18
Nov
2010

One way to migrate Xen virtual machines to KVM in Debian

There are dozens of virtualization products on the market. When we launched our first high-availability cluster in early 2008 we chose Xen due to it's ability to run on non-virtualized hardware, support in Debian 4.0 (Etch) and general flexibility. We've learned a lot about the upstream of Xen, including the challenges that Debian maintainers face, and we were increasingly drawn to another free and open source virtualization technology, Kernel Based Virtual Machine (KVM). The primary downside to KVM is that it requires special CPU hardware support to run, but this hardware support is now almost ubiquitous on modern servers. KVM has the advantage of being supported upstream in the Linux kernel itself, removing the onus of difficult kernel patching from the Debian Developers and has become the supported virtualization option for Ubuntu, Fedora and Red Hat. Additionally, KVM allows the guests to run unaltered, meaning you don't need a special kernel and can run many OSes, from Linux to FreeBSD to Windows 7.

We still work on a number of machines which lack hardware virtualization support, but as our customers upgrade hardware we've begun moving production virtual machines from Xen to KVM. In tackling the migrations of these production virtual machines we encountered several challenges, the major ones being:

- In Xen, partitions were created in separate logical volumes on the host and mounted by Xen itself and as a result we didn't require Logical Volume Manager (LVM) within our Xen guests, in KVM the logical volume on the host for a virtual machine is a single disk image, not separate partitions.
- In Xen, the kernel package is not installed, in KVM it is required
- In Xen, you don't have an independent bootloader on the OS, in KVM you need one to boot

The first step was to create a partition table on the new KVM image which is identical to the one on Xen. We wanted to use LVM within the guest, which required a Matryoshka (Russian) doll approach. First we'd create a volume group on the host to give us the typical flexibility of LVM host-side, and then we'd create one on the disk image giving us the flexibility required within the guest itself to expand any partitions. Finally we'd need to bootstrap the new system and copy the files over.

One way to go about all of this is a manual process. This solution would allow for scripting the procedure but requires a significant investment of time to get everything right so there is the least amount of down time possible. Since we only have a half dozen of these VMs to migrate in total, we looked for some way which already existed for handling all these steps in a familiar way, which is when we looked to the Debian Installer. Assuming a local mirror of core Debian packages (recommended), we could install a skeleton system on a test IP address which was properly partitioned, had LVM configured and bootstrapped in less than 20 minutes. We could then take this skeleton system that we're sure is an functioning, bootable system and move the files over from the Xen installs, arguably decreasing our risk with each migration and the downtime required.

To get started, you’ll first need to calculate the total size of the current VM and lvcreate a slice of that size on the KVM system, then launch the Debian installer against the image using virt-install, something like:

```
virt-install --connect qemu:///system -n guest.example.com -r 768
-f /dev/mapper/VolumeGroup-guest.example.com -s 12 -c
/var/lib/libvirt/images/debian-506-i386-netinst.iso --vnc --
noautoconsole --accelerate --os-type linux --os-variant
debianLenny
```

In this example we assume the debian-506-i386-netinst.iso is in /var/lib/libvirt/images/ and we want 768M of RAM, the information you put here is similar to the information that you would have previously defined in your /etc/xen/guest.example.com.cfg file for Xen.

Then use virt-manager to connect (we actually connected from a remote desktop running virt-manager) to the running install session (the standard Debian installer does not provide serial access) and install Debian, you will need the root password to launch the installer. Proceed to install.

When you get to the step to partition the disks, lay out the partition table to be identical to the VM you want to migrate to it except put it on LVM, put /boot on a separate partition outside of LVM. Complete the install, including installing grub.

Confirm that the system will boot and works on a test IP address and make a copy of your /etc/fstab to the host system, *you’ll need this later.*

You now have a skeleton install of Debian which runs on KVM with the LVM partitions you need.

To begin the actual data migration, you’ll want to mount the volumes within your new KVM disk, this can be done with the help of a great little mapping tool called kpartx. To map and activate the volume group on the guest, following these steps:

```
kpartx -av /dev/VolumeGroup/guest.example.com
vgscan
vgchange -ay guest
```

In this example we assume the Linux Volume on the host is called “guest.example.com” and the Linux Volume within the guest is called “guest”.

Now that the host can see the Volume Group on the guest, and all the Volumes in /dev/mapper/ you’ll want to mount them.

Once mounted, you’ll be able to start your rsync. To incur the least amount of downtime, you’ll want to rsync the large data partitions (like /srv, /home, /var, perhaps /usr) while your production host is still running. *Note: All rsyncs completed during this process must be done with the “–* ***numeric-ids****” option so the permissions are not inherited from the host!*

While you’re rsyncing the data, you’ll want to go into your Xen system and install the following packages (installing these will no impact your Xen system, it doesn’t strictly use them so it will simply ignore them):

- linux-image-2.6-686
- grub
- lvm2

When you’ve completed moving the largest portions of your Xen guest, bring down the production Xen guest (downtime starts now!) and mount the filesystems. And begin rsyncing the data over (preferably over a crossover cable for the fastest transfer, remember to use *–numeric-ids* in your rsync).

Once the rsync is complete, edit the following files on the KVM host:

- /mnt/guest/etc/fstab – use the version you saved to the host in a previous step
- /mnt/guest/etc/inittab – uncomment the ttyS0 line to allow for serial access from the KVM host for virsh
- /mnt/guest/etc/udev/rules.d/70-persistent-net.rules – comment out eth0 line so eth0 can come up on the new KVM MAC address

Unmount the filesystems on both sides and on the KVM side disable the volume group and use kpartx again to unmap the filesystem from the host:

```
sudo vgchange -an guest
sudo kpartx -dv /dev/VolumeGroup/guest.example.com
```

You’re now ready to boot the VM on the KVM side. This can be done with virt-manager or virsh.

Since you just moved the machines to a new server, and probably new MAC addresses, you will probably need to run the arping command to reclaim the IP address of the VM and all its service IP addresses.

Some things to confirm are working:

- Networking (confirm there are no lingering arp caching issues)
- Email (where applicable, confirm system messages etc are being sent)
- All services running (confirm key services, review monitoring dashboard, log in via ssh)
- Confirm that you have contiguous logs

Now that we’ve completed one of these migrations we have a lot of ideas about how to improve the process, including the possibility of making the whole process more scriptable, but this quick method leveraging the Debian installer for easier disk configuration and bootstrapping worked very well.

Posted by Elizabeth Krumbach *in* Debian, Systems Management, Tech Notes, Virtualization, *0 comments*