VirtualBox

VirtualBox (https://www.virtualbox.org) is a hypervisor used to run operating systems in a special environment, called a virtual machine, on top of the existing operating system. VirtualBox is in constant development and new features are implemented continuously. It comes with a Qt GUI interface, as well as headless and SDL command-line tools for managing and running virtual machines.

In order to integrate functions of the host system to the guests, including shared folders and clipboard, video acceleration and a seamless window integration mode, *guest additions* are provided for some guest operating systems.

Related articles

- VirtualBox/Tips and tricks
- Category:Hypervisors
- PhpVirtualBox
- RemoteBox
- Moving an existing install into (or out of) a virtual machine

Contents

Installation steps for Arch Linux hosts

In order to launch VirtualBox virtual machines on your Arch Linux box, follow these installation steps.

Install the core packages

Install the **`virtualbox (https://www.archlinux.org/packages/?name=virtualbox)`** package. You will need to choose a package to provide host modules:

- for linux (https://www.archlinux.org/packages/?name=linux) kernel choose virtualbox-host-modules-arch (https://www.archlinux.org/packages/?name=virtualbox-host-modules-arch)
- for other **kernels** choose virtualbox-host-dkms (https://www.archlinux.org/packages/?name=virtualbox-host-dkms)
  - It is also necessary to install the appropriate headers package(s) for your installed kernel(s): linux-headers (https://www.archlinux.org/packages/?name=linux-headers) or linux-lts-headers (https://www.archlinux.org/packages/?name=linux-lts-headers). **[1] (https://lists.archlinux.org/pipermail/arch-dev-public/2016-March/027808.html)** When either VirtualBox or the kernel is updated, the kernel modules will be automatically recompiled thanks to the **DKMS** Pacman hook.

Sign modules

When using a custom kernel with `CONFIG_MODULE_SIG_FORCE` option enabled, you must sign your modules with a key generated during kernel compilation.

Navigate to your kernel tree folder and execute the following command:

```
# for module in `ls /lib/modules/$(uname -r)/kernel/misc/{vboxdrv.ko,vboxnetadp.ko,vboxnetflt.ko,vboxpci.ko}` ; do ./scripts/sign-file sha1 certs/signing_key.pem certs/signing_key.x509 $module ; done
```

Note: Hashing algorithm does not have to match the one configured, but it must be built into the kernel.

Load the VirtualBox kernel modules

Since version 5.0.16, virtualbox-host-modules-arch (https://www.archlinux.org/packages/?name=virtualbox-host-modules-arch) and virtualbox-host-dkms (https://www.archlinux.org/packages/?name=virtualbox-host-dkms) use `systemd-modules-load.service` to load all four VirtualBox modules automatically at boot time. For the modules to be loaded after installation, either reboot or load the modules once manually.

Note: If you do not want the VirtualBox modules to be automatically loaded at boot time, you have to mask the default `/usr/lib/modules-load.d/virtualbox-host-modules-arch.conf` (or `/usr/lib/modules-load.d/virtualbox-host-dkms.conf`) by creating an empty file (or symlink to `/dev/null`) with the same name in `/etc/modules-load.d/`.

Among the **kernel modules** VirtualBox uses, there is a mandatory module named `vboxdrv`, which must be loaded before any virtual machines can run.

To load the module manually, run:

```
# modprobe vboxdrv
```

The following modules are optional but are recommended if you do not want to be bothered in some advanced configurations (precised here after): `vboxnetadp`, `vboxnetflt` and `vboxpci`.

- `vboxnetadp` and `vboxnetflt` are both needed when you intend to use the **bridged (https://www.virtualbox.org/manual/ch06.html#network_bridged)** or **host-only networking (https://www.virtualbox.org/manual/ch06.html#network_hostonly)** feature. More precisely, `vboxnetadp` is needed to create the host interface in the VirtualBox global preferences, and `vboxnetflt` is needed to launch a virtual machine using that network interface.
- `vboxpci` is needed when your virtual machine needs to pass through a PCI device on your host.

Note: If the VirtualBox kernel modules were loaded in the kernel while you updated the modules, you need to reload them manually to use the new updated version. To do it, run `vboxreload` as root.

Accessing host USB devices in guest

To use the USB ports of your host machine in your virtual machines, add users that will be authorized to use this feature to the `vboxusers` **group**.

Guest additions disc

It is also recommended to install the `virtualbox-guest-iso (https://www.archlinux.org/packages/?name=virtualbox-guest-iso)` package on the host running VirtualBox. This package will act as a disc image that can be used to install the guest additions onto guest systems other than Arch Linux. The *.iso* file will be located at `/usr/lib/virtualbox/additions/VBoxGuestAdditions.iso`, and may have to be mounted manually inside the virtual machine. Once mounted, you can run the guest additions installer inside the guest.

Extension pack

The Oracle Extension Pack provides **additional features (https://www.virtualbox.org/manual/ch01.html#intro-installing)** and is released under a non-free license **only available for personal use**. To install it, the `virtualbox-ext-oracle (https://aur.archlinux.org/packages/virtualbox-ext-oracle/)`AUR package is available, and a prebuilt version can be found in the **seblu** repository.

If you prefer to use the traditional and manual way: download the extension manually and install it via the GUI (*File > Preferences > Extensions*) or via `VBoxManage extpack install <.vbox-extpack>`, make sure you have a toolkit (like **Polkit**, gksu, etc.) to grant privileged access to VirtualBox. The installation of this extension **requires root access (https://www.virtualbox.org/ticket/8473)**.

Front-ends

VirtualBox comes with three front-ends:

- If you want to use VirtualBox with the regular GUI, use `VirtualBox`.
- If you want to launch and manage your virtual machines from the command-line, use the `VBoxSDL` command, which only provides a plain window for the virtual machine without any overlays.
- If you want to use VirtualBox without running any GUI (e.g. on a server), use the `VBoxHeadless` command. With the VRDP extension you can still remotely access the displays of your virtual machines.

Finally, you can also use **phpVirtualBox** to administrate your virtual machines via a web interface.

Refer to the **VirtualBox manual (https://www.virtualbox.org/manual)** to learn how to create virtual machines.

Warning: If you intend to store virtual disk images on a **Btrfs** file system, before creating any images, you should consider disabling **copy-on-write** for the destination directory of these images.

Installation steps for Arch Linux guests

Boot the Arch installation media through one of the virtual machine's virtual drives. Then, complete the installation of a basic Arch system as explained in the **Installation guide**.

Installation in EFI mode

If you want to install Arch Linux in EFI mode inside VirtualBox, in the settings of the virtual machine, choose *System* item from the panel on the left and *Motherboard* tab from the right panel, and check the checkbox *Enable EFI (special OSes only)*. After selecting the kernel from the Arch Linux installation media's menu, the media will hang for a minute or two and will continue to boot the kernel normally afterwards. Be patient.

Once the system and the boot loader are installed, VirtualBox will first attempt to run `/EFI/BOOT/BOOTX64.EFI` from the **ESP**. If that first option fails, VirtualBox will then try the EFI shell script `startup.nsh` from the root of the ESP. This means that in order to boot the system you have the following options:

- **Launch the bootloader manually** from the EFI shell every time;
- Move the bootloader to the default `/EFI/BOOT/BOOTX64.EFI` path;
- Create a script named `startup.nsh` at the ESP root containing the path to the boot loader application, e.g. `\EFI\grub\grubx64.efi`.
- Boot directly from the ESP partition using a **startup.nsh script**.

Do not bother with the VirtualBox Boot Manager (accessible with `F2` at boot), as it is buggy and incomplete. It doesn't store efivars set interactively. Therefore, EFI entries added to it manually in the firmware (accessed with `F12` at boot time) or with **`efibootmgr (https://www.archlinux.org/packages/?name=efibootmgr)`** will persist after a reboot **but are lost when the VM is shut down (https://www.virtualbox.org/ticket/11177)**.

See also **UEFI VirtualBox installation boot problems (https://bbs.archlinux.org/viewtopic.php?id=158003)**.

Install the Guest Additions

VirtualBox **Guest Additions (https://www.virtualbox.org/manual/ch04.html)** provides drivers and applications that optimize the guest operating system including improved image resolution and better control of the mouse. Within the installed guest system, install:

- `virtualbox-guest-utils (https://www.archlinux.org/packages/?name=virtualbox-guest-utils)` for VirtualBox Guest utilities with X support
- `virtualbox-guest-utils-nox (https://www.archlinux.org/packages/?name=virtualbox-guest-utils-nox)` for VirtualBox Guest utilities without X support

Both packages will make you choose a package to provide guest modules:

- for the default `linux (https://www.archlinux.org/packages/?name=linux)` kernel choose `virtualbox-guest-modules-arch (https://www.archlinux.org/packages/?name=virtualbox-guest-modules-arch)`
- for non-default **kernels** choose `virtualbox-guest-dkms (https://www.archlinux.org/packages/?name=virtualbox-guest-dkms)`

To compile the virtualbox modules provided by `virtualbox-guest-dkms (https://www.archlinux.org/packages/?name=virtualbox-guest-dkms)`, it will also be necessary to install the appropriate headers package(s) for your installed kernel(s) (e.g. `linux-lts-headers (https://www.archlinux.org/packages/?name=linux-lts-headers)` for `linux-lts (https://www.archlinux.org/packages/?name=linux-lts)`). [2] (https://lists.archlinux.org/pipermail/arch-dev-public/2016-March/027808.html) When either VirtualBox or the kernel is updated, the kernel modules will be automatically recompiled thanks to the **DKMS** Pacman hook.

Note:

- You can alternatively install the Guest Additions with the ISO from the `virtualbox-guest-iso (https://www.archlinux.org/packages/?name=virtualbox-guest-iso)` package, provided you installed this on the host system. To do this, go to the device menu click Insert Guest Additions CD Image.
- To recompile the vbox kernel modules, run `rcvboxdrv` as root.

The guest additions running on your guest, and the VirtualBox application running on your host must have matching versions, otherwise the guest additions (like shared clipboard) may stop working. If you upgrade your guest (e.g. `pacman -Syu`), make sure your VirtualBox application on this host is also the latest version. "Check for updates" in the VirtualBox GUI is sometimes not sufficient; check the **VirtualBox.org (https://www.virtualbox.org/)** website.

Set optimal framebuffer resolution

Typically after installing Guest Additions, a fullscreen Arch guest running X will be set to the optimal resolution for your display; however, the virtual console's framebuffer will be set to a standard, often smaller, resolution detected from VirtualBox's custom VESA driver.

To use the virtual consoles at optimal resolution, Arch needs to recognize that resolution as valid, which in turn requires VirtualBox to pass this information along to the guest OS.

First, check if your desired resolution is not already recognized by running the command:

```
hwinfo --framebuffer
```

If the optimal resolution does not show up, then you will need to run the `VBoxManage` tool on the host machine and add "extra resolutions" to your virtual machine (on a Windows host, go to the VirtualBox installation directory to find `VBoxManage.exe`). For example:

```
$ VBoxManage setextradata "Arch Linux" "CustomVideoMode1" "1360x768x24"
```

The parameters "Arch Linux" and "1360x768x24" in the example above should be replaced with your VM name and the desired framebuffer resolution. Incidentally, this command allows for defining up to 16 extra resolutions ("CustomVideoMode1" through "CustomVideoMode16").

Afterwards, restart the virtual machine and run `hwinfo --framebuffer` once more to verify that the new resolutions have been recognized by your guest system (which does not guarantee they will all work, depending on your hardware limitations).

Finally, add a `video=`*`resolution`* **kernel parameter** to set the framebuffer to the new resolution, for example `video=1360x768`.

If you use GRUB as your bootloader, you can edit `/etc/default/grub` to include this kernel parameter in the `GRUB_CMDLINE_LINUX_DEFAULT` list, like so:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet video=1360x768"
```

The GRUB menu itself may also be easily set to optimal resolution, by editing the `GRUB_GFXMODE` option on the same configuration file:

```
GRUB_GFXMODE="1360x768x24"
```

On a standard Arch setup, you would then run `grub-mkconfig -o /boot/grub/grub.cfg` to commit these changes to the bootloader.

After these steps, the framebuffer resolution should be optimized for the GRUB menu and all virtual consoles.

Note: The GRUB settings `GRUB_GFXPAYLOAD_LINUX` and `vga` will not fix the framebuffer, since they are overriden by virtue of Kernel Mode Setting, which is mandatory for using X under VirtualBox and only allows for setting the framebuffer resolution by setting the kernel parameter described above.

Load the VirtualBox kernel modules

To load the modules automatically, enable `vboxservice.service` which loads the modules and synchronizes the guest's system time with the host.

To load the modules manually, type:

```
# modprobe -a vboxguest vboxsf vboxvideo
```

Since version 5.0.16, virtualbox-guest-modules-arch (https://www.archlinux.org/packages/?name=virtualbox-guest-modules-arch) and virtualbox-guest-dkms (https://www.archlinux.org/packages/?name=virtualbox-guest-dkms) use **systemd-modules-load** service to load their modules at boot time.

Note: If you do not want the VirtualBox modules to be loaded at boot time, you have to mask the default `/usr/lib/modules-load.d/virtualbox-guest-modules-arch.conf` (or `-dkms.conf`) by creating an empty file (or symlink to `/dev/null`) with the same name in `/etc/modules-load.d/`.

Launch the VirtualBox guest services

After the rather big installation step dealing with VirtualBox kernel modules, now you need to start the guest services. The guest services are actually just a binary executable called `VBoxClient` which will interact with your X Window System. `VBoxClient` manages the following features:

- shared clipboard and drag and drop between the host and the guest;
- seamless window mode;
- the guest display is automatically resized according to the size of the guest window;
- checking the VirtualBox host version

All of these features can be enabled independently with their dedicated flags:

```
$ VBoxClient --clipboard --draganddrop --seamless --display --checkhostversion
```

As a shortcut, the `VBoxClient-all` bash script enables all of these features.

virtualbox-guest-utils (https://www.archlinux.org/packages/?name=virtualbox-guest-utils) installs `/etc/xdg/autostart/vboxclient.desktop` that launches `VBoxClient-all` on logon. If your **desktop environment** or **window manager** does not support this scheme, you will need to set up autostarting yourself, see **Autostarting#Graphical** for more details.

VirtualBox can also synchronize the time between the host and the guest, to do this, **start/enable** the `vboxservice.service`.

Now, you should have a working Arch Linux guest. Note that features like clipboard sharing are disabled by default in VirtualBox, and you will need to turn them on in the per-VM settings if you actually want to use them (e.g. *Settings > General > Advanced > Shared Clipboard*).

Hardware acceleration

Hardware acceleration can be activated in the VirtualBox options. The **GDM** display manager 3.16+ is known to break hardware acceleration support. **[3] (https://bugzilla.gnome.org/show_bug.cgi?id=749390)** So if you get issues with hardware acceleration, try out another display manager (lightdm seems to work fine). **[4] (https://bbs.archlinux.org/viewtopic.php?id=200025) [5] (https://bbs.archlinux.org/viewtopic.php?pid=1607593#p1607593)**

Enable shared folders

Shared folders are managed on the host, in the settings of the Virtual Machine accessible via the GUI of VirtualBox, in the *Shared Folders* tab. There, *Folder Path*, the name of the mount point identified by *Folder name*, and options like *Read-only*, *Auto-mount* and *Make permanent* can be specified. These parameters can be defined with the `VBoxManage` command line utility. See **there for more details (https://www.virtualbox.org/manual/ch04.html#sharedfolders)**.

No matter which method you will use to mount your folder, all methods require some steps first.

To avoid this issue `/sbin/mount.vboxsf: mounting failed with the error: No such device`, make sure the `vboxsf` kernel module is properly loaded. It should be, since we enabled all guest kernel modules previously.

Two additional steps are needed in order for the mount point to be accessible from users other than root:

- the **virtualbox-guest-utils (https://www.archlinux.org/packages/?name=virtualbox-guest-utils)** package created a group `vboxsf` (done in a previous step);
- your username must be in `vboxsf` **group**.

Manual mounting

Use the following command to mount your folder in your Arch Linux guest:

```
# mount -t vboxsf shared_folder_name mount_point_on_guest_system
```

The vboxsf filesystem offers other options which can be displayed with this command:

```
# mount.vboxsf
```

For example if the user was not in the *vboxsf* group, we could have used this command to give access our mountpoint to him:

```
# mount -t vboxsf -o uid=1000,gid=1000 home /mnt/
```

Where *uid* and *gid* are values corresponding to the users we want to give access to. These values are obtained from the `id` command run against this user.

Automounting

Note: Automounting requires the `vboxservice` to be enabled/started.

In order for the automounting feature to work you must have checked the auto-mount checkbox in the GUI or used the optional `--automount` argument with the command `VBoxManage sharedfolder`.

The shared folder should now appear in `/media/sf_shared_folder_name`. If users in `media` cannot access the shared folders, check that `media` has permissions 755 or has group ownership `vboxsf` if using permission 750. This is currently not the default if media is created by installing the `virtualbox-guest-utils`.

You can use symlinks if you want to have a more convenient access and avoid to browse in that directory, e.g.:

```
$ ln -s /media/sf_shared_folder_name ~/my_documents
```

Mount at boot

You can mount your directory with **fstab**. However, to prevent startup problems with systemd, `noauto,x-systemd.automount` should be added to `/etc/fstab`. This way, the shared folders are mounted only when those mount points are accessed and not during startup. This can avoid some problems, especially if the guest additions are not loaded yet when systemd read fstab and mount the partitions.

```
sharedFolderName  /path/to/mntPtOnGuestMachine  vboxsf  uid=user,gid=group,rw,dmode=700,fmode=600,noauto,x-systemd.automount
```

- `sharedFolderName`: the value from the VirtualMachine's *Settings* > *SharedFolders* > *Edit* > *FolderName* menu. This value can be different from the name of the real folder name on the host machine. To see the VirtualMachine's *Settings* go to the host OS VirtualBox application, select the corresponding virtual machine and click on *Settings*.
- `/path/to/mntPtOnGuestMachine`: if not existing, this directory should be created manually (for example by using **mkdir**)
- `dmode`/`fmode` are directory/file permissions for directories/files inside `/path/to/mntPtOnGuestMachine`.}}

As of 2012-08-02, mount.vboxsf does not support the *nofail* option:

```
desktop  /media/desktop  vboxsf  uid=user,gid=group,rw,dmode=700,fmode=600,nofail  0  0
```

SSH from host to guest

The network tab of the virtual machine settings contains, in "Advanced", a tool to create port forwarding. It is possible to use it to forward the Guest ssh port 22 to a Host port, let's say 3022. Then :

```
user@host$ ssh -p 3022 $USER@localhost
```

will establish a connection from Host to Guest.

SSHFS as alternative to the shared folder

Using this port forwarding and sshfs, it is straightforward to mount the Guest filesystem onto the Host one :

```
user@host$ sshfs -p 3022 $USER@localhost:$HOME ~/shared_folder
```

and then transfer files between both.

Virtual disks management

See also VirtualBox/Tips and tricks#Import/export VirtualBox virtual machines from/to other hypervisors.

Formats supported by VirtualBox

VirtualBox supports the following virtual disk formats:

- **VDI**: The Virtual Disk Image is the VirtualBox own open container used by default when you create a virtual machine with VirtualBox.
- **VMDK**: The Virtual Machine Disk has been initially developed by VMware for their products. The specification was initially closed source, but it became now an open format which is fully supported by VirtualBox. This format offers the ability to be split into several 2GB files. This feature is specially useful if you want to store the virtual machine on machines which do not support very large files. Other formats, excluding the HDD format from Parallels, do not provide such an equivalent feature.
- **VHD**: The Virtual Hard Disk is the format used by Microsoft in Windows Virtual PC and Hyper-V. If you intend to use any of these Microsoft products, you will have to choose this format.

  Tip: Since Windows 7, this format can be mounted directly without any additional application.

- **VHDX** (read only): This is the eXtended version of the Virtual Hard Disk format developed by Microsoft, which has been released on 2012-09-04 with Hyper-V 3.0 coming with Windows Server 2012. This new version of the disk format does offer enhanced performance (better block alignment), larger blocks size, and journal support which brings power failure resiliency. VirtualBox **should support this format in read only (https://www.virtualbox.org/manual/ch15.html#idp63002176)**.

- **HDD** (version 2): The HDD format is developed by Parallels Inc and used in their hypervisor solutions like Parallels Desktop for Mac. Newer versions of this format (i.e. 3 and 4) are not supported due to the lack of documentation for this proprietary format.

  Note: There is currently a controversy regarding the support of the version 2 of the format. While the official VirtualBox manual **only reports the second version of the HDD file format as supported (https://www.virtualbox.org/manual/ch05.html#vdidetails)**, Wikipedia's contributors are **reporting the first version may work too**. Help is welcome if you can perform some tests with the first version of the HDD format.

- **QED**: The QEMU Enhanced Disk format is an old file format for QEMU, another free and open source hypervisor. This format was designed from 2010 in a way to provide a superior alternative to QCOW2 and others. This format features a fully asynchronous I/O path, strong data integrity, backing files, and sparse files. QED format is supported only for compatibility with virtual machines created with old versions of QEMU.

- **QCOW**: The QEMU Copy On Write format is the current format for QEMU. The QCOW format does support zlib-based transparent compression and encryption (the latter is flawed and is not recommended). QCOW is available in two versions: QCOW and QCOW2. QCOW2 tends to supersede the first one. QCOW is **currently fully supported by VirtualBox (https://www.virtualbox.org/manual/ch15.html#idp63002176)**. QCOW2 comes in two revisions: QCOW2 0.10 and QCOW2 1.1 (which is the default when you create a virtual disk with QEMU). VirtualBox does not support QCOW2.

- **OVF**: The Open Virtualization Format is an open format which has been designed for interoperability and distributions of virtual machines between different hypervisors. VirtualBox supports all revisions of this format via the **VBoxManage import/export feature (https://www.virtualbox.org/manual/ch08.html#idp55423424)** but with **known limitations (https://www.virtualbox.org/manual/ch14.html#KnownProblems)**.

- **RAW**: This is the mode when the virtual disk is exposed directly to the disk without being contained in a specific file format container. VirtualBox supports this feature in several ways: converting RAW disk **to a specific format (https://www.virtualbox.org/manual/ch08.html#idp59139136)**, or by **cloning a disk to RAW (https://www.virtualbox.org/manual/ch08.html#vboxmanage-clonevdi)**, or by using directly a VMDK file **which points to a physical disk or a simple file (https://www.virtualbox.org/manual/ch09.html#idp57804112)**.

Disk image format conversion

VBoxManage clonehd (https://www.virtualbox.org/manual/ch08.html#vboxmanage-clonevdi) can be used to convert between VDI, VMDK, VHD and RAW.

```
$ VBoxManage clonehd inputfile outputfile --format outputformat
```

For example to convert VDI to VMDK:

```
$ VBoxManage clonehd source.vdi destination.vmdk --format VMDK
```

QCOW

VirtualBox does not support **QEMU**'s QCOW2 disk image format. To use a QCOW2 disk image with VirtualBox you therefore need to convert it, which you can do with **qemu (https://www.archlinux.org/packages/?name=qemu)**'s `qemu-img` command. `qemu-img` can convert QCOW to / from VDI, VMDK, VHDX, RAW and various other formats (which you can see by running `qemu-img --help`).

```
$ qemu-img convert -O output_fmt inputfile outputfile
```

For example to convert QCOW2 to VDI:

```
$ qemu-img convert -O vdi source.qcow2 destination.vdi
```

Tip: The `-p` parameter is used to get the progression of the conversion task.

There are two revisions of QCOW2: 0.10 and 1.1. You can specify the revision to use with `-o compat=revision`.

Mount virtual disks

VDI

Mounting VDI images only works with fixed size images (a.k.a. static images); dynamic (dynamically size allocating) images are not easily mountable.

The offset of the partition (within the VDI) is needed, then add the value of `offData` to `32256` (e.g. 69632 + 32256 = 101888):

```
$ VBoxManage internalcommands dumphdinfo <storage.vdi> | grep "offData"
```

The can now be mounted with:

```
# mount -t ext4 -o rw,noatime,noexec,loop,offset=101888 <storage.vdi> /mntpoint/
```

You can also use **mount.vdi (https://github.com/pld-linux/VirtualBox/blob/master/mount.vdi)** script that, which you can use as (install script itself to `/usr/bin/`):

```
# mount -t vdi -o fstype=ext4,rw,noatime,noexec vdi_file_location /mnt/
```

Alternately you can use `qemu` (https://www.archlinux.org/packages/?name=qemu)'s kernel module that can do this attrib (http://bethesignal.org/blog/2011/01/05/how-to-mount-virtualbox-vdi-image/):

```
# modprobe nbd max_part=16
# qemu-nbd -c /dev/nbd0 <storage.vdi>
# mount /dev/nbd0p1 /mnt/dir/
# # to unmount:
# umount /mnt/dir/
# qemu-nbd -d /dev/nbd0
```

If the partition nodes are not propagated try using `partprobe /dev/nbd0`; otherwise, a VDI partition can be mapped directly to a node by: `qemu-nbd -P 1 -c /dev/nbd0 <storage.vdi>`.

Compact virtual disks

Compacting virtual disks only works with `.vdi` files and basically consists of the following steps.

Boot your virtual machine and remove all bloat manually or by using cleaning tools like `bleachbit` (https://www.archlinux.org/packages/?name=bleachbit) which is available for Windows systems too (http://bleachbit.sourceforge.net/download/windows).

Wiping free space with zeroes can be achieved with several tools:

- If you were previously using Bleachbit, check the checkbox *System > Free disk space* in the GUI, or use `bleachbit -c system.free_disk_space` in CLI;
- On UNIX-based systems, by using `dd` or preferably **`dcfldd (https://www.archlinux.org/packages/?name=dcfldd)`** (see **here (http://superuser.com/a/355322)** to learn the differences) :

  ```
  # dcfldd if=/dev/zero of=/fillfile bs=4M
  ```

  When `fillfile` reaches the limit of the partition, you will get a message like `1280 blocks (5120Mb) written.dcfldd:: No space left on device`. This means that all of the user-space and non-reserved blocks of the partition will be filled with zeros. Using this command as root is important to make sure all free blocks have been overwritten. Indeed, by default, when using partitions with ext filesystem, a specified percentage of filesystem blocks is reserved for the super-user (see the `-m` argument in the `mkfs.ext4` man pages or use `tune2fs -l` to see how much space is reserved for root applications).
  When the aforementioned process has completed, you can remove the file `fillfile` you created.

- On Windows, there are two tools available:

  - `sdelete` from the **Sysinternals Suite (http://technet.microsoft.com/en-us/sysinternals/bb842062.aspx)**, type `sdelete -s -z c:`, where you need to reexecute the command for each drive you have in your virtual machine;

- or, if you love scripts, there is a **PowerShell solution (http://blog.whatsupduck.net/2012/03/powershell-alternative-to-sdelete.html)**, but which still needs to be repeated for all drives.

```
PS> ./Write-ZeroesToFreeSpace.ps1 -Root c:\ -PercentFree 0
```

Note: This script must be run in a PowerShell environment with administrator privileges. By default, scripts cannot be run, ensure the execution policy is at least on `RemoteSigned` and not on `Restricted`. This can be checked with `Get-ExecutionPolicy` and the required policy can be set with `Set-ExecutionPolicy RemoteSigned`.

Once the free disk space have been wiped, shut down your virtual machine.

The next time you boot your virtual machine, it is recommended to do a filesystem check.

- On UNIX-based systems, you can use `fsck` manually;
  - On GNU/Linux systems, and thus on Arch Linux, you can force a disk check at boot **thanks to a kernel boot parameter**;
- On Windows systems, you can use:
  - either `chkdsk c: /F` where `c:` needs to be replaced by each disk you need to scan and fix errors;

- or `FsckDskAll` **from here (http://therightstuff.de/2009/02/14/ChkDskAll-ChkDsk-For-All-Drives.aspx)** which is basically the same software as `chkdsk`, but without the need to repeat the command for all drives;

Now, remove the zeros from the `vdi` file with **VBoxManage modifyhd (https://www.virtualbox.org/manual/ch08.html#vboxmanage-modifyvdi)**:

```
$ VBoxManage modifyhd your_disk.vdi --compact
```

Note: If your virtual machine has snapshots, you need to apply the above command on each `.vdi` files you have.

Increase virtual disks

General procedure

If you are running out of space due to the small hard drive size you selected when you created your virtual machine, the solution adviced by the VirtualBox manual is to use **VBoxManage modifyhd (https://www.virtualbox.org/manual/ch08.html#vboxmanage-modifyvdi)**. However this command only works for VDI and VHD disks and only for the dynamically allocated variants. If you want to resize a fixed size virtual disk disk too, read on this trick which works either for a Windows or UNIX-like virtual machine.

First, create a new virtual disk next to the one you want to increase:

```
$ VBoxManage createhd -filename new.vdi --size 10000
```

where size is in MiB, in this example 10000MiB ~= 10GiB, and *new.vdi* is name of new hard drive to be created.

Note: By default, this command uses the *Standard* (corresponding to dynamic allocated) file format variant and thus will not use the same file format variant as your source virtual disk. If your *old.vdi* has a fixed size and you want to keep this variant, add the parameter `--variant Fixed`.

Next, the old virtual disk needs to be cloned to the new one which this may take some time:

```
$ VBoxManage clonehd old.vdi new.vdi --existing
```

Detach the old hard drive and attach new one, replace all mandatory italic arguments by your own:

```
$ VBoxManage storageattach VM_name --storagectl SATA --port 0 --medium none
$ VBoxManage storageattach VM_name --storagectl SATA --port 0 --medium new.vdi --type hdd
```

To get the storage controller name and the port number, you can use the command `VBoxManage showvminfo VM_name`. Among the output you will get such a result (what you are looking for is in italic):

```
[...]
Storage Controller Name (0):            IDE
Storage Controller Type (0):            PIIX4
Storage Controller Instance Number (0): 0
Storage Controller Max Port Count (0):  2
Storage Controller Port Count (0):      2
Storage Controller Bootable (0):        on
Storage Controller Name (1):            SATA
Storage Controller Type (1):            IntelAhci
Storage Controller Instance Number (1): 0
Storage Controller Max Port Count (1):  30
Storage Controller Port Count (1):      1
Storage Controller Bootable (1):        on
IDE (1, 0): Empty
SATA (0, 0): /home/wget/IT/Virtual_machines/GNU_Linux_distributions/ArchLinux_x64_EFI/Snapshots/{6bb17af7-e8a2-4bbf-baac-fbba05ebd704}.vdi (UUID: 6bb
17af7-e8a2-4bbf-baac-fbba05ebd704)
[...]
```

Download **GParted live image (http://gparted.org/download.php)** and mount it as a virtual CD/DVD disk file, boot your virtual machine, increase/move your partitions, umount GParted live and reboot.

Note: On GPT disks, increasing the size of the disk will result in the backup GPT header not being at the end of the device. GParted will ask to fix this, click on *Fix* both times. On MBR disks, you do not have such a problem as this partition table as no trailer at the end of the disk.

Finally, unregister the virtual disk from VirtualBox and remove the file:

```
$ VBoxManage closemedium disk old.vdi
$ rm old.vdi
```

Increasing the size of VDI disks

If your disk is a VDI one, run:

```
$ VBoxManage modifyhd your_virtual_disk.vdi --resize the_new_size
```

Then jump back to the Gparted step, to increase the size of the partition on the virtual disk.

Replace a virtual disk manually from the .vbox file

If you think that editing a simple *XML* file is more convenient than playing with the GUI or with `VBoxManage` and you want to replace (or add) a virtual disk to your virtual machine, in the *.vbox* configuration file corresponding to your virtual machine, simply replace the GUID, the file location and the format to your needs:

```
ArchLinux_vm.vbox
-----------------
<HardDisk uuid="{670157e5-8bd4-4f7b-8b96-9ee412a712b5}" location="ArchLinux_vm.vdi" format="VDI" type="Normal"/>
```

then in the `<AttachedDevice>` sub-tag of `<StorageController>`, replace the GUID by the new one.

```
ArchLinux_vm.vbox
-----------------
<AttachedDevice type="HardDisk" port="0" device="0">
  <Image uuid="{670157e5-8bd4-4f7b-8b96-9ee412a712b5}"/>
</AttachedDevice>
```

Note: If you do not know the GUID of the drive you want to add, you can use the `VBoxManage showhdinfo file`. If you previously used `VBoxManage clonehd` to copy/convert your virtual disk, this command should have outputted the GUID just after the copy/conversion completed. Using a random GUID does not work, as each **UUID is stored inside each disk image (http://www.virtualbox.org/manual/ch05.html#cloningvdis)**.

Transfer between Linux host and other OS

The information about path to harddisks and the snapshots is stored between `<HardDisks> .... </HardDisks>` tags in the file with the *.vbox* extension. You can edit them manually or use this script where you will need change only the path or use defaults, assumed that *.vbox* is in the same directory with a virtual harddisk and the snapshots folder. It will print out new configuration to stdout.

```
#!/bin/bash
NewPath="${PWD}/"
Snapshots="Snapshots/"
Filename="$1"

 awk -v SetPath="$NewPath" -v SnapPath="$Snapshots" '{if(index($0,"<HardDisk uuid=") != 0){A=$3;split(A,B,"=");
L=B[2];
 gsub(/\"/,"",L);
  sub(/^.*\//,"",L);
  sub(/^.*\\/,"",L);
 if(index($3,"{") != 0){SnapS=SnapPath}else{SnapS=""};
  print $1" "$2" location="\"SetPath SnapS L"\" "$4" "$5}
else print $0}' "$Filename"
```

Note:

- If you will prepare virtual machine for use in Windows host then in the path name end you should use backslash \ instead of / .
- The script detects snapshots by looking for `{` in the file name.
- To make it run on a new host you will need to add it first to the register by clicking on **Machine -> Add...** or use hotkeys Ctrl+A and then browse to *.vbox* file that contains configuration or use command line `VBoxManage registervm filename.vbox`

Clone a virtual disk and assigning a new UUID to it

UUIDs are widely used by VirtualBox. Each virtual machines and each virtual disk of a virtual machine must have a different UUID. When you launch a virtual machine in VirtualBox, VirtualBox will keep track of all UUIDs of your virtual machine instance. See the **VBoxManage list (http://www.virtualbox.org/manual/ch08.html#vboxmanage-list)** to list the items registered with VirtualBox.

If you cloned a virtual disk manually by copying the virtual disk file, you will need to assign a new UUID to the cloned virtual drive if you want to use the disk in the same virtual machine or even in another (if that one has already been opened, and thus registered, with VirtualBox).

You can use this command to assign a new UUID to a virtual disk:

```
$ VBoxManage internalcommands sethduuid /path/to/disk.vdi
```

Tip: To avoid copying the virtual disk and assigning a new UUID to your file manually you can use **VBoxManage clonehd (http://www.virtualbox.org/manual/ch08.html#vboxmanage-clonevdi)**.

Note: The commands above support all **virtual disk formats supported by VirtualBox**.

Tips and tricks

For advanced configuration, see **VirtualBox/Tips and tricks**.

Troubleshooting

Keyboard and mouse are locked into virtual machine

This means your virtual machine has captured the input of your keyboard and your mouse. Simply press the right `Ctrl` key and your input should control your host again.

To control transparently your virtual machine with your mouse going back and forth your host, without having to press any key, and thus have a seamless integration, install the guest additions inside the guest. Read from the **#Install the Guest Additions** step if you guest is Arch Linux, otherwise read the official VirtualBox help.

No 64-bit OS client options

When launching a VM client, and no 64-bit options are available, make sure your CPU virtualization capabilities (usually named `VT-x`) are enabled in the BIOS.

If you are using a Windows host, you may need to disable Hyper-V, as it prevents VirtualBox from using VT-x. [6] (https://www.virtualbox.org/ticket/12350)

VirtualBox GUI does not match host GTK theme

See **Uniform look for Qt and GTK applications** for information about theming Qt-based applications like VirtualBox.

Cannot send Ctrl+Alt+Fn to guest

Your guest operating system is a GNU/Linux distribution and you want to open a new TTY shell by hitting `Ctrl+Alt+F2` or exit your current X session with `Ctrl+Alt+Backspace`. If you type these keyboard shortcuts without any adaptation, the guest will not receive any input and the host (if it is a GNU/Linux distribution too) will intercept these shortcut keys. To send `Ctrl+Alt+F2` to the guest for example, simply hit your *Host Key* (usually the right `Ctrl` key) and press `F2` simultaneously.

USB subsystem not working

Your user must be in the `vboxusers` group and you need to install the **extension pack** if you want USB 2 support. Then you will be able to enable USB 2 in the VM settings and add one or several filters for the devices you want to access from the guest OS.

If `VBoxManage list usbhost` does not show any USB devices even if run as root, make sure that there is no old udev rules (from VirtualBox 4.x) in `/etc/udev/rules.d/`. VirtualBox 5.0 installs udev rules to `/usr/lib/udev/rules.d/`. You can use command like `pacman -Qo /usr/lib/udev/rules.d/60-vboxdrv.rules` to determine if the udev rule file is outdated.

Sometimes, on old Linux hosts, the USB subsystem is not auto-detected resulting in an error `Could not load the Host USB Proxy service: VERR_NOT_FOUND` or in a not visible USB drive on the host, **even when the user is in the vboxusers group (https://bbs.archlinux.org/viewtopic.php?id=121377)**. This problem is due to the fact that VirtualBox switched from *usbfs* to *sysfs* in version 3.0.8. If the host does not understand this change, you can revert to the old behaviour by defining the following environment variable in any file that is sourced by your shell (e.g. your `~/.bashrc` if you are using *bash*):

```
~/.bashrc
```

```
VBOX_USB=usbfs
```

Then make sure, the environment has been made aware of this change (reconnect, source the file manually, launch a new shell instance or reboot).

Also make sure that your user is a member of the `storage` group.

USB modem not working on host

If you have a USB modem which is being used by the guest OS, killing the guest OS can cause the modem to become unusable by the host system. Killing and restarting `VBoxSVC` should fix this problem.

Access serial port from guest

Check you permission for the serial port:

```
$ ls -l /dev/ttyS*
```

```
crw-rw---- 1 root uucp 4, 64 Feb  3 09:12 /dev/ttyS0
crw-rw---- 1 root uucp 4, 65 Feb  3 09:12 /dev/ttyS1
crw-rw---- 1 root uucp 4, 66 Feb  3 09:12 /dev/ttyS2
crw-rw---- 1 root uucp 4, 67 Feb  3 09:12 /dev/ttyS3
```

Add your user to the `uucp` **group**.

Guest freezes after starting Xorg

Faulty or missing drivers may cause the guest to freeze after starting Xorg, see for example **[7] (https://bbs.archlinux.org/viewtopic.php?pid=1167838)** and **[8] (https://bbs.archlinux.org/viewtopic.php?id=156079)**. Try disabling 3D acceleration in *Settings > Display*, and check if all Xorg drivers are installed.

Fullscreen mode shows blank screen

On some window managers (i3, awesome), VirtualBox has issues with fullscreen mode properly due to the overlay bar. To work around this issue, disable "Show in Full-screen/Seamless" option in "Guest Settings > User Interface > Mini ToolBar". See the upstream bug report (https://www.virtualbox.org/ticket/14323) for more information.

Host freezes on virtual machine start

Possible causes/solutions :

- SMAP

This is a known incompatiblity with SMAP enabled kernels affecting (mostly) Intel Broadwell chipsets. A solution to this problem is disabling SMAP support in your kernel by appending the `nosmap` option to your kernel parameters.

- Hardware Virtualisation

Disabling hardware virtualisation (VT-x/AMD-V) may solve the problem.

- Various Kernel bugs

- Fuse mounted partitions (like ntfs) [9] (https://bbs.archlinux.org/viewtopic.php?id=185841), [10] (https://bugzilla.kernel.org/show_bug.cgi?id=82951#c12)

Generally, such issues are observed after upgrading VirtualBox or linux kernel. Downgrading them to the previous versions of theirs might solve the problem.

Linux guests have slow/distorted audio

The AC97 audio driver within the Linux kernel occasionally guesses the wrong clock settings when running inside Virtual Box, leading to audio that is either too slow or too fast. To fix this, create a file in `/etc/modprobe.d/` with the following line:

```
options snd-intel8x0 ac97_clock=48000
```

Analog microphone not working

If the audio input from an analog microphone is working correctly on the host, but no sound seems to get through to the guest, despite the microphone device apparently being detected normally, installing a **sound server** such as **PulseAudio** on the host might fix the problem.

If after installing **PulseAudio** the microphone still refuses to work, setting *Host Audio Driver* (under *VirtualBox > Machine > Settings > Audio*) to *ALSA Audio Driver* might help.

Microphone not working after upgrade

There have been issues reported around sound input in 5.1.x versions. [11] (https://forums.virtualbox.org/viewtopic.php?f=7&t=78797)

Downgrading may solve the problem. You can use `virtualbox-bin-5.0 (https://aur.archlinux.org/packages/virtualbox-bin-5.0/)`AUR to ease downgrading.

Problems with images converted to ISO

Some image formats cannot be reliably converted to ISO. For instance, `ccd2iso (https://www.archlinux.org/packages/?name=ccd2iso)` ignores .ccd and .sub files, which can result in disk images with broken files.

In this case, you will either have to use CDemu for Linux inside VirtualBox or any other utility used to mount disk images.

Failed to create the host-only network interface

Make sure all required kernel modules are loaded. See #Load the VirtualBox kernel modules.

Failed to insert module

When you get the following error when trying to load modules:

```
Failed to insert 'vboxdrv': Required key not available
```

Sign your modules or disable `CONFIG_MODULE_SIG_FORCE` in your kernel config.

VBOX_E_INVALID_OBJECT_STATE (0x80BB0007)

This can occur if a VM is exited ungracefully. Run the following command:

```
$ VBoxManage controlvm virtual_machine_name poweroff
```

NS_ERROR_FAILURE and missing menu items

This happens sometimes when selecting *QCOW*/*QCOW2*/*QED* disk format when creating a new virtual disk.

If you encounter this message the first time you start the virtual machine:

```
Failed to open a session for the virtual machine debian.
Could not open the medium '/home/.../VirtualBox VMs/debian/debian.qcow'.
QCow: Reading the L1 table for image '/home/.../VirtualBox VMs/debian/debian.qcow' failed (VERR_EOF).
VD: error VERR_EOF opening image file '/home/.../VirtualBox VMs/debian/debian.qcow' (VERR_EOF).

Result Code:
NS_ERROR_FAILURE (0x80004005)
Component:
Medium
```

Exit VirtualBox, delete all files of the new machine and from virtualbox config file remove the last line in `MachineRegistry` menu (or the offending machine you are creating):

```
~/.config/VirtualBox/VirtualBox.xml
```

```
...
<MachineRegistry>
  <MachineEntry uuid="{00000000-0000-0000-0000-000000000000}" src="/home/void/VirtualBox VMs/debian/debian.vbox"/>
  <MachineEntry uuid="{00000000-0000-0000-0000-000000000000}" src="/home/void/VirtualBox VMs/ubuntu/ubuntu.vbox"/>
  ~~<MachineEntry uuid="{00000000-0000-0000-0000-000000000000}" src="/home/void/VirtualBox VMs/lastvmcausingproblems/lastvmcausingproblems.qcow"/>~~
</MachineRegistry>
...
```

Arch: pacstrap script fails

If you used *pacstrap* in the **#Installation steps for Arch Linux guests** to also **#Install the Guest Additions** **before** performing a first boot into the new guest, you will need to `umount -l /mnt/dev` as root before using *pacstrap* again; a failure to do this will render it unusable.

OpenBSD unusable when virtualisation instructions unavailable

While OpenBSD is reported to work fine on other hypervisors without virtualisation instructions (VT-x AMD-V) enabled, an OpenBSD virtual machine running on VirtualBox without these instructions will be unusable, manifesting with a bunch of segmentation faults. Starting VirtualBox with the *-norawr0* argument **may solve the problem (https://www.virtualbox.org/ticket/3947)**. You can do it like this:

```
$ VBoxSDL -norawr0 -vm name_of_OpenBSD_VM
```

Windows host: VERR_ACCESS_DENIED

To access the raw VMDK image on a Windows host, run the VirtualBox GUI as administrator.

Windows: "The specified path does not exist. Check the path and then try again."

This error message may appear when running an `.exe` file which requires administrator privileges from a shared folder in windows guests. [12] (https://www.virtualbox.org/ticket/5732#comment:39)

As a workaround, copy the file to the virtual drive or use UNC paths (`\\vboxsvr`). See [13] (https://support.microsoft.com/de-de/help/2019185/copying-files-from-a-mapped-drive-to-a-local-directory-fails-with-erro) for more information.

Windows 8.x error code 0x000000C4

If you get this error code while booting, even if you choose OS Type Win 8, try to enable the `CMPXCHG16B` CPU instruction:

```
$ vboxmanage setextradata virtual_machine_name VBoxInternal/CPUM/CMPXCHG16B 1
```

Windows 8, 8.1 or 10 fails to install, boot or has error "ERR_DISK_FULL"

Update the VM's settings by going to *Settings > Storage > Controller:SATA* and check "Use Host I/O Cache".

WinXP: Bit-depth cannot be greater than 16

If you are running at 16-bit color depth, then the icons may appear fuzzy/choppy. However, upon attempting to change the color depth to a higher level, the system may restrict you to a lower resolution or simply not enable you to change the depth at all. To fix this, run `regedit` in Windows and add the following key to the Windows XP VM's registry:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services]
"ColorDepth"=dword:00000004
```

Then update the color depth in the "desktop properties" window. If nothing happens, force the screen to redraw through some method (i.e. `Host+f` to redraw/enter full screen).

See also

- VirtualBox User Manual (https://www.virtualbox.org/manual/UserManual.html)
- Wikipedia:VirtualBox

Retrieved from "https://wiki.archlinux.org/index.php?title=VirtualBox&oldid=510504"