NAME

virt−resize − Resize a virtual machine disk

SYNOPSIS

```
 virt-resize [--resize /dev/sdaN=[+/-]<size>[%]]
   [--expand /dev/sdaN] [--shrink /dev/sdaN]
   [--ignore /dev/sdaN] [--delete /dev/sdaN] [...] indisk outdisk
```

DESCRIPTION

Virt-resize is a tool which can resize a virtual machine disk, making it larger or smaller overall, and resizing or deleting any partitions contained within.

Virt-resize **cannot** resize disk images in-place. Virt-resize **should not** be used on live virtual machines − for consistent results, shut the virtual machine down before resizing it.

If you are not familiar with the associated tools: **virt−filesystems**(1) and **virt−df**(1), we recommend you go and read those manual pages first.

EXAMPLES

1. This example takes `olddisk` and resizes it into `newdisk`, extending one of the guest's partitions to fill the extra 5GB of space:

```
virt-filesystems --long -h --all -a olddisk

truncate -r olddisk newdisk
truncate -s +5G newdisk

# Note "/dev/sda2" is a partition inside the "olddisk" file.
virt-resize --expand /dev/sda2 olddisk newdisk
```

2. As above, but make the /boot partition 200MB bigger, while giving the remaining space to /dev/sda2:

```
virt-resize --resize /dev/sda1=+200M --expand /dev/sda2 \
  olddisk newdisk
```

3. As in the first example, but expand a logical volume as the final step. This is what you would typically use for Linux guests that use LVM:

```
virt-resize --expand /dev/sda2 --LV-expand /dev/vg_guest/lv_root \
  olddisk newdisk
```

4. As in the first example, but the output format will be qcow2 instead of a raw disk:

```
qemu-img create -f qcow2 -o preallocation=metadata newdisk.qcow2 15G
virt-resize --expand /dev/sda2 olddisk newdisk.qcow2
```

DETAILED USAGE

EXPANDING A VIRTUAL MACHINE DISK

1. Shut down the virtual machine

2. Locate input disk image

Locate the input disk image (ie. the file or device on the host containing the guest's disk). If the guest is managed by libvirt, you can use `virsh dumpxml` like this to find the disk image name:

```
# virsh dumpxml guestname | xpath /domain/devices/disk/source
Found 1 nodes:
-- NODE --
<source dev="/dev/vg/lv_guest" />
```

3. Look at current sizing

Use **virt−filesystems**(1) to display the current partitions and sizes:

```
# virt-filesystems --long --parts --blkdevs -h -a /dev/vg/lv_guest
Name       Type       Size  Parent
/dev/sda1  partition  101M  /dev/sda
/dev/sda2  partition  7.9G  /dev/sda
/dev/sda   device     8.0G  -
```

(This example is a virtual machine with an 8 GB disk which we would like to expand up to 10 GB).

4. Create output disk

Virt-resize cannot do in-place disk modifications. You have to have space to store the resized output disk.

To store the resized disk image in a file, create a file of a suitable size:

```
# rm -f outdisk
# truncate -s 10G outdisk
```

Or use **lvcreate** (1) to create a logical volume:

```
# lvcreate -L 10G -n lv_name vg_name
```

Or use **virsh** (1) vol-create-as to create a libvirt storage volume:

```
# virsh pool-list
# virsh vol-create-as poolname newvol 10G
```

5. Resize

virt-resize takes two mandatory parameters, the input disk and the output disk (both can be e.g. a device, a file, or a URI to a remote disk). The output disk is the one created in the previous step.

```
# virt-resize indisk outdisk
```

This command just copies disk image `indisk` to disk image `outdisk` *without* resizing or changing any existing partitions. If `outdisk` is larger, then an extra, empty partition is created at the end of the disk covering the extra space. If `outdisk` is smaller, then it will give an error.

More realistically you'd want to expand existing partitions in the disk image by passing extra options (for the full list see the "OPTIONS" section below).

"--expand" is the most useful option. It expands the named partition within the disk to fill any extra space:

```
# virt-resize --expand /dev/sda2 indisk outdisk
```

(In this case, an extra partition is *not* created at the end of the disk, because there will be no unused space).

"--resize" is the other commonly used option. The following would increase the size of /dev/sda1 by 200M, and expand /dev/sda2 to fill the rest of the available space:

```
# virt-resize --resize /dev/sda1=+200M --expand /dev/sda2 \
    indisk outdisk
```

If the expanded partition in the image contains a filesystem or LVM PV, then if virt-resize knows how, it will resize the contents, the equivalent of calling a command such as **pvresize** (8), **resize2fs** (8), **ntfsresize** (8), **btrfs** (8), **xfs_growfs** (8), or **resize.f2fs** (8). However virt-resize does not know how to resize some filesystems, so you would have to online resize them after booting the guest.

```
# virt-resize --expand /dev/sda2 nbd://example.com outdisk
```

The input disk can be a URI, in order to use a remote disk as the source. The URI format is compatible with guestfish. See "ADDING REMOTE STORAGE" in **guestfish** (1).

Other options are covered below.

6. Test

   Thoroughly test the new disk image *before* discarding the old one.

   If you are using libvirt, edit the XML to point at the new disk:

   ```
    # virsh edit guestname
   ```

   Change <source ...>, see http://libvirt.org/formatdomain.html#elementsDisks

   Then start up the domain with the new, resized disk:

   ```
    # virsh start guestname
   ```

   and check that it still works. See also the ''NOTES'' section below for additional information.

7. Resize LVs etc inside the guest

   (This can also be done offline using **guestfish** (1))

   Once the guest has booted you should see the new space available, at least for filesystems that virt-resize knows how to resize, and for PVs. The user may need to resize LVs inside PVs, and also resize filesystem types that virt-resize does not know how to expand.

SHRINKING A VIRTUAL MACHINE DISK

Shrinking is somewhat more complex than expanding, and only an overview is given here.

Firstly virt-resize will not attempt to shrink any partition content (PVs, filesystems). The user has to shrink content before passing the disk image to virt-resize, and virt-resize will check that the content has been shrunk properly.

(Shrinking can also be done offline using **guestfish** (1))

After shrinking PVs and filesystems, shut down the guest, and proceed with steps 3 and 4 above to allocate a new disk image.

Then run virt-resize with any of the −−*shrink* and/or −−*resize* options.

IGNORING OR DELETING PARTITIONS

virt-resize also gives a convenient way to ignore or delete partitions when copying from the input disk to the output disk. Ignoring a partition speeds up the copy where you don't care about the existing contents of a partition. Deleting a partition removes it completely, but note that it also renumbers any partitions after the one which is deleted, which can leave some guests unbootable.

QCOW2 AND NON-SPARSE RAW FORMATS

If the input disk is in qcow2 format, then you may prefer that the output is in qcow2 format as well. Alternately, virt-resize can convert the format on the fly. The output format is simply determined by the format of the empty output container that you provide. Thus to create qcow2 output, use:

```
 qemu−img create −f qcow2 −o preallocation=metadata outdisk [size]
```

instead of the truncate command.

Similarly, to get non-sparse raw output use:

```
 fallocate −l size outdisk
```

(on older systems that don't have the **fallocate** (1) command use `dd if=/dev/zero of=outdisk bs=1M count=..`)

LOGICAL PARTITIONS

Logical partitions (a.k.a. */dev/sda5+* on disks using DOS partition tables) cannot be resized.

To understand what is going on, firstly one of the four partitions */dev/sda1−4* will have MBR partition type `05` or `0f`. This is called the**extended partition**. Use**virt−filesystems** (1) to see the MBR partition type.

Logical partitions live inside the extended partition.

The extended partition can be expanded, but not shrunk (unless you force it, which is not advisable). When the extended partition is copied across, all the logical partitions contained inside are copied over implicitly.

Virt-resize does not look inside the extended partition, so it copies the logical partitions blindly.

You cannot specify a logical partition (*/dev/sda5+*) at all on the command line. Doing so will give an error.

OPTIONS

−−help

Display help.

−−align−first auto
−−align−first never
−−align−first always

Align the first partition for improved performance (see also the *−−alignment* option).

The default is *−−align−first auto* which only aligns the first partition if it is safe to do so. That is, only when we know how to fix the bootloader automatically, and at the moment that can only be done for Windows guests.

−−align−first never means we never move the first partition. This is the safest option. Try this if the guest does not boot after resizing.

−−align−first always means we always align the first partition (if it needs to be aligned). For some guests this will break the bootloader, making the guest unbootable.

−−alignment N

Set the alignment of partitions to `N` sectors. The default in virt-resize < 1.13.19 was 64 sectors, and after that is 128 sectors.

Assuming 512 byte sector size inside the guest, here are some suitable values for this:

−−alignment 1 (512 bytes)

The partitions would be packed together as closely as possible, but would be completely unaligned. In some cases this can cause very poor performance. See **virt−alignment−scan**(1) for further details.

−−alignment 8 (4K)

This would be the minimum acceptable alignment for reasonable performance on modern hosts.

−−alignment 128 (64K)

This alignment provides good performance when the host is using high end network storage.

−−alignment 2048 (1M)

This is the standard alignment used by all newly installed guests since around 2008.

−−colors
−−colours

Use ANSI colour sequences to colourize messages. This is the default when the output is a tty. If the output of the program is redirected to a file, ANSI colour sequences are disabled unless you use this option.

−d
−−debug

(Deprecated: use *−v* option instead)

Enable debugging messages.

−−delete PART

Delete the named partition. It would be more accurate to describe this as ''don't copy it over'', since virt-resize doesn't do in-place changes and the original disk image is left intact.

Note that when you delete a partition, then anything contained in the partition is also deleted. Furthermore, this causes any partitions that come after to be *renumbered*, which can easily make your guest unbootable.

You can give this option multiple times.

−−expand PART

Expand the named partition so it uses up all extra space (space left over after any other resize changes that you request have been done).

If virt-resize knows how, it will expand the direct content of the partition. For example, if the partition is an LVM PV, it will expand the PV to fit (like calling **pvresize**(8)). Virt-resize leaves any other content it doesn't know about alone.

Currently virt-resize can resize:

- ext2, ext3 and ext4 filesystems.
- NTFS filesystems, if libguestfs was compiled with support for NTFS.

  The filesystem must have been shut down consistently last time it was used. Additionally, **ntfsresize**(8) marks the resized filesystem as requiring a consistency check, so at the first boot after resizing Windows will check the disk.
- LVM PVs (physical volumes). virt-resize does not usually resize anything inside the PV, but see the *−−LV−expand* option. The user could also resize LVs as desired after boot.
- Btrfs filesystems, if libguestfs was compiled with support for btrfs.
- XFS filesystems, if libguestfs was compiled with support for XFS.
- Linux swap partitions.

  Please note that libguestfs *destroys* the existing swap content by recreating it with `mkswap`, so this should not be used when the guest is suspended.
- f2fs filesystems, if libguestfs was compiled with support for f2fs.

Note that you cannot use *−−expand* and *−−shrink* together.

−−format raw

Specify the format of the input disk image. If this flag is not given then it is auto-detected from the image itself.

If working with untrusted raw-format guest disk images, you should ensure the format is always specified.

Note that this option *does not* affect the output format. See "QCOW2 AND NON-SPARSE RAW FORMATS".

−−ignore PART

Ignore the named partition. Effectively this means the partition is allocated on the destination disk, but the content is not copied across from the source disk. The content of the partition will be blank (all zero bytes).

You can give this option multiple times.

−−LV−expand LOGVOL

This takes the logical volume and, as a final step, expands it to fill all the space available in its volume group. A typical usage, assuming a Linux guest with a single PV */dev/sda2* and a root device called */dev/vg_guest/lv_root* would be:

```
 virt−resize indisk outdisk \
   −−expand /dev/sda2 −−LV−expand /dev/vg_guest/lv_root
```

This would first expand the partition (and PV), and then expand the root device to fill the extra space in the PV.

The contents of the LV are also resized if virt-resize knows how to do that. You can stop virt-resize from trying to expand the content by using the option *−−no−expand−content*.

Use **virt−filesystems**(1) to list the filesystems in the guest.

You can give this option multiple times, *but* it doesn't make sense to do this unless the logical volumes you specify are all in different volume groups.

−−machine−readable
−−machine−readable=format

This option is used to make the output more machine friendly when being parsed by other programs. See "MACHINE READABLE OUTPUT" below.

−n
−−dry−run

Print a summary of what would be done, but don't do anything.

−−no−copy−boot−loader

By default, virt-resize copies over some sectors at the start of the disk (up to the beginning of the first partition). Commonly these sectors contain the Master Boot Record (MBR) and the boot loader, and are required in order for the guest to boot correctly.

If you specify this flag, then this initial copy is not done. You may need to reinstall the boot loader in this case.

−−no−extra−partition

By default, virt-resize creates an extra partition if there is any extra, unused space after all resizing has happened. Use this option to prevent the extra partition from being created. If you do this then the extra space will be inaccessible until you run fdisk, parted, or some other partitioning tool in the guest.

Note that if the surplus space is smaller than 10 MB, no extra partition will be created.

−−no−expand−content

By default, virt-resize will try to expand the direct contents of partitions, if it knows how (see *−−expand* option above).

If you give the *−−no−expand−content* option then virt-resize will not attempt this.

−−no−sparse

Turn off sparse copying. See "SPARSE COPYING" below.

−−ntfsresize−force

Pass the *−−force* option to **ntfsresize**(8), allowing resizing even if the NTFS disk is marked as needing a consistency check. You have to use this option if you want to resize a Windows guest multiple times without booting into Windows between each resize.

−−output−format raw

Specify the format of the output disk image. If this flag is not given then it is auto-detected from the image itself.

If working with untrusted raw-format guest disk images, you should ensure the format is always specified.

Note that this option *does not create* the output format. This option just tells libguestfs what it is so it doesn't try to guess it. You still need to create the output disk with the right format. See "QCOW2 AND NON-SPARSE RAW FORMATS".

−q
−−quiet

Don't print the summary.

−−resize PART=SIZE

Resize the named partition (expanding or shrinking it) so that it has the given size.

`SIZE` can be expressed as an absolute number followed by b/K/M/G to mean bytes, Kilobytes, Megabytes, or Gigabytes; or as a percentage of the current size; or as a relative number or percentage. For example:

```
 --resize /dev/sda2=10G

 --resize /dev/sda4=90%

 --resize /dev/sda2=+1G

 --resize /dev/sda2=-200M

 --resize /dev/sda1=+128K

 --resize /dev/sda1=+10%

 --resize /dev/sda1=-10%
```

You can increase the size of any partition. Virt-resize will expand the direct content of the partition if it knows how (see *−−expand* above).

You can only *decrease* the size of partitions that contain filesystems or PVs which have already been shrunk. Virt-resize will check this has been done before proceeding, or else will print an error (see also *−−resize−force*).

You can give this option multiple times.

−−resize−force PART=SIZE
This is the same as *−−resize* except that it will let you decrease the size of any partition. Generally this means you will lose any data which was at the end of the partition you shrink, but you may not care about that (eg. if shrinking an unused partition, or if you can easily recreate it such as a swap partition).

See also the *−−ignore* option.

−−shrink PART
Shrink the named partition until the overall disk image fits in the destination. The named partition **must** contain a filesystem or PV which has already been shrunk using another tool (eg. **guestfish** (1) or other online tools). Virt-resize will check this and give an error if it has not been done.

The amount by which the overall disk must be shrunk (after carrying out all other operations requested by the user) is called the ''deficit''. For example, a straight copy (assume no other operations) from a 5GB disk image to a 4GB disk image results in a 1GB deficit. In this case, virt-resize would give an error unless the user specified a partition to shrink and that partition had more than a gigabyte of free space.

Note that you cannot use *−−expand* and *−−shrink* together.

−−unknown−filesystems ignore
−−unknown−filesystems warn
−−unknown−filesystems error
Configure the behaviour of virt-resize when asking to expand a filesystem, and neither libguestfs has the support it, nor virt-resize knows how to expand the content of the filesystem.

−−unknown−filesystems ignore will cause virt-resize to silently ignore such filesystems, and nothing is printed about them.

−−unknown−filesystems warn (the default behaviour) will cause virt-resize to warn for each of the filesystem that cannot be expanded, but still continuing to resize the disk.

−−unknown−filesystems error will cause virt-resize to error out at the first filesystem that cannot be expanded.

See also ''unknown/unavailable method for expanding the TYPE filesystem on DEVICE/LV''.

−v
−−verbose
Enable debugging messages.

−V
−−version
Display version number and exit.

−x Enable tracing of libguestfs API calls.

MACHINE READABLE OUTPUT

The *−−machine−readable* option can be used to make the output more machine friendly, which is useful when calling virt-resize from other programs, GUIs etc.

There are two ways to use this option.

Firstly use the option on its own to query the capabilities of the virt-resize binary. Typical output looks like this:

```
$ virt−resize −−machine−readable
virt−resize
ntfsresize−force
32bitok
ntfs
btrfs
```

A list of features is printed, one per line, and the program exits with status 0.

Secondly use the option in conjunction with other options to make the regular program output more machine friendly.

At the moment this means:

1. Progress bar messages can be parsed from stdout by looking for this regular expression:

   ```
   ^[0−9]+/[0−9]+$
   ```

2. The calling program should treat messages sent to stdout (except for progress bar messages) as status messages. They can be logged and/or displayed to the user.

3. The calling program should treat messages sent to stderr as error messages. In addition, virt-resize exits with a non-zero status code if there was a fatal error.

Versions of the program prior to 1.13.9 did not support the *−−machine−readable* option and will return an error.

It is possible to specify a format string for controlling the output; see ‘‘ADVANCED MACHINE READABLE OUTPUT’’ in **guestfs** (3).

NOTES

“Partition 1 does not end on cylinder boundary.”

Virt-resize aligns partitions to multiples of 128 sectors (see the *−−alignment* parameter). Usually this means the partitions will not be aligned to the ancient CHS geometry. However CHS geometry is meaningless for disks manufactured since the early 1990s, and doubly so for virtual hard drives. Alignment of partitions to cylinders is not required by any modern operating system.

GUEST BOOT STUCK AT “GRUB”

If a Linux guest does not boot after resizing, and the boot is stuck after printing `GRUB` on the console, try reinstalling grub.

```
guestfish −i −a newdisk
><fs> cat /boot/grub/device.map
# check the contents of this file are sensible or
# edit the file if necessary
><fs> grub−install / /dev/vda
><fs> exit
```

For more flexible guest reconfiguration, including if you need to specify other parameters to grub-install, use **virt−rescue** (1).

RESIZING WINDOWS BOOT PARTITIONS

In Windows Vista and later versions, Microsoft switched to using a separate boot partition. In these VMs, typically */dev/sda1* is the boot partition and */dev/sda2* is the main (C:) drive. Resizing the first (boot) partition causes the bootloader to fail with `0xC0000225` error. Resizing the second partition (ie. C: drive) should work.

WINDOWS CHKDSK

Windows disks which use NTFS must be consistent before virt-resize can be used. If the ntfsresize operation fails, try booting the original VM and running `chkdsk /f` on all NTFS partitions, then shut down the VM cleanly. For further information see: https://bugzilla.redhat.com/show_bug.cgi?id=975753

After resize Windows may initiate a lengthy ''chkdsk'' on first boot if NTFS partitions have been expanded. This is just a safety check and (unless it find errors) is nothing to worry about.

WINDOWS UNMOUNTABLE_BOOT_VOLUME BSOD

After sysprepping a Windows guest and then resizing it with virt-resize, you may see the guest fail to boot with an `UNMOUNTABLE_BOOT_VOLUME` BSOD. This error is caused by having `ExtendOemPartition=1` in the sysprep.inf file. Removing this line before sysprepping should fix the problem.

WINDOWS 8

Windows 8 ''fast startup'' can prevent virt-resize from resizing NTFS partitions. See ''WINDOWS HIBERNATION AND WINDOWS 8 FAST STARTUP'' in **guestfs** (3).

SPARSE COPYING

You should create a fresh, zeroed target disk image for virt-resize to use.

Virt-resize by default performs sparse copying. This means that it does not copy blocks from the source disk which are all zeroes. This improves speed and efficiency, but will produce incorrect results if the target disk image contains unzeroed data.

The main time this can be a problem is if the target is a host partition (eg. `virt−resizesource.img/dev/sda4`) because the usual partitioning tools tend to leave whatever data happened to be on the disk before.

If you have to reuse a target which contains data already, you should use the *−−no−sparse* option. Note this can be much slower.

''unknown/unavailable method for expanding the TYPE filesystem on DEVICE/LV''

Virt-resize was asked to expand a partition or a logical volume containing a filesystem with the type `TYPE`, but there is no available nor known expanding method for that filesystem.

This may be due to either of the following:

1. There corresponding filesystem is not available in libguestfs, because there is no proper package in the host with utilities for it. This is usually the case for `btrfs`, `ntfs`, `xfs`, and `f2fs` filesystems.

   Check the results of:

   ```
   virt−resize −−machine−readable
   guestfish −a /dev/null run : available
   guestfish −a /dev/null run : filesystem_available TYPE
   ```

   In this case, it is enough to install the proper packages adding support for them. For example, `libguestfs-xfs` on Red Hat Enterprise Linux, CentOS, Debian, Ubuntu, and distributions derived from them, for supporting the `xfs` filesystem.

2. Virt-resize has no support for expanding that type of filesystem.

   In this case, there's nothing that can be done to let virt-resize expand that type of filesystem.

In both cases, virt-resize will not expand the mentioned filesystem; the result (unless *−−unknown−filesystems error* is specified) is that the partitions containing such filesystems will be actually bigger as requested, but the filesystems will still be usable at their older sizes.

ALTERNATIVE TOOLS

There are several proprietary tools for resizing partitions. We won't mention any here.

parted (8) and its graphical shell gparted can do some types of resizing operations on disk images. They can resize and move partitions, but I don't think they can do anything with the contents, and they certainly don't understand LVM.

guestfish (1) can do everything that virt-resize can do and a lot more, but at a much lower level. You will probably end up hand-calculating sector offsets, which is something that virt-resize was designed to avoid. If you want to see the guestfish-equivalent commands that virt-resize runs, use the *−−debug* flag.

dracut (8) includes a module called `dracut−modules−growroot` which can be used to grow the root partition when the guest first boots up. There is documentation for this module in an associated README file.

EXIT STATUS

This program returns 0 if successful, or non-zero if there was an error.

SEE ALSO

virt−filesystems (1), **virt−df** (1), **guestfs** (3), **guestfish** (1), **lvm** (8), **pvresize** (8), **lvresize** (8), **resize2fs** (8), **ntfsresize** (8), **btrfs** (8), **xfs_growfs** (8), **resize.f2fs** (8), **virsh** (1), **parted** (8), **truncate** (1), **fallocate** (1), **grub** (8), **grub−install** (8), **virt−rescue** (1), **virt−sparsify** (1), **virt−alignment−scan** (1), http://libguestfs.org/.

AUTHOR

Richard W.M. Jones http://people.redhat.com/˜rjones/

COPYRIGHT

LICENSE

BUGS

To get a list of bugs against libguestfs, use this link: https://bugzilla.redhat.com/buglist.cgi?component=libguestfs&product=Virtualization+Tools

To report a new bug against libguestfs, use this link: https://bugzilla.redhat.com/enter_bug.cgi?component=libguestfs&product=Virtualization+Tools

When reporting a bug, please supply:

- The version of libguestfs.
- Where you got libguestfs (eg. which Linux distro, compiled from source, etc)

- Describe the bug accurately and give a way to reproduce it.
- Run **libguestfs−test−tool** (1) and paste the **complete, unedited** output into the bug report.