LAB – Intro

Lab infrastructure

The infrastructure

- The test of most of the topics covered in the LAB will require a real cloud infrastructure
- We do not have a real infrastructure available ☹
- However, we have available a cloud infrastructure (provided by UNIPI)
- A set of VMs are available for the students of the course, we are going to pretend that such VMs composes a real computing infrastructure available
- Most of the solutions covered in the LAB can be deployed on top of VMs in the same manner as real servers
- *Whenever some changes are required, it will be highlighted during the exercises*

VMs

- To each student is assigned **ONE** VM
- The VM has pre-installed Ubuntu 22.04 LTS server in its basic installation (no additional software)
- For some LABS you can work by yourselves, for others you'll need to form groups as the deployment of the solution will require more than one VM
- The composition of groups will be needed for final project (it can be the same)
- Start to form groups of 4 people, report the group composition on the shared Word doc on the teams
- Working students can work solo, they will have a dedicated project assignment

Server Architecture

- Each server has the following hardware configuration
  - Single core Virtual CPU
  - 7 GB of RAM
  - 1 Hard Disks
    - 40 GB HD for the OS (/dev/sda)
  - 1 Network Adapters
    - eth0
    - The card is connected to a private network (actually a virtual network) with address 10.1.X.X/9 (private!)
    - IPv4 address are statically configured in an automatic manner
- All the VMs have username/password set to root/ubuntu and can be accessed via SSH
- This is a new platform, please report issues (hopefully not too many)

VPN

- VMs are connected to a private network
- The private network can be reached through a Virtual Private Network
- You need to install the software OpenVPN and configure it with the configuration package

LAB – Full virtualization

Hands on lab on qemu and virsh

References:

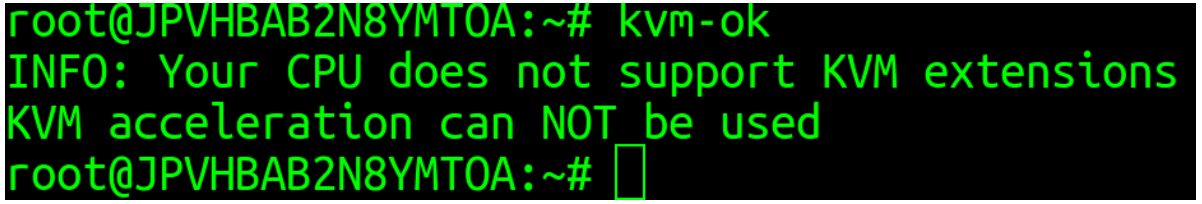
- Documentazione progetto virsh e qemu

Check virtualization support

- Check hardware virtualization support:

```
Sudo apt update
sudo apt install cpu-checker
kvm-ok
```

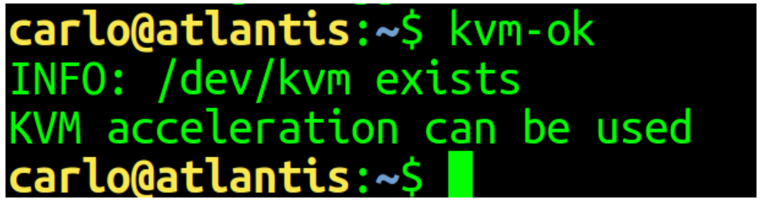
- The VMs does not have access to the hardware acceleration, which is accessed by the VM you are connecting to
- This is OK, the VMs running inside your VM will run slower



```
root@JPVHBAB2N8YMTOA:~# kvm-ok
INFO: Your CPU does not support KVM extensions
KVM acceleration can NOT be used
root@JPVHBAB2N8YMTOA:~# 
```

Without Hardware Acceleration

With Hardware Acceleration

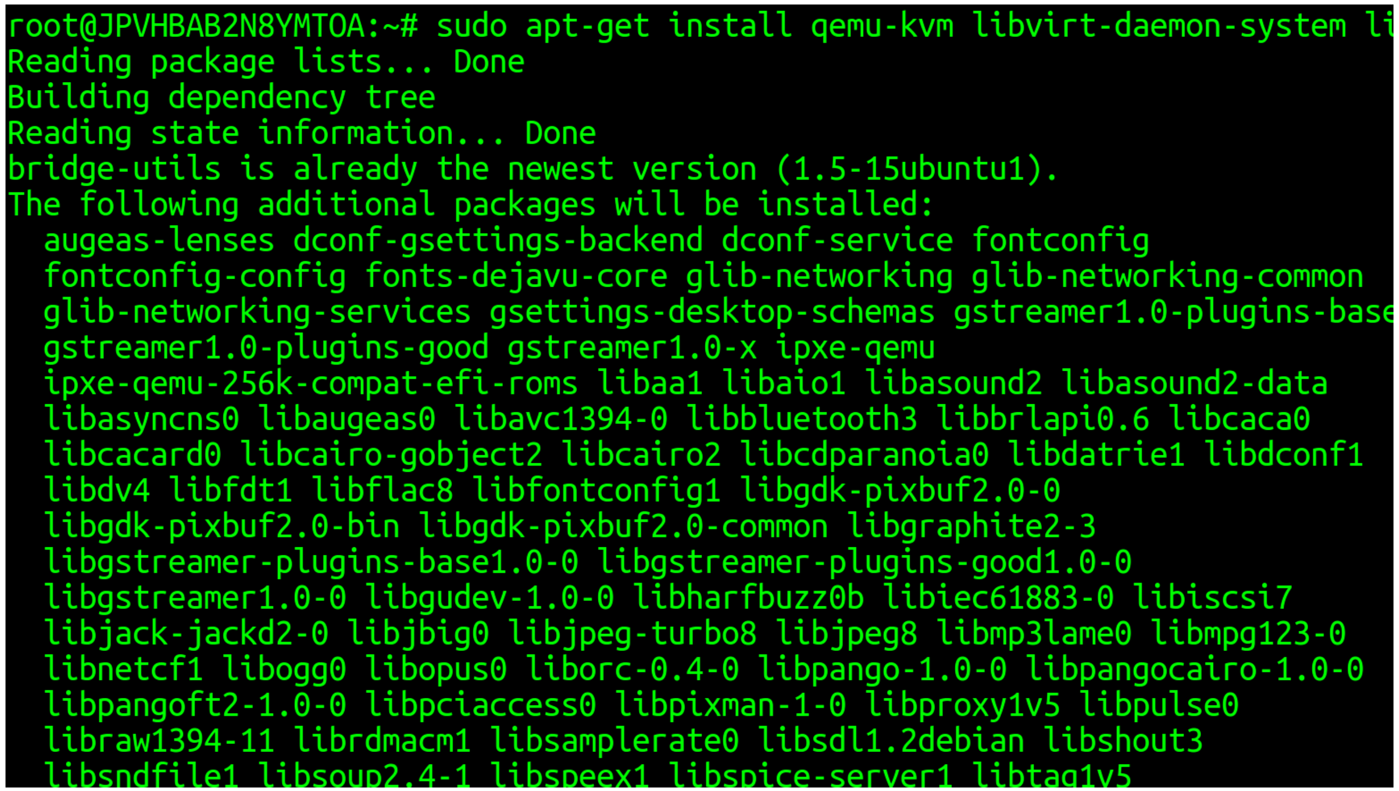


```
carlo@atlantis:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
carlo@atlantis:~$ 
```

Install KVM and libvirt

- Install KVM and libvirt software

```
sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-
clients bridge-utils virtinst
```



```
root@JPVHBAB2N8YMTOA:~# sudo apt-get install qemu-kvm libvirt-daemon-system li
Reading package lists... Done
Building dependency tree
Reading state information... Done
bridge-utils is already the newest version (1.5-15ubuntu1).
The following additional packages will be installed:
  augeas-lenses dconf-gsettings-backend dconf-service fontconfig
  fontconfig-config fonts-dejavu-core glib-networking glib-networking-common
  glib-networking-services gsettings-desktop-schemas gstreamer1.0-plugins-base
  gstreamer1.0-plugins-good gstreamer1.0-x ipxe-qemu
  ipxe-qemu-256k-compat-efi-roms libaa1 libaio1 libasound2 libasound2-data
  libasyncns0 libaugeas0 libavc1394-0 libbluetooth3 libbrlapi0.6 libcaca0
  libcacard0 libcairo-gobject2 libcairo2 libcdparanoia0 libdatrie1 libdconf1
  libdv4 libfdt1 libflac8 libfontconfig1 libgdk-pixbuf2.0-0
  libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgraphite2-3
  libgstreamer-plugins-base1.0-0 libgstreamer-plugins-good1.0-0
  libgstreamer1.0-0 libgudev-1.0-0 libharfbuzz0b libiec61883-0 libiscsi7
  libjack-jackd2-0 libjbig0 libjpeg-turbo8 libjpeg8 libmp3lame0 libmpg123-0
  libnetcf1 libogg0 libopus0 liborc-0.4-0 libpango-1.0-0 libpangocairo-1.0-0
  libpangoft2-1.0-0 libpciaccess0 libpixman-1-0 libproxy1v5 libpulse0
  libraw1394-11 librdmacm1 libsamplerate0 libsdl1.2debian libshout3
  libsndfile1 libsoup2.4-1 libspeex1 libspice-server1 libtag1v5
```

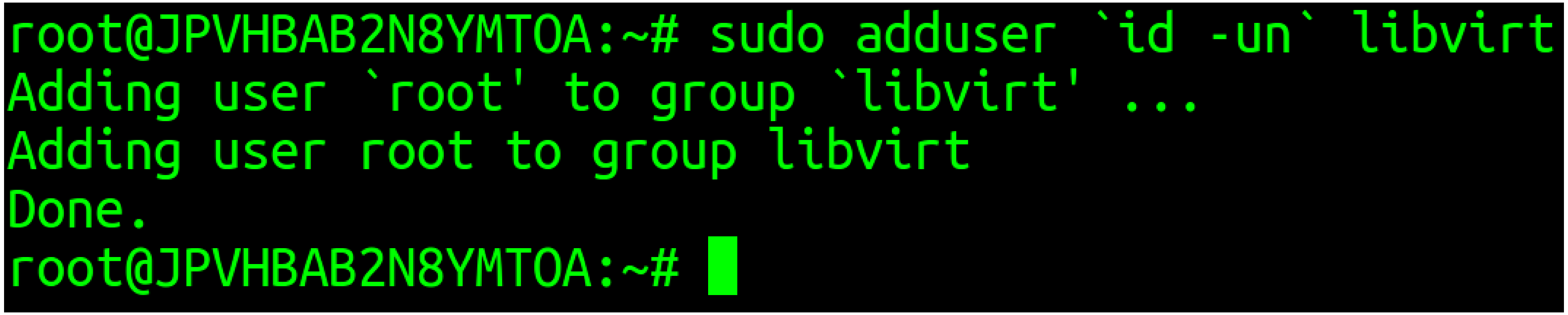
This will take some time.

- KVM is the virtualization module for the linux kernel, it functions as Hypervisor
- LibVirt is an opensource tool for managing platform virtualization (create and configure VMs)
- <u>KVM allows the creation of VMs, LibVirt manages them!</u>

Configuration

- Complete the configuration by adding the user to libvirt group and allows your user to use LibVirt (root is already allowed…)

```
sudo adduser `id -un` libvirt
```



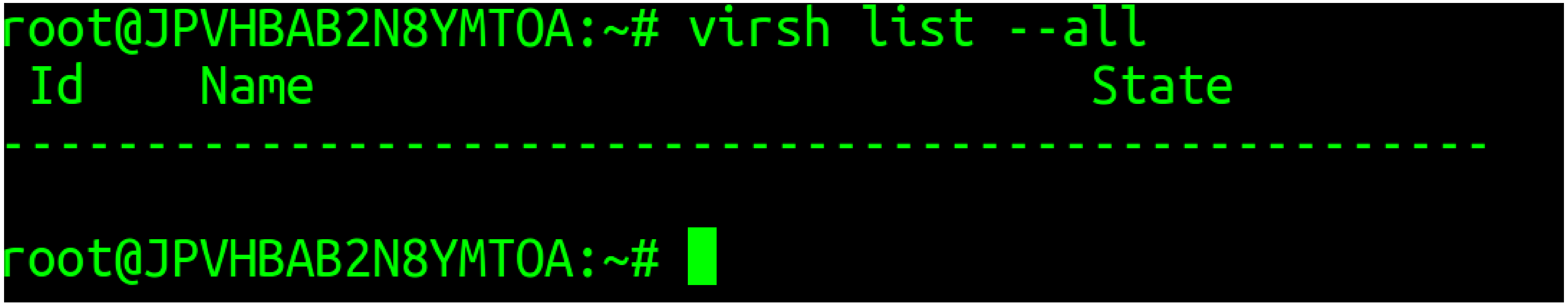
```
root@JPVHBAB2N8YMTOA:~# sudo adduser `id -un` libvirt
Adding user `root' to group `libvirt' ...
Adding user root to group libvirt
Done.
root@JPVHBAB2N8YMTOA:~#
```

Check Configuration

- Check that everything is working fine

  `virsh list`

- This is the list of the VMs running on the platofrm (none so far)

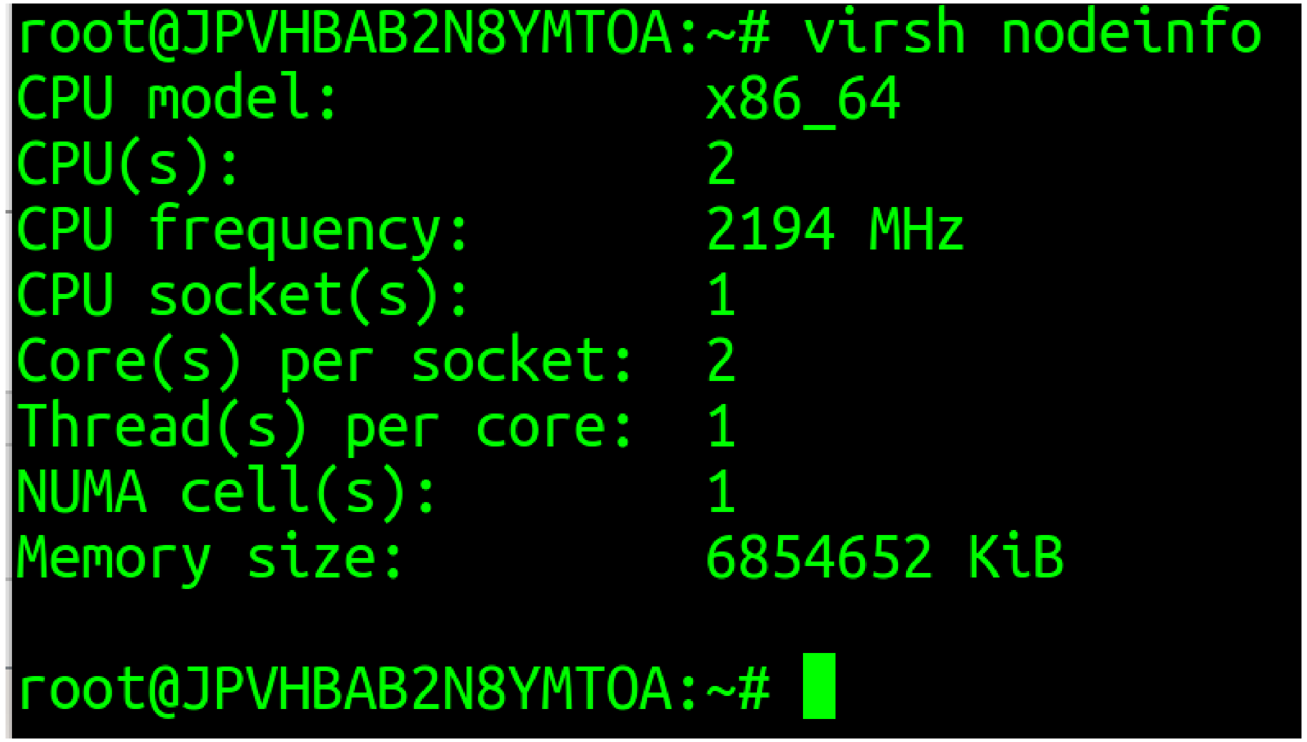


```
root@JPVHBAB2N8YMTOA:~# virsh list --all
 Id    Name                           State
----------------------------------------------------

root@JPVHBAB2N8YMTOA:~# 
```

Gather information on the system

- Let us retrieve some info on the hardware

  `virsh nodeinfo`



```
root@JPVHBAB2N8YMTOA:~# virsh nodeinfo
CPU model:           x86_64
CPU(s):              2
CPU frequency:       2194 MHz
CPU socket(s):       1
Core(s) per socket:  2
Thread(s) per core:  1
NUMA cell(s):        1
Memory size:         6854652 KiB

root@JPVHBAB2N8YMTOA:~#
```

Install a VM from network

- A new VM can be installed in many ways. One possibility is to install the VM from scratch using the installation disk (iso) or fetching the installation packages from Internet
- Let’s try the latter

```
sudo virt-install \
--name Ubuntu \
--description 'Test VM with Ubuntu' \
--ram=512 \
--vcpus=1 \
--os-type=Linux \
--os-variant=ubuntu18.04 \
--disk path=/var/lib/libvirt/images/ubuntu1804.qcow2,bus=virtio,size=1 \
--graphics none \
--location 'http://us.archive.ubuntu.com/ubuntu/dists/bionic/main/installer-amd64/' \
--network bridge:virbr0  \
--console pty,target_type=serial -x 'console=ttyS0,115200n8 serial'
```

Name of the VM

Description

RAM in MB

VCPUs

OS Type

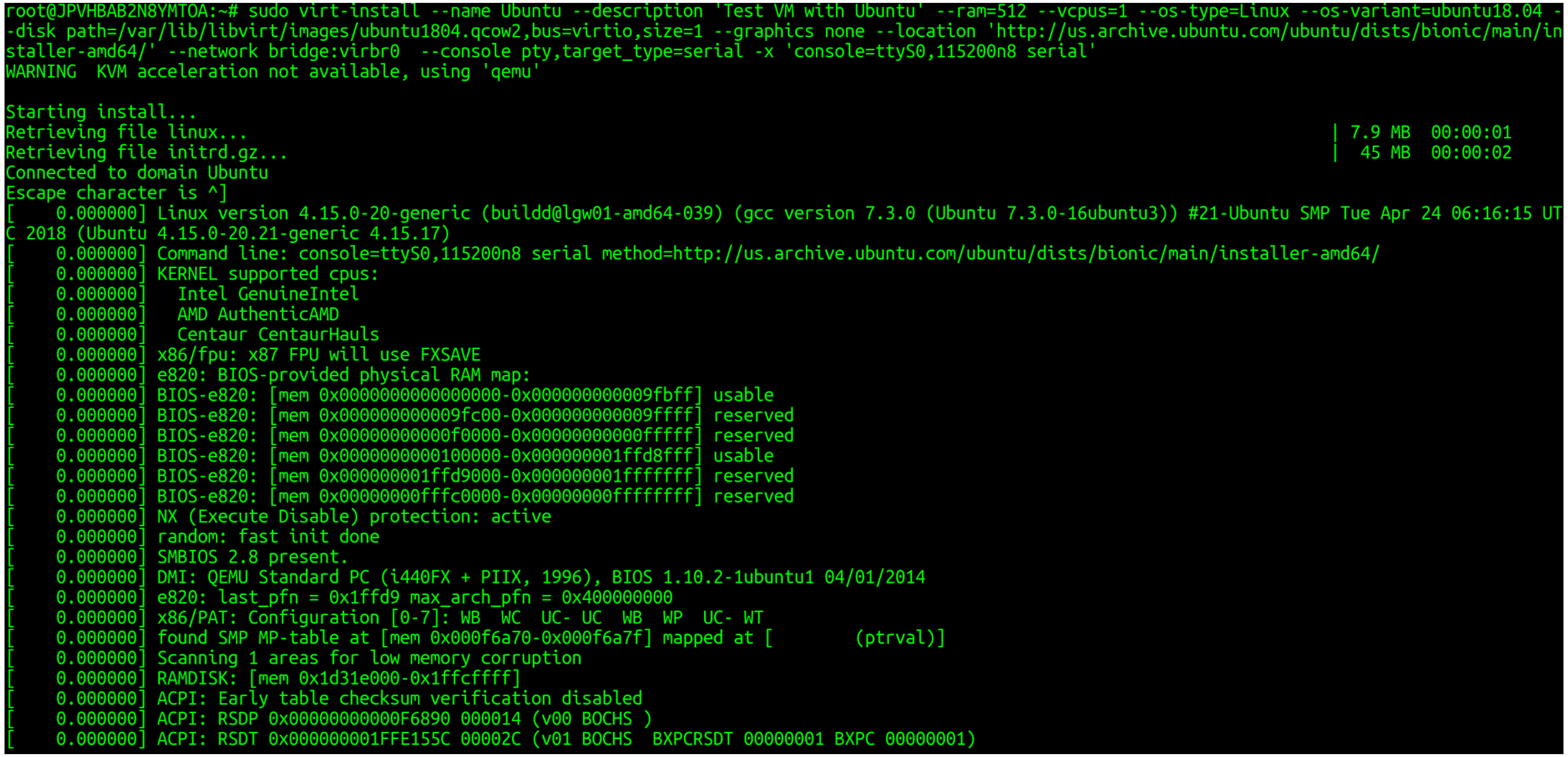
Hard Disk

Type of Graphics (no graphic at this time)

Installation source

Network Configuration

Installation in progress



```
root@JPVHBAB2N8YMTOA:~# sudo virt-install --name Ubuntu --description 'Test VM with Ubuntu' --ram=512 --vcpus=1 --os-type=Linux --os-variant=ubuntu18.04 -
-disk path=/var/lib/libvirt/images/ubuntu1804.qcow2,bus=virtio,size=1 --graphics none --location 'http://us.archive.ubuntu.com/ubuntu/dists/bionic/main/in
staller-amd64/' --network bridge:virbr0  --console pty,target_type=serial -x 'console=ttyS0,115200n8 serial'
WARNING  KVM acceleration not available, using 'qemu'

Starting install...
Retrieving file linux...                                                                                                                  | 7.9 MB  00:00:01
Retrieving file initrd.gz...                                                                                                              |  45 MB  00:00:02
Connected to domain Ubuntu
Escape character is ^]
[    0.000000] Linux version 4.15.0-20-generic (buildd@lgw01-amd64-039) (gcc version 7.3.0 (Ubuntu 7.3.0-16ubuntu3)) #21-Ubuntu SMP Tue Apr 24 06:16:15 UT
C 2018 (Ubuntu 4.15.0-20.21-generic 4.15.17)
[    0.000000] Command line: console=ttyS0,115200n8 serial method=http://us.archive.ubuntu.com/ubuntu/dists/bionic/main/installer-amd64/
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
[    0.000000] x86/fpu: x87 FPU will use FXSAVE
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x000000001ffd8fff] usable
[    0.000000] BIOS-e820: [mem 0x000000001ffd9000-0x000000001fffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] random: fast init done
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.10.2-1ubuntu1 04/01/2014
[    0.000000] e820: last_pfn = 0x1ffd9 max_arch_pfn = 0x400000000
[    0.000000] x86/PAT: Configuration [0-7]: WB  WC  UC- UC  WB  WP  UC- WT
[    0.000000] found SMP MP-table at [mem 0x000f6a70-0x000f6a7f] mapped at [        (ptrval)]
[    0.000000] Scanning 1 areas for low memory corruption
[    0.000000] RAMDISK: [mem 0x1d31e000-0x1ffcffff]
[    0.000000] ACPI: Early table checksum verification disabled
[    0.000000] ACPI: RSDP 0x00000000000F6890 000014 (v00 BOCHS )
[    0.000000] ACPI: RSDT 0x000000001FFE155C 00002C (v01 BOCHS  BXPCRSDT 00000001 BXPC 00000001)
```

Delete a VM

- Let us halt the installation process by destroying the VM

```
virsh destroy Ubuntu
virsh undefine Ubuntu
```

Import a preinstalled VM

- Usually the best installation method is to import a preinstalled VM
- Let us first download the HD image, which includes the OS and SW (CirrOS is a Linux distrubition with small footprint)

```
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-
disk.img
```

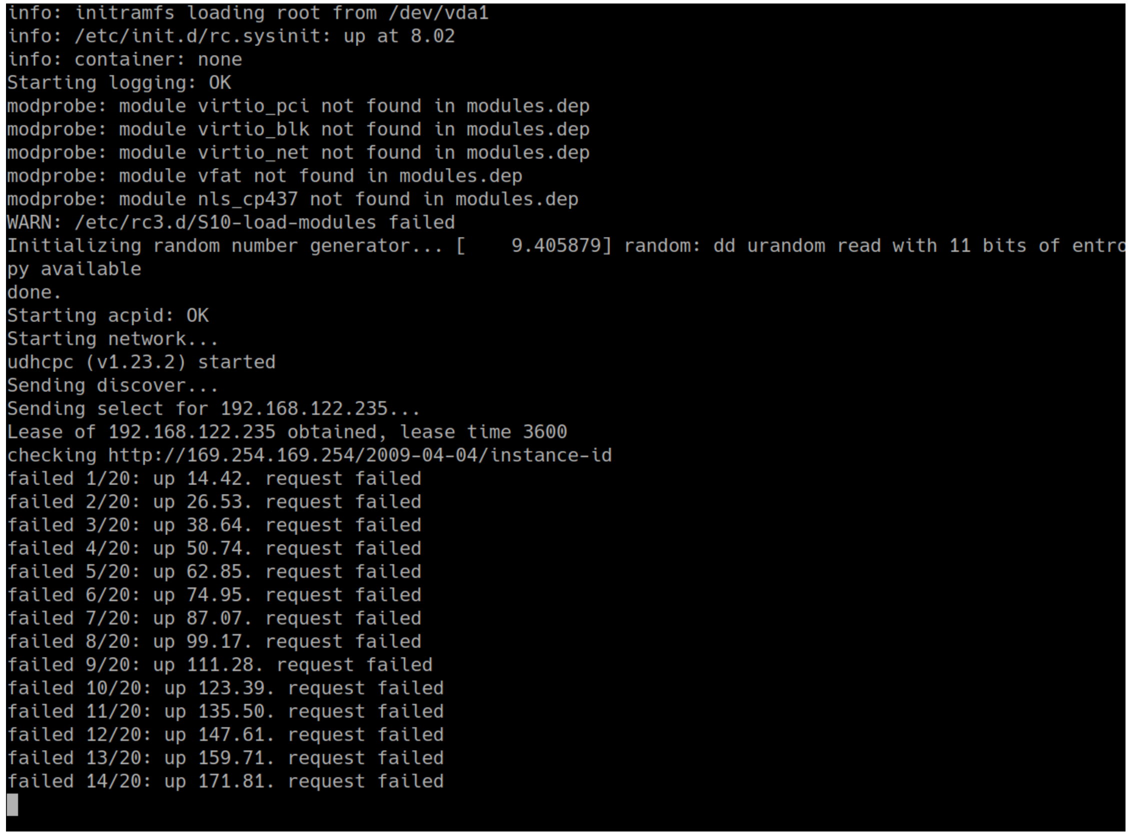
```
mv cirros-0.4.0-x86_64-disk.img /var/lib/libvirt/images
```

- Import the VM in the system

```
sudo virt-install --name Cirros --description 'Cirros' --ram=512
--vcpus=1 --os-type=Linux --os-variant=ubuntu18.04 --disk
path=/var/lib/libvirt/images/cirros-0.4.0-x86_64-
disk.img,bus=virtio,format=raw --network bridge:virbr0 --graphic
none  --import
```

Boot

- The boot can take time (the image is produced for another virtualization platform and at boot performs some operations that cannot be succeeded)

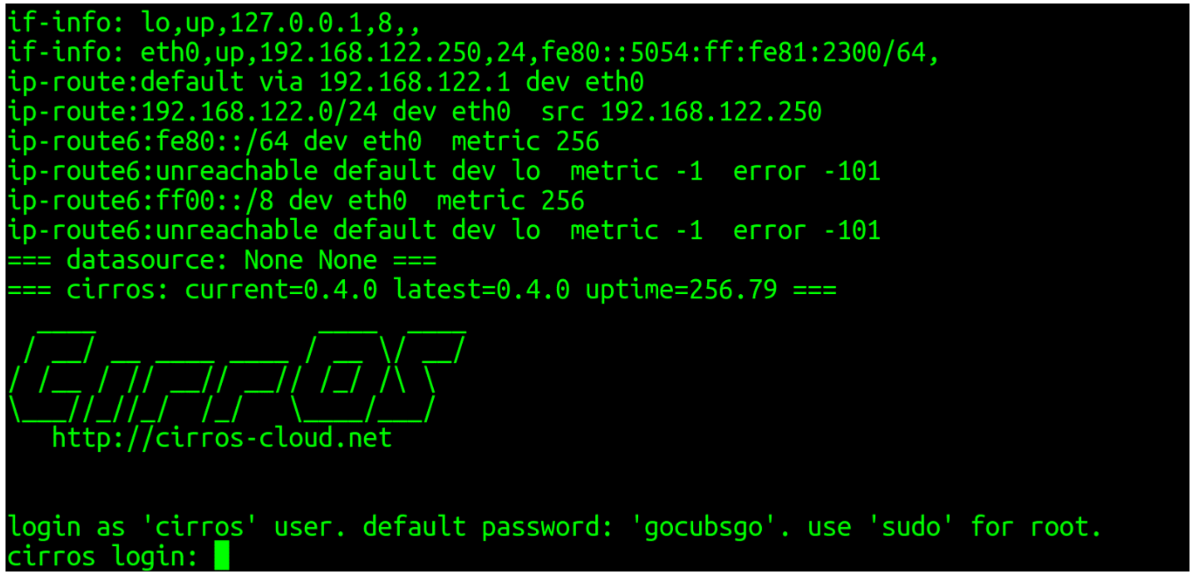


```
info: initramfs loading root from /dev/vda1
info: /etc/init.d/rc.sysinit: up at 8.02
info: container: none
Starting logging: OK
modprobe: module virtio_pci not found in modules.dep
modprobe: module virtio_blk not found in modules.dep
modprobe: module virtio_net not found in modules.dep
modprobe: module vfat not found in modules.dep
modprobe: module nls_cp437 not found in modules.dep
WARN: /etc/rc3.d/S10-load-modules failed
Initializing random number generator... [    9.405879] random: dd urandom read with 11 bits of entro
py available
done.
Starting acpid: OK
Starting network...
udhcpc (v1.23.2) started
Sending discover...
Sending select for 192.168.122.235...
Lease of 192.168.122.235 obtained, lease time 3600
checking http://169.254.169.254/2009-04-04/instance-id
failed 1/20: up 14.42. request failed
failed 2/20: up 26.53. request failed
failed 3/20: up 38.64. request failed
failed 4/20: up 50.74. request failed
failed 5/20: up 62.85. request failed
failed 6/20: up 74.95. request failed
failed 7/20: up 87.07. request failed
failed 8/20: up 99.17. request failed
failed 9/20: up 111.28. request failed
failed 10/20: up 123.39. request failed
failed 11/20: up 135.50. request failed
failed 12/20: up 147.61. request failed
failed 13/20: up 159.71. request failed
failed 14/20: up 171.81. request failed
```

Login

- The import should start the VM and show the console
- If the console is not shown (or you want to connect after you disconnected, e.g. you closed the window) you have to run the following
- Connect to the console (if not shown)

```
virsh  console Cirros
```



```
if-info: lo,up,127.0.0.1,8,,
if-info: eth0,up,192.168.122.250,24,fe80::5054:ff:fe81:2300/64,
ip-route:default via 192.168.122.1 dev eth0
ip-route:192.168.122.0/24 dev eth0  src 192.168.122.250
ip-route6:fe80::/64 dev eth0  metric 256
ip-route6:unreachable default dev lo  metric -1  error -101
ip-route6:ff00::/8 dev eth0  metric 256
ip-route6:unreachable default dev lo  metric -1  error -101
=== datasource: None None ===
=== cirros: current=0.4.0 latest=0.4.0 uptime=256.79 ===
   ____               ____  ____
  / __/ __ ____ ____ / __ \/ __/
 / /__ / // __// __// /_/ /\ \
 \___//_//_/  /_/   \____/___/
   http://cirros-cloud.net


login as 'cirros' user. default password: 'gocubsgo'. use 'sudo' for root.
cirros login:
```

Manage the VM

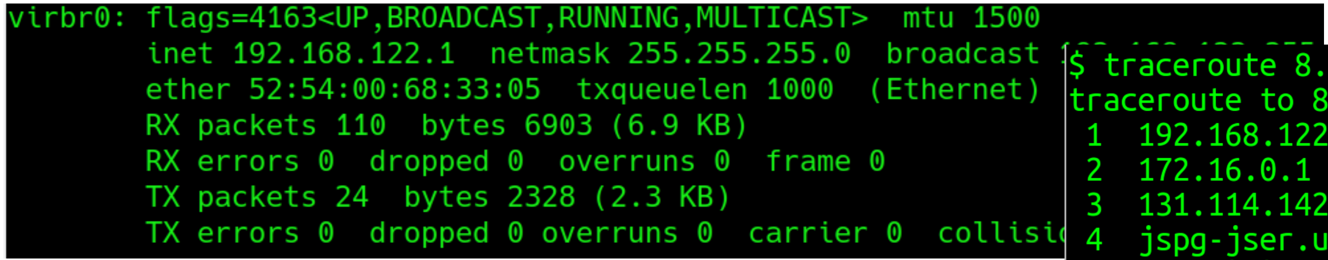
- The VM can be halted or suspended with the following:

```
virsh shutdown Cirros

virsh suspend Cirros

virsh autostart Cirros (to set autostart at Host OS boot)
```

Network

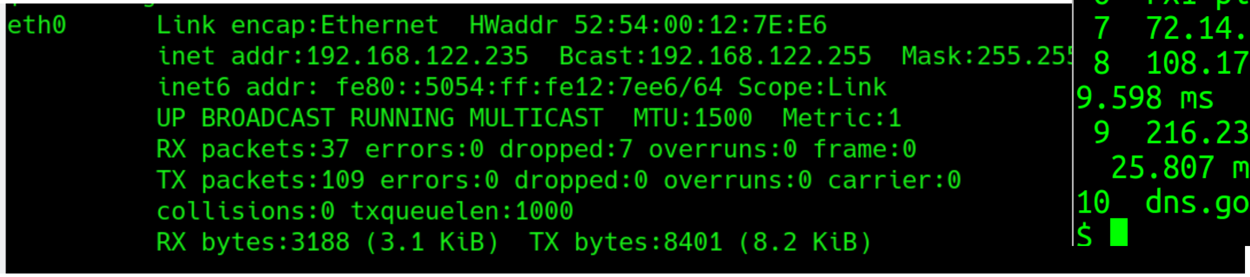
- VM interface added to the bridge virbr0
- LibVirt configures the interface to offer NAT services to VM
- If you run traceroute inside the VM you can see that the traffic goes through 192.168.122.1, the address of virbr0

Host OS

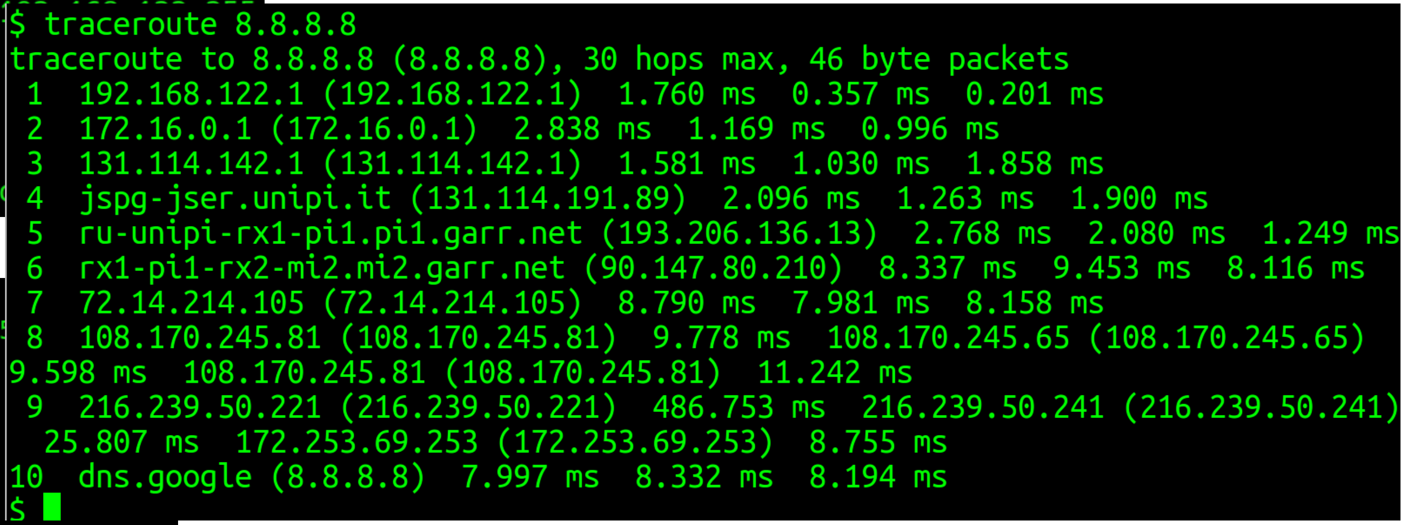


```
virbr0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast
        ether 52:54:00:68:33:05  txqueuelen 1000  (Ethernet)
        RX packets 110  bytes 6903 (6.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 2328 (2.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisi
```

Guest OS



```
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:7E:E6
          inet addr:192.168.122.235  Bcast:192.168.122.255  Mask:255.255
          inet6 addr: fe80::5054:ff:fe12:7ee6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:7 overruns:0 frame:0
          TX packets:109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3188 (3.1 KiB)  TX bytes:8401 (8.2 KiB)
```



```
$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 46 byte packets
 1  192.168.122.1 (192.168.122.1)  1.760 ms  0.357 ms  0.201 ms
 2  172.16.0.1 (172.16.0.1)  2.838 ms  1.169 ms  0.996 ms
 3  131.114.142.1 (131.114.142.1)  1.581 ms  1.030 ms  1.858 ms
 4  jspg-jser.unipi.it (131.114.191.89)  2.096 ms  1.263 ms  1.900 ms
 5  ru-unipi-rx1-pi1.pi1.garr.net (193.206.136.13)  2.768 ms  2.080 ms  1.249 ms
 6  rx1-pi1-rx2-mi2.mi2.garr.net (90.147.80.210)  8.337 ms  9.453 ms  8.116 ms
 7  72.14.214.105 (72.14.214.105)  8.790 ms  7.981 ms  8.158 ms
 8  108.170.245.81 (108.170.245.81)  9.778 ms  108.170.245.65 (108.170.245.65)
9.598 ms  108.170.245.81 (108.170.245.81)  11.242 ms
 9  216.239.50.221 (216.239.50.221)  486.753 ms  216.239.50.241 (216.239.50.241)
  25.807 ms  172.253.69.253 (172.253.69.253)  8.755 ms
10  dns.google (8.8.8.8)  7.997 ms  8.332 ms  8.194 ms
$ 
```

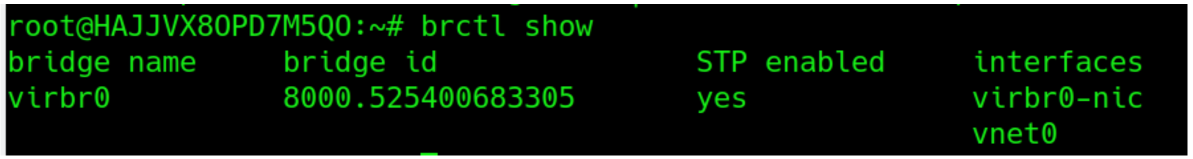
Linux Bridges

- A linux bridge is a virtual interface that bridges different interfaces of a linux system (real interfaces or virtual interfaces)
- The bridge is performed at layer 2, so it is like connecting different networks through a switch
- Check the bridges of the host system:

```
brctl show
```

The system has one bridge created by LibVirt to connect VMs

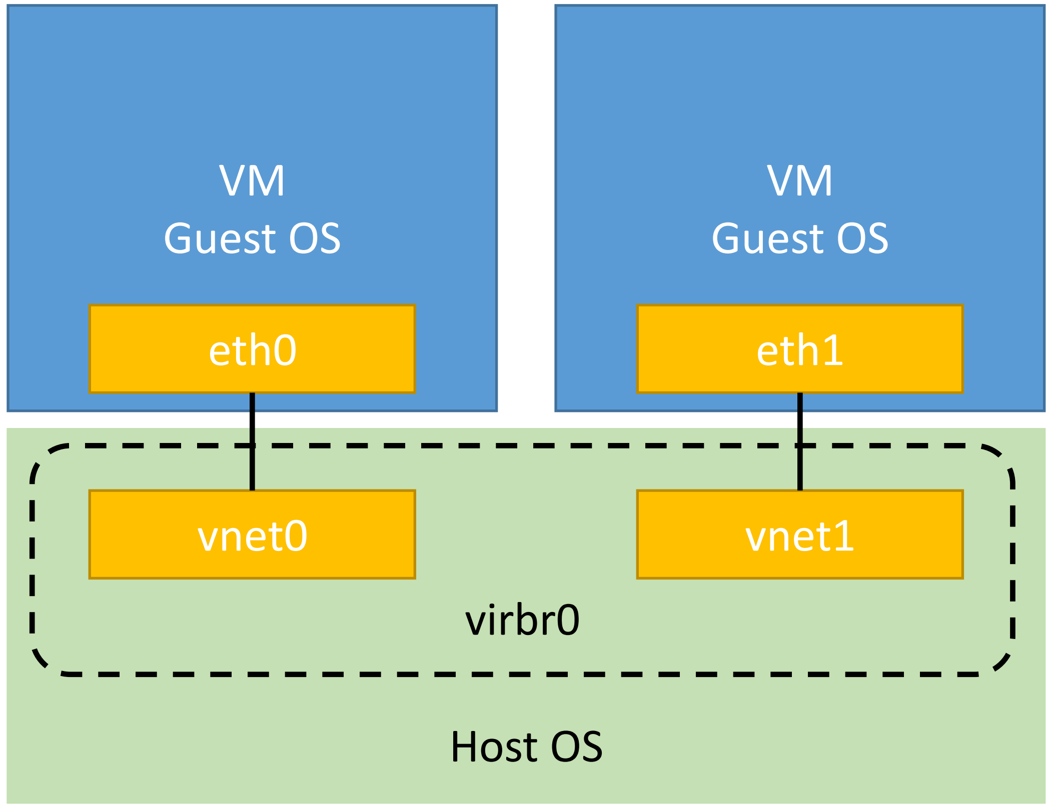
The virbr0 merges two networks: vnet0 and virbr0-nic
vnet0 connects to eth0 of the VM directly
virbr0-nic is a virtual network that is used to connect all the VM hosted by the local LibVirt instance



```
root@HAJJVX8OPD7M5QO:~# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.525400683305       yes             virbr0-nic
                                                        vnet0
```

Libvirt networking

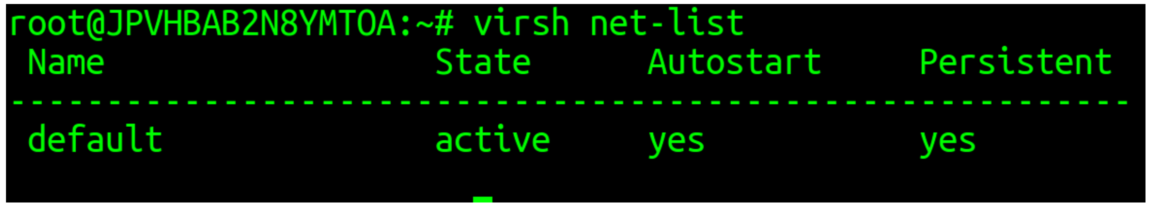
- The virtual NIC of each VM created by the hypervisor has a corresponding virtual interface on the Host OS
- To create a virtual network all the virtual interfaces can be added to a bridge, so data is exchanged among different VMs
- The bridge can be connected to a physical interface to allow data exchange with external networks



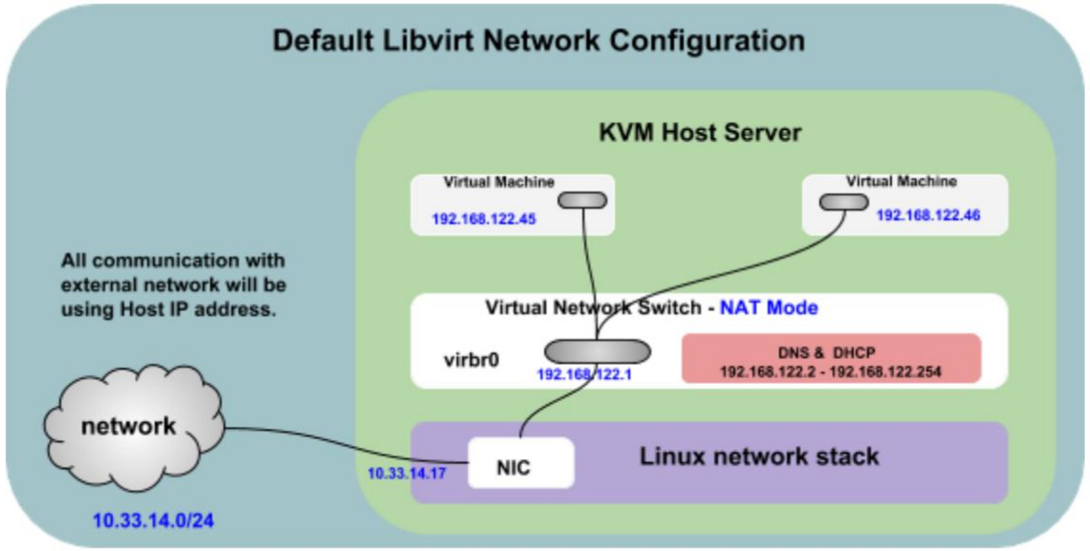
Default Network (NAT)

- By default the VMs are connected to an internal network and get internet access via NAT
- Check networks available

```
virsh net-list
```



```
root@JPVHBAB2N8YMTOA:~# virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
```



Define a new network

- The creation of a new network is managed by LibVirt through the command virsh net-define
- The specifications of the new network must be described in a XML file
- A new internal network with NAT (like the default one) is described as follow (check by running `virsh net-dumpxml default`):

```
<network>
  <name>default</name>
  <bridge name="virbr0"/>
  <forward mode="nat"/>
  <ip address="192.168.122.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2" end="192.168.122.254"/>
    </dhcp>
  </ip>
</network>
```

Define another NAT network

```
<network>
  <name>net2</name>
  <bridge name="virbr1"/>
  <forward mode="nat"/>
  <ip address="192.168.2.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.2.2" end="192.168.2.254"/>
    </dhcp>
  </ip>
</network>
```

Create a new network

- Define the network starting from the XML

```
virsh net-define net2.xml
```

- Set the new network to start automatically

```
virsh net-autostart net2
```

- Start (manually this time)

```
virsh net-start net2
```

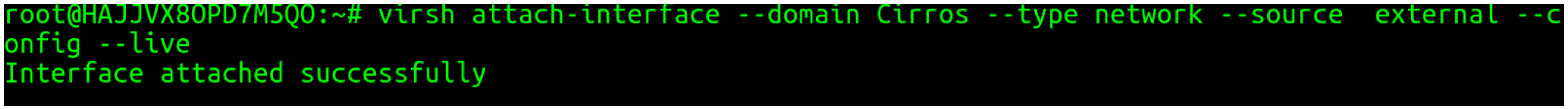
Connect a VM to another network

- Create a new virtual NIC for the VM and attach it to the new network

```
virsh attach-interface --domain Cirros --type network --source  net2 --
config --live
```

- Configure the NIC on the VM

```
sudo ip addr add IP_ADDRESS/MASK dev eth1
sudo ip link set up dev eth1
```



```
root@HAJJVX8OPD7M5QO:~# virsh attach-interface --domain Cirros --type network --source  external --c
onfig --live
Interface attached successfully
```

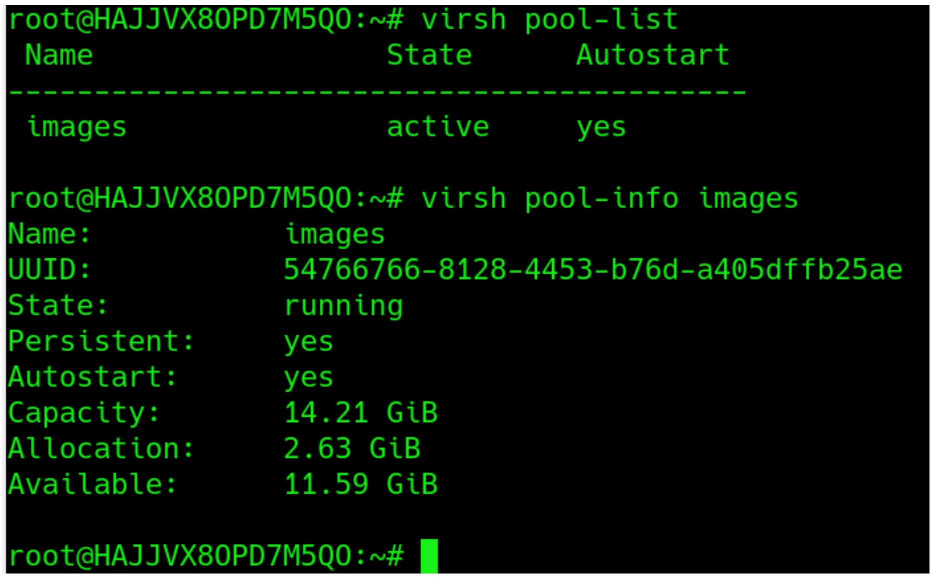
Storage Pool

- Storage for VM hard drives is offered by storage pool
- Each storage pool allows to create different volumes, i.e. different virtual hard drives that can be attached to VMs
- Check the list of pools available via:

```
virsh pool-list
```

- Check the status of the images pool via:

```
virsh pool-info images
```



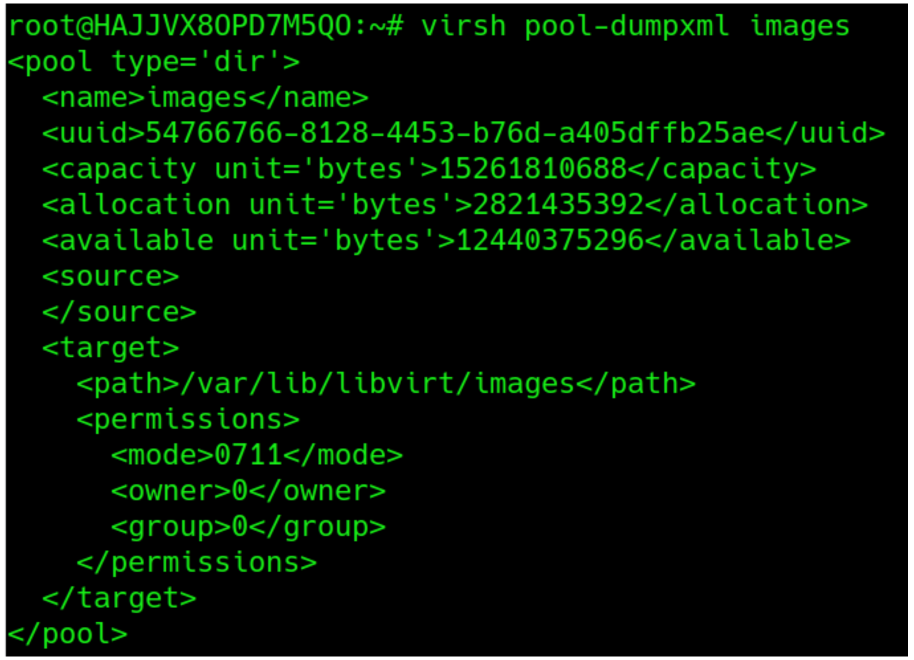
```
root@HAJJVX80PD7M5Q0:~# virsh pool-list
 Name                 State      Autostart
-------------------------------------------
 images               active     yes

root@HAJJVX80PD7M5Q0:~# virsh pool-info images
Name:           images
UUID:           54766766-8128-4453-b76d-a405dffb25ae
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       14.21 GiB
Allocation:     2.63 GiB
Available:      11.59 GiB

root@HAJJVX80PD7M5Q0:~# 
```

Storage pool configuration

- Pools can be created using an existing file system or a whole partition
- To check the configuration of the images pool run

  `virsh pool-dumpxml images`

- *The images pool create complete image (bit by bit) of the virtual HD into a single file*



```
root@HAJJVX8OPD7M5Q0:~# virsh pool-dumpxml images
<pool type='dir'>
  <name>images</name>
  <uuid>54766766-8128-4453-b76d-a405dffb25ae</uuid>
  <capacity unit='bytes'>15261810688</capacity>
  <allocation unit='bytes'>2821435392</allocation>
  <available unit='bytes'>12440375296</available>
  <source>
  </source>
  <target>
    <path>/var/lib/libvirt/images</path>
    <permissions>
      <mode>0711</mode>
      <owner>0</owner>
      <group>0</group>
    </permissions>
  </target>
</pool>
```

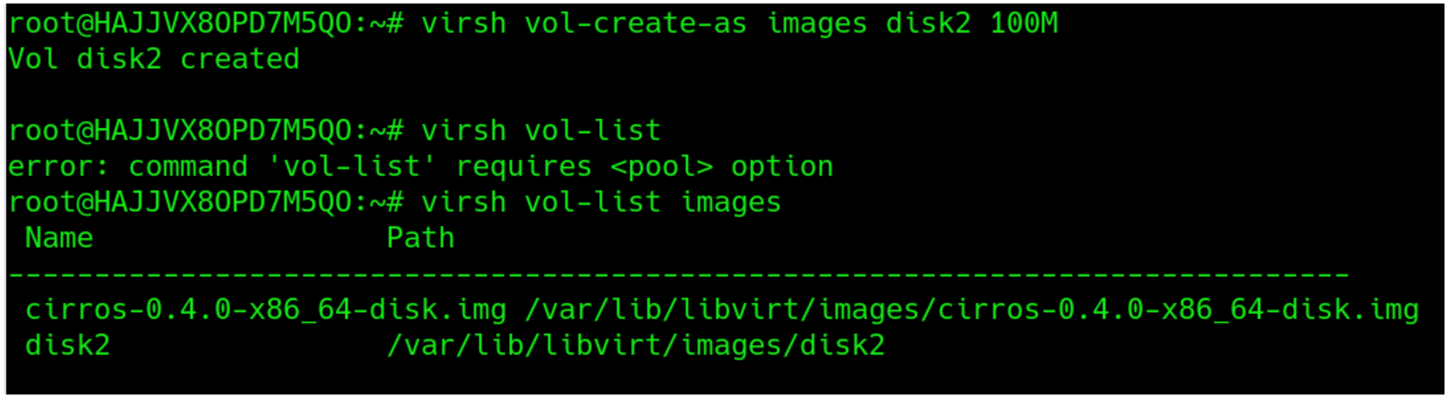
Create a new volume

- Create a new volume in a pool:

```
virsh vol-create-as images disk2 100M
```

- Check the new volume:

```
virsh vol-list images
```



```
root@HAJJVX80PD7M5Q0:~# virsh vol-create-as images disk2 100M
Vol disk2 created

root@HAJJVX80PD7M5Q0:~# virsh vol-list
error: command 'vol-list' requires <pool> option
root@HAJJVX80PD7M5Q0:~# virsh vol-list images
 Name                      Path
------------------------------------------------------------------------------
 cirros-0.4.0-x86_64-disk.img /var/lib/libvirt/images/cirros-0.4.0-x86_64-disk.img
 disk2                     /var/lib/libvirt/images/disk2
```

- Attach the disk to the VM

```
virsh attach-disk Cirros /var/lib/libvirt/images/disk2 vdb --live
```

Use the disk in the VM

- **On the guest VM**
  - Create partitions

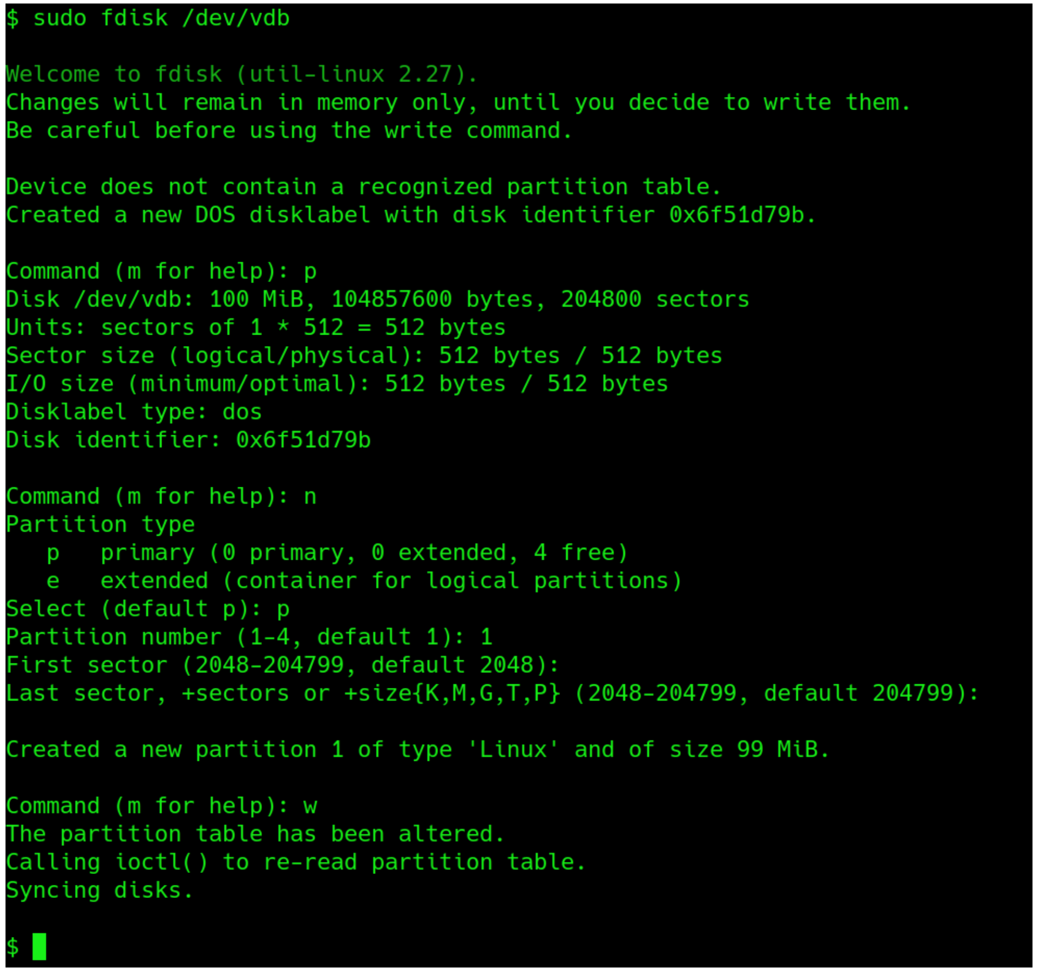
    ```
    sudo fdisk /dev/vdb
    ```

  - Format the partition

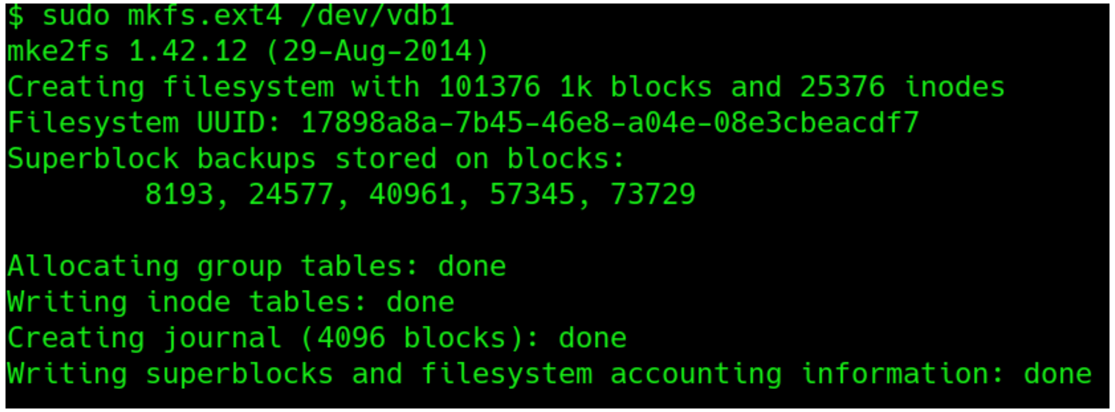
    ```
    sudo mkfs.ext4 /dev/vdb1
    ```

  - Mount and use the partition

    ```
    sudo mount /dev/vdb1 /mnt/
    ```



```
$ sudo fdisk /dev/vdb

Welcome to fdisk (util-linux 2.27).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x6f51d79b.

Command (m for help): p
Disk /dev/vdb: 100 MiB, 104857600 bytes, 204800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x6f51d79b

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-204799, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-204799, default 204799):

Created a new partition 1 of type 'Linux' and of size 99 MiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

$
```



```
$ sudo mkfs.ext4 /dev/vdb1
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 101376 1k blocks and 25376 inodes
Filesystem UUID: 17898a8a-7b45-46e8-a04e-08e3cbeacdf7
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Clean UP!

- Destroy all the VMs

```
virsh destroy Cirros
virsh undefine Cirros
```

- Destroy the networks

```
virsh net-destroy net2
```

- Uninstall

```
sudo apt-get purge qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virtinst
reboot
```

Create a new network with direct external connection

- Create a new bridge with a physical interface

```
virsh iface-bridge --interface eth1 --bridge ext-br
```

- Create a new external network with the following definition

```
<network>
        <name>external</name>
        <forward mode="bridge" />
        <bridge name="ext-br" />
</network>
```

<u>Don’t try this on the VM! This commands are only for servers with at least two network adapters. If you try those on the VM you’ll be kicked out!</u>