StorPool
DISTRIBUTED STORAGE

How to: Migrate VMs from vmware® to KVM

A step-by-step Guide

Table of Contents

Abstract

In a series of papers, we have outlined a detailed methodology for migrating virtual machines from some of the most popular traditional hypervisors to the New Age It stack, powered by KVM, with minimal downtime, using open-source tools.

This Step-by-step Guide is targeted at senior technical staff, who are building & managing IT infrastructure and the applications running on top of this infrastructure.

The presented techniques are successfully used to migrate CentOS and Windows Server virtual machines from VMWare ESXi, Hyper-V and Citrix XenServer to KVM, but it can be used with other guest operating systems and other types of hypervisors.

Introduction

For the last few years, companies have witnessed the rising popularity of open-source (OSS) technologies driven by large- and hyper-scale deployments, the cloud-native movement and the pursuit to eliminate vendor lock-in. These are some of the fundamental drivers to an economy, heavily driven by digital transformation where all businesses have to be agile, efficient and re-invented by and with technology.

The requirements of the business today require any company to heavily invest in New-Age, modern IT, which is resilient, automated, self-service, scalable. This is best achieved by intelligent software, thus we witness the rise of software-defined technologies and datacenters.

This process inevitably means that the old, traditional ways of designing and building IT systems are being replaced with latest-generation designs, tools, and systems, which deliver what the businesses of tomorrow will need.

On the hardware/IT infrastructure side, this usually means adopting SDDC (Software-Defined Datacenter) designs. I.e. using COTS (Commodity of the Shelf servers), standard Ethernet networks and SDS / SDN (Software-Defined Storage / Software-Defined Networking).

On the software side, this means adopting self-service, API-first stacks - typically Linux + KVM + containers with different orchestrators - Kubernetes, OpenStack for example. It also means migrating applications (or the VMs in which they run) from legacy hardware and software stacks (think VMware, Hyper-V, Oracle, etc.) to the modern IT stacks. A process that is not always simple or straight-forward.

There are many tools developed to facilitate the migration of virtual machines from one virtualization platform to another. Migrating a VM from one type of hypervisor to another requires converting the Virtual machine metadata, disk image format, and VM image content or OS morphing. The latter includes updating guest OS configuration settings and installing required drivers for the new target hypervisor.

One of the challenges in the migration process is reducing the downtime. Most tools we know implements more or less this generic workflow:

1. Stop the VM at the source hypervisor
2. Convert VM metadata and define a new VM at the target hypervisor
3. Copy the disk images from the source to the target hypervisor
4. Convert the VM disk image format
5. OS morphing or converting disk image content - update settings, install drivers for the new emulated hardware, etc.
6. Start the VM at the target hypervisor

Some tasks could be combined in a single step and the order could be slightly different, but in general, this workflow imposes a significant downtime. The majority of this downtime is accounted for copying the disk images, especially if this involves a migration of large disk images to remote storage.

The proposed method can reduce the downtime in many cases to less than a minute, even when large disk images have to be copied over low-bandwidth links. This can be achieved by altering the workflow as follows:

1. Convert VM metadata and define a new VM at the target hypervisor
2. Take a snapshot of the VM disks on the source hypervisor
3. Copy the snapshot to the target hypervisor
4. Convert the VM disk image format
5. Take a second snapshot, if needed, and copy the latest changes to the target hypervisor
6. Apply the changes to the target image.
7. Stop the VM at the source hypervisor
8. Copy the data changed since the last snapshot to the target hypervisor
9. Apply the changes to the target image.
10. OS morphing
11. Start the VM at the target hypervisor

While the overall number of steps has been increased, the downtime is reduced by limiting the time when the VM is non-operational to only 3 short steps, while the most time-consuming steps - copying large amounts of data are executed outside of this critical time frame.
The major challenge with this approach is the ability to take snapshots created in one image format on one storage and apply them on a different image format on different storage. In this paper, we'll show how this can be done with a set of open-source tools.

Supported Hypervisors

The method described in this paper supports migration from VMware ESXi, MS Hyper-V, and XenServer to KVM. Other types of hypervisors can be supported with small modifications, as far as source images are stored in the supported image file formats - vmdk (on VMFS6), vhd and vhdx. Check StorPool's website for the Step-by-step guides on how to migrate VMs from XenServer to KVM and from Hyper-V to KVM.

Used Tools

In this paper we use several tools:

- qemu-img - part of qemu project, used to convert base vmdk files to raw images.
- sesparse - open source tool used to load and apply vmdk snapshots onto raw images.
- virt-v2v - part of libguestfs project, this tool is used to morph the OS for use with KVM. It supports major Linux distributions and Windows as guest OS. The used version is compiled from source and includes a patch to install virtio drivers in WIndows guests.
- Support scripts to copy images from the source hypervisor

In this paper, we're using a dedicated Linux VM (called here conversion VM) to do the actual conversion. The conversion VM has an SSH access to the source hypervisors or the storage where it can download images from and has direct access to the target storage, where the target images are created as raw image files or the image content is stored in block devices. All tools listed above are installed on the conversion VM.

Tools can be obtained from:

- qemu-img: use the package supplied by the Linux distro
- sesparse, vhdx, vhd, various support scripts, patch for virt-v2v: https://github.com/storpool/any2kvm
- virt-v2v: http://download.libguestfs.org/ (see "Building virt-v2v" later in this paper for details)

VMware to KVM

Requirements

- A conversion Linux VM with all tools installed.
- ssh access to the source ESXi hypervisor or access to the vmdk files over NFS. In the example below, we're using ssh access to the ESXi.
- access to the vCenter or vSphere, that allows creation of snapshots and start/stop the VMs.

Step-by-step process

- Get the source VM details - CPU, RAM size, number of interfaces, etc.
- Create a target VM on the target hypervisor with the same settings and empty disks of the same size. The actual steps depend on the target orchestration.
- Stop the target VM
- If a shared block storage is used for the target VM (e.g. iSCSI), attach the target block device to the conversion VM (/dev/sdb in this example).

```
[user@libguestfs-2 vmware]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  150G  0 disk
└─sda1   8:1    0  150G  0 part /
sdb      8:16   0   20G  0 disk
```

- Make a snapshot of the source VM.

```
[root@esx:~] vim-cmd vmsvc/getallvms
Vmid             Name                                          File
...
95     Windows-2019               [storpool-ds1] Windows-2019/Windows-2019.vmx
[root@esx:~] vim-cmd vmsvc/snapshot.create 95 snap1
Create Snapshot:

[root@esx:~] cat /vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019.vmsd
.encoding = "UTF-8"
snapshot.lastUID = "3"
snapshot.current = "3"
snapshot0.uid = "1"
snapshot0.filename = "Windows-2019-Snapshot1.vmsn"
snapshot0.displayName = "snap1"
snapshot0.type = "1"
snapshot0.createTimeHigh = "368347"
snapshot0.createTimeLow = "-911790278"
snapshot0.numDisks = "1"
snapshot0.disk0.fileName = "Windows-2019.vmdk"
snapshot0.disk0.node = "scsi0:0"
snapshot.numSnapshots = "1"
```

- Copy all *.vmdk files to the conversion VM:

```
[user@libguestfs-2 vmware]$ export DS='storpool-ds1'
[user@libguestfs-2 vmware]$ export VM='Windows-2019'
[user@libguestfs-2 vmware]$ ssh root@esx ls -l /vmfs/volumes/$DS/$VM/Windows-2019*.vmdk
-rw-------    1 root     root     104861696 Feb 18 16:02
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000001-sesparse.vmdk
-rw-------    1 root     root           342 Feb 18 16:01
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000001.vmdk
-rw-------    1 root     root     21474836480 Feb 18 16:01
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-flat.vmdk
-rw-------    1 root     root           474 Feb 18 14:38
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019.vmdk

[user@libguestfs-2 vmware]$ scp root@esx:/vmfs/volumes/$DS/$VM/Windows-2019*.vmdk .
scp: /vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000001-sesparse.vmdk: Device or
resource busy
Windows-2019-000001.vmdk                                  100%  342   177.6KB/s   00:00
Windows-2019-flat.vmdk                                    100%   20GB  89.8MB/s   03:48
Windows-2019.vmdk                                         100%  474   143.8KB/s   00:00
```

It is expected that the currently used vmdk file is busy and will not be copied.

- Convert the base image to raw format using `qemu-img` :

```
[user@libguestfs-2 vmware]$ qemu-img convert -p -f vmdk Windows-2019.vmdk -O raw /dev/sdb
    (100.00/100%)
```

- (Optional) Make a second snapshot

```
[root@esx:~] vim-cmd vmsvc/snapshot.create 95 snap2
Create Snapshot:
```

- Copy the snapshot content and apply it to the target raw image using sesparse tool.

```
[user@libguestfs-2 vmware]$ ssh root@esx ls /vmfs/volumes/$DS/$VM/Windows-2019*.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000001-sesparse.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000001.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000002-sesparse.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000002.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-flat.vmdk
/vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019.vmdk

[user@libguestfs-2 vmware]$ scp root@esx:/vmfs/volumes/$DS/$VM/Windows-2019-00*-sesparse.vmdk .
Windows-2019-000001-sesparse.vmdk                         100%  100MB 103.5MB/s   00:00
scp: /vmfs/volumes/storpool-ds1/Windows-2019/Windows-2019-000002-sesparse.vmdk: Device or
resource busy

[user@libguestfs-2 vmware]$ ./sesparse Windows-2019-000001-sesparse.vmdk /dev/sdb
capacity 41943040
dir[35] = 1000000000000010
dir[36] = 100000000000000e
...
```

- Shutdown the VM, copy and apply the changes since the last snapshot

```
[root@esx:~] vim-cmd vmsvc/power.shutdown 95  # or power.off if vmware tools are not installed
[user@libguestfs-2 vmware]$ scp
root@esx:/vmfs/volumes/$DS/$VM/Windows-2019-000002-sesparse.vmdk .
Windows-2019-000002-sesparse.vmdk                          100%  116MB  84.9MB/s   00:01
[user@libguestfs-2 vmware]$ ./sesparse Windows-2019-000002-sesparse.vmdk /dev/sdb
capacity 41943040
dir[0] = 1000000000000046
dir[11] = 1000000000000043
dir[34] = 100000000000001c
...
```

- Morph the OS. Use the compiled virt-v2v tool. Use a minimal libvirt domain xml configuration file listed below for the target VM. This file is needed only for this step. The actual domain xml of the target VM is created by the orchestration system and will be different.

```
[user@libguestfs-2 vmware]$ cat domain.xml
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
    <name>tmp-v2v</name>
    <devices>
        <disk type='file' device='disk'>
            <source file='/dev/sdb'/>
            <target dev='sda' bus='scsi'/>
            <driver name='qemu' type='raw' cache='none' io='native' discard='unmap'/>
            <address type='drive' controller='0' bus='0' target='0' unit='0'/>
        </disk>
        <controller type='scsi' index='0' model='virtio-scsi'>
            <driver queues='1'/>
        </controller>
    </devices>
</domain>

[user@libguestfs-2 vmware]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  150G  0 disk
└─sda1   8:1    0  150G  0 part /
sdb      8:16   0   20G  0 disk
├─sdb1   8:17   0  549M  0 part
└─sdb2   8:18   0 19,5G  0 part
[user@libguestfs-2 vmware]$ RUN=/home/user/libguestfs-1.40.2/run
[user@libguestfs-2 vmware]$ $RUN virt-v2v -i libvirtxml domain.xml --in-place
[   0,0] Opening the source -i libvirtxml domain.xml
[   0,0] Opening the source VM
[  12,7] Inspecting the source VM
[  19,8] Checking for sufficient free disk space in the guest
[  19,8] Converting Windows Server 2019 Standard Evaluation to run on KVM
virt-v2v: warning: /usr/local/share/virt-tools/pnp_wait.exe is missing.
Firstboot scripts may conflict with PnP.
virt-v2v: warning: there is no QXL driver for this version of Windows (10.0
x86_64).  virt-v2v looks for this driver in /usr/local/share/virtio-win

The guest will be configured to use a basic VGA display driver.
virt-v2v: This guest has virtio drivers installed.
[  31,0] Mapping filesystem data to avoid copying unused and blank areas
[  31,9] Closing the source VM
[  32,2] Finishing off
```

- Start the VM at the target hypervisor

Notes:

1. The actual location of vmdk files may be different, depending on the storage options of the ESXi.
2. In this example, we're using a raw block device for the target virtual disk. If file storage is used a raw image file will be created and the filename shall be passed to `sesparse` and `qemu-img` tools.
3. If cqow2 output image format is needed, then the processing shall be done in raw format, and as the last step, the image shall be converted from raw to qcow2 with `qemu-img` .
4. In this example, we're converting Windows Server 2019 VM. The output of the morphing will be different for the different guest OS. See below for preparing the `virt-v2v` .
5. If any of the conversion steps after the VM has been stopped at the source hypervisor fails or if the converted VM is not able to start at the target hypervisor, the VM can be re-started on the source HV and the procedure can be repeated.
6. The procedure of converting virtual machines and virtual network definition from VMware vCenter/vSphere to the target KVM-based cloud is not covered in this paper. Only the transfer, conversion and morphing of the disk images is covered here.
7. `sesparse` tool supports only VMDK files in `SESPARSE` format. Disk images in VMFS format can be converted with `qemu-img` . Snapshots in the older `VMFSSPARSE` format created on VMFS5 storage are not supported. Use VMFS6 to create snapshots in `SESPARSE` format.

Building virt-v2v

An important part of the conversion process is morphing the guest OS - installing virtio drivers required to run the OS on KVM and updating the OS configuration. This is done using the virt-v2v tool, part of libguestfs project (http://libguestfs.org/). virt-v2v is available as a package for most Linux distributions, but the packaged version has some limitations. For example, the package available in the CentOS repo doesn't permit in-place image processing. This is a key feature for the method described here to reduce the downtime, by avoiding mass copying of large amounts of data. Also, the bundled package is not capable of installing virtio drivers for windows, because it doesn't register the certificate that is used to sign the drivers.
For these reasons, virt-v2v needs to be compiled from source after applying some changes. Following is the step-by-step procedure for building `virt-v2v` for CentOS 7.

- Get the source tarball of the latest stable libguestfs from http://download.libguestfs.org/
- Get the patch for virt-v2v, link is available in section 'Used Tools'.
- Get virtio-win-*iso from https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/latest-qemu-ga/
- Use a windows workstation or VM to extract a signed driver file form the iso image, for example `amd64\w10\vioscsi.sys` and extract RedHat certificate from it:
Select the .sys file, right-click -> Properties -> Digital Signatures -> Select 'Red Hat, Inc.' -> Details -> View Certificate -> Details -> Copy to File ...; Select 'DER encoded binary X.509 (.CER)'
- Get the created `.cer` file in the conversion VM and save it as `cert1.cer` . It will be used a few steps later.
- Verify `cert1.cer` file is correct, and the certificate is valid (note the 'Not After' date) :

```
[user@libguestfs-2 ~]$ openssl x509 -inform DER -in cert1.cer -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5d:10:cb:18:eb:3a:79:00:87:83:ab:74:77:f9:d3:19
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network, CN=Symantec Class 3
SHA256 Code Signing CA - G2
        Validity
            Not Before: Nov 27 00:00:00 2018 GMT
            Not After : Jan 25 23:59:59 2022 GMT
        Subject: C=US, ST=North Carolina, L=Raleigh, O=Red Hat, Inc., CN=Red Hat, Inc.
        Subject Public Key Info:
...
```

- Untar the libguestfs source:

```
tar zxvf libguestfs-1.40.2.tar.gz
```

- Install the dependencies:

```
sudo yum install yum-utils
sudo yum-builddep libguestfs
sudo ln -s supermin5 /usr/bin/supermin
```

- Apply the patch for virt-v2v:

```
cd libguestfs-1.40.2/
patch -p 1 -i ../v2v-patch
```

- Configure and build libguestfs:

```
./configure
make
make quickcheck
```

- Install windows support. This is needed only if windows VM will be morphed:

```
sudo yum install libguestfs-winsupport
cp /usr/lib64/guestfs/supermin.d/zz-winsupport.tar.gz appliance/supermin.d/
sudo mkdir -p /usr/local/share/virtio-win
sudo mount -o loop,ro virtio-win-0.1.173.iso /usr/local/share/virtio-win
sudo cp /usr/local/share/virtio-win/guest-agent/qemu-ga-x86_64.msi /opt/windows-convert/files/
sudo mkdir -p /opt/windows-convert/files/
sudo cp cert1.cer /opt/windows-convert/files/cert1.cer
sudo cp /usr/local/share/virtio-win/guest-agent/qemu-ga-x86_64.msi /opt/windows-convert/files/
```

We hope this paper gives you a detailed methodology and easy way to migrate your virtual machines. If you have any comments or questions, our technical team will be happy to talk to you! Drop us an email at info@storpool.com.

StorPool
DISTRIBUTED STORAGE

Get in Touch

+1 415 670 9320

+44 (0) 20 7097 8536

info@storpool.com

sales@storpool.com

Get Started | Book a Demo