Red Hat Enterprise Linux 7 Virtualization Getting Started Guide

An introduction to virtualization concepts

Jiri Herrmann
Laura Novich
Yehuda Zimmerman
Jacquelynn East
Dayle Parker
Scott Radvan

Red Hat Enterprise Linux 7 Virtualization Getting Started Guide

An introduction to virtualization concepts

Jiri Herrmann
Red Hat Customer Content Services
jherrman@redhat.com

Yehuda Zimmerman
Red Hat Customer Content Services
yzimmerm@redhat.com

Dayle Parker
Red Hat Customer Content Services

Laura Novich
Red Hat Customer Content Services

Jacquelynn East
Red Hat Customer Content Services

Scott Radvan
Red Hat Customer Content Services

Legal Notice

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. Note: This document is under development, is subject to substantial change, and is provided only as a preview. The included information and instructions should not be considered complete, and should be used with caution.

Table of Contents

Chapter 1. General Introduction to Virtualization

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. Virtualization is accomplished by using a *hypervisor*. This is a software layer or subsystem that controls hardware and enables running multiple operating systems, called *virtual machines (VMs)* or *guests*, on a single (usually physical) machine. This machine with its operating system is called a *host*. There are several virtualization methods:

Full virtualization

Full virtualization uses an unmodified version of the guest operating system. The guest addresses the host's CPU via a channel created by the hypervisor. Because the guest communicates directly with the CPU, this is the fastest virtualization method.

Paravirtualization

Paravirtualization uses a modified guest operating system. The guest communicates with the hypervisor. The hypervisor passes the unmodified calls from the guest to the CPU and other interfaces, both real and virtual. Because the calls are routed through the hypervisor, this method is slower than full virtualization.

Software virtualization (or emulation)

Software virtualization uses binary translation and other emulation techniques to run unmodified operating systems. The hypervisor translates the guest calls to a format that can be used by the host system. Because all calls are translated, this method is slower than virtualization. Note that Red Hat does not support software virtualization on Red Hat Enterprise Linux.

1.2. Virtualization Solutions

Red Hat offers the following major virtualization solutions, each with a different user focus and features:

Red Hat Enterprise Linux

The ability to create, run, and manage virtual machines, as well as a number of virtualization tools and features are included in Red Hat Enterprise Linux 7. This solution supports a limited number of running guests per host, as well as a limited range of guest types. As such, virtualization on Red Hat Enterprise Linux can be useful for example to developers who require testing in multiple environments, or to small businesses running several servers that do not have strict uptime requirements or service-level agreements (SLAs).

Important

This guide provides information about virtualization on Red Hat Enterprise Linux and does not refer to the other virtualization solutions.

Red Hat Virtualization

Red Hat Virtualization (RHV) is based on the Kernel-based Virtual Machine (KVM) technology like virtualization on Red Hat Enterprise Linux is, but offers an enhanced array of features. Designed for enterprise-class scalability and performance, it enables management of your entire virtual infrastructure, including hosts, virtual machines, networks, storage, and users from a centralized graphical interface.

Red Hat Virtualization can be used by enterprises running larger deployments or mission-critical applications. Examples of large deployments suited to Red Hat Virtualization include databases, trading platforms, and messaging systems that must run continuously without any downtime.

Note

For more information about Red Hat Virtualization, or to download a fully supported 60-day evaluation version, see http://www.redhat.com/en/technologies/virtualization/enterprise-virtualization. Alternatively, refer to the Red Hat Virtualization documentation suite.

Red Hat OpenStack Platform

Red Hat OpenStack Platform offers an integrated foundation to create, deploy, and scale a secure and reliable public or private OpenStack cloud.

Note

For more information about Red Hat OpenStack Platform, or to download a 60-day evaluation version, see https://www.redhat.com/en/technologies/linux-platforms/openstack-platform. Alternatively, refer to the Red Hat OpenStack Platform documentation suite.

Chapter 2. Why Use Virtualization?

Virtualization can be useful both for server deployments and individual desktop stations. Desktop virtualization offers cost-efficient centralized management and better disaster recovery. In addition, by using connection tools such as *ssh*, it is possible to connect to a desktop remotely.

When used for servers, virtualization can benefit not only larger networks, but also deployments with more than a single server. Virtualization provides live migration, high availability, fault tolerance, and streamlined backups.

2.1. Virtualization Costs

Virtualization can be expensive to introduce, but it often saves money in the long term. 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 that adequate planning is performed before migrating physical systems to virtualized ones, less time is needed to maintain them. This means less money needs to be spent on parts and labor.

Extended life for installed software

Older versions of software may not be able to run directly on more recent bare-metal machines. By running older software virtually on a larger, faster system, the life of the software may be extended while taking advantage of better performance from a 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 for computer systems.

2.2. Performance

Older virtualization versions supported only a single CPU. As a result, 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:

- Red Hat Enterprise Linux 6.4 and KVM recorded an industry-leading TPC-C benchmark with an IBM DB2 database running in an entirely virtualized x86 environment and delivering 88% of bare-metal performance. Due to resource demands, databases have previously been reserved for bare-metal deployments only.
- The industry standard SAP Sales and Distribution (SD) Standard Application Benchmark found that Red Hat Enterprise Linux 6.2 and KVM performs at the virtualization efficiency of 85% when compared to 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 information on performance tuning for virtualization, refer to the *Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide*.

2.3. 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. Note that this is not possible between all Red Hat Enterprise Linux releases. Consult the *Virtualization Deployment and Administration Guide* for details.

2.3.1. Benefits of Migrating Virtual Machines

Migration is useful for:

Load balancing

When a host machine is overloaded, one or more of its virtual machines could be migrated to other hosts using live migration. Similarly, machines that are not running and tend to overload can be migrated using offline migration.

Upgrading or making changes to the host

When the need arises to upgrade, add, or remove hardware devices on a 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 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 other physical locations for lower latency or for other reasons.

When the migration process moves a virtual machine's memory, the disk volume associated with the virtual machine is also migrated. This process is performed using live block migration.

Note

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

2.3.2. Virtualized to Virtualized Migration (V2V)

As a special type of migration, Red Hat Enterprise Linux 7 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 *V2V Knowledgebase articles*.

In addition, Red Hat Enterprise Linux 7.3 and later support physical-to-virtual (P2V) conversion using the **`virt-p2v`** tool. For details, see the P2V Knowledgebase article.

2.4. Security

KVM virtual machines use the following features to improve their security:

SELinux

Security-Enhanced Linux, or SELinux, provides Mandatory Access Control (MAC) for all Linux systems, and thus benefits also Linux guests. Under the control of SELinux, all processes and files are given a *type*, and their access on the system is limited by fine-grained controls of various types. SELinux limits the abilities of an attacker and works to prevent many common security exploits such as buffer overflow attacks and privilege escalation.

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 hardens the system against hypervisor bugs that might be used to attack the host or another virtual machine.

Note

For more information on security in virtualization, refer to the *Red Hat Enterprise Linux 7 Virtualization Security Guide*.

2.5. Disaster Recovery

Disaster recovery is quicker and easier when the systems are virtualized. On a physical system, if something serious goes wrong, a complete reinstall 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 the 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 from each other, one system's downtime will not affect any others.

Chapter 3. Introduction to Red Hat Virtualization Products and Features

This chapter introduces the main virtualization products and features available in Red Hat Enterprise Linux 7.

3.1. KVM and Virtualization in Red Hat Enterprise Linux

KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on a variety of architectures. It is built into the standard Red Hat Enterprise Linux 7 kernel and integrated with the Quick Emulator (QEMU), and it can run multiple 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.

Warning

Note that Red Hat does not support the QEMU Tiny Code Generator (TCG) emulation.

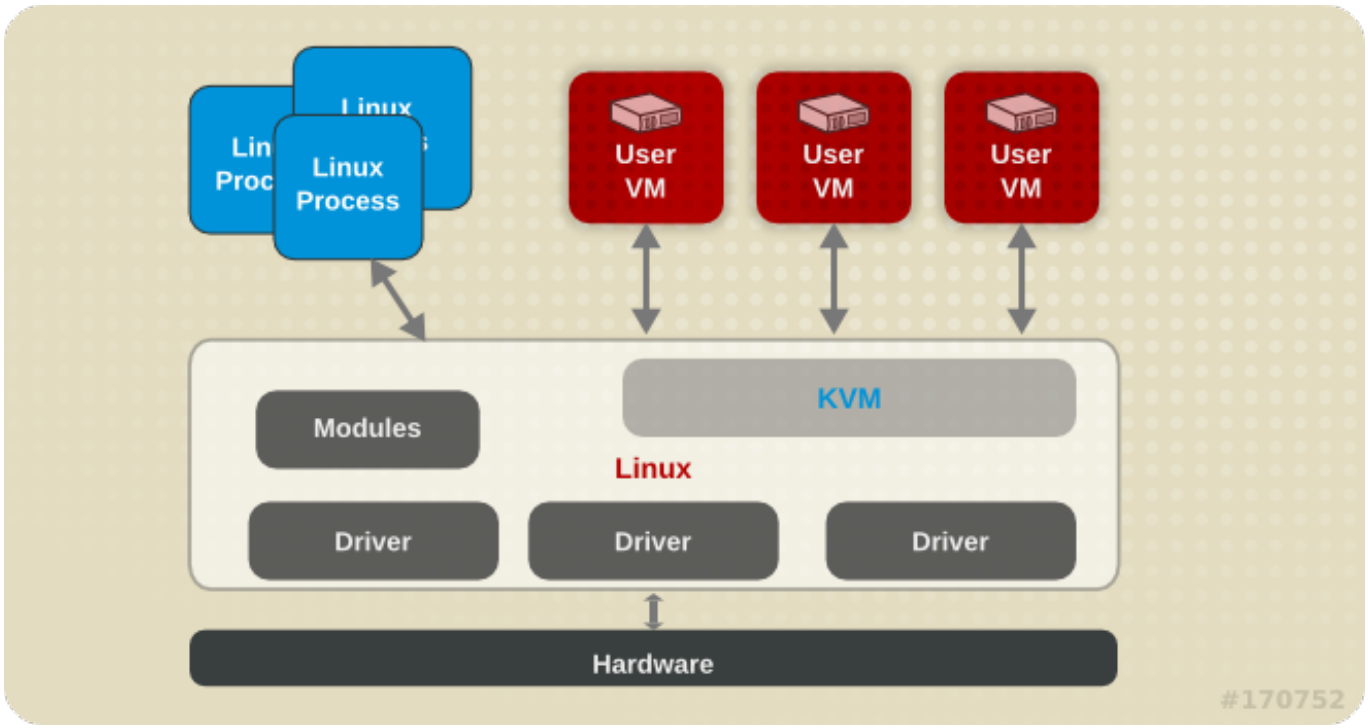


Figure 3.1. KVM architecture

Virtualization features supported by KVM on Red Hat Enterprise 7 include the following:

Overcommitting

The KVM hypervisor supports *overcommitting* of system resources. Overcommitting means allocating more virtualized CPUs or memory than the available resources on the system, so the resources can be dynamically swapped when required by one guest and not used by another. This can improve how efficiently guests use the resources of the host, and can

make it possible for the user to require fewer hosts.

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*.

KSM

Kernel Same-page Merging (KSM), used by the KVM hypervisor, enables 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 Tuning and Optimization Guide*.

QEMU guest agent

The *QEMU guest agent* runs on the guest operating system and makes it possible for 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*.

Disk I/O throttling

When several virtual machines are running simultaneously, they can interfere with the overall 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 individual 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

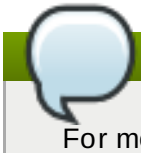
Automatic non-uniform memory access (NUMA) balancing moves tasks, which can be threads or processes closer to the memory they are accessing. This improves the performance of applications running on non-uniform memory access (NUMA) hardware systems, without any manual tuning required for Red Hat Enterprise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

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 on running virtual machines as needed, without shutting down the guests. The vCPUs assigned to a virtual machine can be added to a running guest 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*.

Nested virtualization

As a Technology Preview, Red Hat Enterprise Linux 7.2 and later offers hardware-assisted nested virtualization. This feature enables KVM guests to act as hypervisors and create their own guests.

This can for example be used for debugging hypervisors on a virtual machine or testing larger virtual deployments on a limited amount of physical machines.

Note

For further information on setting up and using nested virtualization, see *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 the host system: 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 Red Hat Customer Portal

Note

To verify whether your processor supports virtualization extensions and for information on enabling 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 provides a hypervisor-independent virtualization API that can interact with the virtualization capabilities of a range of operating systems. It includes:

- A virtualization layer to securely manage virtual machines on a host.
- An interface for managing local and networked hosts.
- The APIs required to provision, create, modify, monitor, control, migrate, and stop virtual machines. Although multiple hosts may be accessed with **libvirt** simultaneously, the APIs are limited to single node operations.

Note

Only operations supported by the hypervisor can be performed using libvirt.

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 do not need to be on the same physical machine as the machines on which the hosts are running. In such a scenario, the machine on which the management tools run communicates with the machines on which the hosts are running using secure protocols.

Red Hat Enterprise Linux 7 supports **libvirt** and includes **libvirt**-based tools as its default method for virtualization management (as in Red Hat 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 KVM on Red Hat Enterprise Linux 5, Red Hat Enterprise Linux 6, and Red Hat Enterprise Linux 7.

Notably, libvirt also provides the two primary tools for controlling virtualization on Red Hat Enterprise Linux 7: `virsh` and `virt-manager`.

3.3. Virtualized Hardware Devices

Virtualization on Red Hat Enterprise Linux 7 allows virtual machines to use the host's physical hardware as three distinct types of devices:

- Virtualized and emulated devices
- Paravirtualized 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 as software. These emulated hardware devices are crucial for virtualizing operating systems. Emulated devices are virtual devices which exist entirely in software.

In addition, KVM provides emulated drivers. These form 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, or mouse) that is recognized by the Linux kernel can be used as the backing source device for the emulated drivers.

Virtual CPUs (vCPUs)

On Red Hat Enterprise Linux 7.2 and above, the host system can have up to 240 virtual CPUs (vCPUs) that can be presented to guests for use, regardless of the number of host CPUs. This is up from 160 in Red Hat Enterprise Linux 7.0.

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 7.3)

Emulated storage drivers

Storage devices and storage pools can use 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.

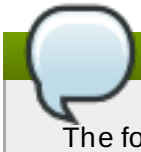
The emulated floppy disk drive driver

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

Emulated sound devices

An emulated (Intel) HDA sound device, **intel-hda**, is supported in the following guest operating systems:

- Red Hat Enterprise Linux 7, for the AMD64 and Intel 64 architecture
- Red Hat Enterprise Linux 4, 5, and 6, for the 32-bit AMD and Intel architecture and the AMD64 and Intel 64 architecture



Note

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 graphics cards

The following emulated graphics cards are provided.

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

Guests can connect to these devices with the Simple Protocol for Independent Computing Environments (SPICE) protocol or with the Virtual Network Computing (VNC) system.

Emulated network devices

The following two emulated network devices are provided:

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

Emulated watchdog devices

A watchdog can be used to automatically reboot a virtual machine when the machine becomes overloaded or unresponsive.

Red Hat Enterprise Linux 7 provides the following emulated watchdog devices:

- **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.

3.3.2. Paravirtualized Devices

3.3.2. Paravirtualized Devices

Paravirtualization provides a fast and efficient means of communication for guests to use devices on the host machine. KVM provides paravirtualized devices to virtual machines using the virtio API as a layer between the hypervisor and guest.

Some paravirtualized devices decrease I/O latency and increase I/O throughput to near bare-metal levels, while other paravirtualized devices add functionality to virtual machines that is not otherwise available. It is recommended to use paravirtualized 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. Paravirtualized device drivers allow the guest operating system access to physical devices on the host system.

To use this device, the paravirtualized device drivers must be installed on the guest operating system. By default, the paravirtualized device drivers are included in Red Hat Enterprise Linux 4.7 and later, Red Hat Enterprise Linux 5.4 and later, and Red Hat Enterprise Linux 6.0 and later.

Note

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

The paravirtualized network device (virtio-net)

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

The paravirtualized block device (virtio-blk)

The paravirtualized block device is a high-performance virtual storage device that provides storage to virtual machines with increased I/O performance and lower latency. The paravirtualized block device is supported by the hypervisor and is attached to the virtual machine (except for floppy disk drives, which must be emulated).

The paravirtualized controller device (virtio-scsi)

The paravirtualized 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

The paravirtualized 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 paravirtualized clock. Additionally, the paravirtualized clock assists with time adjustments needed after a guest runs the sleep (S3) or suspend to RAM operations.

The paravirtualized serial device (virtio-serial)

The paravirtualized 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 *inflating* the balloon), 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 paravirtualized random number generator (virtio-rng)

The paravirtualized 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 paravirtualized graphics card (QXL)

The paravirtualized 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.

3.3.3. Physical Host Devices

Certain hardware platforms enable virtual machines to directly access various hardware devices and components. This process in virtualization is known as *device assignment*, or also 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, providing guests with exclusive access to PCI devices for a range of tasks. This enables 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, up 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, PCI, and SCSI passthrough

The KVM hypervisor supports attaching USB, PCI, and SCSI devices on the host system to virtual machines. USB, PCI, and SCSI device assignment makes it possible for the devices to appear and behave as if they were physically attached to the virtual machine. Thus, it provides guests with exclusive access to these devices for a variety of tasks.

Note

For more information on USB, PCI, and SCSI 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 (PCI-e) 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.

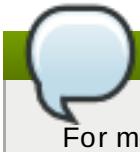
Note

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.



Note

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

3.3.4. Guest CPU Models

CPU models define which host CPU features are exposed to the guest operating system. **KVM** and **libvirt** contain definitions for a number of 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 **KVM** code.

To ensure safe migration of virtual machines between hosts with different sets of CPU features, **KVM** does not expose all features of the host CPU to guest operating system by default. Instead, CPU features are exposed based on the selected CPU model. If a virtual machine has a given CPU feature enabled, it cannot be migrated 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 allocated to the virtual machine. It is attached to the virtual machine using the paravirtualized 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. For more information, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

Local storage pools

Local storage pools are attached directly to the host server. They include local directories, directly attached disks, physical partitions, and Logical Volume Management (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 environment, because 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 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, allowing for maximum flexibility and compatibility with guest operating systems.

virtio-scsi

virtio-scsi is the recommended paravirtualized 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 paravirtualized 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 and make it possible for guests to use ISO files or the host's CD-ROM drive. 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

Emulated 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.

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 can only be 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.

Red Hat Enterprise Linux 7 supports the qcow2 version 3 image file format.

LVM volumes

Logical volumes (LVs) 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, raw disks, and logical unit numbers (LUNs) can be presented to the guest. This enables 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 enables 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*.

3.5. Virtual Networking

A virtual guest's connection to any network uses the software network components of the physical host. These software components can be rearranged and reconfigured by using **`libvirt`**'s virtual network configuration. The host therefore acts as a virtual network switch, which can be configured in a number of different ways to fit the guest's networking needs.

By default, all guests on a single host are connected to the same libvirt virtual network, named **`default`**. Guests on this network can make the following connections:

With each other and with the virtualization host

Both inbound and outbound traffic is possible, but is affected by the firewalls in the guest operating system's network stack and by libvirt network filtering rules attached to the guest interface.

With other hosts on the network beyond the virtualization host

Only outbound traffic is possible, and is affected by Network Address Translation (NAT) rules, as well as the host system's firewall.

However, if needed, guest interfaces can instead be set to one of the following modes:

Isolated mode

The guests are connected to a network that does not allow any traffic beyond the virtualization host.

Routed mode

The guests are connected to a network that routes traffic between the guest and external hosts without performing any NAT. This enables incoming connections but requires extra routing-table entries for systems on the external network.

Bridged mode

The guests are connected to a bridge device that is also connected directly to a physical ethernet device connected to the local ethernet. This makes the guest directly visible on the physical network, and thus enables incoming connections, but does not require any extra routing-table entries.

For basic outbound-only network access from virtual machines, no additional network setup is usually needed, as the **`default`** network is installed along with the *libvirt* package, and automatically started when the **`libvirtd`** service is started. If more advanced functionality is needed, additional networks can be created and configured using either virsh or virt-manager, and the guest XML configuration file can be edited to use one of these new networks.

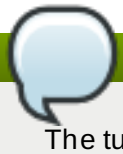
Note

For information on advanced virtual network settings, see the Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide.

From the point of view of the guest operating system, a virtual network connection is the same as a normal physical network connection. For further information on configuring networks in Red Hat Enterprise Linux 7 guests, see the Red Hat Enterprise Linux 7 Networking Guide.

Chapter 4. Quick Start Tutorial on Virtualization in Red Hat Enterprise Linux 7

This chapter provides instructions on installing the essential virtualization tools and creating a virtual machine in Red Hat Enterprise Linux 7.



Note

The tutorial uses Virtual Machine Manager to quickly create a virtual machine for trying out 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* .

4.1. Tutorial overview

- Ensure that your system meets the basic requirements
- Ensure that the required packages are installed
- Create a virtual machine using Virtual Machine Manager

4.2. Basic Requirements and Setup

To set up a KVM virtual machine on Red Hat Enterprise 7, your system must meet the following criteria:

Architecture

Virtualization with the KVM hypervisor is currently only supported on Intel64 and AMD 64 systems.

Disk space and RAM

Minimum:

- 6 GB free disk space
- 2 GB RAM

Customer Portal registration

To install virtualization packages, your host machine must be registered and subscribed to the Red Hat Customer Portal. To register run the **`subscription-manager register`** command and follow the prompts. Alternatively, run the Red Hat Subscription Manager application from **Applications → System Tools** on the desktop to register.

If you do not have a valid Red Hat subscription, visit the Red Hat online store to obtain one. For more information on registering and subscribing a system to the Red Hat Customer Portal, see https://access.redhat.com/solutions/253273.

Required packages

Before you can use virtualization, a basic set of virtualization packages must be installed on your computer.

Procedure 4.1. Installing the virtualization packages with yum

To use virtualization on Red Hat Enterprise Linux, the **qemu-kvm**, **qemu-img**, and **libvirt** packages must be installed. These provide the user-level KVM emulator, disk image manager, and virtualization management tools on the host system.

1. Install the *qemu-kvm*, *qemu-img*, *libvirt*, and *virt-manager* packages:

```
# yum install qemu-kvm qemu-img libvirt virt-manager
```

2. Download a binary DVD ISO image from the Red Hat Customer Portal. This example uses Red Hat Enterprise Linux 6 Workstation. The image file will be used to install the guest virtual machine's operating system.

Note

If you encounter any problems during the installation process, see the Troubleshooting section of the Red Hat Enterprise Linux 7 *Virtualization Deployment and Administration Guide*.

4.3. Creating a Virtual Machine with Virtual Machine Manager

Virtual Machine Manager, also known as **virt-manager**, is a graphical tool for quick deployment of 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.

4.3.1. Introduction to Virtual Machine Manager

To open the **Virtual Machine Manager**, click **Applications** → **System Tools** → **Virtual Machine Manager**; or press the **Super** key → type **virt-manager** → press **Enter**.

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

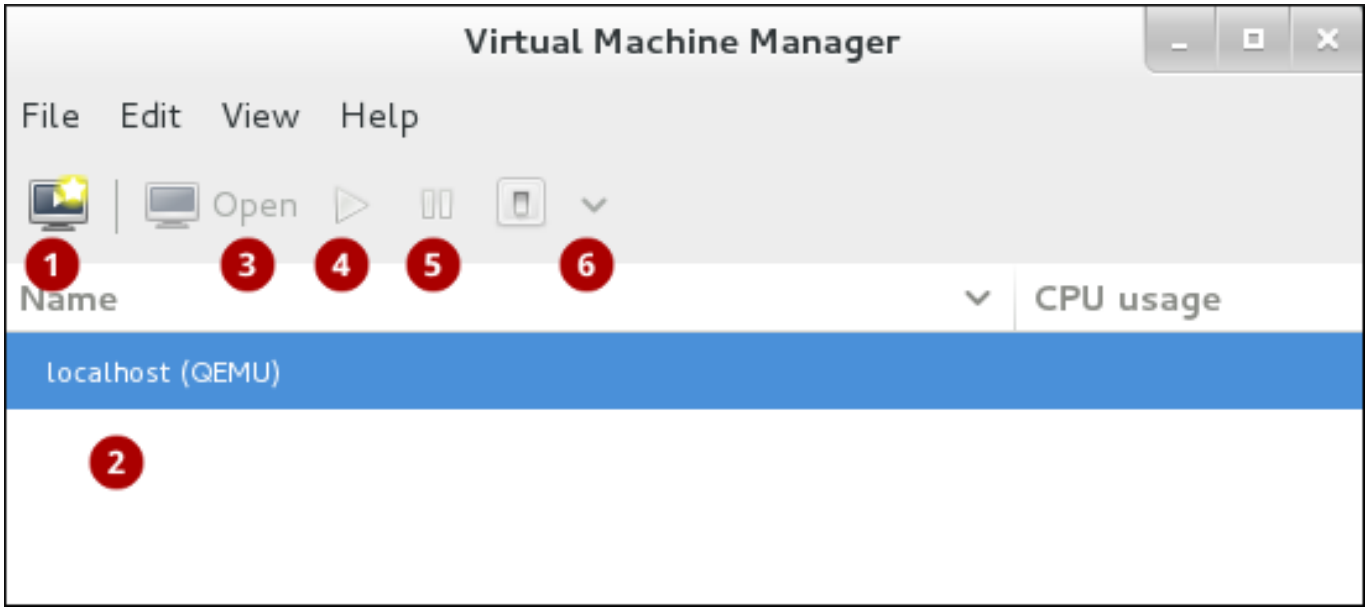


Figure 4.1. The Virtual Machine Manager interface

Commonly used elements of the interface include:

- 1 **`Create new virtual machine`**: Click to create a new virtual machine.
- 2 **`Virtual machines`**: A list of configured connections and all guest virtual machines associated with them. When a virtual machine is created, it is listed here. When a guest is running, an animated graph shows the guest's CPU usage in the **`CPU usage`** column.

  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 the arrow displays a drop-down menu with several options for turning off the virtual machine, including Reboot, Shut Down, Force Reset, Force Off, and Save.

Right-clicking 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.

4.3.2. Creating a Virtual Machine with Virtual Machine Manager

Follow these steps to create a Red Hat Enterprise Linux 6 virtual machine on **Virtual Machine Manager**.

Procedure 4.2. Creating a guest virtual machine with Virtual Machine Manager

1. **Open Virtual Machine Manager**

   Click **Applications → System Tools → Virtual Machine Manager**

   or

   Press the **Super** key, type **`virt-manager`**, and press **Enter**

2. **Create a new virtual machine**

   Click  to open the **New VM** wizard.

3. **Specify the installation method**

   Start the creation process by choosing the method of installing the new virtual machine.

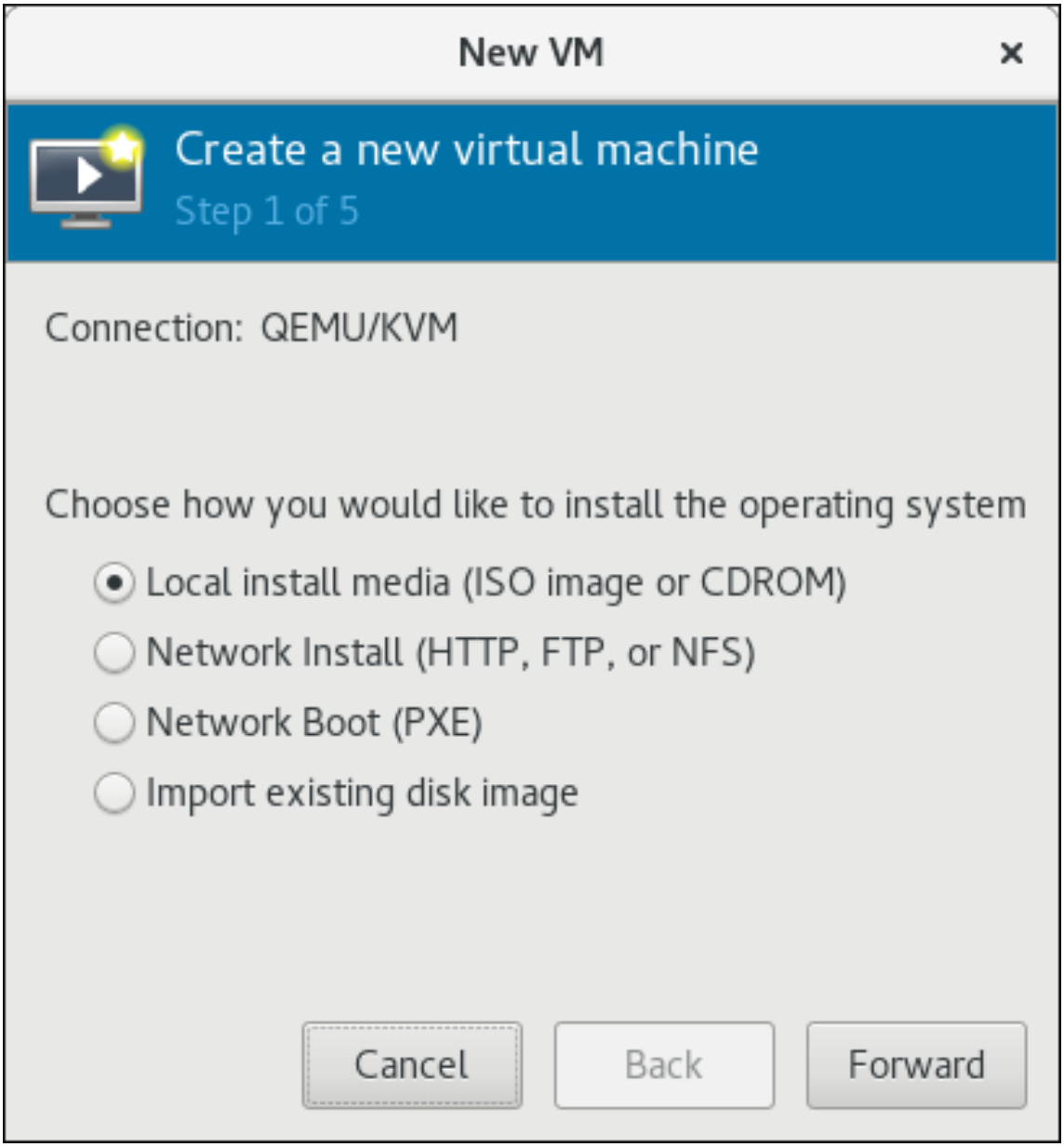


Figure 4.2. Select the installation method

For this tutorial, select **Local install media (ISO image)**. This installation method uses an image of an installation disk (in this case an **.iso** file). Click **Forward** to continue to the next step.

4. **Locate installation media**

   a. Select the **Use ISO Image** option.

   b. Click **Browse** → **Browse Local** buttons.

   c. Locate the ISO downloaded in Procedure 4.1, "Installing the virtualization packages with **yum**" on your machine.

   d. Select the ISO file and click **Open**.

   e. Ensure that Virtual Machine Manager correctly detected the OS type. If not, uncheck **Automatically detect operating system based on install media** and select **Linux** from the **OS type** drop-down and **Red Hat Enterprise Linux 6** from the **Version** drop-down.

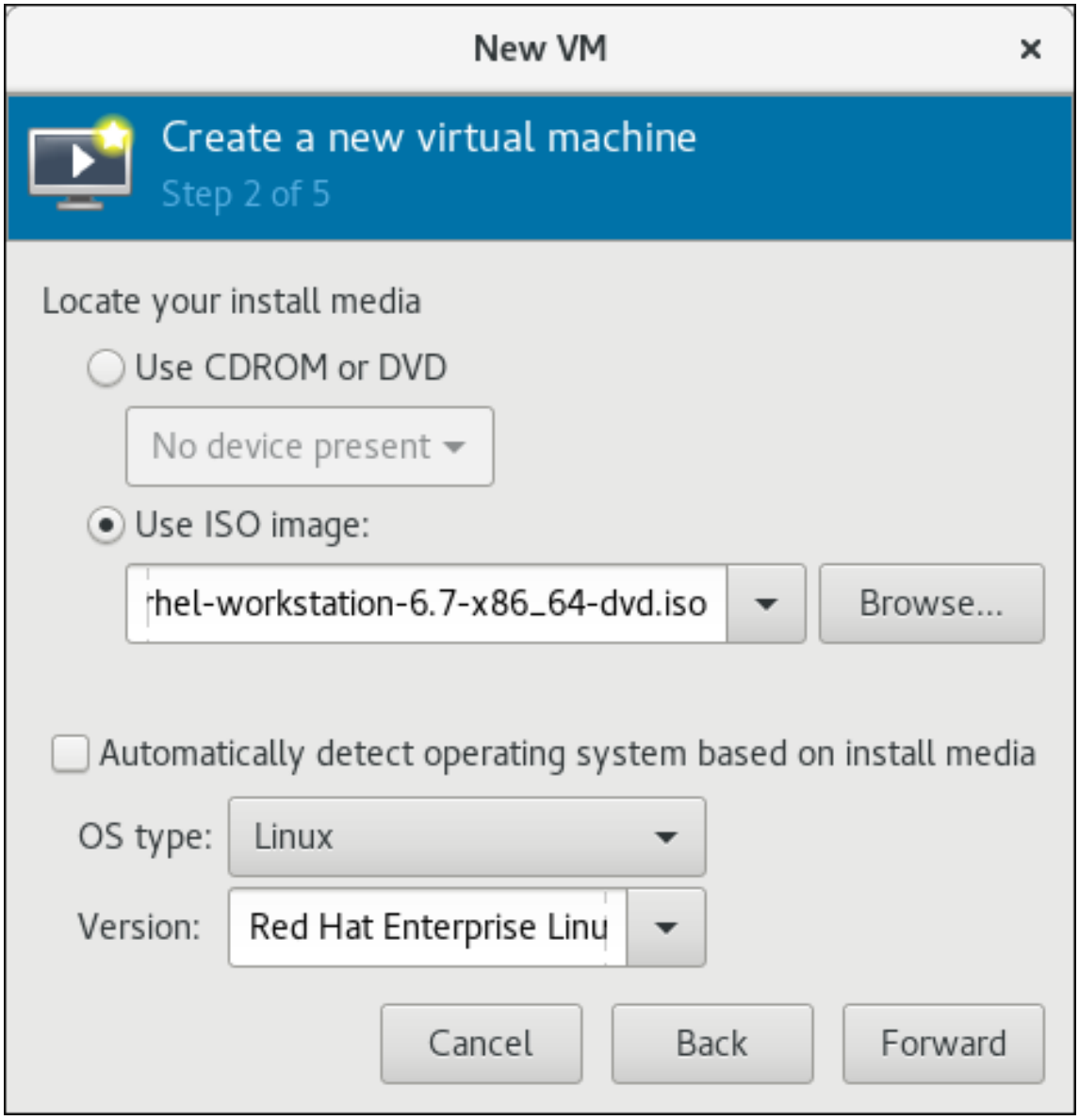


Figure 4.3. Local ISO image installation

5. **Configure memory and CPU**

   You can use step 3 of the wizard to configure 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, leave the default settings and click **Forward**.

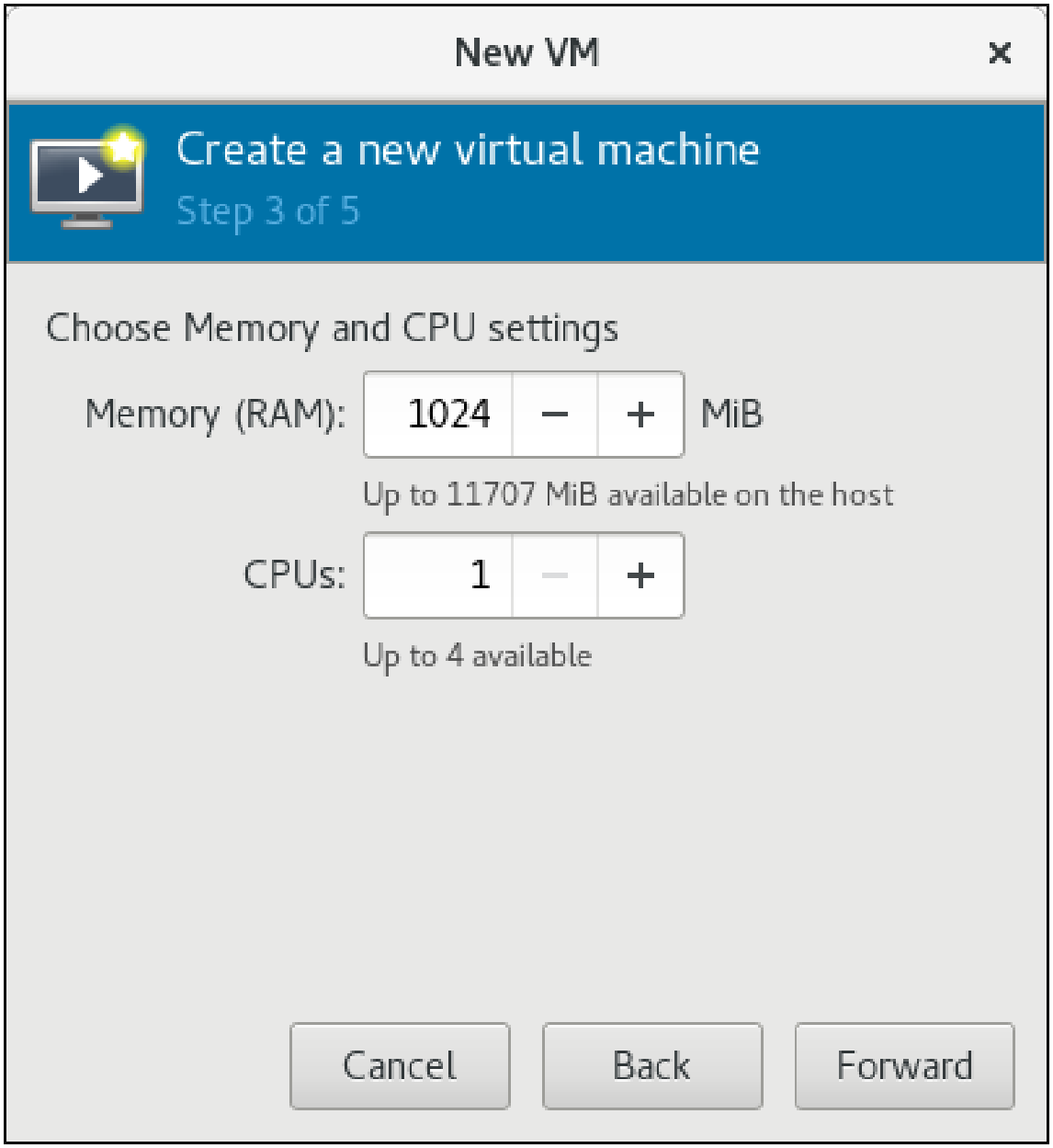


Figure 4.4. Configuring CPU and Memory

6. **Configure storage**

   Using step 4 of the wizard, you can 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, leave the default settings and click **Forward**.

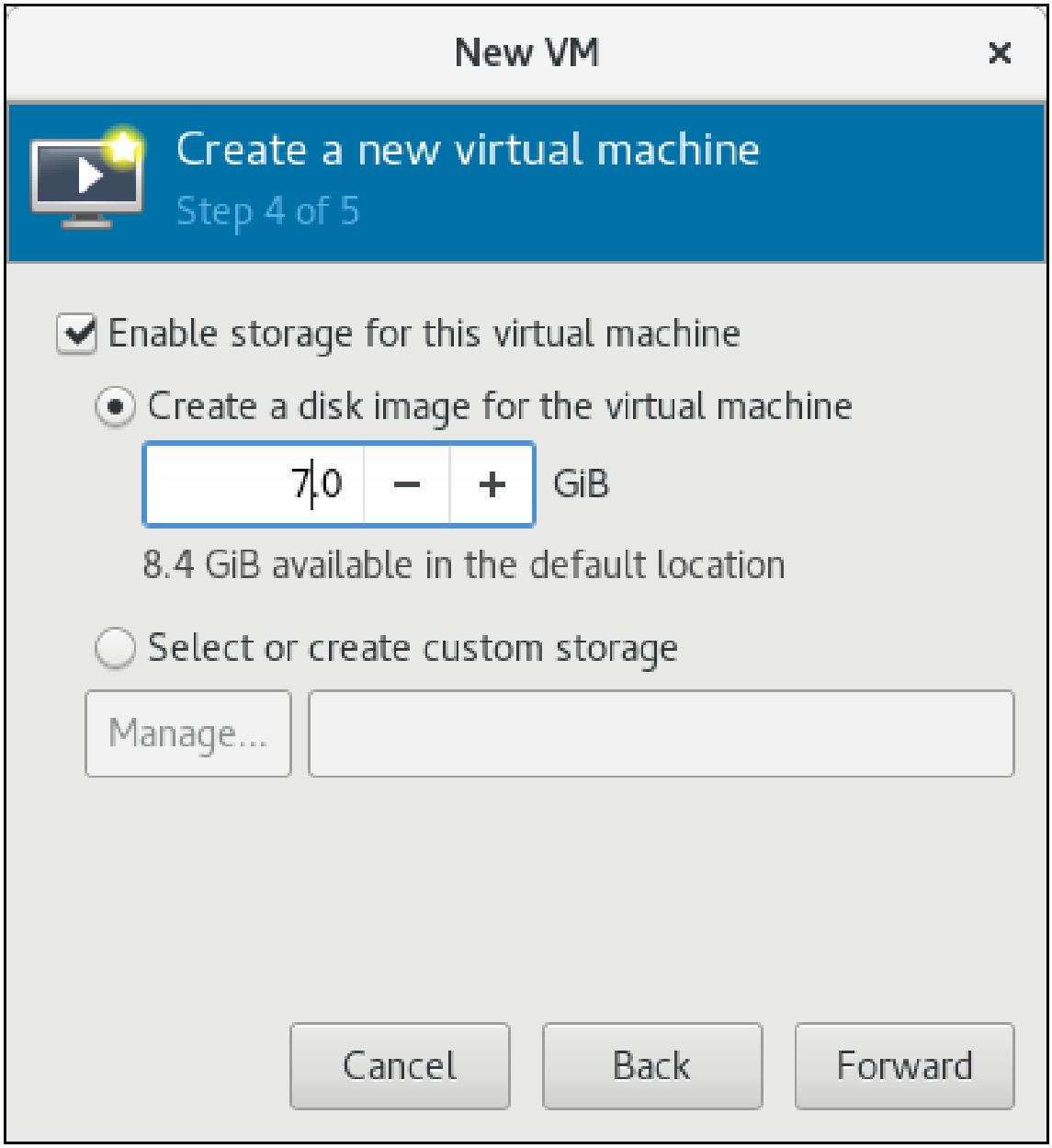


Figure 4.5. Configuring storage

7. **Name and review**

   Using step 5 of the wizard, you can create a name for the virtual machine and configure network settings. For this tutorial, enter a name for the virtual machine, verify the settings, and click **Finish**. **Virtual Machine Manager** will create a virtual machine with the specified hardware settings.

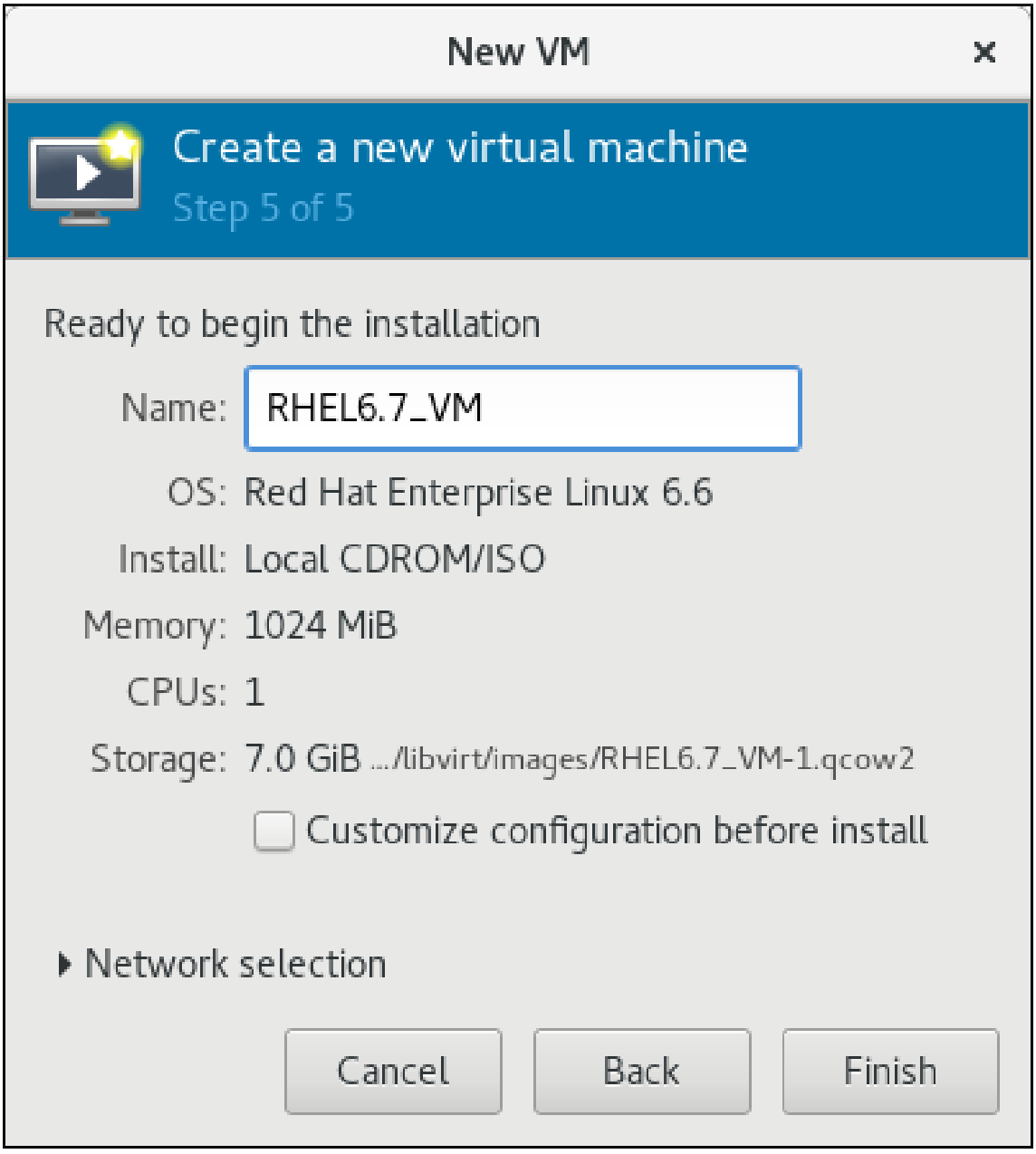


Figure 4.6. Naming and verification

After **Virtual Machine Manager** creates your Red Hat Enterprise Linux 6 virtual machine, the virtual machine's window will open, and the installation of the selected operating system will begin in it. Follow the instructions in the Red Hat Enterprise Linux 6 installer to complete the installation of the virtual machine's operating system.

Note

For help with Red Hat Enterprise Linux 6 installation, refer to the *Red Hat Enterprise Linux 6 Installation Guide*.

4.3.3. 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 6 virtual machine from the console in the same way as a physical system.

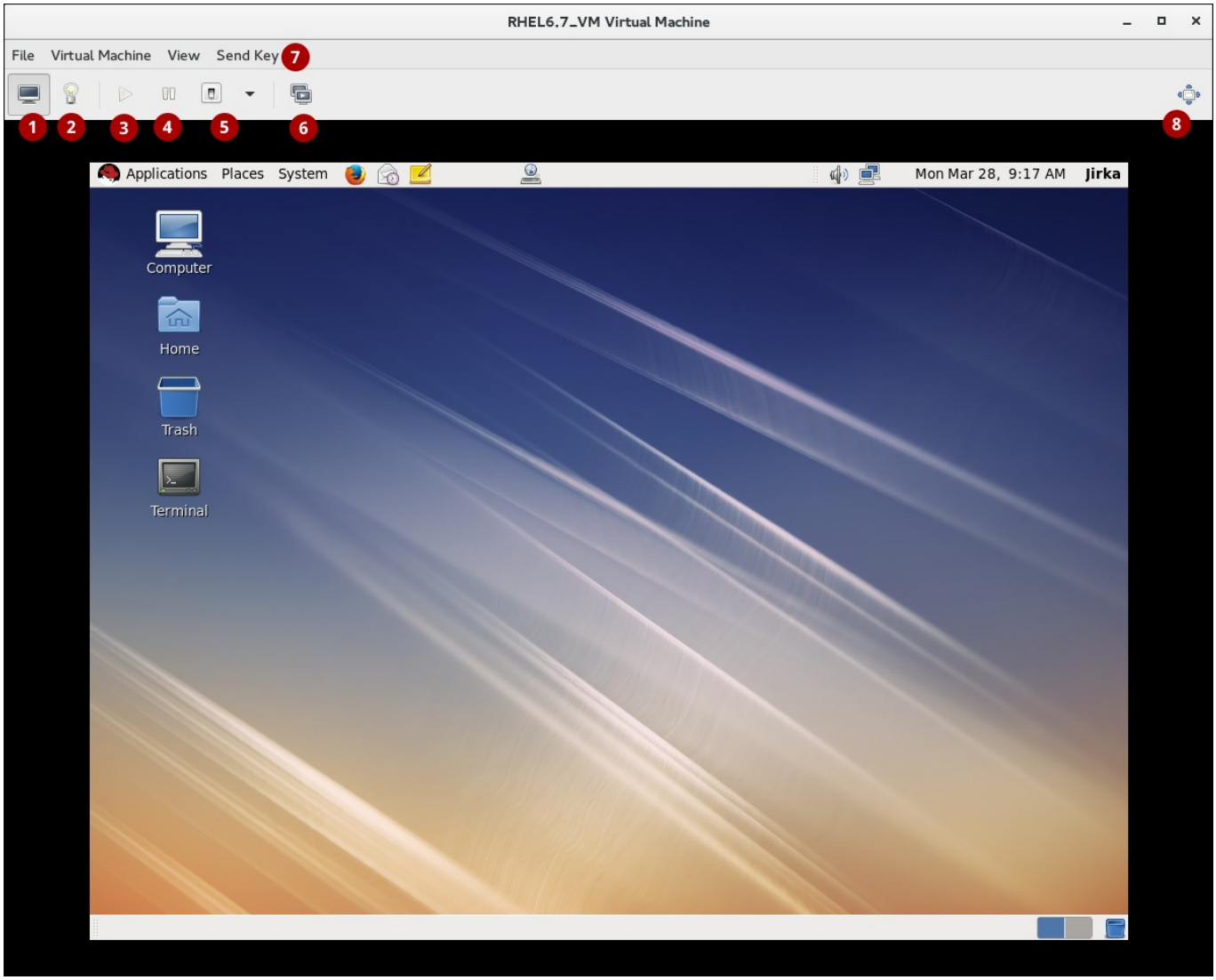


Figure 4.7. The guest virtual machine console

- 1 **Show the graphical console**: Shows the virtual machine's display. The virtual machine can be operated from the console the same as a physical machine.
- 2 **Show virtual hardware details**: Shows details about the virtual hardware that the guest is using. These include an overview of basic system details, performance, processor, memory, and boot settings, and details of the system's virtual devices. For more information, refer to the Virtualization Deployment and Administration Guide.
- 3 **Run**: Turns on the virtual machine.
- 4 **Pause**: Pauses the virtual machine.

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

6. **Manage snapshots**: Enables creating, running, and managing snapshots of the virtual machine. For details, see the Virtualization Deployment and Administration Guide.

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

8. **Full screen**: Switches the virtual machine to full screen view.

Note

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

Chapter 5. Virtualization Tools

This chapter provides an introduction to the basic tools available to interact with virtualization on Red Hat Enterprise Linux 7.

5.1. virsh

virsh is a command-line interface (CLI) tool for managing the hypervisor and guest virtual machines, and works as the primary means of controlling virtualization on Red Hat Enterprise Linux 7. Its capabilities include creating, configuring, pausing, listing, and shutting down virtual machines, as well as managing virtual networks, or loading virtual machine disk images. As such, it is ideal for creating virtualization administration scripts. Users without root privileges can use the **virsh** command as well, but in read-only mode. **virsh** is installed as part of the *libvirt-client* package.

Note

For more information about managing virtual machines with **virsh**, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

5.2. virt-manager

Virtual Machine Manager, also known as *virt-manager*, is a lightweight graphical tool for managing virtual machines, more user-friendly and less powerful than **virsh**. It enables the user to use a GUI for performing actions equivalent to a number of **virsh** and **virt-** commands, such as controlling the life cycle of existing machines, provisioning new machines, managing virtual networks, accessing the graphical console of virtual machines, and viewing performance statistics. For a demonstration of using **virt-manager**, see Section 4.3, “Creating a Virtual Machine with Virtual Machine Manager”. This tool is provided by the *virt-manager* package.

Note

For more information about managing virtual machines with **virt-manager**, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

5.3. virt-install

virt-install is a command-line utility for provisioning 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 use the kickstart method to prepare the guest, allowing for easy automation of installation. This tool is included in the *virt-install* package.

Note

For more information about using **virt-install**, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

5.4. guestfish

guestfish is a shell and command-line utility for examining and modifying virtual machine disk images, as these actions cannot be performed using **virsh** or **virt-manager**. The **guestfish** tool uses the *libguestfs* library and exposes all functionality provided by the **guestfs** API.

Warning

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

Note

For more information about **guestfish**, refer to the *Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide*.

5.5. GNOME Boxes

Boxes is a lightweight graphical desktop virtualization tool used to view and access virtual machines and remote systems. It can serve as an easier-to-use alternative to **virt-manager**, but offers fewer guest management options. 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 pre-configures virtual machines with optimal settings. This tool is provided by the *gnome-boxes* package.

Note

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

5.6. Other Useful Tools

The following tools can be used to access and modify a guest virtual machine's disk image using commands on the host.

guestmount

A command-line utility used to mount virtual machine file systems and disk images on the host machine. To unmount content mounted this way, use **guestunmount** Both tools are

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

Warning

Using **guestmount** in **--rw** (read/write) mode to access a disk that is currently being used by a guest may 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-builder

A command-line utility for quickly building and customizing new virtual machines. This tool is installed in Red Hat Enterprise Linux 7.1 and later as part of the *libguestfs-tools* package.

virt-cat

A command-line utility 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-copy-in and virt-copy-out

A command-line utilitys used for copying files or directories from the host to a guest, and from a guest to the host, respectively. These tools are installed as part of the *libguestfs-tools* package.

Warning

Use **virt-copy-in** only on turned off virtual machines that are not being used by another disk-editing tool. Using **virt-copy-in** on an active guest image or an image that is being edited may cause disk corruption in the virtual machine.

virt-customize

A command-line utility for customizing virtual machine disk images. virt-customize can be used to install packages, edit configuration files, run scripts, and set passwords. This tool is installed in Red Hat Enterprise Linux 7.1 and later as part of the *libguestfs-tools* package.

virt-df

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

virt-diff

A command-line utility for showing differences between the file systems of two virtual machines. This can be useful for example to discover which files have changed between snapshots. This tool is installed in Red Hat Enterprise Linux 7.1 and later as part of the *libguestfs-tools* package.

virt-edit

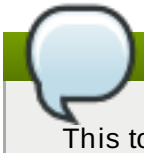
A command-line utility used to edit files that exist on a specified virtual machine. This tool is installed as part of the *libguestfs-tools* package.

Warning

Using **virt-edit** on live virtual machines may 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 utility 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.



Note

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

virt-format

A command-line utility capable of formatting guest image files, but also logical volumes on the host machine. This tool is installed as part of the *libguestfs-tools* package.

Warning

Use **virt-format** only on turned off virtual machines that are not being used by another disk-editing tool. Using **virt-format** on an active guest image or an image that is being edited may cause disk corruption in the virtual machine.

virt-inspector

A command-line utility 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 virtual machine at a time. This tool is installed as part of the *libguestfs-tools* package.

virt-log

A command-line utility for listing log files from virtual machines. This tool is installed in Red Hat Enterprise Linux 7.1 and later as part of the *libguestfs-tools* package.

virt-ls

A command-line utility 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 utility 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-p2v

A bootable tool that, when used in combination with virt-v2v, makes it possible to convert physical machines into KVM virtual machines. Note that virt-p2v is only supported in Red Hat Enterprise Linux 7.3 and later.

virt-rescue

A command-line utility 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 may 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 utility 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 recommended to shut down virtual machines before attempting to resize them.

virt-sparsify

A command-line utility to make a virtual machine disk (or any disk image) thin-provisioned. The tool can convert free space in the disk image to free space on the host.

virt-sysprep

A command-line utility 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-out and virt-tar-in

Command-line archive tools for packing a virtual machine disk image directory into a tarball, and unpacking an uncompressed tarball into a virtual machine disk image or specified libvirt guest, respectively. These tools are installed as part of the *libguestfs-tools* package.

Warning

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

virt-top

A command-line utility similar to **top**, which shows statistics related to guest virtual machines. This tool is contained the *virt-top* package. See **man virt-top** for details.

virt-v2v

A command-line utility for converting virtual machines from a foreign hypervisor to run on KVM managed by libvirt. Currently, virt-v2v can convert Red Hat Enterprise Linux and Windows guests running on Xen and VMware ESX. The virt-v2v tool is installed in Red Hat Enterprise Linux 7.1 and later as part of the *virt-v2v* package.

virt-viewer

A lightweight utility for displaying the graphical console of a virtual machine via the VNC and SPICE protocols. This tool is provided by the *virt-viewer* package.

virt-what

A shell script that detects whether a program is running in a virtual machine. This tool is provided by the *virt-what* package.

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 is provided by the *virt-who* package.

virt-xml-validate

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

Appendix A. Revision History

Revision 1.0-49 **Tue Mar 28 2017** **Jiri Herrmann**
Added multiple asynchronous updates

Revision 1.0-47 **Mon Oct 17 2016** **Jiri Herrmann**
Version for 7.3 GA publication

Revision 1.0-44 **Mon Dec 21 2015** **Laura Novich**
Republished the guide for several bug fixes

Revision 1.0-43 **Thu Oct 08 2015** **Jiri Herrmann**
Cleaned up the Revision History

Revision 1.0-42 **Sun Jun 28 2015** **Jiri Herrmann**
Updated for the 7.2 beta release