

**Red Hat Enterprise Linux**

# **Virtualization Guide**

**5.2**



**Christopher Curran**

**Jan Mark Holzer**

**ISBN:**

**Publication date:**

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.

---

# Red Hat Enterprise Linux: Virtualization Guide

Author	Christopher Curran	<ccurran@redhat.com>
Author	Jan Mark Holzer	<jmh@redhat.com>
Translator	Don Dutile	
Translator	Barry Donahue	
Translator	Rick Ring	
Translator	Michael Kearey	
Translator	Marco Grigull	
Translator	Eugene Teo	

Copyright © 2008 Red Hat, Inc

Copyright © 2008 Red Hat, Inc. This material may only be distributed subject to the terms and conditions set forth in the Open Publication License, V1.0 or later with the restrictions noted below (the latest version of the OPL is presently available at <http://www.opencontent.org/openpub/>).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

Red Hat and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

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

---



---

How should CIO's Think about Virtualization? .....	ix
1. About this book .....	xi
2. Document Conventions .....	xi
3. We Need Feedback .....	xiii
I. System Requirements for Red Hat Enterprise Linux Virtualization .....	1
1. System requirements .....	3
1. Hardware prerequisites .....	3
2. Compatibility of host and guest combinations .....	7
3. Virtualization limitations .....	9
II. Installation Procedures .....	11
Installing Red Hat Enterprise Linux Virtualization .....	xiii
4. Installing Red Hat Virtualization packages on the host .....	15
1. Installing Red Hat Virtualization during a new Red Hat Enterprise Linux installation .....	15
2. Installing Red Hat Virtualization on an existing Red Hat Enterprise Linux system .....	15
5. Installing guests .....	19
1. Create a guest using <code>virt-install</code> .....	19
2. Create a guest using <code>virt-manager</code> .....	21
6. Guest operating system installation processes .....	31
1. Installing Red Hat Enterprise Linux 5 as a para-virtualized guest .....	31
1.1. Graphical Red Hat Enterprise Linux 5 installation .....	39
1.2. The first boot after the guest installation of Red Hat Enterprise Linux 5 .....	54
1.3. First boot configuration .....	56
2. Installing a Windows XP Guest as a fully virtualized guest .....	74
3. Installing a Windows 2003 SP1 Server Guest as a fully-virtualized guest ..	89
III. Configuration .....	93
Configuring Red Hat Enterprise Linux Virtualization .....	xcv
7. Virtualized block devices .....	97
1. Installing a virtualized floppy disk controller .....	97
2. Adding additional storage devices to a guest .....	98
3. Configuring persistent storage in a Red Hat Enterprise Linux 5 environment .....	100
4. Adding an ISO file as a CD-ROM to a guest configuration file .....	102
8. Configuring networks and guests .....	103
9. Server best practices .....	105
10. Securing the host .....	107
11. SELinux and virtualization .....	109
12. Virtualized network devices .....	111
1. Configuring multiple guest network bridges to use multiple ethernet cards	111
2. Laptop network configuration .....	112
13. Introduction to Para-virtualized Drivers .....	117
1. System requirements .....	118
2. Para-virtualization Restrictions and Support .....	119
3. Installation and Configuration of Para-virtualized Drivers .....	122

---

3.1. Common installation steps .....	123
3.2. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 3 .....	124
3.3. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 4 .....	128
3.4. Installation and Configuration of Para-virtualized Drivers on Red Hat Enterprise Linux 5 .....	132
4. Para-virtualized Network Driver Configuration .....	134
5. Additional Para-virtualized Hardware Configuration .....	138
5.1. Virtualized Network Interfaces .....	138
5.2. Virtual Storage Devices .....	139
IV. Administration .....	141
Administering Red Hat Enterprise Linux Virtualization .....	cxliii
14. Starting or stopping a domain during the boot phase .....	145
15. Managing guests with <code>xend</code> .....	147
16. Managing CPUs .....	151
17. Virtualization live migration .....	153
1. A live migration example .....	155
18. Remote management of virtualized guests .....	165
1. Remote management with <code>ssh</code> .....	165
2. Remote management over TLS and SSL .....	166
V. Virtualization Reference Guide .....	169
Tools Reference Guide for Red Hat Enterprise Linux Virtualization .....	clxxi
19. Red Hat Virtualization tools .....	173
20. Managing guests with <code>virsh</code> .....	177
21. Managing guests with Virtual Machine Manager( <code>virt-manager</code> ) .....	185
1. Virtual Machine Manager Architecture .....	185
2. The open connection window .....	185
3. The Virtual Machine Manager main window .....	185
4. The Virtual Machine Manager details window .....	186
5. Virtual Machine graphical console .....	186
6. Starting <code>virt-manager</code> .....	186
7. Creating a new guest .....	187
8. Restoring a saved machine .....	197
9. Displaying guest details .....	198
10. Status monitoring .....	199
11. Displaying domain ID .....	200
12. Displaying a guest's status .....	200
13. Displaying virtual CPUs .....	201
14. Displaying CPU usage .....	201
15. Displaying memory usage .....	202
16. Managing a virtual network .....	202
17. Creating a virtual network .....	202
22. Commands for Red Hat Virtualization .....	205
1. <code>virsh</code> the command line interface tool for virtualization .....	205
2. The <code>xm</code> command line interface .....	208
23. Configuring GRUB .....	211

---

24. Configuring ELILO .....	213
25. Configuration files .....	217
VI. Tips and Tricks .....	227
Tips and Tricks to Enhance Productivity .....	ccxxix
26. Tips and tricks .....	231
1. Automatically starting domains during the host system boot .....	231
2. Modifying <code>/etc/grub.conf</code> .....	231
3. Example guest configuration files and parameters .....	232
4. Duplicating an existing guest and its configuration file .....	233
5. Identifying guest type and implementation .....	234
6. Generating a new unique MAC address .....	235
7. Limit network bandwidth for a guest .....	236
8. Starting domains automatically during system boot .....	237
9. Modifying <code>dom0</code> .....	237
10. Configuring guest live migration .....	238
11. Very Secure <code>ftpd</code> .....	239
12. Configuring LUN Persistence .....	240
13. Disable SMART disk monitoring for guests .....	242
14. Cleaning up the <code>/var/lib/xen/</code> folder .....	242
15. Configuring a VNC Server .....	242
16. Cloning guest configuration files .....	243
27. Creating custom Red Hat Virtualization scripts .....	245
1. Using XML configuration files with <code>virsh</code> .....	245
28. Compiling para-virtualized driver packages from source code .....	247
VII. Troubleshooting .....	249
Introduction to Troubleshooting and Problem Solving .....	ccli
29. How To troubleshoot Red Hat Virtualization .....	253
1. Debugging and troubleshooting Red Hat Virtualization .....	253
2. Log files overview .....	255
3. Log file descriptions .....	256
4. Important directory locations .....	256
5. Troubleshooting with the logs .....	257
6. Troubleshooting with the serial console .....	257
7. Para-virtualized guest console access .....	258
8. Fully virtualized guest console access .....	259
9. SELinux considerations .....	259
10. Accessing data on guest disk image .....	259
11. Common troubleshooting situations .....	260
12. Guest creation errors .....	261
13. Serial console errors .....	261
14. Network bridge errors .....	262
15. Guest configuration files .....	263
16. Interpreting error messages .....	264
17. The layout of the log directories .....	267
18. Online troubleshooting resources .....	268
30. Troubleshooting .....	271
1. Identifying available storage and partitions .....	271

---

2. Virtualized ethernet devices are not found by networking tools .....	271
3. Loop device errors .....	271
4. Failed domain creation caused by a memory shortage .....	271
5. Wrong kernel image error - using a non-Xen kernel in a para-virtualized guest .....	272
6. Wrong kernel image error - non-PAE kernel on a PAE platform .....	272
7. Fully-virtualized x86_64 guest fails to boot .....	273
8. Missing localhost entry in <code>/etc/hosts</code> causing <code>virt-manager</code> to fail .....	273
9. Microcode error during guest boot .....	274
10. Wrong bridge configured on guest causing Xen hot plug scripts to timeout .....	274
11. Python depreciation warning messages when starting a virtual machine .....	275
31. Troubleshooting Para-virtualized Drivers .....	277
1. Red Hat Enterprise Linux 5 Virtualization log file and directories .....	277
2. Para-virtualized guest fail to load on a Red Hat Enterprise Linux 3 guest operating system .....	279
3. A warning message is displayed while installing the para-virtualized drivers on Red Hat Enterprise Linux 3 .....	279
4. What to do if the guest operating system has been booted with <code>virt-manager</code> or <code>virsh</code> .....	280
5. Manually loading the para-virtualized drivers .....	282
6. Verifying the para-virtualized drivers have successfully loaded .....	283
7. The system has limited throughput with para-virtualized drivers .....	283
A. Revision History .....	285
B. Red Hat Virtualization system architecture .....	287
C. Additional resources .....	289
1. Online resources .....	289
2. Installed documentation .....	289
Glossary .....	291



---

## How should CIO's Think about Virtualization?

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 aren't 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/>

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

## 1. About this book

This book is divided into 7 parts:

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

## 2. Document Conventions

Certain words in this manual are represented in different fonts, styles, and weights. This highlighting indicates that the word is part of a specific category. The categories include the following:

Courier font

Courier font represents commands, file names and paths, and prompts.

When shown as below, it indicates computer output:

```
Desktop      about.html    logs          paulwesterberg.png
Mail         backupfiles   mail          reports
```

**Courier font**

Bold Courier font represents text that you are to type, such as: **service jonas start**

If you have to run a command as root, the root prompt (#) precedes the command:

```
# gconftool-2
```

*italic Courier font*

Italic Courier font represents a variable, such as an installation directory:

`install_dir/bin/`

### **bold font**

Bold font represents **application programs** and **text found on a graphical interface**.

When shown like this: **OK** , it indicates a button on a graphical application interface.

Additionally, the manual uses different strategies to draw your attention to pieces of information. In order of how critical the information is to you, these items are marked as follows:



### **Note**

A note is typically information that you need to understand the behavior of the system.



### **Tip**

A tip is typically an alternative way of performing a task.



### **Important**

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.



### **Caution**

A caution indicates an act that would violate your support agreement, such as recompiling the kernel.



### **Warning**

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

### 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 Busily: <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.



---

# **Part I. System Requirements for Red Hat Enterprise Linux Virtualization**

---





# System requirements

Your system will require the attributes listed in this chapter to successfully run virtualization on Red Hat Enterprise Linux. You will require a host running Red Hat Enterprise Linux 5 Server with the virtualization packages. The host will need a configured *hypervisor*. For information on installing the hypervisor, read [Chapter 4, Installing Red Hat Virtualization packages on the host](#).

You require installation media for the *guest systems*. The installation media must be available to the host as follows:

- for para-virtualized guests you require the Red Hat Enterprise Linux 5 installation tree available over NFS, FTP or HTTP.
- for fully-virtualized guest installations you will require DVD or CD-ROM distribution media or a bootable .iso file and a network accessible installation tree.

The types of storage supported by Red Hat Virtualization are:

- a file
- a physical disk partition
- a locally connected physical LUN
- an LVM partition
- or as an iSCSI or Fibre Channel based LUN.



## Tip

Use `/var/lib/xen/images/` for file based guest images. If you use a different directory you must add the directory to your SELinux policy. For more information see [Chapter 11, SELinux and virtualization](#).

## 1. Hardware prerequisites

### Hardware requirements for para-virtualization and full virtualization

The following list is the recommended RAM and disk space for *each* para-virtualized or fully virtualized guest

- a minimum of 2 GB of available RAM.

- a minimum of 6 GB of free disk space.

It is advised to have at least one processing core or hyper-thread for each virtual machine.

Your system will also require hardware virtualization extensions to use fully virtualized guest operating systems. The steps to identify whether your system has virtualization extensions can be found at [Hardware virtualization extensions](#).

### Hardware virtualization extensions.

Full virtualization requires CPUs with hardware virtualization extensions. This section describes how to identify hardware virtualization extensions and enable them in your BIOS if they are disabled. If hardware virtualization extensions are not present you can only use para-virtualization with Red Hat Virtualization.

The virtualization extensions can not be disabled in the BIOS for AMD-V capable processors installed in a Rev 2 socket. The Intel® VT extensions can be disabled in the BIOS. Certain laptop vendors have disabled the Intel® VT extensions by default in their CPUs.

These instructions enable Intel® VT virtualization extensions if they are disabled in BIOS:

1. Run the `xm dmesg | grep VMX` command. The output should display as follows:

```
(XEN) VMXON is done
(XEN) VMXON is done
```

2. Run the `cat /proc/cpuinfo | grep vmx` command to verify the CPU flags have been set. The output should be similar to the following. Note `vmx` in the output:

```
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
```

Do not proceed if the output of `xm dmesg | grep VMX` is not `VMXON is done` for each CPU. Please visit the BIOS if other messages are reported.

The following commands verify that the virtualization extensions are enabled on AMD-V architectures:

1. Run the `xm dmesg | grep SVM` command. The output should look like the following:

```
(XEN) AMD SVM Extension is enabled for cpu 0
```

```
(XEN) AMD SVM Extension is enabled for cpu 1
```

2. Run the `cat /proc/cpuinfo | grep svm` to verify the CPU flags have been set. The output should be similar to the following. Note `svm` in the output:

```
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
```

### Enabling Intel® VT and AMD-V virtualization hardware extensions.

To run unmodified guest operating systems (also known as fully virtualized or [HVM](#) guests) you require a Intel® VT or AMD-V capable system. The steps below will provide an overview about how you can verify whether the virtualization extensions have been enabled, and that Red Hat Enterprise Linux 5 can use them:

First, verify the Intel® VT or AMD-V capabilities are enabled via the BIOS. The BIOS settings for Intel® VT or AMD-V are usually in the **Chipset** or **Processor** menus. However, they can sometimes be hidden under obscure menus, such as **Security Settings** or other non standard menus.

For Intel® VT architectures perform these steps to make sure Intel Virtualization Technology is enabled. Some menu items may have slightly different names:

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. The `vmx` or `svm` instructions should now be enabled.



### Important

After resetting or updating the BIOS you must reboot the system for the updated settings to take effect.

# Compatibility of host and guest combinations

The table below shows the available architecture combinations for hosts and guests.

The table [Table 2.1, “Host and guest architecture compatibility”](#)

Para-virtualized or fully virtualized	Host Architecture	Guest Architecture
Para-virtualized	i686	i686
Fully virtualized	i686	i686
Fully virtualized	x86_64	i686
Para-virtualized	x86_64	x86_64
Fully virtualized	x86_64	x86_64
Para-virtualized	ia64	ia64
Fully virtualized	ia64	ia64

**Table 2.1. Host and guest architecture compatibility**



## Itanium® support

Red Hat Virtualization on the Itanium® architecture requires the guest firmware image, refer to [Installing Red Hat Virtualization with yum](#) for more information.



## Virtualization limitations

This chapter covers the limitations of virtualization in Red Hat Enterprise Linux. There are several aspects of virtualization which make virtualized guests unsuitable for certain types of application.

The following Red Hat Enterprise Linux platforms on para-virtualized guests are unable to subscribe to RHN for the additional services:

- RHN Satellite
- RHN Proxy

It is possible to configure these as fully virtualized guests it should be avoided due to the high I/O requirements imposed by these. This impact may be mitigated by full support of hardware virtualization I/O extensions in future Z-stream releases of Red Hat Virtualization.

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

- **kdump** server
- **netdump** server

You should carefully scrutinize databased 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 (see [Chapter 13, Introduction to Para-virtualized Drivers](#)).

Other platforms and software applications that heavily utilize I/O or real-time should be evaluated carefully. Using full virtualization with the para-virtualized drivers (see [Chapter 13, Introduction to Para-virtualized Drivers](#)) or para-virtualization will result in better performance with I/O intensive applications. However, applications will always suffer some performance degradation running on virtualized environments. The performance benefits of virtualization through consolidating to newer and faster hardware should not be underestimated as they will often outweigh the potential disadvantages of moving to a virtualized platform.



### 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 will be a performance hit in using virtualization with increased I/O usage.



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



---

## **Part II. Installation Procedures**

---



---

## **Installing Red Hat Enterprise Linux Virtualization**

These chapters provide the information for installing host and guest systems utilizing Red Hat Virtualization. These chapters will provide you with the information to get started using Red Hat Virtualization.



# Installing Red Hat Virtualization packages on the host

The virtualization packages, for the host, must be installed on Red Hat Enterprise Linux to utilize Red Hat Virtualization. If the virtualization packages are not present you will need to install them or the instructions in this guide will not work. You can install the necessary host packages

## 1. Installing Red Hat Virtualization during a new Red Hat Enterprise Linux installation

### **Adding Red Hat Virtualization packages to a Kickstart file.**

This section describes how to use kickstart files to install Red Hat Enterprise Linux. 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 Red Hat Virtualization packages.

More information on kickstart files can be found on Red Hat's website, [redhat.com](http://redhat.com)<sup>1</sup>, in the *Installation Guide* for your Red Hat Enterprise Linux version.

## 2. Installing Red Hat Virtualization on an existing Red Hat Enterprise Linux system

The section describes the steps necessary to install Red Hat Virtualization on a working copy of Red Hat Enterprise Linux.

### **Adding packages to your list of Red Hat Network entitlements.**

This section will describe the procedure for enabling Red Hat Network (RHN) entitlements to install the Red Hat Virtualization packages on your systems. You need to have these entitlements enabled and the packages installed to set up host systems.

1. Log in to [RHN](https://rhn.redhat.com/)<sup>2</sup> using your RHN username and password.



**RHN registration**

---

<sup>1</sup> <http://www.redhat.com/docs/manuals/enterprise/>

<sup>2</sup> <https://rhn.redhat.com/>

You machines must be registered with Red Hat Network and you require a valid Red Hat Network account in order to install Red Hat Virtualization on Red Hat Enterprise Linux.

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](#)<sup>3</sup>.

2. Select the systems you want to install Red Hat 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 Red Hat Virtualization packages. The next section covers installing these packages.

### Installing Red Hat Virtualization with `yum`.

To commence using virtualization on Red Hat Enterprise Linux you will need the `xen` and `kernel-xen` packages. The `xen` package contains the Xen hypervisor and Xen tools. The `kernel-xen` package contains a modified linux kernel which runs as a virtual machine guest on the Xen hypervisor.

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

```
# 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
```

### Other recommended virtualization packages:

`python-virtinst`

Provides the `virt-install` command for creating 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` library to use the Xen virtualization framework.

### `libvirt`

`libvirt` is an API library which uses the Xen virtualization framework, and the `virsh` command line tool to manage and control virtual machines.

### `virt-manager`

Virtual Machine Manager provides a graphical tool for administering virtual machines. It uses `libvirt` library as the management API.

To install the other recommended virtualization packages, use the command below:

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





# Installing guests

This chapter will guide you through the process for installing guest virtual machines.

These listed of questions are important to consider and note before commencing the installation process on the host. Th

- are you using DHCP or a static configuration?
- will IPv6 support be enabled?
- is the hostname static or set via DHCP?

Start the installation process using either the **New** button in **virt-manager** or use the command line interface `virt-install`.

## 1. Create a guest using `virt-install`

Instead of **virt-manager** you can use the `virt-install` command. `virt-install` can either be used interactively or in a script to automate the creation of virtual machines. Using `virt-install` with Kickstart files an unattended installation of virtual machines can be achieved.

If you are using the `virt-install` CLI command and you select the `--vnc` option for a graphical installation you will also see the graphical installation screen as shown below.

The `virt-install` script provides a number of options one can pass on the command line. Below is the output from `virt-install -help`:

```
virt-install -help
usage: virt-install [options]
options:
  -h, --help                show this help message and exit
  -n NAME, --name=NAME      Name of the guest instance
  -r MEMORY, --ram=MEMORY   Memory to allocate for guest instance in megabytes
  -u UUID, --uuid=UUID      UUID for the guest; if none is given a random UUID
                             will be generated. If you specify UUID, you should
use
                             a 32-digit hexadecimal number.
  --vcpus=VCPUS             Number of vcpus to configure for your guest
  --check-cpu               Check that vcpus do not exceed physical CPUs and
warn
                             if they do.
  --cpuset=CPUSET           Set which physical CPUs Domain can use.
  -f DISKFILE, --file=DISKFILE
                             File to use as the disk image
  -s DISKSIZE, --file-size=DISKSIZE
                             Size of the disk image (if it doesn't exist) in
                             gigabytes
```

```
--nonsparse          Don't use sparse files for disks. Note that this
will                be significantly slower for guest creation
--nodisks            Don't set up any disks for the guest.
-m MAC, --mac=MAC    Fixed MAC address for the guest; if none or RANDOM
is                  given a random address will be used
-b BRIDGE, --bridge=BRIDGE
                    Bridge to connect guest NIC to; if none given, will
                    try to determine the default
-w NETWORK, --network=NETWORK
                    Connect the guest to a virtual network, forwarding
to                  the physical network with NAT
--vnc                Use VNC for graphics support
--vncport=VNCPORT    Port to use for VNC
--sdl                Use SDL for graphics support
--nographics         Don't set up a graphical console for the guest.
--noautoconsole       Don't automatically try to connect to the guest
                    console
-k KEYMAP, --keymap=KEYMAP
                    set up keymap for a graphical console
--accelerat          Use kernel acceleration capabilities
--accelerate         Connect to hypervisor with URI
--connect=CONNECT    Specify the CDROM media is a LiveCD
--lived              This guest should be a fully virtualized guest
-v, --hvm            File to use a virtual CD-ROM device for fully
-c CDROM, --cdrom=CDROM virtualized guests
--pxe                Boot an installer from the network using the PXE
boot                protocol
--os-type=OS_TYPE    The OS type for fully virtualized guests, e.g.
                    'linux', 'unix', 'windows'
--os-variant=OS_VARIANT
                    The OS variant for fully virtualized guests, e.g.
                    'fedora6', 'rhel5', 'solaris10', 'win2k', 'vista'
--noapic             Disables APIC for fully virtualized guest
(overrides          value in os-type/os-variant db)
--noacpi             Disables ACPI for fully virtualized guest
(overrides          value in os-type/os-variant db)
--arch=ARCH          The CPU architecture to simulate
-p, --paravirt       This guest should be a paravirtualized guest
-l LOCATION, --location=LOCATION
                    Installation source for paravirtualized guest (eg,
                    nfs:host:/path, http://host/path, ftp://host/path)
-x EXTRA, --extra-args=EXTRA
                    Additional arguments to pass to the installer with
                    paravirt guests
-d, --debug          Print debugging information
--noreboot           Disables the automatic rebooting when the
installation        is complete.
--force              Do not prompt for input. Answers yes where
```

applicable,

## 2. Create a guest using `virt-manager`

### Procedure 5.1. Creating a Virtual Machine using `virt-manager`

1. To start **virt-manager** execute the following as root in your shell:

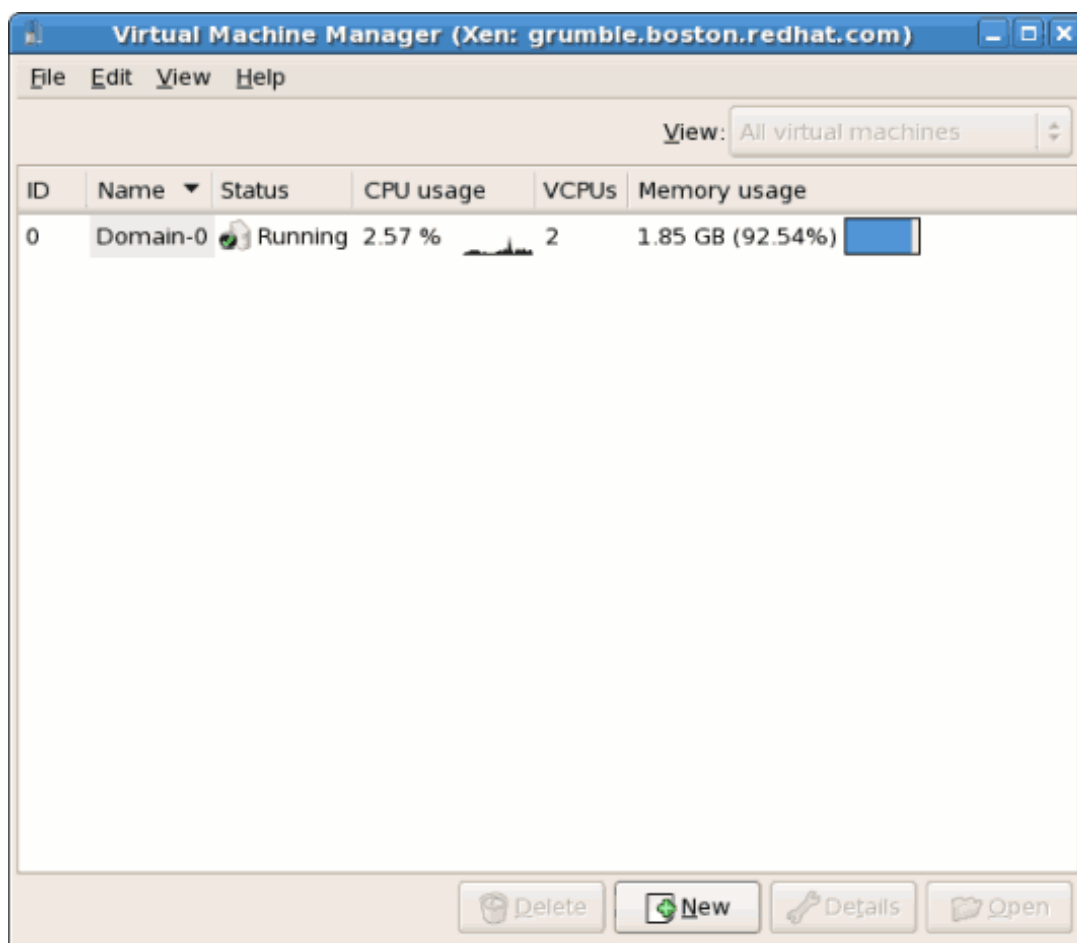
```
# virt-manager &
```

The `virt-manager` opens a new **virt-manager** application graphical user interface. If you do not have root privileges the **New** button will be grayed out and you will not be able to create a new virtual machine.

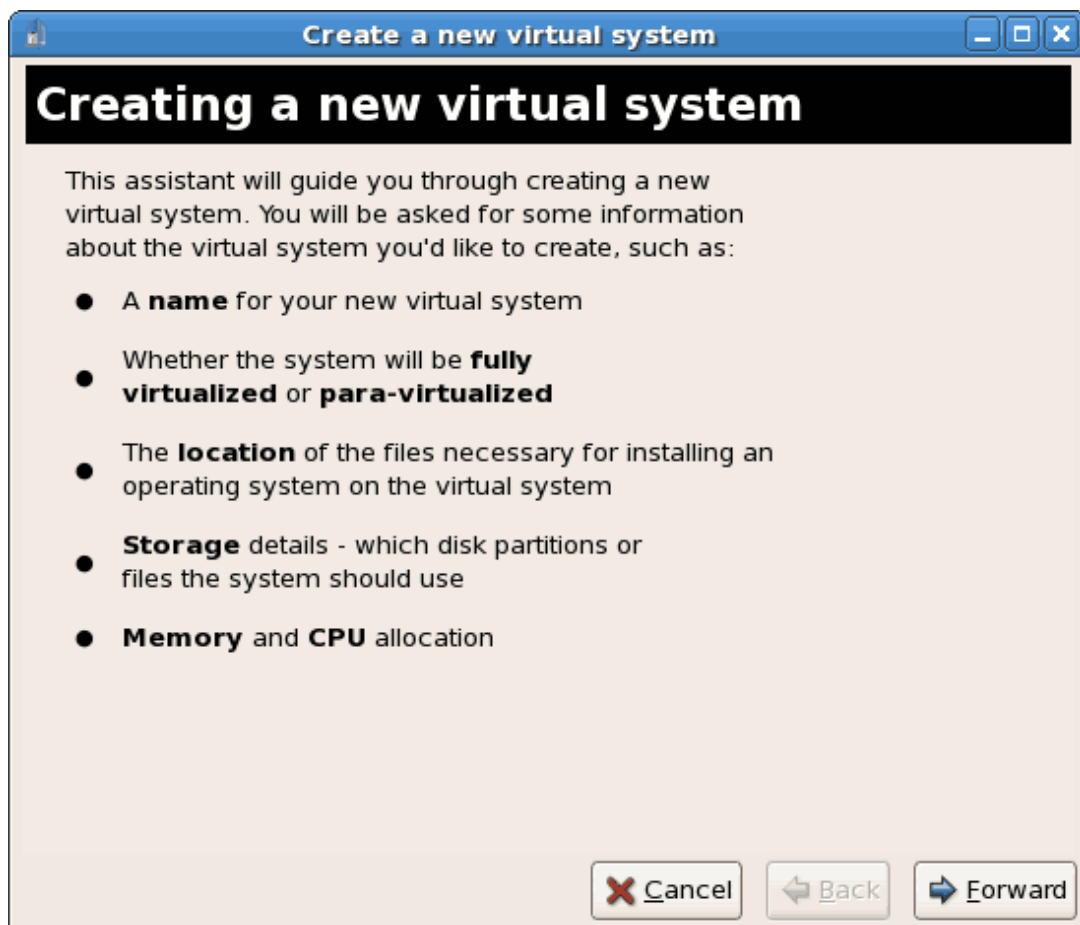
2. You will see a dialog box as the one below. Select the **Connect** button and the main **virt-manager** window will appear:



3. The main **virt-manager** window will allow you to create a new virtual machine using the **New** button:

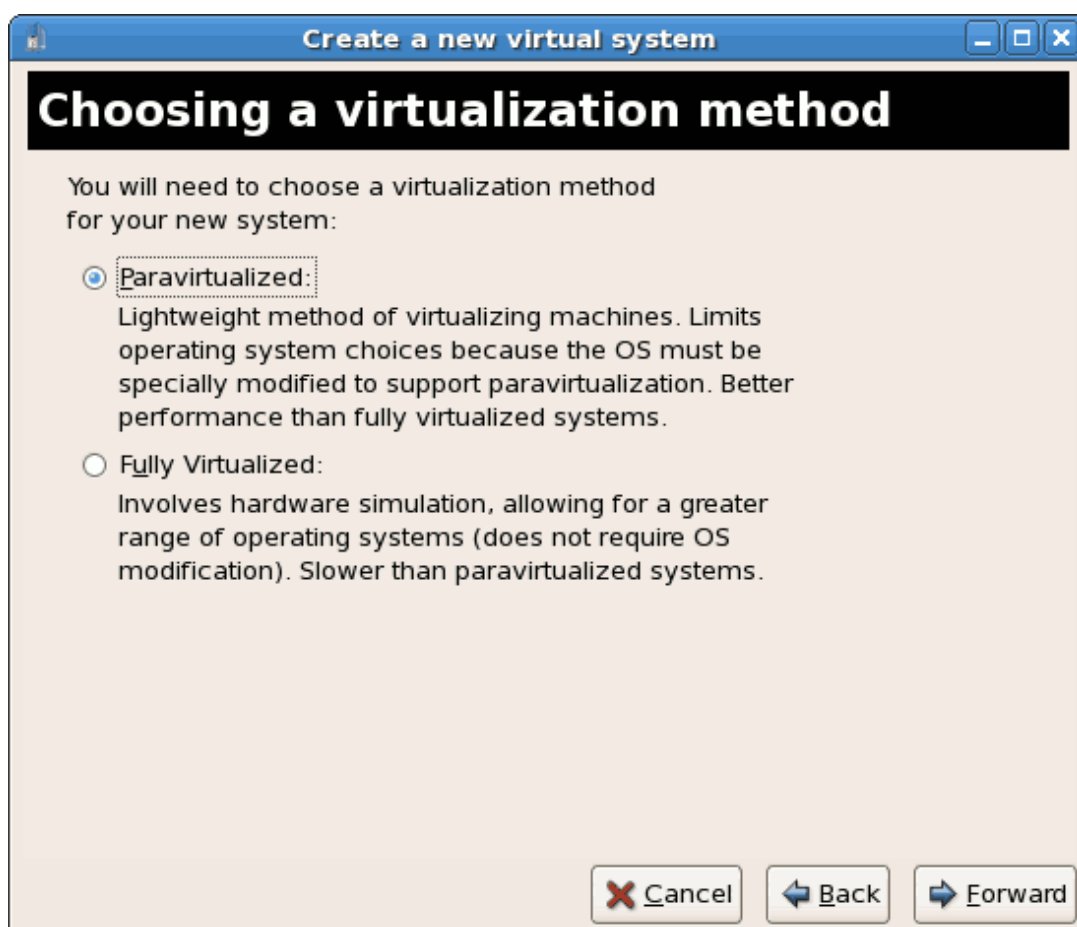


4. The next window provides a summary of the information you will need to provide in order to create a virtual machine:



After you have reviewed all of the information required for your installation you can continue to the next screen.

5. Depending on whether your system has Intel® VT or AMD-V capable processors the next window will either display a single choice to create a para-virtualized guest or two choices. Where one choice will be para-virtualized (modified and optimized operating system for virtualization) guest creation and the second will be for a fully virtualized (unmodified operating system) guest creation:



6. The next screen will ask for the installation media for the type of installation you selected. The para-virtualized installation will require an installation tree accessible either via HTTP, FTP or NFS (can be setup on the same system as where you install the guest). You can easily create an installation by either mounting the installation media DVD to a local directory and exporting it via NFS or making it available via FTP or HTTP. If your media is an .iso file you can loopback mount the file and extract the files onto a local directory.

**Create a new virtual system**

## Locating installation media

Please indicate where installation media is available for the operating system you would like to install on this **paravirtualized** virtual system. Optionally you can provide the URL for a kickstart file that describes your system:

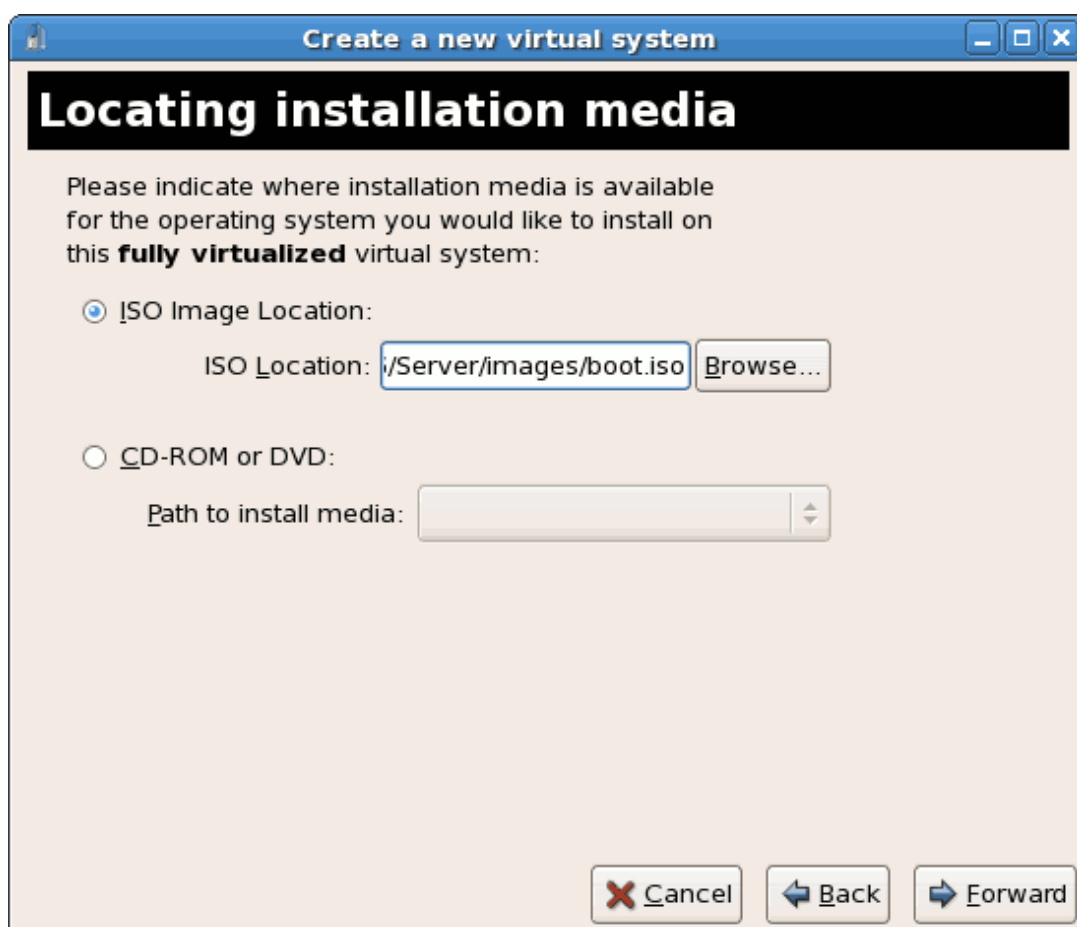
Install Media URL:

**i** **Example:** http://servername.example.com/distro/i386/tree

Kickstart URL:

**i** **Example:** ftp://hostname.example.com/ks/ks.cfg

7. The fully virtualized installation will ask for the location of a boot media (ISO image, DVD or CD-ROM drive). Depending on your installation media/process you can either perform a network based installation after booting your guest off the .iso file or perform the whole installation off a DVD .iso file. typically Windows installations are using DVD/CD .iso files, Linux or unix-like operating systems such as Red Hat Enterprise Linux use use an .iso file for installing a base system to use a a network based tree):



8. This screen is for selecting storage for the guest. Choose a disk partition, LUN or a file based image for the location of the guest image. The convention for Red Hat Enterprise Linux 5 is to install all file based guest images in the `/var/lib/xen/images/` directory as other SELinux blocks access to images located in other directories. If you run SELinux in enforcing mode, see [Chapter 11, SELinux and virtualization](#) for more information on installing guests. You must choose a size for file based guest image storage. The size of your guest image should be larger than the size of the installation, any additional packages and applications, and the size of the guests swap file. The installation process will choose the size of the guest's swap file based on size of the RAM allocated to the guest. Remember to allocate extra space if the guest is to store additional files, for example web server log files or user storage.



**Create a new virtual system**

## Assigning storage space

Please indicate how you'd like to assign space on this physical host system for your new virtual system. This space will be used to install the virtual system's operating system.

☐ Normal Disk Partition:

Partition:

**Example:** /dev/hdc2

☒ Simple File:

File Location:

File Size:  MB

**Note:** File size parameter is only relevant for new files

**Tip:** You may add additional storage, including network-mounted storage, to your virtual system after it has been created using the same tools you would on a physical system.



### Note

It is recommend to use the default directory for virtual machine images which is `/var/lib/xen/images/`. If you are using a different location (such as `/xen/images/` in this example) make sure it is added to your SELinux policy and relabeled before you continue with the installation (later in the document you will find information on how to modify your SELinux policy)

- The last configuration information to enter is the memory size of the guest you are installing and the number of virtual CPUs you would like to assign to your guest. Red Hat Enterprise Linux 5 / Virt will require physical memory to back a guest's memory you must ensure your system is configured with sufficient memory in order to accommodate the guest you may like to run and configure. A good practice for virtual CPU assignments is to not configure more virtual CPUs in a single guest than available physical processors in the host. You can allocate more virtual processors across a number of virtual machine than the number of physical processors available. However, you should generally avoid doing this as it will significantly negatively affect performance of your guests and host.

The screenshot shows a window titled "Create a new virtual system" with a sub-header "Allocate memory and CPU". The window is divided into two main sections: "Memory:" and "CPUs:". The "Memory:" section includes instructions to enter memory configuration, a note about total host memory (2046 GB), and two spinners for "VM Max Memory" and "VM Startup Memory", both set to 500. The "CPUs:" section includes instructions to enter the number of virtual CPUs, a note about logical host CPUs (2), and a spinner for "VCPU" set to 1. A tip icon and text are present below the CPU section. At the bottom are three buttons: "Cancel", "Back", and "Forward".

**Create a new virtual system**

## Allocate memory and CPU

**Memory:**

Please enter the memory configuration for this VM. You can specify the maximum amount of memory the VM should be able to use, and optionally a lower amount to grab on startup.

Total memory on host machine: 2046 GB

VM Max Memory: 500

VM Startup Memory: 500

**CPUs:**

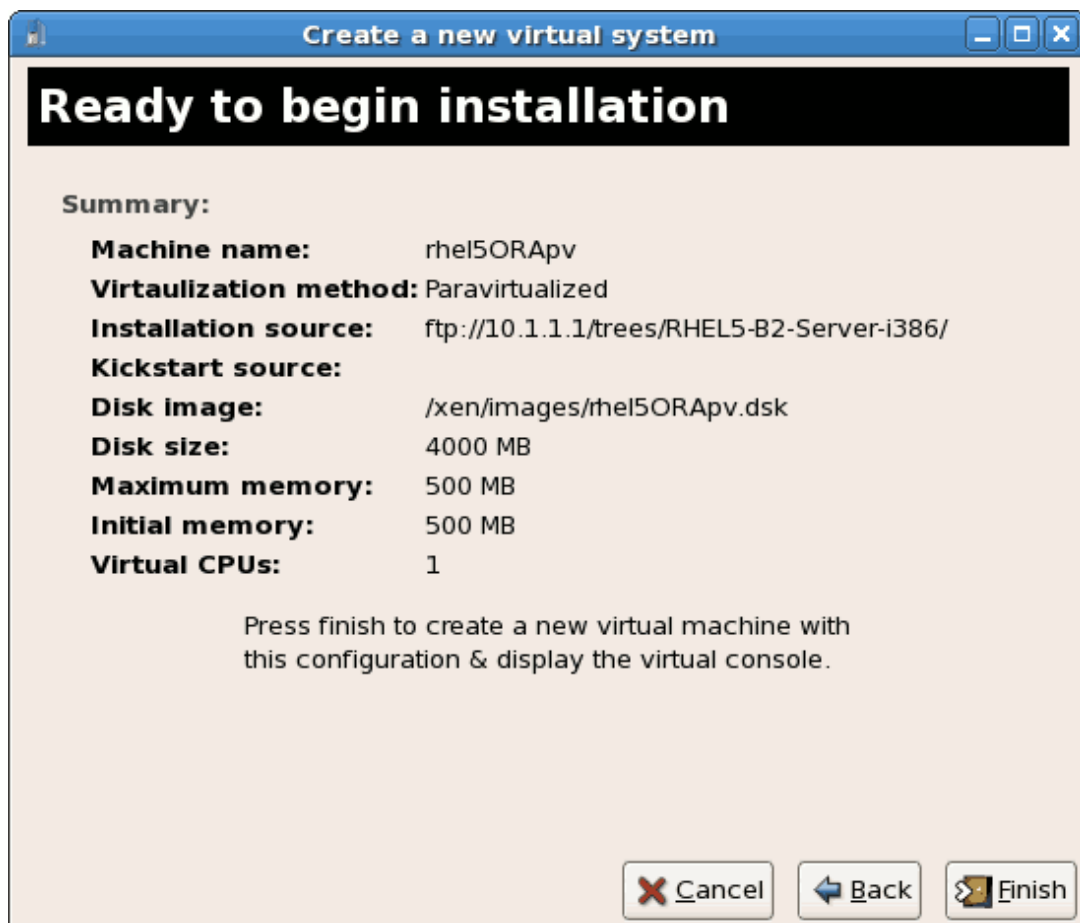
Please enter the number of virtual CPUs this VM should start up with.

Logical host CPUs: 2

VCPUs: 1

**Tip:** For best performance, the number of virtual CPUs should be less than (or equal to) the number of logical CPUs on the host system.

10. At this step you will be presented with a summary screen of all configuration information you entered. Review the information presented and use the **Back** button to make changes. Once you are satisfied with the data entered click the **Finish** button and the installation process will commence:



Press the **Finish** button in **virt-manager** to conclude the installation and automatically launch a VNC based window for the installation process.



# Guest operating system installation processes

This chapter covers, in detail, how to install various guest operating systems in a virtualized environment.

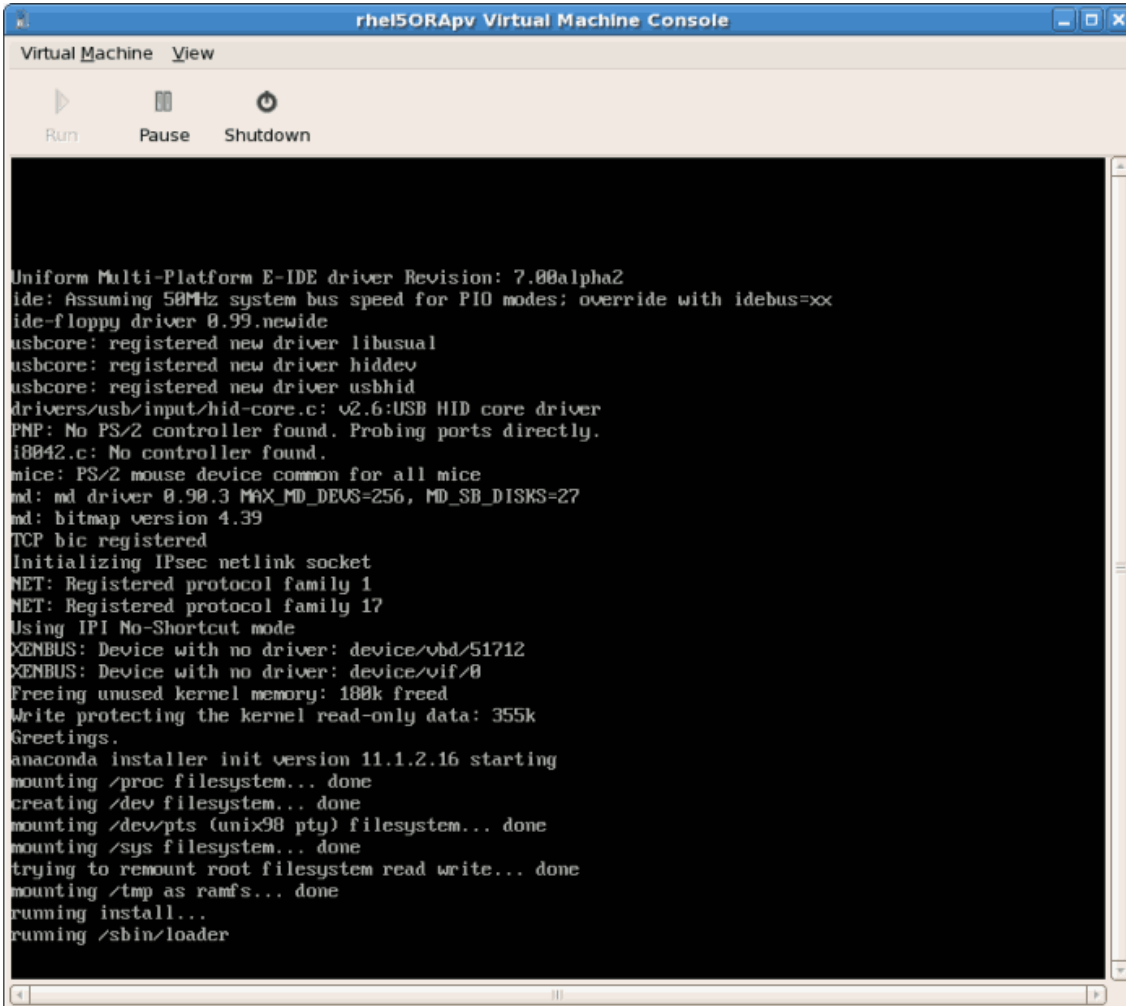
## 1. Installing Red Hat Enterprise Linux 5 as a para-virtualized guest

If you are using the `virt-install` command line interface tool and you select the `--vnc` option for a graphical installation you will also see the graphical installation screen as shown below.

If you used `virt-install` or `virt-manager` the following command line would start a guest installation with the same parameters as selected above via the GUI:

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

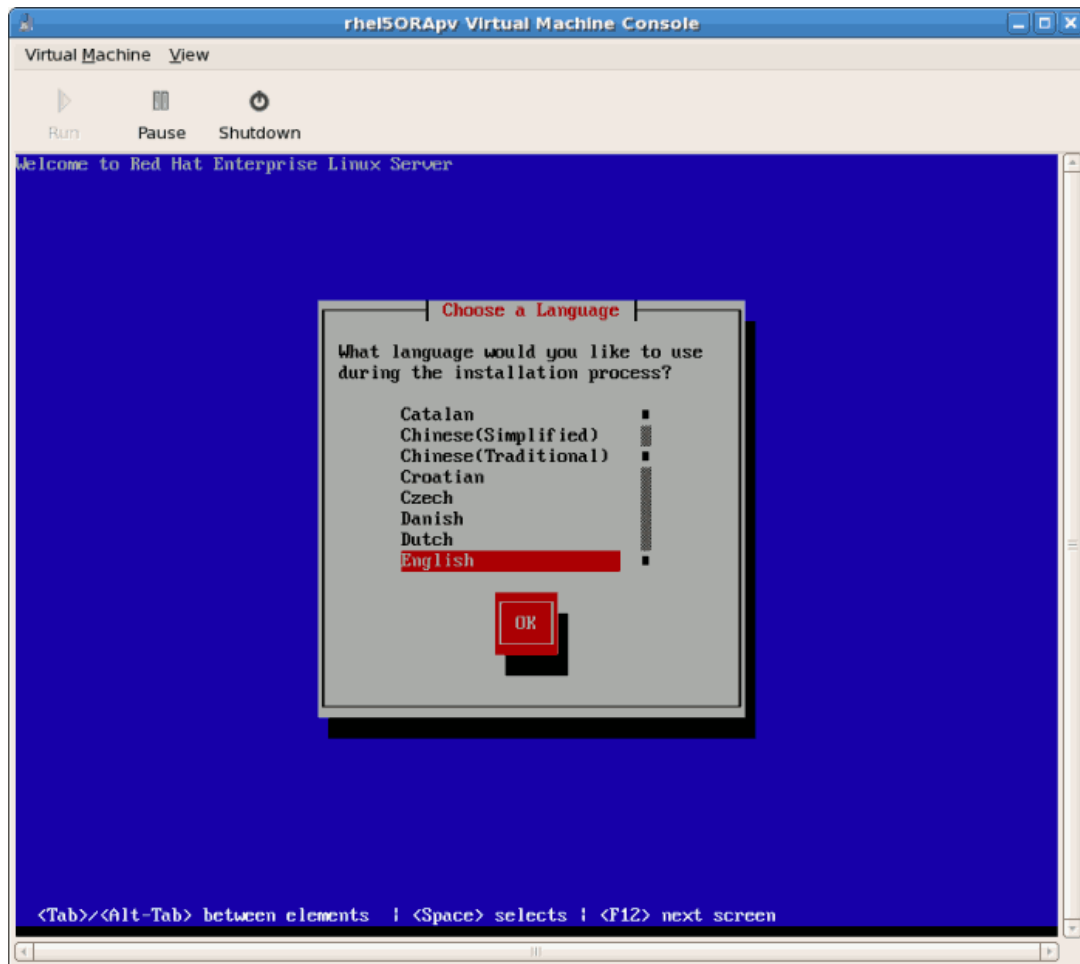
The window will show the initial boot of your guest:



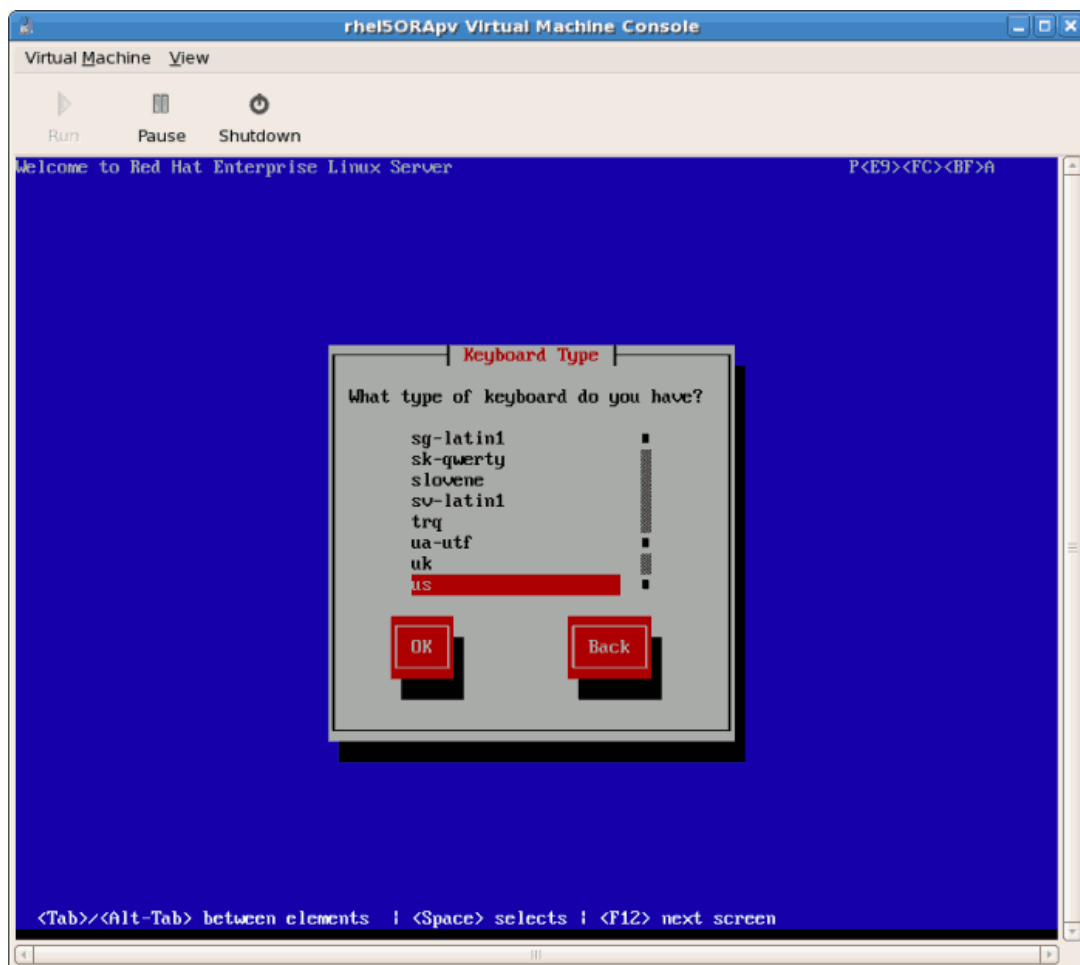
```
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
ide-floppy driver 0.99.newide
usbcore: registered new driver libusual
usbcore: registered new driver hiddev
usbcore: registered new driver usbhid
drivers/usb/input/hid-core.c: v2.6:USB HID core driver
PNP: No PS/2 controller found. Probing ports directly.
i8042.c: No controller found.
mice: PS/2 mouse device common for all mice
md: md driver 0.90.3 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: bitmap version 4.39
TCP bic registered
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI No-Shortcut mode
XENBUS: Device with no driver: device/vbd/51712
XENBUS: Device with no driver: device/vif/0
Freeing unused kernel memory: 188k freed
Write protecting the kernel read-only data: 355k
Greetings.
anaconda installer init version 11.1.2.16 starting
mounting /proc filesystem... done
creating /dev filesystem... done
mounting /dev/pts (unix98 pty) filesystem... done
mounting /sys filesystem... done
trying to remount root filesystem read write... done
mounting /tmp as ramfs... done
running install...
running /sbin/loader
```

After your guest has completed its initial bootstrap you will be dropped into the standard installation process for Red Hat Enterprise Linux or the operating system specific installer you select to install. For most installations the default answers will be acceptable

1. The Red Hat Enterprise Linux installer will ask you for the installation language:

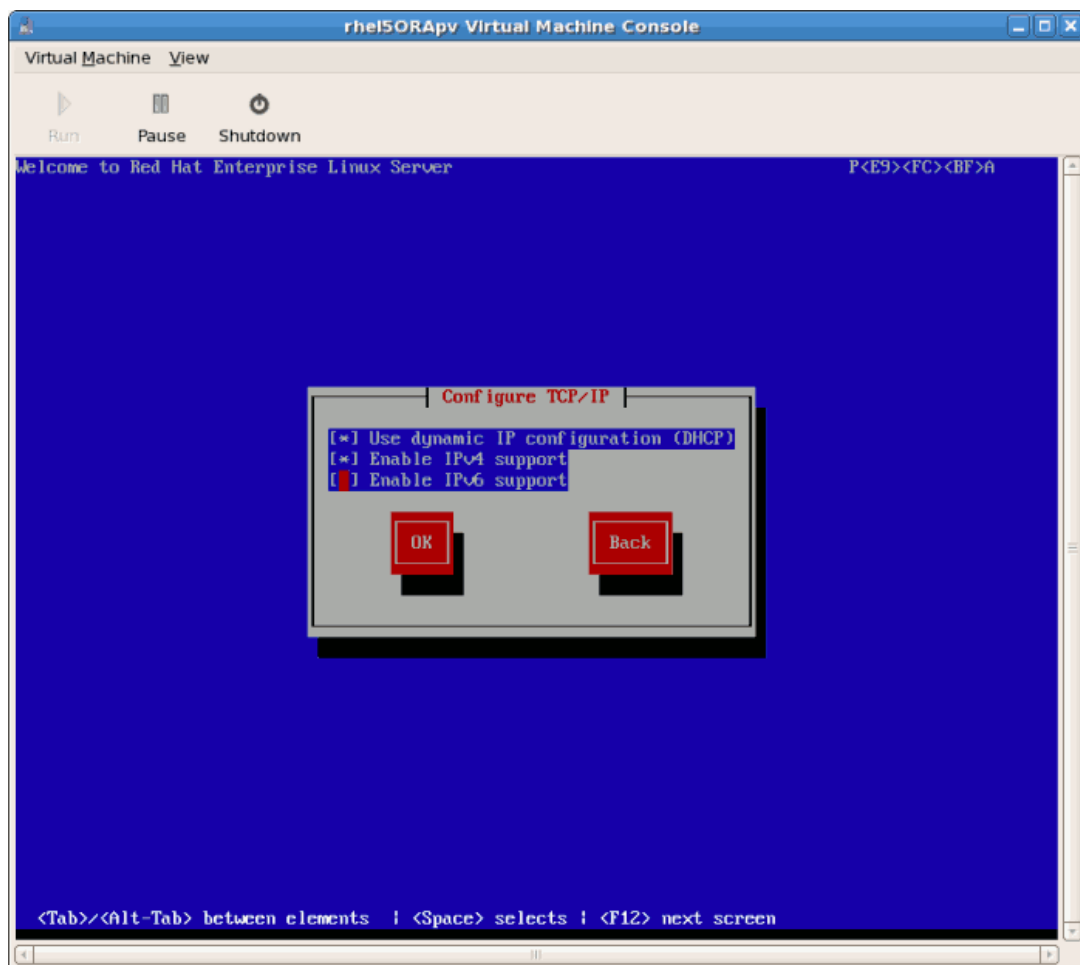


2. Next you have to select the keyboard layout you want to use:

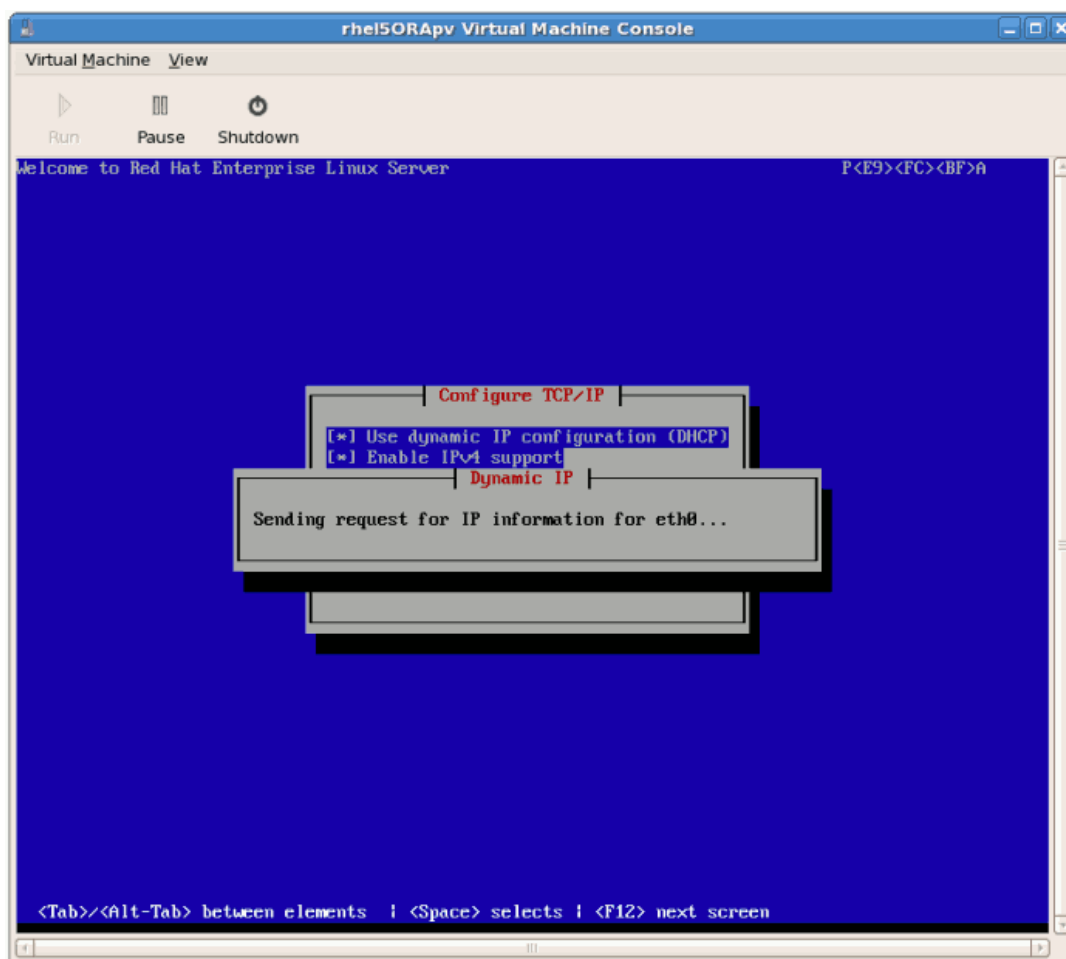


3. Assign the guest's network address. You can either choose DHCP (as shown below) or static:

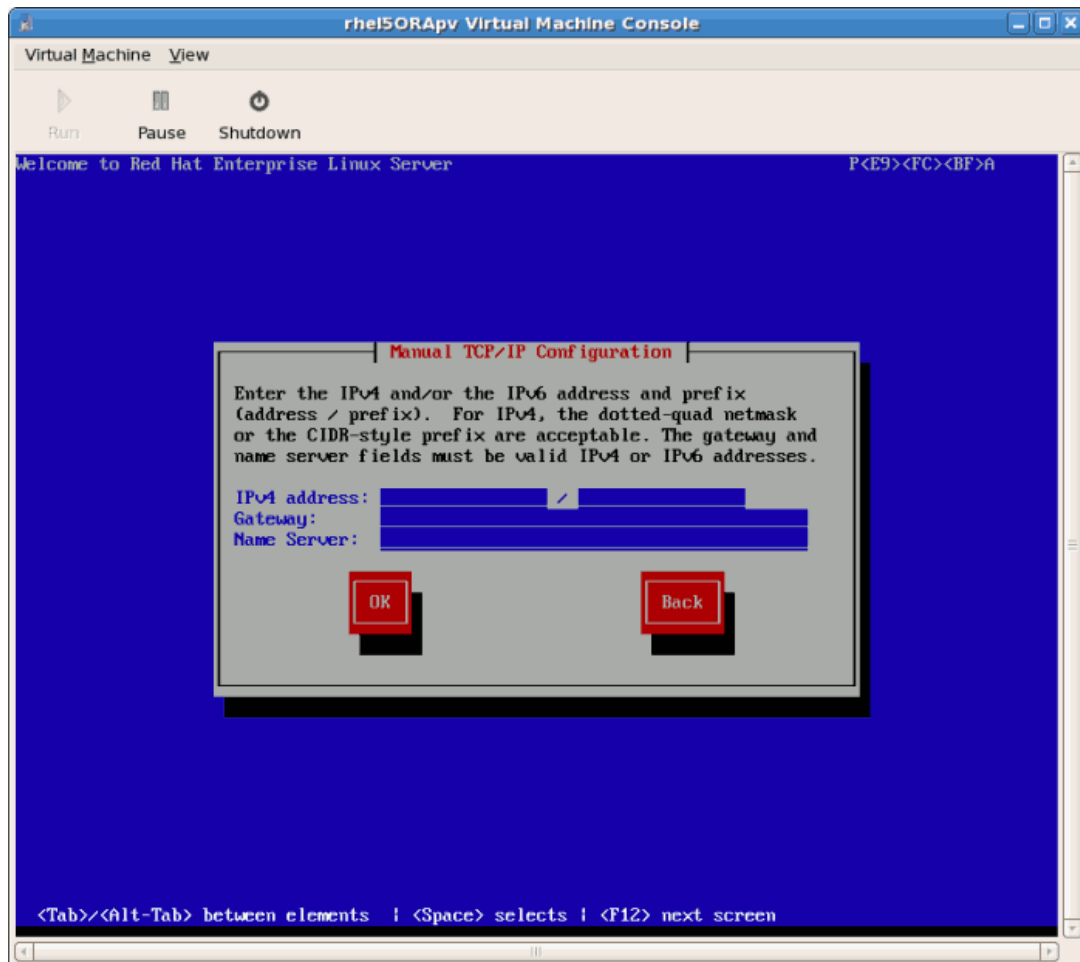




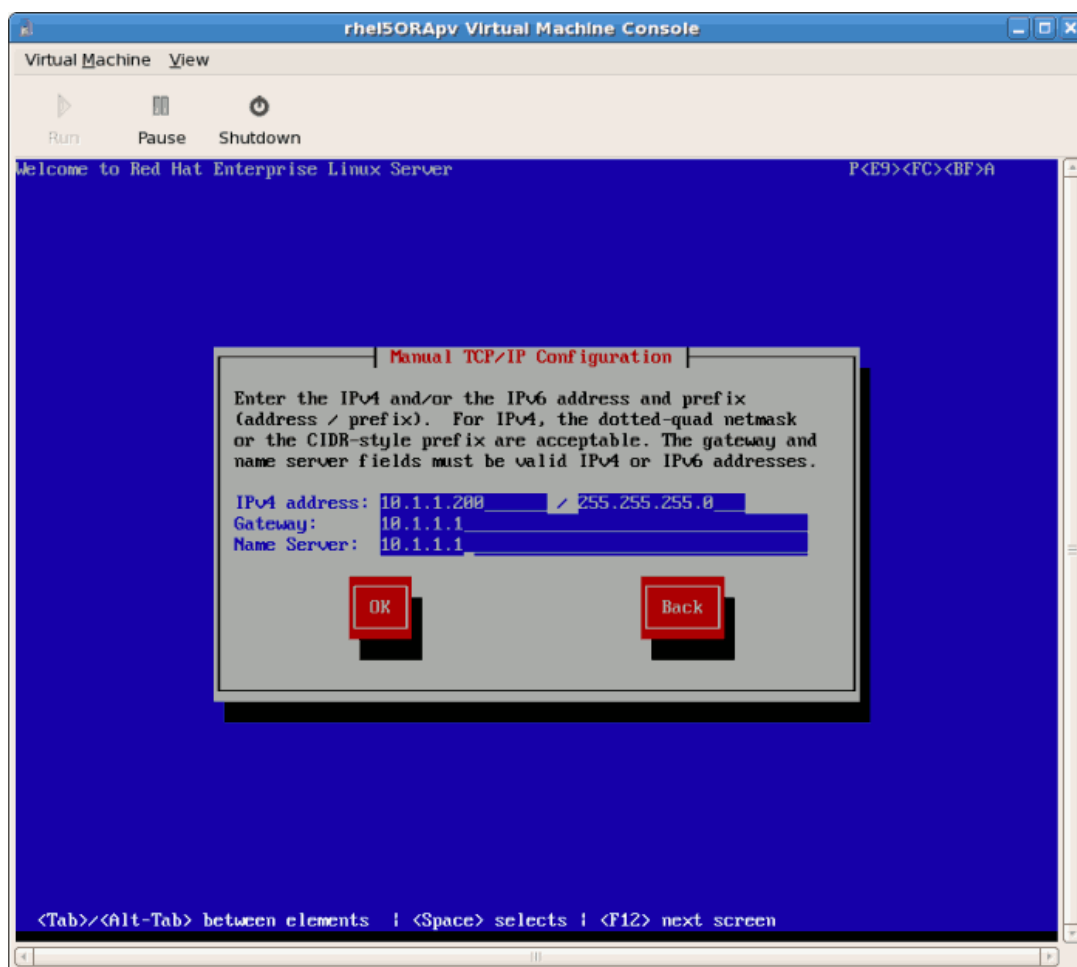
4. If you have chosen DHCP for your guest the installation process will now try to acquire an IP address for your guest via DHCP:



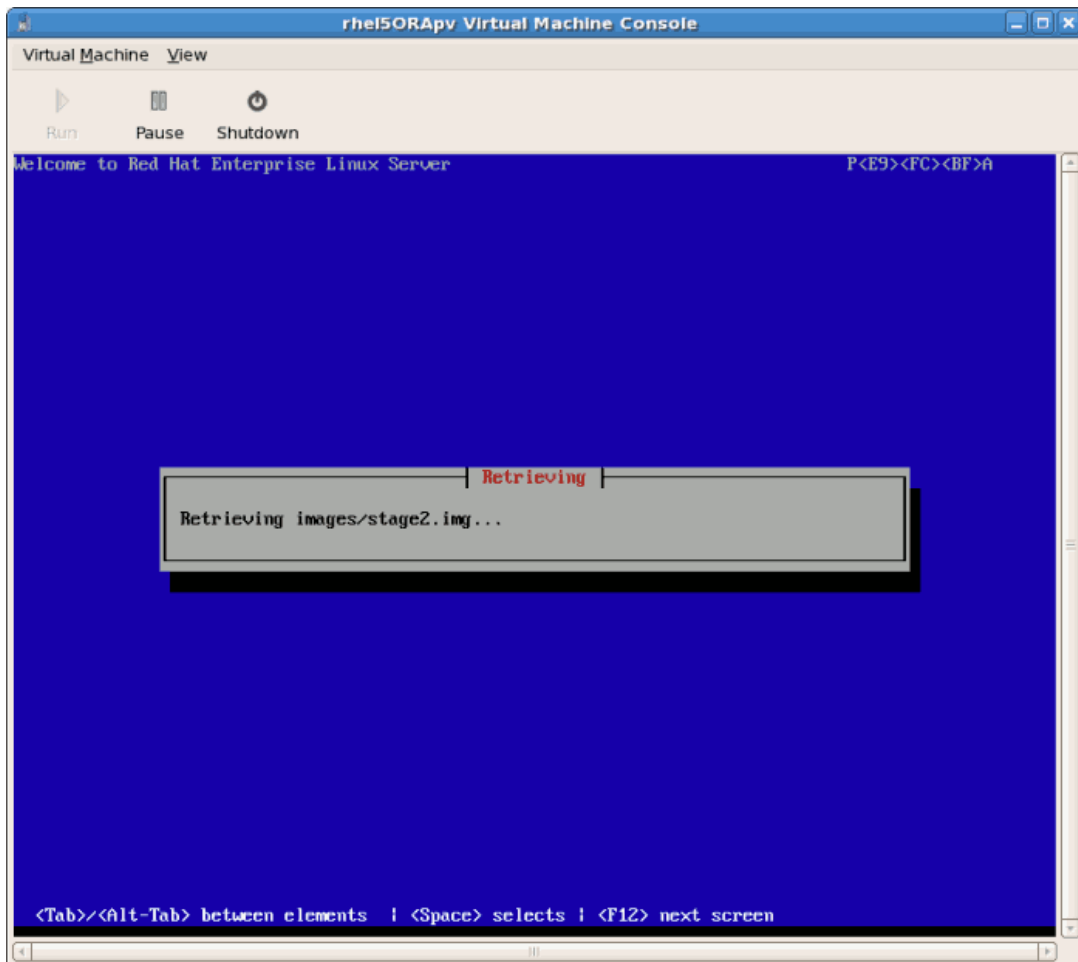
5. If you elected to use a static IP address for your guest the next screen will ask you for the details on the guest's networking configuration:
  - a. Enter a valid IP address, also make sure the IP address you enter can reach your installation server which provides the installation tree.
  - b. Enter a valid Subnet mask, default gateway and name server address.



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

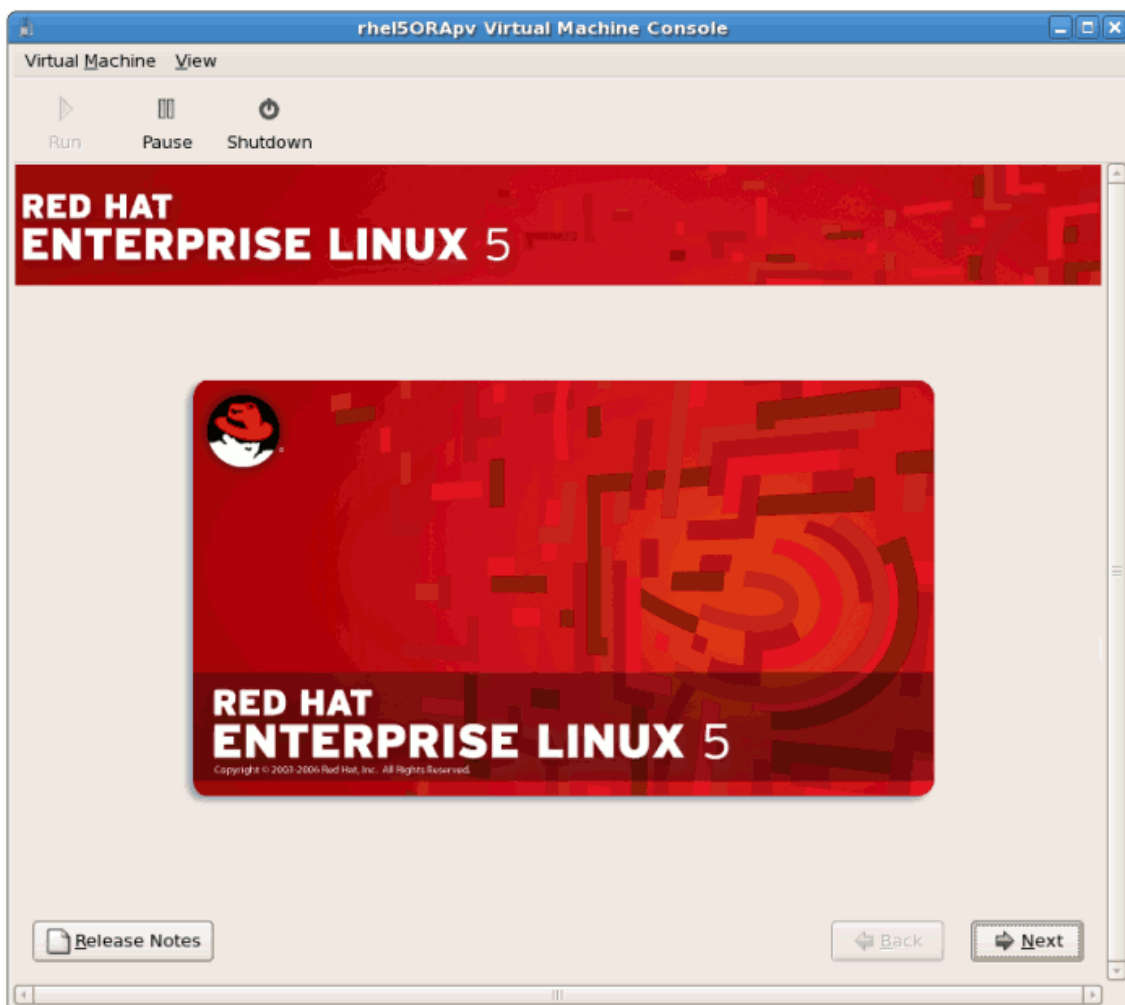


7. After you finished the networking configuration of your guest the installation process will retrieve the installation files required to continue:

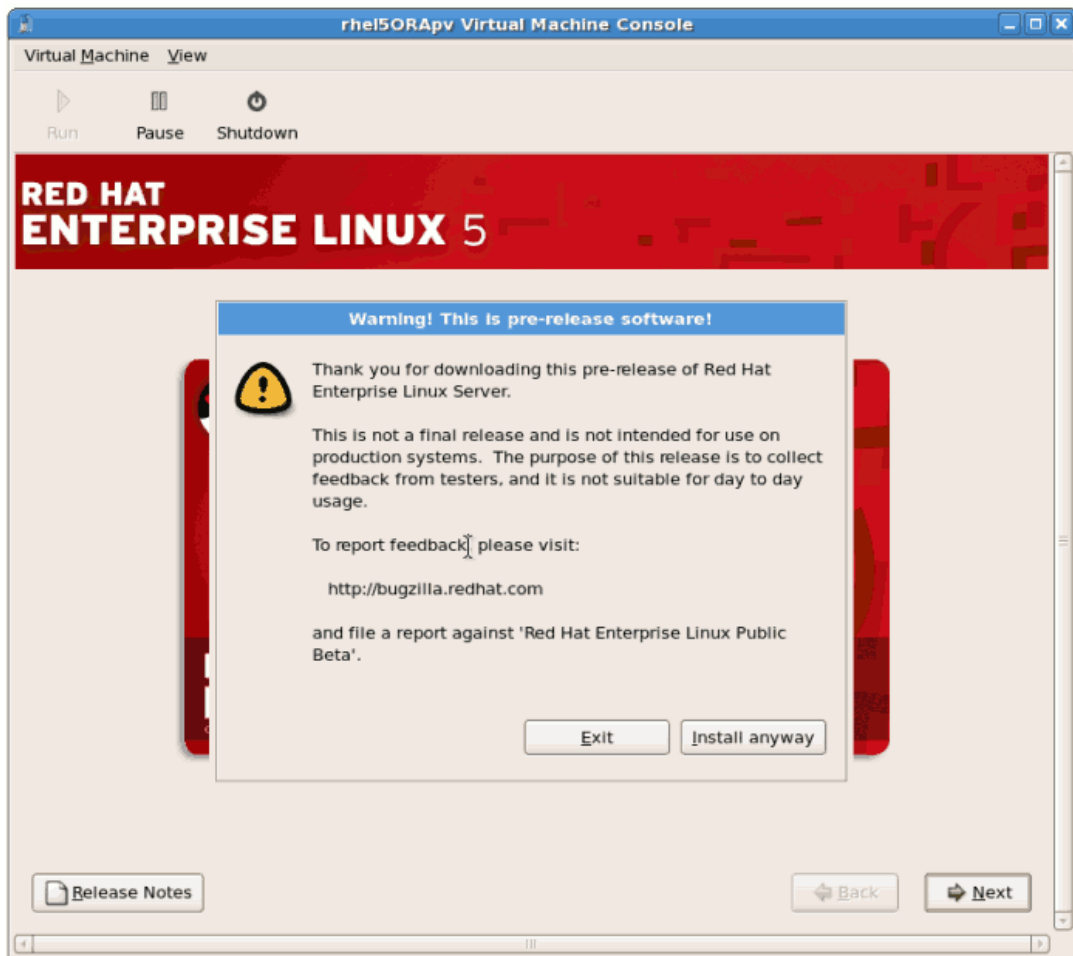


## 1.1. Graphical Red Hat Enterprise Linux 5 installation

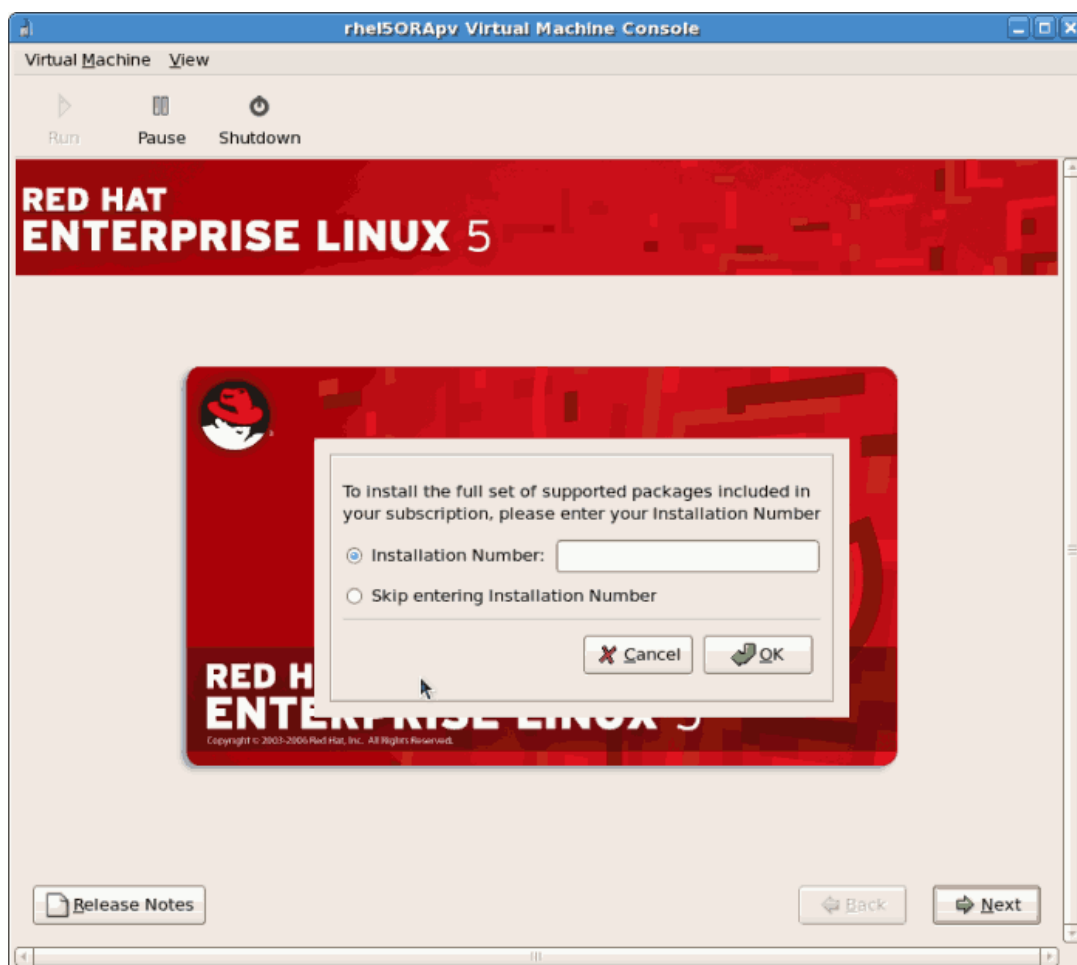
After you have answered the basic questions to get the installation process started and all of the required files have been retrieved you will be greeted by the graphical installation screen of Red Hat Enterprise Linux 5 (or the installation screen of the operating system you decided to install):



1. If you are installing a Beta or early release distribution you need to confirm that you really want to install the operating system:

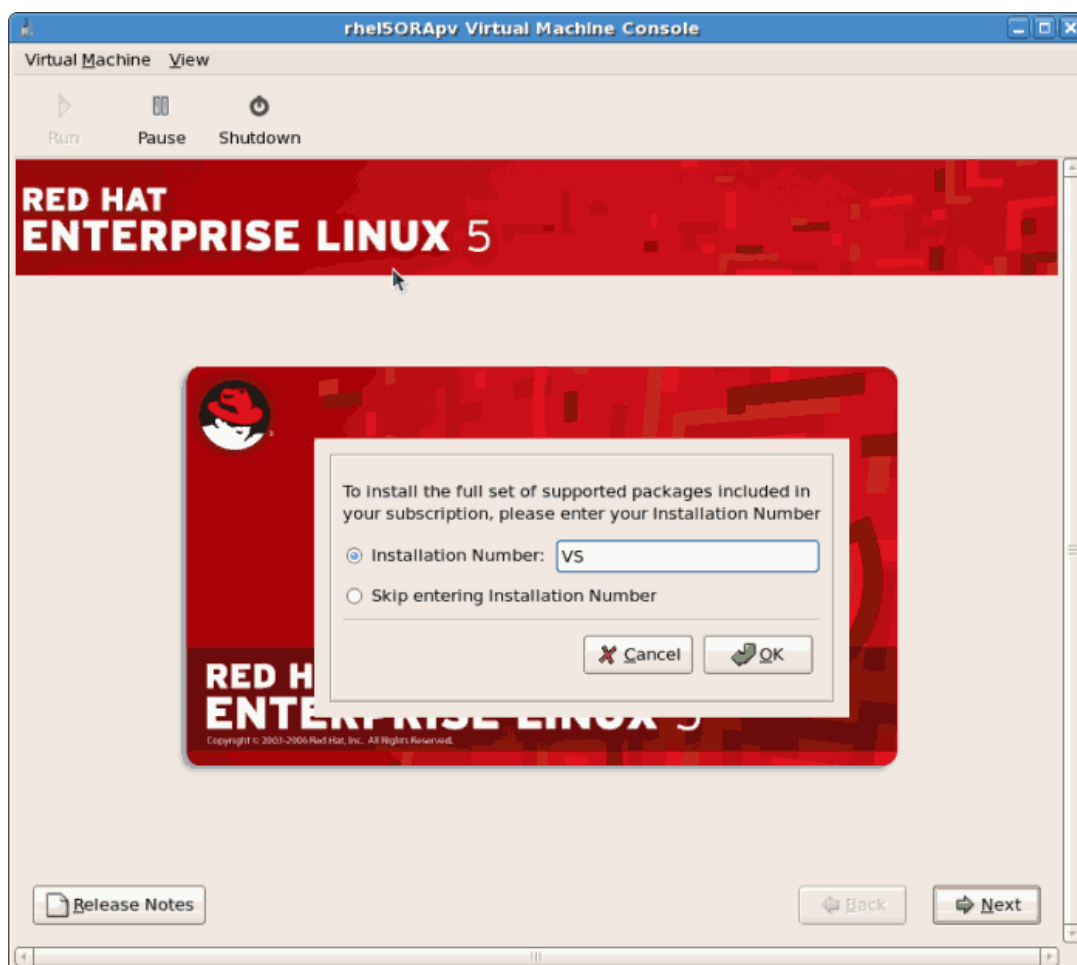


2. The next step is to enter a valid registration code. If you have a valid RHN subscription key please enter in the `Installation Number` field:



3. The release notes for Red Hat Enterprise Linux 5 have temporary subscription keys for Server and Client installations. If you are installing a local environment with no need to access RHN enter **v** for virtualization, **s** for clustered storage (**CLVM** or **GFS**) and **c** for clustering (Red Hat Cluster Suite) if required. In most cases entering **v** is sufficient:

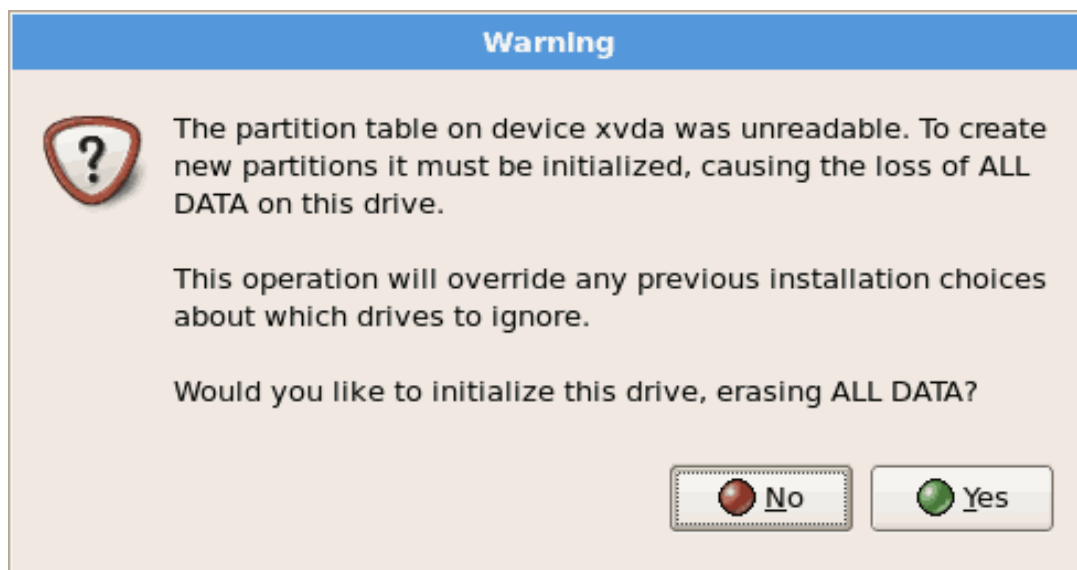




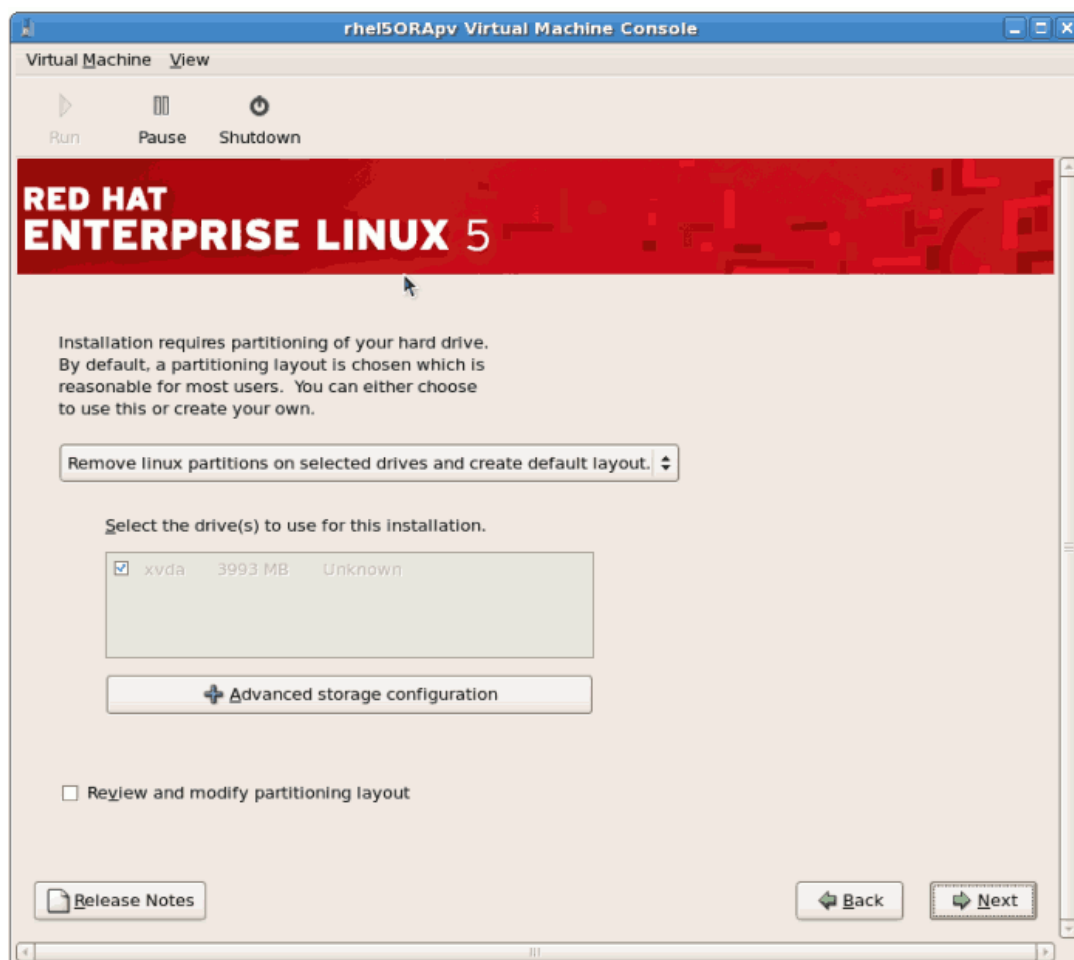
### Note

Depending on your version of Red Hat Enterprise Linux the setup numbers above may not work. In that case you can skip this step and confirm your Red Hat Network account details after the installation using the `rhn_register` command.

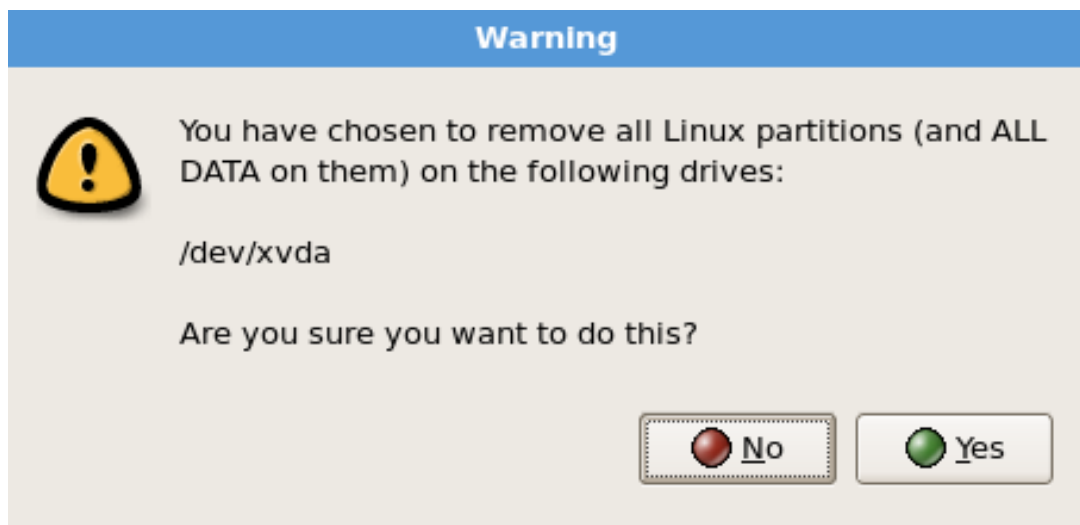
4. The installation will now confirm you want to erase the data on the storage you selected for the installation:



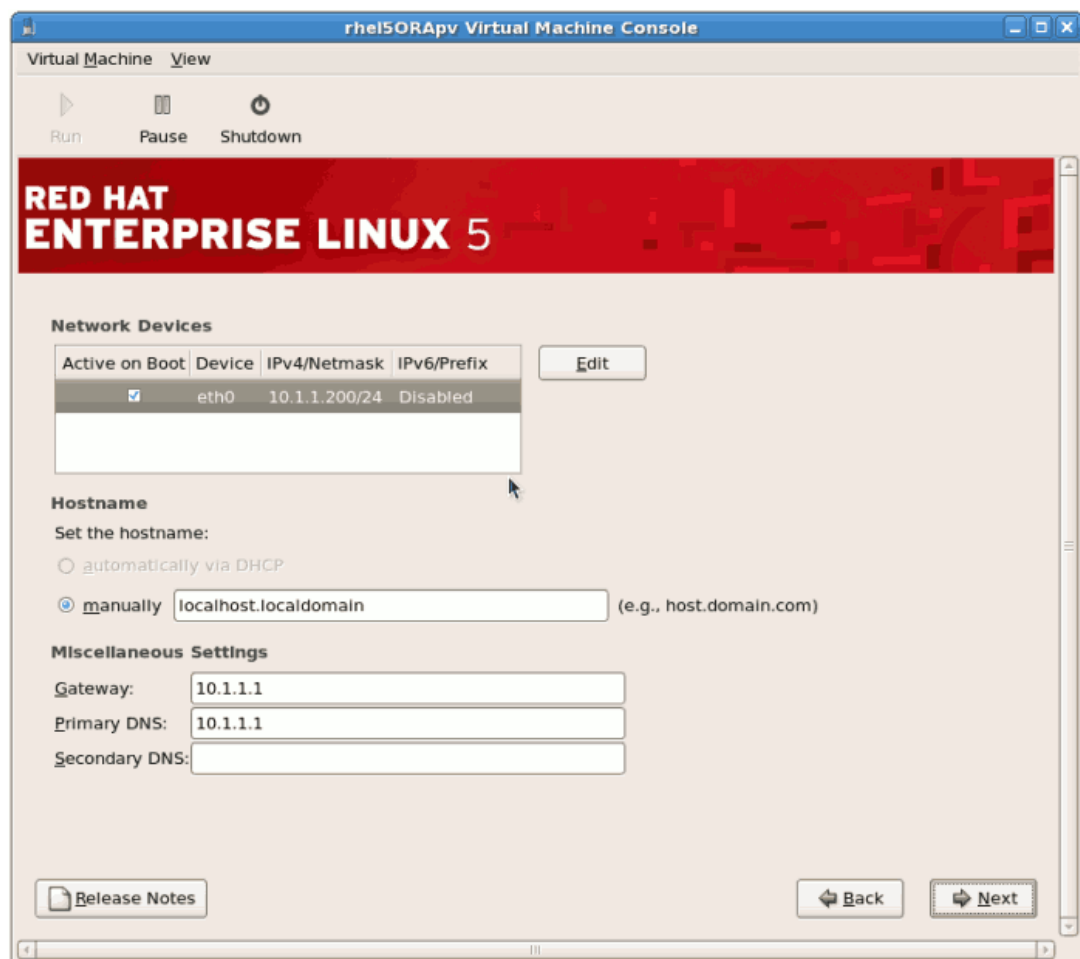
5. This screen will allow you to review the storage configuration and partition layout. You can also chose to select the advanced storage configuration if you want to use iSCSI as guest storage:



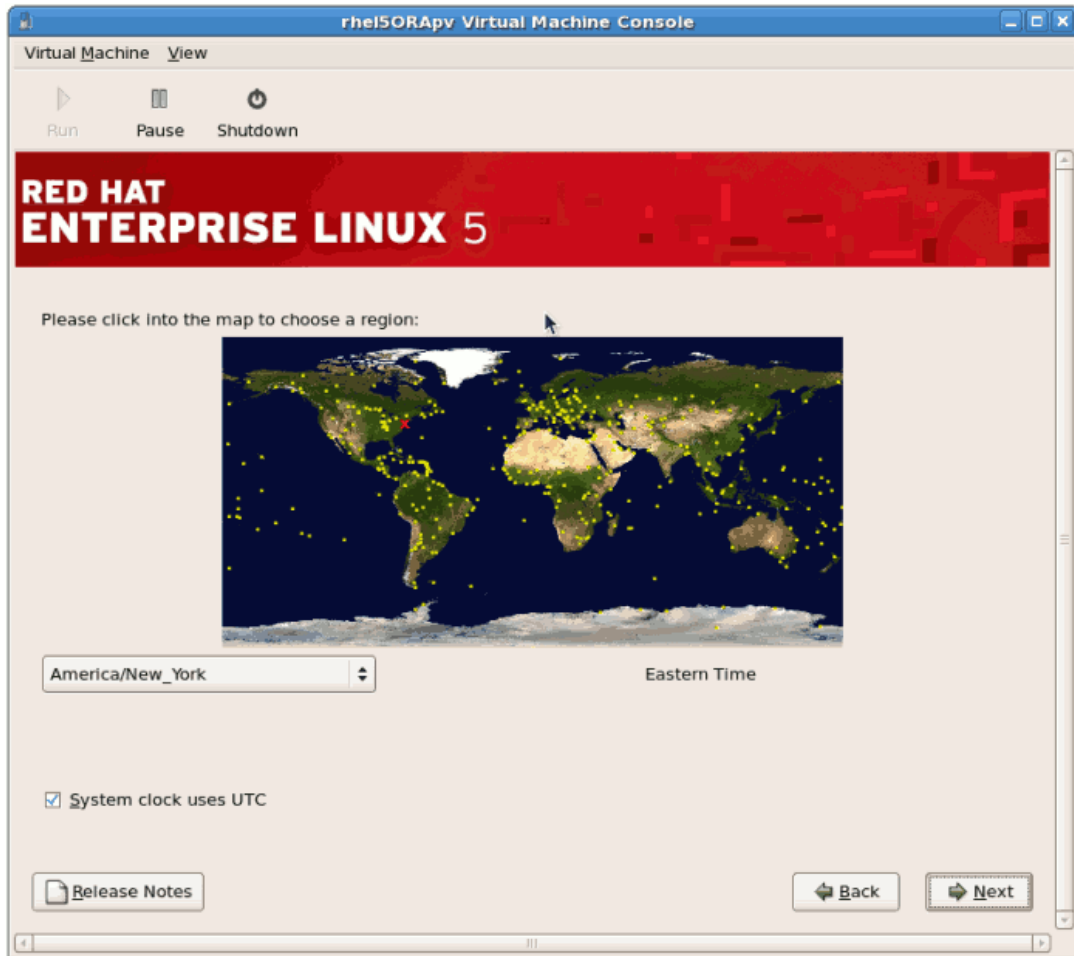
- After you have reviewed the storage choice just confirm that you indeed want to use the drive for the installation:



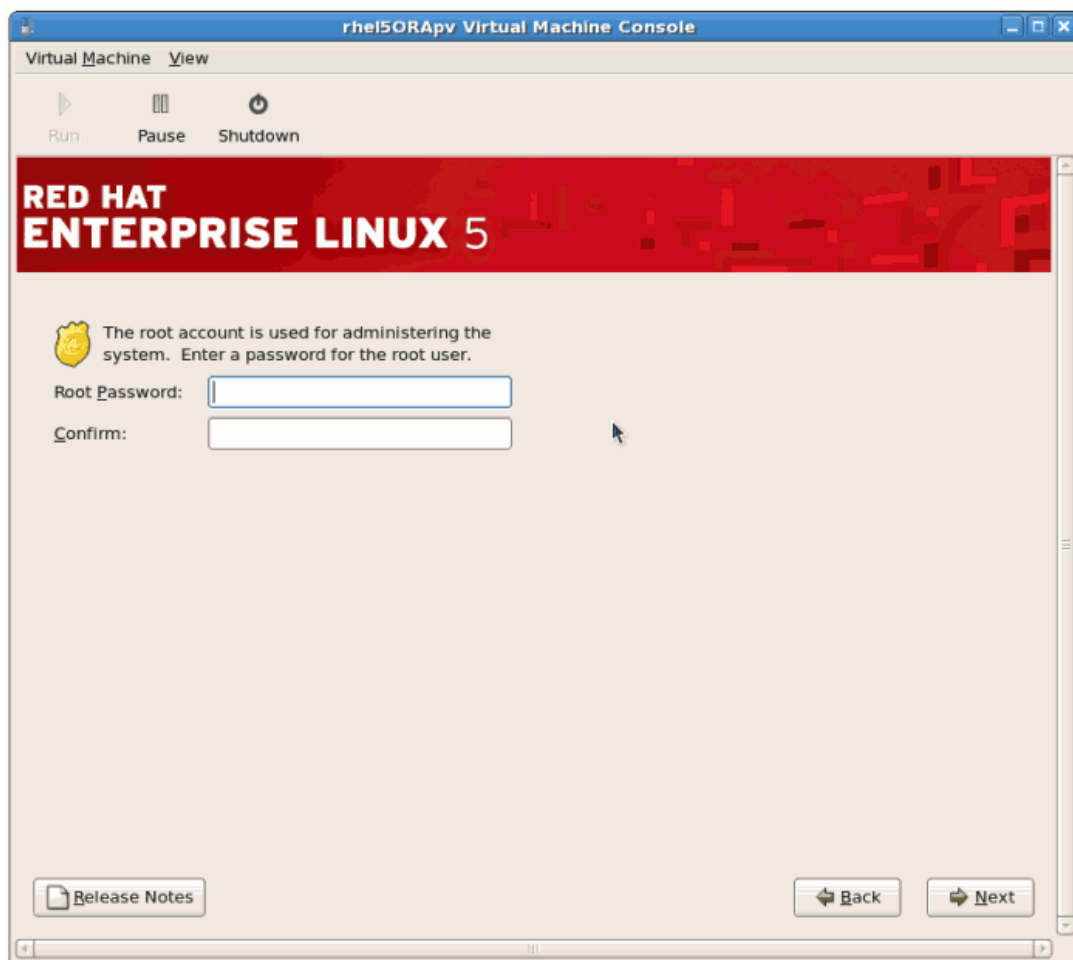
- Next is the guest networking configuration and hostname settings. The information will be populated with the data you entered earlier in the installation process:



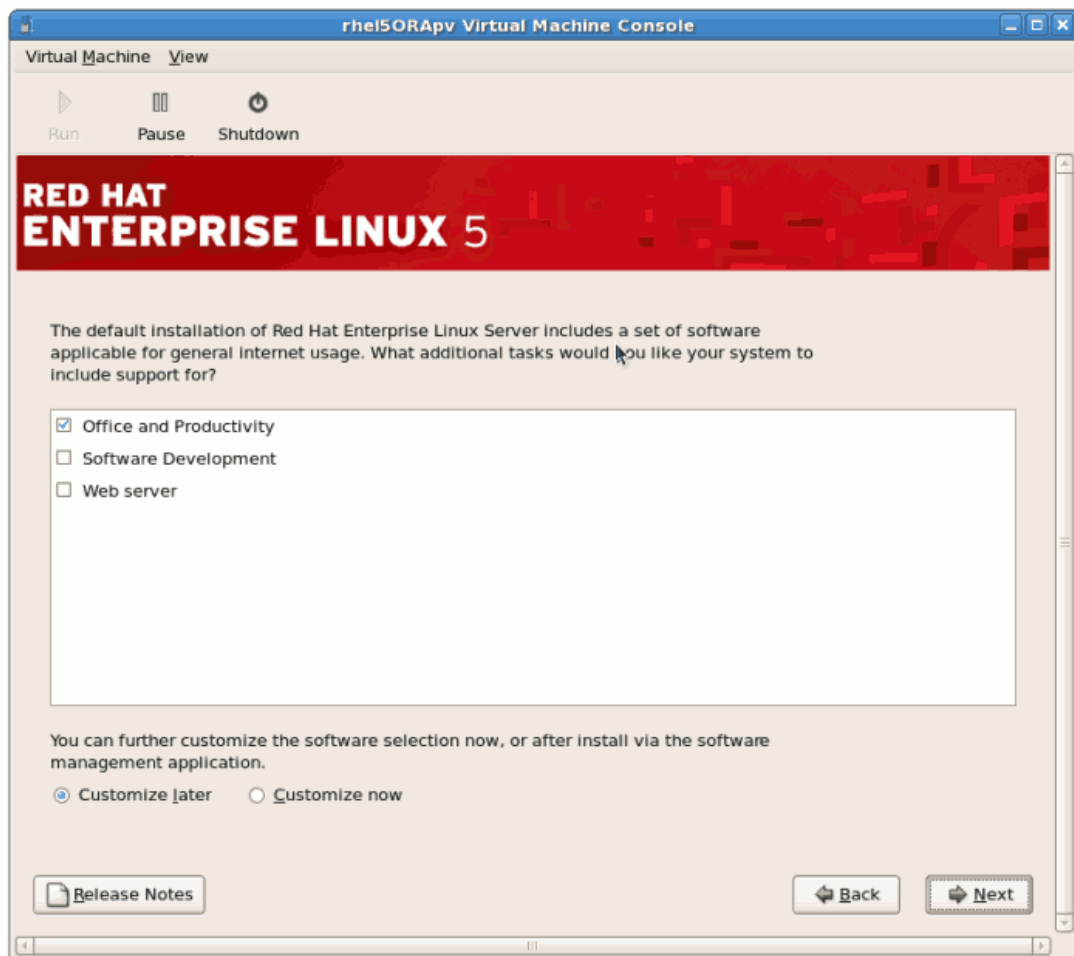
8. Select the appropriate time zone for your environment:



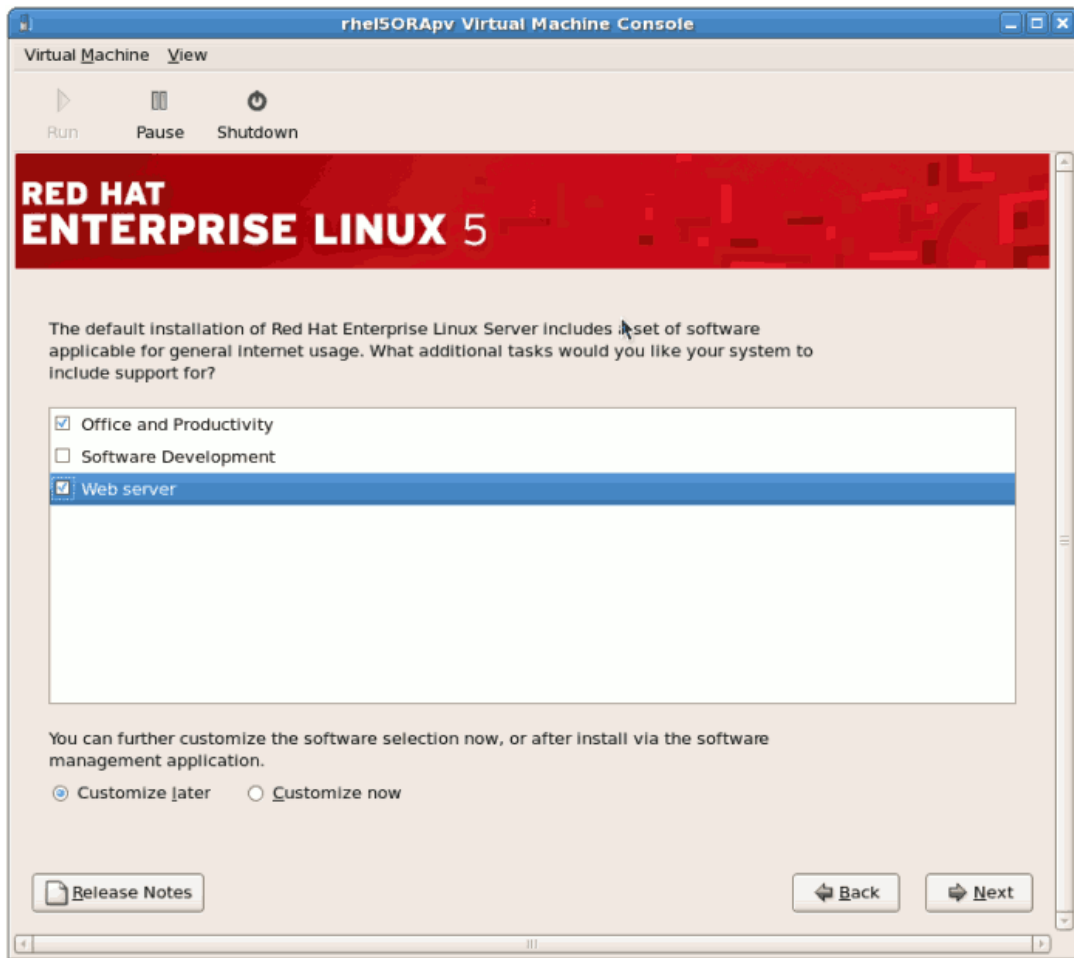
9. Select a root password for your guest:



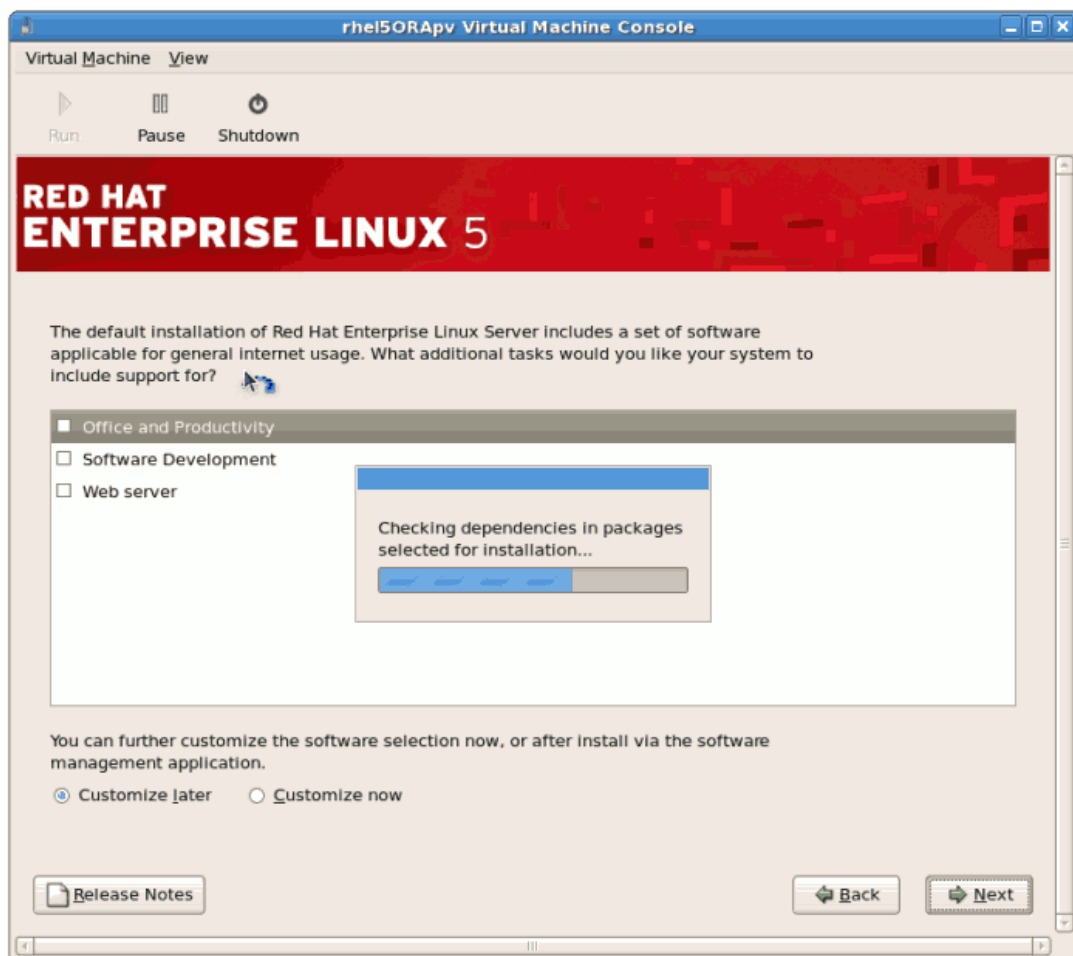
10. Now you get to select the software packages you want to install. You can also choose to customize the software selection by selecting the **Customize Now** button:



11. For our installation we chose the office packages and web server packages:

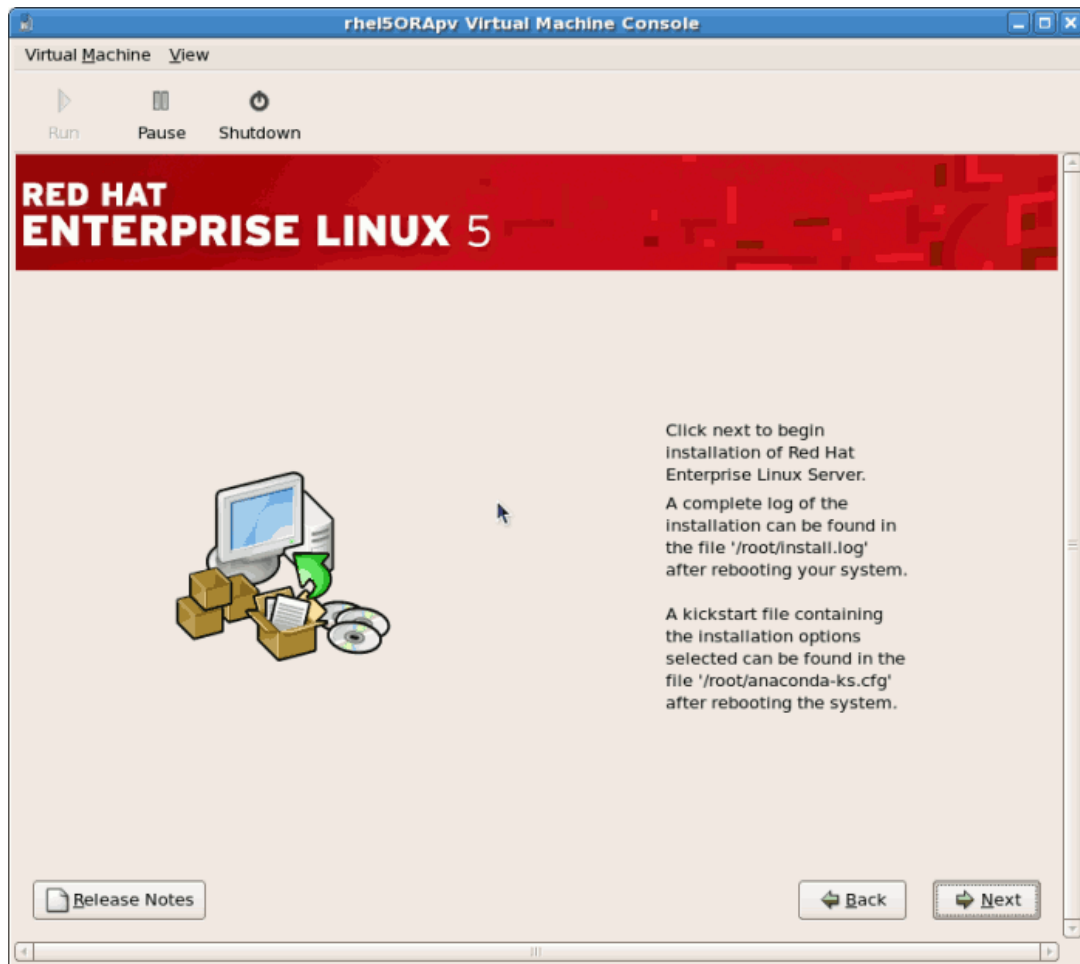


12. After you have selected the packages to install, dependencies and space requirements will be verified:



13. After all of the installation dependencies and space requirements have been verified you need to press the **Next** button to start the actual installation:

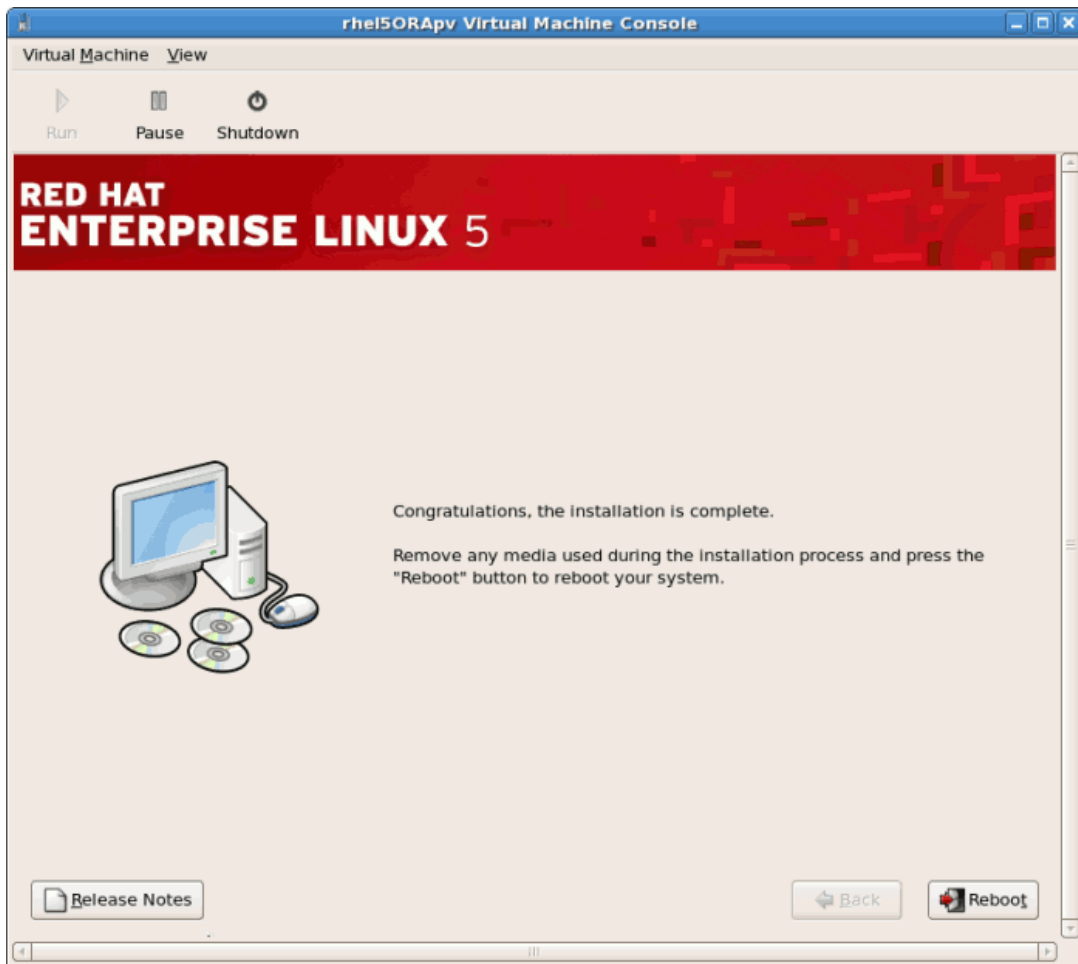




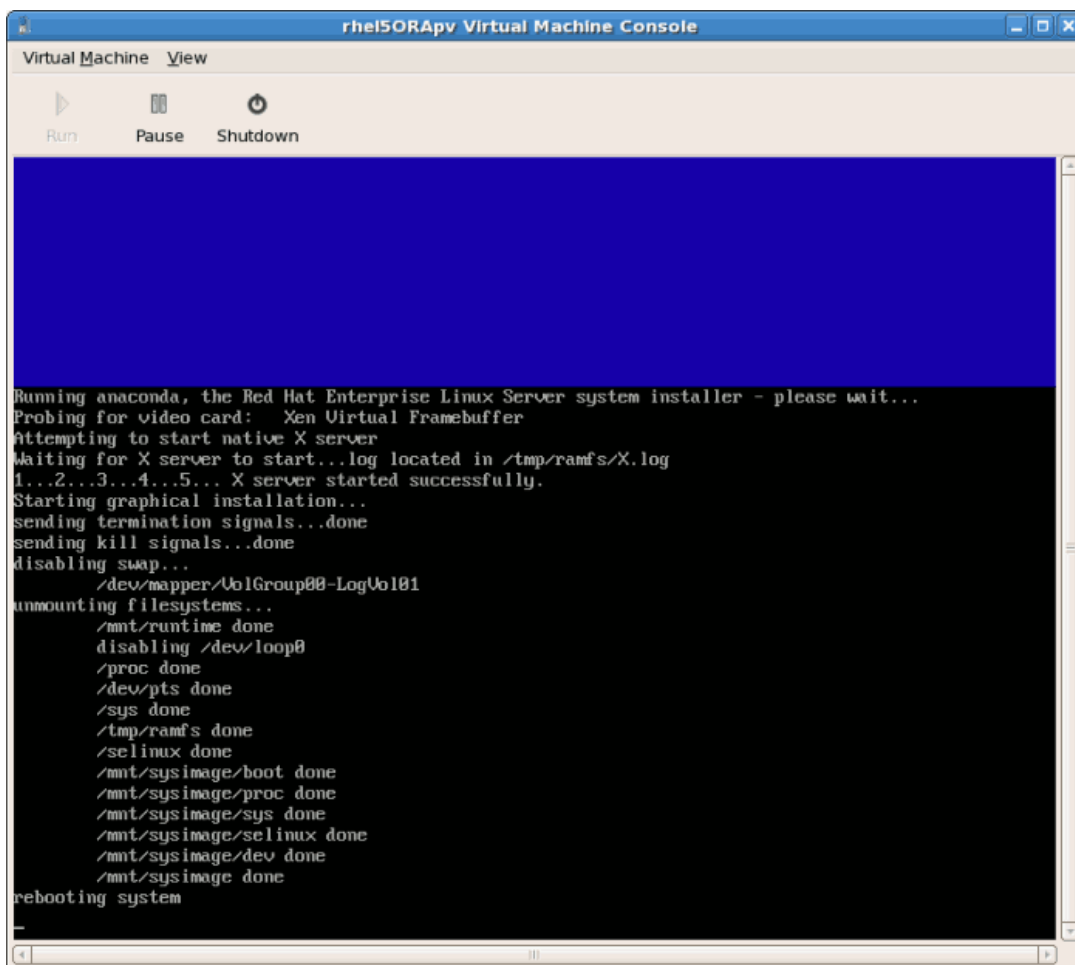
14. Now the installation will automatically install all of the selected software packages:



15. After the installation has finished you need to reboot your guest:



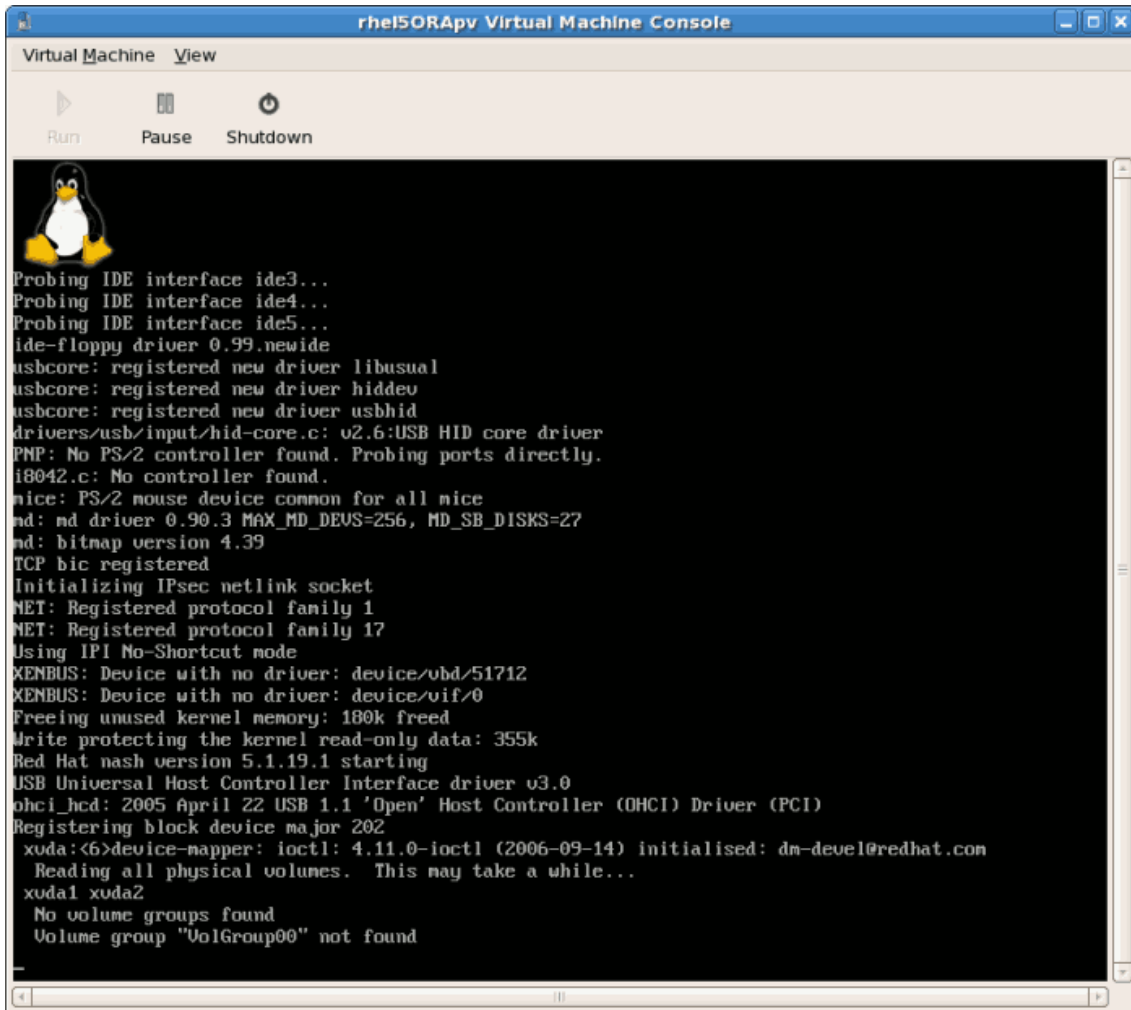
16. Your newly installed guest will not reboot, instead it will shutdown:



### 1.2. The first boot after the guest installation of Red Hat Enterprise Linux 5

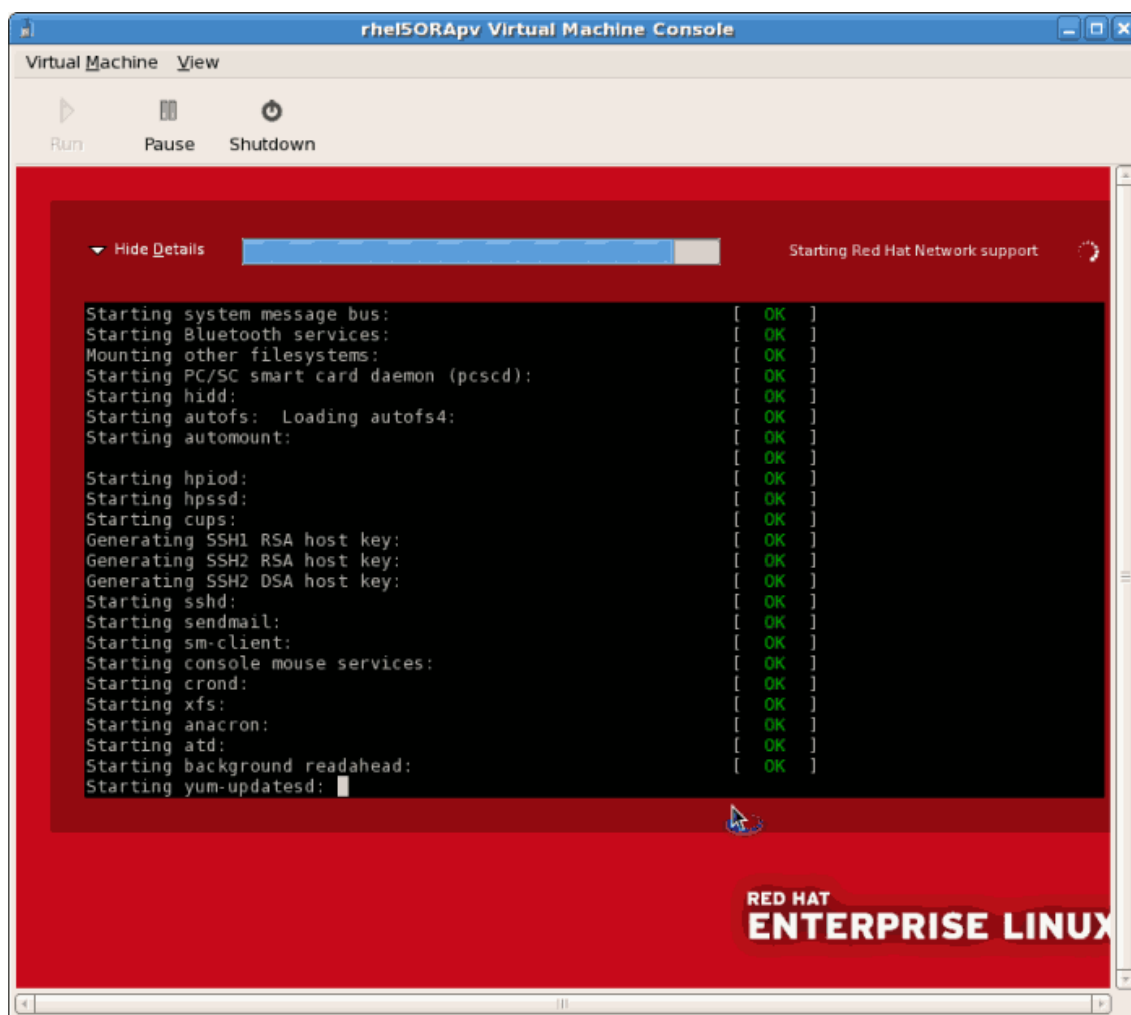
To reboot your new guest use the command `xm create GuestName` where *GuestName* is the name you entered during the initial installation. Guest configuration files are located in `/etc/xen/`.

Once you have started your guest using the command above you can use **virt-manager** to open a graphical console window for your guest. Start **virt-manager**, select your guest from the list and click the **Open** tab, you will see a window similar to the one below:



The image shows a 'Virtual Machine Console' window titled 'rheISOApv Virtual Machine Console'. It features a toolbar with 'Run', 'Pause', and 'Shutdown' buttons. The main area displays a black terminal window with a penguin icon in the top-left corner. The terminal output shows the progress of a Red Hat installation, including device probing, driver registration, and disk initialization.

```
Probing IDE interface ide3...
Probing IDE interface ide4...
Probing IDE interface ide5...
ide-floppy driver 0.99.newide
usbcore: registered new driver libusual
usbcore: registered new driver hiddev
usbcore: registered new driver usbhid
drivers/usb/input/hid-core.c: v2.6:USB HID core driver
PNP: No PS/2 controller found. Probing ports directly.
i8042.c: No controller found.
mice: PS/2 mouse device common for all mice
md: md driver 0.90.3 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: bitmap version 4.39
TCP bic registered
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI No-Shortcut mode
XENBUS: Device with no driver: device/vbd/51712
XENBUS: Device with no driver: device/vif/0
Freeing unused kernel memory: 180k freed
Write protecting the kernel read-only data: 355k
Red Hat nash version 5.1.19.1 starting
USB Universal Host Controller Interface driver v3.0
ohci_hcd: 2005 April 22 USB 1.1 'Open' Host Controller (OHCI) Driver (PCI)
Registering block device major 202
xoda:<6>device-mapper: ioctl: 4.11.0-ioctl (2006-09-14) initialised: dm-devel@redhat.com
Reading all physical volumes. This may take a while...
xoda1 xoda2
No volume groups found
Volume group "VolGroup00" not found
```

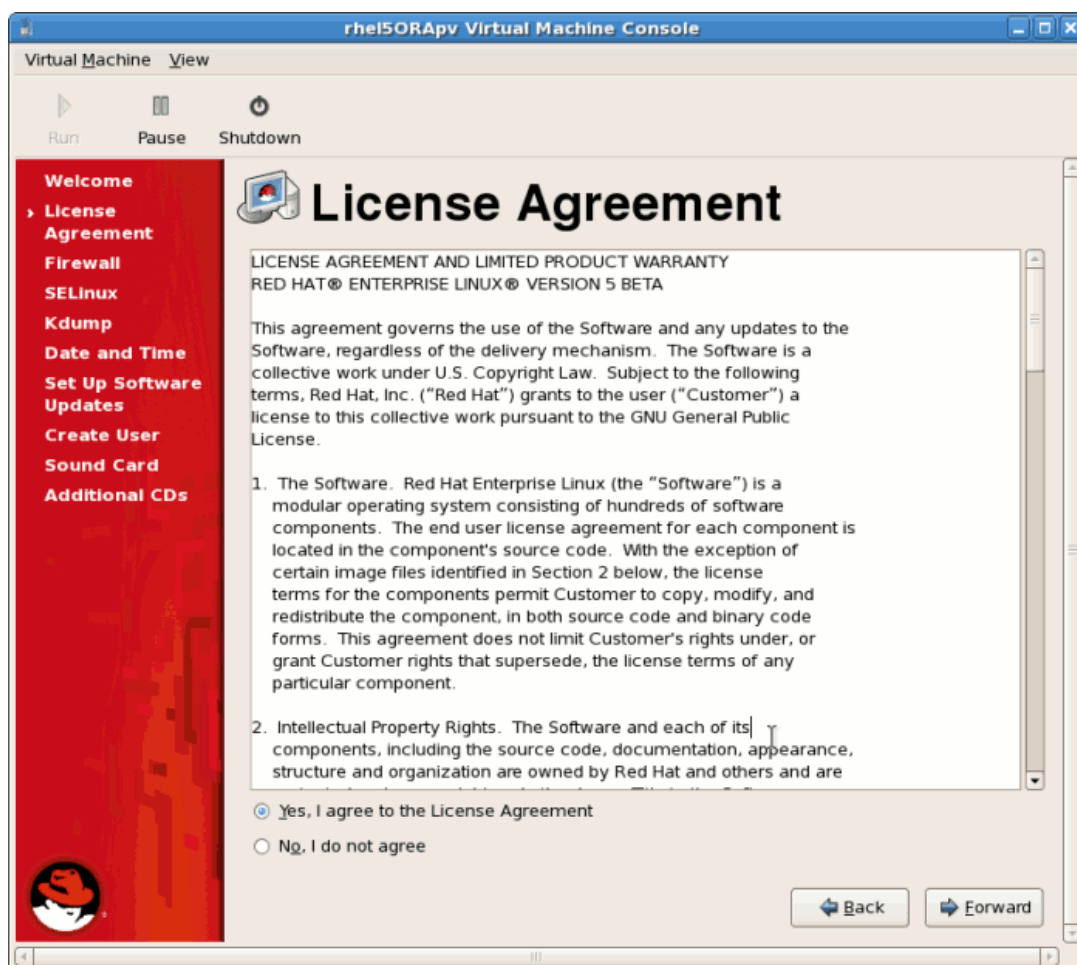


### 1.3. First boot configuration

During the first boot after the installation has finished you will see the *First Boot* configuration screen. This screen will walk you through some basic configuration choices for your guest:



1. First you need to agree to the license agreement:



2. Next is the Firewall configuration:

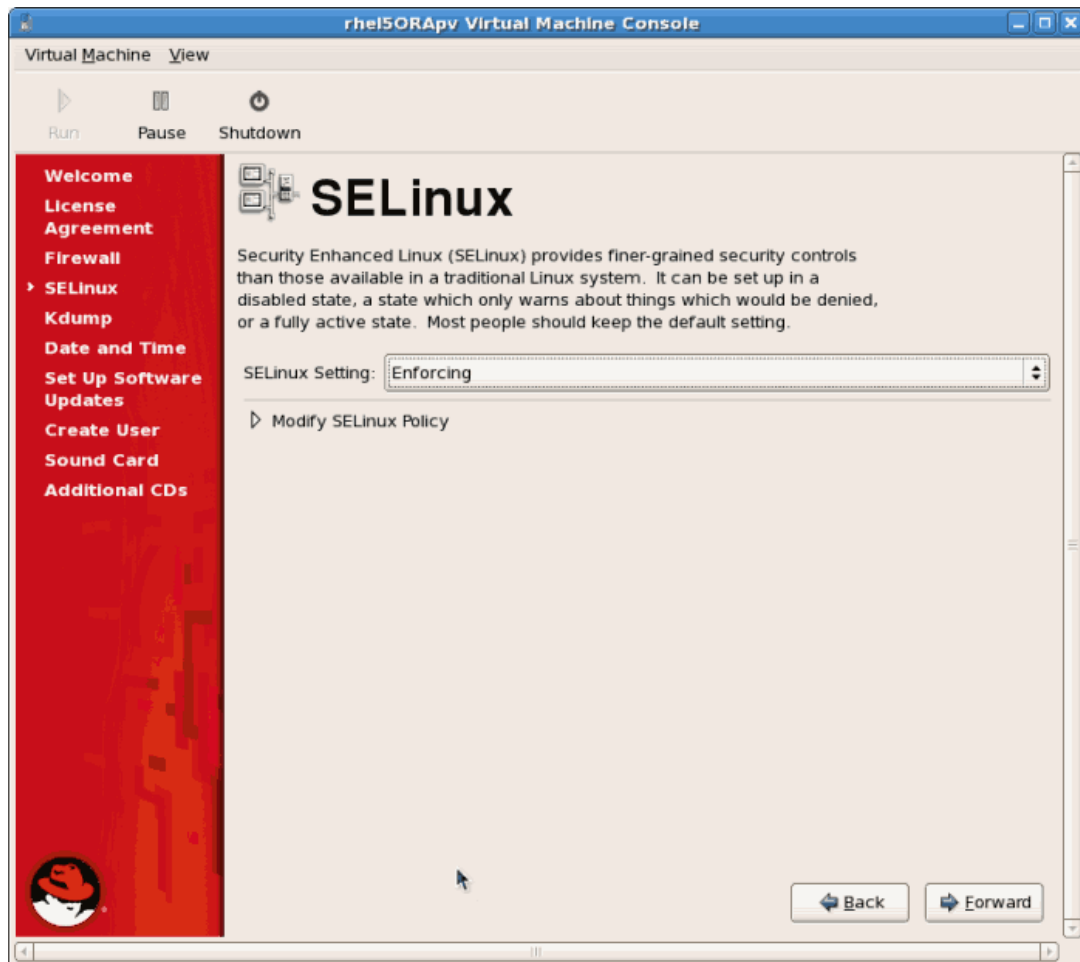




3. If you select to disable the Firewall configuration you need to confirm your choice one more time:



4. Next is SELinux. It is strongly recommended to run SELinux in enforcing mode but you can chose to either run SELinux in permissive mode or completely disable it:



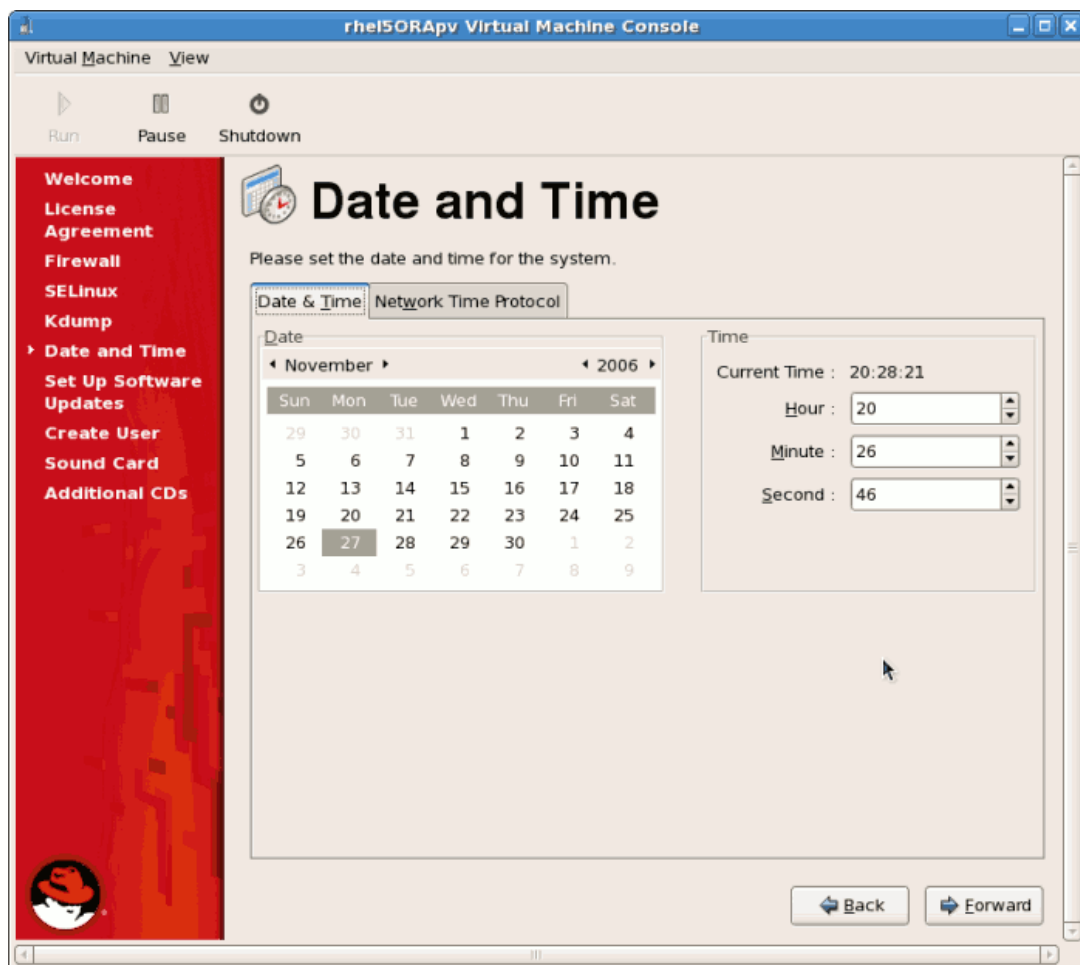
5. Choosing to disable SELinux causes the following warning to appear:



6. You can enable kdump:



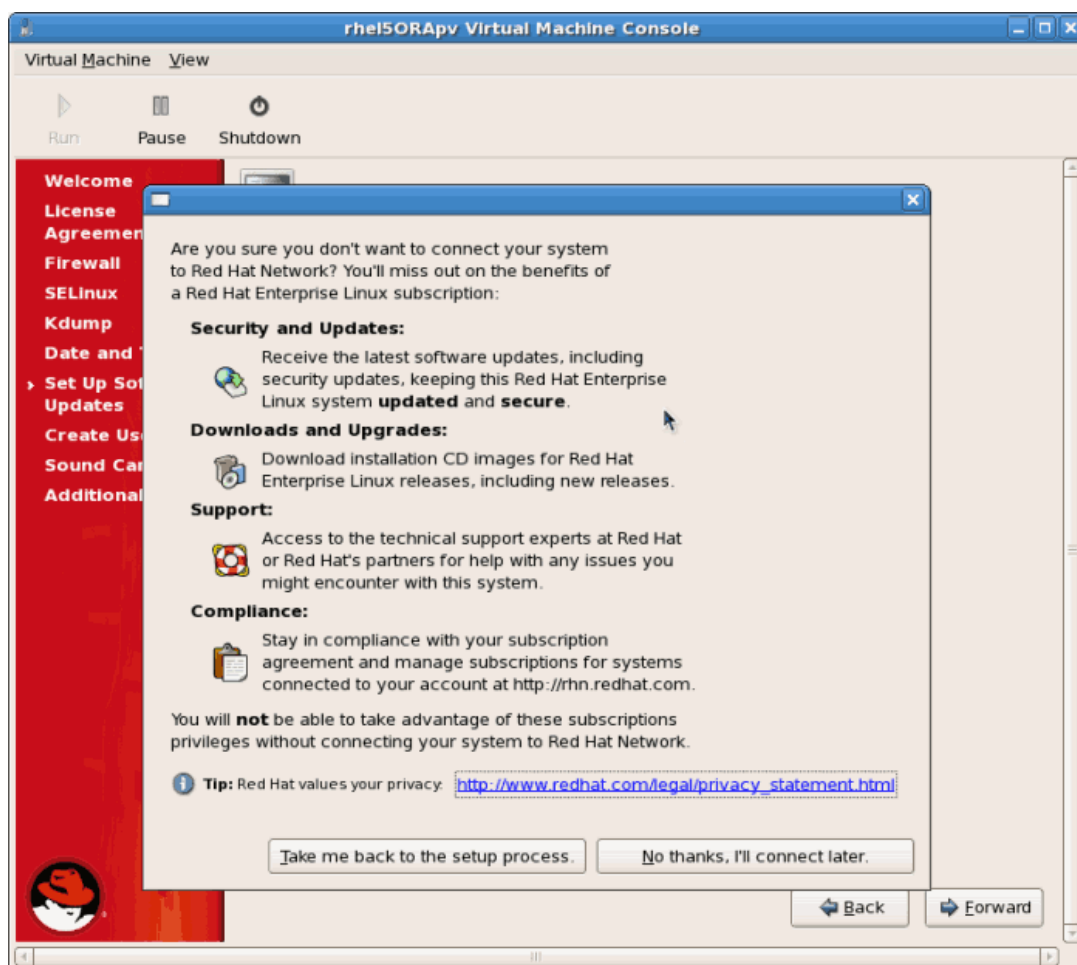
7. Confirm time and date are set correctly for your guest. If you install a para-virtualized guest time and date should be in sync with dom0/Hypervisor as the guest will get its time from dom0. A fully-virtualized guest may need additional configuration such as NTP to avoid skew in time from dom0:



- If you have a Red Hat Network subscription or would like to try one you can use the screen below to register your newly installed guest in RHN:

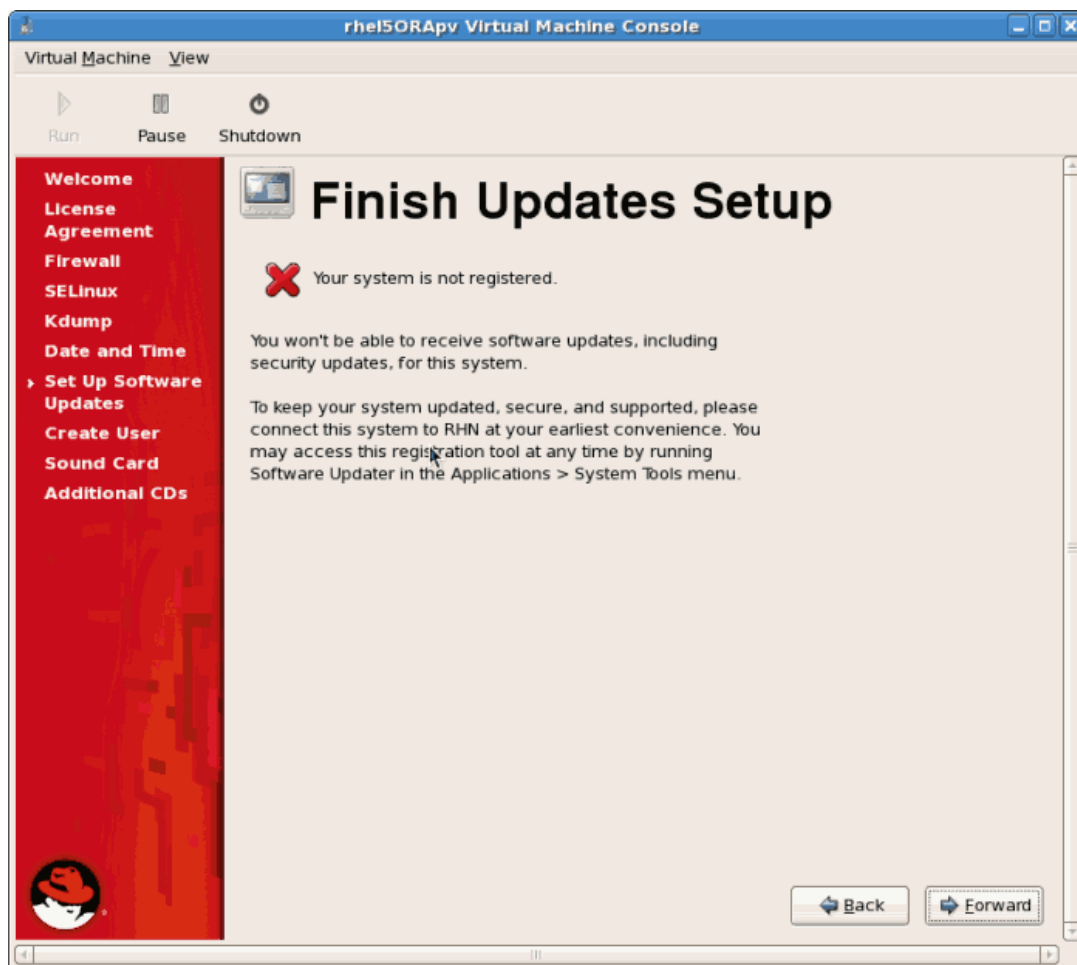


9. Confirm your choices for RHN:

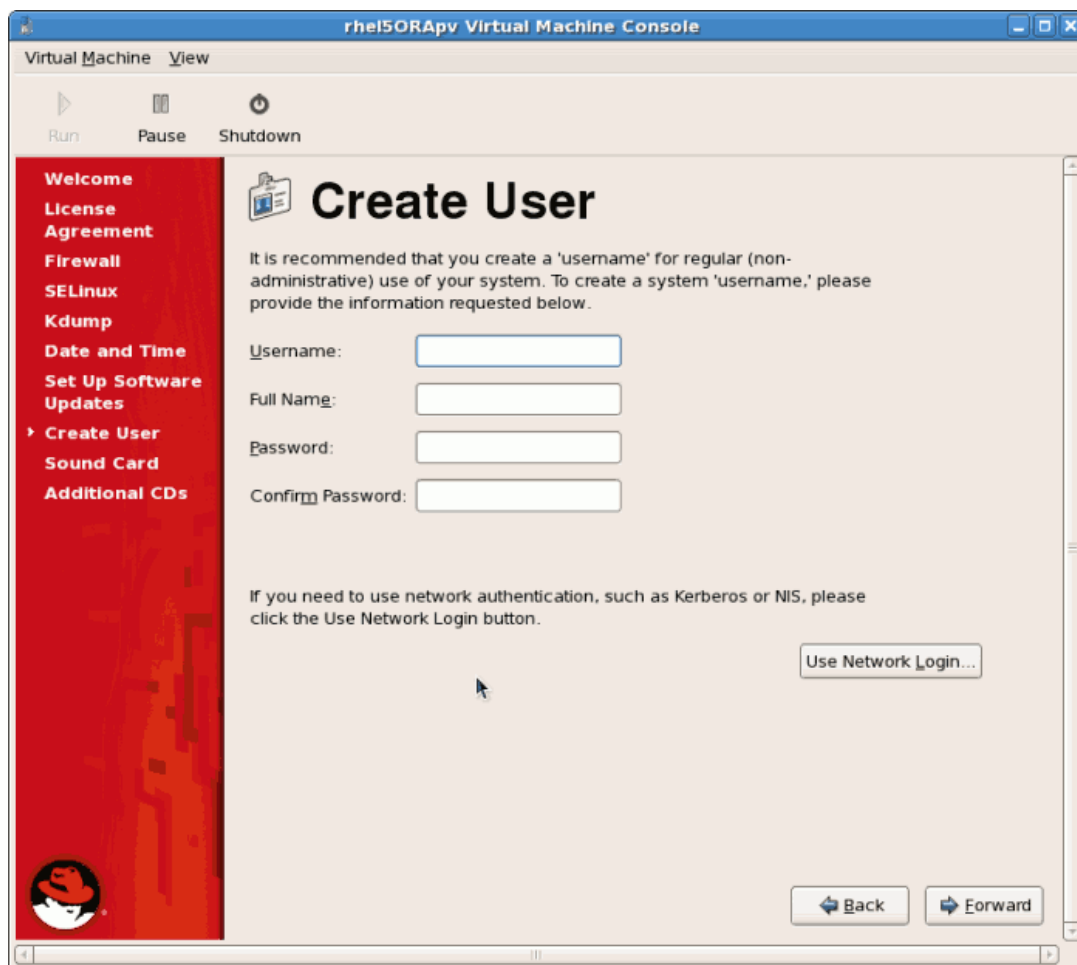


10. Once setup has finished you may see one more screen if you opted out of RHN at this time:

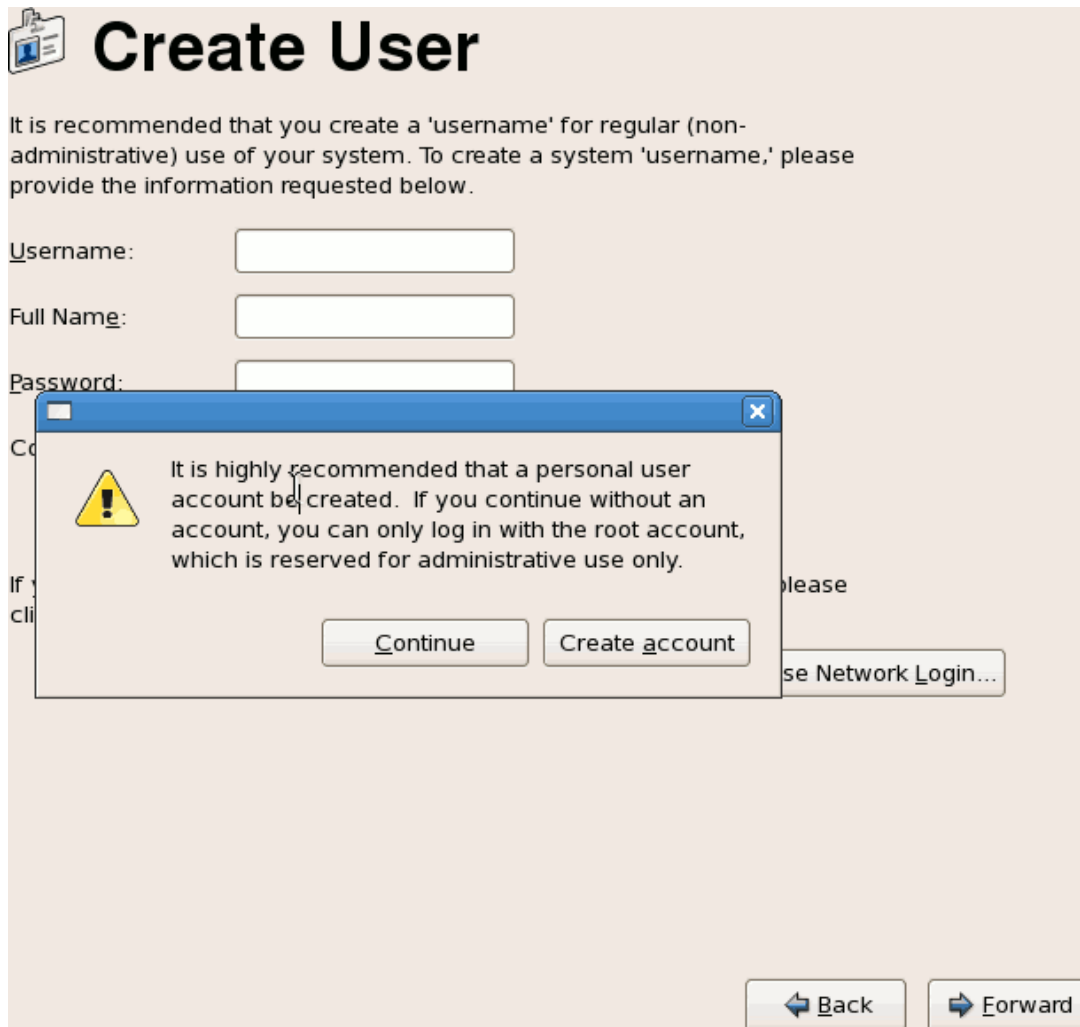




11. This screen will allow you to create a user besides the standard root account:



12. The following warning appears if you do not choose to create a personal account:



**Create User**

It is recommended that you create a 'username' for regular (non-administrative) use of your system. To create a system 'username,' please provide the information requested below.

Username:

Full Name:

Password:

Co

If y  
cli

It is highly recommended that a personal user account be created. If you continue without an account, you can only log in with the root account, which is reserved for administrative use only.

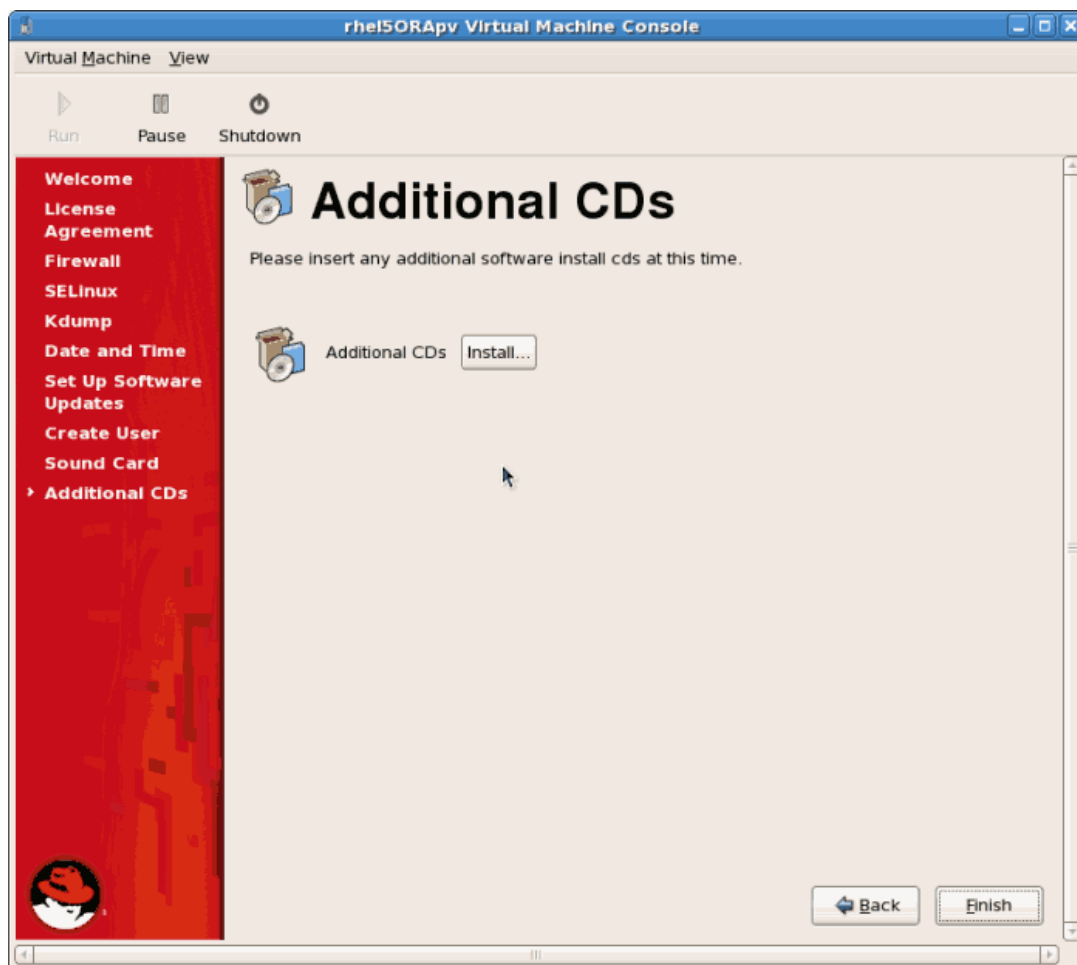
please

se Network Login...

13. If the install detects a sound device you would be asked to calibrate it on this screen:



14. If you want to install any additional software packages from CD you could do so on this screen. It is often more efficient to not install any additional software at this point but add it later using yum:



15. After you press the **Finish** button your guest will reconfigure any settings you may have changed and continue with the boot process:

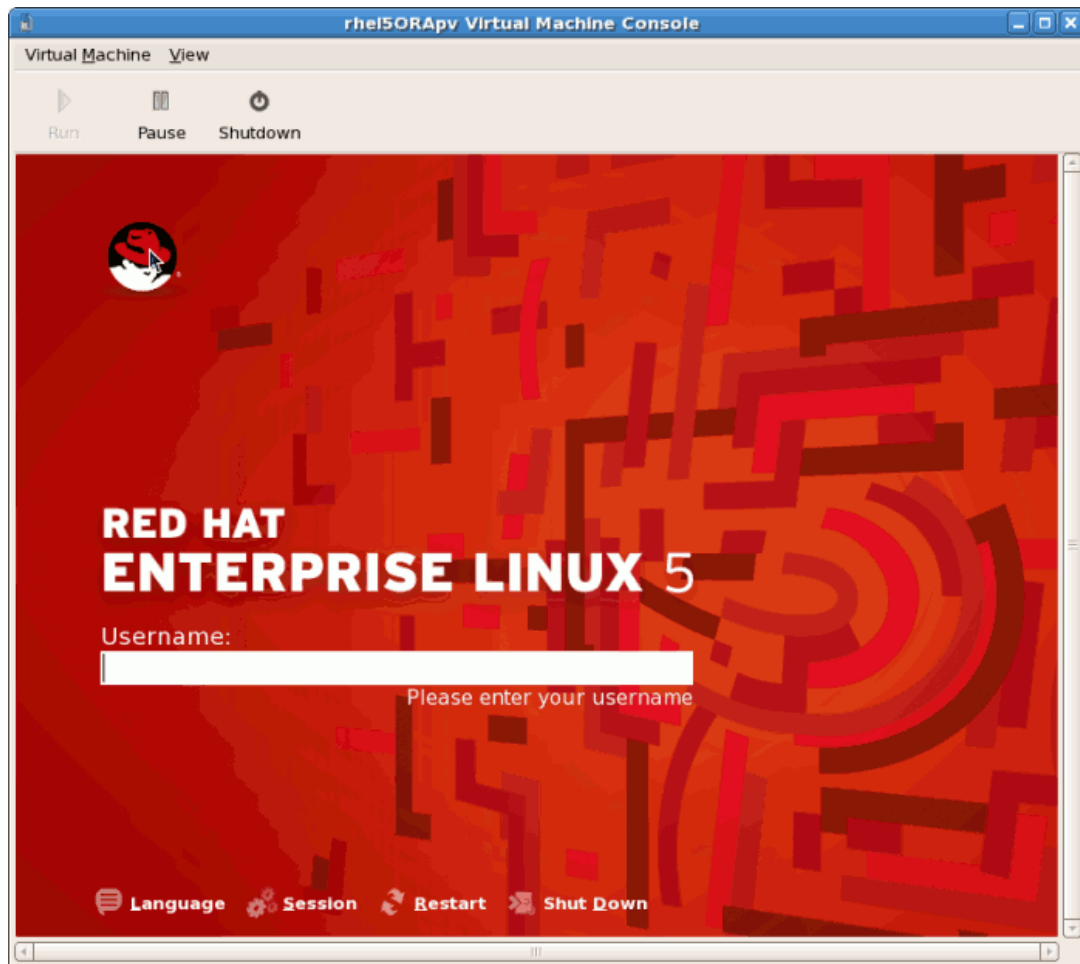
```

rhel5ORApy Virtual Machine Console
Virtual Machine View
Run Pause Shutdown

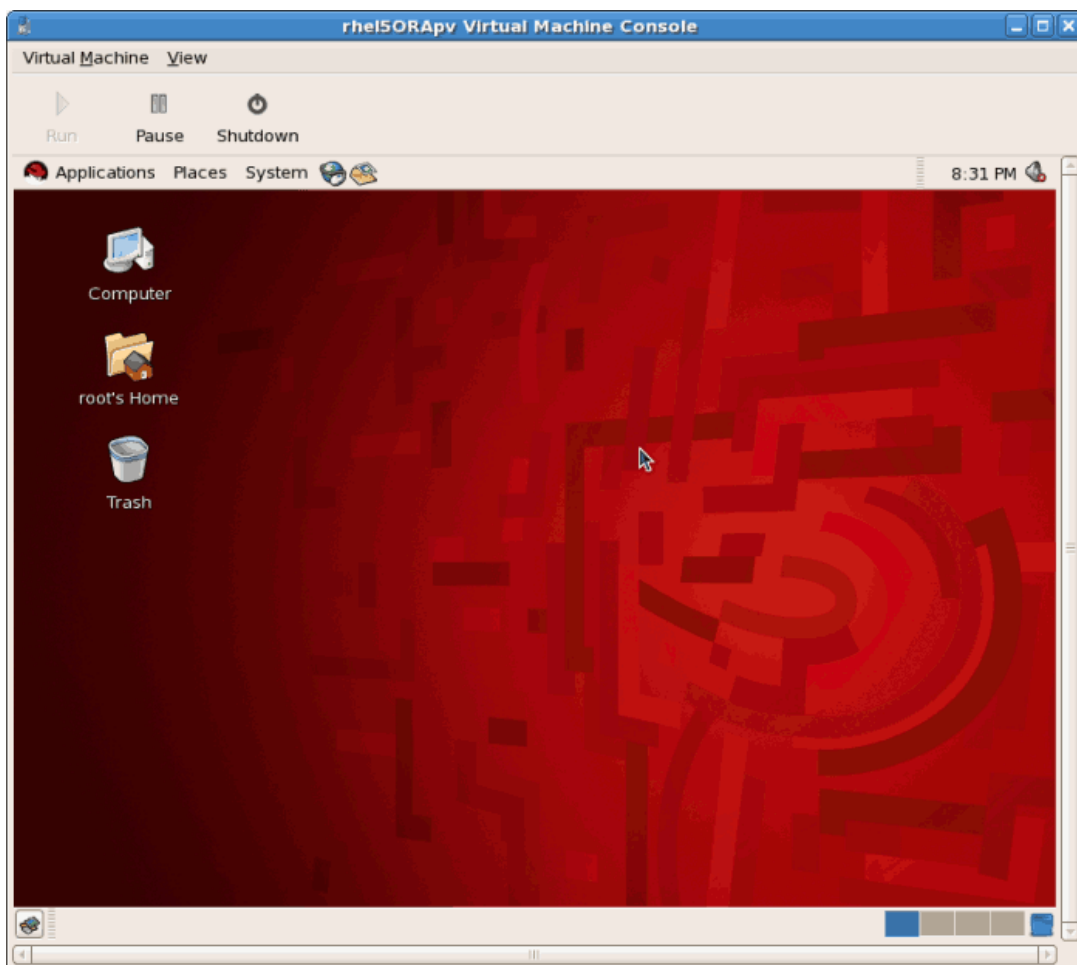
SELinux: Setting up existing superblocks.
SELinux: initialized (dev dm-0, type ext3), uses xattr
SELinux: initialized (dev tmpfs, type tmpfs), uses transition SIDs
SELinux: initialized (dev debugfs, type debugfs), uses genfs_contexts
SELinux: initialized (dev selinuxfs, type selinuxfs), uses genfs_contexts
SELinux: initialized (dev mqueue, type mqueue), uses transition SIDs
SELinux: initialized (dev devpts, type devpts), uses transition SIDs
SELinux: initialized (dev eventpollfs, type eventpollfs), uses task SIDs
SELinux: initialized (dev inotifyfs, type inotifyfs), uses genfs_contexts
SELinux: initialized (dev tmpfs, type tmpfs), uses transition SIDs
SELinux: initialized (dev futexfs, type futexfs), uses genfs_contexts
SELinux: initialized (dev pipefs, type pipefs), uses task SIDs
SELinux: initialized (dev sockfs, type sockfs), uses task SIDs
SELinux: initialized (dev cpuset, type cpuset), not configured for labeling
SELinux: initialized (dev proc, type proc), uses genfs_contexts
SELinux: initialized (dev bdev, type bdev), uses genfs_contexts
SELinux: initialized (dev rootfs, type rootfs), uses genfs_contexts
SELinux: initialized (dev sysfs, type sysfs), uses genfs_contexts
audit(1164677136.867:3): policy loaded auid=4294967295
SELinux: initialized (dev usbfs, type usbfs), uses genfs_contexts
Welcome to Red Hat Enterprise Linux Server
Press 'I' to enter interactive startup.
Setting clock (utc): Mon Nov 27 20:25:41 EST 2006 [ OK ]
Starting udev: [ OK ]
Loading default keymap (us): [ OK ]
Setting hostname localhost.localdomain: [ OK ]
Setting up Logical Volume Management: 2 logical volume(s) in volume group "VolGroup00" now active [ OK ]
Checking filesystems [ OK ]
Remounting root filesystem in read-write mode: [ OK ]
Mounting local filesystems: [ OK ]
Enabling local filesystem quotas: [ OK ]
Enabling /etc/fstab swaps: [ OK ]
audit(1164677411.468:10): user pid=2372 uid=0 auid=4294967295 subj=system_u:system_r:hwclock_t:s0 ms
g='changing system time: exe="/sbin/hwclock" (hostname=?, addr=?, terminal=? res=failed)'

```

16. And finally you will be greeted with the Red Hat Enterprise Linux 5 login screen:



17. After you have logged in you will see the standard Red Hat Enterprise Linux 5 desktop environment:



## 2. Installing a Windows XP Guest as a fully virtualized guest

**virt-manager** can install Windows XP as a fully-virtualized guest. However there are some extra steps which must be followed in order to complete the installation successfully. This section explains the installation process and extra steps needed to get Windows XP fully virtualized guests installed.

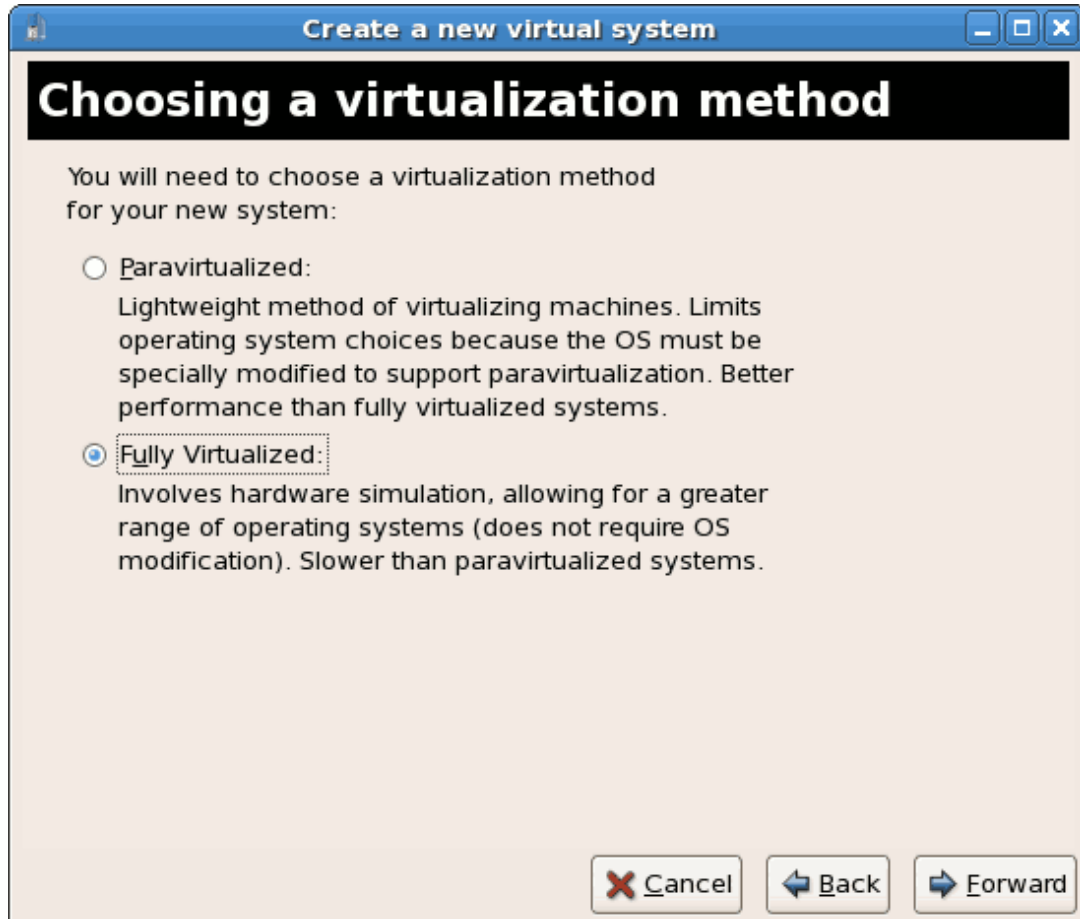


### Itanium® support

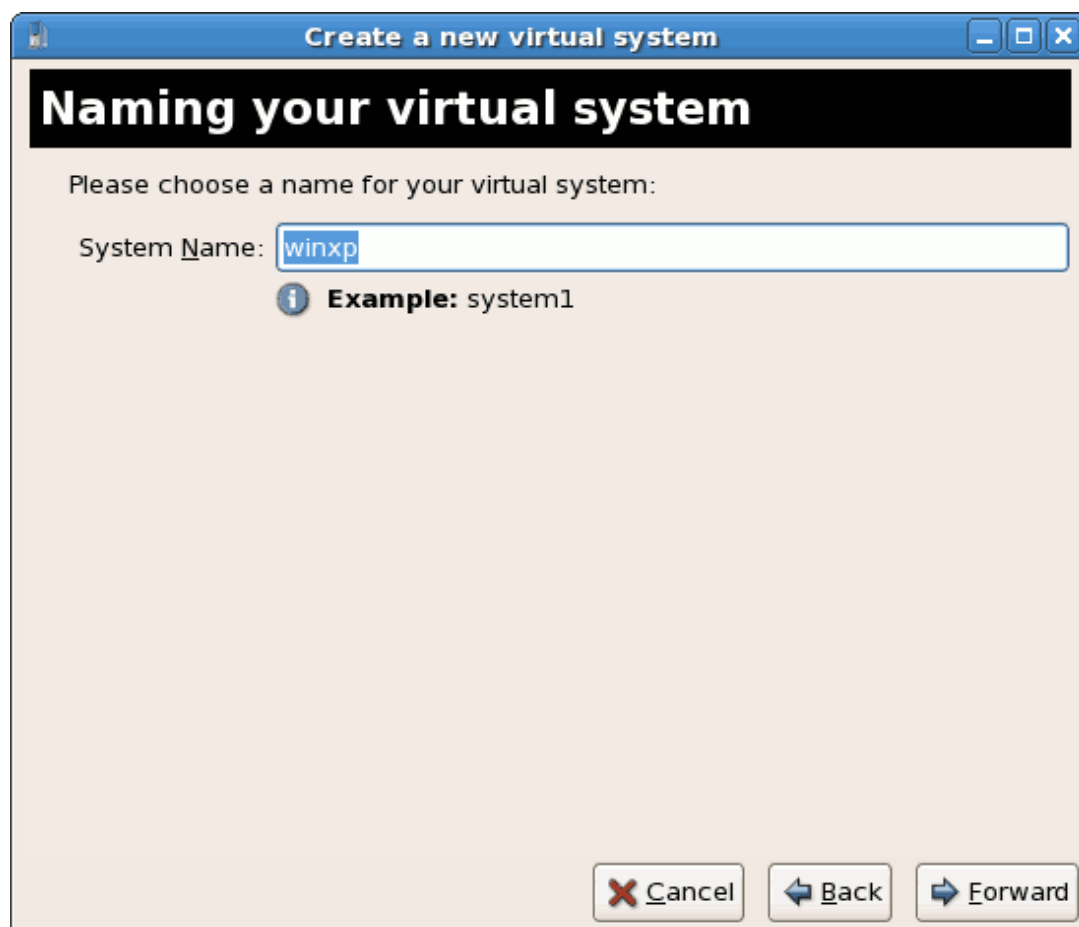
Presently, Red Hat Enterprise Linux hosts on the Itanium® architecture do not support fully virtualized windows guests. This section only applies to x86 and x86-64 hosts.



1. First you start **virt-manager** and select the **New** tab to create a new virtual machine. As you install a Windows based virtual machine you need to select the option to install a **Fully virtualized** guest:



2. Next you choose a descriptive system name:



3. Specify the location for the ISO image you want to use for your Windows installation:



4. Select the storage backing store, either a file based image can be used or a partition or logical volume:

The screenshot shows a window titled "Create a new virtual system" with standard window controls. The main heading is "Assigning storage space". Below this, a paragraph explains that the user needs to indicate how to assign space on the physical host system for the new virtual system, noting that this space will be used to install the operating system.

There are two radio button options:

- ☐ Normal Disk Partition:
  - A text field labeled "Partition:" is empty, followed by a "Browse..." button.
  - An information icon (i) is followed by the text "Example: /dev/hdc2".
- ☒ Simple File:
  - A text field labeled "File Location:" contains the path "/xen/images/winxp.dsk", followed by a "Browse..." button.
  - A text field labeled "File Size:" contains the value "4000", followed by a spin button and the unit "MB".
  - An information icon (i) is followed by the text "Note: File size parameter is only relevant for new files".

At the bottom, there is a "Tip" icon (i) followed by the text: "You may add additional storage, including network-mounted storage, to your virtual system after it has been created using the same tools you would on a physical system."

At the bottom right, there are three buttons: "Cancel" (with a red X icon), "Back" (with a left arrow icon), and "Forward" (with a right arrow icon).

5. Specify the virtual machine resources such as CPU and Memory:

The screenshot shows a window titled "Create a new virtual system" with a sub-header "Allocate memory and CPU". The "Memory:" section instructs the user to enter memory configuration for the VM, noting the total host memory is 2046 GB. It features input fields for "VM Max Memory" (set to 500) and "VM Startup Memory" (set to 500). The "CPUs:" section instructs the user to enter the number of virtual CPUs, showing "Logical host CPUs: 2" and a "VCPU" input field set to 1. A tip icon and text advise that the number of virtual CPUs should be less than or equal to the number of logical CPUs on the host. At the bottom are "Cancel", "Back", and "Forward" buttons.

**Create a new virtual system**

## Allocate memory and CPU

**Memory:**

Please enter the memory configuration for this VM. You can specify the maximum amount of memory the VM should be able to use, and optionally a lower amount to grab on startup.

Total memory on host machine: 2046 GB

VM Max Memory: 500

VM Startup Memory: 500

**CPUs:**

Please enter the number of virtual CPUs this VM should start up with.

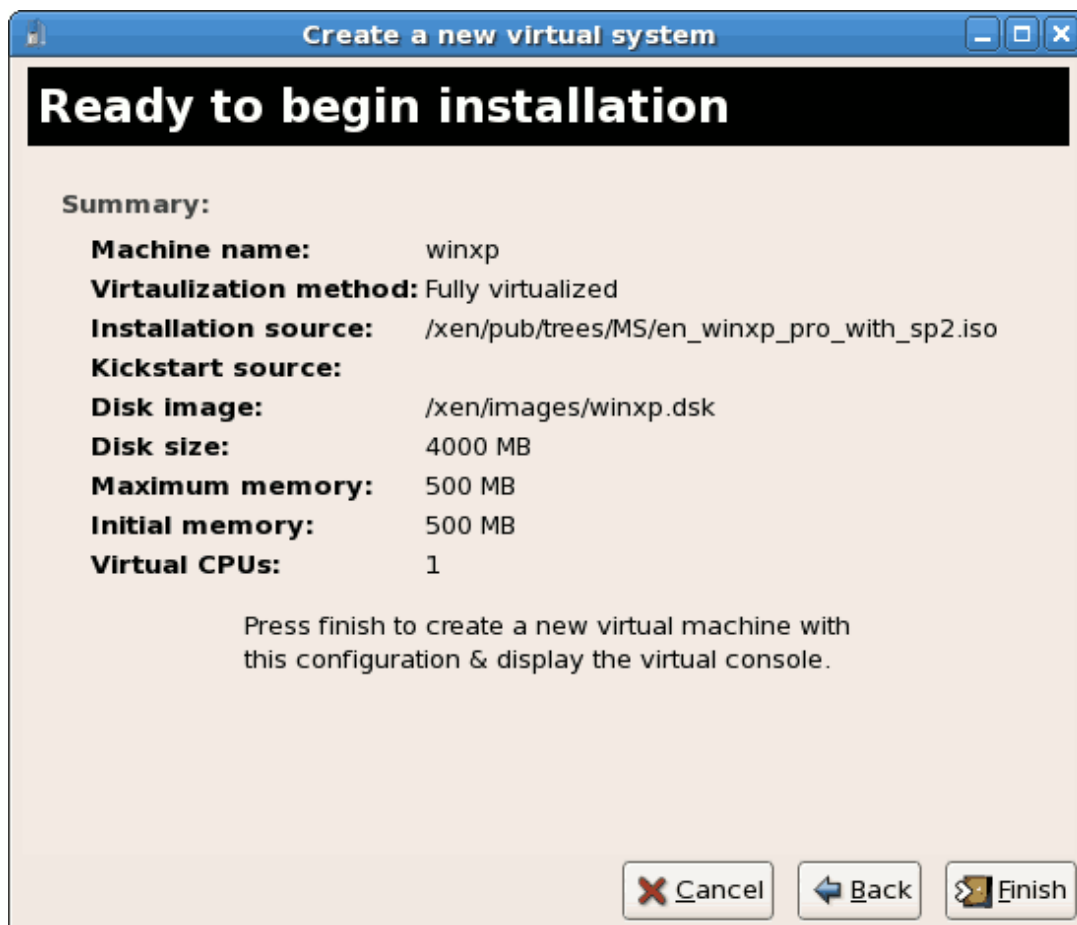
Logical host CPUs: 2

VCPUs: 1

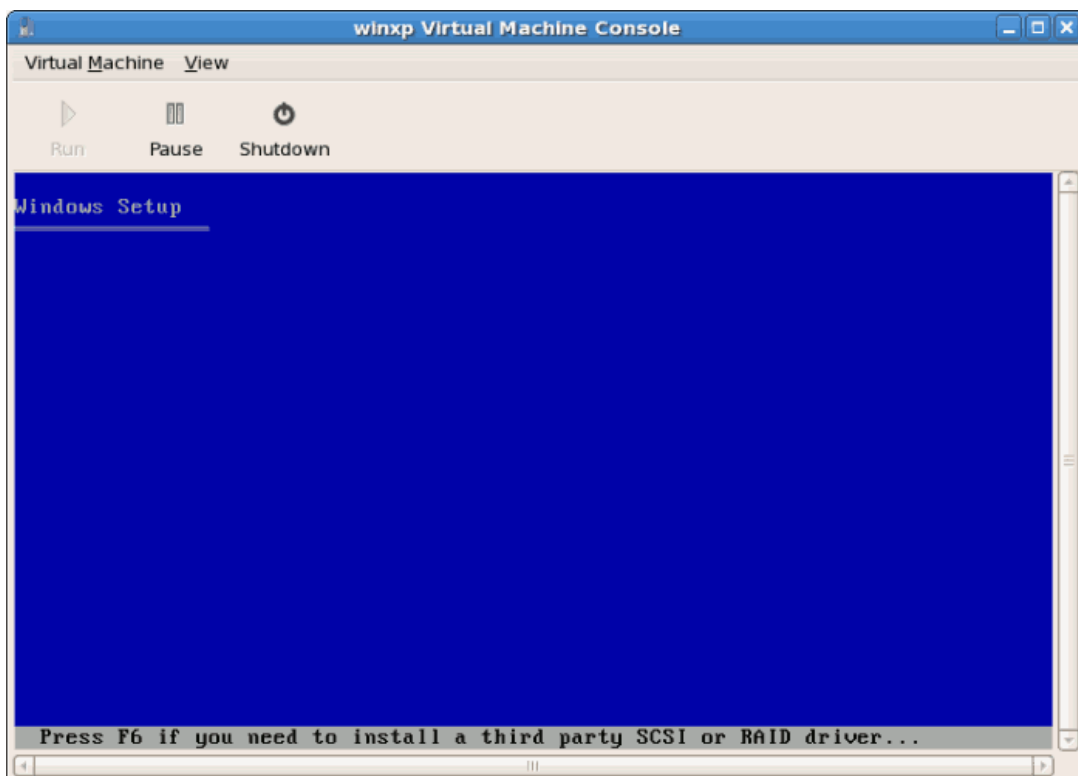
**Tip:** For best performance, the number of virtual CPUs should be less than (or equal to) the number of logical CPUs on the host system.

**Buttons:** Cancel, Back, Forward

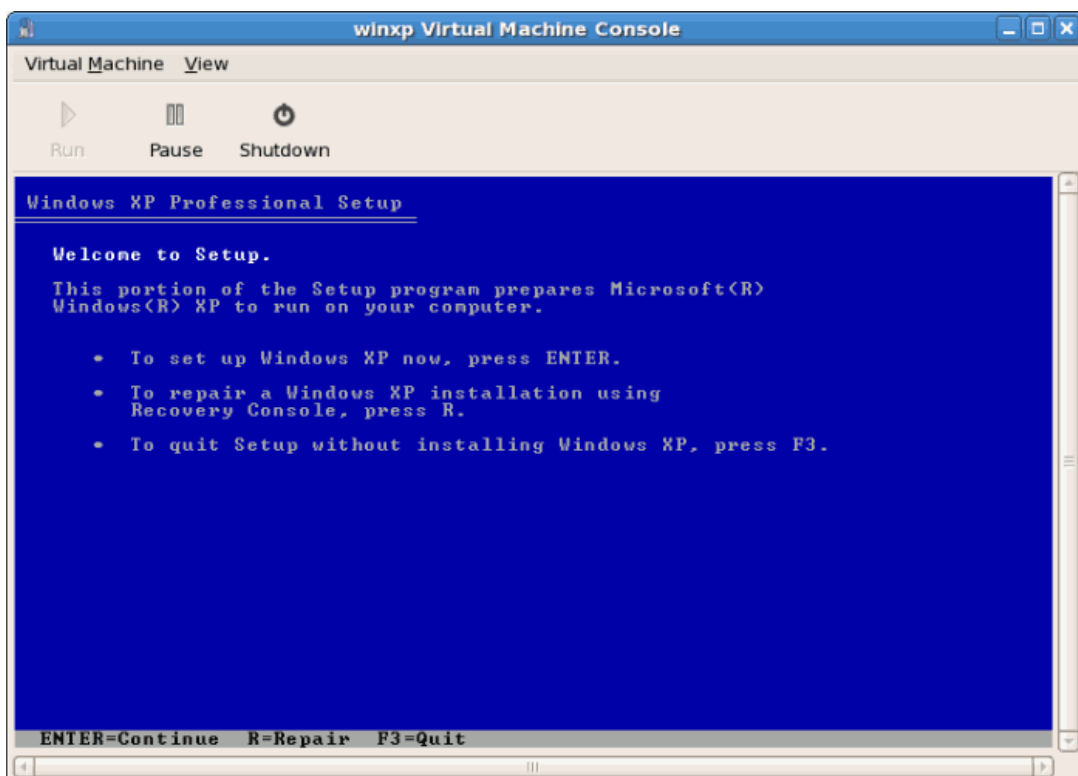
- Before the installation will continue you will see the summary screen. Press **Finish** to proceed to the actual installation:

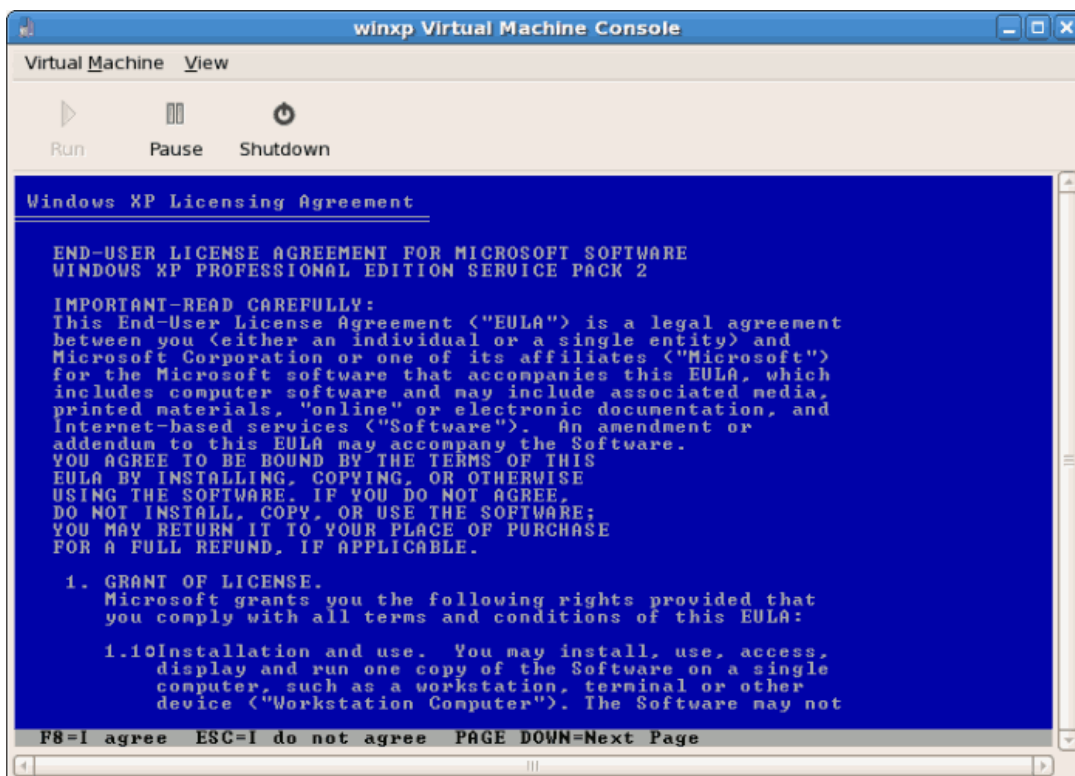


7. Now the actual Windows installation will start. As you need to make a hardware selection it is important that you open a console window very quickly after the installation has started. Once you press **Finish** make sure you set focus to the **virt-manager** summary window and select your newly started Windows guest. Double click on the system name and a console window will open. Quickly press **F5** to select a new HAL, once you get the dialog box in the Windows install select the 'Generic i486 Platform' tab (you can scroll through the selections using the Up and Down arrows).

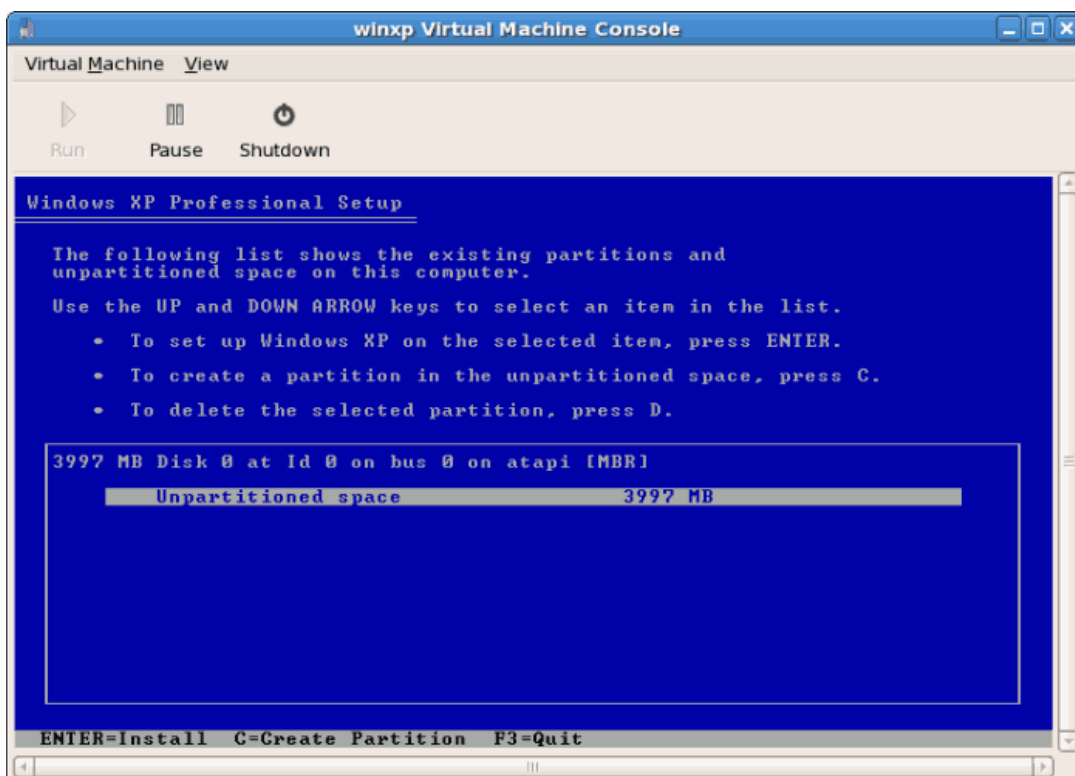


8. The installation will proceed like any other standard Windows installation:

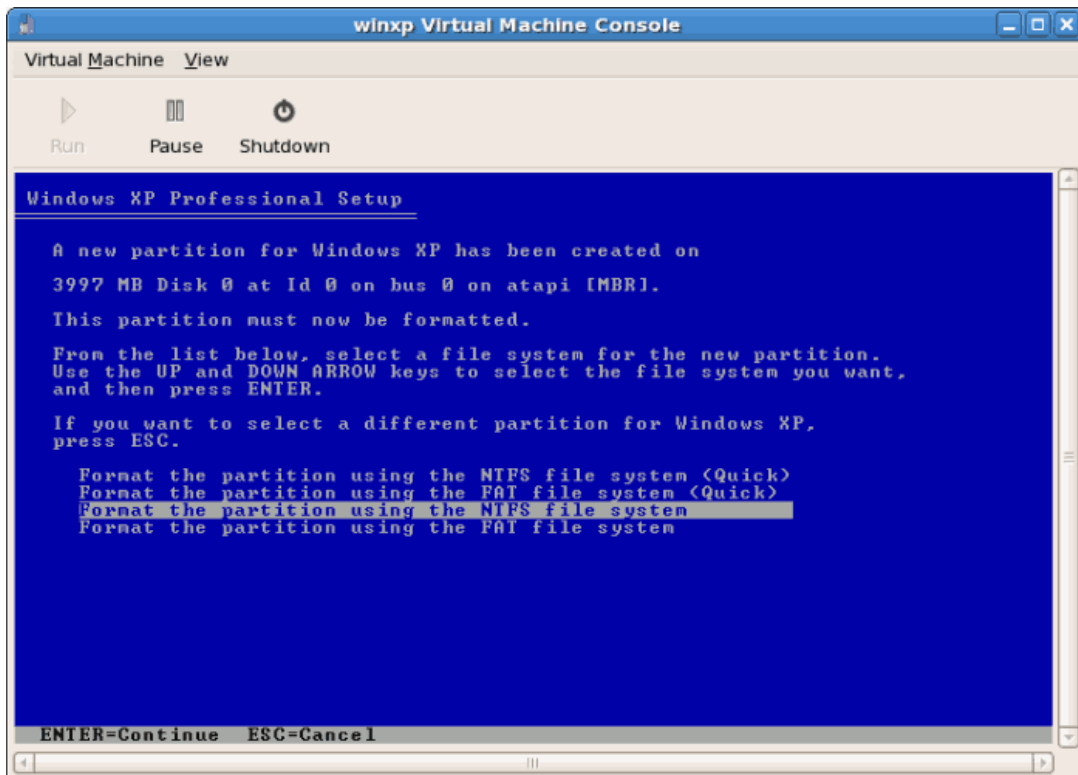




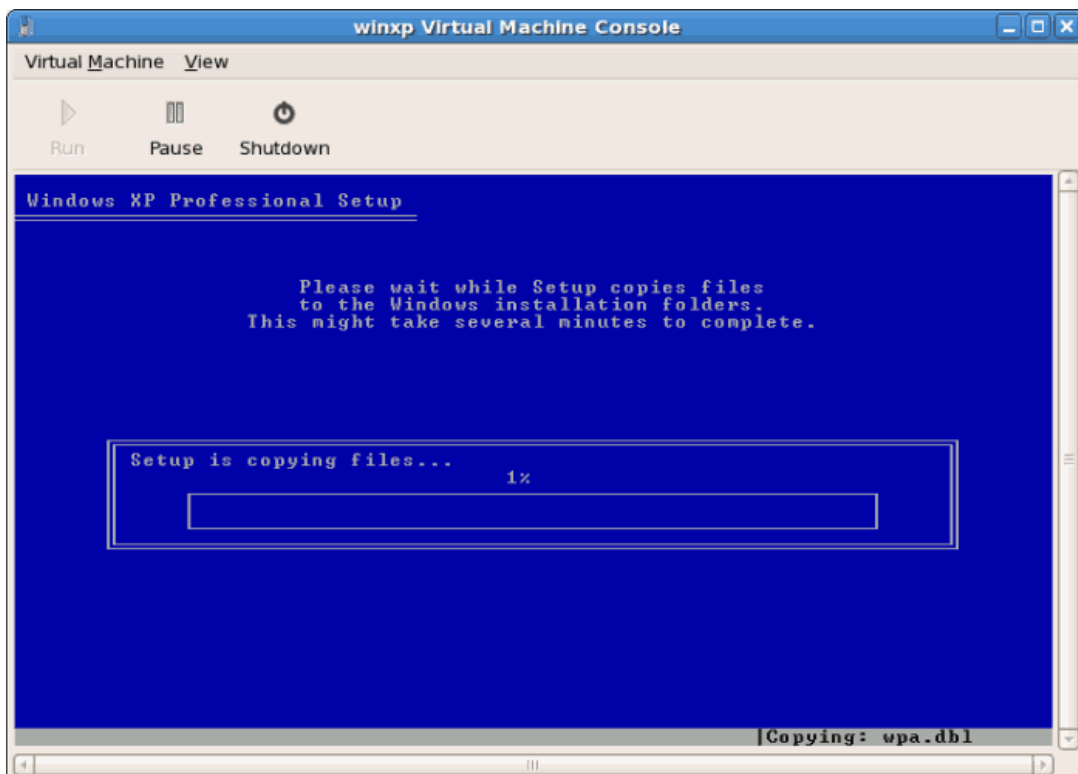
9. You will be asked to partition your drive:

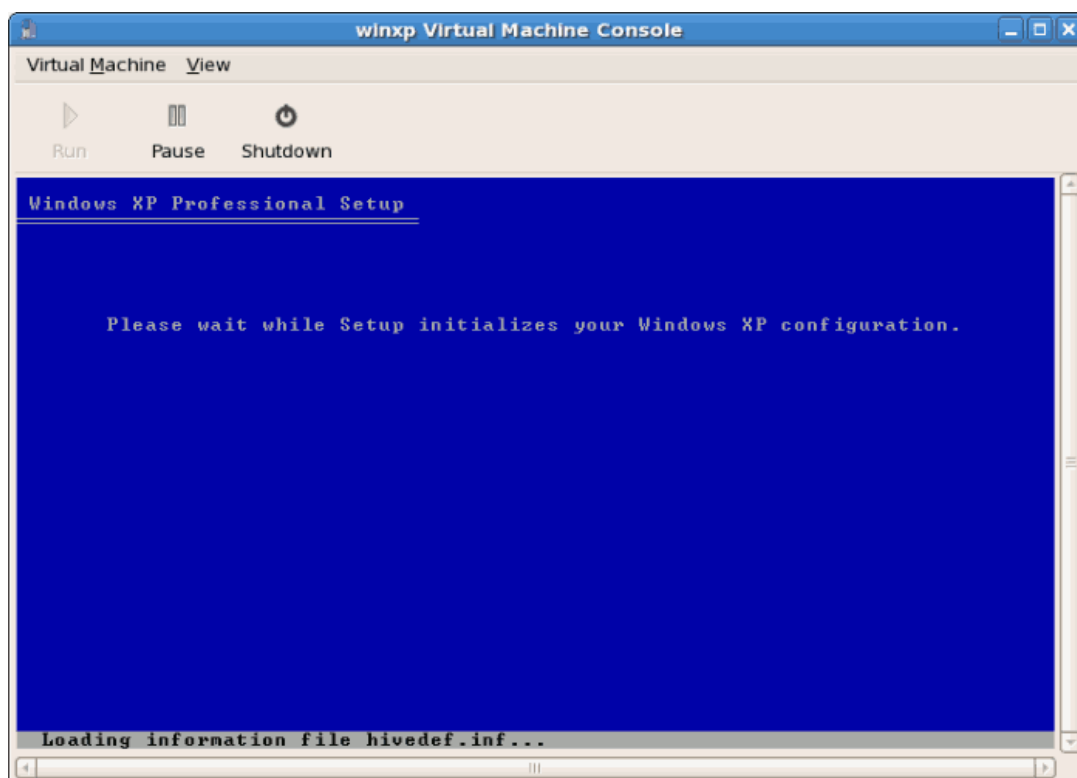




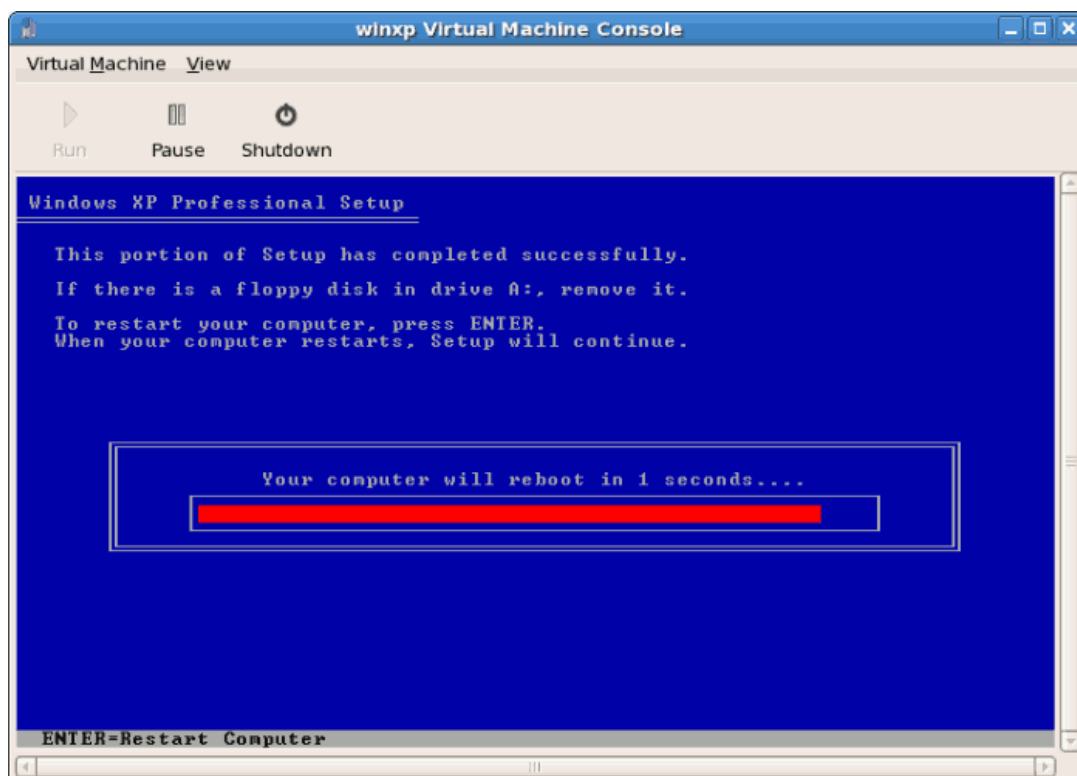


10. After your drive has been formatted Windows will start copying the files onto your new hard drive:





11. After setup has completed your Windows virtual machine will be rebooted:



12. You will have to halt the virtual machine after its initial reboot as you need to manually edit

the virtual machine's configuration file which is located in `/etc/xen/VirtualMachineName`. You can halt the virtual machine using the `xm destroy WindowsGuest` command, where `WindowsGuest` is the name of your virtual machine.

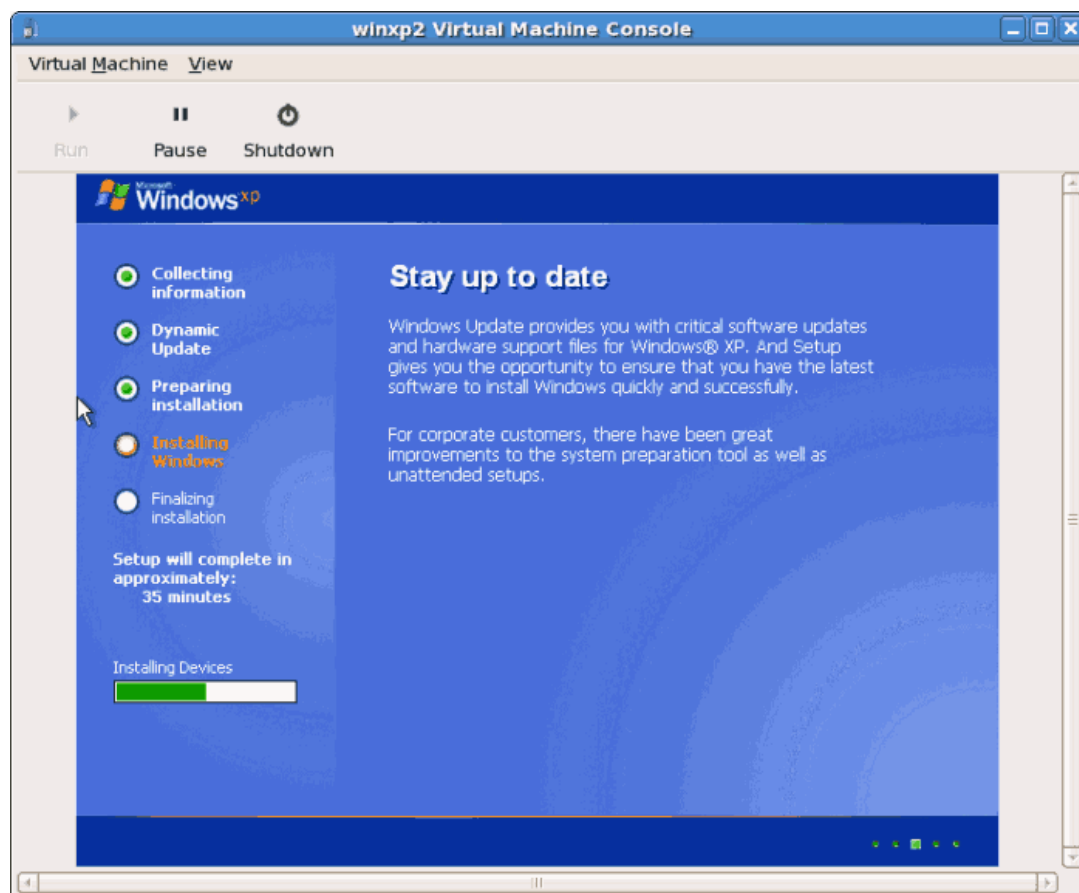
13. You will need to modify the `disk` entry and add a `cdrom` entry to the config file. The old entry will look similar to the following:

```
disk = [ 'file:/var/lib/xen/images/winxp.dsk,hda,w' ]
```

and the new entry should look like the following:

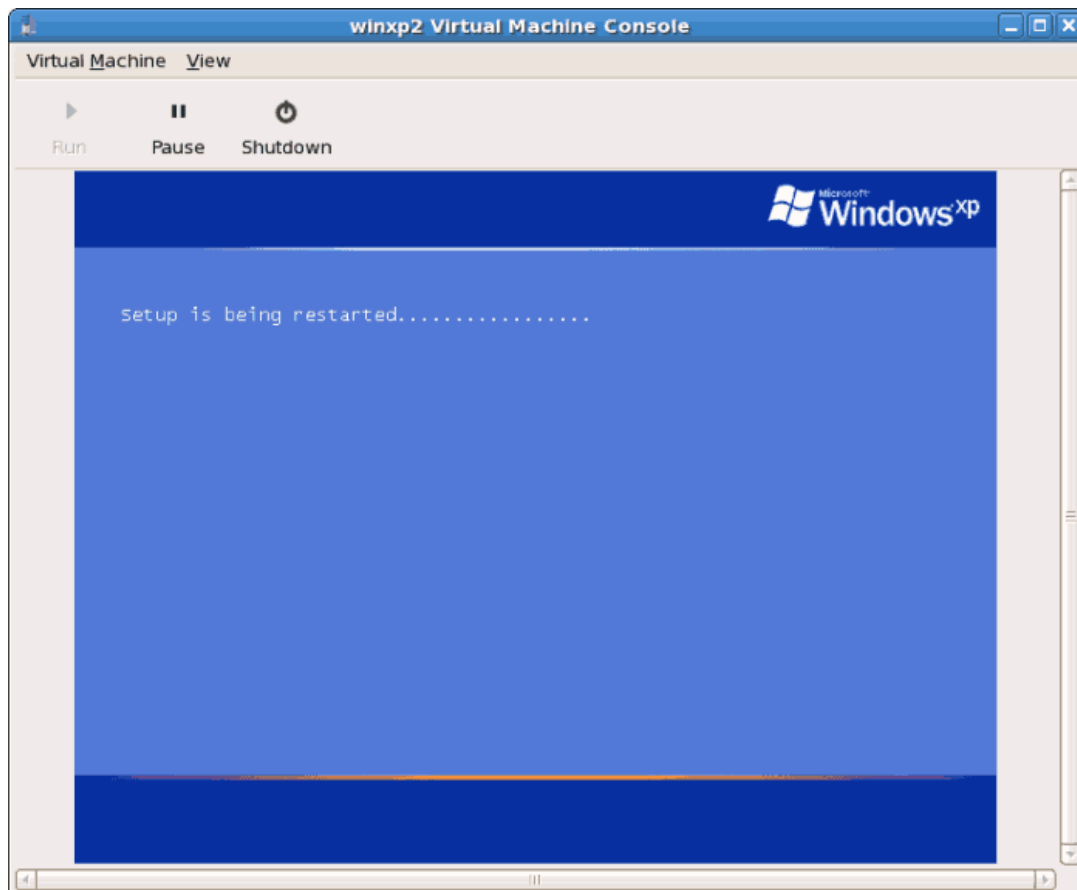
```
disk = [ 'file:/var/lib/xen/images/winxp.dsk,hda,w' ,  
        'file:/xen/pub/trees/MS/en_winxp_pro_with_sp2.iso,hdc:cdrom,r', ]
```

14. Now you can restart your Windows virtual machine using the `xm create WindowsGuest` command, where `WindowsGuest` is the name of your virtual machine.
15. Once you open the console window you will see Windows continuing with the setup phase:

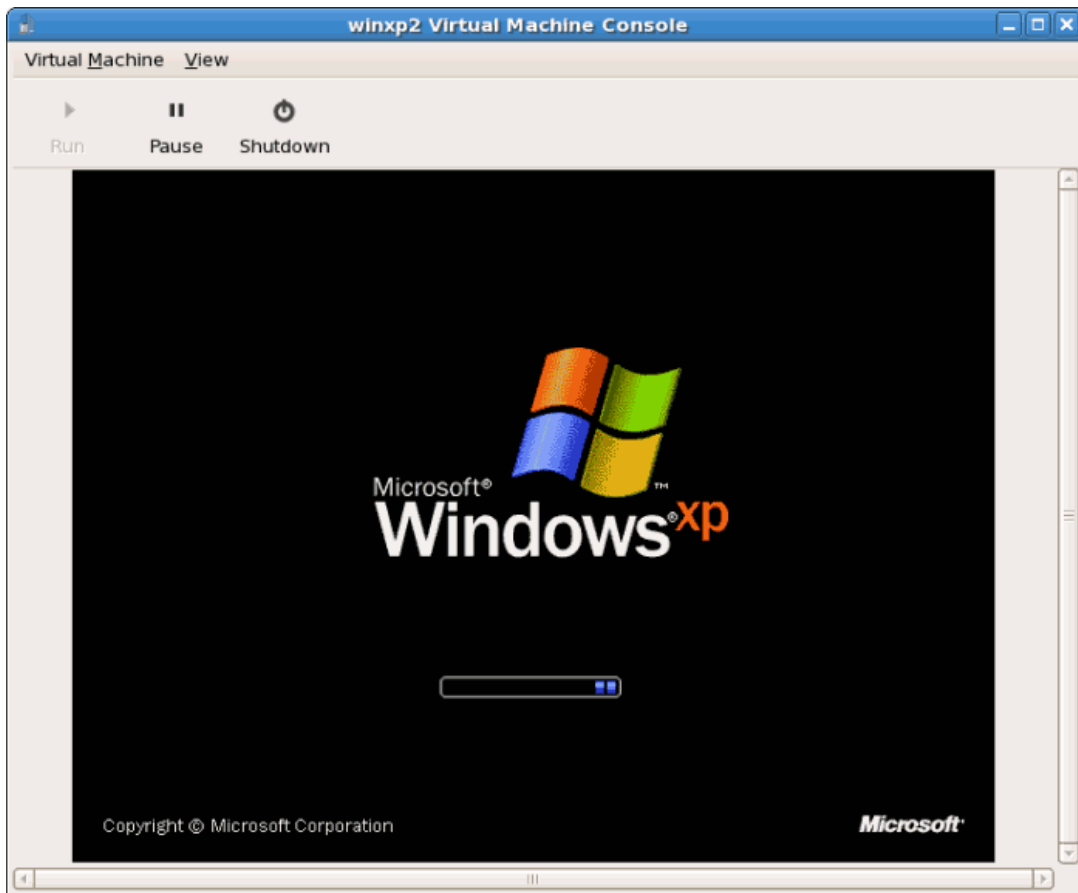


16. If your installation seems to get stuck during the setup phase you can restart the virtual

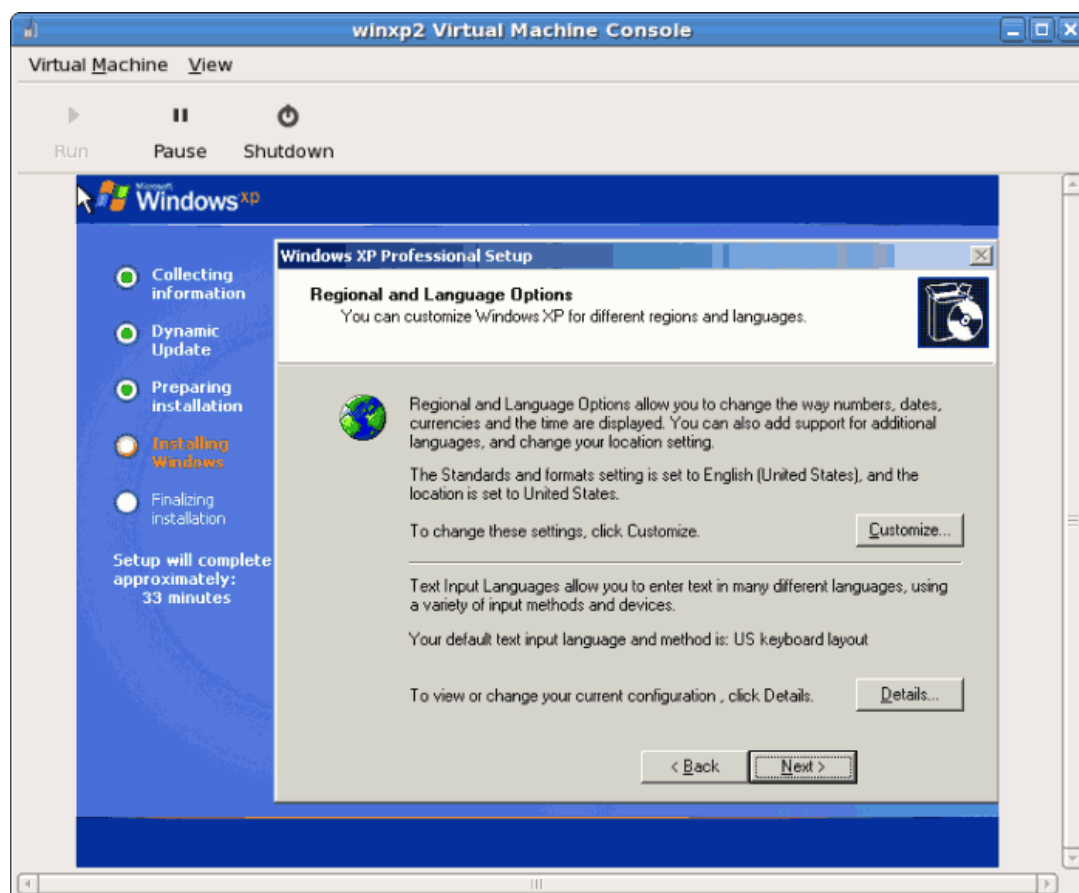
machine using the command mentioned above. This will usually get the installation to continue. As you restart the virtual machine you will see a Setup is being restarted message:



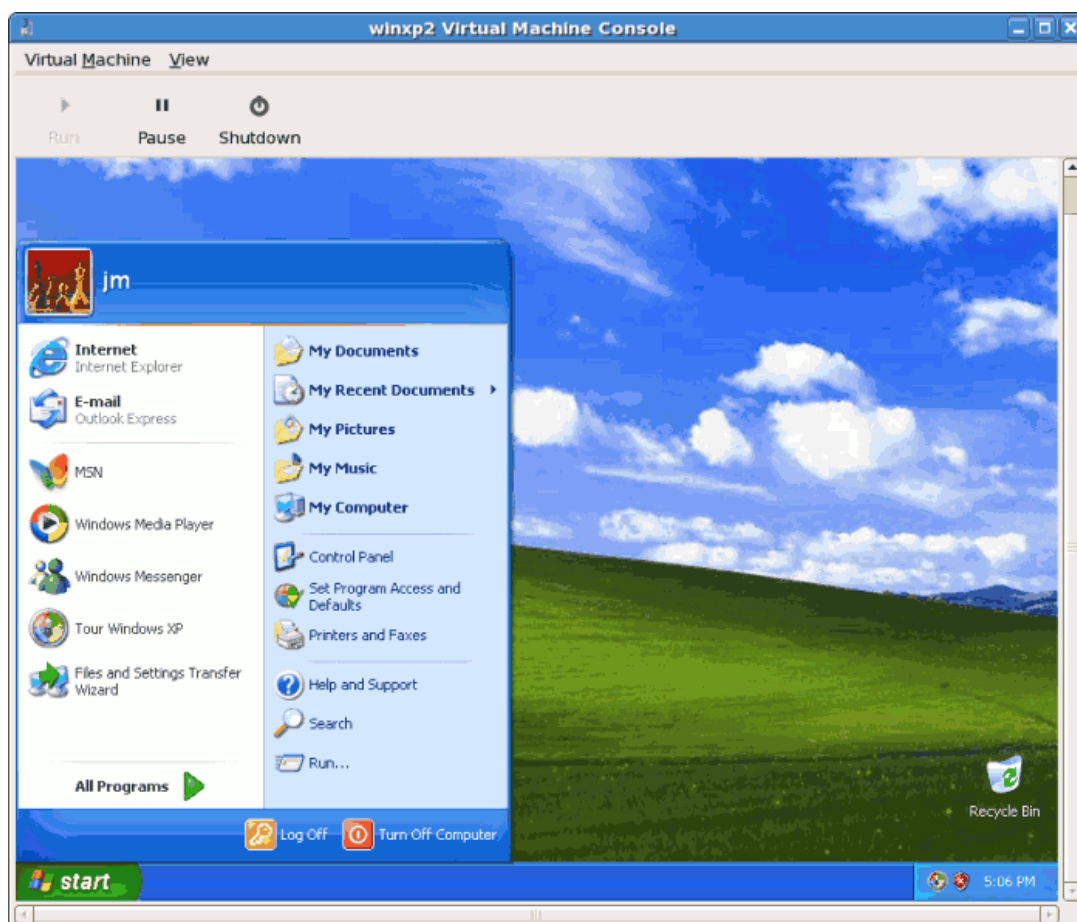
17. After setup has finished you will see the Windows boot screen:



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



19. After you completed the setup process you will be presented with your new Windows desktop or login screen:



### 3. Installing a Windows 2003 SP1 Server Guest as a fully-virtualized guest

**virt-manager** can install Windows XP as a fully-virtualized guest. However there are some extra steps which must be followed in order to complete the installation successfully. This section explains the installation process and extra steps needed to get Windows XP fully virtualized guests installed.

It may be easier to use `virt-install` for installing Windows Server 2003 as the console for the Windows guest will open quicker and allow for **F5** to be pressed which is required to select a new **HAL**.



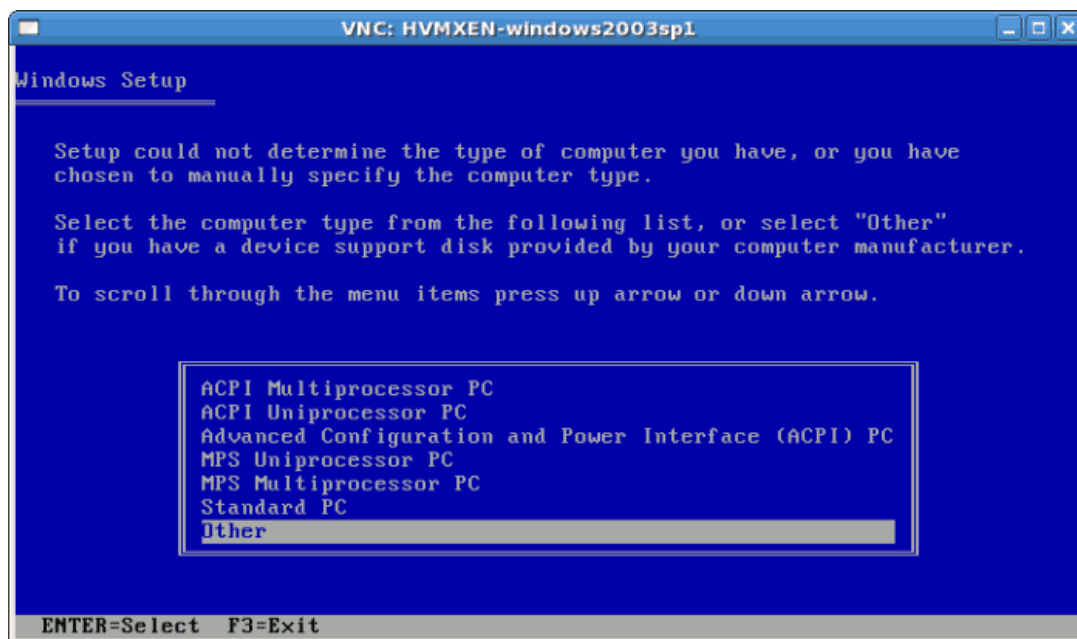
#### Itanium® support

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.

An example of using the `virt-install` for installing a Windows Server 2003 guest:

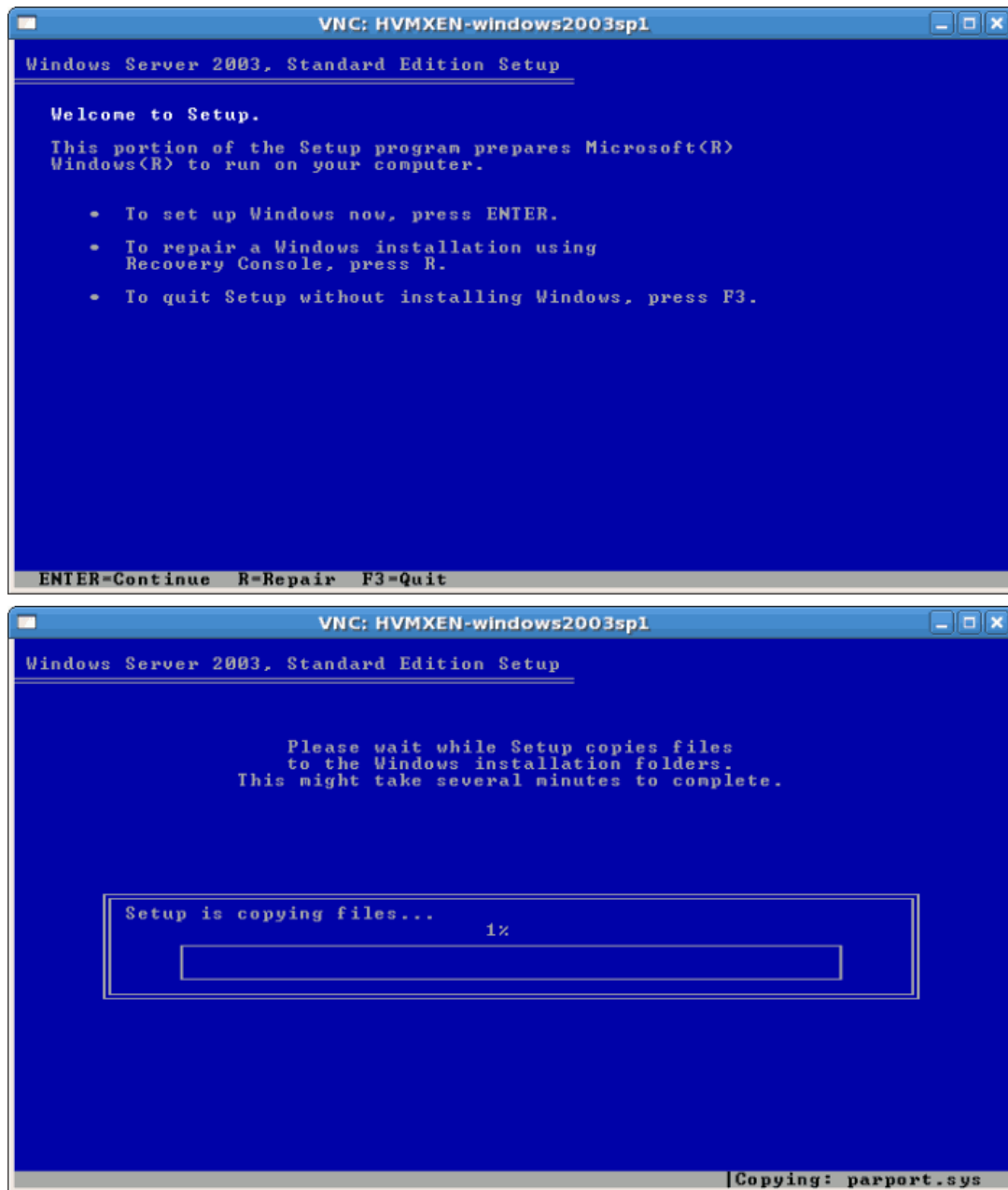
```
virt-install --hvm --s 5 --f /var/lib/xen/images/windows2003sp1.dsk --n
windows2003sp1\
--cdrom=/xen/trees/ISO/WIN/en_windows_server_2003_sp1.iso --vnc --r 1024
```

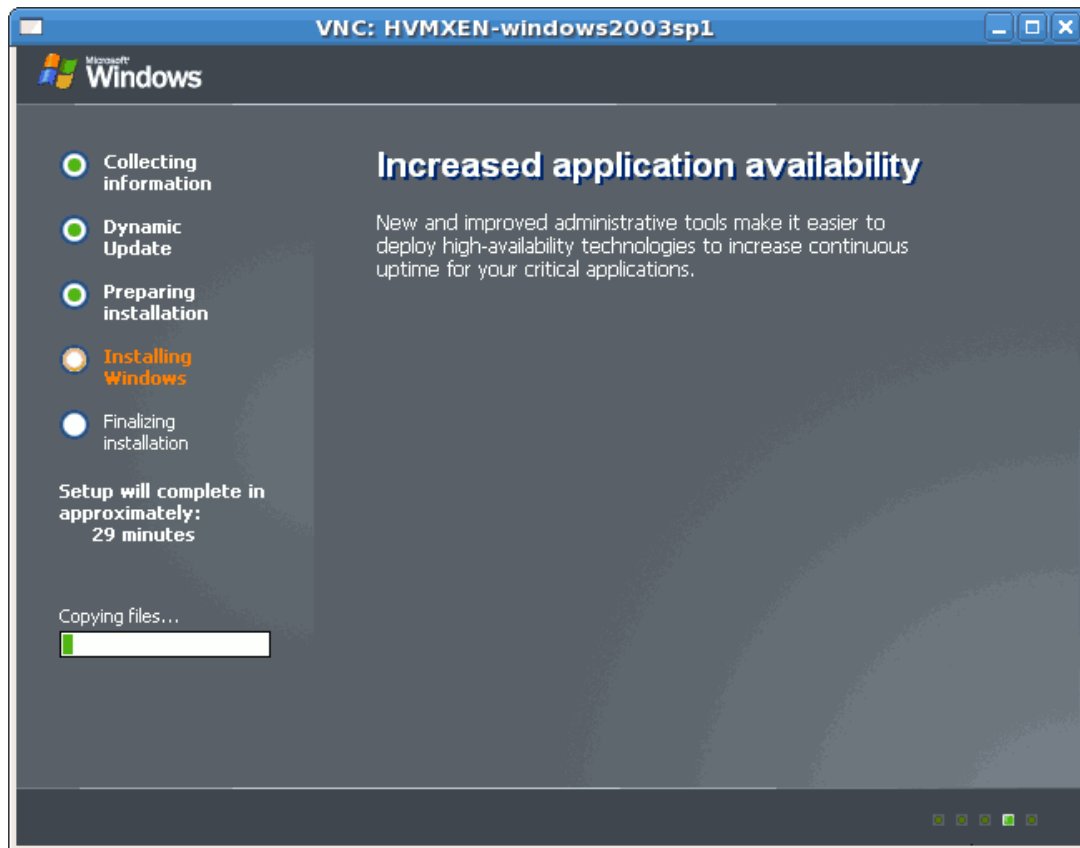
1. After starting your guest installation you need to quickly press **F5**, this opens a dialog window to select a different HAL or Computer Type. Choose `Standard PC` as the Computer Type:



2. Continue with a normal Microsoft Windows Server 2003 installation:







---

## Part III. Configuration

---



---

## Configuring Red Hat Enterprise Linux Virtualization

These chapters contain specialized information for certain advanced virtualization tasks. Users wanting enhanced security, additional devices or better performance are advised to read these chapters.



# Virtualized block devices

This chapter explains installing and configuring block devices in Red Hat Virtualization guests.

## 1. Installing a virtualized floppy disk controller

Presently, physical floppy disks can not be mapped to virtualized guests, however, you can access a virtualized floppy drive created using an image file of a floppy disk and a modified configuration file. The section will guide you through the process.

This section uses a guest system created with `virt-manager` running a fully virtualized Red Hat Enterprise Linux installation with an image located in `/var/lib/xen/images/rhel5FV-1.img`.



### Note

There is no guarantee that this section will work for your system at this time. Para-virtualized guests can access floppy drives as well as using para-virtualized drivers on a fully virtualized system. For more information on using para-virtualized drivers read [Chapter 13, Introduction to Para-virtualized Drivers](#).

Create the XML configuration file for your guest image using the following command on a running guest. This will save the configuration settings as an XML file which can be edited to customize the operations the guest performs when the guest is started. For another example of editing the `virsh` XML files, read [Chapter 27, Creating custom Red Hat Virtualization scripts](#).

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

Execute this instruction to create a floppy image to use on the guest.

```
dd if=/dev/zero of=/var/lib/xen/images/rhel5FV-1flop.img bs=512 count=2880
```

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/xen/images/rhel5FV-1flop.img' />
  <target dev='fda' />
</disk>
```

Stop the guest system, and 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.

## 2. Adding additional storage devices to a guest

After your guests are installed and configured you can add additional storage to your virtualization machines. You can use multiple different types of storage in your guests, these 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 a file based container as additional storage to a guest.

To add a file based container to a guest you must perform the following steps:

1. Create an empty container file or using an existing file container (such as an ISO file).
  - to create a sparse file use the following command (note that using sparse files is not recommended due to data integrity and performance issues, they may be used for testing but not in a production environment)

```
dd if=/dev/zero of=FileName.img bs=1M seek=4096 count=0
```

- or if you want to create a non-sparse file (recommended) just use the command

```
dd if=/dev/zero of=FileName.img bs=1M count=4096
```

The command above will create a 400MB file

2. Once you have created or identified the file you want to assign to your virtual machine you can add it to the virtual machine's configuration file.
3. Edit the virtual machine's configuration in `/etc/xen/VirtualMachineName` and look for an entry starting with `disk=`. A sample entry will look like the following:

```
disk = [ 'tap:aio:/var/lib/xen/images/rhel5vm01.dsk,xvda,w', ]
```



4. To add the additional storage, add a new file based container entry in the `disk=` section of the configuration file. Ensure you have specified a device name for the virtual block device (xvd) which has not yet been used by other storage devices. The following is an example configuration entry adding a file called `oracle.dsk`:

```
disk = [ 'tap:aio:/var/lib/xen/images/rhel5vm01.dsk,xvda,w', \
'tap:aio:/xen/images/oracle.dsk,xvdb,w', ]
```

5. Using the above entry your virtual machine will see file `oracle.dsk` as the device `/dev/xvdb` inside the guest.

### Adding a block device as additional storage to a guest.

To present a block device from your host to a guest you must perform the following steps:

1. Make the block device available to the host and configure for your guests needs (that is, the name, persistence, multipath and so on).
2. Edit the virtual machine's configuration in `/etc/xen/VirtualMachineName` and look for an entry starting with `disk=`. A sample entry will look like the following:

```
disk = [ 'tap:aio:/var/lib/xen/images/rhel5vm01.dsk,xvda,w', ]
```

3. To add the additional storage, add a new file based container entry in the `disk=` section of the configuration file. Ensure you specify the type `phy` and use a virtual block device name for the new virtual block device (xvd) which has not yet been used by other storage devices. The following is an example configuration entry which adds a file called `/dev/sdb1`:

```
disk = [ 'tap:aio:/var/lib/xen/images/rhel5vm01.dsk,xvda,w', \
'phy:/dev/sdb1,xvdb,w', ]
```

4. Using the above entry your virtual machine will see the file `oracle.dsk` as the device `/dev/xvdb` inside the guest

The same procedure can be used to allow a guest machine access to other physical block devices, for example a CD-ROM or DVD drive.

### Dynamically adding storage to a virtual machine.

In some cases one may want to add additional storage to a virtual machine without the need to reboot it. Red Hat Enterprise Linux 5/Virt provides this capability using the command line toolset

`xm`. In order to dynamically add storage to a virtual machine/domain you need to perform the following steps:

1. Identify the block device or image file you want to make available to the virtual machine (for our example we use `/dev/sdb1`)
2. After you have selected the storage you want to present to the guest you can use the `xm block-attach` command to assign it to your virtual machine. The syntax for `xm block-attach` is:

- `xm block-attach domain backdev frontdev mode`
- an example would look like the following:

```
xm block-attach MyVirtualMachine phy:/dev/sdb1 xvdb w
```

The above command will attach `/dev/sdb1` to the virtual machine `MyVirtualMachine` and the device would be seen as `/dev/xvdb` inside the virtual machine.

### 3. Configuring persistent storage in a Red Hat Enterprise Linux 5 environment

In an environment where external storage (for example, Fibre Channel or iSCSI) is used it is advised to configure persistent device names on your hosts. This will also aid in using Red Hat Virtualization's (live) migration feature to implement consistent device names across multiple systems.

#### Single Path Configuration.

On systems not using multipath, `udev` is a good way to implement LUN persistence. For more information on `scsi_id` and its various options please consult the man page.

1. The first step would be to acquire UUIDs. Check/Open your `/etc/scsi_id.config` file and verify that you have the following line commented out:

```
# options=-b
```

2. Add the following line to your `/etc/scsi_id.config` file:

```
options=-g
```

This configuration option will configure `udev` to assume that all attached SCSI devices will

return a UUID (Unique Device Identifier)

3. To display the UUID for a given device execute the following command: `scsi_id -g -s /block/sdc`. The output will look similar to the following:

```
# scsi_id -g -s /block/sdc
3600a0b800013275100000015427b625e
```

The result of the command above represents the device's UUID. At this time you should verify the UUID you just retrieved is the same displayed via each path the device can be accessed through. The UUID will be used as the primary/sole key to the device. UUIDs will be persistent across reboots and as you add additional storage to your environment.

4. The next step is to create a rule to name your device. In `/etc/udev/rules.d` create the file `20-names.rules`. You will be adding the new rules to the file `/etc/udev/rules.d/20-names.rules`, any subsequent rules will be added to the same file using the same format. Rules should have the following 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 the desired name for the device. In the example, the rule would look as follows:

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

This forces the system to verify all devices which correspond to a block device ( `/dev/sd*`) for the given UUID. When it finds a matching device, it will create a device node named `/dev/devicename`. In the example above, the device node is labeled `/dev/mydevice`.

5. As the last step add the following line to `/etc/rc.local`:

```
/sbin/start_udev
```

## Multi-path configuration.

To implement LUN persistence in a multipath environment, you need to define alias names for your multipath devices. To identify a device's UUID or WWID follow the steps from the single path configuration section. The multipath devices will be created in the `/dev/mpath` directory. In

our example below we will define 4 devices in `/etc/multipath.conf`:

```
multipaths {
    multipath {
        wwid      3600805f30015987000000000768a0019
        alias     oramp1
    }
    multipath {
        wwid      3600805f30015987000000000d643001a
        alias     oramp2
    }
    mulitpath {
        wwid      3600805f3001598700000000086fc001b
        alias     oramp3
    }
    mulitpath {
        wwid      3600805f30015987000000000984001c
        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' WWIDs/UUIDs to their new names will now be persistent even after rebooting.

## 4. Adding an ISO file as a CD-ROM to a guest configuration file

You may be prompted for Red Hat Enterprise Linux Installation CD-ROMs during the initial installation to add additional software components or packages. This method can make any ISO, or file based image, available as a device for your guest. This might be useful for installing software from an ISO file or other media source(for example, a downloaded image).

The syntax for adding an ISO image as a new device to a guest is as following:

```
[ 'file:/var/lib/xen/images/win2003sp1.dsk,hda,w',\
  'file:/xen/trees/ISO/WIN/en_windows_server_2003_with_sp1_standard.iso,hdc:cdrom,r',
]
```

In the example above you can see how the file `en_windows_server_2003_with_sp1_standard.iso` has been added to the guest configuration. Once you reboot your guest this ISO image will be configured as a CD Rom driver inside guest.

# Configuring networks and guests

Integrating Red Hat Virtualization into your network architecture is a complicated process and depending upon your infrastructure, may require custom configuration to deploy multiple ethernet interfaces and setup bridging.

Each domain network interface is connected to a virtual network interface in `dom0` by a point to point link. These devices are `vif<domid>` and `<vifid>.vif1.0` for the first interface in `dom1`; `vif3.1` for the second interface in domain 3.

`dom0` handles traffic on these virtual interfaces by using standard Linux conventions for bridging, routing, rate limiting, etc. The **xend** daemon employs two shell scripts to perform initial configuration of your network and new virtual interfaces. These scripts configure a single bridge for all virtual interfaces. You can configure additional routing and bridging by customizing these scripts.

Red Hat Virtualization's virtual networking is controlled by the two shell scripts, `network-bridge` and `vif-bridge`. **xend** calls these scripts when certain events occur. Arguments can be passed to the scripts to provide additional contextual information. These scripts are located in the `/etc/xen/scripts` directory. You can change script properties by modifying the `xend-config.sxp` configuration file located in the `/etc/xen` directory.

Use the `network-bridge` command when **xend** is started or stopped, this script initializes or shuts down the virtual network. Then the configuration initialization creates the bridge `xen-br0` and moves `eth0` onto that bridge, modifying the routing accordingly. When **xend** finally exits, it deletes the bridge and removes `eth0`, thereby restoring the original IP and routing configuration.

`vif-bridge` is a script that is invoked for every virtual interface on the domain. It configures firewall rules and can add the `vif` to the appropriate bridge.

There are other scripts that you can use to help in setting up Red Hat Virtualization to run on your network, such as `network-route`, `network-nat`, `vif-route`, and `vif-nat`. Or these scripts can be replaced with customized variants.



## 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).

- Enable SELinux to run 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/xen/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.





## Securing the host

When deploying Red Hat Virtualization on your corporate infrastructure, you must ensure that the host(*dom0*) cannot be compromised. *dom0* is the privileged domain that handles system management. If *dom0* is insecure, all other domains in the system are vulnerable. There are several ways to enhance security on systems using Red Hat Virtualization. You or your organisation 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 the lowest number of necessary services. You do not want to include too many jobs and services in *dom0*. The fewer processes and services running on *dom0*, the higher the level of security and performance.
- Enable SELinux on the hypervisor(*dom0*). Read [Chapter 11, 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.



# SELinux and virtualization



## Virtualization with SELinux enabled

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

### Adding additional devices and files to the hypervisor SELinux policy.

If you are using a **LVM** volume for your guest you will have to set the SELinux context for the underlying block device and volume group. In the example below the **lvm** volume is `/dev/VirtGroupVol1/rhel4u4Vol01` and the underlying block device is `/dev/sda3`:

```
# semanage fcontext -a -t xen_image_t -f -b /dev/sda3
# restorecon /dev/sda3
# semanage fcontext -a -t xen_image_t -f -b /dev/VirtGroupVol1/rhel4u4Vol01
# restorecon /dev/virtGroupVol1/rhel4u4Vol01
```

Set the SELinux context for a block device used by a guest using the `semanage` and `restorecon` commands. In the example below the block device is `/dev/sda2`:

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

The commands above can be used to add an additional directory which allows you to store guest images in a different directory than `/var/lib/xen/images/`. If you have a guest image outside of `/var/lib/xen/images/` Xen will be unable to access the image. Confirm the problem using `ls` on the file and which should output a `file not found` error.

You can modify your SELinux policy to include other directories you may use to storage images. You will need to add it to the SELinux policy and relabel the directory you want to use for your guest images. To add another directory (in our example the directory `/home/admin/xen/` will be added) to your SELinux policy use the following command:

```
semanage fcontext --add -t xen_image_t '/home/admin/xen(/.*)?'
```

The last step is to relabel the directory using the following command:

```
restorecon /home/admin/xen
```



# Virtualized network devices

This chapter covers special topics for networking and network configuration with Red Hat Enterprise Linux Virtualization.

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 5, Installing guests](#).

Network configuration is also covered in the tool specific reference chapters for `virsh` ([Chapter 20, Managing guests with `virsh`](#)) and `virt-manager` ([Chapter 21, Managing guests with 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 can improve performance on fully virtualized Linux guests. [Chapter 13, Introduction to Para-virtualized Drivers](#) explains how to utilize para-virtualized network drivers.

## 1. Configuring multiple guest network bridges to use multiple ethernet cards

Process to setup multiple Red Hat Virtualization bridges:

1. Configure another network interface using either the `system-config-network` GUI or creating a new configuration file in `/etc/sysconfig/network-scripts`. Below is an example configuration file for a second network interface called `eth1`

```
#/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 `/etc/xen/scripts/network-bridge` to `/etc/xen/scripts/network-bridge.xen`.

3. Edit `/etc/xen/xend-config.sxp` and add line to your new network bridge script (we will call the script `network-xen-multi-bridge`(example below) The new line should read `network-script network-xen-multi-bridge` and remove the commenting on the line called `network-script network-bridge`.
4. Create a custom script to create multiple Red Hat Virtualization network bridges. A sample scripts is below, this example script will create two Red Hat Virtualization 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
```

## 2. Laptop network configuration

The challenge in running Red Hat Virtualization 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 Red Hat Virtualization does not behave well as it 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 Red Hat Virtualization environment due to Red Hat Virtualization's (default) bridged network usage.

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

- You basically will be configuring a 'dummy' network interface which will be used by Red Hat Virtualization. In this example the interface is called `dummy0`. This will also allow you to use a hidden IP address space for your guests/Virtual Machines.
- 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 Red Hat Virtualization environment. Setup and configuration are explained further down in this section/chapter.
- You can also configure NAT/IP masquerading in order to enable access to the network from your guests/virtual machines.

### Configuring a dummy network interface.

Perform the following configuration steps on your host/Dom0:

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 Red Hat Virtualization and wireless card issues. The script below will enable NAT based on the interface currently used for your network connection.

### Configuring NAT(network address translation) for Red Hat Virtualization.

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
;;
*)
;;
esac
```



```

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 Red Hat 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 Red Hat Virtualization:

- Get the latest version of dnsmasq from [here](#)<sup>1</sup>.
- Document for dnsmasq can be found [here](#)<sup>2</sup>.
- 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/dnsmasq.conf`

<sup>1</sup> <http://et.redhat.com/~jmh/tools/xen/> Edit the configuration to suit your local needs and requirements. The following parameters are

<sup>2</sup> <http://www.thekelleys.org.uk/dnsmasq/doc.html>

likely the ones you want to modify:

- `interface` If you want `dnsmasq` to listen for DHCP and DNS requests only on specified (ie dummy interface(s) but not your public interfaces) interfaces (and the loopback) give the name of the interface (eg `dummy0`). Repeat the line for more than one interface. An example would be `interface=dummy0`
- `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 (ie if you start up a VPN session for example).
- Both the `nm-dnsmasq` and `xenDNSmasq` script will also setup NAT if you have your virtual machines in a hidden network to allow them access to the public network.

# Introduction to 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.

The RPM packages for the para-virtualized drivers include the modules for storage and networking para-virtualized drivers for the supported Red Hat Enterprise 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.

The para-virtualized drivers only work on 32 bit Red Hat Enterprise Linux hosts for 32 bit guests.

## 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.

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

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`.

Minimum host operating system version

- Red Hat Enterprise Linux 5.1 or higher

Minimum guest operating system version

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

You require at least 50MB of free disk space in the `/lib` filesystem

## 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
- Red Hat Enterprise Linux 4 Update 6
- Red Hat Enterprise Linux 3 Update 9

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 13.1. Supported kernel architectures for para-virtualized drivers**



### Note

The table above is for guest operating systems. AMD and Intel processors are supported for the Red Hat Enterprise Linux 5.1 host.



### Take note

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(Physical Address Extension), 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.

After installing the para-virtualized drivers on a guest operating system you should only use the `xm` command or `virsh` to start the guests. If `xm` is not used the network interfaces (for example, `eth1` and so on) will not be connected correctly during boot. This problem is known and the Bugzilla number is 300531 and a bug fix is in progress. The bug connects the network interface to `qemu-dm` and subsequently limits the performance dramatically.

**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
i686-HUGEMEM	i686

**Table 13.2. 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 13.3. Required host kernel architectures for guests using para-virtualized drivers on Red Hat Enterprise Linux 3 for AMD processors**



### Note

After installing the para-virtualized drivers on a guest operating system you should only use the `xm` command or the `virsh` command to start the guests. If `xm` or `virsh` are not used the network interfaces (for example, `eth1`) will not be correctly connected during boot. This problem is known and the [Bugzilla](#)<sup>1</sup> number is 300531. The bug connects the network interface to `qemu-dm` and subsequently limits the performance dramatically. A bug fix is presently in development and will be released via RHN.

## 3. Installation and Configuration of 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.

Specifically, use the first disk to install the MBR and the boot loader (GRUB), and to contain the `/boot` partition. (This disk 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 (e.g. `/`, `/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.

### 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 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. Shutdown your guest operating system.
5. Reconfigure the guest operating system 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 using the “`xm create YourGuestName`” command where *YourGuestName* is the name of the guest operating system.
9. Reconfigure the guest network

### 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-level application and data disks can use the para-virtualized block device driver.

#### 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 corresponding to 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.

```
[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/xen/images/rhel3_64_fv.dsk,hda,w"]
```

- Once you have added additional physical devices, LUNs, partitions or volumes your

para-virtualized driver the entry should resemble the entry shown below.

```
disk = [ "file:/var/lib/xen/images/rhel3_64_fv.dsk,hda,w",  
        "tap:aio:/var/lib/xen/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 `xm` command:

```
# xm create YourGuestName
```



### Note

You must use "`xm create <virt-machine-name>`" on Red Hat Enterprise Linux 5.1. The para-virtualized network driver(`xen-vnif`) will not be connected to `eth0` properly if you are using Red Hat Enterprise Linux 5.1 and the `virt-manager` or `virsh` interfaces. This issue is currently a known bug, BZ 300531.

Red Hat Enterprise Linux 5.2 does not have this bug and the `virt-manager` or `virsh` interfaces will correctly load the para-virtualized drivers.



### 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.



### Please note

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

## 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-level application and data disks can use the para-virtualized block device driver.

### 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 corresponding to 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 Linux4-6-z kernel and beyond.

```
[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. If 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/xen/images/rhel4_64_fv.dsk,hda,w" ]
```

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

```
disk = [ "file:/var/lib/xen/images/rhel3_64_fv.dsk,hda,w",
"tap:aio:/var/lib/xen/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 `xm` command:

```
# xm create YourGuestName
```



### Note

You must use `"xm create <virt-machine-name>"` on Red Hat Enterprise Linux 5.1. The para-virtualized network driver(`xen-vnif`) will not be connected to `eth0` properly if you are using Red Hat Enterprise Linux 5.1 and the `virt-manager` or `virsh` interfaces. This issue is currently a known bug, BZ 300531.

Red Hat Enterprise Linux 5.2 does not have this bug and the `virt-manager` or `virsh` interfaces will correctly load the para-virtualized drivers.

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.



### Also note...

IA64 binary RPM packages and builds are not presently available.



### A handy tip

If the `xen-vbd` driver does not automatically load. Issue the following command from the guest's terminal. Substitute `%release` with the correct release version for the para-virtualized drivers.

```
[root@rhel4]# insmod /lib/modules/'uname
-r' /weak-updates/xenpv/%release/xen-vbd.ko
```

### 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-level application and data disks can use the para-virtualized block device driver.

#### 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 corresponding to 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. If 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 GuestName>` 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/xen/images/rhel4_64_fv.dsk,hda,w" ]
```

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

```
disk = [ "file:/var/lib/xen/images/rhel3_64_fv.dsk,hda,w",  
"tap:aio:/var/lib/xen/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 `xm` command:

```
# xm create YourGuestName
```



### Note

You must use “`xm create <virt-machine-name>`” on Red Hat Enterprise Linux 5.1. The para-virtualized network driver(`xen-vnif`) will not be connected to `eth0` properly if you are using Red Hat Enterprise Linux 5.1 and the `virt-manager` or `virsh` interfaces. This issue is currently a known bug, BZ 300531.

Red Hat Enterprise Linux 5.2 does not have this bug and the `virt-manager` or `virsh` interfaces will correctly load the para-virtualized drivers.

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.

## 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.

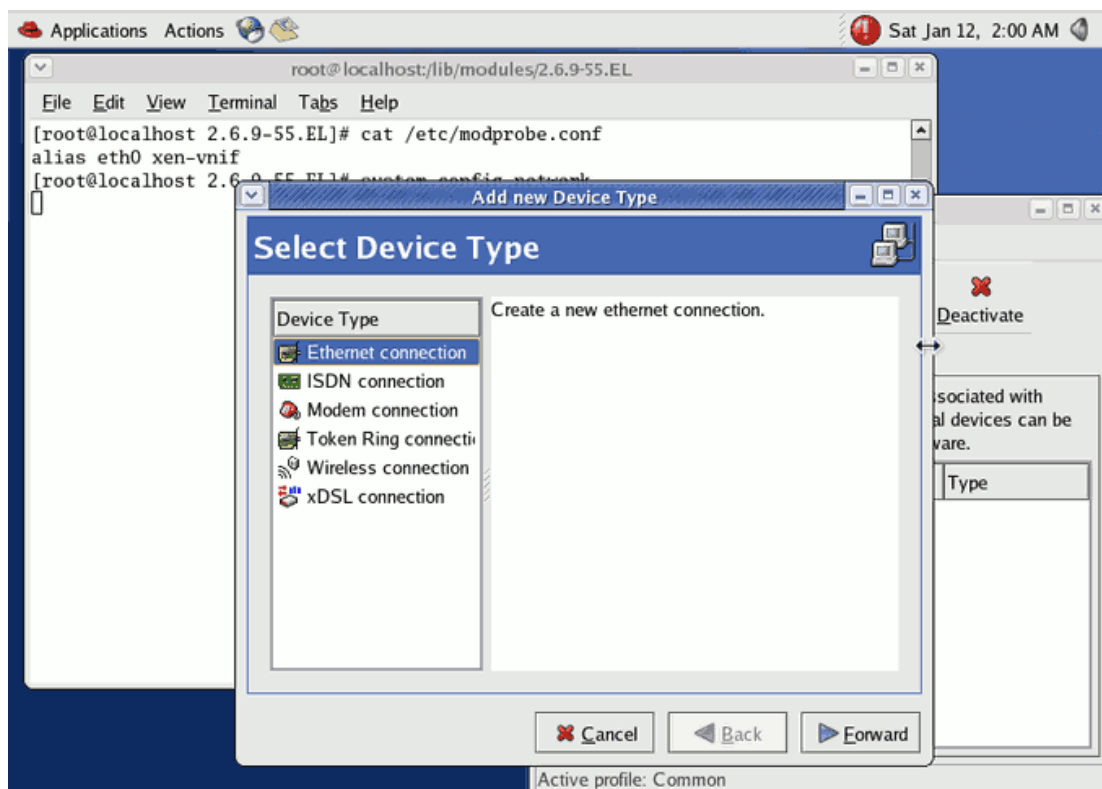
1. In `virt-manager` open the console window for the guest and log in as `root`.
2. On Red Hat Enterprise Linux 4 verify the file `/etc/modprobe.conf` contains the line `"alias eth0 xen-vnif"`.

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

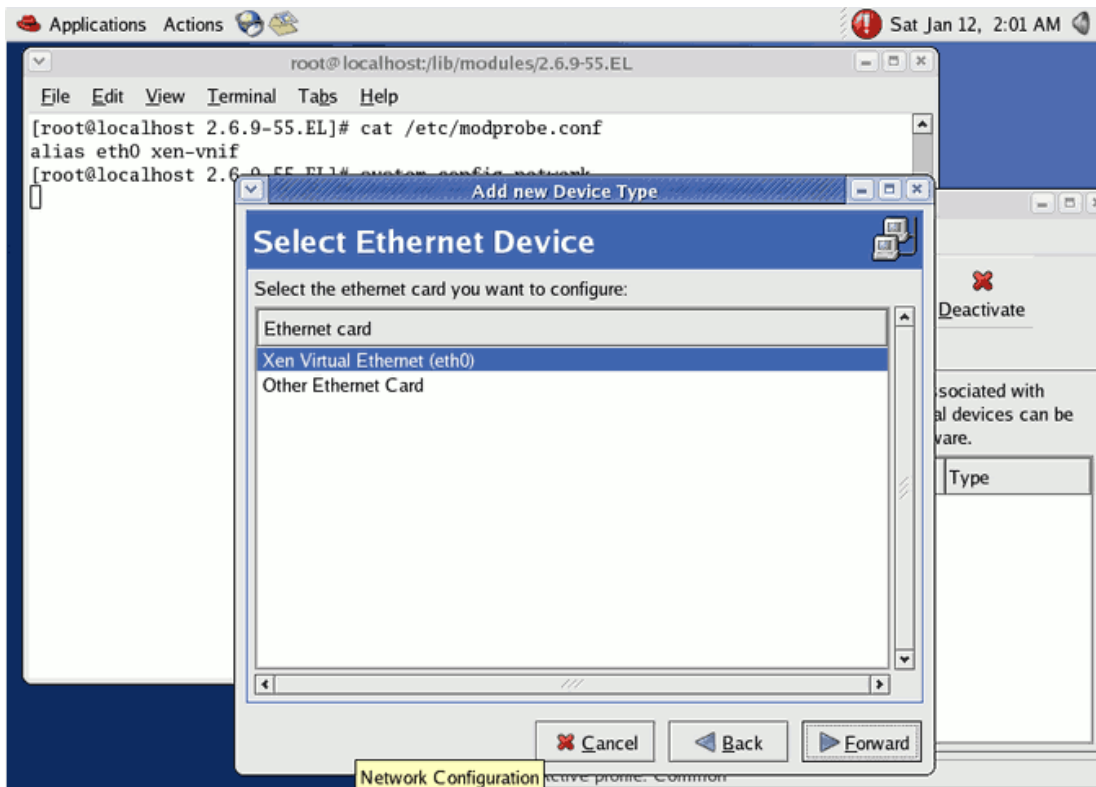
3. To display the present settings for `eth0` execute `"# ifconfig eth0"`. If you receive an error about the device not existing you should load the modules manually as outlined in [Section 5, "Manually loading the para-virtualized drivers"](#).

```
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)
```

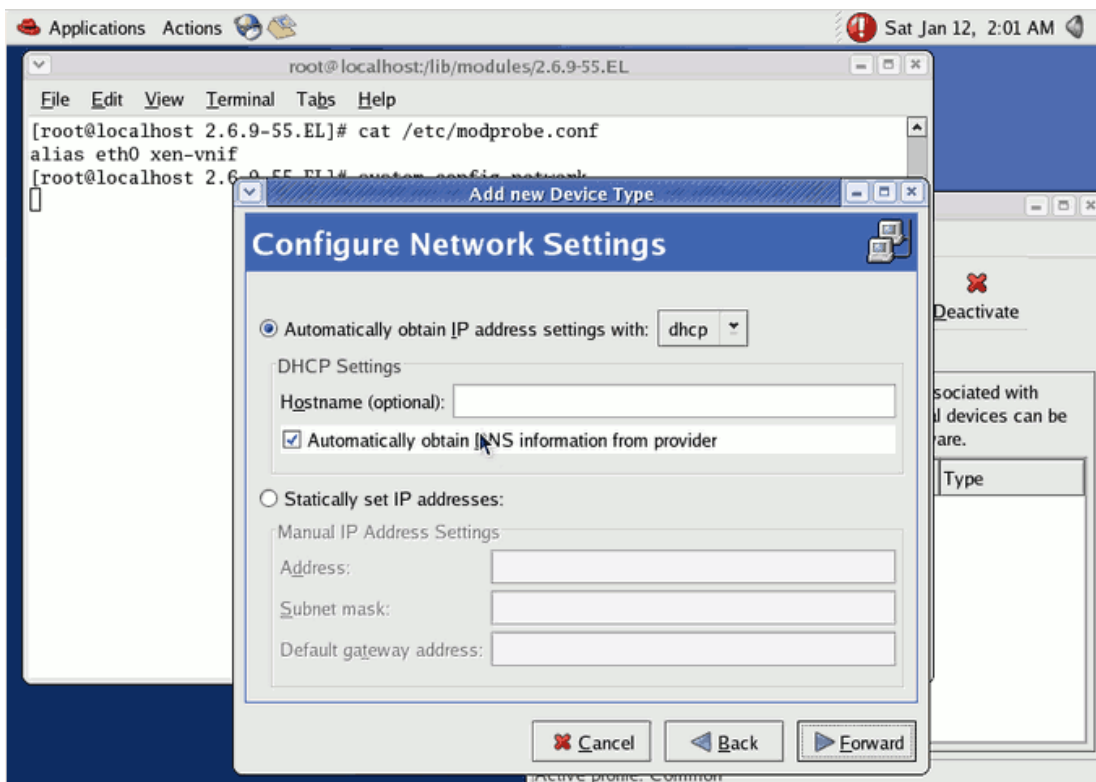
4. Start the network configuration utility(NetworkManager) with the command `"# system-config-network"`. Click on the **"Forward"** button to start the network card configuration.



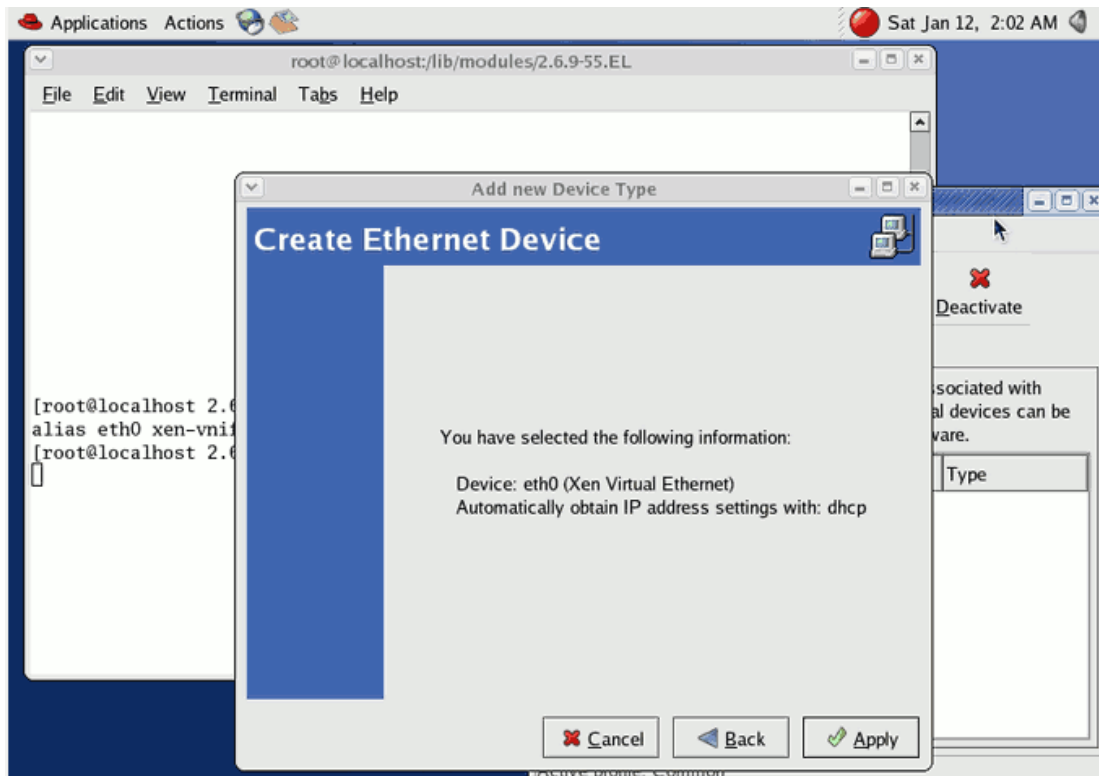
5. Select the '**Xen Virtual Ethernet Card (eth0)**' entry and click '**Forward**'.



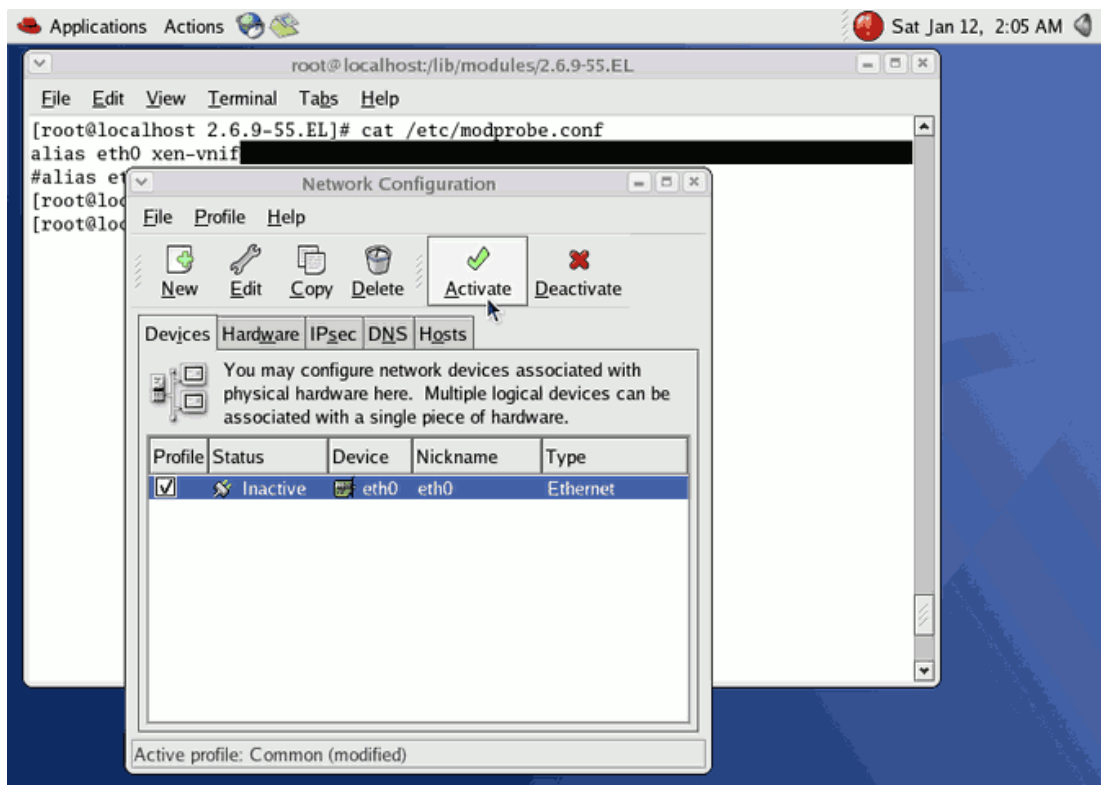
Configure the network settings as required.



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)
```

## 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](http://et.redhat.com/~jmh/docs/Installing_RHEL5_Virt.pdf)<sup>2</sup>

### 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.

---

<sup>2</sup> [http://et.redhat.com/~jmh/docs/Installing\\_RHEL5\\_Virt.pdf](http://et.redhat.com/~jmh/docs/Installing_RHEL5_Virt.pdf)



```
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.

## 5.2. Virtual Storage Devices

Perform the following steps to configure additional virtual storage 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.

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

Add an additional entry for your new physical device, LUN, partition or volume to the “`disk=`” section of the configuration file. The storage entity the para-virtualized driver the updated entry would like the following. Note the use of “`tap:aio`” for the para-virtualized device if a file based image is used.

```
disk = [ "file:/var/lib/xen/images/rhel5_64_fv.dsk,hda,w",  
        "tap:aio:/var/lib/xen/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/xen/images/rhel5_64_fv.dsk,hda,w",  
        "tap:aio:/var/lib/xen/images/UserStorage1.dsk,xvda,w" ]  
        "tap:aio:/var/lib/xen/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/pvdisk_p1
```

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

---

## **Part IV. Administration**

---



---

## **Administering Red Hat Enterprise Linux Virtualization**

These chapters contain information for administering host and guest systems using Red Hat Virtualization tools and technologies.



## Starting or stopping a domain during the boot phase

You can start or stop running domains at any time. Domain0 waits for all running domains to shutdown before restarting. You must place the configuration files of the domains you wish to shut down in the `/etc/xen/` directory. All the domains that you want to start at boot time must be symbolically linked to `/etc/xen/auto`.

```
chkconfig xendomains on
```

The `chkconfig xendomains on` command does not automatically start domains; instead it will start the domains on the next boot.

```
chkconfig xendomains off
```

The `chkconfig xendomains off` terminates all running domains and does *not* start them again during the next boot.





## 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
<b>(console-limit)</b>	Determines the console server's memory buffer limit <code>xend_unix_server</code> and assigns values on a per domain basis.
<b>(min-mem)</b>	Determines the minimum number of megabytes that is reserved for domain0 (if you enter 0, the value does not change).
<b>(dom0-cpus)</b>	Determines the number of CPUs in use by domain0 (at least 1 CPU is assigned by default).
<b>(enable-dump)</b>	Determines that a crash occurs then enables a dump (the default is 0).
<b>(external-migration-tool)</b>	Determines the script or application that handles external device migration. Scripts must reside in <code>etc/xen/scripts/external-device-migrate.</code>
<b>(logfile)</b>	Determines the location of the log file (default is <code>/var/log/xend.log</code> ).
<b>(loglevel)</b>	Filters out the log mode values: DEBUG, INFO, WARNING, ERROR, or CRITICAL (default is DEBUG).
<b>(network-script)</b>	Determines the script that enables the networking environment (scripts must reside in <code>etc/xen/scripts</code> directory).
<b>(xend-http-server)</b>	Enables the http stream packet management server (the default is no).
<b>(xend-unix-server)</b>	Enables the unix domain socket server, which is a socket server is a communications

Item	Description
	endpoint that handles low level network connections and accepts or rejects incoming connections. The default value is set to yes.
( <b>xend-relocation-server</b> )	Enables the relocation server for cross-machine migrations (the default is no).
( <b>xend-unix-path</b> )	Determines the location where the <code>xend-unix-server</code> command outputs data (default is <code>var/lib/xend/xend-socket</code> )
( <b>xend-port</b> )	Determines the port that the http management server uses (the default is 8000).
( <b>xend-relocation-port</b> )	Determines the port that the relocation server uses (the default is 8002).
( <b>xend-relocation-address</b> )	Determines the virtual machine addresses that are allowed for system migration. The default value is the value of <code>xend-address</code>
( <b>xend-address</b> )	Determines the address that the domain socket server binds to. The default value allows all connections.

**Table 15.1. `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:

The `xend` node control daemon performs 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 `xend-config.sxp` which is located in the `/etc/xen` directory.

```
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 initscripts.

```
chkconfig --level 345 xend
```

The `xend` will now start at runlevels 3, 4 and 5.



## Managing CPUs

Red Hat Virtualization allows a domain's virtual CPUs to associate with one or more host CPUs. This can be used to allocate real resources among one or more guests. This approach allows Red Hat Virtualization to make optimal use of processor resources when employing dual-core, hyperthreading, or other advanced CPU technologies. If you are running I/O intensive tasks, it is typically better to dedicate either a hyperthread or entire core to run domain0. The Red Hat Virtualization credit scheduler automatically re-balances virtual cpus between physical ones, to maximize system use. The Red Hat Virtualization system allows the credit scheduler to move CPUs around as necessary, as long as the virtual CPU is pinned to a physical CPU.

To view vcpus using `virsh` refer to [Displaying virtual CPU information](#) for more information.

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 13, “Displaying virtual CPUs”](#) for more information.



# Virtualization live migration

Red Hat Virtualization includes the capabilities to support *migration* of para-virtualized guests between Red Hat Virtualization servers. Migration can either be performed in two ways:

- **Offline mode** using the command `xm migrate VirtualMachineName HostName`. In this mode the virtual machine will be stopped on the original host and restarted on the new host.
- **Live mode** using the `--live` option for the command `xm migrate --live VirtualMachineNameHostName`.



## Word usage note

Take note of the interchangeable use of *relocation* and *migration* throughout these section. The different terms are used to match the different naming conventions of certain configuration files. Both terms can be taken to mean the same thing, that is the relocation of one guest image from one server to another.



## Itanium® support note

Virtual machine migration is presently unsupported on the Itanium® architecture.

To enable the use of migration a few changes must be made to configuration file `/etc/xen/xend-config.sxp`. By default migration is disabled due to the potentially harmful affects on the host's security. Opening the relocation port carries the potential ability of unauthorized hosts and users to initiate migrate or connect to the relocation ports. As there is no specific authentication for relocation requests and the only control mechanism is based on hostnames and IP addresses special care should be taken to make sure the migration port and server is not accessible to unauthorized hosts.



## A note on virtualization migration security

IP address and hostname filters offer only 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 the host and client are on from external and unauthorized internal connections.

## Enabling migration.

Modify the following entries in `/etc/xen/xend-config.sxp` to enable migration, remove the comments preceding the parameters in the configuration file:

```
(xend-relocation-server yes)
```

The default value is `no` to keep the migration server deactivated. Unless you are using a trusted network, the domain virtual memory will be exchanged in raw form without encryption of the communication.

You modify the `xend-relocation-hosts-allow` option to restrict access to the migration server.

```
(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.

```
(xend-relocation-address '')
```

`(xend-relocation-address)` is the address the `xend` should listen on for relocation-socket connections, if `xend-relocation-server` is set.

The default is listen on all active interfaces, the parameter can be used to 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 valid list of addresses or regular expressions surrounded by single quotes.

```
(xend-relocation-hosts-allow '')
```

The `(xend-relocation-hosts-allow )` parameter is used to control the hosts who are allowed to talk to the relocation port.

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 you should reboot the host to restart your environment with the new parameters.



## 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 Red Hat Virtualization 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 /xentest
# mount /dev/sdb /xentest
```



### Important

Ensure the directory is exported with the correct options. If you are exporting the default directory `/var/lib/xen/images/` make sure you *only* export `/var/lib/xen/images/` and *not* `/var/lib/xen/` as this directory is used by the `xend` daemon and other Xen components. Sharing `/var/lib/xen/` will cause unpredictable behavior.

```
# cat /etc/exports
/xentest *(rw,async,no_root_squash)
```

Verify it is exported via **NFS**:

```
# showmount -e et-virt07
Export list for et-virt07:
/xentest *
```

### Install the virtual machine.

The install command in the example used for installing the virtual machine:

```
# virt-install -p -f /xentest/xentestTravelVM01.dsk -s 5 -n\
```

```
xentesttravelvm01 --vnc -r 1024 -l
http://porkchop.devel.redhat.com/rel-eng/RHEL5-\
Server-20070105.0/4.92/x86_64/os/ -b xenbr1
```

### Verify environment for migration.

Make sure the Xen networking 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.feffffffffffff      no           peth1
vif0.1
```

```
[et-virt07 ~]# brctl show
bridge name      bridge id                STP enabled  interfaces
xenbr1           8000.feffffffffffff      no           peth1
vif0.1
```

Check the relocation parameters have been configured in the Xen config file 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 Xen 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 /xentest
Filesystem            1K-blocks      Used Available Use% Mounted on
et-virt07:/xentest    70562400    2379712   64598336   4% /xentest
```

```
[et-virt08 ~]# file /xentest/xentestTravelVM01.dsk
/xentest/xentestTravelVM01.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 /xentest/foo
[et-virt08 ~]# rm -f /xentest/foo
```

### Verification of save and restore on local host.

Start up the virtual machine (if it has not yet):

```
[et-virt07 ~]# xm li
Name           ID Mem(MiB) VCPUs State   Time(s)
Domain-0       0    1880    8    r----- 50.7
```

```
[et-virt07 ~]# xm create xentesttravelvm01
Using config file "/etc/xen/xentesttravelvm01".
Going to boot Red Hat Enterprise Linux Server (2.6.18-1.2961.el5xen)
kernel: /vmlinuz-2.6.18-1.2961.el5xen
initrd: /initrd-2.6.18-1.2961.el5xen.img
Started domain xentesttravelvm01
```

Verify the virtual machine is running:

```
[et-virt07 ~]# xm li
Name           ID Mem(MiB) VCPUs State   Time(s)
Domain-0       0    983     8    r----- 58.2
xentesttravelvm01 1   1024     1    -b----  9.2
```

Save the virtual machine on the local host:

```
[et-virt07 xentest]# time xm save xentesttravelvm01 xentesttravelvm01.sav
real    0m15.744s
user    0m0.188s
sys     0m0.044s
```

```
[et-virt07 xentest]# ls -lrt xentesttravelvm01.sav
-rwxr-xr-x 1 root root 1075657716 Jan 12 06:46 xentesttravelvm01.sav
```

```
[et-virt07 xentest]# xm li
Name          ID Mem(MiB) VCPUs State   Time(s)
Domain-0      0  975      8    r----- 110.7
```

Restore the virtual machine on the local host:

```
[et-virt07 xentest]# time xm restore xentesttravelvm01.sav
real    0m12.443s
user    0m0.072s
sys     0m0.044s
```

```
[et-virt07 xentest]# xm li
Name          ID Mem(MiB) VCPUs State   Time(s)
Domain-0      0  975      8    r----- 118.5
xentesttravelvm01  3 1023      1    -b----  0.0
```

### Live migration test.

Run a simple loop inside the guest to print out time and hostname every 3 seconds.



#### Note

The local host's clock is set to a different time 4hrs ahead of the remote host's clock.

```
# while true
```

```
> do
> hostname ; date
> sleep 3
> done
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:50:16 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:50:19 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:50:22 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:50:25 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:22:24 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:22:27 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:22:30 EST 2007
```

Verify the virtual machine is running:

```
[et-virt08 xen]# xm li
Name                               ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0  975      4    r----- 45.9
xentesttravelvm01                   1 1023      1    -b----  1.3
```

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 xentest]# xm li
Name                               ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0  975      8    r----- 161.1
```

Verify the virtual machine has been migrated to `et-virt08`:

```
[et-virt08 ~]# xm li
Name                               ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0  975      4    r----- 46.3
xentesttravelvm01                   1 1023      1    -b----  1.6
```

### 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 virtual machine'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 ~]# xm li
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                            0  975      4    r----- 46.3
xentesttravelvm01                   1 1023      1    -b----- 1.6
```

Initiate a live migration to et-virt07. You can add the `time` command to see how long the migration takes:

```
[et-virt08 ~]# time xm migrate --live xentesttravelvm01 et-virt07
real    0m10.378s
user    0m0.068s
sys     0m0.052s
```

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 ~]# xm li
Name                      ID Mem(MiB) VCPUs State   Time(s)
Domain-0                  0  975      4    r----- 56.3

```

Verify the virtual machine has started up on et-virt07:

```

[et-virt07 xentest]# xm li
Name                      ID Mem(MiB) VCPUs State   Time(s)
Domain-0                  0  975      8    r----- 198.1
xentesttravelvm01        4 1023      1    -b----- 1.0

```

Run through another cycle migrating from et-virt07 to et-virt08. Initiate a migration from

et-virt07 to et-virt08:

```
[et-virt07 xentest]# time xm migrate --live xentesttravelvm01 et-virt08
real    0m11.490s
user    0m0.084s
sys     0m0.028s
```

Verify the virtual machine has been shut down:

```
[et-virt07 xentest]# xm li
Name                               ID Mem(MiB) VCPUs State   Time(s)
Domain-0                           0  975      8    r----- 221.7
```

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-LogVol100
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-LogVol100
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 ~]# xm li
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                            0  975      4    r----- 67.3
xentesttravelvm01                   2 1023      1    -b----  0.4

```

and run another cycle:

```

[et-virt08 ~]# time xm migrate --live xentesttravelvm01 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.



# Remote management of virtualized guests

This section will explain how to remotely manage your Red Hat Enterprise Linux Virtualization guests using `ssh` or TLS and SSL.

## 1. Remote management with `ssh`

The `ssh` tool can be used to manage remote virtual machines. 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 will be 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 login 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 where you will run `virt-manager`. 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 Red Hat Virtualization may require a special kernel or 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.

## 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` (see [Section 1, “Remote management with `ssh`”](#)). TLS and SSL is the same technology used by web browsers for secure connections. The `libvirt` management connection will open 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.

Using this method you will not need to give users 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 be used to revoke access to users.

### 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

The `libvirt` website has a walk-through on creating certificates, and placing them in the `corr`Refer to <http://libvirt.org/remote.html>

### The Red Hat Virtualization VNC Server

The Red Hat Virtualization 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 `qemu://hostname.domainname/system` or `xen://hostname.domainname/` URIs should be used.

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



---

## **Part V. Virtualization Reference Guide**

---





---

## **Tools Reference Guide for Red Hat Enterprise Linux Virtualization**

These chapters provide in depth description of the tools used by Red Hat Enterprise Linux Virtualization. Users wanting to find information on advanced functionality should read these chapters.



# Red Hat Virtualization tools

The following is a list of tools for Linux and Xen administration, debugging and networking tools that are useful for systems running Xen.

## System Administration Tools

- xentop
- xm dmesg
- xm log
- vmstat
- iostat
- lsof

```
# lsof -i :5900
xen-vncfb 10635  root  5u  IPv4 218738  TCP grumble.boston.redhat.com:5900
(LISTEN)
```

## Advanced Debugging Tools

- XenOprofile
- systemTap
- crash
- xen-gdbserver
- sysrq
- sysrq t
- sysrq w
- sysrq c

## Networking

brctl

•

```
# brctl show
bridge name  bridge id            STP enabled  interfaces
xenbr0       8000.feffffffffffff  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.fefffffffffff
designated root     8000.fefffffffffff
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.fefffffffffff path cost          100
designated bridge   8000.fefffffffffff 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.fefffffffffff path cost          100
designated bridge   8000.fefffffffffff 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.fefffffffffff path cost          100
designated bridge   8000.fefffffffffff message age timer   0.00
designated port     8001          forward delay timer 0.00
designated cost     0            hold timer         0.43
flags
```

- ifconfig

- 
- tcpdump



# 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` tool and the graphical guest Manager(`virt-manager`). Unprivileged users can employ this utility for read-only operations. If you plan on running the `xend`, you should enable `xend` to run as a service. After modifying the respective configuration file, reboot the system, and `xend` will run as a service. You can use `virsh` to load scripts for the guest machines.

## Connecting to the hypervisor.

To initiate a hypervisor session with `virsh`:

```
virsh connect <name>
```

Where `<name>` is the machine name of the hypervisor. If you want to initiate a read-only connection, append the above command with `-readonly`.

## Creating a guest.

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
```

## Creating a virtual machine XML dump(configuration file).

To perform a data dump for an existing guest with `virsh`:

```
virsh dumpxml [domain-id, domain-name or domain-uuid]
```

This command outputs the domain information (in XML) to `stdout`. You save the data by piping the output to a file.

```
virsh dumpxml GuestID > guest.xml
```

The file `guest.xml` can then be used to recreate the guest (refer to [Creating a guest](#)). You can edit this XML configuration file to configure additional devices or to deploy additional guests. Refer to [Section 1, "Using XML configuration files with virsh"](#) for more information on modifying files created with `virsh dumpxml`.

### Suspending a guest.

To suspend a guest with `virsh`:

```
virsh suspend [domain-id, domain-name or domain-uuid]
```

When a domain is in a suspended state, it still consumes system RAM. Disk and network I/O will not occur while the guest is suspended. This operation is immediate and the guest must be restarted with the `resume option`.

### Resuming a guest.

To 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.

### Saving a guest.

To 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 option`.

### Restoring a guest.

To restore a guest that you previously saved with the `virsh save option` using the `virsh` command:

```
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.

### Shutting Down a guest.

To 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 of the `xmdomain.cfg` file.

### Rebooting a guest.

To 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 of the `xmdomain.cfg` file.

### Terminating a guest.

To terminate, destroy or delete guest use the `virsh` command with `destroy`:

```
virsh destroy [domain-id, domain-name or domain-uuid]
```

This command does an immediate ungraceful shutdown and stops any guest domain sessions (which could potentially lead to file corrupted file systems still in use by the guest). You should use the `destroy` option only when the guest is non-responsive. For a para-virtualized guest, you should use the [shutdown option](#).

### Converting a Domain Name to a Domain ID.

To convert a domain name or UUID to a domain id with `virsh`:

```
virsh domid [domain-name or domain-uuid]
```

### Converting a Domain ID to a Domain Name .

To convert a domain id or UUID to a domain name with `virsh`:

```
virsh domname [domain-id or domain-uuid]
```

### Converting a Domain Name to a UUID.

To convert a domain name to a UUID with `virsh` :

```
virsh domuuid [domain-id or domain-name]
```

### 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]
```

### Displaying node information .

To display node information `virsh`:

```
virsh nodeinfo
```

The outputs displays something similar to:

```
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 [ --inactive | --all]
```

The `--inactive` option lists inactive domains (domains that have been defined but are not currently active). The `--all` domain lists all domains, whether active or not. Your output should resemble the this example:

```
Id Name                               State
-----
```

---

0	Domain-0	running
1	Domain202	paused
2	Domain010	inactive
3	Domain9600	crashed

Using `virsh list` will output domains with the following states:

- The `running` state refers to domains which are currently active on a CPU.
- Domains listed as `blocked` or `blocking` are presently idle, waiting for I/O, waiting for the hypervisor or dom0 or .
- The `paused` state lists domains which are suspended.
- The `shutdown` state is for domains in the process of shutting down.
- Domains in the `shutoff` state are off and not using system resources.
- `crashed` domains have failed while running and are no longer running.

### Displaying virtual CPU information.

To display virtual CPU information from a guest with `virsh`:

```
virsh vcpuinfo [domain-id, domain-name or domain-uuid]
```

### Configuring virtual CPU affinity.

To configure the affinity of virtual CPUs with physical CPUs:

```
virsh vcpupin [domain-id, domain-name or domain-uuid] [vcpu] , [cpulist]
```

Where `[vcpu]` is the virtual VCPU number and `[cpulist]` lists the physical number of CPUs.

### Configuring virtual CPU count.

To modify a domain's number of CPUs with `virsh`:

```
virsh setvcpus [domain-name, domain-id or domain-uuid] [count]
```

You cannot increase the count above the amount you specified when you created the guest.

### 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.

### 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:

```
[root@domain ~]# 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 [vnet name]
```

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>
```

---

```
</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 [XML file]` — Generates and starts a new network using a preexisting XML file
- `virsh net-define [XML file]` — Generates a new network from a preexisting XML file without starting it
- `virsh net-destroy [network name]` — Destroy a network specified as [network name]
- `virsh net-name [network UUID]` — Convert a specified [network UUID] to a network name
- `virsh net-uuid [network name]` — Convert a specified [network name] to a network UUID
- `virsh net-start [name of an inactive network]` — Starts a previously undefined inactive network
- `virsh net-undefine [name of an inactive network]` — Undefine an inactive network



# Managing guests with Virtual Machine Manager(virt-manager)

This section describes the Red Hat Virtualization Virtual Machine Manager (VMM) windows, dialog boxes, and various GUI controls.

## 1. Virtual Machine Manager Architecture

Red Hat Virtualization is a collection of software components that work together to host and manage virtual machines. The Virtual Machine Manager (VMM) gives you a graphical view of the virtual machines on your system. You can use VMM to define both para-virtual and full virtual machines. Using Virtual Machine Manager, you can perform any number of virtualization management tasks including assigning memory, assigning virtual CPUs, monitoring operational performance, and save, restore, pause, resume, and shutdown virtual systems. It also allows you to access the textual and graphical console. Red Hat Virtualization abstracts CPU and memory resources from the underlying hardware and network configurations. This enables processing resources to be pooled and dynamically assigned to applications and service requests. Chip-level virtualization enables operating systems with Intel VT and AMD-V hardware to run on hypervisors.

## 2. 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. You start the Virtual Machine Manager test mode by selecting the **Other hypervisor** and then type `test:///default` in the URL field beneath. Once in test mode, you can connect to a libvirt dummy hypervisor. Note that although the **Remote Xen host screen** is visible, the functionality to connect to such a host is not implemented into Red Hat Enterprise Linux 5.

**Figure 21.1. Virtual Machine Manager connection window**

## 3. 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 21.2. Virtual Machine Manager main window**

## 4. The Virtual Machine Manager details window

This window displays graphs and statistics of a guest's live resource utilization data available from the Red Hat Virtualization Virtual Machine Manager. The UUID field displays the globally unique identifier for the virtual machines.

**Figure 21.3. Virtual Machine Manager details window**

## 5. 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 21.4. 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 18, Remote management of virtualized guests](#). TLS can be used to 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`'s '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`'.

## 6. Starting `virt-manager`

To start `virt-manager` session, from the Applications menu, click `System Tools` and select **Virtual Machine Manager**(`virt-manager`).



The `virt-manager` main window appears.

### Figure 21.5. 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 1, “Remote management with `ssh`”](#).

## 7. Creating a new guest

`virt-manager` is the desktop application which can be used to manage guests.

You can use Red Hat's Virtual Machine Manager to:

- Create new domains.
- Configure or adjust a domain's resource allocation and virtual hardware.
- Summarize running domains with live performance and resource utilization statistics.
- Display graphs that show performance and resource utilization over time.
- Use the embedded VNC client viewer which presents a full graphical console to the guest domain.

Before creating new guest virtual machines you should consider the following options. This list is a summary of the installation process using the Virtual Machine Manager.

- The name for your guest virtual machine.
- Decide whether you will use full virtualization (required for non-Red Hat Enterprise Linux guests. Full virtualization provides more flexibility but less performance) or para-virtualization (only for Red Hat Enterprise Linux 4 and 5 guests. Provides performance close to [bare-metal](#)).
- Identify installation media and kickstart (if appropriate) locations.
  - Para-virtualized guests required network based installation media. That is your installation media must be hosted on a `nfs`, `ftp` or `http` server.
  - Fully virtualized guests require iso images, CD-ROMs or DVDs of the installation media

available to the host system.

- If you are creating a fully virtualized guest, identify the operating system type and variant.
- Decide the location and type (for example, a file or partition) of the storage for the virtual disk
- Select the network connection
- Decide how much of your physical memory and cpu cores, or processors, you are going to allocate the guest. Be aware of the physical limitations of your system and the system requirements of your virtual machines.
- Review your selected options and start the installation.

VNC is used for graphical installations.



### Note:

You must install Red Hat Enterprise Linux 5, `virt-manager`, and the kernel packages on all systems that require virtualization. All systems then must be booted and running the Red Hat Virtualization kernel.



### If `virt-manager` is not working properly...

If `virt-manager` is not working, it is usually due to one of these common problems:

1. you have not booted the correct kernel. Verify you are running the `kernel-xen` kernel by running `uname`.

```
$ uname -r
2.6.18-53.1.14.el5xen
```

If the `kernel-xen` is installed it must be enabled in grub, see [Chapter 23, Configuring GRUB](#). If the Red Hat Virtualization packages are not installed, see [Chapter 4, Installing Red Hat Virtualization packages on the host](#).

2. the virtualization extensions are not enabled or available on your hardware. Verify your hardware has the virtualization extensions for full virtualization, read [Chapter 1, System requirements](#).
3. `virt-manager` is not installed. To install Red Hat Virtualization, read [Chapter 4, Installing Red Hat Virtualization packages on the host](#).

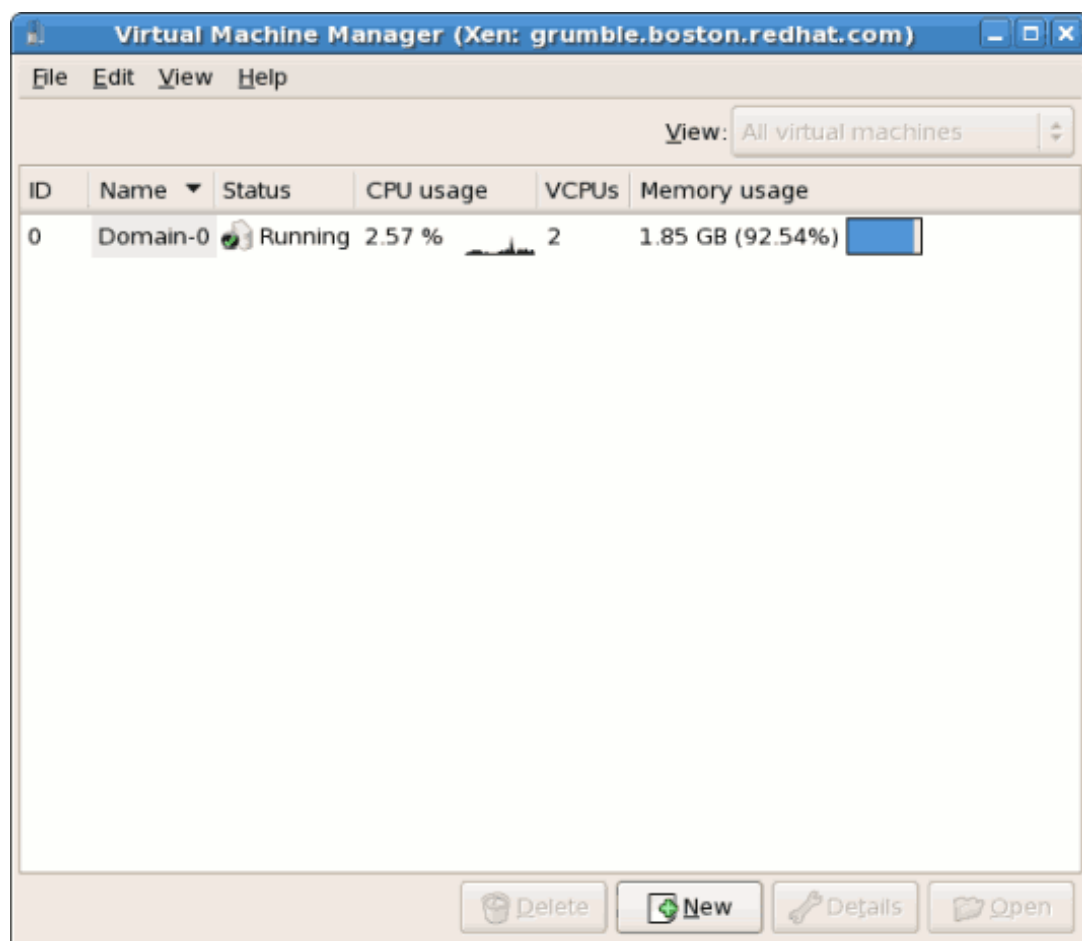
For other issues see the troubleshooting section, [Part VII, "Troubleshooting"](#).

These are the steps required to install a guest operating system on Red Hat Enterprise Linux 5 using the Virtual Machine Monitor:

### Procedure 21.1. Creating a guest with `virt-manager`

1. From the **Applications** menu, select **System Tools** and then **Virtual Machine Manager**.

The Virtual Machine Manager main window appears.



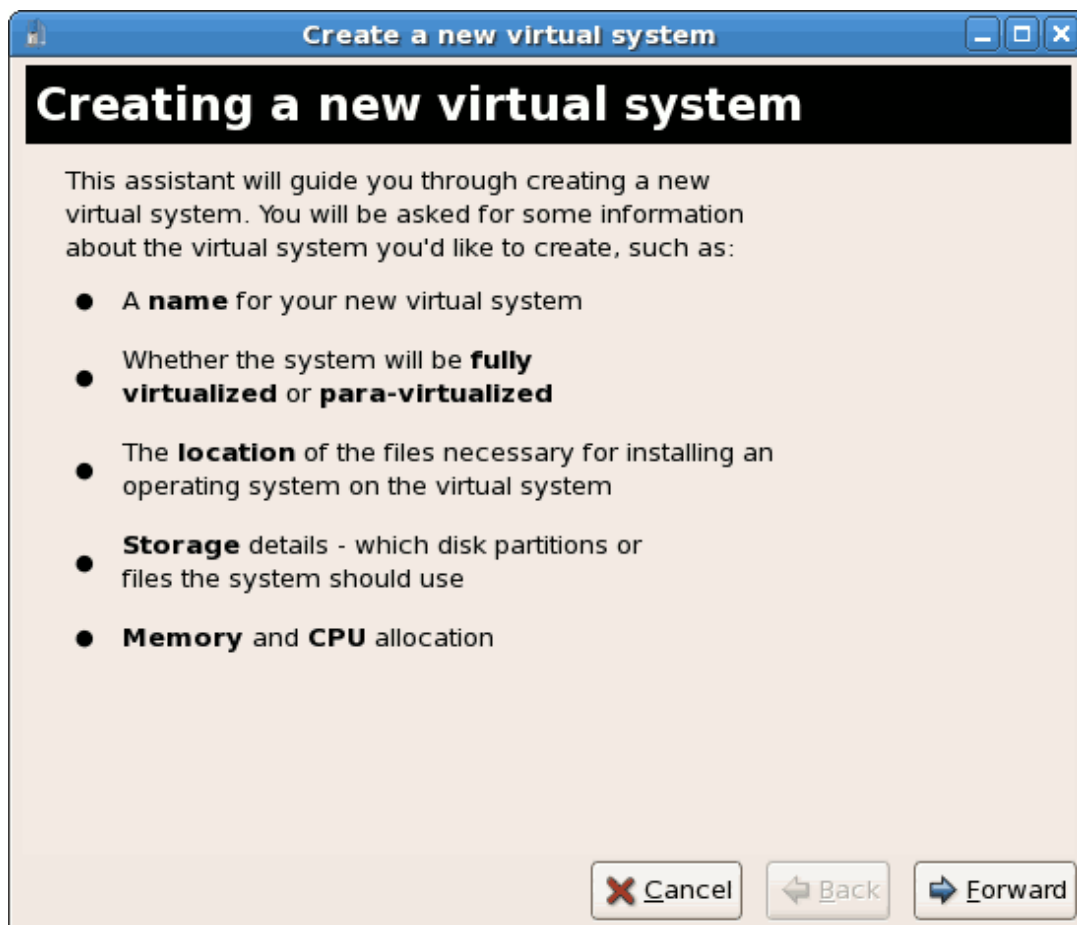
**Figure 21.6. Virtual Machine Manager window**

2. From the **File** menu, select **New machine**.

### Figure 21.7. Selecting a new machine

The Creating a new virtual system wizard appears.

3. Click **Forward**.



**Figure 21.8. Creating a new virtual system wizard**

4. Enter the name of the new virtual system, this name will be the name of the configuration file for the virtual machine, the name of the virtual disk and the name displayed by virt-manager's main screen.

Choose para-virtualization or full virtualization (hardware virtualization). Now you can continue by clicking the **Forward** button.

**Figure 21.9. Naming the virtual system**



### Warning

Do not use the `kernel-xen` as the file name for a Red Hat Enterprise Linux 5 fully virtualized guest. Using this kernel on fully virtualized guests can cause your

system to hang.

5. Choose a virtualization method to use for your guest, either [para-virtualization](#) or [full virtualization](#).



### Fully virtualized guests do not use the `kernel-xen` kernel

If you are using an *Installation Number* when installing Red Hat Enterprise Linux on a fully virtualized guest, be sure to deselect the `Virtualization` package group during the installation. The `Virtualization` package group option installs the `kernel-xen` kernel.

Para-virtualized guests are not affected by this issue. Para-virtualized guests always use the `kernel-xen` kernel.

6. Enter the location of your install media. The location of the kickstart file is optional. Then click **Forward**.

The screenshot shows a window titled "Create a new virtual system" with a sub-header "Locating installation media". The text inside says: "Please indicate where installation media is available for the operating system you would like to install on this **paravirtualized** virtual system. Optionally you can provide the URL for a kickstart file that describes your system:". There are two input fields: "Install Media URL:" with the value "ftp://10.1.1.1/trees/RHEL5-B2-Server-i386/" and "Kickstart URL:". Both fields have a dropdown arrow on the right. Below each field is an information icon and an example: "Example: http://servername.example.com/distro/i386/tree" for the first field, and "Example: ftp://hostname.example.com/ks/ks.cfg" for the second. At the bottom right are three buttons: "Cancel" (with a red X icon), "Back" (with a left arrow icon), and "Forward" (with a right arrow icon).

**Figure 21.10. Locating the installation media for para-virtualized guests**



### Storage media

For installation media on an **http** server the address should resemble "http://servername.example.com/pub/dist/rhel5" where the actual source on your local host is `/var/www/html/pub/dist/rhel5`.

For installation media on an **ftp** server the address should resemble "ftp://servername.example.com/dist/rhel5", where the actual source on your local host is `/var/ftp/pub/dist/rhel5`.

For installation media on an **NFS** server the address should resemble "nfs:servername.example.com:/dist/rhel5". The actual location depends on your **NFS** share. Use the `system-config-nfs` command to configure **NFS** to share media.

For more information on configuring these network services read the relevant sections of your Red Hat Enterprise Linux Deployment Guide in the **System->Documentation** menu.



### Networked installation media must be accessible

The installation media and kickstart files must be accessible for the host and the guest in order to install. You must take into account the IP address of both host and guest and you may need to use the IP addresses instead of hostnames.



### Tip: networked media for para-virtualization

You can use an iso image, a local CD-ROM or DVD to install para-virtualized guests. To enable this, mount the iso file or disk and host the image with **NFS**. To mount an iso image locally use the command:

```
# mount -o loop image.iso /mountpoint
```

7. For fully virtualized guests you must use an .iso file, CD-ROM or DVD.



**Figure 21.11. Locating installation media for fully virtualized guests**

8. Install either to a physical disk partition or install to a virtual file system within a file.



### Note

This example installs a virtual system within a file.

The default SELinux policy only allows storage of virtualization disk images in the `/var/lib/xen/images` folder.

To install images at a different location, `/virtimages` for example, open a terminal and create the `/virtimages` directory and set the SELinux policy settings with the command `restorecon -v /virtimages`. Specify your newly created location and the size of the virtual disk, then click **Forward**.



### A time saving tip

Creating a new disk image may take a while depending on the size and your system configuration. You can create a file before hand by using `dd` – for example to build an empty 6GB file you could use:

```
dd if=/dev/zero of=osimage.img bs=1048576 count=6144
```

Remember if you have created this file outside of the `/var/lib/xen/images` folder the file will need SELinux settings changed. Change the SELinux policy for the file with the command:

```
restorecon -v /path/to/file
```

**Create a new virtual system**

## Assigning storage space

Please indicate how you'd like to assign space on this physical host system for your new virtual system. This space will be used to install the virtual system's operating system.

☐ Normal Disk Partition:

Partition:

**Example:** /dev/hdc2

☒ Simple File:

File Location:

File Size:  MB

**Note:** File size parameter is only relevant for new files

**Tip:** You may add additional storage, including network-mounted storage, to your virtual system after it has been created using the same tools you would on a physical system.

**Figure 21.12. Assigning the storage space**

9. Connecting to the host network



Choose the “Shared Physical Device” option to allow the guest access to the same network as the host and accessible to other computers on the network.

Choose the “Virtual Network” option if you want your guest to on a virtual network. You can bridge a virtual network making it accessible to external networked computers, read [Chapter 8, Configuring networks and guests](#) for configuration instructions.



### Note

The section, [Section 17, “Creating a virtual network”](#), can guide you through the process of creating and configuring network devices or the chapter on network devices, [Chapter 12, Virtualized network devices](#).



**Figure 21.13. Connect to the host network**

10. Select memory to allocate the guest and the number of virtual CPUs then click **Forward**.

**Create a new virtual system**

## Allocate memory and CPU

**Memory:**  
Please enter the memory configuration for this VM. You can specify the maximum amount of memory the VM should be able to use, and optionally a lower amount to grab on startup.

Total memory on host machine: 2046 GB

VM Max Memory: 500

VM Startup Memory: 500

**CPUs:**  
Please enter the number of virtual CPUs this VM should start up with.

Logical host CPUs: 2

VCPUs: 1

**Tip:** For best performance, the number of virtual CPUs should be less than (or equal to) the number of logical CPUs on the host system.

**Buttons:** Cancel, Back, Forward

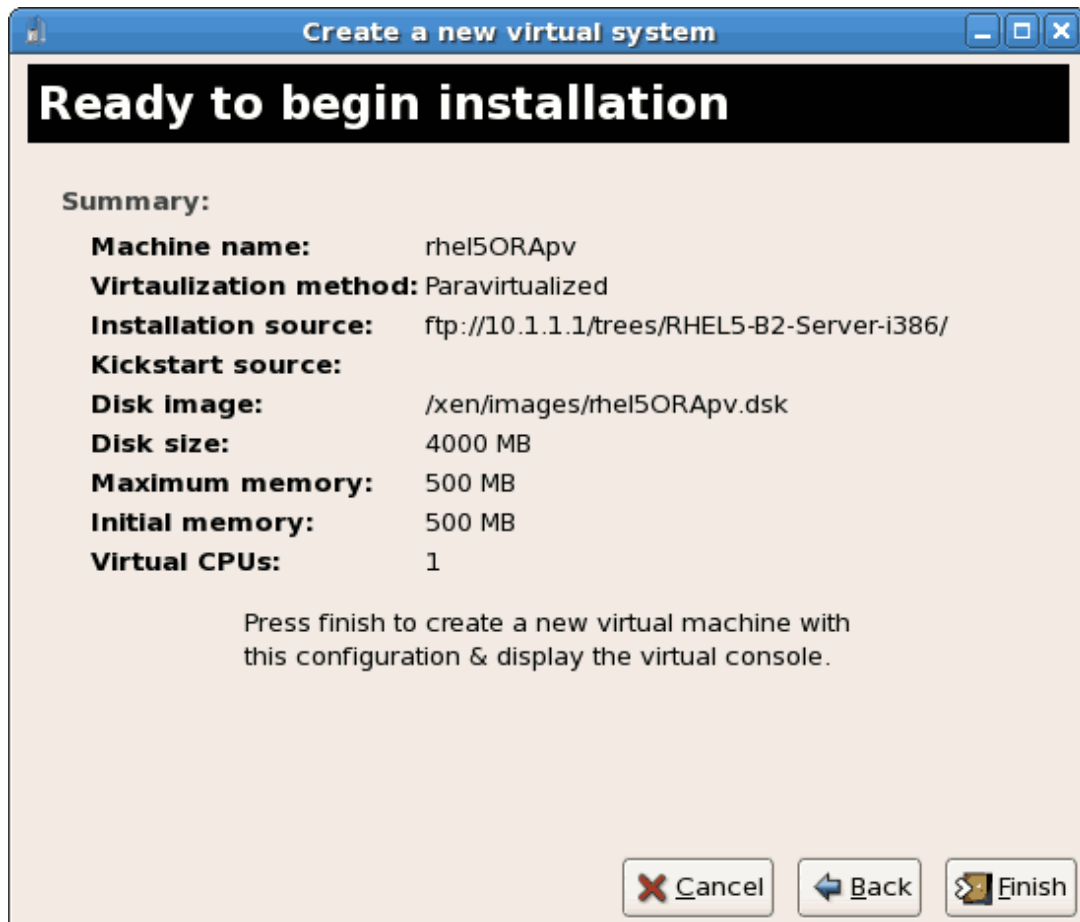
**Figure 21.14. Allocating Memory and CPU**



### Note

Avoid allocating more memory to all of your virtual machines than you have physically available. Over allocating will cause the system to use the swap partition excessively, causing unworkable performance levels.

11. Review your selections, then click **Forward** to open a console and the files start to install.



**Figure 21.15. The final `virt-manager` screen**

12. Your virtual machine will begin to boot.

**Figure 21.16. The virtual machine's boot output**

13. Type `xm create -c xen-guest` to start the Red Hat Enterprise Linux 5 guest. Right click on the guest in the Virtual Machine Manager and choose **Open** to open a virtual console.

**Figure 21.17. A Red Hat Enterprise Linux 5 guest**

## 8. 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 21.18. Restoring a virtual machine

2. The Restore Virtual Machine main window appears.

### Figure 21.19. Selecting saved virtual machine session

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 21.20. A restored virtual machine manager session

## 9. 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 21.21. 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 21.22. Displaying virtual machine details menu

The Virtual Machine Details Overview window appears. This window summarizes CPU and memory usage for the domain(s) you specified.

### Figure 21.23. Displaying guest details overview

3. On the Virtual Machine Details window, click the **Hardware** tab.

The Virtual Machine Details Hardware window appears.

### Figure 21.24. Displaying guest hardware details

4. On the **Hardware** tab, click on **Processor** to view or change the current processor memory allocation.

### Figure 21.25. Displaying processor allocation

5. On the **Hardware** tab, click on **Memory** to view or change the current RAM memory allocation.

### Figure 21.26. Displaying memory allocation

6. On the **Hardware** tab, click on **Disk** to view or change the current hard disk configuration.

### Figure 21.27. Displaying disk configuration

7. On the **Hardware** tab, click on **Network** to view or change the current network configuration.

### Figure 21.28. Displaying network configuration

## 10. 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 21.29. 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 21.30. Configuring status monitoring

3. From the Consoles area, specify how to open a console and specify an input device.

## 11. Displaying domain ID

To view the domain IDs for all virtual machines on your system:

1. From the **View** menu, select the **Domain ID** check box.

### Figure 21.31. Selecting domain IDs

2. The Virtual Machine Manager lists the Domain IDs for all domains on your system.

### Figure 21.32. Displaying domain IDs

## 12. 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 21.33. Selecting a virtual machine's status

2. The Virtual Machine Manager lists the status of all virtual machines on your system.

### Figure 21.34. Displaying a virtual machine's status

## 13. 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 21.35. Selecting the virtual CPUs option

2. The Virtual Machine Manager lists the Virtual CPUs for all virtual machines on your system.

### Figure 21.36. Displaying Virtual CPUs

## 14. 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 21.37. Selecting CPU usage

2. The Virtual Machine Manager lists the percentage of CPU in use for all virtual machines on your system.

### Figure 21.38. Displaying CPU usage

## 15. 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 21.39. 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 21.40. Displaying memory usage

## 16. Managing a virtual network

To configure a virtual network on your system:

1. From the **Edit** menu, select **Host Details**.

### Figure 21.41. Selecting a host's details

2. This will open the **Host Details** menu. Click the **Virtual Networks** tab.

### Figure 21.42. 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.

## 17. Creating a virtual network

To create a virtual network on your system:

1. Open the **Host Details** menu (refer to [Section 16, “Managing a virtual network”](#)) and click the **Add** button.



### Figure 21.43. Virtual network configuration

This will open the **Create a new virtual network** menu. Click **Forward** to continue.

### Figure 21.44. Creating a new virtual network

2. Enter an appropriate name for your virtual network and click **Forward**.

### Figure 21.45. Naming your virtual network

3. Enter an IPv4 address space for your virtual network and click **Forward**.

### Figure 21.46. 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 21.47. Selecting the DHCP range

5. Select how the virtual network should connect to the physical network.

### Figure 21.48. 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 21.49. Ready to create network

7. The new virtual network is now available in the **Virtual Network** tab of the **Host Details** menu.

**Figure 21.50. New virtual network is now available**

# Commands for Red Hat Virtualization

You can use the `xm` and `virsh` commands to create, manage, and troubleshoot virtual machines.

## 1. `virsh` the command line interface tool for virtualization

The `virsh` command is an alternative command to manage a Red Hat Enterprise Linux 5 Xen environment using a Command Line Interface (CLI). The `virsh` command is provided as part of the libvirt API which provides a common API to applications requiring standardized access to interact with Xen using a stable interface. Besides providing `virsh` the libvirt API also provides a higher level language API for C, Python, and Perl (via the CPAN network). The `virsh` command is an interactive shell which can either be used in scripts or interactively. The following `virsh` commands assume you have already run `virsh` and are at a `virsh#` prompt.

### Basic management options.

The following are basic and commonly used `virsh` commands:

command	description
help	print help
list	list domains
create	create a domain from an XML file
start	start a previously created inactive domain
destroy	destroy a domain
define	define (but do not start) a domain from an XML file
domid	convert a domain name or UUID to domain id
domuuid	convert a domain name or id to domain UUID
dominfo	domain information
domname	convert a domain id or UUID to domain name
domstate	domain state
quit	quit this interactive terminal
reboot	reboot a domain
restore	restore a domain from a saved state in a file
resume	resume a domain
save	save a domain state to a file
shutdown	gracefully shutdown a domain
suspend	suspend a domain
undefine	undefine an inactive domain

**Table 22.1. `virsh` commands**

### Resource management options.

Use the following `virsh` commands to manage resources:

command	description
<code>setmem</code>	changes the allocated memory.
<code>setmaxmem</code>	changes maximum memory limit.
<code>setvcpus</code>	changes number of virtual CPUs.
<code>vcpuinfo</code>	domain vcpu information.
<code>vcupin</code>	control the domain vcpu affinity.

**Table 22.2. `virsh` resource management commands**

### Monitoring and troubleshooting options.

Use the following `virsh` commands for monitoring and troubleshooting:

command	description
<code>version</code>	show version
<code>dumpxml</code>	domain information in XML
<code>nodeinfo</code>	node information

**Table 22.3. `virsh` monitoring and troubleshooting commands**

### Presently unsupported options.

The `connect` command that is used to connect or reconnect to a hypervisor.

### `virsh` command output.

The following are example outputs from common `virsh` commands:

- the `list` command:

```
virsh # list
Id   Name                               State
-----
0    Domain-0                           running
13   r5b2-mysql01                       blocked
```

- the `dominfo domain` command:

```
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
```

- the `domstate domain` command:

```
virsh # domstate r5b2-mysql01
blocked
```

- the `domuuid domain` command:

```
virsh # domuuid r5b2-mysql01
4a4c59a7-ee3f-c781-96e4-288f2862f011
```

- the `vcpuinfo domain` command:

```
virsh # vcpuinfo r5b2-mysql01
VCPU:      0
CPU:       0
State:     blocked
CPU time:  0.0s
CPU Affinity: yy
```

- the `dumpxml domain` command:

```
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/xen/vmlinuz.2dgnU_</kernel>
    <initrd>/var/lib/xen/initrd.UQafMw</initrd>
    <cmdline>ro root=/dev/VolGroup00/LogVol100 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>
```

- the version *domain* command:

```
virsh # version
Compiled against library: libvir 0.1.7
Using library: libvir 0.1.7
Using API: Xen 3.0.1
Running hypervisor: Xen 3.0.0
```

## 2. The `xm` command line interface

The `xm` command is used to manage your Red Hat Virtualization environment using a CLI interface. Most operations can be performed by the **virt-manager** application, including a CLI interface which is part of **virt-manager**. However, there are a few operations which currently can not be performed using **virt-manager**. As the `xm` command is part of the Xen environment a few options available with the `xm` command will not work in a Red Hat Enterprise Linux 5 environment. The list below provides an overview of command options available (and unavailable) in a Red Hat Enterprise Linux 5 environment. As an alternative to using the `xm` command one can also use the `virsh` command which is provided as part of the Red Hat Enterprise Linux 5 Xen environment. The `virsh` command is layered on top of the libvirt API which can provide a number of benefits over using the `xm` command. Namely the ability to use `virsh` in scripts and the ability to manage other hypervisors as they are integrated into the libvirt API.

### 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----
```

- `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`: terminate 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 virtual CPU assignments/placements:

```
$ 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 GRUB

The GNU Grand Unified Boot Loader (GRUB) is a program which enables the user to select which installed operating system or kernel to load at system boot time. It also allows the user to pass arguments to the kernel. The GRUB configuration file (located in `/boot/grub/grub.conf`) is used to create a list of operating systems to boot in GRUB's menu interface. When you install the `kernel-xen` RPM, a post script adds `kernel-xen` entries to the GRUB configuration file. You can edit the `grub.conf` file and enable the following GRUB parameter.

```
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/LogVol100
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/LogVol100
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 domain0.

```
dom0_max_vcpus
```

This limits the amount of CPUs visible to 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. It 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`.

The ELILO configuration file has two sections:

- Global options that affect the behavior of ELILO and all the entries. Typically there's 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, this image will be labeled *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 `--` is used to 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 :** `mem=`

**Description :** The *mem* parameter defines the hypervisor maximum RAM usage. Any additional RAM in the system will be 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.

**Parameter :** `dom0_mem=`

**Description :** *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.

**Parameter :** `dom0_max_vcpus=`

**Description :** *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.

**Parameter :** `com1=<baud>,DPS,<io_base>,<irq>`

**Description :** *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.

**Parameter :** `com2=<baud>,DPS,<io_base>,<irq>`

**Description :** Set the parameters for the second serial line. Refer the description of the *com1* parameter above.

**Parameter :** `console=<specifier_list>`

**Description :** The *console* is a comma delimited preference list for the Xen console options. Options include *vga*, *com1* and *com2*. This setting should be omitted because the hypervisor will attempt to inherit EFI console settings.



### For more information on ELILO parameters

A complete list of ELILO parameters are available from [XenSource](#)<sup>1</sup>.

---

A modified example of the configuration above, showing syntax for appending memory and cpu allocation parameters to the hypervisor:

```
image=mlinuz-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.



# Configuration files

Red Hat Virtualization configuration files 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
<i>pae</i>	Specifies the physical address extension configuration data.
<i>apic</i>	Specifies the advanced programmable interrupt controller configuration data.
<i>memory</i>	Specifies the memory size in megabytes.
<i>vcpus</i>	Specifies the numbers of virtual CPUs.
<i>console</i>	Specifies the port numbers to export the domain consoles to.
<i>nic</i>	Specifies the number of virtual network interfaces.
<i>vif</i>	Lists the randomly-assigned MAC addresses and bridges assigned to use for the domain's network addresses.
<i>disk</i>	Lists the block devices to export to the domain and exports physical devices to domain with read only access.
<i>dhcp</i>	Enables networking using DHCP.
<i>netmask</i>	Specifies the configured IP netmasks.
<i>gateway</i>	Specifies the configured IP gateways.
<i>acpi</i>	Specifies the advanced configuration power interface configuration data.

**Table 25.1. Red Hat Virtualization configuration files**

The table below, [Table 25.2, “Red Hat Virtualization configuration files reference”](#), is formatted output from `xm create --help_config`.

Parameter	Description
<i>vncpasswd=NAME</i>	Password for VNC console on HVM domain.
<i>vncviewer=no   yes</i>	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 <i>VNC_SERVER=&lt;host&gt;:&lt;port&gt;</i> . The port used by vnc is 5500 + DISPLAY. A display value with a free port is chosen if possible. Only valid when <i>vnc=1</i> .
<i>vncconsole=no   yes</i>	Spawn a vncviewer process for the domain's graphical console. Only valid when <i>vnc=1</i> .
<i>name=NAME</i>	Domain name. Must be unique.
<i>bootloader=FILE</i>	Path to bootloader.
<i>bootargs=NAME</i>	Arguments to pass to boot loader
<i>bootentry=NAME</i>	DEPRECATED. Entry to boot via boot loader. Use <i>bootargs</i> .
<i>kernel=FILE</i>	Path to kernel image.
<i>ramdisk=FILE</i>	Path to ramdisk.
<i>features=FEATURES</i>	Features to enable in guest kernel
<i>builder=FUNCTION</i>	Function to use to build the domain.
<i>memory=MEMORY</i>	Domain memory in MB.
<i>maxmem=MEMORY</i>	Maximum domain memory in MB.
<i>shadow_memory=MEMORY</i>	Domain shadow memory in MB.
<i>cpu=CPU</i>	CPU to run the VCPU0 on.
<i>cpus=CPUS</i>	CPUS to run the domain on.
<i>pae=PAE</i>	Disable or enable PAE of HVM domain.
<i>acpi=ACPI</i>	Disable or enable ACPI of HVM domain.
<i>apic=APIC</i>	Disable or enable APIC of HVM domain.
<i>vcpus=VCPUS</i>	# of Virtual CPUS in domain.
<i>cpu_weight=WEIGHT</i>	Set the new domain's cpu weight. <i>WEIGHT</i> is a float that controls the domain's share of the cpu.
<i>restart=onreboot   always   never</i>	Deprecated. Use <i>on_poweroff</i> , <i>on_reboot</i> , and <i>on_crash</i> instead. Whether the domain should be restarted on exit. - <i>onreboot</i> : restart on exit with shutdown code <i>reboot</i> - <i>always</i> : always restart on exit, ignore exit code - <i>never</i> : never restart on exit, ignore exit code



Parameter	Description
<code>on_poweroff=destroy   restart   preserve   destroy</code>	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 <code>xm destroy</code> , for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
<code>on_reboot=destroy   restart   preserve   destroy</code>	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 <code>xm destroy</code> , for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
<code>on_crash=destroy   restart   preserve   destroy</code>	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 is done until the domain is manually destroyed (using <code>xm destroy</code> , for example); - rename-restart: the old domain is not cleaned up, but is renamed and a new domain started in its place.
<code>blkif=no   yes</code>	Make the domain a block device backend.
<code>netif=no   yes</code>	Make the domain a network interface backend.
<code>tpmif=no   yes</code>	Make the domain a TPM interface backend.
<code>disk=phy:DEV,VDEV,MODE[,DOM]</code>	Add a disk device to a domain. The physical device is <code>DEV</code> , which is exported to the domain as <code>VDEV</code> . The disk is read-only if <code>MODE</code> is <code>r</code> , read-write if <code>MODE</code> is <code>w</code> . If <code>DOM</code> is specified it defines the backend driver domain to use for the disk. The option may be repeated to add more than one disk.
<code>pci=BUS:DEV.FUNC</code>	Add a PCI device to a domain, using given params (in hex). For example <code>pci=c0:02.1a</code> . The option may be repeated to add more than one pci device.

Parameter	Description
<code>ioports=FROM[-TO]</code>	Add a legacy I/O range to a domain, using given params (in hex). For example <code>ioports=02f8-02ff</code> . The option may be repeated to add more than one i/o range.
<code>irq=IRQ</code>	Add an IRQ (interrupt line) to a domain. For example <code>irq=7</code> . This option may be repeated to add more than one IRQ.
<code>usbport=PATH</code>	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.
<code>vfb=type={vnc,sdl}, vncunused=1,</code> <code>vncdisplay=N,</code> <code>vnclisten=ADDR, display=DISPLAY,</code> <code>xauthority=XAUTHORITY,</code> <code>vncpasswd=PASSWORD,</code> <code>keymap=KEYMAP</code>	Make the domain a framebuffer backend. The backend type should be either <code>sdl</code> or <code>vnc</code> . For <code>type=vnc</code> , connect an external vncviewer. The server will listen on <code>ADDR</code> (default 127.0.0.1) on port <code>N+5900</code> . <code>N</code> defaults to the domain id. If <code>vncunused=1</code> , the server will try to find an arbitrary unused port above 5900. For <code>type=sdl</code> , a viewer will be started automatically using the given <code>DISPLAY</code> and <code>XAUTHORITY</code> , which default to the current user's ones.
<code>vif=type=TYPE, mac=MAC,</code> <code>bridge=BRIDGE, ip=IPADDR,</code> <code>script=SCRIPT, backend=DOM,</code> <code>vifname=NAME</code>	Add a network interface with the given <code>MAC</code> address and bridge. The <code>vif</code> 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 <code>MAC</code> address is used. If not specified then the network backend chooses it's own <code>MAC</code> address. If bridge is not specified the first bridge found is used. If script is not specified the default script is used. If backend is not specified the default backend driver domain is used. If vifname is not specified the backend virtual interface will have name <code>vifD.N</code> where <code>D</code> is the domain id and <code>N</code> 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.
<code>vtpm=instance=INSTANCE,backend=DOM</code>	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

Parameter	Description
	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 <code>/etc/xen/vtpm.db</code> . Use the backend in the given domain.
<code>access_control=policy=POLICY,label=LABEL</code>	Add a security label and the security policy reference that defines it. The local ssid reference is calculated when starting/resuming the domain. At this time, the policy is checked against the active policy as well. This way, migrating through save/restore is covered and local labels are automatically created correctly on the system where a domain is started / resumed.
<code>nics=NUM</code>	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.
<code>root=DEVICE</code>	Set the <code>root=</code> parameter on the kernel command line. Use a device, e.g. <code>/dev/sda1</code> , or <code>/dev/nfs</code> for NFS root.
<code>extra=ARGS</code>	Set extra arguments to append to the kernel command line.
<code>ip=IPADDR</code>	Set the kernel IP interface address.
<code>gateway=IPADDR</code>	Set the kernel IP gateway.
<code>netmask=MASK</code>	Set the kernel IP netmask.
<code>hostname=NAME</code>	Set the kernel IP hostname.
<code>interface=INTF</code>	Set the kernel IP interface name.
<code>dhcp=off/dhcp</code>	Set the kernel dhcp option.
<code>nfs_server=IPADDR</code>	Set the address of the NFS server for NFS root.
<code>nfs_root=PATH</code>	Set the path of the root NFS directory.
<code>device_model=FILE</code>	Path to device model program.
<code>fda=FILE</code>	Path to fda
<code>fdb=FILE</code>	Path to fdb
<code>serial=FILE</code>	Path to serial or pty or vc

Parameter	Description
<code>localtime=no   yes</code>	Is RTC set to localtime?
<code>keymap=FILE</code>	Set keyboard layout used
<code>usb=no   yes</code>	Emulate USB devices?
<code>usbdevice=NAME</code>	Name of USB device to add?
<code>stdvga=no   yes</code>	Use std vga or cirrus logic graphics
<code>isa=no   yes</code>	Simulate an ISA only system?
<code>boot=a/b/c/d</code>	Default boot device
<code>nographic=no   yes</code>	Should device models use graphics?
<code>soundhw=audiodev</code>	Should device models enable audio device?
<code>vnc</code>	Should the device model use VNC?
<code>vncdisplay</code>	VNC display to use
<code>vnclisten</code>	Address for VNC server to listen on.
<code>vncunused</code>	Try to find an unused port for the VNC server. Only valid when <code>vnc=1</code> .
<code>sdl</code>	Should the device model use SDL?
<code>display=DISPLAY</code>	X11 display to use
<code>xauthority=XAUTHORITY</code>	X11 Authority to use
<code>uuid</code>	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 25.2. Red Hat Virtualization configuration files reference

Table 25.4, “Configuration parameter default values” lists all configuration parameters available, the Python parser function used to set the value and each parameter's default value. 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
<code>set_bool</code>	Accepted values:

Parser function	Valid arguments
	<ul style="list-style-type: none"> <li>• yes</li> <li>• y</li> <li>• no</li> <li>• yes</li> </ul>
set_float	<p>Accepts a floating point number with Python's float(). For example:</p> <ul style="list-style-type: none"> <li>• 3.14</li> <li>• 10.</li> <li>• .001</li> <li>• 1e100</li> <li>• 3.14e-10</li> </ul>
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 25.3. Python functions used to set parameter values**

Parameter	Parser function	Default value
<i>name</i>	setter	<i>default value</i>
<i>vncpasswd</i>	set_value	<i>None</i>
<i>vncviewer</i>	set_bool	<i>None</i>
<i>vncconsole</i>	set_bool	<i>None</i>
<i>name</i>	set_value	<i>None</i>
<i>bootloader</i>	set_value	<i>None</i>
<i>bootargs</i>	set_value	<i>None</i>
<i>bootentry</i>	set_value	<i>None</i>
<i>kernel</i>	set_value	<i>None</i>

Parameter	Parser function	Default value
<i>ramdisk</i>	set_value	''
<i>features</i>	set_value	''
<i>builder</i>	set_value	'linux'
<i>memory</i>	set_int	128
<i>maxmem</i>	set_int	None
<i>shadow_memory</i>	set_int	0
<i>cpu</i>	set_int	None
<i>cpus</i>	set_value	None
<i>pae</i>	set_int	0
<i>acpi</i>	set_int	0
<i>apic</i>	set_int	0
<i>vcpus</i>	set_int	1
<i>cpu_weight</i>	set_float	None
<i>restart</i>	set_value	None
<i>on_poweroff</i>	set_value	None
<i>on_reboot</i>	set_value	None
<i>on_crash</i>	set_value	None
<i>blkif</i>	set_bool	0
<i>netif</i>	set_bool	0
<i>tpmif</i>	append_value	0
<i>disk</i>	append_value	[]
<i>pci</i>	append_value	[]
<i>ioports</i>	append_value	[]
<i>irq</i>	append_value	[]
<i>usbport</i>	append_value	[]
<i>vfb</i>	append_value	[]
<i>vif</i>	append_value	[]
<i>vtpm</i>	append_value	[]
<i>access_control</i>	append_value	[]
<i>nics</i>	set_int	-1
<i>root</i>	set_value	''
<i>extra</i>	set_value	''
<i>ip</i>	set_value	''
<i>gateway</i>	set_value	''
<i>netmask</i>	set_value	''

---

Parameter	Parser function	Default value
<i>hostname</i>	set_value	<i>''</i>
<i>interface</i>	set_value	<i>"eth0"</i>
<i>dhcp</i>	set_value	<i>'off'</i>
<i>nfs_server</i>	set_value	<i>None</i>
<i>nfs_root</i>	set_value	<i>None</i>
<i>device_model</i>	set_value	<i>''</i>
<i>fda</i>	set_value	<i>''</i>
<i>fdb</i>	set_value	<i>''</i>
<i>serial</i>	set_value	<i>''</i>
<i>localtime</i>	set_bool	<i>0</i>
<i>keymap</i>	set_value	<i>''</i>
<i>usb</i>	set_bool	<i>0</i>
<i>usbdevice</i>	set_value	<i>''</i>
<i>stdvga</i>	set_bool	<i>0</i>
<i>isa</i>	set_bool	<i>0</i>
<i>boot</i>	set_value	<i>'c'</i>
<i>nographic</i>	set_bool	<i>0</i>
<i>soundhw</i>	set_value	<i>''</i>
<i>vnc</i>	set_value	<i>None</i>
<i>vncdisplay</i>	set_value	<i>None</i>
<i>vnclisten</i>	set_value	<i>None</i>
<i>vncunused</i>	set_bool	<i>1</i>
<i>sdl</i>	set_value	<i>None</i>
<i>display</i>	set_value	<i>None</i>
<i>xauthority</i>	set_value	<i>None</i>
<i>uuid</i>	set_value	<i>None</i>

**Table 25.4. Configuration parameter default values**





---

## Part VI. Tips and Tricks

---



---

## **Tips and Tricks to Enhance Productivity**

These chapters contain useful hints and tips to improve Red Hat Virtualization.



# Tips and tricks

This chapter contains a number of scripts, tips and hints for using and enhancing Red Hat Virtualization.

## 1. Automatically starting domains during the host system boot

This section explains how to make guest systems boot automatically during the host system's boot phase. You need to configure soft links in `/etc/xen/auto` to point to the guest configuration file of the guests you want to automatically boot. It is recommended to keep the number of guests small because the boot order of guests is serialized. More guests automatically started at boot cause the boot sequence to take a significantly longer time.

The example below shows how you can configure the softlink for a guest image named `example` to automatic boot during the system boot.

```
# cd /etc/xen/auto
# ls
# ln -s /var/lib/xen/images/example .
# ls -l
lrwxrwxrwx 1 root root 14 Dec 14 10:02 example -> ../example
```

## 2. 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 virtualization kernel for domain0 in order to successfully run the hypervisor. Copy your existing virtualized 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'). You need specify hypervisor specific values you have to add them to the `xen` line of your grub entry.

The output below is an example of a `grub.conf` entry from a Red Hat Virtualization system. 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 23, Configuring GRUB](#) 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
```

## 3. Example guest configuration files and parameters

The following configuration files can be used as reference examples. Normally the configuration will be created by `virt-install` or **virt-manager** during the installation of a guest. However sometimes it might be useful to have a reference available in case a new configuration needs manual creation.

### Example: para-virtualized guest's configuration file.

An example of a para-virtualized guest's configuration file:

```
name = "rhel5b2vm01"
memory = "2048"
disk = [ 'tap:aio:/var/lib/xen/images/rhel5b2vm01.dsk,xvda,w', ]
vif = [ 'mac=00:16:3e:33:79:3c, bridge=xenbr0', ]
vnc=1
vncunused=1
uuid = "302bd9ce-4f60-fc67-9e40-7a77d9b4e1ed"
bootloader="/usr/bin/pygrub"
vcpus=2
on_reboot = 'restart'
```

```
on_crash      = 'restart'
```

An example of a fully virtualized guest's configuration file:

```
name = "rhel4u4-x86_64"
builder = "hvm"
memory = "500"
disk = [ 'file:/var/lib/xen/images/rhel4u4-x86_64.dsk,hda,w', ]
vif = [ 'type=ioemu, mac=00:16:3e:09:f0:12, bridge=xenbr0', 'type=ioemu,
mac=00:16:3e:09:f0:13, bridge=xenbr1' ]
uuid = "b10372f9-91d7-a05f-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_reboot = 'restart'
```

## 4. 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 6, "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 Red Hat Virtualization bridge information using the command `brctl show`).
- 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.

```
/etc/selinux/config
```

Change the SELinux enforcement policy from `Enforcing` to `Disabled`. Use the GUI tool `system-config-securitylevel` or the command:

```
# setenforce 0
```

## 5. 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. The script below can be used to identify running on:

```
#!/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

```

## 6. 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 Red Hat Virtualization
# guests
#
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 and 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 ""
```

## 7. Limit network bandwidth for a 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 virtual machine will be able to use any bandwidth setting available on your physical network card supports. The physical network card must be mapped to one of virtual machine's virtual network interfaces. In Red Hat Virtualization the “rate” parameter part of the `VIF` entries can be used to achieve the goal of throttling certain virtual machines.

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 to specify a specific granularity of credits during a specified 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

## 8. Starting domains automatically during system boot

You can configure your guests to start automatically when you boot the system. To do this, you must modify the symbolic links that resides in `/etc/xen/auto` . This file points to the guest configuration files that you need to start automatically. The start up process is serialized, meaning that the higher the number of guests, the longer the boot process will take. This example shows you how to use symbolic links for the guest `rhel5vm01` :

```
# cd /etc/xen
# cd auto
# ls
# ln -s ../rhel5vm01 .
# ls -l
lrwxrwxrwx 1 root root 14 Dec 14 10:02 rhel5vm01 -> ../rhel5vm01
#
```

## 9. Modifying dom0

To use Red Hat Virtualization to manage domain0, you will constantly making changes to the `grub.conf` configuration file, that resides in the `/etc` directory. Because of the large number of domains to manage, many system administrators prefer to use the 'cut and paste' method when editing `grub.conf` . If you do this, make sure that you include all five lines in the Virtualization entry (or this will create system errors). If you require Xen hypervisor specific values, you must add them to 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-2.6.17-1.2519.4.21.el5xen ro root=/dev/VolGroup00/LogVol100
module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

For example, if you need to change your dom0 hypervisor's memory to 256MB at boot time, you must edit the 'xen' line and append it with the correct entry, 'dom0\_mem=256M' . This example represents the respective `grub.conf` xen entry:

```
# 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/LogVol100
module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

## 10. Configuring guest live migration

Red Hat Virtualization can migrate virtual machines between other servers running Red Hat Virtualization. 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. This value is correct, just make sure to remove the comment that comes before it.

```
(xend-relocation-address )
```

This parameter is the address that listens for relocation socket connections, after you enable the `xend-relocation-server` . When listening, it restricts the migration to a particular 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 (such as `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 Red Hat Virtualization to accept your new parameters.

## 11. Very Secure `ftpd`

`vsftpd` provides access to installation trees for para-virtualized guests (for example the Red Hat Enterprise Linux 5 repositories) or to allow the storage of public tools/kits etc. 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 the RPM package must be in your current directory).

1. To configure `vsftpd`, edit `/etc/passwd` using `vi` 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 `vsftpd` 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 ]
```

## 12. 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: 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, type:

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

The output should resemble the following:

```
# scsi_id -g -s /block/sdc
*3600a0b80001327510000015427b625e*
```

This long string of characters is the UUID. To get the device names to key off the UUID, check each device path to ensure that the UUID number is the same for each device. The UUIDs do not change when you add a new device to your system. Once you have checked the device

paths, you must create rules for the device naming. To create these rules, you must edit the `20-names.rules` file that resides in the `/etc/udev/rules.d` directory. The device naming rules you create here should 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. So the rule should resemble the following:

```
KERNEL="sd*",  BUS="scsi",  PROGRAM="/sbin/scsi_id",  
RESULT="3600a0b80001327510000015427b625e"  
,  NAME="mydevicename"
```

This causes the system to enable all devices that match `/dev/sd*` 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, you need to append the `rc.local` file that resides in the `/etc` directory with this path:

```
/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.

### 13. 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
```

### 14. Cleaning up the `/var/lib/xen/` folder

Over time you will see a number of files accumulate in `/var/lib/xen`, the usually named `vmlinux.*****` and `initrd.*****`. These files are the `initrd` and `vmlinux` 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.

### 15. Configuring a VNC Server

You can configure a VNC server using by navigating to **System, Preferences**, and selecting **Remote Desktop**. Alternatively you can run the `vino-preferences` command.

Follow these steps to run 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
```

## 16. Cloning guest configuration files

You can copy (or clone) 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.



# Creating custom Red Hat Virtualization scripts

This section will provide some information which may be useful to programmers and system administrators intending to write custom scripts to make their lives easier using Red Hat Virtualization.

[Chapter 26, \*Tips and tricks\*](#) is recommended reading for programmers thinking of making new applications which use Red Hat Virtualization.

## 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/xen/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
```



## Compiling para-virtualized driver packages from source code

You may need to rebuild the RPMs to get them working properly on your system for specific architectures. Use the source RPMs provided by Red Hat. These instructions will also allow you to install from the source RPMs.

Before compiling, the `kverrel` and `kverbase` variables in the `xenpv.spec` file must be changed to match the kernel version that binary modules are being built for. Usually, this is the value returned from `'uname -r'`.

You can then use the following command to rebuild the `kmod-xenpv` package. Change *SOURCE* to reflect the correct path to your `kmod-xenpv` RPM file.

```
cd /usr/src/redhat/SOURCE/xenpv
rpmbuild -bb kmod-xenpv
```



---

## **Part VII. Troubleshooting**

---





---

## Introduction to Troubleshooting and Problem Solving

The following chapters provide information to assist you in troubleshooting issues you may encounter using Red Hat 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/](http://www.redhat.com/docs/manuals/enterprise/)<sup>1</sup>.



### If all else fails...

If you still cannot find a fix to your problem after reading this guide and it is not a known issue, file a bug using Red Hat's Bugzilla. To create a new bug, go to [https://bugzilla.redhat.com/enter\\_bug.cgi](https://bugzilla.redhat.com/enter_bug.cgi) and select Red Hat Enterprise Linux and then fill out the form.



# How To troubleshoot Red Hat Virtualization

This chapter covers how to troubleshoot systems running Red Hat Virtualization. Troubleshooting topics covered in this chapter include:

- general troubleshooting tools,
- virtualization specific troubleshooting tools,
- troubleshooting techniques,
- log file locations and explanations, and
- general virtualization task errors.

## 1. Debugging and troubleshooting Red Hat Virtualization

This section summarizes the System Administrator applications, the networking utilities, and the Advanced Debugging Tools (for more information on using these tools to configure the Red Hat Virtualization services, see the respective configuration documentation). You can employ these standard System Administrator Tools and logs to assist with troubleshooting:

### Useful commands and applications for troubleshooting

`xentop`

`xentop` displays real-time information about a Red HAt Virtualization host system and it's domains.

`xm`

Using the `dmesg` and `log`

- `vmstat`
- `iostat`
- `lsof`

`sysstat` `iostat`, `mpstat` and `sar`.

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`
- `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.fffffefe	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.fefffffffffff		
designated-root	8000.fefffffffffff		
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

Other utilities which can be used to troubleshoot virtualization on Red Hat Enterprise Linux 5. All utilities mentioned can be found in the `Server` repositories of the Red Hat Enterprise Linux 5 Server distribution:

- **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 **vncviewer** 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.

## 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 Red Hat Virtualization logs files are standard ASCII files, and easily accessible with any ASCII based editor:

- The Red Hat Virtualization main configuration directory is `/etc/xen/`. This directory contains the `xend` daemon and other virtual machine configuration files. The networking script files reside here as well (in the `/scripts` subdirectory).
- 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.
- Red Hat Virtualization information for the `/proc` file system reside in the `/proc/xen/` directory.

### 3. Log file descriptions

Red Hat Virtualization 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, etc.) appears here. 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, etc.).
- `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 `/var/log/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.

### 4. Important directory locations

There are other utilities and log files you should when you track errors and troubleshoot problems. within Red Hat Virtualization environments:

- Virtual machines images reside in the `/var/lib/xen/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 `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 called `xend-config.sxp` and you can use this file to implement system-wide changes and configure the networking callouts.
- 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/
```

## 5. Troubleshooting with the logs

When encountering issues with installing Red Hat Virtualization, you can 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. It resides 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.

## 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-2.6.18-8.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 `ttymwatch` 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 24, Configuring ELILO](#).

```
ttymwatch --name myhost --port /dev/ttyS0
```

This pipes the output from `/dev/ttyS0` into the file `/var/log/ttymwatch/myhost.log`.

## 7. Para-virtualized guest console access

Para-virtualized guest operating systems automatically has a virtual text console configured to plumb data to the dom0 operating system. You can do this from the command line by typing:

```
xm console [domain name or number]
```

Where `domain100` represents a running name or number. You can also use the Virtual Machine Manager to display the virtual text console. On the Virtual Machine Details window, select **Serial**



**Console** from the **View** menu.

## 8. Fully virtualized guest console access

Full 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, you can then access the text console by typing:

```
xm console
```

You can also use the Virtual Machine Manager to display the serial console. On the Virtual Machine Details window, select **Serial Console** from the **View** menu.

## 9. SELinux considerations

This sections contains things to you must consider when you implement SELinux into your Red Hat Virtualization environment. 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 be used to set the `xend` in 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.

## 10. 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 volume(s) 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.

## 11. Common troubleshooting situations

When you attempt to start the `xend` service nothing happens. You type `xm list` and receive the following:

```
Error: Error connecting to xend: Connection refused. Is xend running?
```

You 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 is most likely happened here is that you rebooted your host into a kernel that is not a `xen-hypervisor` kernel. To correct this, you must select the `xen-hypervisor` kernel at boot time (or set the `xen-hypervisor` kernel to default in your `grub.conf` file).

## 12. 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.

## 13. Serial console errors

You receive no output to the serial console. To correct this problem, you must modify the `grub.conf` and change the com port parameters to:

```
serial --unit=1 --speed=115200
```

```

title RHEL5 i386 Xen (2.6.18-1.2910.el5xen)
root (hd0, 8)
kernel /boot/xen.gz-2.6.18-1.2910.el5 com2=115200,8n1
module /boot/vmlinuz-2.6.18-1.2910.el5xen to root=LABEL=RHEL5_i386
console=tty console=ttyS1115200
module /boot/initrd-2.8.6.18-12910.el5xen.img

title RHEL5 i386 xen (2.6.18.-1.2910.el5xen)
root (hd0, 8)
kernel /boot/xen.gz-2.6.18-1.2910.el5 com2=115200 console=com21

```

```
module /boot/vmlinuz2.6.18-1.2910.el5xen to root=LABEL=RHEL5_i386
console=xvc xencons=xvc
module /boot/ititrd-2.6.18-1.2910.el5xen.img
```

These changes to the `grub.conf` should enable your serial console to work correctly. You should be able to use any number for the `ttys` and it should work like `ttys0` .

## 14. Network bridge errors

Red Hat Virtualization can configure multiple Virtualization network bridges to use with multiple ethernet cards. To successfully configure multiple network bridges for ethernet cards, you must configure the second network interface by either using the system-config-network TUI/GUI, or by creating a new configuration file in `/etc/sysconfig/network-scripts` . You should use a process to setup multiple Xen bridges. This is an example config file for a second NIC called 'eth1' :

```
#/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
```

Copy the `/etc/xen/scripts/network-bridge` to `/etc/xen/scripts/network-bridge.xen` .

Edit `/etc/xen/xend-config.sxp` and add a line to your new network bridge script (this example uses "network-virtualization-multi-bridge" ).

In the `xend-config.sxp` file, the new line should reflect your new script:

```
network-script network-xen-multi-bridge
```

Remove the commenting on the line that states:

```
network-script network-bridge
```

If you want to create multiple Xen bridges, you must create a custom script. This example below creates two Xen bridges (called `xenbr0` and `xenbr1` ) and attaches them to `eth1` and `eth0` ,

respectively:

```
# !/bin/sh
# network-xen-multi-bridge
# Exit if anything goes wrong
set -e
# First arg is 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
```

If you want to create additional bridges, just use the example script and copy/paste the file accordingly.

## 15. 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:/xen/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 =
[ 'file:/xen/images/rhel5u5-x86_64.dsk.hda,w1' ]
    vif = [ 'type=ioemu, mac=00:16:3e:09:f0:12,
bridge=xenbr0', 'type=ioemu, 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'
```

## 16. Interpreting 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 `"xm mem-set dom0 NewMemSize"` to check memory.

You receive the following error:

---

<sup>1</sup> `.././../home/mhideo/.evolution//xen/images/rhel5u5-x86_64.dsk.hda,w`

```
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. Red Hat Virtualization 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]
XendDomainInfo.destroy: domain=1
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XenDomainInfo: 1457]
XendDomainInfo.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 `paе=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:

```
[root@trumble virt]# 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-2.6.18-12747.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, you must edit your guest configuration file, and modify the `vif` entry. When you locate the `vif` entry of the configuration file, assuming you are using `xenbr0` as the default bridge, ensure that the proper entry resembles the following:

```
# vif = ['mac=00:16:3e:49:1d:11, bridge=xenbr0',]
```

You receive these python deprecation errors:

```
[root@python xen]# xm shutdown win2k3xen12
[root@python xen]# 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.

## 17. 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 containing configuration files used by the `xend` daemon.
- host the script used to setup the Virtualization networking environment can be found in the `scripts` subdirectory.
- contains the configuration files used to describe virtual machines.
- Sometimes the system administrator may decided to keep the virtual machine configuration files in a different or central location. Make sure you are not working off old or stale configuration files.

`/var/log/xen/`

- directory holding all Xen related log files.

`/var/lib/xen/`

- default directory for Virtualization related file (such as XenDB and virtual machine images).

`/var/lib/xen/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/`

- Xen related information in the `/proc` filesystem.

## 18. Online troubleshooting resources

- Red Hat Virtualization Center

<http://www.openvirtualization.com><sup>2</sup>

- Red Hat Enterprise Linux 5 Beta 2 Documentation

<http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/index.html>

- Libvirt API

<http://www.libvirt.org><sup>3</sup>

- virt-manager Project Home Page

<http://virt-manager.et.redhat.com><sup>4</sup>

- Xen Community Center

<http://www.xensource.com/xen/xen/>

---

<sup>2</sup> <http://www.openvirtualization.com/>  
Virtualization Technologies Overview

<sup>3</sup> <http://www.libvirt.org/>

<sup>4</sup> <http://virt-manager.et.redhat.com/>

<http://virt.kernelnewbies.org><sup>5</sup>

- Emerging Technologies Projects

<http://et.redhat.com><sup>6</sup>

---

<sup>5</sup> <http://virt.kernelnewbies.org/>

<sup>6</sup> <http://et.redhat.com/>



# Troubleshooting

This chapter covers common problems and solutions with Red Hat Enterprise Linux virtualization.

## 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    81756888 hda
```

## 2. 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.

## 3. 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.

## 4. 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.

## 5. Wrong kernel image error - using a non-Xen kernel in a para-virtualized guest

If you try to boot a non-xen kernel in a para-virtualized guest you will see the following error message:

```
# 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 do have a kernel-xen installed in your guest you can start your guest using the command `"xm create -c GuestName"` where `GuestName` is the name of the kernel-xen. 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"`.

## 6. Wrong kernel image error - non-PAE kernel on a PAE platform

If you try 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. Red Hat Enterprise Linux 5 and Xen 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 your guest must match the architecture type of your hypervisor. Therefore if you want 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.

## 7. Fully-virtualized x86\_64 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 `paе` setting. Make sure you have an entry “`paе=1`” in your guest's configuration file.

## 8. Missing localhost entry in `/etc/hosts` causing `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
```

## 9. 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
```

## 10. Wrong bridge configured on guest causing Xen hot plug scripts to timeout

If you have moved configuration files between different hosts you may want 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 Xen 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', ]
```

## 11. 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, globals, locals)
```

```
Error: invalid syntax (win2k3xen12, line 1)
```

# Troubleshooting Para-virtualized Drivers

This chapter deals with issues you may encounter with the Red Hat Enterprise Linux hosts and fully virtualized guests using the para-virtualized drivers

## 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`

- used to record error events from `xend` and its subsystems (such as framebuffer and Python scripts etc..)

`xen-hotplug.log`

- used to log events from hotplug events.
- events such as devices not coming online or network bridges not online will be logged in this file

`qemu-dm.PID.log`

- this file is create by the `qemu-dm` process which is started for each fully-virtualized guest.
- the `PID` will be 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 subdirectory called `.virt-manager` in the user's home directory who's running **virt-manager**. For example, `~/.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 Xen environment:

`/var/lib/xen/images/`

the standard directory for file based virtual machine images.

`/var/lib/xen/xend-db/`

directory that hold the xend database which is generated every time the daemon is restarted.

`/etc/xen/`

holds a number of configuration files used to tailor your Red Hat Enterprise Linux 5 Virtualization environment to suite your local needs

- `xend-config.sxp` is the main configuration for the xend daemon. It used to enable/disable specific functionality of the Xen daemon, and to configure the callouts to Xen networking.

`/var/xen/dump/`

hold dumps generate by virtual machines or when using the `xm dump-core` command.

`/proc/xen/`

has a number of entries which can be used to retrieve additional information:

- `/proc/xen/capabilities`
- `/proc/xen/privcmd`
- `/proc/xen/balloon`

- `/proc/xen/xenbus`
- `/proc/xen/xsd_port`
- `/proc/xen/xsd_kva`

## 2. Para-virtualized guest fail to load on a Red Hat Enterprise Linux 3 guest operating system

Red Hat Enterprise Linux 3 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.

## 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.

### 4. What to do if the guest operating system has been booted with `virt-manager` or `virsh`

As mentioned in the installation notes, a guest operating system with network para-virtualized drivers installed must be started using the “# `xm create GuestName`” command. You can only use other methods for starting the guest in Red Hat Enterprise Linux 5.2.

If the guest operating system has been booted using the `virt-manager`(the GUI tool) or `virsh`(the command line application) interface the boot process will detect the “new” old Realtek card. This due to the fact `libvirt`, as the underlying API to `virt-manager` and `virsh`, will always add `type=ioemu` to the networking section followed by prompting the systems administrator to reconfigure networking inside the guest. It is recommend you interrupt the boot process (using `virt-manager`, `virsh` or `xm`) and to boot the guest using the `xm` command. In the event of the guest operating system has booted all the way to multi-user mode you will detect that there is no networking active as the backend and frontend drivers are not connected properly.

To fix this issue, shut down the guest and boot it using “`xm create`”. During the boot process `kudzu` (the hardware detection process) will detect the “old” Realtek card. Simply select “**Remove Configuration**” to delete the Realtek card from the guest operating system. The guest should continue to boot and configure the network interfaces correctly.

You can identify if your guest has been booted with `virt-manager`, `virsh` or “`xm create`” using the command “# `xm list -long YourGuestName`”

In the screenshot below you can see the entry “`ioemu`” highlighted in the “`device vif`” (networking) section. This would mean the guest was booted with `virt-manager` or `virsh` and networking is not configured correctly, that is, without the para-virtualized network driver.

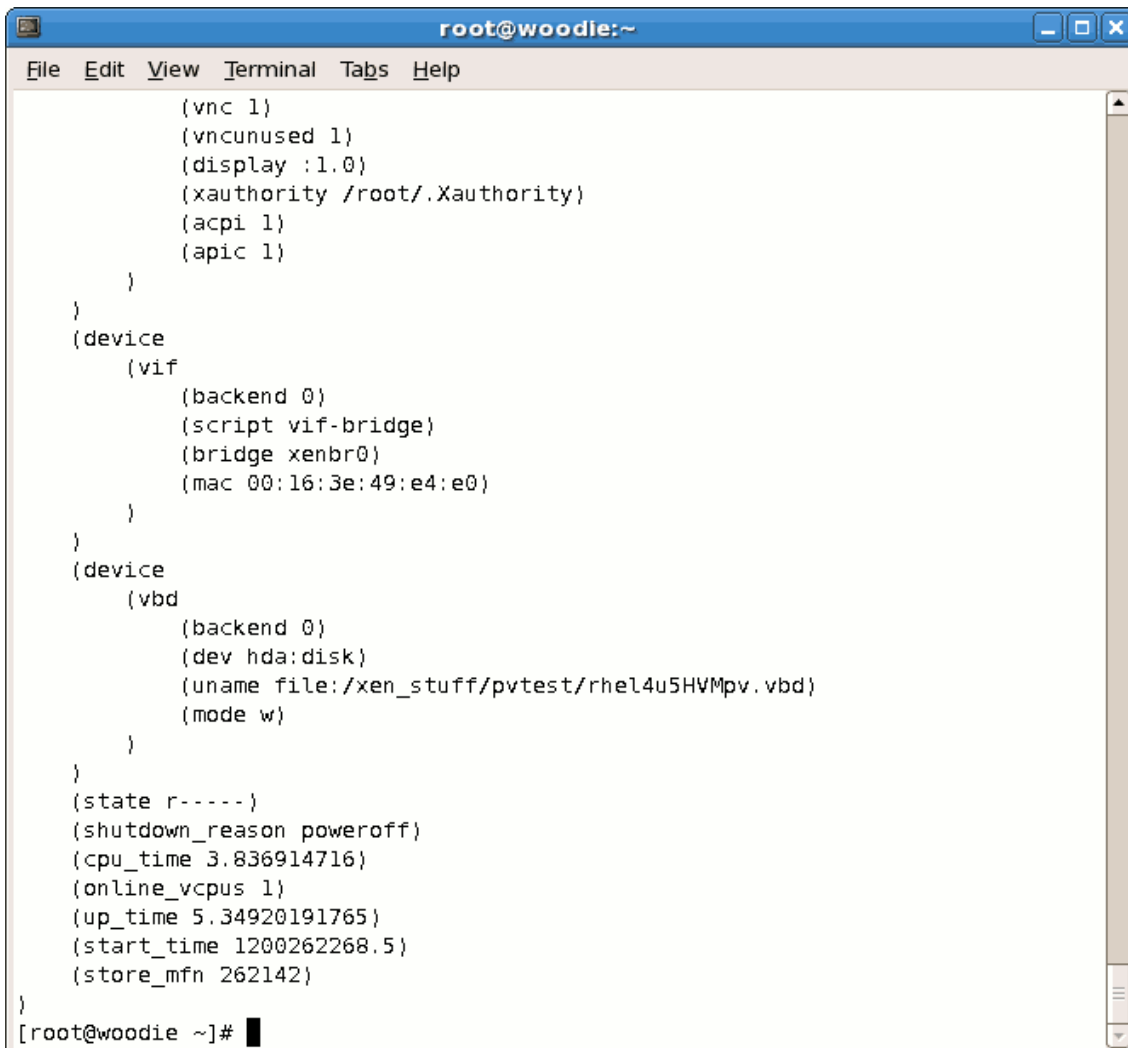
```

root@woodie:~
File Edit View Terminal Tabs Help

    (apic 1)
    (pae 1)
    (serial pty)
    (vnc 1)
    (vncunused 1)
  )
}
(device
  (vif
    (backend 0)
    (script vif-bridge)
    (bridge xenbr0)
    (mac 00:16:3e:49:e4:e0)
    (type ioemu)
  )
)
(device
  (vbd
    (backend 0)
    (dev hda:disk)
    (uname file:/xen_stuff/pvtest/rhel4u5HVMpv.vbd)
    (mode w)
  )
)
(state -b----)
(shutdown_reason poweroff)
(cpu_time 20.085893558)
(online_vcpus 1)
(up_time 130.461688042)
(start_time 1200262038.12)
(store_mfn 262142)
}
[root@woodie ~]#

```

In the screenshot below you can see there is no “type ioemu” entry in the “device vif” section so you can safely assume the guest has been booted with “xm create YourGuestName”. This means networking is configured to use the para-virtualized network driver.



```

root@woodie:~
File Edit View Terminal Tabs Help

    (vnc 1)
    (vncunused 1)
    (display :1.0)
    (xauthority /root/.Xauthority)
    (acpi 1)
    (apic 1)
  }
}
(device
  (vif
    (backend 0)
    (script vif-bridge)
    (bridge xenbr0)
    (mac 00:16:3e:49:e4:e0)
  )
)
(device
  (vbd
    (backend 0)
    (dev hda:disk)
    (uname file:/xen_stuff/pvtest/rhel4u5HVMpv.vbd)
    (mode w)
  )
)
}
(state r-----)
(shutdown_reason poweroff)
(cpu_time 3.836914716)
(online_vcpus 1)
(up_time 5.34920191765)
(start_time 1200262268.5)
(store_mfn 262142)
}
[root@woodie ~]#

```

## 5. 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

```

## 6. 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 xlxbd_add at
/local/domain/0/backend/vbd/21/76
vbd vbd-768: 19 xlxbd_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]

```

## 7. 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 (see [Section 6, “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. Revision History

## Revision History

Revision 5.2-10                      Wednesday May 14 2008                      ChristopherCurran<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 April 7 2008                      ChristopherCurran<ccurran@redhat.com>

Book updated to remove redundant chapters and headings

Virtual Machine Manager updated for 5.1.

Revision 5.2-7                      Mon March 31 2008                      ChristopherCurran<ccurran@redhat.com>

Resolves: #322761

Many spelling and grammar errors corrected.

Chapter on Remote Management added.

Revision 5.2-5                      Tue Mar 19 2008                      ChristopherCurran<ccurran@redhat.com>

Resolves: #428915

New Virtualization Guide created.



---

## Appendix B. Red Hat Virtualization system architecture

A functional Red Hat Virtualization system is multi-layered and is driven by the privileged Red Hat Virtualization component. Red Hat Virtualization can host multiple guest operating systems. Each guest operating system runs in its own domain, Red Hat Virtualization schedules virtual CPUs within the virtual machines to make the best use of the available physical CPUs. Each guest operating system handles its own applications. These guest operating systems schedule each application accordingly.

You can deploy Red Hat Virtualization 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 Red Hat Virtualization, each guest's **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). The Virtual Machine Manager schedules the vcpus 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 logins 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**: domain name (domain-name), identity (domain-id), or UUID. The domain-name is a text string that corresponds to a guest configuration file. The domain-name is used to launch the guests, and when the guest runs the same name is used to identify and control it. The domain-id is a unique, non-persistent number that gets assigned to an active domain and is used to identify and control it. 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 C. Additional resources

To learn more about Red Hat Virtualization, refer to the following resources.

## 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 Red Hat Virtualization 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.
- <http://www.libvirt.org/> is the official website for the `libvirt` virtualization API that interacts with the virtualization framework of a host OS.
- <http://virt-manager.et.redhat.com/> is the project website for the **Virtual Machine Manager** (`virt-manager`), the graphical application for managing virtual machines.

## 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.

## B

**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 natively installed operating system.

## D

**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](#)).

## F

**Full virtualization** You can deploy Red Hat Virtualization in one of two choices: full virtualization or para-virtualization. Full virtualization provides total abstraction of the underlying physical system ([Bare-metal](#)) and creates 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](#).

## G

Guest system                      Also known as guests, virtual machines or [domU](#).

## H

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 the host system allowing virtual machines to run on the host's hardware as well.

Host                      The host operating system, also known as [Domains](#).  
  
The host environment runs the software for [Fully virtualized](#) and [Fully virtualized](#) guest systems.

## I

I/O                      Short for input/output (pronounced "eye-oh"). The term I/O is used to describe 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.

## K

Kernel-based Virtual Machine                      KVM is a [Full virtualization](#) kernel module which will be incorporated into future releases of Red Hat Enterprise Linux. KVM is presently available in the fedora Linux distribution and other Linux distributions.

## L

---

LUN	Logical Unit Numbers(LUN) is the number assigned to a logical unit (a SCSI protocol entity).
-----	--

## M

Migration	See also <a href="#">Relocation</a>
-----------	-------------------------------------

Migration refers to the process of moving a para-virtualized guest images from one Red Hat Virtualization server to another. This other server could be on the same server or a different server, including servers in other locations.

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.
---------------	---

## P

Para-virtualization	Para-virtualization uses a special kernel, sometimes referred to as the xen kernel or kernel-xen to virtualized another environment while using the hosts libraries and devices. A para-virtualized installation will have complete access to all devices on the system. Para-virtualization is significantly faster than full virtualization can can be effectively 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 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.
--------------------------	--

## R

Relocation	Another term for <a href="#">Migration</a> usually used to describe moving a virtual machine image across geographic locations.
------------	---

## V

Virtual cpu	A system running Red Hat Virtualization has a number of virtual cpus, or vcpus. The number of vcpus is finite and represents the total number of vcpus 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.