Red Hat Enterprise Linux 6 Virtualization Getting Started Guide

An introduction to virtualization concepts

Dayle Parker
Laura Novich
Jacquelynn East
Scott Radvan

Red Hat Enterprise Linux 6 Virtualization Getting Started Guide

An introduction to virtualization concepts

Dayle Parker
Red Hat Engineering Content Services
dayleparker@redhat.com

Laura Novich
Red Hat Engineering Content Services
lnovich@redhat.com

Jacquelynn East
Red Hat Engineering Content Services
jeast@redhat.com

Scott Radvan
Red Hat Engineering Content Services
sradvan@redhat.com

Keywords

Abstract
The Red Hat Enterprise Linux Virtualization Getting Started Guide describes the basics of virtualization and the virtualization products and technologies that are available with Red Hat Enterprise Linux.

Table of Contents

Preface

1. Document Conventions

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

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

1.1. Typographic Conventions

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

`Mono-spaced Bold`

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

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

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

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

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

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

Proportional Bold

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

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **`Buttons`** tab, click the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** →

Character Map from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

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

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

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

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

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

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

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

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

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

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

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

import javax.naming.InitialContext;

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

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

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

1.3. Notes and Warnings

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

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

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

Warning

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

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com. Through the customer portal, you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products.
- submit a support case to Red Hat Global Support Services (GSS).
- access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click on the name of any mailing list to subscribe to that list or to access the list archives.

2.2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **Red Hat Enterprise Linux 6.**

When submitting a bug report, be sure to mention the manual's identifier: *doc-Virtualization_Getting_Started_Guide*

If you have a suggestion for improving the documentation, try to 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.

Chapter 1. Introduction

The *Virtualization Getting Started Guide* introduces the basics of virtualization and assists with the navigation of other virtualization documentation and products that Red Hat provides.

This guide also explains the advantages of virtualization and dispels some common myths that exist regarding virtualization.

1.1. Who should read this guide?

This guide is designed for anyone wishing to understand the basics of virtualization, but may be of particular interest to:

- Those who are new to virtualization and seeking knowledge about the benefits offered.
- Those considering deployment of virtualized machines in their environment.
- Those looking for an overview of the virtualization technologies that Red Hat produces and supports.

1.2. Virtualization in Red Hat Enterprise Linux 6

Red Hat Enterprise Linux contains packages and tools to support a variety of virtualized environments.

Virtualization in Red Hat Enterprise Linux 6 is carried out by KVM (Kernel-based Virtual Machine). KVM is a full virtualization solution built into Red Hat Enterprise Linux 6.

Refer to Chapter 4, *Introduction to Red Hat virtualization products* for more about the virtualization products available in Red Hat Enterprise Linux 6.

In addition to this guide, the following titles cover virtualization with Red Hat Enterprise Linux:

- *Virtualization Host Configuration and Guest Installation Guide*: This guide provides information on installing virtualization software and configuring guest machines on a virtualization host.
- *Virtualization Administration Guide*: This guide covers administration of hosts, networking, storage, device and guest management using either virt-manager or virsh, a libvirt and QEMU reference, and troubleshooting information.
- *Virtualization Security Guide*: This guide provides an overview of virtualization security technologies provided by Red Hat. Also included are recommendations for securing hosts, guests, and shared infrastructure and resources in virtualized environments.
- *Virtualization Tuning and Optimization Guide*: This guide provides tips, tricks and suggestions for making full use of virtualization performance features and options for your systems and guest virtual machines.

1.3. Red Hat Enterprise Virtualization (RHEV)

Red Hat Enterprise Virtualization (RHEV) is a complete enterprise virtualization management solution for server and desktop virtualization, based on Kernel-based Virtual Machine (KVM) technology.

Designed for enterprise-class scalability and performance, Red Hat Enterprise Virtualization enables management of your entire virtual infrastructure, including hosts, virtual machines, networks, storage, and users from a centralized graphical interface.

Red Hat Enterprise Virtualization includes the RHEV Manager infrastructure management system and the RHEV Hypervisor, which supports a wide range of Windows and Linux server and desktop operating systems — all while delivering reliability, stability, and the lowest total cost of ownership in its class.

More information on Red Hat Enterprise Virtualization can be found at http://www.redhat.com/products/virtualization/.

Download a fully supported 60-day evaluation version of Red Hat Enterprise Virtualization 3 at http://www.redhat.com/promo/rhev3/.

The full collection of Red Hat Enterprise Virtualization documentation can be found at http://access.redhat.com/knowledge/docs/.

Chapter 2. What is virtualization and migration?

This chapter discusses terms related to virtualization and migration.

2.1. What is virtualization?

Virtualization is a broad computing term used for running software, usually multiple operating systems, concurrently and in isolation from other programs on a single system. Most existing implementations of virtualization use a *hypervisor*, a software layer or subsystem that controls hardware and provides *guest operating systems* with access to underlying hardware. The hypervisor allows multiple operating systems, called *guests*, to run on the same physical system by offering virtualized hardware to the guest operating system. There are various methods for virtualizing operating systems:

Full virtualization

Full virtualization uses the hardware features of the processor to provide guests with total abstraction of the underlying physical system. This creates a new virtual system, called a *virtual machine*, that allows guest operating systems to run without modifications. The guest operating system and any applications on the guest virtual machine are unaware of their virtualized environment and run normally. Hardware-assisted virtualization is the technique used for full virtualization with KVM (Kernel-based Virtual Machine) in Red Hat Enterprise Linux.

Para-virtualization

Para-virtualization employs a collection of software and data structures that are presented to the virtualized guest, requiring software modifications in the guest to use the para-virtualized environment. Para-virtualization can encompass the entire kernel, as is the case for Xen para-virtualized guests, or drivers that virtualize I/O devices.

Software virtualization (or emulation)

Software virtualization uses slower binary translation and other emulation techniques to run unmodified operating systems. Software virtualization is unsupported by Red Hat Enterprise Linux.

Note

For more information and detailed instructions on guest installation, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

2.2. Migration

Migration describes the process of moving a guest virtual machine from one host to another. This is possible because the virtual machines are running in a virtualized environment instead of directly on the hardware. There are two ways to migrate a virtual machine: live and offline.

Migration Types

Offline migration

An offline migration suspends the guest virtual machine, and then moves an image of the virtual machine's memory to the destination host. The virtual machine is then resumed on the

destination host and the memory used by the virtual machine on the source host is freed.

Live migration

Live migration is the process of migrating an active virtual machine from one physical host to another.

2.2.1. Benefits of migrating virtual machines

Migration is useful for:

Load balancing

When a host machine is overloaded, one or many of its virtual machines could be migrated to other hosts using live migration.

Upgrading or making changes to the host

When the need arises to upgrade, add, or remove hardware devices on one host, virtual machines can be safely relocated to other hosts. This means that guests do not experience any downtime due to changes that are made to any of the hosts.

Energy saving

Virtual machines can be redistributed to other hosts and the unloaded host systems can be powered off to save energy and cut costs in low usage periods.

Geographic migration

Virtual machines can be moved to another physical location for lower latency or for other special circumstances.

It is important to understand that the migration process moves the virtual machine's memory, and from Red Hat Enterprise Linux 6.3, the disk volume associated with the virtual machine is also migrated. This process is done using live block migration.

Shared, networked storage can be used to store guest images to be migrated. When migrating virtual machines, it is recommended to use **libvirt**-managed storage pools for shared storage.

Note

For more information on migration, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

2.3. Virtualized to virtualized migration (V2V)

Red Hat Enterprise Linux 6 provides tools for converting virtual machines from other types of hypervisors to KVM. The **`virt-v2v`** tool converts and imports virtual machines from Xen, other versions of KVM, and VMware ESX.

Note

For more information on V2V, refer to the *Red Hat Enterprise Linux 6 V2V Guide*.

Chapter 3. Advantages and misconceptions of virtualization

There are many advantages to virtualization and perhaps an equal amount of misconceptions surrounding it. This chapter explores these points.

3.1. Virtualization costs

A common misconception is that virtualization is too expensive to justify the change. Virtualization can be expensive to introduce but often it saves money in the long term. It is important to perform a Return on Investment (ROI) analysis to determine the best use of virtualization in your environment. Consider the following benefits:

Less power

Using virtualization negates much of the need for multiple physical platforms. This equates to less power being drawn for machine operation and cooling, resulting in reduced energy costs. The initial cost of purchasing multiple physical platforms, combined with the machines' power consumption and required cooling, is drastically cut by using virtualization.

Less maintenance

Provided adequate planning is performed before migrating physical systems to virtualized ones, less time is spent maintaining them. This means less money being spent on parts and labor.

Extended life for installed software

Older versions of software may not run on newer, bare metal machines directly. However, by running the older software virtually on a larger, faster system, the life of the software may be extended while taking advantage of the performance from the newer system.

Predictable costs

A Red Hat Enterprise Linux subscription provides support for virtualization at a fixed rate, making it easy to predict costs.

Less space

Consolidating servers onto fewer machines means less physical space is required. This means the space normally occupied by server hardware can be used for other purposes.

3.2. Virtualization learning curve

A misconception exists that virtualization is difficult to learn. In truth, virtualization is no more difficult or easy to learn than any new process. The skills required for managing and supporting a physical environment are easily transferable to a virtual one. Virtual environments function similarly to their physical counterparts, ensuring the learning curve remains a slight one.

3.3. Performance

On older virtualization versions that supported only a single CPU, virtual machines experienced noticeable performance limitations. This created a long-lasting misconception that virtualization solutions are slow.

This is no longer the case; advances in technology allow virtual machines to run at much faster speeds than previously. Benchmarks show that typical server applications run on virtual machines with nearly the same efficiency as on bare metal systems.

- The industry standard SAP Sales and Distribution (SD) Standard Application Benchmark found Red Hat Enterprise Linux 6.2 and KVM to demonstrate a virtualization efficiency of 85% when comparing a bare metal system running on identical hardware.
- Red Hat Enterprise Linux 6.1 and KVM achieved record-setting virtualization performance in the SPECvirt_sc2010 benchmark recorded by the Standard Performance Evaluation Corporation (SPEC), setting the best virtual performance mark of any published SPECvirt result. The SPECvirt_sc2010 metric measures the end-to-end performance of system components in virtualized data center servers.

Note

For more details on these virtualization benchmarks, visit:

- Red Hat Knowledgebase, *SAP-SD Benchmark running in a VM – Leadership Performance using RHEL 6 / KVM* at https://access.redhat.com/knowledge/articles/216943
- The *Standard Performance Evaluation Corporation (SPEC)* at http://www.spec.org
- *Red Hat Achieves New Top Virtualization Performance Benchmark with HP* at http://investors.redhat.com/releasedetail.cfm?ReleaseID=617594

3.4. Disaster recovery

Disaster recovery is quicker and easier when the systems are virtualized. On a physical system, if something serious goes wrong, a complete re-install of the operating system is usually required, resulting in hours of recovery time. However, if the systems are virtualized this is much faster due to migration ability. If the requirements for live migration are followed, virtual machines can be restarted on another host, and the longest possible delay would be in restoring guest data. Also, because each of the virtualized systems are completely separate to each other, one system's downtime will not affect any others.

3.5. Security

A virtual machine uses SELinux and sVirt to improve security in virtualization. This section includes an overview of the security options available.

3.5.1. Virtualization security features

SELinux

SELinux was developed by the US National Security Agency and others to provide Mandatory Access Control (MAC) for Linux. Under control of SELinux, all processes and files are given what is known as a *type*, and access is limited by fine-grained controls. SELinux limits the abilities of an attacker and works to prevent many common security exploits such as buffer overflow attacks and privilege escalation.

SELinux strengthens the security model of Red Hat Enterprise Linux hosts and virtualized Red Hat Enterprise Linux guests. SELinux is configured and tested to work, by default, with all virtualization tools shipped with Red Hat Enterprise Linux 6.

sVirt

sVirt is a technology included in Red Hat Enterprise Linux 6 that integrates SELinux and virtualization. It applies Mandatory Access Control (MAC) to improve security when using virtual machines, and improves security and hardens the system against bugs in the hypervisor that might be used as an attack vector for the host or to another virtual machine.

Note

For more information on security for virtualization, refer to the *Red Hat Enterprise Linux 6 Virtualization Security Guide*.

3.6. Virtualization for servers and individuals

Virtualization is not just for servers; it can be useful for individuals as well. Desktop virtualization offers centralized management, an improved desktop solution, and better disaster recovery. By using connection software, it is possible to connect to a desktop remotely.

For servers, virtualization is not only for larger networks, but for any situation with two or more servers. It provides live migration, high availability, fault tolerance, and streamlined backups.

3.6.1. Virtualization deployment scenarios

These are examples of common deployment scenarios for virtualization, and the tools that can be used to deploy these scenarios.

Small deployments of up to 3 physical hosts and 10 guests: virt-manager

A tool such as virt-manager can be useful to a small business running several servers that do not have strict uptime requirements or service-level agreements (SLAs). In this environment, a single administrator may be responsible for the entire infrastructure, and maintaining procedural flexibility is important if a component needs to be changed. This environment may contain applications such as web servers, file and print servers, and application servers.

Large deployments or mission-critical applications: Red Hat Enterprise Virtualization (RHEV)

A full virtualization platform such as Red Hat Enterprise Virtualization (RHEV) might suit an enterprise running larger deployments or mission-critical applications. In this environment, the physical infrastructure is large enough to require an IT department and the business requirements demand a defined response to new needs. Some examples of a large deployment suited to Red Hat Enterprise Virtualization may include databases, trading platforms, or messaging systems that must run continuously without any downtime.

Software developers producing management applications: libvirt

Both virt-manager and Red Hat Enterprise Virtualization (RHEV) use libvirt to manage virtual machines. libvirt is a virtualization application programming interface (API) that allows software developers to produce and adapt management applications.

Chapter 4. Introduction to Red Hat virtualization products

This chapter introduces the various virtualization products available in Red Hat Enterprise Linux.

4.1. KVM and virtualization in Red Hat Enterprise Linux

What is KVM?

KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on AMD64 and Intel 64 hardware that is built into the standard Red Hat Enterprise Linux 6 kernel. It can run multiple, unmodified Windows and Linux guest operating systems. The KVM hypervisor in Red Hat Enterprise Linux is managed with the **libvirt** API and tools built for **libvirt** (such as **`virt-manager`** and **`virsh`**). Virtual machines are executed and run as multi-threaded Linux processes controlled by these tools.

Overcommitting

The KVM hypervisor supports *overcommitting* of system resources. Overcommitting means allocating more virtualized CPUs or memory than the available resources on the system. Memory overcommitting allows hosts to utilize memory and virtual memory to increase guest densities.

Important

Overcommitting involves possible risks to system stability. For more information on overcommitting with KVM, and the precautions that should be taken, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

Thin provisioning

Thin provisioning allows the allocation of flexible storage and optimizes the available space for every guest virtual machine. It gives the appearance that there is more physical storage on the guest than is actually available. This is not the same as overcommitting as this only pertains to storage and not CPUs or memory allocations. However, like overcommitting, the same warning applies.

Important

Thin provisioning involves possible risks to system stability. For more information on thin provisioning with KVM, and the precautions that should be taken, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

KSM

Kernel SamePage Merging (KSM), used by the KVM hypervisor, allows KVM guests to share identical memory pages. These shared pages are usually common libraries or other identical, high-use data. KSM allows for greater guest density of identical or similar guest operating systems by avoiding memory duplication.

Note

For more information on KSM, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

QEMU Guest Agent

The *QEMU Guest Agent* runs on the guest operating system and allows the host machine to issue commands to the guest operating system.

Note

For more information on the QEMU Guest Agent, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

KVM guest virtual machine compatibility

To verify whether your processor supports the virtualization extensions and for information on enabling the virtualization extensions if they are disabled, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

Red Hat Enterprise Linux 6 servers have certain support limits.

The following URLs explain the processor and memory amount limitations for Red Hat Enterprise Linux:

- For host systems: http://www.redhat.com/products/enterprise-linux/server/compare.html
- For hypervisors: http://www.redhat.com/resourcelibrary/articles/virtualization-limits-rhel-hypervisors

For a complete chart of supported operating systems and host and guest combinations refer to http://www.redhat.com/resourcelibrary/articles/enterprise-linux-virtualization-support.

4.2. libvirt and libvirt tools

The *libvirt* package is a hypervisor-independent virtualization API that is able to interact with the virtualization capabilities of a range of operating systems.

The *libvirt* package provides:

- A common, generic, and stable layer to securely manage virtual machines on a host.
- A common interface for managing local systems and networked hosts.
- All of the APIs required to provision, create, modify, monitor, control, migrate, and stop virtual machines, but only if the hypervisor supports these operations. Although multiple hosts may be accessed with **libvirt** simultaneously, the APIs are limited to single node operations.

The *libvirt* package is designed as a building block for higher level management tools and applications, for example, **`virt-manager`** and the **`virsh`** command-line management tools. With the exception of migration capabilities, **libvirt** focuses on managing single hosts and provides APIs to enumerate,

monitor and use the resources available on the managed node, including CPUs, memory, storage, networking and Non-Uniform Memory Access (NUMA) partitions. The management tools can be located on separate physical machines from the host using secure protocols.

Red Hat Enterprise Linux 6 supports **libvirt** and included **libvirt**-based tools as its default method for virtualization management (as in Red Hat Enterprise Virtualization Management).

The *libvirt* package is available as free software under the GNU Lesser General Public License. The *libvirt* project aims to provide a long term stable C API to virtualization management tools, running on top of varying hypervisor technologies. The *libvirt* package supports Xen on Red Hat Enterprise Linux 5, and it supports KVM on both Red Hat Enterprise Linux 5 and Red Hat Enterprise Linux 6.

virsh

The **`virsh`** command-line tool is built on the **libvirt** management API and operates as an alternative to the graphical **`virt-manager`** application. The **`virsh`** command can be used in read-only mode by unprivileged users or, with root access, full administration functionality. The **`virsh`** command is ideal for scripting virtualization administration.

`virt-manager`

`virt-manager` is a graphical desktop tool for managing virtual machines. It allows access to graphical guest consoles and can be used to perform virtualization administration, virtual machine creation, migration, and configuration tasks. The ability to view virtual machines, host statistics, device information and performance graphs is also provided. The local hypervisor and remote hypervisors can be managed through a single interface.

Note

For more information on **`virsh`** and **`virt-manager`**, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

4.3. Virtualized hardware devices

Virtualization on Red Hat Enterprise Linux 6 presents three distinct types of system devices to virtual machines. The three types include:

- Virtualized and emulated devices
- Para-virtualized devices
- Physically shared devices

These hardware devices all appear as being physically attached to the virtual machine but the device drivers work in different ways.

4.3.1. Virtualized and emulated devices

KVM implements many core devices for virtual machines in software. These emulated hardware devices are crucial for virtualizing operating systems.

Emulated devices are virtual devices which exist entirely in software.

Emulated drivers may use either a physical device or a virtual software device. Emulated drivers are a

translation layer between the virtual machine and the Linux kernel (which manages the source device). The device level instructions are completely translated by the KVM hypervisor. Any device, of the same type (storage, network, keyboard, and mouse) and recognized by the Linux kernel, may be used as the backing source device for the emulated drivers.

Virtual CPUs (vCPUs)

A host system can have up to 160 virtual CPUs (vCPUs) that can be presented to guests for their use, regardless of the number of host CPUs.

Emulated graphics devices

Two emulated graphics devices are provided. These devices can be connected to with the SPICE (Simple Protocol for Independent Computing Environments) protocol or with VNC:

- A Cirrus CLGD 5446 PCI VGA card (using the *cirrus* device)
- A standard VGA graphics card with Bochs VESA extensions (hardware level, including all non-standard modes)

Emulated system components

The following core system components are emulated to provide basic system functions:

- Intel i440FX host PCI bridge
- PIIX3 PCI to ISA bridge
- PS/2 mouse and keyboard
- EvTouch USB Graphics Tablet
- PCI UHCI USB controller and a virtualized USB hub
- Emulated serial ports
- EHCI controller, virtualized USB storage and a USB mouse

Emulated sound devices

Red Hat Enterprise Linux 6.1 and above provides an emulated (Intel) HDA sound device, **`intel-hda`**. This device is supported on the following guest operating systems:

- Red Hat Enterprise Linux 6, for i386 and x86_64 architectures
- Red Hat Enterprise Linux 5, for i386 and x86_64 architectures
- Red Hat Enterprise Linux 4, for i386 and x86_64 architectures
- Windows 7, for i386 and x86_64 architectures
- Windows 2008 R2, for the x86_64 architecture

The following two emulated sound devices are also available, but are not recommended due to compatibility issues with certain guest operating systems:

- **`ac97`**, an emulated Intel 82801AA AC97 Audio compatible sound card
- **`es1370`**, an emulated ENSONIQ AudioPCI ES1370 sound card

Emulated watchdog devices

Red Hat Enterprise Linux 6.0 and above provides two emulated watchdog devices. A watchdog can be used to automatically reboot a virtual machine when it becomes overloaded or unresponsive.

The *watchdog* package must be installed on the guest.

The two devices available are:

- **i6300esb**, an emulated Intel 6300 ESB PCI watchdog device. It is supported in guest operating system Red Hat Enterprise Linux versions 6.0 and above, and is the recommended device to use.
- **ib700**, an emulated iBase 700 ISA watchdog device. The **ib700** watchdog device is only supported in guests using Red Hat Enterprise Linux 6.2 and above.

Both watchdog devices are supported in i386 and x86_64 architectures for guest operating systems Red Hat Enterprise Linux 6.2 and above.

Emulated network devices

There are two emulated network devices available:

- The **e1000** device emulates an Intel E1000 network adapter (Intel 82540EM, 82573L, 82544GC).
- The **rtl8139** device emulates a Realtek 8139 network adapter.

Emulated storage drivers

Storage devices and storage pools can use these emulated devices to attach storage devices to virtual machines. The guest uses an emulated storage driver to access the storage pool.

Note that like all virtual devices, the storage drivers are not storage devices. The drivers are used to attach a backing storage device, file or storage pool volume to a virtual machine. The backing storage device can be any supported type of storage device, file, or storage pool volume.

The emulated IDE driver

KVM provides two emulated PCI IDE interfaces. An emulated IDE driver can be used to attach any combination of up to four virtualized IDE hard disks or virtualized IDE CD-ROM drives to each virtual machine. The emulated IDE driver is also used for virtualized CD-ROM and DVD-ROM drives.

The emulated floppy disk drive driver

The emulated floppy disk drive driver is used for creating virtualized floppy drives.

4.3.2. Para-virtualized devices

Para-virtualized devices are drivers for virtual devices that increase the I/O performance of virtual machines.

Para-virtualized devices decrease I/O latency and increase I/O throughput to near bare-metal levels. It is recommended to use the para-virtualized drivers for virtual machines running I/O intensive applications.

The para-virtualized devices must be installed on the guest operating system. By default, the para-virtualized drivers are included in Red Hat Enterprise Linux 4.7 and newer, Red Hat Enterprise Linux 5.4 and newer and Red Hat Enterprise Linux 6.0 and newer. The para-virtualized drivers must be manually installed on Windows guests.

Note

For more information on using the para-virtualized drivers, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

The para-virtualized network driver (virtio-net)

The para-virtualized network driver is a Red Hat branded virtual network device. It can be used as the driver for existing network devices or new network devices for virtual machines.

The para-virtualized block driver (virtio-blk)

The para-virtualized block driver is a driver for all storage devices, is supported by the hypervisor, and is attached to the virtual machine (except for floppy disk drives, which must be emulated).

The para-virtualized controller device (virtio-scsi)

The para-virtualized SCSI controller device is a new feature in Red Hat Enterprise Linux 6.4 that provides a more flexible and scalable alternative to virtio-blk. A virtio-scsi guest is capable of inheriting the feature set of the target device, and can handle hundreds of devices compared to virtio-blk, which can only handle 28 devices.

In Red Hat Enterprise Linux 6.4, virtio-scsi is fully supported for the following guest operating systems:

- Red Hat Enterprise Linux 6.4
- Windows Server 2003
- Windows Server 2008
- Windows 7
- Windows Server 2012
- Windows 8 (32/64 bit)

The para-virtualized clock

Guests using the Time Stamp Counter (TSC) as a clock source may suffer timing issues. KVM works around hosts that do not have a constant Time Stamp Counter by providing guests with a para-virtualized clock.

The para-virtualized serial driver (virtio-serial)

The para-virtualized serial driver is a bytestream-oriented, character stream driver, and provides a simple communication interface between the host's user space and the guest's user space.

The balloon driver (virtio-balloon)

The balloon driver can designate part of a virtual machine's RAM as not being used (a process known as balloon *inflation*), so that the memory can be freed for the host (or for other virtual machines on that host) to use. When the virtual machine needs the memory again, the balloon can be *deflated* and the host can distribute the RAM back to the virtual machine.

4.3.3. Physical host devices

Certain hardware platforms allow virtual machines to directly access various hardware devices and components. This process in virtualization is known as *device assignment*. Device assignment is also known as *passthrough*.

PCI device assignment

The KVM hypervisor supports attaching PCI devices on the host system to virtual machines. PCI device assignment allows guests to have exclusive access to PCI devices for a range of tasks. It allows PCI devices to appear and behave as if they were physically attached to the guest virtual machine.

Device assignment is supported on PCI Express devices, with the exception of graphics cards. Parallel PCI devices may be supported as assigned devices, but they have severe limitations due to security and system configuration conflicts.

Note

For more information on device assignment, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

USB passthrough

The KVM hypervisor supports attaching USB devices on the host system to virtual machines. USB device assignment allows guests to have exclusive access to USB devices for a range of tasks. It allows USB devices to appear and behave as if they were physically attached to the virtual machine.

Note

For more information on USB passthrough, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

SR-IOV

SR-IOV (Single Root I/O Virtualization) is a PCI Express standard that extends a single physical PCI function to share its PCI resources as separate, virtual functions (VFs). Each function is capable of being used by a different virtual machine via PCI device assignment.

An SR-IOV capable PCI-e device, provides a Single Root Function (for example, a single Ethernet port) and presents multiple, separate virtual devices as unique PCI device functions. Each virtual device may have its own unique PCI configuration space, memory-mapped registers, and individual MSI-based interrupts.

Note

For more information on SR-IOV, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

NPIV

N_Port ID Virtualization (NPIV) is a functionality available with some Fibre Channel devices. NPIV shares a single physical N_Port as multiple N_Port IDs. NPIV provides similar functionality for Fibre Channel Host Bus Adapters (HBAs) that SR-IOV provides for PCIe interfaces. With NPIV, virtual machines can be provided with a virtual Fibre Channel initiator to Storage Area Networks (SANs).

NPIV can provide high density virtualized environments with enterprise-level storage solutions.

Note

For more information on NPIV, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

4.3.4. Guest CPU models

Historically, CPU model definitions were hard-coded in **qemu**. This method of defining CPU models was inflexible, and made it difficult to create virtual CPUs with feature sets that matched existing physical CPUs. Typically, users modified a basic CPU model definition with feature flags in order to provide the CPU characteristics required by a virtual machine. Unless these feature sets were carefully controlled, safe migration — which requires feature sets between current and prospective hosts to match — was difficult to support.

qemu-kvm has now replaced most hard-wired definitions with configuration file based CPU model definitions. Definitions for a number of current processor models are now included by default, allowing users to specify features more accurately and migrate more safely.

Note

For more information on guest CPU models, refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide*.

4.4. Storage

Storage for virtual machines is abstracted from the physical storage used by the virtual machine. It is attached to the virtual machine using the para-virtualized or emulated block device drivers.

4.4.1. Storage pools

A *storage pool* is a file, directory, or storage device managed by **libvirt** for the purpose of providing storage to virtual machines. Storage pools are divided into storage *volumes* that store virtual machine images or are attached to virtual machines as additional storage. Multiple guests can share the same storage pool, allowing for better allocation of storage resources. Refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide* for more information.

Local storage pools

Local storage pools are directly attached to the host server. They include local directories, directly attached disks, physical partitions, and LVM volume groups on local devices. Local storage pools are useful for development, testing and small deployments that do not require

migration or large numbers of virtual machines. Local storage pools may not be suitable for many production environments as they do not support live migration.

Networked (shared) storage pools

Networked storage pools include storage devices shared over a network using standard protocols. Networked storage is required when migrating virtual machines between hosts with **virt-manager**, but is optional when migrating with **virsh**. Networked storage pools are managed by **libvirt**.

4.4.2. Storage Volumes

Storage pools are further divided into storage volumes. Storage volumes are an abstraction of physical partitions, LVM logical volumes, file-based disk images and other storage types handled by **libvirt**. Storage volumes are presented to virtual machines as local storage devices regardless of the underlying hardware.

Note

For more information on storage and virtualization, refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*.

Chapter 5. Virtualization Tools

This chapter provides an introduction to the many tools available to assist with virtualization.

5.1. `virsh`

virsh is a command line interface (CLI) tool for managing the hypervisor and guest virtual machines. The **`virsh`** command line tool is built on the **libvirt** management API and operates as an alternative to the **`qemu-kvm`** command and the graphical **`virt-manager`** application. The **`virsh`** command can be used in read-only mode by unprivileged users or, with root access, full administration functionality. The **`virsh`** command is ideal for scripting virtualization administration. In addition the **`virsh`** tool is a main management interface for **`virsh`** guest domains and can be used to create, pause, and shut down domains, as well as list current domains. This tool is installed as part of the *libvirt-client* package.

Note

Refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide* for more information about managing virtual machines with **`virsh`**.

5.2. `virt-manager`

virt-manager is a lightweight graphical tool for managing virtual machines. It provides the ability to control the life cycle of existing machines, provision new machines, manage virtual networks, access the graphical console of virtual machines, and view performance statistics. This tool ships in its own package called *virt-manager*.

Note

Refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide* for more information about managing virtual machines with **`virt-manager`**.

5.3. `virt-install`

virt-install is a command line tool to provision new virtual machines. It supports both text-based and graphical installations, using serial console, SDL, SPICE, or VNC client/server pair graphics. Installation media can be local, or exist remotely on an NFS, HTTP, or FTP server. The tool can also be configured to run unattended and kickstart the guest when installation is complete, allowing for easy automation of installation. This tool is installed as part of the *python-virtinst* package.

Note

Refer to the *Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide* for more information about **`virt-install`**.

5.4. `guestfish`

guestfish is a command line tool for examining and modifying the file systems of the host. This tool uses *libguestfs* and exposes all functionality provided by the **guestfs** API. This tool ships in its own package entitled *guestfish*.

Warning

Using **guestfish** on running virtual machines can cause disk-image corruption. Use the **guestfish** command with the **--ro** (read-only) option if the disk image is being used by a running virtual machine.

Note

Refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide* for more information about **guestfish**.

5.5. Other useful tools

The following tools are used to access a guest virtual machine's disk via the host. The guest's disk is usually accessed directly via the **disk-image** file located on the host. However it is sometimes possible to gain access via the **libvirt** domain. The commands that follow are part of the **libvirt** domain and are used to gain access to the guest's disk image.

guestmount

A command line tool used to mount virtual machine file systems and disk images on the host machine. This tool is installed as part of the *libguestfs-mount* package.

Warning

Using **guestmount** in **--r/w** (read/write) mode to access a disk that is currently being used by a guest can cause the disk to become corrupted. Do not use **guestmount** in **--r/w** (read/write) mode on live virtual machines. Use the **guestmount** command with the **--ro** (read-only) option if the disk image is being used.

virt-cat

A command line tool that can be used to quickly view the contents of one or more files in a specified virtual machine's disk or disk image. This tool is installed as part of the *libguestfs-tools* package.

virt-df

A command line tool used to show the actual physical disk usage of virtual machines, similar to the command line tool **df**. Note that this tool does not work across remote connections. It is installed as part of the *libguestfs-tools* package.

virt-edit

A command line tool used to edit files that exist on a specified virtual machine. This tool is

installed as part of the *libguestfs-tools* package.

Warning

Using **virt-edit** on live virtual machines can cause disk corruption in the virtual machine. Although the **virt-edit** command will try to prevent users from editing files on live virtual machines, it is not guaranteed to catch all instances. Do not use **virt-edit** on a live virtual machine.

virt-filesystems

A command line tool used to discover file systems, partitions, logical volumes and their sizes in a disk image or virtual machine. One common use is in shell scripts, to iterate over all file systems in a disk image. This tool is installed as part of the *libguestfs-tools* package.

This tool replaces **virt-list-filesystems** and **virt-list-partitions**.

virt-inspector

A command line tool that can examine a virtual machine or disk image to determine the version of its operating system and other information. It can also produce XML output, which can be piped into other programs. Note that **virt-inspector** can only inspect one domain at a time. This tool is installed as part of the *libguestfs-tools* package.

virt-inspector2

An alternative tool to **virt-inspector**, written in C. This tool is installed as part of the *libguestfs-tools* package.

virt-ls

A command line tool that lists files and directories inside a virtual machine. This tool is installed as part of the *libguestfs-tools* package.

virt-make-fs

A command line tool for creating a file system based on a tar archive or files in a directory. It is similar to tools like **mkisofs** and **mksquashfs**, but it can create common file system types such as ext2, ext3 and NTFS, and the size of the file system created can be equal to or greater than the size of the files it is based on. This tool is provided as part of the *libguestfs-tools* package.

virt-rescue

A command line tool that provides a rescue shell and some simple recovery tools for unbootable virtual machines and disk images. It can be run on any virtual machine known to **libvirt**, or directly on disk images. This tool is installed as part of the *libguestfs-tools* package.

Warning

Using **virt-rescue** on running virtual machines can cause disk corruption in the virtual machine. **virt-rescue** attempts to prevent its own use on running virtual machines, but cannot catch all cases. Using the command with the **--ro** (read-only) option will not cause disk corruption, but may give strange or inconsistent results.
Avoid using **virt-rescue** on a running virtual machine.

virt-resize

A command line tool to resize virtual machine disks, and resize or delete any partitions on a virtual machine disk. It works by copying the guest image and leaving the original disk image untouched. This tool is installed as part of the *libguestfs-tools* package.

Important

Using **virt-resize** on running virtual machines can give inconsistent results. It is best to shut down virtual machines before attempting to resize them.

virt-tar

A command line archive tool for downloading and uploading parts of a virtual machine's file system. This tool is commonly used for making backups, uploading data, reviewing guest activity, and fixing or customizing guests. This tool is installed as part of the *libguestfs-tools* package.

Warning

Using **virt-tar** with the **-u** (upload) option on running virtual machines can cause disk corruption in the virtual machine. **virt-tar** attempts to prevent its own use on running virtual machines, but cannot catch all cases.
Using **virt-tar** with the **-x** (extract) option on running virtual machines will not cause disk corruption, but may give strange or inconsistent results. It is best to shut down virtual machines before attempting to extract files from them.

virt-top

A command line utility similar to **top**, which shows statistics related to virtualized domains. This tool ships in its own package: *virt-top*.

virt-v2v

A graphical tool to convert virtual machines from Xen and VMware hypervisors to run on KVM. This tool ships in its own package: *virt-v2v*.

virt-viewer

A minimal tool for displaying the graphical console of a virtual machine via the VNC and SPICE

protocols. This tool ships in its own package: *virt-viewer*.

`virt-what`

A shell script that detects whether a program is running in a virtual machine. This tool ships in its own package: *virt-what*.

`virt-who`

The *virt-who* package is a Red Hat Enterprise Linux host agent that queries **libvirt** for guest UUIDs. It then passes that data to the local entitlement server for the purposes of issuing certificates. This tool ships in its own package: *virt-who*.

`virt-win-reg`

A command line tool to export and merge Windows Registry entries from a Windows virtual machine, and perform simple Registry operations. This tool is installed as part of the *libguestfs-tools* package.

Warning

Using **`virt-win-reg`** on running virtual machines will cause irreversible disk corruption in the virtual machine. **`virt-win-reg`** attempts to prevent its own use on running virtual machines, but cannot catch all cases.

Warning

Modifying the Windows Registry is an inherently risky operation, as the format is deliberately obscure and undocumented. Changes to the registry can leave the system unbootable, so ensure you have a reliable backup before you use the **`--merge`** option.

`virt-xml-validate`

A command line tool to validate **libvirt** XML files for compliance with the published schema. This tool is installed as part of the *libvirt-client* package.

Revision History

Revision 0.3-24 **Mon Feb 18 2013** **Dayle Parker**
Version for 6.4 GA release.

Revision 0.3-20 **Thurs Jan 31 2013** **Dayle Parker**
Updated 4.4.1 Storage pools.

Revision 0.3-18 **Tue Jan 29 2013** **Dayle Parker**
Updated virtio-scsi support details in 4.3.2. Para-virtualized devices for BZ#903891.

Revision 0.3-16 **Wed Jan 16 2013** **Dayle Parker**
Minor grammatical edits in virtualization products chapter.

Revision 0.3-15 **Tues Nov 27 2012** **Dayle Parker**
Minor grammatical edits throughout book.

Revision 0.3-13 **Thu Oct 18 2012** **Dayle Parker**
Applied SME feedback to RHEV introduction for BZ#798104.
Revised virtualization guide descriptions in introduction.

Revision 0.3-11 **Wed Oct 17 2012** **Dayle Parker**
Made corrections, revised RHEV section in 1.3, added virtualization benchmarks in 3.3 for BZ#798104.
Added virtualization deployment scenarios for BZ#847924.

Revision 0.3-10 **Mon Oct 8 2012** **Dayle Parker**
Revised 1.3 RHEV description in introduction for BZ#798104.

Revision 0.3-8 **Thurs Oct 4 2012** **Dayle Parker**
Added tech preview note about virtio-scsi in Ch.4 from SME review.
Added QEMU Guest Agent description.
Moved configuration-specific part of 4.3.4. Guest CPU Models to Virtualization Host Configuration and Guest Installation Guide for BZ#842970.

Revision 0.3-7 **Wed Oct 3 2012** **Dayle Parker**
Applied peer feedback to Virtualization in Red Hat Enterprise Linux introduction.
Added virtio-scsi description for BZ#847167.

Revision 0.3-6 **Tue Sep 25 2012** **Dayle Parker**
Added Section 1.2: Virtualization in Red Hat Enterprise Linux 6.

Revision 0.3-5 **Mon Sep 3 2012** **Dayle Parker**
Added reference to Virt Security Guide in Ch.3 - BZ#838009.

Revision 0.3-2 **Wed Aug 29 2012** **Dayle Parker**
Corrected links from http://docs.redhat.com/ to new location: http://access.redhat.com/knowledge/docs/ .
Corrected migration details in sections: 2.2. Migration and 4.4.1. Storage pools for BZ#831901.

Revision 0.3-1 **Mon Aug 27 2012** **Dayle Parker**

Placed references to other guides in admonitions; Corrected terms for BZ#813620.

Revision 0.2-83 **Mon June 18 2012** **Dayle Parker**
Version for 6.3 GA release.

Revision 0.2-82 **Mon June 18 2012** **Dayle Parker**
Corrected "64 vCPUs" to "160 vCPUs" for BZ#832415.

Revision 0.2-80 **Tues June 12 2012** **Dayle Parker**
Clarified emulated watchdog device section for BZ#827307.

Revision 0.2-78 **Fri June 8 2012** **Dayle Parker**
Corrected typos and markup for BZ#827305.
General corrections made to Chapter 4 BZ#827307.

Revision 0.2-73 **Mon April 23 2012** **Laura Novich**
Corrections made to chapter 5 (BZ#798108).

Revision 0.2-72 **Mon April 23 2012** **Laura Novich**
Corrections made to chapter 4 (BZ#798106).

Revision 0.2-71 **Thur April 19 2012** **Laura Novich**
Corrections made to chapter 5 (BZ#798108).

Revision 0.2-69 **Wed April 18 2012** **Laura Novich**
Corrections made to chapter 4 (BZ#798106).

Revision 0.2-68 **Tue April 17 2012** **Dayle Parker**
Corrected terminology to "virtual machine" where needed (BZ#798063).

Revision 0.2-64 **Mon April 2 2012** **Laura Novich**
Corrections to Chapter 2 (BZ#800401).

Revision 0.2-61 **Fri March 30 2012** **Dayle Parker**
Made corrections in Chapter 3: Advantages (BZ#800409).
Adjusted terms to "virtual machine" and "virtualized guest" where appropriate; corrected outdated link in 4.1; corrected terms in Emulated network devices (from drivers) in (BZ#798063).

Revision 0.2-52 **Wed January 11 2012** **Jacquelynn East**
BZ#772859 clarified acronym.

Revision 0.2-51 **Fri November 4 2011** **Jacquelynn East**
BZ#750969 minor typos.

Revision 0.2-47 **Fri October 14 2011** **Jacquelynn East**
BZ#744156 added paragraph about emulated watchdogs.

Revision 0.2-45 **Sun September 18 2011** **Scott Radvan**
Minor wording issues.

Revision 0.2-44 **Fri September 16 2011** **Jacquelynn East**
BZ#734614

Revision 0.2-43 **Fri September 16 2011** **Jacquelynn East**
BZ#734618 minor edit.

Revision 0.2-37 **Fri September 2 2011** **Jacquelynn East**
BZ#734619, BZ#734614

Revision 0.2-34 **Thu September 1 2011** **Jacquelynn East**
BZ#734619, BZ#734511, BZ#734618, BZ#734616, BZ#715476, BZ#734613

Revision 0.2-33 **Wed August 31 2011** **Jacquelynn East**
BZ#734618, BZ#734613, BZ#734619

Revision 0.2-32 **Thu August 25 2011** **Scott Radvan**
6.2 development.

Revision 0.2-24 **Fri July 29 2011** **Jacquelynn East**
Extensive edits, combined security section into advantages.

Revision 0.2-22 **Wed July 27 2011** **Jacquelynn East**
Advantages chapter completed (BZ#715476).

Revision 0.2-20 **Tue July 26 2011** **Jacquelynn East**
More of the Advantages draft.

Revision 0.2-17 **Mon July 25 2011** **Jacquelynn East**
Minor edits for BZ#715473 and BZ#715474.

Revision 0.2-15 **Mon July 25 2011** **Jacquelynn East**
Chapter 4 draft BZ#715476.

Revision 0.2-4 **Thu June 23 2011** **Jacquelynn East**
Completed chapter 1.

Revision 0.1-1 **Wed May 4 2011** **Scott Radvan**
Arranged basic layout and book infrastructure. Imported introductory text.

Revision 0.0-1 **Wed May 4 2011** **Scott Radvan**
Initial creation of book by Publican.