

CSI-5-OSY Coursework 3

3807942 Kamran Bhatti

Due: 7th January 2021 23:59

Contents

Deployment.....	4
Steps	4
Challenges.....	27
Configuration.....	28
Steps	28
Programming.....	40
api.py	40
Challenges.....	43
Test Results.....	44
Appraisal.....	50
Bibliography	51
Appendix A – api.py.....	52

Figure 1.....	4
Figure 2.....	5
Figure 3.....	5
Figure 4.....	6
Figure 5.....	7
Figure 6.....	8
Figure 7.....	9
Figure 8.....	10
Figure 9.....	11
Figure 10.....	12
Figure 11.....	13
Figure 12.....	14
Figure 13.....	15
Figure 14.....	16
Figure 15.....	16
Figure 16.....	17
Figure 17.....	18
Figure 18.....	19
Figure 19.....	20
Figure 20.....	20
Figure 21.....	21
Figure 22.....	21
Figure 23.....	22
Figure 24.....	23
Figure 25.....	24
Figure 26.....	25
Figure 27.....	26
Figure 28.....	27
Figure 29.....	28
Figure 30.....	29
Figure 31.....	29

Figure 32.....	30
Figure 33.....	30
Figure 34.....	31
Figure 35.....	31
Figure 36.....	32
Figure 37.....	33
Figure 38.....	33
Figure 39.....	34
Figure 40.....	35
Figure 41.....	35
Figure 42.....	36
Figure 43.....	37
Figure 44.....	38
Figure 45.....	39
Figure 46.....	40
Figure 47.....	40
Figure 48.....	40
Figure 49.....	40
Figure 50.....	40
Figure 51.....	41
Figure 52.....	42
Figure 53.....	42
Figure 54.....	42
Figure 55.....	42
Figure 56.....	42
Figure 57.....	43
Figure 58.....	44
Figure 59.....	44
Figure 60.....	45
Figure 61.....	45
Figure 62.....	46
Figure 63.....	47
Figure 64.....	48
Figure 65.....	49
 Table 1	 44

Deployment

The hypervisor hosts 3 Ubuntu 64 bits servers with minimal configuration.

Therefore, an Ubuntu 64 bit server image must be downloaded and installed on each of the virtual machines.

Steps 1 and 2 refer to downloading the Ubuntu server image via (Canonical Ltd., 2020). (Oracle, 2020) is used from step 3 onwards for creating the virtual machine instances. Except for steps 13 through 19, all the following steps will be used when creating each virtual machine instance.

Steps

Step 1: User left click on "Search with DuckDuckGo or enter an address (edit)" in "Mozilla Firefox"

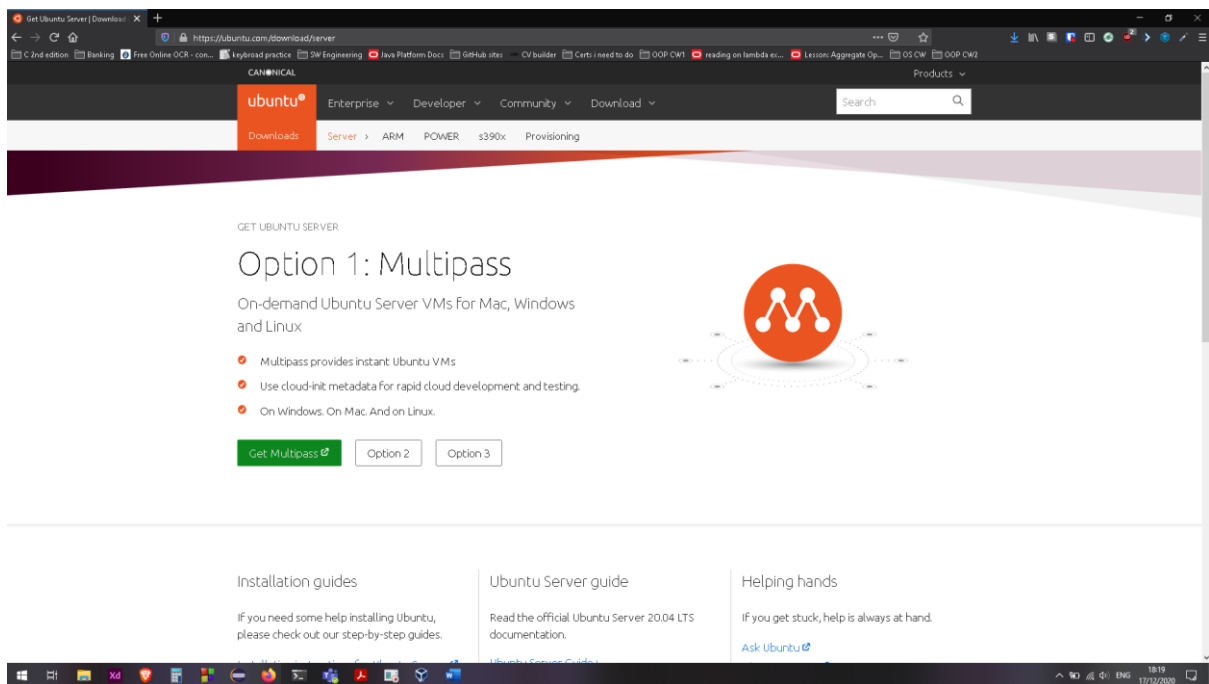


Figure 1

Step 2: User left click on "Option 3 (button)" in "Get Ubuntu Server | Download | Ubuntu — Mozilla Firefox"

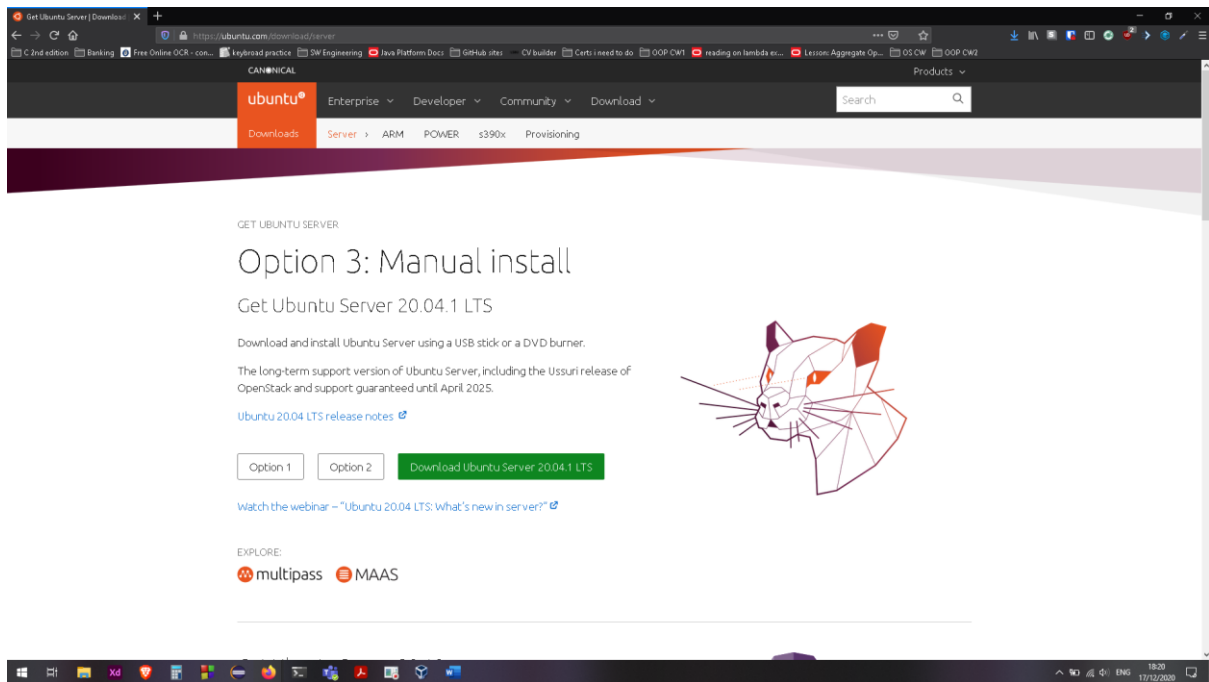


Figure 2

Step 3: User left click on "New (button)" in "Oracle VM VirtualBox Manager"

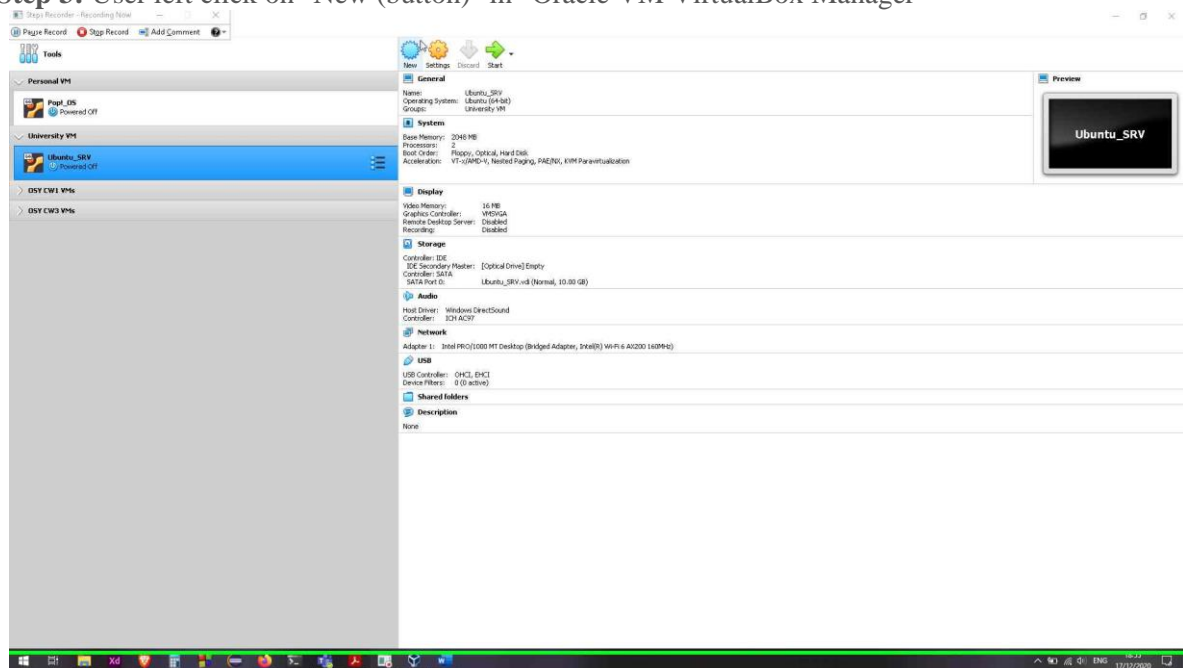


Figure 3

Step 4: User keyboard input on "Create Virtual Machine (window)" in "Create Virtual Machine" [... BACKSPACE ...]

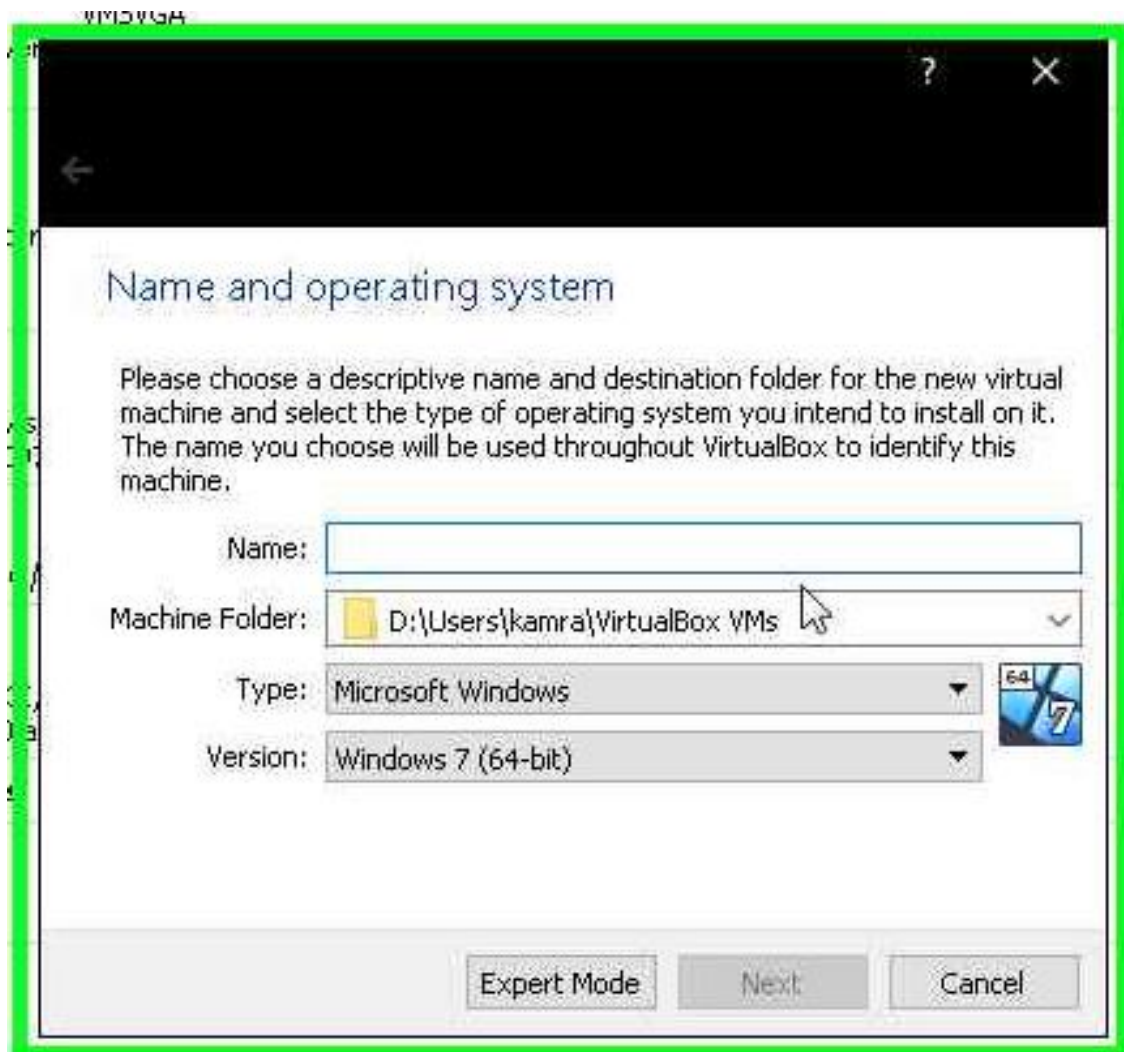


Figure 4

Step 5: User left click on "Version: Down (combo box)" in "Create Virtual Machine"

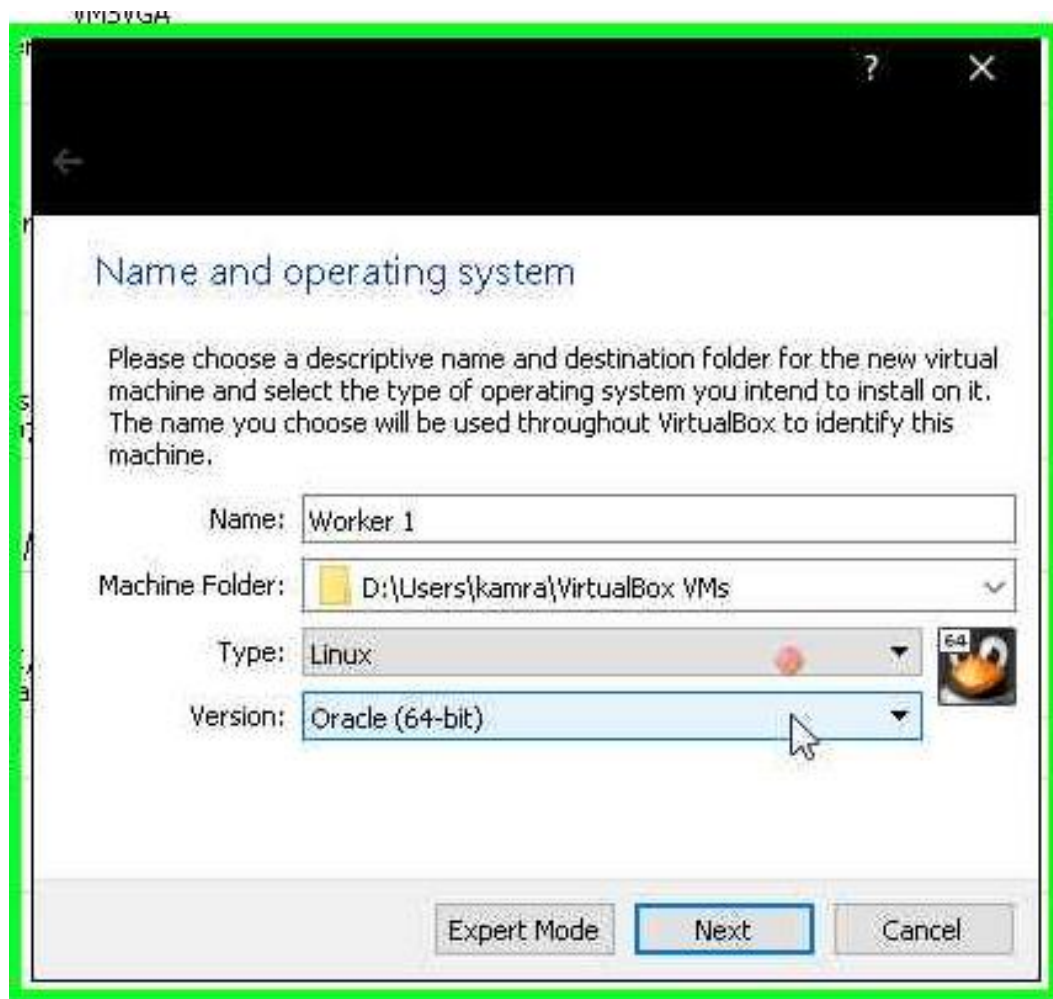


Figure 5

Step 6: User left click in "VirtualBox"

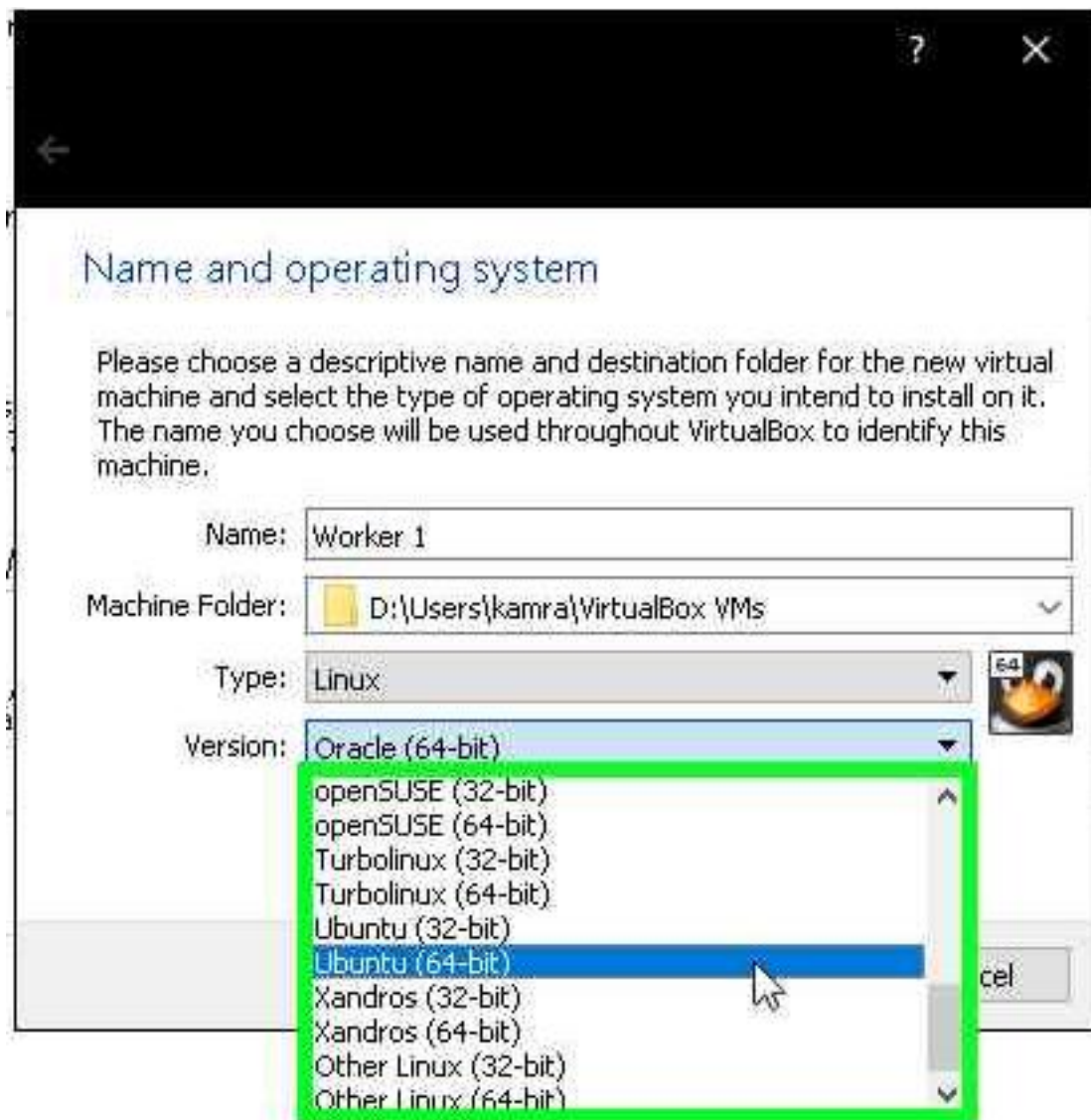


Figure 6

Step 7: User left click on "Next Enter (button)" in "Create Virtual Machine"

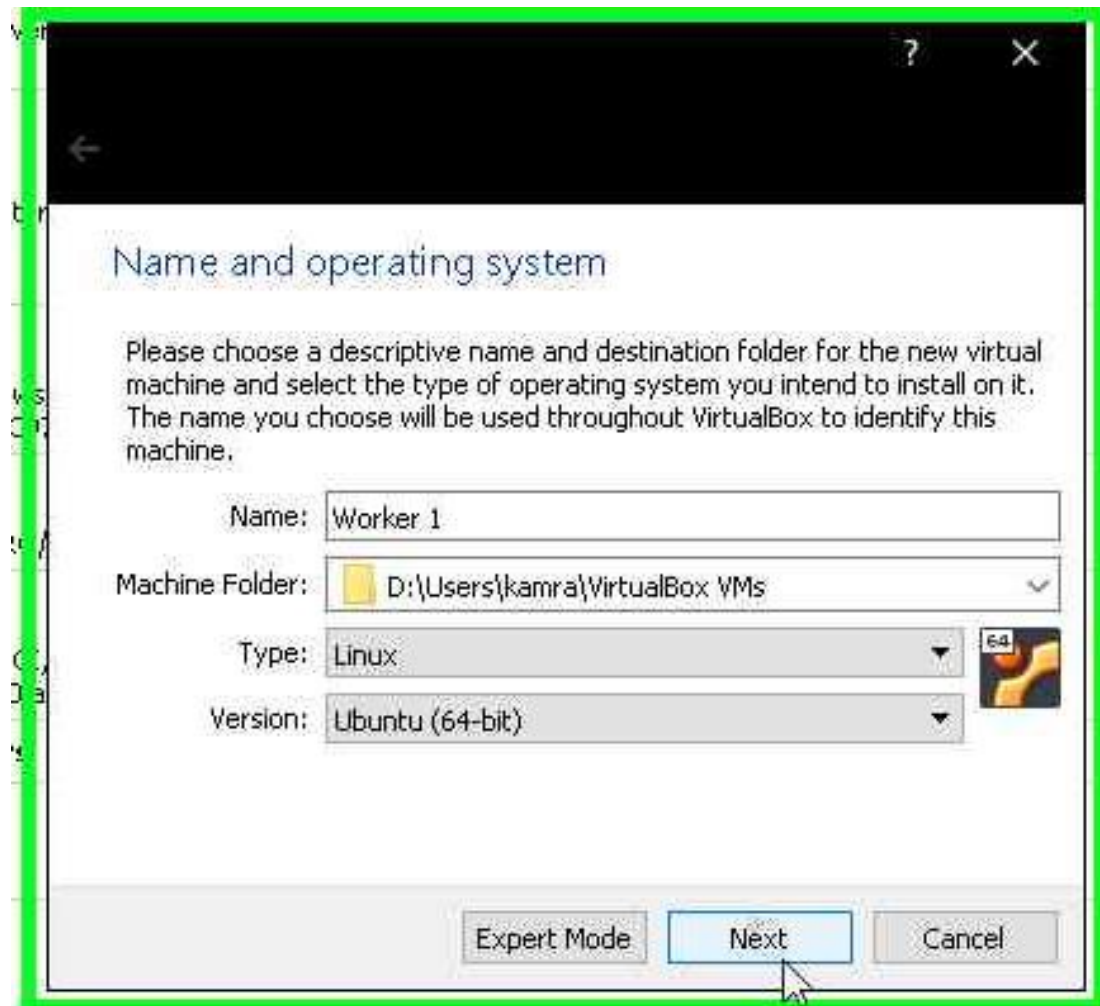


Figure 7

Step 8: User left click on "Next Enter (button)" in "Create Virtual Machine"

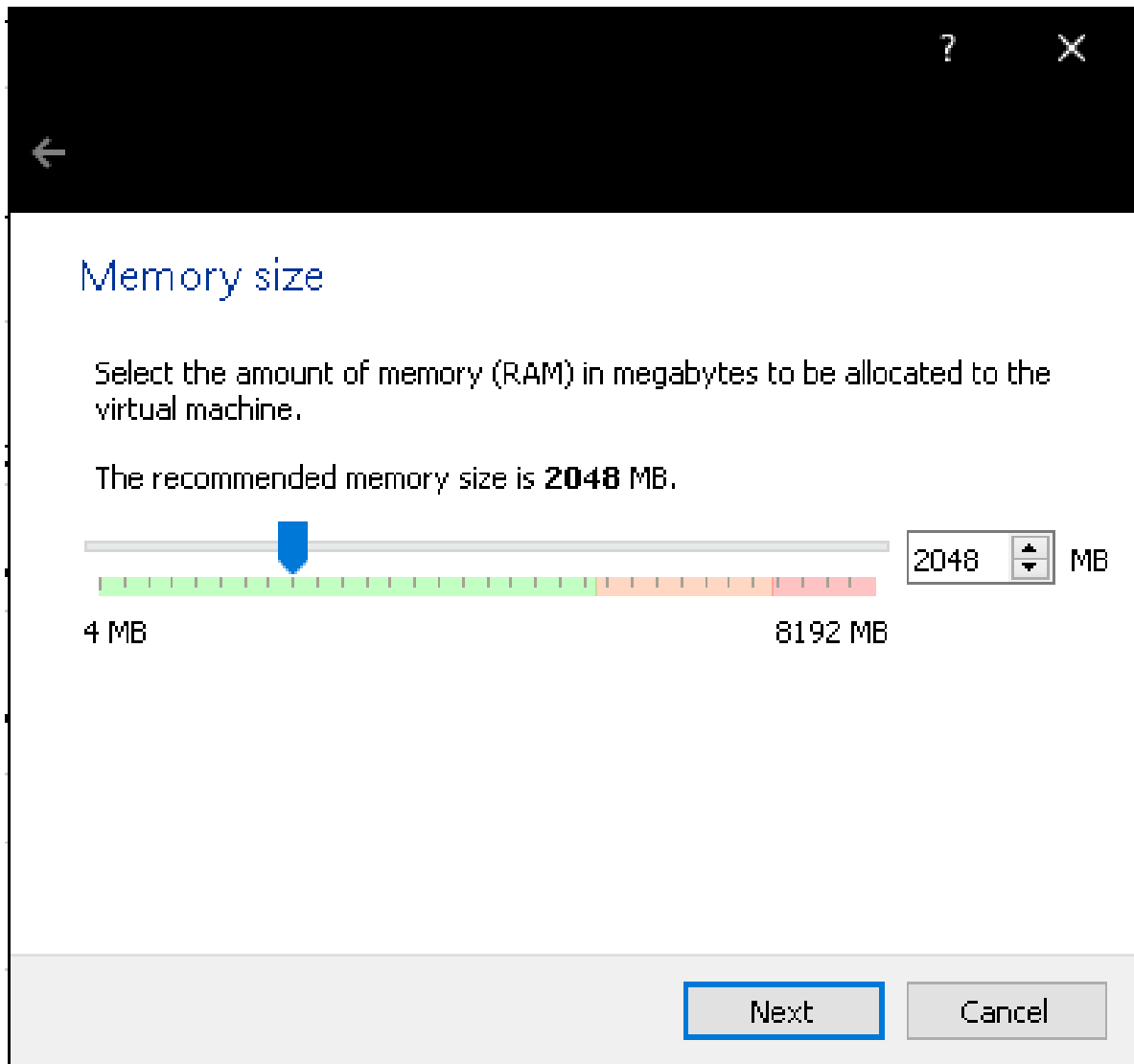


Figure 8

Step 9: User left click on "Create Enter (button)" in "Create Virtual Machine"

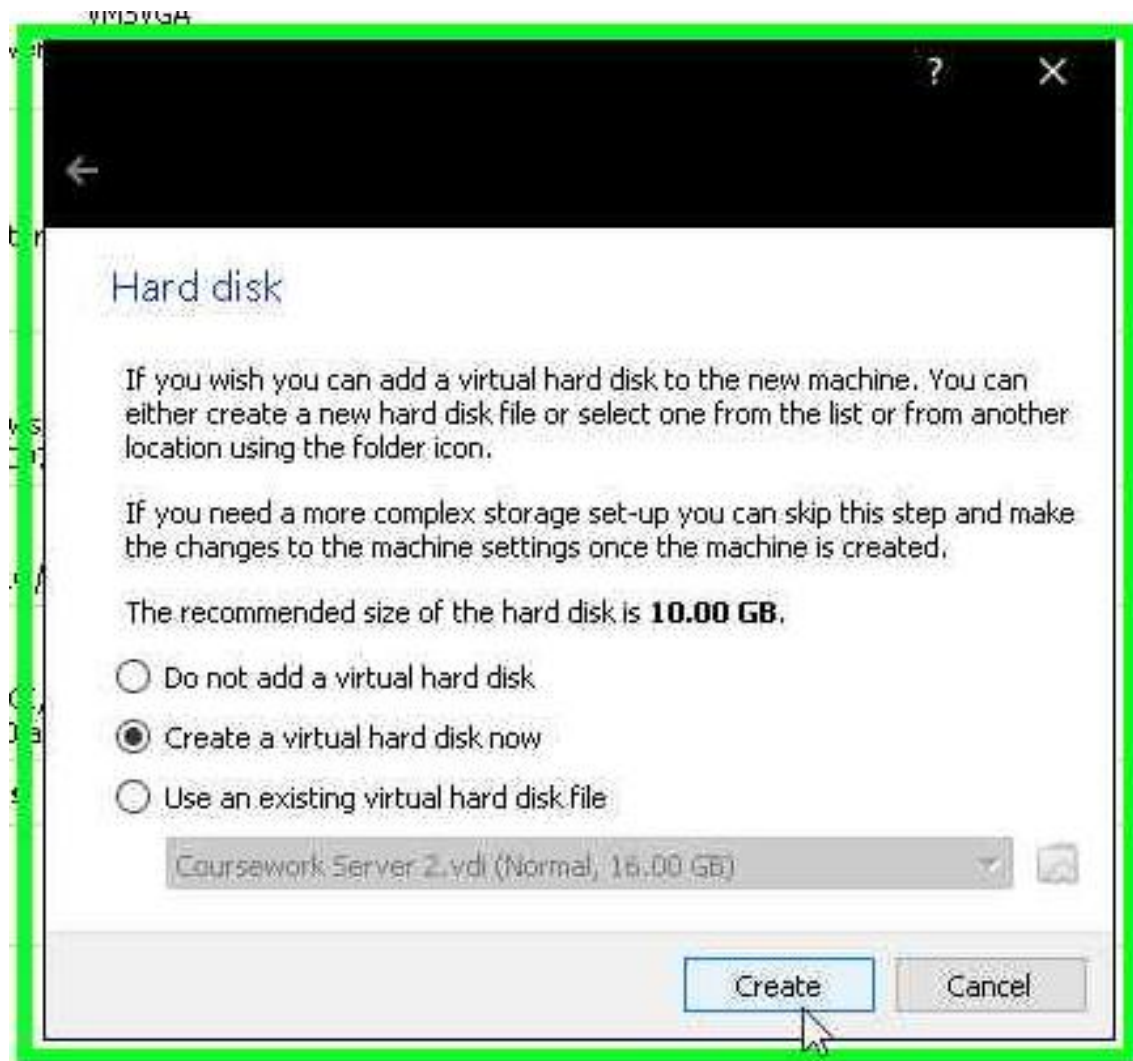


Figure 9

Step 10: User left click on "Next Enter (button)" in "Create Virtual Hard Disk"

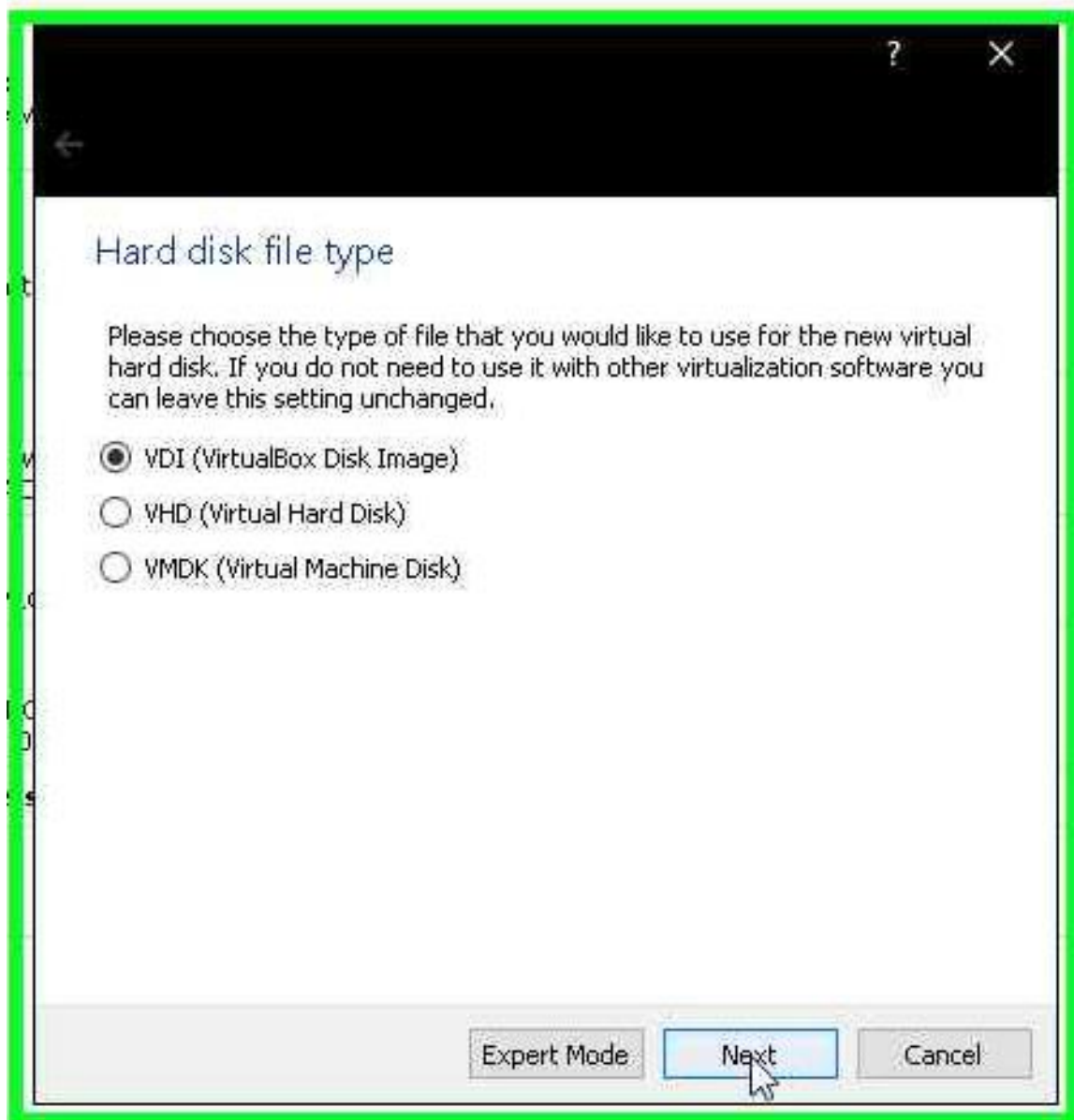


Figure 10

Step 11: User left click on "Next Enter (button)" in "Create Virtual Hard Disk"

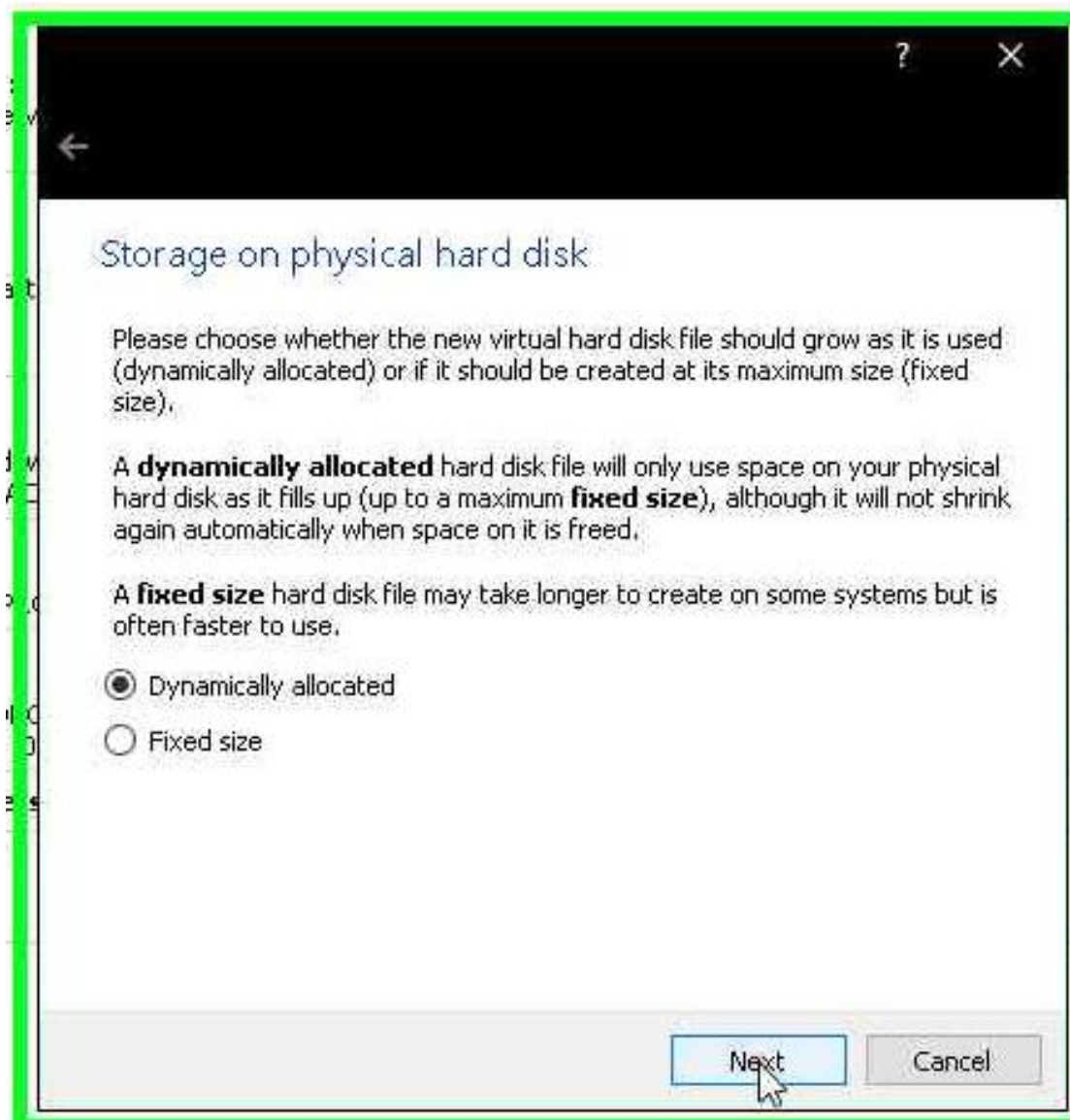


Figure 11

Step 12: User left click on "Create Enter (button)" in "Create Virtual Hard Disk"

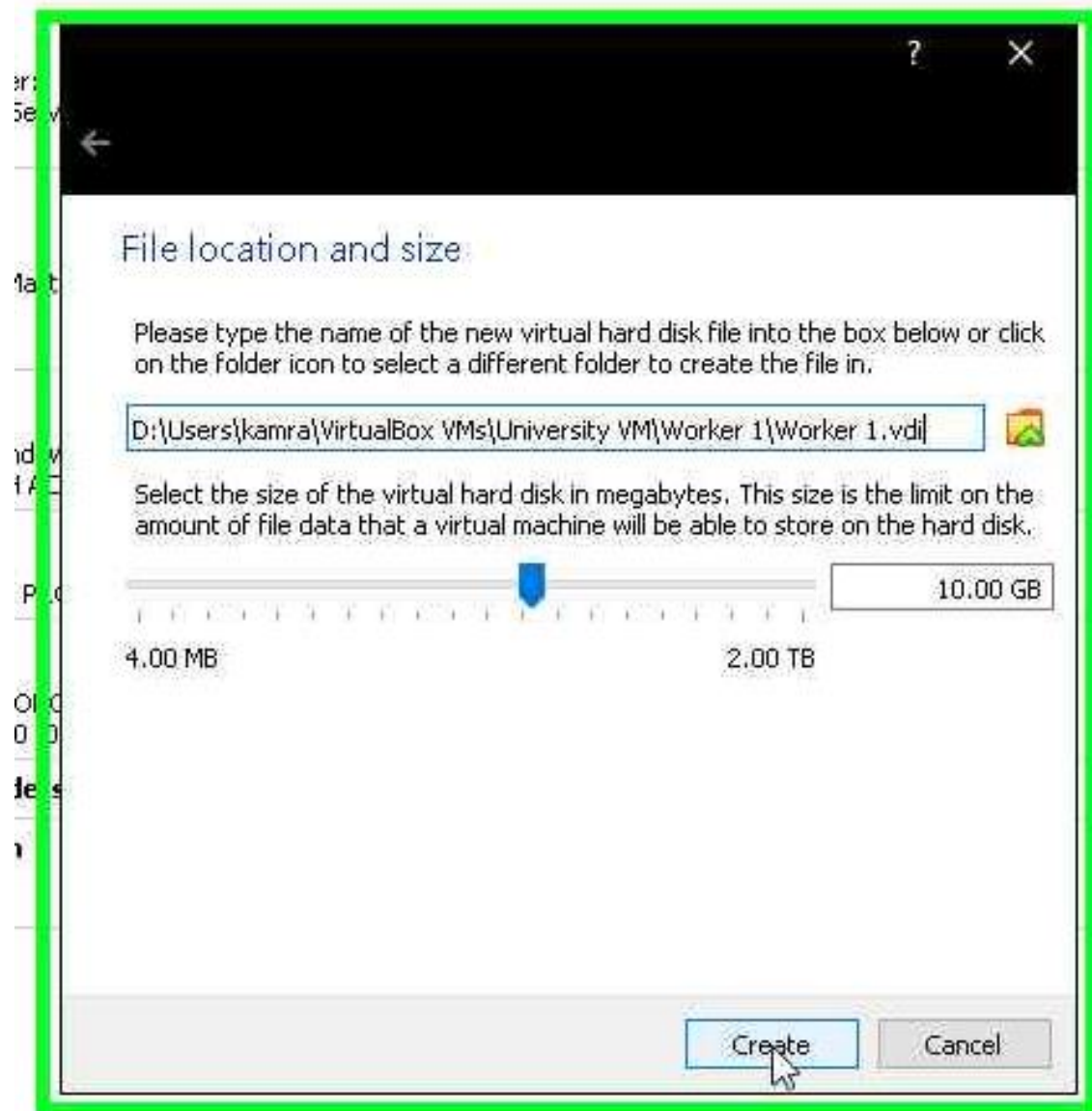


Figure 12

Step 13: User right click on "Worker 1 (list item)" in "Oracle VM VirtualBox Manager"

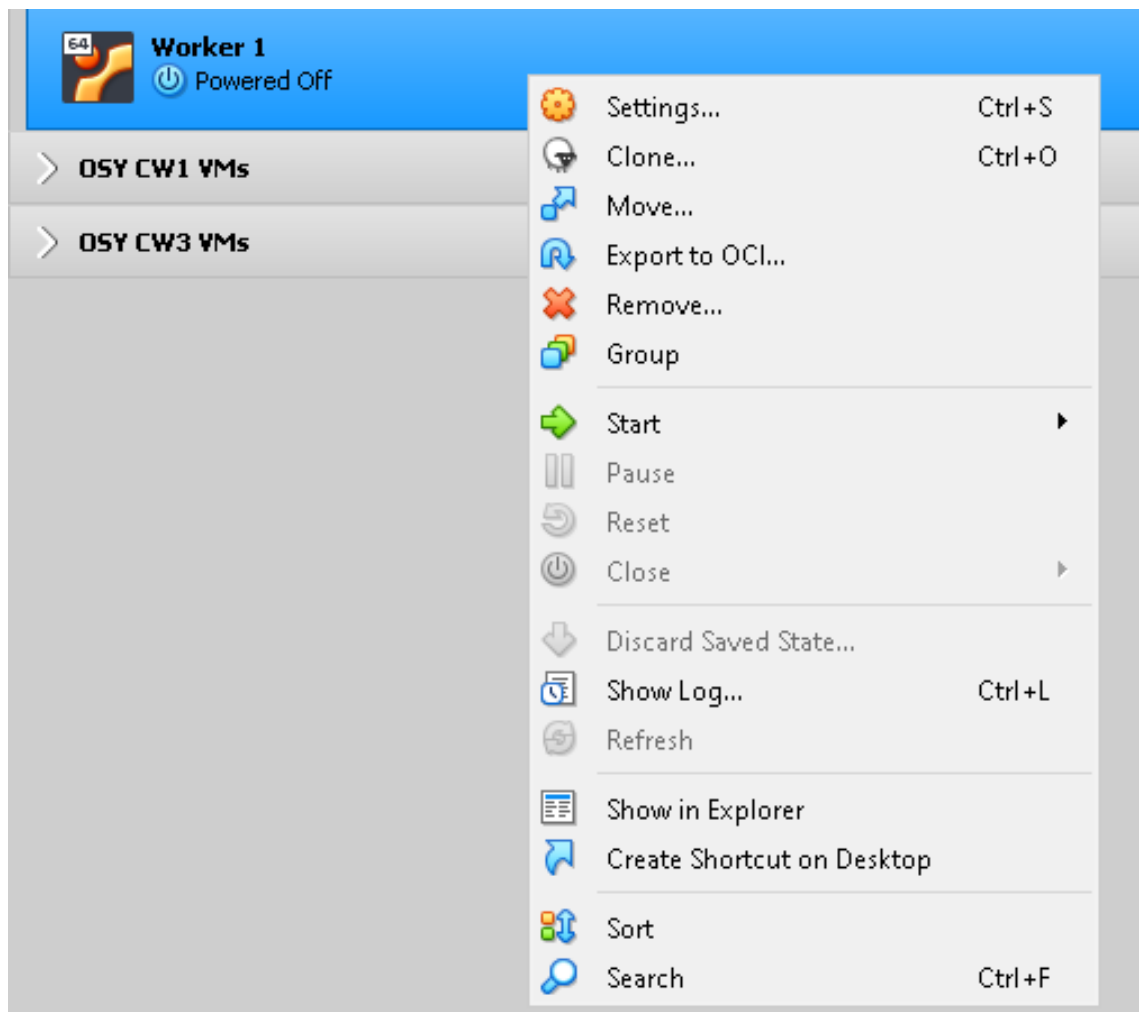


Figure 13

Step 14User left click on "Clone... Ctrl+O (menu item)" in "VirtualBox"

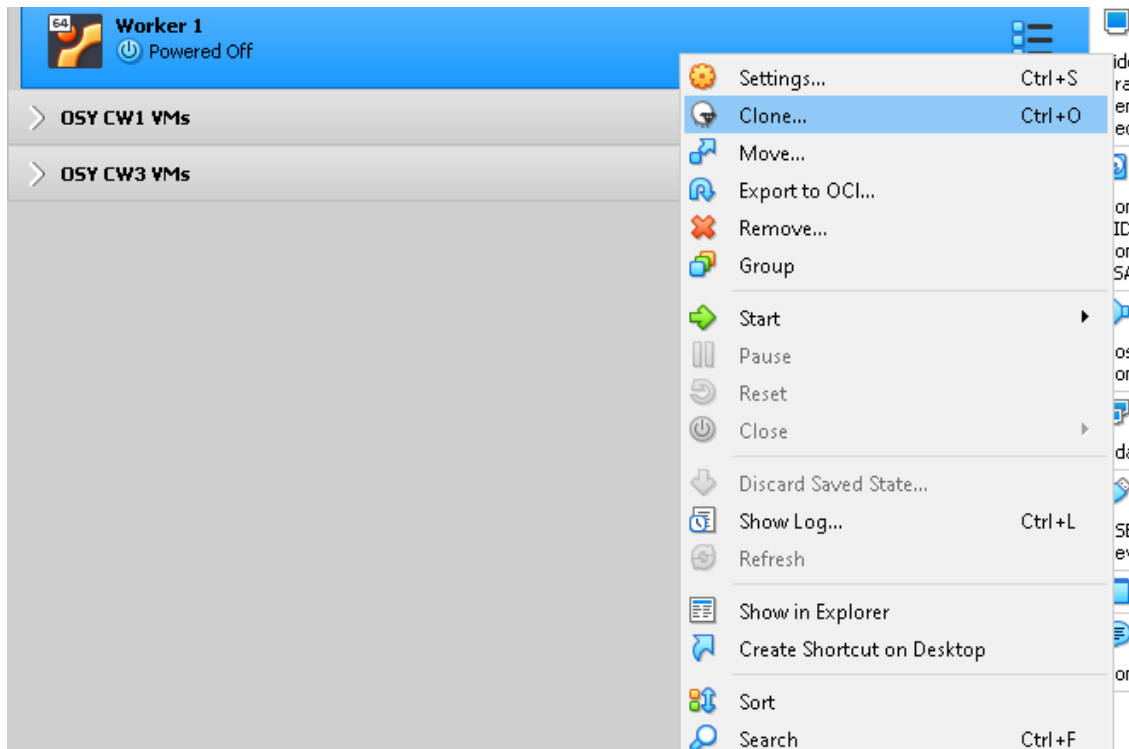


Figure 14

Step 15: User left click on "Clone Virtual Machine (window)" in "Clone Virtual Machine"

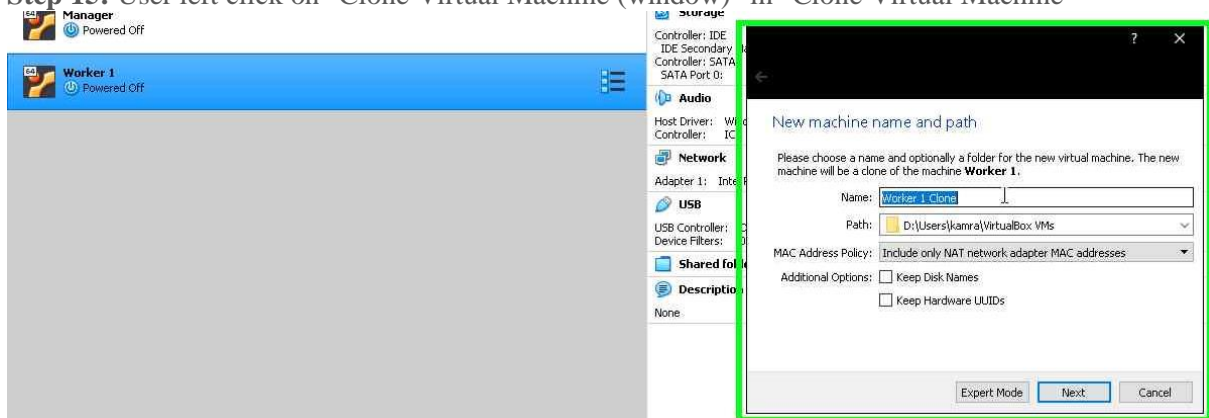


Figure 15

Step 16: User left click on "MAC Address Policy: Down (combo box)" in "Clone Virtual Machine"

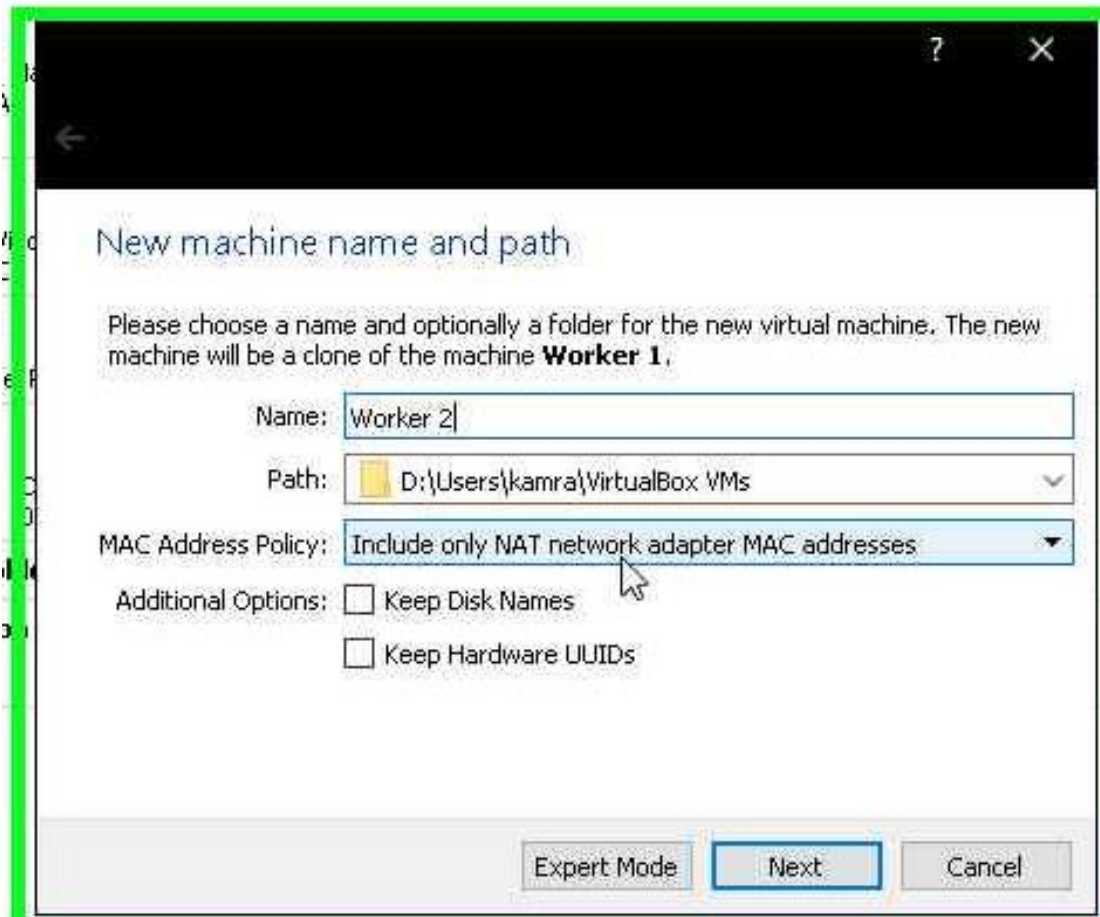


Figure 16

Step 17: User left click in "VirtualBox"

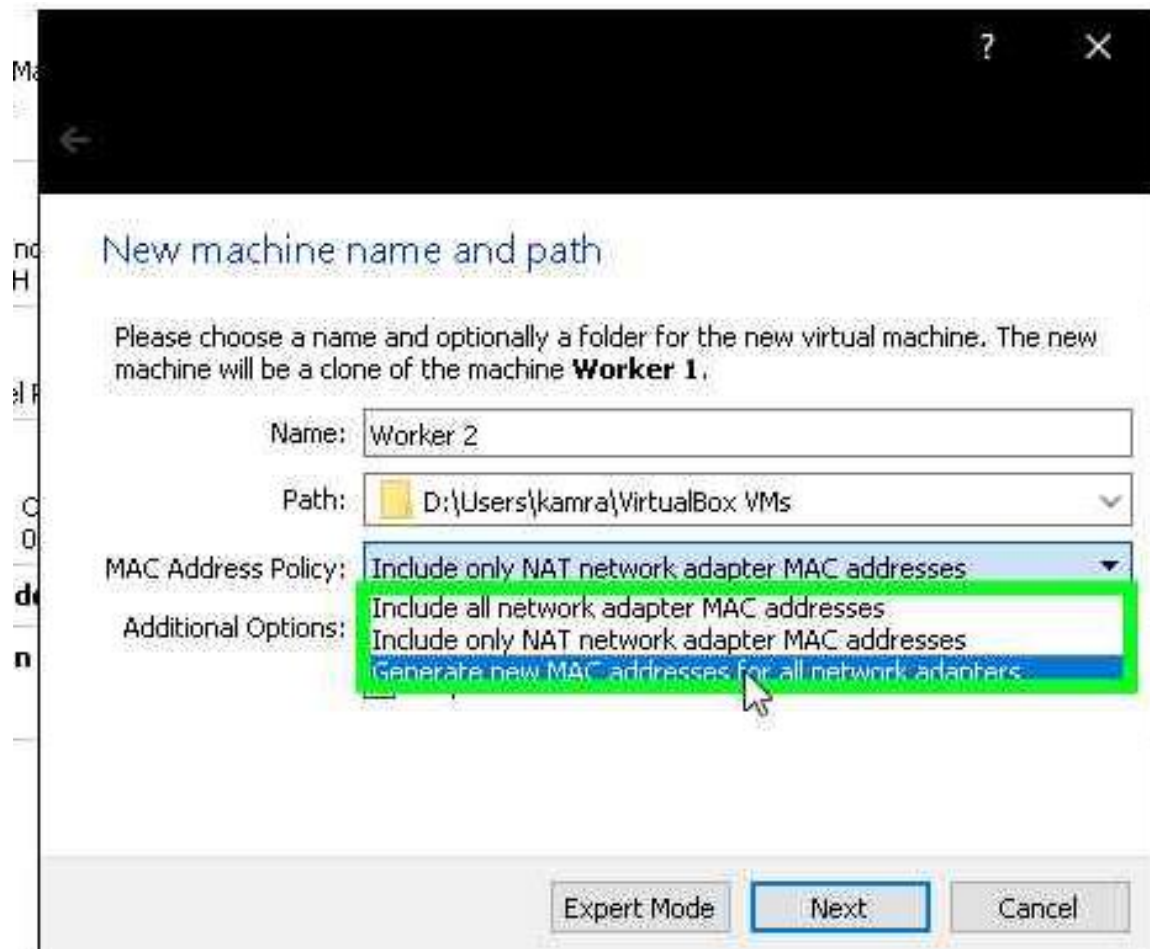


Figure 17

Step 18: User left click on "Next Enter (button)" in "Clone Virtual Machine"

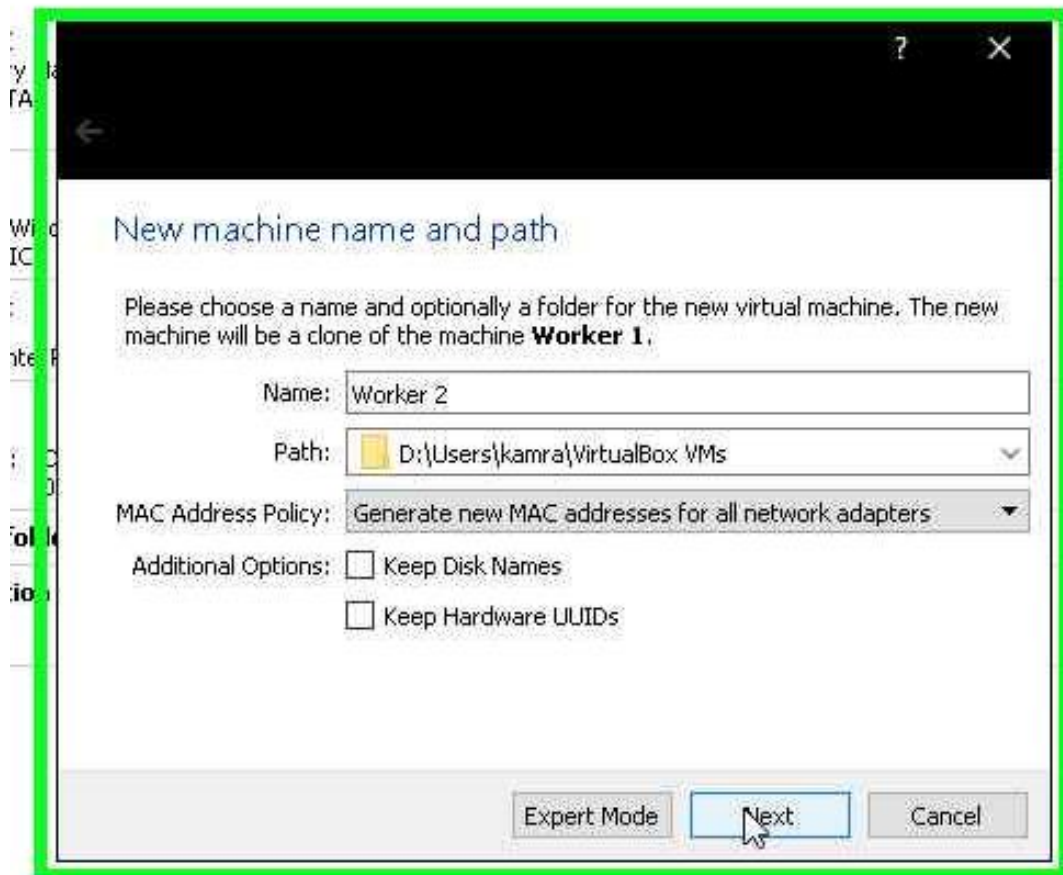


Figure 18

Step 19: User left click on "Clone Enter (button)" in "Clone Virtual Machine"

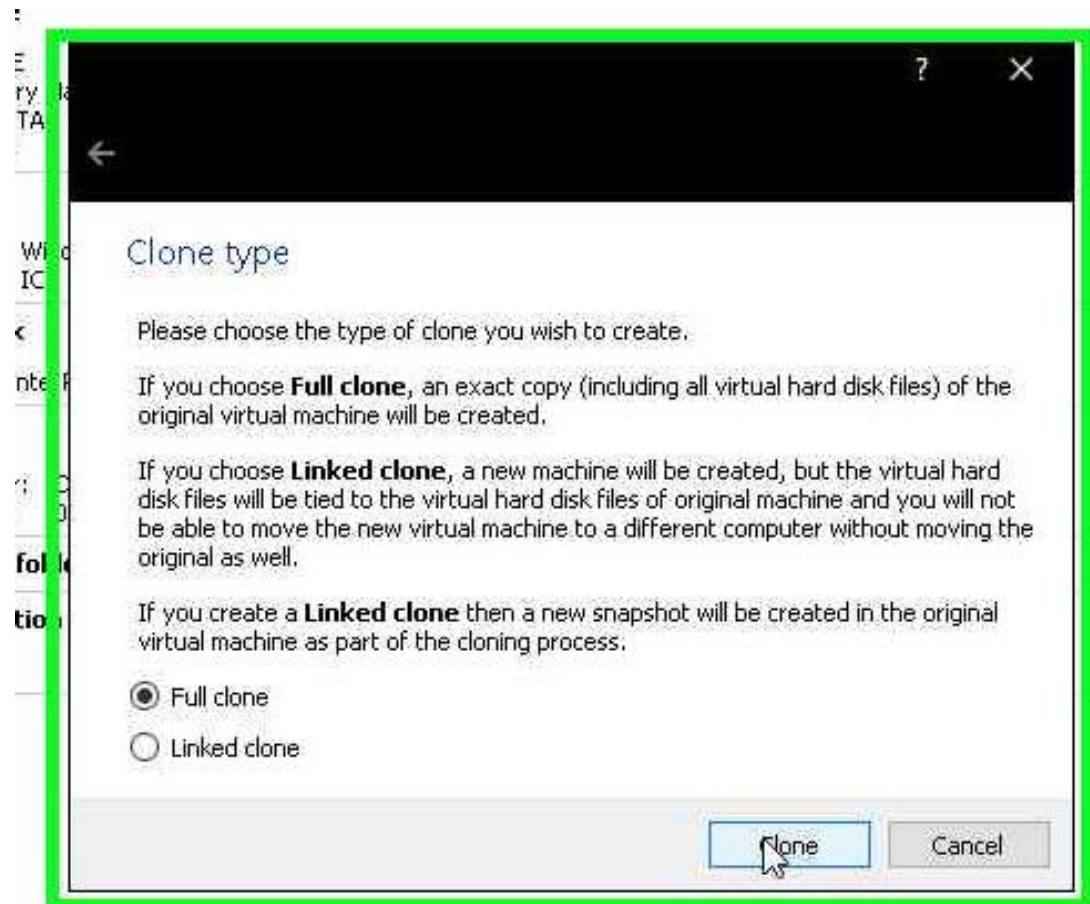


Figure 19

Step 20: User left click on "Worker 1 (list item)" in "Oracle VM VirtualBox Manager"

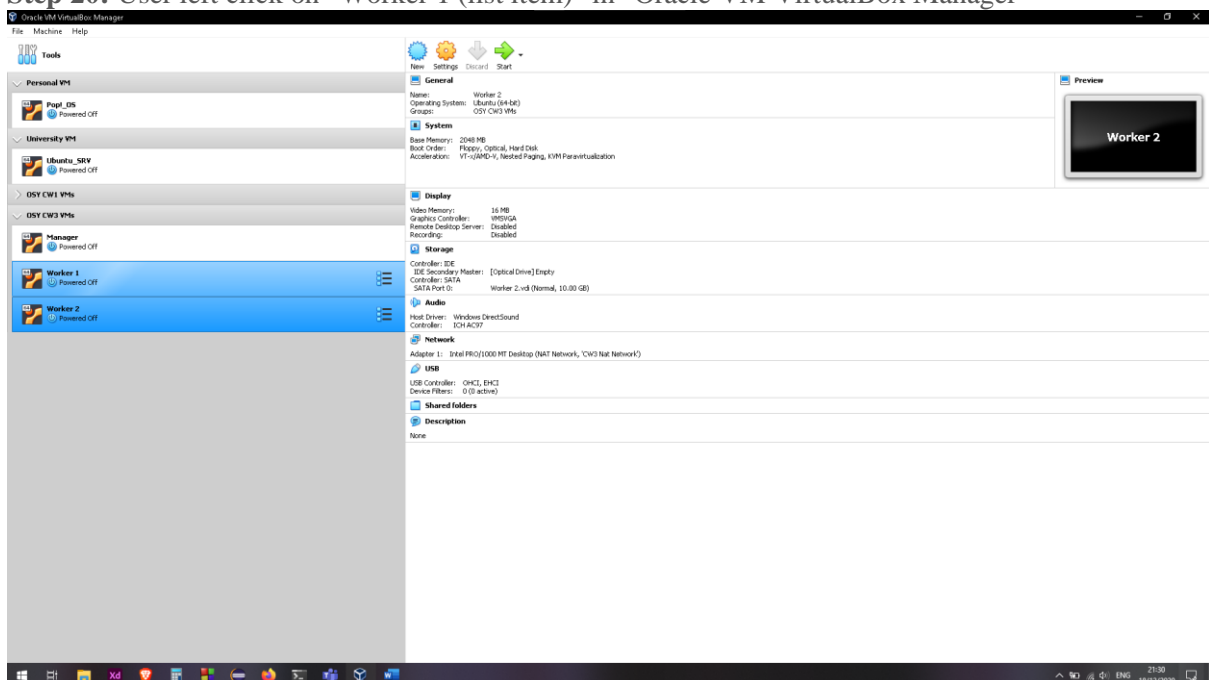


Figure 20

Step 21: User left click on "Oracle VM VirtualBox Manager (window)" in "Oracle VM VirtualBox Manager"

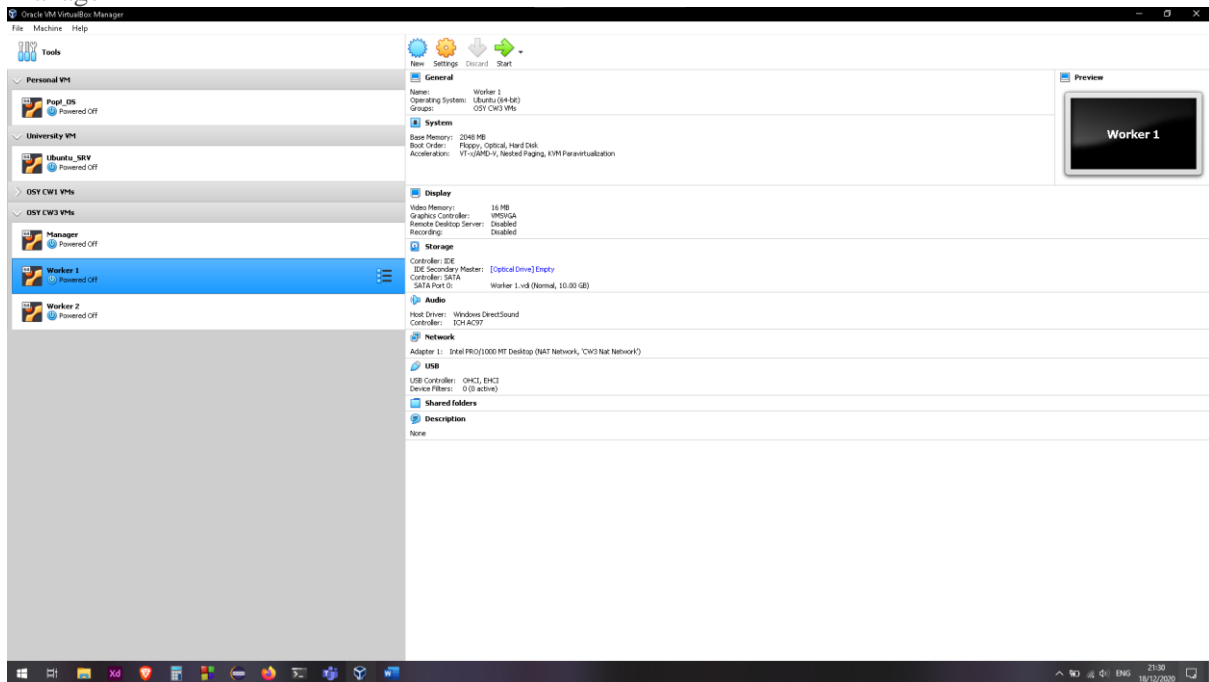


Figure 21

Step 22: User left click on "ubuntu-20.04.1-live-server-amd64.iso (menu item)" in "VirtualBox"



Figure 22

Step 23: User left click on "Start (button)" in "Oracle VM VirtualBox Manager"

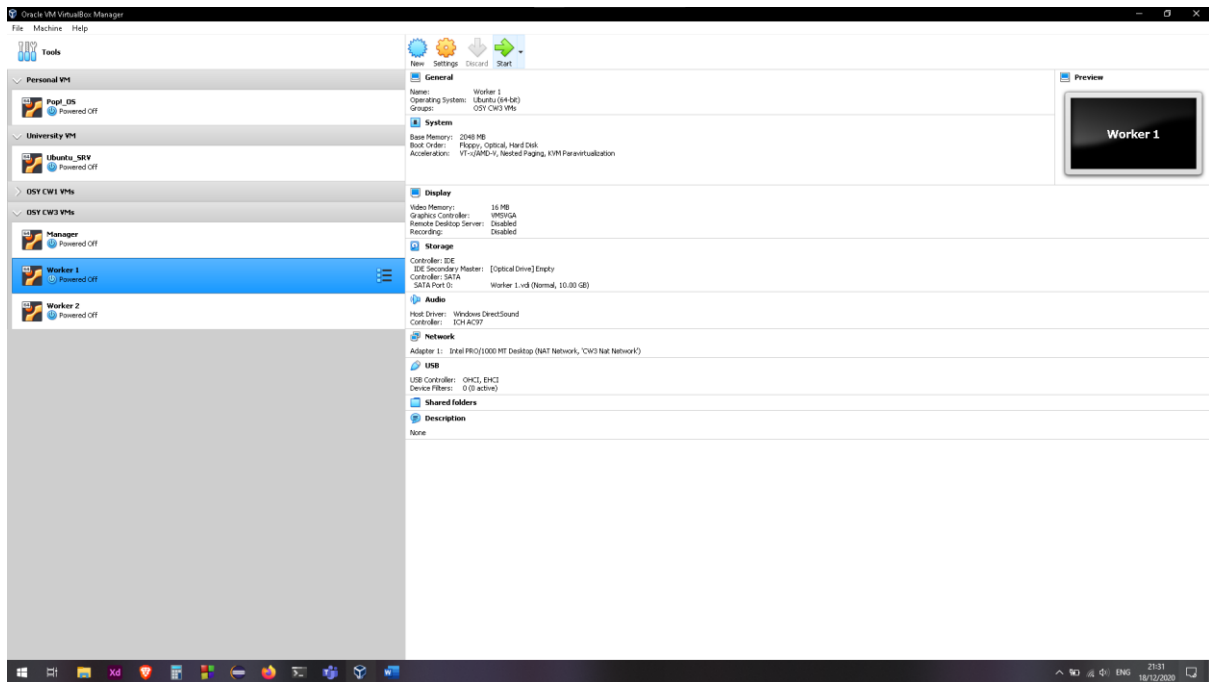


Figure 23

Step 24: User keyboard input on "Worker 1 [Running] - Oracle VM VirtualBox (window)" in "Worker 1 [Running] - Oracle VM VirtualBox" [... SHIFT-SHIFT ... SHIFT-SHIFT]. The server name and username is specific to the virtual machine instance for instance, each server will have a unique server name however, the worker machines will have the same username for simplicity.

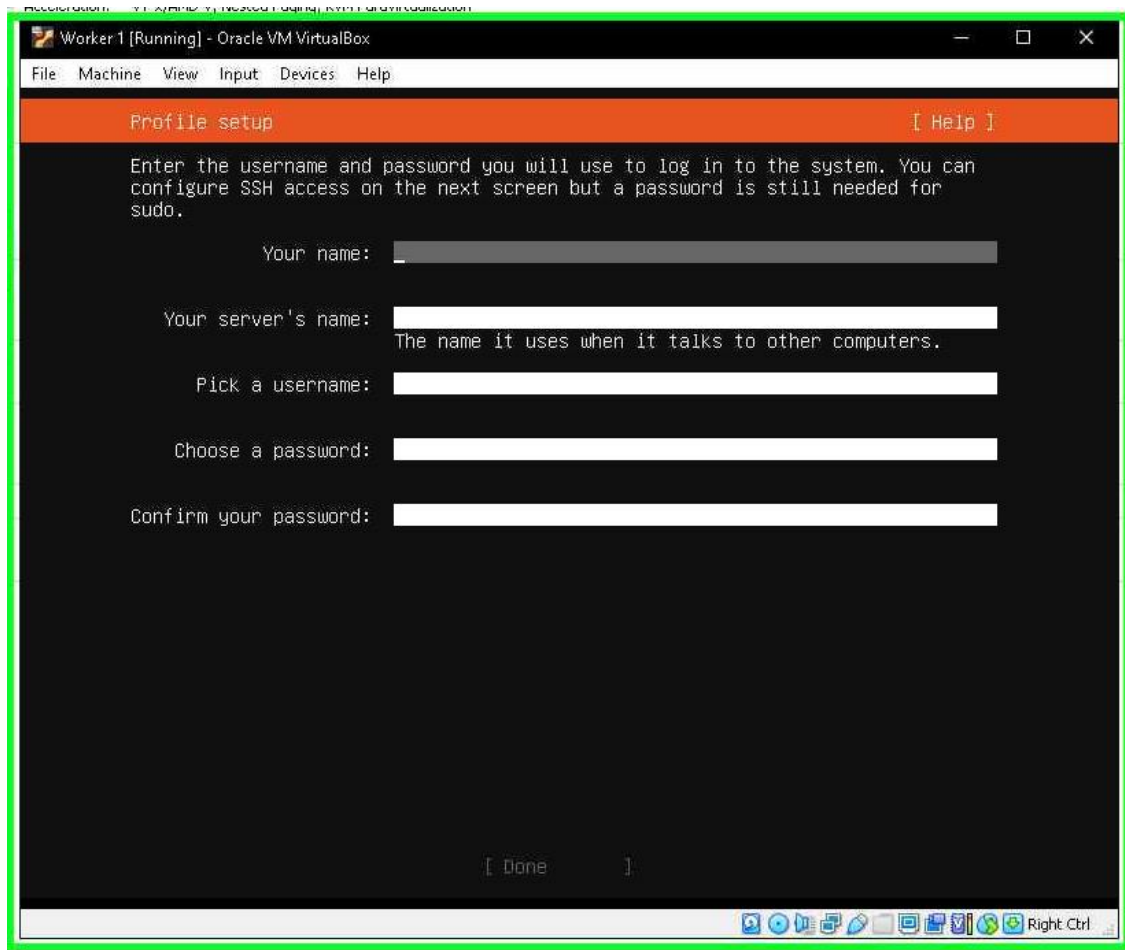


Figure 24

Step 25: User mouse wheel down on "qt_scrollarea_viewportWindow (pane)" in "Worker 1 [Running] - Oracle VM VirtualBox"

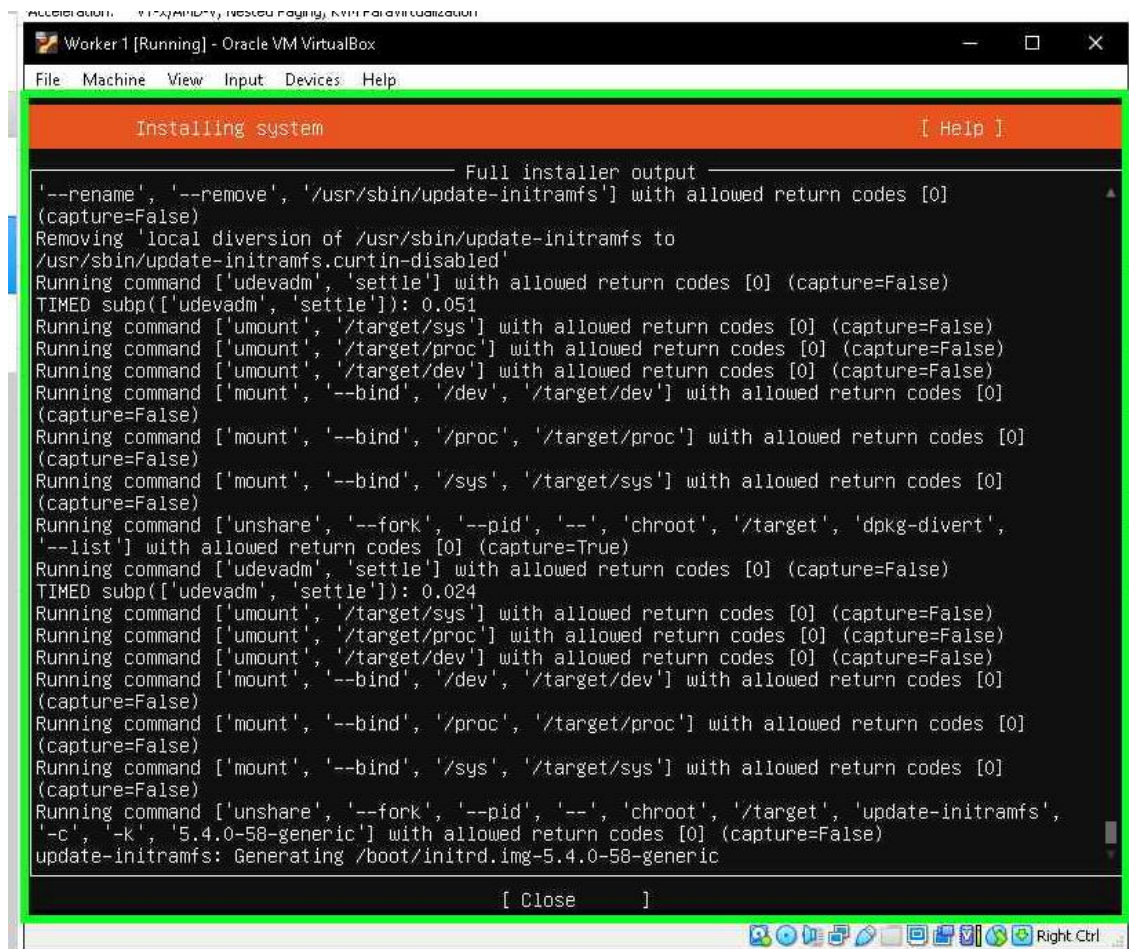


Figure 25

Step 26: User left click on "qt_scrollarea_viewportWindow (pane)" in "Worker 1 [Running] - Oracle VM VirtualBox"

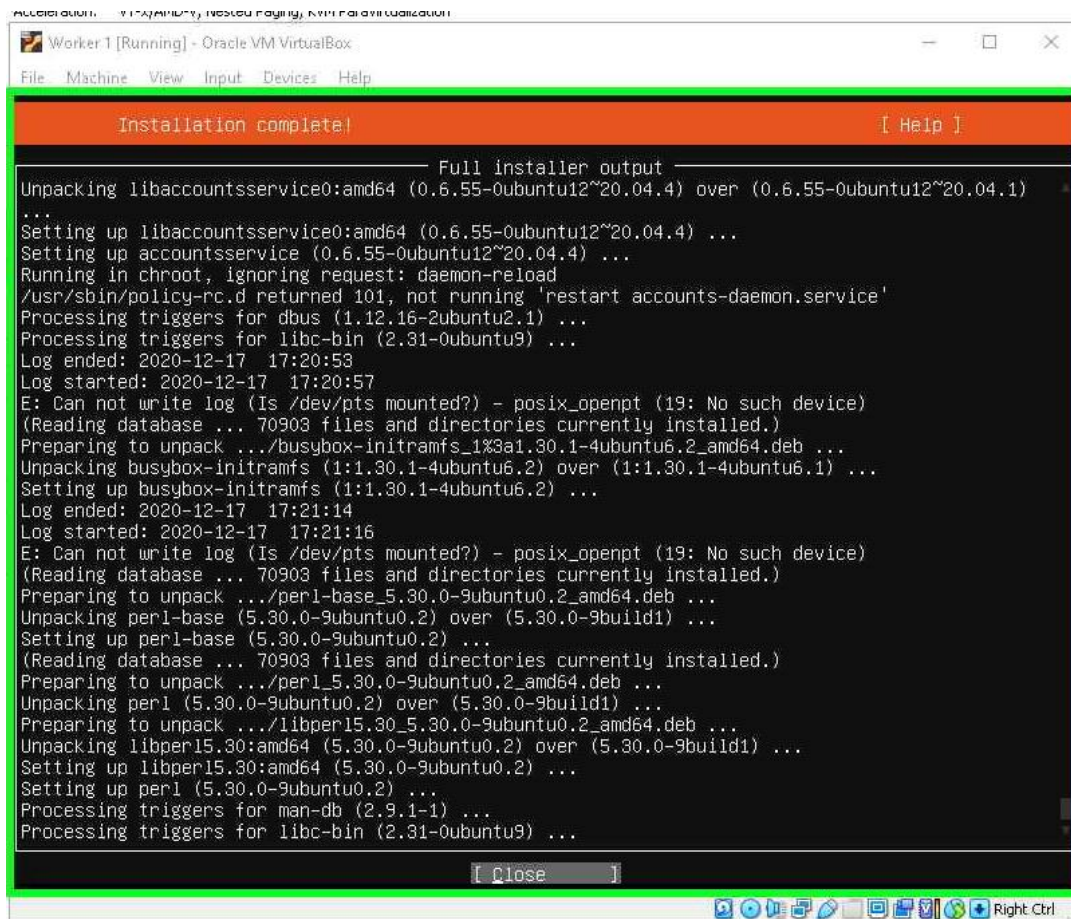


Figure 26

Step 27: User Comment: "This command should fix the /dev/pts is not mounted error"

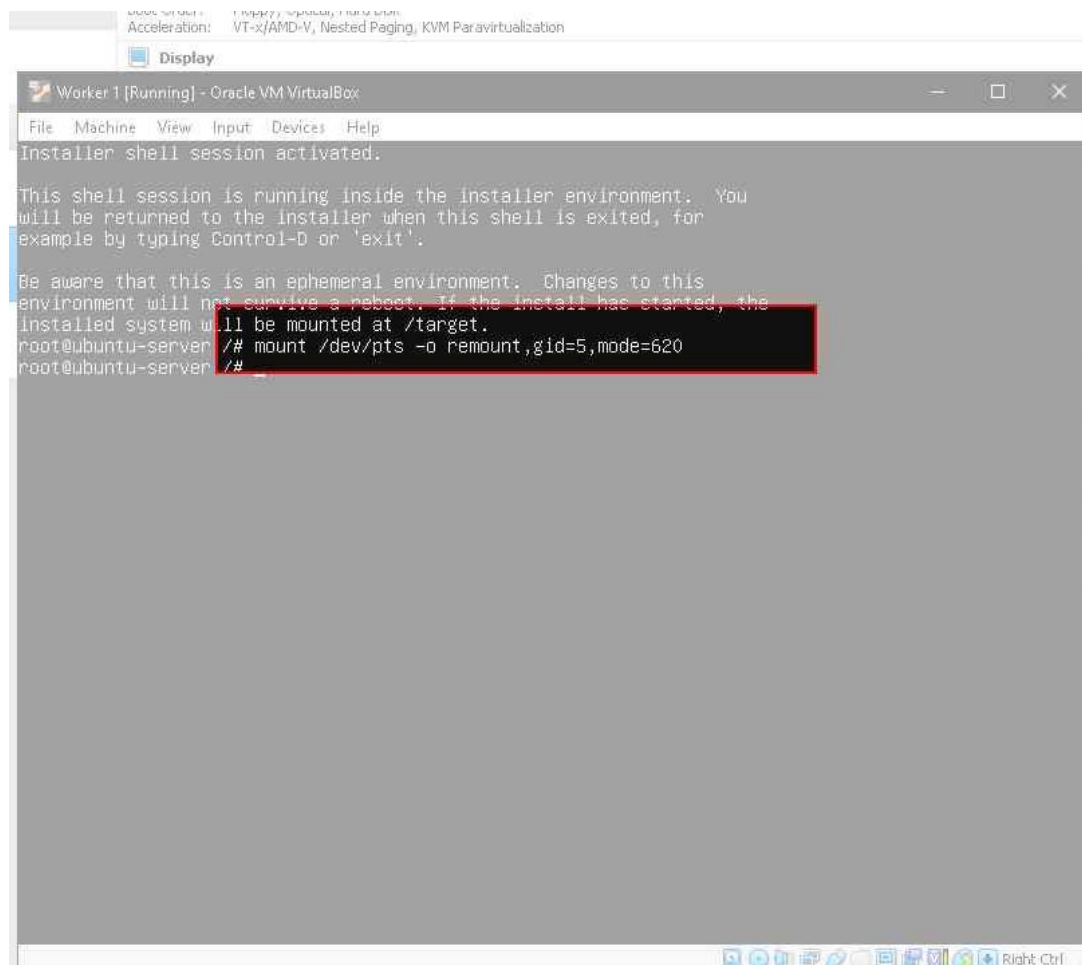


Figure 27

Step 28: User Comment: "We need to execute this command after we first install Ubuntu on each of the Virtual Machines to make sure that all installed packages are up to date"

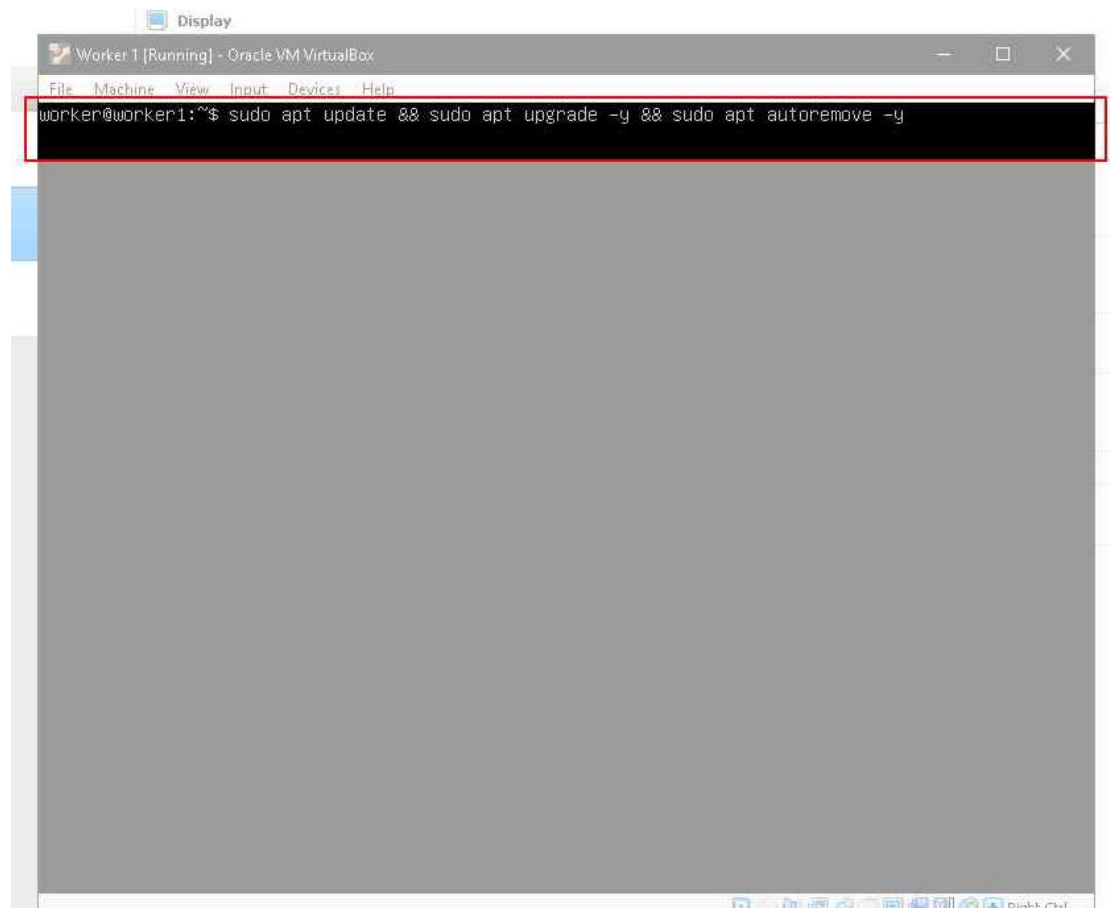


Figure 28

Challenges

As seen in Figure 26, an error was encountered while installing the Ubuntu server image in which a log file couldn't be written to a directory correctly. To rectify the issue, (Score_Under, 2015) was used to mount the /dev/pts directory with the correct user permissions for the installer to be able to write to the directory.

Configuration

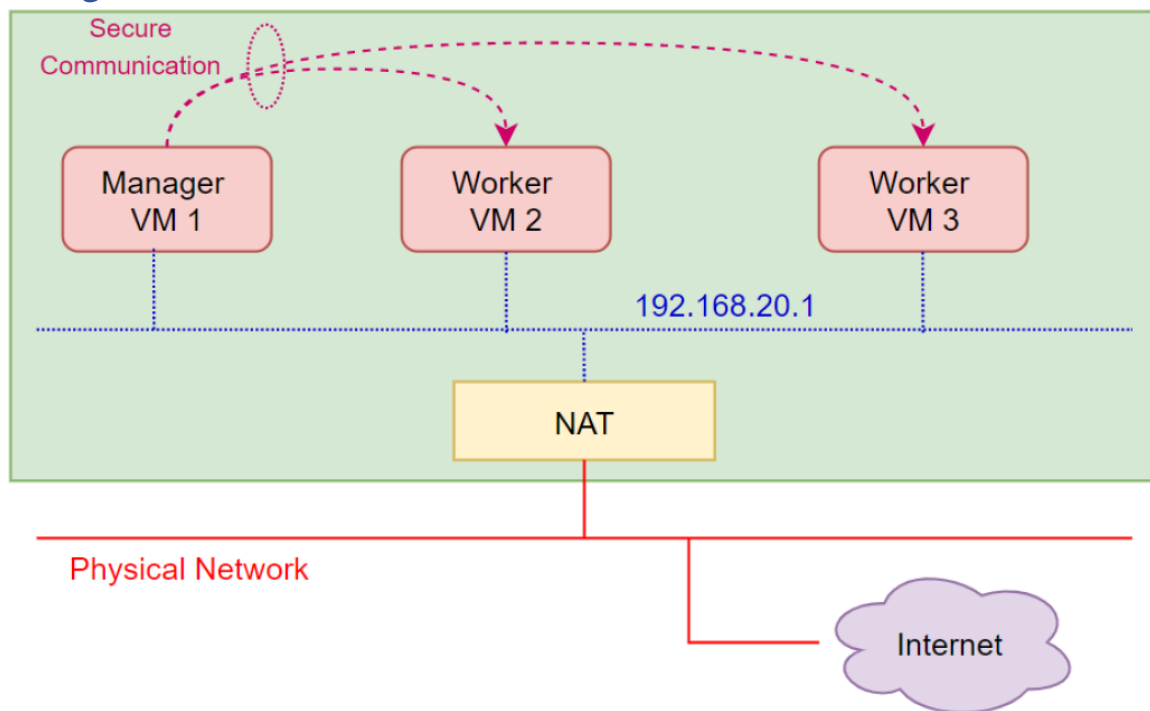


Figure 29

This section involves configuring each virtual machine to operate on a Nat network separate to my host machine as shown in Figure 29. Steps 1 through 12 depict the creation of a custom Nat network which leases IP addresses in the 192.168.20.0/24 range, therefore, all IP addresses connected to the custom Nat network will have addresses beginning with 192.168.20.X where X is a random value between 2 and 255. Steps 13 through 17 will be repeated on the worker 1 and worker 2 virtual machines.

Steps

Step 1: User left click on "File Alt+F (menu item)" in "Oracle VM VirtualBox Manager"

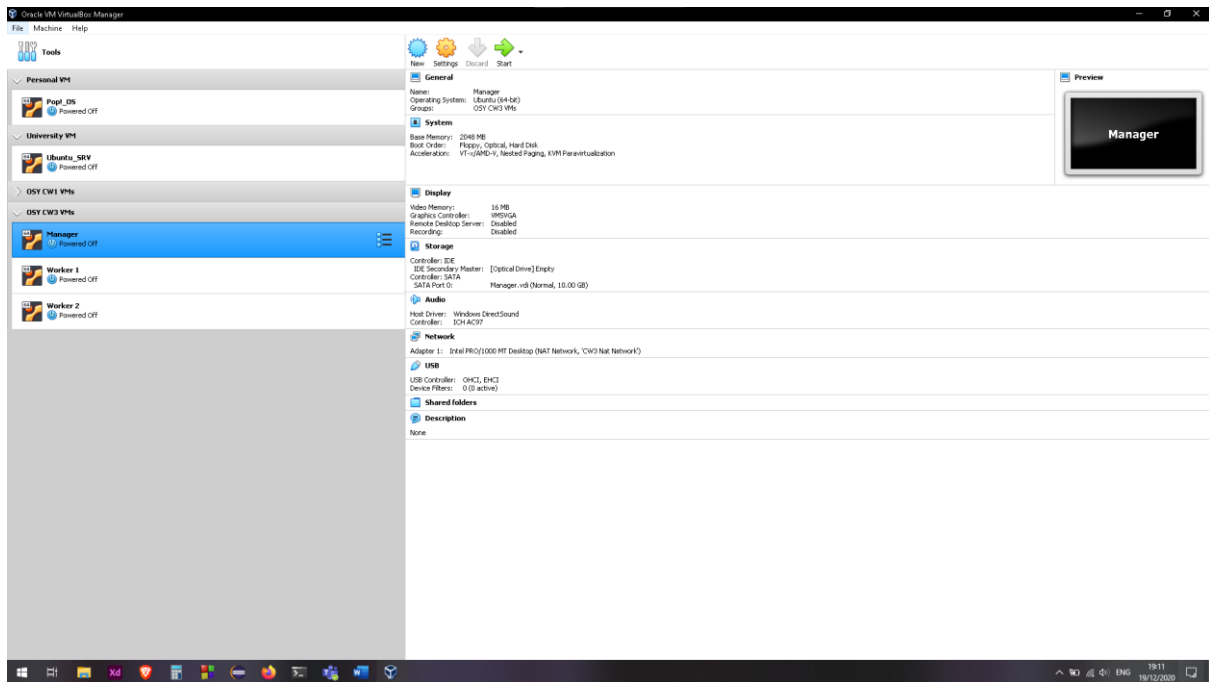


Figure 30

Step 2: User left click on "Preferences... Ctrl+G (menu item)" in "VirtualBox"



Figure 31

Step 3: User left click on " Network (list item)" in "VirtualBox - Preferences"

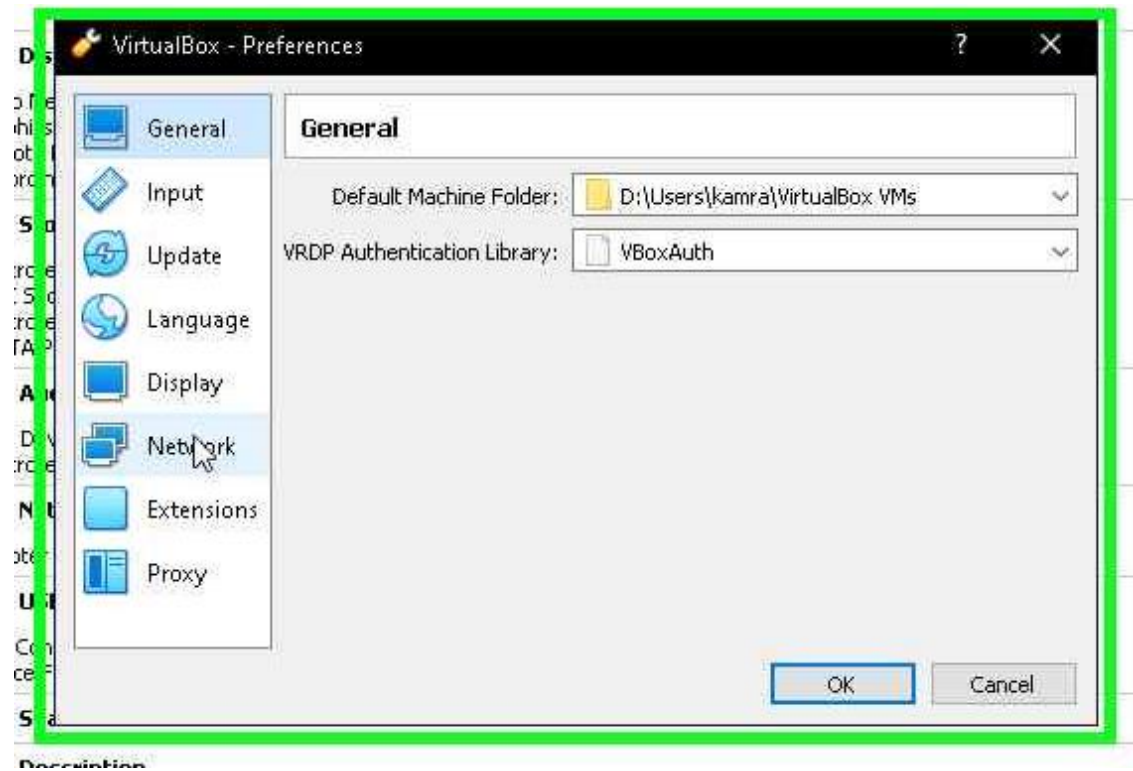


Figure 32

Step 4: User left click on "Add NAT Network (button)" in "VirtualBox - Preferences"

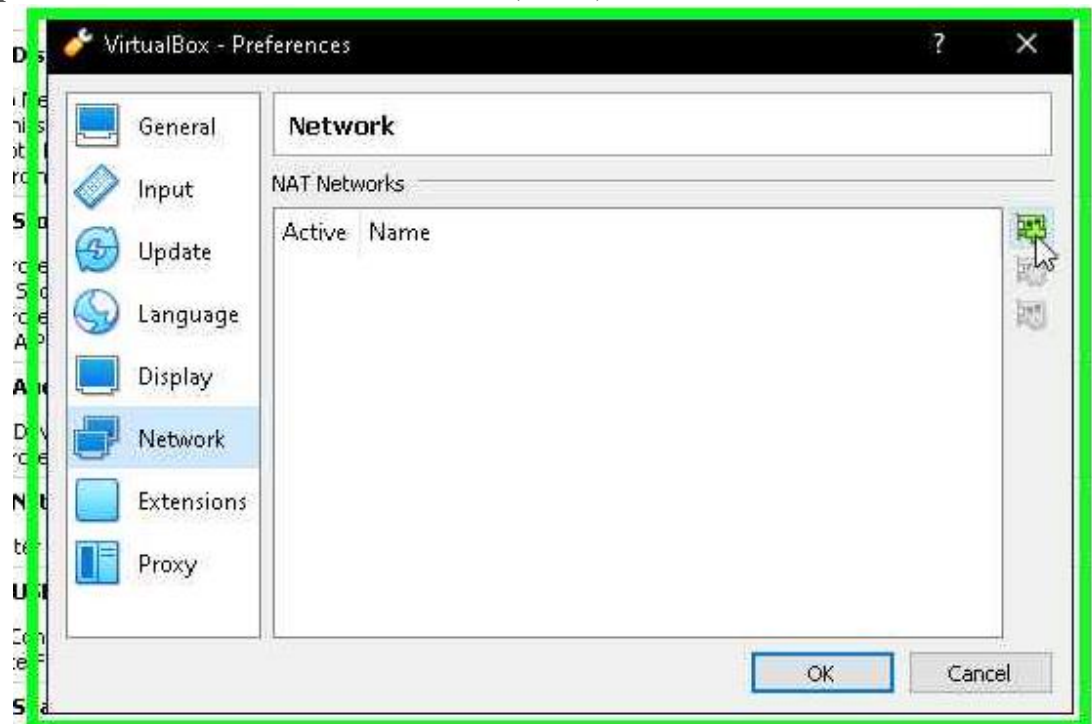


Figure 33

Step 5: User left double click on "NatNetwork, Active (list item)" in "VirtualBox - Preferences"

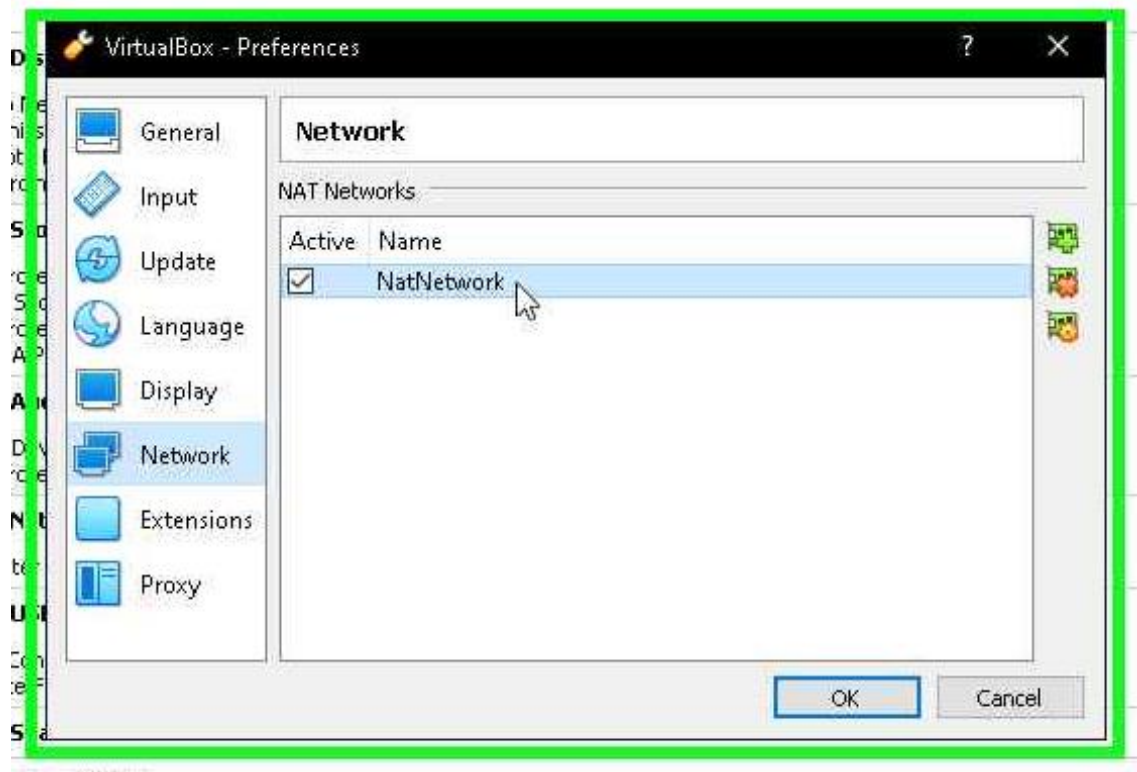


Figure 34

Step 6: User mouse drag start on "Network Name: Alt+N (edit)" in "NAT Network Details"

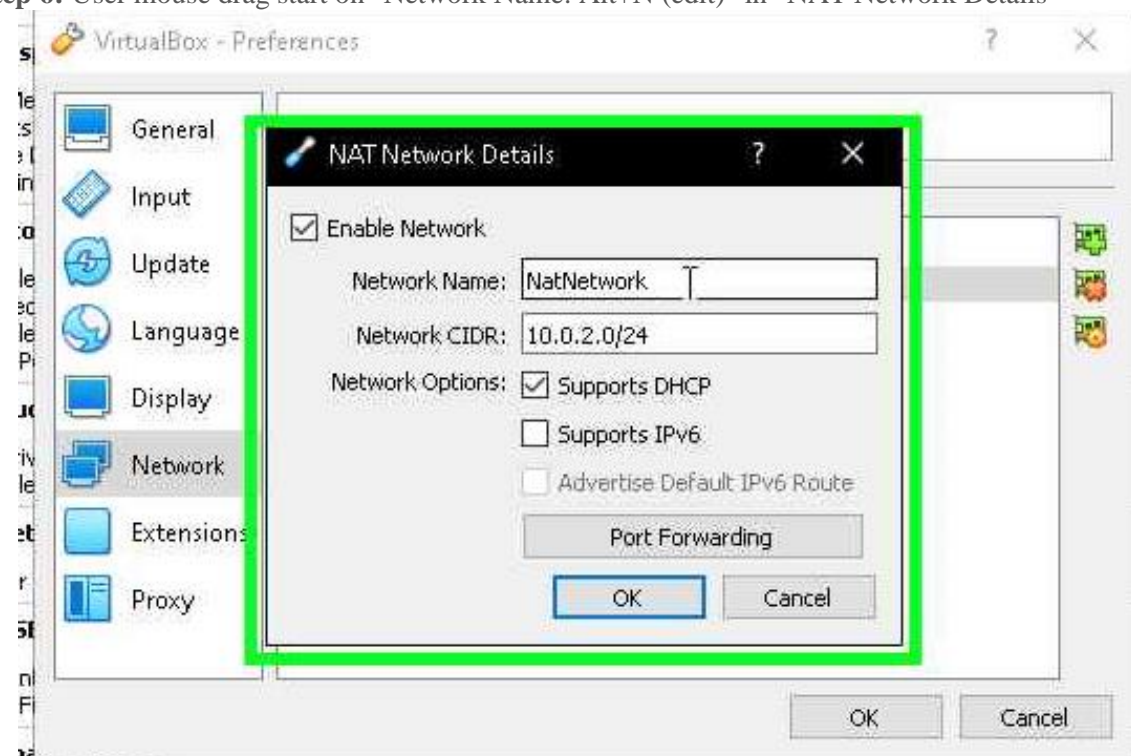


Figure 35

Step 7: User mouse drag end on "Network Name: (text)" in "NAT Network Details"

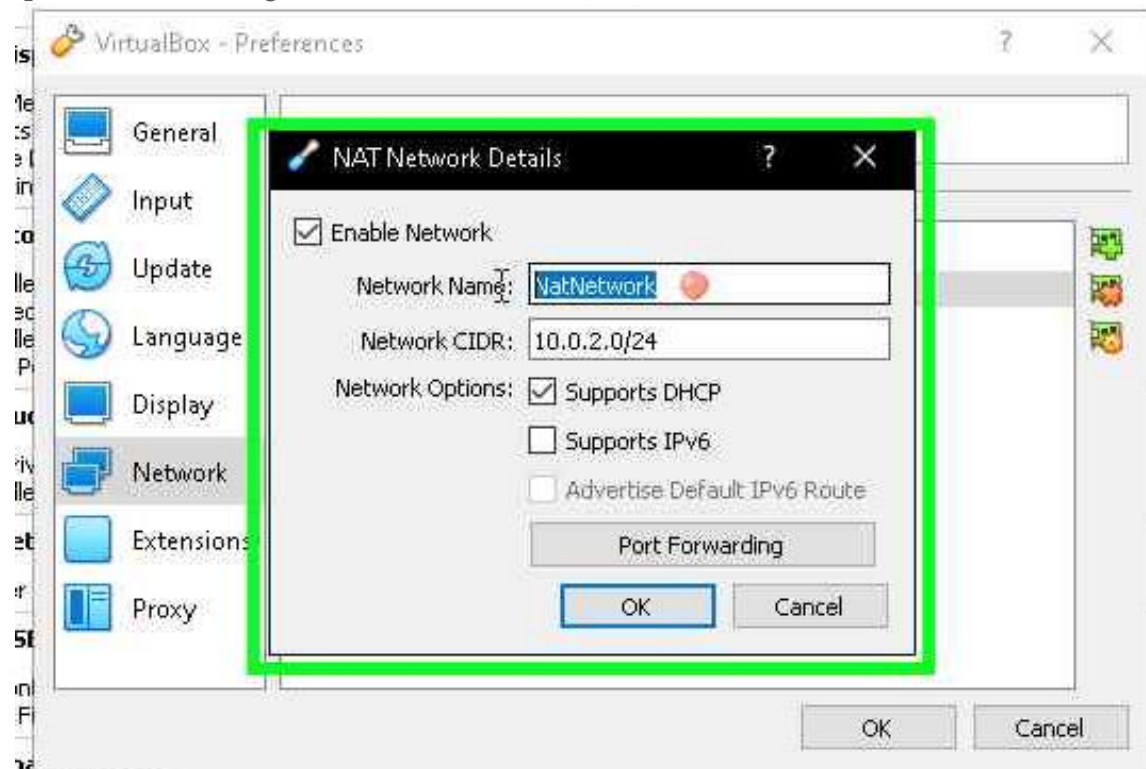


Figure 36

Step 8: User keyboard input on "NAT Network Details (window)" in "NAT Network Details" [...]

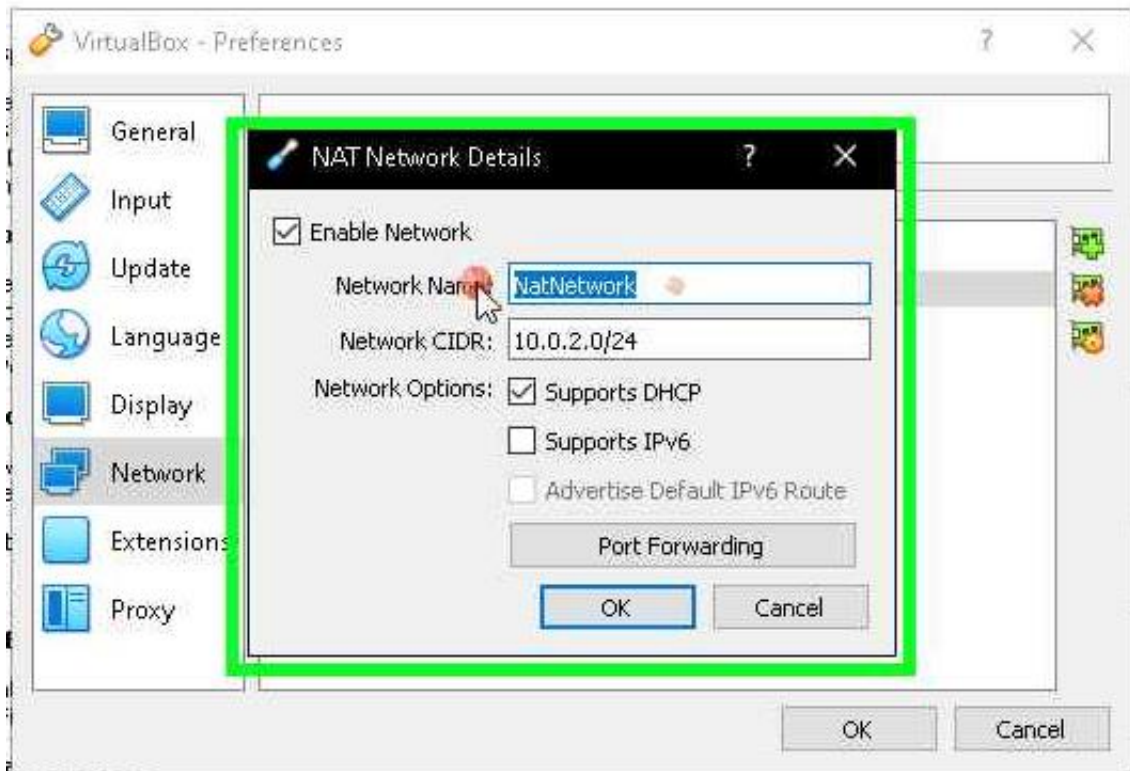


Figure 37

Step 9: User left click on "Network CIDR: Alt+C (edit)" in "NAT Network Details"

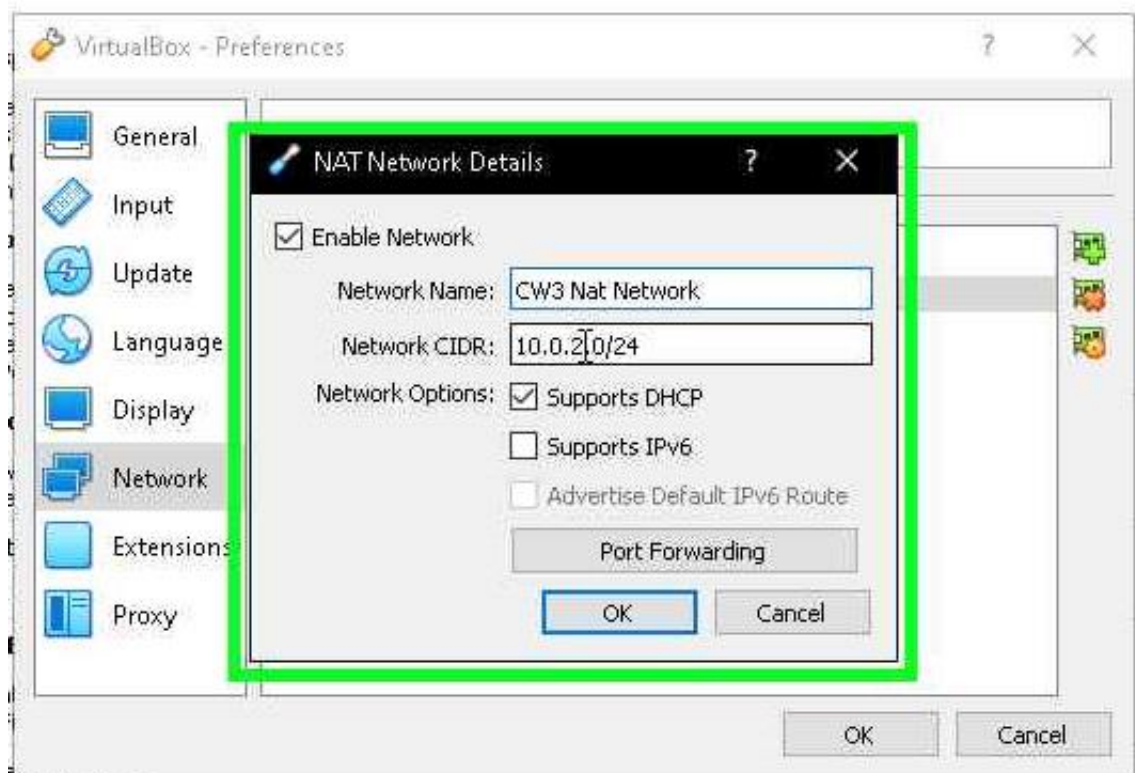


Figure 38

Step 10: User keyboard input on "NAT Network Details (window)" in "NAT Network Details"
[BACKSPACE BACKSPACE BACKSPACE BACKSPACE BACKSPACE ...]

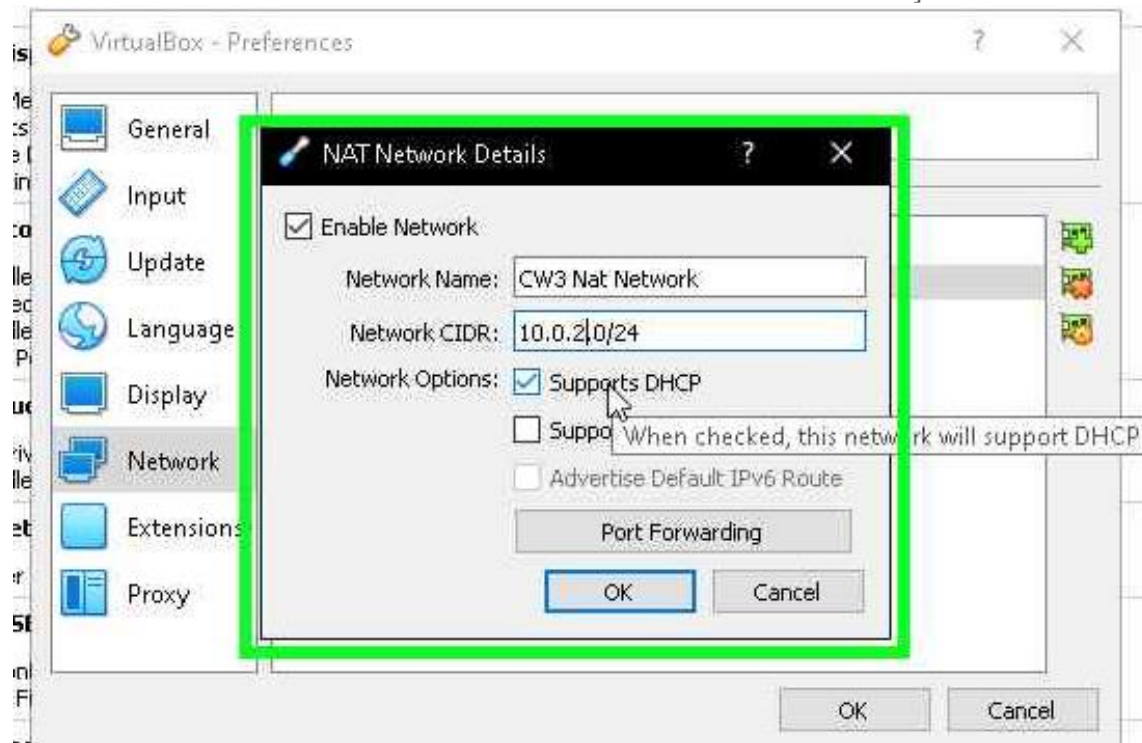


Figure 39

Step 11: User left click on "OK Enter (button)" in "NAT Network Details"

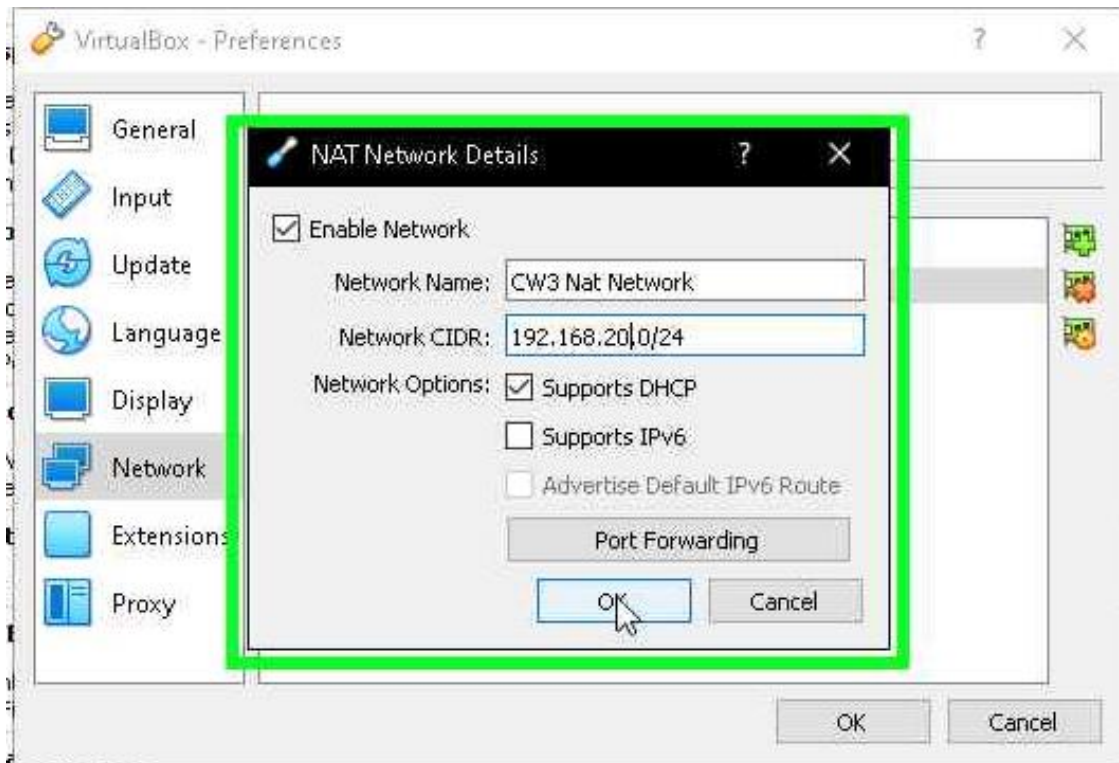


Figure 40

Step 12: User left click on "OK Enter (button)" in "VirtualBox - Preferences"

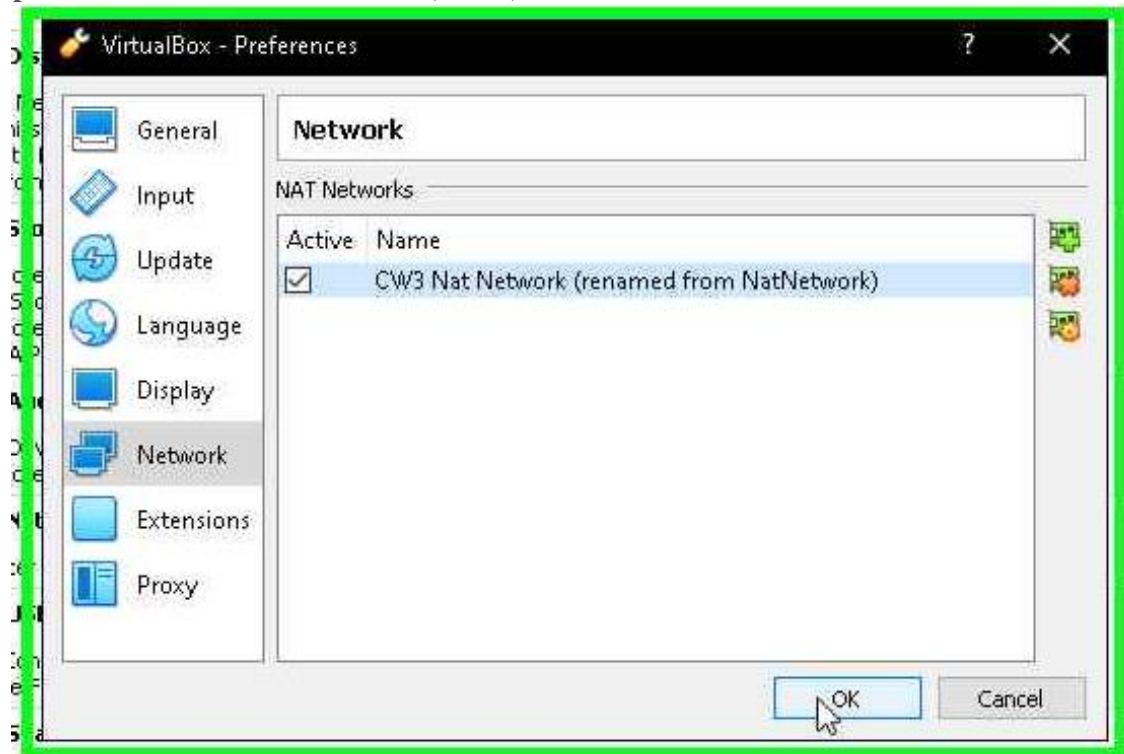


Figure 41

Step 13: User left click on "Oracle VM VirtualBox Manager (window)" in "Oracle VM VirtualBox Manager"

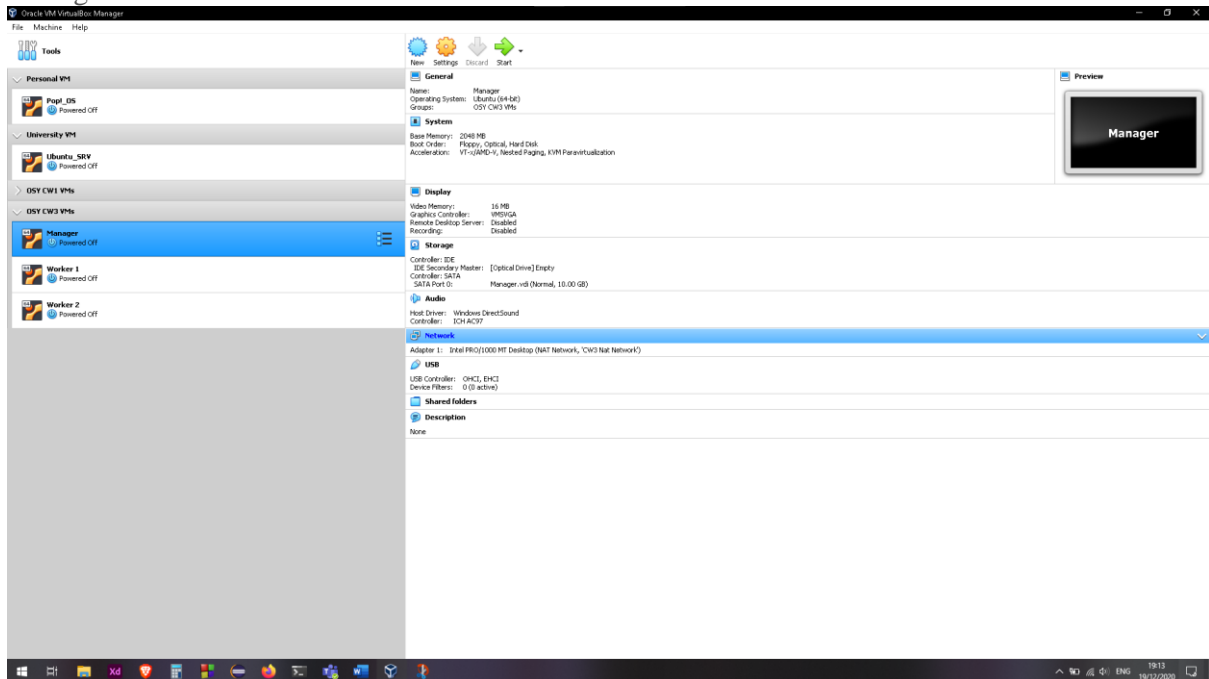


Figure 42

Step 14: User left click on " Down (combo box)" in "Manager - Settings"

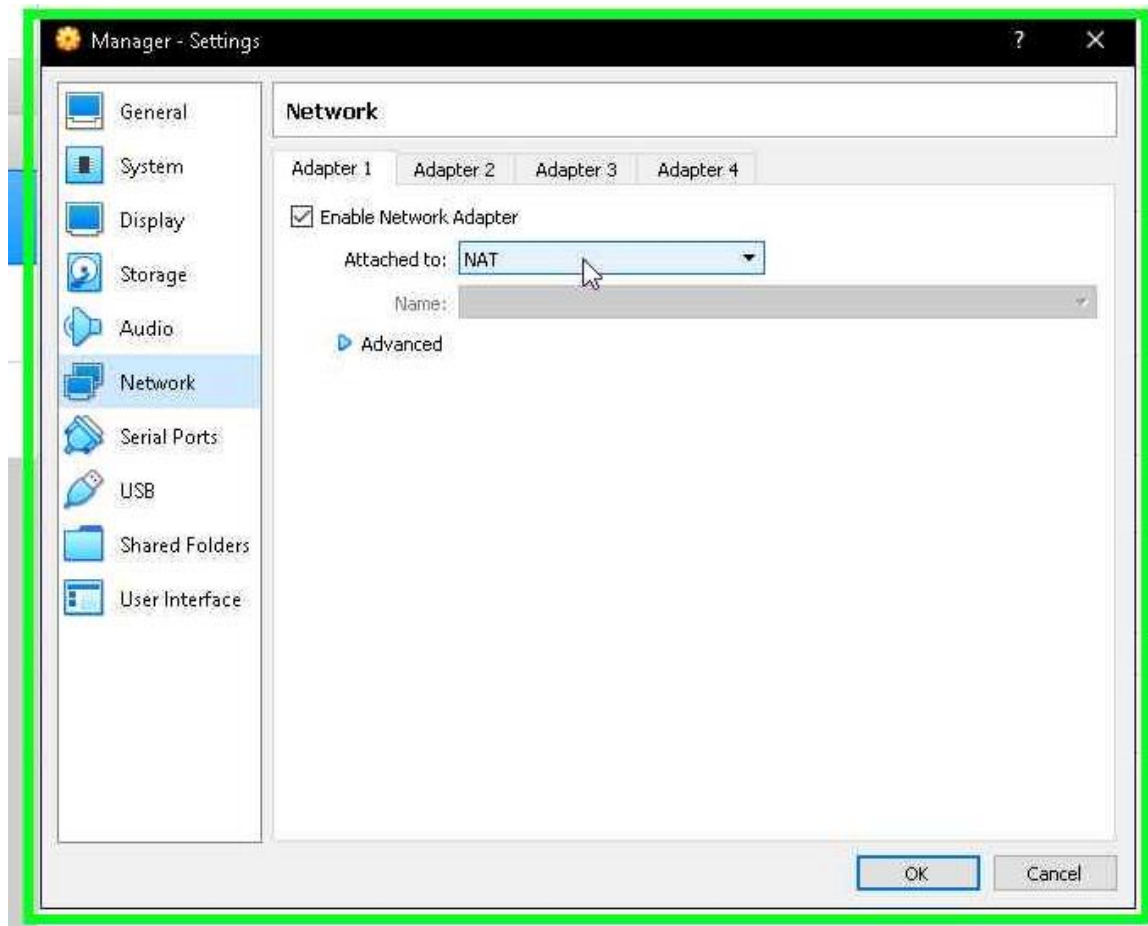


Figure 43

Step 15: User left click in "VirtualBox"

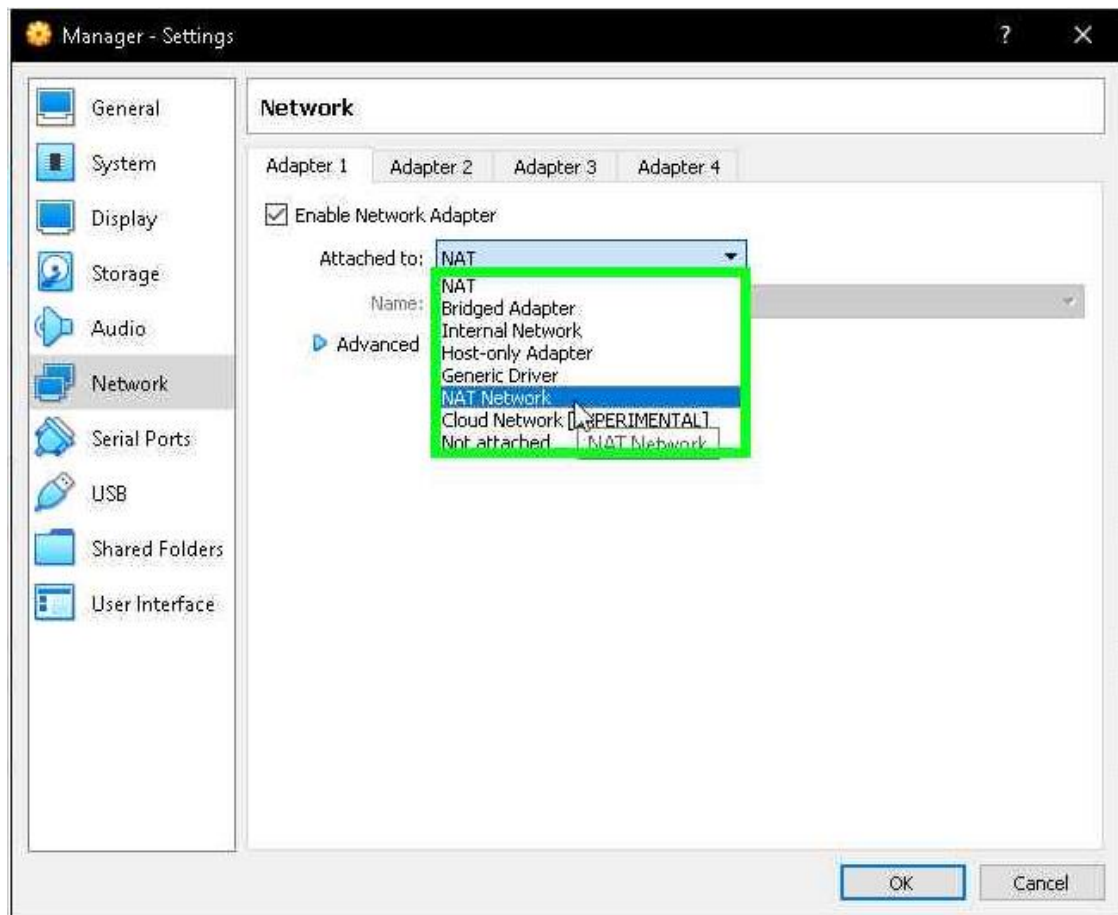


Figure 44

Step 16: User left click on "OK Enter (button)" in "Manager - Settings"

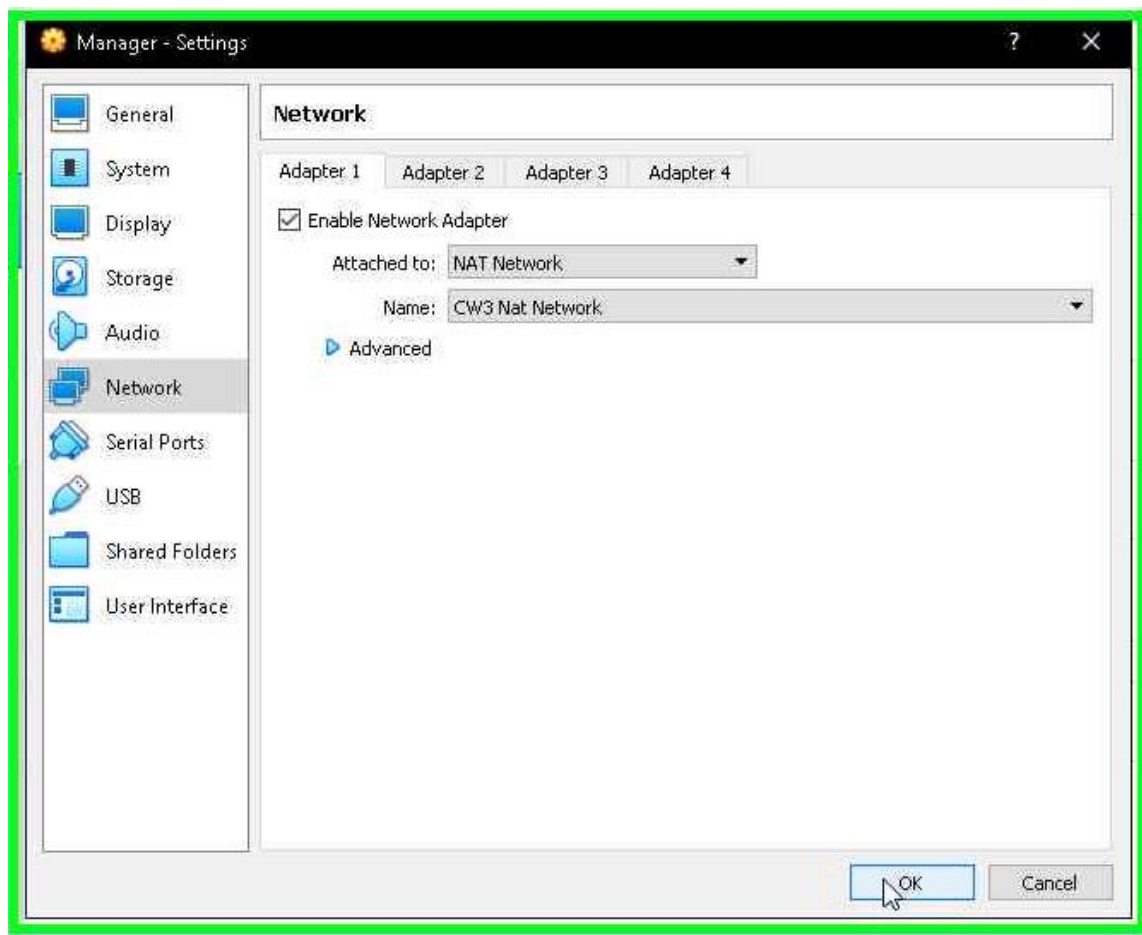


Figure 45

Programming

All source code can be found at (Bhatti, 2020) and in Appendix A – api.py.

```
manager@manager:~/PythonAPI$ sudo apt install python3-pip -y
```

Figure 46

Figure 46 is used to install the pip command for python, which will later be used to install any modules we need for the python API.

Through the Manager, a user automates processes on the Workers over a secure connection (i.e., using SSH).

To use SSH from the python program, the paramiko module must be installed, as dictated in (Ghosh, 2020). To install these modules, the following command must be run:

```
manager@manager:~/PythonAPI$ pip3 install paramiko
Collecting paramiko
```

Figure 47

```
manager@manager:~$ pip3 install scp
```

Figure 48

The SCP module was used to copy the script file from the manager machine to the worker machine inside of the python API file thus, the command in Figure 48 was used to install the SCP module.

api.py

```
os.system("clear")
# command to clear screen
```

Figure 49

Figure 49 references (PythonPool, 2020) to clear the screen in the Linux terminal using the *os* module.

```
host = input('Enter the IP of the remote machine: ')
port = 22
username = input("Enter the username for the remote machine: ")
password = getpass.getpass()
# variables to store host, port, username and password

try:
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, port, username, password)
    # connect to target machine via SSH
```

Figure 50

The host IP, username and password are taken as user input as shown in Figure 50 to make an SSH connection between the manager and worker virtual machines, however, the code used from (Ghosh, 2020) is encapsulated inside a *try* block to catch any errors outputted during program execution.


```

while True:
    print("Scripts in Current Directory: ")
    for line in os.popen("ls *.sh").readlines():
        print(line)
    # prints list of all script files from the current directory on manager vm

    command = input("> ")
    start_time = time.time()
    # store start time when command is executed

    if (command.lower() == "quit") or (command.lower() == "exit"):
        break
    # close program if command entered equals quit or exit
    else:
        file = scp.SCPClient(ssh.get_transport())
        values = command.split(" ")
        # splits the command on every space

        file.put(f"{values[0]}", ".")
        # copies script by name from manager to worker vm

        stdin, stdout, stderr = ssh.exec_command(f"sh ./{command}")
        # runs scripts from manager vm to worker vm

        if stdout:
            for line in stdout.readlines():
                print(line)
            # prints output of running script
        if stderr:
            for line in stderr.readlines():
                print(line)
            # prints errors while running the script

        print(f"Task took {time.time() - start_time} seconds to execute\n")
        # prints total time to execute each command/task

        stdin, stdout, stderr = ssh.exec_command(f"rm {values[0]}")
        # deletes copied script from worker vm

```

Figure 51

(Ghosh, 2020) has been used to execute commands from the manager virtual machine to the worker machine via the parameters passed in the `ssh.exec_command()` function. Lines 3 to 5 will output a list of files ending with `.sh` on separate lines. The user is then asked for user input to enter a shell script to execute which would be stored in the `command` variable.

(Robinson, 2012) was used to calculate the start and end time for executing each shell script where the start time is stored in the `start_time` variable after the user has entered the script to execute. When the script has finished executing, on line 34, the program outputs the total time to execute the shell script in seconds.

The `if` statement on line 11 compares the inputted command to `"quit"` or `"exit"` so that if a user types these as the command, the program will stop running by breaking the `while` loop. Otherwise, the program will interpret the command entered as the name of a shell script file and will attempt to execute the shell script as shown from line 14 onwards. As a `while True` loop has been used, the only way to exit the program is to enter `"quit"` or `"exit"` or by using a `KeyboardInterrupt`, for instance, pressing the Ctrl-C key simultaneously while the program is running.

Line 15 creates a `SCPClient` to copy files securely to the worker machine where line 19 copies the file to the worker machine using logic from (Bardin, 2020). Line 16 splits the command entered by each space so that the first item in the `values` list is the name of the script. This means that on line 22, the

script is executed with any command-line arguments passed when entering the script name. As the script has been copied from the manager to the worker machine, the second to last line removes the script from the worker machine so that the machine appears to be unaffected.

Line 25 checks if there is any data inside of *stdout*, if this evaluates to *True*, the *for* loop under the *if* statement will execute thus reading each line in the *stdout* variable and displaying the output of the shell script to the user. This is similar to line 29 which checks if there are any data inside the *stderr* variable where this evaluates to *True*, the *for* loop under the *if* statement will execute thus outputting any errors that would be encountered while executing the shell script.

```
except paramiko.ssh_exception.NoValidConnectionsError:
    print(f"Couldn't find machine with IP: {host}")
    # print error if can't find IP at specified host address
```

Figure 52

Figure 52 is executed by catching an exception if the IP address stored in the *host* variable cannot be found thus printing an error message.

```
except TimeoutError:
    print("Connection timed out, please recheck the connection details you provided")
    # print error if connection times out
```

Figure 53

Figure 53 is executed by catching an exception if the SSH connection to the IP address the user is trying to connect times out thus printing an error message.

```
except socket.gaierror:
    print(f"IP: {host} is not in the correct format")
    # print error if IP entered isn't in the correct format
```

Figure 54

Figure 54 is executed by catching an exception if the IP address stored in the *host* variable isn't in the correct format thus printing an error message.

```
except paramiko.ssh_exception.AuthenticationException:
    print(f"Authentication failed for IP: {host}")
    # print error if username or password entered is incorrect
```

Figure 55

Figure 55 is executed by catching an exception if the username or password entered when trying to connect to the machine are incorrect thus printing an error message.

```
except FileNotFoundError:
    print("Script doesn't exist, please enter a valid script")
    # print error if script entered isn't found
```

Figure 56

Figure 56 is executed by catching an exception if the script entered doesn't exist thus printing an error message.

Challenges

```
while True:
    print("Scripts in Current Directory: ")
    for line in os.popen("ls *.sh").readlines():
        print(line)
    # prints list of all script files from the current directory on manager vm

    command = input("> ")
    start_time = time.time()
    # store start time when command is executed

    if (command.lower() == "quit") or (command.lower() == "exit"):
        break
    # close program if command entered equals quit or exit
    else:
        script = os.popen(f"cat {command}").read()
        stdin, stdout, stderr = ssh.exec_command(script)
        # runs scripts from manager vm to worker vm

        if stdout:
            for line in stdout.readlines():
                print(line)
            # prints output of running script
        if stderr:
            for line in stderr.readlines():
                print(line)
            # prints errors while running the script

        print(f"Task took {time.time() - start_time} seconds to execute\n")
        # prints total time to execute each command/task
```

Figure 57

Initially, the while loop for the program was structured as seen in Figure 57 where Lines 15 and 16 read the script on the manager virtual machine and execute the script on the worker machine using (Rockikz, 2020) to execute the script remotely over SSH. Although this would execute the script on the worker virtual machine, any command-line arguments passed with the script wouldn't be executed on the script as the script would be executed line by line in the worker virtual machines' terminal thus command line arguments wouldn't be perceived as arguments for the shell script but as other scripts to execute.

To remedy this, methods from (Bardin, 2020) has been used to copy the script file from the manager machine to the worker machine securely so that users can execute the script with additional command-line arguments as seen in Figure 51.

Test Results

Machine Name	IP address	Username	Password
manager	192.168.20.4	manager	password1
worker1	192.168.20.5	worker	password2
worker2	192.168.20.6	worker	password2

Table 1

The shell scripts I will be using to demonstrate the functionality of the python API can be found at (Bhatti, 2020).

```

Worker 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8d:54:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.5/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 496sec preferred_lft 496sec
    inet6 fe80::a00:27ff:fe8d:5414/64 scope link
        valid_lft forever preferred_lft forever
worker@worker1:~$

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192
Enter the username for the remote machine: worker
Password:
Connection timed out, please recheck the connection details you provided
manager@manager:~/PythonAPI$ _
  
```

Figure 58

Figure 58 depicts the triggering condition for Figure 53 to be executed where an IP address in an invalid format has been given.

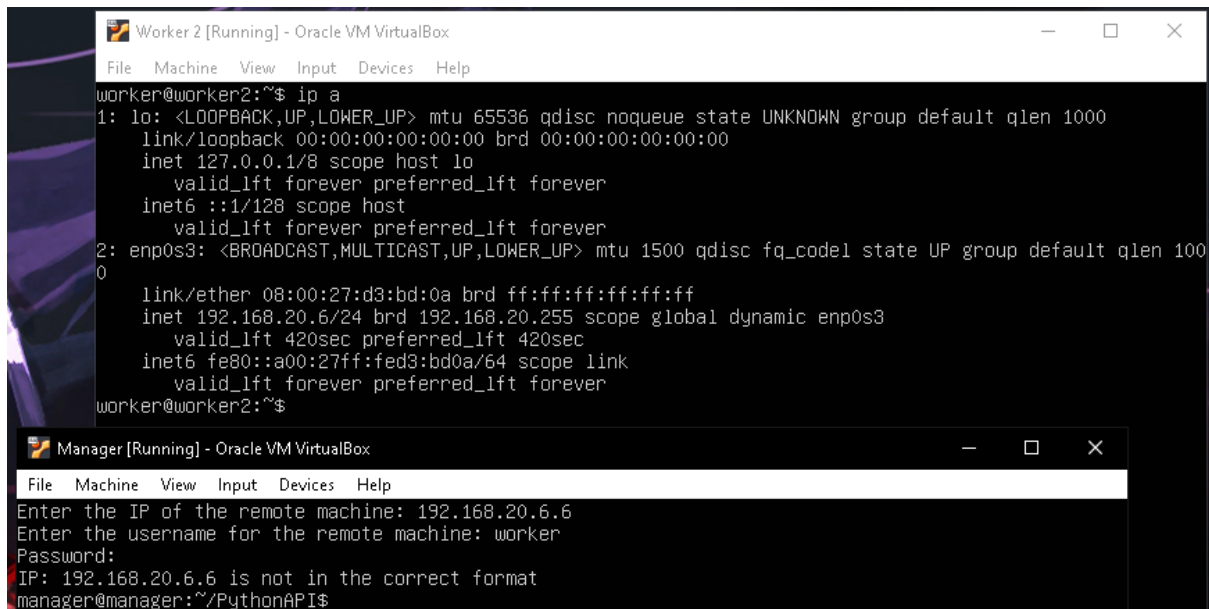
```

Worker 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8d:54:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.5/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 496sec preferred_lft 496sec
    inet6 fe80::a00:27ff:fe8d:5414/64 scope link
        valid_lft forever preferred_lft forever
worker@worker1:~$ _

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.7
Enter the username for the remote machine: worker
Password:
Couldn't find machine with IP: 192.168.20.7
manager@manager:~/PythonAPI$ _
  
```

Figure 59

Figure 59 depicts the triggering conditions for Figure 52 to be executed where the IP address is of the correct format but the machine with that IP address isn't online or the machine with that IP address doesn't exist on the network.

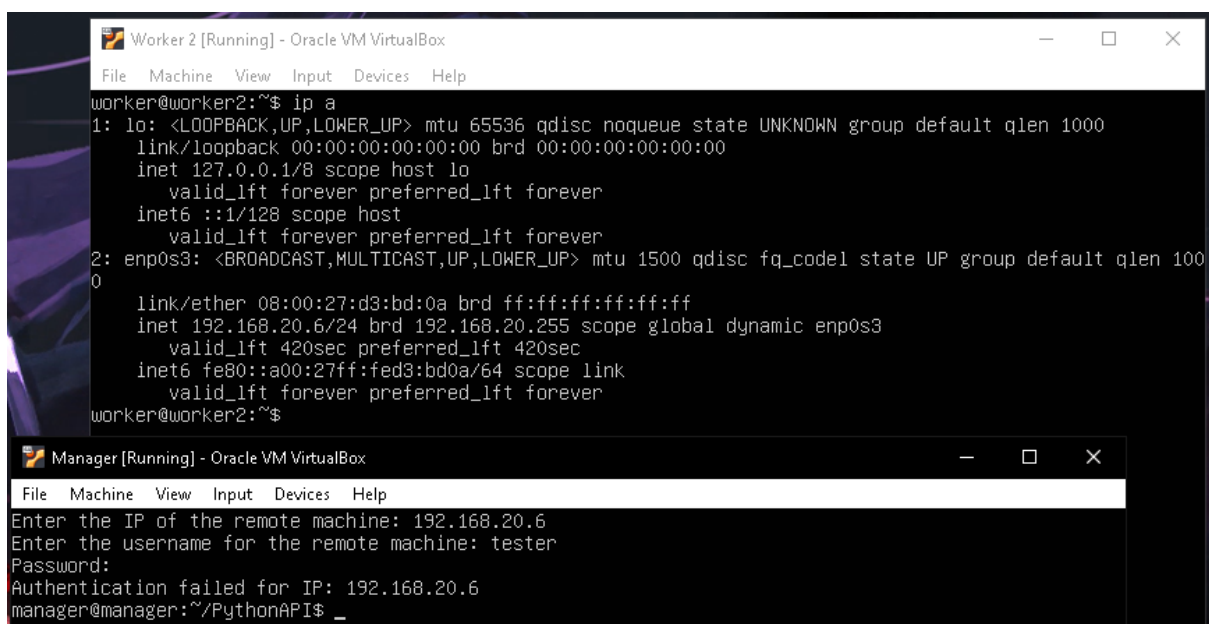


```
Worker 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d3:bd:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.6/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 420sec preferred_lft 420sec
    inet6 fe80::a00:27ff:fed3:bd0a/64 scope link
        valid_lft forever preferred_lft forever
worker@worker2:~$

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.6
Enter the username for the remote machine: worker
Password:
IP: 192.168.20.6.6 is not in the correct format
manager@manager:~/PythonAPI$
```

Figure 60

Figure 60 depicts the triggering conditions for Figure 54 to be executed where the IP address entered isn't in the correct format.



```
Worker 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d3:bd:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.6/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 420sec preferred_lft 420sec
    inet6 fe80::a00:27ff:fed3:bd0a/64 scope link
        valid_lft forever preferred_lft forever
worker@worker2:~$

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.6
Enter the username for the remote machine: tester
Password:
Authentication failed for IP: 192.168.20.6
manager@manager:~/PythonAPI$
```

Figure 61

Figure 61 depicts the triggering conditions for Figure 55 to execute where either the username or password entered is incorrect.

```
Worker 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d3:bd:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.6/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 568sec preferred_lft 568sec
    inet6 fe80::a00:27ff:fed3:bd0a/64 scope link
        valid_lft forever preferred_lft forever
worker@worker2:~$ _

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.6
Enter the username for the remote machine: worker
Password:
Scripts in Current Directory:
task1.sh

task2.sh

task3.sh

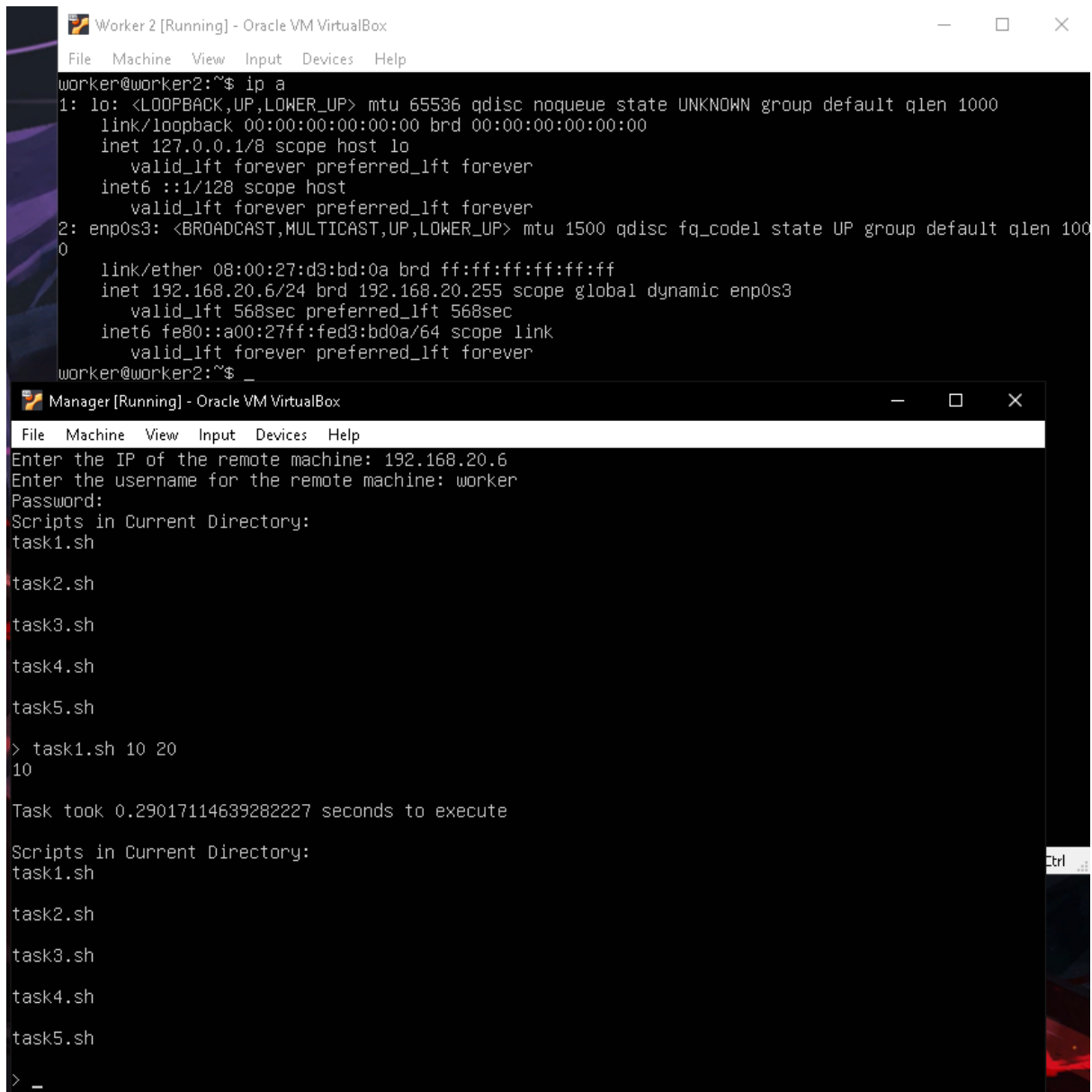
task4.sh

task5.sh

> _
```

Figure 62

If the host, username and password are all correct, the user is asked to enter the name of a script as shown in Figure 62.



```
Worker 2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d3:bd:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.6/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 568sec preferred_lft 568sec
    inet6 fe80::a00:27ff:fed3:bd0a/64 scope link
        valid_lft forever preferred_lft forever
worker@worker2:~$ _

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.6
Enter the username for the remote machine: worker
Password:
Scripts in Current Directory:
task1.sh

task2.sh

task3.sh

task4.sh

task5.sh
> task1.sh 10 20
10
Task took 0.29017114639282227 seconds to execute
Scripts in Current Directory:
task1.sh

task2.sh

task3.sh

task4.sh

task5.sh
> _
```

Figure 63

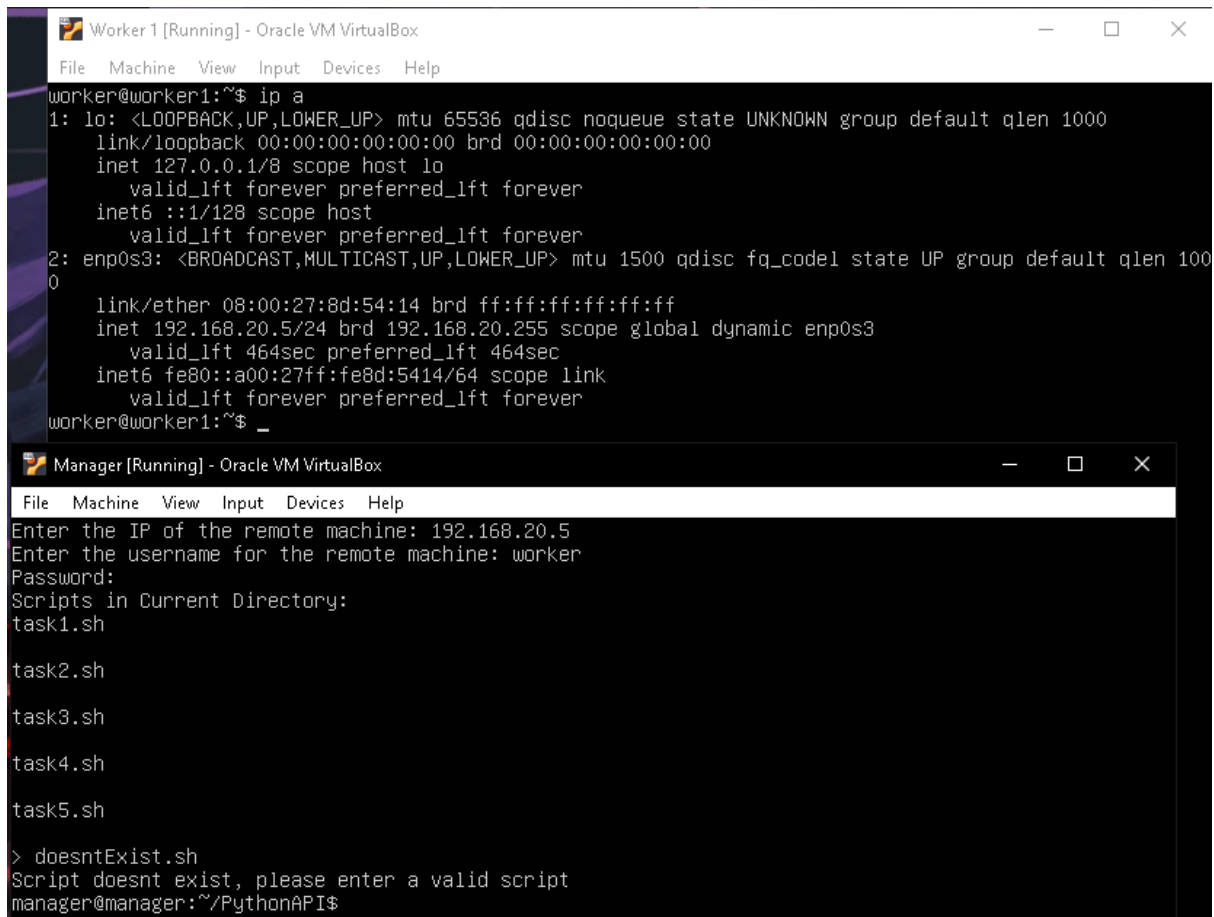
Entering *task1.sh 10 20* executes the script on the worker2 virtual machine from the manager virtual machine.

```
Worker 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8d:54:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.5/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 464sec preferred_lft 464sec
    inet6 fe80::a00:27ff:fe8d:5414/64 scope link
        valid_lft forever preferred_lft forever
worker@worker1:~$

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.5
Enter the username for the remote machine: worker
Password:
Scripts in Current Directory:
task1.sh
task2.sh
task3.sh
task4.sh
task5.sh
> task3.sh 20
Binary 10100
Octal 24
Hexadecimal 14
Task took 0.12398743629455566 seconds to execute
Scripts in Current Directory:
task1.sh
task2.sh
task3.sh
task4.sh
task5.sh
> _
```

Figure 64

Entering *task3.sh* executes the script on Worker1's virtual machine as shown in Figure 64 which will convert a number to a different base.



```
Worker 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
worker@worker1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8d:54:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.5/24 brd 192.168.20.255 scope global dynamic enp0s3
        valid_lft 464sec preferred_lft 464sec
    inet6 fe80::a00:27ff:fe8d:5414/64 scope link
        valid_lft forever preferred_lft forever
worker@worker1:~$ _

Manager [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Enter the IP of the remote machine: 192.168.20.5
Enter the username for the remote machine: worker
Password:
Scripts in Current Directory:
task1.sh

task2.sh

task3.sh

task4.sh

task5.sh

> doesntExist.sh
Script doesnt exist, please enter a valid script
manager@manager:~/PythonAPI$
```

Figure 65

If the user inputs a command that isn't recognised by the program such as a shell script which isn't listed, the program outputs an error as seen in Figure 65 from triggering the exception from Figure 56.

Appraisal

Overall, the system conforms to the specification as the python API can execute scripts from the manager machine to the worker machine over a secure connection which is facilitated by the SSH protocol. Furthermore, the IP address range of each virtual machine connected to the Nat network is from 192.168.20.1/24.

Although the target worker is identified by its IP address, I believe that the program would work better if the user selects the machine to connect to from a list of addresses that are on the network, for example, if the python program were to display a list of machines on the network by their IP address or hostname by utilising the *Nmap* tool, a user could select one of the machines from the displayed list.

To improve the python API, I could have enumerated the list of scripts so that a user would only have to enter the number associated to one of the scripts but this would require multiple input stages which I believe would have made the automated nature of the API redundant. Thus, I have also neglected to implement an interactive shell for if the script executed requires additional input as I believed it wouldn't conform to the traditional nature of how an automated API would execute. A consequence of this is that scripts which require elevated privileges, such as entering a password for the root user, will not execute correctly. Furthermore, the current implementation of the API will only execute shell scripts which are in the same directory as the API file.

Bibliography

Bardin, J., 2020. *scp 0.13.3*. [Online]

Available at: <https://pypi.org/project/scp/#history>
[Accessed 19 December 2020].

Bhatti, K., 2020. *PythonAPI*. [Online]

Available at: <https://github.com/k5924/PythonAPI/>
[Accessed 17 December 2020].

Bhatti, K., 2020. *ShellScripts*. [Online]

Available at: <https://github.com/k5924/ShellScripts>
[Accessed 22 November 2020].

Canonical Ltd., 2020. *Get Ubuntu server*. [Online]

Available at: <https://ubuntu.com/download/server>
[Accessed 17 December 2020].

Ghosh, S., 2020. *Python Basics*. [Online]

Available at:

https://github.com/rishiCSE17/py_Maths/blob/master/System%20Automation%20API%20using%20Python%20.ipynb

[Accessed 18 December 2020].

Oracle, 2020. *Welcome to VirtualBox.org!*. [Online]

Available at: <https://www.virtualbox.org/>
[Accessed 17 December 2020].

PythonPool, 2020. *How to Clear Python Shell in the Most Effective Way*. [Online]

Available at: <https://www.pythonpool.com/how-to-clear-python-shell/>
[Accessed 18 December 2020].

Robinson, D., 2012. *How long does my Python application take to run?*. [Online]

Available at: <https://stackoverflow.com/questions/12444004/how-long-does-my-python-application-take-to-run>

[Accessed 18 December 2020].

Rockikz, A., 2020. *How to Execute Shell Commands in a Remote Machine in Python*. [Online]

Available at: <https://www.thepythoncode.com/article/executing-bash-commands-remotely-in-python>
[Accessed 18 December 2020].

Score_Under, 2015. *How can I fix /dev/pts after mounting it?*. [Online]

Available at: <https://unix.stackexchange.com/questions/214526/how-can-i-fix-dev-pts-after-mounting-it>

[Accessed 17 December 2020].

Appendix A – api.py

```
import paramiko

import getpass

import socket

import os

import time

import scp


os.system("clear")

# command to clear screen


host = input('Enter the IP of the remote machine: ')

port = 22

username = input("Enter the username for the remote machine: ")

password = getpass.getpass()

# variables to store host, port, username and password


try:

    ssh = paramiko.SSHClient()

    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    ssh.connect(host, port, username, password)

    # connect to target machine via SSH


while True:

    print("Scripts in Current Directory: ")

    for line in os.popen("ls *.sh").readlines():

        print(line)

    # prints list of all script files from the current directory on manager vm


    command = input("> ")

    start_time = time.time()

    # store start time when command is executed
```

```

if (command.lower() == "quit") or (command.lower() == "exit"):
    break

    # close program if command entered equals quit or exit
else:
    file = scp.SCPClient(ssh.get_transport())
    values = command.split(" ")
    # splits the command on every space

    file.put(f"{values[0]}", ".")
    # copies script by name from manager to worker vm

    stdin, stdout, stderr = ssh.exec_command(f"sh ./ {command}")
    # runs scripts from manager vm to worker vm

    if stdout:
        for line in stdout.readlines():
            print(line)
            # prints output of running script
    if stderr:
        for line in stderr.readlines():
            print(line)
            # prints errors while running the script

    print(f"Task took {time.time() - start_time} seconds to execute\n")
    # prints total time to execute each command/task

    stdin, stdout, stderr = ssh.exec_command(f"rm {values[0]}")
    # deletes copied script from worker vm

except paramiko.ssh_exception.NoValidConnectionsError:
    print(f"Couldn't find machine with IP: {host}")

```

```
# print error if can't find IP at specified host address

except TimeoutError:

    print("Connection timed out, please recheck the connection details you provided")

    # print error if connection times out


except socket.gaierror:

    print(f"IP: {host} is not in the correct format")

    # print error if IP entered isnt in the correct format


except paramiko.ssh_exception.AuthenticationException:

    print(f"Authentication failed for IP: {host}")

    # print error if username or password entered is incorrect


except FileNotFoundError:

    print("Script doesnt exist, please enter a valid script")

    # print error is script entered isnt found
```