

Diamond Supply Co.

By Kyle Ford

Table of Contents

<i>Inventory</i>	4
<i>Custom Network</i>	4
<i>IDs and Passwords</i>	4
<i>Network Topology Diagram</i>	5
<i>Node.js Application (Ghost) on Docker</i>	5
Install base CentOS 7 Virtual Machine “ITE229-docker (11)”	6
SSH into CentOS VM.....	7
Update CentOS.....	7
Install EPEL Packages	8
Install Docker CE	8
Start and Enable Docker	10
Test Docker (hello-world)	10
Disable SELinux on CentOS 7 Virtual Machine.....	11
Install Ghost Docker Container	12
Test Ghost.....	13
<i>NginX Reverse Proxy</i>	14
Install base CentOS 7 Virtual Machine “ITE229-NginX (10)	14
SSH into CentOS VM.....	14
Update CentOS.....	15
Disable SELinux	15
Disable Firewall.....	16
Install EPEL Packages	16
Install NginX.....	17
Start and Enable NginX	18
Reverse Proxy to Ghost Site	18
<i>Install WordPress on Ubuntu - LAMP Stack</i>	20

Base Ubuntu 18.04 Install	20
Set Static IP	21
SSH into Ubuntu VM.....	21
Update Ubuntu	21
Install and Configure Apache	21
Install and Configure MySQL.....	23
Install and configure PHP	25
Test PHP	25
Database Configuration in MySQL.....	26
<i>Install WordPress.....</i>	27
Clone WordPress.....	27
Edit Ownership	28
Edit .htaccess	30
<i>WordPress Configuration</i>	32
WordPress Configuration Process	33
Test WordPress	34
<i>WordPress Security Settings and Configurations.....</i>	38
Security Summary	38
Defense-in-depth	39
Before/After Configuration.....	39
Testing and Validation Process	40
WordPress Security Conclusion.....	41
<i>Full Report Conclusion</i>	41
<i>Appendix A</i>	42
NginX Config File	42
<i>Appendix B</i>	43
NginX Access Log File.....	43

Inventory

EQUIPMENT	OPERATING SYSTEM	ADDITIONAL INFO	IP ADDRESS
Router/Custom Network			10.10.229.1
Docker	CentOS	Ghost Container	10.10.229.11
NginX Reverse Proxy	CentOS	Reverse Proxy	10.10.229.10
Wordpress	Ubuntu	LAMP Stack running WordPress	10.10.229.12

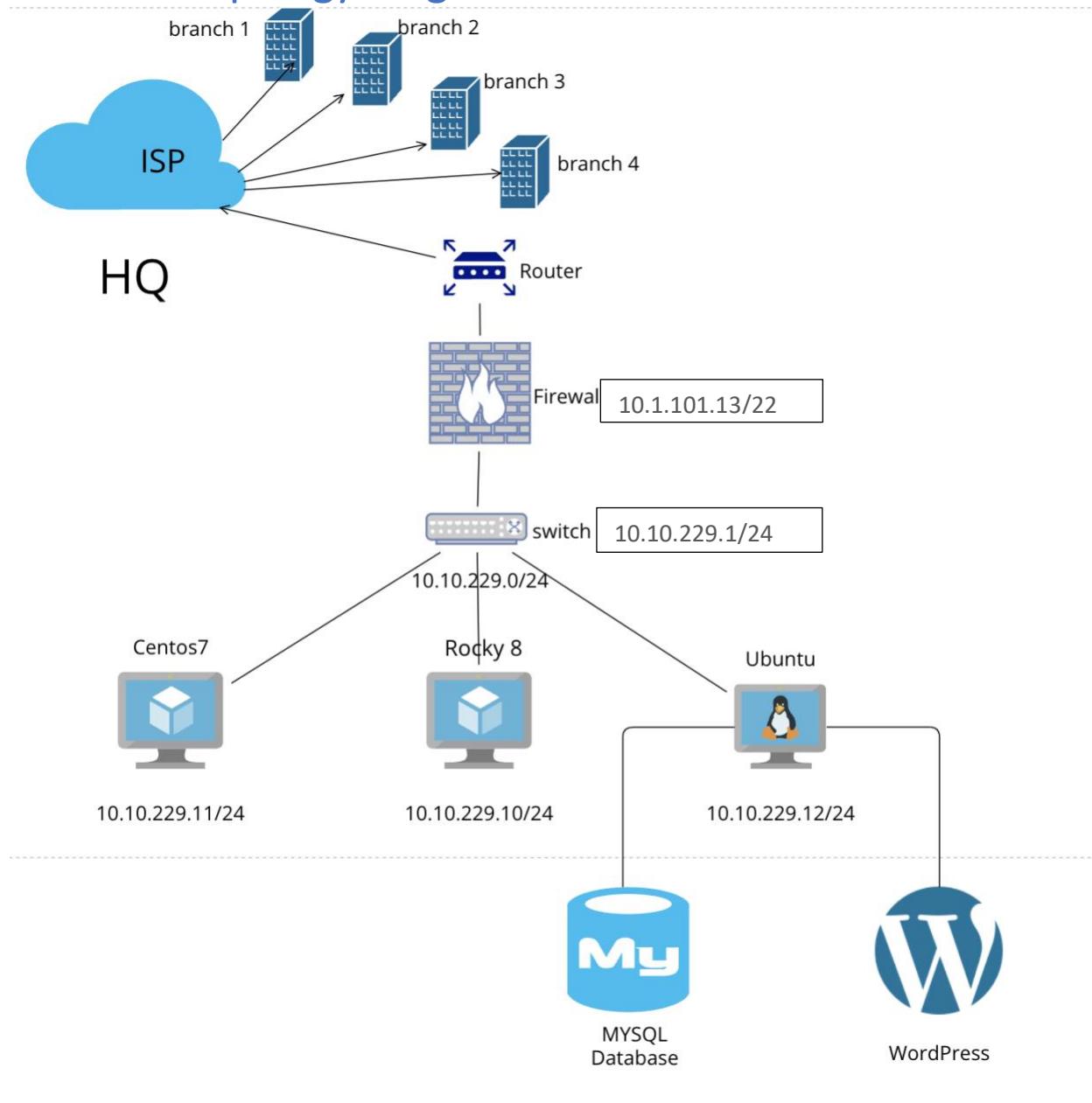
Custom Network

NETWORK NAME	SUBNET IP	SUBNET MASK	DNS	GATEWAY
ITE229	10.10.229.0	255.255.255.0	10.10.229.1	10.10.229.1

IDs and Passwords

ACCOUNT	USER ID	PASSWORD
CentOS Root User	[REDACTED]	[REDACTED]
CentOS Network User	[REDACTED]	[REDACTED]
CentOS Host User	[REDACTED]	[REDACTED]
MySQL Network User	[REDACTED]	[REDACTED]
MySQL Host User	[REDACTED]	[REDACTED]
WordPress Admin	[REDACTED]	[REDACTED]

Network Topology Diagram



Node.js Application (Ghost) on Docker

Install base CentOS 7 Virtual Machine “ITE229-docker (11)”

1. Begin by downloading the CentOS 7 ISO file from the official website. You can choose between a minimal or DVD ISO depending on your specific needs.
2. Next, launch your preferred virtualization software. For example, if you are utilizing Oracle VirtualBox, click on the "New" button to create a new virtual machine.
3. In the "Create Virtual Machine" wizard, assign a name to your virtual machine and select "Linux" as the type, and "Red Hat (64-bit)" as the version.
4. Allocate the appropriate amount of memory and number of CPUs to the virtual machine, taking into consideration your system resources and the anticipated workload.
5. Create a new virtual hard disk or use an existing one if available. Assign the appropriate amount of disk space to accommodate your requirements.
6. In the "Storage" settings, attach the CentOS 7 ISO file to the virtual CD/DVD drive.
7. Start the virtual machine and proceed with the CentOS 7 installation wizard. Select your preferred language, time zone, and keyboard layout.
8. Select "Installation Destination" and choose the disk or partition where CentOS 7 will be installed. You may opt to utilize automatic partitioning as well.
9. Choose the packages you want to install, keeping in mind that a minimal installation necessitates selecting solely the "Base System" package group.
10. Configure the network settings, including the hostname, IP address, and DNS servers.
11. Establish the root password and create a regular user account.
12. Click on "Begin Installation" and wait for the installation to complete.
13. Once the installation is complete, reboot the virtual machine and log in as the user you created.

There you have it! By following these instructions, you'll now have a base CentOS 7 virtual machine installed and ready to utilize.

SSH into CentOS VM

1. Begin by launching your CentOS VM in your preferred virtualization software.
 2. Retrieve the IP address assigned to the CentOS VM by running the command **ip addr show** in the terminal within the VM.
 3. Ensure that the SSH service is installed and running on the CentOS VM. Verify this by running the command **systemctl status sshd.service**.
 4. On your local machine, launch a terminal and input the command **ssh username@ip_address**. Replace "username" with the username of the account on the CentOS VM that you wish to access, and replace "ip_address" with the IP address of the CentOS VM.
 5. If you have not previously connected to that IP address, you will receive a prompt to add the IP address to the list of known hosts. Input "yes" to proceed.
 6. You will then be prompted to enter the password for the username that you specified earlier. Input the password and press Enter.
 7. You should now have successfully logged into the CentOS VM via SSH.

These steps will allow you to manage your CentOS VM using the terminal through SSH.

Update CentOS

- **sudo yum update -y** updates the CentOS 7 system with the latest updates available.
 - **sudo yum install nano** installs the Nano text editor which helps edit configuration files.

sudo yum install epel-release -y installs the EPEL (Extra Packages for Enterprise Linux)

```
QEMU (kgford1-CentOS7-2304) - noVNC

https://itlab.fsemergingtech.com/?console=kvm&novnc=1&vmid=2790&vmname=kgford1-CentOS7-2304&node=t-133&resize=off&cmd=

inet 10.10.229.11/24 brd 10.10.229.255 scope global noforwardroute eth0
    link layer [ether] brd 00:0c:29:11:00:00
        link layer [ether] brd 00:0c:29:11:00:00
            valid_lif forever preferred_lif forever
[root@localhost ~]# sudo yum update -y
Loading mirror speeds from cached hostfile
  * base: ngc.mirrors.cloudflare.net
  * extras: ftp.usf.edu
  * updates: mirror.mail.us.leaseweb.net
base
extras
updates
updates/7-0.6.6-primary_db
Resolving Dependencies
--> Running transaction check
--> Package tzdata.noarch 0:2823c-1.e17 will be updated
--> Package tzdata.noarch 0:2823c-1.e17 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

Transaction Summary
  Upgrade 1 Package

Total download size: 497 k
Downloading packages:
tzdata-2823c-1.e17.noarch.rpm                                              1:497 kB  00:00:00

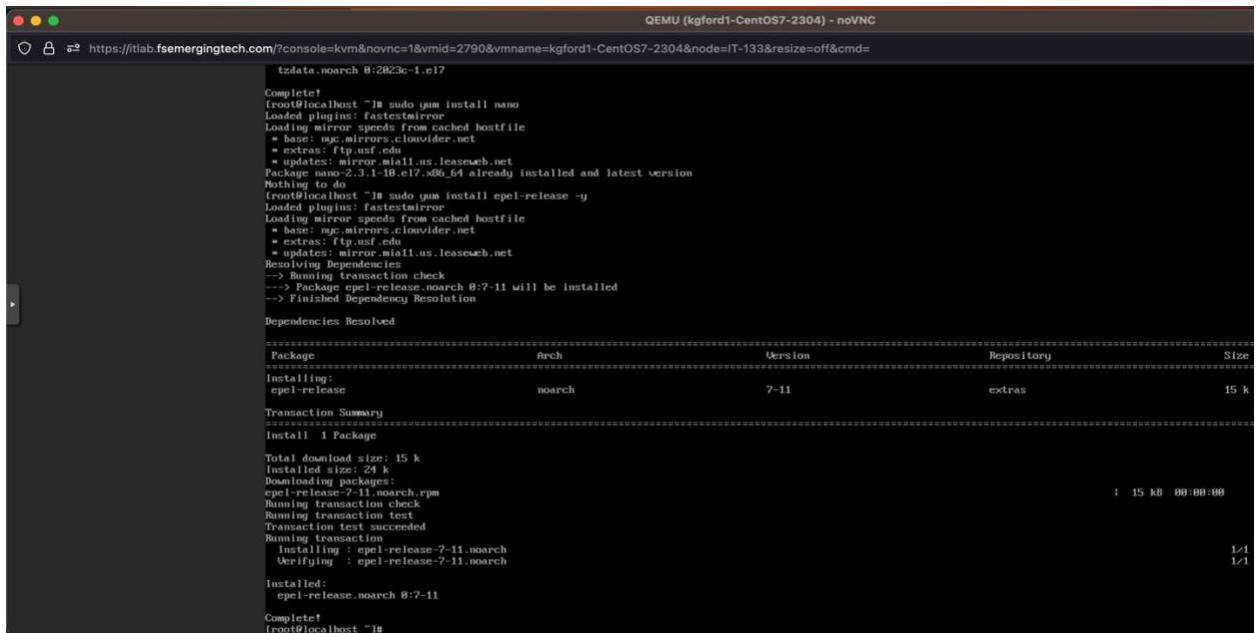
Running transaction check
Running transaction test
  Updating   : tzdata-2823c-1.e17.noarch                                     1/2
  Verifying  : tzdata-2823c-1.e17.noarch                                     1/2
  Verifying  : tzdata-2823c-1.e17.noarch                                     2/2

Updated:
  tzdata.noarch 0:2823c-1.e17

Complete!
[root@localhost ~]
```

Install EPEL Packages

- **sudo yum install epel-release -y** installs the EPEL (Extra Packages for Enterprise Linux) repository which has additional software packages that are not available in the default CentOS repository.
- **yum install -y yum-utils** installs the yum utilities package which provides utilities for managing software packages with yum.



The screenshot shows a terminal session in a QEMU environment. The user is root and is installing the EPEL repository using the command `sudo yum install epel-release -y`. The output shows the repository being added and then a transaction starting to install the `epel-release` package from the `extras` repository. The transaction summary indicates a total download size of 15 kB and an installed size of 24 kB. The transaction test and final transaction steps are also shown.

```
tzdata.noranch 0:2823c-1.el7
Complete!
[root@localhost ~]# sudo yum install nano
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: nyc.mirrors.clouvider.net
 * extras: ftp.usf.edu
 * updates: mirror.mail.us.leaseweb.net
Package nano-2.3.1-18.el7.x86_64 already installed and latest version
Nothing to do
[root@localhost ~]# sudo yum install epel-release -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: nyc.mirrors.clouvider.net
 * extras: ftp.usf.edu
 * updates: mirror.mail.us.leaseweb.net
--> Running transaction check
--> Package epel-release.noarch 0:7-11 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
=====
Transaction Summary
Install 1 Package
=====
Package           Arch      Version       Repository      Size
=====
Installing:
epel-release     noarch    7-11          extras        15 k
=====
Transaction Summary
Install 1 Package
=====
Total download size: 15 kB
Installed size: 24 kB
Downloading packages:
epel-release-7-11.noarch.rpm
Running transaction test
Running transaction test
Transaction test succeeded
Running transaction
  Installing : epel-release-7-11.noarch
  Verifying   : epel-release-7-11.noarch
Installed:
  epel-release.noarch 0:7-11
Complete!
[root@localhost ~]#
```

Install Docker CE

- **yum-config-manager --add-repo <https://download.docker.com/linux/centos/dockerce.repo>** adds the Docker repository to the available list of repositories, allowing the system to download and install the Docker CE engine.

```

QEMU (kgford1-CentOS7-2304) - noVNC
https://itlab.fsemergingtech.com/?console=kvm&novnc=1&vmid=2790&vmmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

Package          Arch    Version           Repository      Size
=====
Install dependencies:
gum-utils        noarch  1.1.31-54.e17_8   base            122 k
Installing for dependencies:
libxml2-python   x86_64  2.9.1-6.e17_9.6   updates         247 k
python-chardet  noarch  2.2.1-3.e17_8   base            227 k
python-kitchen   noarch  1.1.1-5.e17_8   base            209 k

Transaction Summary
Install 1 Package (+3 Dependent packages)

Total download size: 863 k
Installed size: 4.3 M
Downloading packages:
C1:0: python-kitchen-1.1.1-5.e17.noarch.rpm
C2:0: gum-utils-1.1.31-54.e17_8.noarch.rpm
C3:0: libxml2-python-2.9.1-6.e17_9.6.x86_64.rpm
C4:0: python-chardet-2.2.1-3.e17.noarch.rpm

Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : python-chardet-2.2.1-3.e17.noarch
  Installing : python-kitchen-1.1.1-5.e17.noarch
  Installing : libxml2-python-2.9.1-6.e17_9.6.x86_64
  Installing : gum-utils-1.1.31-54.e17_8.noarch
  Verifying  : python-kitchen-1.1.1-5.e17.noarch
  Verifying  : libxml2-python-2.9.1-6.e17_9.6.x86_64
  Verifying  : python-chardet-2.2.1-3.e17.noarch

Installed:
  gum-utils.noarch 0:1.1.31-54.e17_8

Dependency Installed:
  libxml2-python.x86_64 0:2.9.1-6.e17_9.6           python-chardet.noarch 0:2.2.1-3.e17           python-kitchen.noarch 0:1.1.1-5.e17

Complete!
[root@localhost ~]# sudo yum-config-manager --add repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugin: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
Creating mirrorinfo file at /etc/yum.repos.d/docker-ce.repo to file /etc/yum.repos.d/docker-ce.repo: [Errno 141] curl#32 - "Couldn't open file"
[root@localhost ~]#

```

- yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin dockercompose-plugin**
installs the Docker CE engine and additional features such as the Docker CLI, containerd.io, the buildx plugin, and the compose plugin.

```

QEMU (kgford1-CentOS7-2304) - noVNC
https://itlab.fsemergingtech.com/?console=kvm&novnc=1&vmid=2790&vmmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

Transaction test succeeded
Installing : libcgroup-0.41-21.e17.x86_64
Installing : setools-libs-3.3.0-4.e17.x86_64
Installing : docker-compose-plugin-2.17.2-1.e17.x86_64
Installing : docker-buildx-plugin-0.10.4-1.e17.x86_64
Installing : slirp4netns-4.3-1.e17_9.x86_64
Installing : checkpointcy-2.5-0.e17.x86_64
Installing : auditlibs-python-2.8.5-4.e17.x86_64
Installing : python-setuptools-41.1.1-1.e17.x86_64
Installing : fuse3-libs-python-2.5-34.e17.x86_64
Installing : Z:container-selinux-2.119.2-1.911c772.e17_8.noarch
Installing : containerd.io-1.6.28-3.1.e17.x86_64
Installing : docker-ce-23.0.3-1.e17.x86_64
Installing : fuse-overlays-0.7.2-6.e17_8.x86_64
Installing : docker-buildx-plugin-0.10.4-1.e17.x86_64
Installing : docker-ce-cli-23.0.3-1.e17.x86_64
Installing : docker-ce-rootless-extras-0.23.0.3-1.e17.x86_64
Installing : docker-ce-23.0.3-1.e17.x86_64
Verifying: docker-buildx-plugin-0.10.4-1.e17.x86_64
Verifying: fuse3-libs-3.3-1.e17.x86_64
Verifying: auditlibs-0.25-6.e17.noarch
Verifying: auditlibs-python-2.8.5-4.e17.x86_64
Verifying: fuse-overlays-0.7.2-6.e17_8.x86_64
Verifying: docker-ce-23.0.3-1.e17.x86_64
Verifying: docker-ce-23.0.3-1.e17.x86_64
Verifying: Z:container-selinux-2.119.2-1.911c772.e17_8.noarch
Verifying: containerd.io-1.6.28-3.1.e17.x86_64
Verifying: docker-buildx-plugin-0.10.4-1.e17.x86_64
Verifying: fuse3-libs-3.3-1.e17.x86_64
Verifying: slirp4netns-4.3-1.e17_9.x86_64
Verifying: policycoreutils-python-2.5-34.e17.x86_64
Verifying: libsemanage-python-2.5-14.e17.x86_64
Verifying: docker-ce-rootless-extras-0.23.0.3-1.e17.x86_64
Verifying: setools-libs-3.3.0-4.e17.x86_64
Verifying: libcgroup-0.41-21.e17.x86_64

Installed:
  docker-ce.x86_64 3:23.8.3-1.e17

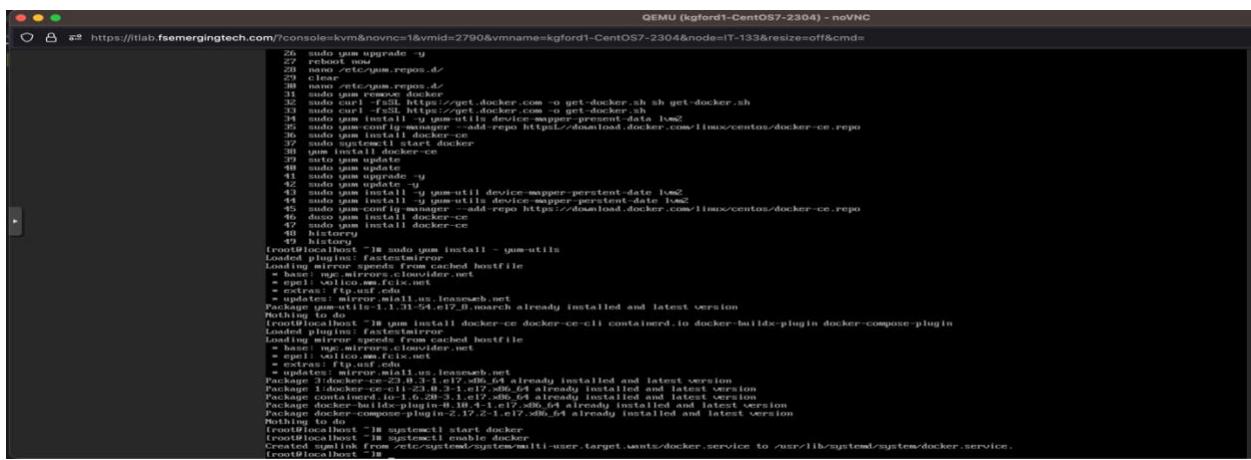
Dependency Installed:
  auditlibs-0.25-6.e17.noarch 0:2.8.5-4.e17           checkpointcy.x86_64 0:2.5-8.e17                docker-buildx-plugin.x86_64 0:0.10.4-1.e17
  container-selinux.noarch 2:2.119.2-1.911c772.e17_8  docker-compose-plugin.x86_64 0:2.17.2-1.e17
  docker-ce-23.0.3-1.e17.x86_64 1:23.8.3-1.e17       docker-ce-rootless-extras.x86_64 0:0.23.0.3-1.e17
  fuse3-libs.x86_64 0:3.3-1.e17                      libcgroup.x86_64 0:0.41-21.e17
  policycoreutils-python.x86_64 0:2.5-34.e17          fuse-overlays.x86_64 0:0.7.2-6.e17_8
  slirp4netns.x86_64 0:4.3-1.e17                     libsemanage-python.x86_64 0:2.5-14.e17
  setools-libs.x86_64 0:3.3-4.e17                   python-setuptools.x86_64 0:41.1.1-1.e17

Complete!
[root@localhost ~]#

```

Start and Enable Docker

- **systemctl start docker** starts the Docker service.
- **systemctl enable docker** enables the Docker service to start automatically during system boot.



The screenshot shows a terminal session in a QEMU environment. The user is performing the following steps:

- 26 sudo yum upgrade -y
- 27 reboot now
- 28 rm -rf /etc/yum.repos.d/*
- 29 rm -rf /etc/yum.repos.d/*
- 30 nano /etc/yum.repos.d/docker.repo
- 31 curl -fsSL https://get.docker.com | sh
- 32 sudo curl -fsSL https://get.docker.com | sh get-docker.sh
- 33 sudo curl -fsSL https://get.docker.com | sh get-docker.sh
- 34 sudo yum install -y yum-utils device-mapper-persistent-data lsof
- 35 sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
- 36 sudo yum install docker-ce
- 37 sudo systemctl start docker
- 38 sudo systemctl enable docker
- 39 sudo yum update
- 40 sudo yum update
- 41 sudo yum upgrade -y
- 42 sudo yum update
- 43 sudo yum install -y yum-util device-mapper-persistent-data lsof
- 44 sudo yum install -y yum-util device-mapper-persistent-data lsof
- 45 curl -fsSL https://download.docker.com/linux/centos/docker-ce.repo
- 46 sudo yum install docker-ce
- 47 sudo yum install docker-ce
- 48 history
- 49 history

After the command "root@localhost ~# sudo yum install -y yum-utils" is run, the output shows:

```
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
base                                | 1.1 kB/s
  • base: mirror.sclouder.net
  • epel: mirror.sclouder.net
  • updates: mirror.sclouder.net
  • updates: mirror.mall.us.leansweb.net
Nothing to do
Nothing to do
root@localhost ~# curl -fsSL https://get.docker.com | sh
Loaded plugin: fastestmirror
Loading mirror speeds from cached hostfile
base                                | 1.1 kB/s
  • base: mirror.sclouder.net
  • epel: mirror.sclouder.net
  • updates: mirror.sclouder.net
  • updates: mirror.mall.us.leansweb.net
Package docker-ce-2.18.0-3.el7_17.1.x86_64 already installed and latest version
Package containerd.io-1.6.28-3.el7_17.x86_64 already installed and latest version
Package docker-ce-cli-2.18.0-3.el7_17.1.x86_64 already installed and latest version
Package docker-compose-plugin-2.17.2-1.el7.x86_64 already installed and latest version
root@localhost ~# systemctl start docker
root@localhost ~# systemctl enable docker
root@localhost ~# cat /etc/systemd/system/docker.service | grep User
User= docker
root@localhost ~#
```

Test Docker (hello-world)

- **docker -v** displays the Docker version currently installed on the system.

- **docker run hello-world** runs the "hello-world" container to ensure Docker is correctly installed and working.

```
QEMU (kgford1-CentOS7-2304) - noVNC

m?console=kvm&novnc=1&vmid=2790&vmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

CentOS Linux 7 (Core)
Kernel 3.10.0-1160.88.1.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Tue Apr 11 21:42:56 on ttys0
[root@localhost ~]# docker -v
Docker version 23.0.3, build 3e7cbfd
[root@localhost ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:ffbb13da98453e8f04d33a6eee5bb8e46ee50d80ebe17735fc0779d0349e889e9
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
 to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
[root@localhost ~]#
```

Disable SELinux on CentOS 7 Virtual Machine

- **nano /etc/selinux/config** opens the SELinux configuration file in the Nano text editor.
- Change the text "Enforcing" option to "Disabled" in the configuration file.
- Save the changes and exit the Nano text editor.
- **sestatus** displays the current status of SELinux.

```
QEMU (kgford1-CentOS7-2304) - noVNC

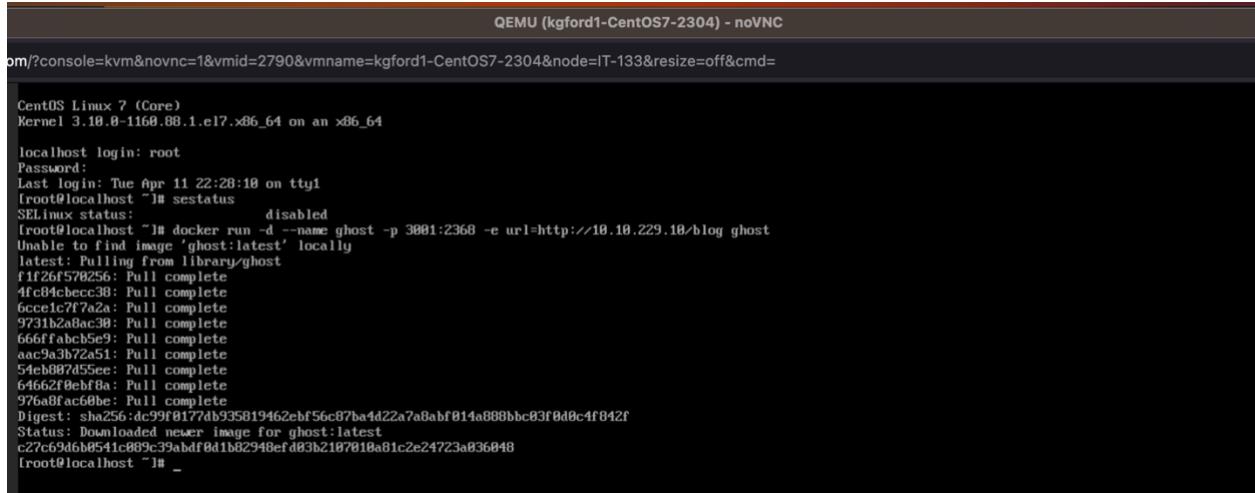
m?console=kvm&novnc=1&vmid=2790&vmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

CentOS Linux 7 (Core)
Kernel 3.10.0-1160.88.1.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Tue Apr 11 22:20:10 on ttys0
[root@localhost ~]# sestatus
SELinux status: disabled
[root@localhost ~]#
```

Install Ghost Docker Container

- **docker run -d --name ghost -p 3001:2368 -e url=http://10.10.229.10/blog ghost** runs the Ghost Docker container in detached mode with the name "ghost", mapping port 3001 on the host to port 2368 in the container, and setting the "url" environment variable to "<http://10.10.229.10/blog>".



```
QEMU (kgford1-CentOS7-2304) - noVNC

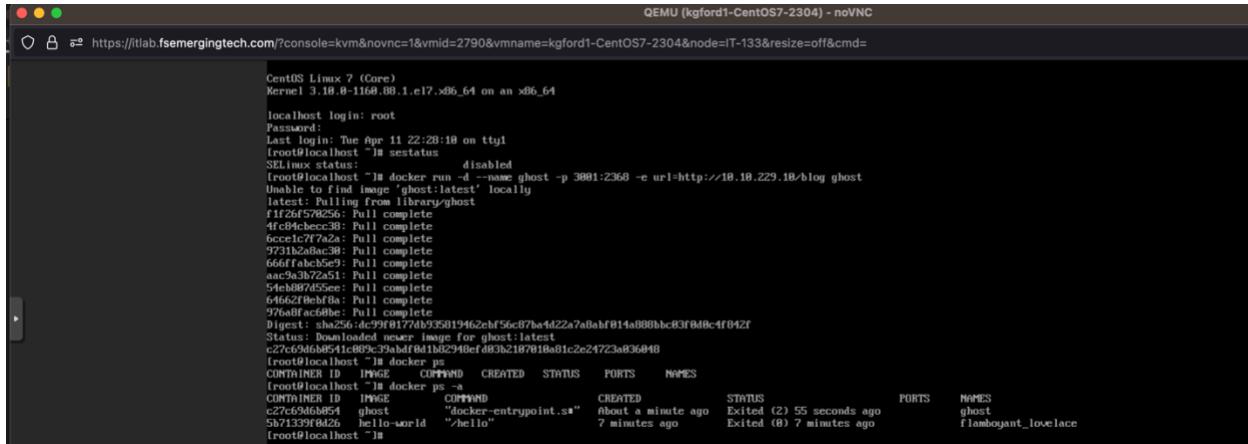
om/?console=kvm&novnc=1&vmid=2790&vmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

CentOS Linux 7 (Core)
Kernel 3.10.0-1160.88.1.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Tue Apr 11 22:28:18 on ttys0
[root@localhost ~]# sestatus
SELinux status:                 disabled
[root@localhost ~]# docker run -d --name ghost -p 3001:2368 -e url=http://10.10.229.10/blog ghost
Unable to find image 'ghost:latest' locally
latest: Pulling from library/ghost
f1f26f57b256: Pull complete
4fc84beccc38: Pull complete
6cce1c7f7a2a: Pull complete
9731b2a8ac38: Pull complete
666ffabch5e9: Pull complete
aac9a3b72a51: Pull complete
54eb007d55ee: Pull complete
64662f9ebfb8a: Pull complete
976a8fac0b0be: Pull complete
Digest: sha256:dc59f8177db35819462cbf56c07ba4d22a7a8abf014a888bbc03f0d0c4f842f
Status: Downloaded newer image for ghost:latest
c27cf946b8541c889c39abdf8d1b82948ef0d3b2107010a81c2e24723a036048
[root@localhost ~]# -
```

- **docker ps -a** lists all the containers that are currently running or have run on the system.

C27c69d6b054



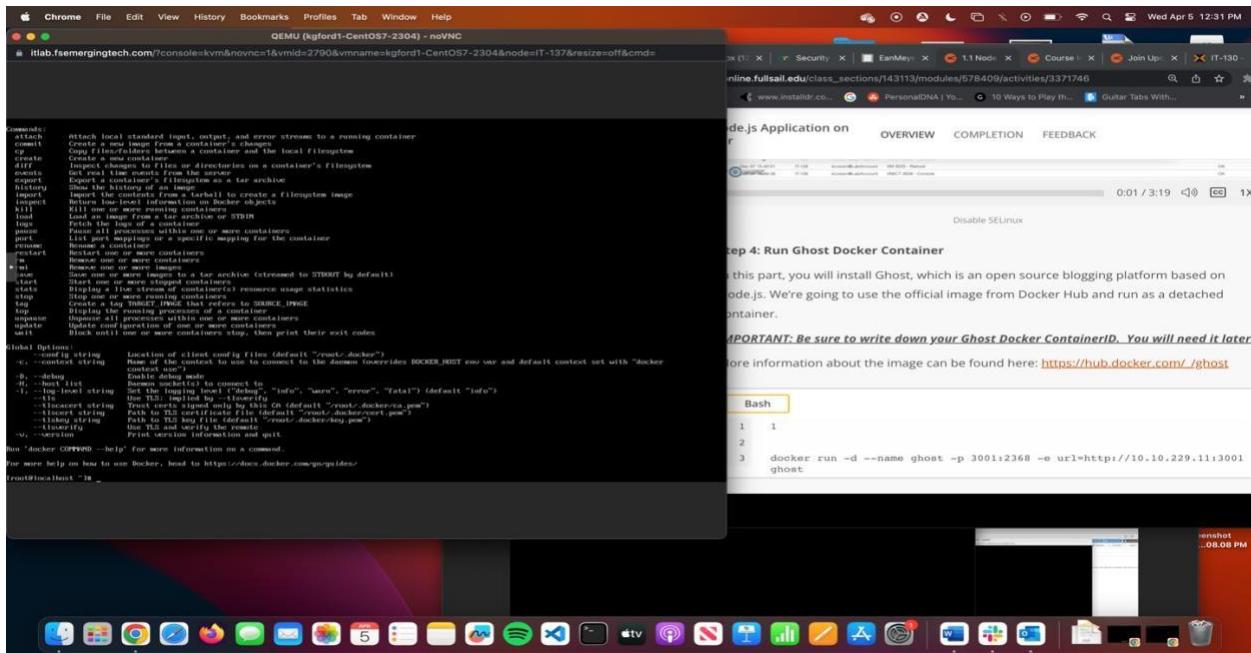
```
QEMU (kgford1-CentOS7-2304) - noVNC

O A https://itlab.fsemergingtech.com/?console=kvm&novnc=1&vmid=2790&vmname=kgford1-CentOS7-2304&node=IT-133&resize=off&cmd=

CentOS Linux 7 (Core)
Kernel 3.10.0-1160.88.1.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Tue Apr 11 22:28:18 on ttys0
[root@localhost ~]# sestatus
SELinux status:                 disabled
[root@localhost ~]# docker run -d --name ghost -p 3001:2368 -e url=http://10.10.229.10/blog ghost
Unable to find image 'ghost:latest' locally
latest: Pulling from library/ghost
f1f26f57b256: Pull complete
4fc84beccc38: Pull complete
6cce1c7f7a2a: Pull complete
9731b2a8ac38: Pull complete
666ffabch5e9: Pull complete
aac9a3b72a51: Pull complete
54eb007d55ee: Pull complete
64662f9ebfb8a: Pull complete
976a8fac0b0be: Pull complete
Digest: sha256:dc59f8177db35819462cbf56c07ba4d22a7a8abf014a888bbc03f0d0c4f842f
Status: Downloaded newer image for ghost:latest
c27cf946b8541c889c39abdf8d1b82948ef0d3b2107010a81c2e24723a036048
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
c27cf946b854        ghost               "docker-entrypoint.s"   About a minute ago   Exited (2) 55 seconds ago
5b71339fb426        hello-world       "/hello"           7 minutes ago      Exited (0) 7 minutes ago
[root@localhost ~]#
```

Test Ghost



NginX Reverse Proxy

Install base Rocky 8 Virtual Machine “ITE229-NginX (10)

1. Begin by downloading the Rocky Linux 8 ISO image from the official website. Ensure that you download the correct ISO image for your virtualization software.
2. Open your virtualization software and create a new virtual machine. Select "Linux" as the operating system and "Red Hat (64-bit)" as the version.
3. Assign a meaningful name to your virtual machine and allocate the necessary resources, such as memory and CPU cores. Ensure that the resources you allocate are sufficient for your intended use.
4. Create a new virtual hard disk with a size that is appropriate for your use case.
5. In the virtual machine settings, select the Rocky Linux 8 ISO image as the installation media.
6. Start the virtual machine and follow the installation wizard. Ensure that you select the appropriate language, time zone, and keyboard layout during the installation process.
7. Customize the installation settings as necessary, including partitioning, software selection, and network configuration.
8. Once the installation is complete, reboot the virtual machine and log in with the root account.
9. Update the system packages and install any necessary additional software.
10. Finally, ensure that you have taken all necessary security precautions, such as disabling unnecessary services and configuring firewalls.

By following these instructions in a professional manner, you can ensure that your base Rocky Linux 8 virtual machine is installed correctly and is ready for use.

SSH into Rocky 8 VM

1. Make sure that your Rocky 8 Linux virtual machine is running and is connected to the network.
2. Open a terminal or command prompt on your local machine.
3. Type the following command to connect to the virtual machine via SSH:

```
ssh [username]@[ip-address]
```

Replace **[username]** with the username that you want to use to log in to the Rocky 8 virtual machine, and replace **[ip-address]** with the IP address of the virtual machine.

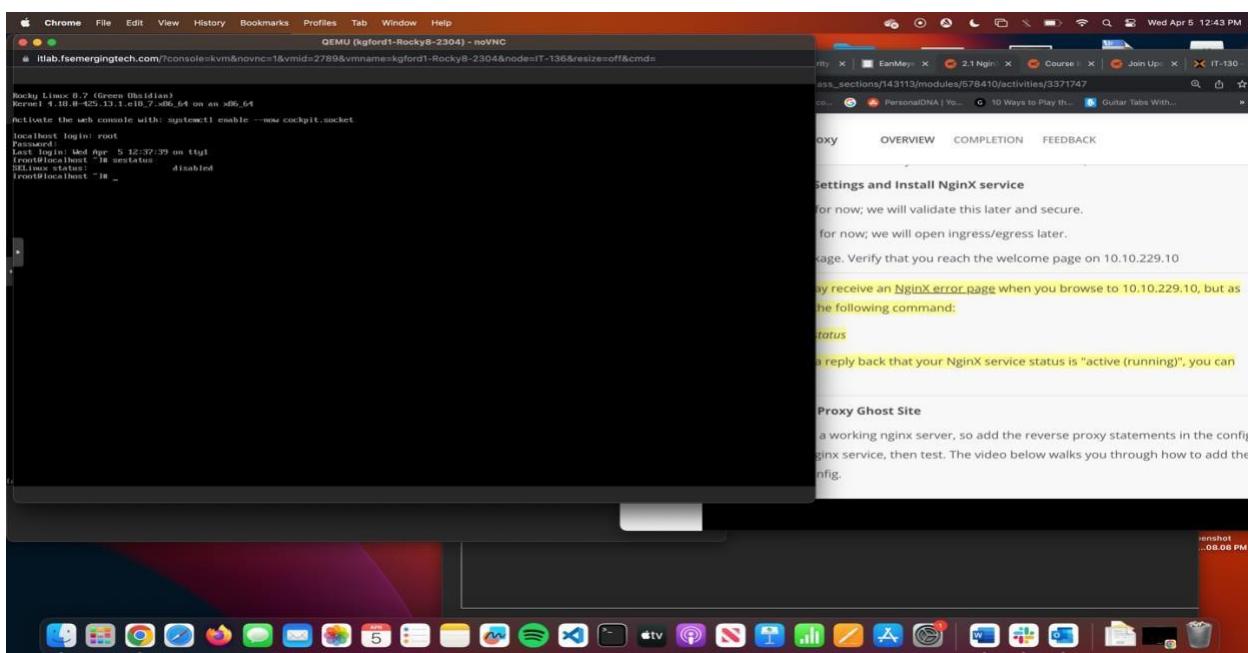
4. If this is the first time you are connecting to the virtual machine from your local machine, you will be prompted to confirm the authenticity of the host by checking the fingerprint of the SSH key. Verify the fingerprint matches the one from the virtual machine's key and type "yes" to continue.
5. Enter the password for the username when prompted.
6. You should now be logged into the Rocky 8 virtual machine via SSH and can run commands in the remote shell.

Update Rocky 8

- `yum update -y`: This command updates all packages on the system to their latest versions, and the `-y` flag automatically answers "yes" to any prompts or questions that may arise during the update process.
- `yum install nano`: This command installs the Nano text editor, which is a lightweight and user-friendly editor for editing configuration files on the system.

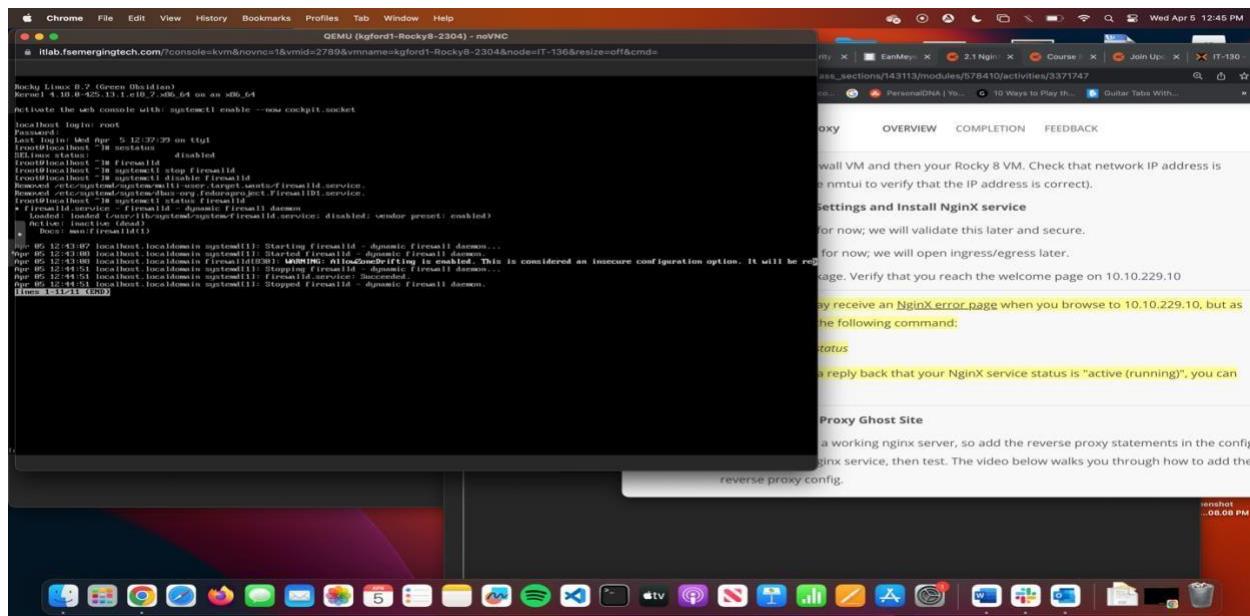
Disable SELinux

- `nano /etc/selinux/config`: This command opens the SELinux configuration file using the Nano text editor, which allows you to edit the configuration settings for SELinux.
- Within the SELinux configuration file, change the "Enforcing" parameter to "Disabled", save the changes, and reboot the system. The command `sestatus` is used to verify that the changes took effect and that SELinux is disabled.



Disable Firewall

- `systemctl status firewalld`: This command checks the status of the firewalld service, which is a firewall management tool for CentOS/RHEL systems.
- `systemctl stop firewalld` and `systemctl disable firewalld`: These commands stop and disable the firewalld service, which temporarily disables the firewall on the system.

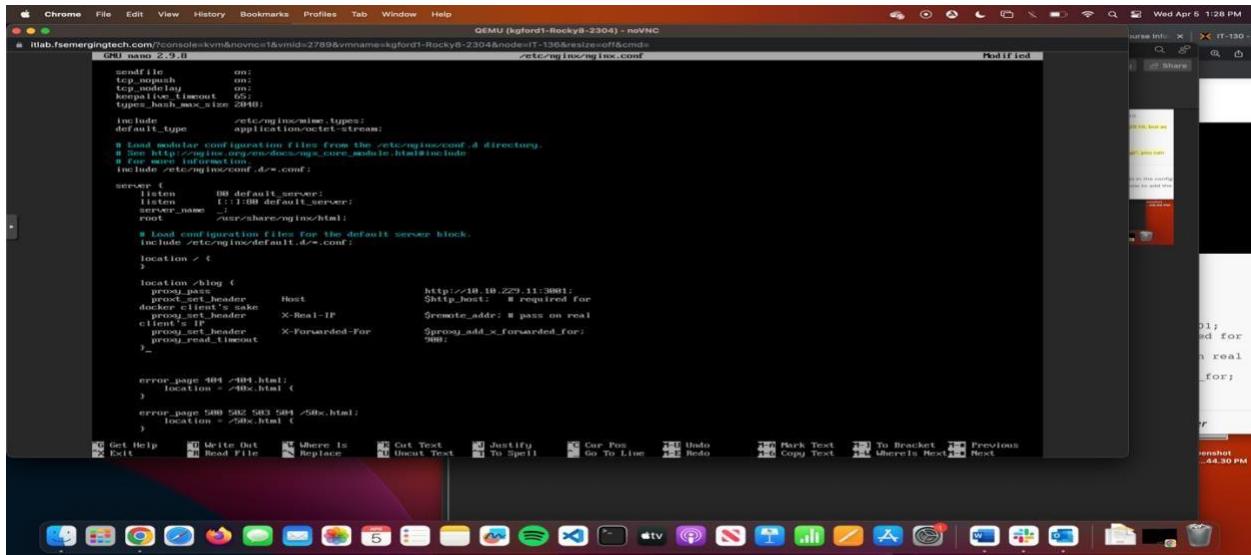


Install EPEL Packages

- `yum install epel-release -y`: This command installs the Extra Packages for Enterprise Linux (EPEL) repository, which provides additional packages that are not included in the default CentOS/RHEL repositories.

Install NginX

- `yum install nginx -y`: This command installs the Nginx web server, which is a popular open-source web server used for serving web content and reverse proxying.



```
GNU nano 2.9.0          /etc/nginx/nginx.conf
include /etc/nginx/mime.types;
default_type application/octet-stream;
# Load modular configuration files from the /etc/nginx/conf.d directory.
# See http://nginx.org/en/docs/http/ngx_http_dseal_module.html#include
# For more information about these files see:
include /etc/nginx/conf.d/*.conf;

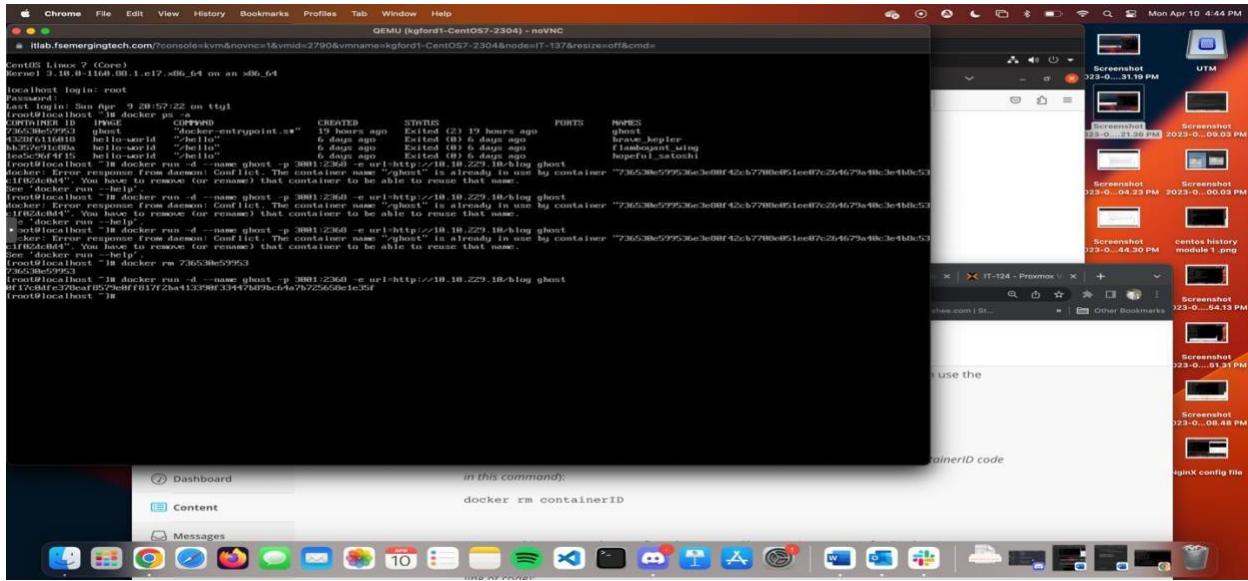
server {
    listen 80 default_server;
    listen [::]:1080 default_server;
    server_name ~*^([^.]+)\.([^.]+\.[^.]+)$;
    root /var/share/nginx/html;
    # load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    location / {
        proxy_pass http://10.10.229.11:38801;
        proxy_set_header Host $http_host; # required
        proxy_set_header X-Real-IP $remote_addr; # pass on real
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 900;
    }
}

error_page 404 /404.html;
location = /404.html {
}
error_page 500 502 503 504 /50x.html;
location = /50x.html {
}
```

Start and Enable NginX

- `systemctl enable nginx` and `systemctl start nginx`: These commands enable and start the Nginx web server service.
 - `service nginx status`: This command checks the status of the Nginx web server service, verifying that it is running.
 - `nano /etc/nginx/nginx.conf`: This command opens the Nginx configuration file in the Nano text editor, allowing you to add or modify configuration settings for the web server.
 - Within the Nginx configuration file, add the reverse proxy statements for the Ghost site, save the changes, and reload the Nginx service. The reverse proxy configuration allows Nginx to act as an intermediary between the client and the Ghost application running on a separate server

Reverse Proxy to Ghost Site



Install WordPress on Ubuntu - LAMP Stack

Base Ubuntu 18.04 Install

1. Begin by downloading the ISO file for Ubuntu 18.04 LTS from the official Ubuntu website.
2. Install virtualization software like Oracle VM VirtualBox or VMware Workstation.
3. To create a new virtual machine, launch the virtualization software and press the "New" button.
4. Choose "Linux" as the operating system and "Ubuntu (64-bit)" as the version for the virtual machine.
5. Give the virtual machine at least 2 GB of RAM and create a new virtual hard disk with a minimum size of 20 GB.
6. Choose "Use an existing virtual hard disk file" and then navigate to the Ubuntu 18.04 LTS ISO file.
7. Configure the virtual machine's settings to meet your needs, such as the number of processors and network settings.
8. Launch the virtual machine and choose "Install Ubuntu" from the boot menu.

9. Select your preferred language and press the "Continue" button.

10. Choose your keyboard layout and press the "Continue" button.

11. Select "Minimal installation" and click "Continue" on the "Updates and Other Software" screen.

12. Select "Erase disk and install Ubuntu" and click "Continue" on the "Installation Type" screen.

13. Select the virtual disk on which you want to install Ubuntu and press the "Install Now" button.
14. Click "Continue" on the pop-up window to confirm the changes.
15. Click "Continue" after selecting your time zone.
16. Enter your user information, which should include your name, computer name, username, and password. When finished, press the "Continue" button.

17. Ubuntu will now begin the installation process. This could take some time.

18. Restart the virtual machine once the installation is complete.

Set Static IP

1. Ctrl+Alt+T will launch a terminal in your Ubuntu virtual machine.
2. To edit the network configuration file, enter the command below:
`sudo nano /etc/netplan/01-netcfg.yaml`
3. Find the section for your network interface in the file. It will probably have the name "eth0" or "ens33" and should be listed under "network".
4. The section should be changed to include a static IP address.
5. By pressing Ctrl+X, Y, and Enter, you can save and close the document.
6. By entering the following command, you can apply the new network configuration:
`sudo netplan apply`
7. By entering the following command, you can check to see if the new IP address has been set:
`ip address display`
8. Under the network interface you changed, you should see the new IP address listed.

SSH into Ubuntu VM

Open the Terminal

- `ssh username@server_ip_address:`

use your user name and the the remote Server's IP address in the command above

This allows remote access to the Ubuntu machine.

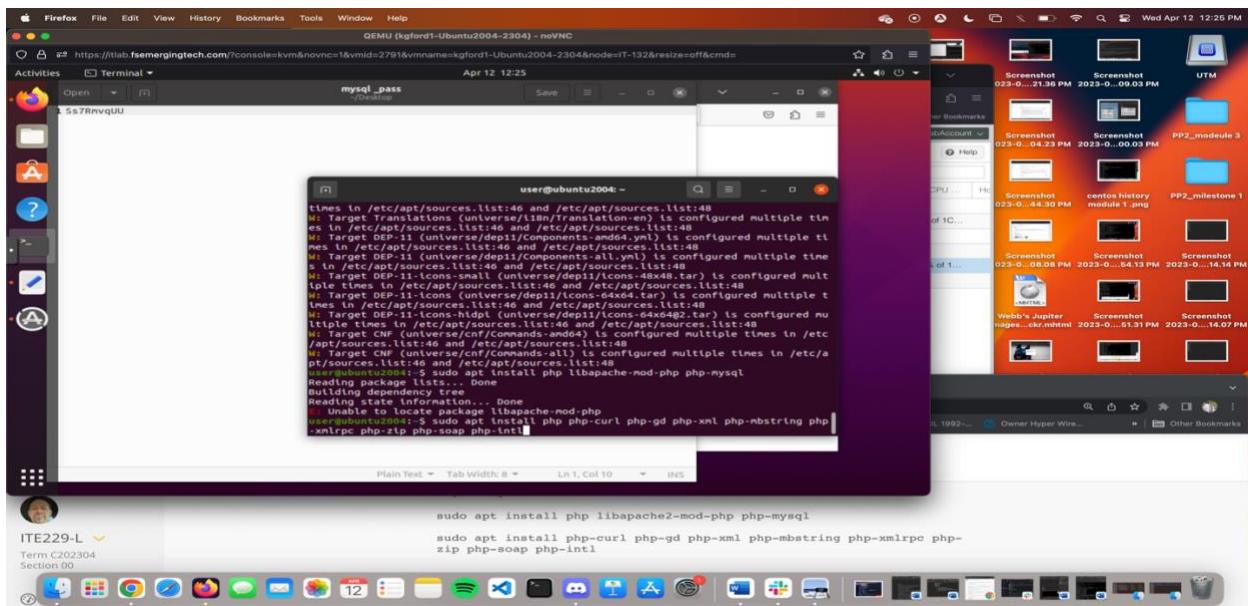
Update Ubuntu

- `sudo apt-get update`
- `sudo apt upgrade`

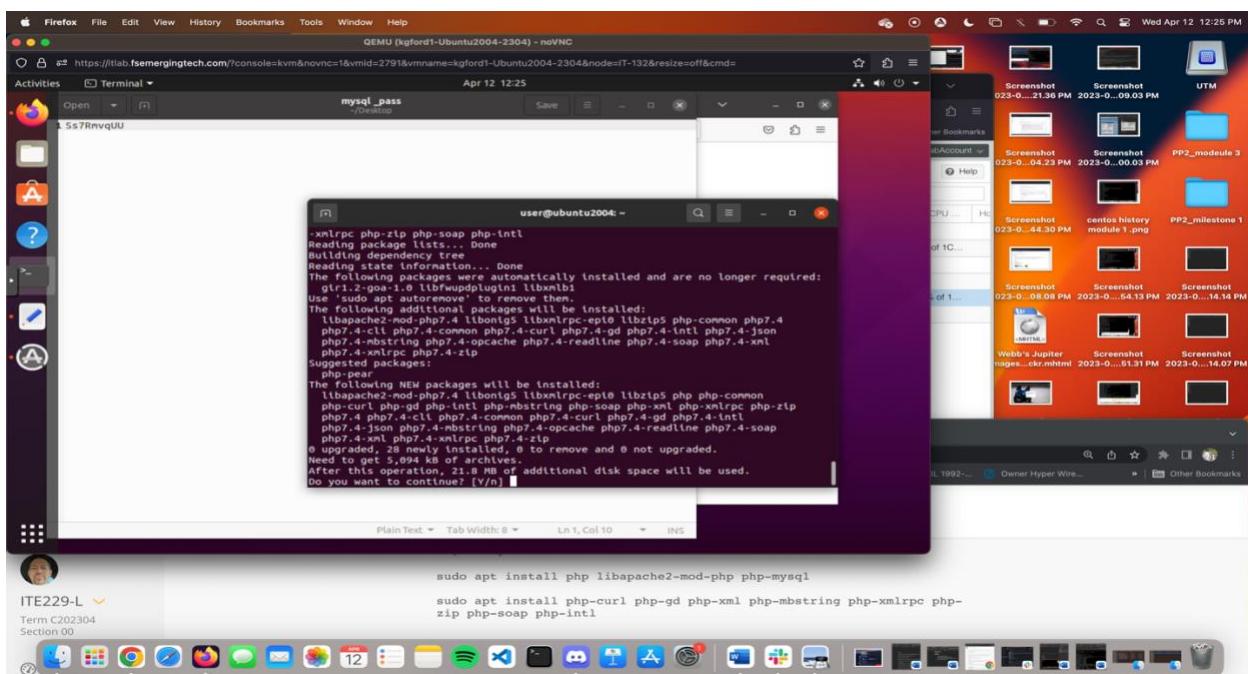
The first command lists upgrades for the installed packages, and the second command updates those upgrades to the most recent versions.

Install and Configure Apache

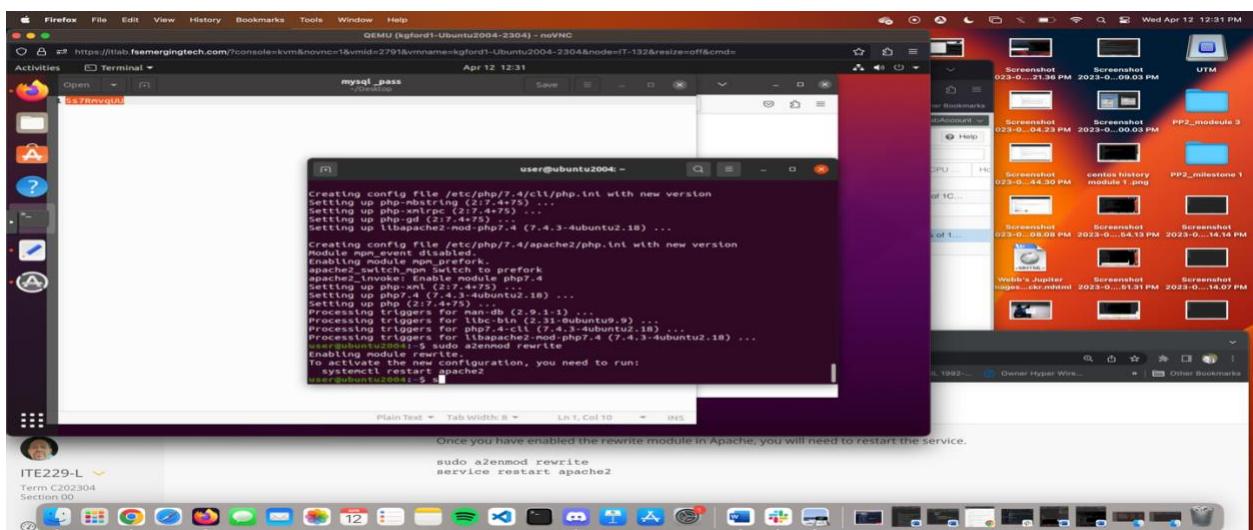
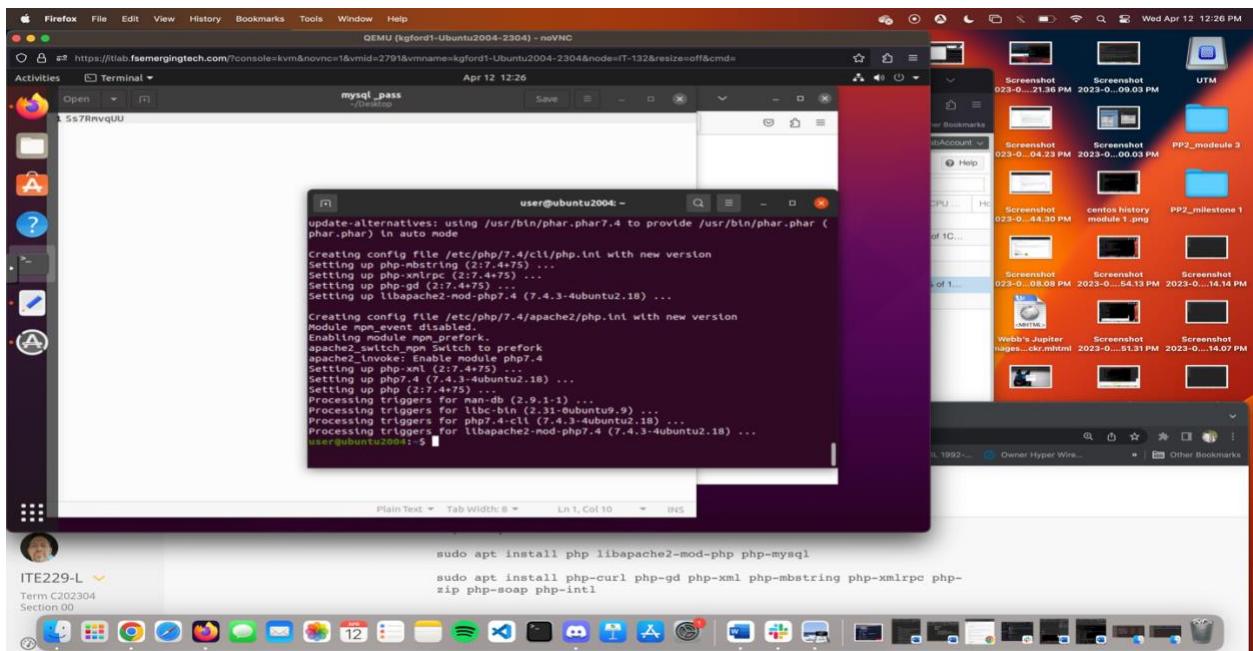
- `sudo apt install apache2`: Command to install packages for Apache web server



Say yes to install packages



Installing packages

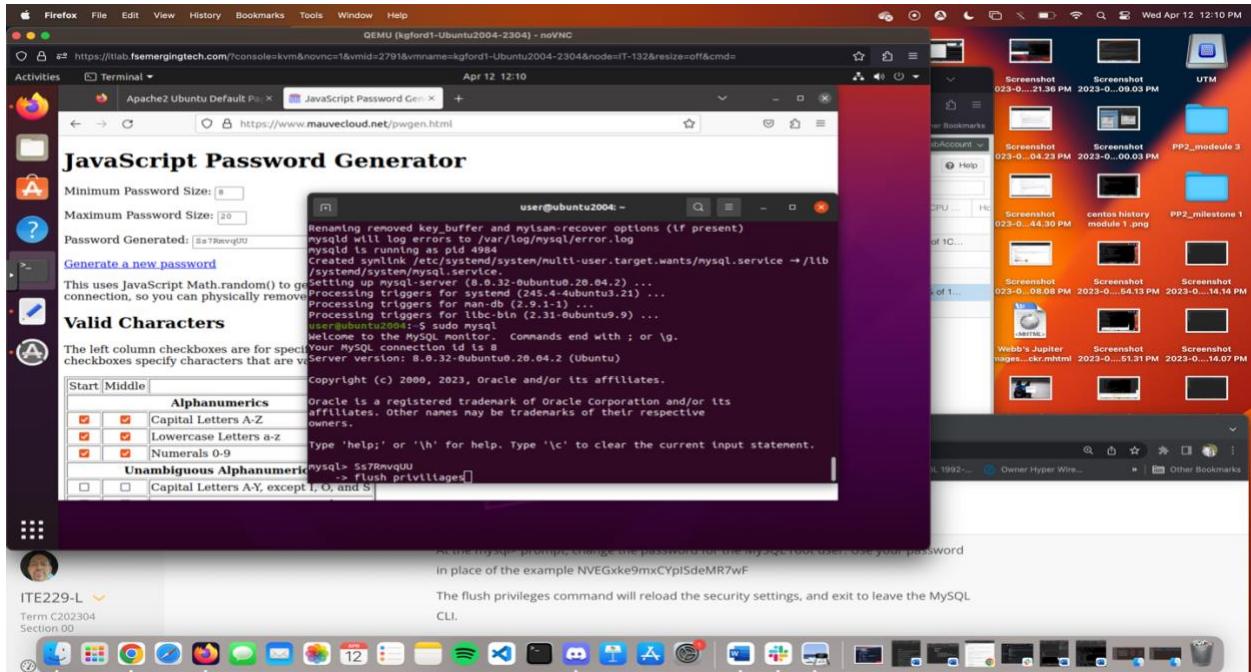


Install and Configure MySQL

- sudo apt install mysql-server:

This command installs the MySQL package

Create a complex password for MySQL root user:



- sudo mysql: This command opens the MySQL command-line interface
- ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'new_password';

FLUSH PRIVILEGES;

Exit; :

Replace **new_password** with the password you created in the previous step. The first command changes the password for the MySQL root user, and the second command reloads the security settings. The **exit** command exits the MySQL CLI.

- sudo nano /etc/apt/sources.list:

This command opens the **sources.list** file in the Nano text editor. Add the keyword **universe** to the end of each line that doesn't already have it. Then save and exit.

- sudo apt update:

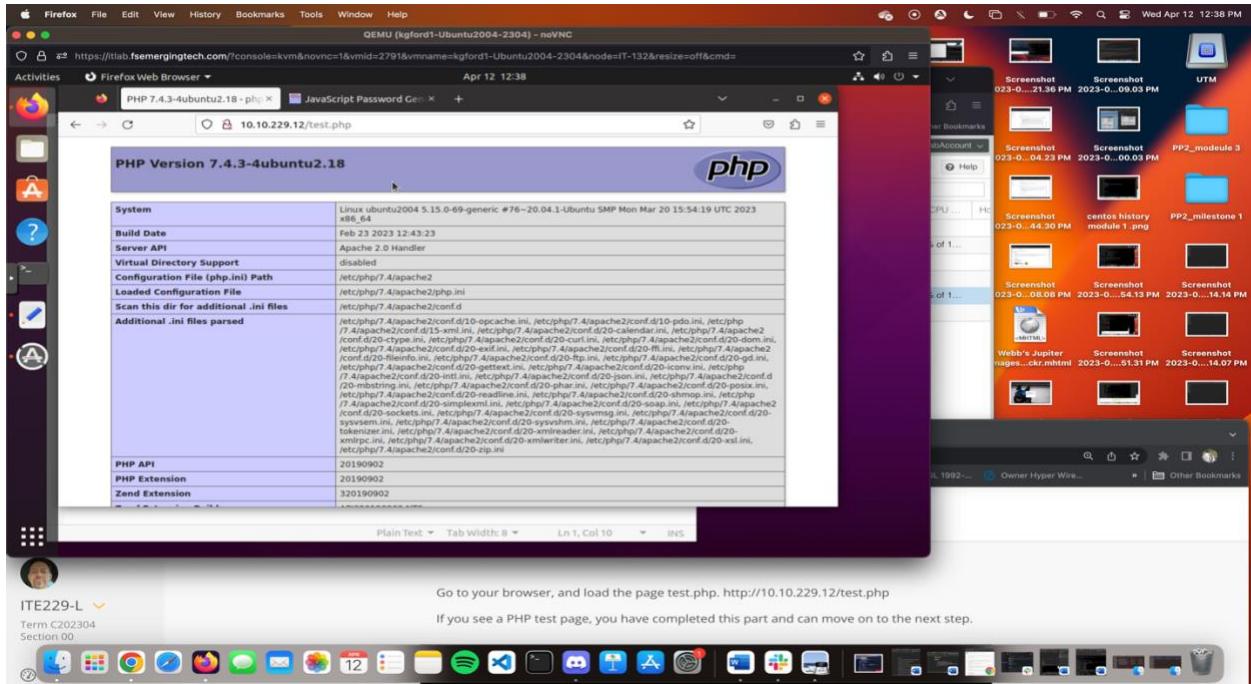
This command updates the package lists to include the newly added repository.

Install and configure PHP

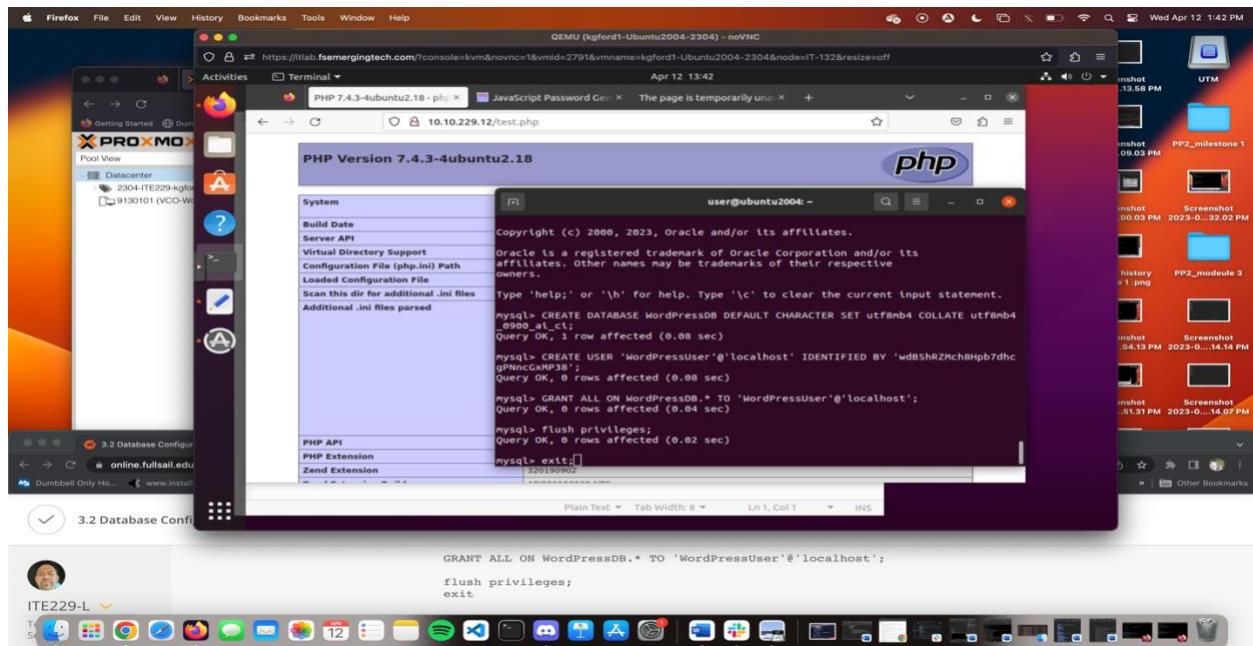
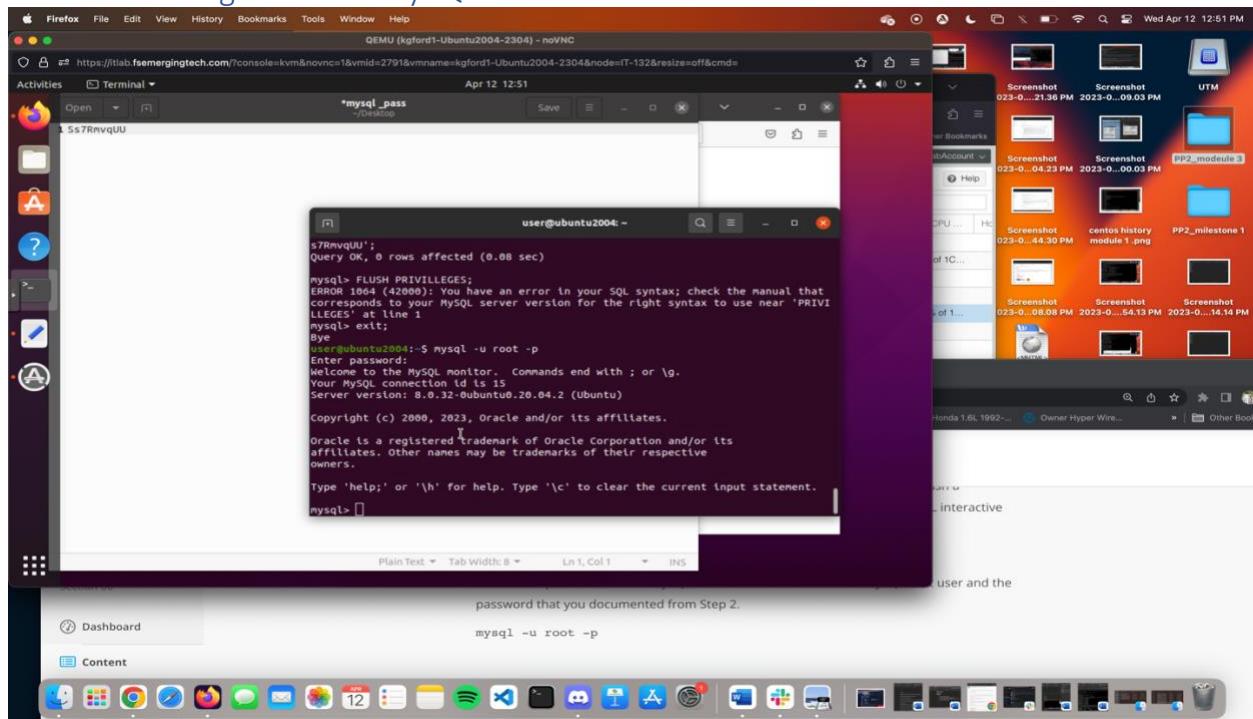
- sudo apt install php libapache2-mod-php php-mysql
- sudo apt install php-curl php-gd php-xml php-mbstring:

installs PHP and the required extensions for WordPress to work.

Test PHP



Database Configuration in MySQL



Install WordPress

WordPress Install using git client, from GitHub:

Empty the /var/www/html directory, then clone the git repository into /var/www/html

- `sudo chown $USER:$USER /var/www/html/*`: removes all files and folders inside the /var/www/html directory.
- `sudo rm /var/www/html/*`

Verify that your html directory is empty:

```
sudo ls /var/www/html/
```

Install git:

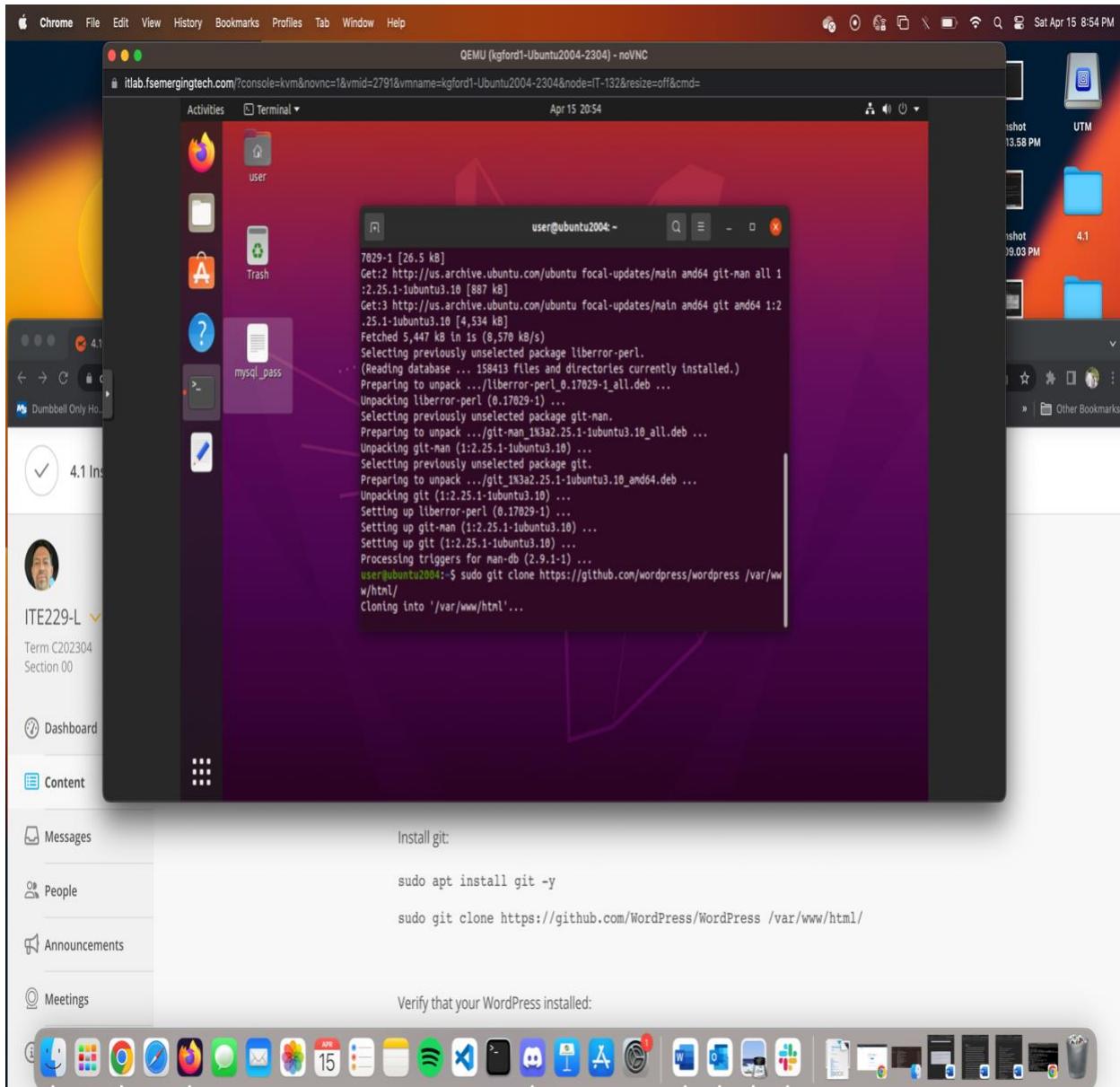
- `sudo apt install git -y`

Clone WordPress

- `sudo git clone https://github.com/WordPress/WordPress /var/www/html/`: clones the WordPress repository from GitHub into the /var/www/html directory.

Verify that your WordPress installed:

- `sudo ls /var/www/html`: verifies that WordPress is installed in the /var/www/html directory



Edit Ownership

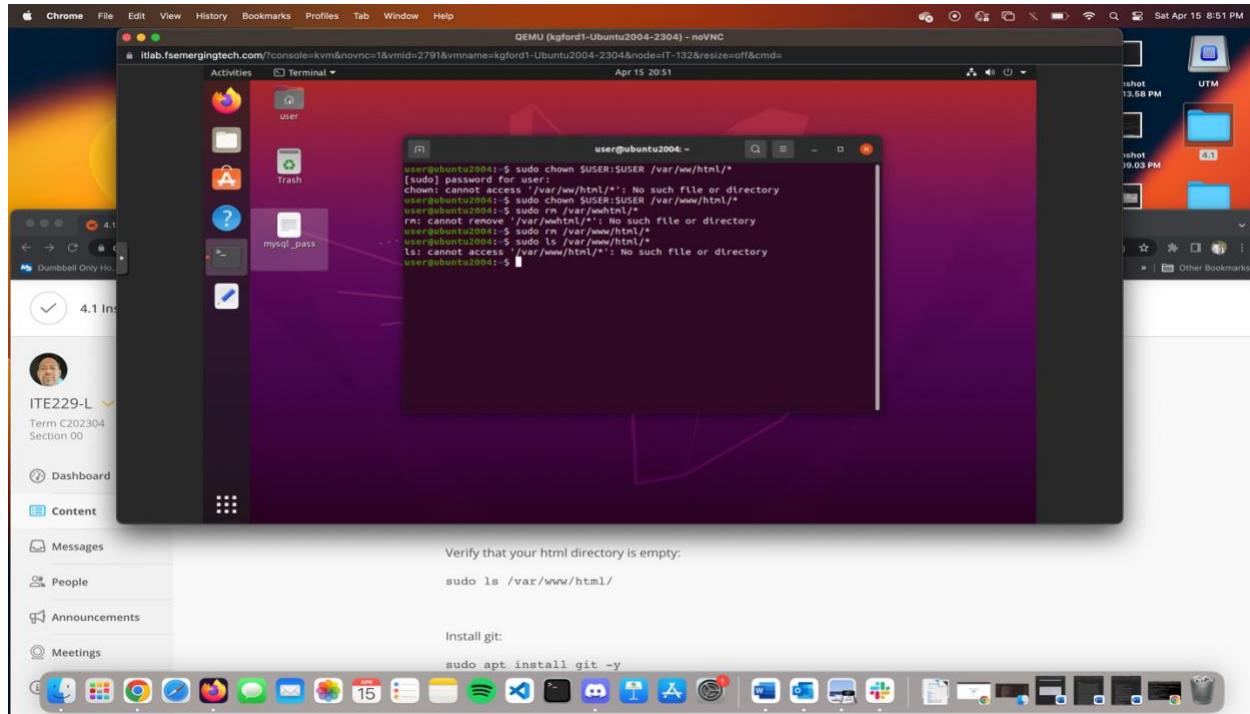
Verify permissions:

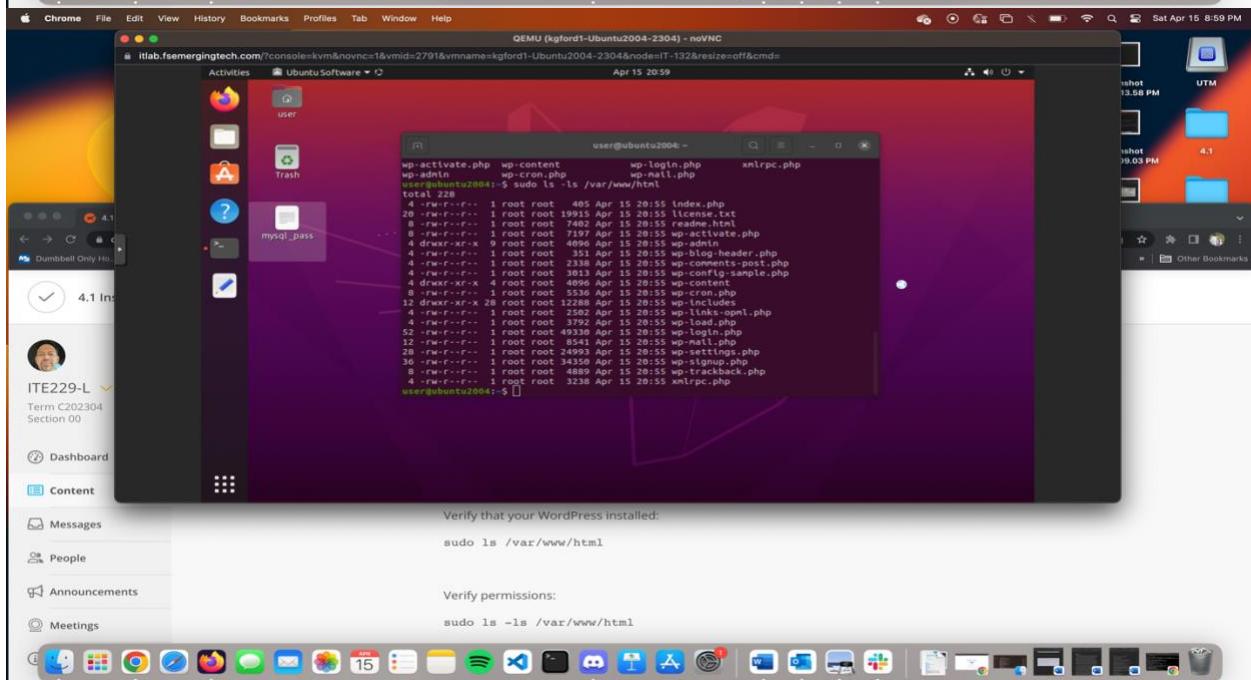
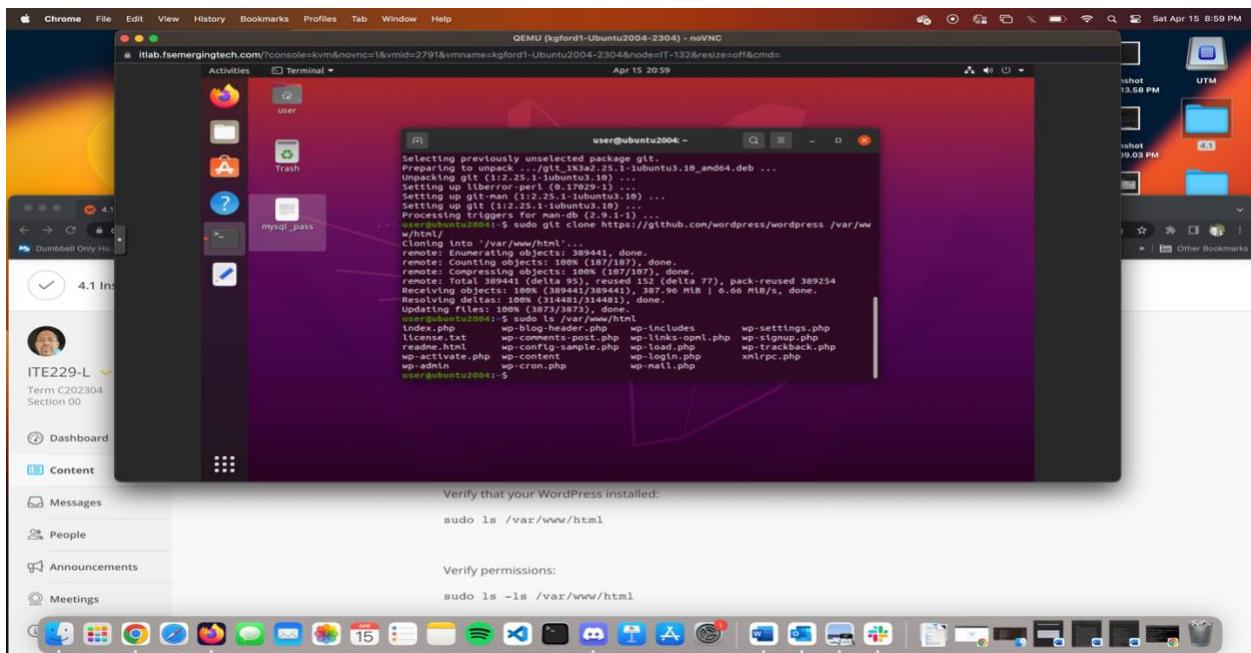
- `sudo ls -ls /var/www/html`: lists the contents of the `/var/www/html` directory with permissions to verify that the ownership is set correctly.

Set the ownership of the /var/www/html contents to the Apache user, and set ownership on the html directory itself.

- `sudo chown -R www-data:www-data /var/www/html/*`: sets the ownership of all files and folders inside the `/var/www/html` directory to the Apache user.

- sudo chown www-data:www-data /var/www/html: sets the ownership of the /var/www/html directory to the Apache user





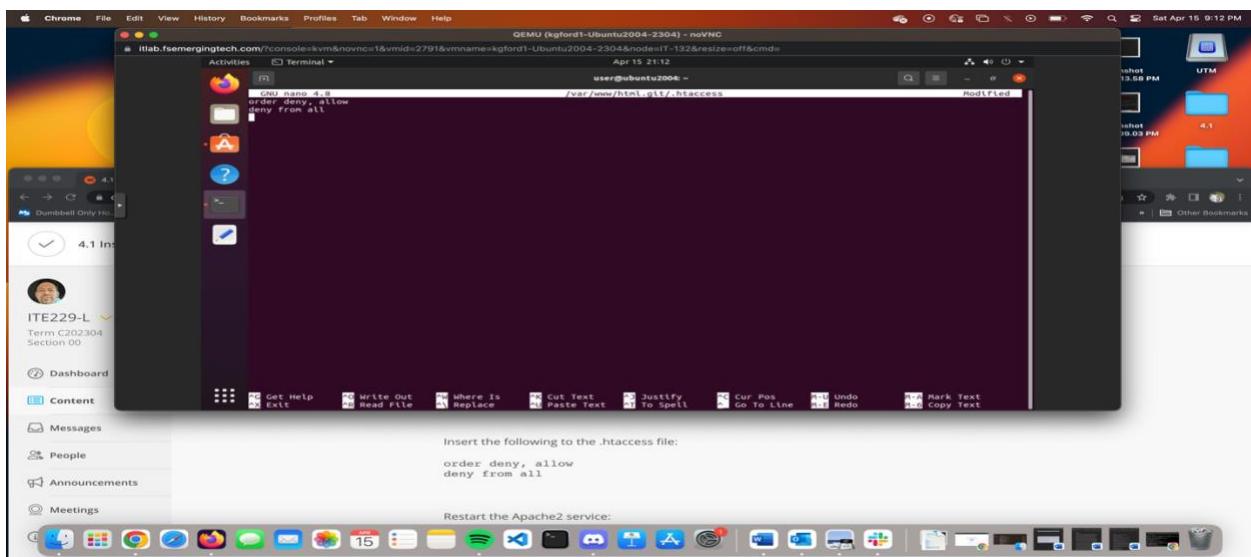
Edit .htaccess

- sudo nano /var/www/html/.git/.htaccess: creates a .htaccess file under the .git folder

Insert the following lines to the .htaccess file to secure the .git directory from unauthorized access:

order deny,

allow deny from all

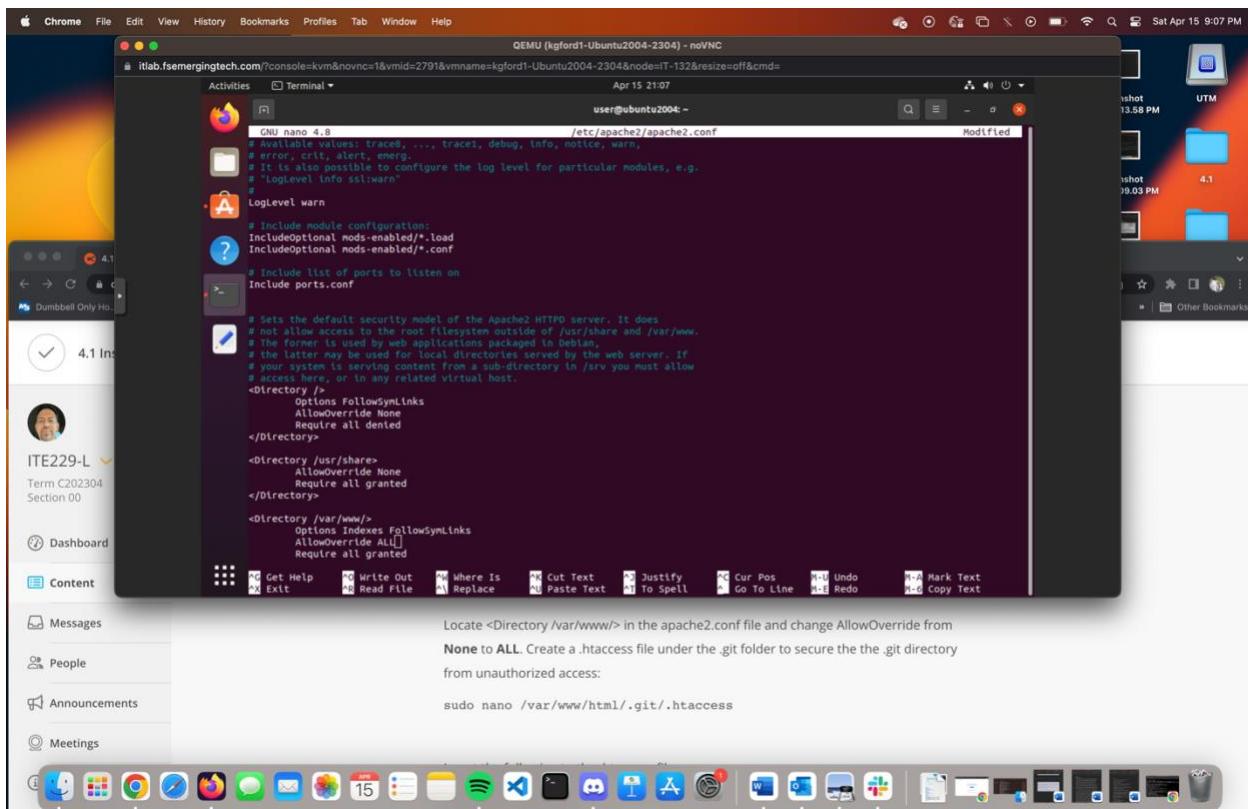


WordPress Configuration

Enable overrides for the /var/www path. Open the /etc/apache2/apache2.conf file.

- sudo nano /etc/apache2/apache2.conf: opens the global Apache configuration file in the Nano text editor

Change AllowOverride from None to ALL for "Directory /var/www" in the apache2.conf file. This allows overrides for the /var/www path.

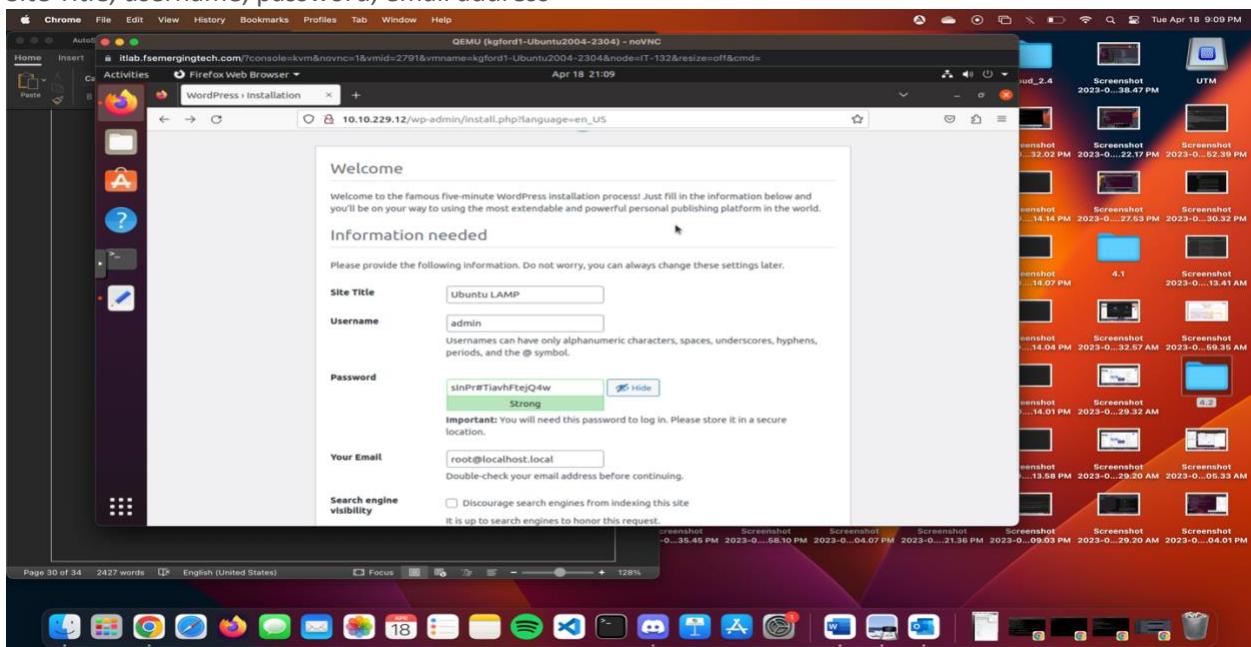


- sudo service apache2 restart: restarts the Apache2 service to implement the configuration changes

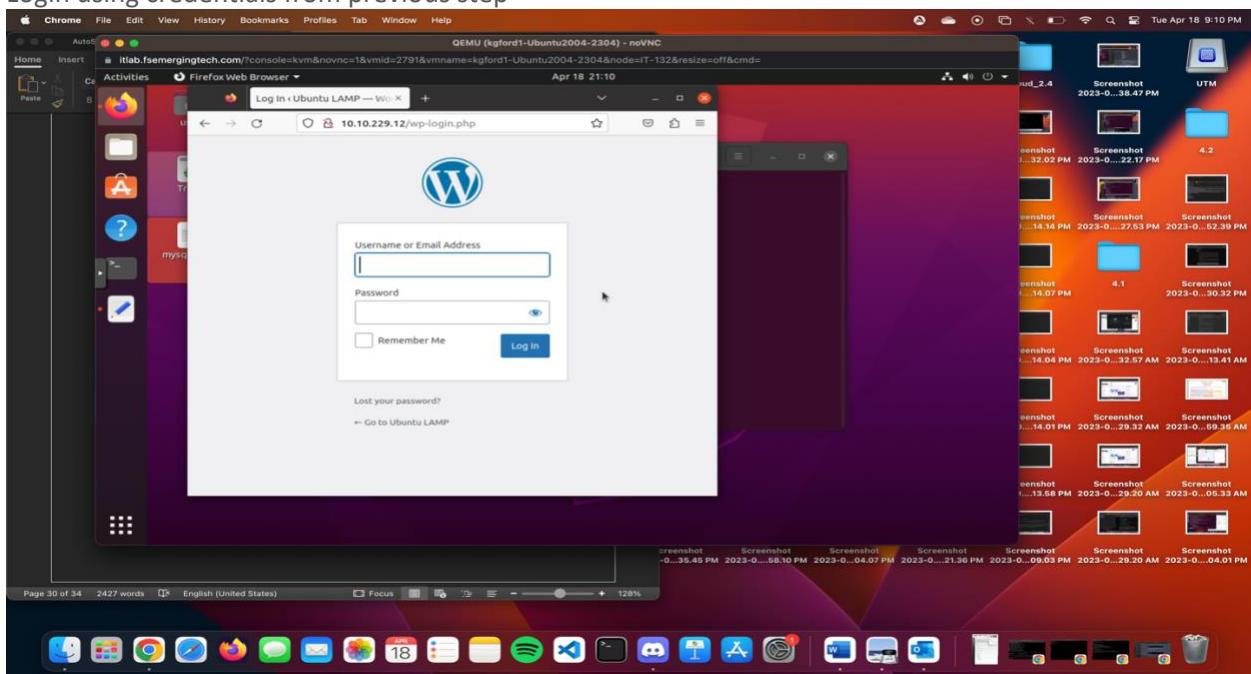
WordPress Configuration Process

Input:

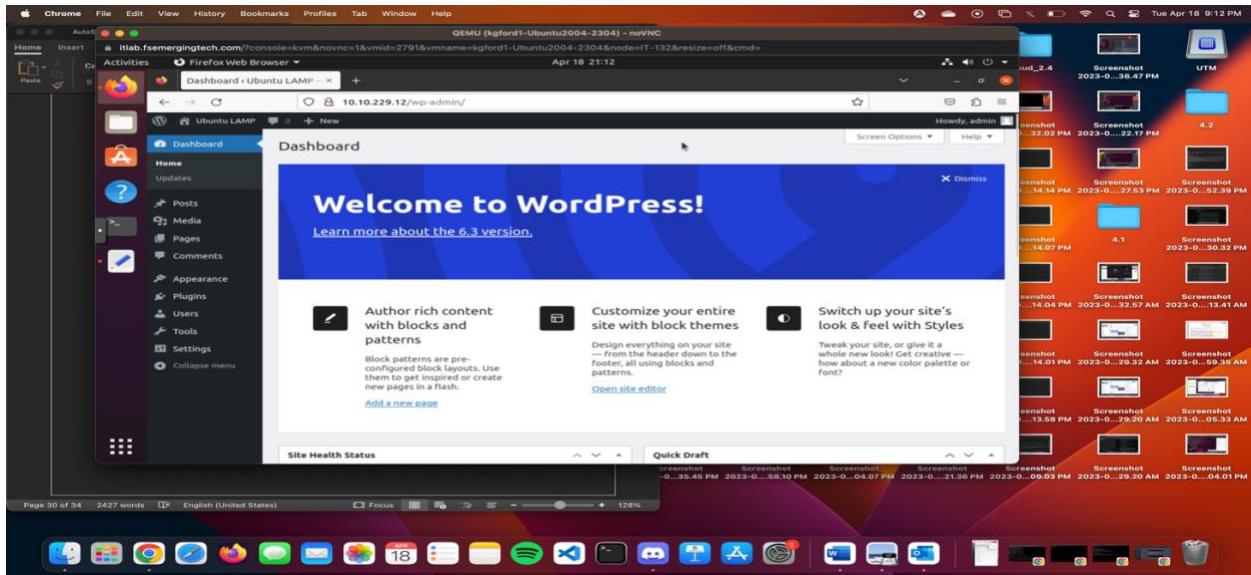
Site Title, username, password, email address



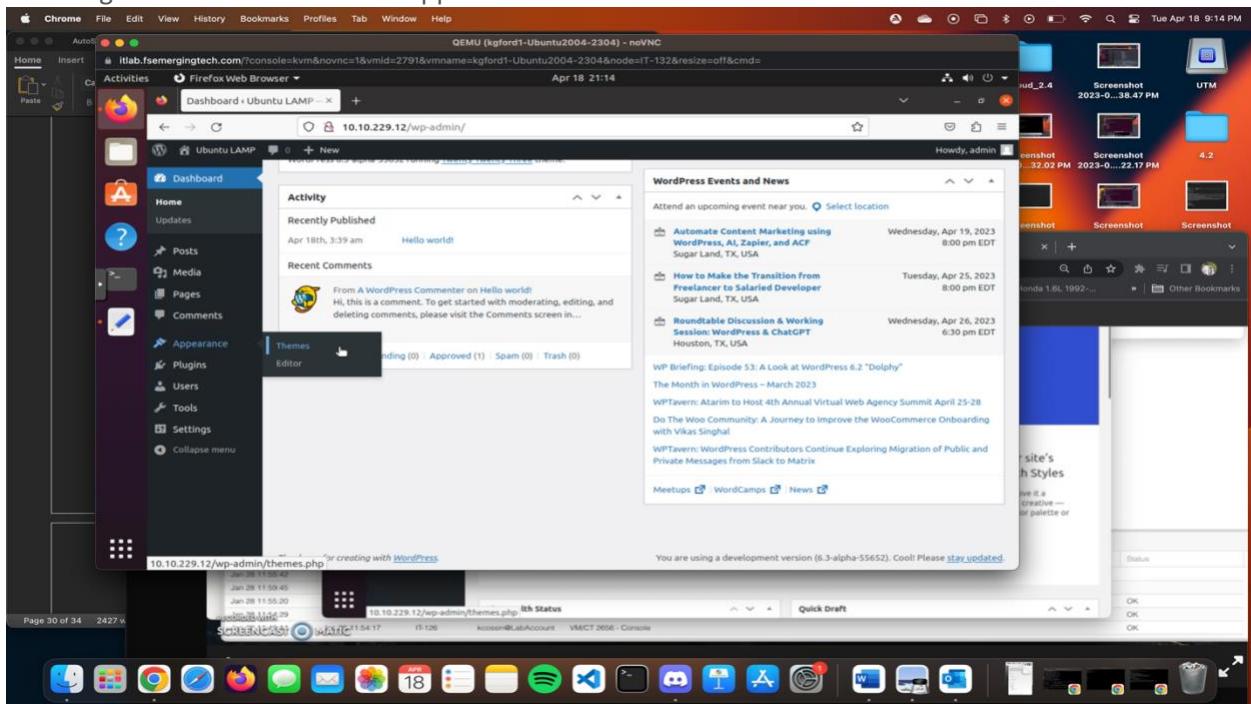
Login using credentials from previous step



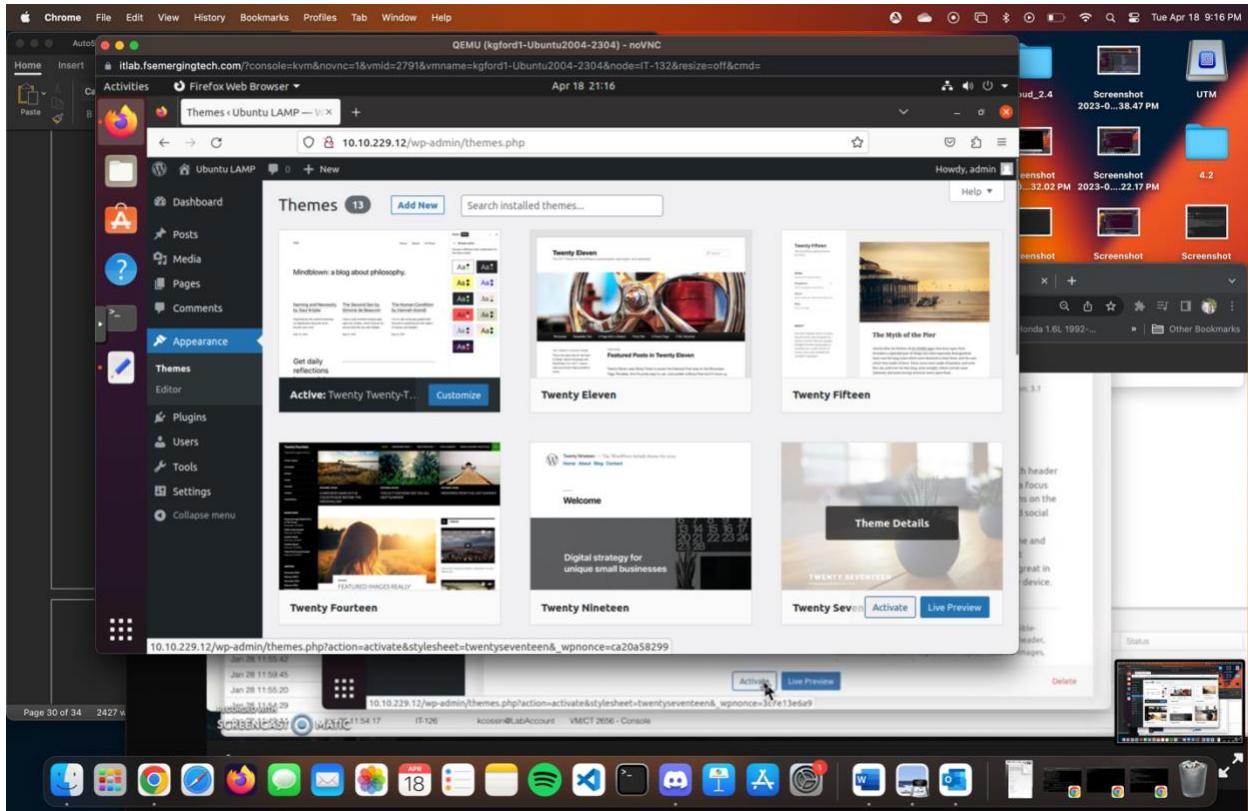
Welcome to WordPress



To change the theme Hover over Appearance and click Themes

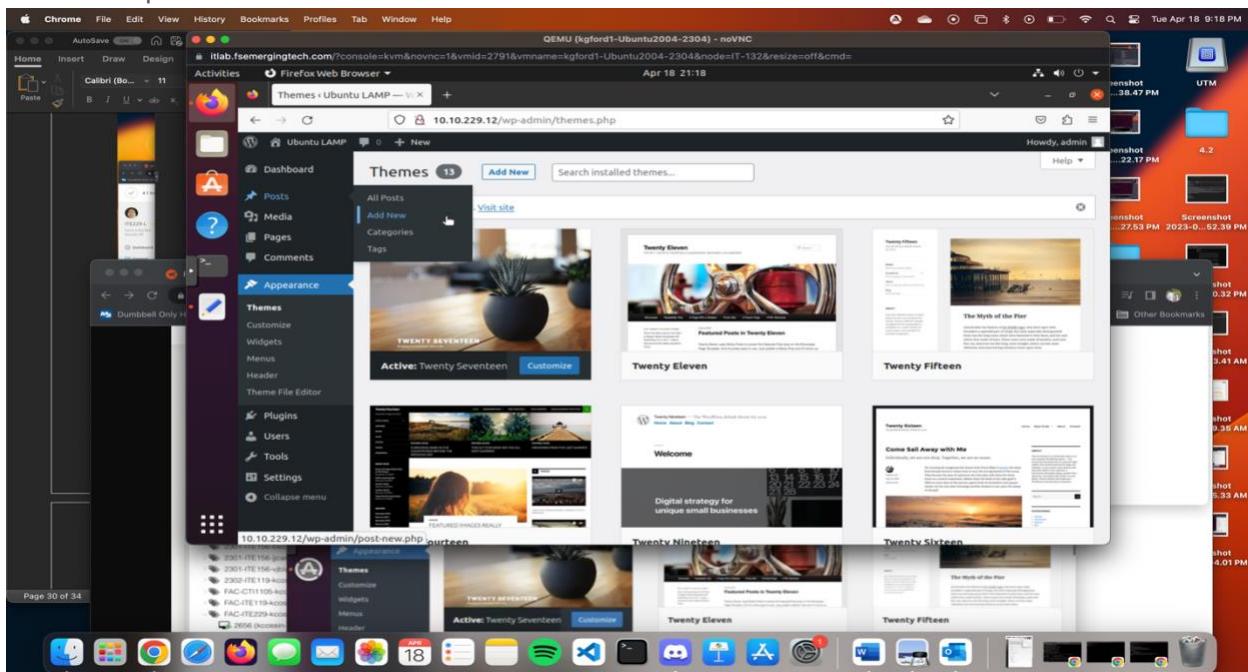


To Apply a theme, click Activate

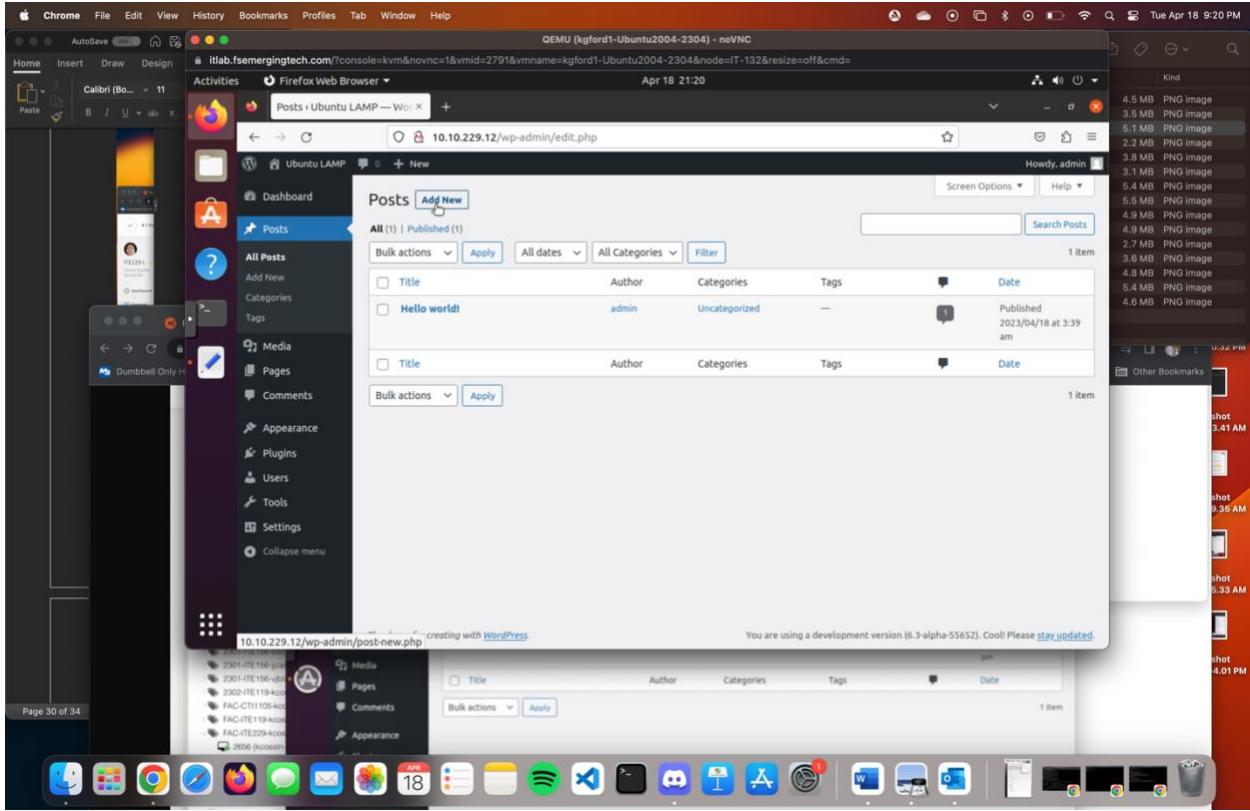


Test WordPress

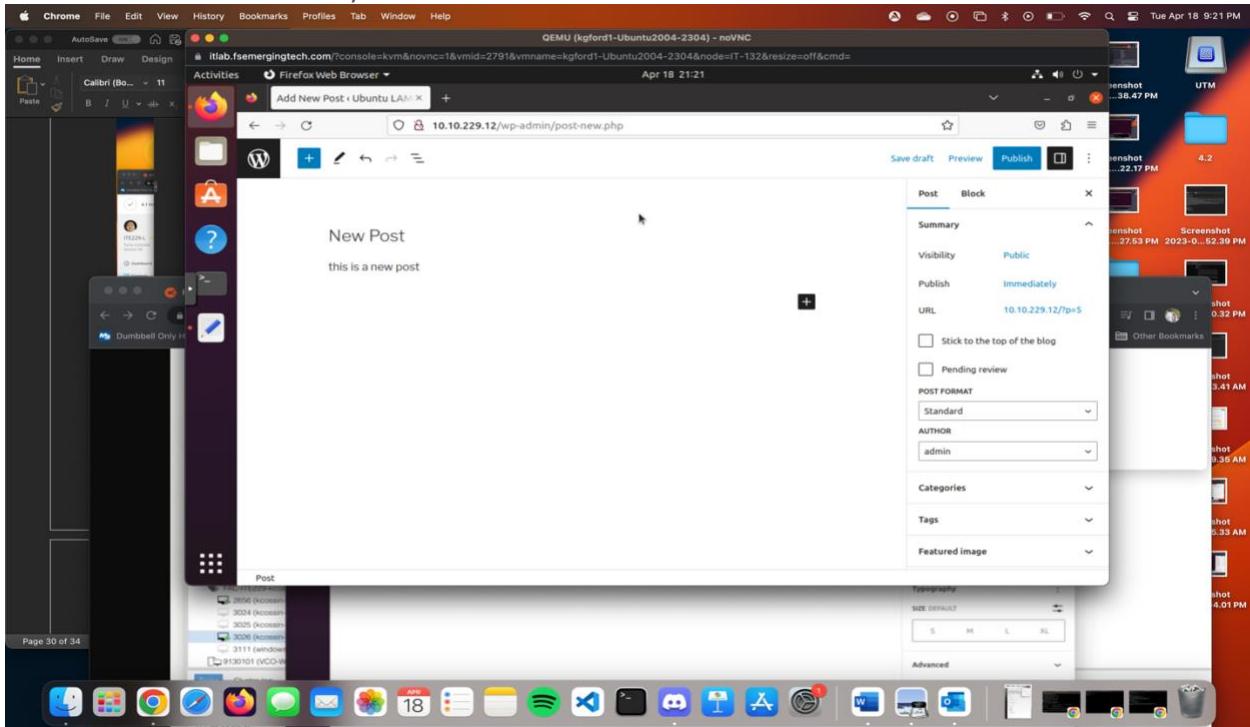
To create a post: Hover over Post and click Add New



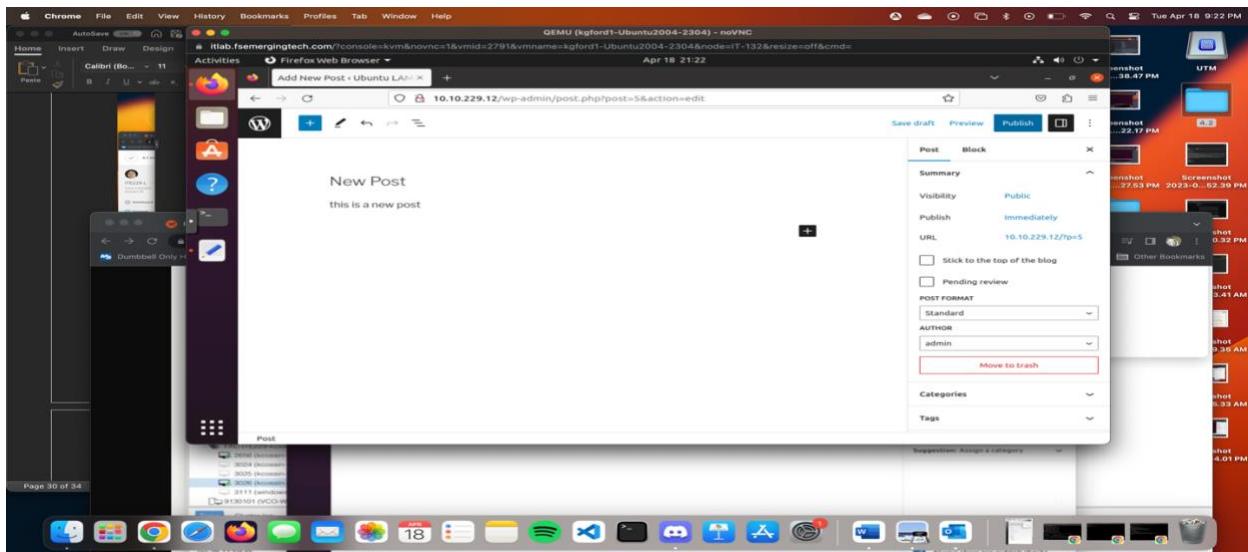
Or Hover over Post, Click All Post, Then Click Add New



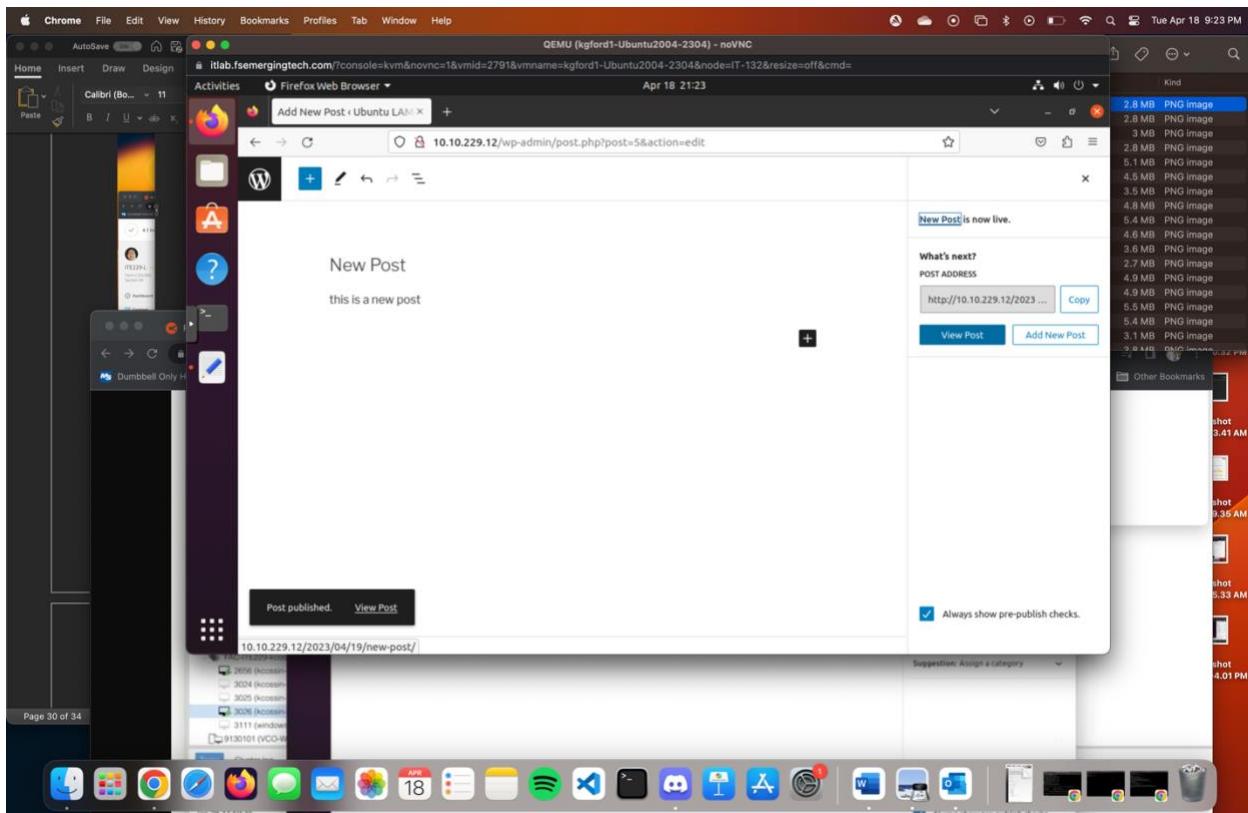
Add text to the Title and Body sections of the Post



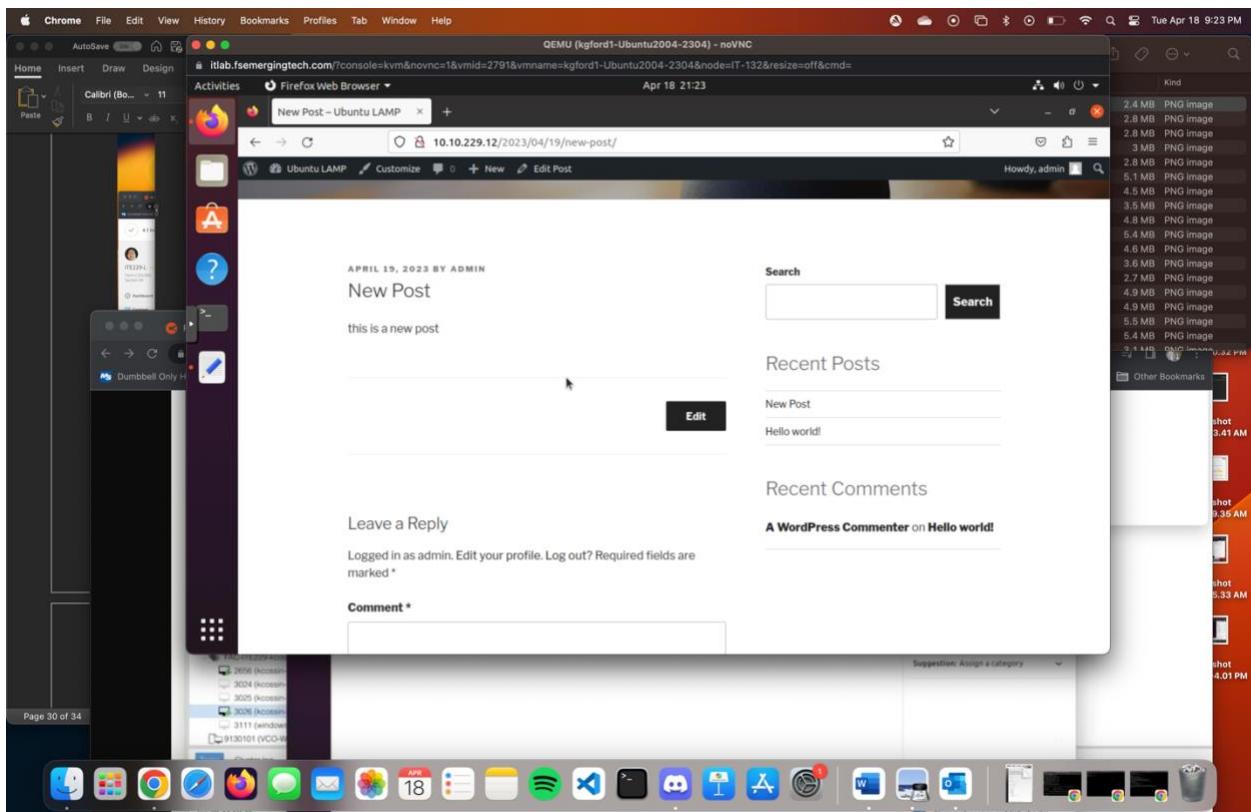
Then Click Publish



Click View Post



Next visit the URL for the WordPress site using a web browser



WordPress Security Settings and Configurations

Security Summary

The security of the WordPress installation has been significantly improved. The following changes have been made:

- File permissions have been updated to restrict access to only the owner and group.
- The WP-Config.php file has been moved to a directory outside the root directory.
- The Shield firewall has been installed and configured.

These changes have made it more difficult for unauthorized users to access sensitive files and directories. They have also helped to protect the WordPress installation from known vulnerabilities and make it more difficult for attackers to gain unauthorized access.

Defense-in-depth

Defense-in-Depth is the idea of using multiple layers of security to shield a system from different types of attacks. I used Defense-in-Depth in this exercise to harden the WordPress installation by putting various security controls in place. Changing file permissions, securing the WP-Config.php file, and installing and configuring the Shield firewall are a few examples of the multiple layers of security measures I used. Each of these steps adds another layer of defense against unauthorized entry and potential attacks.

I have significantly improved the security of the WordPress installation by putting in multiple layers of security measures, making it much more difficult for potential attackers to access the system. The Defense-in-Depth strategy helps to ensure that there are additional layers of security in place to deter a successful attack even if one security measure is circumvented or fails.

Before/After Configuration

steps taken for each of the vulnerabilities checked and the changes made to secure them:

1) Directory and file Permissions:

Before: Accessed the server and ran the command ls -la to view the permissions of the WordPress files and directories.

2) Change File Permissions:

Before: Accessed the server using SSH and ran the command ls -la to view the permissions of a WordPress file.

After: After: Changed permissions of directories to 750 and files to 640 using the command find /path/to/wordpress -type d -exec chmod 750 {} \; and find /path/to/wordpress -type f -exec chmod 640 {} \;.

3) Securing WP-Config.php:

Before: Accessed the server using SSH and located the WP-Config.php file in the WordPress root directory.

After: Moved the WP-Config.php file to a directory outside the root directory and changed its permissions to 400 using the command mv /path/to/wordpress/wp-config.php /path/to/non-public/directory/ && chmod 400 /path/to/non-public/directory/wp-config.php.

4) Firewalls select Shield:

Before: Accessed the WordPress admin dashboard and installed the Shield Security plugin.

After: Configured the Shield firewall by enabling all recommended settings and adding custom rules for specific IPs and user agents.

For each of the changes made, I documented the before and after.

Testing and Validation Process

1. For the file permissions and WP-Config.php changes, I accessed the server using `ssh` and ran the command `ls -la` to view the permissions of the WordPress files and directories, and the WP-Config.php file. I verified that the permissions were changed to the desired values, restricting access to only the owner and group.

Next I ran the following commands:

```
cd /var/www/html
```

```
cd wp-admin
```

```
cd wp-content
```

```
cd wp-includes
```

and permission was denied to all

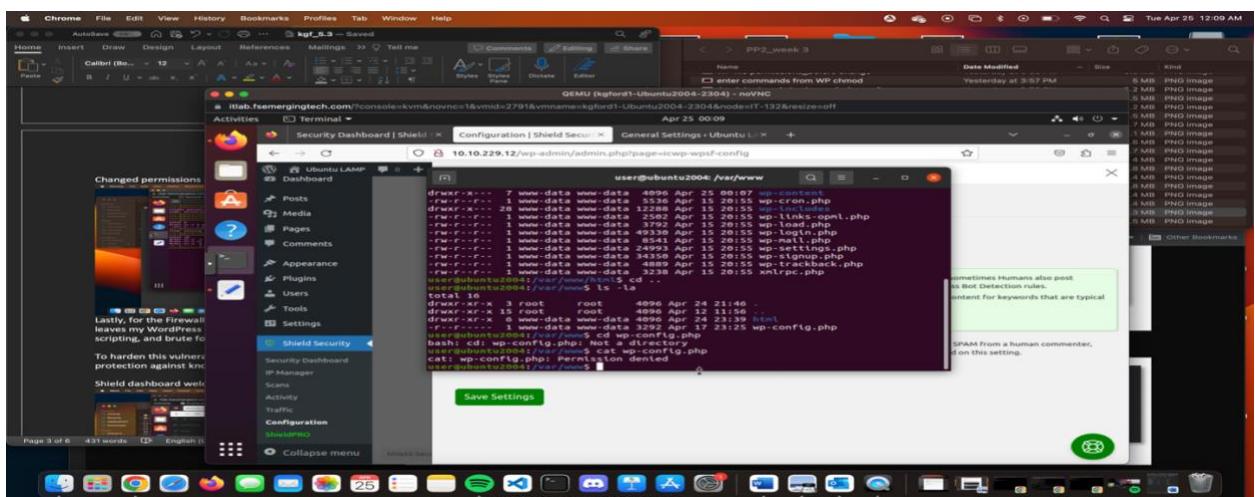
next entered:

```
cd /var/www/
```

then

```
cat wp-config.php
```

permission was denied



2. For the Shield firewall, I logged into the WordPress admin dashboard for the Shield firewall and verified that all suggested settings were active and that the custom rules had been added

properly. After that, I ran a vulnerability scan on the WordPress installation with a vulnerability scanner tool to make sure the system was secure against known threats.

WordPress Security Conclusion

In this exercise, I secured a WordPress installation using the Defense-in-Depth strategy. To defend a system from attack, this method employs multiple layers of security controls. By employing this strategy, I was able to find vulnerabilities in the WordPress installation and fix them, strengthening its security.

By taking similar actions, the Defense-in-Depth strategy can be applied to other systems. Determine any potential system weaknesses first. Next, determine how much risk these vulnerabilities pose. Third, take the necessary security precautions to lessen the risk of an attack.

By taking these precautions, businesses can lower their risk of data breaches, safeguard sensitive data, and preserve the privacy, integrity, and accessibility of their systems and data.

To put the Defense-in-Depth strategy into practice, follow these specific steps:

- Conduct a security audit to find any possible system vulnerabilities.
- Vulnerabilities should be prioritized according to their seriousness and likelihood of being exploited.
- Put security measures in place: Security measures should be implemented to lessen the risk of an attack.
- Monitor the system: The system needs to be monitored for signs of an attack.

Full Report Conclusion

This project covered a variety of topics, including setting up a custom network, installing Ghost on Docker, and setting up WordPress on Ubuntu. I discovered that Nginx is used for reverse proxying and serving web content. The article offered detailed instructions for setting up Nginx, starting it, and adding reverse proxy logs for the Ghost site. Additionally, the project covered the installation and testing of Ghost on Docker as well as the LAMP stack installation of WordPress on Ubuntu. The project was informative overall and gave a good overview of each of the subjects covered.

Appendix A

NginX Config File

The screenshot shows a Mac desktop environment with several windows open:

- Terminal Window:** Displays the NginX configuration file (`nginx.conf`) with syntax highlighting. The configuration includes sections for `events`, `http`, `server`, and `location`. It also includes a `proxy_pass` directive pointing to a Docker container.
- Browser Window:** Shows a QEMU session titled "QEMU (kgford1-Rocky8-2304) - noVNC". The URL is `http://itlab.fsemergingtech.com/?console=kvm&vnc=:1&vmid=2789&vmmname=kgford1-Rocky8-2304&nodeID=136&resize=off&cmd=`.
- Course Page:** A screenshot of a course page titled "2.1 NginX Reverse Proxy" from "online.fullsail.edu/class_sections/143113/modules/578410/activities/3371747". The page has sections for "OVERVIEW", "COMPLETION", and "FEEDBACK".
- System Tray:** Shows various system icons including battery level, signal strength, and volume.

Sample Code:

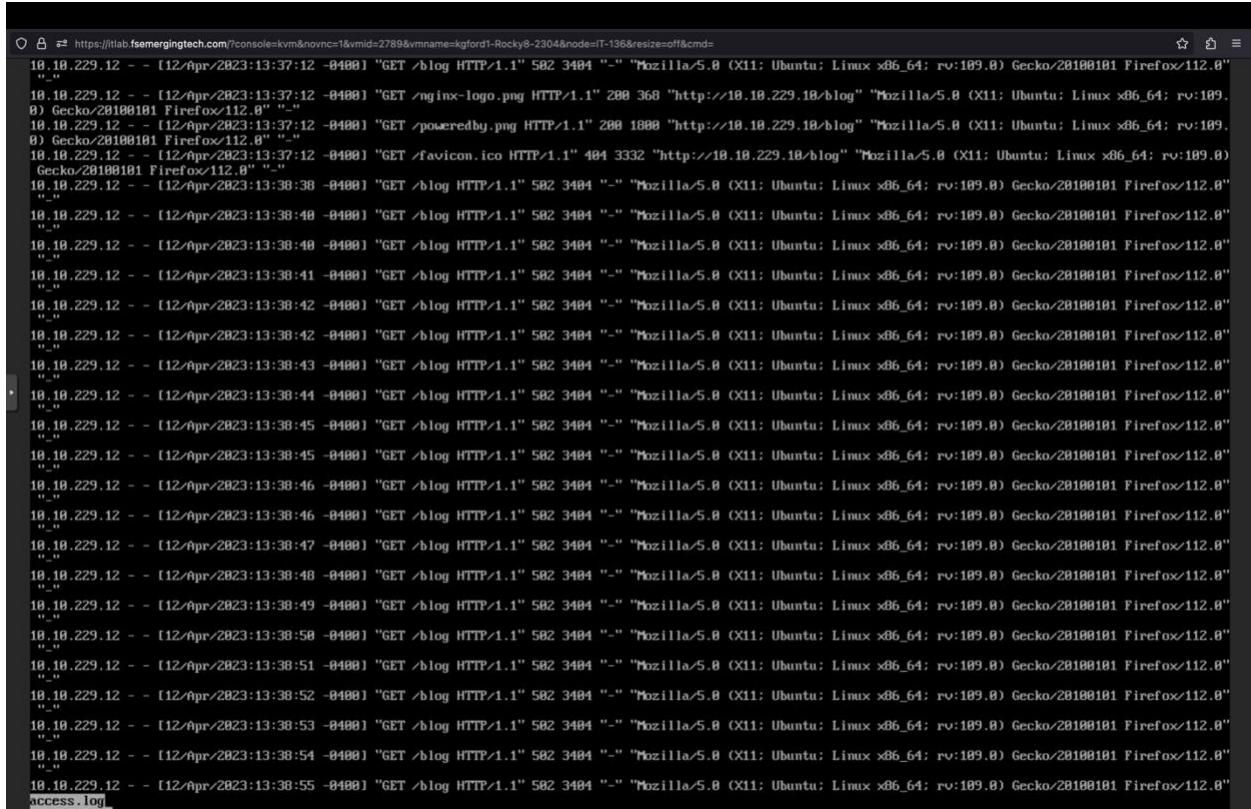
```
location /blog {  
    proxy_pass http://10.10.229.11:3001;  
    proxy_set_header Host $http_host; # required for  
    docker client's sake  
    proxy_set_header X-Real-IP $remote_addr; # pass on real  
    client's IP  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_read_timeout 900;  
}
```

Please run the following code on your Docker machine using the Ghost Docker

```
555 docker -d --name ghost -p 3001:2369 -e url=http://10.10.229.11:3001  
555 history  
root@localhost ~%#
```

Appendix B

NginX Access Log File



The screenshot shows a terminal window with the command `tail -f /var/log/nginx/access.log` running. The log file displays numerous entries from an IP address 10.10.229.12, all timestamped at 13:37:12 on April 12, 2023. Each entry is a GET request for the root URL ('/'). The browser used is Mozilla/5.0 (X11; Ubuntu; Linux x86_64) with a Gecko/20100101 Firefox/112.0 version. The response code is 200 OK, and the content length is 368 bytes. The log also shows a final line: `[access:10]`.

```
tail -f /var/log/nginx/access.log
10.10.229.12 - - [12/Apr/2023:13:37:12 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:37:12 -0400] "GET /nginx-logo.png HTTP/1.1" 200 368 "http://10.10.229.18/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:37:12 -0400] "GET /poweredby.png HTTP/1.1" 200 1000 "http://10.10.229.18/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:37:12 -0400] "GET /favicon.ico HTTP/1.1" 404 3332 "http://10.10.229.18/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:38 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:40 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:40 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:41 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:42 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:42 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:43 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:44 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:45 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:45 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:46 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:46 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:47 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:48 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:49 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:50 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:51 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:52 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:53 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:54 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
10.10.229.12 - - [12/Apr/2023:13:38:55 -0400] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0"
[access:10]
```