Creating multiple VMs from a single VM

Contents

1 Create a "Gold" image with your public key included

To get started Connect to your workshop server as the "nsrc" user. Don't become root except where indicated.

To do this we need to update the authorized_keys file that vmbuilder uses when building a new virtual machine. We will place your public key in a directory on the nsrc user account. When you create the directory please use your name so that you do not clash with other users doing this exercise:

```
cd
mkdir <YOURNAME>
```

Copy your public key to the /home/nsrc/ directory. Use the instructions below for the platform on your laptop.

1.1 For Windows users

Open the psftp.exe program

```
psftp> open s1.ws.nsrc.org
login as: nsrc
nsrc@s1.ws.nsrc.org's password: <usual one>
Remote working directory is /home/nsrc
psftp>cd <YOURNAME>
Remote working directory is /home/nsrc/<YOURNAME>
psftp> put id_rsa.pub
local:id_rsa.pub => remote:/home/nsrc/<YOURNAME>id_rsa.pub
psftp> quit
```

Your public ssh key is now in the /home/nsrc/ directory on your workshop server.

Now we need to fix the key's format as it is not compatible with the SSH server format on Linux and Unix.

Open putty.exe and make a connection to s1.ws.nsrc.org as the user nsrc. Once you have done this do:

```
cd <YOURNAME>
ssh-keygen -i -f id_rsa.pub >> id_rsa.pub.new
```

Note that we use a different name for the resultant public key.

Now that your public key is in the correct format you can add it to the authorized_keys file that vmbuilder uses to create a new virtual machine.

```
sudo -s
cat id_rsa.pub.new >> /etc/vmbuilder/misc/authorized_keys
exit
```

That's it. Your public key is now in the correct location for vmbuilder.

1.2 For Linux / Unix / OS X users

Open a shell (sometimes called terminal) on your laptop. Inside the shell type the command:

```
cd
cat .ssh/id_rsa.pub | ssh root@s1.ws.nsrc.org ' cat >> /etc/vmbuilder/misc/authorized_keys'
```

If you don't understand what this command is doing please ask one of your instructors.

1.3 Use vmbuilder to create a Gold image for workshop student vms

Be sure you log back in to s1.ws.nsrc.org as user nsrc for these steps.

Now we will generate a single virtual machine that we will duplicate multiple times that can be used for students in a workshop. To do this we will use vmbuilder like you have done previously.

Everyone in your group will ssh into the server and build a VM; it's fine for you to do them at the same time. Use the following name for your initial VM:

- 1st person in the group: gold.ws.nsrc.org
- 2nd person in the group: gold2.ws.nsrc.org
- 3rd person in the group: gold3.ws.nsrc.org
- . . . etc

To build your VM, run the following command. Agree amongst yourselves which image you will each build. If you are not the first person in the group then modify the two places where "gold.ws.nsrc.org" appears.

```
sudo vmbuilder kvm ubuntu --hostname gold.ws.nsrc.org --mem 512 --debug \
    --rootsize 20480 --dest /var/lib/libvirt/images/gold.ws.nsrc.org
```

This builds a VM which will initially have 512MB RAM allocated when it runs, and a root disk which can grow up to 20GB.

Check that no errors are shown at the end of the build. You should see something like:

```
...
2013-11-11 10:30:21,390 DEBUG   : Calling deploy method in VMBuilder.plugins.ubuntu.distro p
2013-11-11 10:30:21,391 DEBUG   : No such method
2013-11-11 10:30:21,391 DEBUG   : Calling deploy method in context plugin VMBuilder.plugins.
2013-11-11 10:30:21,391 DEBUG   : ['rm', '-rf', '--one-file-system', '/tmp/tmpjC8QLO']
```

If you have three people in your group, then you would end up with the following three new VM image files:

```
/var/lib/libvirt/images/gold.ws.nsrc.org
/var/lib/libvirt/images/gold2.ws.nsrc.org
/var/lib/libvirt/images/gold3.ws.nsrc.org
```

Verify your VM is available for use:

```
virsh list --all
```

You might see something like:

```
 Id Name                 State
----------------------------------
  - gold.ws.nsrc.org     shut off
  - gold2.ws.nsrc.org    shut off
  - gold3.ws.nsrc.org    shut off
  - noc.ws.nsrc.org      shut off
```

2 Updating your Gold VM

Start the VM that you built. Remember to change "gold.ws.nsrc.org" to the name of your VM.

```
virsh start gold.ws.nsrc.org
virsh list         # show running VMs
```

If you do not see your VM running you can ask for help to troubleshoot this.

Now we want to attach to our VM via the emulated serial console. We don't use ssh to connect at first as we do not know the IP address that has been assigned to your new gold VM. Remeber to use your VM's name below:

```
virsh console gold.ws.nsrc.org
```

Hit Enter a few times. You should get a login prompt. Log in using:

```
user: sysadm
password: <Given in Class>
```

View the VM's IP address and verify network connectivity is working:

```
ifconfig -a
ping 8.8.8.8
ping nsrc.org
```

Now let's add a package that was not included by the vmbuilder process. You may realize after building a Gold VM that you have some specific settings or items that you want all the duplicated VMs to have that will be created from your gold VM image.

In this case let's install nmap (Network MAPper). A useful network forensic tool and do an "apt-get update" to make sure our Ubuntu package database is up to date and an upgrade to make sure all packages are up to date:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nmap
```

You can test that nmap is working:

```
nmap localhost
```

At this point your gold VM is in a state that we would like to freeze and use for multiple VMs that will be created from this image.

Exit from the gold image by typing:

```
ctrl-]
```

This drops you back to the s1.ws.nsrc.org prompt. Now do:

```
virsh list --all
```

You can see your gold image running. Let's shut it down via virsh (remember to use your VM's name):

```
virsh shutdown gold.ws.nsrc.org
```

Verify your VM has shut off:

```
virsh list --all
```

3 Manually clone your gold VM

First, let's update our workshop-kit git repository on s1.ws.nsrc.org.

```
cd
cd workshop-kit
git pull http://trainers@wsnoc.nsrc.org:8000/nsrc/workshop-kit.git
```

When you installed the workshop-kit Git repository on s1.ws.nsrc.org one of the directories that was installed includes some useful scripts for cloning VMs.

```
cd scripts
ls
```

You should see the file clonevm.sh. This is a shell script to help simplify the process of duplicating a VM from an initial image (our gold image). Try running the script with no options:

```
./clonevm.sh
```

You should see:

```
Usage: [BRIDGE=brX] ./clonevm.sh <source> <clone-as> [target-dir]
Then use 'virsh start <clone-as>' to run it
```

Let's use this script to create a new VM cloned from the gold image you just created. For this exercise we are going to split in to three groups per workshop server. This is important to ensure you do not overwrite your neighbor's work. Here are the groups:

```
gold  ==> vm1-vm10
gold2 ==> vm11-vm20
gold3 ==> vm21-vm30
```

To start you are going to manually create two new VMs using the clonevm.sh script, like this:

```
gold creates vm1.ws.nsrc.org and vm2.ws.nsrc.org
gold2 creates vm11.ws.nsrc.org and vm12.ws.nsrc.org
gold3 creates vm21.ws.nsrc.org and vm22.ws.nsrc.org
```

Here is what each person should now do:

For the gold VM:

```
./clonevm.sh gold.ws.nsrc.org vm1.ws.nsrc.org
./clonevm.sh gold.ws.nsrc.org vm2.ws.nsrc.org
```

For the gold2 VM:

```
./clonevm.sh gold2.ws.nsrc.org vm11.ws.nsrc.org
./clonevm.sh gold2.ws.nsrc.org vm12.ws.nsrc.org
```

For the gold3 VM:

```
./clonevm.sh gold3.ws.nsrc.org vm21.ws.nsrc.org
./clonevm.sh gold3.ws.nsrc.org vm22.ws.nsrc.org
```

What you might notice is how fast this is. Each of you now have two new VMs that are duplicate images of your initial gold VM. View what VMs are available:

```
virsh list --all
```

You should see something like:

```
 Id Name                 State
----------------------------------
  - gold.ws.nsrc.org     shut off
  - gold2.ws.nsrc.org    shut off
  - gold3.ws.nsrc.org    shut off
  - noc.ws.nsrc.org      shut off
  - vm1.ws.nsrc.org      shut off
  - vm11.ws.nsrc.org     shut off
  - vm12.ws.nsrc.org     shut off
  - vm2.ws.nsrc.org      shut off
  - vm21.ws.nsrc.org     shut off
  - vm22.ws.nsrc.org     shut off
```

Start your two new VMs now using the "virsh start" command. Hint, for the gold VM user "virst start vm1.ws.nsrc.org" and so on. Once you are all done you should see something like:

```
 Id Name                 State
----------------------------------
  5 vm1.ws.nsrc.org      running
  6 vm2.ws.nsrc.org      running
  7 vm11.ws.nsrc.org     running
  8 vm12.ws.nsrc.org     running
```

```
  9 vm21.ws.nsrc.org     running
 10 vm22.ws.nsrc.org     running
  - gold.ws.nsrc.org     shut off
  - gold2.ws.nsrc.org    shut off
  - gold3.ws.nsrc.org    shut off
  - noc.ws.nsrc.org      shut off
```

Now try pinging some of the new VMs:

```
ping vm1.ws.nsrc.org
ping vm11.ws.nsrc.org
ping vm21.ws.nsrc.org
```

(If you don't have three members in your group, then just ping what is available).

Do you notice something? The DNS already has entries for each of these VMs. This is on purpose. Furthermore, the VMs all have reasonable IP addresses. How did we do this? We will explain the details at the end of this exercise.

Use ssh to connect to your two new VM images and to verify that things look OK. You can ssh directly from your laptop to the vm, or you can use the command line ssh logged in as the nsrc user on s1.ws.nsrc.org.

Once you are logged in on your new VMs let's do this:

```
ifconfig -a
ping 8.8.8.8
ping nsrc.org
ping s1.ws.nsrc.org
ping vm1.ws.nsrc.org
nmap s1.ws.nsrc.org
```

Remember we installed nmap on the gold image? That's why nmap is now installed on your VMs.

Now log out of this vm and be sure you are logged in to s1.ws.nsrc.org as the nsrc user:

4 Clone multiple VMs with a single command

Now we want you to create the following VMs depending on your gold VM image. Here are the machine you need to create:

```
gold.ws.nsrc.org ==> vm3-vm10
gold2.ws.nsrc.org ==> vm13-vm20
gold3.ws.nsrc.org ==> vm23-vm30
```

Note we are not creating the first two VMs in each range as we've already done this manually.

We can use the power of the command line and a for statement to create a programmatic loop with variables to create these VMs. For example, to create sample1.ws.nsrc.org to sample30.ws.nsrc.org from a sample.ws.nsrc.org image you could do (DON'T DO THIS!) on the command line:

```
for i in $(seq 1 1 30); do ./clonevm.sh sample.ws.nsrc.org sample$i.ws.nsrc.org; done
```

This assumes clonevm.sh is in the directory from where you are running this command. Below are the commands each user should run for each gold image to duplicate the remaining 8 VMs we would like you to create:

For the gold VM:

```
cd
cd workshop-kit/scripts
for i in $(seq 3 1 10); do ./clonevm.sh gold.ws.nsrc.org vm$i.ws.nsrc.org; done
```

For the gold2 VM:

```
cd
cd workshop-kit/scripts
for i in $(seq 13 1 20); do ./clonevm.sh gold2.ws.nsrc.org vm$i.ws.nsrc.org; done
```

For the gold3 VM:

```
cd
cd workshop-kit/scripts
for i in $(seq 23 1 30); do ./clonevm.sh gold3.ws.nsrc.org vm$i.ws.nsrc.org; done
```

Now let's see what VMs are available to run:

```
virsh list --all
```

Wow! You've just created multiple and fully functioning Linux servers using a single command.

5 Control multiple VMs from the command line

At this point starting each VM manually using "virsh start" will quickly become tedious. Let's use the same concept and do this with a single command. We'll include the first two VMs you created in case one of them is not running. It is fine to do virsh start for a machine that's already started:

For the gold VM:

```
for i in $(seq 1 1 10); do virsh start vm$i.ws.nsrc.org; done
```

For the gold2 VM:

```
for i in $(seq 11 1 20); do virsh start vm$i.ws.nsrc.org; done
```

For the gold3 VM:

```
for i in $(seq 21 1 30); do virsh start vm$i.ws.nsrc.org; done
```

Once you have all the VMs up and running do:

```
virsh list --all
```

What IP addresses do these VMs have?

```
for i in $(seq 1 1 30); do dig +short vm$i.ws.nsrc.org; done
```

Are you beginng to see how "for" works?

Try pinging each of these addresses to see if your VMs are up and running. If you don't have all 30 VMs see ify can adjust the sample command below to just ping the VMs you are using:

```
for i in $(seq 1 1 30); do ping -c 1 -W 1 vm$i.ws.nsrc.org; done
```

Remember you can scroll back in your terminal windows to see results that scroll off the visible screen.

Can you figure out how to write a single command that would shutdown each of your running VMs using the "virsh shutdown" command? Of so, do this now.

6 Optional exercise - SSH Agent forwarding

When you connect as the user nsrc to your workshop server (s1.ws.nsrc.org) you connect using your private key / public key authentication. To be able to connect to the VMs running on s1.ws.nsrc.org using your public/private key pair you need to enable SSH Agent forwarding so that your key is available for use in your session as user nsrc (or root) on s1.ws.nsrc.org.

6.1 For Linux / Unix / OS X

From the command line:

```
ssh -o ForwardAgent=yes nsrc@s1.ws.nsrc.org
```

To make this permanent:

```
cd
editor .ssh/config
```

and add these lines to the file:

```
Host *
    ForwardAgent=yes
```

But, be careful as this will forward your key to every host you connect to. You could do:

```
Host s1.ws.nsrc.org
    ForwardAgent=yes
```

If you do this, then when you are logged in on s1.ws.nsrc.org you can use keys to connect to the VMs. From a Linux / Unix / OS X laptop, however, you can just connect directly to the VMs if you wish.

6.2 For Windows

- Open putty.exe
- If you have a saved session for connection to s1.ws.nsrc.org Load this (if you don't create one now)
- In the left-hand column go down to SSH, click on the "+" symbol and select Auth

- Check the box that says “Allow agent forwarding”
- In the left-hand column scroll back to the top and click on Session
- Be sure to click “Save” to save these settings for the next time.
- Now click on Open

From now on SSH Agent forwarding is enabled for future SSH connection.

7 Use SSH on s1.ws.nsrc.org to run a command across multiple VMs

Log in on s1.ws.nsrc.org the user nsrc. Be sure you do this with SSH Agent forwarding enabled.

Now start your VMs.

For the gold VM:

```
for i in $(seq 1 1 10); do virsh start vm$i.ws.nsrc.org; done
```

For the gold2 VM:

```
for i in $(seq 11 1 20); do virsh start vm$i.ws.nsrc.org; done
```

For the gold3 VM:

```
for i in $(seq 21 1 30); do virsh start vm$i.ws.nsrc.org; done
```

Since your SSH key is available you should be able to ssh to each VM using your private key’s passphrase. But, we don’t want to type this each time so we’ll enable the ssh agent on s1.ws.nsrc.org:

```
ssh-add
```

Now try running a command on your VMs using SSH. Note, you *do not* need to log in on each VM. You can issue a command via SSH by using SSH in this form:

```
ssh user@remote-host command
```

So, to issue the command hostname on vm1.ws.nsrc.org without having to log in and log back out you can do:

```
ssh sysadm@vm1.ws.nsrc.org hostname
```

Give that a try.

Now let's do this across all 10 of your VMs for each gold group. Can you figure out how to do this without looking at the example below?

The first time you connect you'll have to accept each VM's public key, so this will not be all that impressive. Run the command twice. The second time will be much faster and give you an idea of how useful this could be when administering multiple VMs:

For the gold VM:

```
for i in $(seq 1 1 10); do ssh sysadm@vm$i.ws.nsrc.org hostname; done
```

For the gold2 VM:

```
for i in $(seq 11 1 20); do ssh sysadm@vm$i.ws.nsrc.org hostname; done
```

For the gold3 VM:

```
for i in $(seq 21 1 30); do ssh sysadm@vm$i.ws.nsrc.org hostname; done
```

Now we'll explain some of the details behind how things were built.