COEN 241: HW 1
System vs OS Virtualization

Name : Surya Kiran U
ID : W1610385

Table of Contents:

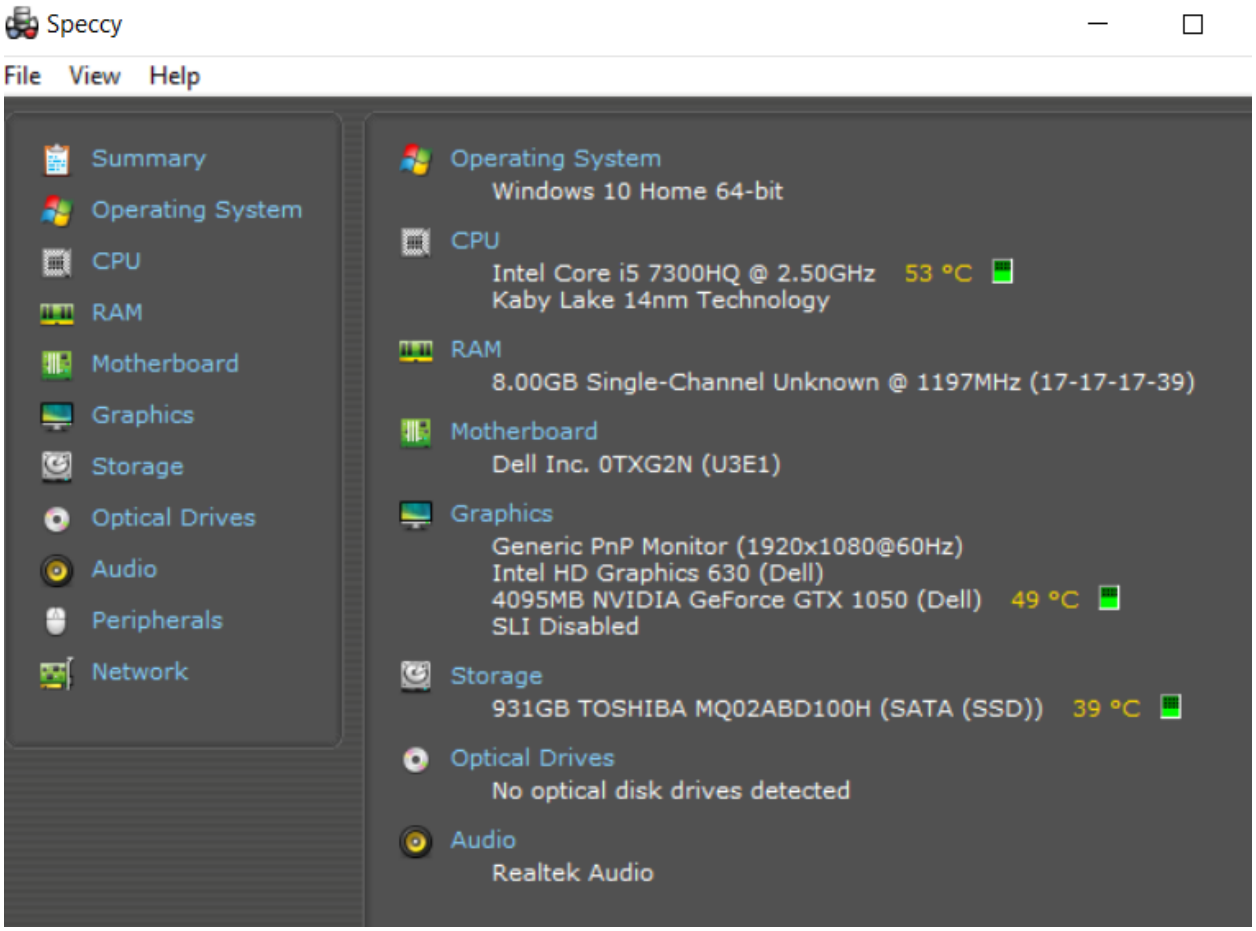
1. System Configurations……....2
   1.1) Native system config…….....2
   1.2) Qemo-Ubuntu config…….....3
2. Steps to enable a QEMU VM…….3
3. Enabling Docker Container..…….5
4. Proof of Experiments……..……9
   4.1) CPU Test 1…….....11
   4.2) CPU Test 2…….....11
   4.3) CPU Test 3…….....11
   4.4) FILE IO Test 1…….....11
   4.5) FILE IO Test 2……..11
   4.6) FILE IO Test 3……..11
   4.7) Shell Scripts for the experiments…….....11
   4.8) Analysis of the experiments..…….....11
5. GIT repository information……..12
6. Writing Vagrant File with results……...13

1. System Configurations:

1.1) Personal Computer(Base) Setup:

1. **OS:** Windows 10 Home 64-bit
2. **CPU:** Intel Core i5 7300HQ @ 2.50GHz(Kaby Lake 14nm Technology)
3. **RAM:** 8.00GB@ 1197MHz
4. **MOTHERBOARD**: Dell Inc. 0TXG2N (U3E1)
5. **STORAGE:** 931GB TOSHIBA MQ02ABD100H (SATA (SSD))
6. **CORES:** 4
7. **THREADS:** 4
8. **VIRTUALIZATION:** Supported

We can either check the system specs by executing “dxdiag.exe” from windows run or execute “systeminfo” in CMD prompt. Also, to get extensive information, the speccy tool can be used as shown below.



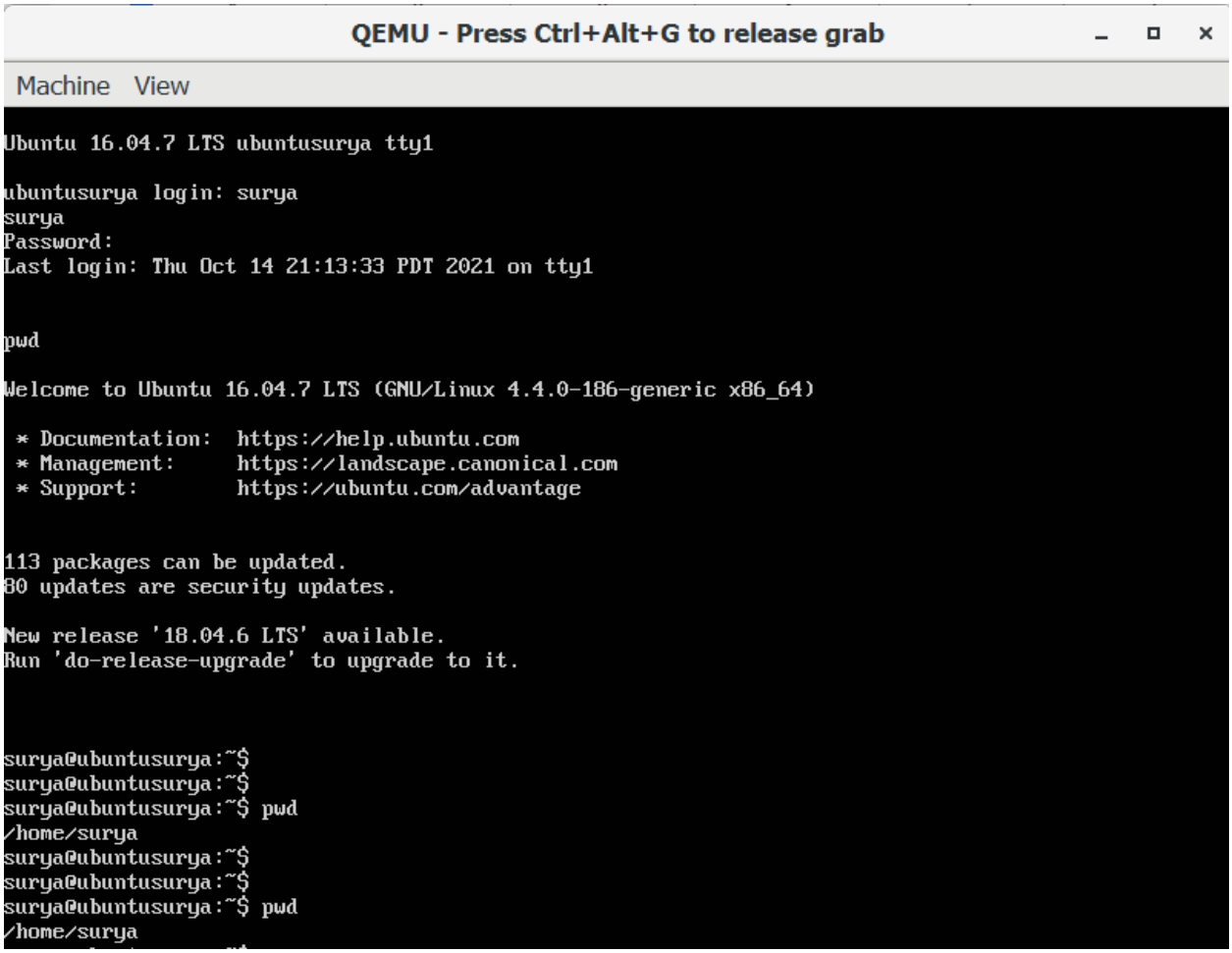
1.2) QEMU: UBUNTU Virtual OS:

1. **OS:** Ubuntu 16.04 xenial
2. **KERNEL:** x86_64 Linux 4.4.0-186-generic
3. **Shell:** bash 4.3.48
4. **CPU:** QEMU Virtual CPU version 2.5+ @ 2.496GHZ
5. **RAM:** 72MiB/1998MiB

2) Steps to enable a QEMU VM:

1. First, we will **download Ubuntu 16.04 server**(AMD64 version) and need to **install qemu binaries for windows** system from their official webpage.
2. Now, place the installed qemu binaries path in the environment variables.
3. Create a qemu image for the Ubuntu guest virtual machine. It can be done by executing the following in Powershell, navigate to the virtual machine iso folder and execute: "**qemu-img.exe create ubuntu.img 10G -f qcow2**".
4. Once image is created, execute the installation command:
   "qemu-system-x86_64.exe -L "C:\Program Files\qemu" -hda .\ubuntu.img -boot d -cdrom .\ubuntu-16.04.7-server-amd64.iso -m 2046 -boot strict=on"

5. After installation is complete, you can now boot from the image created, execute **“qemu-system-x86_64.exe -hda .\ubuntu.img -boot d -m 2046 -boot strict=on”**.
6. You will be able to see ubuntu booted through qemu as shown below:



QEMU-Ubuntu VM Configurations:

1. Host : ubuntusurya
2. Kernel: 4.4.0-186-generic x86_64(64 bit) Consol :tty 1
3. Qemu-harddisk: 10.7GB
4. Cache: 512KB
5. Distro: Ubuntu 16.04 xenial

We can get the system information either by $screenfetch or $inxi -F.

```
surya@ubuntusurya:~$ inxi -F
System:    Host: ubuntusurya Kernel: 4.4.0-186-generic x86_64 (64 bit) Console: tty 1
           Distro: Ubuntu 16.04 xenial
Machine:   System: QEMU product: Standard PC (i440FX + PIIX 1996) v: pc-i440fx-5.2
           Mobo: N/A model: N/A
           Bios: Sea v: rel-1.14.0-0-g155821a1990b-prebuilt.qemu.org date: 04/01/2014
CPU:       Single core QEMU Virtual version 2.5+ (-UP-) cache: 512 KB speed: 2491 MHz (max)
Graphics:  Card: Device 1234:1111
           Display Server: N/A driver: N/A tty size: 100x37 Advanced Data: N/A out of X
Network:   Card: Intel 82540EM Gigabit Ethernet Controller driver: e1000
           IF: ens3 state: up speed: 1000 Mbps duplex: full mac: 52:54:00:12:34:56
Drives:    HDD Total Size: 10.7GB (26.0% used) ID-1: /dev/sda model: QEMU_HARDDISK size: 10.7GB
Partition: ID-1: / size: 8.1G used: 1.7G (22%) fs: ext4 dev: /dev/dm-0
           ID-2: /boot size: 720M used: 59M (9%) fs: ext2 dev: /dev/sda1
           ID-3: swap-1 size: 1.03GB used: 0.00GB (0%) fs: swap dev: /dev/dm-1
RAID:      No RAID devices: /proc/mdstat, md_mod kernel module present
Sensors:   None detected - is lm-sensors installed and configured?
Info:      Processes: 102 Uptime: 26 min Memory: 84.7/1998.0MB Init: systemd runlevel: 5
           Client: Shell (bash) inxi: 2.2.35
surya@ubuntusurya:~$ _
```

3) Enabling Docker Container.

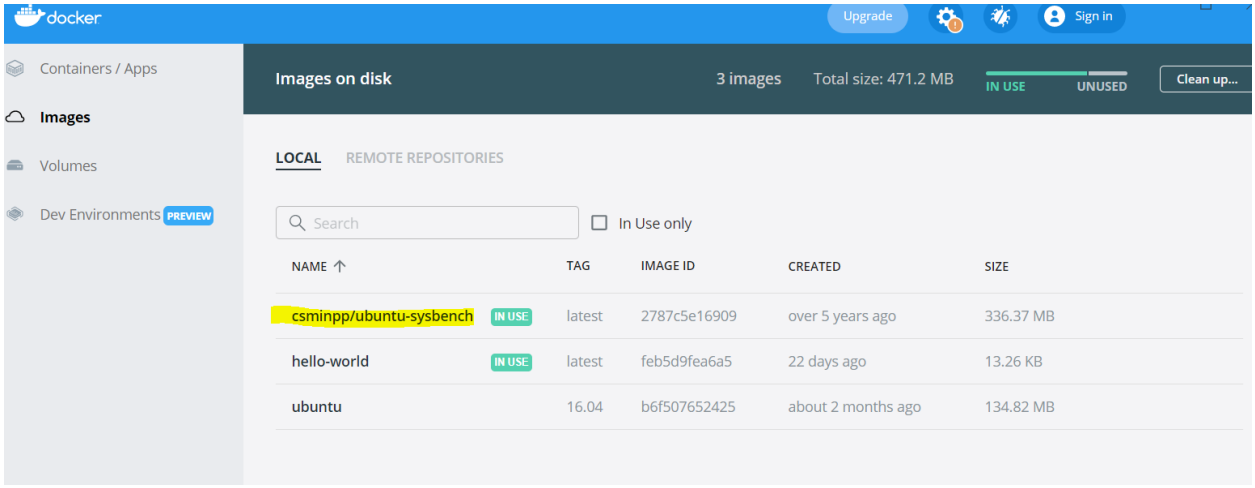
1. We will be downloading and installing Docker Desktop (for Windows).
2. Once installation is complete, open the CMD prompt and check the docker version to verify the installation as shown below: “docker --version”.

```
C:\Users\usury>docker --version
Docker version 20.10.8, build 3967b7d
```

3. After Docker desktop installation is successful, we will install a image **csminpp/ubuntu-sysbench**, which has testbench preinstalled.
4. Execute the command **“docker pull csminpp/ubuntu-sysbench”** in cmd.

```
C:\Users\usury>docker pull csminpp/ubuntu-sysbench
Using default tag: latest
latest: Pulling from csminpp/ubuntu-sysbench
Image docker.io/csminpp/ubuntu-sysbench:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image
for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
d89e1bee20d9: Already exists
9e0bc8a71bde: Already exists
27aa681c95e5: Already exists
a3ed95caeb02: Already exists
55734f896640: Already exists
Digest: sha256:90fd06985472eec3aa99b665618c23f074deb326fcc87a5fb59d2be1f9d97435
Status: Image is up to date for csminpp/ubuntu-sysbench:latest
docker.io/csminpp/ubuntu-sysbench:latest
```

5. Verify the image installation in docker desktop:
   Shown as highlighted below:



   Or in CMD: **docker images**

```
C:\Users\usury>docker images
REPOSITORY                TAG       IMAGE ID       CREATED        SIZE
hello-world               latest    feb5d9fea6a5   3 weeks ago    13.3kB
ubuntu                    16.04     b6f507652425   6 weeks ago    135MB
csminpp/ubuntu-sysbench   latest    2787c5e16909   5 years ago    336MB
```

6. Check the currently running processes:(Empty as we arent running anything).

```
C:\Users\usury>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

7. Check the dangling docker images:**docker volume ls -f dangling=true**

```
C:\Users\usury>docker volume ls -f dangling=true
DRIVER    VOLUME NAME
```

8. Examine the docker containers:**docker container ls -a**

```
C:\Users\usury>docker container ls -a
CONTAINER ID   IMAGE                          COMMAND       CREATED      STATUS                          PORTS     NAMES
a55fe867d3df   hello-world                    "/hello"      3 days ago   Exited (0) 3 days ago                     romantic_blackwell
c763ce787526   csminpp/ubuntu-sysbench        "/bin/bash"   3 days ago   Exited (255) 50 minutes ago               pedantic_hodgkin
48ea6efdf7cc   csminpp/ubuntu-sysbench:latest "/bin/bash"   3 days ago   Exited (255) 50 minutes ago               Sysbench
c3b07efe484b   hello-world                    "/hello"      4 days ago   Exited (0) 4 days ago                     wizardly_jones
ca0bfcc452b7   csminpp/ubuntu-sysbench:latest "/bin/bash"   4 days ago   Exited (255) 50 minutes ago               surya
```

9. Inspecting a particular container configuration:
   docker container inspect surya

```
C:\Users\usury>docker container ls -a
CONTAINER ID   IMAGE                          COMMAND       CREATED      STATUS                      PORTS     NAMES
a55fe867d3df   hello-world                    "/hello"      3 days ago   Exited (0) 3 days ago                 romantic_blackwell
c763ce787526   csminpp/ubuntu-sysbench        "/bin/bash"   3 days ago   Exited (255) 50 minutes ago           pedantic_hodgkin
48ea6efdf7cc   csminpp/ubuntu-sysbench:latest "/bin/bash"   3 days ago   Exited (255) 50 minutes ago           Sysbench
c3b07efe484b   hello-world                    "/hello"      4 days ago   Exited (0) 4 days ago                 wizardly_jones
ca0bfcc452b7   csminpp/ubuntu-sysbench:latest "/bin/bash"   4 days ago   Exited (255) 50 minutes ago           surya

C:\Users\usury>docker container inspect surya
[
    {
        "Id": "ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712",
        "Created": "2021-10-11T18:55:02.1832302Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
            "Status": "exited",
            "Running": false,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 0,
            "ExitCode": 255,
            "Error": "",
            "StartedAt": "2021-10-11T18:55:05.4297131Z",
            "FinishedAt": "2021-10-15T19:18:39.0574771Z"
        },
        "Image": "sha256:2787c5e169096c2187f94a4ab58071d0876de08722c32dbd0d9c6e9f400a01f6",
        "ResolvConfPath": "/var/lib/docker/containers/ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712/hostname",
        "HostsPath": "/var/lib/docker/containers/ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712/hosts",
        "LogPath": "/var/lib/docker/containers/ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712/ca0bfcc452b767b1aac45e1352f59ebcd9b49bf35bf9b5042887c71c1ef5e712-json.log",
        "Name": "/surya",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "",
        "ExecIDs": null,
```

10. Finally, we will run the docker image in an interactive mode:**docker container run -it csminpp/ubuntu-sysbench**

```
C:\Users\usury>docker run -it csminpp/ubuntu-sysbench
root@1042ef9f1fde:/#
```

Since our image is running: **docker ps** will show the running container.

```
C:\Users\usury>docker ps
CONTAINER ID   IMAGE                    COMMAND       CREATED         STATUS         PORTS     NAMES
1042ef9f1fde   csminpp/ubuntu-sysbench  "/bin/bash"   2 minutes ago   Up 2 minutes             pedantic_sutherland
```

docker container run -it "csminpp/ubuntu-sysbench" /bin/bash

```
C:\Users\usury>docker ps
CONTAINER ID   IMAGE                    COMMAND       CREATED          STATUS          PORTS     NAMES
199f03aacda9   csminpp/ubuntu-sysbench  "/bin/bash"   13 seconds ago   Up 11 seconds             affectionate_allen

C:\Users\usury>docker container run -it "csminpp/ubuntu-sysbench" /bin/bash
root@444b136c36b0:/#
```

11. Now, we can stop the running container by executing: **docker stop container_name** (The interactive run will be exited).

```
C:\Users\usury>docker ps
CONTAINER ID   IMAGE                    COMMAND       CREATED         STATUS         PORTS     NAMES
1042ef9f1fde   csminpp/ubuntu-sysbench  "/bin/bash"   6 minutes ago   Up 6 minutes             pedantic_sutherland

C:\Users\usury>docker stop pedantic_sutherland
pedantic_sutherland

C:\Users\usury>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

4) Proof of Experiments:

1. Testing different arguments of qemu: Booting with -smp(specifying number of cores) and -m(amount of memory):**qemu-system-x86_64.exe -L "C:/Program Files/qemu" -smp 3 -hda .\ubuntu.img -boot d -m 2046 -boot strict=on**

```
PS C:\Users\usury\Downloads> qemu-system-x86_64.exe -L "C:/Program Files/qemu" -smp 3 -hda .\ubuntu.img -boot d -m 2046
-boot strict=on
WARNING: Image format was not specified for '.\ubuntu.img' and probing guessed raw.
        Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.

        Specify the 'raw' format explicitly to remove the restrictions.
```

We can see the number of cores in CPU is 3 as highlighted below:

```
surya@ubuntusurya:~$ screenfetch
                           ./+o+-       surya@ubuntusurya
                   yyyyy- -yyyyyy+      OS: Ubuntu 16.04 xenial
                ://+//////-yyyyyyo      Kernel: x86_64 Linux 4.4.0-186-generic
            .++ .:/++++++/-.+sss/`      Uptime: 1m
          .:++o:  /++++++++/:--:/-      Packages: 504
         o:+o+:++.`..```.-/oo+++++/     Shell: bash 4.3.48
        .:+o:+o/.          `+sssoo+/    CPU: 3x QEMU Virtual CPU version 2.5+ @ 2.496GHz
  .++/+:+oo+o:`             /sssooo.    RAM: 69MiB / 1997MiB
 /+++//+:`oo+o               /::--:.
 \+/+o+++`o++o                ++////.
  .++.o+++oo+:`              /dddhhh.
       .+.o+oo:.             `oddhhhh+
        \+.++o+o``-````.:ohdhhhhh+
         `:o+++ `ohhhhhhhhyo++os:
           .o:`.syhhhhhhh/.oo++o`
               /osyyyyyyyo++ooo+++/
                   ````  +oo+++o\:
                          `oo++.
surya@ubuntusurya:~$
```
```

The number of **cores helped in improving the startup time** of the virtual machine as shown below:**systemd-analyze(with 3 cores).**

```
surya@ubuntusurya:~$ systemd-analyze
Startup finished in 16.360s (kernel) + 26.086s (userspace) = 42.446s
```

The number of cores helped in improving the startup of the virtual machine as shown below: **systemd-analyze(with 1 cores).**

```
surya@ubuntusurya:~$ systemd-analyze
Startup finished in 17.983s (kernel) + 1min 6.072s (userspace) = 1min 24.055s
```

2. Sysbench version in Docker container:**sysbench --version**

```
C:\Users\usury>docker container run -it "csminpp/ubuntu-sysbench" /bin/bash
root@444b136c36b0:/# sysbench --version
sysbench 0.4.12
root@444b136c36b0:/#
```

3. Sysbench version in Ubuntu VM:

```
surya@ubuntusurya:~$ sysbench --version
sysbench 0.4.12
surya@ubuntusurya:~$
```

4.1) sysbench --test=cpu --cpu-max-prime=20000 Experiment:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for user-level and kernel-level usage.
  Results: COEN241:HW1:TEST1

4.2) sysbench --test=cpu --cpu-max-prime=30000 Experiment:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for user-level and kernel-level usage.
  Results: COEN241:HW1:TEST2

4.3) sysbench --test=cpu --cpu-max-prime=35000 Experiment:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for user-level and kernel-level usage.
  Results: COEN241:HW1:TEST3

4.4) FILE IO Test -1:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for disk IO, Latency and disk utilization.

  If you want to see results at each stage execute :

  sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare

  sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run

  sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup

  Results: COEN241:HW1:FILE TEST1

4.5) FILE IO Test -2:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for disk IO, Latency and disk utilization.

  sysbench --num-threads=32--test=fileio --file-total-size=3G --file-test-mode=seqwr prepare

  sysbench --num-threads=32 --test=fileio --file-total-size=3G --file-test-mode=seqwr run

  sysbench --num-threads=32 --test=fileio --file-total-size=3G --file-test-mode=seqwr cleanup

  Result: COEN241:HW1:FILE TEST2

4.6) FILE IO Test -3:

- Check for any current processes by executing “top -i” and make sure no other tasks are running in qemu and container.
- Execute the bash script created to get the results.
- Check the memory during each execution for disk IO, Latency and disk utilization.

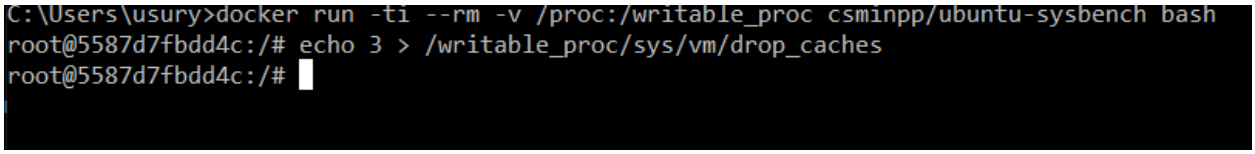
  sysbench --num-threads=16--test=fileio --file-total-size=4G --file-test-mode=rndrw prepare

  sysbench --num-threads=16 --test=fileio --file-total-size=4G --file-test-mode=rndrw run

  sysbench --num-threads=16--test=fileio --file-total-size=4G --file-test-mode=rndrw cleanup

  Result: COEN241:HW1:FILE TEST3

  Dropping cache for the docker container after each run :



4.7) Shell Scripts for the experiments:

The shell scripts are as attached, to run the script make sure to follow the instructions above and execute using “./script_name.sh”.

https://github.com/skiranu/COEN241-HW1.git

4.8) Analysis of the experiments:

Thorough analyses for each of the experiments are present in the attached excel file. It was seen that the docker containers were always faster than the qemu VM overall. The impact of the native system was highest on Qemu VM as compared to the container image, the recover of container image after a halt/ freeze scenario was much quicker than Qemu VM.

5) Git repository information:

Repository name:COEN241-HW1

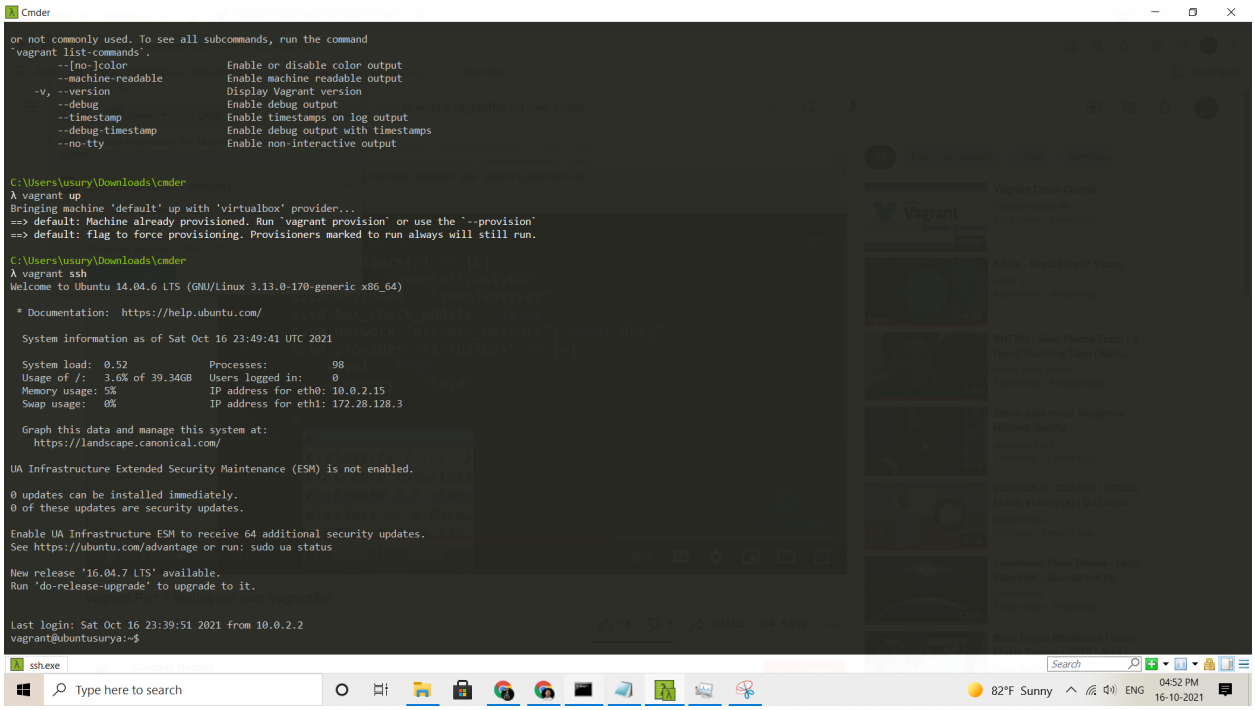
URL:https://github.com/skiranu/COEN241-HW1.git

Since it is a private repository, I have sent the invite to the instructor.

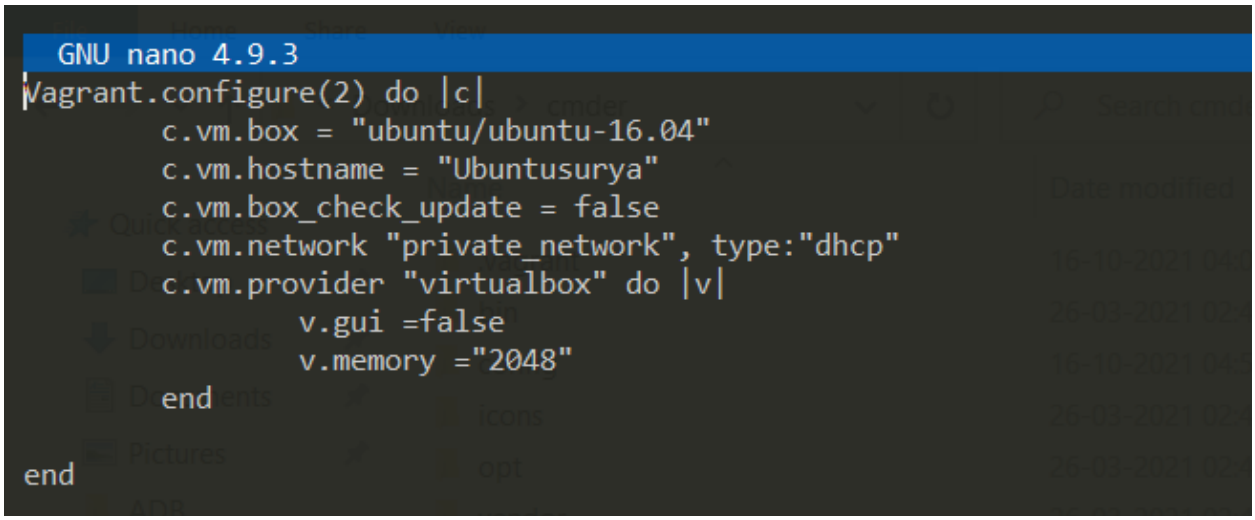
6)Vagrant file:

I had fun writing ruby code, the vagrant script was tested multiple times with different parameters for IP/DHCP. After multiple failed attempts to use the virtual box, I disabled all the anti-virus software and finally was able to boot into GUI mode through VirtualBox.

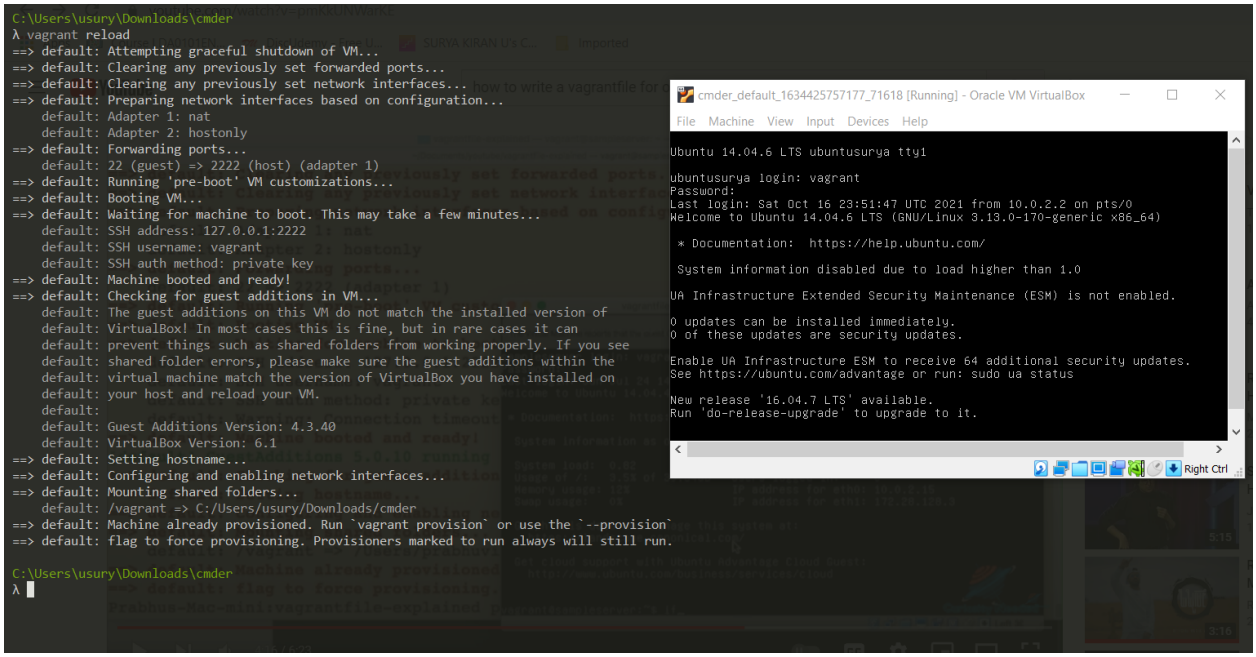
Booting of ubuntu through **“vagrant ssh”**



A snippet of vagrant file:



Ubuntu logged in through GUI(Virtual Box):**vagrant reload**



Steps to execute vagrant file:

1) nano/touch vagrantfile.
2) Execute **vagrant up.**
3) Execute **vagrant ssh** for ssh connectivity.
4) Execute **vagrant reload** after making changes to vagrant file to boot using Virtual box.