Getting Started with Kernel-based Virtual Machine (KVM)

Agenda / Tasks / 90 Minutes

- Setup Kernel-based Virtual Machine (KVM)
- Manage KVM Using:
  - Command Line Interface (CLI)
  - Graphical User Interface (GUI)

Abstract (Part 1/5)

- Want to get started with Kernel-based Virtual Machine (KVM)?
- Want to run a virtual machine on your system using open source technologies?
- Want to interact with KVM virtual machines from command line interface (CLI)?

Abstract (Part 2/5)

In this tutorial, Leonard will be teaching people to familiarize themselves with KVM technologies, which allows virtual machines to run with near native performance.

Abstract (Part 3/5)

Participants must have a basic knowledge on how Linux operating system works, and must have a Linux based operating system running natively on a laptop in order to join this tutorial. We will be focusing on doing some tasks of creating, accessing, modifying, and deleting KVMs, from both graphical user interface (GUI) and CLI.

Abstract (Part 4/5)

We will busy with these virt-manager tools in the tutorial. - `virt-install` is a command line tool which provides an easy way to provision operating systems into virtual machines. - `virt-viewer` is a lightweight UI interface for interacting with the graphical display of virtualized guest OS. It can display VNC or SPICE, and uses libvirt to lookup the graphical connection details.

Abstract (Part 5/5)

At the end of this tutorial, participants are expected to know how to check if KVM is supported on their hardware, create and manage KVMs with confident, from both GUI and CLI.

Overview of Kernel-based Virtual Machine (KVM) (Part 1/2)

- An open source virtualization technology built into Linux®. Turn Linux into a hypervisor.
- Allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).
- Available from Linux 2.6.20 or newer.

Overview of Kernel-based Virtual Machine (KVM) (Part 2/2)

- QEMU (Quick Emulator) is part of the KVM experience being the userspace backend for it, but it also can be used for hardware without virtualization extensions by using its Tiny Code Generator (TCG) mode.

Task 1

Setup Kernel-based Virtual Machine (KVM)

Hardware Virtualization Support (Part 1/2)

- KVM requires a CPU with virtualization extensions.
  - Intel® Virtualization Technology (Intel® VT)
    - CPU flag is `vmx` (*Virtual Machine Extensions*).
  - AMD virtualization (AMD-V)
    - CPU flag is `svm` (*Secure Virtual Machine*).

Hardware Virtualization Support (Part 2/2)

```
egrep --count '^flags.*(vmx|svm)' /proc/cpuinfo
```

- If output is 0, your system does not support the relevant virtualization extensions *or* disabled on BIOS. You can still use QEMU/KVM, but the emulator will fall back to software virtualization, which is much slower.

Installing Virtualization Packages (Fedora)

```
dnf groupinfo virtualization

dnf group install \
    virtualization \
    --with-optional \
    --assumeyes
```

- See Fedora's Installation Documentation.

Installing Virtualization Packages (Ubuntu)

```
# apt-get install \
    bridge-utils \
    qemu-kvm \
    virt-manager
```

Installing Virtualization Packages (CentOS)

```
# yum install \
    libvirt \
    qemu-kvm \
    virt-install \
    virt-install \
    virt-manager
```

Enable `libvirtd` Service

- The **`libvirtd`** service is a server side daemon and driver required to manage the virtualization capabilities of the KVM hypervisor.
- Start **`libvirtd`** service and enable it on boot.

```
systemctl start libvirtd

systemctl enable libvirtd
```

Verify KVM Kernel Modules

- Verify that the KVM kernel modules are properly loaded.

```
lsmod | egrep 'kvm_*(amd|intel)'
```

- If output contains **kvm_intel** or **kvm_amd**, KVM is properly configured.

Append Groups to Manage KVM

- Append current user to `kvm` and `libvirt` groups to create and manage virtual machines.

```
usermod --append --groups=kvm,libvirt ${USER}

cat /etc/group | egrep "^(kvm|libvirt).*${USER}"
```

- Log out and log in again to apply this modification.

Update QEMU Configuration

```
# cp /etc/libvirt/qemu.conf /etc/libvirt/qemu.conf.original
# sed --in-place \
    "s,\#user = \"root\",\#user = \"${USER}\",g" \
    /etc/libvirt/qemu.conf
# sed --in-place \
    "s,\#group = \"root\",\#group = \"libvirt\",g" \
    /etc/libvirt/qemu.conf
# diff --unified \
    /etc/libvirt/qemu.conf.original \
    /etc/libvirt/qemu.conf
systemctl restart libvirtd
```

Task 2

Manage KVM using Command Line Interface (CLI)

- Install Debian from Network.

Install Debian from Network

```
$ virt-install \
    --name Debian --os-variant debian11 --description 'Debian' \
    --vcpus 1 --ram 1024 \
    --location \
    https://ftp.debian.org/debian/dists/stable/main/installer-amd64 \
    --network bridge=virbr0 \
    --graphics vnc,listen=127.0.0.1,port=5901 \
    --noreboot --noautoconsole \
    --extra-args 'console=ttyS0,115200n8 serial'
$ virt-viewer --connect qemu:///session --wait Debian
```

Guest Virtual Machine States and Types (Part 1/2)

Several `virsh` commands are affected by the state of the guest virtual machine: `Transient` or `Persistent`.

Guest Virtual Machine States and Types (Part 2/2)

During the life cycle of a virtual machine, `libvirt` will classify the guest as any of the following states:

`Undefined`, `Shut off`, `Running`, `Paused`, `Saved`

Display `virsh` Version

```
virsh version
virsh version --daemon
```

Connect to Hypervisor (Part 1/2)

```
virsh connect [hostname-or-URI] [--readonly]
```

The most commonly used URIs are:

`qemu:///system`, `qemu:///session`, `lxc:///`

Connect to Hypervisor (Part 2/2)

For example, establish a session to connect to your set of guest virtual machines (VMs), with you as the local user:

```
virsh connect qemu:///session
```

List Guest VM Connected to Hypervisor

```
virsh list --all
virsh list --inactive
```

Display Information about Hypervisor

```
virsh hostname
virsh sysinfo
```

Extra: Start Guest Virtual Machine

```
virsh start <Guest_VM> [--console] [--paused]
[--autodestroy] [--bypass-cache] [--force-
boot]
```

Starts the `<Guest_VM>` that you already created and is currently in the inactive state.

Configuring a Virtual Machine to be Started Automatically at Boot

```
virsh autostart [--disable] <Guest_VM>
```

```
Example: virsh autostart Debian
```

Extra: Rebooting a Guest Virtual Machine

```
virsh reboot <Guest_VM> [--mode
<RebootModeName>]
```

Example: `virsh reboot Debian --mode initctl`

Extra: Save Guest Virtual Machine's Configuration

```
virsh save [--bypass-cache] domain file [--xml string] [--running] [--paused] [--verbose]
```

```
Example: virsh save Debian Debian-Configuration.xml --running
```

Extra: Restore Guest Virtual Machine

```
virsh restore <file> [--bypass-cache] [--xml
/path/to/file] [--running] [--paused]
```

```
Example: virsh restore Debian-Configuration.xml
--running
```

Extra: Resuming a Guest Virtual Machine

```
virsh resume <Guest_VM>
```

Display Host Physical Machine Name

```
virsh domhostname <Guest_VM>
```

Display Guest VM General Information

```
virsh dominfo <Guest_VM>
```

Display Guest VM's ID Number

```
virsh domid <Guest_VM>
```

Abort Running Jobs on a Guest VM

```
virsh domjobabort <Guest_VM>
```

List Statistic about Guest VM

```
virsh domjobinfo <Guest_VM>
```

Display Guest Virtual Machine's Name

```
virsh domname <Guest_VM_ID>
```

Display Virtual Machine's State

```
virsh domstate <Guest_VM>
```

Display Connection State to the Virtual Machine

```
virsh domcontrol <Guest_VM>
```

Extra: Install Ubuntu from ISO Image

```
$ virt-install \
    --name Ubuntu --os-variant ubuntu22.04 --description 'Ubuntu' \
    --vcpus 2 --ram 2048 \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5902 \
    --cdrom ~/Downloads/ubuntu-22.04-desktop-amd64.iso \
    --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait Ubuntu
```

Extra: Install Ubuntu from Network

```
$ virt-install \
    --name Ubuntu --os-variant ubuntu20.04 --description 'Ubuntu' \
    --vcpus 2 --ram 2048 \
    --location \
    http://archive.ubuntu.com/ubuntu/dists/focal/main/installer-amd64/ \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5902 \
    --noreboot --noautoconsole \
    --extra-args='console=ttyS0,115200n8 serial edd=off'
$ virt-viewer --connect qemu:///session --wait Ubuntu
$ virsh console Ubuntu
```

Extra: Install Fedora from ISO Image

```
$ virt-install \
    --name Fedora --os-variant fedora36 --description 'Fedora' \
    --vcpus 2 --ram 2048 \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5904 \
    --cdrom ~/Downloads/Fedora-Workstation-Live-x86_64-36-1.5.iso \
    --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait Fedora
```

Extra: Install Fedora from Network

```
$ virt-install \
    --name Fedora --os-variant fedora36 --description 'Fedora' \
    --vcpus 2 --ram 2048 \
    --location \
    https://download.fedoraproject.org/pub/fedora/linux/releases/36/Server/x86_64/os \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5904 \
    --noreboot \
    --extra-args='console=ttyS0,115200n8 edd=off'
$ virsh console Fedora
```

Extra: Install AlmaLinux from ISO Image

```
$ virt-install \
    --name AlmaLinux --os-variant almalinux9 --description 'AlmaLinux' \
    --vcpus 2 --ram 3072 \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5903 \
    --cdrom ~/Downloads/AlmaLinux-9.0-x86_64-dvd.iso \
    --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait AlmaLinux
```

Extra: Install AlmaLinux from Network

```
$ virt-install \
    --name AlmaLinux --os-variant almalinux9 --description 'AlmaLinux' \
    --vcpus 2 --ram 3072 \
    --location \
    https://almalinux.uib.no/9.0/BaseOS/x86_64/os/ \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5905 \
    --noreboot \
    --extra-args='console=ttyS0,115200n8 edd=off'
$ virsh console AlmaLinux
```

Extra: Install CentOS from ISO Image

```
$ virt-install \
    --name CentOS --os-variant centos-stream9 --description 'CentOS' \
    --vcpus 2 --ram 3072 \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5902 \
    --cdrom ~/Downloads/CentOS-Stream-9-latest-x86_64-dvd1.iso \
    --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait CentOS
```

Extra: Install CentOS from Network

```
$ virt-install \
    --name CentOS --os-variant centos-stream9 --description 'CentOS' \
    --vcpus 2 --ram 3072 \
    --location \
    https://mirror.netsite.dk/centos-stream/9-stream/BaseOS/x86_64/os/ \
    --network bridge=virbr0,model=virtio \
    --graphics vnc,listen=127.0.0.1,port=5904 \
    --noreboot \
    --extra-args='console=ttyS0,115200n8 edd=off'
$ virt-viewer --connect qemu:///session --wait CentOS
$ virsh console CentOS
```

Delete Virtual Machine

```
$ virsh shutdown Debian # Graceful Shutown
Domain 'Debian' is being shutdown

$ virsh destroy Debian # Force Shutdown
Domain 'Debian' destroyed

$ virsh undefine Debian
Domain 'Debian' has been undefined
```

Related `vir*` Commands

```
$ virt-install --os-variant list

$ virsh nodeinfo
$ virsh edit
$ virt-df
$ virt-top
$ virt-viewer

$ virsh pool-list --all
$ virsh pool-destroy
$ virsh pool-undefine
```

View Serial Console Message

```
$ virsh console Fedora
Connected to domain 'Fedora'
Escape character is ^] (Ctrl + ])
```

Error: Refusing to Undefine

```
$ virsh undefine Ubuntu --remove-all-storage
error: Refusing to undefine while domain managed save image exists
```

```
$ virsh managedsave-remove Ubuntu
Removed managedsave image for domain 'Ubuntu'
$ virsh undefine Ubuntu
Domain 'Ubuntu' has been undefined
```

Error: Failed to Get MTU of Bridge

```
stderr=failed to get mtu of bridge `virbr0': No such device
```

```
# systemctl restart libvirtd
$ brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.525400a87247       yes
```

Error: Hangs on Probing EDD

```
Booting from Hard disk....
Probing EDD (edd=off to disable)... ok
```

```
$ virt-install \
    ...
    --extra-args='... edd=off'
```

Error: Failed to Get Domain

- Ensure that specified storage pool has correct permissions and path.

```
$ virsh pool-list --all
$ virsh pool-info default
$ virsh pool-dumpxml default
$ virsh pool-dumpxml default \
    | xmlstarlet sel --template --copy-of "/pool/target"
$ virsh pool-dumpxml default \
    | xmlstarlet sel --template --value-of "/pool/target/path"
```

Error: Cannot Access Storage File (UID:107, GID:107)

```
# cp /etc/libvirt/qemu.conf /etc/libvirt/qemu.conf.original
# sed --in-place \
    "s,\#user = \"root\",\#user = \"${USER}\",g" \
    /etc/libvirt/qemu.conf
# sed --in-place \
    "s,\#group = \"root\",\#group = \"libvirt\",g" \
    /etc/libvirt/qemu.conf
# systemctl restart libvirtd
```

Error: Missing 'Default' Network?

```
$ virsh net-list --all
 Name   State   Autostart   Persistent
----------------------------------------

$ sudo virsh net-list --all
 Name      State    Autostart   Persistent
--------------------------------------------
 default   active   yes         yes
```

Read this post if default network is still missing.

Task 3

Manage KVM using Graphical User Interface (GUI)

Use `virt-manager` to create, manage, & delete KVMs.

Bonus: Unattended Install

- Preseeding (Debian-based Linux Distributions) or Kickstart (Red Hat-based Linux Distributions) provides a way to set answers to questions asked during the installation process, without having to manually enter the answers while the installation is running.

Bonus: Assign Host USB Device

https://www.linux-kvm.org/page/USB_Host_Device_Assigned_to_Guest

Related Links

https://gist.github.com/sheeeng/6f43d75d6d58fee44690208cbe6c5dd9

End