



Red Hat Enterprise Linux 7

Virtualization Getting Started Guide

Introduction to virtualization technologies available with RHEL

Red Hat Enterprise Linux 7 Virtualization Getting Started Guide

Introduction to virtualization technologies available with RHEL

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

Copyright © 2019 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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. GENERAL INTRODUCTION TO VIRTUALIZATION	3
1.1. WHAT IS VIRTUALIZATION?	3
1.2. VIRTUALIZATION SOLUTIONS	3
CHAPTER 2. WHY USE VIRTUALIZATION?	5
2.1. VIRTUALIZATION COSTS	5
2.2. PERFORMANCE	5
2.3. MIGRATION	6
2.4. SECURITY	7
2.5. DISASTER RECOVERY	8
CHAPTER 3. INTRODUCTION TO RED HAT VIRTUALIZATION PRODUCTS AND FEATURES	9
3.1. KVM AND VIRTUALIZATION IN RED HAT ENTERPRISE LINUX	9
3.2. LIBVIRT AND LIBVIRT TOOLS	11
3.3. DOMAIN XML CONFIGURATION	12
3.4. VIRTUALIZED HARDWARE DEVICES	13
3.5. STORAGE	18
3.6. VIRTUAL NETWORKING	21
CHAPTER 4. GETTING STARTED WITH VIRTUALIZATION COMMAND-LINE INTERFACE	23
4.1. PRIMARY COMMAND-LINE UTILITIES FOR VIRTUALIZATION	23
4.2. DEMONSTRATION: CREATING AND MANAGING A GUEST WITH COMMAND-LINE UTILITIES	27
CHAPTER 5. GETTING STARTED WITH VIRTUAL MACHINE MANAGER	33
5.1. RUNNING VIRTUAL MACHINE MANAGER	33
5.2. THE VIRTUAL MACHINE MANAGER INTERFACE	34
APPENDIX A. REVISION HISTORY	45

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*. For more information, see the [Red Hat Customer Portal](#).

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.

Containerization

While KVM virtualization creates a separate instance of OS kernel, operating-system-level virtualization, also known as *containerization*, operates on top of an existing OS kernel and creates isolated instances of the host OS, known as *containers*. For more information on containers, see the [Red Hat Customer Portal](#).

Containers do not have the versatility of KVM virtualization, but are more lightweight and flexible to handle. For a more detailed comparison, see the [Introduction to Linux Containers](#).

To use containers on Red Hat Enterprise Linux, [install the docker packages from the Extras channel](#). Note that Red Hat also offers optimized solutions for using containers, such as [Red Hat Enterprise Linux Atomic Host](#) and [Red Hat OpenShift Container Platform](#). For details on container support, see the [Red Hat KnowledgeBase](#).

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).

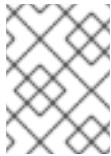


IMPORTANT

This guide provides information about virtualization on Red Hat Enterprise Linux and does not go into detail about 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.



NOTE

For more information about the differences between virtualization in Red Hat Enterprise Linux and Red Hat Virtualization, see the [Red Hat Customer Portal](#).

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, see [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, see [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, see 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, see 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, see 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

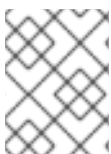
KVM virtual machines use the following features to improve their security:

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, see 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

QEMU and **libvirt** also support a dynamic translation mode using the QEMU Tiny Code Generator (TCG), which does not require hardware virtualization support. This configuration is not supported by Red Hat.

For more information about this limitation, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

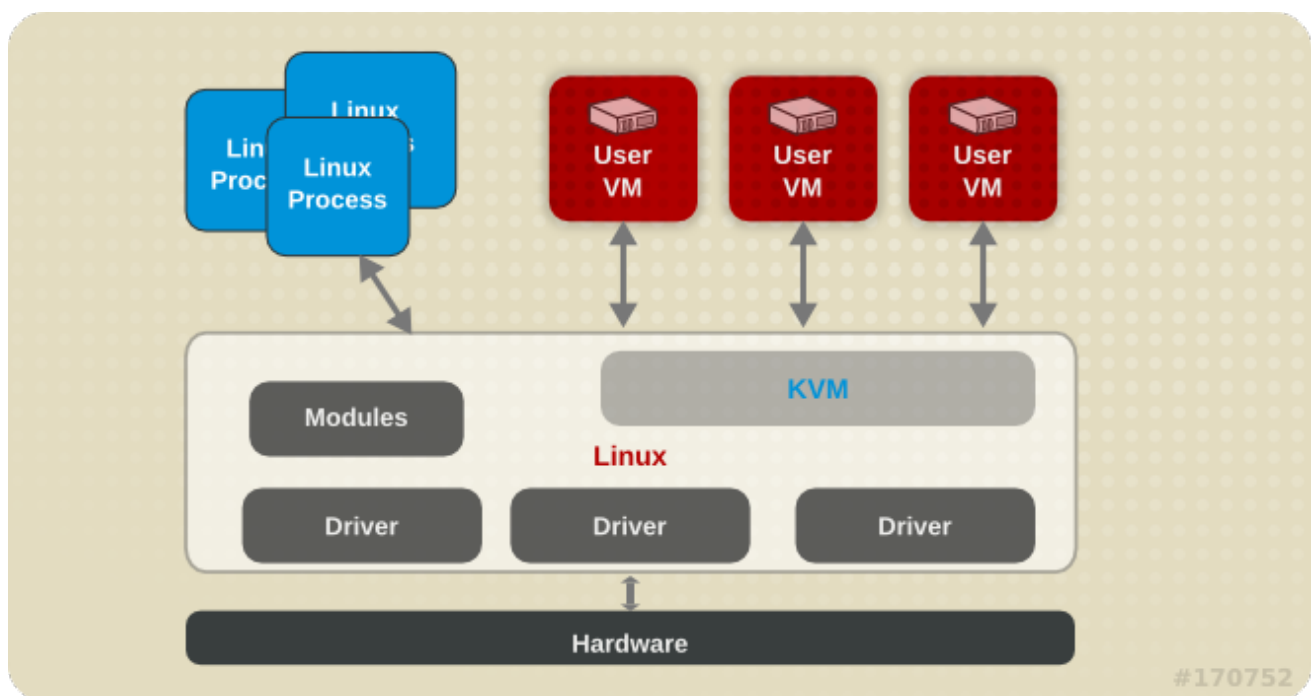


Figure 3.1. KVM architecture

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, see 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, see 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.

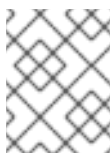


NOTE

For more information on the QEMU guest agent, see 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.

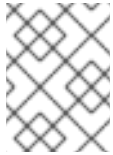


NOTE

For instructions on using disk I/O throttling, see 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, see 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.

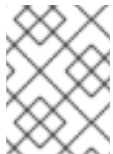
**NOTE**

For more information on virtual CPU hot add, see 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 see [Red Hat Customer Portal](#)

**NOTE**

To verify whether your processor supports virtualization extensions and for information on enabling virtualization extensions if they are disabled, see 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. DOMAIN XML CONFIGURATION

The host-based configuration of each KVM guest virtual machine (also referred to as a *domain*) is stored in the guest's XML configuration (or *domain XML*). This includes the settings for virtual hardware, boot options, resource allocation, network interfaces, and more. Using an application or utility that persistently modifies a guest property, such as **virsh setmem**, writes this change into the guest's XML configuration, which the hypervisor then reads when booting the guest and modifies the virtual machine accordingly.

To display the XML configuration of a specific guest, use the **virsh dumpxml *guestname*** command.

To edit the XML configuration of a guest, use one of the following:

- The **virsh** commands - Persistent changes made to a guest virtual machine using **virsh** commands are recorded in the domain XML.
- The **virt-xml** command - This command configures the domain XML file of a specified guest according to the provided options.
- The **Virtual Machine Manager** - Changes made to a guest virtual machine in the **Virtual Machine Manager** are recorded in the domain XML.



NOTE

For the rare cases in which the domain XML files must be edited directly, use the **virsh edit *guestname***. This command opens the domain XML configuration of the specified guest in the text editor determined by root's bash configuration, by default **vi**.

To change the editor for **virsh edit**, set or modify the **EDITOR** variable in the **.bashrc** file of the intended user. For example, for the root user, this file is located in the **/root/** directory.

For the modifications performed using **virsh edit** to take effect, save the edited XML configuration and restart the guest.



WARNING

Do not edit the XML configuration of a guest by opening it as a file in a text editor, for example by using **gedit /etc/libvirt/qemu/guestname.xml**. Changes made this way do not take effect and are automatically overwritten.

For detailed information on domain XML configuration files, see the [Virtualization Deployment and Administration Guide](#).

3.4. 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.

3.4.1. 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 on 32-bit and 64-bit AMD and Intel architectures for guest operating systems Red Hat Enterprise Linux 6.2 and above.

3.4.2. 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, see 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.

3.4.3. 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.

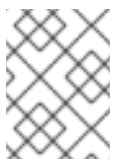


NOTE

For more information on VFIO device assignment, see 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.



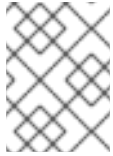
NOTE

For more information on USB, PCI, and SCSI passthrough, see 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 can be 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.

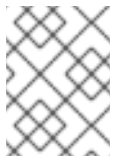
**NOTE**

For more information on SR-IOV, see 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.

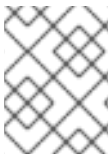
**NOTE**

For more information on NPIV, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

3.4.4. 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, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

3.5. 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.

3.5.1. 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, see 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**.

3.5.2. 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.

3.5.3. 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.

3.5.4. 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, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

3.6. 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. GETTING STARTED WITH VIRTUALIZATION COMMAND-LINE INTERFACE

The standard method of operating virtualization on Red Hat Enterprise Linux 7 is using the command-line user interface (CLI). Entering CLI commands activates system utilities that create or interact with virtual machines on the host system. This method offers more detailed control than using graphical applications such as **virt-manager** and provides opportunities for scripting and automation.

4.1. PRIMARY COMMAND-LINE UTILITIES FOR VIRTUALIZATION

The following subsections list the main command-line utilities you can use to set up and manage virtualization on Red Hat Enterprise Linux 7. These commands, as well as numerous other virtualization utilities, are included in packages provided by the Red Hat Enterprise Linux repositories and can be installed [using the Yum package manager](#).

For more information about installing virtualization packages, see the [Virtualization Deployment and Administration Guide](#).

4.1.1. virsh

virsh is a CLI utility for managing hypervisors and guest virtual machines. It is the primary means of controlling virtualization on Red Hat Enterprise Linux 7. Its capabilities include:

- Creating, configuring, pausing, listing, and shutting down virtual machines
- Managing virtual networks
- Loading virtual machine [disk images](#)

The **virsh** utility is ideal for creating virtualization administration scripts. Users without root privileges can use **virsh** as well, but in read-only mode.

Using virsh

The **virsh** utility can be used in a standard command-line input, but also as an interactive shell. In shell mode, the **virsh** command prefix is not needed, and the user is always registered as root. The following example uses the **virsh hostname** command to display the hypervisor's host name – first in standard mode, then in interactive mode.

```
$ virsh hostname
localhost.localdomain

$ virsh
Welcome to virsh, the virtualization interactive terminal.

Type: 'help' for help with commands
      'quit' to quit

virsh # hostname
localhost.localdomain
```



IMPORTANT

When using **virsh** as a non-root user, you enter an unprivileged [libvirt session](#), which means you cannot see or interact with guests or any other virtualized elements created by the root.

To gain read-only access to the elements, use **virsh** with the **-c qemu:///system** option.

Getting help with virsh

Like with all Linux bash commands, you can obtain help with **virsh** by using the **man virsh** command or the **--help** option. In addition, the **virsh help** command can be used to view the help text of a specific **virsh** command, or, by using a keyword, to list all **virsh** commands that belong to a certain group.

The **virsh** command groups and their respective keywords are as follows:

- Guest management – keyword **domain**
- Guest monitoring – keyword **monitor**
- Host and hypervisor monitoring and management – keyword **host**
- Host system network interface management – keyword **interface**
- Virtual network management – keyword **network**
- Network filter management – keyword **filter**
- Node device management – keyword **nodedev**
- Management of secrets, such as passphrases or encryption keys – keyword **secret**
- Snapshot management – keyword **snapshot**
- Storage pool management – keyword **pool**
- Storage volume management – keyword **volume**
- General **virsh** usage – keyword **virsh**

In the following example, you need to learn how to rename a guest virtual machine. By using **virsh help**, you first find the proper command to use and then learn its syntax. Finally, you use the command to rename a guest called *Fontaine* to *Atlas*.

Example 4.1. How to list help for all commands with a keyword

```
# virsh help domain
Domain Management (help keyword 'domain'):
  attach-device      attach device from an XML file
  attach-disk        attach disk device
  [...]
  domname            convert a domain id or UUID to domain name
  domrename          rename a domain
  [...]
# virsh help domrename
NAME
  domrename - rename a domain
```

SYNOPSIS

```
domrename <domain> <new-name>
```

DESCRIPTION

Rename an inactive domain.

OPTIONS

```
[--domain] <string> domain name, id or uuid
[--new-name] <string> new domain name
```

```
# virsh domrename --domain Fontaine --new-name Atlas
Domain successfully renamed
```

**NOTE**

For more information about managing virtual machines using **virsh**, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

4.1.2. virt-install

virt-install is a CLI utility for creating 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.

**IMPORTANT**

When using **virt-install** as a non-root user, you enter an unprivileged [libvirt session](#). This means that the created guest will only be visible to you, and it will not have access to certain capabilities that guests created by the root have.

**NOTE**

For more information about using **virt-install**, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

4.1.3. virt-xml

virt-xml is a command-line utility for editing domain XML files. For the XML configuration to be modified successfully, the name of the guest, the XML action, and the change to make must be included with the command.

For example, the following lists the suboptions that relate to guest boot configuration, and then turns on the boot menu on the **example_domain** guest:

```
# virt-xml boot=?
--boot options:
arch
cdrom
[...]
```

```

menu
network
nvram
nvram_template
os_type
smbios_mode
uefi
useserial
# virt-xml example_domain --edit --boot menu=on
Domain 'example_domain' defined successfully.

```

Note that each invocation of the command can perform one action on one domain XML file.



NOTE

This tool is included in the `virt-install` package. For more information about using **virt-xml**, see the **virt-xml** man pages.

4.1.4. guestfish

guestfish is a command-line utility for examining and modifying virtual machine disk images. It uses the `libguestfs` library and exposes all functionalities provided by the **libguestfs** API.

Using guestfish

The **guestfish** utility can be used in a standard command-line input mode, but also as an interactive shell. In shell mode, the **guestfish** command prefix is not needed, and the user is always registered as `root`. The following example uses the **guestfish** to display the file systems on the *testguest* virtual machine – first in standard mode, then in interactive mode.

```

# guestfish domain testguest : run : list-file systems
/dev/sda1: xfs
/dev/rhel/root: xfs
/dev/rhel/swap: swap
# guestfish

```

Welcome to guestfish, the guest filesystem shell for
editing virtual machine filesystems and disk images.

Type: 'help' for help on commands
'man' to read the manual
'quit' to quit the shell

```

><fs> domain testguest
><fs> run
><fs> list-file systems
/dev/sda1: xfs
/dev/rhel/root: xfs
/dev/rhel/swap: swap

```

In addition, **guestfish** can be used [in bash scripts](#) for automation purposes.



IMPORTANT

When using **guestfish** as a non-root user, you enter an unprivileged [libvirt session](#). This means you cannot see or interact with disk images on guests created by the root.

To gain read-only access to these disk images, use **guestfish** with the **-ro -c qemu:///system** options. In addition, you must have read privileges for the disk image files.

Getting help with guestfish

Like with all Linux bash commands, you can obtain help with **guestfish** by using the **man guestfish** command or the **--help** option. In addition, the **guestfish help** command can be used to view detailed information about a specific **guestfish** command. The following example displays information about the **guestfish add** command:

```
$ guestfish help add
NAME
  add-drive - add an image to examine or modify

SYNOPSIS
  add-drive filename [readonly:true|false] [format:...] [iface:...] [name:...] [label:...] [protocol:...] [server:...]
  [username:...] [secret:...] [cachemode:...] [discard:...] [copyonread:true|false]

DESCRIPTION
  This function adds a disk image called filename to the handle. filename
  may be a regular host file or a host device.
  [...]
```



NOTE

For more information about **guestfish**, see the [Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide](#).

4.2. DEMONSTRATION: CREATING AND MANAGING A GUEST WITH COMMAND-LINE UTILITIES

To show how virtualization tasks are performed in the CLI, this section provides a demonstration where a new guest virtual machine is created, an OS is installed on it, and the guest is afterwards interacted with and managed using CLI commands.

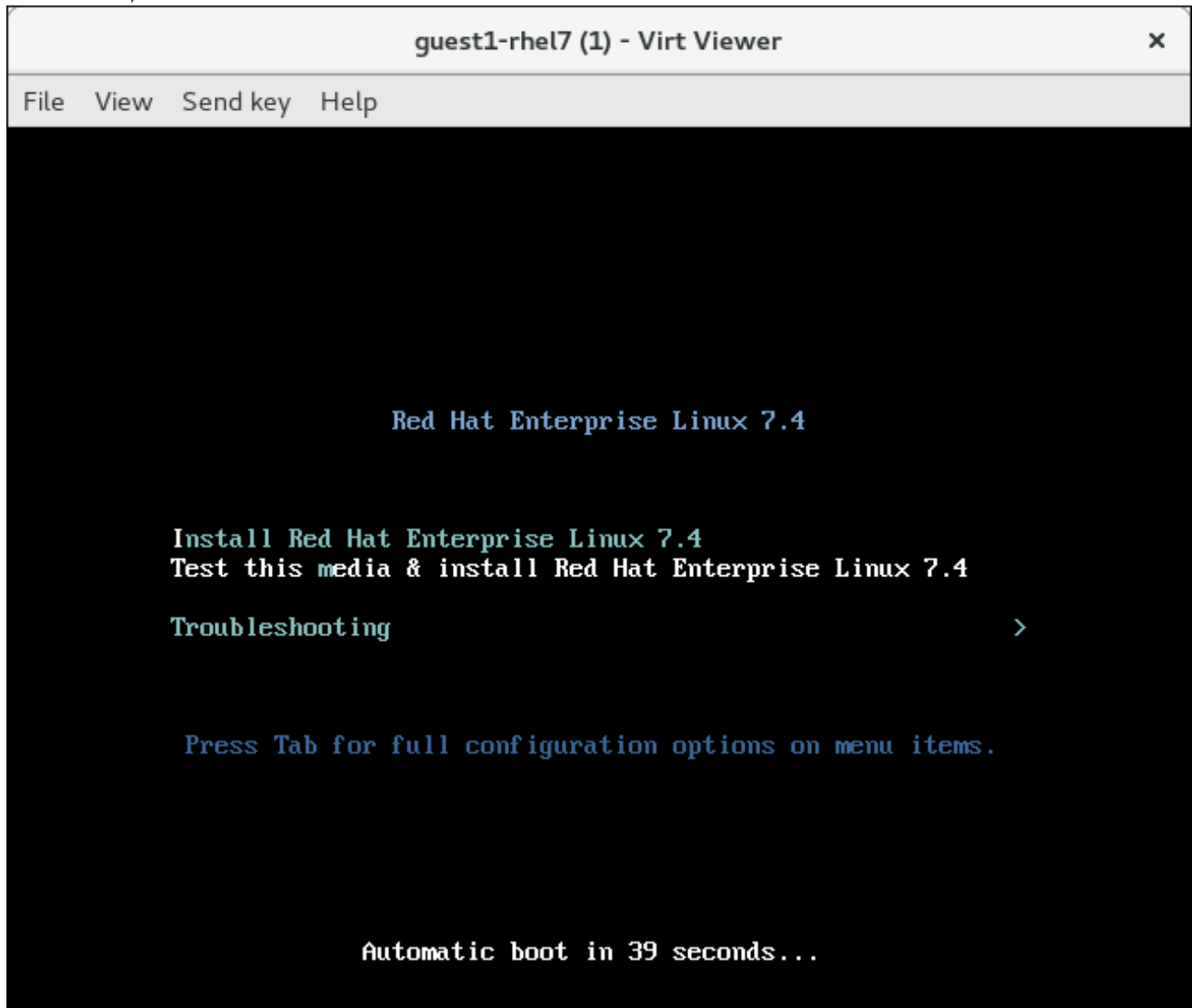
4.2.1. Installation

The following creates a new guest, here named **guest1-rhel7**, and starts an OS installation from a Red Hat Enterprise Linux 7 Workstation ISO image. This image is obtainable on the [Customer Portal](#) and in this example, it is currently located in your **~/Downloads/** folder. The guest is allocated with two virtual CPUs, 2048 MB RAM, and 8 GB of disk space.

```
# virt-install --name guest1-rhel7 --memory 2048 --vcpus 2 --disk size=8 --cdrom
/home/username/Downloads/rhel-workstation-7.4-x86_64-dvd.iso --os-variant rhel7

Starting install...
Allocating 'guest1-rhel7.qcow2' | 8.0 GB 00:00:00
```

This launches a graphical Anaconda installer in the **virt-viewer** application. For more information on the installation, see the [Installation Guide](#).



NOTE

On host systems without access to a graphical interface, it is possible to install the guest OS using [text-based Anaconda](#), using a **virt-install** command similar to the following:

```
# virt-install -name rhel7anaconda-guest -r 1024 --
location=/home/jherrman/Downloads/rhel-workstation-7.4-x86_64-dvd.iso --disk
size=8 --nographics --extra-args="console=tty0 console=ttyS0,115200n8"
```

If the installation completes successfully, the command line displays the following:

```
Domain creation completed.
Restarting guest.
```

It is now possible to make any desirable configuration to the guest. However, to manage the guest settings safely, it is advisable to shut down the guest first.

```
# virsh shutdown guest1-rhel7
Domain guest1-rhel7 is being shutdown
```


4.2.2. Attaching a Device

To make the guest detect and use a USB device connected to the host, a Samsung mobile phone in this example, start by using the **lsusb** command on the host to retrieve the device's IDs.

```
# lsusb

[...]
Bus 003 Device 007: ID 04e8:6860 Samsung Electronics Co., Ltd Galaxy (MTP)
```

Afterwards, use your text editor of choice on the host to create an XML file for the device, **samsung_USB_device.xml** in this example, where you will input the vendor and product IDs.

```
# vim samsung_USB_device.xml

<hostdev mode='subsystem' type='usb' managed='yes'>
  <source>
    <vendor id='0x04e8'>
    <product id='0x6860'>
  </source>
</hostdev>
```

Finally, use the **virsh attach-device** command to attach the device to the guest.

```
# virsh attach-device guest1-rhel7 --file samsung_USB_device.xml --config
Device attached successfully
```



NOTE

It is possible to attach devices to a running guest as well. To do so, use the **--live** option.

4.2.3. Interacting with the Guest

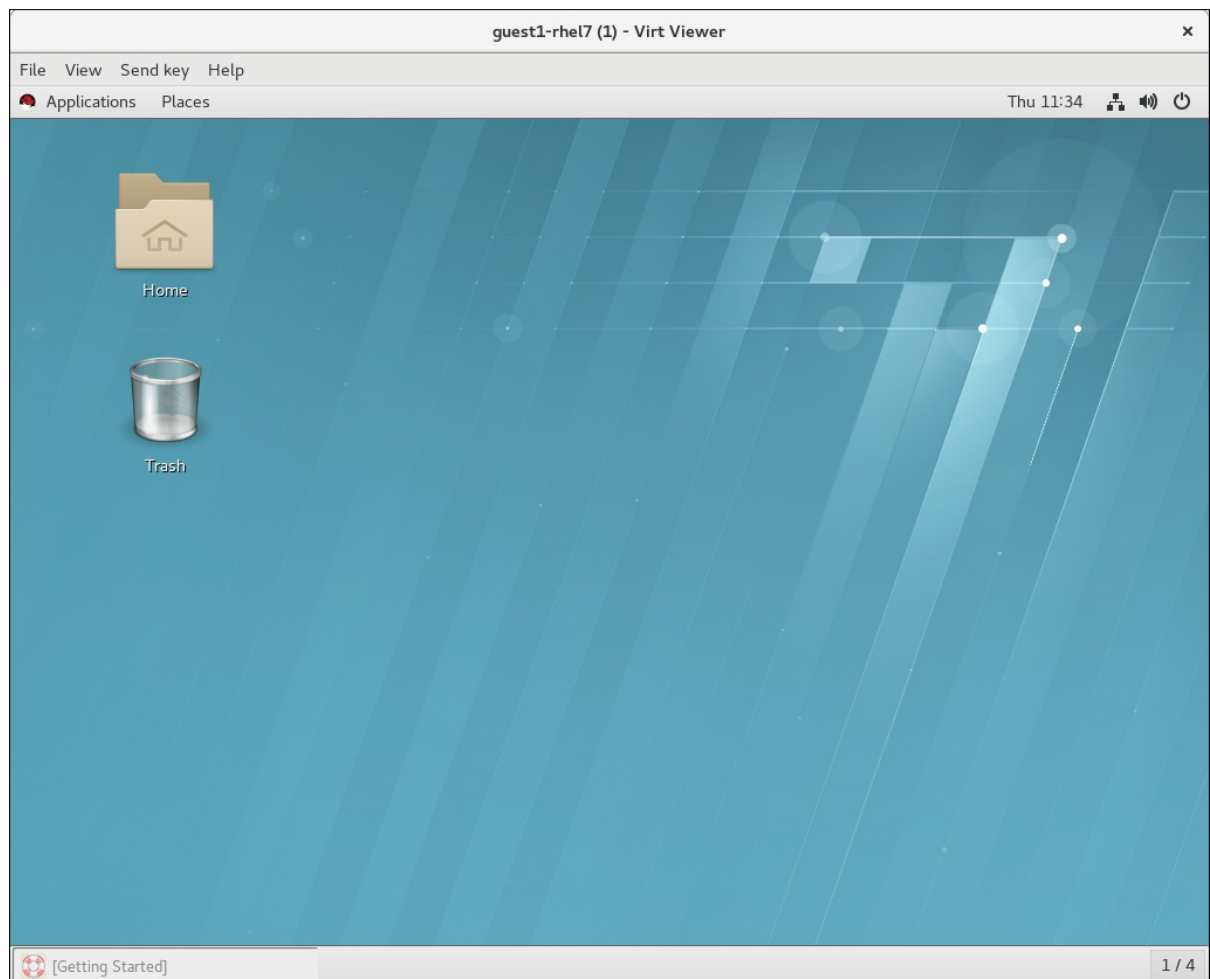
To begin using the guest1-rhel7 guest, start it first.

```
# virsh start guest1-rhel7
Domain guest1-rhel7 started
```

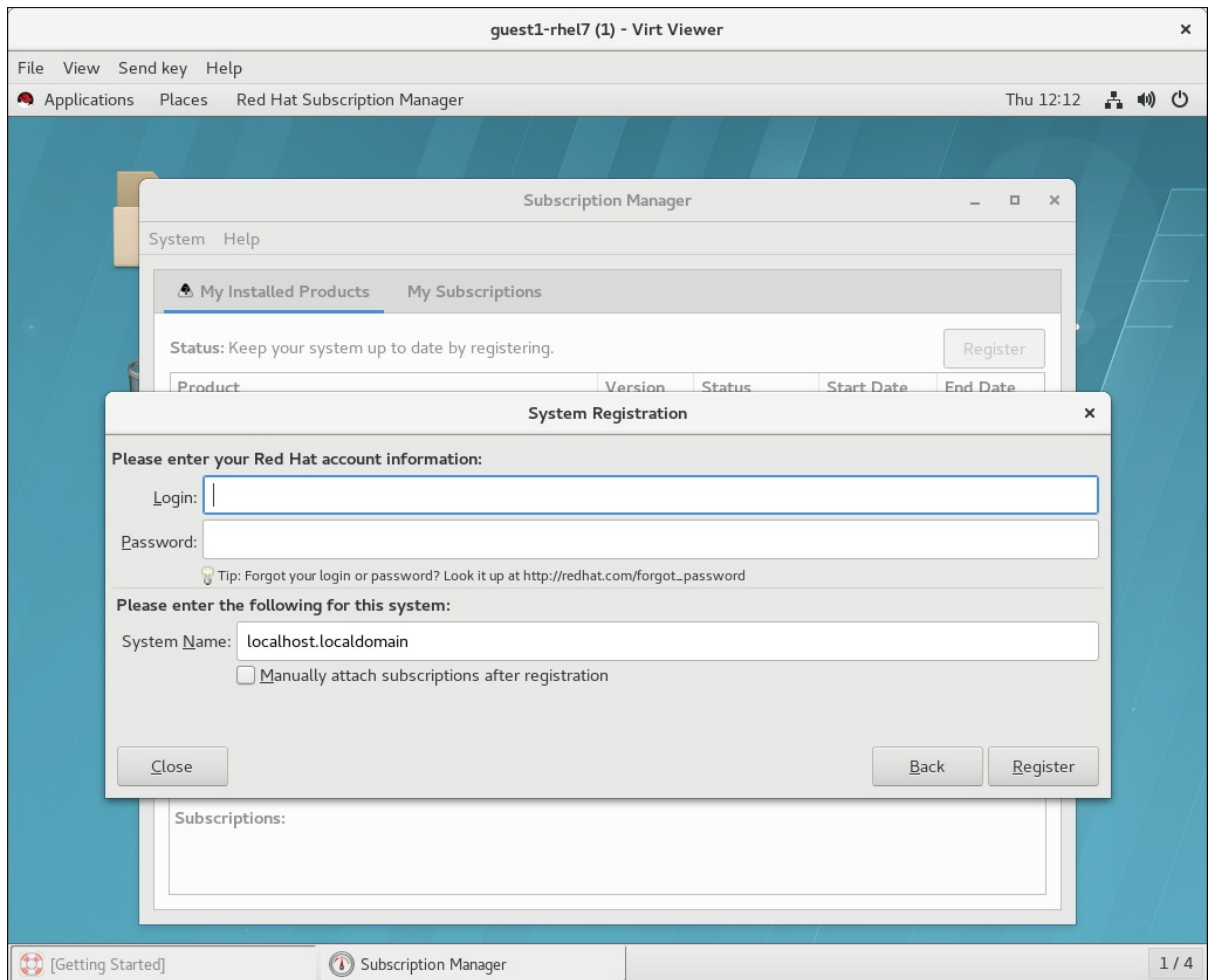
Depending on whether your host system has a graphical display, you can either interact with the guest using the **virt-viewer** application, or using an SSH shell.

- On systems with a graphical display, use **virt-viewer**:

```
# virt-viewer guest1-rhel7
```



Afterwards, you can interact with the screen output in the **virt-viewer** window like with an OS GUI on a physical machine. For example, you can use the [Subscription Manager](#) application to register your Red Hat Enterprise Linux guest OS:



- If the host or guest has a text-only interface, use SSH. This requires knowing the IP address of the guest. If you do not know the IP address, it can be obtained using the **virsh domifaddr** command.

```
# virsh domifaddr guest1-rhel7
Name      MAC address      Protocol  Address
-----
vnet0     52:54:00:65:29:21  ipv4      10.34.3.125/24
# ssh root@10.34.3.125
root@10.34.3.125's password:
Last login: Wed Jul 19 18:27:10 2017 from 192.168.122.1
[root@localhost ~]#
```



NOTE

For **virsh domifaddr** to work, the guest must be running, reachable on a network, and may require the [QEMU guest agent](#) to be activated.

Afterwards, you can interact with the host terminal as if using a terminal on the guest machine. For example, you can use the [subscription-manager](#) utility to register your Red Hat Enterprise Linux guest OS:

```
[root@localhost ~]# subscription-manager register
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: username@sample.com
```

Password:

The system has been registered with ID: 30b5e666-67f9-53bb-4b90-c2a88e5be789

4.2.4. Diagnostics

Display general information about the state of the guest:

```
# virsh dominfo guest1-rhel7
Id:          1
Name:        guest1-rhel7
UUID:        ec0c0122-fb63-4a54-b602-5cf84f5e2dfd
OS Type:     hvm
State:        running
CPU(s):       2
CPU time:     33.4s
Max memory:   2097152 KiB
Used memory:  2097152 KiB
Persistent:   yes
Autostart:    disable
Managed save: no
Security model: selinux
Security DOI:  0
Security label: unconfined_u:unconfined_r:svirt_t:s0:c102,c792 (enforcing)
```

4.2.5. Creating Snapshots

To back up the state of the guest, you can use the **virsh snapshot-create** command.

```
# virsh snapshot-create guest1-rhel7
Domain snapshot 1500563241 created
```

You can display your current snapshots and the XML setting of each one.

```
# virsh snapshot-list guest1-rhel7
Name                Creation Time          State
-----
1500563241          2017-07-20 17:07:21 +0200 shutoff

# virsh snapshot-dumpxml guest1-rhel7 1500563241
<domainsnapshot>
  <name>1500563241</name>
  <state>shutoff</state>
  <creationTime>1500563241</creationTime>
  <memory snapshot='no'/>
  <disks>
    <disk name='vda' snapshot='internal'/>
  [...]

```

This snapshot can later be loaded to revert the guest to the state saved in the snapshot.

```
# virsh snapshot-revert guest1-rhel7 --snapshotname 150056324
```

CHAPTER 5. GETTING STARTED WITH VIRTUAL MACHINE MANAGER

The Virtual Machine Manager, also known as **virt-manager**, is a graphical tool for creating and managing guest virtual machines. This chapter provides a description of the **Virtual Machine Manager** and how to run it.



NOTE

You can only run the Virtual Machine Manager on a system that has a graphical interface.

For more detailed information about using the Virtual Machine Manager, see the other [Red Hat Enterprise Linux virtualization guides](#).

5.1. RUNNING VIRTUAL MACHINE MANAGER

To run the Virtual Machine Manager, select it in the list of applications or use the following command:

```
# virt-manager
```

The Virtual Machine Manager opens to the main window.

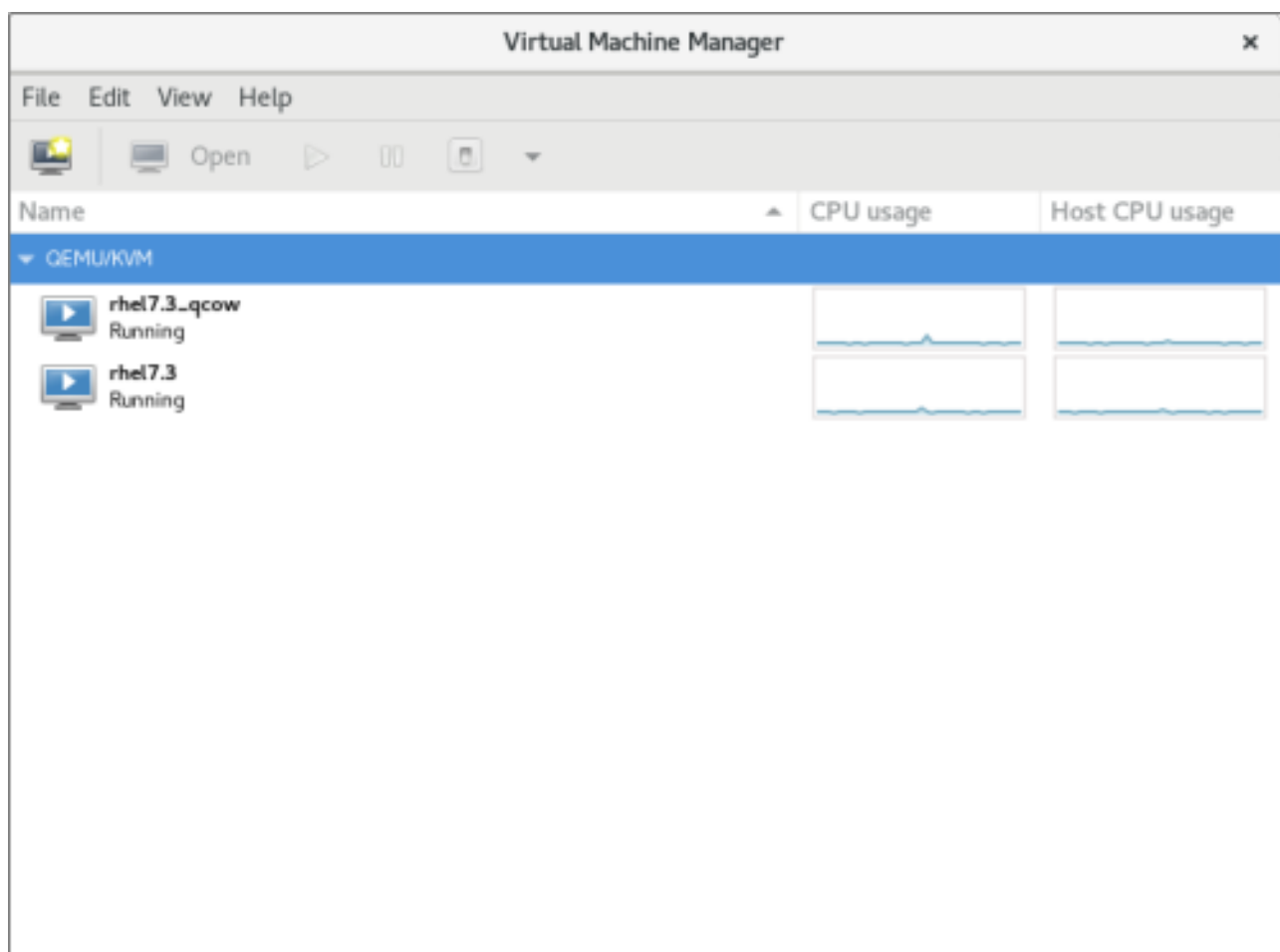


Figure 5.1. The Virtual Machine Manager



NOTE

If running **virt-manager** fails, ensure that the virt-manager package is installed. For information on installing the virt-manager package, see [Installing the Virtualization Packages](#) in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.

5.2. THE VIRTUAL MACHINE MANAGER INTERFACE

The following sections provide information about the Virtual Machine Manager user interface. The user interface includes [The Virtual Machine Manager main window](#) and [The Virtual Machine window](#).

5.2.1. The Virtual Machine Manager Main Window

This following figure shows the Virtual Machine Manager main window interface.

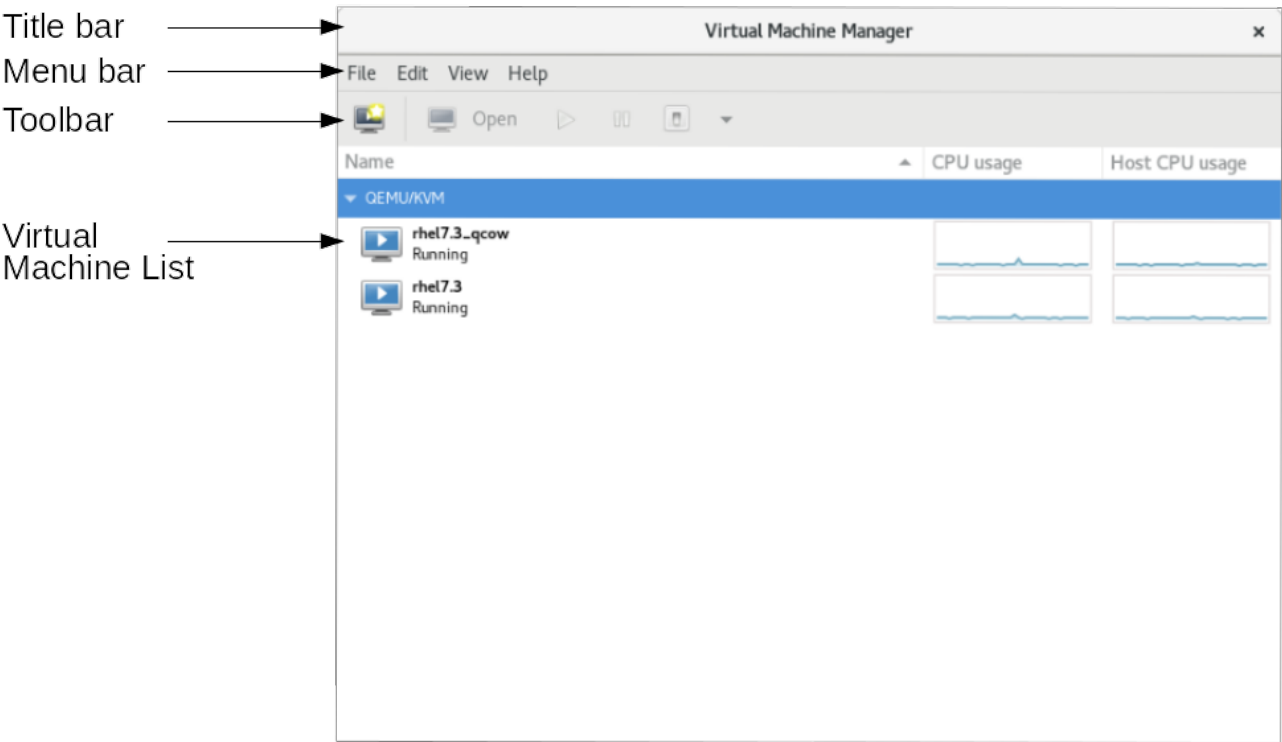


Figure 5.2. The Virtual Machine Manager window

The Virtual Machine Manager main window title bar displays **Virtual Machine Manager**.

5.2.1.1. The main window menu bar

The following table lists the entries in the Virtual Machine Manager main window menus.

Table 5.1. Virtual Machine Manager main window menus


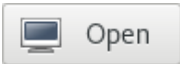




Menu name	Menu item	Description
File	Add Connection	Opens the Add Connection dialog to connect to a local or remote hypervisor. For more information, see Adding a Remote Connection in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.

Menu name	Menu item	Description
	New Virtual Machine	Opens the New VM wizard to create a new guest virtual machine. For more information, see Creating Guests with virt-manager in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.
	Close	Closes the Virtual Machine Manager window without closing any Virtual Machine windows. Running virtual machines are not stopped.
	Exit	Closes the Virtual Machine Manager and all Virtual Machine windows. Running virtual machines are not stopped.
Edit	Connection Details	Opens the Connection Details window for the selected connection.
	Virtual Machine Details	Opens the Virtual Machine window for the selected virtual machine. For more information, see The Virtual Machine pane .
	Delete	Deletes the selected connection or virtual machine.
	Preferences	Opens the Preferences dialog box for configuring Virtual Machine Manager options.
View	Graph <ul style="list-style-type: none"> ● Guest CPU Usage ● Host CPU Usage ● Memory Usage ● Disk I/O ● Network I/O 	Toggles displays of the selected parameter for the virtual machines in the Virtual Machine Manager main window.
Help	About	Displays the About window with information about the Virtual Machine Manager.

5.2.1.2. The main window toolbar

The following table lists the icons in the Virtual Machine Manager main window.

Table 5.2. Virtual Machine Manager main window toolbar

Icon	Description
	Opens the New VM wizard to create a new guest virtual machine.
	Opens the Virtual Machine window for the selected virtual machine.
	Starts the selected virtual machine.
	Pauses the selected virtual machine.
	Stops the selected virtual machine.
	<p>Opens a menu to select one of the following actions to perform on the selected virtual machine:</p> <ul style="list-style-type: none"> • Reboot - Reboots the selected virtual machine. • Shut Down - Shuts down the selected virtual machine. • Force Reset - Forces the selected virtual machine to shut down and restart. • Force Off - Forces the selected virtual machine to shut down. • Save - Saves the state of the selected virtual machine to a file. For more information, see Saving a Guest Virtual Machine's Configuration in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.

5.2.1.3. The Virtual Machine list

The virtual machine list displays a list of virtual machines to which the Virtual Machine Manager is connected. The virtual machines in the list are grouped by connection. You can sort the list by clicking on the header of a table column.







Name	CPU usage	Host CPU usage
▼ QEMU/KVM		
 rhel7.3 Running		
 rhel7.3_qcow Running		

Figure 5.3. The Virtual Machine list

The virtual machine list displays graphs with information about the resources being used by each virtual machine. You make resources available for display from the **Polling** tab of the **Preferences** dialog in the **Edit** menu. The following is a list of the resources that can be displayed in the virtual machine list:

- CPU usage
- Host CPU usage
- Memory usage
- Disk I/O
- Network I/O

You can select the resources to display using the **Graph** menu item in the **View** menu.

5.2.2. The Virtual Machine Window

This section provides information about the Virtual Machine window interface.

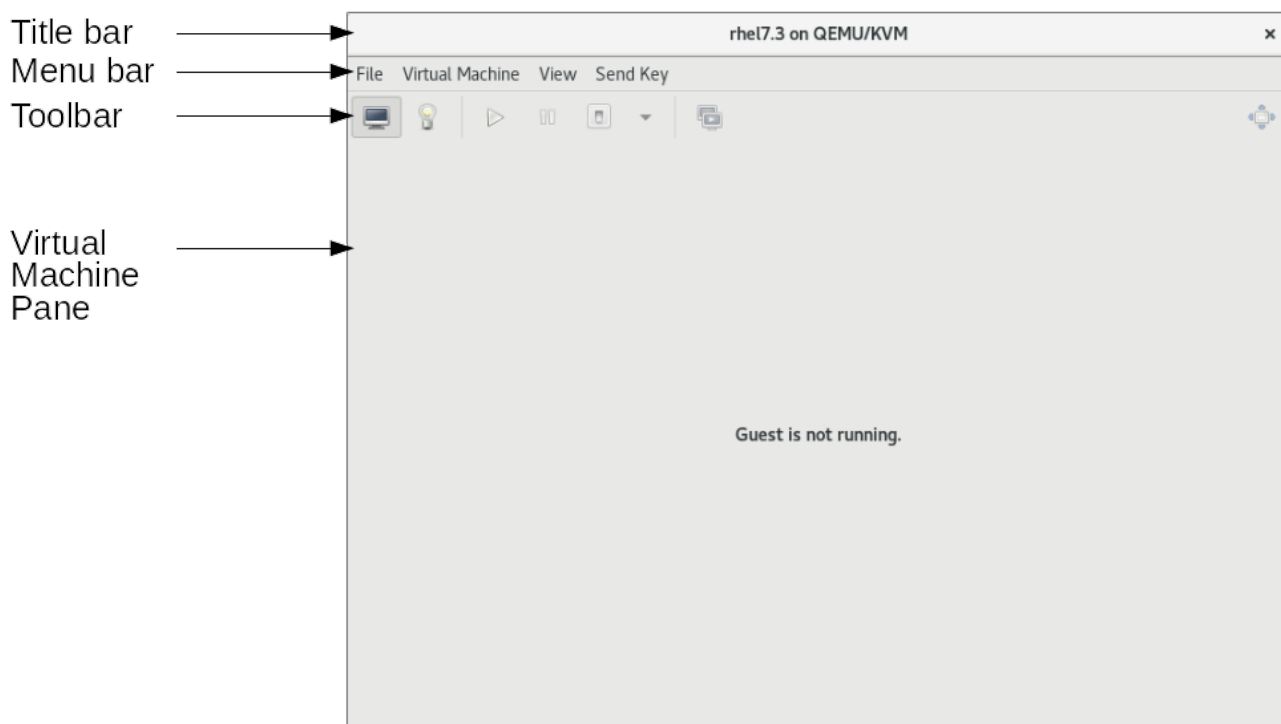


Figure 5.4. The Virtual Machine window

The title bar displays the name of the virtual machine and the connection that it uses.

5.2.2.1. The Virtual Machine window menu bar

The following table lists the entries in the Virtual Machine window menus.

Table 5.3. Virtual Machine window menus

Menu name	Menu item	Description
File	View Manager	Opens the main Virtual Machine Manager window.

Menu name	Menu item	Description
	Close	Closes only the Virtual Machine window without stopping the virtual machine.
	Exit	Closes the all Virtual Machine Manager windows. Running virtual machines are not stopped.
Virtual Machine	Run	Runs the virtual machine. This option is only available if the virtual machine is not running.
	Pause	Pauses the virtual machine. This option is only available if the virtual machine is already running.
	Shut Down	<p>Opens a menu to select one of the following actions to perform on the virtual machine:</p> <ul style="list-style-type: none"> ● Reboot - Reboots the virtual machine. ● Shut Down - Shuts down the virtual machine. ● Force Reset - Forces the virtual machine to shut down and restart. ● Force Off - Forces the virtual machine to shut down. ● Save - Saves the state of the virtual machine to a file.
	Clone	Creates a clone of the virtual machine. For more information, see Cloning Guests with virt-manager in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.
	Migrate	Opens the Migrate the virtual machine dialog to migrate the virtual machine to a different host. For more information, see Migrating with virt-manager in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.
	Delete	Deletes the virtual machine.
	Take Screenshot	Takes a screenshot of the virtual machine console.
	Redirect USB Device	Opens the Select USB devices for redirection dialog to select USB devices to redirect. For more information, see USB Redirection in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.
View	Console	Opens the Console display in the Virtual Machine pane.





Menu name	Menu item	Description
	Details	Opens the Details display in the Virtual Machine pane. For more information, see The virtual machine details window
	Snapshots	Opens the Snapshots display in the Virtual Machine pane. For more information, see The snapshots window .
	Fullscreen	Displays the virtual machine console in full screen mode.
	Resize to VM	Resizes the display on the full screen to the size and resolution configured for the virtual machine.
	Scale Display	<p>Scales the display of the virtual machine based on the selection of the following sub-menu items:</p> <ul style="list-style-type: none"> ● Always - The display of the virtual machine is always scaled to the Virtual Machine window. ● Only when Fullscreen - The display of the virtual machine is only scaled to the Virtual Machine window when the Virtual Machine window is in Full screen mode.. ● Never - The display of the virtual machine is never scaled to the Virtual Machine window. ● Auto resize VM with window - The display of the virtual machine resizes automatically when the Virtual Machine window is resized.
	Text Consoles	Displays the virtual machine display selected in the list. Examples of virtual machine displays include Serial 1 and Graphical Console Spice .
	Toolbar	Toggles the display of the Virtual Machine window toolbar.





Menu name	Menu item	Description
Send Key	Ctrl+Alt+Backspace	Sends the selected key to the virtual machine.
	Ctrl+Alt+Delete	
	Ctrl+Alt+F1	
	Ctrl+Alt+F2	
	Ctrl+Alt+F3	
	Ctrl+Alt+F4	
	Ctrl+Alt+F5	
	Ctrl+Alt+F6	
	Ctrl+Alt+F7	
	Ctrl+Alt+F8	
	Ctrl+Alt+F9	
	Ctrl+Alt+F10	
	Ctrl+Alt+F11	
	Ctrl+Alt+F12	
	Ctrl+Alt+Printscreen	

5.2.2.2. The Virtual Machine window toolbar

The following table lists the icons in the Virtual Machine window.

Table 5.4. Virtual Machine window toolbar

Icon	Description
	Displays the graphical console for the virtual machine.
	Displays the details pane for the virtual machine.
	Starts the selected virtual machine.
	Pauses the selected virtual machine.

Icon	Description
	Stops the selected virtual machine.
	<p>Opens a menu to select one of the following actions to perform on the selected virtual machine:</p> <ul style="list-style-type: none"> ● Reboot - Reboots the selected virtual machine. ● Shut Down - Shuts down the selected virtual machine. ● Force Reset - Forces the selected virtual machine to shut down and restart. ● Force Off - Forces the selected virtual machine to shut down. ● Save - Saves the state of the selected virtual machine to a file.
	Opens the Snapshots display in the Virtual Machine pane.
	Displays the virtual machine console in full screen mode.

5.2.2.3. The Virtual Machine pane

The Virtual Machine pane displays one of the following:

- [The virtual machine console](#)
- [The virtual machine details window](#)
- [The snapshots window](#)

The virtual machine console

The virtual machine console shows the graphical output of the virtual machine.

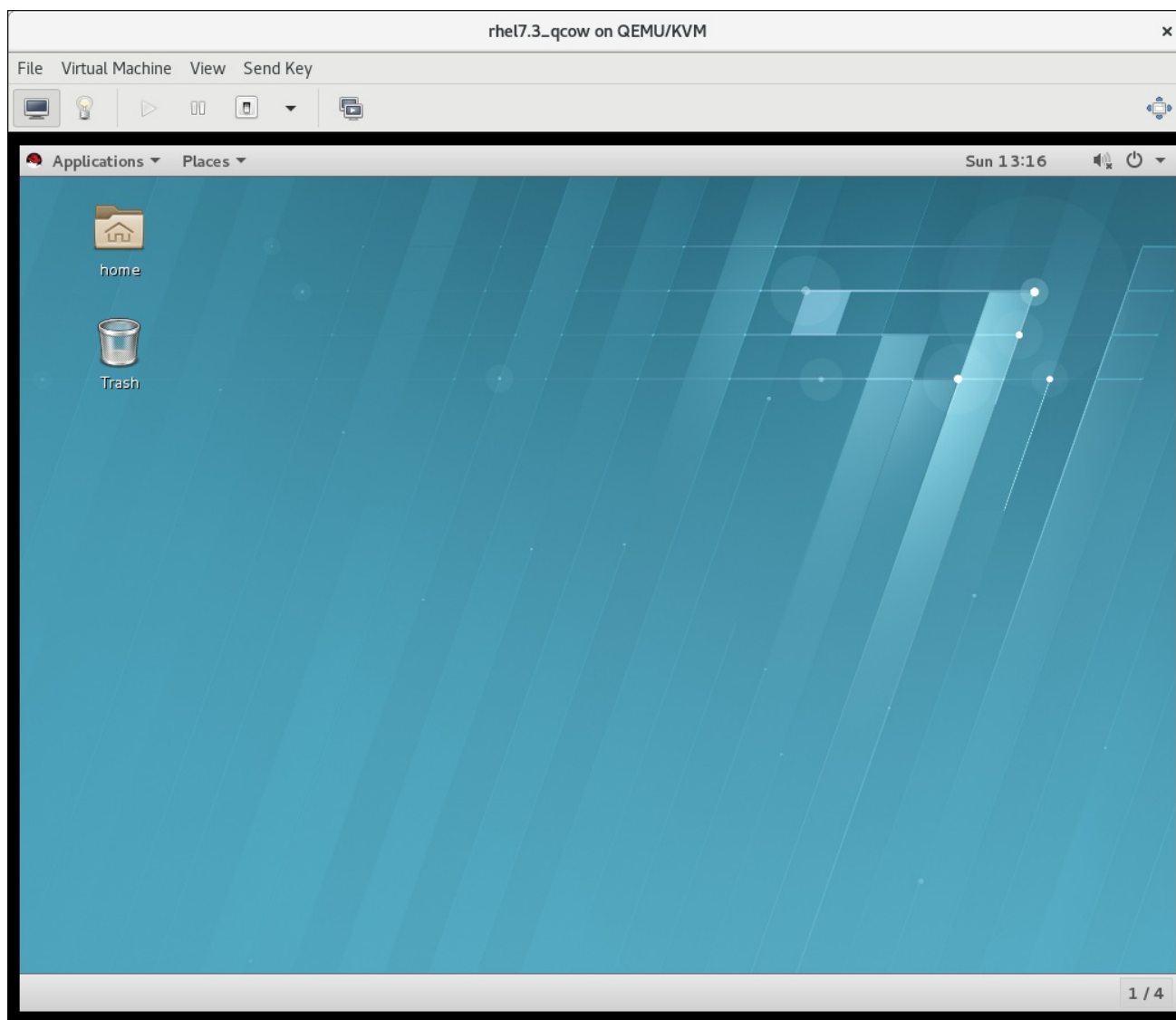


Figure 5.5. The Virtual Machine console

You can interact with the virtual machine console using the mouse and keyboard in the same manner you interact with a real machine. The display in the virtual machine console reflects the activities being performed on the virtual machine.

The virtual machine details window

The virtual machine details window provides detailed information about the virtual machine, its hardware and configuration.

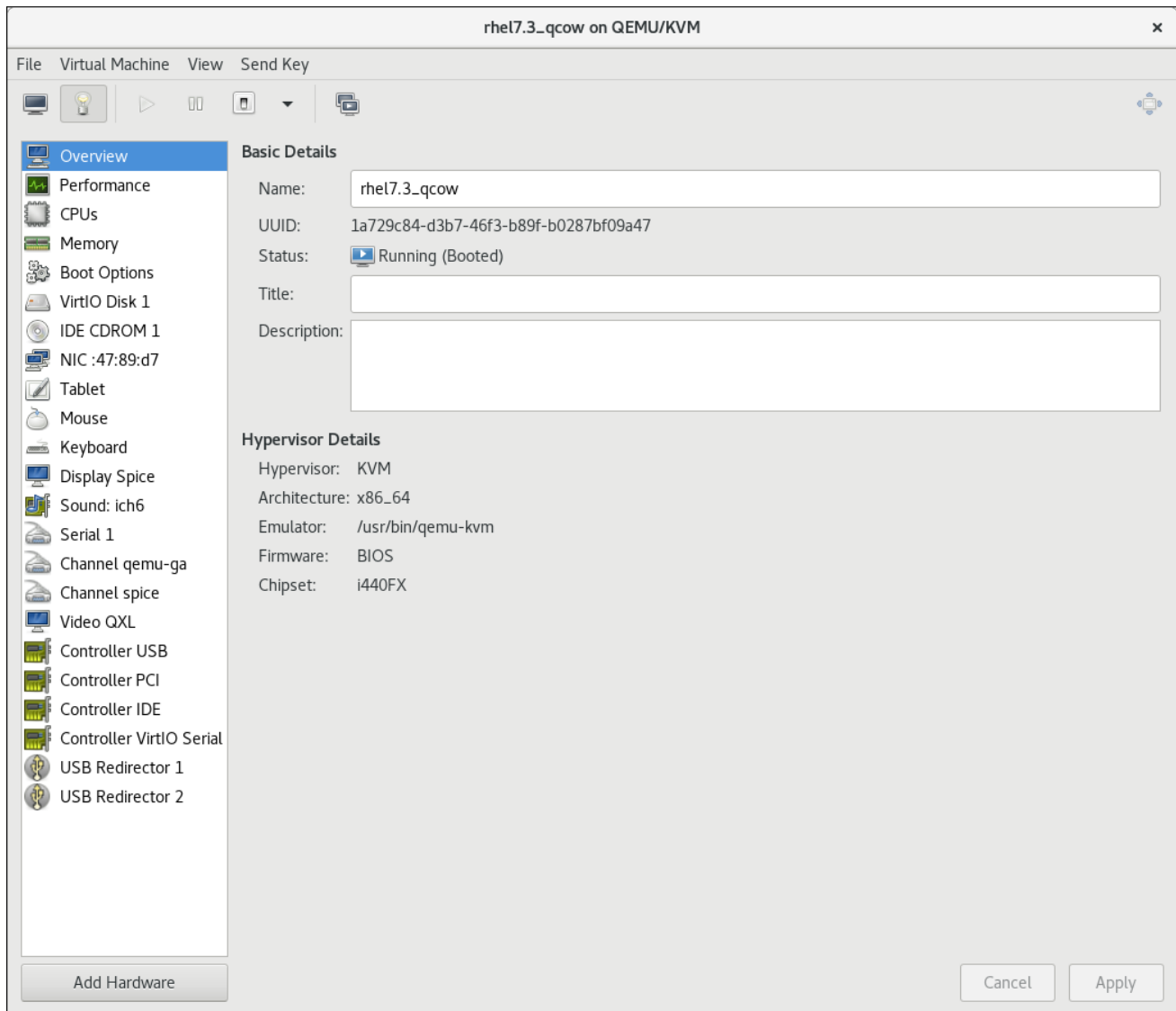


Figure 5.6. The Virtual Machine details window

The virtual machine details window includes a list of virtual machine parameters. When a parameter in the list is selected, information about the selected parameter appears on the right side of the virtual machine details window. You can also add and configure hardware using the virtual machine details window.

For more information on the virtual machine details window, see [The Virtual Hardware Details Window](#) in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.

The snapshots window

The virtual machine snapshots window provides a list of snapshots created for the virtual machine.

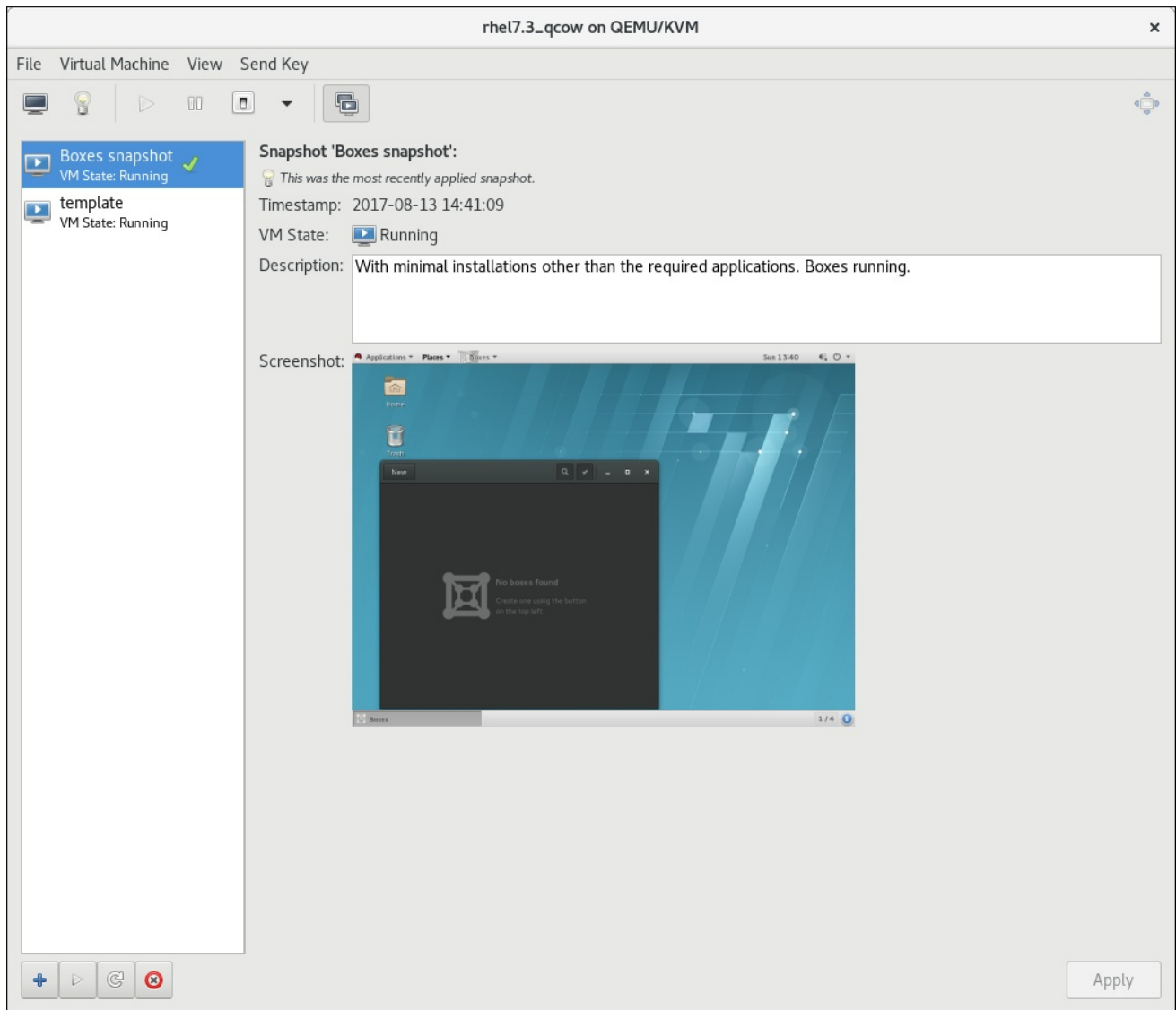


Figure 5.7. The Virtual Machine snapshots window

The virtual machine snapshots window includes a list of snapshots saved for the virtual machine. When a snapshot in the list is selected, details about the selected snapshot, including its state, description, and a screenshot, appear on the right side of the virtual machine snapshots window. You can add, delete, and run snapshots using the virtual machine snapshots window.

For more information about managing snapshots, see [Managing Snapshots](#) in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide.

APPENDIX A. REVISION HISTORY

Revision 1.0-56 Version for 7.6 GA publication	Thu May 23 2019	Jiri Herrmann
Revision 1.0-55 Version for 7.6 GA publication	Thu Oct 25 2018	Jiri Herrmann
Revision 1.0-53 Version for 7.6 Beta publication	Thu Aug 5 2018	Jiri Herrmann
Revision 1.0-52 Version for 7.5 GA publication	Thu Apr 5 2018	Jiri Herrmann
Revision 1.0-49 Version for 7.4 GA publication	Thu Jul 27 2017	Jiri Herrmann
Revision 1.0-46 Version for 7.3 GA publication	Mon Oct 17 2016	Jiri Herrmann
Revision 1.0-44 Republished the guide for several bug fixes	Mon Dec 21 2015	Laura Novich
Revision 1.0-43 Cleaned up the Revision History	Thu Oct 08 2015	Jiri Herrmann
Revision 1.0-42 Updated for the 7.2 beta release	Sun Jun 28 2015	Jiri Herrmann