CLOUD COMPUTING HOMEWORK #1 – VIRTUALIZATION

JUAN MANUEL ZULUAGA FERNANDEZ.

1. QEMU Installation and VM configuration.

1.1 First step to install qemu in our device is to use the command sudo apt-get install as can bee seen in the image below. Prior to this, we should do sudo apt-get update in case is needed.

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo apt-get install qemu-system-x86
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  cpu-checker ipxe-qemu ipxe-qemu-256k-compat-efi-roms libcacard0 libdecor-0-0 libdecor-0-plugin-1-cairo libfdt1 libsdl2-2.0-0 libslirp0
  libspice-server1 libusbredirparser1 libvirglrenderer1 msr-tools ovmf qemu-system-common qemu-system-data qemu-system-gui seabios
Suggested packages:
  gstreamer1.0-libav gstreamer1.0-plugins-ugly samba vde2
The following NEW packages will be installed:
  cpu-checker ipxe-qemu ipxe-qemu-256k-compat-efi-roms libcacard0 libdecor-0-0 libdecor-0-plugin-1-cairo libfdt1 libsdl2-2.0-0 libslirp0
  libspice-server1 libusbredirparser1 libvirglrenderer1 msr-tools ovmf qemu-system-common qemu-system-data qemu-system-gui qemu-system-x86
  seabios
0 upgraded, 19 newly installed, 0 to remove and 150 not upgraded.
Need to get 24.9 MB of archives.
After this operation, 83.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 msr-tools amd64 1.3-4 [10.3 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 cpu-checker amd64 0.7-1.3build1 [6,800 B]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 ipxe-qemu all 1.21.1+git-20220113.fbbdc3926-0ubuntu1 [1,569 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 ipxe-qemu-256k-compat-efi-roms all 1.0.0+git-20150424.a25a16d-0ubuntu4 [552 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libcacard0 amd64 1:2.8.0-3build2 [38.0 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libdecor-0-0 amd64 0.1.0-3build1 [15.1 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libdecor-0-plugin-1-cairo amd64 0.1.0-3build1 [20.4 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsdl2-2.0-0 amd64 2.0.20+dfsg-2ubuntu1.22.04.1 [582 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-1build1 [61.5 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libspice-server1 amd64 0.15.0-2ubuntu4 [351 kB]
```

1.2 We also install qemu utils in case we need them in the future.

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo apt-get install qemu-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ibverbs-providers libaio1 libdaxctl1 libgfapi0 libgfrpc0 libgfxdr0 libglusterfs0 libibverbs1 libiscsi7 libndctl6 libpmem1 libpmemobj1
  librados2 librbd1 librdmacm1 liburing2 qemu-block-extra
Suggested packages:
  debootstrap
The following NEW packages will be installed:
  ibverbs-providers libaio1 libdaxctl1 libgfapi0 libgfrpc0 libgfxdr0 libglusterfs0 libibverbs1 libiscsi7 libndctl6 libpmem1 libpmemobj1
  librados2 librbd1 librdmacm1 liburing2 qemu-block-extra qemu-utils
0 upgraded, 18 newly installed, 0 to remove and 150 not upgraded.
Need to get 9,967 kB of archives.
After this operation, 41.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libibverbs1 amd64 39.0-1 [69.3 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 ibverbs-providers amd64 39.0-1 [341 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libaio1 amd64 0.3.112-13build1 [7,176 B]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libdaxctl1 amd64 72.1-1 [19.8 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libgfxdr0 amd64 10.1-1 [22.1 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libglusterfs0 amd64 10.1-1 [288 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libgfrpc0 amd64 10.1-1 [47.1 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libgfapi0 amd64 10.1-1 [77.3 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 librdmacm1 amd64 39.0-1 [71.2 kB]
```

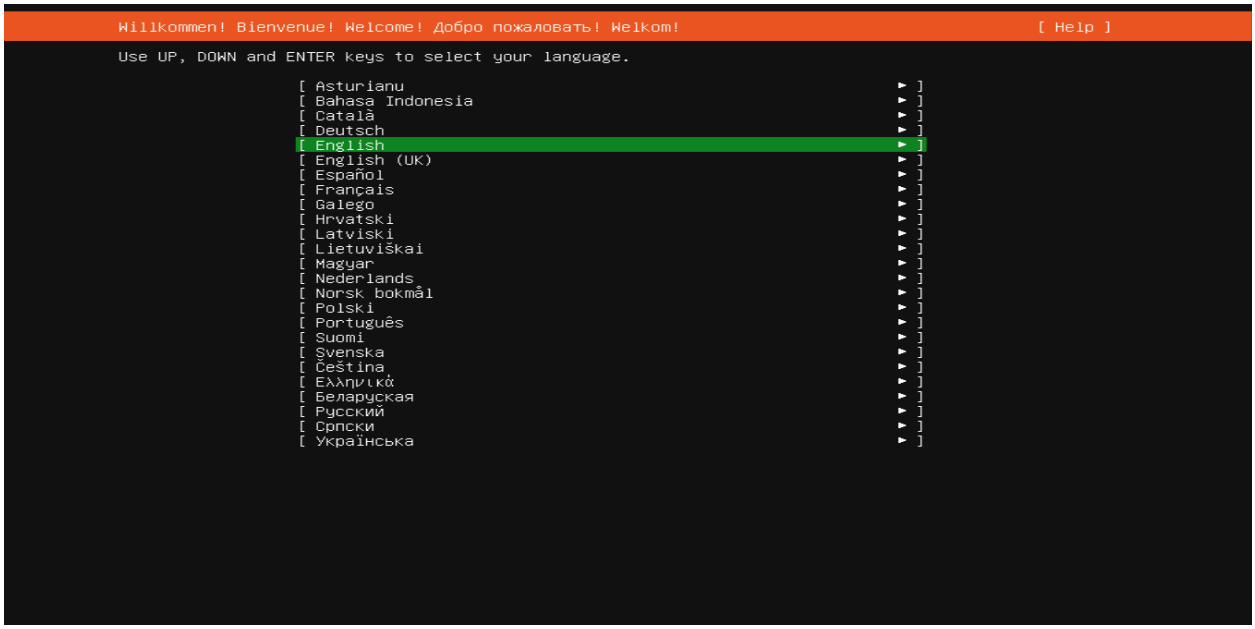
1.3 We create our image, the one we are going to use to run the VM

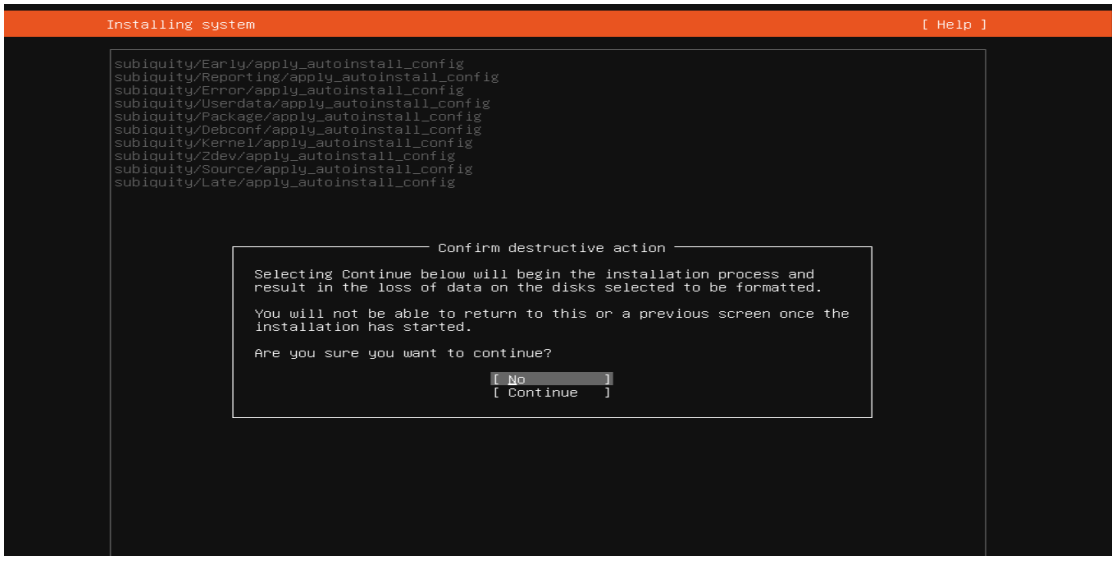
```
juan@juan-IdeaPad-3-15IIL05:~$ sudo qemu-img create ubuntu.img 10G -f qcow2
Formatting 'ubuntu.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=10737418240 lazy_refcounts=off refcount_bits=
16
```

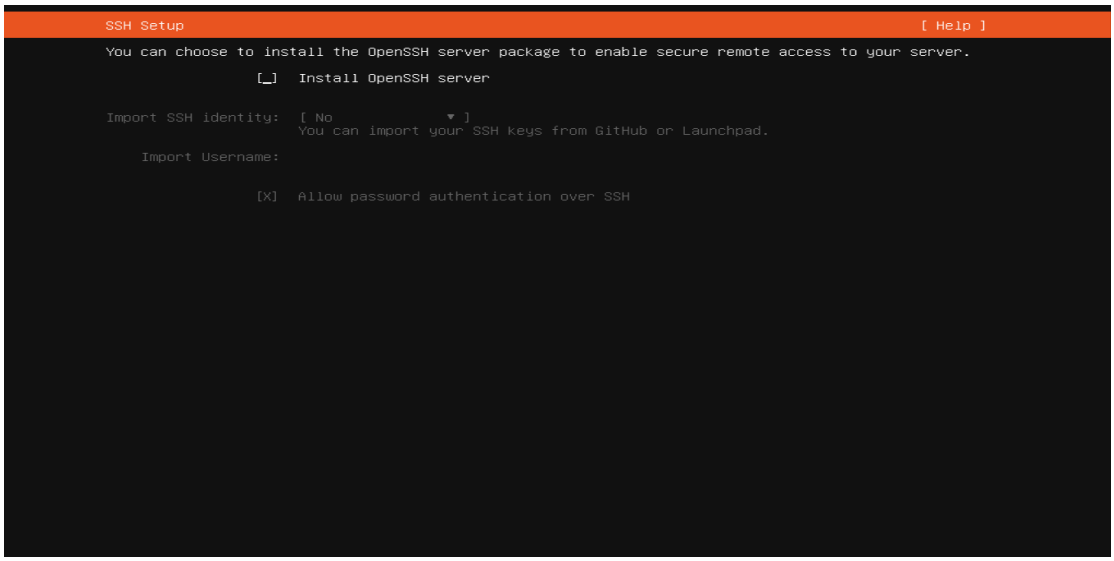
1.4 Once we have our image ready, we can proceed to install the VM, which takes the iso file as a cdrom and the image with created in the previous command as a hard disk.

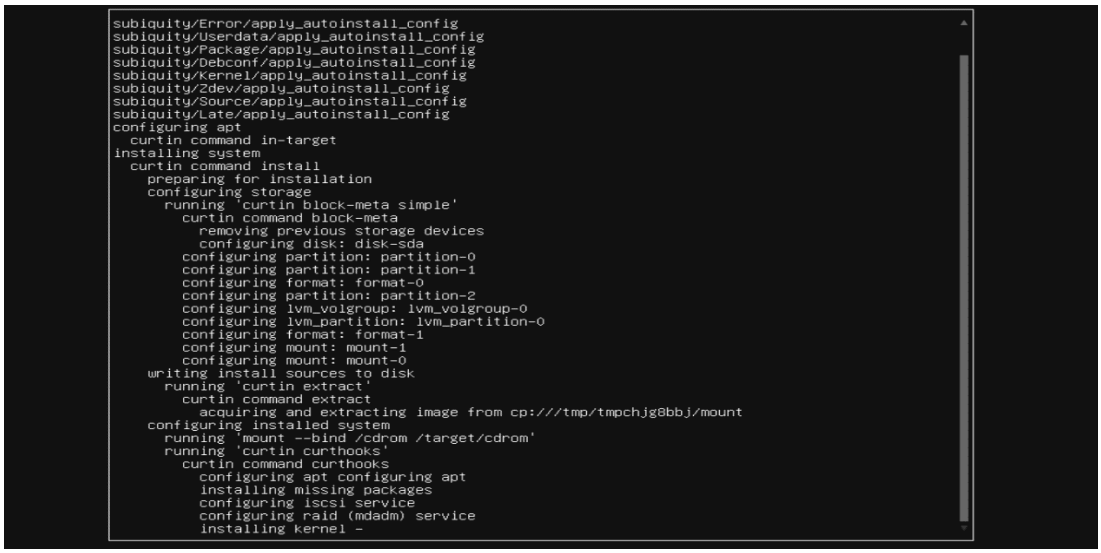
```
juan@juan-IdeaPad-3-15IIL05:~$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ubuntu-20.04.5-live-server-amd64.iso -m 2046 -boot strict
t=on
```

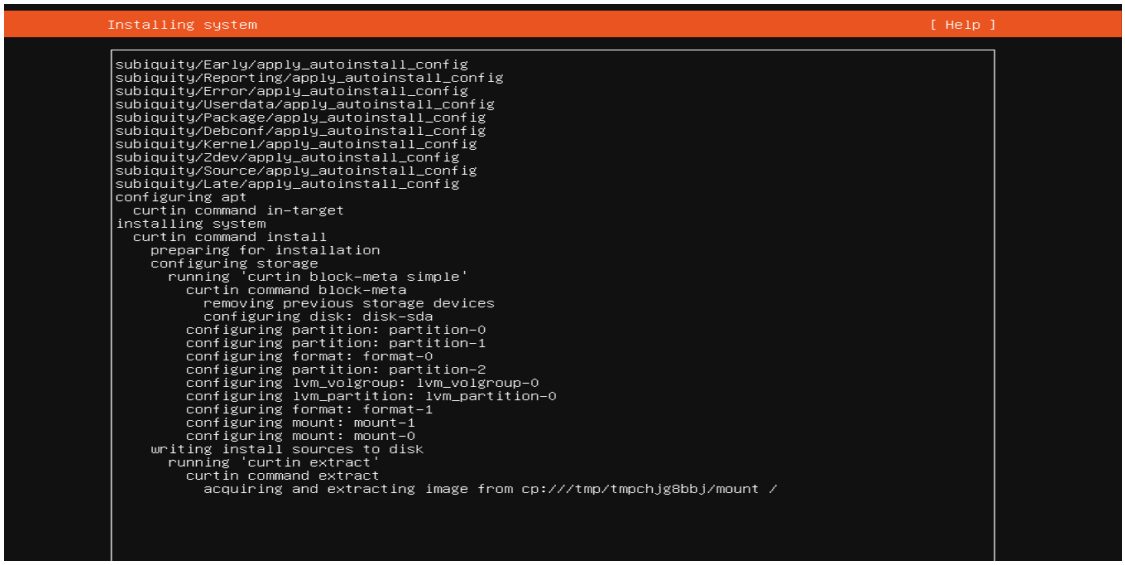
1.5 Then, we launched the virtual machine installation, and we can do different basic configurations to personalize our VM. Such as choosing the language, configuration of packages, timezone, partition configuration and installation of the kernel.

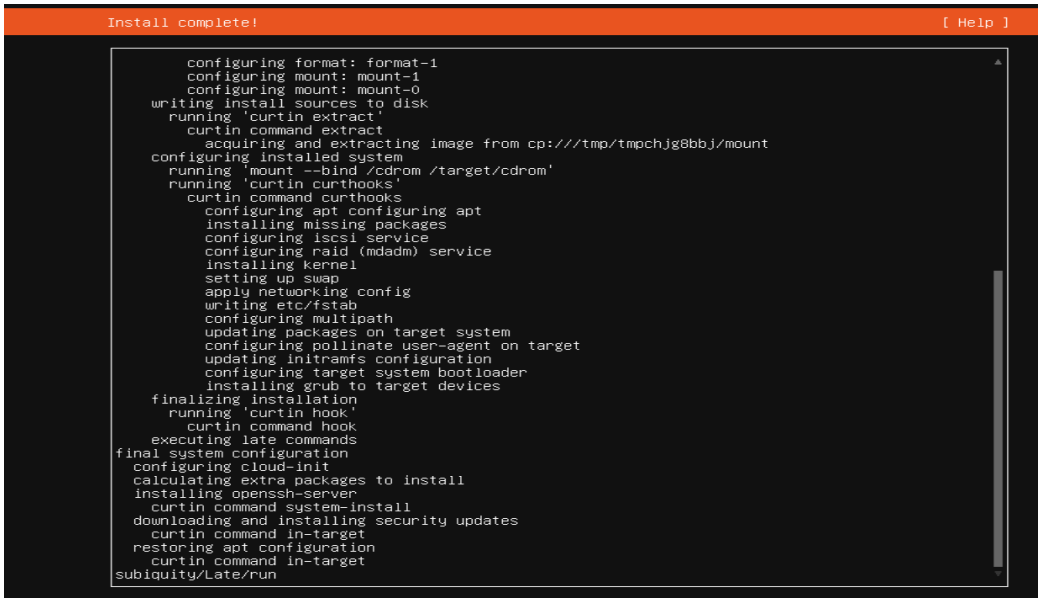



Installing system
[Help]
subiquity/Early/apply_autoinstall_config
subiquity/Reporting/apply_autoinstall_config
subiquity/Error/apply_autoinstall_config
subiquity/Userdata/apply_autoinstall_config
subiquity/Package/apply_autoinstall_config
subiquity/Debconf/apply_autoinstall_config
subiquity/Kernel/apply_autoinstall_config
subiquity/Zdev/apply_autoinstall_config
subiquity/Source/apply_autoinstall_config
subiquity/Late/apply_autoinstall_config
Confirm destructive action
Selecting Continue below will begin the installation process and result in the loss of data on the disks selected to be formatted.
You will not be able to return to this or a previous screen once the installation has started.
Are you sure you want to continue?
[No]
[Continue]


SSH Setup
[Help]
You can choose to install the OpenSSH server package to enable secure remote access to your server.
[_] Install OpenSSH server
Import SSH identity: [No ▾]
You can import your SSH keys from GitHub or Launchpad.
Import Username:
[X] Allow password authentication over SSH


subiquity/Error/apply_autoinstall_config
subiquity/Userdata/apply_autoinstall_config
subiquity/Package/apply_autoinstall_config
subiquity/Debconf/apply_autoinstall_config
subiquity/Kernel/apply_autoinstall_config
subiquity/Zdev/apply_autoinstall_config
subiquity/Source/apply_autoinstall_config
subiquity/Late/apply_autoinstall_config
configuring apt
curtin command in-target
installing system
curtin command install
preparing for installation
configuring storage
running 'curtin block-meta simple'
curtin command block-meta
removing previous storage devices
configuring disk: disk-sda
configuring partition: partition-0
configuring partition: partition-1
configuring format: format-0
configuring partition: partition-2
configuring lvm_volgroup: lvm_volgroup-0
configuring lvm_partition: lvm_partition-0
configuring format: format-1
configuring mount: mount-1
configuring mount: mount-0
writing install sources to disk
running 'curtin extract'
curtin command extract
acquiring and extracting image from cp:///tmp/tmpchjg8bbj/mount
configuring installed system
running 'mount --bind /cdrom /target/cdrom'
running 'curtin curthooks'
curtin command curthooks
configuring apt configuring apt
installing missing packages
configuring iscsi service
configuring raid (mdadm) service
installing kernel -

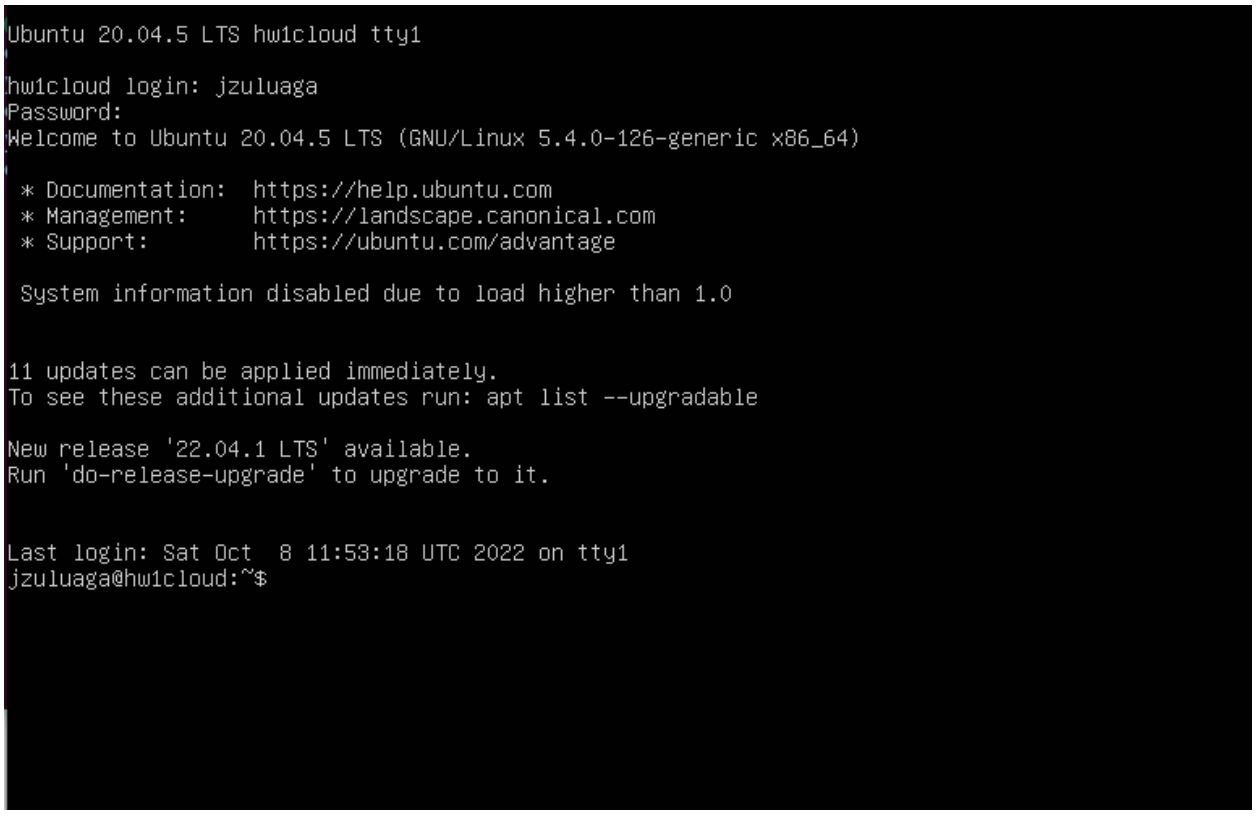

Installing system
[Help]
subiquity/Early/apply_autoinstall_config
subiquity/Reporting/apply_autoinstall_config
subiquity/Error/apply_autoinstall_config
subiquity/Userdata/apply_autoinstall_config
subiquity/Package/apply_autoinstall_config
subiquity/Debconf/apply_autoinstall_config
subiquity/Kernel/apply_autoinstall_config
subiquity/Zdev/apply_autoinstall_config
subiquity/Source/apply_autoinstall_config
subiquity/Late/apply_autoinstall_config
configuring apt
curtin command in-target
installing system
curtin command install
preparing for installation
configuring storage
running 'curtin block-meta simple'
curtin command block-meta
removing previous storage devices
configuring disk: disk-sda
configuring partition: partition-0
configuring partition: partition-1
configuring format: format-0
configuring partition: partition-2
configuring lvm_volgroup: lvm_volgroup-0
configuring lvm_partition: lvm_partition-0
configuring format: format-1
configuring mount: mount-1
configuring mount: mount-0
writing install sources to disk
running 'curtin extract'
curtin command extract
acquiring and extracting image from cp:///tmp/tmpchjg8bbj/mount /



1.6 Once, the vm configuration is over, we can always access it by using the next command:

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo qemu-system-x86_64 -m 1024 -hda ubuntu.img
```

We have to log in with the same credentials we used during the installation process.



We can also test our machine to check that everything was installed correctly.

```
jzuluaga@hw1cloud:~$ ls
jzuluaga@hw1cloud:~$ mkdir hola
jzuluaga@hw1cloud:~$ ls
hola
jzuluaga@hw1cloud:~$ sudo apt-get install python
[sudo] password for jzuluaga:
Sorry, try again.
[sudo] password for jzuluaga:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-is-python2' instead of 'python'
The following additional packages will be installed:
  libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib python2 python2-minimal python2.7 python2.7-minimal
Suggested packages:
  python2-doc python-tk python2.7-doc binutils binfmt-support
The following NEW packages will be installed:
  libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib python-is-python2 python2 python2-minimal python2.7
  python2.7-minimal
0 upgraded, 8 newly installed, 0 to remove and 11 not upgraded.
Need to get 3,815 kB of archives.
After this operation, 16.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-minimal amd64 2.7.18-1~20.04.3 [336 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python2.7-minimal amd64 2.7.18-1~20.04.3 [1,280 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 python2-minimal amd64 2.7.17-2ubuntu4 [27.5 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-stdlib amd64 2.7.18-1~20.04.3 [1,888 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python2.7 amd64 2.7.18-1~20.04.3 [248 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 libpython2-stdlib amd64 2.7.17-2ubuntu4 [7,072 B]
Get:7 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 python2 amd64 2.7.17-2ubuntu4 [26.5 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 python-is-python2 all 2.7.17-4 [2,496 B]
Fetched 3,815 kB in 2s (2,305 kB/s)
```

Finally, we can move onto sysbench installation process, which only needs two steps to be followed:

```
jzuluaga@hw1cloud:~$ sudo apt update
[sudo] password for jzuluaga:
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
0% [Working]
```

```
jzuluaga@hw1cloud:~$ sudo apt install sysbench
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libluajit-5.1-2 libluajit-5.1-common libmysqlclient21 libpq5 mysql-common
The following NEW packages will be installed:
  libluajit-5.1-2 libluajit-5.1-common libmysqlclient21 libpq5 mysql-common sysbench
0 upgraded, 6 newly installed, 0 to remove and 11 not upgraded.
Need to get 1,827 kB of archives.
After this operation, 9,267 kB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

2. Docker configuration and installation.

2.1 Below are presented some of the main steps and commands to be followed in order to install docker.

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
[sudo] password for juan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20211016).
ca-certificates set to manually installed.
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following package was automatically installed and is no longer required:
```

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo mkdir -p /etc/apt/keyrings
juan@juan-IdeaPad-3-15IIL05:~$ curl -fsSL https://download.docker.com/linux/ub
tu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
juan@juan-IdeaPad-3-15IIL05:~$
```

```
juan@juan-IdeaPad-3-15IIL05:~$  echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg
] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /d
ev/null
juan@juan-IdeaPad-3-15IIL05:~$
```

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Hit:2 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [7,0
5 B]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metad
ta [93.2 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 M
tadata [248 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-1
 Metadata [940 B]
Get:11 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-1
 Metadata [12.6 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metad
ta [13.0 kB]
Hit:13 https://ppa.launchpadcontent.net/git-core/ppa/ubuntu jammy InRelease
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 M
tadata [12.4 kB]
Fetched 760 kB in 1s (553 kB/s)
```

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo apt-get install docker-ce docker-ce-cli cont
ainerd.io docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  docker-ce-rootless-extras docker-scan-plugin pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin docker-scan-plugin pigz slirp4netns
0 upgraded, 8 newly installed, 0 to remove and 151 not upgraded.
Need to get 109 MB of archives.
After this operation, 423 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io
amd64 1.6.8-1 [28.1 MB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1
[63.6 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64
```

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo service docker start
juan@juan-IdeaPad-3-15IIL05:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:62af9efd515a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
```

```
juan@juan-IdeaPad-3-15IIL05:~$ docker --version
Docker version 20.10.18, build b40c2f6
juan@juan-IdeaPad-3-15IIL05:~$
```

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo docker run -t -d hello-world
d418afee073a6824eb6d5a713b46a11ef2a4f3c9f44f4258c10e68bdaf40eee3
juan@juan-IdeaPad-3-15IIL05:~$ docker ps -a
Got permission denied while trying to connect to the Docker daemon socket at un
x:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containe
s/json?all=1": dial unix /var/run/docker.sock: connect: permission denied
juan@juan-IdeaPad-3-15IIL05:~$ sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS
     PORTS     NAMES
d418afee073a   hello-world   "/hello"   18 seconds ago   Exited (0) 16 seconds
go             confident_darwin
02fe12f87675   hello-world   "/hello"   5 minutes ago    Exited (0) 5 minutes a
o              interesting_galois
```

Now, we can proceed to pull a base ubuntu image, in order to create our own.

```
juan@juan-IdeaPad-3-15IIL05:~/CloudComputing/HW1/Screenshots$ sudo docker pull u
buntu
[sudo] password for juan:
Using default tag: latest
latest: Pulling from library/ubuntu
cf92e523b49e: Pull complete
Digest: sha256:35fb073f9e56eb84041b0745cb714eff0f7b225ea9e024f703cab56aaa5c7720
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Then, we can get our image running

```
root@5e1c7d1ab803: /
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit cod
es

Run 'docker COMMAND --help' for more information on a command.

To get more help with docker, check out our guides at https://docs.docker.com/go
/guides/

juan@juan-IdeaPad-3-15IIL05:~$ sudo docker run -it ubuntu:20.04
Unable to find image 'ubuntu:20.04' locally
20.04: Pulling from library/ubuntu
fb0b3276a519: Pull complete
Digest: sha256:9c2004872a3a9fcec8cc757ad65c042de1dad4da27de4c70739a6e36402213e3
Status: Downloaded newer image for ubuntu:20.04
root@5e1c7d1ab803:/#
```

Commit the image, since when it’s closed any progress will not be saved. We can call the image image_with_sysbench, since that’s going to be its main purpose.

```
juan@juan-IdeaPad-3-15IIL05:~$ sudo docker commit 5e1c7d1ab803 image_with_sysben
ch
sha256:45524f7d955e0394dd75c6bb4c2fa4ae1cc9f91654c567a571e4c2da140ef111
juan@juan-IdeaPad-3-15IIL05:~$
```

We execute the image, and now we can install basic tools such as nano, bash, sudo and ping.

```
root@5e1c7d1ab803:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [
916 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2183
 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2650
```

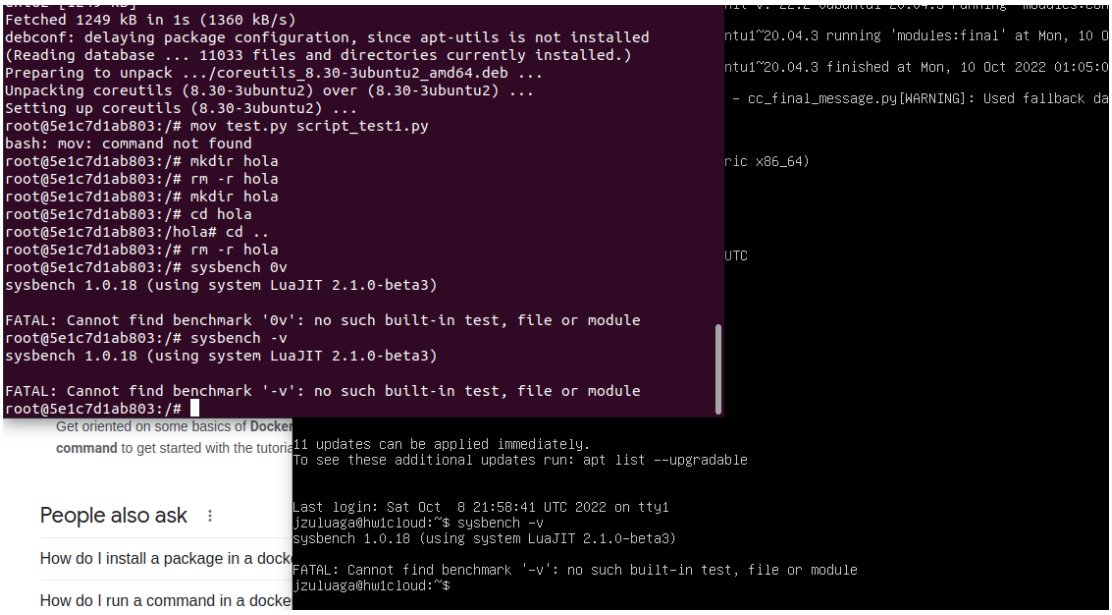
```
root@5e1c7d1ab803:/# apt install iputils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcap2 libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2 libcap2-bin libpam-cap
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 90.5 kB of archives.
After this operation, 333 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 libcap2 amd64 1:2
15.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libcap2-bin amd64
-1 [26.2 kB]
```

```
root@5e1c7d1ab803:/# apt-get -y install sudo
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sudo
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 514 kB of archives.
After this operation, 2257 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 sudo amd64 1.8.
1-1ubuntu1.2 [514 kB]
Fetched 514 kB in 1s (366 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package sudo.
(Reading database ... 4126 files and directories currently installed.)
Preparing to unpack .../sudo_1.8.31-1ubuntu1.2_amd64.deb ...
Unpacking sudo (1.8.31-1ubuntu1.2) ...
Setting up sudo (1.8.31-1ubuntu1.2) ...
root@5e1c7d1ab803:/#
```

Just like we did in the vm, we can repeat the process for the sysbench installation.

```
root@5e1c7d1ab803:/# sudo apt install sysbench
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  krb5-locales libaio1 libasn1-8-heimdal libgssapi-krb5-2 libgssapi3-heimdal
  libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal
  libhx509-5-heimdal libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3
  libkrb5support0 libldap-2.4-2 libldap-common libluajit-5.1-2
  libluajit-5.1-common libmysqlclient21 libpq5 libroken18-heimdal libsasl2-2
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssl1.1 libwind0-heimdal
  mysql-common
Suggested packages:
  krb5-doc krb5-user libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp
  libsasl2-modules-sql
The following NEW packages will be installed:
  krb5-locales libaio1 libasn1-8-heimdal libgssapi-krb5-2 libgssapi3-heimdal
  libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal
  libhx509-5-heimdal libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3
  libkrb5support0 libldap-2.4-2 libldap-common libluajit-5.1-2
  libluajit-5.1-common libmysqlclient21 libpq5 libroken18-heimdal libsasl2-2
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssl1.1 libwind0-heimdal
```

We check the version of sysbench installed in both virtual spaces, in order to see that they coincide.



3. Bash scrips and results gathering.

Sysbench allows to do different performance tests in our machines, it goes from memory tests to data basing test, CPU, and I/O devices. In this case, we will focus on CPU and I/O, and we will see how these two different ways of virtualization differ from each other. Furthermore, to conduct a fair test, we will use the same parameters for each one of the different scenarios (4), we will also do 5 attempts to have a better grasp of what these machines can do.

Below are presented the different commands that will be use in this test:

CPU TEST COMMANDS:

sysbench --test=cpu --cpu-max-prime=1000 run (1)

sysbench --test=cpu --cpu-max-prime=5000 run (2)

sysbench --test=cpu --cpu-max-prime=11000 run (3)

sysbench --test=cpu --cpu-max-prime=16000 run (4)

We will use a bash script with a for loop to automate the process of changing the parameter.

```
#! /bin/bash

for j in {1..5}
do
 for i in {1000..20000..5000}
  do
    printf "Test with  cpu max prime of $i\n" >> output.txt
    sysbench --test=cpu  --cpu-max-prime=$i run >> output.txt
  done
done
```

I/O TEST COMMANDS:

sysbench --test=fileio --file-test-mode=seqwr run (1)

sysbench --test=fileio --file-test-mode=seqwr --file-num=5 --file-block-size=64 run (2)

sysbench --test=fileio --file_test-mode=rndwr --file-num=10 --file-block-size=128 --file-io-mode=async run (3)

sysbench --test=fileio --file-test-mode=seqrewr --file-num=45 --file-block-size=1024 --file-io-mode=async run (4)

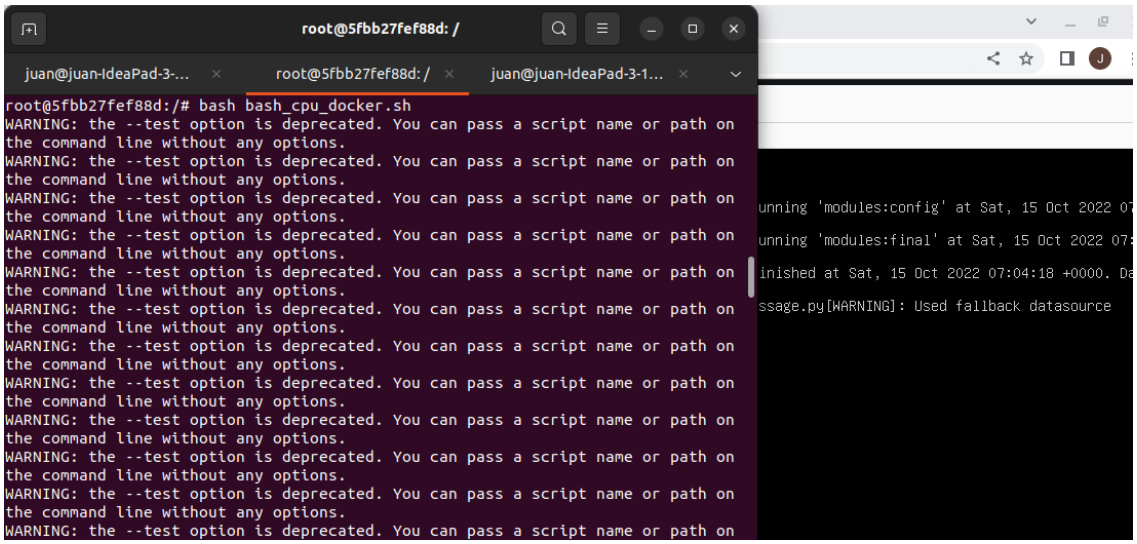
In the case of I/O Test, we can change different parameters, like the way we read/write the information with file-test-mode, the number of files we will be managing and the block size of each one of these files. Lastly, the io-mode allows to decide whether we want other processing to occur while the transmission is still taking place.

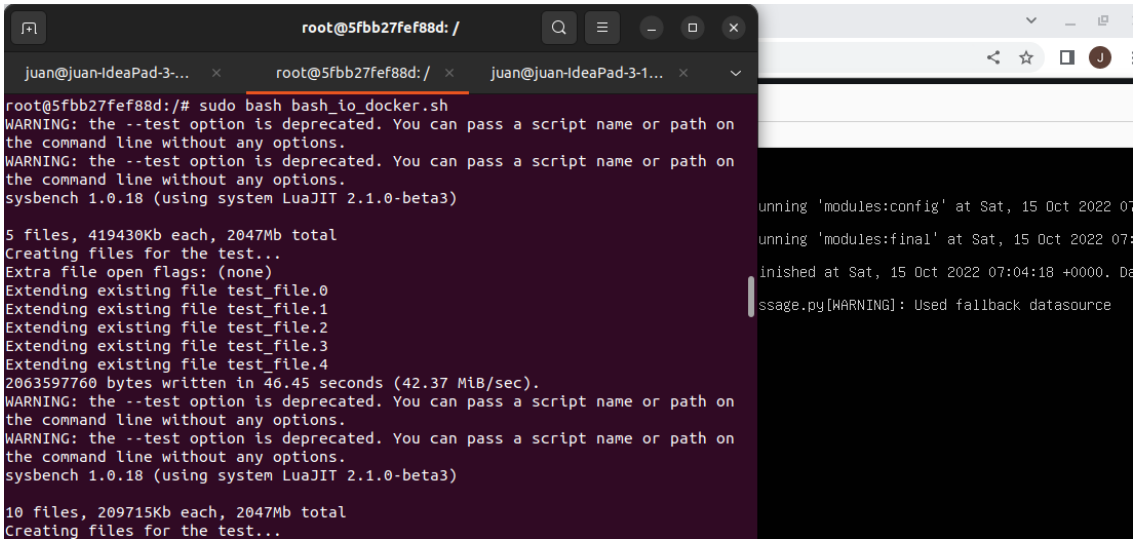
In a similar way to what we did with the CPU, we create a bash script to automate this process, repeating it for 5 attemps.

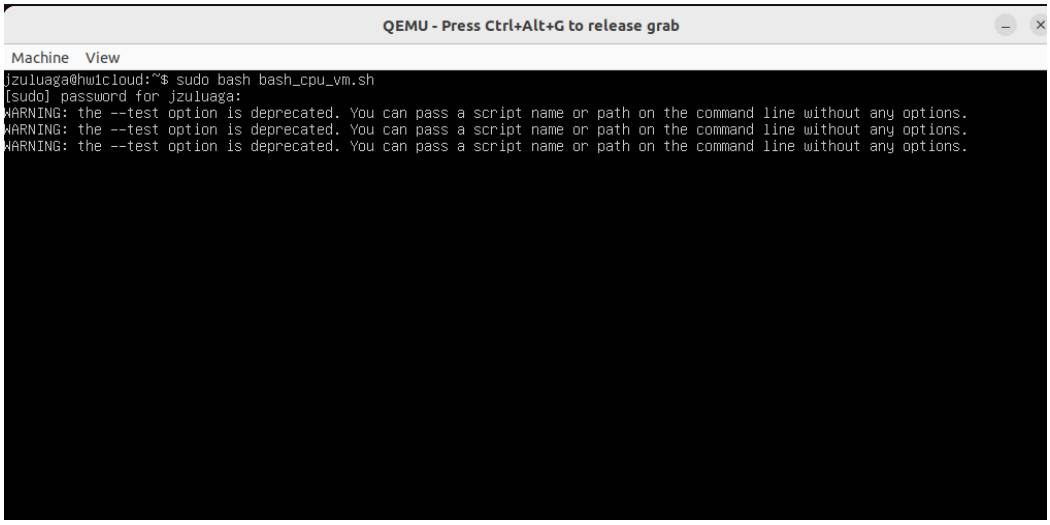
```
#! /bin/bash

for i in {1..5}
do
  sysbench --test=fileio --file-test-mode=seqwr run >> output2.txt
  sysbench --test=fileio --file-test-mode=seqwr --file-num=5 --file-block-size=64 prepare
  sysbench --test=fileio --file-test-mode=seqwr --file-num=5 --file-block-size=64 run >> output2.txt
  sysbench --test=fileio --file_test-mode=rndwr --file-num=10 --file-block-size=128 --file-io-mode=async prepare
  sysbench --test=fileio --file_test-mode=rndwr --file-num=10 --file-block-size=128 --file-io-mode=async run >> output2.txt
  sysbench --test=fileio --file-test-mode=seqrewr --file-num=45 --file-block-size=1024 --file-io-mode=async prepare
  sysbench --test=fileio --file-test-mode=seqrewr --file-num=45 --file-block-size=1024 --file-io-mode=async run >> output2.t
done
```

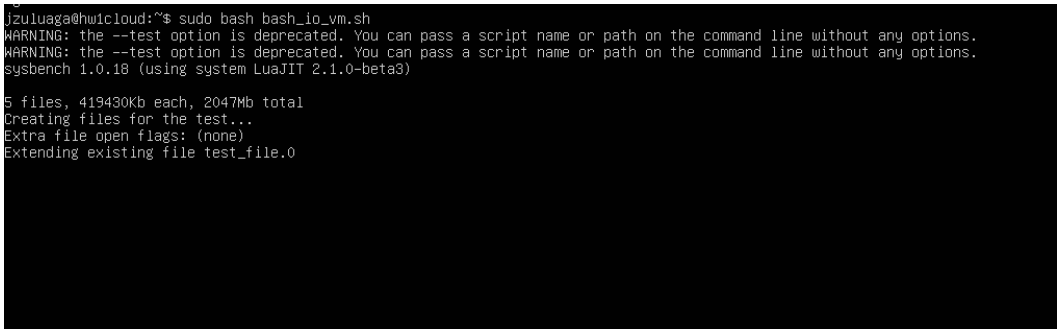
It’s important to note that the final outputs that these commands are obtaining are getting redirected to text files that are going to be also available in the repository.

Now, having both scripts ready to be run, we can do it in both machines to obtains the results that are going to be presented in the next section. We can leave in both cases the scrips running on the background since they are going to be written to a text file.

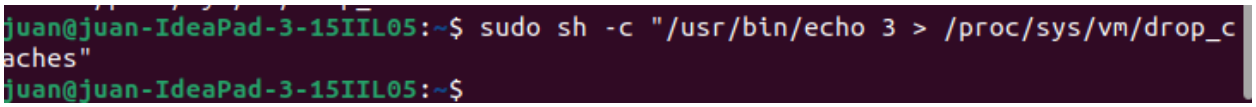








Finally, let's not forget to drop the cache files, so that the tests don't finish before the time expected.

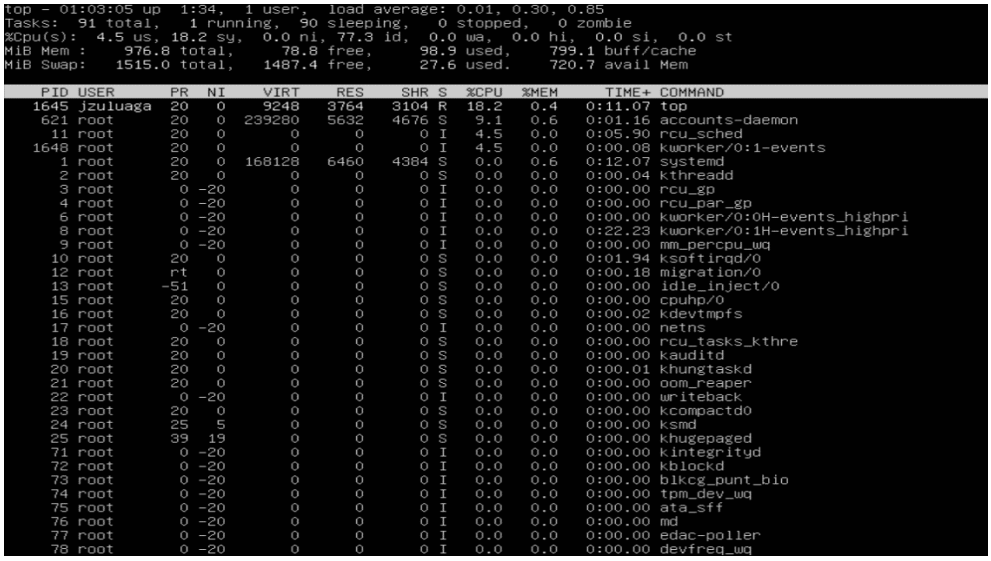


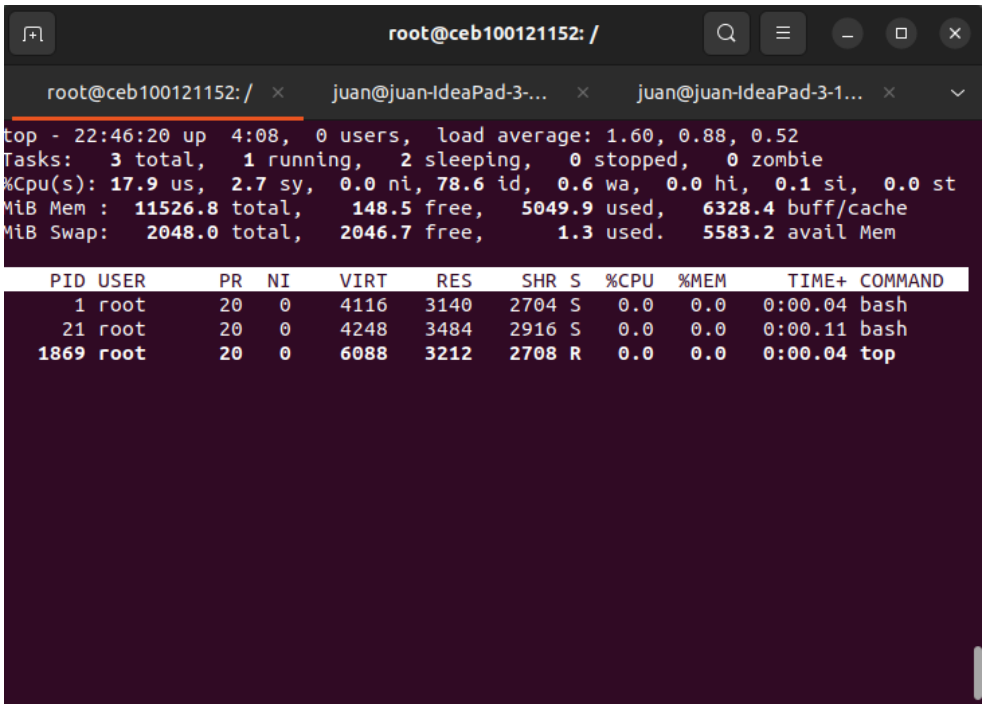
4. Results presentation.

For the results, we will use tables and graphs to have a clear comparison of the performance of both machines in each one of the metrics that sysbench commands provide.

4.1 CPU test performance.

a) User-level and kernel utilization with the command top.





We can clearly see how in the case of the docker container the user space has a larger percentage of the CPU assigned to it. While in the case of the VM, it's completely opposite. This also shows how container are much lighter than virtual machines and make less usage of the resources of the host machine, being that one of the advantages that containers have.

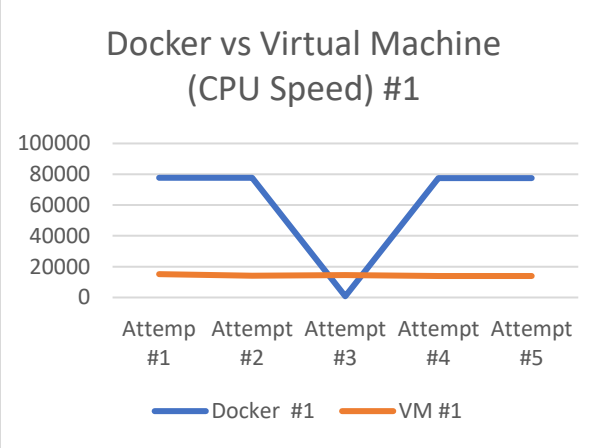
b) CPU Speed.

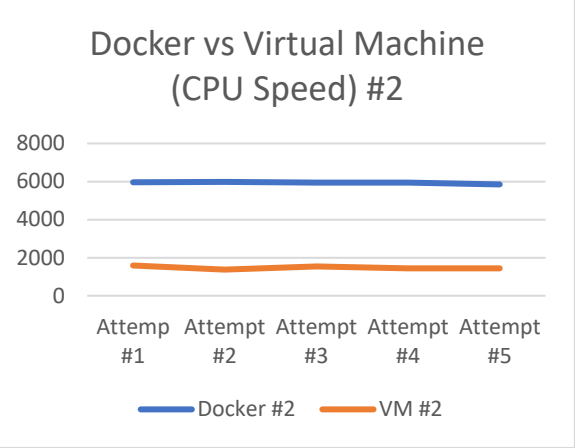
CPU Speed (Events per second)					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	77710.34	77616.2	780.462	77566.58	77472.86
VM #1	15154.05	14047.51	14453.91	13943.91	13943.98
Docker #2	5961.17	5979.9	5947.38	5935.8	5847.38
VM #2	1588	1373.34	1535.81	1436.94	1437.52
Docker #3	2590.96	2568.01	2593.01	2590.96	2582.57
VM #3	702.76	643.21	672.22	651.16	653.59
Docker #4	1547.25	1547	1546.23	1557.92	1537.54
VM #4	397.78	382.25	397.32	392.27	359.42

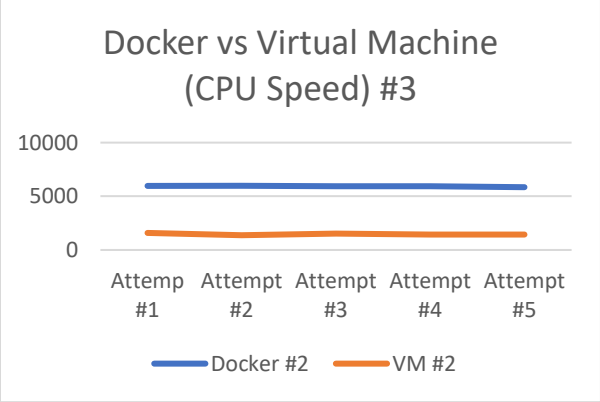
Table I. CPU Speed for each one of the commands

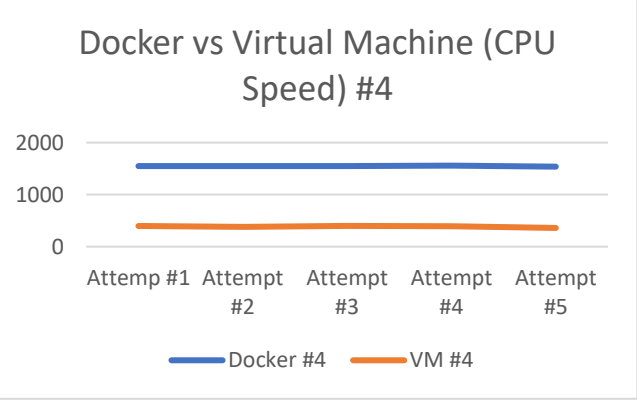
Table I presents the performance of the docker container and virtual machine in each one of the scenarios mentioned in the section 3, the numbers corresponding to the ones assigned in that same section.

We can also look at some graphs to see how both machines do in each one of the # commands and the attemps.





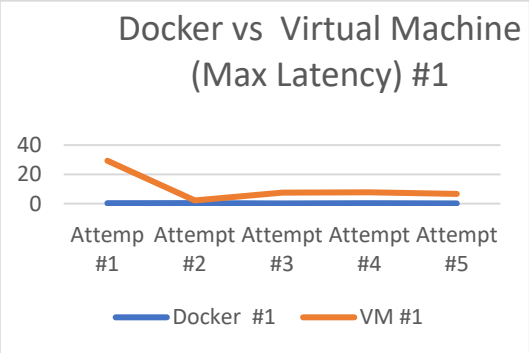


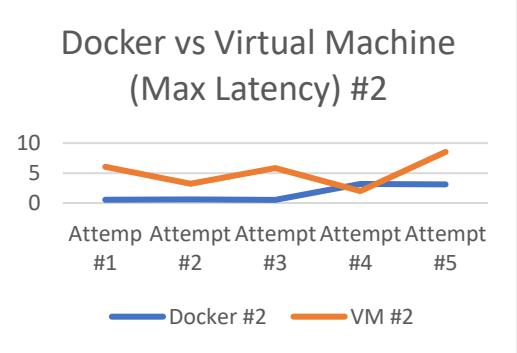


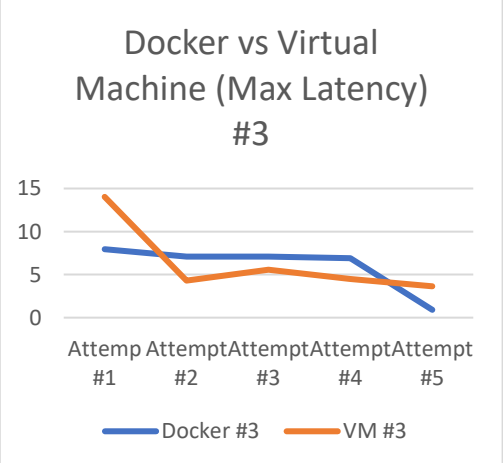
As can be seen from the graphs and the table, the CPU is mostly constant in both cases. However, the VM in general presents a steadier behavior, this is due to the capacities of the VM, since it has actual access to its own virtualized kernel and it's more powerful in general.

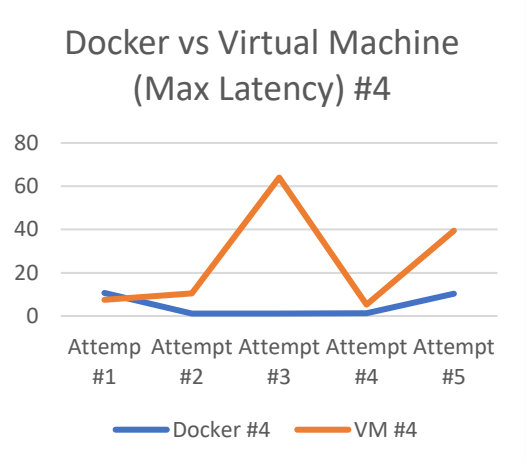
c) CPU Latency

CPU Max. Latency					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	0.26	0.38	0.17	0.31	0.17
VM #1	29.33	2.21	7.58	7.85	6.74
Docker #2	0.54	0.58	0.52	3.18	3.11
VM #2	6.06	3.19	5.81	1.99	8.53
Docker #3	7.95	7.09	7.11	6.92	0.91
VM #3	14.03	4.3	5.57	4.5	3.64
Docker #4	10.68	1.17	1.15	1.25	10.33
VM #4	7.47	10.54	63.99	5.27	39.47







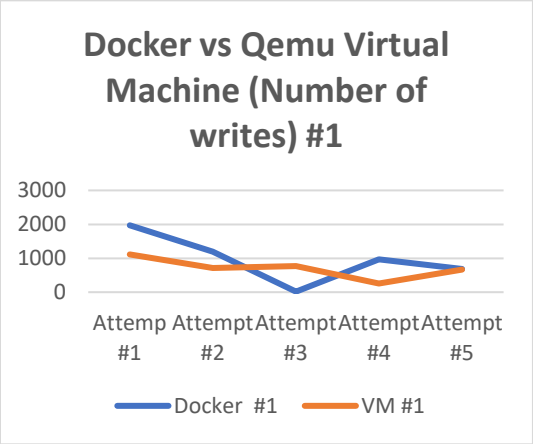


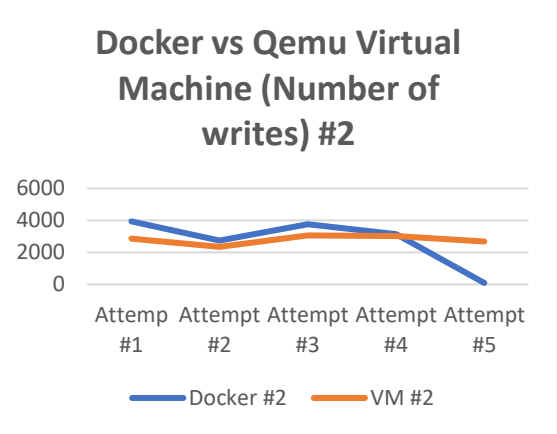
By looking at the graphs we can see how the VM in general has larger peaks of latency, meaning that there are some cases in which there's a lot of delay in the data transfer. This makes a lot of sense, since the VMs don't have a direct connection and interaction with the hardware of the host, making the container a better solution if we are looking for a system with low latency.

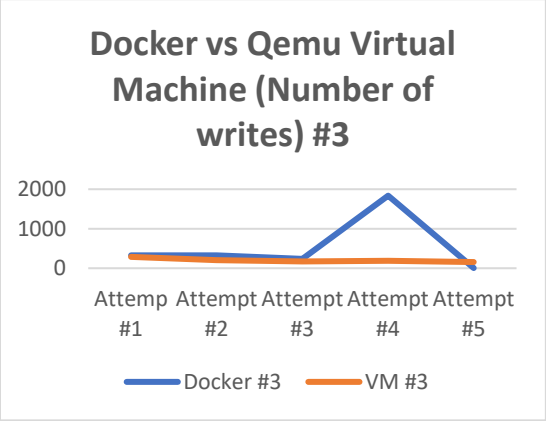
4.2 I/O Test Performance:

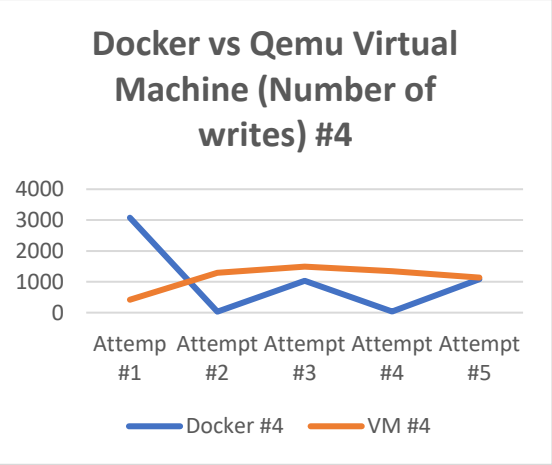
a) I/O Number of writes in Disk.

Disk operations: number of writes					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	1971	1194	16.32	970	683
VM #1	1113	712	767	256	668
Docker #2	3937	2744	3763	3131	85
VM #2	2867	2344	3060	3008	2669
Docker #3	328.73	326	233	1837	4.58
VM #3	287	207	171	188	156
Docker #4	3077	31	1029	39	1088
VM #4	418	1296	1491	1338	1132.02





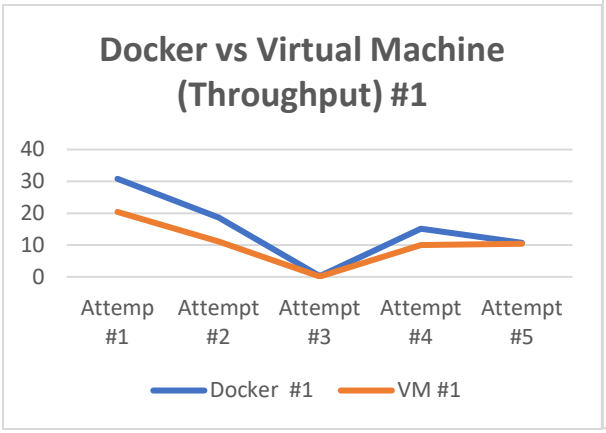


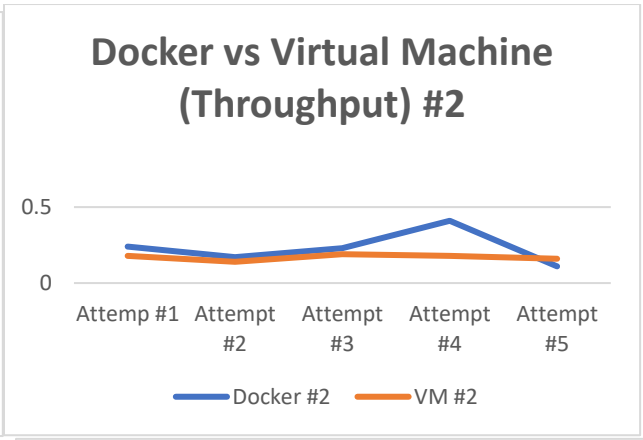


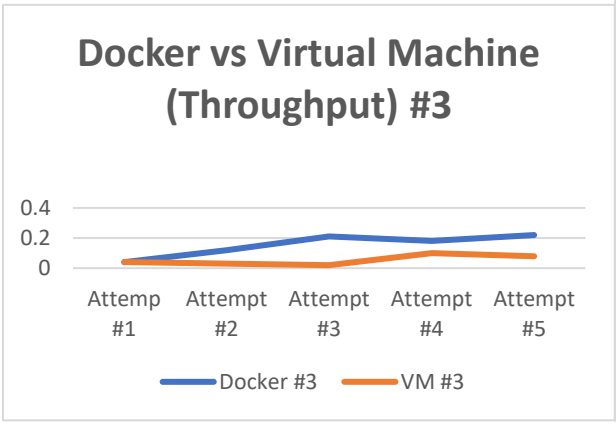
In general, we can see that both machines have a similar behavior in most attempts, but the docker was the one that reached the largest peak in the third command and fourth command. Therefore, one could say in general that docker is slightly better in terms of writing operations in disk.

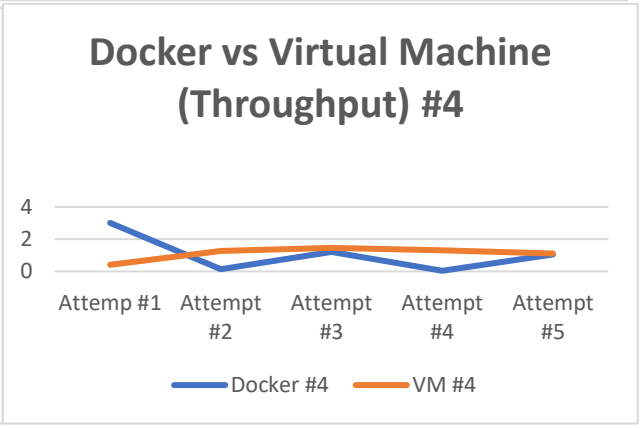
b) Throughput:

Throughput: MiB /s					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	30.8	18.66	0.25	15.17	10.68
VM #1	20.4	11.14	0.12	10.05	10.45
Docker #2	0.24	0.17	0.23	0.41	0.11
VM #2	0.18	0.14	0.19	0.18	0.16
Docker #3	0.04	0.12	0.21	0.18	0.22
VM #3	0.04	0.03	0.02	0.1	0.08
Docker #4	3.01	0.13	1.21	0.04	1.06
VM #4	0.41	1.27	1.46	1.31	1.11





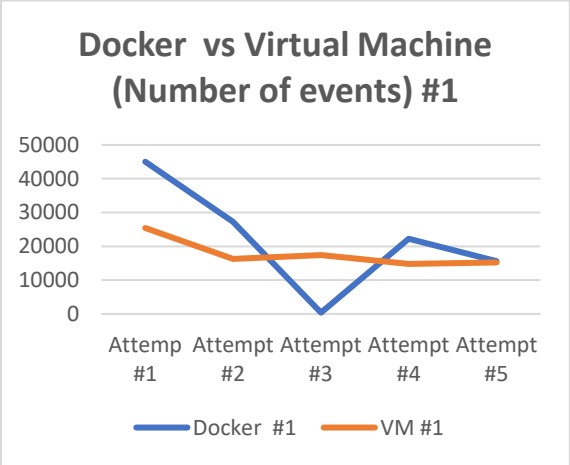


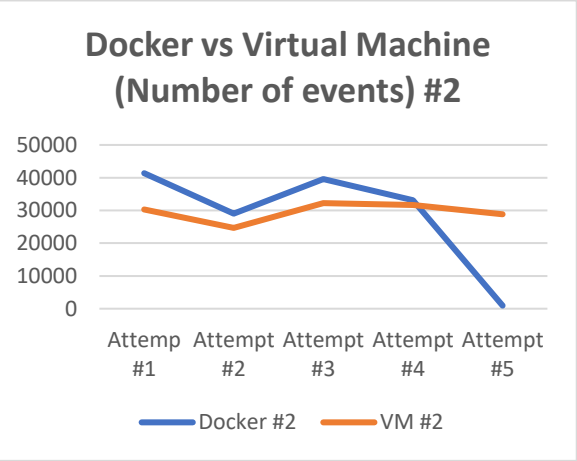


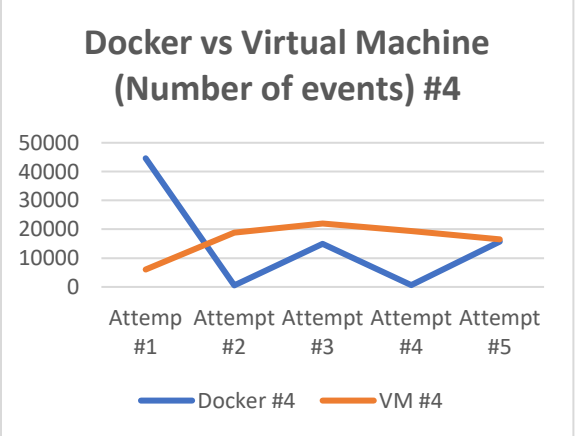
The graphs show that for 3 of the different scenarios proposed containers have the edge over the VM in terms of throughput, which makes sense considering how fast and agile docker container are in comparison to VMs.

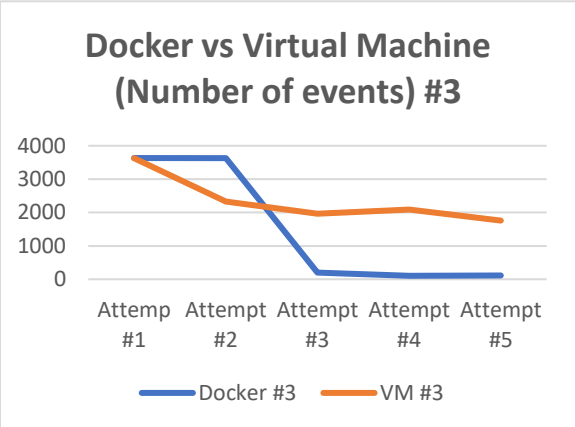
C) Number of events

Number of events					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	45036	27245	369	22226	15611
VM #1	25419	16295	17442	14788	15192
Docker #2	41366	28977	39581	33177	941
VM #2	30341	24671	32231	31706	28871
Docker #3	3628	3629	204	104	110
VM #3	3628	2332	1960	2086	1758
Docker #4	44619	536	14922	681	15761
VM #4	6046	18871	21996	19425	16486





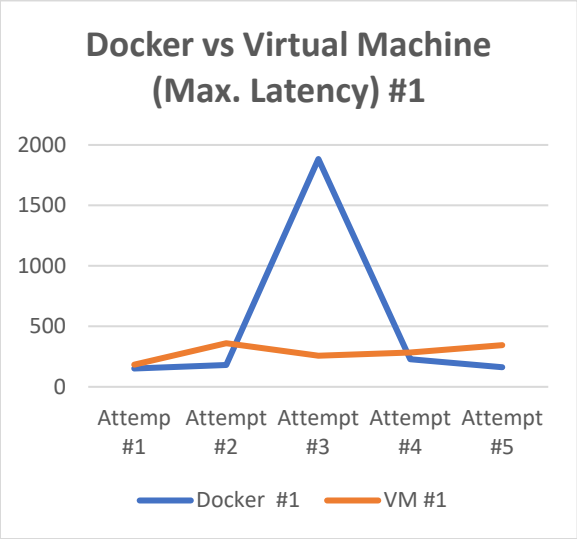


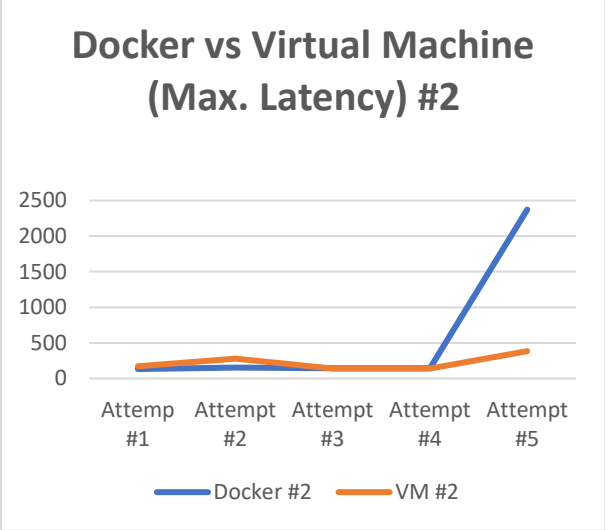


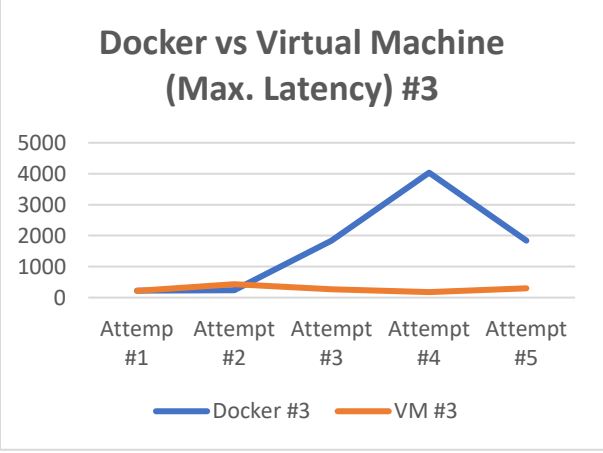
We mentioned for previous tests above that the virtual machine is more powerful than the docker container and in terms of processing should be better. This is an argument that the graphs above support, since we can see that in general it can reach a larger number of events.

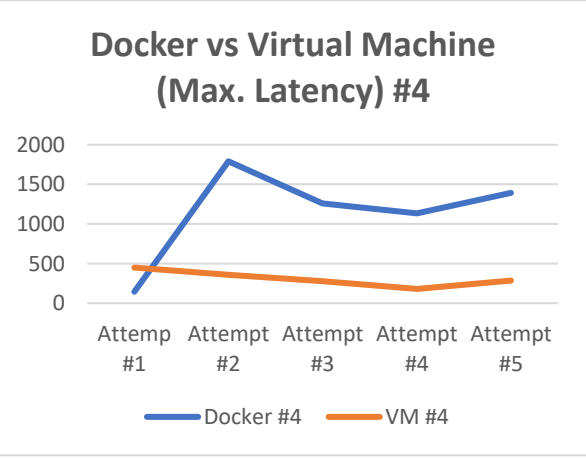
d) Latency max

Latency (Max)					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	151.43	181.87	1883.61	226.94	163
VM #1	183.43	360.6	256.9	284.9	343.59
Docker #2	133.07	154.73	143.62	144.32	2373.42
VM #2	172.64	280.88	139.8	140.64	383.9
Docker #3	223.89	233.89	1837.75	4034.64	1839
VM #3	219.36	432.82	270.59	177.58	298.6
Docker #4	144.64	1790.85	1260.29	1133.67	1393.36
VM #4	449.49	360.55	277.31	180.22	286.6





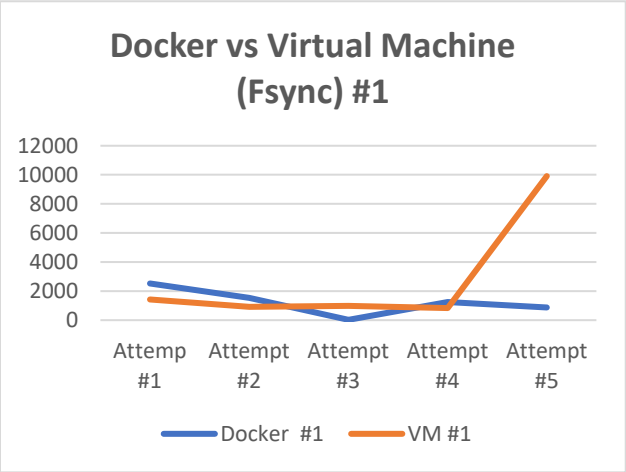


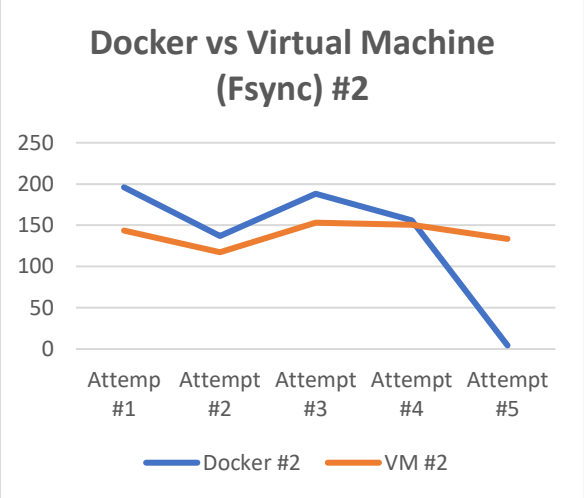


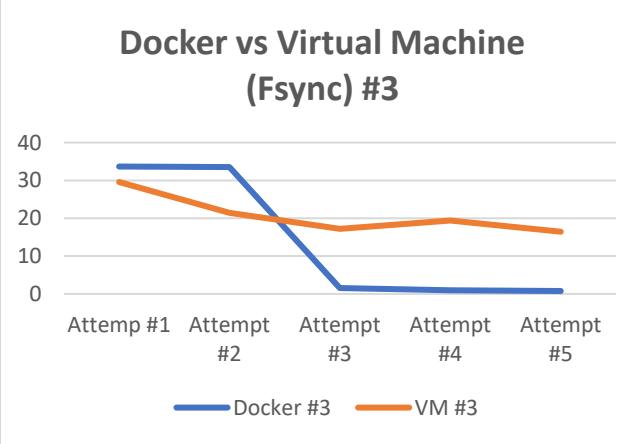
In terms of maximum latency reached for I/O operations, we can see that docker is the clear winner here, since the VM is a machine with its own independent kernel and doesn't need the help of the host device that often.

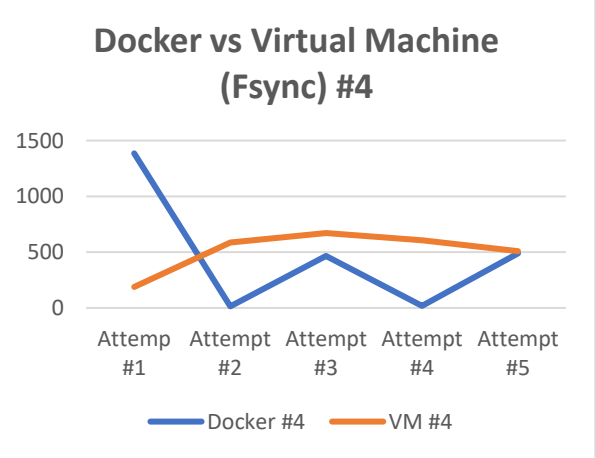
e) Disk operations (fsync)

Disk operations: fsync/s					
Device/# command	Attemp #1	Attempt #2	Attempt #3	Attempt #4	Attempt #5
Docker #1	2524	1529	24.23	1243	875
VM #1	1426	912.93	984.16	832.99	9910.85
Docker #2	196	137	188	156	4.37
VM #2	143.49	117.3	153.12	150.53	133.57
Docker #3	33.67	33.53	1.54	0.95	0.78
VM #3	29.58	21.46	17.23	19.42	16.44
Docker #4	1385.19	14.14	466.4	17.84	489.74
VM #4	188.5	586.87	671.14	606.29	509.51









The VMs in this case clearly has the edge over the containers, meaning that can have more files open at the same time, which makes the VMs a lot more powerful and with a better processing capacity.

5. **General Statistics:**

5.1 CPU test.

a) CPU Speed.

First command:

Max: 77710.34 events per second obtained by the docker container #1 attempt
Min: 780 events per second obtained by the docker container #3 attempt.
Average: 62229.29 for the container and 34351.05 for the virtual machine.
Standard Deviation: 34351 for the container and 14308 for the virtual machine.

Second command:

Max: 5979 events per second obtained by the docker container #2 attempt
Min: 1373 events per second obtained by the virtual machine #2 attempt.
Average: 5934 for the container and 1474 for the virtual machine.
Standard Deviation: 51.30 for the container and 86.09 for the virtual machine.

Third command:

Max: 2593 events per second obtained by the docker container #3 attempt
Min: 643.21 events per second obtained by the virtual machine #2 attempt.
Average: 2585 for the container and 664 for the virtual machine.
Standard Deviation: 10.36 for the container and 23.84 for the virtual machine.

Fourth command:

Max: 1557.92 events per second obtained by the docker container #4 attempt
Min: 359.42 events per second obtained by the virtual machine #3 attempt.
Average: 1547 for the container and 385 for the virtual machine.
Standard Deviation: 7.23 for the container and 16.02 for the virtual machine.

b) CPU Latency.

First command:

Max: 29.23 obtained by the virtual machine #1 attempt
Min: 0.17 obtained by the docker container #3 attempt.
Average: 0.258 for the container and 10.742 for the virtual machine.
Standard Deviation: 0.09094 for the container and 10.63 for the virtual machine.

Second command:

Max: 8.53 obtained by the virtual machine #5 attempt
Min: 0.52 obtained by the docker container #3 attempt.
Average: 1.586 for the container and 5.116 for the virtual machine.
Standard Deviation: 1.423 for the container and 2.57 for the virtual machine.

Third command:

Max: 14.03 obtained by the virtual machine #3 attempt
Min: 4.3 events per second obtained by the docker container #2 attempt.
Average: 5.996 for the container and 6.408 for the virtual machine.
Standard Deviation: 4.316 for the container and 25.6664 for the virtual machine.

Fourth command:

Max: 63.99 obtained by the virtual machine #3 attempt
Min: 1.15 obtained by the docker container #3 attempt.
Average: 4.916 for the container and 25.348 for the virtual machine.
Standard Deviation: 5.103 for the container and 25.66 for the virtual machine.

c) Number of writes in Disk (I/O)

First command:

Max: 1971 obtained by the docker container #1 attempt
Min: 256 obtained by the virtual machine #4 attempt.
Average: 966.864 for the container and 703.2 for the virtual machine.
Standard Deviation: 714.84 for the container and 305.49 for the virtual machine.

Second command:

Max: 3937 obtained by the docker container #1 attempt
Min: 85 obtained by the docker container #5 attempt.
Average: 2732 for the container and 545.86 for the virtual machine.
Standard Deviation: 291.5 for the container and 733.72 for the virtual machine.

Third command:

Max: 1837 obtained by the docker container #4 attempt
Min: 4.58 events obtained by the docker container #5 attempt.
Average: 5.46 for the container and 201 for the virtual machine.
Standard Deviation: 733 for the container and 51.29 for the virtual machine.

Fourth command:

Max: 3077 obtained by the docker container #1 attempt
Min: 31 obtained by the docker container #2 attempt.
Average: 1052.8 for the container and 1135 for the virtual machine.
Standard Deviation: 1242 for the container and 420.7 for the virtual machine.

d) Throughput

First command:

Max: 30.8 obtained by the docker container #1 attempt
Min: 0.12 obtained by the virtual machine #3 attempt.
Average: 15.112 for the container and 10.432 for the virtual machine.
Standard Deviation: 11.17 for the container and 7.182 for the virtual machine.

Second command:

Max: 0.41 obtained by the docker container #4 attempt
Min: 0.14 obtained by the virtual machine #5 attempt.
Average: 0.232 for the container and 0.17 for the virtual machine.
Standard Deviation: 0.1123 for the container and 0.02 for the virtual machine.

Third command:

Max: 0.22 obtained by the docker container #5 attempt
Min: 0.02 obtained by the virtual machine #3 attempt.
Average: 0.154 for the container and 0.054 for the virtual machine.
Standard Deviation: 0.07469 for the container and 0.034351 for the virtual machine.

Fourth command:

Max: 3.01 obtained by the docker container #1 attempt
Min: 0.04 obtained by the docker container #4 attempt.
Average: 1.09 for the container and 1.112 for the virtual machine.
Standard Deviation: 1.1964 for the container and 0.411728 for the virtual machine.

e) Latency (I/O)

First command:

Max: 1883 obtained by the docker container #3 attempt
Min: 163 obtained by the docker container #1 attempt.
Average: 521 for the container and 285 for the virtual machine.
Standard Deviation: 762 for the container and 71 for the virtual machine.

Second command:

Max: 2373.42 obtained by the docker container #5 attempt
Min: 133.07 obtained by the docker container #1 attempt.
Average: 589 for the container and 223 for the virtual machine.
Standard Deviation: 997 for the container and 106 for the virtual machine.

Third command:

Max: 4034 obtained by the docker container #4 attempt
Min: 177.58 obtained by the virtual machine #4 attempt.
Average: 1633 for the container and 279 for the virtual machine.
Standard Deviation: 1564 for the container and 97 for the virtual machine.

Fourth command:

Max: 1790 obtained by the docker container #2 attempt
Min: 144.64 obtained by the docker container #1 attempt.
Average: 1144 for the container and 310.83 for the virtual machine.
Standard Deviation: 610 for the container and 100 for the virtual machine.

6. CONCLUSIONS:

I personally think that we can summarize the results obtained in the different experiments by saying that in general docker had a better CPU performance and virtual machines a better I/O performance. This might be due to the differences between those two in terms of architecture, being virtual machines more complete devices by themselves but at the same time having a lot of overhead. Docker container in general are the opposite case.