Red Hat Enterprise Linux 7 Virtualization Getting Started Guide

An introduction to virtualization concepts

Dayle Parker
Scott Radvan

Laura Novich

Jacquelynn East

Red Hat Enterprise Linux 7 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

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

Chapter 1. What is virtualization and migration?

This chapter discusses terms related to virtualization and migration.

1.1. 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 several methods for virtualizing operating systems.

Virtualization methods

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 7 Virtualization Deployment and Administration Guide*.

1.2. What is 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 or shuts down 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.

It is important to understand that the migration process moves the virtual machine's memory, and the disk volume associated with the virtual machine is also migrated. This process is done using live block migration — information about this can be found in the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

1.2.1. 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.

Chapter 2. 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.

2.1. 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.

2.2. 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.

2.3. 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; modern virtualization technology has greatly improved the speed of virtual machines. Benchmarks show that virtual machines can run typical server applications nearly as efficiently as 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

2.4. 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.

2.5. Security

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

2.5.1. 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 7.

sVirt

sVirt is a technology included in Red Hat Enterprise Linux 7 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 in virtualization, refer to the *Red Hat Enterprise Linux 7 Virtualization Security Guide*.

2.6. 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.

2.6.1. 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 3. Introduction to Red Hat virtualization products

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

3.1. 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 7 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 7 Virtualization Deployment and 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 7 Virtualization Deployment and Administration Guide*.

KSM

Kernel Same-page 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 7 Virtualization Deployment and 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 7 Virtualization Deployment and Administration Guide*.

Hyper-V Enlightenment

KVM in Red Hat Enterprise Linux 7 implements several Hyper-V compatible functions that are used by Windows guests to improve performance and stability, allowing Windows guests to perform as if they were running on a Microsoft Hyper-V hypervisor.

Note

For more information on Hyper-V functionality, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

Disk I/O throttling

When several virtual machines are running simultaneously, they can interfere with system performance by using excessive disk I/O. *Disk I/O throttling* in KVM provides the ability to set a limit on disk I/O requests sent from virtual machines to the host machine. This can prevent a virtual machine from over utilizing shared resources, and impacting the performance of other virtual machines.

Note

For instructions on using disk I/O throttling, refer to the *Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide*.

Automatic NUMA balancing

Automatic NUMA balancing improves the performance of applications running on NUMA hardware systems, without any manual tuning required for Red Hat Enterprise Linux 7 guests. Automatic NUMA balancing moves tasks, which can be threads or processes, closer to the memory they are accessing.

Note

For more information on automatic NUMA balancing, refer to the *Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide*.

Virtual CPU hot add

Virtual CPU (vCPU) hot add capability provides the ability to increase processing power as needed on running virtual machines, without downtime. The vCPUs assigned to a virtual machine can be added to a running guest in order to either meet the workload's demands, or to maintain the Service Level Agreement (SLA) associated with the workload.

Note

For more information on virtual CPU hot add, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

KVM guest virtual machine compatibility

Red Hat Enterprise Linux 7 servers have certain support limits.

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

- For host systems: https://access.redhat.com/site/articles/rhel-limits
- For the KVM hypervisor: https://access.redhat.com/site/articles/rhel-kvm-limits

For a complete chart of supported operating systems and host and guest combinations, refer to https://access.redhat.com/site/supported-hypervisors.

Note

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 7 Virtualization Deployment and Administration Guide*.

3.2. 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 7 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 Red Hat Enterprise Linux 5, Red Hat Enterprise Linux 6, and Red Hat Enterprise Linux 7.

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 and provides many functions such as installing, listing, starting, and stopping virtual machines.

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 7 Virtualization Deployment and Administration Guide*.

3.3. Virtualized hardware devices

Virtualization on Red Hat Enterprise Linux 7 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.

3.3.1. 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
- USB 3.0 xHCI host controller (Technology Preview in Red Hat Enterprise Linux 7.0)
- Q35 host PCI bridge (Technology Preview in Red Hat Enterprise Linux 7.0)

  Note, the Q35 host PCI bridge should be used only with the emulated AHCI controller, also available as a Technology Preview in Red Hat Enterprise Linux 7.0.
- PCI Express bus (Technology Preview in Red Hat Enterprise Linux 7.0)

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 7, for the x86_64 architecture
- 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 emulated sound device is also available, but is not recommended due to compatibility issues with certain guest operating systems:

- **`ac97`**, an emulated Intel 82801AA AC97 Audio compatible 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 drivers 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.

The emulated AHCI controller

The emulated Advanced Host Controller Interface (AHCI) bus is an alternative to IDE. It is available as a Technology Preview in Red Hat Enterprise Linux 7.0, and should be used only with the Q35 host PCI bridge (also Technology Preview).

Note

For more information on KVM storage features, refer to Section 3.4, “Storage”.

3.3.2. Para-virtualized devices

Para-virtualization provides a fast and efficient means of communication for guests to use devices on the host machine. KVM provides para-virtualized devices to virtual machines using the Virtio API as a layer between the hypervisor and guest.

Some para-virtualized devices decrease I/O latency and increase I/O throughput to near bare-metal levels, while other para-virtualized devices add functionality to virtual machines that is not otherwise available. It is recommended to use para-virtualized devices instead of emulated devices for virtual machines running I/O intensive applications.

All virtio devices have two parts: the host device and the guest driver. Para-virtualized device drivers allow the guest operating system access to physical devices on the host system.

The para-virtualized device drivers must be installed on the guest operating system. By default, the para-virtualized device 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 device drivers must be manually installed on Windows guests.

Note

For more information on using the para-virtualized devices and drivers, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

The para-virtualized network device (virtio-net)

The para-virtualized network device is a virtual network device that provides network access to virtual machines with increased I/O performance and lower latency.

The para-virtualized block device (virtio-blk)

The para-virtualized block device is a high-performance virtual storage device that provides storage to virtual machines with increased I/O performance and lower latency. The para-virtualized block device 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 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.

virtio-scsi is fully supported for the following guest operating systems:

- Red Hat Enterprise Linux 7
- Red Hat Enterprise Linux 6.4 and above
- Windows Server 2008 (32/64 bit)
- Windows Server 2008 R2
- Windows 7 (32/64 bit)
- Windows Server 2012
- Windows Server 2012 R2
- Windows 8 (32/64 bit)
- Windows 8.1 (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. Additionally, the para-virtualized clock assists with time adjustments needed after a guest runs S3 or suspend to RAM operations. The para-virtualized clock is not supported in Windows guests.

The para-virtualized serial device (virtio-serial)

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

The balloon device (virtio-balloon)

The balloon device 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.

The para-virtualized random number generator (virtio-rng)

The para-virtualized random number generator enables virtual machines to collect entropy, or randomness, directly from the host to use for encrypted data and security. Virtual machines can often be starved of entropy because typical inputs (such as hardware usage) are unavailable. Sourcing entropy can be time-consuming; virtio-rng makes this process faster by injecting entropy directly into guest virtual machines from the host.

The para-virtualized graphics card (QXL)

The para-virtualized graphics card works with the QXL driver to provide an efficient way to display a virtual machine's graphics from a remote host. The QXL driver is required to use SPICE. It is downloaded as part of the Windows driver package, *virtio-win*.

3.3.3. 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*.

VFIO device assignment

Virtual Function I/O (VFIO) is a new kernel driver in Red Hat Enterprise Linux 7 that provides virtual machines with high performance access to physical hardware.

VFIO attaches PCI devices on the host system directly to virtual machines, allowing guests exclusive access to PCI devices for a range of tasks. This allows PCI devices to appear and behave as if they were physically attached to the guest virtual machine.

VFIO improves on previous PCI device assignment architecture by moving device assignment out of the KVM hypervisor, and enforcing device isolation at the kernel level. VFIO offers better security and is compatible with secure boot. It is the default device assignment mechanism in Red Hat Enterprise Linux 7.

VFIO increases the number of assigned devices to 32 in Red Hat Enterprise Linux 7, from a maximum 8 devices in Red Hat Enterprise Linux 6. VFIO also supports assignment of NVIDIA GPUs.

Note

For more information on VFIO device assignment, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration 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 7 Virtualization Deployment and 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 7 Virtualization Deployment and Administration 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.

3.3.4. Guest CPU models

CPU models define which host CPU features are exposed to the guest operating system. **qemu-kvm** and **libvirt** contain definitions for several current processor models, allowing users to enable CPU features that are available only in newer CPU models. The set of CPU features that can be exposed to guests depends on support in the host CPU, the kernel, and **qemu-kvm** code.

To allow safe migration of virtual machines between hosts with different sets of CPU features, **qemu-kvm** does not expose all features from the host CPU to guest operating systems by default. Instead, CPU features are exposed based on the chosen CPU model. If a virtual machine has a given CPU feature enabled, it is not possible to migrate it to a host that does not support exposing that feature to guests.

Note

For more information on guest CPU models, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

3.4. 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.

3.4.1. 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 7 Virtualization Deployment and 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**.

3.4.2. 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.

3.4.3. Emulated storage devices

Virtual machines can be presented with a range of storage devices that are emulated by the host. Each type of storage device is appropriate for specific use cases. Choice between different types of storage devices allows for maximum flexibility and compatibility with guest operating systems.

virtio-scsi

virtio-scsi is the recommended para-virtualized device for guests using large numbers of disks or advanced storage features such as TRIM. Guest driver installation may be necessary on guests using operating systems other than Red Hat Enterprise Linux 7.

virtio-blk

virtio-blk is a para-virtualized storage device suitable for exposing image files to guests. virtio-blk can provide the best disk I/O performance for virtual machines, but has fewer features than virtio-scsi.

IDE

IDE is recommended for legacy guests that do not support virtio drivers. IDE performance is lower than virtio-scsi or virtio-blk, but it is widely compatible with different systems.

CD-ROM

ATAPI CD-ROMs and virtio-scsi CD-ROMs are available for presenting ISO files or the host CD-ROM drive to guests. virtio-scsi CD-ROMs can be used with guests that have the virtio-scsi driver installed. ATAPI CD-ROMs offer wider compatibility but lower performance.

USB mass storage devices and floppy disks

USB mass storage devices and floppy disks are available when removable media are required. USB mass storage devices are preferable to floppy disks due to their larger capacity.

AHCI

The emulated Advanced Host Controller Interface (AHCI) bus is an alternative to IDE which provides increased features and performance, including communication with Serial ATA (SATA) devices.

AHCI is available as a Technology Preview in Red Hat Enterprise Linux 7.0. It should be used only with the Q35 host PCI bridge (also Technology Preview in Red Hat Enterprise Linux 7.0).

3.4.4. Host storage

Disk images can be stored on a range of local and remote storage technologies connected to the host.

Image files

Image files are stored on a host file system. The image files can be stored on a local file system, such as ext4 or xfs, or a network file system, such as NFS.

Tools such as **libguestfs** can manage, back up, and monitor files. Disk image formats on KVM include:

raw

Raw image files contain the contents of the disk with no additional metadata.

Raw files can either be pre-allocated or sparse, if the host file system allows it. Sparse files allocate host disk space on demand, and are therefore a form of thin provisioning. Pre-allocated files are fully provisioned but have higher performance than sparse files.

Raw files are desirable when disk I/O performance is critical and transferring the image file over a network is rarely necessary.

qcow2

qcow2 image files offer a number of advanced disk image features including backing files, snapshots, compression, and encryption. They can be used to instantiate virtual machines from template images.

qcow2 files are typically more efficient to transfer over a network, because only sectors written by the virtual machine are allocated in the image.

LVM volumes

Logical volumes can be used for disk images and managed using the system's LVM tools. LVM offers higher performance than file systems because of its simpler block storage model.

LVM thin provisioning offers snapshots and efficient space usage for LVM volumes, and can be used as an alternative to migrating to qcow2.

Host devices

Host devices such as physical CD-ROMs and raw disks or LUNs can be presented to the guest. This allows SAN or iSCSI LUNs as well as local CD-ROM media to be used by the guest with good performance.

Host devices can be used when storage management is done on a SAN instead of on hosts.

Distributed storage systems

Gluster volumes can be used as disk images. This allows high-performance clustered storage over the network.

Red Hat Enterprise Linux 7 includes native support for disk images on GlusterFS. This enables a KVM host to boot virtual machine images from GlusterFS volumes, and to use images from a GlusterFS volume as data disks for virtual machines. When compared to GlusterFS FUSE, the native support in KVM delivers higher performance.

Note

For more information on storage and virtualization, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

Chapter 4. Virtualization Tools

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

4.1. `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 administrative 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 7 Virtualization Deployment and Administration Guide* for more information about managing virtual machines with **`virsh`**.

4.2. `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 7 Virtualization Deployment and Administration Guide* for more information about managing virtual machines with **`virt-manager`**.

4.3. `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, 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 *virt-install* package.

Note

Refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide* for more information about using **`virt-install`**.

4.4. `guestfish`

guestfish is a command line tool for examining and modifying the file systems of the guest. This tool uses *libguestfs* and exposes all functionality provided by the **`guestfs`** API. This tool ships in its own package called *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 7 Virtualization Deployment and Administration Guide* for more information about **`guestfish`**.

4.5. GNOME Boxes

Boxes is a lightweight graphical desktop virtualization tool used to view and access virtual machines and remote systems. Boxes provides a way to test different operating systems and applications from the desktop with minimal configuration. Virtual systems can be installed manually or with the express installation function, which automatically preconfigures virtual machines with optimal settings. This tool ships in its own package called *gnome-boxes*.

Note

Refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide* for more information about using **GNOME Boxes**.

4.6. 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-tools* package.

Warning

Using **`guestmount`** in **`--rw`** (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 **`--rw`** (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-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-sysprep`

A command line tool to reset, customize, or unconfigure virtual machines to prepare a template for creating clones. This tool is installed as part of the *libguestfs-tools* package.

Important

Virtual machines must be shut down before running **`virt-sysprep`**. To preserve a virtual machine's existing contents, snapshot, copy or clone the disk before running **`virt-sysprep`**.

`virt-tar-in`

A command line archive tool for unpacking an uncompressed tarball into a virtual machine disk image or specified libvirt domain. This tool is installed as part of the *libguestfs-tools* package.

Warning

Using **`virt-tar-in`** command on a live virtual machine can cause disk corruption in the virtual machine. The virtual machine must be shut down before using this command.

`virt-tar-out`

A command line archive tool for packing a virtual machine disk image directory into a tarball. This tool is installed as part of the *libguestfs-tools* package.

`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-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.

Chapter 5. Quick Start Guide to Virtualization in Red Hat Enterprise Linux 7

This chapter enables you to try virtualization in Red Hat Enterprise Linux 7.

This chapter begins with an outline of the minimum system specifications and required packages to use virtualization. After these basic packages are installed, follow the procedure in Section 5.2.2, "Creating a virtual machine with Virtual Machine Manager" to create a basic virtual machine on **Virtual Machine Manager**.

Note

This tutorial uses Virtual Machine Manager to quickly create a virtual machine to trial KVM virtualization. To set up virtual machines with the capabilities necessary for a production environment, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide* for detailed information about system requirements and recommendations for running virtual machines.

5.1. Basic requirements and setup

Virtualization is available for Red Hat Enterprise Linux 7 on the Intel 64 and AMD64 architecture. The KVM hypervisor is provided with Red Hat Enterprise Linux 7.

Before you can set up a virtual machine on your Red Hat Enterprise Linux 7 system, your system must have certain packages installed.

System requirements

For virtualization to work on your system, you must have at minimum:

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

Required packages for virtualization

Before you can use virtualization, the virtualization packages must be installed on your computer. The virtualization packages can be installed either during the host installation sequence, or after host installation using the **`yum`** command and the Red Hat Customer Portal. This section describes the steps for installing the KVM hypervisor on a working Red Hat Enterprise Linux 7 system.

To install the packages, your host machine must be registered. To register via Red Hat Subscription Manager, run the **`subscription-manager register`** command and follow the prompts.

If you do not have a valid Red Hat subscription, visit the Red Hat online store to obtain one.

Procedure 5.1. Installing the virtualization packages with yum

To use virtualization on Red Hat Enterprise Linux you require at minimum the **`qemu-kvm`** and **`qemu-img`** packages. These packages provide the user-level KVM emulator and disk image manager on the host Red Hat Enterprise Linux system.

1. Install the *qemu-kvm* and *qemu-img* packages with the following command:

   ```
   # yum install qemu-kvm qemu-img
   ```

2. Install the *virt-manager* package with the following command:

   ```
   # yum install virt-manager
   ```

3. Additionally, for this tutorial, download a Red Hat Enterprise Linux 7 ISO image to your system to create the guest virtual machine's operating system.

   Red Hat Enterprise Linux ISO files can be found on the Red Hat Customer Portal at https://rhn.redhat.com/rhn/software/downloads/SupportedISOs.do.

5.2. Deploying a virtual machine with Virtual Machine Manager

The **Virtual Machine Manager**, also known as **virt-manager**, is a graphical tool for quickly deploying virtual machines in Red Hat Enterprise Linux. In this tutorial, you will become familiar with its basic functions, and will be able to use **Virtual Machine Manager** to create a virtual machine.

5.2.1. Introduction to Virtual Machine Manager

Open the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu.

The following image shows the **Virtual Machine Manager** interface. This interface allows you to control all of your virtual machines from one central location:

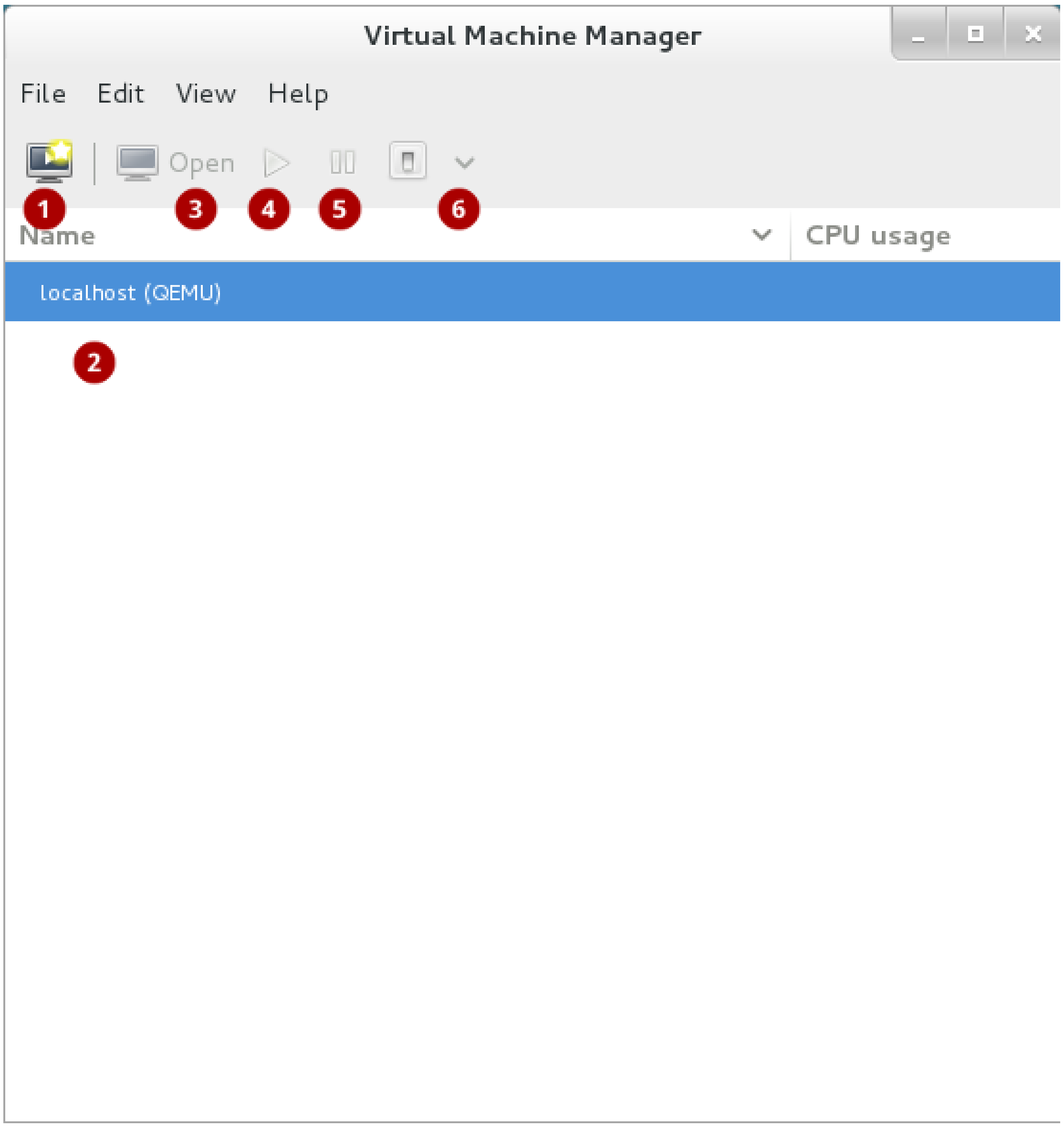


Figure 5.1. The Virtual Machine Manager interface

Commonly-used functions include:

- (1) **`Create new virtual machine`**: Start here to create a new virtual machine.
- (2) **`Virtual machines`**: A list of all virtual machines, or guests. When a virtual machine is created, it will be listed here. When a guest is running, an animated graph shows the guest's CPU usage under **`CPU usage`**.

  After selecting a virtual machine from this list, use the following buttons to control the selected virtual machine's state:

  - (3) **`Open`**: Opens the guest virtual machine console and details in a new window.
  - (4) **`Run`**: Turns on the virtual machine.

- 5 **`Pause`**: Pauses the virtual machine.
- 6 **`Shut down`**: Shuts down the virtual machine. Clicking on the arrow displays a dropdown menu with several options for turning off the virtual machine, including Reboot, Shut Down, Force Reset, Force Off, and Save.

Right-clicking on a virtual machine shows a menu with more functions, including:

- **`Clone`**: Clones the virtual machine.
- **`Migrate`**: Migrates the virtual machine to another host.
- **`Delete`**: Deletes the virtual machine.

5.2.2. Creating a virtual machine with Virtual Machine Manager

Follow these steps to create a Red Hat Enterprise Linux 7 virtual machine on **Virtual Machine Manager**.

Procedure 5.2. Creating a guest virtual machine with Virtual Machine Manager

1. **Open Virtual Machine Manager**

   Launch the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu.

2. **Create a new virtual machine**

   Click the **`Create a new virtual machine`** button (Figure 5.2, “The Create a new virtual machine button”) to open the **New VM** wizard.

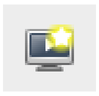
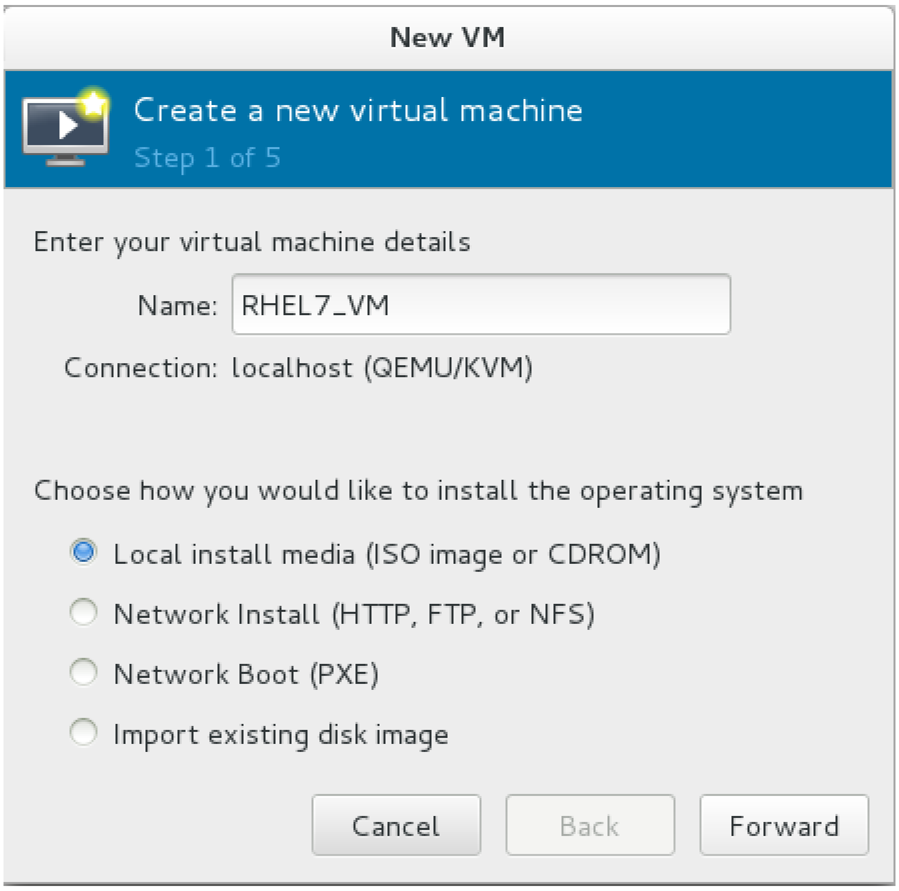
   

   Figure 5.2. The Create a new virtual machine button

3. **Specify name and installation method**

   Type in a virtual machine name and choose an installation type to install the guest virtual machine's operating system.

   The virtual machine creation process starts with the selection of a name and installation method. Virtual machine names can have underscores (_), periods (`.`), and hyphens (`-`).



Enter your virtual machine details

Name: RHEL7_VM

Connection: localhost (QEMU/KVM)

Choose how you would like to install the operating system

● Local install media (ISO image or CDROM)

○ Network Install (HTTP, FTP, or NFS)

○ Network Boot (PXE)

○ Import existing disk image

Figure 5.3. Name virtual machine and select installation method

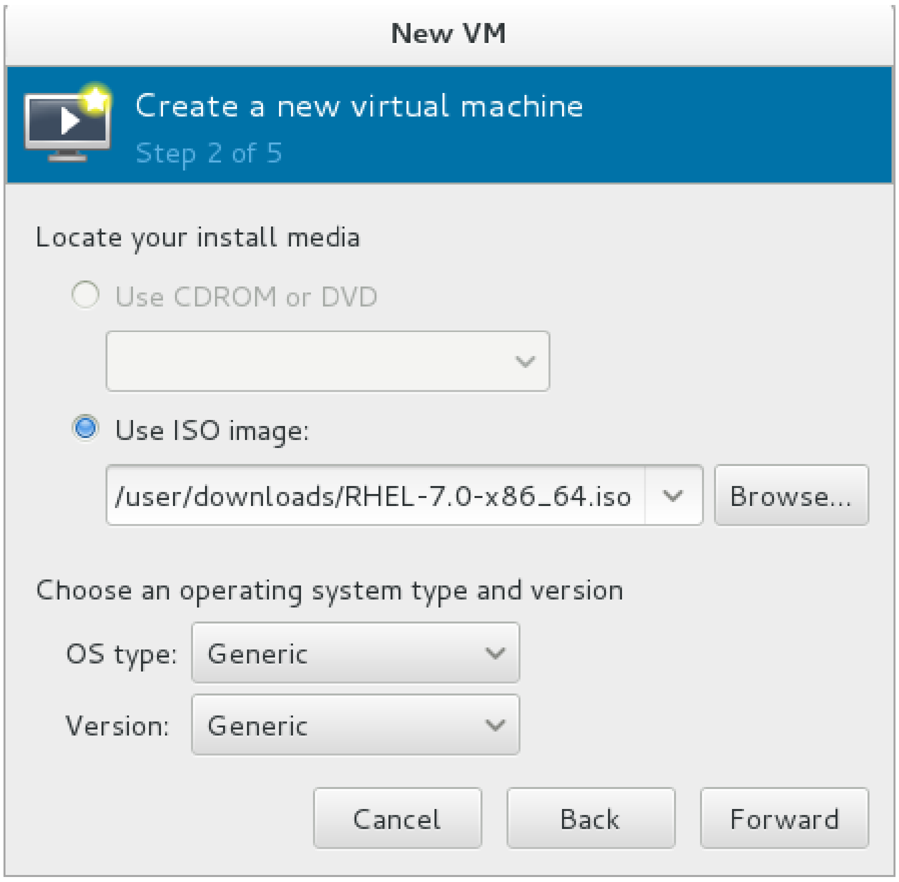
For this tutorial, select **`Local install media (ISO image)`**. This installation method uses an image of an installation disk (for example, an **`.iso`** file). Click the **`Forward`** button to continue.

4. **Locate installation media**

   Provide the location of the ISO that the guest virtual machine will install the operating system from.

   For this example, locate the ISO downloaded in Procedure 5.1, “Installing the virtualization packages with **`yum`**” on your machine by using the **`Browse`** to provide the existing storage path.

   Select the **OS type** and **Version** of the installation, ensuring that you select the appropriate OS type for your virtual machine. For this example, select **`Linux`** from the dropdown in **OS type** and choose **`Red Hat Enterprise Linux 7`** from the dropdown in **Version**, and click the **`Forward`** button.



Locate your install media

( ) Use CDROM or DVD

(●) Use ISO image: /user/downloads/RHEL-7.0-x86_64.iso

Choose an operating system type and version

OS type: Generic

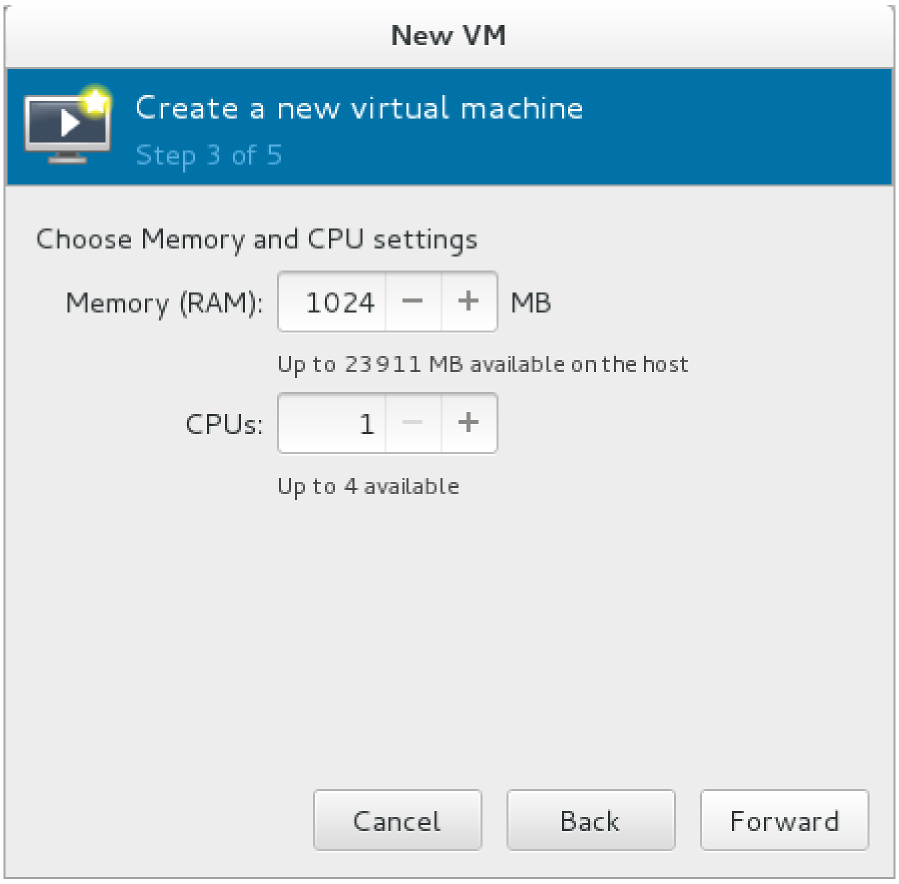
Version: Generic

Figure 5.4. Local ISO image installation

5. **Configure memory and CPU**

   Select the amount of memory and the number of CPUs to allocate to the virtual machine. The wizard shows the number of CPUs and amount of memory available to allocate.

   For this tutorial, select the default settings and click the **`Forward`** button.



Choose Memory and CPU settings

Memory (RAM): 1024 MB

Up to 23911 MB available on the host

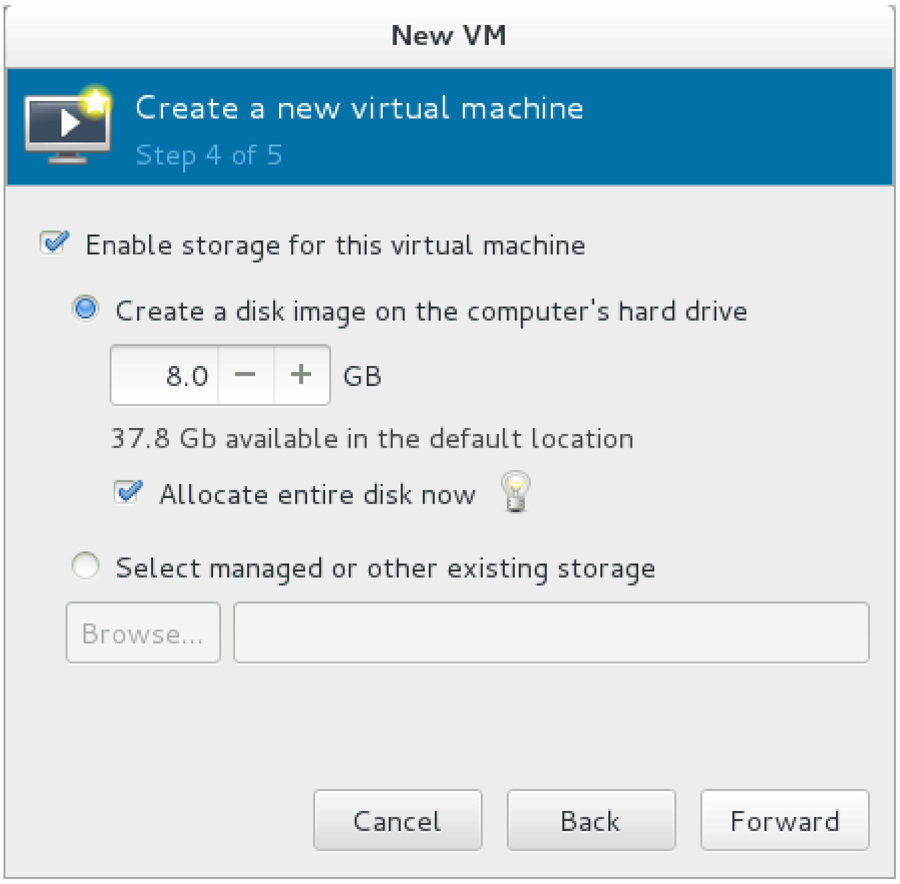
CPUs: 1

Up to 4 available

Figure 5.5. Configuring CPU and Memory

6. **Configure storage**

   Assign storage to the guest virtual machine. The wizard shows options for storage, including where to store the virtual machine on the host machine. For this tutorial, select the default settings and click the **`Forward`** button.



New VM

Create a new virtual machine

Step 4 of 5

☑ Enable storage for this virtual machine

● Create a disk image on the computer's hard drive

8.0 GB

37.8 Gb available in the default location

☑ Allocate entire disk now

○ Select managed or other existing storage

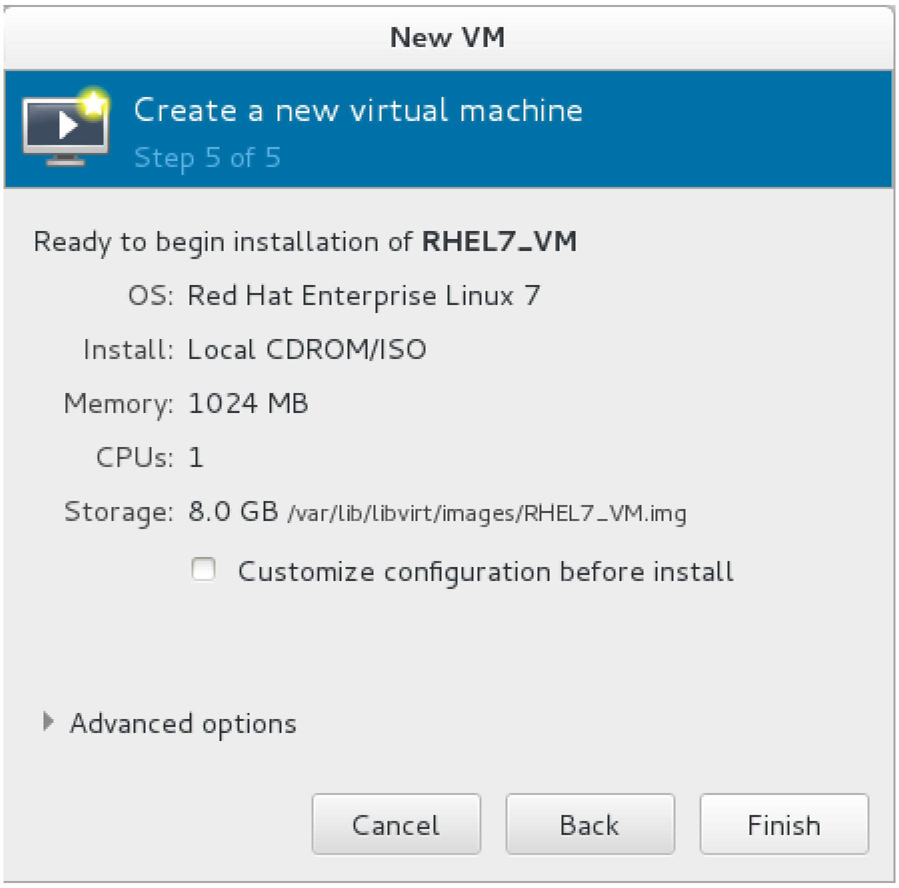
Browse...

Cancel | Back | Forward

Figure 5.6. Configuring storage

7. **Final configuration**

   Verify the settings of the virtual machine and click the **`Finish`** button. **Virtual Machine Manager** will then create a virtual machine with your specified hardware settings.


New VM

Create a new virtual machine
Step 5 of 5

Ready to begin installation of **RHEL7_VM**

OS: Red Hat Enterprise Linux 7

Install: Local CDROM/ISO

Memory: 1024 MB

CPUs: 1

Storage: 8.0 GB /var/lib/libvirt/images/RHEL7_VM.img

☐ Customize configuration before install

Figure 5.7. Verifying the configuration

After **Virtual Machine Manager** has created your Red Hat Enterprise Linux 7 virtual machine, follow the instructions in the Red Hat Enterprise Linux 7 installer to complete the installation of the virtual machine's operating system.

Note

For help with Red Hat Enterprise Linux 7 installation, refer to the *Red Hat Enterprise Linux 7 Installation Guide*.

5.2.3. Exploring the guest virtual machine

You can view a virtual machine's console by selecting a virtual machine in the **Virtual Machine Manager** window and clicking **Open**. You can operate your Red Hat Enterprise Linux 7 virtual machine from the console in the same way as a physical system.

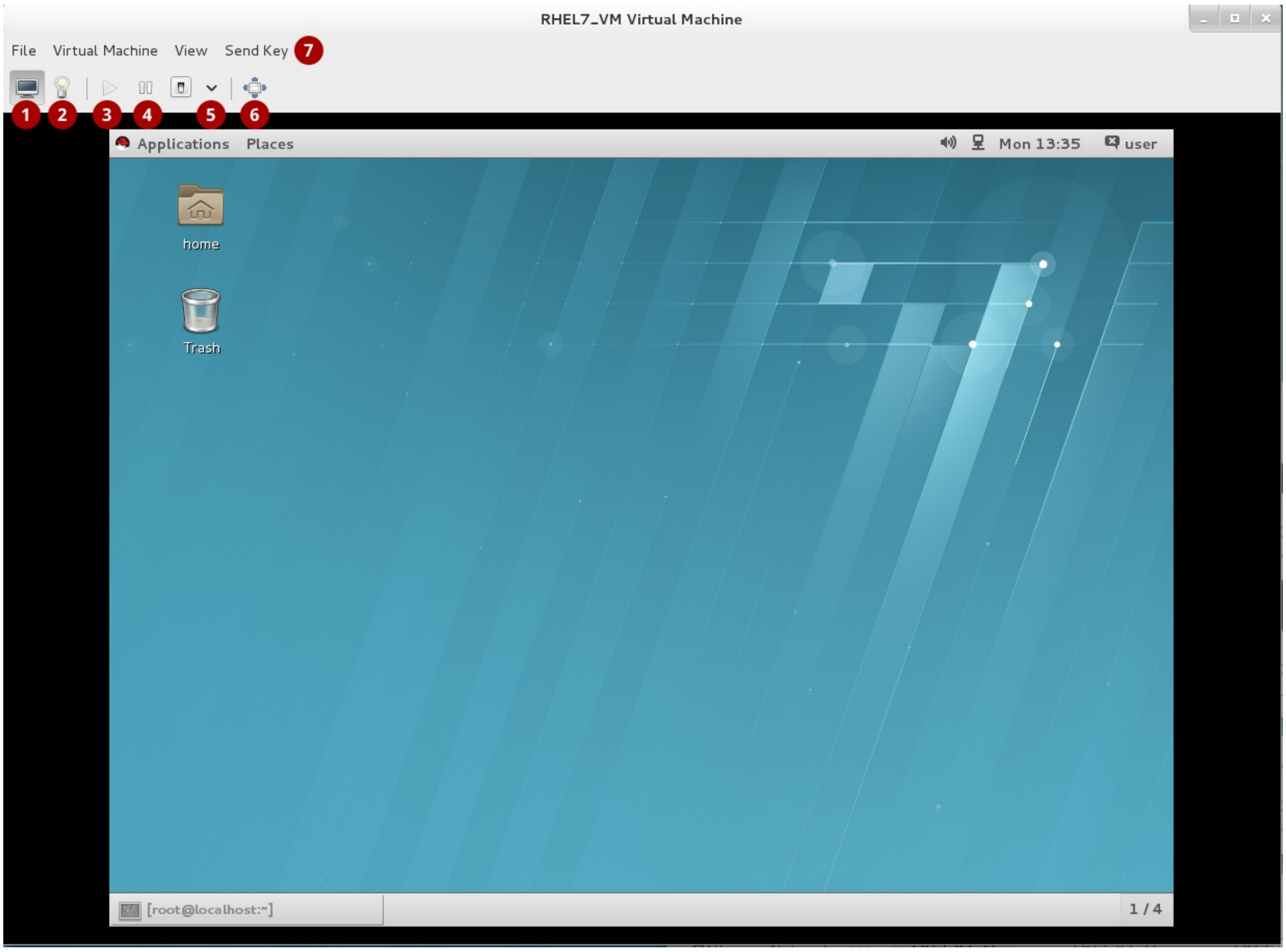


Figure 5.8. The guest virtual machine console

- 1 **`Show the graphical console`**: This shows what is on the virtual machine's screen. The virtual machine can be operated the same as a physical machine from the console.

- 2 **`Show virtual hardware details`**: This window shows details of the virtual hardware that the guest is using. It provides an overview of basic system details, performance, processor, memory, and boot settings, plus details of the system's virtual devices.

- These buttons control the virtual machine's state:

  - 3 **`Run`**: Turns on the virtual machine.

  - 4 **`Pause`**: Pauses the virtual machine.

  - 5 **`Shut down`**: Shuts down the virtual machine. Clicking on the arrow displays a dropdown menu with several options for turning off the virtual machine, including Reboot, Shut Down, Force Reset, Force Off, and Save.

- 6 **`Full screen`**: Switches the virtual machine to full screen view.

- 7 **Send Key**: Sends key combinations such as Ctrl+Alt+Backspace, Ctrl+Alt+Delete, Ctrl+Alt+F1, PrintScreen, and more to the virtual machine.

Note

Refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide* for more information about using the **Virtual Machine Manager** to create and run virtual machines.

Revision History

Revision 1.0-22	**Tues June 3 2014**	**Dayle Parker**

Version for 7.0 GA release

Revision 1.0-21	**Tues June 3 2014**	**Dayle Parker**

Updated support limit URLs under "KVM guest virtual machine compatibility" for BZ#1064610.
Updated warning under guestmount tool for BZ#1064611.

Revision 1.0-20	**Tues May 20 2014**	**Dayle Parker**

Updated support limit URLs under "KVM guest virtual machine compatibility" for BZ#1064610.

Revision 1.0-19	**Mon May 19 2014**	**Dayle Parker**

Added "shuts down" to offline migration description for QE feedback in BZ#1064608.
Minor edits to Products chapter for QE feedback in BZ#1064610.
Edited tools list found in QE review in BZ#1064611.
Removed introduction and preface material.

Revision 1.0-18	**Fri May 9 2014**	**Dayle Parker**

Rebuild for style changes.

Revision 1.0-14	**Tues May 6 2014**	**Dayle Parker**

Edited Virtual CPU hot add description in Products chapter.

Revision 1.0-12	**Mon April 28 2014**	**Dayle Parker**

Added callouts to images in Quick Start chapter for BZ#919320.

Revision 1.0-10	**Tues April 16 2014**	**Dayle Parker**

Edited virsh section for BZ#1056848.

Revision 1.0-09	**Mon April 14 2014**	**Dayle Parker**

Edited libvirt section for BZ#1056848.
Edited guest CPU models section for BZ#952964.

Revision 1.0-08	**Wed April 9 2014**	**Dayle Parker**

Version for beta refresh.

Revision 1.0-07	**Mon April 7 2014**	**Dayle Parker**

Updated references to other virtualization guides for BZ#872060.
Edited vCPU hot add description for SME feedback in BZ#953006.

Revision 1.0-06	**Thurs April 3 2014**	**Dayle Parker**

Edited quick start chapter for QE feedback in BZ#919320.
Added vCPU hot add description to KVM products list for BZ#953006.

Revision 1.0-04	**Wed April 2 2014**	**Dayle Parker**

Edited quick start chapter.
Added PCI Express bus to emulated system components list for BZ#952442.

Revision 1.0-03	**Wed April 2 2014**	**Dayle Parker**

Added ACL description to libvirt section for BZ#1056848.
Moved KVM compatibility information to end of "KVM and virtualization in Red Hat Enterprise Linux" sectio
for BZ#895819.

Revision 1.0-02 **Mon March 31 2014** **Dayle Parker**

Added virtualization quick start chapter for BZ#919320 and BZ#919282
Updated reference to additional disk I/O throttling information to the correct guide.
Edited para-virtualized devices section for QE feedback in BZ#1056429.

Revision 0.1-23 **Mon March 17 2014** **Dayle Parker**

Updated virtualization docs suite list to include new Red Hat Enterprise Virtualization documentation.
Added automatic NUMA balancing description for BZ#1071730.
Edited performance section for BZ#1056535.

Revision 0.1-22 **Wed March 12 2014** **Dayle Parker**

Updated references to other Red Hat Enterprise Linux 7 guides for BZ#872060.
Added QXL driver description for BZ#1056429.
Edited storage concepts section for BZ#1010806.
Edited emulated storage drivers section for BZ#1056424.

Revision 0.1-21 **Thurs Feb 27 2014** **Dayle Parker**

Edited virtualized and emulated devices section for BZ#1056424.
Added AHCI description.
Edited storage concepts section for BZ#1010806.

Revision 0.1-20 **Fri Feb 7 2014** **Dayle Parker**

Edited virt-install section for BZ#1056896.
Edited Boxes description for BZ#921813.

Revision 0.1-19 **Mon Jan 13 2014** **Dayle Parker**

Added storage concepts description for BZ#1010806.
Edited Boxes description for BZ#921813.
Applied QE feedback throughout guide.

Revision 0.1-18 **Thurs Dec 12 2013** **Dayle Parker**

Edited VFIO description based on SME feedback in BZ#952004.

Revision 0.1-17 **Fri Dec 6 2013** **Dayle Parker**

Version for beta.

Revision 0.1-16 **Wed Dec 4 2013** **Dayle Parker**

Added VFIO description for BZ#952004.

Revision 0.1-15 **Thurs Nov 28 2013** **Dayle Parker**

Added to GlusterFS description for BZ#1010826.

Revision 0.1-12 **Wed Oct 2 2013** **Dayle Parker**

Added disk I/O throttling description for BZ#1010821.
Edited terminology in Para-virtualized devices section for BZ#928129.
Updated references to other virtualization guides for BZ#872060.

Revision 0.1-11	**Mon June 10, 2013**	**Dayle Parker**

Added virtio-rng description to Para-virtualized Devices section for BZ#923008.
Added GNOME Boxes description to Tools.

Revision 0.1-10	**Mon May 6, 2013**	**Dayle Parker**

Rearranged Migration section and included live storage migration feature description for BZ#953341.
Added xHCI host controller to Emulated system components list.

Revision 0.1-7	**Mon Apr 29 2013**	**Dayle Parker**

Added author names to Author Group file.

Revision 0.1-6	**Fri Apr 26 2013**	**Dayle Parker**

Added documentation suite list to Chapter 1.

Revision 0.1-3	**Thurs Jan 31 2013**	**Dayle Parker**

Updated 4.4.1 Storage pools.

Revision 0.1-2	**Wed Jan 23 2013**	**Dayle Parker**

Updated Red Hat Enterprise Linux 6 references to Red Hat Enterprise Linux 7 to build first draft.

Revision 0.1-1	**Wed Jan 16 2013**	**Dayle Parker**

Branched from the Red Hat Enterprise Linux 6 version of the document.