NAME

virt−dib − Run diskimage−builder elements

SYNOPSIS

```
virt−dib −B DIB−LIB [options] elements...
```

DESCRIPTION

Virt-dib is a tool for using the elements of `diskimage−builder` to build a new disk image, generate new ramdisks, etc.

Virt-dib is intended as safe replacement for `diskimage−builder` and its `ramdisk−image−create` mode, see ‘‘COMPARISON WITH DISKIMAGE-BUILDER’’ for a quick comparison with usage of `diskimage−builder`.

`diskimage−builder` is part of the TripleO OpenStack project: https://wiki.openstack.org/wiki/TripleO.

EXAMPLES

Build simple images of distributions

```
virt−dib \
  −B /path/to/diskimage−builder/lib \
  −p /path/to/diskimage−builder/elements \
  −−envvar DIB_RELEASE=jessie \
  −−name debian−jessie \
  debian vm
```

This builds a Debian Jessie (8.x) disk image, suitable for running as virtual machine, saved as *debian−jessie.qcow2*.

Build ramdisks

```
virt−dib \
  −B /path/to/diskimage−builder/lib \
  −p /path/to/diskimage−builder/elements \
  −−ramdisk \
  −−name ramdisk \
  ubuntu deploy−ironic
```

This builds a ramdisk for the Ironic OpenStack component based on the Ubuntu distribution.

OPTIONS

−−help
Display help.

−B PATH
Set the path to the library directory of `diskimage−builder`. This is usually the *lib* subdirectory in the sources and when installed, and */usr/share/diskimage−builder/lib* when installed in */usr*.

This parameter is **mandatory**, as virt-dib needs to provide it for the elements (as some of them might use scripts in it). Virt-dib itself does not make use of the library directory.

−−arch ARCHITECTURE
Use the specified architecture for the output image. The default value is the same as the host running virt-dib.

Right now this option does nothing more than setting the `ARCH` environment variable for the elements, and it’s up to them to produce an image for the requested architecture.

−−checksum
Generate checksum files for the generated image. The supported checksums are MD5, and SHA256.

−−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.

--debug LEVEL
Set the debug level to `LEVEL`, which is a non-negative integer number. The default is `0`.

This debug level is different than what *-x* and *-v* set, and it increases the debugging information printed out. Specifically, this sets the `DIB_DEBUG_TRACE`, and any value > `0` enables tracing in the scripts executed.

--docker-target TARGET
Set the repository and tag for docker.

This is used only when the formats include `docker`, and it is required in that case.

--drive DISK
Add the specified disk to be used as helper drive where to cache files of the elements, like disk images, distribution packages, etc.

See ''HELPER DRIVE''.

--drive-format raw
--drive-format qcow2
Specify the format of the helper drive. If this flag is not given then it is auto-detected from the drive itself.

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

This option is used only if *--drive* is specified.

See ''HELPER DRIVE''.

-p PATH
--element-path PATH
Add a new path with elements. Paths are used in the same order as the *-p* parameters appear, so a path specified first is looked first, and so on.

Obviously, it is recommended to add the path to the own elements of `diskimage-builder`, as most of the other elements will rely on them.

--extra-packages PACKAGE,...
Install additional packages in the image being built.

This relies on the `install-packages` binary provided by the package management elements.

This option can be specified multiple times, each time with multiple packages separated by comma.

--envvar VARIABLE
--envvar VARIABLE=VALUE
Carry or set an environment variable for the elements.

See ''ENVIRONMENT VARIABLES'' below for more information on the interaction and usage of environment variables.

This option can be used in two ways:

--envvar VARIABLE
Carry the environment variable `VARIABLE`. If it is not set, nothing is exported to the elements.

--envvar VARIABLE=VALUE
Set the environment variable `VARIABLE` with value `VALUE` for the elements, regardless whether an environment variable with the same name exists.

This can be useful to pass environment variable without exporting them in the environment where virt-dib runs.

--exclude-element ELEMENT
Ignore the specified element.

--exclude-script SCRIPT
Ignore any element script named `SCRIPT`, whichever element it is in.

This can be useful in case some script does not run well with virt-dib, for example when they really need `diskimage-builder`'s environment.

--formats FORMAT,...
Set the list of output formats, separating them with comma.

Supported formats are:

`docker`
Import the image to docker, running **docker import**. The target for the image **must** be specified using *--docker-target*.

Please note this operation usually requires the docker service to be enabled, otherwise it will fail. Furthermore, **docker** is run using **sudo** (8), so make sure the user has the permissions to run at least **docker**.

`qcow2` (enabled by default)
QEMU's qcow2. This output format requires the`qemu-img` tool.

`raw`
Raw disk format.

`squashfs`
An squashfs filesystem, compressed with XZ. This output format requires the`squashfs` feature; see also ''AVAILABILITY'' in **guestfs** (3).

`tar`
An uncompressed tarball.

`tgz`
A tarball compressed with gzip.

`vhd`
`Virtual Hard Disk`disk image. This output format requires the `vhd-util` tool.

Please note that the version of `vhd-util` tool needs to be patched to support the `convert` subcommand, and to be bootable. The patch is available here: https://github.com/emonty/vhd-util/blob/master/debian/patches/citrix.

--fs-type FILESYSTEM
Set the filesystem type to use for the root filesystem. The default is `ext4`.

See also ''guestfs_filesystem_available'' in **guestfs** (3).

--image-cache DIRECTORY
Set the path in the host where cache the resources used by the elements of the `extra-data.d` phase. The default is *~/.cache/image-create*.

Please note that most of the resources fetched in phases other than `extra-data.d` will be cached in the helper drive specified with *--drive*; see also ''HELPER DRIVE''.

--install-type TYPE
Specify the default installation type. Defaults to `source`.

Set to `package` to use package based installations by default.

--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.

-m MB
--memsize MB
Change the amount of memory allocated to the appliance. Increase this if you find that the virt-dib execution runs out of memory.

The default can be found with this command:

```
guestfish get-memsize
```

--mkfs-options `OPTION STRING`
Add the specified options to **mkfs**(1), to be able to fine-tune the root filesystem creation; the options are passed to the driver of **mfks**(1), and not to **mfks**(1) itself. Note that *--fs-type* is used to change the filesystem type.

You should use *--mkfs-options* at most once. To pass multiple options, separate them with space, eg:

```
virt-dib ... --mkfs-options '-O someopt -I foo'
```

--network
--no-network
Enable or disable network access from the guest during the installation.

Enabled is the default. Use*--no-network* to disable access.

The network only allows outgoing connections and has other minor limitations. See ‘‘NETWORK’’ in **virt-rescue**(1).

This does not affect whether the guest can access the network once it has been booted, because that is controlled by your hypervisor or cloud environment and has nothing to do with virt-dib.

If you use *--no-network*, then the environment variable `DIB_OFFLINE` is set to `1`, signaling the elements that they should use only cached resources when available. Note also that, unlike with `diskimage-builder` where elements may still be able to access to the network even with `DIB_OFFLINE=`, under virt-dib network will not be accessible at all.

--name NAME
Set the name of the output image file. The default is `image`.

According to the chosen name, there will be the following in the current directory:

$NAME.ext
For each output format, a file named after the output image with the extension depending on the format; for example: *$NAME.qcow2*, *$NAME.raw*, etc.

Not applicable in ramdisk mode, see ‘‘RAMDISK BUILDING’’.

$NAME.d
A directory containing any files created by the elements, for example *dib-manifests* directory (created by the `manifests` element), ramdisks and kernels in ramdisk mode, and so on.

$NAME.ext.checksum
When *--checksum* is specified, there will be files for each supported checksum type; for example: *$NAME.ext.md5*, *$NAME.ext.sha256*, etc.

Not applicable in ramdisk mode, see ‘‘RAMDISK BUILDING’’.

--no-delete-on-failure
Don’t delete the output files on failure to build. You can use this to debug failures to run scripts.

The default is to delete the output files if virt-dib fails (or, for example, some script that it runs fails).

--python PYTHON
Specify a different Python interpreter to use. Parts of `diskimage-builder` are implemented in Python, and thus an interpreter is needed.

`PYTHON` can either be an executable filename (e.g. *python2*, which is then searched in `$PATH`), or a full path (e.g. */usr/bin/python2*). If not specified, the default value is *python*.

-q
--quiet
Don't print ordinary progress messages.

--qemu-img-options option[,option,...]
Pass *--qemu-img-options* option(s) to the **qemu-img** (1) command to fine-tune the output format. Options available depend on the output format (see *--formats*) and the installed version of the qemu-img program.

You should use *--qemu-img-options* at most once. To pass multiple options, separate them with commas, eg:

```
virt-dib ... --qemu-img-options cluster_size=512,preallocation=metadata ...
```

--ramdisk
Set the ramdisk building mode.

See ''RAMDISK BUILDING''.

--ramdisk-element NAME
Set the name for the additional element added in ramdisk building mode. The default is `ramdisk`.

See ''RAMDISK BUILDING''.

--root-label LABEL
Set the label for the root filesystem in the created image.

Please note that some filesystems have different restrictions on the length of their labels; for example, on `ext2/3/4` filesystems labels cannot be longer than 16 characters, while on `xfs` they have at most 12 characters.

The default depends on the actual filesystem for the root partition (see *--fs-type*): on `xfs` is `img-rootfs`, while `cloudimg-rootfs` on any other filesystem.

--size SIZE
Select the size of the output disk, where the size can be specified using common names such as `32G` (32 gigabytes) etc. The default size is `5G`.

To specify size in bytes, the number must be followed by the lowercase letter *b*, eg: `--size10737418240b`.

See also **virt-resize** (1) for resizing partitions of an existing disk image.

--skip-base
Skip the inclusion of the `base` element.

--smp N
Enable N ≥ 2 virtual CPUs for scripts to use.

-u Do not compress resulting qcow2 images. The default is to compress them.

-v
--verbose
Enable debugging messages.

-V
--version
Display version number and exit.

-x Enable tracing of libguestfs API calls.

ENVIRONMENT VARIABLES

Unlike with `diskimage-builder`, the environment of the host is **not** inherited in the appliance when running most of the elements (i.e. all except the ones in the `extra-data.d` phase).

To set environment for the elements being run, it is necessary to tell virt-dib to use them, with the option *--envvar*. Such option allows to selectively export environment variables when running the elements, and it is the preferred way to pass environment variables to the elements.

To recap: if you want the environment variable `MYVAR` (and its content) to be available to the elements, you can do either

```
export MYVAR   # whichever is its value
virt-dib ... --envvar MYVAR ...
```

or

```
virt-dib ... --envvar MYVAR=value_of_it ...
```

HELPER DRIVE

Virt-dib runs most of the element in its own appliance, and thus not on the host. Because of this, there is no possibility for elements to cache resources directly on the host.

To solve this issue, virt-dib allows the usage of an helper drive where to store cached resources, like disk images, distribution packages, etc. While this means that there is a smaller space available for caching, at least it allows to limit the space on the host for caches, without assuming that elements will do that by themselves.

Currently this disk is either required to have a single partition on it, or the first partition on it will be used. A disk with the latter configuration can be easily created with **guestfish** (1) like the following:

```
guestfish -N filename.img=fs:ext4:10G exit
```

The above will create a disk image called *filename.img*, 10G big, with a single partition of type ext4; see "PREPARED DISK IMAGES" in **guestfish** (1).

It is recommended for it to be ≥ 10G or even more, as elements will cache disk images, distribution packages, etc. As with any disk image, the helper disk can be easily resized using **virt-resize** (1) if more space in it is needed.

The drive can be accessed like any other disk image, for example using other tools of libguestfs such as **guestfish** (1):

```
guestfish -a filename.img -m /dev/sda1
```

If no helper drive is specified with *--drive*, all the resources cached during a virt-dib run will be discarded.

RESOURCES INSIDE THE DRIVE

Inside the helper drive, it is possible to find the following resources:

/home
: This directory is set as `HOME` environment variable during the build. It contains mostly the image cache (saved as */home/.cache/image-create*), and whichever other resource is cached in the home directory of the user running the various tools.

/virt-dib-.log*
: These are the logs of the elements being run within the libguestfs appliance, which means all the phases except `extra-data.d`.

RAMDISK BUILDING

Virt-dib can emulate also `ramdisk-image-create`, which is a secondary operation mode of `diskimage-builder`. Instead of being a different tool name, virt-dib provides easy access to this mode using the *--ramdisk* switch.

In this mode:

- there is an additional ramdisk element added (see *--ramdisk-element*)
- no image is produced (so *--formats* is ignored)
- *$NAME.d* (see *--name*) will contain initrd, kernel, etc

TEMPORARY DIRECTORY

Virt-dib uses the standard temporary directory used by libguestfs, see ''ENVIRONMENT VARIABLES'' in **guestfs** (3).

By default this location is */tmp* (default value for `TMPDIR`), which on some systems may be on a tmpfs filesystem, and thus defaulting to a maximum size of *half* of physical RAM. If virt-dib exceeds this, it may hang or exit early with an error. The solution is to point`TMPDIR` to a permanent location used as temporary location, for example:

```
mkdir local-tmp
env TMPDIR=$PWD/local-tmp virt-dib ...
rm -rf local-tmp
```

EXTRA DEPENDENCIES

Because of virt-dib runs most of the elements in its own appliance, all the tools and libraries used by elements running outside the guest (typically `root.d`, `block-device.d`, and `cleanup.d`) need to be present in the appliance as well. In case they are not, scripts will fail typically with a `command not found` error.

For tools and libraries packaged by the distribution, the easy solution is to tell libguestfs to include additional packages in the appliance. This is doable by e.g. creating a new file with the additional packages:

```
# echo wget > /usr/lib64/guestfs/supermin.d/dib-my-extra
```

The actual path to the *supermin.d* directory depends on the distribution; additional files can list more packages, each in its own line. For more details, see **supermin** (1).

COMPARISON WITH DISKIMAGE-BUILDER

Virt-dib is intended as safe replacement for `diskimage-builder` and its `ramdisk-image-create` mode; the user-notable differences consist in:

- the command line arguments; some of the arguments are the same as available in `diskimage-builder`, while some have different names:

disk-image-create	virt-dib
-a ARCH	--arch ARCH
--image-size SIZE	--size SIZE
--max-online-resize SIZE	doable using --mkfs-options
-n	--skip-base
-o IMAGENAME	--name IMAGENAME
-p PACKAGE(S)	--extra-packages PACKAGE(S)
-t FORMAT(S)	--formats FORMAT(S)
-x	--debug 1
-x -x	--debug 2
-x -x [-x ...]	--debug 3/4/etc

- the location of non-image output files (like ramdisks and kernels)
- the way some of the cached resources are saved: using an helper drive, not directly on the disk where virt-dib is run
- the need to specify a target size for the output disk, as opposed to `diskimage-builder` calculating an optimal one
- the handling of environment variables, see ''ENVIRONMENT VARIABLES''.

Furthermore, other than the libguestfs own environment variables (see ‘‘ENVIRONMENT VARIABLES’’ in **guestfs** (3)), virt-dib does not read any other environment variable: this means that all the options and behaviour changes are specified solely using command line arguments

- extra tools needed on some out-of-chroot phases need to be available in the appliance, see ‘‘EXTRA DEPENDENCIES’’.

Elements themselves should notice no difference in they way they are run; behaviour differences may due to wrong assumptions in elements, or not correct virt-dib emulation.

Known issues at the moment:

- (none)

MACHINE READABLE OUTPUT

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

Use the option on its own to query the capabilities of the virt-dib binary. Typical output looks like this:

```
$ virt-dib --machine-readable
virt-dib
output:qcow2
output:tar
output:raw
output:vhd
```

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

The `output:` features refer to the output formats (`--formats` command line option) supported by this binary.

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

TESTING

Virt-dib has been tested with `diskimage-builder` (and its elements) ≥ 0.1.43; from time to time also with `tripleo-image-elements` and `sahara-image-elements`.

Previous versions may work, but it is not guaranteed.

EXIT STATUS

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

SEE ALSO

guestfs (3), **guestfish** (1), **virt-resize** (1), http://libguestfs.org/.

AUTHOR

Pino Toscano (`ptoscano at redhat dot com`)

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.