Tricks and Techniques

Following are just some notes on how to “do things” with virtuals, especially converting from one form to another

Migration

One nice thing about some hypervisors it the ability to move a virtual machine (vm) from one physical machine to another. There are two main ways to do this, shutting down the running vm, or doing it while the vm is still running.

In the following sections, *source* is the hypervisor moving from, *target* is the hypervisor being moved to, and 'vm' is the virtual machine being moved.

Shutting Down and moving

This method has the least limitations, but takes longer and may require significant manual intervention. The configuration of the source and target may be different, and the block devices do not need to be on a shared system. It is even feasible to have Xen running on the source and KVM running on the target, in theory.

1. Copy the vm configuration from *source* to *target*
    1. Modify the configuration on *target*
        1. If network bridges are different, either create the network bridge on the target, or change configuration on the *target*
        2. Perform any additional changes to the configuration file to adapt to *target* hypervisor. **Note** if source and target use different hypervisors and/or managers, this may require a complete rewrite of the configuration on *target*
2. Shut down the vm on *source*
3. Move all block devices to *target*.
    1. If iSCSI, NFS, or some other network shared storage, this may be as simple as mounting on *target*, and can be done before shutting down vm.
    2. If this is a physical partition,Logical Volume, or file on *source*, you must copy to *target* while the vm is down (or, snapshot and lose changes over the migration period)
4. Bring the vm up on *target*

Live Migration

Live Migration rapidly migrates a running vm from source to target, in many cases with no downtime detectable by the end user.

This generally has some limitations imposed. For example, migrating a running Xen machine must have Xen on both *source* and *target*, and the version of Xen must be very close on both machines. Same with KVM and virtlib. The CPU's on both machines should be similar also.

But, if you have compatible systems, you can migrate machines with little or no impact on end users, either as precursor to maintenance, or simply to manually load balance your hypervisors.

Limitations

- Both hypervisors must be running compatible hypervisors.
  - Should be same hypervisor family (xen, kvm, or a manager like virtlib that knows both)
  - Hypervisor versions should be the same, though this will generally work within major versions, ie v7.1 and v7.5 should be able to work together.
- Block devices should be on shared storage.
  - Usually uses iSCSI or a file on an NFS share
  - Paths to block devices on *source* and *target* must be exact
  - *source* and *target* should never use block devices simultaneously.
    - *source* should shut down vm, first, freeing up block device,
    - *target* may now bring up block devices
- Network definitions used by vm configuration must be accessible with same name on both *source* and *target*.
  - *source* and *target* should be on same subnet
- *source* root user must have ssh access to *target*
  - It is preferred if *source*.root can log into *target*.root without password.

Procedure

1. Ensure root access from *source* to *target* (required for Xen, preferred for all others)
   1. **Xen**

      ```
      ssh target 'ls'
      ```

      should give you a listing of all files in /root on *target*

   2. **virtlib** issue the command

      ```
      virsh -c qemu+ssh:%%//%%target/system list
      ```

      should return a list of running virtuals on *target*

2. Verify compatibility
   1. Ensure hypervisor compatibility between *source* and *target*
   2. Ensure network bridges with same name on *source* and *target*
   3. Ensure path to block devices same on *source* and *target*
3. Issue command for migration from *source*. **Note** you can use the running virtual number instead of it's actual name in many cases
   1. **Xen** -

      ```
      xl migrate vm targetmachine
      ```

   2. **KVM** - unknown
   3. **virtlib** -

      ```
      virsh migrate --live //vm// qemu+ssh://target/system
      ```

Move VirtualBox images to KVM/Xen

Oracle VirtualBox uses their own format for the disk back end. In order to move the block device from VirtualBox to something something else, the image needs to be converted to a *raw* format, then optionally converted to a *qcow2* format to conserve disk space. **Note:** in each case below, the commands need to be executed from a machine which has the virtualization tool installed. VBoxManage is only available on a VirtualBox machine, and qemu-img is only available on a machine which uses it, ie something that has had virt-manager, kvm, xen or qemu installed.

Also, note that neither of these commands modify the original disk image. They convert the image, copying the converted file to a new one.

The following code will take VirtualBox's sparse disk and convert it to a raw format, in essence a disk image **with no compression**. In other words, if you have a 40G disk image, but are only using 2.4G of disk space because VirtualBox realizes you are only using that much space, /target/disk.raw will require the full 40 gigabytes.

```
VBoxManage clonehd /source/disk.vdi /target/disk.raw --format raw
```

Virt-Manager can work with qcow2 formatted disk images, which are similar to VirtualBox's vdi (sparse) format, so we would really like to convert the raw image above to qcow2 whenever disk space is important. To do this, issue the following command:

```
qemu-img convert -f raw /path/to/raw/disk.raw -O qcow2
/path/to/qcow/disk.qcow2
```

You can now delete the original and intermediate files when you feel confident the process has gone well

Sticky IP's (Reservations) with virt-manager

virt-manager defaults to NAT on the networking. I find this useful on my laptop, where I don't always have a DHCP server available and the running virtuals should be private to my laptop. However, the dhcp server will occasionally give a different IP to the same machine.

To get around this, find the MAC address of the virtual, then set the network to always give it the same IP address. By default, the dhcp server built into virt-manager sets up an entire /24 range except for the primary IP, so we'll need to adjust that range also.

First, find the MAC address of the virtual you want to set up this way, and get the network list (usually “default”)

```
# get network name
virsh net-list
# get MAC address of a virtual's network interface
virsh  dumpxml  //vm_name// | grep 'mac address'
```

Now, edit the DHCP server

```
# change default to whatever you got in net-list, if it is different
virsh net-edit default
```

Look through this list and find the XML block which defines the DHCP server. Adjust the range for automatic assignments to give you room to add your reservations, then add a new line within the dhcp block to assign a sticky IP based on MAC address found above. Below, we have adjusted the *range start* address ufrom 192.168.122.2 to 192.168.122.100. .2-.99 are then available for sticky IP reservations. We then added three lines for vm1, vm2 and vm3, giving them an IP based on their mac addresses

```
<dhcp>
  <range start='192.168.122.100' end='192.168.122.254'/>
  <host mac='52:54:00:6c:3c:01' name='vm1' ip='192.168.122.11'/>
  <host mac='52:54:00:6c:3c:02' name='vm2' ip='192.168.122.12'/>
  <host mac='52:54:00:6c:3c:03' name='vm3' ip='192.168.122.12'/>
</dhcp>
```

Save your work, then you need to do something to get it all to begin work. https://serverfault.com/questions/627238/kvm-libvirt-how-to-configure-static-guest-ip-addresses-on-the-virtualisation-ho#627245, where I stole this idea, has a lot of things to do, but I just rebooted the damned thing and it began working just fine.

Links

- https://www.utappia.org/2016/04/how-to-migrate-your-virtual-box.html
- https://serverfault.com/questions/627238/kvm-libvirt-how-to-configure-static-guest-ip-addresses-on-the-virtualisation-ho#627245