Stefan Hajnoczi

Open source and virtualization blog

Friday, 25 February 2011

How to access virtual machine image files from the host

I am going to explain how to mount or **access virtual machine disk images from the host** using `qemu-nbd`.

Often you want to access an image file from the host to:

- Copy files in before starting a new virtual machine.
- Customize configuration like setting a hostname or networking details.
- Troubleshoot a virtual machine that fails to boot.
- Retrieve files after you've decided to stop using a virtual machine.

There is actually a toolkit for accessing image files called libguestfs. Take a **look at that first** but what follows is the poor man's version using tools that come with QEMU.

Required packages

The required programs are the `qemu-nbd` tool and (optionally) `kpartx` for detecting partitions.

On Debian-based distros the packages are called `qemu-utils` and `kpartx`.

On RHEL 6 nbd support is not available out-of-the-box but you can build from source if you wish.

Please leave package names for other distros in the comments!

Remember to back up important data

Consider making a **backup copy** of the image file before trying this out. Especially if you don't work with disk images often it can be easy to lose data with a wrong command.

Attaching an image file

The goal is to make an image file appear on the host as a block device so it can be mounted or accessed with tools like `fdisk` or `fsck`. The image file can be in **any format** that QEMU supports including raw, qcow2, qed, vdi, vmdk, vpc, and others.

1. Ensure the nbd driver is loaded

The Network Block Device driver in Linux needs to be loaded:

```
modprobe nbd
```

The `qemu-nbd` tool will use the nbd driver to create block devices and perform I/O.

2. Connect qemu-nbd

Before you do this, make sure the virtual machine is not running! It is generally not safe to access file systems from two machines at once and this applies for virtual machines and the host.

There should be many `/dev/nbdX` devices available now and you can pick an unused one as the block device through which to access the image:

```
sudo qemu-nbd -c /dev/nbd0 path/to/image/file
```

Don't be surprised that there is no output from this command. On success the `qemu-nbd` tool exits and **leaves a daemon running** in the background to perform I/O. You can now access `/dev/nbd0` or whichever nbd device you picked like a regular block device using `mount`, `fdisk`, `fsck`, and other tools.

3. (Optionally) detect partitions

The `kpartx` utility automatically sets up partitions for the disk image:

```
sudo kpartx -a /dev/nbd0
```

Blog Archive

- ► 2016 (3)
- ► 2015 (8)
- ► 2014 (5)
- ► 2013 (7)
- ► 2012 (3)
- ▼ 2011 (26)
  - ► December (1)
  - ► November (1)
  - ► September (4)
  - ► August (2)
  - ► June (1)
  - ► May (1)
  - ► April (3)
  - ► March (10)
  - ▼ February (3)
    - How to access virtual machine image files from the...
    - Observability using QEMU tracing
    - Near instant kernel development cycle with KVM

Labels

- block
- gsoc
- internals
- kvm
- libvirt
- linux
- nbd
- qemu
- tracing

Blogs I follow

- aglitke
- Avi Kivity's blog
- Daniel P. Berrangé
- Richard WM Jones
- Rusty Russell's Coding Blog
- Tales of a Code Monkey

About me

I am an open source developer and contribute to QEMU and KVM. I work at Red Hat. The postings on this site are my own and don't necessarily represent Red Hat's positions, strategies or opinions.

They would be named `/dev/nbd0p1`, `/dev/nbd0p2`, and so on.

Detaching an image file

When all block devices and partitions are no longer mounted or in use you can clean up as follows.

1. (Optionally) forget partitions

```
sudo kpartx -d /dev/nbd0
```

2. Disconnect qemu-nbd

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

3. Remove the nbd driver

Once there are no more attached nbd devices you may wish to unload the nbd driver:

```
sudo rmmod nbd
```

More features: read-only, throwaway snapshots, and friends

The `qemu-nbd` tool has more features that are worth looking at:

- Ensuring **read-only** access using the `--read-only` option.
- Allowing write access but **not saving changes to the image file** using the `--snapshot` option. You can think of this as throwaway snapshots.
- Exporting the image file over the network using the `--bind` and `--port` options. Drop the `-c` option because no local nbd device is used in this case. **Only do this on secure private networks** because there is no access control.

Hopefully this has helped you quickstart `qemu-nbd` for accessing image files from the host. Feel free to leave questions in the comments below.

Posted by stefanha at 10:40

Labels: block, kvm, nbd, qemu

13 comments:

aglitke 28 February 2011 at 21:56

Thanks for this tutorial Stefan. One thing I would add is that if your disk image is using LVM (ie. you notice a 'Linux LVM' partition type in the fdisk output), you will need to use some additional tools to access the data in your image.

This brief tutorial should get you off to a good start.

Reply

stefanha 1 March 2011 at 09:44

Thanks for mentioning LVM. I've been confused in the past when my LVM volume group was called something generic like "vg" in the guest and on the host. Libguestfs solves this issue since it accesses the image from inside a special-purpose VM and does not affect the host.

By the way, Blogger flagged your comment as spam for some reason but I do not normally moderate comments. Hopefully in the future your comments will be published straight away :).

Reply

Anonymous 27 July 2011 at 11:50

thanks, stefan. only a mention for fedora OS. After running kpartx, we maybe need to check the partition device in /dev/mapper, then mount the device files under it. similar as below: mount /dev/mapper/nbd0p1 /tmp/1

Reply

Anonymous 16 February 2012 at 13:09

can i access virtual disk with libguestfs while the VM is running ?

Reply

Replies

Anonymous 22 February 2012 at 14:09

No, you cannot normally use libguestfs while the guest is running. When two things access an on-disk structure like a file system at the same time it is easy to cause corruption or see an inconsistent view.

However, in some cases you can run the guestfsd daemon inside your guest and use it carefully (for

example, formatting a file system that is in use would be a bad idea). See here for more info: https://rwmj.wordpress.com/2011/07/06/libguestfs-live/

Stefan

Reply

Gustavo Diaz 21 February 2013 at 16:24

guestmount is the sysadmin tool for libguestfs, looks powerfull, I find it thanks to your post :)
http://libguestfs.org/guestmount.1.html

Reply

Sathees 16 April 2013 at 09:27

Thanks for the beautifully explained post.

Reply

Sathees 16 April 2013 at 10:19

What is the usage of exporting the image file, using --bind and --port options of qemu-nbd ?, I was just looking for the usecase

Reply

Replies

stefanha 16 April 2013 at 10:33

From the qemu-nbd help output:
-p, --port=PORT port to listen on (default `%d')
-b, --bind=IFACE interface to bind to (default `0.0.0.0')

These are standard options for servers that use TCP (http://en.wikipedia.org/wiki/Transmission_Control_Protocol).

Only one socket can be bound to a TCP port number at any given time. That means you need to choose port numbers if you run multiple qemu-nbd server instances at the same time on a machine.

The bind option sets the address on which qemu-nbd server listens. The default (0.0.0.0) means it will listen on all network interfaces. If you wish to limit network access to qemu-nbd server to just this machine, use --bind=127.0.0.1. Then external machines will be unable to connect.

Reply

Varad 16 December 2013 at 10:33

I'd like to add that if you want to mount the nbd device thus created, make sure that max_part for nbd module is set to something >0, i.e. use `modprobe nbd max_part=15` instead. I recently faced this issue where `mount /dev/nbd0p1 /mnt` failed with `mount: special device /dev/nbd0p1 does not exist` because max_part wasn't set.

Reply

Replies

stefanha 16 December 2013 at 11:04

Good idea, that way you can skip the kpartx(8) step.

Reply

Kinnari Jangid 6 November 2015 at 10:25

Thanks, really helped a lot about nbd.

Reply

Unknown 8 April 2016 at 17:46

On Ubuntu 14.04 the partitions show up as dm-2 and dm-3:
brw-rw---- 1 root disk 253, 2 Apr 8 12:41 dm-2
brw-rw---- 1 root disk 253, 3 Apr 8 12:41 dm-3

The only way I figured this out was to look at verbose output of kpartx and match the device nodes:

add map nbd0p1 (253:2): 0 1523712 linear /dev/nbd0 2048
add map nbd0p2 (253:3): 0 41502720 linear /dev/nbd0 1525760

Reply