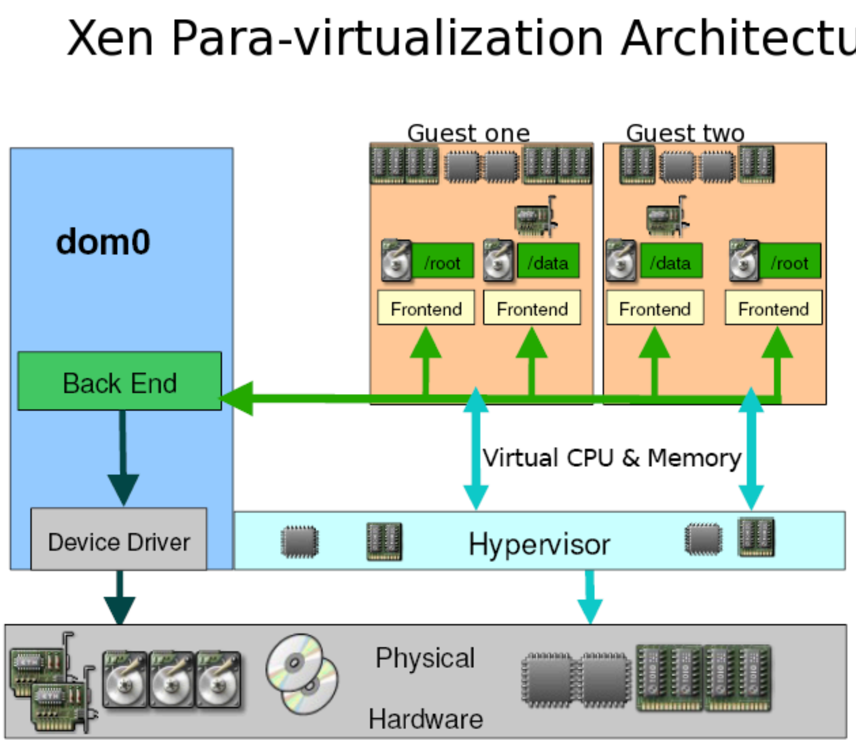
- Hardware-assisted virtualization is the technique used for full virtualization with Xen and KVM (definition: *Full virtualization*)
- Para-virtualization is a technique used by Xen to run Linux guests (definition: *Para-virtualization*)
- Software virtualization or emulation. Software virtualization uses binary translation and other emulation techniques to run unmodified operating systems. Software virtualization is significantly slower than hardware-assisted virtualization or para-virtualization. Software virtualization, in the form of QEMU, is unsupported by Red Hat Enterprise Linux.

Virtualized CPU
: A system has a number of virtual CPUs (VCPUs) relative to the number of physical processor cores. The number of virtual CPUs is finite and represents the total number of virtual CPUs that can be assigned to guest virtual machines.

Virtual machines
: A virtual machine is a software implementation of a physical machine or programming language (for example the Java Runtime Environment or LISP). Virtual machines in the context of virtualization are operating systems running on virtualized hardware.

Xen
: Red Hat Enterprise Linux supports the Xen hypervisor and the KVM hypervisor (refer to *Kernel-based Virtual Machine*). Both hypervisors have different architectures and development approaches. The Xen hypervisor runs underneath a Red Hat Enterprise Linux operating system which acts as a host managing system resources and virtualization APIs. The host is sometimes referred to as as *dom0* or Domain0.


Xen Full Virtualization Architecture
With the para-virtualized drivers
Guest one
Guest two
dom0
/root
/data
/data
/root
Frontend
Frontend
Frontend
Frontend
Back End
qemu-dm
Virtual CPU & Memory
Device Driver
Hypervisor
Physical
Hardware


Xen Para-virtualization Architectu
Guest one
Guest two
dom0
/root
/data
/data
/root
Frontend
Frontend
Frontend
Frontend
Back End
Virtual CPU & Memory
Device Driver
Hypervisor
Physical
Hardware

Appendix C. Revision History

Revision 5.4-64 | Wed Oct 07 2009 | Christopher Curran ccurran@redhat.com

Fixes VCPU allocation bug.

Revision 5.4-61 | Wed Sep 23 2009 | Christopher Curran ccurran@redhat.com

Fixes BZ#524350, BZ#508606 and BZ#517221
Expands and corrects Remote management, timing management and para-virtualized drivers.
Adds localization credits.

Revision 5.4-59 | Thu Sep 17 2009 | Christopher Curran ccurran@redhat.com

Fixes bug in the Remote Management chapter.

Revision 5.4-58 | Tue Sep 15 2009 | Christopher Curran ccurran@redhat.com

Fixes various errors.

Revision 5.4-55 | Fri Sep 11 2009 | Christopher Curran ccurran@redhat.com

Fixes BZ#510058 and BZ#513651

Revision 5.4-53 | Tue Sep 08 2009 | Christopher Curran ccurran@redhat.com

Updated restrictions, Expanded KVM virtio drivers, and various edits.

Revision 5.4-51 | Tue Aug 25 2009 | Christopher Curran ccurran@redhat.com

Fixed restrictions and various edits.

Revision 5.4-48 | Thu Aug 20 2009 | Christopher Curran ccurran@redhat.com

New PXE installation section.

Revision 5.4-47 | Wed Aug 19 2009 | Christopher Curran ccurran@redhat.com

Updated restrictions (BZ#517439) and default image storage location (BZ#514117).

Revision 5.4-46 | Thu Aug 13 2009 | Christopher Curran *ccurran@redhat.com*

Updated limitations and para-virtualized drivers content.

Revision 5.4-45 | Mon Aug 10 2009 | Christopher Curran *ccurran@redhat.com*

Updated the KVM para-virtualized drivers section.

Revision 5.4-44 | Thu Aug 06 2009 | Christopher Curran *ccurran@redhat.com*

32 bugs fixed.

Revision 5.4-42 | Tue Aug 04 2009 | Christopher Curran *ccurran@redhat.com*

New Red Hat Enterprise Linux installation chapter.

Revision 5.4-41 | Mon Aug 03 2009 | Christopher Curran *ccurran@redhat.com*

Live migration chapter completed. Various other bug fixes and content updates.

Revision 5.4-38 | Thu Jul 30 2009 | Christopher Curran *ccurran@redhat.com*

New storage section. Updated live migrations. 11 bug fixes.

Revision 5.4-36 | Thu Jul 30 2009 | Christopher Curran *ccurran@redhat.com*

Various edits.

Revision 5.4-35 | Wed Jul 29 2009 | Christopher Curran *ccurran@redhat.com*

Expanded live migration for KVM. Various screen shot errors resolved. Updated the Xen live migration section.

Revision 5.4-33 | Tue Jul 28 2009 | Christopher Curran *ccurran@redhat.com*

Fixes *BZ #513887*[1].

Revision 5.4-32 | Mon Jul 27 2009 | Christopher Curran *ccurran@redhat.com*

Added a hypervisor switching section.

Revision 5.4-31	Fri Jul 24 2009	Christopher Curran ccurran@redhat.com

Installation chapter updated for KVM.

Revision 5.4-29	Thu Jul 23 2009	Christopher Curran ccurran@redhat.com

KVM support and restrictions updates.

Revision 5.4-26	Fri Jul 17 2009	Christopher Curran ccurran@redhat.com

New Windows Server 2008 installation section.

Revision 5.4-24	Wed Jun 24 2009	Christopher Curran ccurran@redhat.com

Various copy edits and corrections.

Revision 5.4-23	Mon May 11 2009	Christopher Curran ccurran@redhat.com

Fixes restrictions errors.

Revision 5.4-20	Mon Mar 09 2009	Christopher Curran ccurran@redhat.com

Resolves 487407 and minor copy edits.

Revision 5.3-19	Mon Feb 23 2009	Christopher Curran ccurran@redhat.com

Resolves 486294 and minor copy edits.

Revision 5.2-18	Tue Jan 20 2009	Christopher Curran ccurran@redhat.com

Resolves various bugs and other documentation fixes including:
Resolves: BZ #461440
Resolves: BZ #463355
Fixes various spelling and typographic errors

Revision 5.2-16	Mon Jan 19 2009	Christopher Curran ccurran@redhat.com

Resolves various bugs and other documentation fixes including:
Resolves: BZ #469300
Resolves: BZ #469316
Resolves: BZ #469319
Resolves: BZ #469326
Resolves: BZ #444918

Resolves: BZ #449688
Resolves: BZ #479497

Revision 5.2-14	Tue Nov 18 2008	Christopher Curran *ccurran@redhat.com*

Resolves various bugs and other documentation fixes including:
Resolves: BZ #469300
Resolves: BZ #469314
Resolves: BZ #469319
Resolves: BZ #469322
Resolves: BZ #469326
Resolves: BZ #469334
Resolves: BZ #469341
Resolves: BZ #371981
Resolves: BZ #432235
Resolves: BZ #432394
Resolves: BZ #441149
Resolves: BZ #449687
Resolves: BZ #449688
Resolves: BZ #449694
Resolves: BZ #449704
Resolves: BZ #449710
Resolves: BZ #454706

Revision 5.2-11	Fri Aug 01 2008	Christopher Curran *ccurran@redhat.com*

Resolves: BZ #449681
Resolves: BZ #449682
Resolves: BZ #449683
Resolves: BZ #449684
Resolves: BZ #449685
Resolves: BZ #449689
Resolves: BZ #449691
Resolves: BZ #449692
Resolves: BZ #449693
Resolves: BZ #449695
Resolves: BZ #449697
Resolves: BZ #449699
Resolves: BZ #449700
Resolves: BZ #449702
Resolves: BZ #449703
Resolves: BZ #449709
Resolves: BZ #449711
Resolves: BZ #449712 - Various typos and spelling mistakes.
Resolves: BZ #250272
Resolves: BZ #251778
Resolves: BZ #285821
Resolves: BZ #322761
Resolves: BZ #402161

Resolves: BZ #422541
Resolves: BZ #426954
Resolves: BZ #427633
Resolves: BZ #428371
Resolves: BZ #428917
Resolves: BZ #428958
Resolves: BZ #430852
Resolves: BZ #431605
Resolves: BZ #448334
Resolves: BZ #449673
Resolves: BZ #449679
Resolves: BZ #449680

Revision 5.2-10	Wed May 14 2008	Christopher Curran `ccurran@redhat.com`

New or rewritten sections for installation, troubleshooting, networking and installation
Various updates for spelling, grammar and language
Formatting and layout issues resolved
Updated terminology and word usage to enhance usability and readability

Revision 5.2-9	Mon Apr 7 2008	Christopher Curran `ccurran@redhat.com`

Book updated to remove redundant chapters and headings
Virtual Machine Manger updated for 5.1.

Revision 5.2-7	Mon Mar 31 2008	Christopher Curran `ccurran@redhat.com`

Resolves: #322761
Many spelling and grammar errors corrected.
Chapter on Remote Management added.

Revision 5.2-5	Wed Mar 19 2008	Christopher Curran `ccurran@redhat.com`

Resolves: #428915
New Virtualization Guide created.

Appendix D. Colophon

This manual was written in the DocBook XML v4.3 format.

This book is based on the work of Jan Mark Holzer and Chris Curran.

Other writing credits go to:

- Don Dutile contributed technical editing for the para-virtualized drivers section.
- Barry Donahue contributed technical editing for the para-virtualized drivers section.
- Rick Ring contributed technical editing for the Virtual Machine Manager Section.
- Michael Kearey contributed technical editing for the sections on using XML configuration files with virsh and virtualized floppy drives.
- Marco Grigull contributed technical editing for the software compatibility and performance section.
- Eugene Teo contributed technical editing for the Managing Guests with virsh section.

Publican, the publishing tool which produced this book, was written by Jeffrey Fearn.

The Red Hat Localization Team consists of the following people:

East Asian Languages

- Simplified Chinese
  - Leah Wei Liu
- Traditional Chinese
  - Chester Cheng
  - Terry Chuang
- Japanese
  - Junko Ito
- Korean
  - Eun-ju Kim

Latin Languages

- French
  - Sam Friedmann
- German
  - Hedda Peters
- Italian

  - Francesco Valente
- Brazilian Portuguese
  - Glaucia de Freitas
  - Leticia de Lima
- Spanish
  - Angela Garcia
  - Gladys Guerrero
- Russian
  - Yuliya Poyarkova