Debian 12 on Windows:

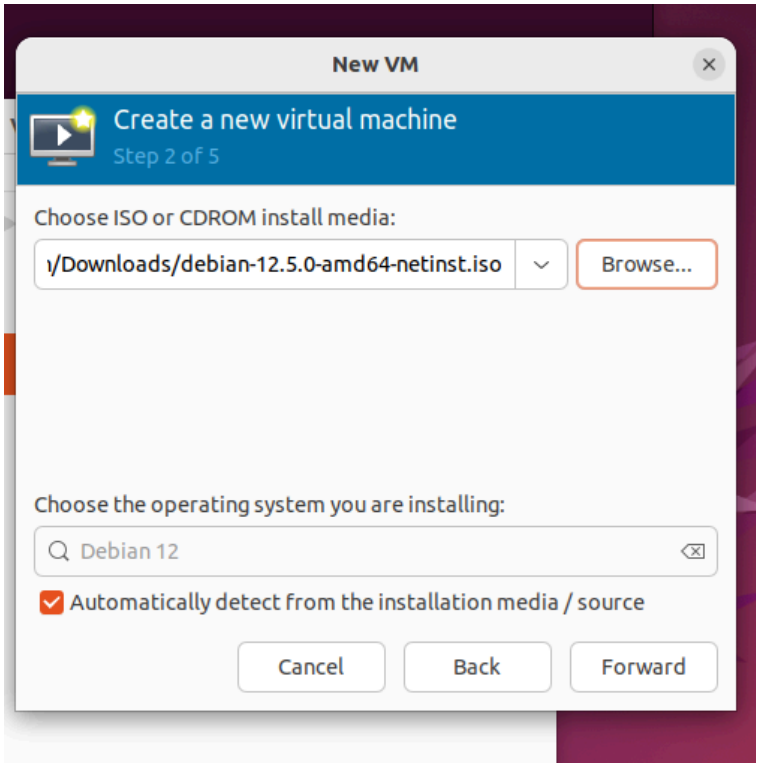
As far as I know, you can't put the KVM/QEMU virt-manager on windows without nesting it into an Ubuntu virtual machine, so this is the first step. This was pretty easy with VMware. Once you have your Ubuntu up and running, open a terminal login as root and then enter these commands:

1. sudo apt upgrade
2. sudo apt install cpu-checker
3. kvm-ok
4. sudo apt install qemu-kvm virt-manager libvirt-daemon-system
5. virtinst libvirt-clients bridge-utils
6. sudo apt install qemu-kvm virt-manager libvirt-daemon-system virtinst libvirt-clients bridge-utils
7. reboot
8. Kvm-ok

One line/enter button press per number, rebooting in between lines 4-8 may help this process, but definitely reboot at #7. Once this is done, hopefully when you do #8 it will say it does support it, but if not it should still work just a bit slow.

Now it's time to install the debian iso and create a virtual machine inside your virtual machine. Vmception.

Make a new vm and click forward in the first screen, select the debian iso and make sure it says debian 12 as the OS.



Take the defaults for the rest. During the install process make sure you add the ssh and web server and probably not a gui since mine never worked if I tried to install one, I don't have screenshots for this.

Once it's running we are going to set up more stuff on it like sudo, ssh, serial port, and Logical Volume Manager(LVM).

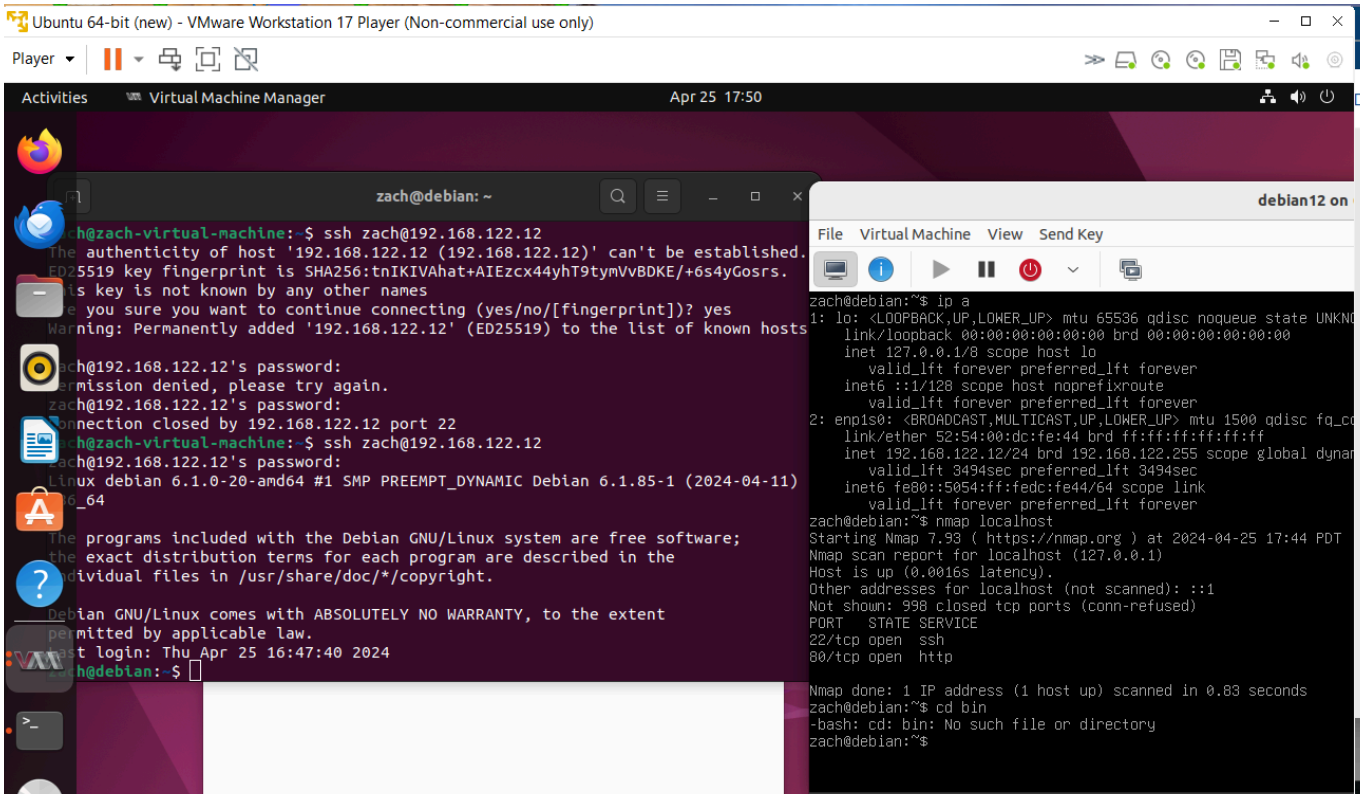
Install sudo and add user to the group:

1. apt install sudo lvm
2. usermod -aG sudo zach

You'll have to do these as root and of course replace zach with your user.

Then install ssh with this: apt install openssh-server

And try to ssh in. I did it from the Ubuntu vm:



It would not work from the windows command prompt on the host machine, insert windows slander here.

Now if you have a serial port on the hardware you are doing this with you will have to change the settings in the Ubuntu and add the serial port, then go back the debian terminal and run these commands:

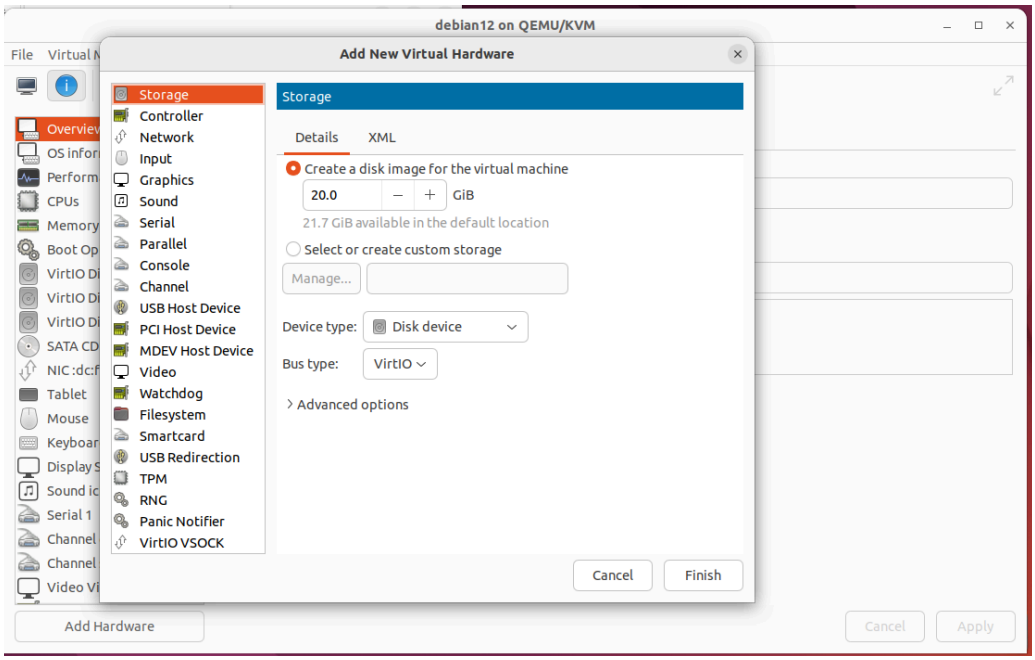
systemctl enable serial-getty@S0.service

systemctl start serial-getty@S0.service

systemctl status serial-getty@S0.service

It does not work for me because I don't have a serial port on this laptop.

Next we are going to set up an LVM which means we first have to add, hardware, make a physical group and add it to a volume group which we can then put in our Logical Volume Manager, and then mount it.



To add the hardware, click add hardware in the bottom left, make it whatever size, and finish. Then reboot so it will be created.

Now we add the drives, naming starts with vdb, vdc, and so on. Login as root then follow this:

1. fdisk -l
2. fdisk -l | grep vd
3. fdisk /dev/vdb
4. #currently at the fdisk commands
    a. n
    b. p
5. ENTER three times

6. At the fdisk commands
    a. p (lists out partition name)
7. t
8. L
9. 8e
10.
11. (back at the fdisk commands) w

Now do it again with fdisk /dev/vdc

Now we create the physical volume:

1. pvcreate /dev/vdb1 /dev/vdc1
2. pvs

Create volume group and check:

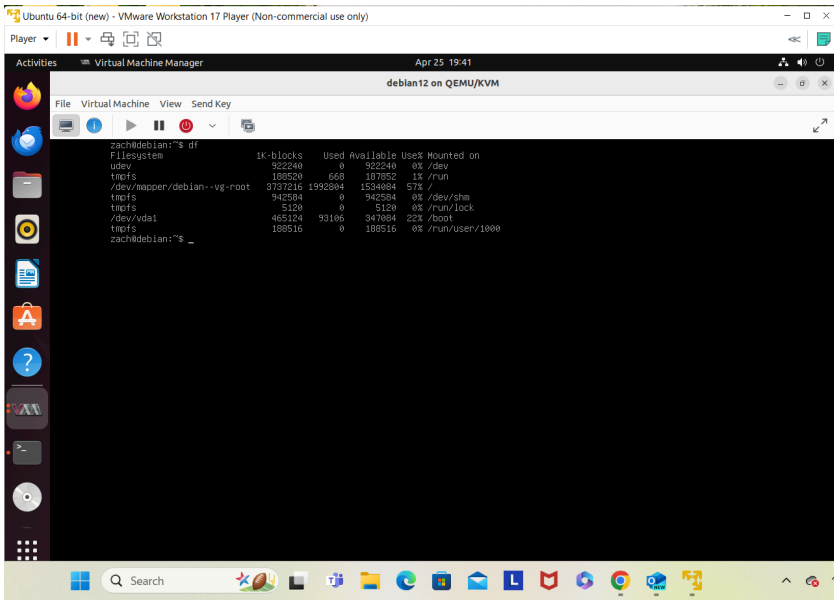
1. vgcreate deblvm /dev/vdb1 /dev/vdc1
2. vgs
3. Vgscan

Create LVM:

1. lvcreate –name data –size 3G deblvm *change this it wrong?*
2. Lvs
3. mkfs.ext4 /dev/deblvm/data
4. cd /mnt
5. mkdir data
6. mount /dev/deblvm/data /mnt/data
7. df

8. vi(or your editor of choice) /etc/fstab
9. Type in: /dev/mapper/deblvm-data /mnt/data ext4 defaults 0 2
10. df

If everything was correct, you should see something like this:



Next I am going to make a bin and an inclass directory (not /bin) and then run a cool script from home directory.

1. Mkdir bin
2. Wget https://phillipsd.com/210/sp24_mkstuff.txt

I am going to paste the entire script here as well in case my future self needs it and that link no longer works:

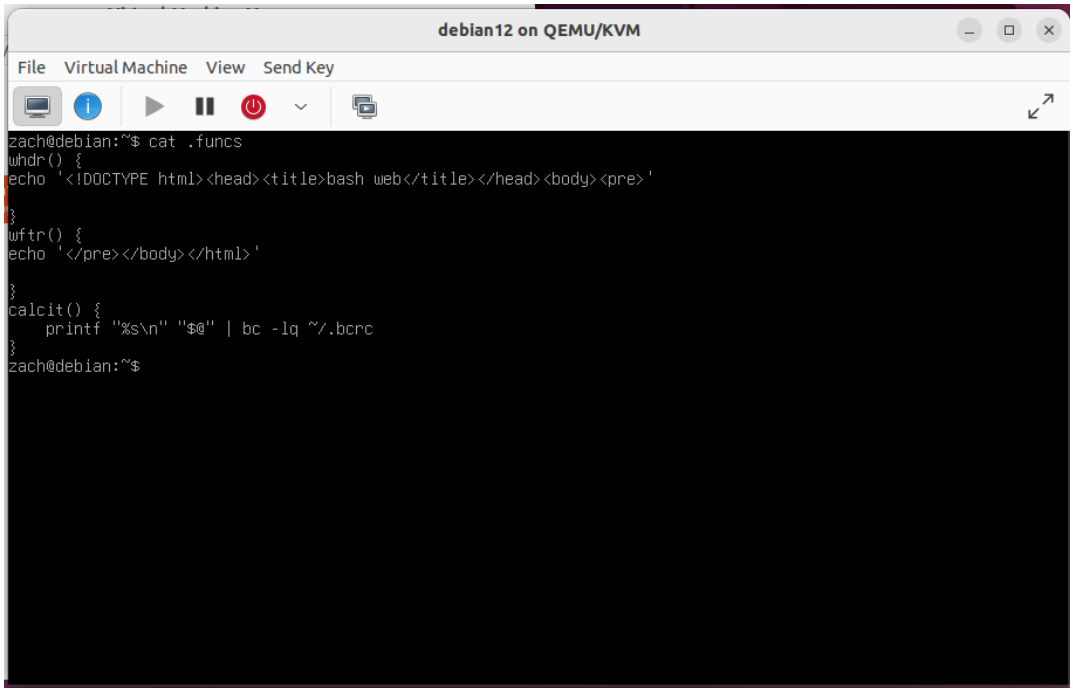
```
#!/bin/bash
# mkstuff
# create .funcs and .bash_aliases
#
echo "Did you create you $HOME/bin dir? "
```

```
read X

cat - > ~/.bash_aliases << "EOF"
#.bash_aliases
export PATH="$HOME/bin:$HOME/inclass:$PATH"
alias h=' history'
alias l=' ls -sailF --color=auto'
alias la=' ls -A --color=auto'
alias lc=' ls -CF --color=auto'
alias ll=' ls -alF --color=auto'
alias lr=' ls -rtlF --color=auto'
EOF

cat - > ~/.funcs << "FUN"
whdr() {
echo '<!DOCTYPE html><head><title>bash web</title></head><body><pre>'

}
wftr() {
echo '</pre></body></html>'

}
calcit() {
    printf "%s\n" "$@" | bc -lq ~/.bcrc
}
FUN

cat - > ~/.bcrc << BCRC
scale=2
BCRC

cat -> ~/bin/upit << UPIT
#!/bin/bash
# upit
#snap refresh
sudo apt-get check
sudo apt-get clean
sudo apt-get -y autoclean
sudo apt-get remove
sudo apt-get -y autoremove
sudo apt-get update
sudo apt-get -y upgrade
sudo apt-get -y dist-upgrade
uname -a
cat /proc/version
cat /etc/os-release
```

```
#--
UPIT
chmod +x ~/bin/upit

cat - >> ~/.bashrc << SETF
if [ -f ~/.funcs ]; then
    . ~/.funcs
fi
#--
SETF


# enable the getty for ttyS0 (getty allows login)
sudo systemctl enable serial-getty@ttyS0.service
sudo systemctl start serial-getty@ttyS0.service
sudo systemctl daemon-reload

# install web server
sudo tasksel install web-server

# multiline comments - unorthodox
:<<'CMT'
here docs are awesome!
Linux runs the world.
Amazing for overwieght penguin.
CMT
#--
```

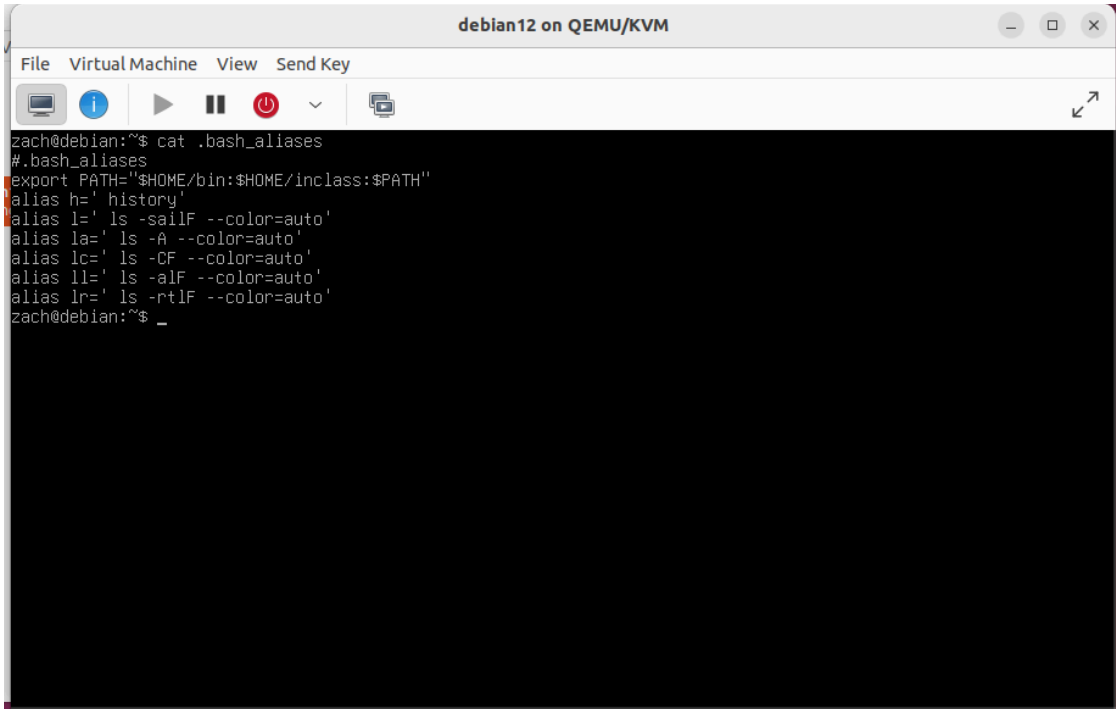
Basically what this does is creates files, .funcs, .bash_aliaes, and upit.

.funcs has some cool functions in it that we will use later.



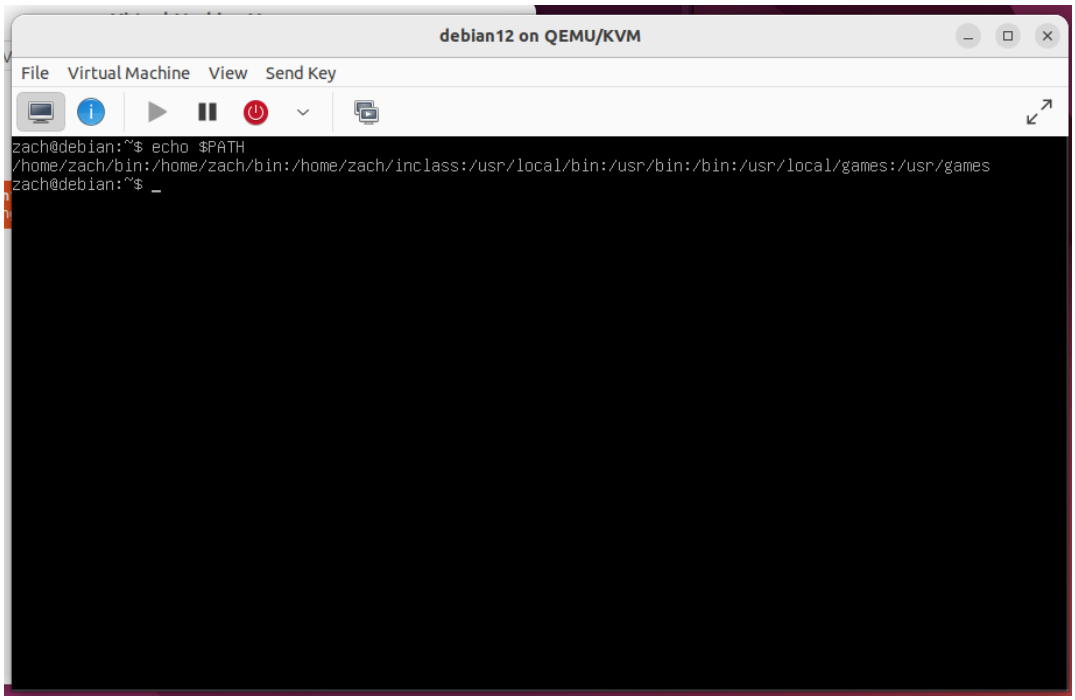
I don't know why the screenshot is so blurry.

.bash_aliases creates some bash aliases (huge surprise I know).



And upit is in the bin directory we made and it updates/upgrades stuff.

The script also adds the bin and inclass directories to your path.



Now we are going to use some of the cool functions from .funcs to do some web server stuff.

Make sure port 80 is open with nmap localhost.

If/once it is, go to a web browser and type in your ip address which you can find with ip a.

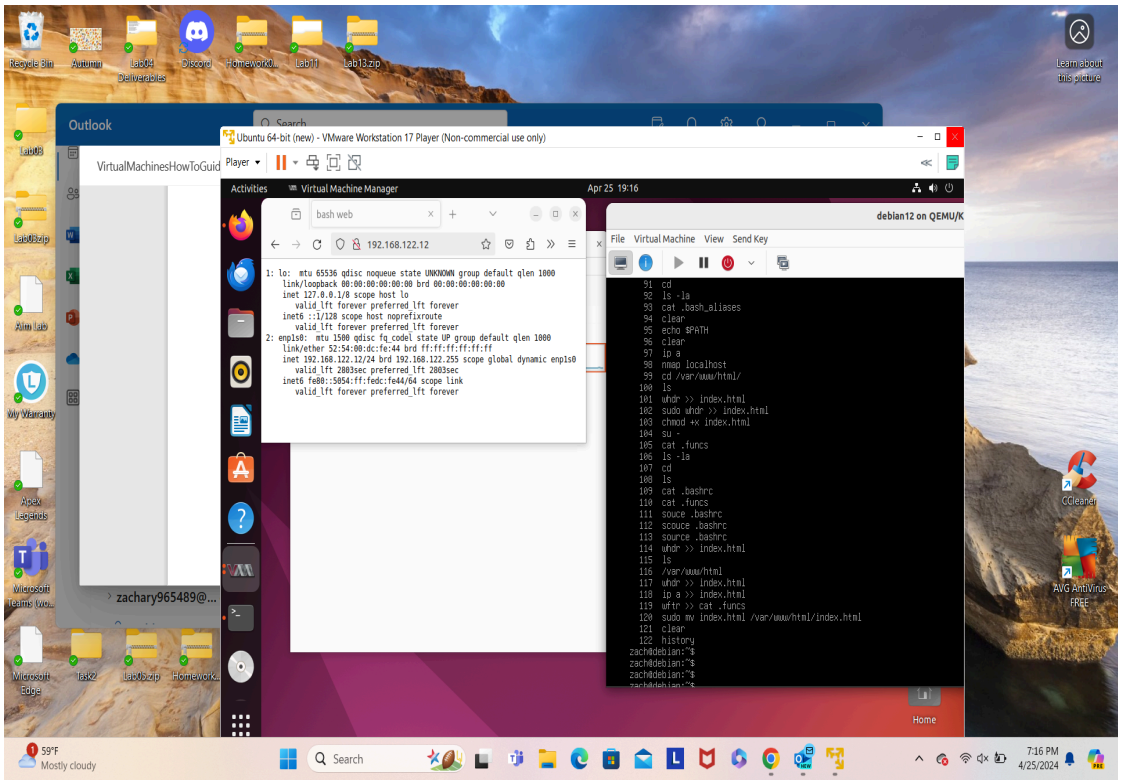
It should say somewhere “This is the default web page” and “It works”.

To change this page:

1. cd /var/www/html
2. whdr >> index.html

3. ip a >> index.html
4. wftr >> cat .funcs
5. sudo mv index.html /var/www/html/index.html

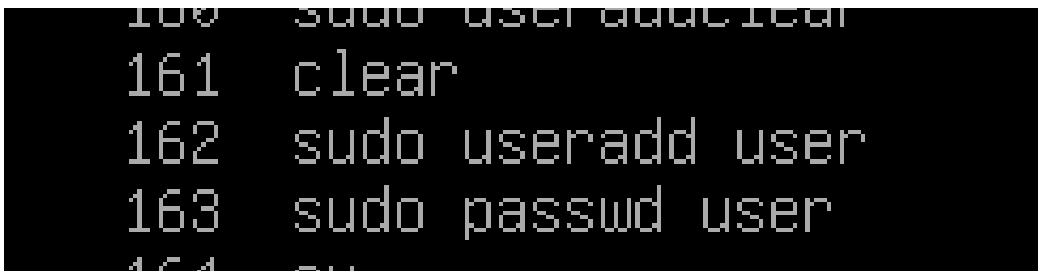
Type the ip into a web browser again and you should see the changes.



To add new users use these:

1. useradd user
2. passwd user

You can do this as root or sudo.



To add a user to sudo:

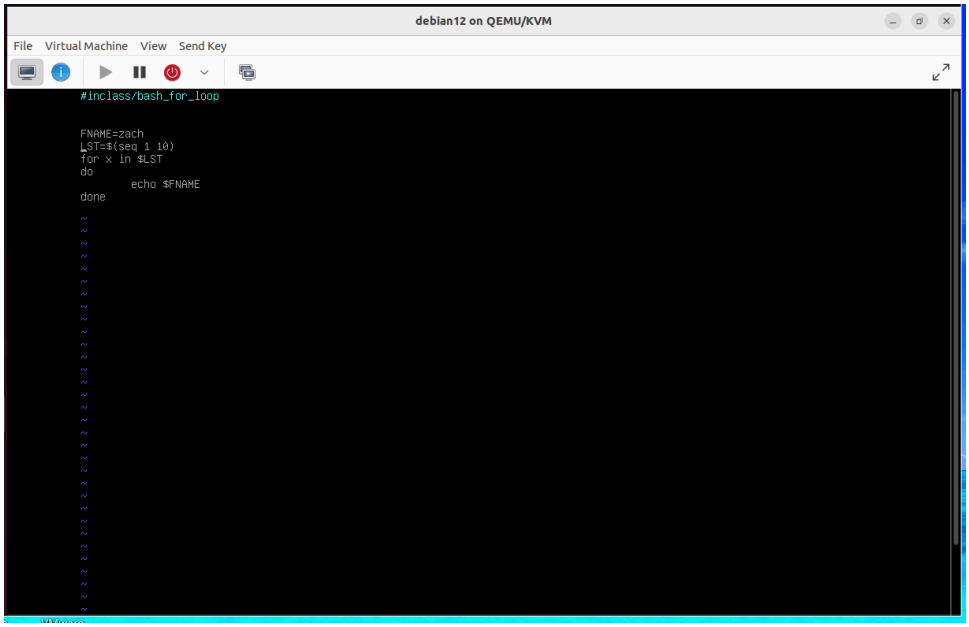
1. sudo usermod -aG sudo user

Check with: groups user.

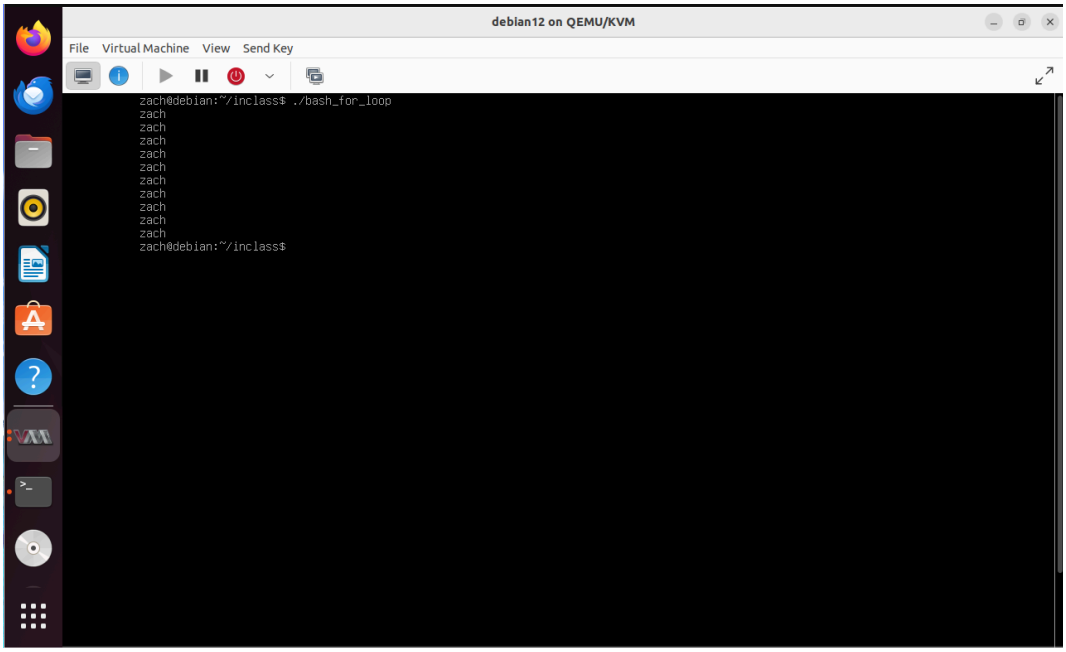
If this is your first time adding a user to sudo, you'll have to do it as root.

```
zach@debian:~$ su -
Password:
root@debian:~# usermod -aG sudo tux
root@debian:~# groups tux
tux : tux sudo
```

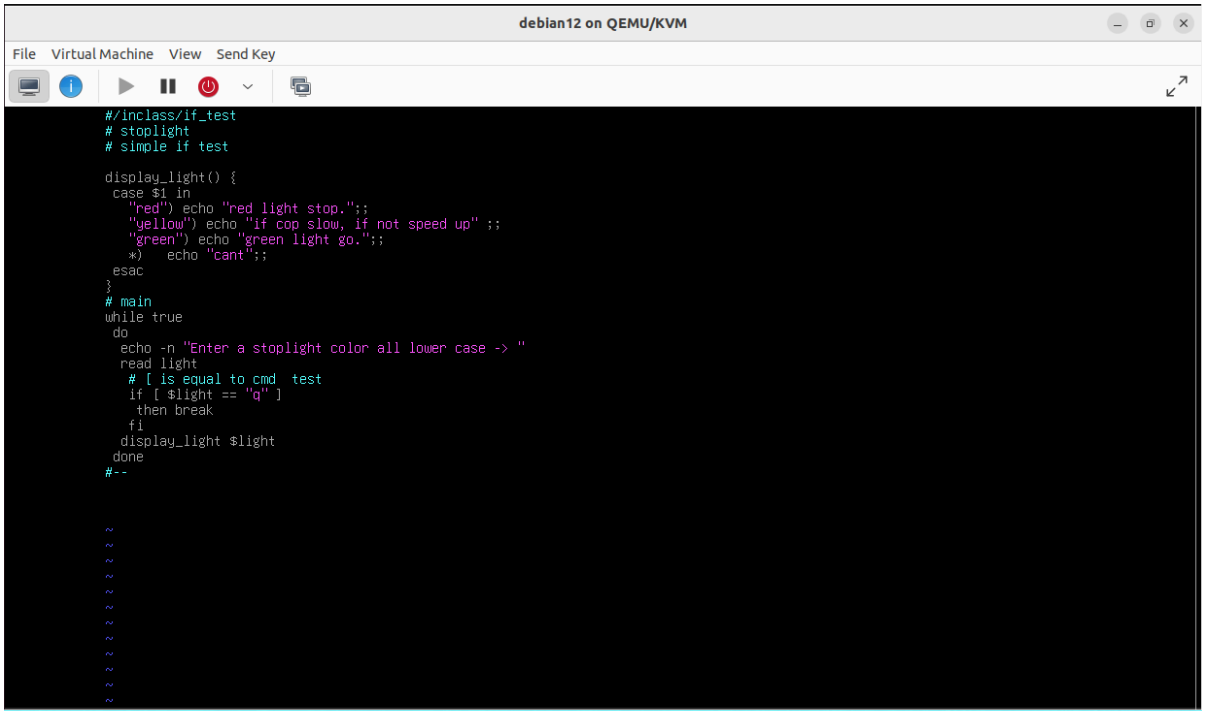
Bash for loop code:



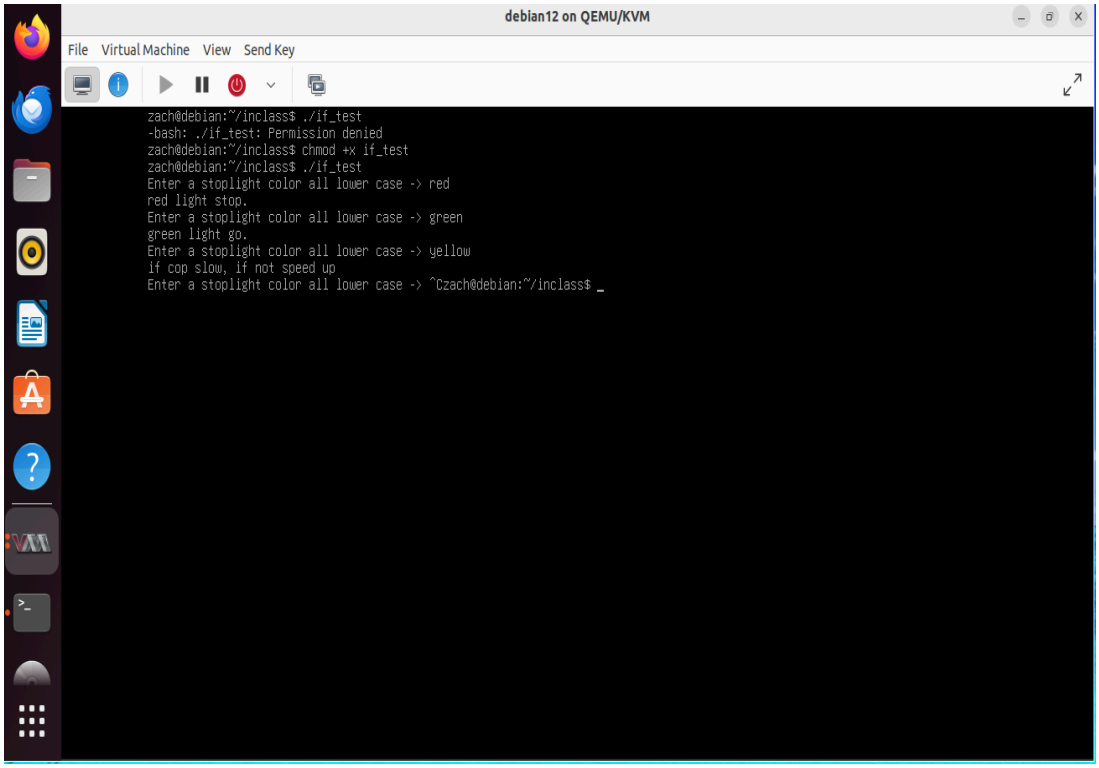
Demo:



If test code:



Demo:


debian12 on QEMU/KVM
File Virtual Machine View Send Key
zach@debian:~/inclass$ ./if_test
-bash: ./if_test: Permission denied
zach@debian:~/inclass$ chmod +x if_test
zach@debian:~/inclass$ ./if_test
Enter a stoplight color all lower case -> red
red light stop.
Enter a stoplight color all lower case -> green
green light go.
Enter a stoplight color all lower case -> yellow
if cop slow, if not speed up
Enter a stoplight color all lower case -> ^Czach@debian:~/inclass$