QEMU

This article was made for the following Ubuntu versions tested:

- **Ubuntu 22.04** Jammy Jellyfish

Would you like to test the article for another Ubuntu version? Collaboration in the wiki is always welcome! For this purpose, the instructions **for testing articles** must be observed.

To understand this article, the following pages are helpful:

1. **of programs**
2. **Editing package sources**
3. **Opening a terminal**
4. **Opening an editor**
5. **Handling kernel modules (drivers)**

Table of contents

QEMU [http://wiki.qemu.org/Main_Page] is a free **emulator** [https://de.wikipedia.org/wiki/Emulator] and virtualizer. It makes it possible to boot another operating system under Linux, displaying the output in a window, and then using it as if it had been started natively on a computer. A basic introduction to this topic can be found in the article **Virtualization** [https://wiki.ubuntuusers.de/Virtualisierung/].

Some features of QEMU include:

- runs in combination with **KVM** [https://wiki.ubuntuusers.de/KVM/] as a hypervisor with almost native speed
- can also other processor architectures such as Emulating PowerPC or ARM
- can emulate and include different types of drives
- many **options** for starting the guest system, such as Multiprocessor emulation also for single processor systems

QEMU is also available for **other platforms** [https://www.qemu.org/download/] such as Windows and MacOS.

Installation

The QEMU package also includes **KVM;** [https://wiki.ubuntuusers.de/KVM/] it can be installed by the following package **[1]**:

- **qemu-system-x86**

Command to install the packages:

```
sudo apt-get install qemu-system-x86
```

Or install with **apturl** [https://wiki.ubuntuusers.de/apturl/], link: **apt://qemu-system-x86** [apt://qemu-system-x86]

This package only supports guest systems with x86 architecture (32- and 64-bit). If you want to emulate other hardware platforms, you can install the support of other systems with the **qemu system** package. This is a meta package that installs packages for all available target systems.

- **qemu system**

Command to install the packages:

```
sudo apt-get install qemu-system
```

Or install with **apturl** [https://wiki.ubuntuusers.de/apturl/], link: **apt://qemu system** [apt://qemu-system]

Symlink

After that, a link to the desired standard system architecture (typically `i386`or `x86_64`) to be set:

```
sudo ln -s /usr/bin/qemu-system-ARCHITEKTUR /usr/bin/qemu
```

`ARCHITEKTUR`is to be replaced by the desired processor architecture.

QEMU User Mode

If you just want to install the **QEMU user** [https://wiki.ubuntuusers.de/QEMU/#Reine-Prozessemulation] mode:

- **qemu-user**

Command to install the packages:

```
sudo apt-get install qemu-user
```

Or install with **apturl** [https://wiki.ubuntuusers.de/apturl/], link: **apt://qemu-user** [apt://qemu-user]

Hypervisor and emulator

Running host and guest system on the same computer architecture (e.g. x86-64), QEMU can work as a hypervisor through the KVM installed. Then the guest runs directly on the processor of the host and is therefore as fast as a native installation. How to find out whether the host supports this hardware virtualization is described in the [https://wiki.ubuntuusers.de/KVM/]article **KVM** [https://wiki.ubuntuusers.de/KVM/].

For guest systems running on another computer architecture, QEMU emulates the target architecture. This significantly reduces the speed of the guest compared to a native system.

Graphical interfaces and command line tools for QEMU

This article only describes the setup and start parameters for QEMU directly via a terminal[3]. For the sake of completeness, however, some graphical programs for QEMU are listed here, which make it much easier to get started in QEMU. They are sorted by the graphical toolkits used.

- **GTK** [https://wiki.ubuntuusers.de/GTK/]:
  - **virt-manager** [https://wiki.ubuntuusers.de/virt-manager/] - the default tool for QEMU/KVM virtualization, builds on **libvirt** [https://libvirt.org/]
  - **Boxes** [https://wiki.gnome.org/Apps/Boxes] - a simplified variant for gnomes, builds on **libvirt** [https://libvirt.org/]
- **QT** [https://wiki.ubuntuusers.de/QT/]:
  - **AQEMU** [https://wiki.ubuntuusers.de/AQEMU/]
- **Shell** [https://wiki.ubuntuusers.de/Shell/]:
  - **virsh** [https://wiki.ubuntuusers.de/virsh/] - the command line tool for **libvirt** [https://libvirt.org/]

[https://media-cdn.ubuntu-de.org/wiki/attachments/18/28/develop.png]

[https://media-cdn.ubuntu-de.org/wiki/attachments/28/28/download_manager.png]

Installation of a guest operating system

All the following commands are entered in a terminal [3].

Set up the virtual hard disk

To be able to install an operating system in a virtual machine, the first thing to create a virtual hard drive must be created as a file:

```
qemu-img create BOOTSYSTEM.img 10G
```

Instead of `BOOTSYSTEM`one should choose a suitable name. In this case, you create a 10 GB hard disk image in raw format. You can also specify the size of the image in MiB, i.e. by reference to the example above `10240M`(10 * 1024) instead `10G`.

Depending on the type of **file system** [https://wiki.ubuntuusers.de/Dateisystem/] this [https://wiki.ubuntuusers.de/Dateisystem/]image is in, the file may not grow dynamically until data is stored in it. This means that it is initially large 0 bytes, even if most of the programs (e.g.) 10 GiB show. This is the case, for example, with a typical Ubuntu installation on an **ext** [https://wiki.ubuntuusers.de/ext/] file system (based on the host system; the guest in the VM does not matter). If you want to use such **sparse files** [https://de.wikipedia.org/wiki/Sparse-Datei] on older file systems or use other improvements, you can use a different format such as `qed`or `qcow2`indicate:

```
qemu-img create -f qcow2 BOOTSYSTEM.img 10G
```

Installation of CD or CD image

If the hard drive image is created, you can now install an operating system in it. It should be considered from the outset which memory requirements (RAM) have the system to be installed. Accordingly, QEMU will be `-m SPEICHERGRÖSSE`started. The guide values can be found in the table in **the guest system** [https://wiki.ubuntuusers.de/Virtualisierung/#Gastsystem].

Start of the installation process using CD-ROM drive, it will be provided to the 2048 guest system MiB memory:

```
qemu-system-ARCHITEKTUR -enable-kvm -hda BOOTSYSTEM.img -cdrom /dev/cdrom -boot d -m 2048
```

The `-boot d`means that QEMU should boot from CD-ROM. If the installation CD is available as an ISO image, it goes similarly to:

```
qemu-system-ARCHITEKTUR -enable-kvm -hda BOOTSYSTEM.img -cdrom DATEINAME.iso -boot d -m 2048
```

For ARCHITECTUR, the specified default architecture must be inserted (for AMD64, for example, the command is `qemu-system-x86_64`)

After successful installation, the option must be `-boot d`to boot from the virtual hard disk instead of the CD-ROM drive. An ISO image (image file), for example, an installation DVD from Windows can be created with a **burning program** [https://wiki.ubuntuusers.de/Brennprogramme/]. QEMU copes better with ISO images than with DVDs.

Creating overlay images

QEMU offers the possibility to work with overlay files (translated "overlay files"). New and changed data are not written to the original image file but to the overlay file. The original image remains unchanged. This is useful, for example, if you have a "clean" basic installation of a system and are made "Experiments" with the overlay image that do not affect the original. You can also create several overlays for an image.

The following command is used to create an overlay image:

```
qemu-img create -b mein_image.img -F <Format von mein_image.img, z.B. qcow2> -f qcow2 mein_overlay.ovl
```

The name of the overlay file can be arbitrary and do not have to do with the names of the image. After that, you can boot the virtual machine normally, only that instead of the image, the overlay file must be specified as a hard disk, e.g.

```
qemu -hda mein_overlay.ovl
```

Note:

The overlay files are "Delta images", i.e. in the overlay file, changes are saved relative to the original image. If you change the underlying image after an overlay file has been created, the overlay file no longer works!

It is also possible to write changes to the overlay file in the underlying image.

Use hard drives from Virtualbox in QEMU

In the event that you use already existing virtual hard disks in **VirtualBox** [https://wiki.ubuntuusers.de/VirtualBox/] format **.vdi**, you can make them usable for QEMU.

```
VBoxManage clonehd /pfad/zur/virtualbox_platte/system.vdi /pfad/zur/kvm_platte/system.img --format raw
```

Depending on the size of the image, the conversion may take some time.

Full command line to start QEMU

An exemplary, complete command line to start a Lubuntu image:

```
qemu-system-x86_64 \
-enable-kvm \
-cpu host \
-machine q35 \
-device amd-iommu \
-m 4096 \
-smp 4 \
-device virtio-vga-gl -display sdl,gl=on \
-device intel-hda -device hda-duplex \
-device virtio-serial -chardev spicevmc,id=vdagent,debug=0,name=vdagent \
-device virtserialport,chardev=vdagent,name=com.redhat.spice.0 \
-hda lubuntu/lubuntu22.04.qcow2 \
-name "Lubuntu 22.04"
```

[https://media-cdn.ubuntu-de.org/wiki/attachments/22/28/globe.png]

QEMU surroundings

While the operating system is being installed, you should familiarize yourself with the QEMU environment.

Keyboard shortcuts

Keyboard shortcut of QEMU

Keys	Statement
Strg + Old	Free the mouse from the QEMU window
Strg + Old + 2	from guest to the **QEMU monitor**
Strg + Old + 1	switch from QEMU monitor to the guest operating system
Strg + Old + F	Switch between window and full screen mode

QEMU monitor

The QEMU monitor (sometimes called the QEMU console) offers a number of ways to manage and control the virtual machine. This means that there can also be changed from CD-ROM media or floppy disks.

Some of them are:

Commands for the QEMU Monitor

Command	Statement
`info GERÄT`	prints information about the virtual device; possible devices are among other things. `block`(hard disk(s), CD-ROM `snapshot`, `usb`and `network`
`change GERÄT GERÄTEDATEI`	Replaces a removable medium (CD/DVD, floppy disk) (see: **removable drive**)
`commit`	writes a snapshot, if QEMU is `-snapshot`has been started
`screendump DATEI`	creates a screenshot, using the rather unusual file format **ppm**. Example: `screendump BILDNAME.ppm`
`sendkey ctrl-alt-f1`	sends the key combination Strg + Old + F1 to the guest system (calls the **virtual console** [https://wiki.ubuntuusers.de/Terminal/#Virtuelle-Konsole] in an Ubuntu guest)
`help [befehl]`	shows help for all commands or only for the command `befehl`

Removable data storage devices

QEMU cannot automatically determine during virtualization whether a CD or diskette has been inserted or changed. This is done via the QEMU monitor. In the QEMU monitor, you ask about the command `info block`first, which devices are connected and what they are called:

```
info block
ide1-cd0: type=cdrom removable=1 locked=0 file=/dev/cdrom ro=0 drv=host_cdrom encrypted=0
ide0-hd0: type=hd removable=0 file=/PFAD/ZUM/CONTAINER.img ro=0 drv=qcow2 encrypted=0
...
```

In this example, the CD-ROM device is called `ide1-cd0`and the device file **/dev/cdrom**. Accordingly, QEMU is created by entering

```
change ide1-cd0 /dev/cdrom
```

informed that a CD has been inserted/changeed and you can access it with the guest system. When changing floppy disks, the options must be adjusted accordingly.

Screen output

Graphics cards

The default graphics card under QEMU is vga-std. This can be used by the guest operating system without any further drivers.

Graphics card	Option	Properties
Std	-vga std	Standard VGA graphics card, option does not need to be specified
qxl	-vga qxl or -device qxl-vga	Para-virtualized graphics card, is required for SPICE
virtio	-device virtio-vga-gl and -display sdl,glÃ3on or -display gtk,glÃ3on	Paravirtualized graphics card, can also 3D graphics acceleration
vnc	-vnc :0	Graphic output via VNC
no	-nographic	Only one text console is provided

VNC

If you want to operate the virtual machine from another computer, you can run the graphical output of the guest via VNC. To do this, you have to use the option instead of a graphics card `-vnc x`where there are xthe display is. So you start QEMU, for example. over

```
qemu -hda image.img -vnc :1
```

the virtual machine can then **be** [https://wiki.ubuntuusers.de/VNC/#Viewer] reached via port 5901 (5900 + display number) via a **VNC viewer** [https://wiki.ubuntuusers.de/VNC/#Viewer].

Attention!

The connection runs completely unencrypted and without authentication. If necessary for use, precautions urgently need to be taken. Details can be found at **this point** - on the QEMU documentation.

SPICE

SPICE [https://www.spice-space.org/] is similar to guest extensions in VirtualBox or the VMware Tools. Via SPICE, for example, you can adjust the resolution of the guest to the window size in the host and enable a common clipboard of the host and guest.

Turn on Spice on the host

The following parameters must be passed to qemu:

```
qemu-system-x86_64 \
-device qxl-vga,vgamem_mb=32 \
-spice port=3001,password=MeinGeheimesPasswort \
-device virtio-serial -chardev spicevmc,id=vdagent,debug=0,name=vdagent \
-device virtserialport,chardev=vdagent,name=com.redhat.spice.0
```

For remote connections via the port, the graphics card qxl must be used. For resolutions higher than 2560x1440, the graphics card memory must be increased with the option vgamem-mb. In addition, a serial interface between host and guest will be set up, through which the SPICE drivers can communicate with QEMU on the guest. If you do not need a network connection between the viewer and the VM, you can also use a partially faster Unix domain socket instead of the IP port:

```
# Verbindung über Port 3001
-spice port=3001,password=MeinGeheimesPasswort

# Verbindung über Unix-Domain-Socket run/user/$UID/spice.sock
-spice unix=on,addr=/run/user/$UID/spice.sock,password=MeinGeheimesPasswort
```

Resolution	Minimum size graphics card memory
1920x1080, FullHD	8MB, default setting is enough
2560x1440	16MB, default setting is enough
3840x2160, 4k	32MB

If 3D acceleration is activated, Spice must run over a local Unix socket. Then you can set virtio as a graphics card. Unfortunately, this does not work with all systems.

```
-device virtio-vga-gl \
-spice unix=on,addr=/run/user/$UID/spice.sock,password=MeinGeheimesPasswort,gl=on \
```

Attention!

The password in the option "password-MySecre Password" must be replaced by a separate, secure password. It is also not very safe to pass the password from the command line. Better methods can be read in the **Spice user manual**. There you will also find methods of encrypting the connection if this is necessary for the purpose.

Turn on Spice on the guest

The package spice-vdagent and the qxl graphics driver are required. Under Ubuntu these packages must be installed:

- **spice-vdagent**
- **xserver-xorg-video-qxl**

Command to install the packages:

```
sudo apt-get install spice-vdagent xserver-xorg-video-qxl
```

Or install with **apturl** [https://wiki.ubuntuusers.de/apturl/], link: **apt://spice-vdagent,xserver-xorg-video-qxl** [apt://spice-vdagent,xserver-xorg-video-qxl]

In case of problems, it can be checked whether the spice-vdagent daemon is running.

Connecting with the guest via Spice

The Spice client recommended by Spice developers is **remote viewer**.

- **virt viewer**

Command to install the packages:

```
sudo apt-get install virt-viewer
```

Or install with **apturl** [https://wiki.ubuntuusers.de/apturl/], link: **apt://virt-viewer** [apt://virt-viewer]

He is called upon

```
# Über Netzwerk
remote-viewer spice://localhost:3001

# Über Unix-Socket
remote-viewer spice+unix:///run/user/$UID/spice.sock
```

Audio

As a sound card, for example, IntelHD are used:

```
qemu-system-x86_64 -device intel-hda -device hda-duplex
```

Available (audio) devices can be displayed via

```
qemu-system-x86_64 -device help
```

Network

In QEMU, there are several ways to set up a network for the virtual machines.

User Network

The simplest variant is the "user network" where each VM is in its own network segment. The individual network segments are connected to the host network via NAT.

Communication Host - Guest

The host can be reached by a guest system via its normal IP address (routed) and via the IP address 10.0.2.2 (QEMU internal router).

To reach the guest from the host, a port forwarding must be configured.

Communication guest - guest

The individual virtual machines are located in different network segments. To enable them to communicate with each other, a port forwarding is necessary

DHCP

If you have configured your network via **DHCP** [https://de.wikipedia.org/wiki/DHCP], the Internet and the network should work automatically.

TAP Network

With TAP, a virtual network card is created in the host, which is connected to the guest. This allows guests and hosts to communicate with each other without port forwarding. In order for the individual virtual machines to be able to communicate with each other, a network bridge still has to be created in the host.

Creating a TAP with **NetworkManager** [https://wiki.ubuntuusers.de/NetworkManager/]:

```
nmcli connection add type tun ifname tap0 con-name tap0 mode tap owner `id -u`
```

In QEMU, the TAP is configured with these parameters:

```
-netdev tap,id=tap0,ifname=tap0,script=no,downscript=no -device e1000,netdev=tap0,mac=52:54:00:12:34:56
```

Another MAC address has to be set for each VM.

You can then assign static IP addresses to the TAP Device on the host and client sides in order to be able to communicate directly in both directions. If you want to connect several VMs and possibly also allow Internet access via the host, you still have to create a network bridge.

Network bridge

In order for the individual virtual machines to be able to communicate with each other, the TAP networks still need to be connected via a **network** [https://wiki.ubuntuusers.de/Netzwerkbr%C3%BCcke/] **bridge** [https://wiki.ubuntuusers.de/Netzwerkbr%C3%BCcke/].

If you also connect the bridge to the host's network card, the virtual machines have access to the host's network, typically also with Internet access.

The network bridge for Ethernet cannot be connected directly to a WLAN interface; for this purpose you have to route it at the IP level.

Expert information:

Other VMM solutions seem to generate a kind of bridge to Wi-Fi maps by performing NAT at MAC level. There are descriptions on the Internet, as you can achieve this with ebtables, see e.g. **https://wiki.debian.org/BridgeNetworkConnections-Bridging-with-a-wireless-NIC** (untested).

The easiest way to connect the bridge to the host's Wi-Fi map is **NAT** [https://wiki.ubuntuusers.de/Router/NAT/]. In this case, the QEMU guests remain in their own network segment and are routed via the host's normal Internet connection.

Routing

If you want to access the host network without connecting the bridge directly to the real network card, you can instead route the network over the IP level, see **Routing function** [https://wiki.ubuntuusers.de/Router/Routing-Funktion/].

[https://media-cdn.ubuntu-de.org/wiki/attachments/11/28/home.png]

Data exchange between host and guest

Since the host and guest are isolated from each other by QEMU, they cannot simply access each other's data. There are various options for data exchange.

SSH

A **ssh** [https://wiki.ubuntuusers.de/SSH/#Dateitransfer] connection to the guest can be established from the host if it is connected via a TAP network or a port forwarding has been set up. An **SSH server** [https://wiki.ubuntuusers.de/SSH/#Der-SSH-Server] must run on the [https://wiki.ubuntuusers.de/SSH/#Der-SSH-Server]guest system.

Single files per scp

```
# Host sendet an Gast
scp [-P port] datei1 datei2 <IP-Adresse der VM>:/Zielpfad/

# Host liest von Gast
scp [-P port] <IP-Adresse der VM>:/Pfad/datei1 /Zielpfad_Host
```

Folder with sshfs

With **sshfs** [https://wiki.ubuntuusers.de/FUSE/sshfs/], entire folders can be included via ssh.

Single files per netcat

To exchange individual files, you can use the tools **tar** [https://wiki.ubuntuusers.de/tar/] and `nc`if they are available in the host and guest system (e.g. both Linux).

To do this, you execute one command in a terminal **[3]**:

```
# Host sendet an Gast
tar c DATEIEN | nc -v -l 8080           # auf dem Host
nc 10.0.2.2 8080 | tar x                # auf dem Gast

# Gast sendet an Host
nc -v -l 8080 | tar x                   # auf dem Host
tar c DATEIEN | nc 10.0.2.2 8080        # auf dem Gast
```

Partition image

Entire partitions can be copied into a **.img** file and included as a **dd** [https://wiki.ubuntuusers.de/dd/]second hard disk in QEMU:

```
qemu -hda BOOTSYSTEM.img -hdb PFAD/ZUR/PARTITIONSKOPIE.img -m 2048
```

Selected data in an iso-file

A **burning program** [https://wiki.ubuntuusers.de/Brennprogramme/] can be used **to** [https://wiki.ubuntuusers.de/Brennprogramme/] save directories and data from different partitions in an ISO file. This can be integrated as a CD-ROM in QEMU:

```
qemu -hda BOOTSYSTEM.img -cdrom PFAD/ZUR/ISODATEI.iso -m 2048
```

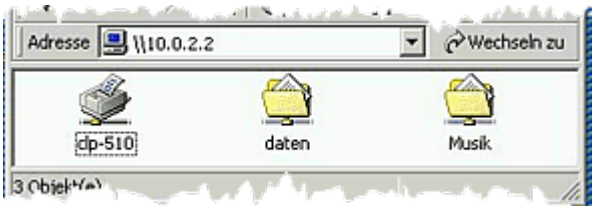
Per Samba

If a **Samba** [https://wiki.ubuntuusers.de/Samba/] **server** [https://wiki.ubuntuusers.de/Samba/] is installed in the host system, you can also access its shares via network with QEMU:

1. When calling QEMU, the option must also `-nic user,smb=/path/to/local/dir`used:

```
qemu -hda BOOTSYSTEM.img -m 2048 -nic user,smb=/path/to/local/dir/
```

   QEMU automatically starts the Samba server with a customized configuration.
2. The guest is accessed via the URL `10.0.2.4`in the directory qemu:
   1. Windows: In the address bar (e.g. from the workstation) simply the URL `\\10.0.2.4\qemu`enter
   2. Ubuntu GUI: In the address bar (e.g. from **Nautilus** [https://wiki.ubuntuusers.de/Nautilus/] or **Dolphin** [https://wiki.ubuntuusers.de/Dolphin/]) the URL `smb://10.0.2.4/qemu`enter
   3. Ubuntu Console: smbclient //10.0.2.4/qemu

 [https://media-cdn.ubuntu-de.org/wiki/attachments/13/04/samba_win.png]	 [https://media-cdn.ubuntu-de.org/wiki/attachments/46/04/samba_ubuntu.png]
Windows	Ubuntu

[https://media-cdn.ubuntu-de.org/wiki/attachments/11/28/settings.png]

Options

Start options

Among the numerous start options, the following can be particularly useful:

Selected options from QEMU

Option	Statement
`-hda Datei`	indicates the image of the primary hard disk. Other plates can be used with `-hdb`, `-hdc`and `-hdd`be specified
`-fda Datei`	indicates floppy drives. You can use the real floppy drive if you `/dev/fd0`indicated as file name
`-cdrom Datei`	specifies the CD drive to use. A device such as **/dev/cdrom** or an image file can be specified
`-daemonize`	solves the QEMU process from the terminal; the terminal can be closed without interference after take-off
`-boot Laufwerksbuchstabe`	indicates which drive should be started. `a`stands for diskette, `c`for hard drive, `d`for CD-ROM and `n`for a network boot
`-m Speichergröße`	specifies the memory to use in MiB. Preparation for this s.o.
`-usb`	USB is emulated with or the interfaces of the host are available. With `-usb -usbdevice tablet`the mouse cursor can be used both in the guest and host system, without constantly cooperating Strg + Old to have to switch
`-vga qxl`	a paravirtualized graphics driver is used. Requires a suitable driver in the guest.
`-soundhw KARTE`	it will be the sound card `KARTE`emulated; there are: `sb16`, `es1370`and `all`
`-smp X`	X CPUs are used in the virtual machine, the number of virtual CPUs may be higher than that of the real of the host
`-vnc :X`	The screen is output via **VNC** [https://wiki.ubuntuusers.de/VNC/] on display `X`and not on the normal screen of the host, see also **here**
`-snapshot`	This causes changes not to be written to the hard drive image, but to be stored in a temporary file. Only with the keys Strg + Old + S or the command **commit** in the QEMU console the changes are adopted
`-k XX`	sets the keyboard layout to the specified value `XX`e.g. `-k de`for German, `-k en`for English, etc... (Helpful for problems with the input and special characters in connection with VNC)
`-hostfwd=[tcp|udp]:[Hostadresse]:Hostport-[Gastadresse]:Gastport`	Continue to a port of the guest on a port of the host. That is. `-hostfwd=::8008::80`makes an Apache server (if default configuration) of the guest system below `http://localhost:8008`visible on the host. Or `-hostfwd=::8022::22`allows ssh access (after standard configuration) to the guest system from the host via `ssh -p 8022 localhost`
`-no-quit`	Disables the window-closing option. Prevents, for example, that when pressing Old + F4 the window is closed and the VM is terminated when the combination was actually intended for the guest (e.g. close window there or to the 4. switch **virtual console** [https://wiki.ubuntuusers.de/Terminal/#Virtuelle-Konsole]). QEMU can continue with `quit`in the **QEMU monitor**
`-cpu X`	Sets the type of the CPU, by `-cpu host`the host CPU can be defined (works only in conjunction with KVM enabled: `-enable-kvm`)

This is just a (very) small section of the options. There are many possibilities, especially in the area of network options. A complete overview can be found in the **man pages** [https://wiki.ubuntuusers.de/man/] or in the **QEMU wiki** [http://wiki.qemu.org/QEMU-doc.html].

Portforwarding (e.g. SSH)

As listed under the **start options**, you can reach services from the guest on the host.

```
qemu [weitere Optionen] -hostfwd=::2222-10.0.2.20:22
```

In this example, port 22 of the guest will be redirected to port 2222 of the host to the guest from the host with

```
ssh -p 2222 localhost
```

to be able to reach.

For this to work properly, the option may need to be embedded in the correct network context - by default, this is the very easy to use, but comparatively very slow and inflexible mode `user network`(User network):

```
qemu [weitere Optionen] -net nic,macaddr=52:54:00:12:34:56 -net user,hostfwd=::2222-10.0.2.20:22
```

The MAC address is the by default preset with `55`at the end, increased by one, and can continue to change or `56`be further increased. Every virtual machine in the network must have its own distinctive MAC.

Emulating Other Architectures

QEMU is not limited to the virtualization / emulation of x86 processors, a variety of other architectures can also be emulated. Whichse are these can be viewed **here** [https://www.qemu.org/docs/master/system/targets.html].

The general syntax is

```
qemu-system-ARCHITEKTUR [OPTIONEN]
```

where `ARCHITEKTUR`must be replaced accordingly.

Pure process emulation

QEMU also masters the "pure" process emulation, also known as "user space emulation". This means that instead of a complete system, "only" a single program ("Binary") will be executed in emulation mode. The process emulation for a 32-bit i386 system is called with the following command:

```
qemu-i386 PROGRAMMNAME
```

The emulation only works, of course, if the program does not dynamically invite any other libraries.

In addition to the i386 emulation, QEMU also masters the process emulation for SPARC, PPC, ARM and some more. Detailed information can be found in the **QEMU documentation**.

Integrating QEMU hard drive images under Linux

QEMU offers the ability to offer the images over the network. However, this can also be used to integrate it on your own computer. But beware: The image must not be in use of QEMU or any other program.

First NBD must be loaded:

```
sudo modprobe nbd
```

If you have a lot of partitions in the image, it may be necessary to increase the number of partitions:

```
sudo modprobe nbd max_part=63
```

Now the image is mentioned in a kind of loopdevice gem. **pfad/zu/qemu.img**, of course, you have to adjust accordingly and you should already be assigned **/dev/nbd0** you can also adjust this number.

```
sudo qemu-nbd --connect=/dev/nbd0 /kompletter/pfad/zu/qemu.img
```

You can use the following command to display the partitions:

```
sudo fdisk -l /dev/nbd0
```

And so you mount the image:

```
sudo mount /dev/nbd0p1 /mnt
```

When you are done, you should release the image again. Adjust numbers if necessary.

```
sudo umount /dev/nbd0p1
sudo qemu-nbd -d /dev/nbd0
```

USB devices through

QEMU is able to pass through USB devices of the host to the guest. For USB support to work in principle, the start option must

`-usb`be specified. There are numerous possibilities to pass through a USB device to the guest system. The following two methods are particularly practicable:

1. Specification of "vendor" and "product ID" of the USB device in question
2. Specification of USB bus and port of the host to which you intend to connect the USB device

1. Method

The "Vendor" and "Product-ID" with

```
lsusb
```

. For example, the output contains a line as follows:

```
Bus 002 Device 026: ID 090c:1000 Silicon Motion, Inc. - Taiwan (formerly Feiya Technology Corp.) 64MB QDI U2
DISK
```

This USB flash drive has the "Vendor-ID" `090c`and "Product-ID" `1000`. The syntax for the corresponding start option is:

```
qemu [OPTIONEN] -usb -device usb-host,bus=usb-bus.0,vendorid=0x<vendor_id>,productid=0x<product_id>
```

In the QEMU monitor you do this with:

```
device_add usb-host,vendorid=0x<vendor_id>:productid=0x<product_id>
```

If everything has gone well, the USB device should now log in to the guest. Unfortunately, there are also more exotic USB devices that cannot be moved to work. Under certain circumstances, the "vendor" and "Product-ID" will not be passed on to the guest system correctly. In a Windows XP guest, for example, this problem is noticeable as an "Unknown Device".

2. Method

If a USB device refuses to cooperate or wants to pass on a selected USB port to a guest in principle, then you can try out the following method. After plugging can be done with:

```
dmesg | tail -n 20
```

the USB bus and port of the device concerned. The interesting part of the output looks like this:

```
[29383.460263] usb 6-1: new full-speed USB device number 14 using uhci_hcd
[29383.622414] usb 6-1: New USB device found, idVendor=10c4, idProduct=8044
[29383.622427] usb 6-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[29383.622436] usb 6-1: Product: USB Debug Adapter
[29383.622444] usb 6-1: Manufacturer: Silicon Laboratories
[29383.622450] usb 6-1: SerialNumber: EC30000D370
[29385.340305] usb 6-1: reset full-speed USB device number 14 using uhci_hcd
[29385.688125] usb 6-1: reset full-speed USB device number 14 using uhci_hcd
```

In this case, the device depends on USB bus 6 port 1. The syntax for the corresponding start option is:

```
qemu [OPTIONEN] -usb -device usb-host,bus=usb-bus.0,hostbus=<bus>,hostport=<port>
```

This is `bus`the USB bus address just determined and `port`the corresponding USB port. In the QEMU monitor this is how it works:

```
device_add usb-host,bus=usb-bus.0,hostbus=<bus>,hostport=<port>
```

Further information can be found in the **USB 2.0 Quick Start** [http://www.kraxel.org/cgit/qemu/tree/docs/usb2.txt] Ã document.

[https://media-cdn.ubuntu-de.org/wiki/attachments/47/28/hint.png]

Troubleshooting

Virtual machine does not get enough RAM

Start QEMU with the option `-m XXX`, but the virtual machine still does not get enough memory, the virtual file system is perhaps too small under **/dev/shm** or from **Ubuntu 11.10** [https://wiki.ubuntuusers.de/Oneiric/] **/run/shm**. Then you can make the following addition in **/etc/fstab** [https://wiki.ubuntuusers.de/fstab/] (adjust directory if necessary) **[4]**:

```
# /dev/shm Vergrößern für QEMU
none            /dev/shm        tmpfs   defaults,size=528M
```

Here stands "size 528M" for 528 MiB. This value may have to be adjusted and should be slightly larger than the RAM actually required for the guest, since further data is stored under **/dev/shm**. Without having to boot again, you can reconnect the tmpfs [3]:

```
sudo mount -o remount /dev/shm
```

If you start QEMU, you can use the command

```
df | egrep 'shm|File'
```

check how much of the virtual RAM is used.

Note:

One should make sure that you no longer assign QEMU to RAM than the host calculator, since QEMU could then run unstable. In addition, the host system should be left enough RAM that it is still running smoothly. The exact system requirements can be found in the release notes of Ubuntu and its derivatives. The lower limit for RAM is 128-512 MiB.

---

ping does not work on the guest after installation

QEMU starts a guest by default in mode `user network` [https://en.wikibooks.org/wiki/QEMU/Networking#User_mode_networking]which is not a `ping`supported.

An alternative to `user network`with `TUN/TAP interface`is explained in the **Network** [https://wiki.ubuntuusers.de/QEMU/#Netzwerk] section.

Alternatively to `ping`other tools can be used to check on the guest after installation whether the Internet connection was established by Gast with the worldwide Internet. An alternative command would be:

```
wget -O - fsf.org 2>&1|grep free|wc
```

where the output should look like this or similar:

```
27     269    2980
```

The VM runs very slow in graphics mode

Most desktop environments use 3D acceleration of graphics for their effects. This cannot be used in all configurations of QEMU in the guest. An alternative is LXQt with the package `lubuntu-desktop`(**Lubuntu** [https://wiki.ubuntuusers.de/Lubuntu/]). 2D acceleration is supported with the virtual graphics card qxl.

If 3D acceleration is to be used in the guest, **you** can **set** as a **graphics** card **-device virtio-vga-gl** and as a display **-display sdl,glÃ3on**.

Virtual hard drive is full

The hard disk image can be enlarged in two steps.

Attention!

Before changing the virtual hard drive, don't forget the backup

First, the image file that is located on the real hard disk of the host is enlarged, e.g. by 10 GB:

```
qemu-img resize Pfad/zum/Image.img +10GB
```

Now the partition of the virtual disk has to be enlarged. This happens like the image enlargement of a real hard disk. To this end, it is best to download an iso of gparted**herunter** [https://gparted.org/download.php]. Then you start the virtual machine from the image, just as you would start a real computer from a CD:

```
qemu -hda Pfad/zum/Image.img -boot d -cdrom Pfad/zum/iso/gparted-live-1.4.0-6-amd64.iso -m 2048
```

Now you first select the language and keyboard layout and then select the option to start the Xserver. Now the size of the partition can be graphically adjusted to the new virtual disk size. To start, only the icon of gparted needs to be clicked.

Links

- **Instructions on Linuxforen.de** [https://www.linuxforen.de/forums/showthread.php?t=141201] - A comprehensive introduction
- **QEMU User Documentation** [https://www.qemu.org/documentation/] - The official documentation
- **Bochs** [http://bochs.sourceforge.net/] - Another free platform emulator
- **Guide to running Windows 7 in QEMU** [https://www.reddit.com/r/archlinux/comments/1fg3y9/guide_to_running_windows_7_in_qemu/] - Manual from 2013 for Arch Linux. Easy to understand and concise, works just as well under Ubuntu
- **Instructions in the OpenMoko-Wiki for QEMU** [http://wiki.openmoko.org/wiki/OpenMoko_under_QEMU] - The **OpenMoko** [https://de.wikipedia.org/wiki/OpenMoko] operating system under QEMU
- **Windows XP under QEMU HowTo** [https://help.ubuntu.com/community/WindowsXPUnderQemuHowTo] - Documentation from Ubuntu.com
- **https://www.linuxtechi.com/install-configure-kvm-ubuntu-18-04-server/** [https://www.linuxtechi.com/install-configure-kvm-ubuntu-18-04-server/] - HowTo Set up network bridge with Netplan

---

This revision [https://wiki.ubuntuusers.de/QEMU/a/revision/1009061/] was held on 12th. January 2023 07:59 created by **kB**.
The following keywords have been assigned to the article: **Emulation and Virtualization** [https://wiki.ubuntuusers.de/wiki/tags/Emulation%20und%20Virtualisierung/], **KVM** [https://wiki.ubuntuusers.de/wiki/tags/KVM/]