Jensd's I/O buffer
random technotes…

Install and use CentOS 7 or RHEL 7 as KVM virtualization host

Posted on **27/08/2014**

Tweet

When thinking about virtualization, everybody immediately thinks about VMWare. And it must be said, the product they offer is very decent but also comes with a “decent” price. As an alternative, it’s worth looking into KVM for your virtualization. As with the VMWare product range, KVM offers full virtualization and it can compete with VMWare regarding stability and performance.

Virtualization-terminology

To prevent things getting confused I would first like to clear out some terminology used for virtualization. For somebody working on a daily basis in virtual environments, these might be clear but can be rather confusing for others.

Host: the machine that hosts other system, KVM will be installed on this machine
Guest: the system running on the host, also referred to as VM, Virtual Machine or domain.
Hypervisor: the piece of software that enables virtualization on the host. For example: KVM, ESXi, Xen, …

Part1: KVM installation and preparation

KVM hypervisor and VM-extensions

As mentioned earlier, KVM offers, as VMWare, full virtualization. This means that a full system, which looks like a real physical system to the guest-OS, will be offered. Besides full virtualization, there is also such a thing as paravirtualization, as Xen can offer. Paravirtualization gives you higher performance but needs a modified guest-OS and is basically limited to *nix-systems. Full virtualization enables you to run unmodified guest-systems and thus also most proprietary systems as Windows . In order to be able to use full virtualization, you either need some virtualization-extensions on your CPU or use emulation.

First thing to do is to check if the host-machine supports VM-extensions. On the x86 platofrom, those are either AMD-V or Intel's VT-X. In order to check if the installed CPU's support those extensions, we need to check if the vmx (for VT-X) or svm (for AMD-V) flag exists in the cpuinfo-output:

```
[jensd@kvmhost ~]$ egrep -c '(vmx|svm)' /proc/cpuinfo
2
```

When the output is 0, meaning that neither vmx or svm is found in the flags, it probably means that your CPU doesn't support those extensions and there is little you can do. When the extensions are listed, be sure to check if they are enabled in the systems BIOS since that would cause problems later on. In case your CPU doesn't support VM-extensions, you are limited to QEMU-emulation in combination with KVM, which delivers a much worse performance in comparison. For this tutorial, I'll assume that the VM-extensions are supported and enabled in the BIOS of the host-system.

KVM installation

The first step in the KVM installation is installing the necessary packages. Package virt-manager, xauth and dejavu-lgc-sans-fonts are also needed if you want to manage KVM with the graphical interface in combination with X11 forwarding. (for more information, check this previous post about X11 forwarding)

To install the required packages

```
[jensd@kvmhost ~]$ sudo yum install kvm virt-manager libvirt virt-ins
...
Complete!
```

Networking

For the networking part, our KVM-host will act as a router for its guests and we will need to create a bridge interface to allow the guest to communicate out of the host. Guests will use NAT on the host to connect to the real network. To allow such type of setup it's needed to allow ip forwarding in the kernel parameters.

```
[jensd@kvmhost ~]$ echo "net.ipv4.ip_forward = 1"|sudo tee /etc/sysct
```

```
net.ipv4.ip_forward = 1
[jensd@kvmhost ~]$ sudo sysctl -p /etc/sysctl.d/99-ipforward.conf
net.ipv4.ip_forward = 1
```

After allowing the host to do ip forwarding, we need to change the network configuration. Basically we will keep our original physical interface as it is but will assign its IP-address to the brige. In the example host-machine there is one real interface called eno16777736 and the script in /etc/sysconfig/network-scripts/ifcfg-eno16777736 looks like this:

```
1 DEVICE="eno16777736"
2 ONBOOT=yes
3 IPADDR="192.168.202.111"
4 NETMASK="255.255.255.0"
5 GATEWAY="192.168.202.2"
6 HWADDR="00:0c:29:32:d0:4c"
7 DNS1="192.168.202.2"
```

The first thing to change here, is to comment out everything that is IP-related and tell the interface which interface will be the bridge. Resulting in /etc/sysconfig/network-scripts/ifcfg-eno16777736 to look like this:

```
1 DEVICE="eno16777736"
2 ONBOOT=yes
3 #IPADDR="192.168.202.111"
4 #NETMASK="255.255.255.0"
5 #GATEWAY="192.168.202.2"
6 HWADDR="00:0c:29:32:d0:4c"
7 #DNS1="192.168.202.2"
8 BRIDGE=virbr0
```

Next, we can create the config-script for the bridge interface virbr0 in /etc/sysconfig /network-scripts/ifcfg-virbr0. Most details can be copied from the original script for eno16777736:

```
1 DEVICE="virbr0"
2 TYPE=BRIDGE
3 ONBOOT=yes
4 BOOTPROTO=static
5 IPADDR="192.168.202.111"
6 NETMASK="255.255.255.0"
7 GATEWAY="192.168.202.2"
8 DNS1="192.168.202.2"
```

Finish and check the KVM installation

Basically all components are now ok but before KVM can be used it's a good idea to perform a reboot in order to load the kvm-modules and to relaod the new network settings.

After the reboot, we should check if the necessary kernel modules are loaded, which

means that KVM successfully can handle the VM-extensions of our CPU:

```
[jensd@kvmhost ~]$ lsmod|grep kvm
kvm_intel 138567 0
kvm 441119 1 kvm_intel
```

Check if the bridge is installed and in an up-state:

```
[jensd@kvmhost ~]$ ip a show virbr0
3: virbr0: &lt;BROADCAST,MULTICAST,UP,LOWER_UP&gt; mtu 1500 qdisc noq
 link/ether 00:0c:29:32:d0:4c brd ff:ff:ff:ff:ff:ff
 inet 192.168.202.111/24 brd 192.168.202.255 scope global virbr0
 valid_lft forever preferred_lft forever
 inet6 fe80::20c:29ff:fe32:d04c/64 scope link
 valid_lft forever preferred_lft forever
```

Last thing to check is if we can connect to KVM by asking for a simple list of systems:

```
[jensd@kvmhost ~]$ sudo virsh -c qemu:///system list
 Id Name State
----------------------------------------------------
```

If it returns something else, then you should go trough the earlier steps to check where something went wrong.

Part 2: Using KVM with the CLI

After completing the KVM installation, it's time to start using the host. First thing we need to do is to create a new domain or VM.

Adding a new VM

To create a new virtual machine using the CLI, we need to know which template we will use to install the system. To get a list of templates that are known in our KVM installation, you can do the following:

```
[jensd@kvmhost ~]$ virt-install --os-variant=list
 win7 : Microsoft Windows 7
 vista : Microsoft Windows Vista
 winxp64 : Microsoft Windows XP (x86_64)
 winxp : Microsoft Windows XP
 win2k : Microsoft Windows 2000
 win2k8 : Microsoft Windows Server 2008
 win2k3 : Microsoft Windows Server 2003
 openbsd4 : OpenBSD 4.x
 freebsd8 : FreeBSD 8.x
 freebsd7 : FreeBSD 7.x
 freebsd6 : FreeBSD 6.x
 solaris9 : Sun Solaris 9
 solaris10 : Sun Solaris 10
 opensolaris : Sun OpenSolaris
 netware6 : Novell Netware 6
 netware5 : Novell Netware 5
 netware4 : Novell Netware 4
 msdos : MS-DOS
 generic : Generic
 debianwheezy : Debian Wheezy
```

```
debiansqueeze : Debian Squeeze
debianlenny : Debian Lenny
debianetch : Debian Etch
fedora19 : Fedora 19
fedora18 : Fedora 18
fedora17 : Fedora 17
fedora16 : Fedora 16
fedora15 : Fedora 15
fedora14 : Fedora 14
fedora13 : Fedora 13
fedora12 : Fedora 12
fedora11 : Fedora 11
fedora10 : Fedora 10
fedora9 : Fedora 9
fedora8 : Fedora 8
fedora7 : Fedora 7
fedora6 : Fedora Core 6
fedora5 : Fedora Core 5
mageia1 : Mageia 1 and later
mes5.1 : Mandriva Enterprise Server 5.1 and later
mes5 : Mandriva Enterprise Server 5.0
mandriva2010 : Mandriva Linux 2010 and later
mandriva2009 : Mandriva Linux 2009 and earlier
rhel7 : Red Hat Enterprise Linux 7
rhel6 : Red Hat Enterprise Linux 6
rhel5.4 : Red Hat Enterprise Linux 5.4 or later
rhel5 : Red Hat Enterprise Linux 5
rhel4 : Red Hat Enterprise Linux 4
rhel3 : Red Hat Enterprise Linux 3
rhel2.1 : Red Hat Enterprise Linux 2.1
sles11 : Suse Linux Enterprise Server 11
sles10 : Suse Linux Enterprise Server
opensuse12 : openSuse 12
opensuse11 : openSuse 11
ubuntusaucy : Ubuntu 13.10 (Saucy Salamander)
ubunturaring : Ubuntu 13.04 (Raring Ringtail)
ubuntuquantal : Ubuntu 12.10 (Quantal Quetzal)
ubuntuprecise : Ubuntu 12.04 LTS (Precise Pangolin)
ubuntuoneiric : Ubuntu 11.10 (Oneiric Ocelot)
ubuntunatty : Ubuntu 11.04 (Natty Narwhal)
ubuntumaverick : Ubuntu 10.10 (Maverick Meerkat)
ubuntulucid : Ubuntu 10.04 LTS (Lucid Lynx)
ubuntukarmic : Ubuntu 9.10 (Karmic Koala)
ubuntujaunty : Ubuntu 9.04 (Jaunty Jackalope)
ubuntuintrepid : Ubuntu 8.10 (Intrepid Ibex)
ubuntuhardy : Ubuntu 8.04 LTS (Hardy Heron)
virtio26 : Generic 2.6.25 or later kernel with virtio
generic26 : Generic 2.6.x kernel
generic24 : Generic 2.4.x kernel
```

Virtual disk images for the KVM-guests can be placed in /var/lib/libvirt by default. In case you prefer to use another location to store the disk images, SELinux will, by default, prevent access and the security context of that location needs to be changed in order to use it for KVM. To change the SELinux context when storing the images in another location (/vm for example):

```
[jensd@kvmhost ~]$ sudo semanage fcontext -a -t virt_image_t "/vm(/.*
 [jensd@kvmhost ~]$ sudo restorecon -R /vm
```

Now, to add a new VM, we can use virt-install.

Example to add a windows-guest:

```
[jensd@kvmhost ~]$ sudo virt-install --connect qemu:///system -n vmwi
Starting install...
Allocating 'vmwin7.img' | 10 GB 00:00:00
Creating domain... | 0 B 00:00:00
Domain installation still in progress. Waiting for installation to co
```

Explanation of the arguments that were given to virt-install:

- –connect qemu:///system : connect to KVM on the local system, we could also connect to another KVM-host and define our new VM there
- -n vmwin7 : name of the new VM: vmwin7
- -r 1024 : amount of memory for the VM: 1GB
- –vcpus=2 : amount of virtual CPU's for the VM: 2
- –disk path=/var/lib/libvirt/images/vmwin7.img,size=10 : where to store the virtual disk image of the VM and the size: 10GB
- –graphics vnc,listen=0.0.0.0 : how to display the VM's console: via VNC accessible from outside
- –noautoconsole : do not automatically connect to the console
- –os-type windows –os-variant win7 : type of guest OS (from the list given above)
- –accelerate : use KVM HW-acceleration
- –network=bridge:virbr0 : network bridge to use
- –hvm : full virtualisation
- –cdrom /var/X17-59186.iso : location of the installation ISO

After launching the above command, you should be able to connect with VNC to the host and get on the console-display of the VM. The console displays what would normally, on a physical machine, appear on the attached monitor.

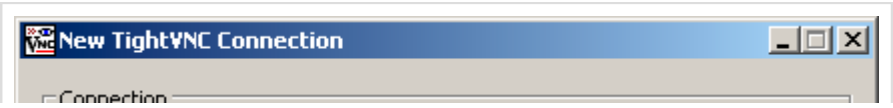
By default, VNC will use the first available screen on port 5900. To be sure which screen is used, we can use virsh to show the attached console-screens for VNC:

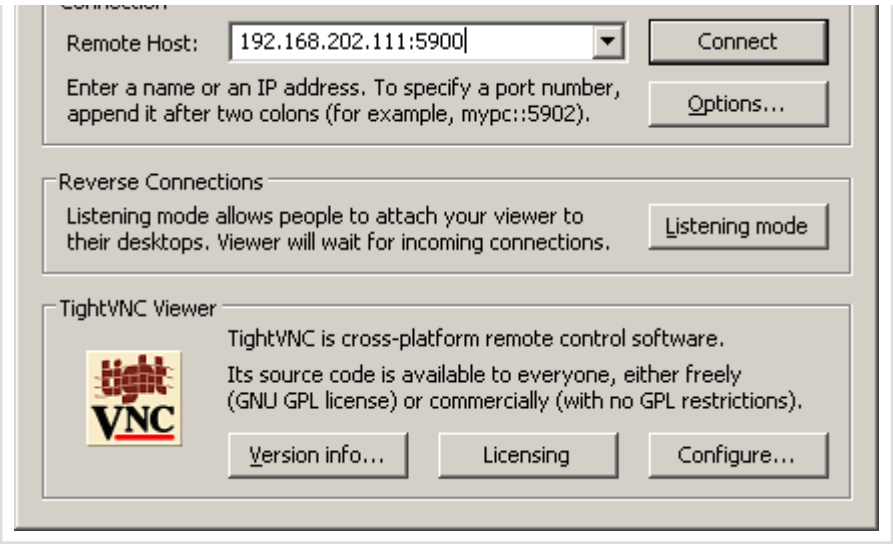
```
[jensd@kvmhost ~]$ sudo virsh vncdisplay vmwin7
:0
```

:0 means the first screen and real port 5900 as you can also see when checking with netstat which ports are currently listening:

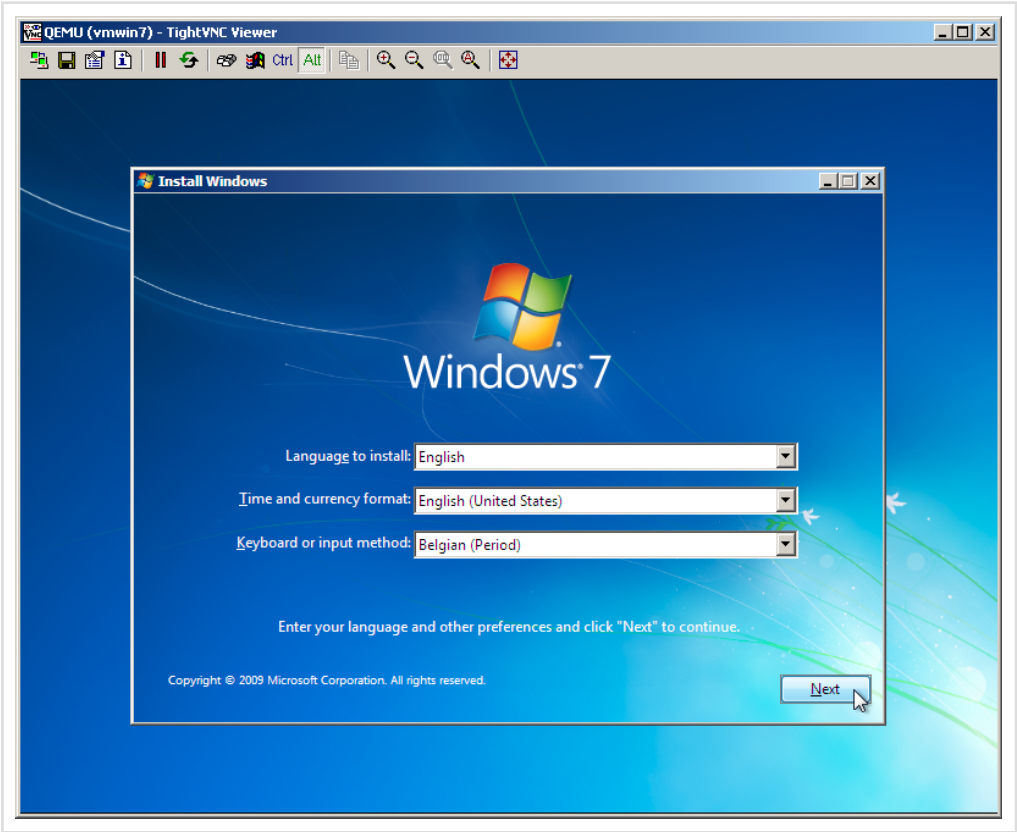
```
[jensd@kvmhost ~]$ netstat -tln|grep :59
tcp 0 0 0.0.0.0:5900 0.0.0.0:* LISTEN
```

Now, connect to the KVM-host with a VNC viewer. I'm using TightVNC but every VNC viewer should do:

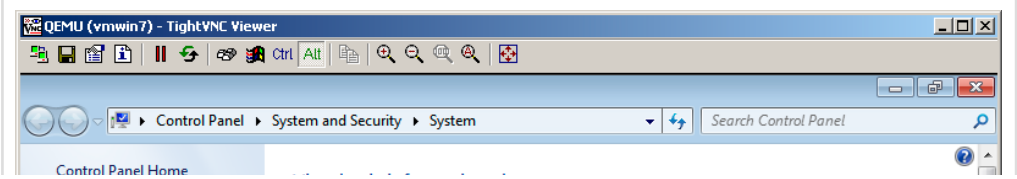


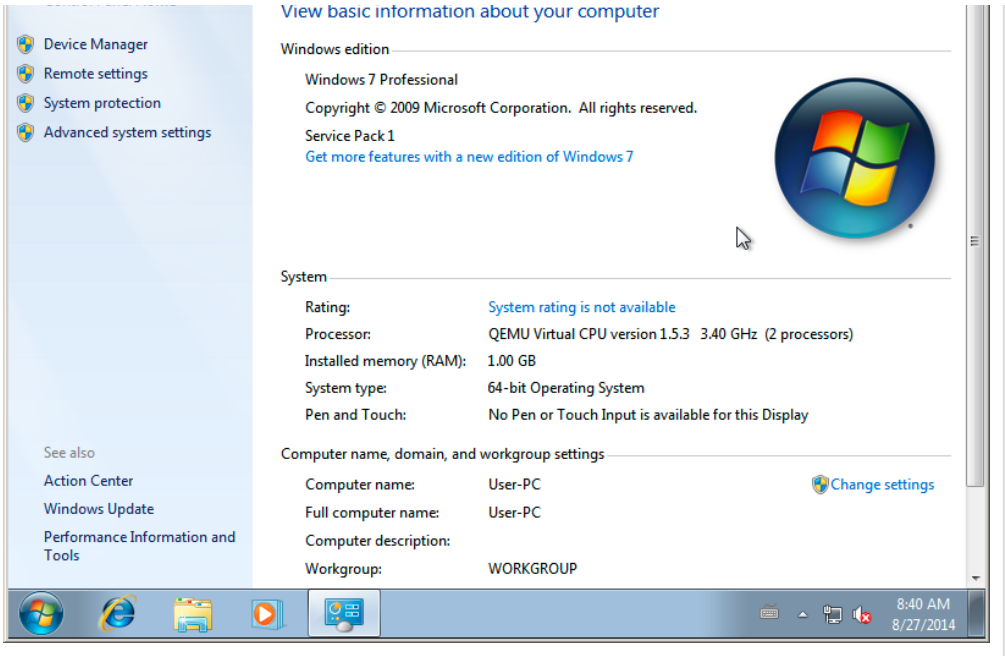


From this point, we can complete the windows installation as if it would be a normal physical system:

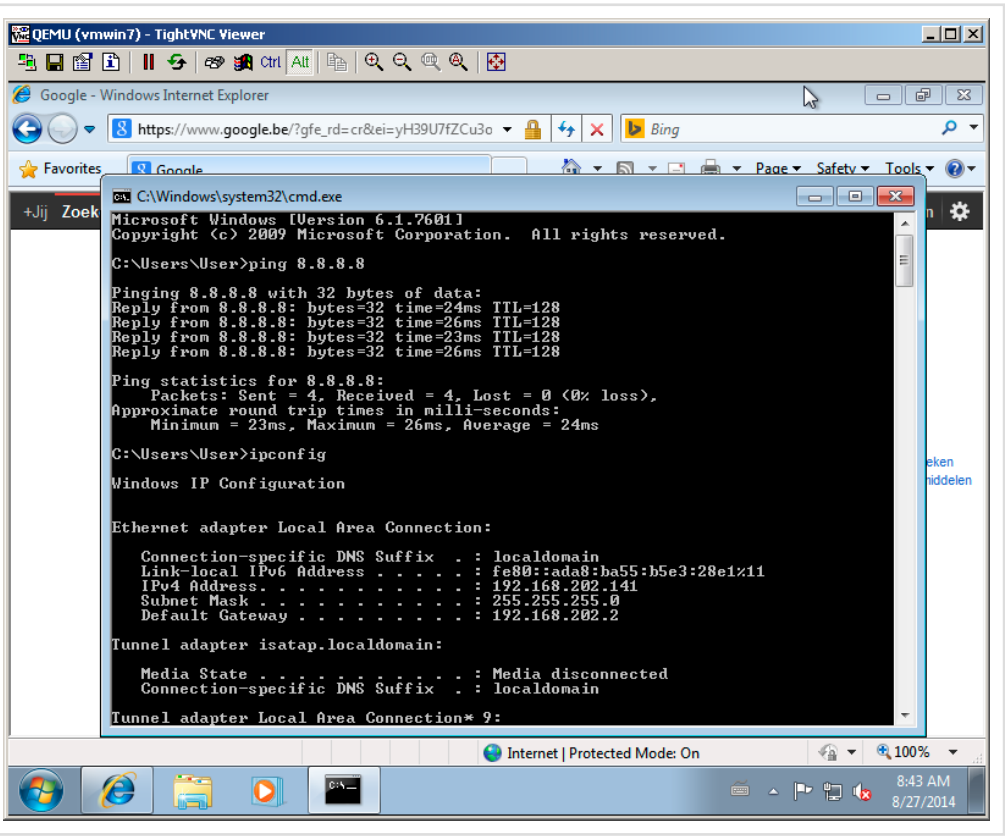


After completing the installation with VNC, we end up with a Windows-VM that is running on our KVM-host:





As for the networking part, we use the earlier created bridge (virbr0) to do NAT. This means that the KVM-host NAT's all our connections to the real network connected to the KVM-host. If DHCP is active on that network, it can be used in the VM. Otherwise you will have to configure a static IP in the same subnet.



Example to add a Linux-guest:

To add a Linux guest, next to the already added Windows-guest is quite similar:

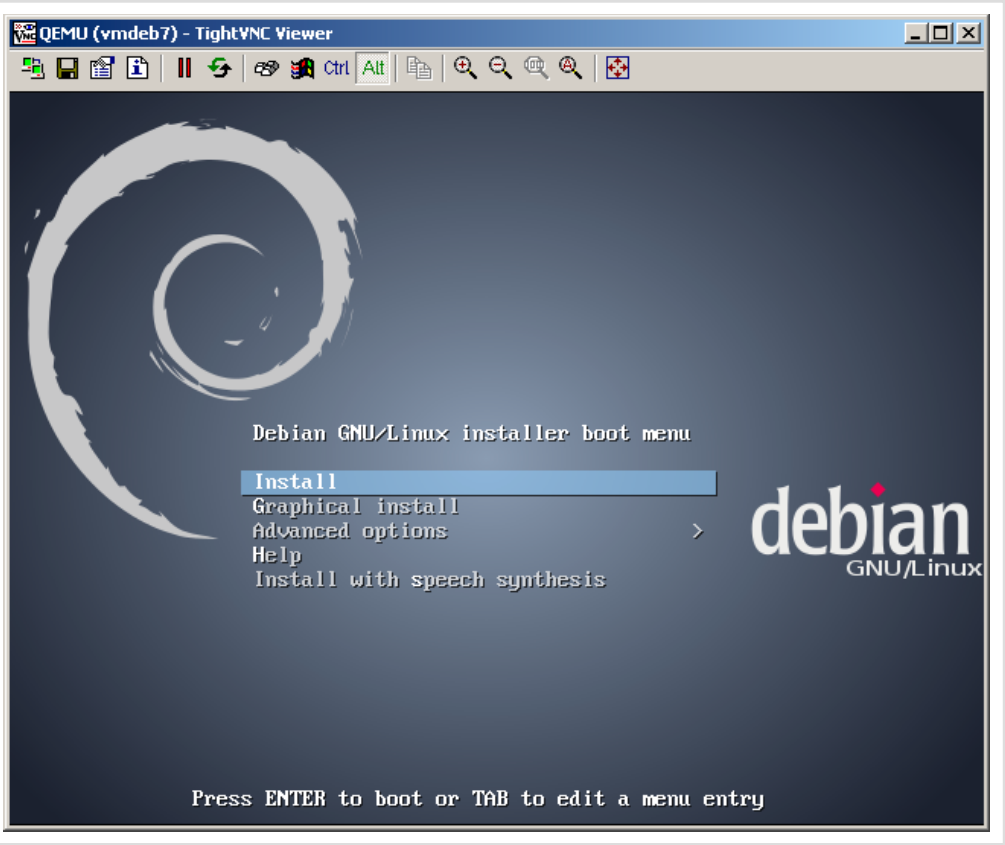
```
[jensd@kvmhost ~]$ sudo virt-install --connect qemu:///system -n vmde
 Starting install...
Allocating 'vmdeb7.img' | 2.0 GB 00:00:00
Creating domain... | 0 B 00:00:02
Domain installation still in progress. You can reconnect to the conso
```

As with the Windows-VM, after launching this command, you should be able to connect with VNC to the host and get on the console of the VM to complete the Debian installation.

To know which VNC-display number (and port) is used for a certain VM, the same command as used earlier should do:

```
[jensd@kvmhost ~]$ sudo virsh vncdisplay vmdeb7
 :1
```

Above command gives :1 as result, meaning that the guest vmdeb7 can be contacted with VNC on port 5901:



After finishing the installation, we end up with a Linux guest running on top of our KVM-host. Which Linux distro we are using doesn't matter since we're doing full virtualization.

```
root@deb:~# uname -a
Linux deb 3.2.0-4-amd64 #1 SMP Debian 3.2.60-1+deb7u3 x86_64 GNU/Linu
root@deb:~# cat /proc/cpuinfo |grep model
model : 13
model name : QEMU Virtual CPU version 1.5.3
```

Considering network, the same as with the Windows VM applies here. Our connections are NATted trough the KVM-host and we can use the DHCP-server of our real network.

```
root@deb:~# ping -c1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=128 time=23.8 ms
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 23.855/23.855/23.855/0.000 ms
root@deb:~# ip a show eth0
2: eth0: &lt;BROADCAST,MULTICAST,UP,LOWER_UP&gt; mtu 1500 qdisc pfifo
link/ether 52:54:00:33:65:75 brd ff:ff:ff:ff:ff:ff
inet 192.168.202.140/24 brd 192.168.202.255 scope global eth0
inet6 fe80::5054:ff:fe33:6575/64 scope link
valid_lft forever preferred_lft forever
```

More KVM actions

Besides creating VM's, it's a good thing to know some basic operations regarding VM-managent.

List the active virtual machines:

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system list
 Id Name State
----------------------------------------------------
 7 vmwin7 running
 8 vmdeb7 running
```

Get more information about a guest:

```
[jensd@kvmhost ~]$ sudo virsh dominfo vmwin7
Id: 7
Name: vmwin7
UUID: f913c6fa-b597-437d-b6f5-797314e34847
OS Type: hvm
State: running
CPU(s): 2
CPU time: 20955.1s
Max memory: 1048576 KiB
Used memory: 1048576 KiB
Persistent: yes
Autostart: disable
Managed save: no
Security model: selinux
Security DOI: 0
Security label: system_u:system_r:svirt_t:s0:c638,c926 (enforcing)
```

Stop a running guest:

To stop a running VM in a clean way (as you would press the power button to start the shutdown sequence):

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system shutdown vmdeb
Domain vmdeb7 is being shutdown
```

This triggers a normal, clean, shutdown on the guest:

```
root@deb:~#
Broadcast message from root@deb (Wed Aug 27 09:09:16 2014):
Power button pressed
The system is going down for system halt NOW!
```

To force stop a running VM that doesn't want to shutdown in a clean way:

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system destroy vmdeb7
Domain vmdeb7 destroyed
```

Start a guest:

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system start vmdeb7
Domain vmdeb7 started
```

Delete a guest:

First we need to make sure that the guest is stopped before it can be deleted. In case you don't want the virtual disk image anymore either, you'll have to delete it manually after undefining the guest.

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system destroy vmcen6
Domain vmcen6 destroyed
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system undefine vmcen
Domain vmcen6 has been undefined
[jensd@kvmhost ~]$ sudo rm -f /var/lib/libvirt/images/vmcen6.img
```

After removing a disk-image, it's a good thing to refresh the storage pool of KVM:

```
[jensd@kvmhost ~]$ sudo virsh pool-refresh default
 Pool default refreshed
```

Automatically let a guest start when the host starts

When rebooting your host, you probably want some or all the guests that are defined on that host to start at the same time. By default, the guest are not automatically started.

```
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system autostart vmde
Domain vmdeb7 marked as autostarted
[jensd@kvmhost ~]$ sudo virsh --connect qemu:///system dominfo vmdeb7
Autostart: enable
```

Part 3: Using KVM with the virt-manager GUI

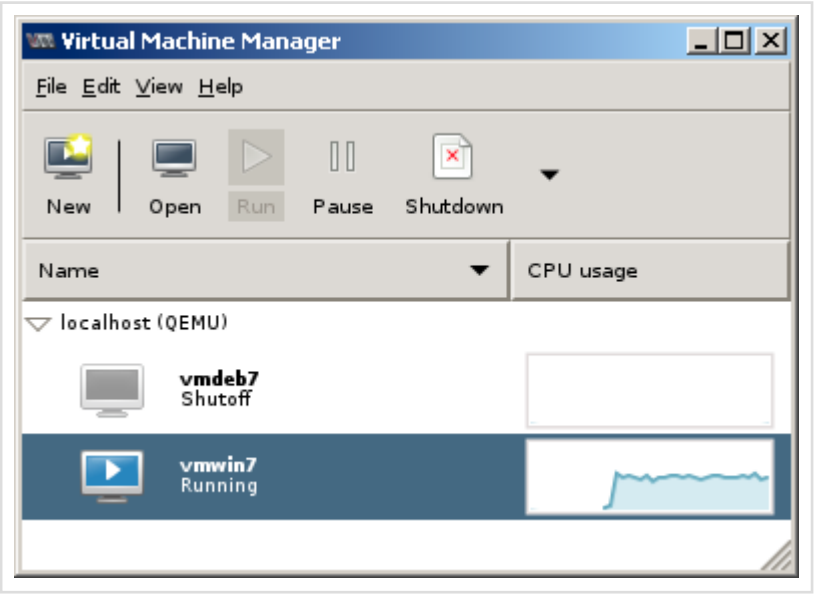
Starting the GUI

Managing KVM with the CLI is not so difficult and it can be very handy to script certain day-to-day tasks. Sometimes, you just need to keep an overview and require a little more user-friendliness. For that, you can use virt-manager, which is a graphical interface for libvirt and is mainly built for KVM. When you want to manage your guest with virt-manager, you can either do it on the host itself, by starting an X-server locally or use X11 forwarding on a headless server (more information here).

Make sure that you have enough permissions to use virt-manager and simply execute virt-manager from the command line:

```
[root@kvmhost ~]# virt-manager
```

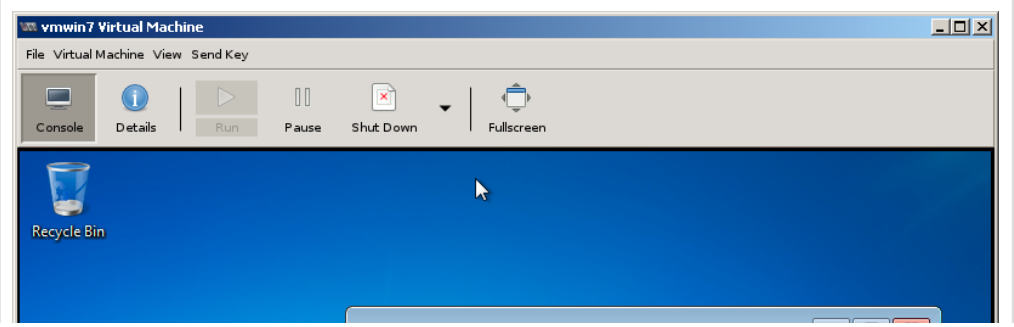
If all goes well, you should be presented with the virt-manager GUI:

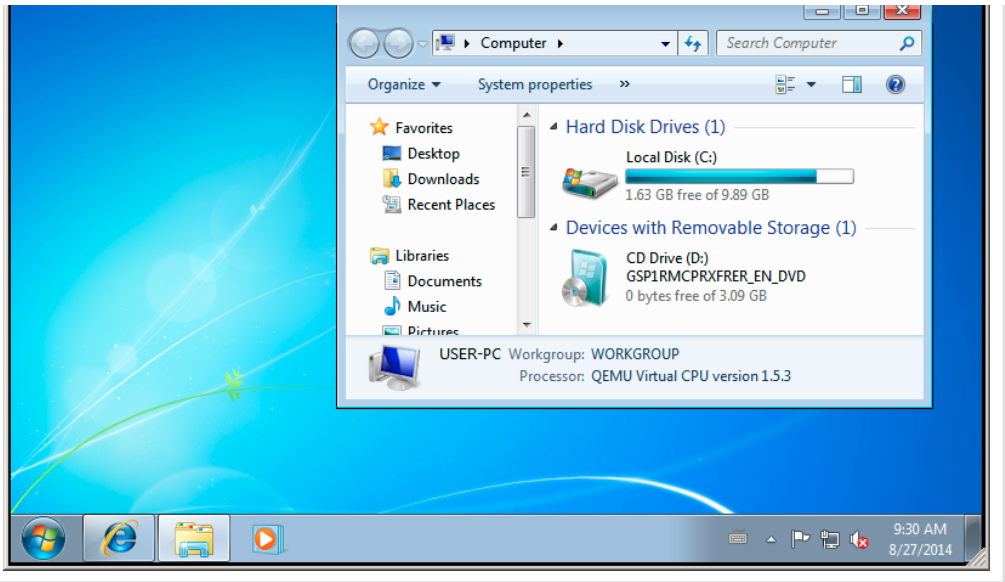


Basic actions

From the initial start-up screen, you can immediately see a list of configured guests on this host and take actions on them like: Run, Pause, Shutdown, Reboot, Force off,…

When selecting a guest, you can also click on Open to display the console as we did earlier using VNC:





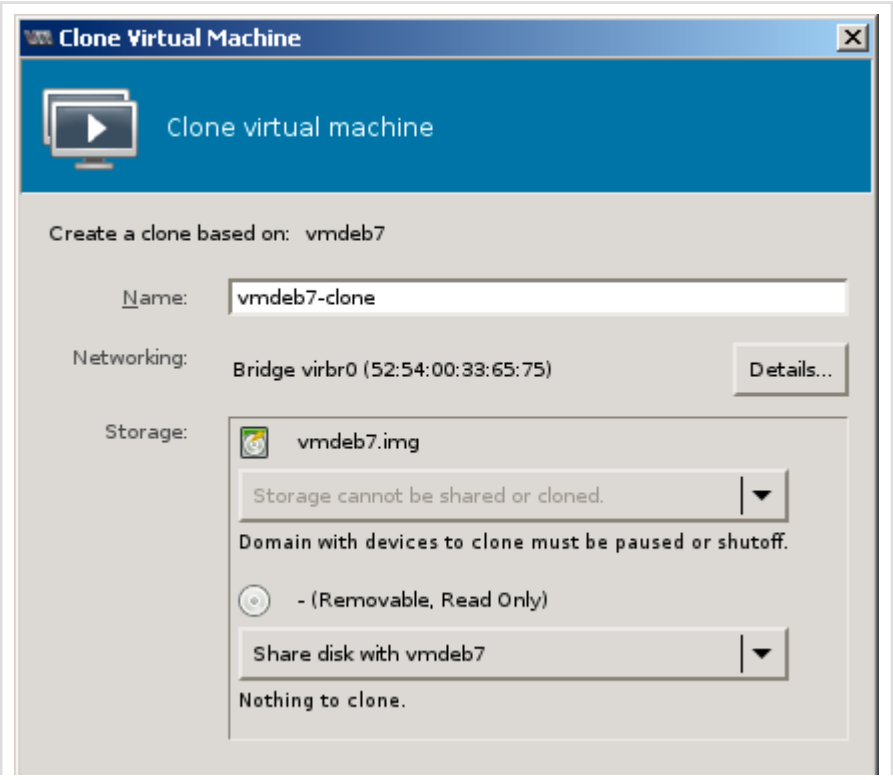
Advanced actions

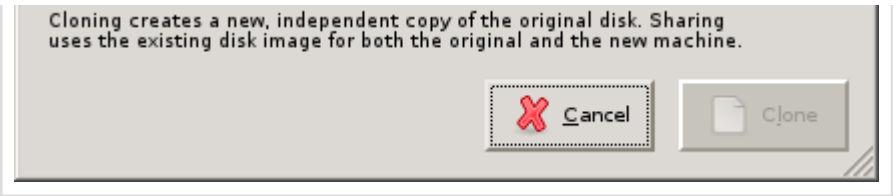
Other possibilities using the virt-manager interface:

Connect to another host-system to manage the VM's running there, using File -> Add connection (like using the –connect on the CLI)

Migrate a VM to another KVM-host: right click on the VM and choose Migrate…

Clone a VM to the same or another KVM-host: right click on the VM and choose Clone…





As you can see, the virt-manager interface is not very complicated and most of the basic tasks don't need any explanation.

After completing all of the above steps, basic installation and using KVM shouldn't have any secrets anymore for you. The next thing to do is experiment and test a little more with KVM and hopefully start to use it in your production environment.

This entry was posted in **CentOS**, **Kernel**, **KVM**, **Linux**, **Red Hat**, **RHEL**, **Virtualization**, **Windows** by **jensd**. Bookmark the **permalink [http://jensd.be/207/linux/install-and-use-centos-7-as-kvm-virtualization-host]** .

29 THOUGHTS ON "INSTALL AND USE CENTOS 7 OR RHEL 7 AS KVM VIRTUALIZATION HOST"

Hugh
on **26/09/2014 at 11:13** said:

Hi there,

I'm trying to install CentOS 6.5

virt-install
–name=vm1
–disk path=/var/lib/libvirt/images/vm1.img,size=50
–ram=8192
–os-type=linux
–os-variant=rhel6
–network bridge:br0
–nographics
–cdrom=/tmp/CentOS-6.5-x86_64-minimal.iso

It fails with an error "ERROR internal error: process exited while connecting to monitor: qemu-kvm: -chardev pty,id=charserial0: Failed to create chardev"

Thanks

jensd
on **26/09/2014 at 13:37** said:

Hi,

Difficult to tell you exactly what's wrong.
I suspect a authorization problem.

You can try to (temporary) turn of SELinux to make sure that isn't causing the issue.

Next, you could try to add/replace the following line in /etc/fstab:
devpts /dev/pts devpts gid=5,mode=620 0 0

Could you send me the contents of /etc/libvirt/qemu.conf
and the output of the following command: ls -al /var/lib/libvirt/qemu

Suleman
on **23/10/2014 at 21:52** said:

Running into an odd issue. Wonder if you had an answer for it. The libvirtd adds 4 lines to the head of the IP Chain. It is supposed to limit it to the bridge by default according to their documentation. However, it does not limit it for me. Did you come across anything like this?

Documentation:
ACCEPT udp — virbr0 * 0.0.0.0/0 0.0.0.0/0 udp dpt:53
ACCEPT tcp — virbr0 * 0.0.0.0/0 0.0.0.0/0 tcp dpt:53
ACCEPT udp — virbr0 * 0.0.0.0/0 0.0.0.0/0 udp dpt:67
ACCEPT tcp — virbr0 * 0.0.0.0/0 0.0.0.0/0 tcp dpt:67

Installation:
ACCEPT udp — 0.0.0.0/0 0.0.0.0/0 udp dpt:53
ACCEPT tcp — 0.0.0.0/0 0.0.0.0/0 tcp dpt:53
ACCEPT udp — 0.0.0.0/0 0.0.0.0/0 udp dpt:67

ACCEPT tcp — 0.0.0.0/0 0.0.0.0/0 tcp dpt:67

Jorge
on **05/11/2014 at 15:58** said:

Hi

I try to connect with vnc but does not work, with vm-manager dont have any issue.

[root@hostX ~]# netstat -tln|grep :59
tcp 0 0 127.0.0.1:5900 0.0.0.0:* LISTEN
[root@hostX ~]#

[root@hostX ~]# virsh vncdisplay vm-test
127.0.0.1:0
[root@hostX ~]#

[root@hostX ~]# systemctl status iptables
iptables.service – IPv4 firewall with iptables
Loaded: loaded (/usr/lib/systemd/system/iptables.service; disabled)
Active: inactive (dead)
nov 05 09:47:44 hostX.local.inet systemd[1]: Stopped IPv4 firewall with iptables.
[root@hostX ~]#

Jorge
on **05/11/2014 at 21:48** said:

I found the issue, in the domain libvirt XML config. the the graphics type, listen had ‘127.0.0.1’, so I changed to:

Jorge
on **05/11/2014 at 21:49** said:

graphics type=’vnc’ port=’-1′ autoport=’yes’ listen=’0.0.0.0′
listen type=’address’ address=’0.0.0.0′

graphics

Ton
on **20/11/2014 at 15:28** said:

Where is domain libvirt XML config file located?

jensd
on **20/11/2014 at 15:33** said:

At this moment, I don't have access to a KVM host but from my head it's located in: /etc/libvirt/qemu/

Jack
on **06/11/2014 at 15:48** said:

Hi, I modyfied network scripts as U show but after reboot i can't ping outside (GW or DNS) , ip a show virbr0 returns "state DOWN"

Pingback: December 2014 Meeting Notes | South Orlando Linux User Group

Alain
on **23/02/2015 at 16:41** said:

Hi.

I would like to make a windows-guest running on a host system and display there too on this video card.
Thus, a hardware-independent windows should be generated in which applications I can use my usual way on the local computer and this video output …
The auto-start of the VM works, but I want to the video output also starts in

full screen mode, so I no longer see the ubuntu.
Terminating of the virt.maschine should also stop the linux …
Is it that possible?
Thanks for any response and suggestion.
Alain.

jensd
on **23/02/2015 at 17:46** said:

I think what you're looking for is VGA passthrough: https://wiki.debian.org/VGAPassthrough

Alain
on **24/02/2015 at 21:22** said:

Hi jensd.
This is a solution, but do I need a second VGA card and I can not install them in the industrial-PC's. Is this possible with only a VGA?
The goal is to install our standard windows-os with programs and sysprep on computers with different hardware without driver installation information.
For that I want to be a linux system that is not hardware sensitive, operating as kvm host and display on the same graphics card, the guest system – such as in virt-manager full screen mode…?
It's just a windows guest system on the computer needed, and the linux is not to be reached by the end user.
Computer-Start = VM-Guest-Start
Guest-Shutdown = Host-Shutdown

navthink
on **06/04/2015 at 04:47** said:

I tried
[root@localhost ~]# sysctl -p /etc/sysctl.d/99-ipforward.conf

but my Centos7 told me that

sysctl: cannot open “/etc/sysctl.d/99-ipforward.conf”: No such file or directory

What can I do?

jensd
on **07/04/2015 at 13:14** said:

something went wrong with the preceding command (echo “net.ipv4.ip_forward = 1″|sudo tee /etc/sysctl.d/99-ipforward.conf). You could try to run it again or manually create the file /etc/sysctl.d /99-ipforward.conf containing net.ipv4.ip_forward = 1.

hos7ein
on **12/04/2015 at 08:10** said:

hi
for enable ipforward on centos 7 do it :

#nano /etc/sysctl.conf

then Add this line to above file :

net.ipv4.ip_forward = 1

then save file and run this commend :

sysctl -p /etc/sysctl.conf

Phil
on **30/06/2015 at 00:26** said:

If using ‘virt-install –os-variant=list’ to list available os tempates fails as it did for me try using ‘osinfo-query os’

Mark
on **23/08/2015 at 21:13** said:

You can use osinfo-query os instead, I'm running KVM on

[root@kvm ~]# cat /etc/*-release
CentOS Linux release 7.1.1503 (Core)

Harold
on **14/10/2015 at 03:16** said:

Excellent, very informative article. Thankyou. Essential for CentOS-7 users.

Just a few typos could be corrected:
> since that would problems later on.

Insert "cause".

> act as a router for it's guests

Change "it's" to "its". "Its" is the possessive pronoun (cf hers, ours, yours, theirs – none have apostrophes).

> assign it's IP-address

Change "it's" to "its".

> appear on the attache monitor.

"Attached".

> it can be ver handy

"Very".

Thanks again!

jensd

on **23/10/2015 at 09:53** said:

Thanks for your time spent on pointing this out :)
English isn't my native language (excuse alert) and I tend to try to type faster than I can :)

I've corrected your remarks in the text.

Gryd3
on **23/10/2015 at 23:57** said:

Can we please make a couple corrections?
"Guests will use NAT on the host to connect to the real network."

This is only the case if the –network switch is set to user or a named virtual network. If you use 'bridge' and connect to a bridged interface, then the guest will have transparent access to the hosts network and will NOT be behind a NAT created by the host. The host will only be a 'router' offering NAT if the –network switch is set to user or a named virtual network.
Please see the networking section here : http://linux.die.net/man/1/virt-install

Bridged networking is incredibly easy to use, and does not require the entry of NAT rules or port forwards in the host to the guest as the other two guest network types do.

Please note that although you 'can' add more than one physical network adaptor to a bridge, you run the risk of creating a loop in the network if both network adaptors are plugged into the same or related network switches. This should only be done if you understand how to use spanning tree or link aggregation to prevent network storms and problems… this is a really easy way to kill an improperly setup network.

geocrasher
on **18/04/2016 at 05:53** said:

This is a very good point. I also had trouble with

virt-install –os-variant=list

not working, and had to use

osinfo-query os

Which worked much better. Otherwise, thanks for the good tutorial!

Marco Varanda
on **24/11/2015 at 20:20** said:

Thanks !

It will help me a lot

;-)

Manuel
on **29/11/2015 at 21:04** said:

I've followed the directions here exactly and everything works as expected. However, my knowledge (not ability to understand) falls short when it comes time to set up the IP for the guest machine, inside the guest machine.

I have a host server running bare CentOS7 and I've installed a guest OS of ClearOS7. I connected from my home machine to the Guest server VNC connection as outlined with no problem. I am able to see the ClearOS GUI and complete the initial install.
The Guest server indicates it has a Link but no IP assigned so I assume I am to use the IP provided for virb0. Of course the guest system says IP already in use.

What I need to accomplish is this:
1) I need the host server to be able to be accessed from pubic (internet) via SSH so I can manage and maintain it. I do not necessarily need VNC, HTTP, or anything else.
2) I need the Guest OS to be able to server up HTTP, SSH and other service like a normal server and be able to get out to the internet to surf, download

content etc.
3) I will be adding another Guest Server and it will also need the same access as the other guests.

I have only one IP from my ISP.
IP: 199.x.x.10
SNM: 255.255.224.0
DG: 199.x.x.1

Any help is appreciated.
Thanks

Michael Cooper
on **04/01/2016 at 15:40** said:

Hello New to this style of Virtualization

I followed your article for X11Forwarding (http://jensd.be/68/linux/remote-graphical-linux-applications-on-linux-and-windows) then I followed this (http://jensd.be/207/linux/install-and-use-centos-7-as-kvm-virtualization-host) to install KVM on CentOS 7. When I try to start virt-manager I get the following:

[root@cfkvm1 ~]# virt-manager
[root@cfkvm1 ~]#
** (virt-manager:11657): WARNING **: Could not open X display

(virt-manager:11657): Gtk-CRITICAL **: gtk_settings_get_for_screen: assertion ‘GDK_IS_SCREEN (screen)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_settings_get_style_cascade: assertion ‘GTK_IS_SETTINGS (settings)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_style_provider_private_lookup: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_lookup_resolve: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba:

assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_style_provider_private_lookup: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_lookup_resolve: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_style_provider_private_lookup: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_lookup_resolve: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_style_provider_private_lookup: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_lookup_resolve: assertion ‘GTK_IS_STYLE_PROVIDER_PRIVATE (provider)’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

(virt-manager:11657): Gtk-CRITICAL **: _gtk_css_rgba_value_get_rgba: assertion ‘rgba->class == &GTK_CSS_VALUE_RGBA’ failed

So I am not sure what the real issue is, I need some help to get this going.

Thanks,

Jo
on **08/01/2016 at 23:12** said:

Gotta say that this walk through helped me a lot.
Got me up and running quickly and started using my environments.

Thanks for this.

Ivan
on **18/03/2016 at 01:43** said:

Você esta executando no putty?
Pois não é possível, somente logado na maquina em modo gráfico.

Ivan
on **18/03/2016 at 01:45** said:

1- logar na parte gráfica e chamar o console e apos isso executar o comando virt-manager…….
At+

davei
on **16/04/2016 at 03:57** said:

2
3
4
5
6
7
8
DEVICE=”virbr0″
TYPE=BRIDGE
ONBOOT=yes

BOOTPROTO=static
IPADDR=”192.168.202.111″
NETMASK=”255.255.255.0″
GATEWAY=”192.168.202.2″
DNS1=”192.168.202.2″

should be

2
3
4
5
6
7
8
DEVICE=”virbr0″
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
IPADDR=”192.168.202.111″
NETMASK=”255.255.255.0″
GATEWAY=”192.168.202.2″
DNS1=”192.168.202.2″