

## ***Creation, Installation, and Setup of a Virtual Ubuntu Server on a Macbook M2 chip Using UTM + Setup of SSH Keys***

*[Note: It is assumed that you have successfully downloaded the latest version of an Ubuntu server iso file and the UTM program.]*

### 1. Problem Statement



#### Description:

- Setting up virtualization on a macbook machine can be tricky, and with new proprietary (M1, M2... and more to come) chips, the difficulties and discrepancies between macbooks and other machines for the installation and setup of an Ubuntu server are growing in number. The goal of this document is to walk a reader through the process of creating a virtual Ubuntu server on a Macbook M2 chip; as well as, demonstrate how to set up SSH keys for a more secure connection through an SSH connection.

#### Initial Requirements:

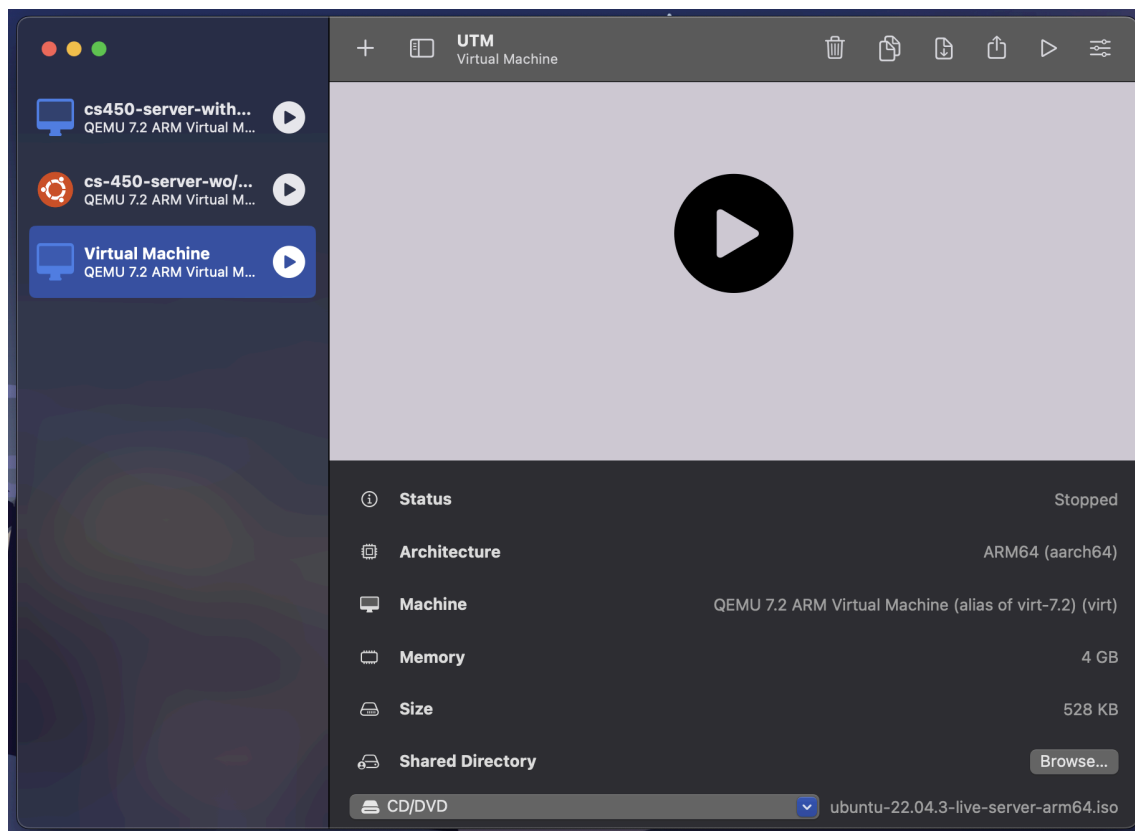
- As noted above, this document covers virtualization using the software “UTM” and thus it is highly recommended to use this in order to follow along with the setup and installation phase. If you wish to use a different software, you can follow along with the first section’s step 3 and then start from section 1.2 and onward.
  - If you are using “UTM” I would recommend you to read the documentation for the software at <https://docs.getutm.app/>
- It is assumed you have already downloaded the latest version of an ARM architecture Ubuntu server iso file. Before continuing, ensure this is the case and note its location.

## 1.1 Installation of Ubuntu Server Using UTM

### Step 1:

Start by opening UTM and selecting “Create a New Virtual Machine”. Then select “Virtualize” and then “Other”. Finally you’ll be prompted to mount the iso image of the Ubuntu server. Select browse and then navigate to where you stored the iso file and then select it. From here, UTM goes through the standard virtual machine (VM) setup, you can choose to change the default setting for memory, cpu cores, storage, and even add a shared directory if you choose. These settings are fully dependent on what you wish to use your server for so change according to your needs. For me in particular, I did the default memory, 2 cores total, 25 GB of storage and no shared drive.

Having completed the directions above, the UTM app should look something like the photo below (minus the other servers I have installed).

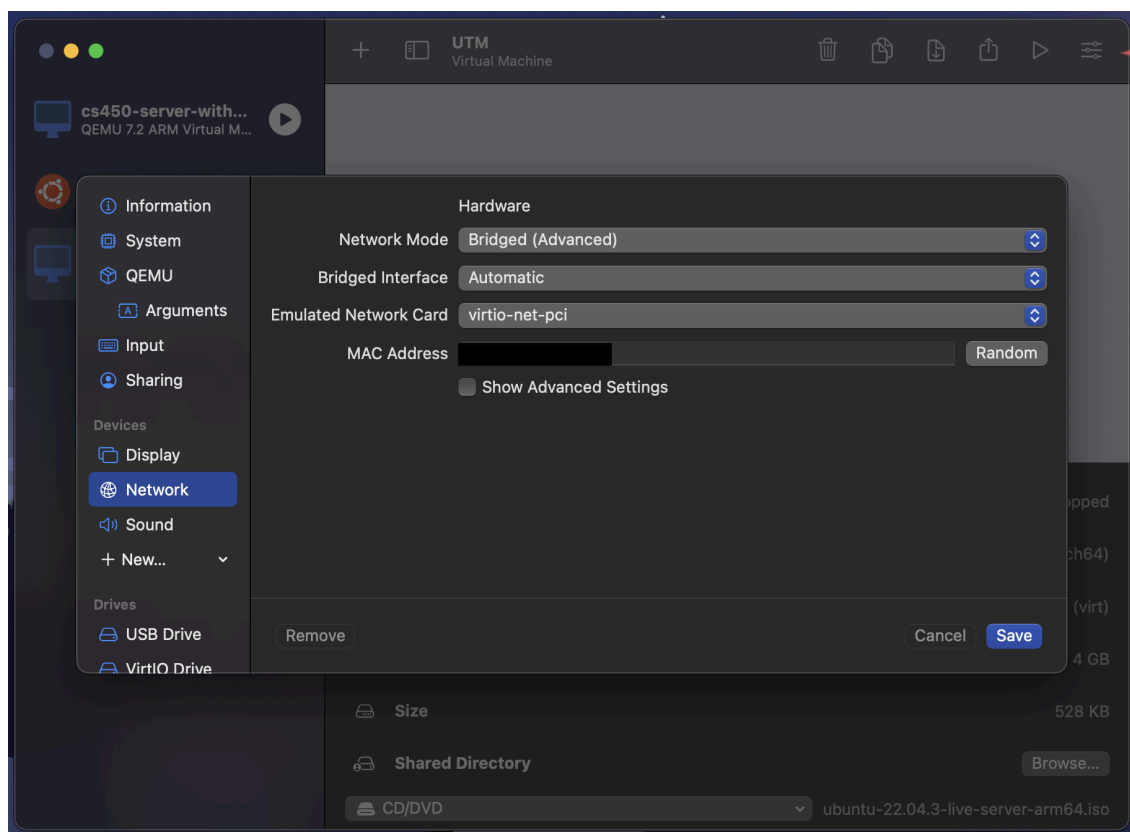


### Step 2:

Before starting the server, there is one major thing that needs to be changed before continuing. The default network settings in UTM uses the “Shared Network” mode. In a Shared network, as is described in the UTM documentation “Traffic is routed directly by the host operating system

and the guest shares a VLAN with the host. Services running on the guest and the host can see each other without additional configuration,” (Home, n.d.). However, other machines, not including your macbook, running the virtual machine are unable to connect to the VM with this mode, so we need to change the network mode to Bridged where “the host creates a layer 2 bridge with the specified interface,” which allows for connection to the server from any machine on the local network (Home, n.d.).

To change this setting, you can right click on the “Virtual Machine” to the left and select edit, or you can find the “Edit selected VM” button (button furthest to the right at the top of the UTM app). Then, navigate to the Network tab and select the dropdown menu for “Network Mode” and select “Bridged (Advanced)”. Finally, hit save and then you are ready to begin the server operating system installation.



### Step 3:

You can then hit play/run on the VM in the UTM app which will open a VM window. In your first prompt, it asks if you wish to “Try to install Ubuntu Server”, simply select this option and then the Ubuntu os installation will begin.

Note: The Ubuntu server does not have mouse integration, so navigation in the window is done using the keyboard. You can move your cursor with the arrow keys and select options with the spacebar and enter key.

Select a language of your choice, and then if your installer is out of date, it may ask if you wish to update to the newer version and continue, make sure to update if prompted.

Then select your keyboard layout, and then you'll be prompted to select the base installation. Here we are installing Ubuntu Server, which is checked by default, so we will just continue. Next is network connections, which is done automatically so we can continue here too. This walkthrough of an Ubuntu server is not utilizing any HTTP proxies so we can continue here too.

A small test is run, it only takes a few seconds to complete, once it has you can continue on.

Next you can select if you wish to use your entire disk, this is highly recommended if you are running off of a VM since there is no separate hardware the server has access to other than the allocated space you previously provided it. You also can choose to set up the disks into an LVM group, which is selected by default, and choose to encrypt it using LUKS, the decision for this is dependent on the purposes of the server you intend to create, for this demonstration, I will not need this encryption so I will leave this unselected and not encrypt the disks.

Finally you are presented with a file system summary, you can review this and ensure that everything looks correct and then continue. Continuing from here is a destructive action, installing an Ubuntu server directly on hardware would be cause for concern here since continuing completely wipes the disk being used, although, the VM has already been allocated an empty section of the disk and thus we do not need to worry about continuing here.

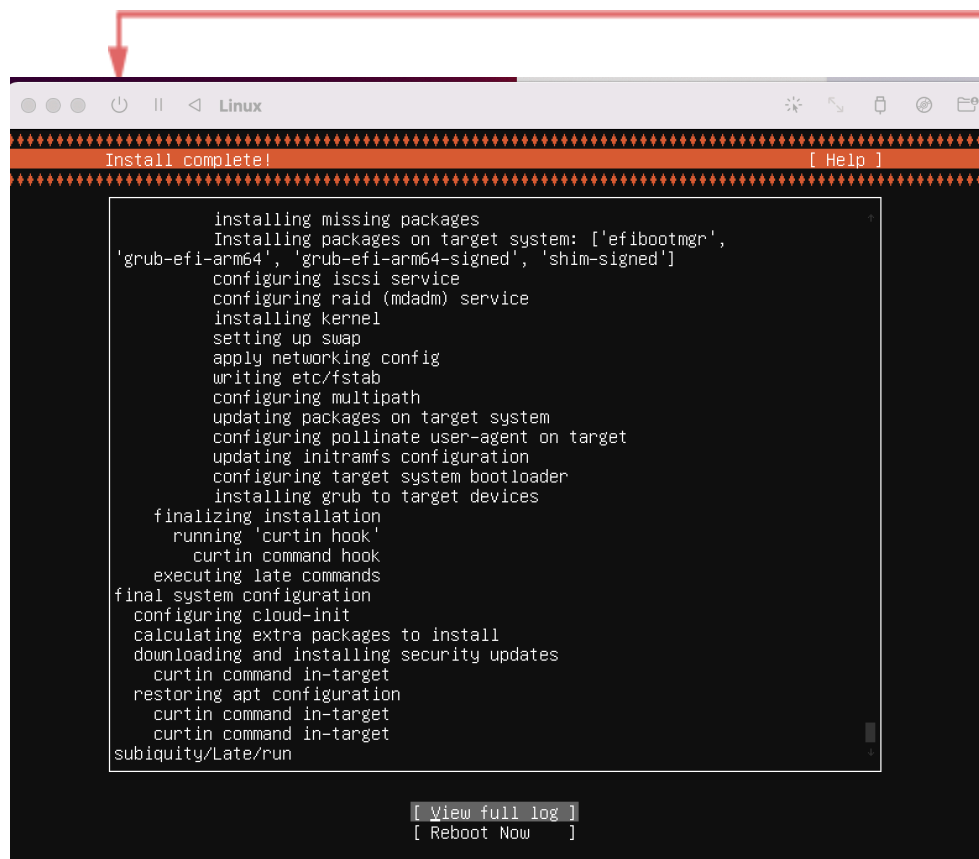
Next you are prompted to input your name, the server's name, and then the username and password for the user with root privileges. This username and password will be used later to log into a user's account with SSH as well so remember your credentials.

Next you are asked if you wish to upgrade to Ubuntu Pro, which is not necessary so we will continue.

Then you are asked if you wish to install OpenSSH server, which can be done from command line with a pre-existing server but, since we intend on using SSH we can skip the need to do that later on so we should select that we wish to install it and then continue. If you have already configured SSH keys and have them stored on Github or Launchpad, you can select one of those options; however, in our case we do not have any keys yet so we can select "No" and continue.

Finally, the last question for the setup asks if you wish to add any popular snap environments. If you wish you can add some here now, or wait until later after having completed this installation. For our case, none of the snap environments are necessary so we can skip this and continue.

From here, the server will begin to install. This may take a few minutes so make sure you wait until you are presented with the option of “Reboot Now” at the bottom, but **DO NOT select the “Reboot Now” and instead move on to the next step.**



<sup>1</sup> Image

#### Step 4:

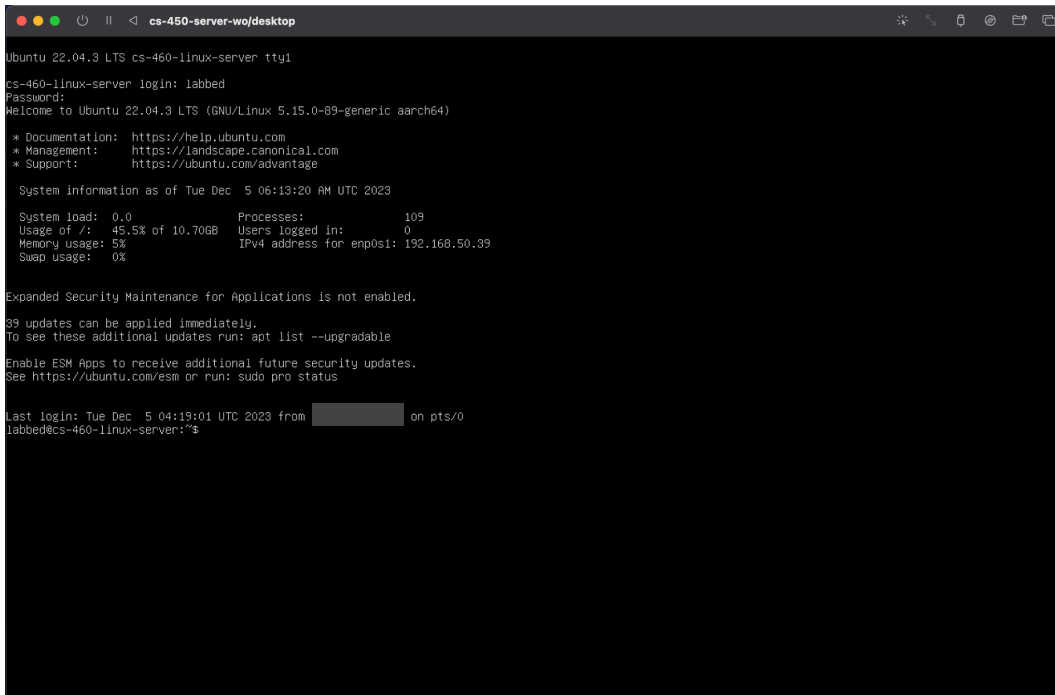
When using UTM, you cannot simply select reboot now at this point. If you do, the iso image would not have been dismounted yet, so when the server reboots, it would do so from the iso file again and restart the installation process. Instead click the power button at the top left of the window and hit “OK” when the confirmation window opens. You can then close the VM window and navigate back to the UTM app.

At the bottom of the information displayed about the currently selected VM, there should be a CD/DVD dropdown menu that is currently attached to the iso file. Click to open the menu and select “Clear”. This unmounts the iso file and allows for the boot manager to use the “VirtIO Drive” with the new Ubuntu Server installation. However, the boot order for the VM still begins with a now empty CD/DVD drive. This means that when booting, the boot manager first

<sup>1</sup> Image courtesy of Github, taken from a discussion asking about help on the issue described in step 4: <https://github.com/utmapp/UTM/discussions/3716>

attempts to use the empty drive, find nothing in it, and then move onto the second drive in the order. In order to prevent this, we need to swap the boot order. You can do this by going into the settings of the VM again and finding the list of Drives on the left. Drag the drive labeled “VirtIO Drive” above the “USB Drive” and now the boot order has also been changed to the correct order.

You can then run the VM again and wait until the server has fully booted and you are left at the console log in screen for an Ubuntu server. You can then login with the username and password you specified earlier in the installation process.



```

cs-450-server-wo/desktop
Ubuntu 22.04.3 LTS cs-460-linux-server tty1
cs-460-linux-server login: labbed
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Dec  5 06:13:20 AM UTC 2023

System load:  0.0               Processes:    109
Usage of /:   45.5% of 10.70GB   Users logged in:  0
Memory usage: 5%               IPv4 address for enp0s1: 192.168.50.39
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

39 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Dec  5 04:19:01 UTC 2023 from [REDACTED] on pts/0
labbed@cs-460-linux-server:~$

```

## 1.2 Configure the Ubuntu Server

### Step 1:

Having now successfully installed and logged into a new Ubuntu Server, it is best practice to ensure that the server is up to date. You can do this by entering the command:

“`sudo apt update`”

Sudo essentially means that the command proceeding it is an administrative command. This requires password authentication, so ensure you enter your password.

Next we will set up an Uncomplicated Firewall or “ufw” in order to better secure/explicitly dictate what traffic is allowed to connect to the server. Initially the firewall will be inactive, so before we activate it, let's add the rules we need.

The Ubuntu wiki provides a lot more information on the uses of “ufw” (Can be found here: <https://wiki.ubuntu.com/UncomplicatedFirewall>) however, for our purposes, we only need to enable SSH which is by default ported in through port 22. So we can add the rule by entering this command:

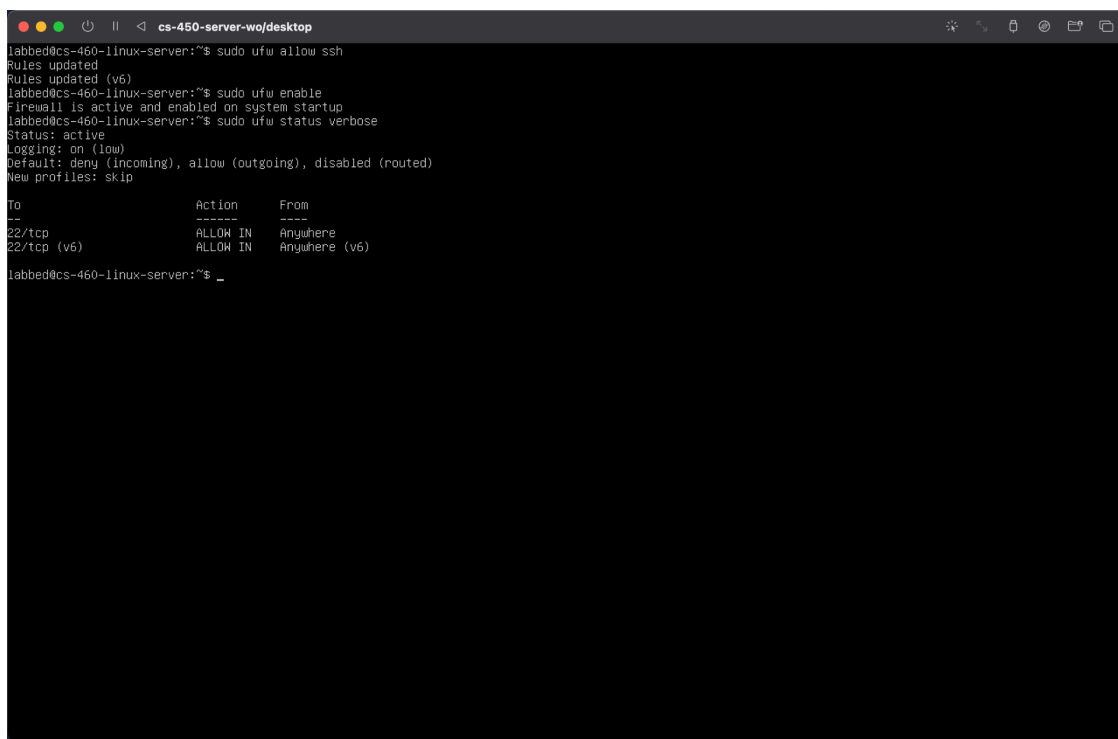
```
“sudo ufw allow ssh”
```

And then start the firewall by entering:

```
“sudo ufw enable”
```

Now you can view that the firewall is both active and what rules are in place by entering:

```
“sudo ufw status verbose”
```

A terminal window titled 'cs-450-server-wo/desktop' showing the execution of ufw commands. The user runs 'sudo ufw allow ssh', 'sudo ufw enable', and 'sudo ufw status verbose'. The output shows the firewall is active and allows SSH connections on port 22.

```
labbed@cs-460-linux-server:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
labbed@cs-460-linux-server:~$ sudo ufw enable
Firewall is active and enabled on system startup
labbed@cs-460-linux-server:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
22/tcp ALLOW IN Anywhere
22/tcp (v6) ALLOW IN Anywhere (v6)

labbed@cs-460-linux-server:~$ _
```

Now that we have our firewall up and allowing SSH connections, we can test our server’s SSH ability with a separate machine. In my case, I have a windows machine and I will be using puTTY to SSH into the server.

Before doing so, we need to gather the server’s public ip address in order to access the server. In order to find the public ip address, you can run the command:

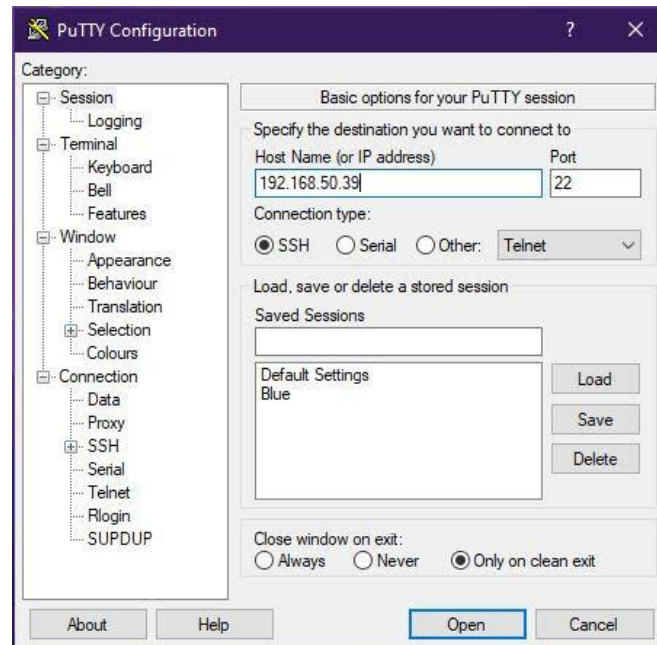
```
“hostname -I”
```

The number following this command is the server’s public ip address (might be something like 192.168.50.39). This address can be used on another machine with the ssh command in a

terminal or using a separate software like PuTTY. If you're using a terminal, you can use the following command and then entering your password:

```
"ssh [your username]@[the public ip address of your server]"
```

If you are using PuTTY, you simply need to put the IP address in the host name (or IP address) box and click "Open"



Then you can use your username and password to login.

```
labbed@cs-460-linux-server: ~
login as: labbed
labbed@192.168.50.39's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Dec  5 06:38:09 AM UTC 2023

System load:  0.0078125      Processes:           104
Usage of /:   46.4% of 10.70GB Users logged in:        1
Memory usage: 5%            IPv4 address for enp0s1: 192.168.50.39
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

39 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Dec  5 06:13:20 2023
labbed@cs-460-linux-server:~$
```



You have now successfully connected to your virtual Ubuntu server over ssh connection from a separate machine!

### ***1.3 Add and Implement SSH Key***

#### **Step 1:**

Now that our Ubuntu server has been installed and configured, we now need to secure the SSH connections with SSH keys. For this, we need to make a keypair on the macbook that you created the VM on. You can do this by opening a terminal on the macbook and entering:

```
"ssh-keygen"
```

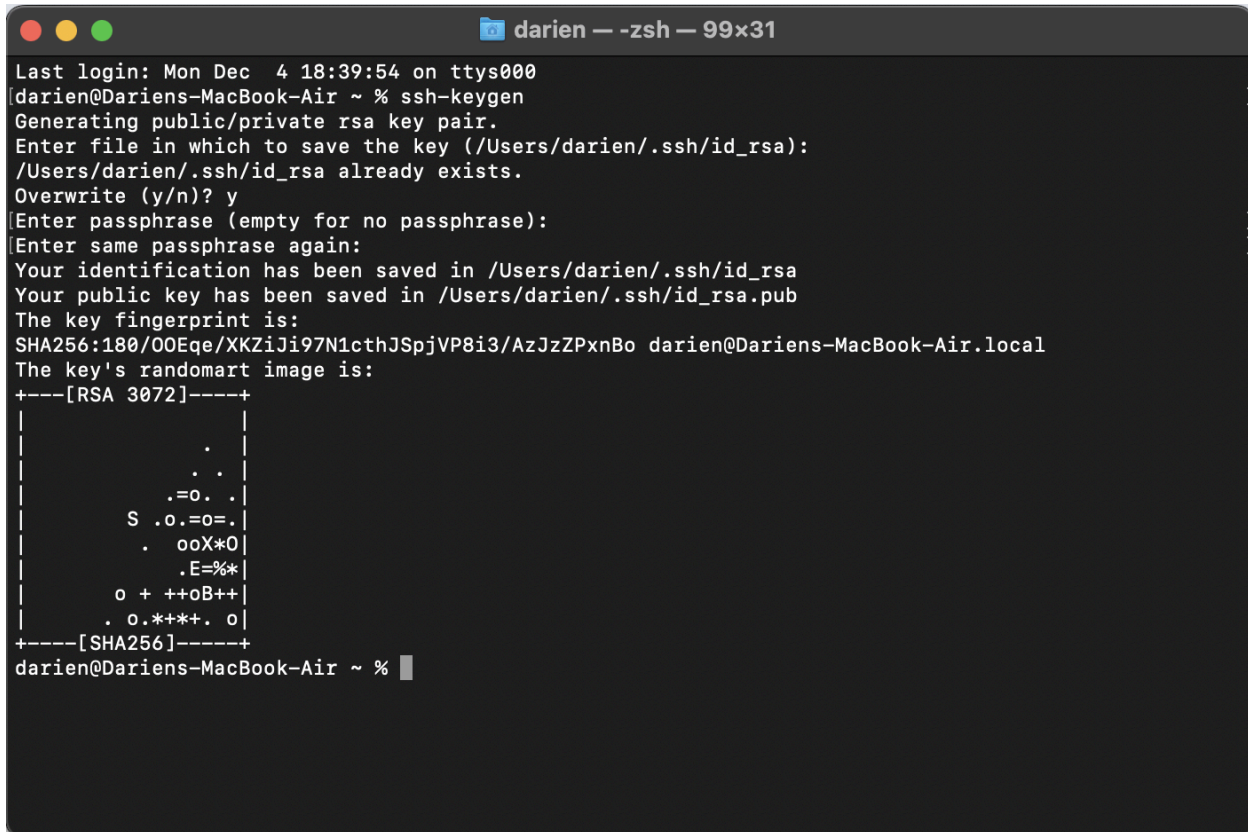
You can also add to the end of this command `"-b 4096"` if you wish to create a larger 4096-bit key but this is optional.

You will then be presented with where you wish to store your key, by default it is in your `.ssh/id_rsa` folder. You can specify a different location or hit enter to store it there. If you already have a key stored there, you will be asked if you wish to overwrite it. Make sure that you do not need this key for authentication anywhere else since overwriting it cannot be reversed and thus wherever the key was used will no longer have access.

Next you can set up a passphrase for extra security. Making a passphrase is highly recommended for a number of reasons:

- The private SSH key is never exposed on the network and the passphrase is only used to decrypt the key on the local machine. What this means is that a network-based brute forcing will not be possible against the passphrase.
- The private key is kept in a restricted directory and the SSH client won't recognize private keys not kept in restricted directories. The key needs to also be read and write permissions which are only available for the owner. This prevents other users from finding this information.
- Attackers attempting to crack the private SSH key passphrase need to have already gained access into the system. This means they already have access to your user account and/or the root account. In this situation, the passphrase can be used to prevent the attacker from being able to immediately log into your other servers which will hopefully give you time to create/implement a new SSH key pair, as well as, remove access from the key that the attacker found.

So, if you choose to enter a passphrase, treat it like creating a new password, enter something secure to ensure a safe connection.



```

Last login: Mon Dec  4 18:39:54 on ttys000
[darien@Dariens-MacBook-Air ~ % ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/darien/.ssh/id_rsa):
/Users/darien/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/darien/.ssh/id_rsa
Your public key has been saved in /Users/darien/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:180/00Ege/XKZiJi97N1cthJSpjVP8i3/AzJzZPxNBo darien@Dariens-MacBook-Air.local
The key's randomart image is:
+----[RSA 3072]-----+
|
|      .
|      .
|    .=.
|   S .o.=o=
|    . ooX*0
|    .E=%*
|   o + ++oB++
|   . o.*+++. o
+----[SHA256]-----+
darien@Dariens-MacBook-Air ~ %

```

You now have a key saved in the location that was specified.

## Step 2:

Next, we need to copy over the public SSH key to the server we made. To do so, enter:

```
"ssh-copy-id [username]@[ip address of your server]"
```

You may be asked if you are sure you wish to run this command, this would most likely be due to the local computer not recognizing the remote host. If you have already attempted to SSH into the server with your macbook's terminal, then you may not be prompted with this, otherwise, type "yes" and continue.

Now the public key should have been added to the remote account. We can verify this by using the terminal to SSH into the server. If you added a passphrase, you no longer need to enter your server password, and instead will be asked for your passphrase. After entering it, you will be logged into the server with your key!

**Step 3:**

While we did bypass the need to enter our password to the user account on the server, the password authentication system is still enabled on the account and thus still susceptible to brute force attacks. In order to remove this vulnerability, we need to edit the `sshd_config` file. To do this, enter the following command:

```
"sudo nano /etc/ssh/sshd_config"
```

```
GNU nano 6.2 /etc/ssh/sshd_config
# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location  ^M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line ^M-E Redo
```

From here we need to navigate to “PasswordAuthentication”. You can do so quickly by pressing “Ctrl+W” and then typing “PasswordAuthentication” to navigate straight to it. It should be commented out so remove the ‘#’ and change the config value of “yes” to “no” and then exit and save the file by pressing “Ctrl+X” then ‘Y’ and then hit enter.

To activate these changes, we just need to restart the SSH system, so run the command:

```
"sudo systemctl restart ssh"
```

You have now successfully configured SSH key authentication on your server. You have also entirely disabled password authentication and thus the server only responds to key authentication.

If you wished to add new users and wanted to access them from the same computer, repeat steps 2 and 3, changing the username entered when copying the key from the computer to the server (besides changing and updating the SSH config rules).

If you wanted to allow a new computer to have access, you would need to repeat all three steps on that new computer (besides changing and updating the SSH config rules).

## 2. Conclusion

Congratulations, you now have an Ubuntu server that can be accessed from any computer on the network with secure SSH keys.

From here you can do what you wish with your server, perhaps add users and computers, you could add a desktop GUI, you could simply use it to test out programs without using your own system. The possibilities are up to you.

## References

- Boucheron, B. (2020, April 23). *How to Set Up SSH Keys on Ubuntu 20.04*. DigitalOcean | Cloud Hosting for Builders.  
<https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-ubuntu-20-04>
- Cannot Install Ubuntu, reboot returns to install screen · utmapp/UTM · Discussion #3716*. (n.d.). GitHub. <https://github.com/utmapp/UTM/discussions/3716>
- Ellingwood, J., & Boucheron, B. (2014, October 20). *How To Configure SSH Key-Based Authentication on a Linux Server*. DigitalOcean | Cloud Hosting for Builders.  
<https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server>
- Home*. (n.d.). UTM Documentation. <https://docs.getutm.app/>
- Okamura, J. (2023, May 28). *Install Ubuntu on Mac M1 Powered by UTM*. Medium.  
<https://jun1okamura.medium.com/install-ubuntu-on-mac-m1-powered-by-utm-499aba3ba7e9>
- UncomplicatedFirewall - Ubuntu Wiki*. (n.d.). Home - Ubuntu Wiki.  
<https://wiki.ubuntu.com/UncomplicatedFirewall>
- OKAMURA, Jun. "Install Ubuntu on Mac M1 Powered by UTM." Medium, Medium, 28 May 2023, [jun1okamura.medium.com/install-ubuntu-on-mac-m1-powered-by-utm-499aba3ba7e9](https://jun1okamura.medium.com/install-ubuntu-on-mac-m1-powered-by-utm-499aba3ba7e9)
- Utmapp. "Cannot Install Ubuntu Reboot Returns to Install Screen - Utmapp/UTM - Discussion #3716." Github, <https://github.com/utmapp/UTM/discussions/3716>. Accessed 4 Dec. 2023.