KVM Hypervisor on Red Hat / CentOS 8.x

KVM is an open source hardware virtualization solution which can be used to run several Linux-based or Windows-based systems in parallel on one host.

KVM is known as a so-called "**Kernel based Virtual Machine**". This is because after installing the package, the KVM module is loaded with the kernel at the next boot, turning a *normal Linux server* into a bare metal hypervisor.

Now I will describe how to set up such a KVM Hypervisor and how to manage it afterwards.

- Configure KVM to suspend/restore virtual machines on host reboot
- OLD - KVM Installation on CentOS / Red Hat 7.x

Before we start, verify support for Virtual Technology of the CPU by issueing the following command:

```
# lscpu | grep Virtualization
```

```
Virtualization:      VT-x
Virtualization type: full
```

The output of the above command shows that, our server kvm-virtualization-01.recipes.com supports Virtualization.

However, if the above command returns no result on your server then,

1. In case of bare-metal machine, you have to enable the VT support from system BIOS.
2. In case of virtual machine, you have to enable the VT support from VM's CPU Settings.

Installing KVM and QEMU

In CentOS 8 / RHEL 8, virtualization components including KVM and QEMU hypervisors are bundled in virt module. Therefore, it is really simple now to configure a KVM virtualization host in CentOS 8.

We are installing virt module using dnf command.

```
# dnf install -y @virt
```

```
...
================================================================================
====
Installed:
  libguestfs-1:1.38.4-11.1.module_el8.0.0+189+f9babebb.x86_64
  libvirt-client-4.5.0-24.3.module_el8.0.0+189+f9babebb.x86_64
  libvirt-daemon-config-
network-4.5.0-24.3.module_el8.0.0+189+f9babebb.x86_64
  libvirt-daemon-kvm-4.5.0-24.3.module_el8.0.0+189+f9babebb.x86_64
  alsa-lib-1.1.6-3.el8.x86_64
  autogen-libopts-5.18.12-7.el8.x86_64
  boost-atomic-1.66.0-6.el8.x86_64
  boost-chrono-1.66.0-6.el8.x86_64
  boost-date-time-1.66.0-6.el8.x86_64
  boost-iostreams-1.66.0-6.el8.x86_64
  boost-program-options-1.66.0-6.el8.x86_64
  boost-random-1.66.0-6.el8.x86_64
  boost-regex-1.66.0-6.el8.x86_64
  boost-system-1.66.0-6.el8.x86_64
  boost-thread-1.66.0-6.el8.x86_64
  cairo-1.15.12-3.el8.x86_64
  celt051-0.5.1.3-15.el8.x86_64
  dnsmasq-2.79-4.el8.x86_64
  edk2-ovmf-20180508gitee3198e672e2-9.el8_0.1.noarch
  fribidi-1.0.4-6.el8.x86_64
  genisoimage-1.1.11-39.el8.x86_64
  glusterfs-api-3.12.2-40.2.el8.x86_64
  glusterfs-cli-3.12.2-40.2.el8.x86_64
  gnutls-dane-3.6.5-2.el8.x86_64
  gnutls-utils-3.6.5-2.el8.x86_64
  graphite2-1.3.10-10.el8.x86_64
  gstreamer1-1.14.0-3.el8.x86_64
  gstreamer1-plugins-base-1.14.0-4.el8.x86_64
  harfbuzz-1.7.5-3.el8.x86_64
  hivex-1.3.15-7.module_el8.0.0+189+f9babebb.x86_64
  ipxe-roms-qemu-20181214-1.git133f4c47.el8.noarch
  iso-codes-3.79-2.el8.noarch
  libX11-1.6.7-1.el8.x86_64
  libX11-common-1.6.7-1.el8.noarch
  libX11-xcb-1.6.7-1.el8.x86_64
  libXau-1.0.8-13.el8.x86_64
...
```

We are also installing virt-install package, because it provides some very useful command line tools.

```
# dnf install -y virt-install
```

```
...
```

```
================================================================================
====
Installing:
 virt-install        noarch 2.0.0-5.1.el8                       AppStream
100 k
Installing dependencies:
 libosinfo           x86_64 1.2.0-5.el8                         AppStream
244 k
 osinfo-db           noarch 20181011-8.el8_0.1                  AppStream
172 k
 osinfo-db-tools     x86_64 1.2.0-1.el8                         AppStream
90 k
 python3-libvirt     x86_64 4.5.0-2.module_el8.0.0+189+f9babebb AppStream
291 k
 virt-manager-common noarch 2.0.0-5.1.el8                       AppStream
921 k
 python3-chardet     noarch 3.0.4-7.el8                         BaseOS
195 k
 python3-pysocks     noarch 1.6.8-3.el8                         BaseOS
34 k
 python3-requests    noarch 2.20.0-1.el8                        BaseOS
123 k
 python3-urllib3     noarch 1.23-5.el8                          BaseOS
178 k

Transaction Summary
================================================================================
====
Install  10 Packages
...
```

Validate all the components on your KVM host can support virtualization.

```
# virt-host-validate
```

```
  QEMU: Checking for hardware virtualization
: PASS
  QEMU: Checking if device /dev/kvm exists
: PASS
  QEMU: Checking if device /dev/kvm is accessible
: PASS
  QEMU: Checking if device /dev/vhost-net exists
: PASS
  QEMU: Checking if device /dev/net/tun exists
: PASS
  QEMU: Checking for cgroup 'memory' controller support
: PASS
  QEMU: Checking for cgroup 'memory' controller mount-point
```

```
: PASS
  QEMU: Checking for cgroup 'cpu' controller support
: PASS
  QEMU: Checking for cgroup 'cpu' controller mount-point
: PASS
  QEMU: Checking for cgroup 'cpuacct' controller support
: PASS
  QEMU: Checking for cgroup 'cpuacct' controller mount-point
: PASS
  QEMU: Checking for cgroup 'cpuset' controller support
: PASS
  QEMU: Checking for cgroup 'cpuset' controller mount-point
: PASS
  QEMU: Checking for cgroup 'devices' controller support
: PASS
  QEMU: Checking for cgroup 'devices' controller mount-point
: PASS
  QEMU: Checking for cgroup 'blkio' controller support
: PASS
  QEMU: Checking for cgroup 'blkio' controller mount-point
: PASS
  QEMU: Checking for device assignment IOMMU support
: PASS
  QEMU: Checking if IOMMU is enabled by kernel
: WARN (IOMMU appears to be disabled in kernel. Add intel_iommu=on to kernel
cmdline arguments)
```

It looks like **`IOMMU`** `(input-output memory management unit)` support is not yet enabled in the CentOS 8 Kernel.

The solution is already suggested by the above command. Therefore, we are adding the same in the Kernel command line options.

```
# grub2-editenv - set "$(grub2-editenv - list | grep kernelopts)
intel_iommu=on"
```

To take effect, restart your machine!

After reboot, again run the `virt-host-validate` command.

```
  QEMU: Checking for hardware virtualization
: PASS
  QEMU: Checking if device /dev/kvm exists
: PASS
  QEMU: Checking if device /dev/kvm is accessible
: PASS
  QEMU: Checking if device /dev/vhost-net exists
: PASS
  QEMU: Checking if device /dev/net/tun exists
```

```
: PASS
  QEMU: Checking for cgroup 'memory' controller support
: PASS
  QEMU: Checking for cgroup 'memory' controller mount-point
: PASS
  QEMU: Checking for cgroup 'cpu' controller support
: PASS
  QEMU: Checking for cgroup 'cpu' controller mount-point
: PASS
  QEMU: Checking for cgroup 'cpuacct' controller support
: PASS
  QEMU: Checking for cgroup 'cpuacct' controller mount-point
: PASS
  QEMU: Checking for cgroup 'cpuset' controller support
: PASS
  QEMU: Checking for cgroup 'cpuset' controller mount-point
: PASS
  QEMU: Checking for cgroup 'devices' controller support
: PASS
  QEMU: Checking for cgroup 'devices' controller mount-point
: PASS
  QEMU: Checking for cgroup 'blkio' controller support
: PASS
  QEMU: Checking for cgroup 'blkio' controller mount-point
: PASS
  QEMU: Checking for device assignment IOMMU support
: PASS
  QEMU: Checking if IOMMU is enabled by kernel
: PASS
```

```
Everything is fine now.
```

KVM and QEMU hypervisors has been installed on CentOS 8.

Configure KVM Environment

Allow QEMU/KVM Commands for personal user:

If you want to allow your login user (non-root) to run virsh command or other KVM/QEMU commands, or use these commands without sudo, (needed by cockpit) then add your login user to the libvirt group as follows:

```
# usermod -aG libvirt YOURUSERNAME
```

Setup own VM-Storage / ISO-Storage configuration:

Create needed directories:

```
# mkdir -p /data01/vm-storage
# mkdir /data01/iso-images
```

Make the virsh pool configuration changes:

1. Listing current pools:

```
# virsh pool-list
```

```
Name                 State      Autostart
-------------------------------------------
default              active     yes
```

2. Destroying current default pool:

```
# virsh pool-destroy default
```

```
Pool default destroyed
```

3. Undefine current default pool:

```
# virsh pool-undefine default
```

```
Pool default has been undefined
```

4. Defining a new pool with name “default”:

```
# virsh pool-define-as --name default --type dir --target /data01/vm-storage
```

```
Pool default defined
```

5. Set pool to be started when libvirt daemons starts:

```
# virsh pool-autostart default
```

```
Pool default marked as autostarted
```

6. Start pool:

```
# virsh pool-start default
```

```
Pool default started
```

7. Checking pool state:

```
# virsh pool-list
```

```
Name                 State      Autostart
-------------------------------------------
default              active     yes
```

From now, when creating virtual machines, Virtual Machine Manager will inform you that the *.img file (virtual disk of your VM), will be saved at `/data01/vm-storage/`.

Installing Cockpit Web Interface in CentOS 8

Although, KVM commandline-tools are quite sufficient for managing a Virtualization environment. But, we can also use the CentOS 8 native Web UI i.e. Cockpit to manage virtual machines via a graphical interface.

We are installing Cockpit using dnf command. To add support of managing virtual machines, we have to install cockpit-machines package as well.

```
# dnf install -y cockpit cockpit-machines
```

```
...
================================================================================
====
Installing:
 cockpit                      x86_64 185.1-1.el8_0              BaseOS
68 k
 cockpit-machines             noarch 184.1-1.el8                AppStream
669 k
Installing dependencies:
 PackageKit                   x86_64 1.1.12-2.el8               AppStream
600 k
 PackageKit-glib              x86_64 1.1.12-2.el8               AppStream
141 k
 cairo-gobject                x86_64 1.15.12-3.el8              AppStream
33 k
 python3-cairo                x86_64 1.16.3-6.el8               AppStream
90 k
 python3-gobject              x86_64 3.28.3-1.el8               AppStream
25 k
 python3-systemd              x86_64 234-8.el8                  AppStream
```

```
81 k
 setroubleshoot-plugins       noarch 3.3.10-1.el8               AppStream
365 k
 checkpolicy                  x86_64 2.8-2.el8                  BaseOS
338 k
 cockpit-bridge               x86_64 185.1-1.el8_0              BaseOS
596 k
 cockpit-system               noarch 185.1-1.el8_0              BaseOS
1.6 M
 cockpit-ws                   x86_64 185.1-1.el8_0              BaseOS
834 k
 gdk-pixbuf2                  x86_64 2.36.12-2.el8              BaseOS
466 k
 glib-networking              x86_64 2.56.1-1.1.el8             BaseOS
155 k
 gsettings-desktop-schemas    x86_64 3.28.1-1.el8               BaseOS
619 k
...
```

Enable and start Cockpit Unit.

```
# systemctl enable --now cockpit.socket
```

```
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket at
/usr/lib/systemd/system/cockpit.socket
```

Cockpit service is by-default allowed in CentOS 8 firewall.

Browse URL https://YOUR-SERVERS-IP:9090/ in a client's browser.

The Cockpit uses a self-signed SSL certificate, therefore, you may see a Security warning. Ignore the Security warning and continue to the website.

Last update: **2020/09/02 12:37**