Lab CMP-I – Compute Virtualisation

Introduction & Prerequisites

This laboratory is to learn how to:

- setup basic storage and networking to support the VMs you'll create.
- create, manipulate and delete virtual machines from the command line.
- access virtual machines remotely from the command line.
- scale up a VM by adding more CPU, RAM and/or disk capacity.

In this lab you will learn how to use open source virtualisation technologies. KVM, QEMU and libvirt. These technologies are the foundation of many cloud computing technologies such as CloudStack and OpenStack. You will use these tools to exercise the full life cycle of virtual machines, from creation to deletion.

The following resources and tools are required for this laboratory session:

- Any modern web browser,
- Any modern SSH client application
- OpenStack Horizon dashboard: https://ned.cloudlab.zhaw.ch
- OpenStack account details: please contact the lab assistant in case you already have not received your access credentials.
- VNC Client (suggested, there are others)
    - Windows, Mac, Linux: https://www.realvnc.com/en/connect/download/viewer/

Time

The entire session will take 90 minutes.

Task 1 – Setup Your Working Environment

In this task you will set up the virtual lab environment ready for you to work with KVM and libvirt. The environment is provided by the respective image on NED, called
Note the access credentials to work with this image: CCP1-EN-CMP-I

- Username to login with SSH into VMs on ned.cloudlab.zhaw.ch OpenStack cloud from your laptops: **`ubuntu`**
- The pre-created libvirt vm disk image ***`my_vm.img`*** has a built in user account enabled
  - Username: **`user`**
  - Password: **`password`**

Subtask – Start Necessary VMs and Verify Environment

From previous labs you now know how to configure a virtual network, connect it using a router to an external public network and launch VMs in OpenStack. Please perform the following steps -

- Create a VM (`image:`CCP1-EN-CMP-I)
- Enable SSH, port `5900` (VNC) via security groups.
- Assign a Floating IP to your VM
- Check if the KVM module is loaded by the operating system using:
  - `Lsmod and kvm-ok`
- Check if libvirt is installed properly by querying for the default network
  - `sudo virsh net-list`
  - `sudo virsh net-dumpxml default`

Questions

ubuntu@lab-cmp1:~$ lsmod and kvm-ok
Usage: lsmod

- Is the KVM module loaded? What information is shows if it is loaded?
- Enter virsh and check what you can find out about the compute capabilities of your system.

it has a NAT, Bridget Interface and a DHCP

Subtask – QEMU basics

First, identify if QEMU is installed on your system and if so, what are the different tools for and which modes are supported by our installation?

dpkg -l | grep qemu

Subtask – VM Creation using QEMU and libvirt

You should see the following files already present in your home directory, see lab directory cmp-one-two

- `my_vm.img` (existing vm image)
- `my_vm.xml`
- `my_vm_precreated.xml`

Analyze the difference between `my_vm.xml` and `my_vm_precreated.xml` files.

- Which one allows you to boot from a virtual CDROM and install a new operating system from scratch?
- What is the type of virtualization used within these XML files?

QEMU Basics

```
ubuntu@lab-cmp1:~$ dpkg -l | grep qemu
ii  ipxe-qemu                         1.0.0+git-20190109.133f4c4-0ubuntu3.2 all          PXE boot firmware - ROM images for qemu
ii  ipxe-qemu-256k-compat-efi-roms    1.0.0+git-20150424.a25a16d-0ubuntu4   all          PXE boot firmware - Compat EFI ROM images for qemu
ii  libvirt-daemon-driver-qemu        6.0.0-0ubuntu8.16                     amd64        Virtualization daemon QEMU connection driver
ii  qemu-block-extra:amd64            1:4.2-3ubuntu6.27                     amd64        extra block backend modules for qemu-system and qemu-utils
ii  qemu-kvm                          1:4.2-3ubuntu6.27                     amd64        QEMU Full virtualization on x86 hardware
ii  qemu-system-common                1:4.2-3ubuntu6.27                     amd64        QEMU full system emulation binaries (common files)
ii  qemu-system-data                  1:4.2-3ubuntu6.27                     all          QEMU full system emulation (data files)
ii  qemu-system-gui:amd64             1:4.2-3ubuntu6.27                     amd64        QEMU full system emulation binaries (user interface and audio support)
ii  qemu-system-x86                   1:4.2-3ubuntu6.27                     amd64        QEMU full system emulation binaries (x86)
ii  qemu-utils                        1:4.2-3ubuntu6.27                     amd64        QEMU utilities
```

Analyze the difference between my_vm.xml and my_vm_precreated.xml files.

● Which one allows you to boot from a virtual CDROM and install a new operating system from scratch?

my_vm.xml has setup a boot device

```
<os>
   <type>hvm</type>
   <boot dev='cdrom'/>
   <boot dev='hd'/>
   <bootmenu enable='yes' timeout='10000'/>
 </os>
```

and also a section on what to do

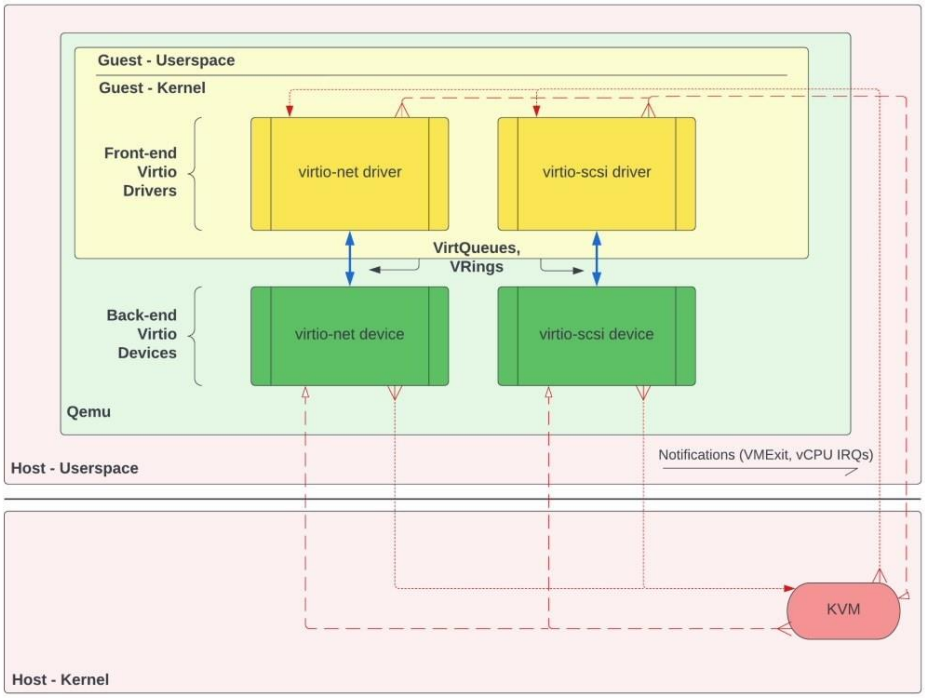
```
<disk type="file" device="cdrom">
     <driver name="qemu" type="raw"/>
     <source file="/home/ubuntu/ubuntu-16.04.3-server-amd64.iso"/>
     <target dev="hdc" bus="ide"/>
     <readonly/>
</disk>
```

● What is the type of virtualization used within these XML files?

KVM

```
<domain type='kvm'>
```

- Libvirt provides a convenient way to use underlying virtualisation technology. Issue the following command and analyse what will be done by libvirt and what features it uses with respect to virtualization of compute and devices. Recall the image below.
    - `sudo virsh domxml-to-native qemu-argv my_vm.xml`



Source: IBM

Question: from the command's output what parameter specifies the amount of RAM supplied to the guest VM? What parameter specifies the number of CPUs given to the guest VM?

Hint: If for some reason you need to create an empty disk image to use with a VM, issue the following command:

- `sudo qemu-img create -f qcow2 my_vm_temp.img 5G`

Question: Did you try the above command? This will create a disk using the QCOW2 format. It's size is `5G` but notice the actual size on the disk. Why is it not 5GB and a few KB only on disk?

Let's now create the VM itself. Do this by issuing the following command:

- `sudo virsh create my_vm_precreated.xml`
    - `Make sure that all references in the xml file are correct. You can edit the file using vim or any other editor.`
- Try and login to the just created VM via VNC console on your laptop
    - Use floating IP of your host / OpenStack VM and 5900 port to access the VNC console
    - Do you know what password to use in the VNC viewer? [Hint: check the XML definitions]

Question: from the command's output what parameter specifies the amount of RAM supplied to the guest VM? What parameter specifies the number of CPUs given to the guest VM?

in my_vm.xml this is defined:

<memory>512000</memory>
<vcpu placement='static' current='1'>4</vcpu>

in the output this are defined:

-cpu qemu64
-m 500
-overcommit mem-lock=off
-smp 1,maxcpus=4,sockets=4,cores=1,threads=1

Question: Did you try the above command? This will create a disk using the QCOW2 format. It's size is 5G but notice the actual size on the disk. Why is it not 5GB and a few KB only on disk?

QCOW2 is a sparse file format, meaning that it doesn't pre-allocate the entire disk space when the image is created. Instead, it allocates space on an as-needed basis. The file will start small and grow dynamically as data is written to it.

The initial size of the QCOW2 file will be relatively small, reflecting the minimal amount of data needed to represent an empty virtual disk. As you add data to the virtual machine, the QCOW2 file will grow to accommodate the new information

- What driver is the guest OS using for it's harddrive? (Block Device). Mind, directory /sys/ provides you with information about devices
- Play with the following commands and discuss with your partner what do you see
    - `sudo virsh domblklist my_vm`
    - `sudo virsh dumpxml my_vm`
    - `sudo virsh net-list`
    - `sudo virsh list`
    - `sudo virsh domiflist my_vm`
    - `sudo arp -e`

- What is the IP address allocated to your `my_vm` VM?
- Can you SSH into this IP from your host VM?

Subtask – Basic Management with libvirt

In this task you will exercise some of the basic management features of libvirt.

- VM Listing
    - `sudo virsh list`

Question: Can you see the process associated with the VM by issuing the command '`ps ax`'? What can you tell from this?

- VM Lifecycle management
    - Reboot
        - `sudo virsh reboot my_vm`

Question: What do you see in the VNC console?

    - Suspend & Resume
        - `sudo virsh suspend my_vm`

Question: What is the state of the VM after executing this command (use `sudo virsh list`)? Can you use the VM (login, run a command)?

    - Resume
        - `sudo virsh resume my_vm`

Question: Again, what is the state of the VM after executing this command (use `sudo virsh list`)?

Subtask – Snapshots with libvirt

Snapshots allow you to capture the state of the VM as it changes over time. Can you imagine a situation where snapshots can be useful?

To create a snapshot of your VM issue the following command:

- `sudo virsh snapshot-create-as --domain my_vm --name my_vm_s0 --description "Test snapshot"`

You can get a listing of your snapshots by executing:

- `sudo virsh snapshot-list my_vm`

Let's test the effect of rollback now:

- Create a new file in your running vm, using the linux `touch` command.

- Do a rollback to previously created snapshot
    - `sudo virsh snapshot-revert --domain my_vm --snapshotname my_vm_s0`

Question: what do you observe when this is executed? Is the file (using touch) you just created there?

Subtask – Basic Monitoring and Analysis with libvirt

Libvirt supports the extraction of many metrics from a created VM. Use the following command to explore what can be monitored:

- `virsh help monitor`

Questions:

- How would you get statistics on the VM's memory use?
- How would you get statistics on the VM itself?
- How would you get statistics on the network traffic to and from the VM?

Subtask – Remove Resources with libvirt

When the VM is no longer needed, we can delete the VM with the following command:

- `sudo virsh destroy my_vm`

Question: If I destroy the VM process, is my data also destroyed? Do I have to reinstall the VM again?

Task 2 - Advanced Tasks

By now you should understand how to setup a host machine with a KVM hypervisor, create a VM on KVM using libvirt along with the required inputs (disk and image) and carry out some basic management and monitoring tasks upon that VM, `my_vm`. In the next part, we'll look into some more advanced aspects available to us including scaling and live migration of a VM from one host to another host.

Subtask – Scaling a VM with libvirt

In this task you will learn how to scale up/down (add more or less) system resources to your already running virtual machine my_vm. Use '`sudo virsh help`' to find out which command to run.

- Add Another CPU: The current settings of `my_vm` is to have one CPU. Use appropriate command to increase the cpu count to 2
- Verify if the CPU has been added in your `my_vm` virtual machine
- `sudo virsh dominfo my_vm`
- RAM (memory): The current settings of `my_vm` is `512MB`. Scale down the amount of memory to 256MB
- Verify the new memory limit allocated for your `my_vm` instance
- Attach More Storage (Disk): Your VM, `my_vm`, already has a thinly provisioned disk which has a capacity of `10GB`.
- Imagine a scenario where you need to add more storage into your VM. How can you do it with libvirt?
- Allocate the new storage on the host machine using:
- `sudo qemu-img create -f qcow2 /home/ubuntu/disk2.qcow2 1G`
- Then we tell libvirt to attach the storage into the existing instance my_vm
- `sudo virsh attach-disk my_vm --source /home/ubuntu/disk2.qcow2 --target vdb`
- Now go to the guest VM, `my_vm`, VNC console and validate that the new disk is available by issuing this command: `sudo fdisk -l`

Subtask – Working with Qemu

After extensive use of libvirt, which facilitates working with Qemu heavily, now work with Qemu directly. Start a VM on your host "source" using the qemu command line directly. Use the image my_vm.img and make it available via VNC on port 5900 with security enabled using the same configuration as before. Also enable ssh via a port redirection to map host port 5555 to guest port 22.. Hint, Qemu allows you to monitor its operation by using the -monitor stdio option.

Cleanup

IMPORTANT: At the end of the lab session:

- **Delete** all OpenStack - unused - VMs, volumes, security group rules that were created by your team.
- **Release** all floating IPs back to the central pool for others to use.
    - Go to Network -> Floating IPs to release IPs back to the pool

Additional Documentation

- User Guide: https://docs.openstack.org/horizon/latest/
- http://qemu-buch.de/
- https://en.wikibooks.org/wiki/QEMU/Networking
- Linux man pages of qemu and libvirt