Inspecting and modifying virtual machines with Red Hat Enterprise Linux 6.1

Richard W.M. Jones
Senior Software Engineer
Red Hat
`rjones@redhat.com`

Wednesday May 4th 2011

1 Introduction

libguestfs[1] is a library, scripting language and a set of tools that let you look into virtual machines and make changes to them without needing to boot them up. "Inspection" is the process of getting a formal description of what's in a virtual machine, how it is configured, what software is installed, the contents of the filesystem and Windows Registry and so on. "Modification" here means making repeatable changes to these guests, to their configuration files, filesystems and Registries, from programs and scripts.

Big advances have happened in libguestfs since an early version was added to RHEL 6.0, and most of those changes will appear in RHEL 6.1. For a start, RHEL 6.1 libguestfs is 4 or 5 times faster, so if you tried libguestfs in RHEL 6.0 and were disappointed with the performance, then try RHEL 6.1. Hundreds of individual features have been added, and we're only going to be able to show a handful of the most important features in the talk today.

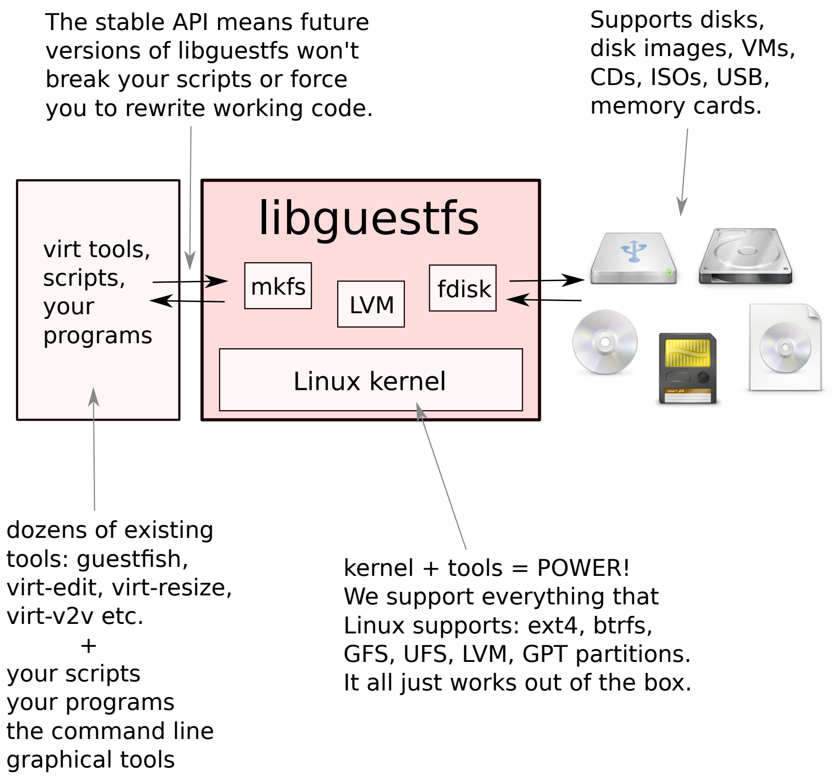
libguestfs is now the basis for several important projects inside and outside Red Hat, including the Boxgrinder cloud image builder, at least one proprietary ISP/cloud VM deployment system, and virt-p2v/virt-v2v which my colleague Matthew Booth is going to talk about after me.

[1]`http://libguestfs.org/`

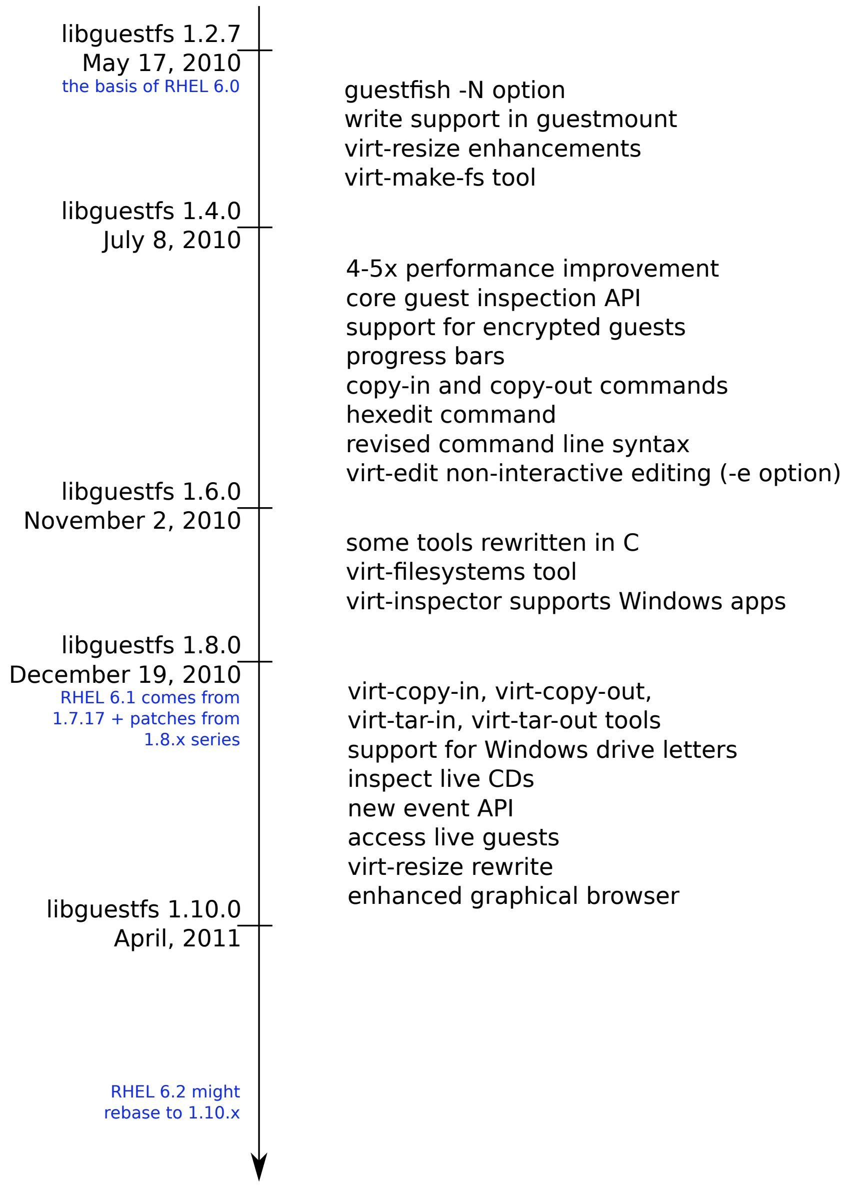
Red Hat thinks that managing these previously "opaque" disk images and virtual machines is very important, and we want our customers to have the best possible open source tools available. The libguestfs project was started over two years ago and has had one or two full time developers working on it ever since then. Here are some stats from the project:

24	command line tools
171	pages in the manual
over 300	API calls
555	automated tests run on each release
2,885	git commits (about $3\frac{1}{2}$ commits per day, including weekends and holidays)
313,247	lines of code

2 What is libguestfs?


The stable API means future versions of libguestfs won't break your scripts or force you to rewrite working code.
Supports disks, disk images, VMs, CDs, ISOs, USB, memory cards.
virt tools, scripts, your programs
libguestfs
mkfs
LVM
fdisk
Linux kernel
dozens of existing tools: guestfish, virt-edit, virt-resize, virt-v2v etc. + your scripts your programs the command line graphical tools
kernel + tools = POWER! We support everything that Linux supports: ext4, btrfs, GFS, UFS, LVM, GPT partitions. It all just works out of the box.

3 Timeline


libguestfs 1.2.7
May 17, 2010
the basis of RHEL 6.0
guestfish -N option
write support in guestmount
virt-resize enhancements
virt-make-fs tool
libguestfs 1.4.0
July 8, 2010
4-5x performance improvement
core guest inspection API
support for encrypted guests
progress bars
copy-in and copy-out commands
hexedit command
revised command line syntax
virt-edit non-interactive editing (-e option)
libguestfs 1.6.0
November 2, 2010
some tools rewritten in C
virt-filesystems tool
virt-inspector supports Windows apps
libguestfs 1.8.0
December 19, 2010
RHEL 6.1 comes from
1.7.17 + patches from
1.8.x series
virt-copy-in, virt-copy-out,
virt-tar-in, virt-tar-out tools
support for Windows drive letters
inspect live CDs
new event API
access live guests
virt-resize rewrite
enhanced graphical browser
libguestfs 1.10.0
April, 2011
RHEL 6.2 might
rebase to 1.10.x

4 Introducing guestfish

guestfish[2] is a shell you can use to open up and modify disk images. You can just open up any libvirt guest or disk image by doing:

```
# guestfish --ro -i -d RHEL60

Welcome to guestfish, the libguestfs filesystem interactive shell for
editing virtual machine filesystems.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell

Operating system: Red Hat Enterprise Linux Server release 6.0 (Santiago)
/dev/vg_rhel6brewx64/lv_root mounted on /
/dev/vda1 mounted on /boot

><fs> ll /
total 138
dr-xr-xr-x.  26 root root  4096 Apr 11 09:49 .
drwxr-xr-x   24 root root  4096 Apr 11 17:13 ..
-rw-r--r--.   1 root root     0 Apr 11 09:49 .autofsck
drwx------.   3 root root  4096 Sep 17  2010 .dbus
dr-xr-xr-x.   2 root root  4096 Nov  6 15:21 bin
dr-xr-xr-x.   5 root root  1024 Sep 18  2010 boot
drwxr-xr-x.   2 root root  4096 Jul 14  2010 cgroup
drwxr-xr-x.   2 root root  4096 Sep 17  2010 dev
[etc]
```

A note about those options:

`--ro` This means open the disk read-only: you don't want to make any changes to it. Opening a disk which is in use (eg. used by a running VM) is *unsafe* unless you use this option.

`-i` This means "inspect" the disk image and mount up the filesystems as they would be mounted if the guest was running. You can leave

[2] `http://libguestfs.org/guestfish.1.html`

out this option and instead look for the filesystems yourself using the `list-filesystems` command.

`-d` This means open the named libvirt guest. You can get a list of libvirt guests by doing `virsh list --all`. You can use the `-a` option to open a disk image file or device directly.

There are hundreds of guestfish commands for reading and writing files, listing directories, creating partitions, extending logical volumes and so on. You can also use guestfish from shell scripts if you want to make repeatable scripted changes to guests. A few useful commands include:

`cat`	Display small text files.
`edit`	Edit a file.
`less`	Display longer files.
`ll`	List (long) directory.
`ls`	List directory.
`mkdir`	Make a directory.
`rm`	Remove a file.
`touch`	Touch a file.
`upload`	Upload a local file to the disk.
`write`	Create a file with content.

The best place to start is the guestfish man page:

```
$ man guestfish
```

or by reading the webpage `http://libguestfs.org/guestfish.1.html`

guestfish doesn't normally need root. The only time you need to run guestfish as root is if you need root in order to be able to access the disk images themselves. There are some better alternatives, such as adding users to the "disk" group.

5 Introducing virt-rescue

virt-rescue[3] is a good way to rescue virtual machines that don't boot, or just generally make ad hoc changes to virtual machines. It's like a rescue CD for virtual machines.

virt-rescue is a little different from guestfish in that you get an ordinary shell and ordinary tools. However unlike guestfish, virt-rescue cannot be used from shell scripts, so it's not useful if you want to make repeatable changes to lots of your guests.

You must not use virt-rescue on running VMs.

If you had a libvirt guest called "Fedora" then:

```
# virt-rescue -d Fedora
[lots of boot messages]

Welcome to virt-rescue, the libguestfs rescue shell.

Note: The contents of / are the rescue appliance.
You have to mount the guest's partitions under /sysroot
before you can examine them.

><rescue> lvs
  LV      VG        Attr   LSize Origin Snap%  Move Log Copy%  Convert
  lv_root vg_f13x64 -wi-a- 7.56g
  lv_swap vg_f13x64 -wi-a- 1.94g
><rescue> mount /dev/vg_f13x64/lv_root /sysroot/
[  107.912813] EXT4-fs (dm-0): mounted filesystem with ordered data mode.
Opts: (null)
><rescue> ls -l /sysroot/etc/fstab
-rw-r--r--. 1 root root 781 Sep 16  2010 /sysroot/etc/fstab
><rescue> vi /sysroot/etc/fstab
```

There is a lot more information about virt-rescue in the man page:

```
$ man virt-rescue
```

or you can read the manual online `http://libguestfs.org/virt-rescue.1.html`

[3]`http://libguestfs.org/virt-rescue.1.html`

6 Introducing the other virt-tools

In the following sections I will be demonstrating some of the other virt tools that come with RHEL 6.1. Here I'll provide a quick overview of the tools available.

guestfish	Interactive and scriptable shell.
guestmount	Mount filesystems from any guest or disk image on the host.
virt-cat	Display a file from a guest.
virt-copy-in	Copy files and directories into a guest.
virt-copy-out	Copy files and directories out of a guest.
virt-df	Display disk usage of a guest.
virt-edit	Edit a file in a guest.
virt-filesystems	Display the partitions, filesystems, logical volumes etc. in a guest.
virt-inspector	The old RHEL 6.0 virt-inspector program. Use virt-inspector2 instead.
virt-inspector2	Inspect a guest and produce a report detailing the operating system, version, applications installed and more.
virt-ls	List a directory in a guest.
virt-make-fs	Make a new filesystem.
virt-rescue	Rescue mode for guests.
virt-resize	Resize a guest.
virt-tar-in	Copy files from a tarball into a guest.
virt-tar-out	Copy files out of a guest into a tarball.
virt-win-reg	Display and edit the Windows Registry in a guest.

To get more information about any command, read the manual page. Type (for example):

```
$ man virt-cat
```

or see the upstream website: `http://libguestfs.org/`

7 Exercise: charting disk usage with virt-df

The virt-df utility[4] displays disk usage for virtual machines. Normally the output looks like the ordinary "df" command:

```
# virt-df -h
Filesystem                              Size       Used  Available  Use%
cooking:/dev/sda                        3.0G       1.5G       1.3G   52%
cooking:/dev/sdb                        128M        95M        26M   75%
database:/dev/sda                       3.0G       733M       2.1G   25%
database:/dev/sdb                       128M        95M        26M   75%
database:/dev/sdc                        49G        25G        22G   51%
```

However you can also get virt-df to produce comma-separated values (CSV) output which is useful for monitoring and tracking disk usage. CSV can be imported directly into many databases and spreadsheet programs.

On my production server I capture virt-df CSV output every day using a simple cron job `/etc/cron.daily/local-virt-df`:

```
#!/bin/bash -
date=$(date +%F)
virt-df --csv > /var/local/virt-df.$date
```

I then import these files into a spreadsheet which allows me to chart disk usage and look for trends. Figure 1 on page 13 charts a virtual machine over a five month period.

8 Exercise: using guestfish -N

In this exercise we will use the guestfish "-N" option to create a new disk image from scratch containing some files and directories. For the content I'm going to use a source tarball of libguestfs[5].

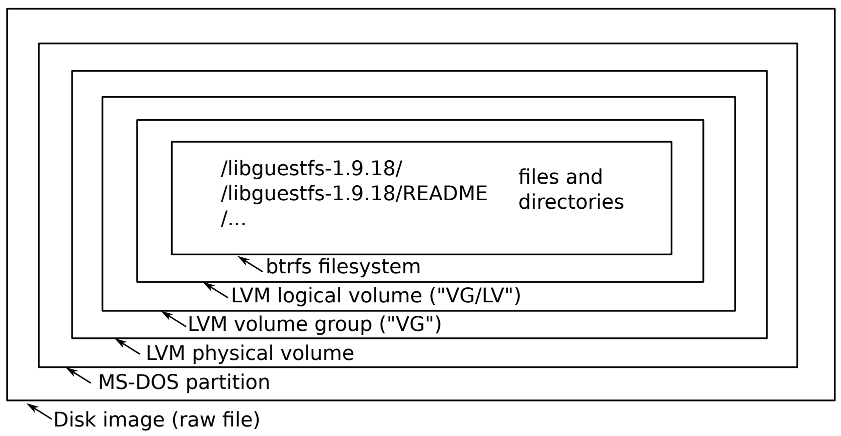
To make this exercise more exciting I'm going to specify that I want my files stored in an LVM logical volume inside the disk image, and I want to

[4] `http://libguestfs.org/virt-df.1.html`

[5] Source code for libguestfs is available from `http://libguestfs.org/download/` or for Red Hat subscribers from RHN.

format my filesystem using the smart new btrfs[6] filesystem. The files from the tarball are about 5 MB in size, so I'm going to choose a disk image size which is easily large enough to store them with plenty of space: 500 MB! It turns out that the minimum size for a btrfs filesystem is 256 MB, and both LVM and btrfs impose a large overhead.

In effect my disk image will be wrapped up in several layers as in this diagram:



The guestfish "-N" option below creates the complex nested filesystem structure[7]. Notice that you do not need to run this command as root –

[6]`https://secure.wikimedia.org/wikipedia/en/wiki/Btrfs`

[7]For more information about use of the "-N" option, type: `guestfish -N help`

creating disk images is something that everyone can do.

```
$ guestfish -N lvfs:/dev/VG/LV:btrfs:500M
><fs> list-filesystems
/dev/VG/LV: btrfs
><fs> mount-options "" /dev/VG/LV /      Mount the filesystem so we can write to it
><fs> df-h                              Notice that 96 MB has been lost!
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VG-LV     496M   56K  404M   1% /sysroot
><fs> tgz-in libguestfs-1.9.18.tar.gz /  Unpack the tarball into the new filesystem
><fs> ll /
total 8
dr-xr-xr-x  1 root root   34 Apr 12 14:08 .
drwxr-xr-x 24  500  500 4096 Apr 12 14:08 ..
drwxrwxr-x  1 root root 1076 Apr  9 22:25 libguestfs-1.9.18
><fs> exit

$ file test1.img
test1.img: x86 boot sector; partition 1: ID=0x83, starthead 1,
startsector 64, 1023873 sectors, code offset 0xb8
```

The output disk image is in `test1.img`. How do you prove that it contains a filesystem? One way is to open it again with guestfish:

```
$ guestfish -a test1.img -m /dev/VG/LV
```

Another way is to take this disk image and attach it to a virtual machine.

9 Exercise: find vulnerable versions of Firefox

In this exercise we will use virt-inspector[8] to find out if any vulnerable versions of Firefox[9] are installed in Windows guests. At the time of writing, any

[8]This example uses the Fedora virt-inspector program. In RHEL 6.1 this program is called `virt-inspector2` so you need to change any references to "virt-inspector" to "virt-inspector2". RHEL 6.1 ships with a known bug: it is not able to list 32 bit applications installed in a 64 bit Windows guest (using the WOW64 emulator). A fix for this bug will be included in RHEL 6.2. (RHBZ#692545)

[9]`https://www.mozilla.org/security/known-vulnerabilities/`

version of Firefox < 3.6.16 was vulnerable, so we'd like to scan our Windows guests to check this.

First run virt-inspector and have a look at the output:

```
# virt-inspector -d WindowsGuest
<operatingsystems>
  <operatingsystem>
    <name>windows</name>                                     it's a Windows guest
    <arch>i386</arch>                                                  it's 32 bit
    <product_name>Windows 7 Enterprise</product_name>
    <major_version>6</major_version>       ''6.1'' = Windows 7 -- blame Microsoft!
    <minor_version>1</minor_version>
    ...
    <applications>                               the list of applications starts here
      <application>
        <name>Mozilla Firefox (3.6.12)</name>
        <display_name>Mozilla Firefox (3.6.12)</display_name>
        <version>3.6.12 (en-GB)</version>
      ...
      </application>
    </applications>
  </operatingsystem>
</operatingsystems>
```

One way to extract and process XML documents is to use W3C standard XPath expressions. In this example I will use a short Python program with the libxml2 library to find vulnerable versions of Firefox:

```
#!/usr/bin/python
import libxml2, re, sys
from distutils import version
```

Read the XML piped from standard input

```
doc = libxml2.readFd (sys.stdin.fileno(), None, None, 0)
```

Use XPath to find all <application> nodes

```
ctx = doc.xpathNewContext()
res = ctx.xpathEval ("//application")
for node in res:
```

Use XPath to find the <name> and <version> within current <application> node

```
    ctx.setContextNode(node)
    name = ctx.xpathEval ("./name//text()")[0]
    ver = ctx.xpathEval ("./version//text()")[0]
```

Python StrictVersion lets me compare version numbers

```
    ver = version.StrictVersion (str(ver).split(’ ’)[0])
    if re.search ("Mozilla Firefox", str(name)) and \
            ver < version.StrictVersion ("3.6.16"):
        print "Vulnerable version of Firefox found (%s)!" % ver
```

Putting this together gives:

```
# virt-inspector -d WindowsGuest | ./vulnerable.py
Vulnerable version of Firefox found (3.6.12)!
```

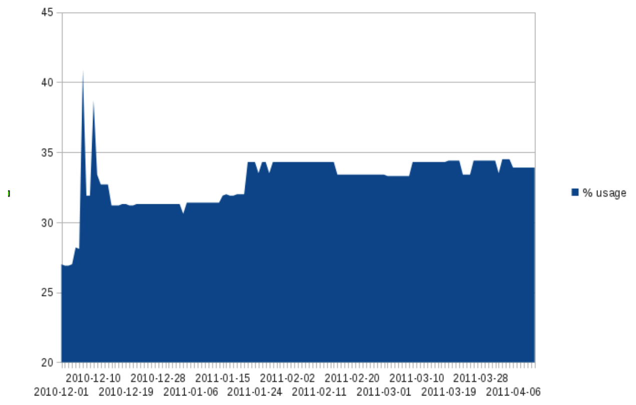


Figure 1: Disk usage of a virtual machine over the 5 months starting with installation. Notice the spikes when the VM was first installed, followed by a broad trend of very gradually increasing disk usage.