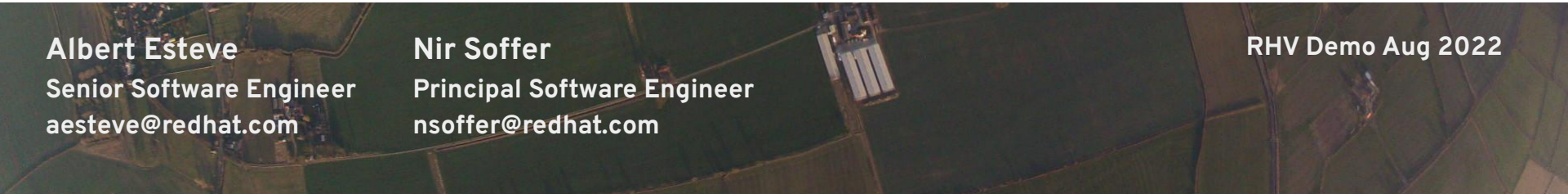
ovirt-img

Flying with NBD



Albert Esteve
Senior Software Engineer
aesteve@redhat.com

Nir Soffer
Principal Software Engineer
nsoffer@redhat.com

RHV Demo Aug 2022

Agenda

- Transferring images is hard
- The ovirt-img tool
- Live demo
- Future work

The "secret" plan

Transferring images is hard

Detecting image format

- ISO images needs special handling
- QCOW2 images needs special handling
  - More on this later

Getting image virtual size

- For QCOW2 we cannot use the file size
- We need to set the provisioned size to the virtual size

Determining disk initial size

- When upload QCOW2 sparse disk to block storage we must allocate enough space or the upload will fail when the disk becomes full
- We cannot use the file size since the source may be compressed

Determine disk format and allocation

- Do you want incremental backup?
  - Works only with QCOW2 disks
- Do you want best possible performance?
  - Use RAW preallocated
- How to convert RAW image to QCOW2 disk (or the other way around)?

Does storage domain support this?

- Can we upload RAW sparse image to this storage domain?
  - RAW sparse is not allowed on block based domain
  - QCOW2 preallocated requires backup="incremental"

How to transfer the data?

- Uploading or downloading data using HTTP is not trivial
  - Specially if you want to do this efficiently

How to transfer images faster?

- How to use multiple HTTP connections?
- How avoid transferring unallocated areas (read as zeros)?

How to optimize?

- If you run on an oVirt host in the same data center, you can upload faster and with minimal bandwidth using unix socket

Use transfer_url or proxy_url?

- transfer_url is the host URL
  - Much faster to upload directly to host
- proxy_url is engine URL
  - Much slower to upload via engine proxy, but if the host is not accessible this is the only way

Handling errors

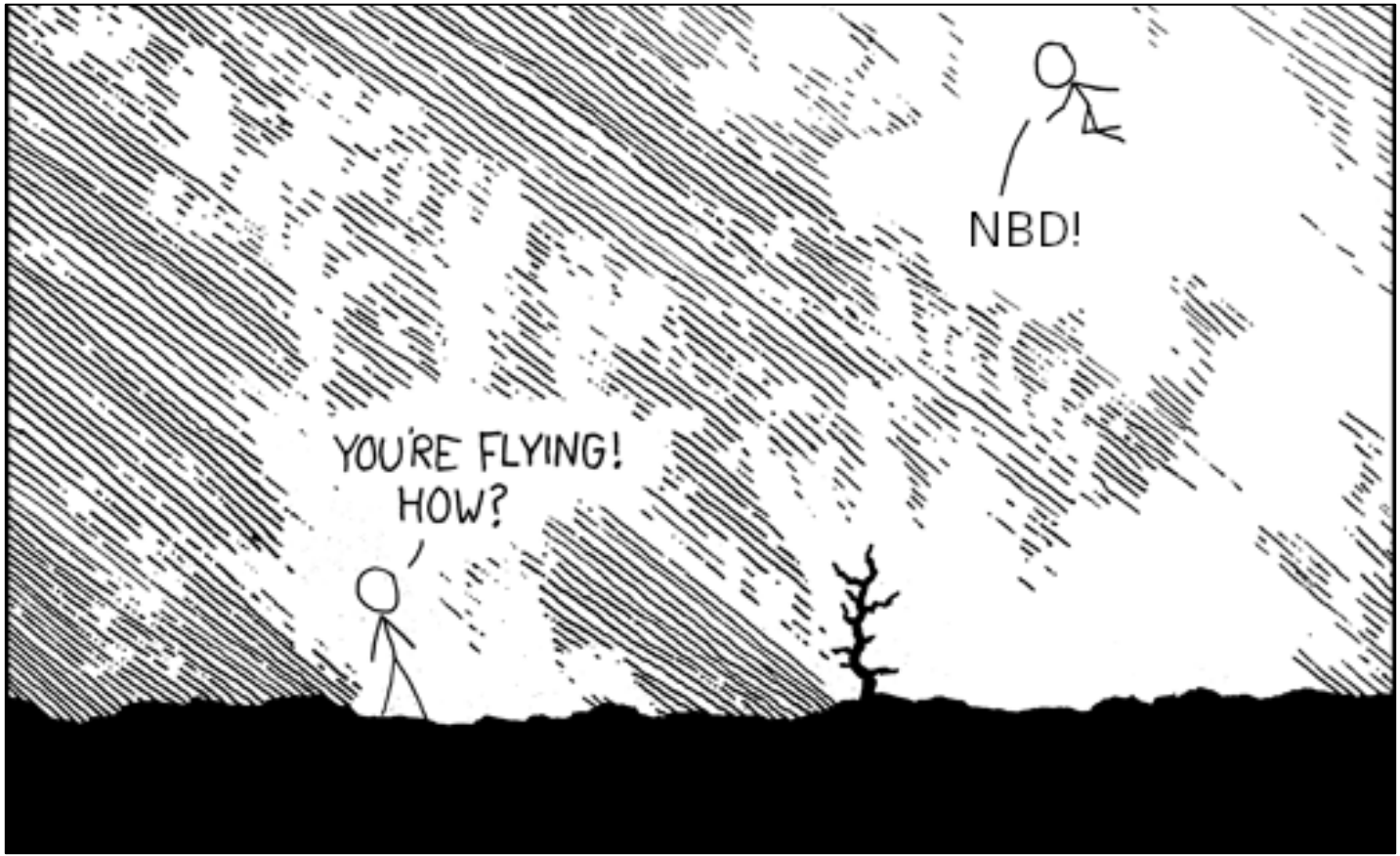
- On upload: cancel transfer on failures
- On download: finalize the transfer on failures

We have engine SDK examples right?

We have upload_disk.py, download_disk.py, but:

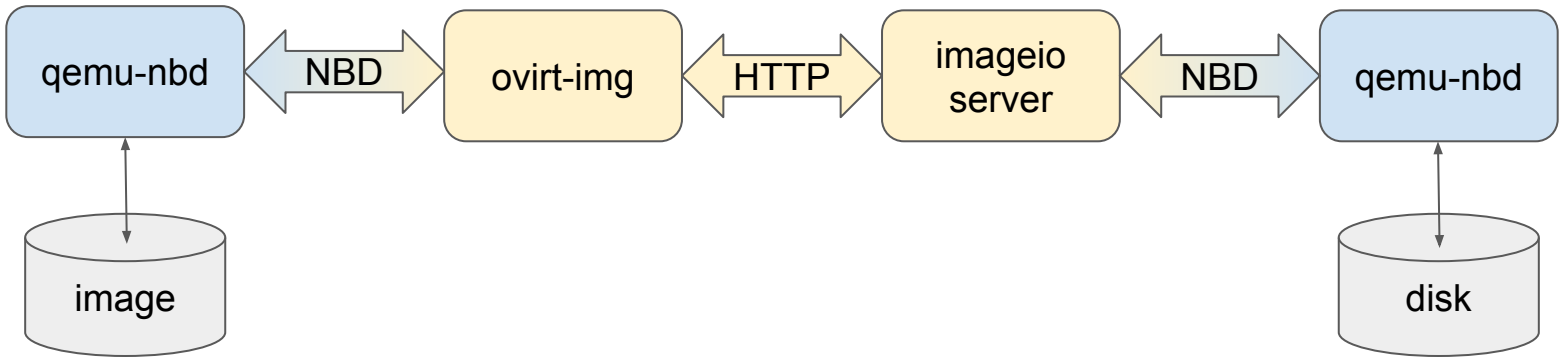
- On RHVH/oVirt node: Not installed
- On RHEL/CentOS: Installed in `/usr/share/doc/something (not in the PATH)`
- They don't have good defaults, easy to shoot yourself in the foot
- Not supported - just an example code

The ovirt-img tool



based on https://xkcd.com/353/

NBD-based pipeline



Always installed

- Part of ovirt-imageio-client package
  - Depends on python3-ovirt-engine-sdk4 package
- Can be installed via pip
- Will also be available via a container

Easy to use

Only requires server-side settings

```
$ ovirt-img download-disk \
    --engine-url https://engine.com \
    --username admin@internal \
    --cafile /path/to/cert.pem \
    {disk-id} download.qcow2
password:
[ 100% ] 6.00 GiB, 16.85 s, 364.64 MiB/s | download completed
```

Use configuration file

Add configuration for your setup (you can have many):

```
$ cat ~/.config/ovirt-img.conf
[engine1]
engine_url = https://engine.com
username = admin@internal
password = password
cafile = /path/to/cert.pem
```

Use configuration file

Specify the section name:

```
$ ovirt-img download-disk --config engine1 {disk-id} image.qcow2
[ 100% ] 6.00 GiB, 16.85 s, 364.64 MiB/s | download completed
```

It just works

ovirt-img does the right thing for the image and storage domain:

```
$ ovirt-img upload-disk -c engine --storage-domain iscsi-01 image.compressed.qcow2
[ 100% ] 6.00 GiB, 25.39 s, 241.97 MiB/s | upload completed

$ ovirt-img upload-disk -c engine --storage-domain nfs-01 image.raw
[ 100% ] 6.00 GiB, 34.19 s, 179.72 MiB/s | upload completed

$ ovirt-img upload-disk -c engine --storage-domain fc-01 image.iso
[ 100% ] 6.00 GiB, 30.24 s, 201.32 MiB/s | upload completed
```

Under the hood

- Inspects the image format and virtual size
- Measures the required size for the image
  - Supports compressed QCOW2 image
- Enables incremental backup by default
- Converts image format on the fly to QCOW2
- Detects ISO images and upload them as RAW preallocated
- Handles errors correctly
- Uploads efficiently (more on this later)

Efficient data transfer

- Detects and skips zero extents
- Using multiple HTTP connections
- Sparsify images (convert data with zeros to holes)
- Use unix socket when running on oVirt host

```
$ ./ovirt-img upload-disk -c engine -s fc-01 fedora-35-8t.qcow2
[ 100% ] 8.00 TiB, 95.97 s, 85.36 GiB/s | upload completed

$ ./ovirt-img download-disk -c engine d9c37bc3d155 fedora-35-8t.qcow2
[ 100% ] 8.00 TiB, 67.01 s, 122.25 GiB/s | download completed
```

Is flexible

Options to tweak the command behaviour for various use cases

Upload	
`--preallocated`	Create preallocated disk instead of sparse
`--format raw`	Create disk in RAW format
`--disk-id`	Set the UUID for the new disk
`--name`	Set the alias for the new disk
Download	
`--format`	Specify the format of the downloaded image


Live demo

Future work

- ovirt-img container
- More commands:
  - download snapshot
  - upload snapshot
  - backup disk
  - mirror disk (using incremental backup for warm import)
- Upload any image format supported by qemu-nbd
- Improve error handling
- Add system tests for actual commands

More info

- Project: https://github.com/oVirt/ovirt-imageio
- Open issues: ovirt-img open issues

Questions?