DevCloud TP1

Alleaume Julien

3 Environnement du TP

3.1 Installations nécessaires au TP

On check le support des flags de virtualisation sur le CPU :

```
grep -E '(vmx|svm)' /proc/cpuinfo
virt-host-validate
```

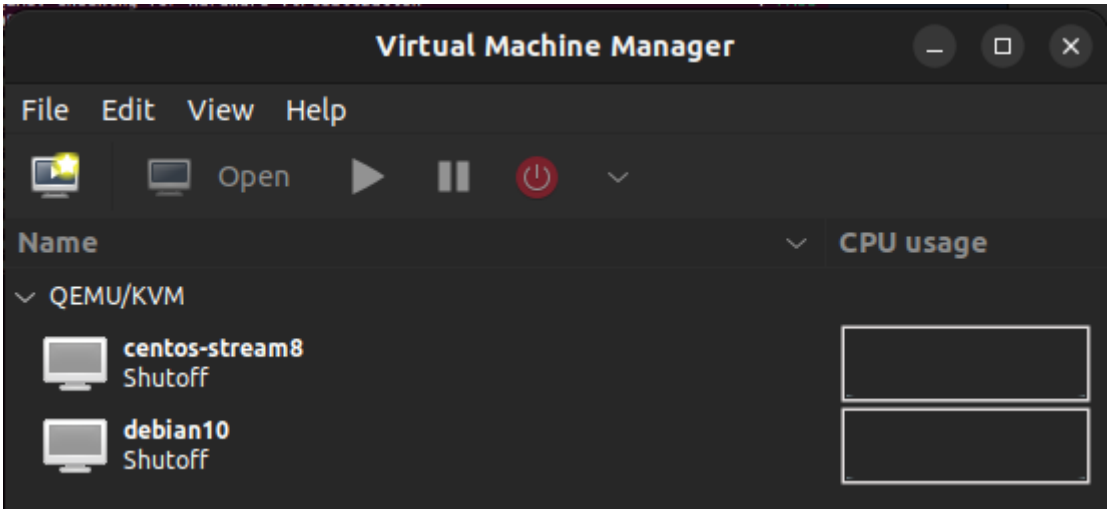
```
 QEMU: Checking for hardware virtualization                                 : PASS
 QEMU: Checking if device /dev/kvm exists                                   : PASS
 QEMU: Checking if device /dev/kvm is accessible                            : PASS
 QEMU: Checking if device /dev/vhost-net exists                             : PASS
 QEMU: Checking if device /dev/net/tun exists                               : PASS
 QEMU: Checking for cgroup 'cpu' controller support                         : PASS
 QEMU: Checking for cgroup 'cpuacct' controller support                     : PASS
 QEMU: Checking for cgroup 'cpuset' controller support                      : PASS
 QEMU: Checking for cgroup 'memory' controller support                      : PASS
 QEMU: Checking for cgroup 'devices' controller support                     : WARN
 QEMU: Checking for cgroup 'blkio' controller support                       : PASS
 QEMU: Checking for device assignment IOMMU support                         : PASS
 QEMU: Checking if IOMMU is enabled by kernel                               : PASS
 QEMU: Checking for secure guest support                                    : WARN
  LXC: Checking for Linux >= 2.6.26                                         : PASS
  LXC: Checking for namespace ipc                                           : PASS
  LXC: Checking for namespace mnt                                           : PASS
  LXC: Checking for namespace pid                                           : PASS
  LXC: Checking for namespace uts                                           : PASS
  LXC: Checking for namespace net                                           : PASS
  LXC: Checking for namespace user                                          : PASS
  LXC: Checking for cgroup 'cpu' controller support                         : PASS
  LXC: Checking for cgroup 'cpuacct' controller support                     : PASS
  LXC: Checking for cgroup 'cpuset' controller support                      : PASS
  LXC: Checking for cgroup 'memory' controller support                      : PASS
  LXC: Checking for cgroup 'devices' controller support                     : FAIL
  LXC: Checking for cgroup 'freezer' controller support                     : FAIL
  LXC: Checking for cgroup 'blkio' controller support                       : PASS
  LXC: Checking if device /sys/fs/fuse/connections exists                   : PASS
```

`osinfo-query os` permet de montrer les images compatible avec 'libvirt' sur le systeme d'exploitation.

```
ubuntu20.04          | Ubuntu 20.04 LTS                    | 20.04    | http://ubuntu.com/ubuntu/20.04
ubuntu20.10          | Ubuntu 20.10                        | 20.10    | http://ubuntu.com/ubuntu/20.10
ubuntu21.04          | Ubuntu 21.04                        | 21.04    | http://ubuntu.com/ubuntu/21.04
ubuntu21.10          | Ubuntu 21.10                        | 21.10    | http://ubuntu.com/ubuntu/21.10
ubuntu22.04          | Ubuntu 22.04 LTS                    | 22.04    | http://ubuntu.com/ubuntu/22.04
ubuntu4.10           | Ubuntu 4.10                         | 4.10     | http://ubuntu.com/ubuntu/4.10
ubuntu5.04           | Ubuntu 5.04                         | 5.04     | http://ubuntu.com/ubuntu/5.04
ubuntu5.10           | Ubuntu 5.10                         | 5.10     | http://ubuntu.com/ubuntu/5.10
ubuntu6.06           | Ubuntu 6.06 LTS                     | 6.06     | http://ubuntu.com/ubuntu/6.06
ubuntu6.10           | Ubuntu 6.10                         | 6.10     | http://ubuntu.com/ubuntu/6.10
ubuntu7.04           | Ubuntu 7.04                         | 7.04     | http://ubuntu.com/ubuntu/7.04
ubuntu7.10           | Ubuntu 7.10                         | 7.10     | http://ubuntu.com/ubuntu/7.10
ubuntu8.04           | Ubuntu 8.04 LTS                     | 8.04     | http://ubuntu.com/ubuntu/8.04
ubuntu8.10           | Ubuntu 8.10                         | 8.10     | http://ubuntu.com/ubuntu/8.10
ubuntu9.04           | Ubuntu 9.04                         | 9.04     | http://ubuntu.com/ubuntu/9.04
ubuntu9.10           | Ubuntu 9.10                         | 9.10     | http://ubuntu.com/ubuntu/9.10
unknown              | Unknown                             |          | http://libosinfo.org/unknown
voidlinux            | Void Linux                          |          | http://voidlinux.org/voidlinux/rolling
win1.0               | Microsoft Windows 1.0               | 1.0      | http://microsoft.com/win/1.0
win10                | Microsoft Windows 10                | 10.0     | http://microsoft.com/win/10
win2.0               | Microsoft Windows 2.0               | 2.0      | http://microsoft.com/win/2.0
win2.1               | Microsoft Windows 2.1               | 2.1      | http://microsoft.com/win/2.1
win2k                | Microsoft Windows 2000              | 5.0      | http://microsoft.com/win/2k
win2k12              | Microsoft Windows Server 2012       | 6.3      | http://microsoft.com/win/2k12
win2k12r2            | Microsoft Windows Server 2012 R2    | 6.3      | http://microsoft.com/win/2k12r2
win2k16              | Microsoft Windows Server 2016       | 10.0     | http://microsoft.com/win/2k16
```

3.2 URLs pour réaliser une netinstall

Réalisation de la `net install` sans problème d'une CentOs et d'une debian10.



4 Création de machines virtuelles K-VMs

4.1 Création de VMs avec virt-manager

4.1.1 Accès à virt-manager

Configuration du port série de la console sur les deux VM sans soucis.

```
systemctl start serial-getty@ttyS0
systemctl enable serial-getty@ttyS0
```

4.1.2 Installation d'une VM Centos avec virt-manager

Démarrage de `virsh` faire :

```
virsh -c qemu:///system
#ou
virsh -c qemu:///session
```

Explication problèmes : J'ai deux 'type' de virsh : `virsh -c qemu:///system` et `virsh -c qemu:///session`. Les VMs de VirtualManager seront dans system et les autres dans sessions.

1. `virsh nodeinfo`
   #Affiche les infos systèmes sur lequel virsh est installé.
2. `virsh list –state-running`
   #Affiche les infos des machines démarrées.
3. `virsh domstate –domain centOS –reason`
   #Affiche les informations du status de la VM CentOs.
4. `virsh start –domain centOS`
   #Démarre la VM CentOs.
5. `virsh autostart –domain centOS`
   #Démarre automatiquement la VM CentOs à chaque démarrage.
6. `virsh dominfo –domain centOS`
   #Donne les informations VM CentOs.
7. `virsh destroy –graceful`
   #Eteint la VM CentOs de manière propre.

4.2 Création d'une K-VM Debian avec virt-install

- 1. Commande exemple prise d'internet :

```
sudo virt-install --name vmname --ram 1024 --os-type=linux --os-
variant=ubuntutrusty –disk
path=/data/vm/vmname_sda.qcow2,bus=virtio,size=10,sparse=false --
noautoconsole --console pty,target_type=virtio --accelerate --hvm --
network=network:default --graphics spice,port=20001,listen=127.0.0.1
```

- Ma commande final après test:

```
virt-install --name debianmini --ram=1024 --vcpus=1 --os-
variant=debianbuster --cdrom=/home/julien/Downloads/debian-10.6.0-amd64-
netinst.iso --disk bus=virtio,size=5 --graphics spice
```

- Commande données par le professeur :

```
 virt-install –name debian8 –description ‘’test’’ –ram=512 –vcpus=1 os-
type=Linux –os-variant=debian8 –disk path=/var/lib/images/debian/.qcow2 --
graphics spice,listen=127,0,0,1,keymap=fr –cdrom /varlib/libvirt/.iso –
```

```
network bridge:virbr0 –console pty.target_type=serial -x
‘console=ttyS0,115200N8 SERIAL’
```

- 2. Test de plusieurs commande d'info :

```
virsh dominfo --domain debianmini
```


```
virsh schedinfo --domain debianmini
#Et
virsh domiflist --domain debianmini
```

```
virsh # schedinfo --domain debian10
Scheduler      : posix
cpu_shares     : 0
vcpu_period    : 0
vcpu_quota     : 0
emulator_period: 0
emulator_quota : 0
global_period  : 0
global_quota   : 0
iothread_period: 0
iothread_quota : 0

virsh # domi
domid           domif-getlink  domif-setlink  domiftune
domifaddr       domiflist      domifstat      dominfo
virsh # domiflist
error: command 'domiflist' requires <domain> option
virsh # domiflist --
--domain    --inactive
virsh # domiflist --domain
centos-stream8  debian10
virsh # domiflist --domain debian10
 Interface   Type     Source            Model    MAC
-------------------------------------------------------------------
 -          bridge   br-7abd9b640f91   virtio   52:54:00:ad:cc:a9
 -          bridge   virbr1            virtio   52:54:00:e7:1d:0c
```

```
virsh domvlklist
#Et
virsh vcpucount
```

```
virsh # domblklist --domain debian10
 Target   Source
------------------------------------------------
 vda      /var/lib/libvirt/images/debian10.qcow2
 sda      -

virsh # vcpucount --domain debian10
maximum      config         6
current      config         6
```

Photo précedente on des valeurs plus élevés, qui me permette d'effectuer les installes rapidement, d'ou les 6VCPUs.

- 3. Pour modifier les vcpus à chaud on à la commande :

```
virsh setvcpus –domain debian8 –count ‘nombres vcpus’
```

Pour modifier à froid on utilise :

```
virsh edit –domain debian8
```

selectioné l'editeur de texte puis on modifie le nombre vcpus ou autre options recherchés.

4.3 Création de VMs avec virt-builder

Je cherche dans la liste l'OS souhaité :

```
virt-builder –list |grep cent
```

J'installe ensuite l'OS souhaité avec comme mots de passe root= root :

```
virt-builder centosstream-9 --size 20G --root-password password:root
```

La monter ensuite dans virt-install :

```
virt-install --name debianmini --ram=1024 --vcpus=2 --os-
variant=debianbuster –disk=/home/julien/Downloads/debian-
11.qcow2,bus=virtio,size=5 --pxe --graphics spice
```

4.4 Création de VMs avec virt-customize

Je crée la custom avec une image .qcow2 :

```
virt-customize -a /home/julien/Downloads/CentOS.qcow2 –root-password
password:root
```

La monter dans `virt-install` :

```
virt-install --name debianmini --ram=1024 --vcpus=2 --os-
variant=debianbuster –disk=/home/julien/Downloads/debian-
11.qcow2,bus=virtio,size=5 --pxe --graphics spice
```

TIPS :

- .qcow2 est fichier, ancien format, 'copy on write' , possible de faire des snapshot est bien plus effficace.
- L'autre est .img le format le plus récents.

5 Découverte de l'architecture KVM

5.1 Gestion du réseau

1. Lister les bridges virtuel : Avec brctl :

```
brctl show bridge
bridge name bridge      id     STP enabled  interfaces
br-7abd9b640f91     8000.02425c0c412a          no
docker0             8000.0242e032f3d6          no
virbr0              8000.5254000153b9        yes vnet4
```

Avec virsh :

```
virsh # net-info --network default
Name:           default
UUID:           b6692718-7d77-4ff6-a5d0-1fdf463cdc2b
Active:         yes
Persistent:     yes
Autostart:      no
Bridge:         virbr0
```

```
virsh # net-info default
Name:           default
UUID:           b6692718-7d77-4ff6-a5d0-1fdf463cdc2b
Active:         yes
Persistent:     yes
Autostart:      no
Bridge:         virbr0
```

2. Pour voir les infos réseau d'une VM :

```
domiflist --domain 'nom domaine'
#Affiche les interface utilisé par le domain
```

```
virsh # domiflist --domain debian10
 Interface   Type     Source            Model    MAC
-------------------------------------------------------------------
 -           bridge   br-7abd9b640f91   virtio   52:54:00:ad:cc:a9
 -           bridge   virbr1            virtio   52:54:00:e7:1d:0c

virsh # domiflist --domain centos-stream8
 Interface   Type   Source   Model   MAC
-----------------------------------------
```

3. Pour dump la configuration réseau, je fait dans Virsh :

```
net-dumpxml –network default #Permet de print la conf en xml d'un
bridge
```

Ensuite j'éteint le bridge 'default' :

```
net-destroy –network default
```

Pour la création j'utilise le fichier dump et l'edit à mon besoins à cette position : `usrshare/libvirt/network` Je valide et initialise la configuration `Autre.xml` dans virsh :

```
net-create --file Autre.xml --validate
```

Pensez à bien changer au moins le nom, l'interface d'utilisation ou sont ip.

4. Rattacher le bridge à la vm :

```
attach-interface --domain debian10 --type bridge --source virbr1 --
model virtio --config --live
```

On attache l'interface b1 aux domaine debian10 de type bridge avec la source l'interface virbr1 crée précedement avec un modele virtio.
Pour effacer une bridge crée par erreurs par exemple :

```
virsh net-undefine (nom bridge)
```

Et pensez à éteindre l'autostart, etc...

5. Crée le liens macvtap :

```
ip link add link eth0 name macvtap0 type macvtap
#Je la 'set up' et configure mes deux VMs
```

TIPS : MACVtap est une connection réseau sans nat pour plus de perf lors de fort transfert de données.

5.2 : Commencé la configuration mais je n'ai pas finis la configuration final. J'ai reussi la partie création serveur NFS.
Brouillon : Procedure install NFS virt-manager Crée la vm sur le partage et à chaud la cloner.