SUSE Linux Enterprise Server

11 SP1 www.novell.com

January 17, 2011 | Virtualization with KVM

Virtualization with KVM

Contents

About This Manual

This manual offers an introduction to setting up and managing virtualization with KVM (Kernel-based Virtual Machine) on SUSE Linux Enterprise Server. The first part introduces KVM by describing it's requirements and Novell's support status. The second part deals with managing KVM with `libvirt`, while the last part covers management with QEMU.

Many chapters in this manual contain links to additional documentation resources. This includes additional documentation that is available on the system as well as documentation available on the Internet.

For an overview of the documentation available for your product and the latest documentation updates, refer to http://www.novell.com/documentation.

1 Available Documentation

We provide HTML and PDF versions of our books in different languages. The following manuals for users and administrators are available on this product:

Deployment Guide (↑*Deployment Guide*)
: Shows how to install single or multiple systems and how to exploit the product inherent capabilities for a deployment infrastructure. Choose from various approaches, ranging from a local installation or a network installation server to a mass deployment using a remote-controlled, highly-customized, and automated installation technique.

Administration Guide (↑*Administration Guide*)
: Covers system administration tasks like maintaining, monitoring and customizing an initially installed system.

Security Guide (↑*Security Guide*)
: Introduces basic concepts of system security, covering both local and network security aspects. Shows how to make use of the product inherent security software like Novell AppArmor (which lets you specify per program which files the program may read, write, and execute) or the auditing system that reliably collects information about any security-relevant events.

System Analysis and Tuning Guide (↑*System Analysis and Tuning Guide*)
An administrator's guide for problem detection, resolution and optimization. Find how to inspect and optimize your system by means of monitoring tools and how to efficiently manage resources. Also contains an overview of common problems and solutions and of additional help and documentation resources.

Virtualization with Xen (↑*Virtualization with Xen*)
Offers an introduction to virtualization technology of your product. It features an overview of the various fields of application and installation types of each of the platforms supported by SUSE Linux Enterprise Server as well as a short description of the installation procedure.

Storage Administration Guide
Provides information about how to manage storage devices on a SUSE Linux Enterprise Server.

In addition to the comprehensive manuals, several quick start guides are available:

Installation Quick Start (↑*Installation Quick Start*)
Lists the system requirements and guides you step-by-step through the installation of SUSE Linux Enterprise Server from DVD, or from an ISO image.

Linux Audit Quick Start
Gives a short overview how to enable and configure the auditing system and how to execute key tasks such as setting up audit rules, generating reports, and analyzing the log files.

Novell AppArmor Quick Start
Helps you understand the main concepts behind Novell® AppArmor.

Find HTML versions of most product manuals in your installed system under `/usr/share/doc/manual` or in the help centers of your desktop. Find the latest documentation updates at `http://www.novell.com/documentation` where you can download PDF or HTML versions of the manuals for your product.

2 Feedback

Several feedback channels are available:

Bugs and Enhancement Requests

For services and support options available for your product, refer to http://www.novell.com/services/.

To report bugs for a product component, please use http://support.novell.com/additional/bugreport.html.

Submit enhancement requests at https://secure-www.novell.com/rms/rmsTool?action=ReqActions.viewAddPage&return=www.

User Comments

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Use the User Comments feature at the bottom of each page in the online documentation or go to http://www.novell.com/documentation/feedback.html and enter your comments there.

3 Documentation Conventions

The following typographical conventions are used in this manual:

- `/etc/passwd`: directory names and filenames
- *`placeholder`*: replace *`placeholder`* with the actual value
- `PATH`: the environment variable PATH
- `ls`, `--help`: commands, options, and parameters
- `user`: users or groups
- Alt, Alt + F1: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* > *Save As*: menu items, buttons
- This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block.

- *Dancing Penguins* (Chapter *Penguins*, ↑Another Manual): This is a reference to a chapter in another manual.

Part I. Requirements, Limitations, and Support Status

KVM Installation and Requirements

1

KVM is a full virtualization solution for x86 processors supporting hardware virtualization (Intel VT or AMD-V). It consists of two main components: A set of Kernel modules (`kvm.ko`, `kvm-intel.ko`, and `kvm-amd.ko`) providing the core virtualization infrastructure and processor specific drivers and a userspace program (`qemu-kvm`) that provides emulation for virtual devices and control mechanisms to manage VM Guests (virtual machines). The term KVM more properly refers to the Kernel level virtualization functionality, but is in practice more commonly used to reference the userspace component.

VM Guests (virtual machines), virtual storage and networks can be managed with `libvirt`-based and QEMU tools. `libvirt` is a library that provides an API to manage VM Guests based on different virtualization solutions, among them KVM and Xen. It offers a graphical user interface as well as a command line program. The QEMU tools are KVM/QEMU specific and are only available for the command line.

1.1 Hardware Requirements

Currently, Novell only supports KVM full virtualization on x86_64 hosts. KVM is designed around hardware virtualization features included in AMD (AMD-V) and Intel (VT-x) CPUs. It supports virtualization features of chipsets, and PCI devices, such as an I/O Memory Mapping Unit (IOMMU) and Single Root I/O Virtualization (SR-IOV)).

You can test whether your CPU supports hardware virtualization with the following command:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

If this command returns no output, your processor either does not support hardware virtualization, or this feature has been disabled in the BIOS.

The following websites identify processors which support hardware virtualization:

http://wiki.xensource.com/xenwiki/HVM_Compatible_Processors
http://en.wikipedia.org/wiki/X86_virtualization

NOTE

The KVM Kernel modules will not load if the CPU does not support hardware virtualization or if this feature is not enabled in the BIOS.

The general minimum hardware requirements for the VM Host Server are the same as outlined in Section “Hardware for AMD64 and Intel 64” (Chapter 2, *Installation on x86, AMD64, Intel 64, and Itanium*, ↑*Deployment Guide*). However, additional RAM for each virtualized guest is needed. It should at least be the same amount that is needed for a physical installation. It is also strongly recommended to have at least one processor core or hyper-thread for each running guest.

1.2 Supported Guest Operating Systems

The following table lists guest operating systems tested and their support status offered by Novell. All operating systems listed are supported in both 32 and 64-bit x86 versions. Para-virtualized drivers (PV drivers) are listed where available.

Para-virtualized drivers for KVM

- `virtio-net`: Virtual network driver.
- `virtio-blk`: Virtual block device driver for para-virtualized block devices.
- `virtio-balloon`: Memory driver for dynamic memory allocation. Allows to dynamically change the amount of memory allocated to a guest.
- `kvm-clock`: Clock synchronization driver.

SUSE Linux Enterprise Server 11 SP1

Virtualization Type:	Fully virtualized
PV drivers:	kvm-clock, virtio-net, virtio-blk, virtio-balloon
Support Status:	Fully Supported

SUSE Linux Enterprise Server 10 SP3

Virtualization Type:	Fully virtualized
PV drivers:	kvm-clock, virtio-net, virtio-blk, virtio-balloon
Support Status:	Fully Supported

SUSE Linux Enterprise Server 9 SP4

Virtualization Type:	Fully virtualized
Support Status:	Fully Supported
Note:	• 32-bit Kernel: specify `clock=pmtmr` on Linux boot line • 64-bit Kernel: specify `ignore_lost_ticks` on Linux boot line

SUSE Linux Enterprise Desktop 11 SP1

Virtualization Type:	Fully virtualized

PV drivers: kvm-clock, virtio-net, virtio-blk, virtio-balloon

Support Status: Tech Preview

RedHat Enterprise Linux 4.x / 5.x

Virtualization Type: Fully Virtualized

PV drivers: See http://www.redhat.com/

Support Status: Best Effort

Note: Refer to the RHEL Virtualization guide for more information.

Windows XP SP3+ / Windows Server 2003 SP2+ / Windows Server 2008+ / Windows Vista SP1+ / Windows 7

Virtualization Type: Best Effort

PV drivers: virtio-net, virtio-blk, virtio-balloon

Support Status: Best Effort

IMPORTANT

Guest images created under a previous SUSE Linux Enterprise version are not supported.

1.2.1 Availability of para-virtualized Drivers

To improve the performance of the guest operating system, para-virtualized drivers are provided when available. Although they are not required, it is strongly recommended to use them. The para-virtualized drivers are available as follows:

SUSE Linux Enterprise Server 11 SP1
: included in Kernel

SUSE Linux Enterprise Server 10 SP3
: Available from http://drivers.suse.com/novell/Novell-virtio-drivers-2.6.27/sle10-sp3/install-readme.html. Refer to Section 5.3.1, “Adding para-virtualized Drivers During the Installation” (page 35) or Section A.1.1, “Installing Para-Virtualized Drivers for SUSE Linux Enterprise Server 10 SP3” (page 143) for installation instructions.

SUSE Linux Enterprise Server 9 SP4
: not available

RedHat
: available in RedHat Enterprise Linux 5.4 and newer

Windows
: Available from `/usr/share/qemu-kvm/win-virtio-drivers.iso`. Refer to Section A.1.2, “Installing virtio Drivers for Microsoft Windows*” (page 144) for installation instructions.

1.3 The `kvm` package

The `kvm` package provides `qemu-kvm`, the program that performs the actual emulation. In addition to the qemu-kvm program, the `kvm` package also comes with a monitoring utility (`kvm-stat`), firmware components, key-mapping files, scripts, and Windows drivers (`/usr/share/qemu-kvm/win-virtio-drivers.iso`).

Originally, the `kvm` package also provided the KVM Kernel modules. Now, these modules are included with the Kernel and only userspace components are included in the current `kvm` package.

Using the `libvirt`-based tools is the recommended way of managing VM Guests. Interoperability with other virtualization tools has been tested and is an essential part of Novell's support stance. All tools are provided by packages carrying the tool's name.

- `libvirt`: A toolkit that provides management of VM Guests, virtual networks, and storage. `libvirt` provides an API, a daemon, and a shell (`virsh`).

- Virtual Machine Manager: A graphical management tool for VM Guests.
- `vm-install`: Define a VM Guest and install its operating system.
- `virt-viewer`: An X viewer client for VM Guests which supports TLS/SSL encryption of x509 certificate authentication and SASL authentication.

Support for creating and manipulating file-based virtual disk images is provided by `qemu-img`. `qemu-img` is provided by the package `virt-utils`.

1.4 Installing KVM

KVM is not installed by default. To install KVM and all virtualization tools, proceed as follows:

1. Start YaST and choose *Virtualization > Installing Hypervisor and Tools*.
2. Select *KVM* and confirm with *Accept*.
3. Confirm the list of packages that is to be installed with *Install*.
4. Agree to set up a network bridge by clicking *Yes*. It is recommended using a bridge on a VM Host Server (virtual machine host). If you prefer to manually configure a different network setup, you can safely skip this step by clicking *No*.
5. After the setup has been finished reboot the machine as YaST suggests. Alternatively load the required kernel modules and start `libvirtd` to avoid a reboot:

```
modprobe kvm-intel # on Intel machines only
modprobe kvm-amd   # on AMD machines only
rclibvirtd start
```

2 KVM Limitations

Although virtualized machines behave almost like physical machines, some limitations apply. These affect both, the VM Guest as well as the VM Host Server system.

2.1 General Limitations

The following general restrictions apply when using KVM:

Overcommits
: KVM allows for both memory and disk space overcommit. It is up to the user to understand the implications of doing so. However, hard errors resulting from exceeding available resources will result in guest failures. CPU overcommit is also supported but carries performance implications.

Time Synchronization
: Most guests require some additional support for accurate time keeping. Where available, `kvm-clock` is to be used. NTP or similar network based time keeping protocols are also highly recommended (for VM Host Server and VM Guest) to help maintain a stable time. Running NTP inside the guest is not recommended when using the `kvm-clock` . Refer to Section 9.4, “Clock Settings” (page 80) for details.

MAC addresses
: If no MAC address is specified for a NIC, a default MAC address will be assigned. This may result in network problems when more than one NIC receives the same

MAC address. It is recommended to always assure a unique MAC address has been assigned for each NIC.

Live Migration
: Live Migration is only possible between VM Host Servers with a processor from the same vendor. Guest storage has to be accessible from both VM Host Servers.

User Permissions
: The management tools (Virtual Machine Manager, `virsh`, `vm-install`) need to authenticate with `libvirt`—see Chapter 7, *Connecting and Authorizing* (page 45) for details. In order to invoke `qemu-kvm` from the command line, a user has to be a member of the group `kvm`.

Suspending/Hibernating the VM Host Server
: Suspending or hibernating the VM Host Server system while guests are running is not supported.

2.2 Hardware Limitations

The following virtual hardware limits for guests have been tested:

max. Guest RAM size	512 GB
max. Virtual CPUs per guest	16
max. NICs per guest	8
max. Block devices per guest	4 emulated, 20 para-virtual (using `virtio-blk`)
max, number of guests	no more than 8 times the number of CPU cores in the VM Host Server

2.3 Performance Limitations

Basically, workloads designed for physical installations can be virtualized and therefore inherit the benefits of modern virtualization techniques. However, virtualization comes

at the cost of a slight to moderate performance impact. You should always test your workload with the maximum anticipated CPU and I/O load to verify if it is suited for being virtualized. Although every reasonable effort is made to provide a broad virtualization solution to meet disparate needs, there will be cases where the workload itself is unsuited for KVM virtualization.

We therefore propose the following performance expectations for guests performance to be used as a guideline. The given percentage values are a comparison of performance achieved with the same workload under non-virtualized conditions. The values are rough approximations and cannot be guaranteed.

Category	**Fully Virtualized**	**Paravirtualized**	**Host-Passthrough**
CPU, MMU	7%	not applicable	97% (Hardware Virtualization with Extended Page Tables(Intel) or Nested Page Tables (AMD) 85% (Hardware Virtualization with shadow page tables)
Network I/O (1GB LAN)	20% (Realtek emulated NIC)	75% (`virtio-net`)	95%
Disk I/O	40% (IDE emulation)	85% (`virtio-blk`)	95%
Graphics (non-accelerated)	50% (VGA or Cirrus)	not applicable	not applicable
Time accuracy (worst case, using recommended settings without NTP)	95% - 105% (where 100% = accurate)	100% (`kvm-clock`)	not applicable

3 KVM Support Status

The following list contains features and tools as supported by Novell—this does not necessarily reflect the support status of the software itself. For a list of `qemu-kvm` command switches supported by Novell, refer to Section A.3, "QEMU Command Line Options" (page 150).

3.1 Supported Features and Tools

`vm-install`
: Define and install VM Guests via `vm-install` including specifying RAM, disk type and location, video type, keyboard mapping, NIC type, binding, MAC address, and boot method.

  Restrictions: Currently only the `raw` disk format is supported. NIC creation is restricted to using Realtek or Virtio NICs.

Virtual Machine Manager
: Manage guests via Virtual Machine Manager using the following functions: autostart, start, stop, restart, pause, unpause, save, restore, clone, migrate, special key sequence insertion, guest console viewers, performance monitoring, and CPU pinning. Furthermore, static modifications of CPU, RAM, boot method, disk, NIC, mouse, display, video and host PCI assignments are supported.

  Restrictions: The following features are currently not supported: sound devices, vmvga (vmware), Xen video, and adding physical USB devices.

`virsh`
: Manage guests via the command line.

  Restrictions: Requires XML descriptions as created by `vm-install` or `virt-manager`. Altering these descriptions via `virsh edit` is not supported. The supported `virsh` functionality is restricted to life cycle functions.

`qemu-kvm`
: Manage guests via the command line. Although managing via Virtual Machine Manager should be the preferred option, `qemu-kvm` may be used for greater flexibility. See Section A.3.1, “Supported `qemu-kvm` Command Line Options” (page 150) for a list of supported options.

  Restrictions: See Section A.3.2, “Unsupported `qemu-kvm` Command Line Options” (page 152) for a list of not supported options.

`kvm_stat`
: Debugging and monitoring tool.

  Restrictions: none.

Kernel Samepage Merging (KSM)
: When enabled on the VM Host Server Kernel, KSM allows for automatic sharing of identical memory pages between guests to save host memory.

  Restrictions: none.

PCI Passthrough
: PCI Passthrough improves performance of PCI devices. It requires a AMD CPU with IOMMU (I/O Memory Mapping Unit) or an Intel CPU with VT-d (Virtualization Technology for Directed I/O). VT-d requires the Kernel parameter "intel_iommu=on". Many PCIe cards from major vendors should be supportable. Refer to system level certifications for specific details, or contact the vendor for support statements.

Non-Uniform Memory Access (NUMA)
: NUMA machines are supported. Using numactl to pin `qemu-kvm` processes to specific nodes is recommended.

3.2 Unsupported Features and Tools

CPU hotplugging
: Dynamically changing the number of virtual CPUs assigned to the guest is currently not supported.

Device hotplugging
: Dynamically adding or removing devices in the guest is currently not supported.

KVM Kernel Module Parameters
: Specifying parameters for the KVM Kernel modules is currently not supported unless done under the direction of Novell support personnel.

Using QEMU without KVM
: `qemu-kvm` can be invoked with the `-no-kvm` parameter. In this case guest CPU instructions are emulated instead of being executed directly by the processor. This mode is not supported, but may be useful for problem resolution.

Part II. Managing Virtual Machines with `libvirt`

4 Overview

`libvirt` is a library that provides a common API for managing popular virtualization solutions, among them KVM and Xen. The library provides a normalized management API for these virtualization solutions, allowing a stable, cross-hypervisor interface for higher-level management tools. The library also provides APIs for management of virtual networks and storage on the VM Host Server. The configuration of each VM Guest is stored in an XML file.

With `libvirt` you can also manage your VM Guests remotely. It supports TLS encryption and x509 certificates as well as authentication with SASL.

The communication between the virtualization solutions and `libvirt` is managed by the daemon `libvirtd`. It is also used by the management tools. `libvirtd` needs to run on the VM Host Server and on any remote machine on which the `libvirt`-based tools are started. Use the following commands to start, stop it or check its status:

```
~ # rclibvirtd start
Starting libvirtd                              done
~ # rclibvirtd status
Checking status of libvirtd                 running
~ # rclibvirtd stop
Shutting down libvirtd                         done
~ # rclibvirtd status
Checking status of libvirtd                  unused
```

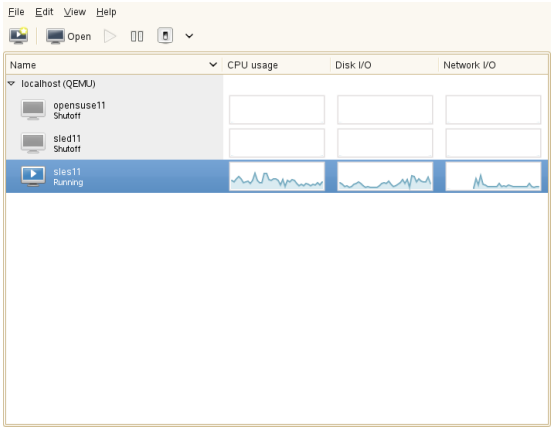
To automatically start `libvirtd` at boot time, either activate it using the YaST *System Services (Runlevel)* module or by entering the following command:

```
insserv libvirtd
```

The following `libvirt`-based tools are available on SUSE Linux Enterprise Server:

Virtual Machine Manager (`virt-manager`)

The Virtual Machine Manager is a desktop tool for managing VM Guests. It provides the ability to control the life cycle of existing machines (bootup/shutdown, pause/resume, suspend/restore). It lets you create new VM Guests and various types of storage, and manage virtual networks. Access the graphical console of VM Guests with the built-in VNC viewer, and view performance statistics, all done locally or remotely.

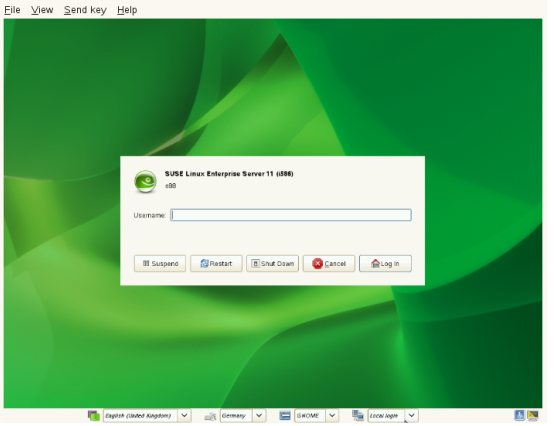


The Virtual Machine Manager does not need to run on the VM Host Server, it also lets you control VM Guests via remote connections. This enables you to manage VM Guests centrally from a single workstation without having to log in on the VM Host Server.

To start the Virtual Machine Manager, enter `virt-manager` at the command prompt.

`virt-viewer`

A viewer for the graphical console of a VM Guest. It uses the VNC protocol and supports TLS and x509 certificates. VM Guests can be accessed by name, ID, or UUID. If the guest is not already running, the viewer can be told to wait until the guest starts, before attempting to connect to the console.



`vm-install`
: A tool to set up a VM Guest, configure its devices and start the operating system installation. Starts a GUI wizard when called from a graphical user interface. When invoked on a terminal, starts the wizard in command-line mode. `vm-install` is also started when creating a new virtual machine in the Virtual Machine Manager.

`virsh`
: A command line tool to manage VM Guests with similar functionality as the Virtual Machine Manager. Allows you to change a VM Guest's status (start, stop, pause, etc.) to set up new guests and devices and to edit existing configurations. `virsh` is also useful to script VM Guest management operations.

  `virsh` basically works like Subversion's `svn` command or `zypper`: it takes the first arguments as a command and further arguments as options to this command:

  ```
  virsh [-c URI] command domain-id [OPTIONS]
  ```

  Just like `zypper`, `virsh` can also be called without a command. In this case it starts a shell waiting for your commands. This mode is useful when having to run subsequent commands:

  ```
  ~> virsh -c qemu+ssh://root@mercury.example.com/system
  Enter passphrase for key '/home/wilber/.ssh/id_rsa':
  Welcome to virsh, the virtualization interactive terminal.

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

```
virsh # hostname
mercury.example.com
```

5 Guest Installation

A VM Guest is comprised of an image containing an operating system and data files and a configuration file describing the VM Guest's virtual hardware resources. VM Guests are hosted on and controlled by the VM Host Server.

5.1 Guest Installation with Virtual Machine Manager

Clicking *New* in the Virtual Machine Manager launches `vm-install`. It provides the graphical *Create Virtual Machine Wizard* that guides you through the guest installation. `vm-install` can also be run directly from the command-line or from YaST by choosing *Virtualization* > *Create Virtual Machine*.

1 Start the *Create Virtual Machine Wizard* as described above and click *Forward*.

2 Choose whether to install an operation system or an already existing image or disk.

3 Select the operating system you want to install from the list. Each entry provides reasonable defaults for the chosen operating system.

4 The *Summary* page shows the default configuration for the chosen operating system. Edit the configuration by clicking on a headline. When having chosen to install a system, you at least have to specify either an image or a CD/DVD device from which to boot or choose PXE boot. When accepting the configuration with *OK*, the guest system boots to start the installation.

5.1.1 Customizing the Default Settings

Change the proposed configuration by clicking on a headline in the Summary page of the Create Virtual Machine Wizard:

Name of Virtual Machine

Specify a *Name* and an optional *Description* for the guest. The *Name* must contain only alphanumeric and `_-.:+` characters. It must be unique among all VM Guests on the VM Host Server. It is used to create and name the guest's configuration file and you will be able to access the guest with this name from `virsh`.

Hardware

Change memory and CPU assignments in this screen. It is recommended not to specify values larger than the resources the VM Host Server can provide (overcommit), since it may result in fatal errors or performance penalties.

The *Advanced Settings* let you activate or deactivate ACPI, APIC, and PAE. It is recommended not to change the default settings. You can also enable or disable para-virtualization with `virtio` here.

IMPORTANT: Enabling *virtio* (Para-Virtualization)

If you enable para-virtualization by activating *virtio*, all hard disks you create will be configured as `virtio` disks. If your operating system does not have appropriate drivers, the installation will fail. A Windows operating system installation even fails if providing a driver. By default, this feature is only activated for operating systems known to ship with `virtio` drivers.

Overwriting the default by activating para-virtualization for non-supported operating system currently only makes sense for SUSE Linux Enterprise Server SP3, if you plan to add the para-virtualized drivers during the installation (see Section 5.3.1, "Adding para-virtualized Drivers During the Installation" (page 35) for installation instructions).

Graphics and Keyboard

Configure the type of virtualized graphics hardware and the keymap in this dialog. If you disable the graphics card support, the machine is only accessible via network services (ssh) or serial port.

Disks

Disks: Manage virtual hard disks and CD/DVD drives in this dialog. A VM Guest must have at least one virtual disk—either an existing one or a newly created disk. Virtual disks can be:

- a single file with a fixed size
- a single file that grows on demand (`Sparse Image File`)

IMPORTANT: Sufficient Space for Sparse Image Files

When creating sparse image files, the partition on which you create them always needs sufficient free space. The VM Guest has no means to check the VM Host Server disk space. Having no space left on the host partition causes write errors and loss of data on the guest system.

- a block device, such as an entire disk, partition, or a network volume.

For best performance, create each virtual disk from an entire disk or a partition. For the next best performance, create an image file but do not create it as a sparse image file. A virtual disk based on a sparse image file delivers the most disk space flexibility but slows installation and disk access speeds.

TIP: Live Migration

If you need to be able to migrate your VM Guest to another host without shutting it down (live migration), all disks must reside on a network resource (network file system or iSCSI volume) that is accessible from both hosts.

By default, a single, file-backed virtual disk is created as a sparse image file in `/var/lib/kvm/images/`*`VM_NAME`*`/` where *`VM_NAME`* is the name of the virtual machine.

NOTE: Supported Disk format

Currently, only the raw disk format is supported by Novell.

Procedure 5.1 *Creating a Virtual Disk*

1. Click *Hard disk*.

2. Enter a *Source*. If creating a file-backed disk, either enter the path directly or click *New*. When creating a disk from a device, enter the device node, for example `/dev/disk/by-path/`*`path`*. It is strongly recommended not to use the simple device paths such as `/dev/sdb` or `/dev/sda5`, since they may change (by adding a disk or by changing the disk order in the BIOS).

3. Specify the *Protocol*. Choose *file* for file-backed virtual disks and *phy* for device-backed disks. Choosing those protocols will create the disk in `raw` format.

4. Enter a *Size* in GB. This option is only available for file-backed disks.

5. Choose whether to create a *Sparse Image File*. This option is only available for file-backed disks. If you want to disable write-access to the disk, choose *Read-Only Access*.

If you want to install from DVD or CD-ROM, add the drive to the list of available hard disks. To learn about device nodes of the available optical drives, run:

```
hwinfo --cdrom | egrep "(Device File:|Model:)"
```

Instead of the real DVD or CD-ROM drive, you can also add the ISO image of an installation medium. Note that each CD-Rom drive or ISO image can only be used by one guest at the same time.

To add a CD/DVD-ROM device or an ISO image, proceed as follows:

1. Click *CD-ROM*.

2. Enter a *Source*. If adding a device, enter its node. If adding an ISO image, either enter the path directly or click *Browse* to open a file browser.

3. Specify the *Protocol*. Choose *file* for an ISO image and *phy* for a device.

The disks are listed in the order in which they have been created. This order also represents the boot order. Use the *Up* and *Down* arrows to change the disk order.

Network Adapters

By default, a single virtual network card is created for the virtual machine. It has a randomly generated MAC address that you can change to fit your desired configuration. The virtual network card will be attached to a default bridge configured on the VM Host Server. You can adjust the configuration in this dialog or create additional virtual network cards.

To add a new network adapter or edit an existing one, proceed as follows:

1 Click *New* to add a card or *Edit* to change the configuration of the selected card.

2 Choose a *Type* from the drop-down list.

NOTE: Supported Virtual Network Adapter Types

Currently, only *Fully Virtualized Realtek 8139* or the paravirtualized *QEMU Virtualized NIC Card* (`virtio-net`) are supported by Novell.

---

3 Choose a *Source* from the drop-down list.

4 Choose whether to assign a randomly generated MAC address or manually specify an address.

NOTE: MAC addresses need to be unique

When choosing to manually specify a MAC address, make sure it is not already used in your network. If so, it may result in network problems, especially when using DHCP. Therefore avoid specifying MAC addresses such as 52:54:00:12:34:56 or 52:54:00:11:22:33. It is strongly recommended to always use a randomly generated MAC address for each adapter.

Operating System Installation

This dialog is only available when having chosen to install an operating system. The installation can be booted from a virtual disk, from a CD/DVD device, from an ISO image or via PXE boot over the network— use this dialog to configure the boot device.

Also use this dialog to configure the behavior of the VM Guest when the operating system is powered off, rebooted or if it crashes. The following options are available

destroy
: normal cleanup

restart
: a new VM Guest is started in place of the old one

preserve
: no cleanup, do not delete temporary, configuration and image files

rename-restart
: the VM Guest is not cleaned up but is renamed and a new domain started in its place

coredump-destroy
: a crashed machine's core is dumped before a normal cleanup is performed

coredump-restart
: a crashed machine's core is dumped before a normal restart is performed

5.2 Installing from the Command Line with vm-install

If `$DISPLAY` is not set (for example, when operating on a console or on an ssh shell with no X-forwarding), `vm-install` offers a command-line wizard to interactively set up a VM Guest for installation. Once the setup is completed, the newly created guest boots into the installation system which can be connected via VNC.

IMPORTANT: Graphical User Interface needed for Installation

Once the VM Guest boots into the installation, you need to connect to the graphical console via VNC to attend the installation. Therefore, you need to start the viewer from a graphical user interface.

If you are working from a console with no access to a graphical user interface, you can set up the VM Guest configuration and start the installation at a later

time. Refer to Section 5.2.1, “Defining a VM Guest without Starting the Installation” (page 34) for instructions.

To start the wizard, just type `vm-install` to start. For a lot of parameters, the installation wizard already provides reasonable defaults which you can confirm by just pressing Enter. Here is a log of an interactive setup for a SUSE Linux Enterprise Server 11 installation:

Example 5.1 *Interactive Setup on the Command Line Using vm-install*

```
~ # vm-install
Gathering settings...

Please specify the type of operating system that will run within the virtual
machine.  This defines many defaults, and helps decide how to start
paravirtualized operating systems.
Press 'q' or the Escape key to exit.
  1: Novell Open Enterprise Server 2 (Linux)
  2: Novell Open Enterprise Server 2 (NetWare)
  3: Other operating system
  4: PXE
  5: RedHat (other)
  6: RedHat Enterprise Linux 3
  7: RedHat Enterprise Linux 4
  8: RedHat Enterprise Linux 5
  9: SUSE (other)
 10: SUSE Linux Enterprise Desktop 10
 11: SUSE Linux Enterprise Desktop 11
 12: SUSE Linux Enterprise Server 8
 13: SUSE Linux Enterprise Server 9
 14: SUSE Linux Enterprise Server 10
 15: SUSE Linux Enterprise Server 11
 16: Solaris 9 and older
 17: Solaris 10
 18: Windows (other)
 19: Windows (other, x64)
 20: Windows NT
 21: Windows Server 2008
 22: Windows Server 2008 (x64)
 23: Windows Vista, Windows 7
 24: Windows Vista, Windows 7 (x64)
 25: Windows XP, 2000, 2003
 26: Windows XP, 2003 (x64)
 27: openSUSE
 28: openSUSE 11
[15] >

PXE Boot
(Y / N) [N] >
Please choose a name for the virtual machine.
[sles11] >

Description > SLES 11 SP1

Specify the amount of memory and number of processors to allocate for the VM.
Initial Memory [512] >
Maximum Memory [512] > 768
Warning: Setting the maximum memory greater than the initial memory requires
the VM operating system to have a memory balloon driver.
Virtual Processors [2] >
```

```
Please specify the type of virtualized graphics hardware.
  1: Cirrus Logic GD5446 VGA
  2: No Graphics Support
  3: VESA VGA
[1] >

Virtual Disks:
  (None)
Do you want to add another virtual disk?
(Y / N) [Y] >

Create a virtual disk based on a device (CD or other block device), an existing
image file (ISO), or a new file.  Specify a device by its device node, such
as
/dev/cdrom, not its mount point.
What type of virtual disk do you want to add?
  1: CD-ROM or DVD
  2: Floppy
  3: Hard Disk
[3] > 3
Where will the virtual disk physically reside?
[/var/lib/kvm/images/sles11/hda] >
Size (GB) [4.0] > 8.0
Create a sparse image file for the virtual disk?
(Y / N) [Y] >

Virtual Disks:
  8.0 GB Hard Disk (file:/var/lib/kvm/images/sles11/hda)
Do you want to add another virtual disk?
(Y / N) [N] > y

Create a virtual disk based on a device (CD or other block device), an existing
image file (ISO), or a new file.  Specify a device by its device node, such
as
/dev/cdrom, not its mount point.
What type of virtual disk do you want to add?
  1: CD-ROM or DVD
  2: Floppy
  3: Hard Disk
[3] > 1
Where will the virtual disk physically reside?
[/var/lib/kvm/images/sles11/hdb] > /isos/SLES-11-SP1-CD-i386-GM-CD1.iso

Virtual Disks:
  8.0 GB Hard Disk (file:/var/lib/kvm/images/sles11/hda)
  2.9 GB CD-ROM or DVD (file:/isos/SLES-11-SP1-DVD-x86_64-GM-DVD1.iso)
Do you want to add another virtual disk?
(Y / N) [N] >

Network Adapters
  (None)
Do you want to add another virtual network adapter?
(Y / N) [Y] >
```

```
What type of virtual network adapter do you want to add?
  1: Fully Virtualized AMD PCnet 32
  2: Fully Virtualized Intel e100
  3: Fully Virtualized Intel e1000
  4: Fully Virtualized NE2000 (ISA Bus)
  5: Fully Virtualized NE2000 (PCI Bus)
  6: Fully Virtualized Realtek 8139
  7: Paravirtualized
[6] > 7

Network Adapters
  Paravirtualized; Randomly generated MAC address
Do you want to add another virtual network adapter?
(Y / N) [N] >

Preparing to start the installation...

Installing...
```

You may also provide parameters on the command line. The wizard will then prompt you for any missing parameters. In the following all parameters from Example 5.1, "Interactive Setup on the Command Line Using `vm-install`" (page 31) for which a command line switch exists, are specified. See `man 8 vm-install` for a full list of parameters.

Example 5.2 *vm-install command line switches*

```
vm-install --os-type sles11❶ --name "sles11_test"❷ \
--vcpus 2❸ --memory 512❹ --max-memory 768❺ \
--disk /var/lib/kvm/images/sles11/hda,0,disk,w,8000,sparse=1 ❻ \
--disk /iso/SLES-11-SP1-DVD-x86_64-GM-DVD1.iso,1,cdrom❼ \
--nic mac=52:54:00:05:11:11,model=virtio❽ \
--graphics cirrus❾ --config-dir "/etc/libvirt/qemu"❿
```

❶ Specifies the guest operating system to define suitable defaults. A list of valid values can be obtained with `vm-install -O`.

❷ Name of the VM Guest. This name must be unique.

❸ Number of virtual processors.

❹ Initial amount of memory.

❺ Maximum amount of memory. Requires an operating system with a para-virtualized *virtio-balloon* driver.

❻ Defines a virtual hard disk. The file is located at `/var/lib/kvm/images/sles11/hda`. It is configured as the first (`0`) hard disk (`disk`). It is writable

(w) with a size of 8 GB (8000). The file on the VM Host Server is a sparse file (sparse=1).

❼ Defines an ISO image for a CD-ROM as the second (1) block device.

❽ Configures a para-virtualized network device with the MAC address 52:54:00:05:11:11.

❾ Specifies the graphics card.

❿ The directory in which the XML configuration file for the virtual machine will be stored. It is strongly recommended to use the default directory /etc/libvirt/qemu.

IMPORTANT: No Automated and Non-Interactive Setup

An automated installation in the background utilizing, for example, AutoYaST or a NetWare Response File with vm-install --os-settings is currently not supported with KVM (it only works for Xen PV guests).

The non-interactive setup of a VM Guest with vm-install --vm-settings is currently not supported with KVM, as well.

5.2.1 Defining a VM Guest without Starting the Installation

vm-install provides the --no-install parameter. With this parameter the XML configuration file defining the VM Guest is created, but the guest does not booted automatically. You may use it regardless whether you start vm-install in wizard mode or whether you specify all other options in the command line.

WARNING: No Virtual Disk Creation

When using the --no-install parameter with vm-install, no virtual disks will be created. Therefore, you have to create the disks in advance using either qemu-img or virsh.

Once the VM Guest XML configuration file is successfully created, you need to "register" it so it is recognized by Virtual Machine Manager or virsh. Do so by running:

```
virsh -c qemu:///system define PATH_TO_XMLFILE
```

5.3 Advanced Guest Installation Scenarios

This section provides instructions for operations exceeding the scope of a normal installation, such as adding para-virtualized drivers during the installation or including add-on packages.

5.3.1 Adding para-virtualized Drivers During the Installation

Installing an operating system out of the box on machines with para-virtualized devices is currently only possible with SUSE Linux Enterprise Server 11 SP1 or RedHat Enterprise Linux 5.4 and newer. Novell offers para-virtualized drivers for SUSE Linux Enterprise Server 10 SP3 and Windows operating systems, that can be installed manually.

In case of SUSE Linux Enterprise Server 10 SP3 these drivers can be added at the start of the installation. Thus it is possible to install on a virtual machine with para-virtualized devices (hard disk, network adapter and memory ballooning). See Section "*Hardware*" (page 25) on how to activate para-virtualization when setting up a virtual machine. Windows operating systems refuse to install on para-virtualized drives, even if adding the drivers during the installation. Therefore para-virtualization for Windows can only be activated in an already installed system.

See Section A.1, "Installing Para-Virtualized Drivers" (page 143) for instructions on installing para-virtualized drivers in an already installed system.

SUSE Linux Enterprise Server 10 SP3

SUSE Linux Enterprise Server 10 SP3 does not support para-virtualized devices out of the box, because the drivers are not included. In order to use such devices, add the missing drivers during the installation. Proceed as follows:

1 Download the ISO image containing the para-virtualized drivers from http://drivers.suse.com/novell/Novell-virtio-drivers-2.6.27/sle10-sp3/novell-virtio-drivers-2.6.27-sle10sp3.iso.

2 When setting up the VM Guest, specify the ISO image containing the para-virtualized drivers as CDROM device. If you are installing from CDROM or from an ISO image, make sure to configure the para-virtualized driver image as the *second* CDROM device.

3 Finish the guest setup and start the installation. Enter

```
dud=1
```

at the boot prompt and boot into the installation.

4 After the installation system has loaded, a pop-up window opens and asks for the device from which to install the drivers. Choose the CDROM device containing the drivers.

5.3.2 Including Add-On Products in the Installation

Some operating systems such as SUSE Linux Enterprise Server offer to include add-on products in the installation process. In case the add-on product installation source is provided via network, no special VM Guest configuration is needed. If it is provided via CD/DVD or ISO image, it is necessary to provide the VM Guest installation system with both, the standard installation images and the image for the add-on product.

First add the standard installation image, and second the physical CD/DVD-ROM or add-on image. The image or device added first is automatically chosen as the boot image. In case you install SUSE Linux Enterprise Server, it will be configured as `/dev/sr0`, while the add-on product source will be configured as `/dev/sr1`.

6 Basic VM Guest Management

Basic management tasks such as starting or stopping a VM Guest, can either be done using the graphical application Virtual Machine Manager or on the command line using `virsh`. Connecting to the graphical console via VNC is only possible from a graphical user interface.

6.1 Listing VM Guests

In order to be able to list VM Guests, you need to connect to a VM Host Server first. If you start the management tool on the VM Host Server itself, you are automatically connected. When operating from remote, refer to Section 7.3, “Connecting to a VM Host Server” (page 63) for instructions.

6.1.1 Listing VM Guests with Virtual Machine Manager

The main Window of the Virtual Machine Manager shows a list of all VM Guests for each VM Host Server it is connected to. Each VM Guest entry contains the machine's name, its status (*Running*, *Paused*, or *Shutoff*) displayed as icon and literal, and a CPU usage bar.

6.1.2 Listing VM Guests with virsh

Use the command `virsh list` to get a list of VM Guests:

```
# list running guests on localhost
virsh -c qemu:///system list
# list running + inactive guests on mercury over TLS connection
virsh -c qemu+tls://mercury.example.com/system list --all
# list running + inactive guests on mercury over SSH connection
virsh -c qemu+ssh://tux@mercury.example.com/system list --inactive
```

6.2 Opening a Graphical Console

Opening a Graphical Console to a VM Guest lets you interact with the machine like a physical host via a VNC connection. If accessing the VNC server requires authentication, you are prompted to enter a user name (if applicable) and a password.

Once you click into the VNC console, the cursor is "grabbed" and cannot be used outside the console anymore. To release it, press Alt + Ctrl.

TIP: Absolute Cursor Movement

In order to prevent the console from grabbing the cursor and to enable absolute (seamless) cursor movement, add a tablet input device to the VM Guest.

Certain key combinations such as Ctrl + Alt + Del are interpreted by the host system and are not passed to the VM Guest.

To pass such key combinations to a VM Guest, open the *Send Key* menu from the VNC window and choose the desired key combination entry.

NOTE: Supported VNC Viewer

Principally all VNC viewers are able to connect to the console of a VM Guest. However, if you are using SASL authentication and/or TLS/SSL connection to access the guest, the options become limited. Common VNC viewers such as `tightvnc` or `tigervnc` support neither SASL authentication or TSL/SSL. The only supported alternative to Virtual Machine Manager and `virt-viewer` is `vinagre`.

6.2.1 Opening a Graphical Console with Virtual Machine Manager

1 In the Virtual Machine Manager, right-click on a VM Guest entry.

2 Choose *Open* from the pop-up menu.

6.2.2 Opening a Graphical Console with virt-viewer

`virt-viewer` is a simple VNC viewer with added functionality for displaying VM Guest consoles. It can, for example, be started in "wait" mode, where it waits for a VM Guest to start before it connects. It also supports automatically reconnecting to a VM Guest that is rebooted.

`virt-viewer` addresses VM Guests by name, by ID or by UUID. Use `virsh list --all` to get this data.

To connect to a guest that is running or paused, either use the ID, UUID, or name. VM Guests that are shut off do not have an ID—you can only connect by UUID or name.

```
# local connect to guest with ID 8
virt-viewer -c qemu:///system 8

# local connect to the inactive guest sles11; will connect once
# the guest starts
virt-viewer --wait sles11
virt-viewer -c qemu:///system --wait sles11

# remote connect via ssh
viewer -c qemu+ssh://tux@mercury.example.com/system 8

# remote connect via ssh wait / reconnect mode:
virt-viewer -c qemu+ssh://root@mercury.example.com/system -w sles11
```

For more information, see `virt-viewer --help` or `man 1 virt-viewer`.

6.3 Changing a VM Guest's State: Start, Stop, Pause

Starting, stopping or pausing a VM Guest can either be done with Virtual Machine Manager or `virsh`. You can also configure a VM Guest to be automatically started when booting the VM Host Server.

When shutting down a VM Guest, you may either shut it down gracefully, or force the shutdown. The latter is equivalent to pulling the power plug on a physical host and is only recommended if there are no alternatives. Forcing a shutdown may cause file system corruption and loss of data on the VM Guest.

TIP: Graceful Shutdown

In order to be able to perform a graceful shutdown, the VM Guest must be configured to support ACPI. If you have created the guest with `vm-install` or with Virtual Machine Manager, ACPI should be available. Use the following procedure in Virtual Machine Manager to check:

Double-click the VM Guest entry in Virtual Machine Manager. Choose *View* > *Details* and then *Overview* > *Machine Settings*. *ACPI* should be checked.

Depending on the guest operating system, enabling ACPI may not be sufficient. It is strongly recommended to test shutting down and rebooting a guest before releasing it to production. openSUSE or SUSE Linux Enterprise Desktop, for example, may require PolicyKit authorization for shutdown and reboot. Make sure this policy is turned off on all VM Guests.

If ACPI was enabled during a Windows XP/Server 2003 guest installation, turning it on in the VM Guest configuration alone is not sufficient. See the following articles for more information:

`http://support.microsoft.com/kb/314088/EN-US/`
`http://support.microsoft.com/?kbid=309283`

A graceful shutdown is of course always possible from within the guest operating system, regardless of the VM Guest's configuration.

6.3.1 Changing a VM Guest's State with Virtual Machine Manager

Changing a VM Guest's state can either be done from Virtual Machine Manager's main window, or from a VNC window.

Procedure 6.1 *State Change from the Virtual Machine Manager Window*

1. Right-click on a VM Guest entry.
2. Choose *Run*, *Pause*, or one of the *Shutdown options* from the pop-up menu.

Procedure 6.2 *State change from the VNC Window*

1. Open a VNC Window as described in Section 6.2.1, “Opening a Graphical Console with Virtual Machine Manager” (page 39).
2. Choose *Run*, *Pause*, or one of the *Shut Down* options either from the toolbar or from the *Virtual Machine* menu.

Autostarting a VM Guest

Automatically starting a guest when the VM Host Server boots is not enabled by default. This feature needs to be turned on for each VM Guest individually. There is no way to activate it globally.

1. Double-click the VM Guest entry in Virtual Machine Manager to open its console.
2. Choose *View* > *Details* to open the VM Guest configuration window.
3. Choose *Boot Options* and check *Start virtual machine on host boot up*.
4. Save the new configuration with *Apply*.

6.3.2 Changing a VM Guest's State with virsh

In the following examples the state of a VM Guest named “sles11” is changed. All commands are run directly on the VM Host Server.

```
virsh -c qemu:///system start sles11                # start
virsh -c qemu:///system suspend sles11              # pause
virsh -c qemu:///system reboot sles11               # reboot
virsh -c qemu:///system shutdown sles11             # graceful shutdown
virsh -c qemu:///system destroy sles11              # force shutdown
virsh -c qemu:///system autostart sles11            # turn on autostart
virsh -c qemu:///system autostart --disable sles11  # turn off autostart
```

6.4 Saving and Restoring VM Guests

Saving a VM Guest preserves the exact state of the guest’s memory. The operation is slightly similar to *hibernating* a computer. A saved VM Guest can be quickly restored to its previously saved running condition.

When saved, the VM Guest is paused, its current memory state saved to disk, and then the guest is stopped. The operation does not make a copy of any portion of the VM Guest’s virtual disk. The amount of time to save the virtual machine depends on the amount of memory allocated. When saved, a VM Guest’s memory is returned to the pool of memory available on the VM Host Server.

The restore operation loads a VM Guest’s previously saved memory state file and starts it. The guest is not booted but rather resumes at the point where it was previously saved. The operation is slightly similar to coming out of hibernation.

The VM Guest is saved to a state file. Make sure there is enough space on the partition you are going to save to. Issue the following command on the guest to get a rough estimation of the file size in megabytes to be expected:

```
free -m | awk '/^Mem:/ {print $3}'
```

WARNING

After using the save operation, do not boot, start, or run the saved VM Guest. Doing so would cause the machine's virtual disk and the saved memory state getting out of sync and can result in critical errors when restoring the guest.

6.4.1 Saving / Restoring with Virtual Machine Manager

Procedure 6.3 *Saving a VM Guest*

1. Open a VNC connection window to a VM Guest. Make sure the guest is running.
2. Choose *Virtual Machine* > *Save*
3. Choose a location and a file name.
4. Click *Save*. Saving the guest's state may take some time. After the operation has finished, the VM Guest will automatically shut down.

Procedure 6.4 *Restoring a VM Guest*

1. Start the Virtual Machine Manager.
2. Type Alt + R or choose *File* > *Restore Saved Machine*.
3. Choose the file you want to restore and proceed with *Open*. Once the file has been successfully loaded, the VM Guest is up and running.

6.4.2 Saving / Restoring with virsh

Save a running VM Guest with the command `virsh save`:

```
# save the guest named opensuse11
virsh save opensuse11 /virtual/saves/opensuse11.vmsave

# save the guest with Id 37
virsh save 37 /virtual/saves/opensuse11.vmsave
```

To restore it, use `virsh restore`:

```
virsh restore /virtual/saves/opensuse11.vmsave
```

6.5 Deleting a VM Guest

Deleting a VM Guest removes its XML configuration by default. Since the attached storage is not deleted, you will be able to use it with another VM Guest. With Virtual Machine Manager you may also delete a guest's storage files as well—this will completely erase the guest.

In order to delete a VM Guest, it has to be shut down first (refer to Section 6.3, “Changing a VM Guest's State: Start, Stop, Pause” (page 40) for instructions). It is not possible to delete a running guest.

6.5.1 Deleting a VM Guest with Virtual Machine Manager

1. In the Virtual Machine Manager, right-click on a VM Guest entry.
2. Choose *Delete* from the pop-up menu.
3. A confirmation window opens. Clicking *Delete* will permanently erase the VM Guest. The deletion is not recoverable.

   You may also choose to permanently delete the guest's virtual disk by ticking *Delete Associated Storage Files*. The deletion is not recoverable either.

6.5.2 Deleting a VM Guest with virsh

To delete a VM Guest with `virsh` run `virsh undefine` *`VM_NAME`*. There is no option to automatically delete the attached storage files.

7 Connecting and Authorizing

Having to manage several VM Host Servers, each hosting a couple of VM Guests, quickly becomes difficult to handle. One of the major benefits of `libvirt` is the ability to connect to several VM Host Servers at once, providing a single interface to manage all VM Guests and to connect to their graphical console.

In order to ensure only authorized users can connect, `libvirt` offers several connection types that can be combined with different authorization mechanisms.

7.1 Authentication

The power to manage VM Guests and to access their graphical console obviously is something that should be restricted to a well defined circle of persons. In order to achieve this goal, you can use the following authentication techniques on the VM Host Server:

- Access control for UNIX sockets with permissions and group ownership. This method is available for `libvirtd` connections only.
- Access control for UNIX sockets with PolicyKit. This method is available for local `libvirtd` connections only.
- Username and password authentication with SASL (Simple Authentication and Security Layer). This method is available for both, `libvirtd` and VNC connections. Using SASL does not require real user accounts on the server, since it uses its own database to store usernames and passwords.

- Kerberos authentication. This method, available for `libvirtd` connections only, is not covered in this manual. Please refer to `http://libvirt.org/auth.html#ACL_server_kerberos` for details.
- Single password authentication. This method is available for VNC connections only.

IMPORTANT: Authentication for `libvirtd` and VNC need to be configured separately

Access to the VM Guest management functions (via `libvirtd`) on the one hand and to their graphical console on the other hand, always needs to be configured separately. When restricting the access to the management tools, these restrictions do *not* automatically apply to VNC connections!

When accessing VM Guests from remote via TLS/SSL connections, access can be indirectly controlled on each client by restricting read permissions to the certificate's key file to a certain group. See Section “Restricting Access (Security Considerations)” (page 60) for details.

7.1.1 `libvirtd` Authentication

`libvirtd` authentication is configured in `/etc/libvirt/libvirtd.conf`. The configuration made here applies to all `libvirt` tools such as the Virtual Machine Manager or `virsh`.

`libvirt` offers two sockets: a read-only socket for monitoring purposes and a read-write socket to be used for management operations. Access to both sockets can be configured independently. By default, both sockets are owned by `root.root`. Default access permissions on the read-write socket are restricted to the user `root` (`0700`) and fully open on the read-only socket (`0777`).

In the following instructions you learn how to configure access permissions for the read-write socket. The same instructions also apply to the read-only socket. All configuration steps have to be carried out on the VM Host Server.

NOTE: Default Authentication Settings on SUSE Linux Enterprise Server

The default authorization method on SUSE Linux Enterprise Server is access control for UNIX sockets with PolicyKit. Every user accessing the read-write socket, has to provide the `root` password once and is granted access for the current and for future sessions.

Recommended Authorization Methods

Local Connections

Section “Access Control for UNIX Sockets with PolicyKit” (page 48)
Section “Access Control for UNIX Sockets with Permissions and Group Ownership” (page 47)

Remote Tunnel over SSH

Section “Access Control for UNIX Sockets with Permissions and Group Ownership” (page 47)

Remote TLS/SSL Connection

Section “Username and Password Authentication with SASL” (page 49)
none (access controlled on the client side by restricting access to the certificates)

Access Control for UNIX Sockets with Permissions and Group Ownership

In order to grant access for non-`root` accounts, configure the sockets to be owned and accessible by a certain group (`libvirt` in the following example). This authentication method can be used for local and remote SSH connections.

1 In case it does not exist, create the group which should own the socket:

```
groupadd libvirt
```

IMPORTANT: Group Needs to Exist

The group must exist prior to restarting `libvirtd`. If not, the restart will fail.

2 Add the desired users to the group:

```
usermod -A libvirt tux
```

3 Change the configuration in `/etc/libvirt/libvirtd.conf` as follows:

```
unix_sock_group = "libvirt"❶
unix_sock_rw_perms = "0770"❷
auth_unix_rw = "none"❸
```

❶ Group ownership will be set to group `libvirt`.

❷ Sets the access permissions for the socket (srwxrwx---).

❸ Disables other authentication methods (PolicyKit or SASL). Access is solely controlled by the socket permissions.

4 Restart `libvirtd`:

```
rclibvirtd restart
```

Access Control for UNIX Sockets with PolicyKit

Access control for UNIX sockets with PolicyKit is the default authentication method on SUSE Linux Enterprise Server. Therefore, no `libvirt` configuration changes are needed. With PolicyKit authorization enabled, permissions on both sockets default to `0777` and each application trying to access a socket needs to authenticate via PolicyKit. SUSE Linux Enterprise Server currently supports this method for local connections only.

Two policies for accessing `libvirt`'s sockets exist:

- *org.libvirt.unix.monitor*: accessing the read-only socket
- *org.libvirt.unix.manage*: accessing the read-write socket

By default, the policy for accessing the read-write socket is to authenticate with `root` password once and grant the privilege for the current and for future sessions (auth_admin_keep_always).

In order to grant users access to the read-write socket without having to provide the `root` password, there are two possibilities:

1. Using the `polkit-auth` command, you can grant the privilege without any restrictions:

```
polkit-auth --user tux --grant org.libvirt.unix.manage    # grant privilege
polkit-auth --user tux --revoke org.libvirt.unix.manage   # revoke privilege
```

2. Editing `/etc/PolicyKit/PolicyKit.conf` offers more advanced options. Add the following XML snippet in between the existing `<config version="0.1">` and `</config>` tags:

```
<match action="org.libvirt.unix.manage">❶
  <match user="tux">❷
    <return result="yes"/>❸
  </match>
</match>
```

❶ The name of the policy; `org.libvirt.unix.manage` stands for accessing the read-write socket.

❷ The username(s) which to grant the privilege. Use the `|` symbol to separate entries (`user="tux|wilber"`).

❸ The privilege that is granted. The following options exist: `yes` (no restrictions), `no` (block access completely), `auth_self` or `auth_admin` (authenticate with own password/`root` password every time the privilege is requested), `auth_self_keep_session` or `auth_admin_keep_session` (authenticate with own password/`root` password once per session) and `auth_self_keep_always` or `auth_admin_keep_always` (authenticate only once with own password/`root` password).

Username and Password Authentication with SASL

SASL provides username and password authentication as well as data encryption (digest-md5, by default). Since SASL maintains its own user database, the users do not need to exist on the VM Host Server. SASL is required by TCP connections and on top of TLS/SSL connections.

IMPORTANT: Plain TCP and SASL with digest-md5 Encryption

Using digest-md5 encryption on an otherwise unencrypted TCP connection does not provide enough security for production environments. It is recommended to only use it in testing environments.

TIP: SASL Authentication on Top of TLS/SSL

Access from remote TLS/SSL connections can be indirectly controlled on the *client side* by restricting access to the certificate's key file. However, this might prove error-prone when dealing with a large number of clients. Utilizing SASL with TLS adds security by additionally controlling access on the server side.

To configure SASL authentication, proceed as follows:

1 Change the configuration in `/etc/libvirt/libvirtd.conf` as follows:

1a To enable SASL for TCP connections:

```
auth_tcp = "sasl"
```

1b To enable SASL for TLS/SSL connections:

```
auth_tls = "sasl"
```

2 Restart `libvirtd`:

```
rclibvirtd restart
```

3 The libvirt SASL configuration file is located at `/etc/sasl2/libvirtd.conf`. Normally, there is no need to change the defaults. However, if using SASL on top of TLS, you may turn off session encryption to avoid additional overhead— TLS connections are already encrypted— by commenting the `mech_list`. For TCP connections this parameter must be set to digest-md5:

```
mech_list: digest-md5   # mandatory for TCP connections
#mech_list: digest-md5   # apply default (username+password) TLS/SSL only!
```

4 By default, no SASL users are configured, so no logins are possible. Use the following commands to add, list, and delete users:

```
mercury:~ # saslpasswd2 -a libvirt tux                    # add user tux
Password:
Again (for verification):
mercury:~ # sasldblistusers2 -f /etc/libvirt/passwd.db  # list users
tux@mercury.example.com: userPassword
mercury:~ # saslpasswd2 -a libvirt -d tux                 # delete user tux
```

TIP: `virsh` and SASL Authentication

When using SASL authentication you will be prompted for a username and password every time you issue a `virsh` command. Avoid this by using `virsh` in shell mode.

7.1.2 VNC Authentication

Since access to the graphical console of a VM Guest is not controlled by `libvirt`, but rather by `QEMU`, it is always necessary to additionally configure VNC authentication. The main configuration file is `/etc/libvirt/qemu.conf`.

Two authentication types are available: SASL and single password authentication. If you are using SASL for `libvirt` authentication, it is strongly recommended to use it for VNC authentication as well—it is possible to share the same database.

A third method to restrict access to the VM Guest is to enable the use of TLS encryption on the VNC server. This requires the VNC clients to have access to x509 client certificates. By restricting access to these certificates, access can indirectly be controlled on the client side. Refer to Section “VNC over TLS/SSL: Client Configuration” (page 58) for details.

Username and Password Authentication with SASL

SASL provides username and password authentication as well as data encryption. Since SASL maintains its own user database, the users do not need to exist on the VM Host Server. As with SASL authentication for `libvirt`, you may use SASL on top of TLS/SSL connections. Refer to Section “VNC over TLS/SSL: Client Configuration” (page 58) for details on configuring these connections.

To configure SASL authentication for VNC, proceed as follows:

1 Create a SASL configuration file. It is recommended to use the existing `libvirt` file. If you have already configured SASL for `libvirt` and are planning to use the same settings including the same username/password database, a simple link is suitable:

```
ln –s /etc/sasl2/libvirt.conf /etc/sasl2/qemu.conf
```

In case you are setting up SASL for VNC only or planning to use a different configuration than for `libvirt`, copy the existing file to use as a template and edit it according to your needs:

```
cp /etc/sasl2/libvirt.conf /etc/sasl2/qemu.conf
```

2 By default, no SASL users are configured, so no logins are possible. Use the following commands to add, list, and delete users:

```
mercury:~ # saslpasswd2 –a libvirt tux                  # add user tux
Password:
Again (for verification):
mercury:~ # sasldblistusers2 –f /etc/libvirt/passwd.db  # list users
tux@mercury.example.com: userPassword
mercury:~ # saslpasswd2 –a libvirt –d tux               # delete user tux
```

3 Change the configuration in `/etc/libvirt/qemu.conf` as follows:

```
vnc_listen = "0.0.0.0"
vnc_sasl = 1
```

The first parameter enables VNC to listen on all public interfaces (rather than to the local host only), and the second parameter enables SASL authentication.

4 Restart `libvirtd`:

```
rclibvirtd restart
```

5 Restart all VM Guests that have been running prior to changing the configuration. VM Guests that have not been restarted will not use SASL authentication for VNC connects.

NOTE: Supported VNC Viewers

Currently only the same VNC viewers that also support TLS/SSL connections, support SASL authentication, namely Virtual Machine Manager, `virt–viewer`, and `vinagre`.

Single Password Authentication

Access to the VNC server may also be controlled by setting a VNC password. You can either set a global password for all VM Guests or set individual passwords for each guest. The latter requires to edit the VM Guest's config files.

NOTE: Always Set a Global Password

If you are using the single password authentication, it is good practice to set a global password even if setting passwords for each VM Guest. This will always leave your virtual machines protected with a “fallback” password if you forget to set a per-machine password. The global password will only be used if no other password is set for the machine.

Procedure 7.1 *Setting a Global VNC Password*

1. Change the configuration in `/etc/libvirt/qemu.conf` as follows:

   ```
   vnc_listen = "0.0.0.0"
          vnc_password = "PASSWORD"
   ```

   The first parameter enables VNC to listen on all public interfaces (rather than to the local host only), and the second parameter sets the password. The maximum length of the password is eight characters.

2. Restart `libvirtd`:

   ```
   rclibvirtd restart
   ```

3. Restart all VM Guests that have been running prior to changing the configuration. VM Guests that have not been restarted will not use password authentication for VNC connects.

Procedure 7.2 *Setting a VM Guest Specific VNC Password*

1. Change the configuration in `/etc/libvirt/qemu.conf` as follows to enable VNC to listen on all public interfaces (rather than to the local host only).

   ```
   vnc_listen = "0.0.0.0"
   ```

2 Open the VM Guest's XML configuration file in an editor. Replace *VM NAME* in the following example with the name of the VM Guest. The editor that is used defaults to `$EDITOR`. If that variable is not set, `vi` is used.

```
virsh edit VM NAME
```

3 Search for the element `<graphics>` with the attribute `type='vnc'`, for example:

```
<graphics type='vnc' port='-1' autoport='yes'/>
```

4 Add the `passwd=`*PASSWORD* attribute, save the file and leave the editor. The maximum length of the password is eight characters.

```
<graphics type='vnc' port='-1' autoport='yes' passwd='PASSWORD'/>
```

5 Restart `libvirtd`:

```
rclibvirtd restart
```

6 Restart all VM Guests that have been running prior to changing the configuration. VM Guests that have not been restarted will not use password authentication for VNC connects.

WARNING: Security

The VNC protocol is not considered to be safe. Although the password is sent encrypted, it might be vulnerable, when an attacker is able to sniff both, the encrypted password and the encryption key. Therefore, it is recommended to use VNC with TLS/SSL or tunneled over SSH. `virt-viewer`, as well as the Virtual Machine Manager and `vinagre` from version 2.30 on, support both methods.

7.2 Configuring Remote Connections

A major benefit of `libvirt` is the ability to manage VM Guests on different remote hosts from a central location. This section gives detailed instructions on how to configure server and client to allow remote connections.

7.2.1 Remote Tunnel over SSH

Enabling a remote connection that is tunneled over SSH on the VM Host Server only requires the ability to accept SSH connections. Make sure the SSH daemon is started (`rcsshd status`) and that the ports for service `SSH` are opened in the firewall.

User authentication for SSH connections can be done using traditional file user/group ownership and permissions as described in Section "Access Control for UNIX Sockets with Permissions and Group Ownership" (page 47).

7.2.2 Remote TLS/SSL Connection with x509 Certificate

Using TCP connections with TLS/SSL encryption and authentication via x509 certificates is much more complicated to set up than SSH, but it is a lot more scalable. Use this method if you have to manage several VM Host Servers with a varying number of administrators.

Basic concept

Basically, TLS (Transport Layer Security) encrypts the communication between two computers by using certificates. The computer starting the connection is always considered as the "client" using a "client certificate", while the receiving computer is always considered as the "server", using a "server certificate". This scenario applies, for example, if you manage your VM Host Servers from a central desktop.

If connections are initiated from both computers, each needs to have a client *and* a server certificate. This is the case, for example, if you migrate a VM Guest from one host to another.

Each x509 certificate has a matching private key file. Only the combination of certificate and private key file is able to identify itself correctly. In order to assure that a certificate was issued by the assumed owner, it is signed and issued by a central certificate called certification authority (CA). Both the client and the server certificates must be issued by the same CA.

IMPORTANT: User Authentication

Using a remote TLS/SSL connection basically only ensures that two computers are allowed to communicate in a certain direction. Restricting access to certain users can indirectly be achieved on the client side by restricting access to the certificates. Refer to Section "Restricting Access (Security Considerations)" (page 60) for details. `libvirt` also supports user authentication on the server with SASL. Read more in Section "Central User Authentication with SASL for TLS Sockets" (page 62).

Configuring the VM Host Server

The VM Host Server is the machine receiving connections. Therefore, the *server* certificates have to be installed. The CA certificate needs to be installed, as well. Once the certificates are in place, TLS support can be turned on for `libvirt`.

1. Create the server certificate and export it together with the CA certificate as described in Section A.2, "Generating x509 Client/Server Certificates" (page 149).

2. Create the following directories on the VM Host Server:

   ```
   mkdir –p /etc/pki/CA/ /etc/pki/libvirt/private/
   ```

   Install the certificates as follows:

   ```
   /etc/pki/CA/cacert.pem
   /etc/pki/libvirt/servercert.pem
   /etc/pki/libvirt/private/serverkey.pem
   ```

   IMPORTANT: Restrict Access to Certificates

   Make sure to restrict access to certificates as explained in Section "Restricting Access (Security Considerations)" (page 60).

3. Enable TLS support by editing `/etc/libvirt/libvirtd.conf` and setting `listen_tls = 1`. Restart `libvirtd`:

   ```
   rclibvirtd restart
   ```

4. By default, `libvirt` uses the TCP port 16514 for accepting secure TLS connections. Open this port in the firewall.

IMPORTANT: Restarting `libvirtd` with TLS enabled

If you enable TLS for `libvirt`, the server certificates need to be in place, otherwise restarting `libvirtd` will fail. You also need to restart `libvirtd` in case you change the certificates.

Configuring the Client and Testing the Setup

The client is the machine initiating connections. Therefore the *client* certificates have to be installed. The CA certificate needs to be installed, as well.

1. Create the client certificate and export it together with the CA certificate as described in Section A.2, “Generating x509 Client/Server Certificates” (page 149).

2. Create the following directories on the client:

   ```
   mkdir -p /etc/pki/CA/ /etc/pki/libvirt/private/
   ```

   Install the certificates as follows:

   ```
   /etc/pki/CA/cacert.pem
   /etc/pki/libvirt/clientcert.pem
   /etc/pki/libvirt/private/clientkey.pem
   ```

   IMPORTANT: Restrict Access to Certificates

   Make sure to restrict access to certificates as explained in Section “Restricting Access (Security Considerations)” (page 60).

3. Test the client/server setup by issuing the following command. Replace *`mercury.example.com`* with the name of your VM Host Server. Specify the same full qualified hostname as used when creating the server certificate.

   ```
   virsh -c qemu+tls://mercury.example.com/system list --all
   ```

   If your setup is correct, you will see a list of all VM Guests registered with `libvirt` on the VM Host Server.

Enabling VNC for TLS/SSL connections

Currently, VNC communication over TLS is only supported by few tools. The widespread `tightvnc` or `tigervnc` viewer, for example, do not support TLS. Known to work are the Virtual Machine Manager (`virt-manager`), `virt-viewer` and the GNOME VNC viewer `vinagre`.

VNC over TLS/SSL: VM Host Server Configuration

In order to access the graphical console via VNC over TLS/SSL, you need to configure the VM Host Server as follows:

1. Open ports for the service `VNC` in your firewall.

2. Create a directory `/etc/pki/libvirt-vnc` and link the certificates into this directory as follows:

```
mkdir -p /etc/pki/libvirt-vnc && cd /etc/pki/libvirt-vnc
 ln -s /etc/pki/CA/cacert.pem ca-cert.pem
 ln -s /etc/pki/libvirt/servercert.pem server-cert.pem
 ln -s /etc/pki/libvirt/private/serverkey.pem server-key.pem
```

3. Edit `/etc/libvirt/qemu.conf` and set the following parameters:

```
vnc_listen = "0.0.0.0"
 vnc_tls = 1
 vnc_tls_x509_verify = 1
```

4. Restart the `libvirtd`:

```
rclibvirtd restart
```

IMPORTANT: VM Guests Need to be Restarted

The VNC TLS setting is only set when starting a VM Guest. Therefore, you need to restart all machines that have been running prior to making the configuration change.

VNC over TLS/SSL: Client Configuration

The only action needed on the client side is to place the x509 client certificates in a location recognized by the client of choice. Unfortunately, each supported client—Virtual

Machine Manager, `virt-viewer`, and `vinagre`—expects the certificates in a different location. However, Virtual Machine Manager and `vinagre` can either read from a system wide location applying to all users, or from a per user location.

Virtual Machine Manager (`virt-manager`)

In order to connect to the remote host, Virtual Machine Manager requires the setup explained in Section "Configuring the Client and Testing the Setup" (page 57). In order to be able to connect via VNC the client certificates also need to be placed in the following locations:

System wide location

```
/etc/pki/CA/cacert.pem
/etc/pki/libvirt-vnc/clientcert.pem
/etc/pki/libvirt-vnc/private/clientkey.pem
```

Per user location

```
/etc/pki/CA/cacert.pem
~/.pki/libvirt-vnc/clientcert.pem
~/.pki/libvirt-vnc/private/clientkey.pem
```

`virt-viewer`

`virt-viewer` only accepts certificates from a system wide location:

```
/etc/pki/CA/cacert.pem
/etc/pki/libvirt-vnc/clientcert.pem
/etc/pki/libvirt-vnc/private/clientkey.pem
```

`vinagre`

System wide location

```
/etc/pki/CA/cacert.pem
/etc/pki/vinagre/clientcert.pem
/etc/pki/vinagre/private/clientkey.pem
```

Per user location

```
$HOME/.pki/CA/cacert.pem
```

```
~/.pki/vinagre/clientcert.pem
~/.pki/vinagre/private/clientkey.pem
```

IMPORTANT: Restrict Access to Certificates

Make sure to restrict access to certificates as explained in Section “Restricting Access (Security Considerations)” (page 60).

Restricting Access (Security Considerations)

Each x509 certificate consists of two pieces: the public certificate and a private key. A client can only authenticate using both pieces. Therefore, any user that has read access to the client certificate and its private key can access your VM Host Server. On the other hand, an arbitrary machine equipped with the full server certificate can pretend to be the VM Host Server. Since this is probably not desirable, access to at least the private key files needs to be restricted as much as possible. The easiest way to control access to a key file is to use access permissions.

Server Certificates

Server certificates need to be readable for QEMU processes. On SUSE Linux Enterprise Server QEMU processes started from `libvirt` tools are owned by `root`, so it is sufficient if `root` is able to read them certificates:

```
chmod 700 /etc/pki/libvirt/private/
chmod 600 /etc/pki/libvirt/private/serverkey.pem
```

If you change the ownership for QEMU processes in `/etc/libvirt/qemu.conf`, you also need to adjust the ownership of the key file.

System Wide Client Certificates

To control access to a key file that is available system wide, restrict read access a certain group, so that only members of that group can read the key file. In the following example, a group `libvirt` is created and the group ownership of the `clientkey.pem` and its parent directory is set to `libvirt`. Afterwards, the access permissions are restricted to owner and group. Finally the user tux is added to the group `libvirt`, so he will be able to access the key file.

```
CERTPATH="/etc/pki/libvirt/"
# create group libvirt
groupadd libvirt
# change ownership to user root and group libvirt
```

```
chown root.libvirt $CERTPATH/private $CERTPATH/clientkey.pem
# restrict permissions
chmod 750 $CERTPATH/private
chmod 640 $CERTPATH/private/clientkey.pem
# add user tux to group libvirt
usermod -A libvirt tux
```

Per User Certificates

User specific client certificates for accessing the graphical console of a VM Guest via VNC need to be placed in the users home directory in `~/.pki`. Contrary to, for example, the VNC viewer using these certificates do not check the access permissions of the private key file. Therefore, it is solely on the user's responsibility to make sure the key file is not readable by others.

Restricting Access from the Server Side

By default, every client that is equipped with appropriate client certificates may connect to a VM Host Server accepting TLS connections. Therefore, it is possible to use additional server side authentication with SASL as described in Section “Username and Password Authentication with SASL” (page 49).

It is also possible to restrict access with a whitelist of DNs (distinguished names), so only clients with a certificate matching a DN from the list can connect.

Add a list of allowed DNs to `tls_allowed_dn_list` in `/etc/libvirt/libvirtd.conf`. This list may contain wildcards. Do not specify an empty list, since that would result in refusing all connections.

```
tls_allowed_dn_list = [
   "C=US,L=Provo,O=SUSE Linux Products GmbH,OU=*,CN=venus.example.com,EMAIL=*",

    "C=DE,L=Nuremberg,O=SUSE Linux Products GmbH,OU=Documentation,CN=*"]
```

Get the distinguished name of a certificate with the following command:

```
certtool -i --infile /etc/pki/libvirt/clientcert.pem | grep "Subject:"
```

Restart `libvirtd` after having changed the configuration:

```
rclibvirtd restart
```

Central User Authentication with SASL for TLS Sockets

A direct user authentication via TLS is not possible - this is handled indirectly on each client via the read permissions for the certificates as explained in Section “Restricting Access (Security Considerations)” (page 60). However, if a central, server based user authentication is needed `libvirt` also allows to use SASL (Simple Authentication and Security Layer) on top of TLS for direct user authentication. See Section “Username and Password Authentication with SASL” (page 49) for configuration details.

Troubleshooting

Virtual Machine Manager/virsh Cannot Connect to Server

Check the following in the given order:

Is it a firewall issue (TCP port 16514 needs to be open on the server)?
Is the client certificate (certificate and key) readable by the user that has started Virtual Machine Manager/`virsh`?
Has the same full qualified hostname as in the server certificate been specified with the connection?
Is TLS enabled on the server (`listen_tls = 1`)?
Has `libvirtd` been restarted on the server?

VNC Connection fails

Ensure that you can basically connect to the remote server using Virtual Machine Manager. If so, check whether the virtual machine on the server has been started with TLS support. The virtual machine's name in the following example is “sles11”.

```
ps ax | grep qemu | grep "\-name sles11" | awk -F" -vnc " '{ print FS $2 }'
```

If the output does not begin with a string similar to the following, the machine has not been started with TLS support and must be restarted.

```
-vnc 0.0.0.0:0,tls,x509verify=/etc/pki/libvirt
```

7.3 Connecting to a VM Host Server

In order to connect to a hypervisor with `libvirt`, you need to specify a uniform resource identifier (URI). This URI is needed with `virsh` and `virt-viewer` (except when working as `root` on the VM Host Server) and is optional for the Virtual Machine Manager. Although the latter can be called with a connection parameter (for example, `virt-manager -c qemu:///system`), it also offers a graphical interface to create connection URIs. See Section 7.3.1, "Managing Connections with Virtual Machine Manager" (page 64) for details.

*HYPERVISOR*❶*+PROTOCOL*❷*://USER@REMOTE*❸*/CONNECTION_TYPE*❹

❶ Specify the hypervisor. SUSE Linux Enterprise Server currently supports the following hypervisors: `test` (dummy for testing), `qemu` (KVM), and `xen` (Xen). This parameter is mandatory.

❷ When connecting to a remote host, specify the protocol here. Can be one of: `ssh` (connection via SSH tunnel), `tcp` (TCP connection with SASL/Kerberos authentication), `tls` (TLS/SSL encrypted connection with authentication via x509 certificates).

❸ When connecting to a remote host, specify the user and the remote hostname. If no user is specified, the username that has called the command (`$USER`) is used. Please see below for more information. For TLS connections the hostname has to be specified exactly as in the x509 certificate.

❹ When connecting to `QEMU` hypervisor, two connections types are accepted: `system` for full access rights, or `session` for restricted access. Since `session` access is not supported on SUSE Linux Enterprise Server, this documentation focuses on `system` access.

Example Hypervisor Connection URIs

`test:///default`
: Connect to the local dummy hypervisor. Useful for testing.

`qemu:///system`
: Connect to the QEMU hypervisor on the local host having full access (type system). This usually requires that the command is issued by the user `root`.

`qemu+ssh://tux@mercury.example.com/system`
: Connect to the QEMU hypervisor on the remote host mercury.example.com. The connection is established via an SSH tunnel.

`qemu+tls://saturn.example.com/system`
: Connect to the QEMU hypervisor on the remote host mercury.example.com. The connection is established TLS/SSL.

For more details and examples, refer to the `libvirt` documentation at http://libvirt.org/uri.html.

NOTE: Usernames in URIs

A username needs to be specified when using Unix socket authentication (regardless whether using the username/password authentication scheme or PolicyKit). This applies to all SSH and local connections.

There is no need to specify a username when using SASL authentication (for TCP or TLS connections) or when doing no additional server side authentication for TLS connections. With SASL the username will not be evaluated—you will be prompted for a SASL user/password combination in any case.

7.3.1 Managing Connections with Virtual Machine Manager

The Virtual Machine Manager uses a `Connection` for every VM Host Server it manages. Each connection contains all VM Guests on the respective host. By default, a connection to the localhost is already configured and connected.

All configured connections are displayed in the Virtual Machine Manager main window. Active connections are marked with a small triangle which you can click in order to fold or unfold the list of VM Guests for this connection.

Inactive connections are listed gray and are marked with `Not Connected`. Either double-click or right-click it and choose *Connect* from the context menu. You can also *Delete* an existing connection from this menu.

NOTE: Editing Existing Connections

It is not possible to edit an existing connection. In order to change a connection, create a new one with the desired parameters and delete the "old" one.

To add a new connection in the Virtual Machine Manager, proceed as follows:

1. Choose *File* > *Add Connection*
2. Choose the host's *Hypervisor* (*Xen* or *QEMU/KVM*)
3. Choose a *Connection* type—either *Local* for connecting to the host the Virtual Machine Manager was started on, or one of the remote connections (see Section 7.2, "Configuring Remote Connections" (page 54) for more information).
4. In case of a remote connection, enter the *Hostname* of the remote machine as `USERNAME@REMOTE_HOST`. Usernames must be specified for local connections as well as for SSH

   IMPORTANT: Specifying a Username

   There is no need to specify a username for TCP and TLS connections; it will not be evaluated anyway. A username must be specified for local connections as well as for SSH connections—if not, the default user `root` will be used.
5. If you do not want the connection to be automatically activated when starting the Virtual Machine Manager, remove the tick from *Autoconnect*.
6. Finish the configuration by clicking *Connect*.

8 Managing Storage

When managing a VM Guest on the VM Host Server itself, it is possible to access the complete file system of the VM Host Server in order to attach disks or images to the VM Guest. However, this is not possible when managing VM Guests from a remote host. For this reason, `libvirt` supports so called "Storage Pools" which can be accessed from remote machines. `libvirt` knows two different types of storage: volumes and pools.

Storage Volume
: A storage volume is a storage device that can be assigned to a guest—a virtual disk or a CD/DVD/floppy image. Physically (on the VM Host Server) it can be a block device (a partition, a logical volume, etc.) or a file.

Storage Pool
: A storage pool basically is a storage resource on the VM Host Server that can be used for storing volumes, similar to network storage for a desktop machine. Physically it can be one of the following types:

  File System Directory (*dir*)
  : A directory for hosting image files. The files can be fully allocated raw files, sparsely allocated raw files, or ISO images.

  Physical Disk Device (*disk*)
  : Use a complete physical disk as storage. A partition is created for each volume that is added to the pool. It is recommended to use a device name from `/dev/disk/by-*` rather than the simple `/dev/sd`*X*, since the latter may change.

Pre-Formatted Block Device (*fs*)
: Specify a partition to be used in the same way as a file system directory pool (a directory for hosting image files). The only difference to using a file system directory is the fact that `libvirt` takes care of mounting the device.

iSCSI Target (iscsi)
: Set up a pool on an iSCSI target. You need to have been logged into the volume once before, in order to use it with `libvirt` (use the YaST *iSCSI Initiator* to detect and log into a volume, see SLES 11 SP1: Storage Administration Guide (↑SLES 11 SP1: Storage Administration Guide) for details). It is recommended to use a device name from `/dev/disk/by-id` rather than the simple `/dev/sd`*X*, since the latter may change. Volume creation on iSCSI pools is not supported, instead each existing Logical Unit Number (LUN) represents a volume. Each volume/LUN also needs a valid (empty) partition table or disk label before you can use it. If missing, use for example, `fdisk` to add it:

```
~ # fdisk -cu
/dev/disk/by-path/ip-192.168.2.100:3260-iscsi-iqn.2010-10.com.example:[...]-lun-2
Device contains neither a valid DOS partition table, nor Sun, SGI
or OSF disklabel
Building a new DOS disklabel with disk identifier 0xc15cdc4e.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected
by w(rite)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

LVM Volume Group (logical)
: Use a LVM volume group as a pool. You may either use a pre-defined volume group, or create a group by specifying the devices to use. Storage volumes are created as partitions on the volume.

WARNING: Deleting the LVM Based Pool

When the LVM based pool is deleted in the Storage Manager, the volume group is deleted as well. This results in a non-recoverable loss of all data stored on the pool!

Network Exported Directory (*netfs*)
: Specify a network directory to be used in the same way as a file system directory pool (a directory for hosting image files). The only difference to using a file system directory is the fact that `libvirt` takes care of mounting the directory. Supported protocols are NFS and glusterfs.

SCSI Host Adapter (*scsi*)
: Use an SCSI host adapter in almost the same way a an iSCSI target. It is recommended to use a device name from `/dev/disk/by-id` rather than the simple `/dev/sd`*X*, since the latter may change. Volume creation on iSCSI pools is not supported, instead each existing LUN (Logical Unit Number) represents a volume.

WARNING: Security Considerations

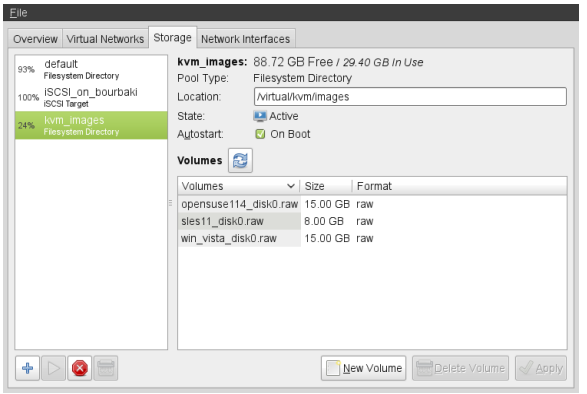
In order to avoid data loss or data corruption, do not attempt to use resources such as LVM volume groups, iSCSI targets, etc. that are used to build storage pools on the VM Host Server, as well. There is no need to connect to these resources from the VM Host Server or to mount them on the VM Host Server—`libvirt` takes care of this.

Do not mount partitions on the VM Host Server by label. Under certain circumstances it is possible that a partition is labeled from within a VM Guest with a name already existing on the VM Host Server.

When managing VM Guests from remote, a volume can only be accessed if it is located in a storage pool. On the other hand, creating new volumes is only possible within an existing storage pool. Although remote installation of a guest is currently not supported on SUSE Linux Enterprise Server, there are other use cases for storage pools, such as adding an additional virtual disk. If VM Guests should be able to access CDROM or DVDROM images, they also need to be in a storage pool.

8.1 Managing Storage with Virtual Machine Manager

The Virtual Machine Manager provides a graphical interface—the Storage Manager—to manage storage volumes and pools. To access it, either right-click on a connection and choose *Details*, or highlight a connection and choose *Edit* > *Connection Details*. Select the *Storage* tab.

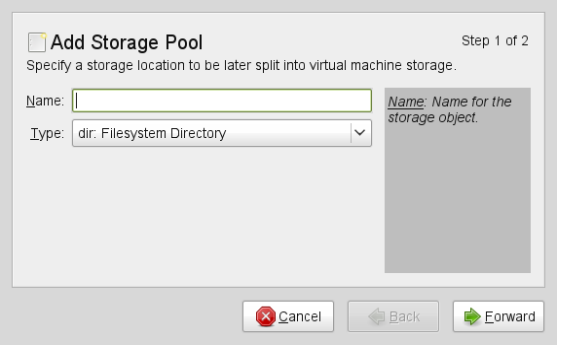


8.1.1 Adding a Storage Pool

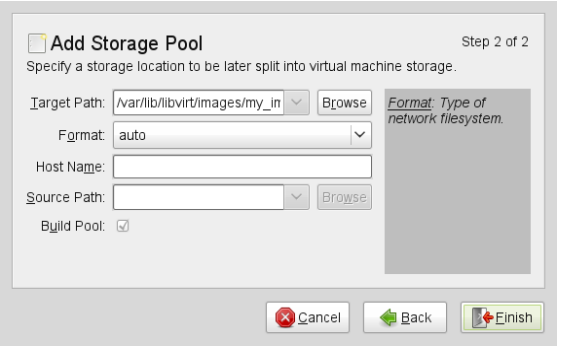
To add a storage pool, proceed as follows:

1. Click the plus symbol in the bottom left corner to open the *Add a New Storage Pool Window*.

2. Provide a *Name* for the pool (consisting of alphanumeric characters plus _-.) and select a *Type*. Proceed with *Forward*.



3 Specify the needed details in the following window. The data that needs to be entered depends on the type of pool you are creating.



Type *dir:*

- *Target Path*: Specify an existing directory.

Type *disk:*

- *Target Path*: The directory which hosts the devices. It is recommended to use a directory from `/dev/disk/by-*` rather than from `/dev`, since device names in the latter directory may change.
- *Format*: Format of the device's partition table. Using *auto* should work in most cases. If not, get the needed format by running `parted -l`.

- *Source Path:* The device, for example `sda`.
- *Build Pool:* Activating this option formats the device. Use with care— all data on the device will be lost!

Type *fs:*

- *target Path:* Mount point on the VM Host Server file system.
- *Format:* File system format of the device. the default value `auto` should work.
- *Source Path:* Path to the device file. It is recommended to use a device name from `/dev/disk/by-*` rather than the simple `/dev/sd`*X*, since the latter may change.

Type *iscsi:*

Get the necessary data by running the following command on the VM Host Server:

```
iscsiadm --mode node
```

It will return a list of iSCSI volumes with the following format. The elements highlighted with a bold font are the ones needed:

```
IP_ADDRESS:PORT,TPGT TARGET_NAME_(IQN)
```

- *Target Path:* The directory containing the device file. Do not change the default `/dev/disk/by-path`.
- *Host Name:* Host name or IP address of the iSCSI server.
- *Source Path:* The iSCSI target name (IQN).

Type *logical:*

- *Target Path:* In case you use an existing volume group, specify the existing device path. In case of building a new LVM volume group, specify a device name in the `/dev` directory that does not already exist.
- *Source Path:* Leave empty when using an existing volume group. When creating a new one, specify its devices here.
- *Build Pool:* Only activate when creating a new volume group.

Type *netfs:*

- *target Path:* Mount point on the VM Host Server file system.
- *Format:* Network file system protocol
- *Host Name:* IP address or hostname of the server exporting the network file system.
- *Source Path:* Directory on the server that is being exported.

Type *scsi:*

- *Target Path:* The directory containing the device file. Do not change the default `/dev/disk/by-path`.
- *Source Path:* Name of the SCSI adapter.

NOTE: File Browsing

Using the file browser by clicking on *Browse* is not possible when operating from remote.

---

4 Click *Finish* to add the storage pool.

8.1.2 Managing Storage Pools

Virtual Machine Manager's Storage Manager lets you create or delete volumes in a pool. You may also temporarily deactivate or permanently delete existing storage pools. Changing the basic configuration of a pool is not supported.

Starting, Stopping and Deleting Pools

The purpose of storage pools is to provide block devices located on the VM Host Server, that can be added to a VM Guest when managing it from remote. In order to make a pool temporarily inaccessible from remote, you may *Stop* it by clicking on the stop symbol in the bottom left corner of the Storage Manager. Stopped pools are marked with *State: Inactive* and are grayed out in the list pane. By default, a newly created pool will be automatically started *On Boot* of the VM Host Server.

To *Start* an inactive pool and make it available from remote again click on the play symbol in the bottom left corner of the Storage Manager.

NOTE: A Pool's State Does not Affect Attached Volumes

Volumes from a pool attached to VM Guests are always available, regardless of the pool's state (*Active* or *Inactive*). The state of the pool solely affects the ability to attach volumes to a VM Guest via remote management.

To permanently make a pool inaccessible, you can *Delete* it by clicking on the shredder symbol in the bottom left corner of the Storage Manager. You may only delete inactive pools. Deleting a pool does not physically erase its contents on VM Host Server—it only deletes the pool configuration. However, you need to be extra careful when deleting pools, especially when deleting LVM volume group-based tools:

WARNING: Deleting Storage Pools

Deleting storage pools based on local file system directories, local partitions or disks has no effect on the availability of volumes from these pools currently attached to VM Guests.

Volumes located in pools of type iSCSI, SCSI, LVM group or Network Exported Directory will become inaccessible from the VM Guest in case the pool will be deleted. Although the volumes themselves will not be deleted, the VM Host Server will no longer have access to the resources.

Volumes on iSCSI/SCSI targets or Network Exported Directory will be accessible again when creating an adequate new pool or when mounting/accessing these resources directly from the host system.

When deleting an LVM group-based storage pool, the LVM group definition will be erased and the LVM group will no longer exist on the host system. The configuration is not recoverable and all volumes from this pool are lost.

Adding Volumes to a Storage Pool

Virtual Machine Manager lets you create volumes in all storage pools, except in pools of types iSCSI or SCSI. A volume in these pools is equivalent to a LUN and cannot be changed from within `libvirt`.

1 A new volume can either be created using the Storage Manager or while adding a new storage device to a VM Guest. In both cases, select a *Storage Pool* and then click *New Volume*.

2 Specify a *Name* for the image and choose an image format (note that Novell currently only supports `raw` images). The latter option is not available on LVM group-based pools.

   Specify a *Max Capacity* and the amount of space that should initially be allocated. If both values differ, a `sparse` image file, growing on demand, will be created.

3 Start the volume creation by clicking *Finish*.

Deleting Volumes From a Storage Pool

Deleting a volume can only be done from the Storage Manager, by selecting a volume and clicking *Delete Volume*. Confirm with *Yes*. Use this function with extreme care!

WARNING: No Checks Upon Volume Deletion

A volume will be deleted in any case, regardless whether it is currently used in an active or inactive VM Guest. There is no way to recover a deleted volume.

At the moment `libvirt` offers no tools to list all volumes currently being used in VM Guest definitions. This makes it even more dangerous to delete volumes with the Storage Manager. The following procedure describes a way to create such a list by processing the VM Guest XML definitions with `XSLT`:

Procedure 8.1 *Listing all Storage Volumes Currently Used on a VM Host Server*

1 Create an XSLT style sheet by saving the following content to a file, for example, ~/libvirt/guest_storage_list.xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="text()"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="disk">
    <xsl:text>  </xsl:text>
    <xsl:value-of select="(source/@file|source/@dev|source/@dir)[1])"/>
```

```
    <xsl:text>&#10;</xsl:text>
  </xsl:template>
</xsl:stylesheet>
```

2 Run the following commands in a shell. It is assumed that the guest's XML definitions are all stored in the default location (`/etc/libvirt/qemu`). `xsltproc` is provided by the package `libxslt`.

```
SSHEET="$HOME/libvirt/guest_storage_list.xsl"
cd /etc/libvirt/qemu
for FILE in *.xml; do
  basename $FILE .xml
  xsltproc $SSHEET $FILE
done
```

9 Configuring Virtual Machines

9.1 Adding a CD/DVD-ROM Device with Virtual Machine Manager

KVM supports CD or DVD-ROMs in VM Guest either by directly accessing a physical drive on the VM Host Server or by accessing ISO images. To create an ISO image from an existing CD or DVD, use `dd`:

```
dd if=/dev/cd_dvd_device of=my_distro.iso bs=2048
```

To add a CD/DVD-ROM device to your VM Guest proceed as follows:

1. Double-click a VM Guest entry in the Virtual Machine Manager to open its console and switch to the configuration screen with *View > Details*.

2. Click *Add Hardware* and choose *Storage* in the pop-up window. Proceed with *Forward*.

3. Change the *Device Type* to *IDE CDROM*.

4. Select *Select Managed or Other Existing Storage*.

   4a To assign the device to a physical medium, enter the path to the VM Host Server's CD/DVD-ROM device (for example, `/dev/cdrom`) next to the *Browse* button. Alternatively you may use the *Browse* button to open a file browser and then click *Browse Local* to select the device. Assigning the device

to a physical medium is only possible, when the Virtual Machine Manager was started on the VM Host Server.

4b To assign the device to an existing image, click *Browse* to choose an image from a storage pool. If the Virtual Machine Manager was started on the VM Host Server, you may alternatively choose an image from another location on the file system by clicking *Browse Local*. Select an image and close the file browser with *Choose Volume*.

5 Proceed with *Forward* to review the settings. Apply them with *Finish*, *Yes*, and *Apply*.

6 Reboot the VM Guest to make the new device available. For further information also see Section 9.3, “Ejecting and Changing Floppy or CD/DVD-ROM Media with Virtual Machine Manager” (page 79).

9.2 Adding a Floppy Device with Virtual Machine Manager

Currently KVM only supports the use of floppy disk images—using a physical floppy drive is not supported. Create a floppy disk image from an existing floppy using `dd`:

```
dd if=/dev/fd0 of=/var/lib/libvirt/images/floppy.img
```

To create an empty floppy disk image use one of the following commands:

```
# raw image
dd if=/dev/zero of=/var/lib/libvirt/images/floppy.img bs=512 count=2880

# FAT formatted image
mkfs.msdos -C /var/lib/libvirt/images/floppy.img 1440
```

To add a floppy device to your VM Guest proceed as follows:

1 Double-click a VM Guest entry in the Virtual Machine Manager to open its console and switch to the configuration screen with *View* > *Details*.

2 Click *Add Hardware* and choose *Storage* in the pop-up window. Proceed with *Forward*.

3 Change the *Device Type* to *Floppy Disk*.

4 Choose *Select Managed or Other Existing Storage* and click *Browse* to choose an existing image from a storage pool. If Virtual Machine Manager was started on the VM Host Server, you may alternatively choose an image from another location on the file system by clicking *Browse Local*. Select an image and close the file browser with *Choose Volume*.

5 Proceed with *Forward* to review the settings. Apply them with *Finish*, *Yes*, and *Apply*.

6 Reboot the VM Guest to make the new device available. For further information also see Section 9.3, “Ejecting and Changing Floppy or CD/DVD-ROM Media with Virtual Machine Manager” (page 79).

9.3 Ejecting and Changing Floppy or CD/DVD-ROM Media with Virtual Machine Manager

Regardless whether you are using the VM Host Server's physical CD/DVD-ROM device or an ISO image: before you can change the media or image of an existing device in the VM Guest, you first need to `disconnect` the media from the guest.

1 Double-click a VM Guest entry in the Virtual Machine Manager to open its console and switch to the configuration screen with *View* > *Details*.

2 Choose the Floppy or CD/DVD-ROM device and “eject” the media by clicking *Disconnect*.

3 To “insert” a new media, click *Connect*.

   3a If using the VM Host Server's physical CD/DVD-ROM device, first change the media in the device (this may require to unmount it before it can be ejected). Then choose *CD-ROM or DVD* and select the device from the drop-down list.

   3b If using an ISO image, choose *ISO image Location* and select an image by clicking *Browse*. When connecting from a remote host, you may only choose images from existing storage pools.

4 Click *OK* to finish. The new media can now be accessed in the VM Guest.

9.4 Clock Settings

Keeping the correct time in a VM Guest is one of the more difficult aspects of virtualization. Keeping the correct time is especially important for network applications and is also a prerequisite to do a live migration of a VM Guest.

TIP: Time Keeping on the VM Host Server

It is strongly recommended to ensure the VM Host Server keeps the correct time as well, for example, by utilizing NTP (see Chapter 20, *Time Synchronization with NTP* (↑*Administration Guide*) for more information).

9.4.1 Using `kvm_clock`

KVM provides a para-virtualized clock which is currently supported by SUSE Linux Enterprise Server 10 SP3 and newer and RedHat Enterprise Linux 5.4 and newer via the `kvm_clock` driver. It is strongly recommended to use `kvm_clock` when available.

Use the following command inside a VM Guest running Linux to check whether the driver `kvm_clock` has been loaded:

```
~ # dmesg | grep kvm-clock
[    0.000000] kvm-clock: cpu 0, msr 0:7d3a81, boot clock
[    0.000000] kvm-clock: cpu 0, msr 0:1206a81, primary cpu clock
[    0.012000] kvm-clock: cpu 1, msr 0:1306a81, secondary cpu clock
[    0.160082] Switching to clocksource kvm-clock
```

To check which clock source is currently used, run the following command in the VM Guest. It should output `kvm-clock`:

```
echo /sys/devices/system/clocksource/clocksource0/current_clocksource
```

IMPORTANT: `kvm-clock` and NTP

When using `kvm-clock`, it is not recommended to use NTP in the VM Guest, as well. Using NTP on the VM Host Server, however, is still recommended.

9.4.2 Other Time Keeping Methods

The para-virtualized `kvm-clock` is currently not available for SUSE Linux Enterprise Server 9 and Windows operating systems. For Windows, use the `Windows Time Service Tools` for time synchronization (see `http://technet.microsoft.com/en-us/library/cc773263%28WS.10%29.aspx` for more information).

Correct time keeping in SUSE Linux Enterprise Server 9 SP4 can be achieved by using special boot parameters:

32-bit Kernel: `clock=pmtmr`
64-bit Kernel: `ignore_lost_ticks`

10 Administrating VM Guests

10.1 Migrating VM Guests

One of the major advantages of virtualization is the fact that VM Guests are portable. When a VM Host Server needs to go down for maintenance, or when the host gets overloaded, the guests can easily be moved to another VM Host Server. KVM and Xen even support "live" migrations during which the VM Guest is constantly available (live migrations).

In order to successfully migrate a VM Guest to another VM Host Server, the following requirements need to be met:

- Host and target must have same processor manufacturer (Intel or AMD).
- Storage devices must be accessible from both machines (for example, via NFS or iSCSI) and must be configured as a storage pool on both machines (see Chapter 8, *Managing Storage* (page 67) for more information). This is also true for CD-ROM or floppy images that are connected during the move (however, you may disconnect them prior to the move as described in Section 9.3, "Ejecting and Changing Floppy or CD/DVD-ROM Media with Virtual Machine Manager" (page 79)).
- `libvirtd` needs to run on both VM Host Servers and you must be able to open a remote `libvirt` connection between the target and the source host (or vice versa). Refer to Section 7.2, "Configuring Remote Connections" (page 54) for details.

- If a firewall is running on the target host ports need to be opened to allow the migration. If you do not specify a port during the migration process, `libvirt` chooses one from the range 49152:49215. Make sure that either this range (recommended) or a dedicated port of your choice is opened in the firewall on the *target host*.

- Host and target machine should be in the same subnet on the network, otherwise networking will not work after the migration.

- No running or paused VM Guest with the same name must exist on the target host. If a shut down machine with the same name exists, its configuration will be overwritten.

10.1.1 Migrating with virt-manager

When using the Virtual Machine Manager to migrate VM Guests, it does not matter on which machine it is started. You can start Virtual Machine Manager on the source or the target host or even on a third host. In the latter case you need to be able to open remote connections to both the target and the source host.

1. Start Virtual Machine Manager and establish a connection to the target or the source host. If the Virtual Machine Manager was started neither on the target nor the source host, connections to both hosts need to be opened.

2. Right-click on the VM Guest that is to be migrated and choose *Migrate*. Make sure the guest is running or paused - it is not possible to migrate guests that are shut off.

3. Choose a *New Host* for the VM Guest. If the desired target host does not show up, make sure a connection to this host has been established.

   By default, a “live” migration is performed. If you prefer an “offline” migration where the VM Guest is paused during the migration, tick *Migrate offline*.

4. Click *Migrate* to start a migration with the default port and bandwidth.

   In order to change these defaults, make the advanced options available by clicking the triangle at *Advanced Options*. Here you can enter the target host's *Address* (IP address or hostname), a port and the bandwidth in megabit per second (Mbps). If you specify a *Port*, you must also specify an *Address*; the *Bandwidth* is optional.

5 Once the migration is complete, the *Migrate* window closes and the VM Guest is now listed on the new host in the Virtual Machine Manager Window. The original VM Guest will still be available on the target host (in state shut off).

10.1.2 Migrating with virsh

To migrate a VM Guest with `virsh migrate`, you need to have direct or remote shell access to the VM Host Server, because the command needs to be run on the host. Basically the migration command looks like this

```
virsh migrate [OPTIONS] VM_ID_or_NAME CONNECTION URI [--migrateuri
tcp://REMOTE_HOST:PORT]
```

The most important options are listed below. See `virsh help migrate` for a full list.

`--live`
: Does a live migration. If not specified, an offline migration where the VM Guest is paused during the migration, will be performed.

`--suspend`
: Does an offline migration and does not restart the VM Guest on the target host.

`--persistent`
: By default a migrated VM Guest will be migrated transient, so its configuration is automatically deleted on the target host if it is shut down. Use this switch to make the migration persistent.

`--undefinesource`
: When specified, the VM Guest definition on the source host will be deleted after a successful migration.

WARNING

It is recommended to use `--persistent` when specifying `--undefinesource`, otherwise the VM Guest configuration will be lost on both hosts when the guest is shut down on the target host.

The following examples use mercury.example.com as the source system and jupiter.example.com as the target system, the VM Guest's name is `opensuse11` with Id `37`.

```
# offline migration with default parameters
virsh migrate 37 qemu+ssh://root@jupiter.example.com/system

# transient live migration with default parameters
virsh migrate --live opensuse11 qemu+ssh://root@jupiter.example.com/system

# persistent live migration; delete VM definition on source
virsh migrate --live --persistent --undefinesource 37 \
qemu+tls://root@jupiter.example.com/system

# offline migration using port 49152
virsh migrate opensuse11 qemu+ssh://root@jupiter.example.com/system \
--migrateuri tcp://@jupiter.example.com:49152
```

10.2 Monitoring

10.2.1 Monitoring with Virtual Machine Manager

After starting Virtual Machine Manager and connecting to the VM Host Server, a CPU usage graph of all the running guests is displayed.

It is also possible to get information about disk and network usage with this tool, however, you must first activate this in the *Preferences*:

1. Run `virt-manager`.
2. Select *Edit* > *Preferences*.
3. Change the tab from *General* to *Stats*.
4. Activate the check boxes for *Disk I/O* and *Network I/O*.
5. If desired, also change the update interval or the number of samples that are kept in the history.

6 Close the *Preferences* dialog.

7 Activate the graphs that should be displayed under *View* > *Graph*.

Afterwards, the disk and network statistics are also displayed in the main window of the Virtual Machine Manager.

More precise data is available from the VNC window. Open a VNC window as described in Section 6.2, “Opening a Graphical Console” (page 38). Choose *Details* from the toolbar or the *View* menu. The statistics are displayed from the *Performance* entry of the left-hand tree menu.

10.2.2 Monitoring with kvm_stat

`kvm_stat` can be used to trace KVM performance events. It monitors `/sys/kernel/debug/kvm`, so it needs the debugfs to be mounted. On SUSE Linux Enterprise Server it should be mounted by default. In case it is not mounted, use the following command:

```
mount -t debugfs none /sys/kernel/debug
```

`kvm_stat` can be used in three different modes:

```
kvm_stat                    # update in 1 second intervals
kvm_stat -1                 # 1 second snapshot
kvm_stat -l > kvmstats.log  # update in 1 second intervals in log format
                            # can be imported to a spreadsheet
```

Example 10.1 *Typical Output of kvm_stat*

```
kvm statistics

 efer_reload                 0       0
 exits                11378946  218130
 fpu_reload              62144     152
 halt_exits             414866     100
 halt_wakeup            260358      50
 host_state_reload      539650     249
 hypercalls                  0       0
 insn_emulation        6227331  173067
 insn_emulation_fail         0       0
 invlpg                 227281      47
 io_exits               113148      18
 irq_exits              168474     127
 irq_injections         482804     123
 irq_window              51270      18
 largepages                  0       0
 mmio_exits               6925       0
 mmu_cache_miss          71820      19
 mmu_flooded             35420       9
 mmu_pde_zapped          64763      20
 mmu_pte_updated             0       0
 mmu_pte_write          213782      29
 mmu_recycled                0       0
 mmu_shadow_zapped      128690      17
 mmu_unsync                 46      -1
 nmi_injections              0       0
 nmi_window                  0       0
 pf_fixed              1553821     857
 pf_guest              1018832     562
 remote_tlb_flush       174007      37
 request_irq                 0       0
 signal_exits                0       0
 tlb_flush              394182     148
```

See http://clalance.blogspot.com/2009/01/kvm-performance-tools.html for further information on how to interpret these values.

Part III. Managing Virtual Machines with QEMU

11 Overview

QEMU is a fast, cross-platform Open Source machine emulator which can emulate a huge number of hardware architectures for you. QEMU supports two basic operation modes: with a *full system virtualization* you can run a complete unmodified operating system (VM Guest) on top of your existing system (VM Host Server), while *user mode emulation* lets you run a single Linux process compiled for a certain CPU on another CPU.

You can also use QEMU for debugging purposes - you can easily stop your running virtual machine, inspect its state and save and restore it later.

QEMU consists of the following parts:

- processor emulator (x86, PowerPC, Sparc ...)
- emulated devices (graphic card, network card, hard drives, mice ...)
- generic devices used to connect the emulated devices to the related host devices
- descriptions of the emulated machines (PC, Power Mac ...)
- debugger
- user interface used to interact with the emulator

As a virtualization solution, QEMU can be run together with the KVM kernel module. If the VM Guest hardware architecture is the same as the architecture of VM Host Server, QEMU can take advantage of the KVM acceleration.

12 Guest Installation

A virtual machine is comprised of data and operating system files that define the virtual environment. Virtual machines are hosted and controlled by the VM Host Server. This chapter provides generalized instructions for installing virtual machines.

Before creating a virtual machine, consider the following:

- If you want to use an automated installation file (AutoYaST, NetWare® Response File, or RedHat Kickstart), you should create and download it to a directory on the host machine server or make it available on the network.
- When creating sparse image files, make sure the partition on which you create them always has sufficient free space. The guest system has no means to check the host's disk space. Having no space left on the host partition causes write errors and loss of data on the guest system.
- NetWare and OES Linux virtual machines need a static IP address for each virtual machine you create.
- If you are installing Open Enterprise Server (OES) 2 Linux, you need a network installation source for OES 2 Linux software. For procedures to create the installation sources, see the *Deployment Guide* (↑*Deployment Guide*).

For further prerequisites, consult the manuals of the respective operating system to install.

12.1 Basic Installation with qemu-kvm

The `libvirt`-based tools such as `virt-manager` or `vm-install` offer convenient interfaces to set up and manage virtual machines. They act as a kind of wrapper for `qemu-kvm`. However, it is also possible to utilize `qemu-kvm` directly without using `libvirt`-based tools at all.

WARNING

Virtual machines created with `qemu-kvm` are not "visible" for the `libvirt`-based tools.

In the following example, a virtual machine with the same parameters as in Example 5.1, “Interactive Setup on the Command Line Using `vm-install`” (page 31) will be set up using `qemu-kvm`. For detailed information on the commands, refer to the respective man pages.

If you do not already have an image of a system which you want to run in a virtualized environment, you need to create one from the installation media. In such case, you need to prepare a hard disk image, and obtain an image of the installation media or the media itself.

Create a hard disk with `qemu-img`.

```
qemu-img create❶ -f raw❷ /images/sles11/hda❸ 8G❹
```

❶ The subcommand `create` tells `qemu-img` to create a new image.

❷ Specify the disk's format with the `-f` parameter. Currently, only the `raw` disk format is supported by Novell.

❸ The full path to the image file.

❹ The size of the image—8 GB in this case. The image is created as a sparse file that grows when the disk is filled with data. The specified size defines the maximum size to which the image file can grow.

After at least one hard disk image is created, you can set up a virtual machine with `qemu-kvm` that will boot into the installation system:

```
qemu-kvm -name "sles11"❶ -M pc-0.12❷ -m 768❸ \
-smp 2❹ -boot d❺ \
```

```
-drive file=/images/sles11/hda,if=virtio,index=0,media=disk,format=raw❻ \
-drive file=/isos/SLES-11-SP1-DVD-x86_64-GM-DVD1.iso,index=1,media=cdrom❼ \
-net nic,model=virtio,macaddr=52:54:00:05:11:11❽ \
-vga cirrus❾ -balloon virtio❿
```

❶ Name of the virtual machine that will be displayed in the window caption and also used for the VNC server. This name must be unique.

❷ Specifies the machine type (*Standard PC*, *ISA-only PC*, or *Intel-Mac*). Use `qemu-kvm -M ?` to display a list of valid parameters. `pc-0.12` is the default *Standard PC*.

❸ Maximum amount of memory for the virtual machine.

❹ Defines an SMP system with two processors.

❺ Specifies the boot order. Valid values are `a`, `b` (floppy 1 and 2), `c` (first harddisk), d (first CDROM), or `n` to `p` (Ether-boot from network adapter 1-3). Defaults to `c`.

❻ Defines the first (`index=0`) hard disk. It will be accessed as a paravirtualized (`if=virtio`) drive in `raw` format.

❼ The second (`index=1`) image drive will act as a CD-ROM.

❽ Defines a paravirtualized (`model=virtio`) network adapter with the MAC address `52:54:00:05:11:11`. Be sure to specify a unique MAC address, otherwise a network conflict may occur.

❾ Specifies the graphic card. If you specify *`none`*, the graphic card will be disabled and it is then possible to connect and view the graphical output through VNC.

❿ Defines the paravirtualized balloon device that allows to dynamically change the amount of memory (up to the maximum value specified with the parameter `-m`).

After the installation of the guest operating system finishes, you can easily start the related virtual machine without the need to specify the CD-ROM device:

```
qemu-kvm -name "sles11" -M pc-0.12 -m 768 \
-smp 2 -boot c \
-drive file=/images/sles11/hda,if=virtio,index=0,media=disk,format=raw \
-net nic,model=virtio,macaddr=52:54:00:05:11:11 \
-vga cirrus -balloon virtio
```

12.2 Managing Disk Images with qemu-img

In the previous section (see Section 12.1, “Basic Installation with `qemu-kvm`” (page 94), we used the `qemu-img` command to create an image of a hard disk. You can, however, use `qemu-img` for a general disk image manipulation. This section introduces useful `qemu-img` subcommands to help manage the disk images flexibly.

12.2.1 General Information on qemu-img Invocation

`qemu-img` uses subcommands (like `zypper` does) to do specific tasks. Each subcommand understands a different set of options. Some of these options are general and used by more of these subcommands, while some of them are unique to the related subcommand. See the qemu-img manual page (`man 1 qemu-img`) for a complete list of all supported options. `qemu-img` uses the following general syntax:

```
qemu-img subcommand [options]
```

and supports the following subcommands:

`create`
: Creates a new disk image on the filesystem.

`check`
: Checks an existing disk image for errors.

`convert`
: Converts an existing disk image to a new one in a different format.

`info`
: Displays information about the relevant disk image.

`snapshot`
: Manages snapshots of an existing disk images.

`commit`
Applies changes made to an existing disk image.

`rebase`
Creates a new base image based on an existing image.

12.2.2 Creating, Converting and Checking Disk Images

This section describes how to create disk images, check their condition, convert a disk image from one format to another, and get detailed information about a particular disk image.

qemu-img create

Use `qemu-img create` to create a new disk image for your VM Guest operating system. The command uses the following syntax:

```
qemu-img create -f fmt❶ -o options❷ fname❸ size❹
```

❶ The format of the target image. To get a complete list of image formats supported by QEMU, run `qemu-img -h` and look at the last line of the output. Currently, Novell only supports the `raw` image format. It is a plain binary image with no advanced features, and, therefore, very portable from and to other emulators. Moreover, if your filesystem supports *sparse files* (most UNIX modern filesystems or NTFS do), the image occupies only the space actually used by its data.

❷ Some of the image formats support additional options to be passed on the command line. You can specify them here with the `-o` option. The `raw` image format supports only the `size` option, so it is possible to insert `-o size=8G` instead of adding the size option at the end of the command.

❸ Path to the target disk image to be created.

❹ Size of the target disk image (if not already specified with the `-o size=<image_size>` option. Optional suffixes for the image size are `K` (kilobyte), `M` (megabyte), `G` (gigabyte), or `T` (terabyte).

To create a new disk image `sles11sp1.raw` in the directory `/images` growing up to a maximum size of 4 GB, run the following command:

```
tux@venus:~> qemu-img create -f raw -o size=4G /images/sles11sp1.raw
Formatting '/images/sles11sp1.raw', fmt=raw size=4294967296

tux@venus:~> ls -l /images/sles11sp1.raw
-rw-r--r-- 1 tux users 4294967296 Nov 15 15:56 /images/sles11sp1.raw

tux@venus:~> qemu-img info /images/sles11sp1.raw
image: /images/sles11sp1.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 0
```

As you can see, the *virtual* size of the newly created image is 4 GB, but the actual reported disk size is 0 as no data has been written to the image yet.

qemu-img convert

Use `qemu-img convert` to convert disk images to another format. To get a complete list of image formats supported by QEMU, run `qemu-img -h` and look at the last line of the output. Currently, Novell only supports the `raw` image format. The command uses the following syntax:

```
qemu-img convert -c❶ -f fmt❷ -O out_fmt❸ -o options❹ fname❺ out_fname❻
```

❶ Applies compression on the target disk image. Only `qcow` and `qcow2` formats support compression.

❷ The format of the source disk image. It is autodetected in most cases and can therefore be omitted.

❸ The format of the target disk image.

❹ Specify additional options relevant for the target image format. Use `-o ?` to view the list of options supported by the target image format:

❺ Path to the source disk image to be converted.

❻ Path to the converted target disk image.

```
tux@venus:~> qemu-img convert -O vmdk /images/sles11sp1.raw \
/images/sles11sp1.vmdk

tux@venus:~> ls -l /images/
-rw-r--r-- 1 tux users 4294967296 16. lis 10.50 sles11sp1.raw
-rw-r--r-- 1 tux users 2574450688 16. lis 14.18 sles11sp1.vmdk
```

To see a list of options relevant for the selected target image format, run the following command (replace `with your image fomrmat`):

```
tux@venus:~> qemu-img convert -O vmdk /images/sles11sp1.raw \
/images/sles11sp1.vmdk -o ?
Supported options:
size             Virtual disk size
backing_file     File name of a base image
compat6          VMDK version 6 image
```

qemu-img check

Use `qemu-img check` to check the existing disk image for errors. Not all disk image formats support this feature. The command uses the following syntax:

```
qemu-img check -f fmt❶ fname❷
```

❶ The format of the source disk image. It is autodetected in most cases and can therefore be omitted.

❷ Path to the source disk image to be converted.

If no error is found, the command returns no output. Otherwise, the type and number of errors found is shown.

```
tux@venus:~> qemu-img check -f qcow2 /images/sles11sp1.qcow2
ERROR: invalid cluster offset=0x2af0000
[...]
ERROR: invalid cluster offset=0x34ab0000
378 errors were found on the image.
```

Increasing the Size of an Existing Disk Image

When creating a new image, you must specify its maximum size before the image is created (see Section “qemu-img create” (page 97)). After you install and run VM Guest for some time, the initial size of the image may no longer be sufficient and you need to add more space to it.

To increase the size of an existing disk image, follow these steps:

1 Create a new additional disk image with the `raw` format. It must be as big as the size by which you decided to increase your existing disk image (2 GB in our example).

```
tux@venus:~> qemu-img create -f raw /images/additional.raw 2G
Formatting '/images/additional.raw', fmt=raw size=2147483648
```

2 Convert your existing disk image from your native format into the `raw` one. If your disk image format already is `raw`, skip this step.

```
tux@venus:~> qemu-img convert -f vmdk -O raw /images/sles11sp1.vmdk \
/images/original.raw
```

3 Append the newly created disk image `/images/additional.raw` to your converted existing one `/images/original.raw`.

```
tux@venus:~> cat /images/additional.raw >> /images/original.raw
```

4 If the format of your existing image was not `raw`, convert it back to its original format.

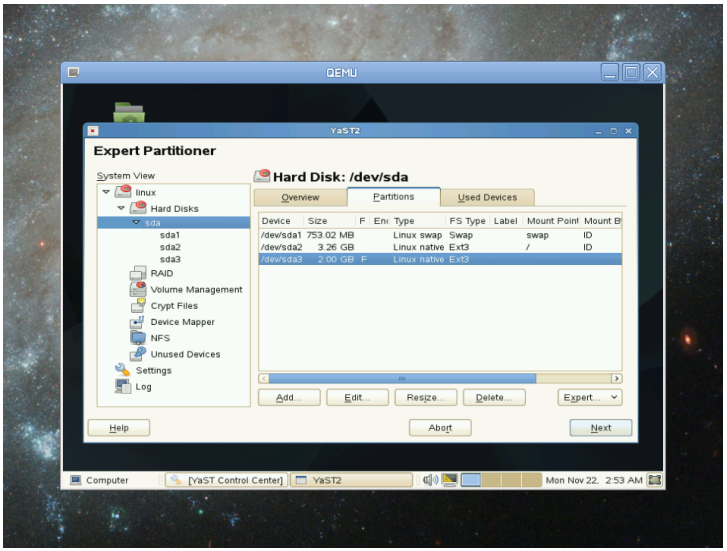
```
tux@venus:~> qemu-img convert -f raw /images/original.raw -O vmdk \
/images/expanded.vmdk
```

5 View information about the new disk image.

```
tux@venus:~> qemu-img info /images/expanded.vmdk
image: /images/expanded.vmdk
file format: vmdk
virtual size: 6.0G (6442450944 bytes)
disk size: 2.4G
```

The `/images/expanded.vmdk` image now contains an empty space of 2 GB after the final partition. You can resize the existing partitions or add new ones.

Figure 12.1 *New 2GB Partition in Guest YaST Partitioner*



12.2.3 Managing Snapshots of Virtual Machines with qemu-img Snapshot

Virtual machine snapshots are snapshots of the complete environment in which VM Guest is running. The snapshot includes the state of the processor (CPU), memory (RAM), devices, and all writable disks.

Snapshots are helpful when you need to save your virtual machine in a particular state. For example, after you configured network services on a virtualized server and want to quickly start the virtual machine in the same state you last saved it. Or you can create a snapshot after the virtual machine has been powered off to create a backup state before you try something experimental and possibly make VM Guest unstable. This section introduces the latter case, while the former is described in Chapter 14, *Administrating Virtual Machines with QEMU Monitor* (page 133).

To use snapshots, your VM Guest must contain at least one writable hard disk image in `qcow2` format. This device is usually the first virtual hard disk.

NOTE

Currently, only the `raw` disk image format is supported by Novell.

Virtual machine snapshots are created with the `savevm` command in the interactive QEMU monitor. You can assign a 'tag' to each snapshot which makes its identification easier. For more information on QEMU monitor, see Chapter 14, *Administrating Virtual Machines with QEMU Monitor* (page 133).

Once your `qcow2` disk image contains saved snapshots, you can inspect them with the `qemu-img snapshot` command.

WARNING

Do not create or delete virtual machine snapshots with the `qemu-img snapshot` command while the virtual machine is running. Otherwise, you can damage the disk image with the state of the virtual machine saved.

Listing Existing Snapshots

Use `qemu-img snapshot -l` *`disk_image`* to view a list of all existing snapshots saved in the `disk_image` image. You can get the list even while the VM Guest is running.

```
tux@venus:~> qemu-img snapshot -l /images/sles11sp1.qcow2
Snapshot list:
ID❶        TAG❷                VM SIZE❸        DATE❹           VM CLOCK❺
1         booting                4.4M 2010-11-22 10:51:10   00:00:20.476
2         booted                 184M 2010-11-22 10:53:03   00:02:05.394
3         logged_in              273M 2010-11-22 11:00:25   00:04:34.843
4         ff_and_term_running    372M 2010-11-22 11:12:27   00:08:44.965
```

❶ Unique identification number of the snapshot. Usually auto-incremented.

❷ Unique description string of the snapshot. It is meant as a human readable version of the ID.

❸ The disk space occupied by the snapshot. Note that the more memory is consumed by running applications, the bigger the snapshot is.

❹ Time and date the snapshot was created.

❺ The current state of the virtual machine's clock.

Creating Snapshots of a Powered-Off Virtual Machine

Use `qemu-img snapshot -c snapshot_title disk_image` to create a snapshot of the current state of a virtual machine which was previously powered off.

```
tux@venus:~> qemu-img snapshot -c backup_snapshot /images/sles11sp1.qcow2
```

```
tux@venus:~> qemu-img snapshot -l /images/sles11sp1.qcow2
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         booting                4.4M 2010-11-22 10:51:10   00:00:20.476
2         booted                 184M 2010-11-22 10:53:03   00:02:05.394
3         logged_in              273M 2010-11-22 11:00:25   00:04:34.843
4         ff_and_term_running    372M 2010-11-22 11:12:27   00:08:44.965
5         backup_snapshot           0 2010-11-22 14:14:00   00:00:00.000
```

Once something breaks in your VM Guest and you need to restore the state of the saved snapshot (id 5 in our example), power off your VM Guest and do the following:

```
tux@venus:~> qemu-img snapshot -a 5 /images/sles11sp1.qcow2
```

Next time you run the virtual machine with `qemu-kvm`, it will be in the state of snapshot number 5.

NOTE

The `qemu-img snapshot -c` command is not related to the `savevm` command of QEMU monitor (see Chapter 14, *Administrating Virtual Machines with QEMU Monitor* (page 133). For example, you cannot apply a snapshot with `qemu-img snapshot -a` on a snapshot created with `savevm` in QEMU's monitor.

Deleting Snapshots

Use `qemu-img snapshot -d snapshot_id disk_image` to delete old or unneeded snapshots of a virtual machine. This saves some disk space inside the `qcow2` disk image as the space occupied by the snapshot data is restored:

```
tux@venus:~> qemu-img snapshot -d 2 /images/sles11sp1.qcow2
```

12.2.4 Disk Images Effectively

Imagine the following real-life situation: you are a server administrator who runs and manages a number of virtualized operating systems. One group of these systems are based on one specific distribution, while another group (or groups) is based on different versions of the distribution or even on a different (and maybe non-Unix) platform. And to make the case even more complex, individual virtual guest systems based on the same distribution usually differ according to the department and deployment: a file server typically uses different setup and services than a Web server does, while both may still be based on SUSE® Linux Enterprise Server 11 SP1.

With QEMU it is possible to create a 'base' disk images. You can use them as template virtual machines. These base images will save you plenty of time because you will never need to install the same operation system more than once.

Base and Derived Images

First, build a disk image as usual and install the target system on it. For more information, see Section 12.1, “Basic Installation with `qemu-kvm`” (page 94) and Section 12.2.2, “Creating, Converting and Checking Disk Images” (page 97). Then build a new image while using the first one as a base image. The base image is also called a 'backing' file. After your new 'derived' image is built, never boot the base image again, but boot the derived image instead. Several derived images may depend on one base image at the same time. Therefore, changing the base image can damage the dependencies. While using your derived image, QEMU writes changes to it and uses the base image only for reading.

It is a good practice to create a base image from a freshly installed (and, if needed, registered) operating system with no patches applied and no additional applications installed or removed. Later on, you can create another base image with the latest patches applied and based on the original base image.

Creating Derived Images

NOTE

Currently, Novell supports only the `raw` image format.

While you can use the `raw` format for base images, you cannot use it for derived images because the `raw` format does not support the `backing_file` option. Use the `qcow2` format for the derived images.

For example, `/images/sles11sp1_base.raw` is the base image holding a freshly installed system.

```
tux@venus:~> qemu-img info /images/sles11sp1_base.raw
image: /images/sles11sp1_base.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 2.4G
```

The image's reserved size is 4 GB, the actual size is 2.4 GB, and its format is `raw`. Create an image derived from the `/images/sles11sp1_base.raw` base image with:

```
tux@venus:~> qemu-img create -f qcow2 /images/sles11sp1_derived.qcow2 \
-o backing_file=/images/sles11sp1_base.raw
Formatting '/images/sles11sp1_derived.qcow2', fmt=qcow2 size=4294967296 \
backing_file='/images/sles11sp1_base.raw' encryption=off cluster_size=0
```

Look at the derived image details:

```
tux@venus:~> qemu-img info /images/sles11sp1_derived.qcow2
image: /images/sles11sp1_derived.qcow2
file format: qcow2
virtual size: 4.0G (4294967296 bytes)
disk size: 140K
cluster_size: 65536
backing file: /images/sles11sp1_base.raw \
(actual path: /images/sles11sp1_base.raw)
```

Although the reserved size of the derived image is the same as the size of the base image (4 GB), the actual size is 140 kB only. The reason is that only changes made to the system inside the derived image are saved and the new image is in fact empty. Run the derived virtual machine, register it if needed, and apply the latest patches. Do any other changes in the system such as removing unneeded or installing new software packages. Then shut the VM Guest down and examine its details once more:

```
tux@venus:~> qemu-img info /images/sles11sp1_derived.qcow2
image: /images/sles11sp1_derived.qcow2
file format: qcow2
virtual size: 4.0G (4294967296 bytes)
disk size: 1.1G
cluster_size: 65536
backing file: /images/sles11sp1_base.raw \
(actual path: /images/sles11sp1_base.raw)
```

The `disk size` value has grown to 1.1 GB, which is the disk space occupied by the changes on the filesystem compared to the base image.

Rebasing Derived Images

Once you modify the derived image (apply patches, install specific applications, or change the environment settings etc.) into a satisfactory shape, at some point you probably want to create a new base image 'merged' from the base image and the derived one. Your first base image (`/images/sles11sp1_base.raw`) holds a freshly installed system and can be a template for new modified base images, while the new one can contain the same system as the first one plus all security and update patches applied, for example. After you created this new base image, you can use it as a template for more specialized derived images as well. The new base image becomes independent of the original one. The process of creating base images from derived ones is called 'rebasing':

```
tux@venus:~> qemu-img convert /images/sles11sp1_derived.qcow2 \
-O raw /images/sles11sp1_base2.raw
```

This command created the new base image `/images/sles11sp1_base2.raw` using the `raw` format.

```
tux@venus:~> qemu-img info /images/sles11sp1_base2.raw
image: /images/sles11sp1_base2.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 2.8G
```

The new image is 0.4 gigabytes bigger than the original base image. It uses no backing file, which means it is independent and therefore a base image. You can create new derived images based upon it. This lets you create a sophisticated hierarchy of virtual disk images for your organization, saving a lot of time and work.

Mounting an Image on VM Host Server

Sometimes it is useful to mount a virtual disk image under the host system. For example, if VM Host Server does not have a network support, this can be the only way to transfer files in and out of VM Guest.

Linux systems can mount an internal partition of a `raw` disk image using a 'loopback' device. The first example procedure is more complex but more illustrative, while the second one is straightforward:

Procedure 12.1 *Mounting Disk Image by Calculating Partition Offset*

1 Set a *loop* device on the disk image whose partition you want to mount.

```
tux@venus:~> losetup /dev/loop0 /images/sles11sp1_base.raw
```

2 Find the *sector size* and the starting *sector number* of the partition you want to mount.

```
tux@venus:~> fdisk -lu /dev/loop0

Disk /dev/loop0: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
Units = sectors of 1 * 512 = 512❶ bytes
Disk identifier: 0x000ceca8

       Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1              63     1542239      771088+  82  Linux swap
/dev/loop0p2   *     1542240❷    8385929     3421845   83  Linux
```

❶ The disk sector size.

❷ The starting sector of the partition.

3 Calculate the partition start offset:

```
sector_size * sector_start = 512 * 1542240 = 789626880
```

4 Delete the loop and mount the partition inside the disk image with the calculated offset on a prepared directory.

```
tux@venus:~> losetup -d /dev/loop0
tux@venus:~> mount -o loop,offset=789626880 \
/images/sles11sp1_base.raw /mnt/sles11sp1/
tux@venus:~> ls -l /mnt/sles11sp1/
total 112
drwxr-xr-x   2 root root  4096 Nov 16 10:02 bin
drwxr-xr-x   3 root root  4096 Nov 16 10:27 boot
drwxr-xr-x   5 root root  4096 Nov 16 09:11 dev
[...]
drwxrwxrwt  14 root root  4096 Nov 24 09:50 tmp
drwxr-xr-x  12 root root  4096 Nov 16 09:16 usr
drwxr-xr-x  15 root root  4096 Nov 16 09:22 var
```

5 Copy a file or files on the mounted partition and unmount it when finished.

```
tux@venus:~> cp /etc/X11/xorg.conf /mnt/sles11sp1/root/tmp
tux@venus:~> ls -l /mnt/sles11sp1/root/tmp
tux@venus:~> umount /mnt/sles11sp1/
```

Procedure 12.2 *Mounting Disk Image while Utilizing kpartx*

1 Set a *loop* device on the disk image whose partition you want to mount.

```
tux@venus:~> losetup /dev/loop0 /images/sles11sp1_base.raw
```

2 Create a device map from the disk image's partitions.

```
tux@venus:~> kpartx -a /dev/loop0
```

3 Mount any partition of the disk image on a prepared mount point.

```
tux@venus:~> mount /dev/mapper/loop0p1 /mnt/p1
```

You can replace `loop0p1` with the number of the partition you want to mount, for example `loop0p3` to mount the third partition on the disk image.

4 Copy or move files or directories to and from the mounted partition as you like. Once you finish, unmount the partition and delete the loop.

```
tux@venus:~> umount /mnt/p1
tux@venus:~> losetup -d /dev/loop0
```

WARNING

Never mount a partition of an image of a running virtual machine. This could corrupt the partition and break the whole VM Guest.

Running Virtual Machines with qemu-kvm

13

Once you have a virtual disk image ready (for more information on disk images, see Section 12.2, “Managing Disk Images with `qemu-img`” (page 96)), it is time to start the related virtual machine. Section 12.1, “Basic Installation with `qemu-kvm`” (page 94) introduced simple commands to install and run VM Guest. This chapter focuses on a more detailed explanation of `qemu-kvm` usage, and shows solutions of more specific tasks. For a complete list of `qemu-kvm`'s options, see its manual page (`man 1 qemu-kvm`).

13.1 Basic qemu-kvm Invocation

The `qemu-kvm` command uses the following syntax:

```
qemu-kvm options❶ disk_img❷
```

❶ `qemu-kvm` understands a large number of options. Most of them define parameters of the emulated hardware, while others affect more general emulator behavior. If you do not supply any options, default values are used, and you need to supply the path to a disk image to be run.

❷ Path to the disk image holding the guest system you want to virtualize. `qemu-kvm` supports a large number of image formats. Use `qemu-img --help` to list them. If you do not supply the path to a disk image as a separate argument, you have to use the `-drive file=` option.

13.2 General qemu-kvm Options

This section introduces general `qemu-kvm` options and options related to the basic emulated hardware, such as virtual machine's processor, memory, model type, or time processing methods.

`-name` *`name_of_guest`*
: Specifies the name of the running guest system. The name is displayed in the window caption and also used for the VNC server.

`-boot` *`options`*
: Specifies the order in which the defined drives will be booted. Drives are represented by letters, where 'a' and 'b' stands for the floppy drives 1 and 2, 'c' stands for the first hard disk, 'd' stands for the first CD-ROM drive, and 'n' to 'p' stand for Etherboot network adapters.

  For example, `qemu-kvm [...] -boot order=ndc` first tries to boot from network, then from the first CD-ROM drive, and finally from the first hard disk.

`-pidfile` *`fname`*
: Stores the QEMU's process identification number (PID) in a file. This is useful if you run QEMU from a script.

`-nodefaults`
: By default QEMU creates basic virtual devices even if you do not specify them on the command line. This option turns this feature off, and you must specify every single device manually, including graphical and network cards, parallel or serial ports, or virtual consoles. Even QEMU monitor is not attached by default.

`-daemonize`
: 'Daemonizes' the QEMU process after it is started. QEMU will detach from the standard input and standard output after it is ready to receive connections on any of its devices.

13.2.1 Basic Virtual Hardware

`-M` *machine_type*

Specifies the type of the emulated machine. Run `qemu-kvm -M ?` to view a list of supported machine types.

```
tux@venus:~> qemu-kvm -M ?
Supported machines are:
pc          Standard PC (alias of pc-0.12)
pc-0.12     Standard PC (default)
pc-0.11     Standard PC, qemu 0.11
pc-0.10     Standard PC, qemu 0.10
isapc       ISA-only PC
mac         Intel-Mac
```

NOTE

Currently, Novell supports only the default `pc-0.12` machine type.

`-m` *megabytes*

Specifies how many megabytes are used for the virtual RAM size. Default is 128 MB.

`-balloon virtio`

Specifies a paravirtualized device to dynamically change the amount of virtual RAM memory assigned to VM Guest. The top limit is the amount of memory specified with `-m`.

`-cpu` *cpu_model*

Specifies the type of the processor (CPU) model. Run `qemu-kvm -cpu ?` to view a list of supported CPU models.

```
tux@venus:~> qemu-kvm -cpu ?
x86           qemu64
x86           phenom
x86         core2duo
x86            kvm64
x86           qemu32
x86          coreduo
x86              486
x86          pentium
x86         pentium2
x86         pentium3
x86           athlon
x86             n270
```

NOTE

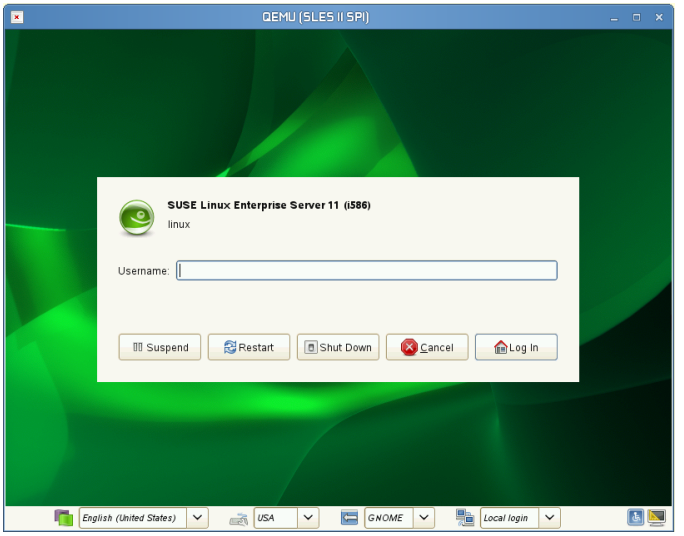
Currently, Novell supports only the `kvm64` CPU model.

`-smp` *`number_of_cpus`*
: Specifies how many CPUs will be emulated. QEMU supports up to 255 CPUs on the PC platform. This option also takes other CPU-related parameters, such as number of *sockets*, number of *cores* per socket, or number of *threads* per core.

Following is an example of a working `qemu-kvm` command line:

```
qemu-kvm -name "SLES 11 SP1" -M pc-0.12 -m 512 -cpu kvm64 \
-smp 2 /images/sles11sp1.raw
```

Figure 13.1 *QEMU Window with SLES 11 SP1 as VM Guest*



`-no-acpi`
: Disables ACPI support. Try to use it if VM Guest reports problems with ACPI interface.

`-S`
: QEMU starts with CPU stopped. To start CPU, enter `c` in QEMU monitor. For more information, see Chapter 14, *Administrating Virtual Machines with QEMU Monitor* (page 133).

13.2.2 Storing and Reading Configuration of Virtual Devices

`-readconfig` *`cfg_file`*

Instead of entering the devices configuration options on the command line each time you want to run VM Guest, `qemu-kvm` can read it from a file which was either previously saved with `-writeconfig` or edited manually.

`-writeconfig` *`cfg_file`*

Dumps the current virtual machine devices configuration to a text file. It can be consequently re-used with the `-readconfig` option.

```
tux@venus:~> qemu-kvm -name "SLES 11 SP1" -M pc-0.12 -m 512 -cpu kvm64 \
-smp 2 /images/sles11sp1.raw -writeconfig /images/sles11sp1.cfg
(exited)
tux@venus:~> more /images/sles11sp1.cfg
# qemu config file

[drive]
  index = "0"
  media = "disk"
  file = "/images/sles11sp1_base.raw"
```

This way you can effectively manage the configuration of your virtual machines' devices in a well-arranged way.

13.2.3 Guest Real-time Clock

`-rtc` *`options`*

Specifies the way the RTC is handled inside VM Guest. By default, the clock of VM Guest is derived from that of the host system. Therefore, it is recommended that the host system clock is synchronized with an accurate external clock (for example, via NTP service).

If you need to isolate the VM Guest clock from the host one, specify `clock=vm` instead of the default `clock=host`.

You can also specify a 'starting point' for VM Guest clock with the `base` option:

```
qemu-kvm [...] -rtc clock=vm,base=2010-12-03T01:02:00
```

Instead of a timestamp, you can specify `utc` or `localtime`. The former instructs VM Guest to start at UTC (Coordinated Universal Time, see http://en.wikipedia.org/wiki/Utc), while the latter applies the local time setting.

13.3 QEMU Virtual Devices

QEMU virtual machines emulate all devices needed to run VM Guest. QEMU supports, for example, several types of network cards, block devices (hard and removable drives), USB devices, character devices (serial and parallel ports), or multimedia devices (graphic and sound cards). For satisfactory operation and performance of the virtual machine, some or all of these devices must be configured correctly. This section introduces options to configure various types of supported devices.

13.3.1 Block Devices

Block devices are vital for virtual machines. In general, these are fixed or removable storage media usually referred to as 'drives'. One of the connected hard drives typically holds the guest operating system to be virtualized.

Virtual machine drives are defined with `-drive`. This option uses many suboptions, some of which are described in this section. For their complete list, see the manual page (`man 1 qemu-kvm`).

Suboptions for the `-drive` Option

`file=image_fname`
: Specifies the path to the disk image which will be used with this drive. If not specified, an empty (removable) drive is assumed.

`if=drive_interface`
: Specifies the type of interface to which the drive is connected. Currently only `floppy`, `ide`, or `virtio` are supported by Novell. `virtio` defines a paravirtualized disk driver. Default is `ide`.

`index=index_of_connector`
: Specifies the index number of a connector on the disk interface (see the `if` option) where the drive is connected. If not specified, the index is automatically incremented.

`media=type`
: Specifies the type of the media. Can be `disk` for hard disks, or `cdrom` for removable CD-ROM drives.

`format=img_fmt`
: Specifies the format of the connected disk image. If not specified, the format is autodetected. Currently, Novell supports only the `raw` format.

`boot='on' or 'off'`
: Specifies whether booting from 'uncommon' devices (such as `virtio`) is allowed. If not specified, disks connected via `virtio` interface will refuse to boot.

`cache=method`
: Specifies the caching method for the drive. Possible values are `unsafe`, `writethrough`, `writeback`, or `none`. For the `qcow2` image format, choose `writeback` if you care about performance. `none` disables the host page cache and, therefore, is the safest option. Default is `writethrough`.

TIP

To simplify defining of block devices, QEMU understands several shortcuts which you may find handy when entering the `qemu-kvm` command line.

You can use

```
qemu-kvm -cdrom /images/cdrom.iso
```

instead of

```
qemu-kvm -drive file=/images/cdrom.iso,index=2,media=cdrom
```

and

```
qemu-kvm -hda /images/imagei1.raw -hdb /images/image2.raw -hdc \
/images/image3.raw -hdd /images/image4.raw
```

instead of

```
qemu-kvm -drive file=/images/image1.raw,index=0,media=disk \
-drive file=/images/image2.raw,index=1,media=disk \
-drive file=/images/image3.raw,index=2,media=disk \
-drive file=/images/image4.raw,index=3,media=disk
```

TIP: Using Host Drives Instead of Images

Normally you will use disk images (see Section 12.2, “Managing Disk Images with `qemu-img`” (page 96)) as disk drives of the virtual machine. However, you can also use existing VM Host Server disks, connect them as drives, and access them from VM Guest. Use the host disk device directly instead of disk image filenames.

To access the host CD-ROM drive, use

```
qemu-kvm [...] -drive file=/dev/cdrom,media=cdrom
```

To access the host hard disk, use

```
qemu-kvm [...] -drive file=/dev/hdb,media=disk
```

When accessing the host hard drive from VM Guest, always make sure the access is *read-only*. You can do so by modifying the host device permissions.

13.3.2 Graphic Devices and Display Options

This section describes QEMU options affecting the type of the emulated video card and the way VM Guest graphical output is displayed.

Defining Video Cards

QEMU uses `-vga` to define a video card used to display VM Guest graphical output. The `-vga` option understands the following values:

`none`
: Disables video cards on VM Guest (no video card is emulated). You can still access the running VM Guest via the QEMU monitor and the serial console.

`std`
: Emulates a standard VESA 2.0 VBE video card. Use it if you intend to use high display resolution on VM Guest.

`cirrus`
: Emulates Cirrus Logic GD5446 video card. Good choice if you insist on high compatibility of the emulated video hardware. Most operating systems (even Windows 95) recognize this type of card.

TIP

For best video performance with the `cirrus` type, use 16-bit color depth both on VM Guest and VM Host Server.

Display Options

The following options affect the way VM Guest graphical output is displayed.

`-nographic`
: Disables QEMU's graphical output. The emulated serial port is redirected to the console.

  After starting the virtual machine with `-nographic`, press Ctrl + A H in the virtual console to view the list of other useful shortcuts, for example, to toggle between the console and the QEMU monitor.

```
tux@venus:~> qemu-kvm -hda /images/sles11sp1_base.raw -nographic

C-a h    print this help
C-a x    exit emulator
C-a s    save disk data back to file (if -snapshot)
C-a t    toggle console timestamps
C-a b    send break (magic sysrq)
C-a c    switch between console and monitor
C-a C-a  sends C-a
(pressed C-a c)

QEMU 0.12.5 monitor - type 'help' for more information
(qemu)
```

`-no-frame`
: Disables decorations for the QEMU window. Convenient for dedicated desktop workspace.

`-full-screen`
Starts QEMU graphical output in full screen mode.

`-no-quit`
Disables the 'close' button of QEMU window and prevents it from being closed by force.

`-alt-grab, -ctrl-grab`
By default QEMU window releases the 'captured' mouse after Ctrl + Alt is pressed. You can change the key combination to either Ctrl + Alt + Shift (`-alt-grab`), or Right Ctrl (`-ctrl-grab`).

13.3.3 USB Devices

To emulate USB devices in QEMU you first need to enable the generic USB driver with the `-usb` option. Then you can specify individual devices with the `-usbdevice` option. Although QEMU supports much more types of USB devices, Novell currently only supports the types `mouse` and `tablet`.

Types of USB devices for the `-usbdevice` Option

`mouse`
Emulates a virtual USB mouse. This option overrides the default PS/2 mouse emulation. The following example shows the hardware status of a mouse on VM Guest started with `qemu-kvm [...] -usbdevice mouse`:

```
tux@venus:~> hwinfo --mouse
20: USB 00.0: 10503 USB Mouse
[Created at usb.122]
UDI: /org/freedesktop/Hal/devices/usb_device_627_1_1_if0
[...]
Hardware Class: mouse
Model: "Adomax QEMU USB Mouse"
Hotplug: USB
Vendor: usb 0x0627 "Adomax Technology Co., Ltd"
Device: usb 0x0001 "QEMU USB Mouse"
[...]
```

`tablet`
Emulates a pointer device that uses absolute coordinates (such as touchscreen). This option overrides the default PS/2 mouse emulation. The tablet device is useful

if you are viewing VM Guest via the VNC protocol. See Section 13.5, “Viewing VM Guest with VNC” (page 126) for more information.

13.3.4 Character Devices

Use `-chardev` to create a new character device. The option uses the following general syntax:

```
qemu-kvm [...] -chardev backend_type,id=id_string
```

where *`backend_type`* can be one of `null`, `socket`, `udp`, `msmouse`, `vc`, `file`, `pipe`, `console`, `serial`, `pty`, `stdio`, `braille`, `tty`, or `parport`. All character devices must have a unique identification string up to 127 characters long. It is used to identify the device in other related directives. For the complete description of all back-end's suboptions, see the man page (`man 1 qemu-kvm`). A brief description of the available `backends` follows:

`null`
: Creates an empty device which outputs no data and drops any data it receives.

`stdio`
: Connects to QEMU's process standard input and standard output.

`socket`
: Creates a two-way stream socket. If *`path`* is specified, a Unix socket is created:

```
qemu-kvm [...] -chardev \
socket,id=unix_socket1,path=/tmp/unix_socket1,server
```

The *`server`* suboption specifies that the socket is a listening socket.

If *`port`* is specified, a TCP socket is created:

```
qemu-kvm [...] -chardev \
socket,id=tcp_socket1,host=localhost,port=7777,server,nowait
```

The command creates a local listening (`server`) TCP socket on port 7777. QEMU will not block waiting for a client to connect to the listening port (`nowait`).

`udp`
: Sends all network traffic from VM Guest to a remote host over the UDP protocol.

```
qemu-kvm [...] -chardev udp,id=udp_fwd,host=mercury.example.com,port=7777
```

The command binds port 7777 on the remote host mercury.example.com and sends VM Guest network traffic there.

`vc`

Creates a new QEMU text console. You can optionally specify the dimensions of the virtual console:

```
qemu-kvm [...] -chardev vc,id=vc1,width=640,height=480 -mon chardev=vc1
```

The command creates a new virtual console called `vc1` of the specified size, and connects the QEMU monitor to it.

`file`

Logs all traffic from VM Guest to a file on VM Host Server. The `path` is required and will be created if it does not exist.

```
qemu-kvm [...] -chardev file,id=qemu_log1,path=/var/log/qemu/guest1.log
```

By default QEMU creates a set of character devices for serial and parallel ports, and a special console for QEMU monitor. You can, however, create your own character devices and use them for just mentioned purposes. The following options will help you:

`-serial` *`char_dev`*

Redirects the VM Guest's virtual serial port to a character device *`char_dev`* on VM Host Server. By default, it is a virtual console (`vc`) in graphical mode, and `stdio` in non-graphical mode. The `-serial` understands many suboptions. See the manual page `man 1 qemu-kvm` for their complete list.

You can emulate up to 4 serial ports. Use `-serial none` to disable all serial ports.

`-parallel` *`device`*

Redirects the VM Guest's parallel port to a *`device`*. This option supports the same devices as `-serial`.

TIP

If your VM Host Server is Linux, you can directly use the hardware parallel port devices `/dev/parportN` where `N` is the number of the port.

You can emulate up to 3 parallel ports. Use `-parallel none` to disable all parallel ports.

`-monitor char_dev`
: Redirects the QEMU monitor to a character device *`char_dev`* on VM Host Server. This option supports the same devices as `-serial`. By default, it is a virtual console (`vc`) in a graphical mode, and `stdio` in non-graphical mode.

13.4 Networking with QEMU

Use the `-net` option to define a network interface and a specific type of networking for your VM Guest. Currently, Novell supports the following options: `none`, `nic`, `user`, and `tap`. For a complete list of `-net` suboptions, see the man page (`man 1 qemu-kvm`).

Supported `-net` Suboptions

`none`
: Disables a network card emulation on VM Guest. Only the loopback `lo` network interface is available.

`nic`
: Creates a new Network Interface Card (NIC) and connects it to a specified Virtual Local Area Network (VLAN). For more information, see Section 13.4.1, “Defining a Network Interface Card” (page 121)

`user`
: Specifies a user-mode networking. For more information , see Section 13.4.2, “User-mode Networking” (page 122).

`tap`
: Specifies a bridged networking. For more information, see Section 13.4.3, “Bridged Networking” (page 124).

13.4.1 Defining a Network Interface Card

Use `-net nic` to add a new emulated network card:

```
qemu-kvm [...] -net nic,vlan=1❶,macaddr=00:16:35:AF:94:4B❷,\
model=virtio❸,name=ncard1❹
```

❶ Connects the network interface to VLAN number 1. You can specify your own number, it is mainly useful for identification purpose. If you omit this suboption, QEMU uses the default 0.

❷ Specifies the Media Access Control (MAC) address for the network card. It is a unique identifier and you are advised to always specify it. If not, QEMU supplies its own default MAC address and creates a possible MAC address conflict within the related VLAN.

❸ Specifies the model of the network card. Use `-net nic,model=?` to get the list of all network card models supported by QEMU on your platform:

Currently, Novell supports the models `rtl8139` and `virtio`.

13.4.2 User-mode Networking

The `-net user` option instructs QEMU to use a user-mode networking. This is the default if no networking mode is selected. Therefore, these command lines are equivalent:

```
qemu-kvm -hda /images/sles11sp1_base.raw
```

```
qemu-kvm -hda /images/sles11sp1_base.raw -net nic -net user
```

This mode is useful if you want to allow VM Guest to access the external network resources, such as Internet. By default, no incoming traffic is permitted and therefore, VM Guest is not visible to other machines on the network. No administrator privileges are required in this networking mode. The user-mode is also useful to do a 'network-booting' on your VM Guest from a local directory on VM Host Server.

The VM Guest allocates an IP address from a virtual DHCP server. VM Host Server (the DHCP server) is reachable at 10.0.2.2, while the IP address range for allocation starts from 10.0.2.15. You can use `ssh` to connect to VM Host Server at 10.0.2.2, and `scp` to copy files back and forth.

Command Line Examples

This section shows several examples on how to set up user-mode networking with QEMU.

Example 13.1 *Restricted User-mode Networking*

```
qemu-kvm [...] -net user❶,vlan=1❷,name=user_net1❸,restrict=yes❹
```

❶ Specifies user-mode networking.

❷ Connect to VLAN number 1. If omitted, defaults to 0.

❸ Specifies a human readable name of the network stack. Useful when identifying it in the QEMU monitor.

❹ Isolates VM Guest. It will not be able to communicate with VM Host Server and no network packets will be routed to the external network.

Example 13.2 *User-mode Networking with Custom IP Range*

```
qemu-kvm [...] -net user,net=10.2.0.0/8❶,host=10.2.0.6❷,dhcpstart=10.2.0.20❸,\
hostname=tux_kvm_guest❹
```

❶ Specifies the IP address of the network that VM Guest sees and optionally the netmask. Default is 10.0.2.0/8.

❷ Specifies the VM Host Server IP address that VM Guest sees. Default is 10.0.2.2.

❸ Specifies the first of the 16 IP addresses that the built-in DHCP server can assign to VM Guest. Default is 10.0.2.15.

❹ Specifies the hostname that the built-in DHCP server will assign to VM Guest.

Example 13.3 *User-mode Networking with Network-boot and TFTP*

```
qemu=kvm [...] -net
user,tftp=/images/tftp_dir❶,bootfile=/images/boot/pxelinux.0❷
```

❶ Activates a built-in TFTP (a file transfer protocol with the functionality of a very basic FTP) server. The files in the specified directory will be visible to VM Guest as the root of a TFTP server.

❷ Broadcasts the specified file as a BOOTP (a network protocol which offers an IP address and a network location of a boot image, often used in diskless workstations) file. When used together with `tftp`, VM Guest can boot from network from the local directory on the host.

Example 13.4 *User-mode Networking with Host Port Forwarding*

```
qemu-kvm [...] -net user,hostfwd=tcp::2222-:22
```

Forwards incoming TCP connections to the port 2222 on the host to the port 22 (`SSH`) on VM Guest. If `sshd` is running on VM Guest, enter

```
ssh qemu_host -p 2222
```

where `qemu_host` is the hostname or IP address of the host system, to get a `SSH` prompt from VM Guest.

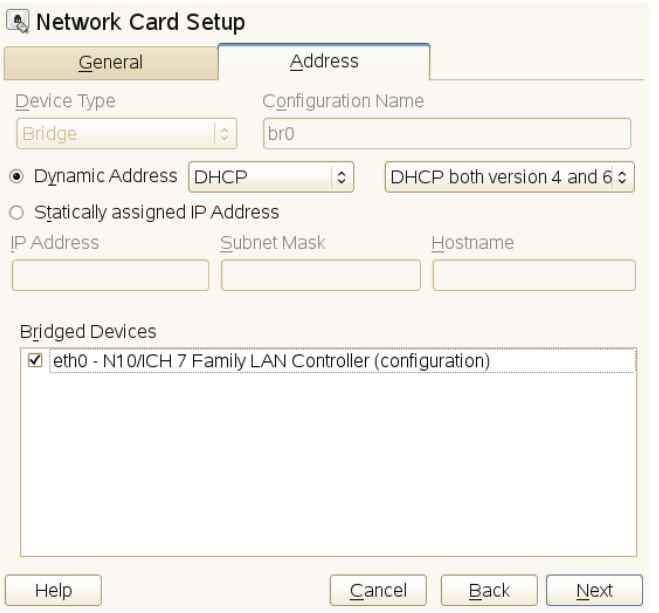
13.4.3 Bridged Networking

With the `-net tap` option, QEMU creates a network bridge by connecting the host TAP network device to a specified VLAN of VM Guest. Its network interface is then visible to the rest of the network. This method does not work by default and has to be explicitly specified.

First, create a network bridge and add a VM Host Server physical network interface (usually `eth0`) to it:

1. Start YaST Control Center and select *Network Devices* > *Network Settings*.
2. Click *Add* and select *Bridge* from the *Device Type* drop-down list in the *Hardware Dialog* window. Click *Next*.
3. Choose whether you need a dynamically or statically assigned IP address, and fill the related network settings if applicable.
4. In the *Bridged Devices* pane, select the Ethernet device to add to the bridge.

Figure 13.2 *Configuring Network Bridge with YaST*



Click *Next*. If asked about adapting already configured device, click *Continue*.

5 Click *OK* to apply the changes. Check if the bridge is created:

```
tux@venus:~> brctl show
bridge name bridge id          STP enabled  interfaces
br0         8000.001676d670e4  no           eth0
```

Use the following example script to connect VM Guest to the newly created bridge interface `br0`. Several commands in the script are run via the `sudo` mechanism because they require `root` privileges.

NOTE

Make sure the `tunctl` and `bridge-utils` packages are installed on VM Host Server. If not, install them with `zypper in tunctl bridge-utils`.

```
#!/bin/bash
bridge=br0❶
tap=$(/usr/bin/sudo /bin/tunctl -u $(/usr/bin/whoami) -b)❷
```

```
/usr/bin/sudo /sbin/ip link set $tap up❸
sleep 1s❹
/usr/bin/sudo /sbin/brctl addif $bridge $tap❺
qemu-kvm -m 512 -hda /images/sles11sp1_base.raw \
-net nic,vlan=0,model=virtio,macaddr=00:16:35:AF:94:4B \
-net tap,vlan=0,ifname=$tap❻,script=no❼,downscript=no
/usr/bin/sudo /sbin/brctl delif $bridge $tap❽
/usr/bin/sudo /sbin/ip link set $tap down❾
/usr/bin/sudo /bin/tunctl -d $tap❿
```

❶ Name of the bridge device.

❷ Prepare a new TAP device and assign it to the user who runs the script. TAP devices are virtual network devices often used for virtualization and emulation setups.

❸ Bring up the newly created TAP network interface.

❹ Make a 1 second pause to make sure the new TAP network interface is really up.

❺ Add the new `TAP` device to the network bridge `br0`.

❻ The `ifname=` suboption specifies the name of the TAP network interface used for bridging.

❼ Before `qemu-kvm` connects to a network bridge, it checks the `script` and `downscript` values. If it finds the specified scripts on the VM Host Server filesystem, it runs the `script` before it connects to the network bridge and `downscript` after it exits the network environment. You can use these scripts to first set up and bring up the bridged network devices, and then to deconfigure them. By default, `/etc/qemu-ifup` and `/etc/qemu-ifdown` are examined. If `script=no` and `downscript=no` are specified, the script execution is disabled and you have to take care manually.

❽ Deletes the TAP interface from a network bridge `br0`.

❾ Sets the state of the TAP device to 'down'.

❿ Deconfigures the TAP device.

13.5 Viewing VM Guest with VNC

QEMU normally uses an SDL (a cross-platform multimedia library) window to display the graphical output of VM Guest. With the `-vnc` option specified, you can make QEMU listen on a specified VNC display and redirect its graphical output to the VNC session.

TIP

When working with QEMU's virtual machine via VNC session, it is useful to work with the `-usbdevice tablet` option.

Moreover, if you need to use another keyboard layout than the default `en-us`, specify it with the `-k` option.

The first suboption of `-vnc` must be a *display* value. The `-vnc` option understands the following display specifications:

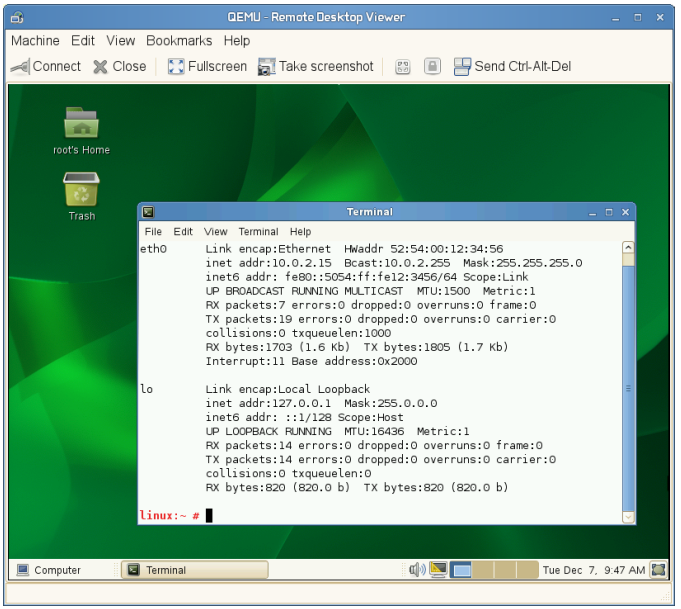
`host:display`
: Only connections from `host` on the display number `display` will be accepted. The TCP port on which the VNC session is then running is normally a 5900 + `display` number. If you do not specify `host`, connections will be accepted from any host.

`unix:path`
: The VNC server listens for connections on Unix domain sockets. The `path` option specifies the location of the related Unix socket.

`none`
: The VNC server functionality is initialized, but the server itself is not started. You can start the VNC server later with the QEMU monitor. For more information, see Chapter 14, *Administrating Virtual Machines with QEMU Monitor* (page 133).

```
tux@venus:~> qemu-kvm [...] -vnc :5
(on the client:)
wilber@jupiter:~> vinagre venus:5905 &
```

Figure 13.3 *QEMU VNC Session*



13.5.1 Secure VNC Connections

The default VNC server setup does not use any form of authentication. In the previous example, any user can connect and view the QEMU VNC Session from any host on the network.

There are several levels of security which you can apply to your VNC client/server connection. You can either protect your connection with a password, use x509 certificates, use SASL authentication, or even combine some of these authentication methods in one QEMU command.

See Section A.2, “Generating x509 Client/Server Certificates” (page 149) for more information about the x509 certificates generation. For more information about configuring x509 certificates on VM Host Server and the client, see Section 7.2.2, “Remote TLS/SSL Connection with x509 Certificate” (page 55) and Section “Configuring the Client and Testing the Setup” (page 57).

The Vinagre VNC viewer supports advanced authentication mechanisms. Therefore, it will be used to view the graphical output of VM Guest in the following examples. For this example, lets assume that the server x509 certificates `ca-cert.pem`, `server-cert.pem`, and `server-key.pem` are located in the `/etc/pki/qemu` directory on the host, while the client's certificates are distributed in the following locations on the client:

```
/etc/pki/CA/cacert.pem
/etc/pki/libvirt-vnc/clientcert.pem
/etc/pki/libvirt-vnc/private/clientkey.pem
```

Example 13.5 *Password Authentication*

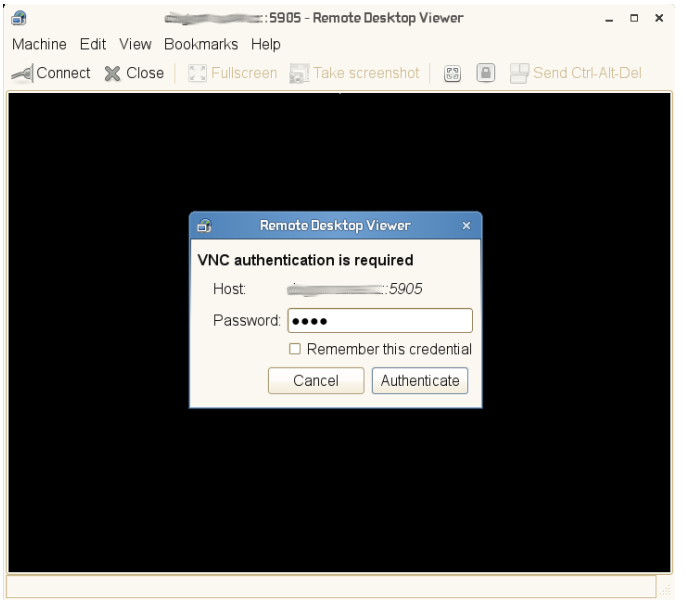
```
qemu-kvm [...] -vnc :5,password -monitor stdio
```

Starts the VM Guest graphical output on VNC display number 5 (usually port 5905). The `password` suboption initializes a simple password-based authentication method. There is no password set by default and you have to set one with the `change vnc password` command in QEMU monitor:

```
QEMU 0.12.5 monitor - type 'help' for more information
(qemu) change vnc password
Password: ****
```

You need the `-monitor stdio` option here, because you would not be able to manage the QEMU monitor without redirecting its input/output.

Figure 13.4 *Authentication Dialog in Vinagre*



Example 13.6 *x509 Certificate Authentication*

The QEMU VNC server can use TLS encryption for the session and x509 certificates for authentication. The server asks the client for a certificate and validates it against the CA certificate. Use this authentication type if your company provides an internal certificate authority.

```
qemu-kvm [...] -vnc :5,tls,x509verify=/etc/pki/qemu
```

Example 13.7 *x509 Certificate and Password Authentication*

You can combine the password authentication with TSL encryption and x509 certificate authentication to create a two-layer authentication model for clients. Remember to set the password in the QEMU monitor after you run the following command:

```
qemu-kvm [...] -vnc :5,password,tls,x509verify=/etc/pki/qemu -monitor stdio
```

Example 13.8 *SASL Authentication*

Simple Authentication and Security Layer (SASL) is a framework for authentication and data security in Internet protocols. It integrates several authentication mechanisms, like PAM, Kerberos, LDAP and more. SASL keeps its own user database, so the connecting user accounts do not need to exist on VM Host Server.

For security reasons, you are advised to combine SASL authentication with TLS encryption and x509 certificates:

```
qemu-kvm [...] -vnc :5,tls,x509,sasl -monitor stdio
```

Administrating Virtual Machines with QEMU Monitor

14

When QEMU is running, a monitor console is provided for performing interaction with the user. Using the commands available in the monitor console, it is possible to inspect the running operating system, change removable media, take screenshots or audio grabs and control several other aspects of the virtual machine.

14.1 Accessing Monitor Console

To access the monitor console from QEMU, press Ctrl + Alt + 2. To return back to QEMU from the monitor console, press Ctrl + Alt + 1.

To get help while using the console, use `help` or `?`. To get help for a specific command, use `help` *`command`*.

14.2 Getting Information about the Guest System

To get information about the guest system, use the `info option` command. If used without any option, the list of possible options is printed. Options determine which of the system will be analyzed:

`info version`
: Shows the version of QEMU

`info commands`
: Lists available QMP commands

`info network`
: Shows the network state

`info chardev`
: Shows the character devices

`info block`
: Information about block devices, such as hard drives, floppy drives, or CD-ROMs

`info blockstats`
: Read and write statistics on block devices

`info registers`
: Shows the CPU registers

`info cpus`
: Shows information about available CPUs

`info history`
: Shows the command line history

`info irq`
: Shows the interrupts statistics

`info pic`
: Shows the i8259 (PIC) state

`info pci`
: Shows the PCI information

`info tlb`
: Shows virtual to physical memory mappings

`info mem`
: Shows the active virtual memory mappings

`info hpet`
Shows state of HPET

`info jit`
Shows dynamic compiler information

`info kvm`
Shows the KVM information

`info numa`
Shows the NUMA information

`info usb`
Shows the guest USB devices

`info usbhost`
Shows the host USB devices

`info profile`
Shows the profiling information

`info capture`
Shows the capture (audio grab) information

`info snapshots`
Shows the currently saved virtual machine snapshots

`info status`
Shows the current virtual machine status

`info pcmcia`
Shows the guest PCMCIA status

`info mice`
Shows which guest mice is receiving events

`info vnc`
Shows the VNC server status

`info name`
: Shows the current virtual machine name

`info uuid`
: Shows the current virtual machine UUID

`info usernet`
: Shows the user network stack connection states

`info migrate`
: Shows the migration status

`info balloon`
: Shows the balloon device information

`info qtree`
: Shows the device tree

`info qdm`
: Shows the qdev device model list

`info roms`
: Shows the ROMs

14.3 Changing VNC Password

To change the VNC password, use the `change vnc password` command and enter the new password:

```
(qemu) change vnc password
Password: ********
(qemu)
```

14.4 Managing Devices

To release the device or file connected to the removable media device, use the `eject` *`device`* command. Use the optional `-f` to force ejection.

To change removable media (like CD-ROMs), use the `change` *`device`* command. The name of the removable media can be determined using the `info block` command:

```
(qemu) info block
ide1-cd0: type=cdrom removable=1 locked=0 file=/dev/sr0 ro=1 drv=host_device
(qemu) change ide1-cd0 /path/to/image
```

14.5 Controlling Keyboard and Mouse

It is possible to use the monitor console to emulate keyboard and mouse input if necessary. For example, if your graphical user interface intercepts some key combinations at low level (such as Ctrl + Alt + F1 in X Window), you can still enter them using the `sendkey` *`keys`*:

```
sendkey ctrl-alt-f1
```

To list the key names used in the *`keys`* option, enter `sendkey` and press Tab.

To control the mouse, the following commands can be used:

`mouse_move` *`dx dy [dz]`*
: Move the active mouse pointer to the specified coordinates dx, dy with the optional scroll axis dz.

`mouse_button` *`val`*
: Change the state of the mouse buttons (1=left, 2=middle, 4=right).

`mouse_set` *`index`*
: Set which mouse device receives events. Device index numbers can be obtained with the `info mice` command.

14.6 Changing Available Memory

If the virtual machine was started with the `-balloon virtio` option and the paravirtualized balloon device that allows to dynamically change the amount of memory available is therefore enabled, it is possible to change the available memory dynamically. For more information about enabling the baloon device, see Section 12.1, "Basic Installation with `qemu-kvm`" (page 94).

To get information about the balloon device in the monitor console and to determine whether the device is enabled, use the `info balloon` command:

```
(qemu) info balloon
```

If the balloon device is enabled, use the `balloon` *`memory_in_MB`* command to set the requested amount of memory:

```
(qemu) balloon 400
```

14.7 Dumping Virtual Machine Memory

To save the content of the virtual machine memory to a disk or console output, use the following commands:

`memsave` *`addrsizefilename`*
: Saves virtual memory dump starting at *`addr`* of size *`size`* to file *`filename`*

`pmemsave` *`addrsizefilename`*
: Saves physical memory dump starting at *`addr`* of size *`size`* to file *`filename`*-

x /*`fmtaddr`*
: Makes a virtual memory dump starting at address *`addr`* and formatted according to the *`fmt`* string. The *`fmt`* string consists of three parameters *`countformatsize`*:

  The *`count`* parameter is the number of the items to be dumped.

The *format* can be x (hex), d (signed decimal), u (unsigned decimal), o (octal), c (char) or i (assembly instruction).

The *size* parameter can be b (8 bits), h (16 bits), w (32 bits) or g (64 bits). On x86, h or w can be specified with the i format to respectively select 16 or 32-bit code instruction size. is the number of the items to be dumped.

xp /*fmtaddr*

Makes a physical memory dump starting at address *addr* and formatted according to the *fmt* string. The *fmt* string consists of three parameters *countformatsize*:

The *count* parameter is the number of the items to be dumped.

The *format* can be x (hex), d (signed decimal), u (unsigned decimal), o (octal), c (char) or i (asm instruction).

The *size* parameter can be b (8 bits), h (16 bits), w (32 bits) or g (64 bits). On x86, h or w can be specified with thei format to respectively select 16 or 32-bit code instruction size. is the number of the items to be dumped.

14.8 Managing Virtual Machine Snapshots

Virtual machine snapshots are snapshots of the complete virtual machine including the state of CPU, RAM, and the content of all writable disks. To use virtual machine snapshots, you must have at least one non-removable and writable block device using the qcow2 disk image format.

NOTE

Currently, only the raw disk image format is supported by Novell.

Snapshots are helpful when you need to save your virtual machine in a particular state. For example, after you configured network services on a virtualized server and want to quickly start the virtual machine in the same state that has been saved last. You can

also create a snapshot after the virtual machine has been powered off to create a backup state before you try something experimental and possibly make VM Guest unstable. This section introduces the former case, while the latter is described in Section 12.2.3, “Managing Snapshots of Virtual Machines with qemu-img Snapshot” (page 101).

The following commands are available for managing snapshosts in QEMU monitor:

`savevm` *`name`*
: Creates a new virtual machine snapshot under the tag *`name`* or replaces an existing snapshot.

`loadvm` *`name`*
: Loads a virtual machine snapshot tagged *`name`*.

`delvm`
: Deletes a virtual machine snapshot.

`info snapshots`
: Prints information about available snapshots.

```
(qemu) info snapshots
Snapshot list:
ID❶       TAG❷                VM SIZE❸        DATE❹           VM CLOCK❺
1         booting                4.4M 2010-11-22 10:51:10   00:00:20.476
2         booted                 184M 2010-11-22 10:53:03   00:02:05.394
3         logged_in              273M 2010-11-22 11:00:25   00:04:34.843
4         ff_and_term_running    372M 2010-11-22 11:12:27   00:08:44.965
```

❶ Unique identification number of the snapshot. Usually auto-incremented.

❷ Unique description string of the snapshot. It is meant as a human readable version of the ID.

❸ The disk space occupied by the snapshot. Note that the more memory is consumed by running applications, the bigger the snapshot is.

❹ Time and date the snapshot was created.

❺ The current state of the virtual machine's clock.

14.9 Suspending and Resuming Virtual Machine Execution

The following commands are available for suspending and resuming virtual machines:

`stop`
: Suspends the execution of the virtual machine.

`cont`
: Resumes the execution of the virtual machine.

`system_reset`
: Resets the virtual machine. The effect is similar to the reset button on a physical machine. This may leave the filesystem in an unclean state.

`system_powedown`
: Sends an ACPI shutdown request to the machine. The efect is similar to the power button on a physical machine.

`q` or `quit`
: Terminates QEMU immediately.

14.10 Live Migration

The live migration process allows to transmite any virtual machine from one host system to another host system without any interruption in availability. It is possible to change hosts permanently or just during a maintenance. It is recommended that the source and destination systems have the same architecture, however it is possible to migrate between hosts with AMD and Intel architecture or even between Linux and Windows hosts.

The requirements for the live migration:

- The virtual machine image must be accessible on both source and destination hosts. For example, it can be located on a shared NFS disk.
- The image directory should be located in the same path on both hosts.

- Both hosts must be located in the same subnet.
- The guest on the source and destination hosts must be started in the same way.

The live migration process has the following steps:

1. The virtual machine instance is running on the source host.
2. The virtual machine is started on the destination host in the frozen listening mode. The parameters used are the same as on the source host plus the `-incoming tcp:`*`ip`*`:`*`port`* parameter, where *`ip`* specifies the IP address and *`port`* specifies the port for listening to the incoming migration. If 0 is set as IP address, the virtual machine listens on all interfaces.
3. On the source host, switch to the monitor console and use the `migrate -d tcp:`*`destination_ip`*`:`*`port`* command to initiate the migration.
4. To determine the state of the migration, use the `info migrate` command in the monitor console on the source host.
5. To cancel the migration, use the `migrate_cancel` command in the monitor console on the source host.
6. To set the maximum tolerable downtime for migration in seconds, use the `migrate_set_downtime` *`number_of_seconds`* command.
7. To set the maximum speed for migration in bytes per second, use the `migrate_set_speed` *`bytes_per_second`* command.

Appendix A

A.1 Installing Para-Virtualized Drivers

A.1.1 Installing Para-Virtualized Drivers for SUSE Linux Enterprise Server 10 SP3

Support for para-virtualized drivers is already built into all SUSE Linux Enterprise Server 11 SP1 Kernels, so virtio devices are supported out of the box. Para-virtualized drivers for SUSE Linux Enterprise Server 10 SP3 are not shipped with the product and need to be installed from a repository provided by Novell. It is recommended to install para-virtualized drivers during the installation as described in Section 5.3.1, “Adding para-virtualized Drivers During the Installation” (page 35). If you need to install the drivers on an existing virtual machine, follow the instructions below.

1. Add the para-virtualized drivers repository and the corresponding drivers update repositories with either the YaST *Software Repositories* module or with `zypper ar`.

2. Determine the flavor of the installed Kernel by running `uname -r`. The output string has the form *`Version-Flavor`* (for example `2.6.32.24-0.2-default`).

3. Search for packages matching the string `novell-virtio-drivers` in the YaST *Software Management* module or with `zypper se`.

4 Install the `novell-virtio-drivers` package matching your Kernel flavor.

A.1.2 Installing virtio Drivers for Microsoft Windows*

Providing para-virtualized drivers during a Microsoft Windows installation does currently not work, since the installation refuses to boot from a para-virtualized hard disk. Therefore, the para-virtualized drivers need to be installed on a running Windows installation.

The following instructions assume that the existing Windows installation uses a single IDE hard disk and a single network adapter. An ISO image containing the virtio drivers for Windows is part of the `kvm` package and is available on the KVM host under `/usr/share/qemu-kvm/win-virtio-drivers.iso`. Make this ISO image available as a CD-ROM on your virtual machine as described in Section 9.3, "Ejecting and Changing Floppy or CD/DVD-ROM Media with Virtual Machine Manager" (page 79). In case your virtual machine is configured without a CD-ROM device or you prefer to add a second one, see Section 9.1, "Adding a CD/DVD-ROM Device with Virtual Machine Manager" (page 77) for setup instructions.

Finding the virtio drivers for Windows

Windows XP 32-bit

Memory Ballooning: `balloon\install\XP\x86\balloon.inf`
Network: `NetKVM\install\XP_Win2003\x86\netkvm.inf`
Storage: `viostor\install\XP\x86\viostor.inf`

Windows XP 64-bit

Memory Ballooning: not available
Network: `NetKVM\install\XP_Win2003\amd64\netkvm.inf`
Storage: `viostor\install\XP\amd64\viostor.inf`

Windows Server 2003 32-bit

Memory Ballooning: `balloon\install\Win2003\x86\balloon.inf`
Network: `NetKVM\install\XP_Win2003\x86\netkvm.inf`

Storage: `viostor\install\Win2003\x86\viostor.inf`

Windows Server 2003 64-bit

Memory Ballooning: `balloon\install\Win2003\amd64\balloon.inf`
Network: `NetKVM\install\XP_Win2003\amd64\netkvm.inf`
Storage: `viostor\install\XP\amd64\viostor.inf`

Windows Vista/Server 2008 32-bit

Memory Ballooning: `balloon\install\Vista_Win2008\x86\balloon.inf`
Network: `NetKVM\install\Vista_Win2008\x86\netkvm.inf`
Storage: `viostor\install\Vista_Win2008\x86\viostor.inf`

Windows Vista/Server 2008 64-bit

Memory Ballooning: `balloon\install\Vista_Win2008\amd64\balloon.inf`
Network: `NetKVM\install\Vista_Win2008\amd64\netkvm.inf`
Storage: `viostor\install\Vista_Win2008\amd64\viostor.inf`

Windows 7 32-bit

Memory Ballooning: `balloon\install\Win7\x86\balloon.inf`
Network: `NetKVM\install\Win7\x86\netkvm.inf`
Storage: `viostor\install\Win7\x86\viostor.inf`

Windows 7 64-bit

Memory Ballooning: `balloon\install\Win7\amd64\balloon.inf`
Network: `NetKVM\install\Win7\amd64\netkvm.inf`
Storage: `viostor\install\Win7\amd64\viostor.inf`

Windows 7

The following instructions show how to install para-virtualized storage an network drivers for Windows 7. Please make sure to *exactly* follow the instructions for installing

the storage drivers, otherwise your system will either completely refuse to boot or will boot into a “blue screen”!

IMPORTANT: Technical Support

The following instructions require to use `virsh edit`. Using this command in principle is not supported by the Novell Technical Support. However, this special context (Installing Para-Virtualized Storage Drivers for Windows) is an exception from this rule. It will be supported with reasonable effort.

Procedure A.1 *Installing Para-Virtualized Storage Drivers for Windows 7 32-bit*

1. Shut down the Windows 7 VM Guest and use Virtual Machine Manager to add an additional hard disk of type `virtio` (a para-virtualized hard disk). This disk is only temporarily needed and will be removed again from the VM Guest.

2. If necessary, use Virtual Machine Manager to adjust the *Boot Device Order*. It *must* start with *Hard Disk*, otherwise the system will not boot once the system disk is para-virtualized. You need to confirm your changes with *Apply*, otherwise they will not be written to the configuration.

3. Reboot the VM Guest. Once it has booted, open the *Device Manager*, for example, by opening the main menu and entering `devmgmt.msc` followed by Enter into the *Start programs and files* field.

4. Search for the entry *Other devices* > *SCSI Controller*. The entry is marked with an exclamation mark as being problematic. Right-click this entry and choose *Update Driver Software*.

5. Install the driver. Choose to *Browse my computer for driver software*. Use the *Browse* button to select the directory on the driver CD containing the storage drivers for your operating system and architecture (`viostor\install\Win7\x86\`). Confirm the security exception by clicking *Install*.

6. Once the driver installation is finished, a new *Storage Controller* named *Novell VirtIO SCSI Adapater* is listed in the *Device Manager*. Additionally, the entry *Disk Drives* now contains the temporary para-virtualized disk. It is listed as *Novell VirtIO SCSI Disk Device*.

7 Shut down the Windows 7 VM Guest and use Virtual Machine Manager to remove the temporary para-virtualized disk added earlier.

8 Changing the type of a virtual hard disk is currently not supported by Virtual Machine Manager—therefore the XML configuration needs to be changed directly. Open a terminal and enter the following command (replace *NAME* with the name of you Windows 7 VM Guest). If operating from a remote host, also specify a connection URl with the `-c` parameter.

```
virsh edit NAME
```

An editor (`vi` by default) opens. Search for a block similar to the following:

```
<disk type='block' device='disk'>
  <driver name='qemu' type='raw'/>
  <source file='/var/lib/libvirt/images/win7.raw'/>
  <target dev='hda' bus='ide'/>
  <address type='drive' controller='0' bus='0' unit='0'/>
</disk>
```

Remove the <address> tag. Change the attributes of the <target> tag to `dev='vda'` and `bus='virtio'`:

```
<disk type='block' device='disk'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/Virtual/win7'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Save the file. A successful save results in `Domain NAME XML configuration edited.` In case an error is reported (for example, when having produced invalid XML), the configuration has not been changed.

9 Restart the VM Guest. If starting it via Virtual Machine Manager, make sure the hardware change is visible in the *Details* screen before you start (this may take a few seconds after you have saved the configuration changes from `virsh`). Otherwise your changes will be overwritten with the configuration last used by Virtual Machine Manager.

Your Windows 7 VM Guest now uses a para-virtualized system disk.

Installing para-virtualized network drivers is very similar to installing the storage drivers:

Procedure A.2 *Installing Para-Virtualized Network Drivers for Windows 7*

1. Shut down the Windows 7 VM Guest and use Virtual Machine Manager to add an additional network adapter of type `virtio` (a para-virtualized network adapter). This ensures that you still have network connectivity while installing the drivers.

2. Reboot the VM Guest and install the driver via the *Device Manager* as described above. The new network adapter can be found under *Other devices* > *Ethernet controller*. After a successful driver installation, a *Novell VirtIO Ethernet Adapter* is listed in the *Device Manager* under *Network Adapters*.

3. Shut down the VM Guest and remove the original, non-para-virtualized network adapter from the guest configuration using Virtual Machine Manager. Reboot the guest—now it uses a para-virtualized network adapter.

Other Windows Versions (XP, Server 2003/2008, Vista)

Installing para-virtualized drivers for other Windows versions is very similar to installing them on Windows 7 (see Section "Windows 7" (page 145)). You do not need to manually start the *Device Manager*— Windows will rather prompt you to install the missing drivers. Make sure to manually choose the location of the driver during the installation process.

WARNING: Para-Virtualized Storage Drivers on Windows Vista

Currently the para-virtualized storage drivers for Windows Vista do not support booting from a para-virtualized disk. Using para-virtualized storage devices for non-bootable disks is supported.

NOTE: Para-Virtualized Storage Drivers on Windows XP not Recommended

Using the para-virtualized storage drivers on Windows XP does not result in any performance gain—it may even result in performance penalties. Therefore, it is *not* recommended to use them. See `http://www.mail-archive.com/kvm@vger.kernel.org/msg22834.html` for technical details.

Note that this only affects para-virtualized *storage* drivers for Windows XP! Using para-virtualized storage drivers on other Windows versions will result in

better performance. Using para-virtualized network drivers on Windows XP is also beneficial.

A.2 Generating x509 Client/Server Certificates

In order to be able to create x509 client and server certificates you need to issue them by a Certificate Authority (CA). It is recommended to set up an independent CA that only issues certificates for `libvirt`.

1 Set up a CA as described in Section "Creating a Root CA" (Chapter 17, *Managing X.509 Certification*, ↑*Security Guide*).

2 Create a server and a client certificate as described in Section "Creating or Revoking User Certificates" (Chapter 17, *Managing X.509 Certification*, ↑*Security Guide*). The Common Name (CN) for the server certificate must be the full qualified hostname, the Common Name for the client certificate can be freely chosen. For all other fields stick with the defaults suggested by YaST.

Export the client and server certificates to a temporary location (for example, `/tmp/x509/`) by performing the following steps:

2a Select the certificate on the *certificates* tab.

2b Choose *Export* > *Export to File* > *Certificate and the Key Unencrypted in PEM Format*, provide the *Certificate Password* and the full path and the filename under *File Name*, for example, `/tmp/x509/server.pem` or `/tmp/x509/client.pem`.

2c Open a terminal and change to the directory where you have saved the certificate and issue the following commands to split it into certificate and key (this example splits the server key):

```
csplit -z -f s_ server.pem '/-----BEGIN/' '{1}'
       mv s_00 servercert.pem
       mv s_01 serverkey.pem
```

2d Repeat the procedure for each client and server certificate you would like to export.

3 Finally export the CA certificate by performing the following steps:

3a Switch to the *Description* tab.

3b Choose*Advanced* > *Export to File* > *Only the Certificate in PEM Format* and enter the full path and the filename under *File Name*, for example, `/tmp/x509/cacert.pem`.

A.3 QEMU Command Line Options

A.3.1 Supported qemu-kvm Command Line Options

```
-alt-grab
-append ...
-audio-help
-balloon ...
-boot ...
-cdrom ...
-chardev ..
-clock
-cpu [?|kvm64 ]
-ctrl-grab
-d ...
-daemonize
-device driver [isa-serial|isa-parallel|isa-fdc
|ide-drive|VGA|cirrus-vga|rtl8139|virtio-net-pci
|virtio-blk-pci|virtio-balloon-pci] ...
-drive if=[ide|floppy|virtio] format=raw snapshot=off ...
-echr ...
```

```
-enable-kvm
-fda/-fdb ...
-full-screen
-gdb ...
-global ...
-h
-hda/-hdb/-hdc/-hdd ...
-help
-incoming ...
-initrd ...
-k ...
-kernel ...
-loadvm ...
-m ...
-mem-path ...
-mem-prealloc
-mon ...
-monitor ...
-M [pc|pc-0.12]
-name ...
-netdev ...
-net [nic|user|tap|none] mode=[rtl8139|virtio]
-no-acpi
-nodefaults
-no-frame
-nographic
-no-hpet
-no-quit
-no-reboot
-no-shutdown
-parallel ...
-pcidevice ...
-pidfile ...
-readconfig ...
-rtc ...
-runas ...
```

```
-s
-S
-sdl
-serial ...
-smbios ...
-smp ...
-tdf
-usb
-usbdevice [tablet|mouse]
-uuid ..
-version
-vga [std|cirrus|none]
-vnc ...
-watchdog ...
-watchdog-action ...
-writeconfig ...
```

A.3.2 Unsupported qemu-kvm Command Line Options

The following `qemu-kvm` command line options are *not* supported by Novell:

```
-acpitable ...
-bios ...
-bt ...
-chroot ...
-cpu [phenom|core2duo|qemu32|qemu64|coreduo|486|pentium
|pentium2|pentium3|athlon|n270]
-curses
-device driver (where driver is not in [isa-serial
|isa-parallel|isa-fdc|ide-drive|VGA|cirrus-vga|rtl8139
|virtio-net-pci|virtio-blk-pci|virtio-balloon-pci])
-drive if=[scsi|mtd|pflash], snapshot=yes, format=[anything
apart from raw]
-enable-nesting
```

```
-icount ...
-kvm-shadow-memory ...
-L ...
-M [pc-0.11|pc-0.10|isapc|mac]
-mtdblock ...
-net dump ...
-net socket ...
-no-fd-bootchk
-no-kvm
-no-kvm-irqchip
-no-kvm-pit
-no-kvm-pit-reinjection
-numa ...
-nvram ...
-option-rom ...
-osk
-pflash ...
-portrait
-qmp ...
-sd ...
-set ...
-show-cursor
-singlestep
-snapshot
-soundhw ...
-tb-size ...
-usbdevice [disk|host|serial|braille|net]
-vga [vmware|xenfb]
-virtioconsole ...
-win2k-hack
```

A.3.3 Supported qemu-kvm `monitor` Command Line Options

The following `qemu-kvmmonitor` command line options are supported by Novell:

```
?
balloon target ...
[c|cont]
change device ...
cpu ...
eject ...
gdbserver ...
help
info ...
logfile ...
logitem ...
mce ...
memsave ...
migrate ...
migrate_set_downtime ...
migrate_set_speed ...
mouse_button ...
mouse_move ...
mouse_set ...
pmemsave ...
[p|print] ...
q
sendkey ...
stop
system_powerdown
watchdog_action ...
x ...
xp ...
```

A.3.4 Unsupported qemu-kvm `monitor` Command Line Options

The following `qemu-kvmmonitor` command line options are *not* supported by Novell:

```
acl_add ...
acl_policy ...
acl_remove ...
acl_reset ...
acl_show ...
block_passwd ...
boot_set
close_fd ...
commit ...
cpu_set ...
delvm ...
device_add ...
device_del ...
drive_add ...
hostfwd_add ...
hostfwd_remove ...
host_net_add ...
host_net_remove ...
i ...
loadvm ...
migrate_cancel
nmi ...
o ...
pci_add ...
pci_del...
savevm ...
screendump ...
set_link ...
singlestep ...
stopcapture ...
sum ...
system_reset
usb_add ...
watchdog_action ...
wavcapture ...
```