KVM Installation and Requirements

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, andkvm-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.

Hardware Requirements

Currently, SUSE 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 Web site identifies processors which support hardware virtualization:<http://ark.intel.com/Products/VirtualizationTechnology> (for Intel CPUs), and<http://products.amd.com/> (for AMD CPUs).

**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 2.2, System Requirements for Operating Linux, (↑*Deployment Guide*)](https://www.suse.com/documentation/sles11/book_sle_deployment/data/sec_x86_sysreqs.html#sec_x86_sysreqs). 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.

# Supported Guest Operating Systems

The following table lists guest operating systems tested and their support status offered by SUSE. All guest operating systems are supported both fully virtualized and paravirtualized, with the exception of Windows guests, which are only supported fully virtualized and OES and Netware guests, which are supported only paravirtualized. All operating systems except Netware (32-bit only) 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 paravirtualized block devices.
* virtio-balloon: Memory driver for dynamic memory allocation. Allows to dynamically change the amount of memory allocated to a guest.
* virtio-scsi: Storage interface that supports advanced SCSI hardware.
* kvm-clock: Clock synchronization driver.

**Table 1-1**Supported Guest Operating Systems on KVM with SUSE Linux Enterprise Server

|  |  |  |
| --- | --- | --- |
| ****SLES 12**** | | |
|  | *PV drivers:* | kvm-clock, virtio-net, virtio-blk, virtio-balloon, virtio-console, virtio-rng, virtio-scsi |
|  | *Support Status:* | Fully supported (L3) |
| ****SLES 11 SP3 / SP4**** | | |
|  | *PV drivers:* | kvm-clock, virtio-net, virtio-blk, virtio-balloon, virtio-console, virtio-rng, virtio-scsi (SP3 and SP4 only) |
|  | *Support Status:* | Fully supported (L3) |
| ****SLES 10 SP4**** | | |
|  | *PV drivers:* | kvm-clock, virtio-net, virtio-blk, virtio-balloon, virtio-console |
|  | *Support Status:* | Fully supported (L3) |
| ****SLES 9 SP4**** | | |
|  | *PV drivers:* | n/a |
|  | *Support Status:* | Fully supported (L3) |
|  | *Mandatory boot parameters:* | * 32 bit kernel: clock=pmtmr * 64 bit kernel: ignore\_lost\_ticks |
| ****SLED 11 SP4**** | | |
|  | *PV drivers:* | kvm-clock, virtio-net, virtio-blk, virtio-balloon, virtio-console, virtio-rng |
|  | *Support Status:* | Technology Preview (L2) |
| ****RedHat Enterprise Linux 5.11+ / RHEL 6.6+ / RHEL 7.0+**** | | |
|  | *PV drivers:* | See <http://www.redhat.com/> |
|  | *Support Status:* | Best Effort (L2) |
|  | *Note:* | Refer to the RHEL Virtualization guide for more information. |
| ****Windows 2003 SP2+ / 2008 SP2+ / 2008 R2 SP1+ / 2012+ / 2012 R2+**** | | |
|  | *PV drivers:* | virtio-net, virtio-blk, virtio-balloon; drivers from the Virtual Machine Driver Pack (<http://www.suse.com/products/vmdriverpack/>) are preferred |
|  | *Support Status:* | Fully supported (L3) |
|  | *Note:* | Host processor must have constant\_tsc CPU feature (check with the following command: grep "constant\_tsc" /proc/cpuinfo). |
| ****Windows XP SP3+ / Vista SP2+ / 7 SP1+ / 8+ / 8.1+**** | | |
|  | *PV drivers:* | virtio-net, virtio-blk, virtio-balloon; drivers from the Virtual Machine Driver Pack (<http://www.suse.com/products/vmdriverpack/>) are preferred |
|  | *Support Status:* | Best effort (L2) |
| ****OES 11 SPx**** | | |
|  | *PV drivers:* |  |
|  | *Support Status:* | Fully supported (L3) |
| ****Netware 6.5 SP8 (32b only)**** | | |
|  | *PV drivers:* |  |
|  | *Support Status:* | Fully supported (L3) |

**IMPORTANT:** Guest images created under SUSE Linux Enterprise Server 11 SP1 and newer are supported, but guest images created under a previous SUSE Linux Enterprise version are not supported.

## 1.2.1Availability of Paravirtualized Drivers[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_kvm_requires_guests.html#sec_kvm_requires_guests_virt_drivers)

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

SUSE Linux Enterprise Server 11 SP1 / SP2 / SP3

included in Kernel

SUSE Linux Enterprise Server 10 SP4

included in Kernel

SUSE Linux Enterprise Server 9 SP4

not available

RedHat

available in RedHat Enterprise Linux 5.4 and newer

Windows

SUSE has developed virtio based drivers for Windows, which are available in the Virtual Machine Driver Pack (VMDP). See <http://www.suse.com/products/vmdriverpack/> for more information.

# The kvm package

The kvm package provides qemu-kvm, the program that performs the I/O emulation for the VM Guest. In addition to the qemu-kvm program, the kvm package also comes with a debug level monitoring utility (kvm\_stat), firmware components, key-mapping files, and scripts. The deprecated Windows drivers (win-virtio-drivers.iso) are no longer provided. For more information, see [Deprecated features](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_qemu-commands.html#kvm_deprecated_features).

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 SUSE'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. libvirtprovides an API, a daemon, and a shell (virsh).
* virt-manager (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.

# 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 manually and start libvirtd to avoid a reboot:
6. modprobe kvm-intel # on Intel machines only
7. modprobe kvm-amd # on AMD machines only
8. modprobe vhost-net

rclibvirtd start

**NOTE: The vhost-net Kernel Module**

The vhost-net kernel module allows for a more efficient network transport to the guest. It is automatically used by libvirt if loaded, or when using the qemu-kvm command line, by adding,vhost=on to the networking option.

# 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.

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 10.8, Clock Settings](https://www.suse.com/documentation/sles11/book_kvm/data/sec_kvm_managing_clock.html) 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 the same CPU features. The only supported CPU model for migration is -cpu qemu64 (default) with no additional features specified. No physical devices can be passed from host to guest. Guest storage has to be accessible from both VM Host Servers and guest definitions need to be compatible. VM Host Server and VM Guests need to have proper timekeeping installed. The use of the AHCI interface, the virtfs feature, and the -mem-path command-line option are not compatible with migration. Migration from SP3 to SP2 or SP1 hosted guests is not supported.

User Permissions

The management tools (Virtual Machine Manager, virsh, vm-install) need to authenticate withlibvirt—see [Section 8.0, Connecting and Authorizing](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_connect.html) 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.

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 10.8, Clock Settings](https://www.suse.com/documentation/sles11/book_kvm/data/sec_kvm_managing_clock.html) 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 the same CPU features. The only supported CPU model for migration is -cpu qemu64 (default) with no additional features specified. No physical devices can be passed from host to guest. Guest storage has to be accessible from both VM Host Servers and guest definitions need to be compatible. VM Host Server and VM Guests need to have proper timekeeping installed. The use of the AHCI interface, the virtfs feature, and the -mem-path command-line option are not compatible with migration. Migration from SP3 to SP2 or SP1 hosted guests is not supported.

User Permissions

The management tools (Virtual Machine Manager, virsh, vm-install) need to authenticate withlibvirt—see [Section 8.0, Connecting and Authorizing](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_connect.html) 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.

Hardware Limitations

The following virtual hardware limits for guests have been tested. We ensure host and VMs install and work successfully, even when reaching the limits and there are no major performance regressions (CPU, memory, disk, network) since the last release.

|  |  |
| --- | --- |
| *Max. Guest RAM Size* | 4 TB |
| *Max. Virtual CPUs per Guest* | 256 |
| *Max. Virtual Network Devices per Guest* | 8 |
| *Max. Virtual Block Devices per Guest* | 4 emulated (IDE), 20 para-virtual (using virtio-blk) |
| *Max. Number of VM Guests per VM Host Server* | Limit is defined as the total number of virtual CPUs in all guests being no greater than 8 times the number of CPU cores in the host. |

The following hardware limits for the host server have been tested.

|  |  |
| --- | --- |
| *Max. Physical CPUs* | 4096 |
| *Max. Physical Memory* | 16 TB |

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 Pass-through** |
| *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)* | 60% (e1000 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 |

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 Pass-through** |
| *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)* | 60% (e1000 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 1 |  |  |

# KVM Support Status

The following list contains features and tools as supported by SUSE—this does not necessarily reflect the support status of the software itself. For a list of qemu-kvm command switches supported by SUSE, refer to [Section A.3, QEMU Command Line Options](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_qemu-commands.html).

Supported Features and Tools

vm-install

Define and install VM Guests via vm-install including specifying the number of virtual processors, RAM, disk type and location, video type, keyboard mapping, NIC type, binding, MAC address, and boot method.

*Restrictions:* Currently only the raw, qcow2 and qed disk formats are supported in read and write mode. The vmdk, vpc and vhd/vhdx formats are only supported in read-only mode. NIC creation is restricted to using Realtek, e1000 or virtio NICs. Sound cards are not supported.

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 and USB assignments are supported.

*Restrictions:* The following features are currently not supported: sound devices, qxl, vmvga (vmware), Xen video, pcnet, ne2k\_pci, eepro100, emulated SCSI disks, Spice graphics. Raw,qed, and qcow2 are the only supported storage formats in read and write mode. The vmdk, vpcand vhd/vhdx formats are only supported in read-only mode.

virsh

Manage guests via the command line.

Most virsh subcommands are supported, including creation, modification, and destruction of guests and all life cycle operations. Any virsh subcommands which translate to unsupported qemu-kvm command-line or monitor syntax are also unsupported. Guest XML descriptions used by virsh can be created manually, using vm-install, the Virtual Machine Manager, or external tools and scripts.

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](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_qemu-commands.html#app_kvm_qemu-commands_supported) for a list of supported options.

*Restrictions:* See [Section A.3.2, Unsupported qemu-kvm Command Line Options](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_qemu-commands.html#app_kvm_qemu-commands_unsupported) for a list of not supported options.

kvm\_stat

Debugging and monitoring tool.

USB Host Device Pass-through

A physical USB device may be passed from the VM Host Server to the VM Guest. Given the very wide variety of USB devices available, it is anticipated that some devices may not work properly.

PCI Pass-through

PCI Pass-through improves performance of PCI devices. It requires underlying support from the hardware (e.g. Intel VT-d extensions, or AMD IOMMU extensions). VT-d requires the Kernel parameter "intel\_iommu=on". Additionally, some host hardware will require the use of the KVM Kernel module parameter: allow\_unsafe\_assigned\_interrupts=1 (enabling this parameter has security implications).

Many PCIe cards from major vendors should be supportable. Refer to system level certifications for specific details, or contact the vendor for support statements.

Device hotplugging

Dynamically adding or removing emulated or pass-through physical devices in the VM Guest is supported.

Memory ballooning

Dynamically changing the amount of memory allocated to a guest is supported.

Sharing Folders between VM Host Server and VM Guest

Sharing folders between host and VM Guest is supported via VirtFs.

Kernel Samepage Merging (KSM)

KSM allows for automatic sharing of identical memory pages between guests to save host memory. KVM is optimized to use KSM if enabled on the VM Host Server.

Transparent Huge Pages (THP)

THP allows CPUs to address memory using pages larger than the default 4 KB. This helps reducing memory consumption and CPU cache usage. KVM is optimized to use THP (via madvise and opportunistic methods) if enabled on the VM Host Server.

KVM Security

A kvm group is created by the KVM package, which permits a non-root user to access the KVM control device file (/dev/kvm). Where possible, guests should not be run as root. Steps have been taken to enable this for libvirt as well. A setuid bridge helper has been added so that a bridged network interface can be set up without needing root privileges.

Seccomp2 based sandboxing

The VM Guest can be run in a sandboxed environment where only predetermined system calls are permitted for added protection against malicious behavior.

APIC Virtualization

Hardware APIC Virtualization, allowing the processor to directly inject interrupts into the VM Guest to achieve better performance, is supported.

VirtFS (file system pass-through)

Directories in the host file system can be shared between the host and VM Guest or guests usingvirtfs. A virtfs proxy helper is provided to enable virtfs usage when KVM is used as non-root user.

Vhost-net kernel module support

The vhost-net kernel module allows for a more efficient network transport to the VM Guest. It is automatically used by libvirt if loaded, or when using the qemu-kvm command line, by addingvhost=on to the networking option.

AHCI guest storage interface

The AHCI interface for SATA storage has been recently added. It permits much higher block I/O performance than the IDE interface, and is particularly useful for use in recent Windows OS versions.

qcow2 and qed storage formats

qcow2 and qed storage formats can now be used with live migration.

Trim and Online Disk Resizing

Trim and online disk resizing support depends on the storage format used.

Virtio SCSI

Virtio SCSI allows for passing through host SCSI block or generic SCSI devices to the VM Guest, and provides additional storage options in a virtio SCSI interface within the guest.

Macvtap / vhost-net zero-copy transmits

Zero-copy packet transmits from the VM Guest are now possible using vhost-net and macvtap changes that have been added to the latest kernels.

Disk caching modes

The default caching mode for disk images is now *writeback* due to improvements in the handling of the image format. The virtio-blk back-end now automatically switches from 'writeback' to 'writethrough' if the VM Guest virtio driver does not support flushes.

Non-Uniform Memory Access (NUMA)

NUMA machines are supported. Using numactl to pin qemu-kvm processes to specific nodes is recommended.

Supported live migration scenarios

The following host operating system combinations are fully supported for live migrating guests from one host to another: SLES 11 SP3 to SLES 11 SP4, SLES 11 SP4 to SLES 11 SP4 and SLES 11 SP4 to SLES 12. When released, live migrating from SLES 11 SP4 to SLES 12 SP1 will be also supported.

Backwards migration is not supported: SLES 12 to SLES 11 SP4 and SLES 11 SP4 to SLES 11 SP3.

All supported guest systems can be migrated.

Unsupported Features and Tools

Power Management

Changing power states in the host while guests are running is not supported.

Virtio-data-block - data-plane

An experimental block I/O back-end is available using the, x-data-plane=on parameter to -device virtio-blk-pci. This interface allows higher I/O rates. This is not yet supported.

Q35 Machine

A more modern machine type based on the Intel q35 chipset is available. This is not yet supported.

Nested Virtualization

When the svm or vmx CPU feature is passed through to the VM Guest, nested virtualization is possible. This is not yet supported.

Spice

Spice interoperability is not supported.

Glusterfs

Glusterfs interoperability is not supported.

ISCSI

ISCSI integration is not supported. It is however possible for guests to access ISCSI targets available to the host via the blockio interfaces.

RBD (Rados Block Devices)

RBD integration is not supported.

CPU hotplugging

Dynamically changing the number of virtual CPUs assigned to the VM 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 SUSE support personnel.

Guest Agent

The guest agent (qemu-ga) allows programs on the VM Host Server to directly communicate with a VM Guest via an emulated or paravirtualized serial console. This feature is currently not supported.

Using QEMU without KVM (TCG (Tiny Code Generator) mode)

qemu-kvm can be invoked with the -no-kvm parameter. In this case VM 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.

# I/O Virtualization

VM Guests not only share CPU and memory resources of the host system, but also the I/O subsystem. Because software I/O virtualization techniques deliver less performance than bare metal, hardware solutions, that deliver almost "native" performance have been developed recently. SUSE Linux Enterprise Server supports the following I/O virtualization techniques:

Full Virtualization

Fully virtualized drivers emulate widely supported real devices, which can be used with an existing driver in the VM Guest. Since the physical device on the VM Host Server may differ from the emulated one, the hypervisor needs to process all I/O operations before handing them over to the physical device. Therefore all I/O operations need to traverse two software layers, a process that not only significantly impacts I/O performance, but also consumes CPU time.

Paravirtualization

Paravirtualization allows a direct communication between the hypervisor and the VM Guest. With less overhead involved, performance is much better than with full virtualization. However, paravirtualization requires either the guest operating system to be modified to support the paravirtualization API or paravirtualized drivers. See [Section 1.2.1, Availability of Paravirtualized Drivers](https://www.suse.com/documentation/sles11/book_kvm/data/sec_kvm_requires_guests.html#sec_kvm_requires_guests_virt_drivers) for a list of guest operating systems supporting parvirtualization.

Direct Assignment via PCI-Passthrough

Directly assigning a PCI device to a VM Guest (PCI passthrough) avoids performance issues caused by avoiding any emulation in peformance critical paths. With PCI passthough, a VM Guest can directly access the real hardware using a native driver getting almost native performance. This method does not allow to share devices—each device can only be assigned to a single VM Guest. PCI-Passthrough needs to be supported by the VM Host Server CPU, chipset and the BIOS/EFI. The VM Guest needs to be equipped with drivers for the device. See [Section 10.5, Adding a PCI Device with Virtual Machine Manager](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_pci.html) or [Section 10.6, Adding a PCI Device with virsh](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_pci_virsh.html) for setup instructions.

Single Root I/O Virtualization (SR-IOV)

The latest I/O virtualization technique, SR-IOV combines the benefits of the aforementioned techniques— performance and the ability to share a device with several VM Guests. SR-IOV requires special I/O devices, which are capable of replicating resources, so they appear as multiple separate devices. Each such "pseudo" device can be directly used by a single guest. However, for network cards for example the number of concurrent queues that can be used is reduced, potentially reducing performance for the VM Guest compared to paravirtualized drivers. On the VM Host Server SR-IOV must be supported by the I/O device, the CPU and chipset, the BIOS/EFI and the hypervisor. . See [Section 10.7, Adding SR-IOV Devices](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_io.html) for setup instructions.

**IMPORTANT: I/O Virtualization and Live Migration**

Live migration is currently not supported when using devices with PCI passthrough or SR-IOV. In case live migration needs to be supported, you need to use software virtualization (paravirtualization or full Virtualization).

# I/O Virtualization

VM Guests not only share CPU and memory resources of the host system, but also the I/O subsystem. Because software I/O virtualization techniques deliver less performance than bare metal, hardware solutions, that deliver almost "native" performance have been developed recently. SUSE Linux Enterprise Server supports the following I/O virtualization techniques:

Full Virtualization

Fully virtualized drivers emulate widely supported real devices, which can be used with an existing driver in the VM Guest. Since the physical device on the VM Host Server may differ from the emulated one, the hypervisor needs to process all I/O operations before handing them over to the physical device. Therefore all I/O operations need to traverse two software layers, a process that not only significantly impacts I/O performance, but also consumes CPU time.

Paravirtualization

Paravirtualization allows a direct communication between the hypervisor and the VM Guest. With less overhead involved, performance is much better than with full virtualization. However, paravirtualization requires either the guest operating system to be modified to support the paravirtualization API or paravirtualized drivers. See [Section 1.2.1, Availability of Paravirtualized Drivers](https://www.suse.com/documentation/sles11/book_kvm/data/sec_kvm_requires_guests.html#sec_kvm_requires_guests_virt_drivers) for a list of guest operating systems supporting parvirtualization.

Direct Assignment via PCI-Passthrough

Directly assigning a PCI device to a VM Guest (PCI passthrough) avoids performance issues caused by avoiding any emulation in peformance critical paths. With PCI passthough, a VM Guest can directly access the real hardware using a native driver getting almost native performance. This method does not allow to share devices—each device can only be assigned to a single VM Guest. PCI-Passthrough needs to be supported by the VM Host Server CPU, chipset and the BIOS/EFI. The VM Guest needs to be equipped with drivers for the device. See [Section 10.5, Adding a PCI Device with Virtual Machine Manager](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_pci.html) or [Section 10.6, Adding a PCI Device with virsh](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_pci_virsh.html) for setup instructions.

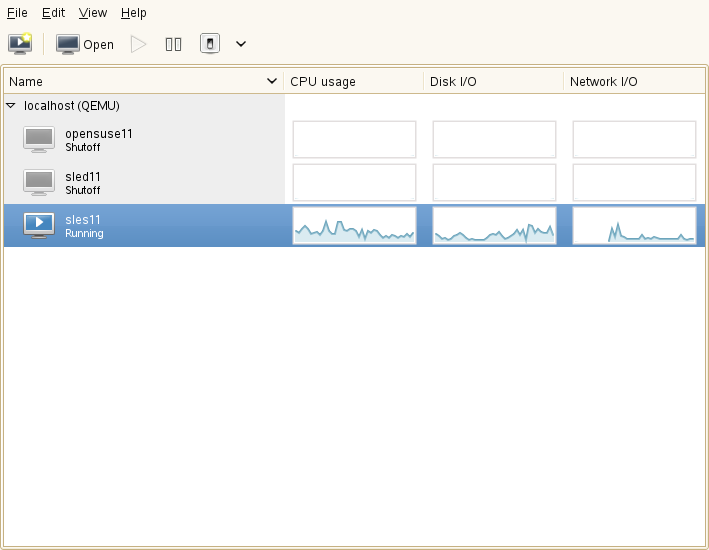
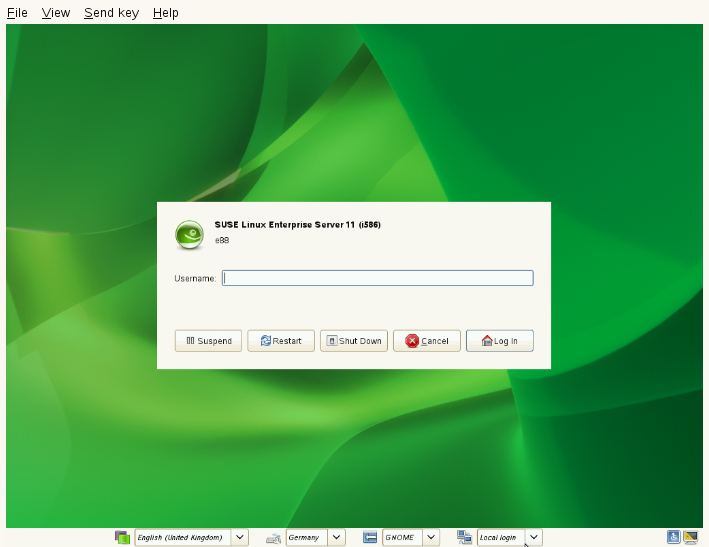
Single Root I/O Virtualization (SR-IOV)

The latest I/O virtualization technique, SR-IOV combines the benefits of the aforementioned techniques— performance and the ability to share a device with several VM Guests. SR-IOV requires special I/O devices, which are capable of replicating resources, so they appear as multiple separate devices. Each such "pseudo" device can be directly used by a single guest. However, for network cards for example the number of concurrent queues that can be used is reduced, potentially reducing performance for the VM Guest compared to paravirtualized drivers. On the VM Host Server SR-IOV must be supported by the I/O device, the CPU and chipset, the BIOS/EFI and the hypervisor. . See [Section 10.7, Adding SR-IOV Devices](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_io.html) for setup instructions.

**IMPORTANT: I/O Virtualization and Live Migration**

Live migration is currently not supported when using devices with PCI passthrough or SR-IOV. In case live migration needs to be supported, you need to use software virtualization (paravirtualization or full Virtualization).

# Managing Virtual Machines with libvirt

* [Section 5.0, Overview](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_overview.html)
* [Section 6.0, Guest Installation](https://www.suse.com/documentation/sles11/book_kvm/data/cha_kvm_inst.html)
* [Section 7.0, Basic VM Guest Management](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_managing.html)
* [Section 8.0, Connecting and Authorizing](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_connect.html)
* [Section 9.0, Managing Storage](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_storage.html)
* [Section 10.0, Configuring Virtual Machines](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_config.html)
* [Section 11.0, Administrating VM Guests](https://www.suse.com/documentation/sles11/book_kvm/data/cha_libvirt_admin.html)
* 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 daemonlibvirtd. 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*] *commanddomain-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://wilber@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
* 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.

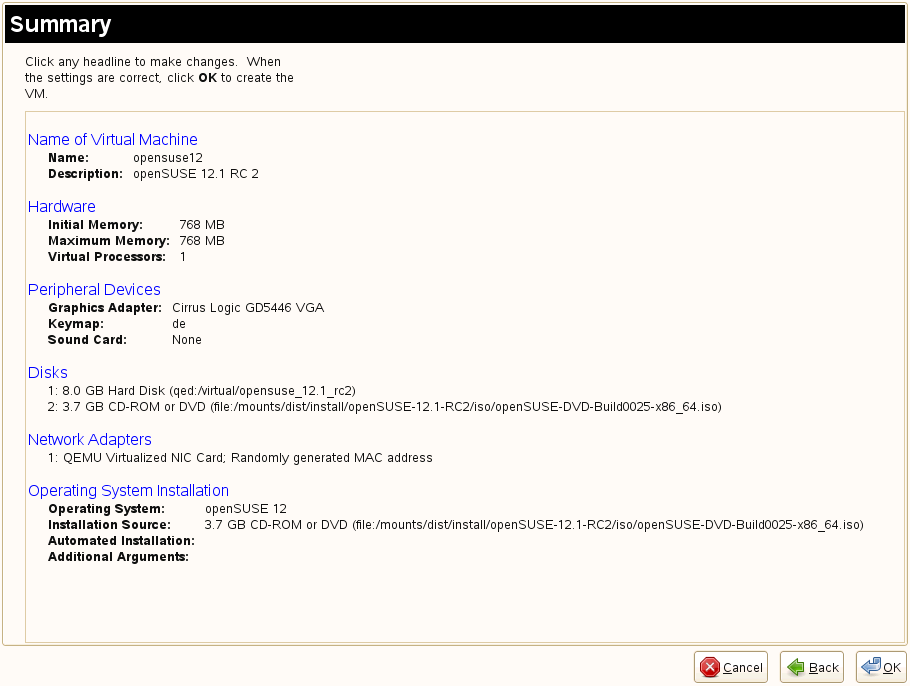
# Guest Installation with Virtual Machine Manager

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.

## 6.1.1Customizing the Default Settings[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_virtman_custom)

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



### *Name of Virtual Machine* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_name)

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* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_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 errors or performance penalties.

This dialog also allows you to assign PCI devices (for example a network card) that can be directly used by the VM Guest (PCI passthrough). Click *Host Devices* > *Manage VM Devices* to get a list of available devices. Choose a device from the list and click *Add* to add it to the devices list for the VM Guest. It is recommended to activate the *Managed* option for each device—it assures that libvirtautomatically takes care of binding and unbinding drivers—see ['managed' vs. 'unmanaged'](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_pci_virsh.html#tip_libvirt_config_pci_virsh_managed) for more information.

**IMPORTANT: PCI devices cannot be shared**

PCI devices cannot be shared between host and VM Guest or between VM Guests—each device can only be used by a single instance. Make sure to only add PCI devices not used elsewhere.

The *Advanced Settings* lets you activate or deactivate ACPI, APIC, and PAE. It is recommended not to change the default settings. You can also enable or disable paravirtualized I/O with virtio or choose to execute the kernel on boot (linux only) here.

**IMPORTANT: Para-Virtualized I/O**

If you enable paravirtualized I/O 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.

### *Peripheral Devices* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_gfx)

Configure the type of virtualized graphics hardware, the keymap and sound device in this dialog. If you disable the graphics card support, the machine is only accessible via network services (ssh) or serial port. Sound in VM Guests is currently not supported by SUSE, therefore *Sound* should be set to None.

### *Disks* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_disk)

*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.

**HINT: 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 sparse raw disk image file is created in /var/lib/kvm/images/*VM\_NAME*/ where*VM\_NAME* is the name of the virtual machine.

**NOTE: Supported Disk format**

Currently, only the disk formats raw, qed and qcow2 are supported by SUSE with read and write support. The vmdk, vpc and vhd/vhdx formats are only supported in read-only mode.

# Creating a Virtual Disk

1. Click *Harddisk*.
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*. For creating raw disks, choose either *file* for file-backed virtual disks or *phy*for device-backed disks. qcow2 or qed disks can be created by choosing the corresponding value.
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* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_nic)

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. If a bridge exists on the VM Host Server, the virtual network card will be attached to it, otherwise it will be attached to the libvirt's default virtual bridge (virbr1).

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*, *Fully Virtualized Intel e1000* or the paravirtualized *QEMU Virtualized NIC Card* (virtio) are supported by SUSE.

1. Choose a *Source* from the drop-down list.
2. 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 obvious MAC addresses such as 52:54:00:12:34:56 or 52:54:00:11:22:33, because they may already be in use. It is strongly recommended to always use a randomly generated MAC address for each adapter.

### *Operating System Installation* [#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vmm.html#cha_kvm_inst_vmm_custom_install)

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, a network resource or via PXE boot—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

# 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 6.2.1, Defining a VM Guest without Starting the Installation](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vm-install.html#sec_libvirt_inst_vm-install_prepare) 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 6-1Interactive 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 6-1](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vm-install.html#ex_libvirt_inst_vm-install_interactive) for which a command line switch exists, are specified. See man 8 vm-install for a full list of parameters.

Example 6-2vm-install command line switches

vm-install --os-type sles11https://www.suse.com/documentation/docui_a/images/1.png --name "sles11\_test"https://www.suse.com/documentation/docui_a/images/2.png \

--vcpus 2https://www.suse.com/documentation/docui_a/images/3.png --memory 512https://www.suse.com/documentation/docui_a/images/4.png --max-memory 768https://www.suse.com/documentation/docui_a/images/5.png \

--disk /var/lib/kvm/images/sles11/hda,0,disk,w,8000,sparse=1 https://www.suse.com/documentation/docui_a/images/6.png \

--disk /iso/SLES-11-SP1-DVD-x86\_64-GM-DVD1.iso,1,cdromhttps://www.suse.com/documentation/docui_a/images/7.png \

--nic mac=52:54:00:05:11:11,model=virtiohttps://www.suse.com/documentation/docui_a/images/8.png \

--graphics cirrushttps://www.suse.com/documentation/docui_a/images/9.png --config-dir "/etc/libvirt/qemu"https://www.suse.com/documentation/docui_a/images/10.png

|  |  |
| --- | --- |
| https://www.suse.com/documentation/docui_a/images/1.png | Specifies the guest operating system to define suitable defaults. A list of valid values can be obtained with vm-install -O. |
| https://www.suse.com/documentation/docui_a/images/2.png | Name of the VM Guest. This name must be unique. |
| https://www.suse.com/documentation/docui_a/images/3.png | Number of virtual processors. |
| https://www.suse.com/documentation/docui_a/images/4.png | Initial amount of memory. |
| https://www.suse.com/documentation/docui_a/images/5.png | Maximum amount of memory. Requires an operating system with a paravirtualized *virtio-balloon*driver. |
| https://www.suse.com/documentation/docui_a/images/6.png | 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). |
| https://www.suse.com/documentation/docui_a/images/7.png | Defines an ISO image for a CD-ROM as the second (1) block device. |
| https://www.suse.com/documentation/docui_a/images/8.png | Configures a paravirtualized network device with the MAC address 52:54:00:05:11:11. |
| https://www.suse.com/documentation/docui_a/images/9.png | Specifies the graphics card. |
| https://www.suse.com/documentation/docui_a/images/10.png | 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. |

## 6.2.1Defining a VM Guest without Starting the Installation[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_vm-install.html#sec_libvirt_inst_vm-install_prepare)

vm-install provides the --no-install parameter. With this parameter the XML configuration file defining the VM Guest is created, but the guest is 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. You can start the installation.

**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*

# Advanced Guest Installation Scenarios

This section provides instructions for operations exceeding the scope of a normal installation, such as including add-on packages.

## 6.3.1Including Add-On Products in the Installation[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_inst_advanced.html#cha_kvm_inst_virtman_advanced_addons)

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.

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.

# 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 8.3, Connecting to a VM Host Server](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_connecting.html) for instructions.

## 7.1.1Listing VM Guests with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_list.html#sec_libvirt_managing_list_vmm)

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.

## 7.1.2Listing VM Guests with virsh[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_list.html#sec_libvirt_managing_list_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 and inactive guests as user wilber on a remote host over a TLS connection

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

List running and inactive guests as user tux on a remote host over an SSH tunnel

virsh -c qemu+ssh://tux@mercury.example.com/system list --inactive

# 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 username (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.

**HINT: Seamless (Absolute) Cursor Movement**

In order to prevent the console from grabbing the cursor and to enable seamless cursor movement, add a tablet input device to the VM Guest. See [Section 10.1, Enabling Seamless and Synchronized Cursor Movement](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_config_tablet.html) for more information.

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. The *Send Key* menu is only available when using Virtual Machine Manager and virt-viewer

**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 isvinagre.

## 7.2.1Opening a Graphical Console with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_vnc.html#sec_libvirt_managing_vnc_vmm)

1. In the Virtual Machine Manager, right-click a VM Guest entry.
2. Choose *Open* from the pop-up menu.

## 7.2.2Opening a Graphical Console with virt-viewer[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_vnc.html#sec_libvirt_managing_vnc_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 -c qemu:///system --wait sles11

With the --wait option, the connection will be upheld even if the VM Guest is not running at the moment. Once the guest starts, the viewer will be launched.

Remote connect via ssh:

virt-viewer -c qemu+ssh://tux@mercury.example.com/system -w sles11

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

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

## 7.3.1Changing a VM Guest's State with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_status.html#sec_libvirt_managing_status_vmm)

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

# 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.

# State change from the VNC Window

1. Open a VNC Window as described in [Section 7.2.1, Opening a Graphical Console with Virtual Machine Manager](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_vnc.html#sec_libvirt_managing_vnc_vmm).
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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_status.html#sec_libvirt_managing_status_vmm_autostart)

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*.

## 7.3.2Changing a VM Guest's State with virsh[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_status.html#sec_libvirt_managing_status_virsh)

In the following examples the state of a VM Guest named sles11 is changed.

Start

virsh -c qemu:///system start sles11

Pause

virsh -c qemu:///system suspend sles11

Reboot

virsh -c qemu:///system reboot sles11

Graceful shutdown

virsh -c qemu:///system shutdown sles11

Force shutdown

virsh -c qemu:///system destroy sles11

Turn on autostart

virsh -c qemu:///system autostart sles11

Turn off autostart

virsh -c qemu:///system autostart --disable sles11

# 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 is 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.

## 7.4.1Saving / Restoring with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_manage_save.html#sec_libvirt_manage_save_vmm)

# 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.

# 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.

## 7.4.2Saving / Restoring with virsh[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_manage_save.html#sec_libvirt_manage_save_virsh)

Save a running VM Guest with the command virsh save and specify the file to where it is saved.

Save the guest named opensuse11

virsh save opensuse11 /virtual/saves/opensuse11.vmsav

Save the guest with the ID 37

virsh save 37 /virtual/saves/opensuse11.vmsave

To restore it, use virsh restore:

virsh restore /virtual/saves/opensuse11.vmsave

# Deleting a VM Guest

Deleting a VM Guest removes its XML configuration by default. Since the attached storage is not deleted by default, 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 7.3, Changing a VM Guest's State: Start, Stop, Pause](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_status.html) for instructions). It is not possible to delete a running guest.

## 7.5.1Deleting a VM Guest with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_delete.html#sec_libvirt_managing_delete_vmm)

1. In the Virtual Machine Manager, right-click 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.

## 7.5.2Deleting a VM Guest with virsh[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_managing_delete.html#sec_libvirt_managing_delete_virsh)

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

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 (via TLS, SSH, Unix sockets, and TCP) that can be combined with different authorization mechanisms (socket, PolicyKit, SASL and Kerberos).

# 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 libvirtdconnections 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. Connections authenticated with SASL are encrypted.
* 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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security) for details.

## 8.1.1libvirtd Authentication[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt)

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 authentication method on SUSE Linux Enterprise Server is access control for UNIX sockets. Only the user root may authenticate. When accessing the libvirt tools as a non-root user directly on the VM Host Server, you need to provide the root password through PolicyKit once and are granted access for the current and for future sessions.

Alternatively you can configure libvirt to allow system access to non-privileged users. See[Section 8.3.1, system Access for Non-Privileged Users](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_connecting.html#sec_libvirt_connect_connecting_dac) for details.

**Recommended Authorization Methods**

Local Connections

* [Local Access Control for UNIX Sockets with PolicyKit](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_pk)
* [Access Control for UNIX Sockets with Permissions and Group Ownership](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_traditional)

Remote Tunnel over SSH

* [Access Control for UNIX Sockets with Permissions and Group Ownership](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_traditional)

Remote TLS/SSL Connection

* [Username and Password Authentication with SASL](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_sasl)
* none (access controlled on the client side by restricting access to the certificates)

### Access Control for UNIX Sockets with Permissions and Group Ownership[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_traditional)

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.

1. Add the desired users to the group:

usermod -A libvirt tux

1. Change the configuration in /etc/libvirt/libvirtd.conf as follows:
2. unix\_sock\_group = "libvirt"https://www.suse.com/documentation/docui_a/images/1.png
3. unix\_sock\_rw\_perms = "0770"https://www.suse.com/documentation/docui_a/images/2.png

auth\_unix\_rw = "none"https://www.suse.com/documentation/docui_a/images/3.png

|  |  |
| --- | --- |
| https://www.suse.com/documentation/docui_a/images/1.png | Group ownership will be set to group libvirt. |
| https://www.suse.com/documentation/docui_a/images/2.png | Sets the access permissions for the socket (srwxrwx---). |
| https://www.suse.com/documentation/docui_a/images/3.png | Disables other authentication methods (PolicyKit or SASL). Access is solely controlled by the socket permissions. |

1. Restart libvirtd:

rclibvirtd restart

### Local Access Control for UNIX Sockets with PolicyKit[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_pk)

Access control for UNIX sockets with PolicyKit is the default authentication method on SUSE Linux Enterprise Server for non-remote connections. 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

**IMPORTANT: PolicyKit Authentication for Local Connections Only**

Authentication with PolicyKit can only be used for local connections on the VM Host Server itself, since PolicyKit does not handle remote authentication.

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:
2. polkit-auth --user tux --grant org.libvirt.unix.manage # grant privilege

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

1. 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:
2. <match action="org.libvirt.unix.manage">https://www.suse.com/documentation/docui_a/images/1.png
3. <match user="tux">https://www.suse.com/documentation/docui_a/images/2.png
4. <return result="yes"/>https://www.suse.com/documentation/docui_a/images/3.png
5. </match>

</match>

|  |  |
| --- | --- |
| https://www.suse.com/documentation/docui_a/images/1.png | The name of the policy; org.libvirt.unix.manage stands for accessing the read-write socket. |
| https://www.suse.com/documentation/docui_a/images/2.png | The username(s) which to grant the privilege. Use the | symbol to separate entries (user="tux|wilber"). |
| https://www.suse.com/documentation/docui_a/images/3.png | 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 orauth\_admin\_keep\_always (authenticate only once with own password/root password). |

### Username and Password Authentication with SASL[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_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.

**HINT: 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:
   1. To enable SASL for TCP connections:

auth\_tcp = "sasl"

* 1. To enable SASL for TLS/SSL connections:

auth\_tls = "sasl"

1. Restart libvirtd:

rclibvirtd restart

1. 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:
2. mech\_list: digest-md5 # mandatory for TCP connections

#mech\_list: digest-md5 # apply default (username+password) TLS/SSL only!

1. By default, no SASL users are configured, so no logins are possible. Use the following commands to add, list, and delete users:
2. mercury:~ # saslpasswd2 -a libvirt tux # add user tux
3. Password:
4. Again (for verification):
5. mercury:~ # sasldblistusers2 -f /etc/libvirt/passwd.db # list users
6. tux@mercury.example.com: userPassword

mercury:~ # saslpasswd2 -a libvirt -d tux # delete user tux

**HINT: 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.

## 8.1.2VNC Authentication[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_vnc)

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 [VNC over TLS/SSL: Client Configuration](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_vnc_client) for details.

### Username and Password Authentication with SASL[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_vnc_tls)

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 [VNC over TLS/SSL: Client Configuration](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_vnc_client) 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

1. By default, no SASL users are configured, so no logins are possible. Use the following commands to add, list, and delete users:
2. mercury:~ # saslpasswd2 -a libvirt tux # add user tux
3. Password:
4. Again (for verification):
5. mercury:~ # sasldblistusers2 -f /etc/libvirt/passwd.db # list users
6. tux@mercury.example.com: userPassword

mercury:~ # saslpasswd2 -a libvirt -d tux # delete user tux

1. Change the configuration in /etc/libvirt/qemu.conf as follows:
2. 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.

1. Restart libvirtd:

rclibvirtd restart

1. 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_vnc_pw)

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.

# Setting a Global VNC Password

1. Change the configuration in /etc/libvirt/qemu.conf as follows:
2. 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.

1. Restart libvirtd:

rclibvirtd restart

1. 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.

# 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"

1. 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*

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

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

1. 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'/>

1. Restart libvirtd:

rclibvirtd restart

1. 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.

# 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.

## 8.2.1Remote Tunnel over SSH (qemu+ssh)[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_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 [Access Control for UNIX Sockets with Permissions and Group Ownership](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_traditional). Connecting as user tux (qemu+ssh://tuxsIVname;/system) works out of the box and does not require additional configuration on the libvirt side.

When connecting via SSH qemu+ssh://*USER*@*SYSTEM* you need to provide the password for *USER*. This can be avoided by copying your public key to ~*USER*/.ssh/authorized\_keys on the VM Host Server as explained in [Section 14.5.2, Copying an SSH Key, (↑*Security Guide*)](https://www.suse.com/documentation/sles11/book_security/data/sec_ssh_authentic.html#sec_ssh_authentic_key_copy). Using an ssh-agent on the machine from which you are connecting adds even more convenience—see [Section 14.5.3, Using the ssh-agent, (↑*Security Guide*)](https://www.suse.com/documentation/sles11/book_security/data/sec_ssh_authentic.html#sec_ssh_authentic_agent) for instructions.

## 8.2.2Remote TLS/SSL Connection with x509 Certificate (qemu+tls)[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls)

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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_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 aclient 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 certificate 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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security) for details. libvirt also supports user authentication on the server with SASL. Read more in [Central User Authentication with SASL for TLS Sockets](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_sasl).

### Configuring the VM Host Server[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_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](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_certificates.html).
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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security).

1. Enable TLS support by editing /etc/libvirt/libvirtd.conf and setting listen\_tls = 1. Restartlibvirtd:

rclibvirtd restart

1. 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 libvirtdwill fail. You also need to restart libvirtd in case you change the certificates.

### Configuring the Client and Testing the Setup[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_client)

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](https://www.suse.com/documentation/sles11/book_kvm/data/app_kvm_certificates.html).
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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security).

1. 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_vnc)

Currently, VNC communication over TLS is only supported by few tools. The widespread tightvnc ortigervnc 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_vnc_server)

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:
3. mkdir -p /etc/pki/libvirt-vnc && cd /etc/pki/libvirt-vnc
4. ln -s /etc/pki/CA/cacert.pem ca-cert.pem
5. ln -s /etc/pki/libvirt/servercert.pem server-cert.pem

ln -s /etc/pki/libvirt/private/serverkey.pem server-key.pem

1. Edit /etc/libvirt/qemu.conf and set the following parameters:
2. vnc\_listen = "0.0.0.0"
3. vnc\_tls = 1

vnc\_tls\_x509\_verify = 1

1. 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_vnc_client)

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[Configuring the Client and Testing the Setup](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_client). 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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security).

### Restricting Access (Security Considerations)[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security)

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 libvirtis 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_server_restrict)

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 [Username and Password Authentication with SASL](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_sasl).

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 wild cards. 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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_sasl)

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 [Restricting Access (Security Considerations)](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_security). 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[Username and Password Authentication with SASL](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_sasl) for configuration details.

### Troubleshooting[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html#sec_libvirt_connect_remote_tls_troubleshoot)

#### Virtual Machine Manager/virsh Cannot Connect to Server[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html)

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[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html)

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

# 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 8.3.2, Managing Connections with Virtual Machine Manager](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_connecting.html#sec_libvirt_connect_connecting_vmm) for details.

*HYPERVISOR*https://www.suse.com/documentation/docui_a/images/1.png*+PROTOCOL*https://www.suse.com/documentation/docui_a/images/2.png://*USER@REMOTE*https://www.suse.com/documentation/docui_a/images/3.png/*CONNECTION\_TYPE*https://www.suse.com/documentation/docui_a/images/4.png

|  |  |
| --- | --- |
| https://www.suse.com/documentation/docui_a/images/1.png | 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. |
| https://www.suse.com/documentation/docui_a/images/2.png | 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). |
| https://www.suse.com/documentation/docui_a/images/3.png | 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. |
| https://www.suse.com/documentation/docui_a/images/4.png | 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).

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.

## 8.3.1system Access for Non-Privileged Users[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_connecting.html#sec_libvirt_connect_connecting_dac)

As mentioned above, a connection to the QEMU hypervisor can be established using two different protocols: session and system. A session connection is spawned with the same privileges as the client program. Such a connection is intended for desktop virtualization, since it is restricted (for example no USB/PCI device assignments, no virtual network setup, limited remote access tolibvirtd).

The system connection intended for server virtualization has no functional restrictions but is, by default, only accessible by root. However, with the addition of the DAC (Discretionary Access Control) driver to libvirt it is now possible to grant non-privileged users system access. To grantsystem access to the user tux, proceed as follows:

# Granting system Access to a Regular User

1. Enable access via UNIX sockets as described in [Access Control for UNIX Sockets with Permissions and Group Ownership](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_auth.html#sec_libvirt_connect_auth_libvirt_traditional). In that example access to libvirt is granted to all members of the group libvirt and tux is made a member of this group. This ensures that tux can connect using virsh or Virtual Machine Manager.
2. Edit /etc/libvirt/qemu.conf and change the configuration as follows:
3. user = "tux"
4. group = "libvirt"

dynamic\_ownership = 1

This ensures that the VM Guests are started by tux and that resources bound to the guest (for example virtual disks) can be accessed and modified by tux.

1. Make tux a member of the group kvm:

usermod -A kvm tux

This step is needed to grant access to /dev/kvm which is required to start VM Guests.

1. Restart libvirtd:

rclibvirtd restart

## 8.3.2Managing Connections with Virtual Machine Manager[#](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_connecting.html#sec_libvirt_connect_connecting_vmm)

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 8.2, Configuring Remote Connections](https://www.suse.com/documentation/sles11/book_kvm/data/sec_libvirt_connect_remote.html) for more information).
4. In case of a remote connection, enter the *Hostname* of the remote machine asUSERNAME@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.

1. If you do not want the connection to be automatically activated when starting the Virtual Machine Manager, remove the tick from *Autoconnect*.
2. Finish the configuration by clicking *Connect*.