

03

ITE229 (SERVER-SIDE / LINUX WEB INFRASTRUCTURE)

# Proof-of-Concept Technical Solution: Secured WordPress Website for Marconi Law Firm

A three-VM LAMP, Docker/Ghost, and Nginx reverse-proxy build with defense-in-depth hardening on a custom 10.10.229.0/24 network

---

COURSE

ITE229 (Server-side / Linux web infrastructure)

DATE

2025

ENVIRONMENT

Custom network ITE229: subnet 10.10.229.0/24, mask 255.255.255.0, DNS/gateway 10.10.229.1

PAGES (REPORT)

54

STUDENT

Cody Richard

ORGANIZATION

Black Sands Security LLC (client: Marconi Law Firm, LLC – Orlando, Florida)

# Proof-of-Concept Technical Solution: Secured WordPress Website for Marconi Law Firm

I authored a 54-page proof-of-concept and audit-documentation package delivering a production-style WordPress website for the Marconi Law Firm, built across three Linux VMs on a custom 10.10.229.0/24 network behind a firewall/router VM gateway (10.10.229.1). I stood up a full LAMP stack on Ubuntu (10.10.229.12) running WordPress cloned from GitHub against a dedicated MySQL database, deployed a containerized Ghost CMS on a Rocky 8 Docker host (10.10.229.11), and fronted the Ghost site with an Nginx reverse proxy on a second Rocky 8 host (10.10.229.10) that routes /blog traffic to the container. I then applied defense-in-depth hardening: chmod/chown lockdown of /var/www/html (755 dirs, 644 files), relocating wp-config.php above the web root with 600 permissions, and installing the Wordfence Layer-7 WAF plugin — resolving a 403 error and a redirect loop in the process. Every step is documented as graded audit evidence with commands, explanations, and verification screenshots.

## ► Objectives

- Deliver a working WordPress website for a law-firm client as a documented proof of concept and audit artifact
- Design and build a segmented custom network (ITE229, 10.10.229.0/24) with a firewall/router VM gateway
- Deploy a full LAMP stack (Linux/Apache2/MySQL/PHP) on Ubuntu hosting WordPress
- Containerize a Ghost (Node.js) CMS on a Rocky 8 Docker host and expose it via an Nginx reverse proxy
- Harden the WordPress deployment with defense-in-depth controls (file permissions, config secrecy, application firewall)
- Produce comprehensive audit documentation supporting compliance, validation, records maintenance, and decision-making

## ► Environment

Custom network ITE229: subnet 10.10.229.0/24, mask 255.255.255.0, DNS/gateway 10.10.229.1

Firewall/Router VM gateway at 10.10.229.1

Rocky Linux 8 Docker host 'DockerRichard' at 10.10.229.11 (Ghost container)

Rocky Linux 8 Nginx reverse-proxy host 'NGINXRichard' at 10.10.229.10

Ubuntu LAMP host 'UbuntuXRichard' at 10.10.229.12 running WordPress

CLI-only VMs administered over OpenSSH; Firefox on Ubuntu used for browser verification

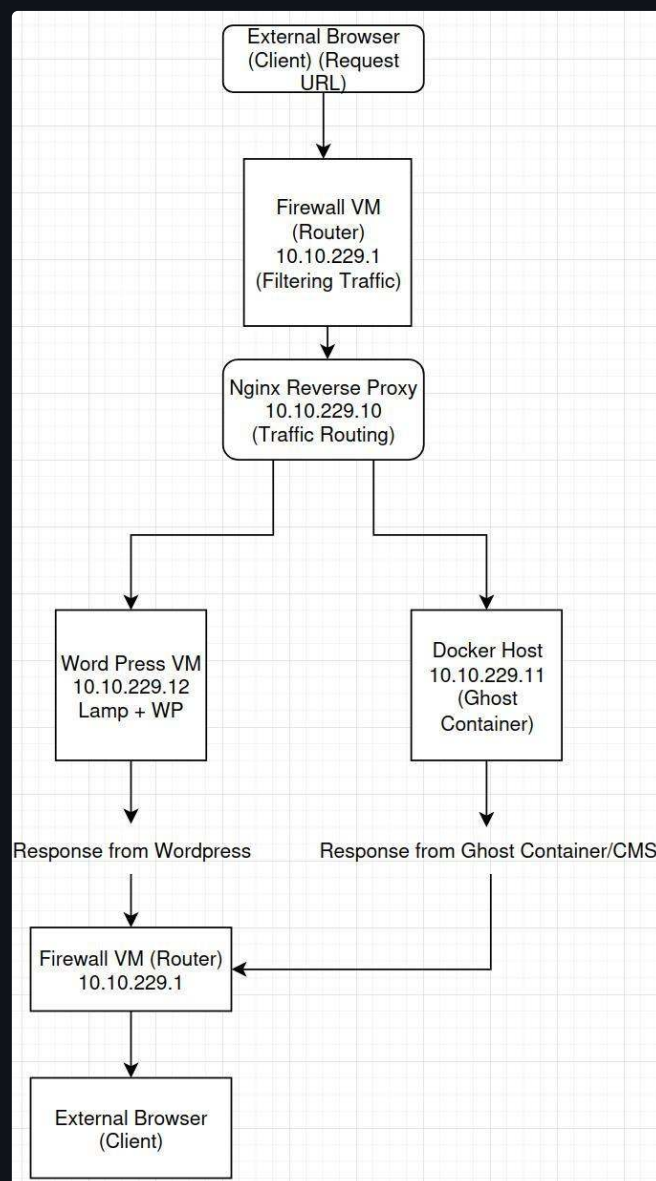
## WALKTHROUGH & EVIDENCE

This is the proof-of-concept build I delivered for the Marconi Law Firm: a production-style, secured WordPress site spanning three Linux VMs on a custom 10.10.229.0/24 network behind a firewall/router gateway at 10.10.229.1. The work covers a full LAMP stack on Ubuntu (10.10.229.12), a containerized Ghost CMS on a Rocky 8 Docker host (10.10.229.11), an Nginx reverse proxy on a second Rocky 8 host (10.10.229.10), and defense-in-depth hardening of the WordPress deployment. Every figure below is the original graded audit evidence, re-presented with commands, IPs, and outcomes preserved exactly as captured.

### ARCHITECTURE

## Network Topology

Three-tier segmented design on the custom ITE229 network (10.10.229.0/24, mask 255.255.255.0) with the firewall/router VM as gateway and DNS at 10.10.229.1.

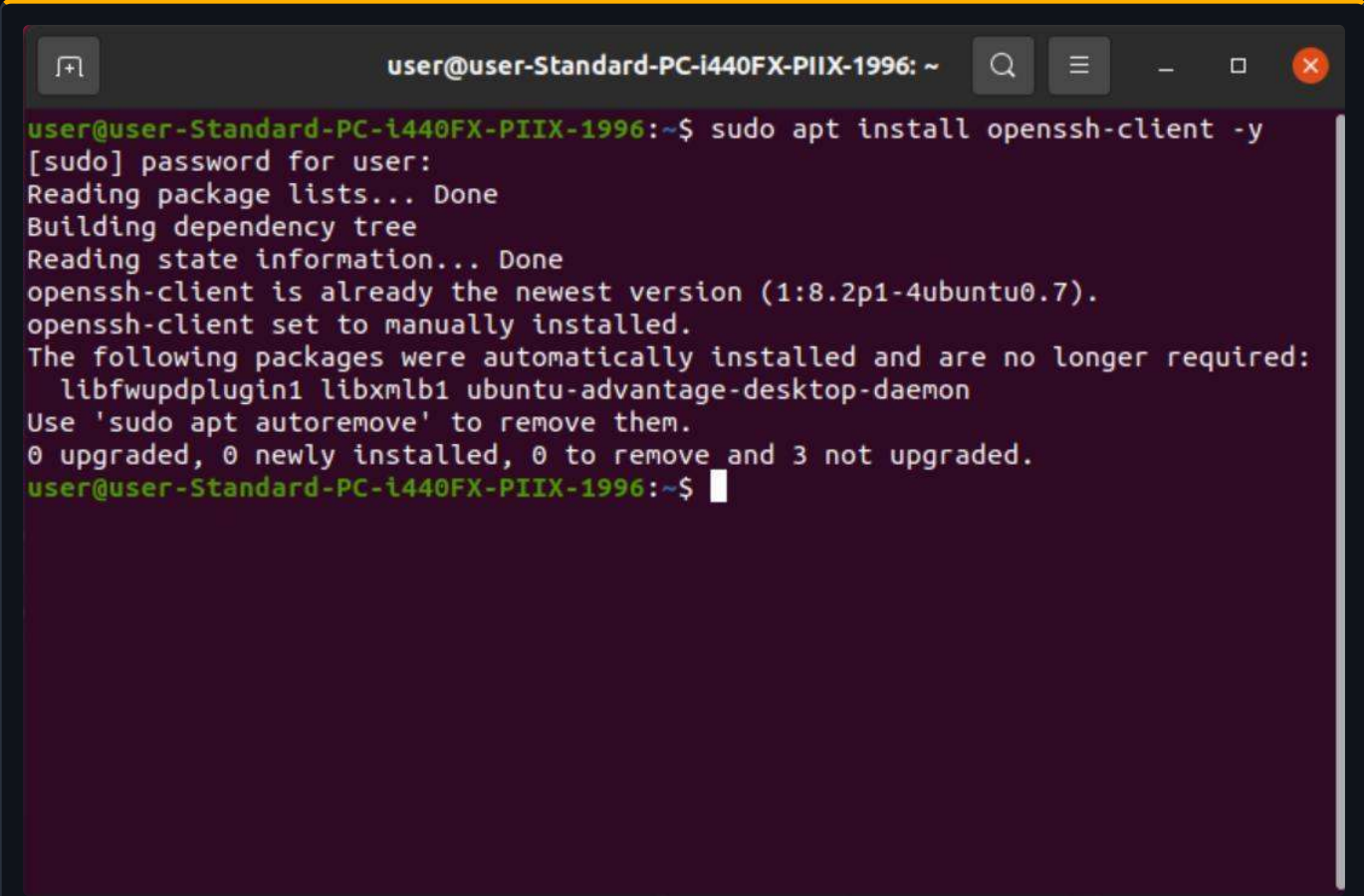


**FIG 01** — Network topology diagram for the ITE229 build: firewall/router gateway at 10.10.229.1 fronting the Docker host (.11), Nginx reverse proxy (.10), and Ubuntu LAMP/WordPress host (.12) on the 10.10.229.0/24 subnet.

## SSH ACCESS

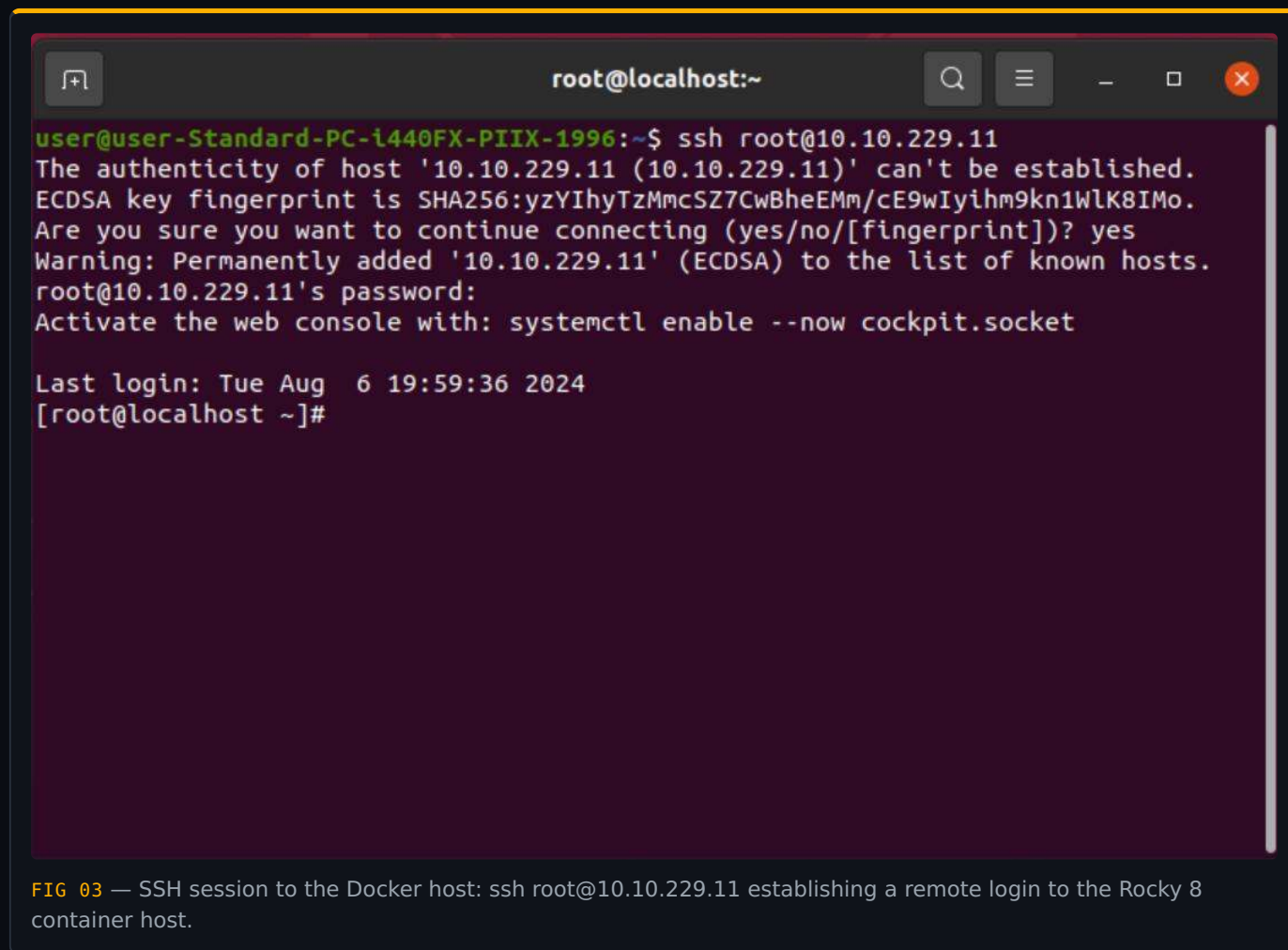
## OpenSSH Client and Host Login

Installed the OpenSSH client on Ubuntu to administer the CLI-only VMs remotely.

A terminal window screenshot showing the installation of the OpenSSH client on Ubuntu. The terminal title is 'user@user-Standard-PC-i440FX-PIIX-1996: ~'. The command entered is 'sudo apt install openssh-client -y'. The output shows that the package is already installed and lists some automatically installed packages that are no longer required. The terminal text is as follows:

```
user@user-Standard-PC-i440FX-PIIX-1996:~$ sudo apt install openssh-client -y
[sudo] password for user:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:8.2p1-4ubuntu0.7).
openssh-client set to manually installed.
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 libxmlb1 ubuntu-advantage-desktop-daemon
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
user@user-Standard-PC-i440FX-PIIX-1996:~$
```

**FIG 02** — Installing the OpenSSH client on Ubuntu with `sudo apt install openssh-client -y` to enable remote administration of the Docker and Nginx hosts.



## SSH ACCESS

## Reverse-Proxy Host Login and Docker Hostname

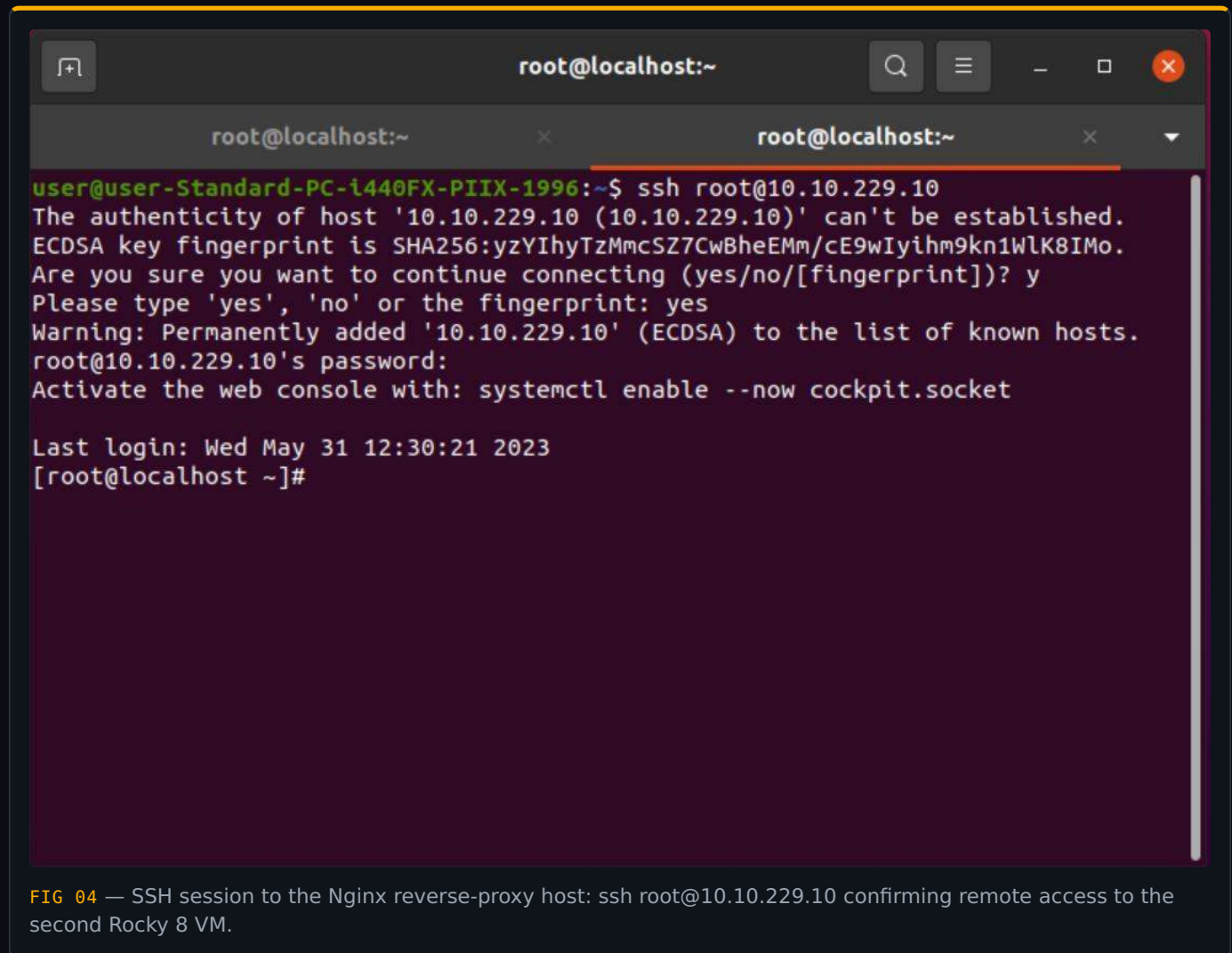


FIG 04 — SSH session to the Nginx reverse-proxy host: `ssh root@10.10.229.10` confirming remote access to the second Rocky 8 VM.

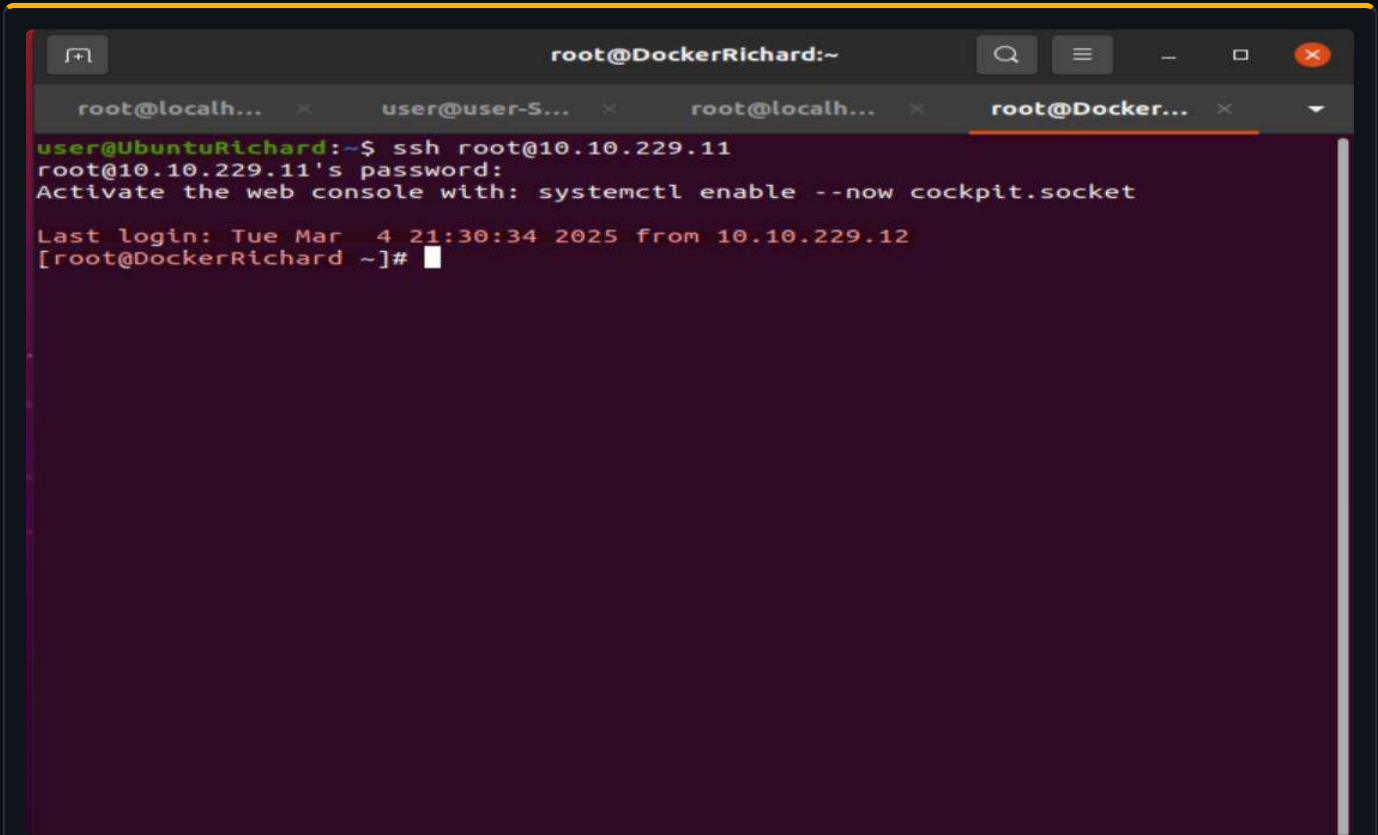


FIG 05 — nmtui System Hostname editor on the Docker host, preparing to set a meaningful hostname for the CLI-only VM.

DOCKER HOST

## Set Hostname and Update Packages

Renamed the Docker host to DockerRichard and brought the system fully up to date.

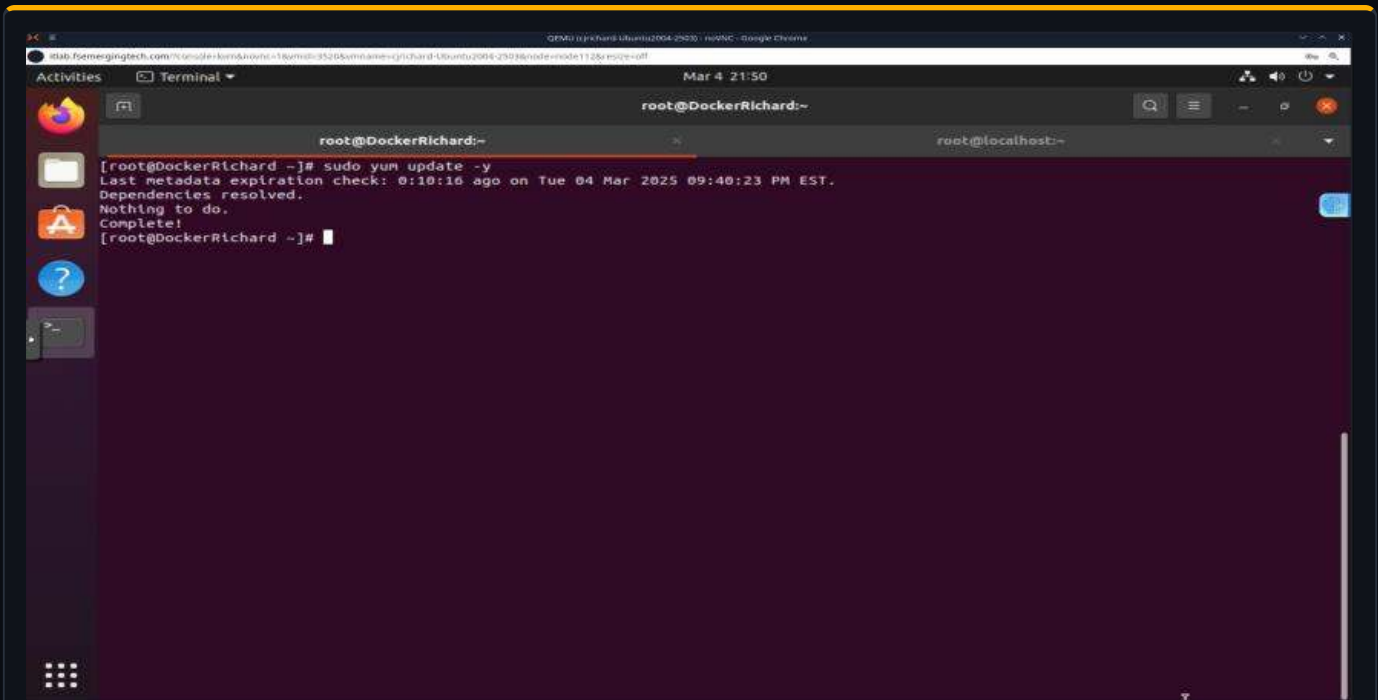


FIG 06 — Docker host renamed to DockerRichard via nmtui, simplifying SSH identification across the VM fleet.

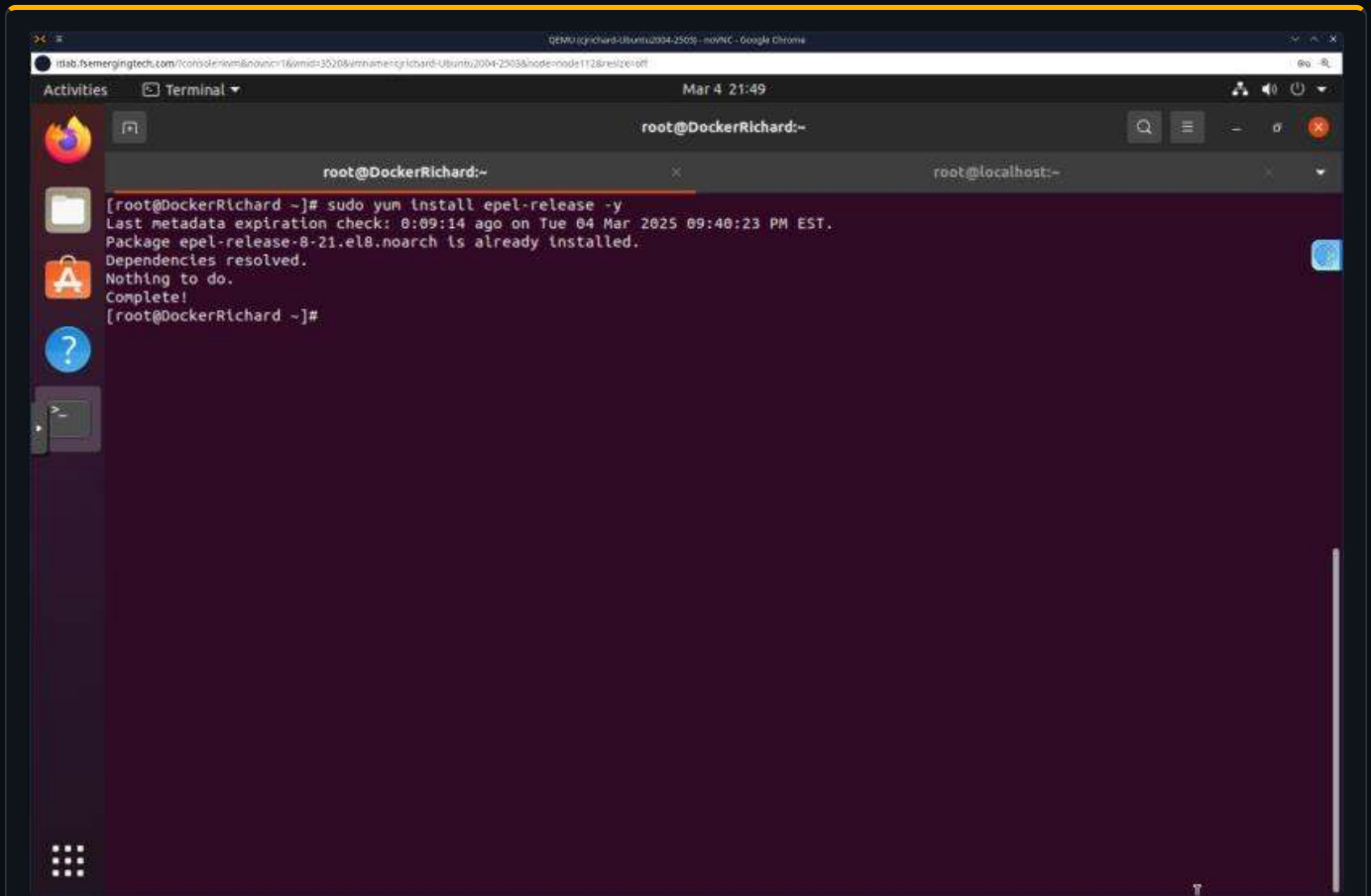


FIG 07 — sudo yum update -y on the Rocky 8 Docker host, refreshing all installed packages for security and stability.

## DOCKER HOST

## EPEL and Nano

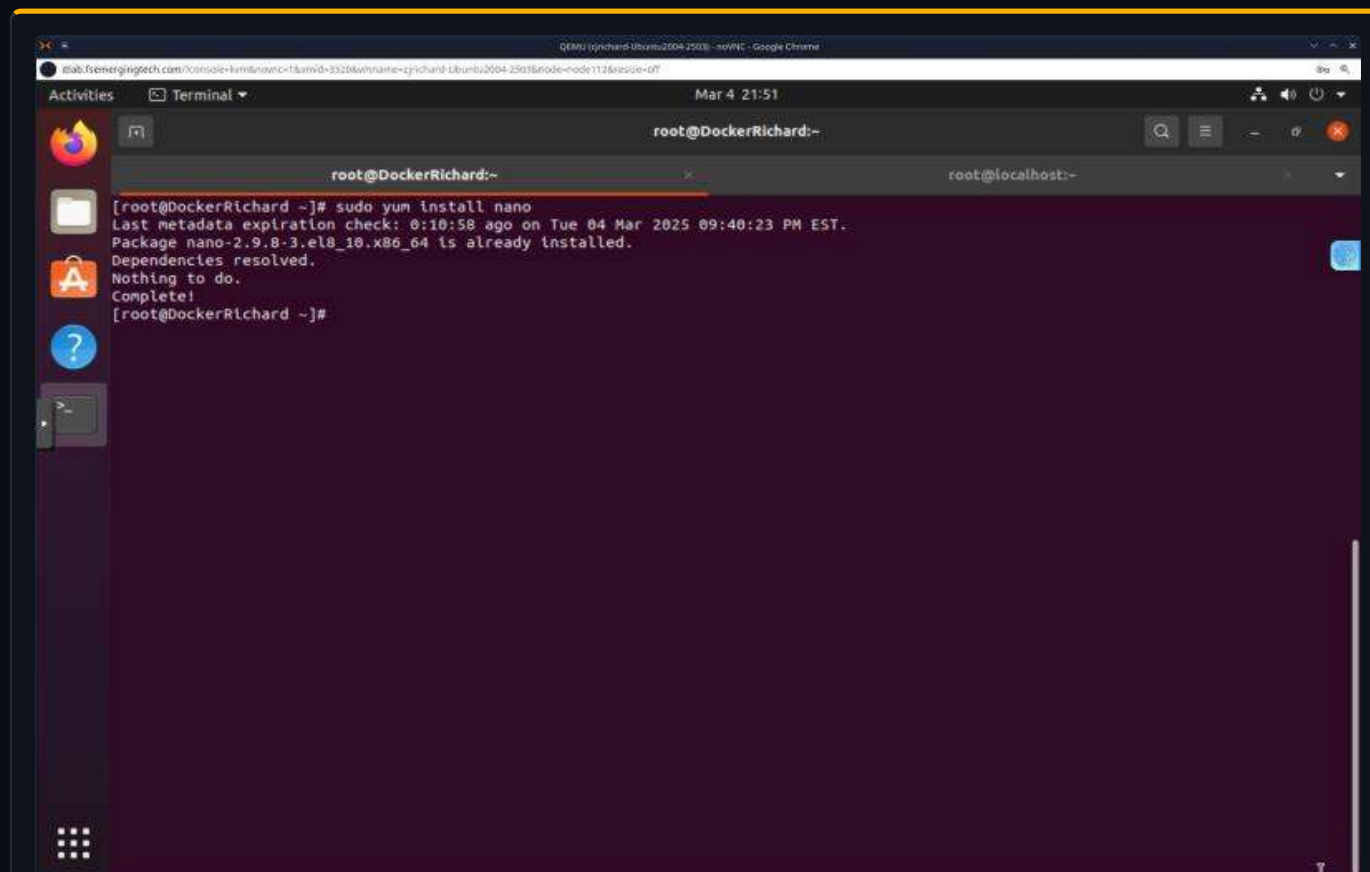


FIG 08 — Installing the EPEL repository with `sudo yum install epel-release -y` to make additional packages available on Rocky 8.

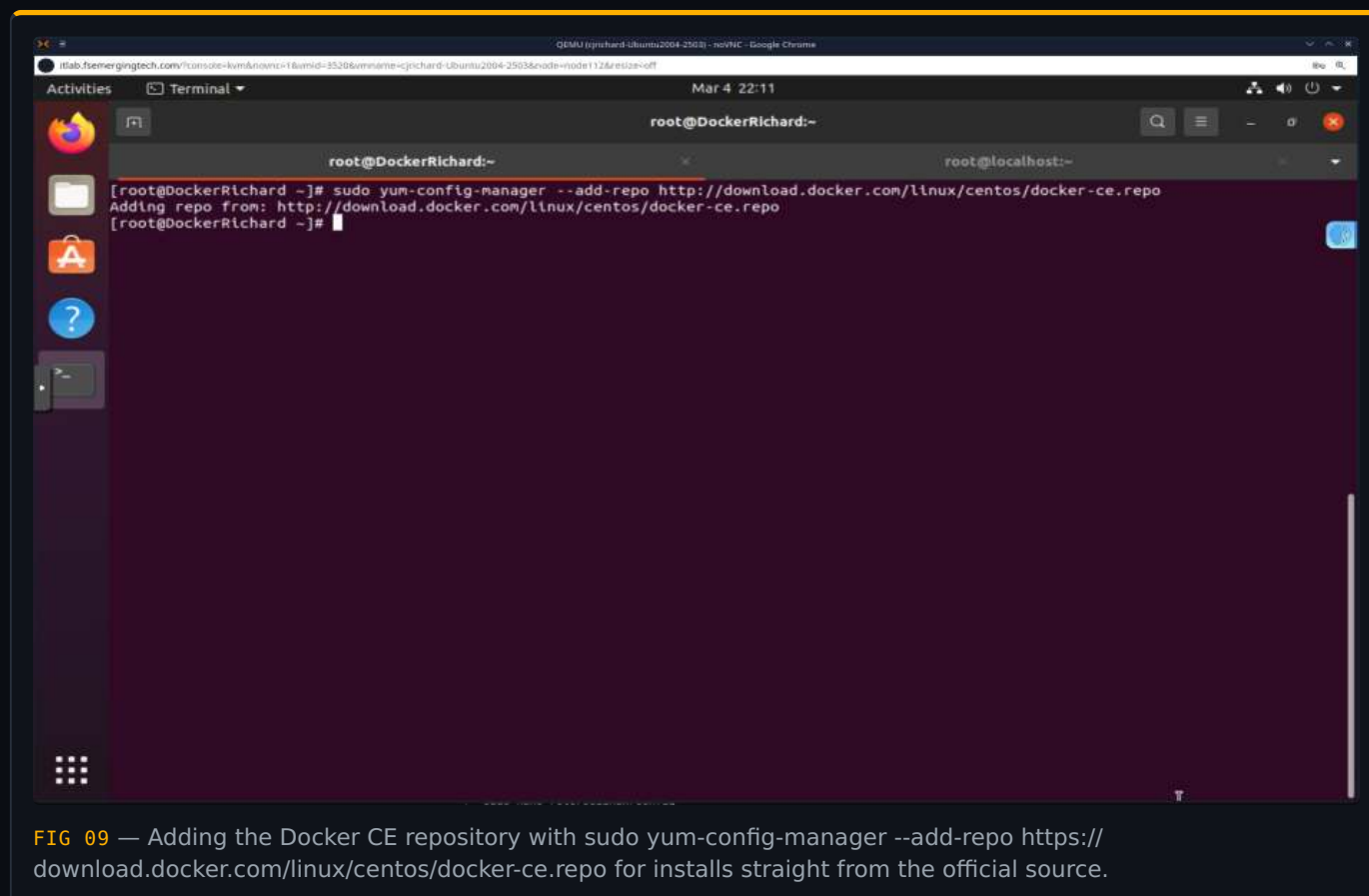


FIG 09 — Adding the Docker CE repository with `sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo` for installs straight from the official source.

DOCKER HOST

# Install Docker CE and Verify Version

Installed Docker CE with its CLI, runtime, buildx, and compose plugins, then confirmed the version.

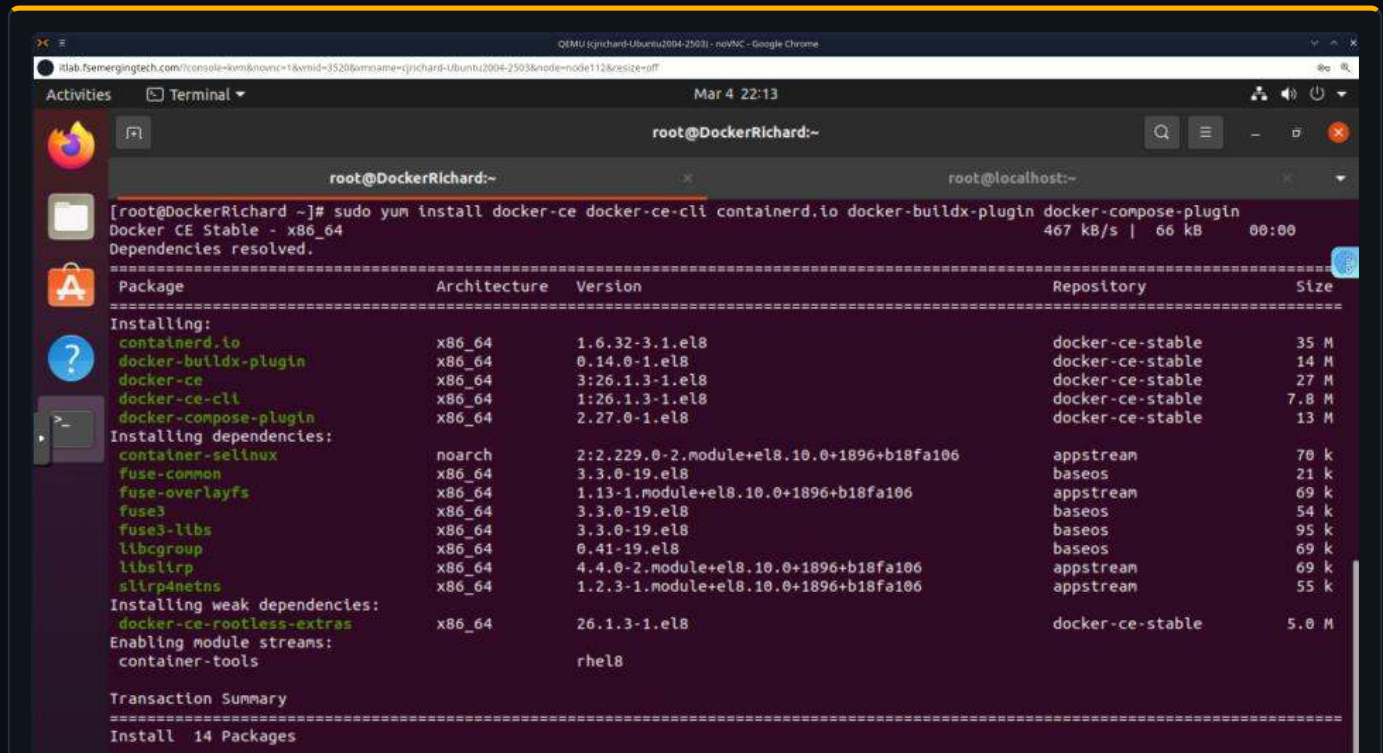


FIG 10 — Installing Docker CE and components: `sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`.

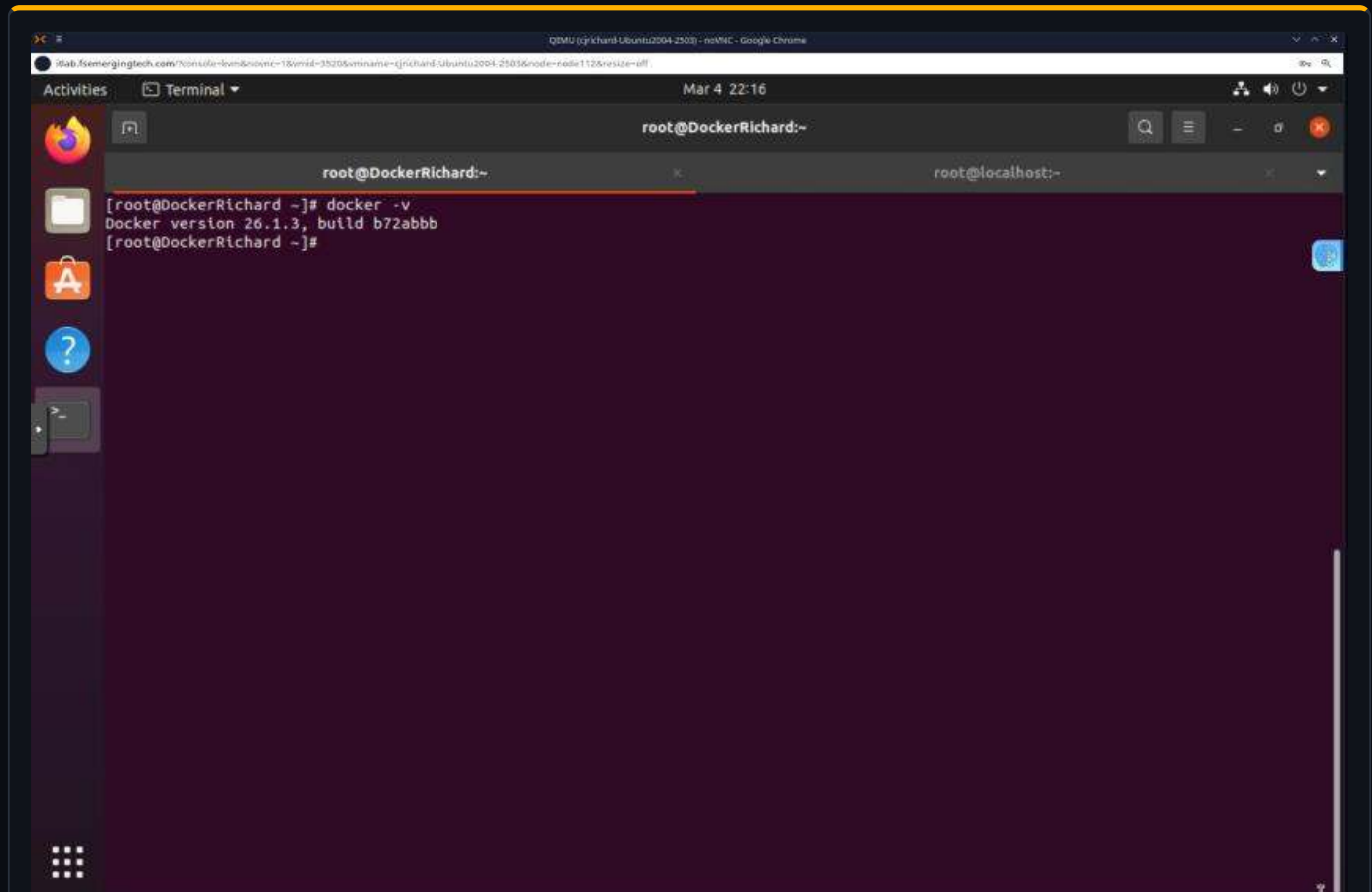
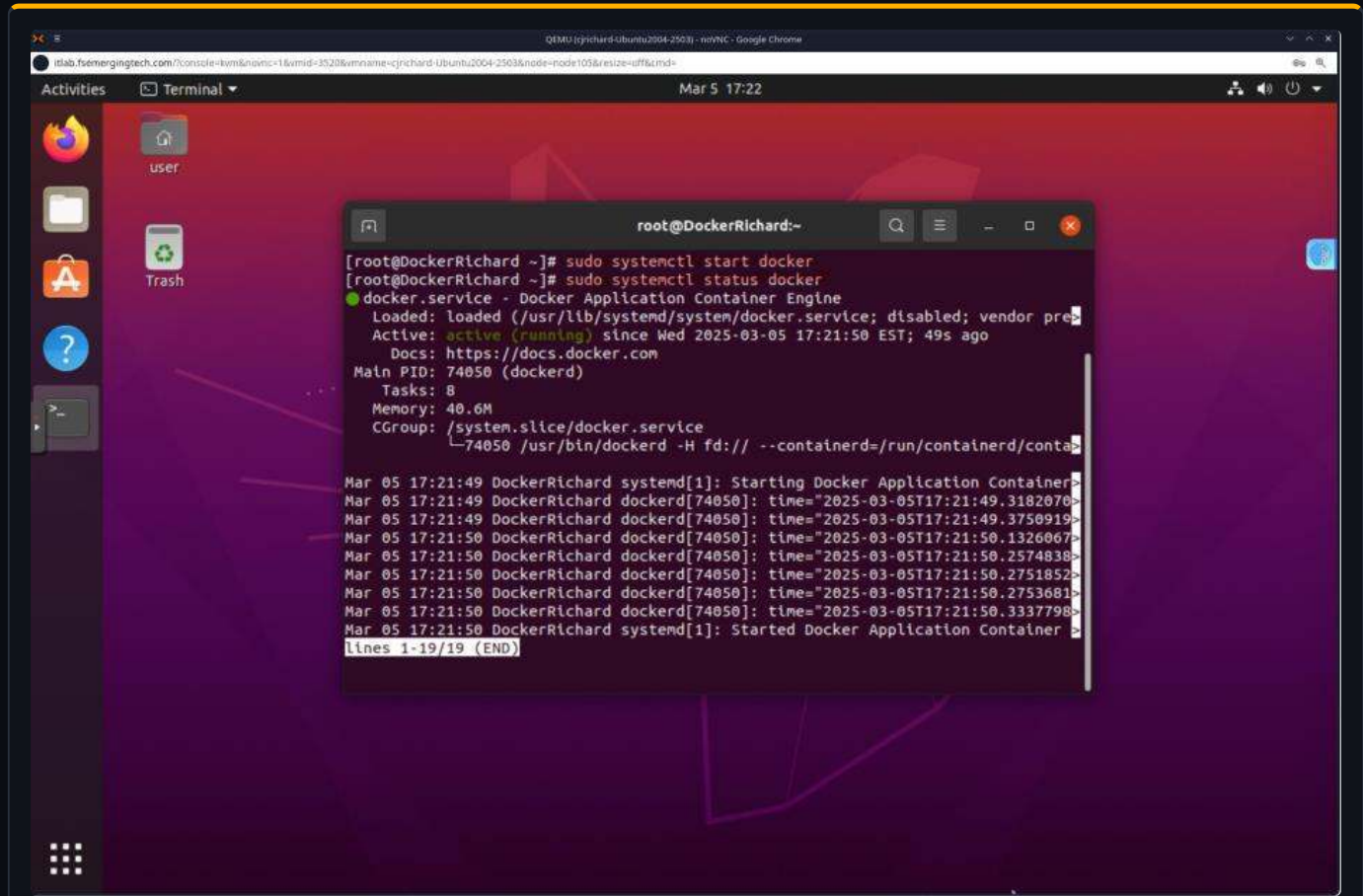


FIG 11 — Verifying the install with `docker -v`, confirming the Docker CE version was deployed correctly.

## DOCKER HOST

## Start and Enable the Docker Service

A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the following commands and output:

```
[root@DockerRichard ~]# sudo systemctl start docker
[root@DockerRichard ~]# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2025-03-05 17:21:50 EST; 49s ago
     Docs: https://docs.docker.com
   Main PID: 74050 (dockerd)
      Tasks: 8
     Memory: 40.6M
    CGroup: /system.slice/docker.service
            └─74050 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contai...
```

```
Mar 05 17:21:49 DockerRichard systemd[1]: Starting Docker Application Container Engine:
Mar 05 17:21:49 DockerRichard dockerd[74050]: time="2025-03-05T17:21:49.3182070Z"
Mar 05 17:21:49 DockerRichard dockerd[74050]: time="2025-03-05T17:21:49.3750919Z"
Mar 05 17:21:50 DockerRichard dockerd[74050]: time="2025-03-05T17:21:50.1326067Z"
Mar 05 17:21:50 DockerRichard dockerd[74050]: time="2025-03-05T17:21:50.2574830Z"
Mar 05 17:21:50 DockerRichard dockerd[74050]: time="2025-03-05T17:21:50.2751852Z"
Mar 05 17:21:50 DockerRichard dockerd[74050]: time="2025-03-05T17:21:50.2753681Z"
Mar 05 17:21:50 DockerRichard dockerd[74050]: time="2025-03-05T17:21:50.3337798Z"
Mar 05 17:21:50 DockerRichard systemd[1]: Started Docker Application Container Engine:
lines 1-19/19 (END)
```

FIG 12 — Starting the daemon with `sudo systemctl start docker` and checking `sudo systemctl status docker` to confirm it is active and running.

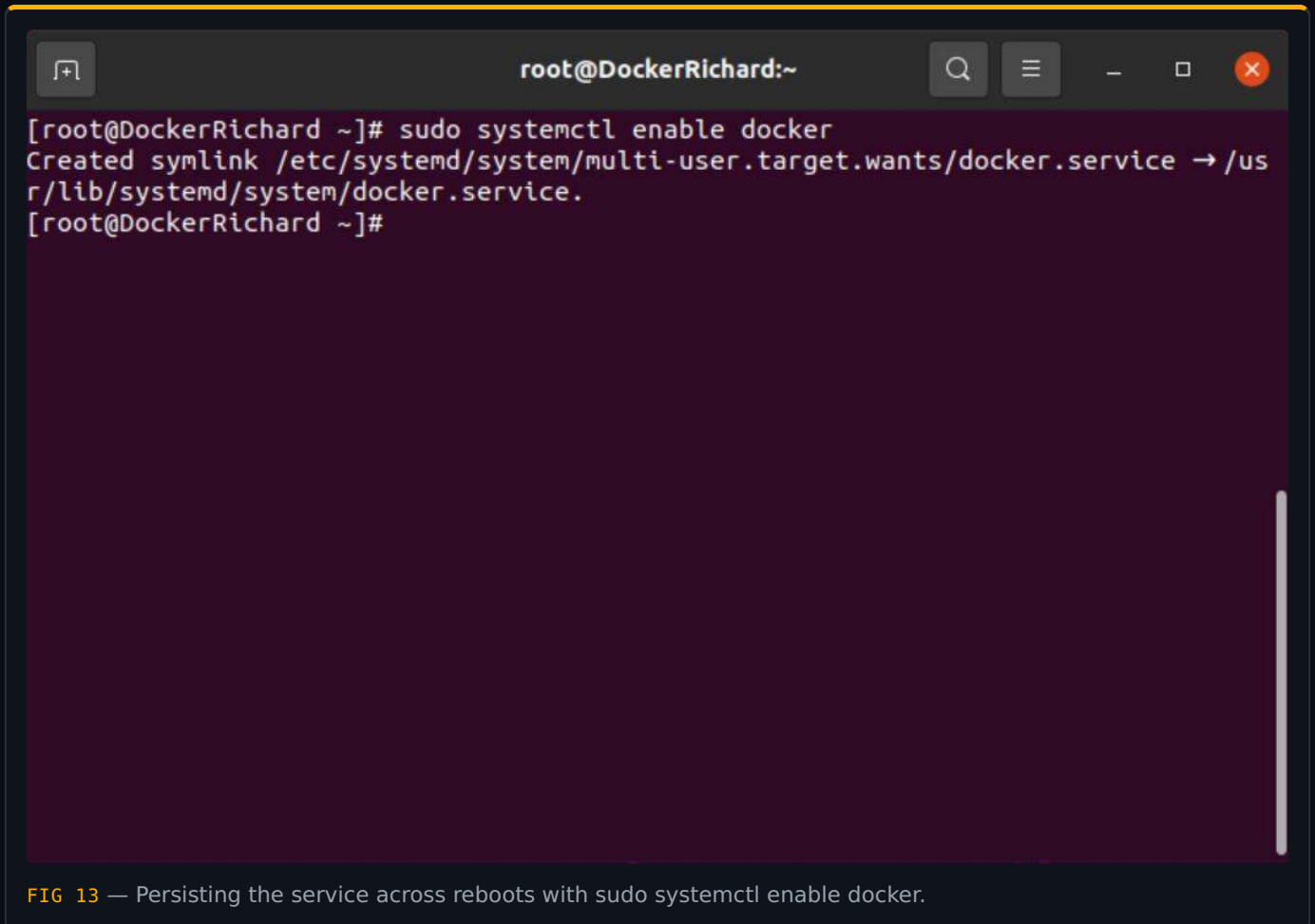
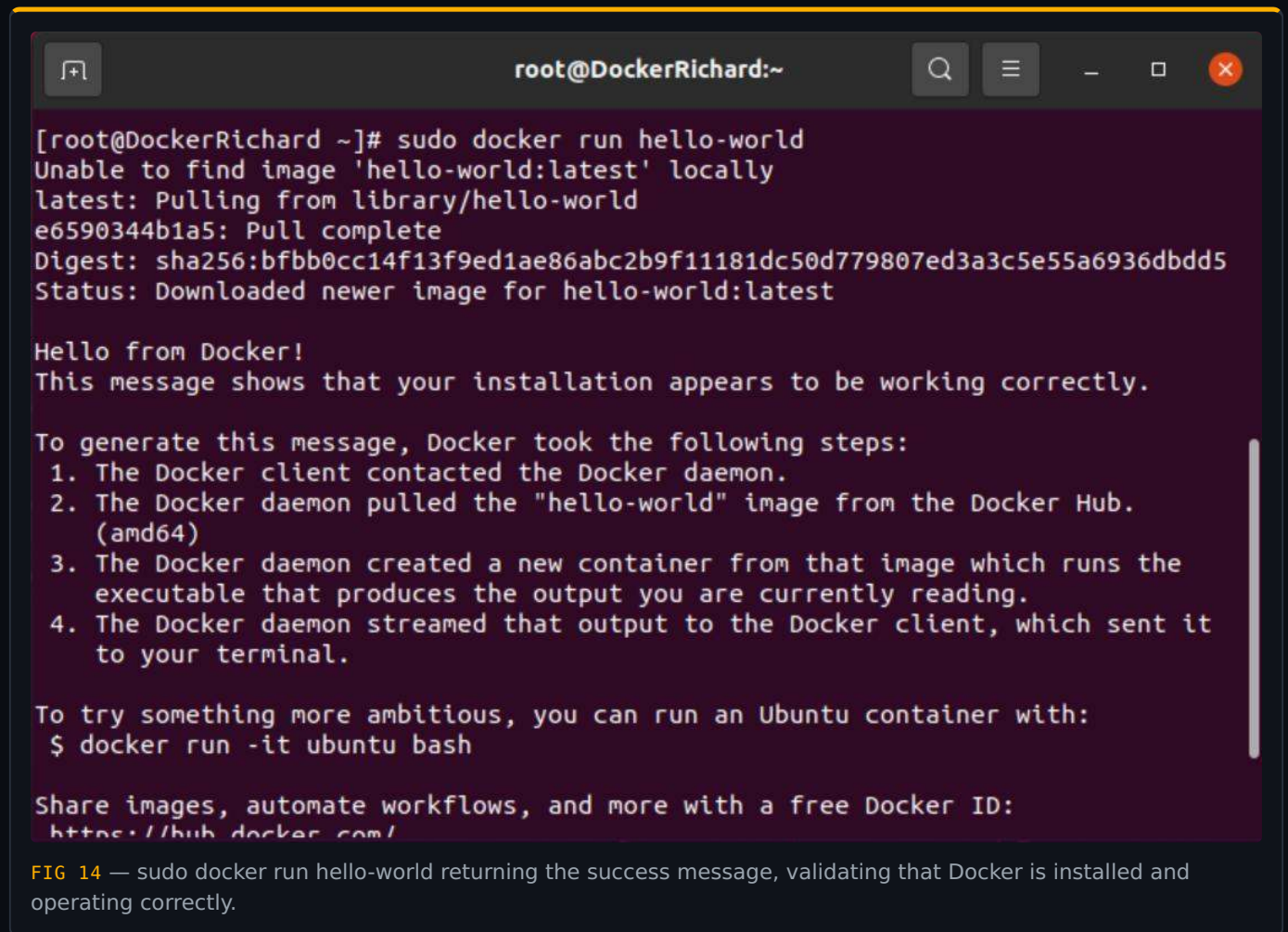


FIG 13 — Persisting the service across reboots with sudo systemctl enable docker.

## DOCKER HOST

## Validate Docker and Edit SELinux

Ran the hello-world container to prove the install, then disabled SELinux to avoid conflicts for this build.

A terminal window titled 'root@DockerRichard:~' with standard window controls. The terminal output shows the command 'sudo docker run hello-world' being executed. It reports that the 'hello-world:latest' image was not found locally and was pulled from the Docker Hub. The output includes the image ID 'e6590344b1a5', the digest 'sha256:bfbb0cc14f13f9ed1ae86abc2b9f11181dc50d779807ed3a3c5e55a6936dbdd5', and the status 'Downloaded newer image for hello-world:latest'. The container then prints 'Hello from Docker!' followed by a success message and a list of steps Docker took to run the container. It also provides instructions for running an Ubuntu container and a link to Docker Hub.

```
[root@DockerRichard ~]# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:bfbb0cc14f13f9ed1ae86abc2b9f11181dc50d779807ed3a3c5e55a6936dbdd5
Status: Downloaded newer image for hello-world:latest

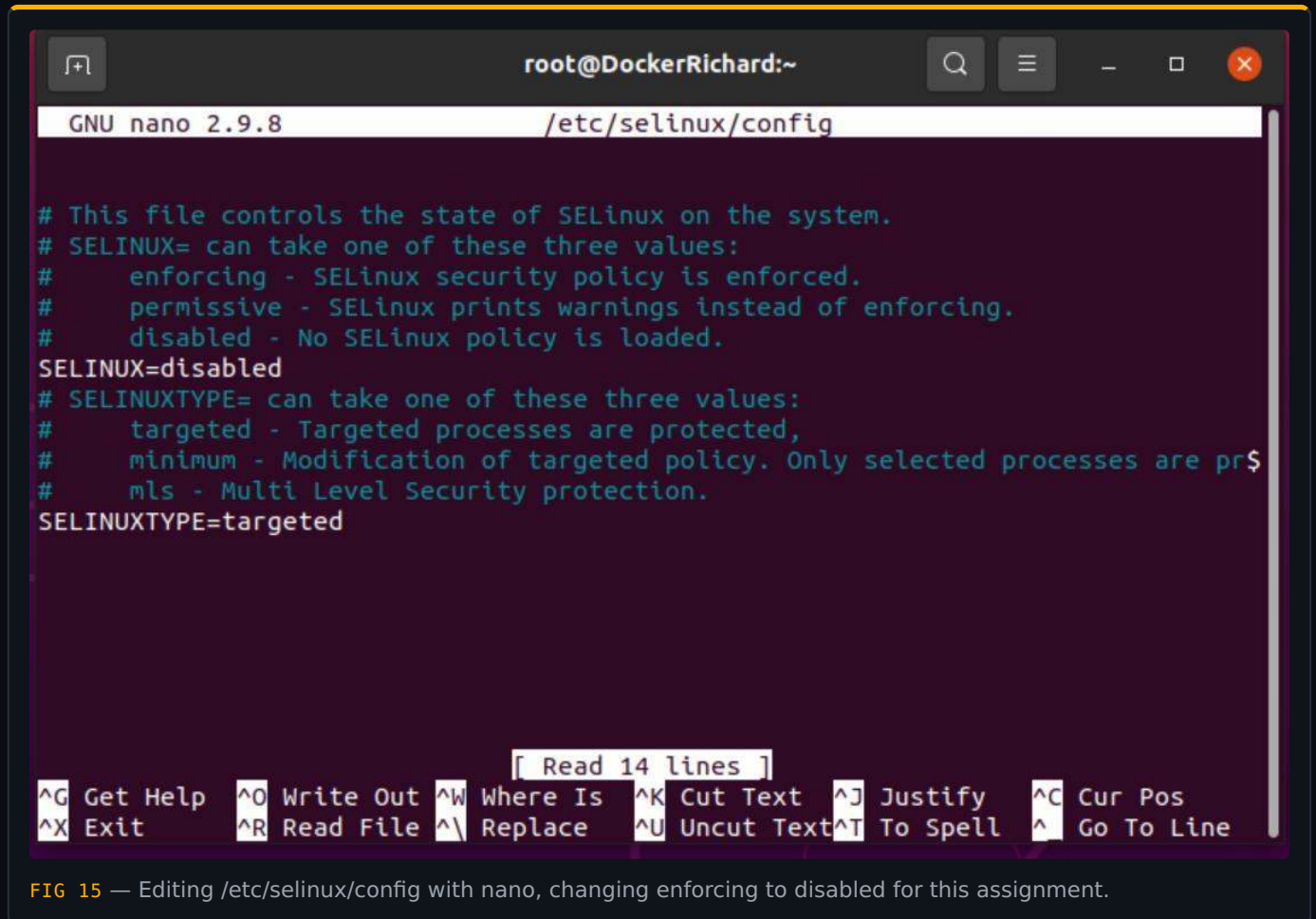
Hello from Docker!
This message shows that your installation appears to be working correctly.

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

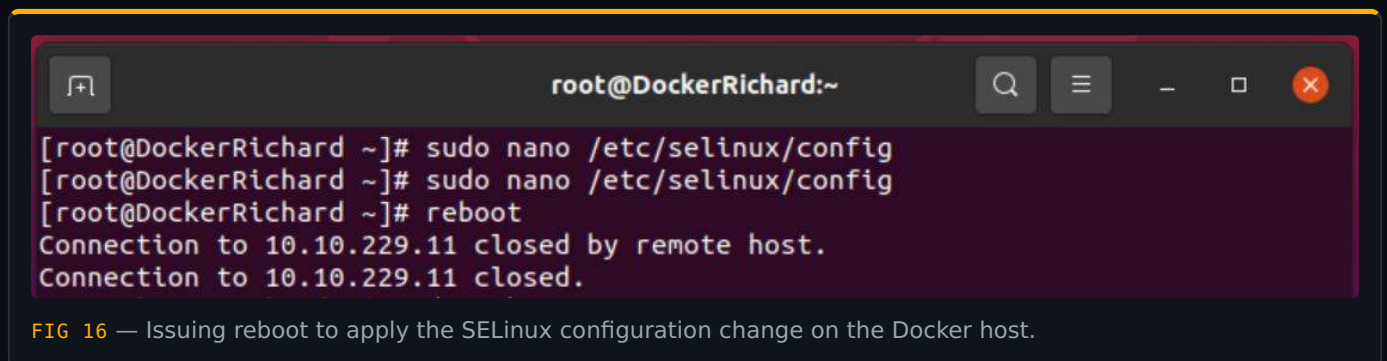
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
```

FIG 14 — sudo docker run hello-world returning the success message, validating that Docker is installed and operating correctly.



## DOCKER HOST

### Reboot and Confirm SELinux Status



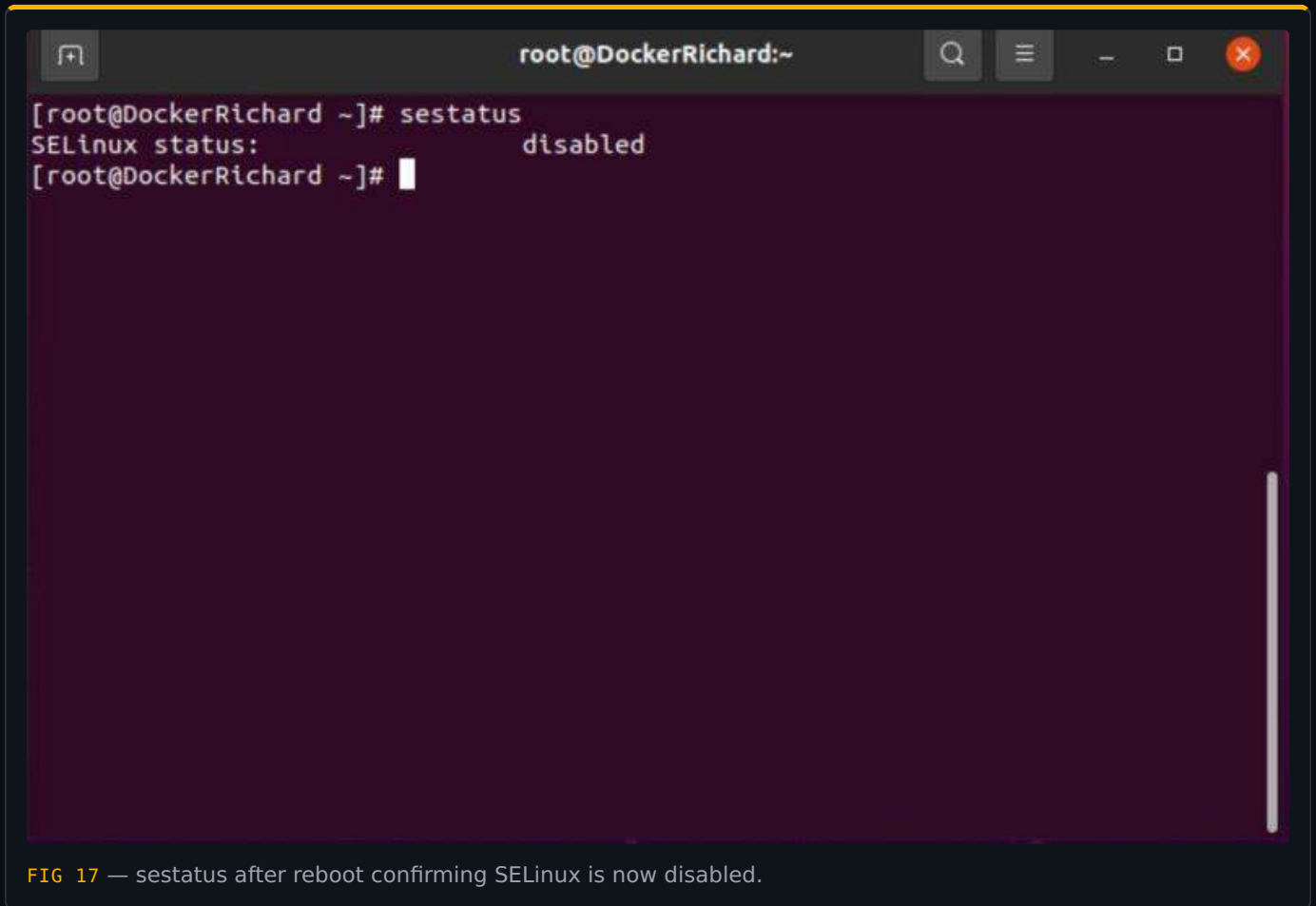
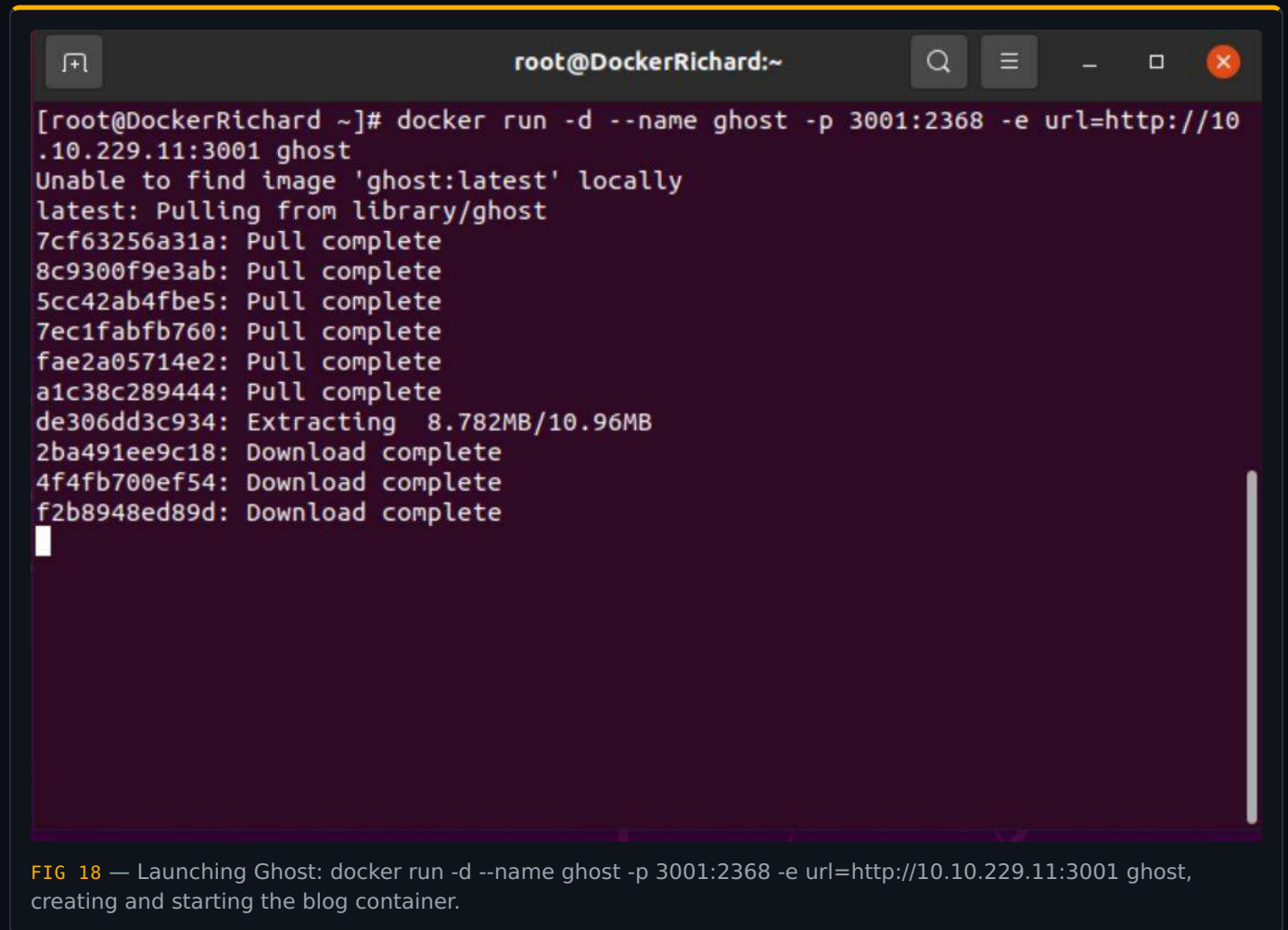


FIG 17 — sestatus after reboot confirming SELinux is now disabled.

## GHOST CMS

## Deploy the Ghost Container

Stood up the Ghost CMS container, mapping host port 3001 to the container's 2368.



```
root@DockerRichard:~  
[root@DockerRichard ~]# docker run -d --name ghost -p 3001:2368 -e url=http://10.10.229.11:3001 ghost  
Unable to find image 'ghost:latest' locally  
latest: Pulling from library/ghost  
7cf63256a31a: Pull complete  
8c9300f9e3ab: Pull complete  
5cc42ab4fbe5: Pull complete  
7ec1fabfb760: Pull complete  
fae2a05714e2: Pull complete  
a1c38c289444: Pull complete  
de306dd3c934: Extracting 8.782MB/10.96MB  
2ba491ee9c18: Download complete  
4f4fb700ef54: Download complete  
f2b8948ed89d: Download complete  
█
```

FIG 18 — Launching Ghost: `docker run -d --name ghost -p 3001:2368 -e url=http://10.10.229.11:3001 ghost`, creating and starting the blog container.

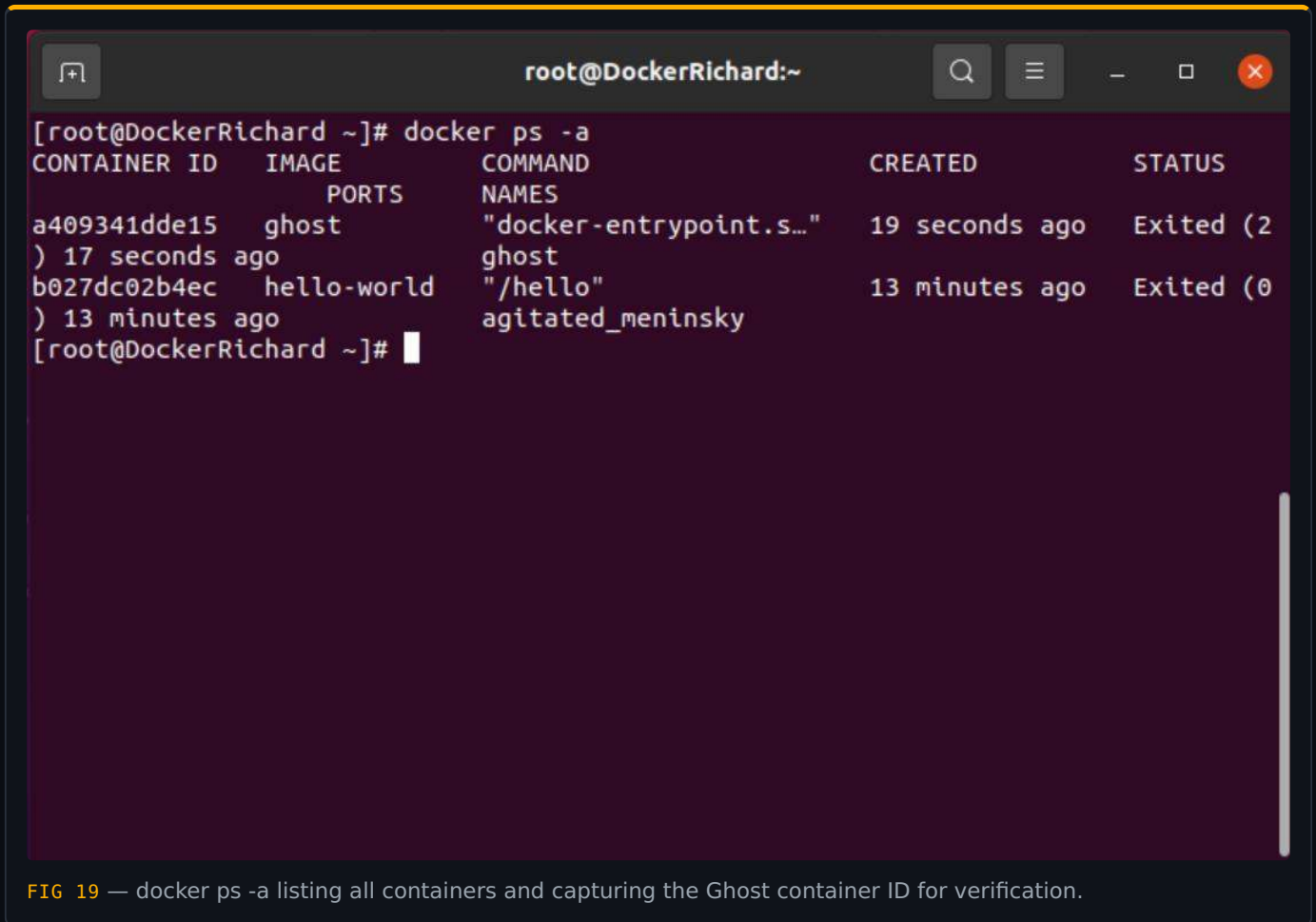


FIG 19 — docker ps -a listing all containers and capturing the Ghost container ID for verification.

## PROXY HOST

## Set Hostname and Update Packages

Prepared the second Rocky 8 host (the reverse proxy): renamed to NGINXRichard and updated.

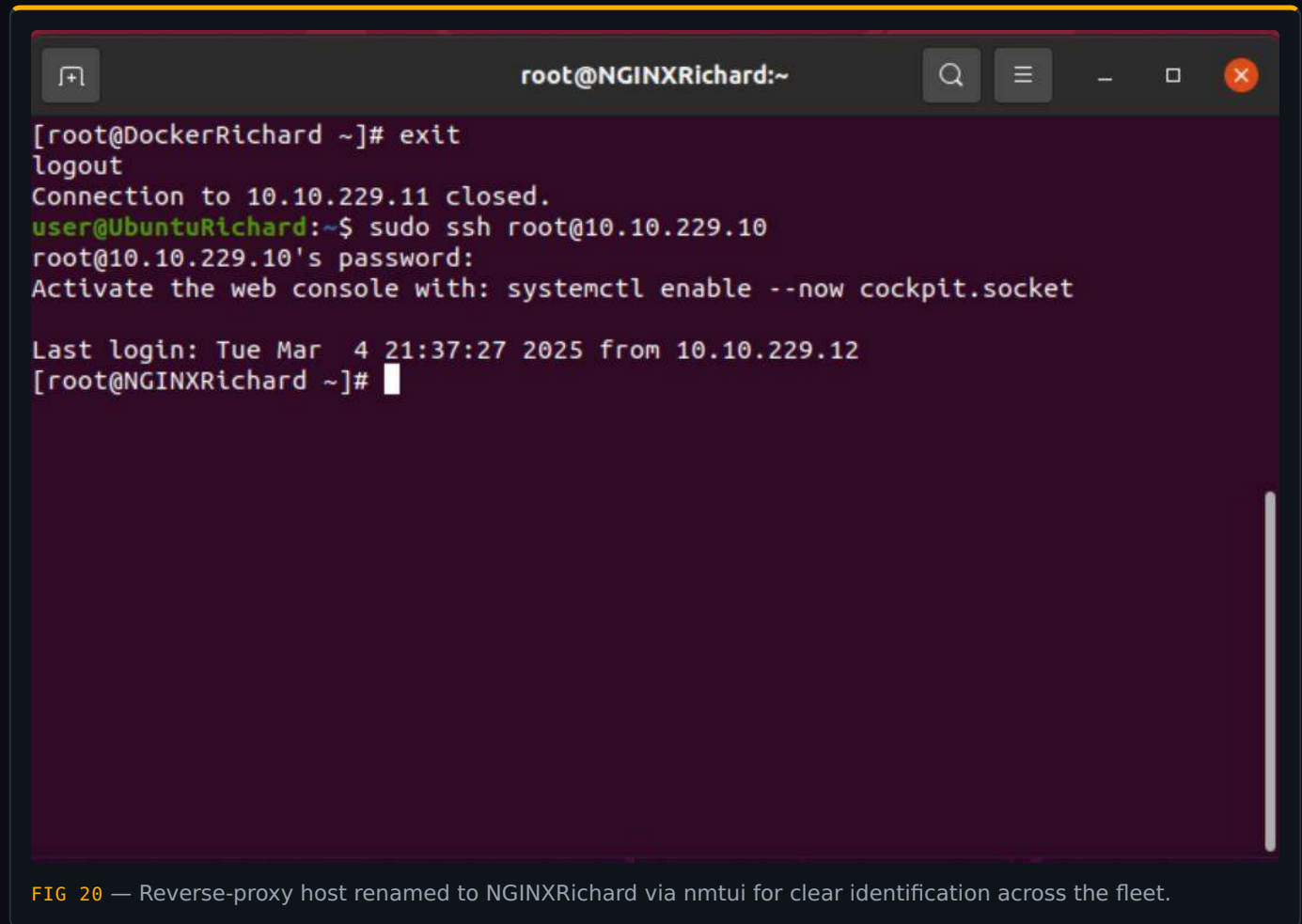


FIG 20 — Reverse-proxy host renamed to NGINXRichard via nmtui for clear identification across the fleet.

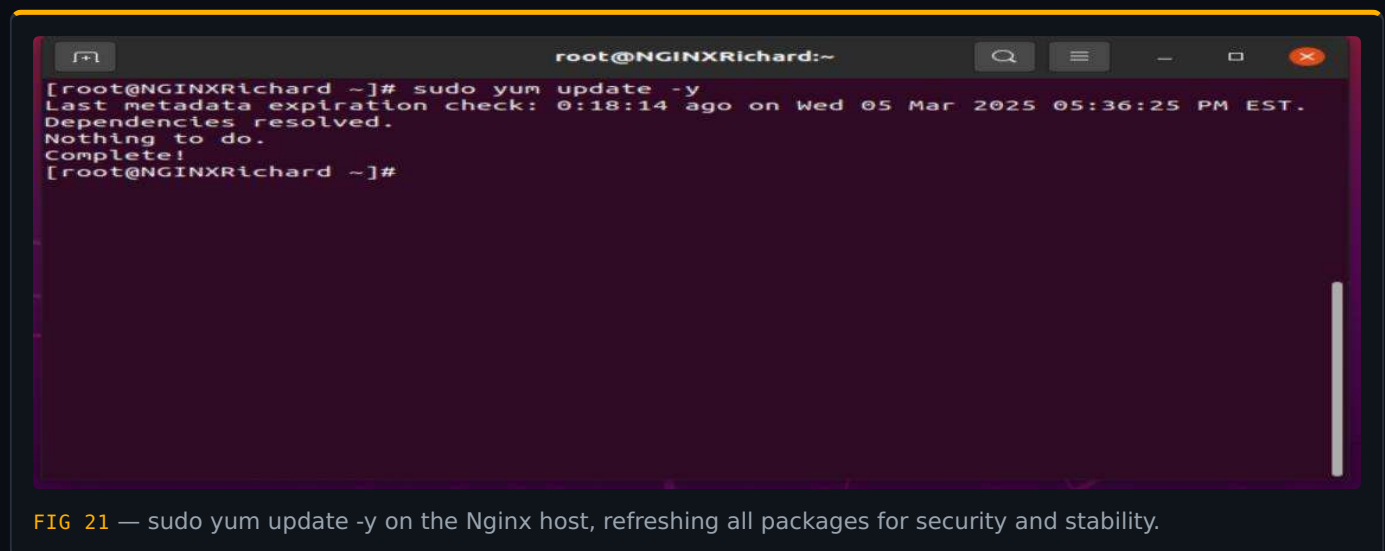
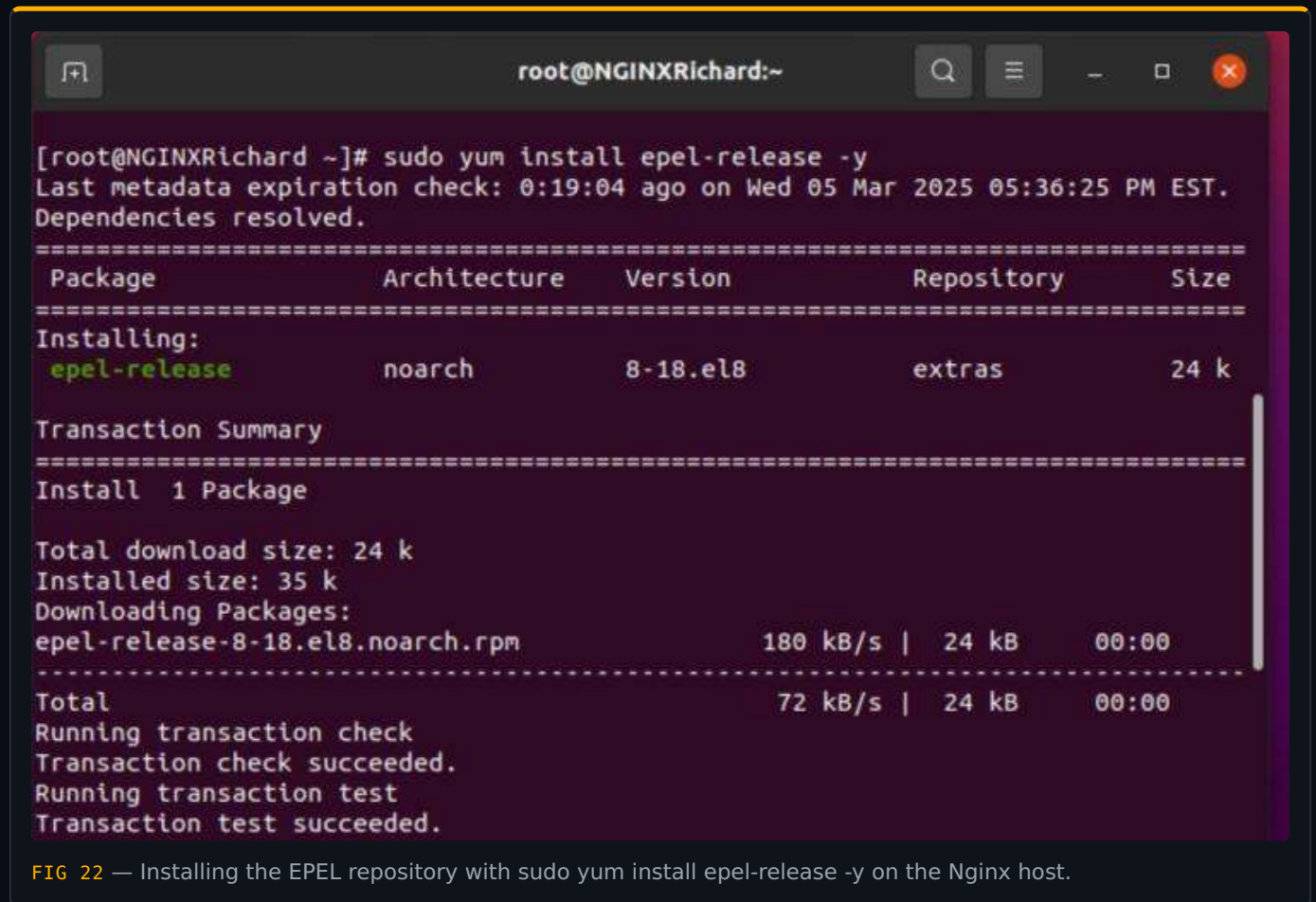


FIG 21 — sudo yum update -y on the Nginx host, refreshing all packages for security and stability.

## PROXY HOST

## EPEL and Nano



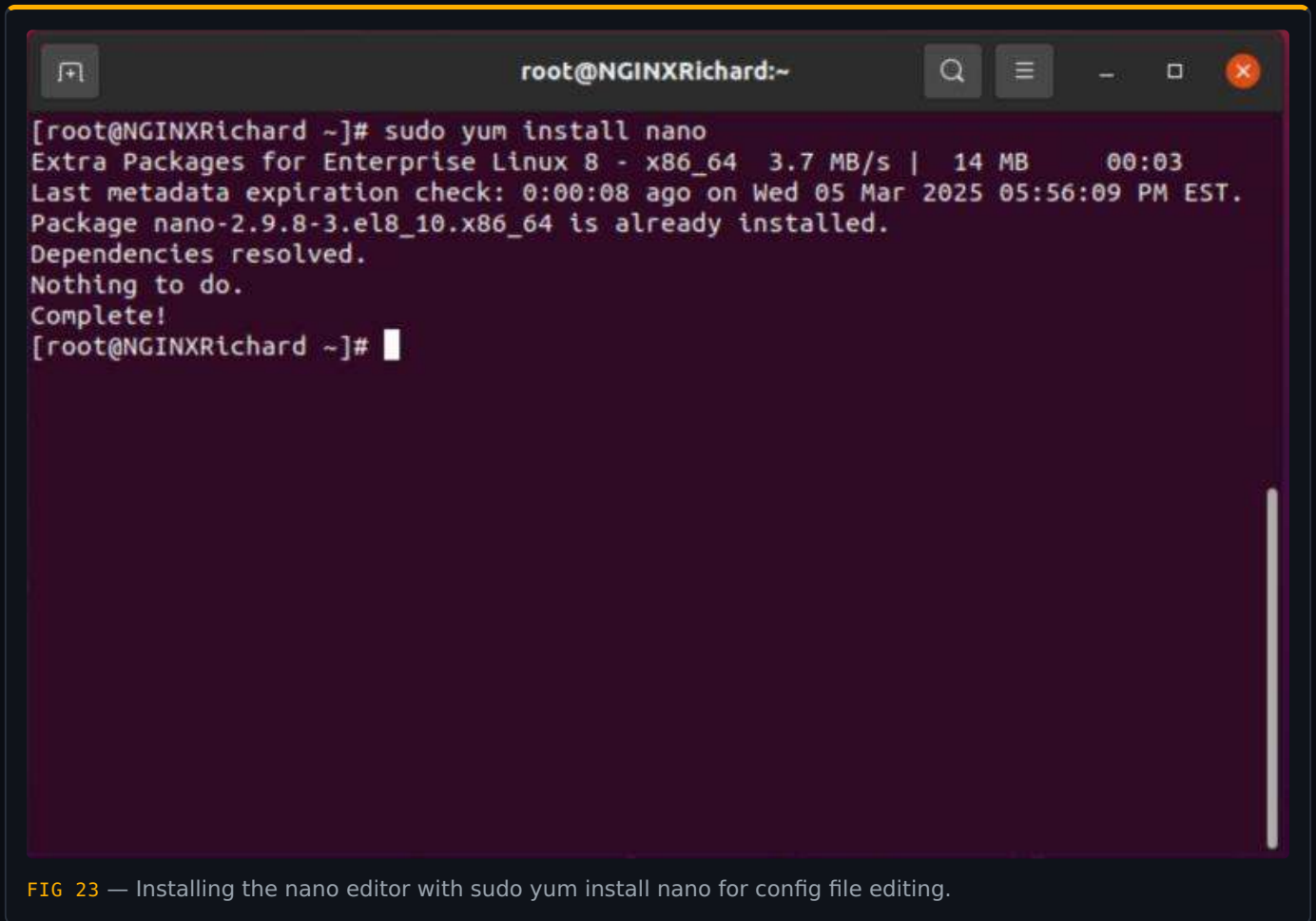


FIG 23 — Installing the nano editor with `sudo yum install nano` for config file editing.

PROXY HOST

## Disable SELinux and Reboot

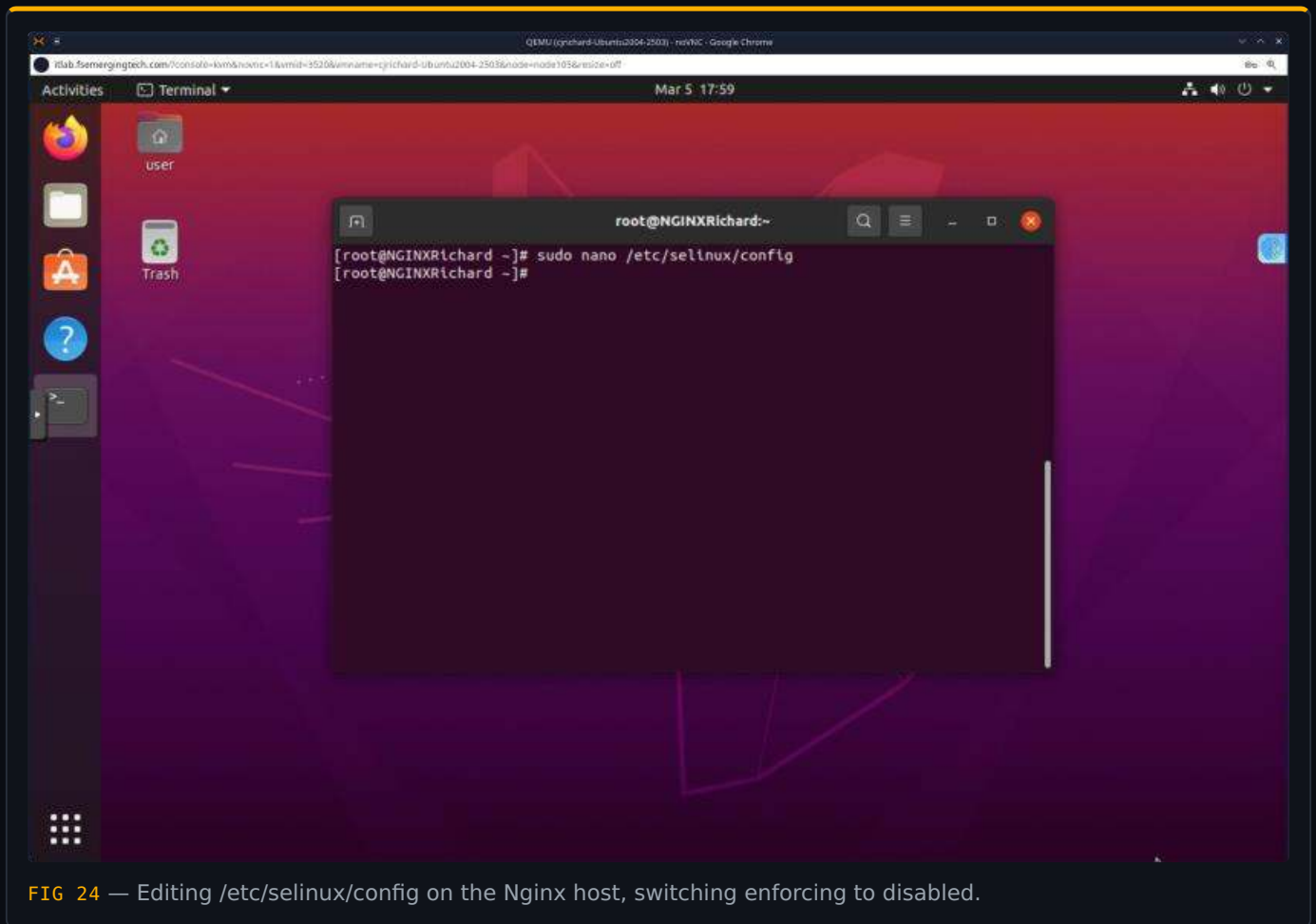


FIG 24 — Editing `/etc/selinux/config` on the Nginx host, switching enforcing to disabled.

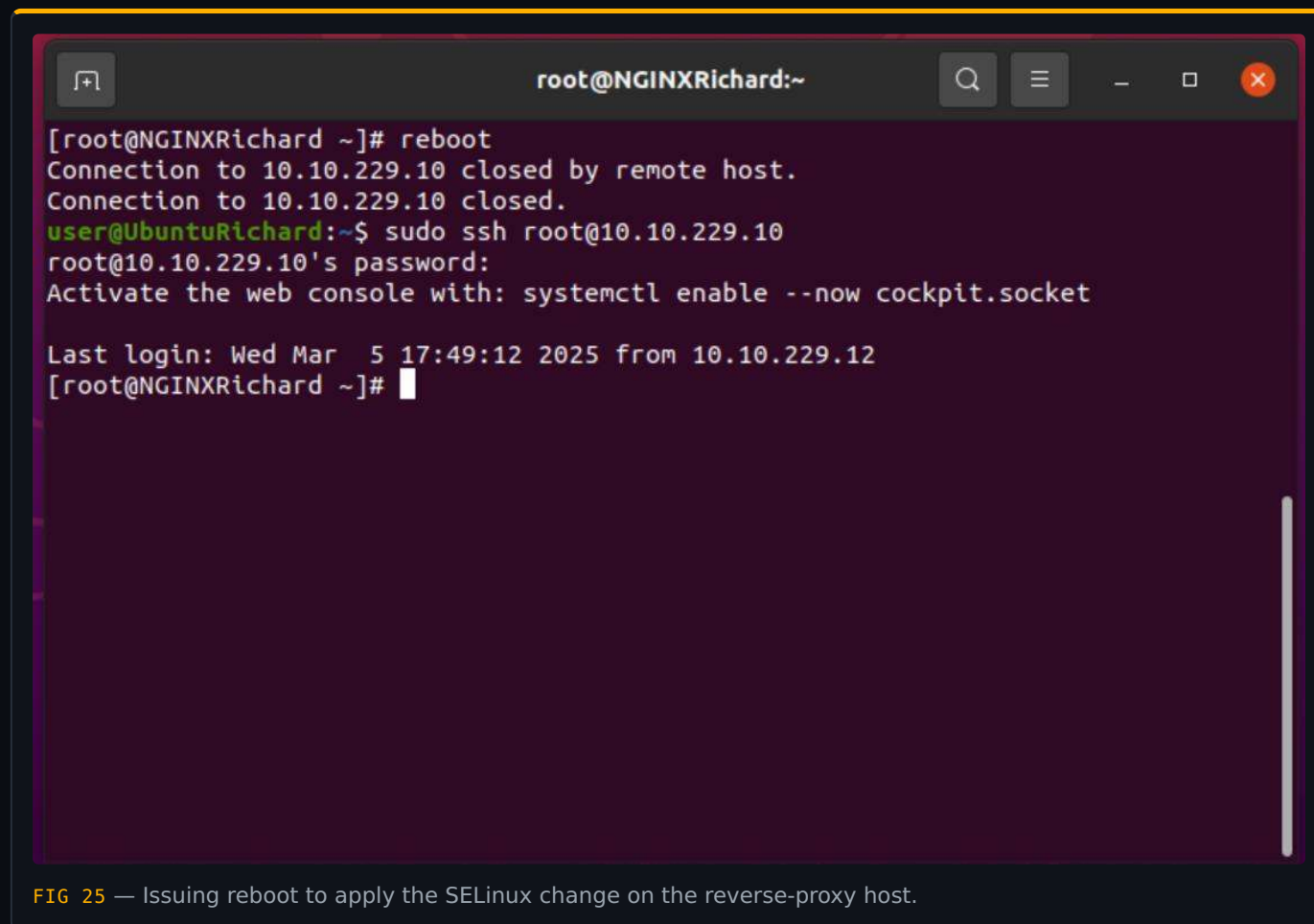


FIG 25 — Issuing reboot to apply the SELinux change on the reverse-proxy host.

## PROXY HOST

## Confirm SELinux and Stop firewalld

Verified SELinux disabled, then stopped firewalld so the proxy could receive HTTP/HTTPS on ports 80 and 443.

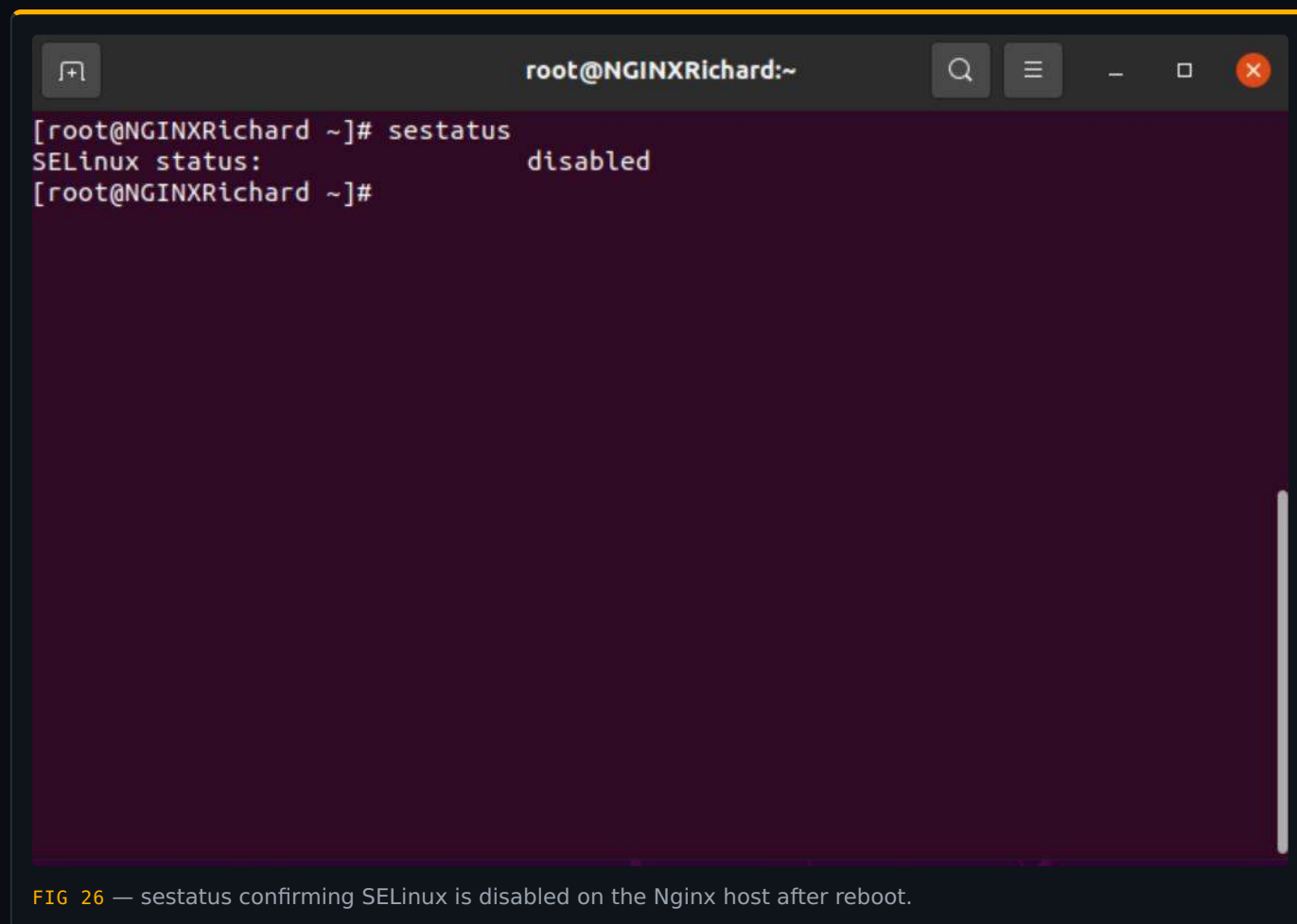
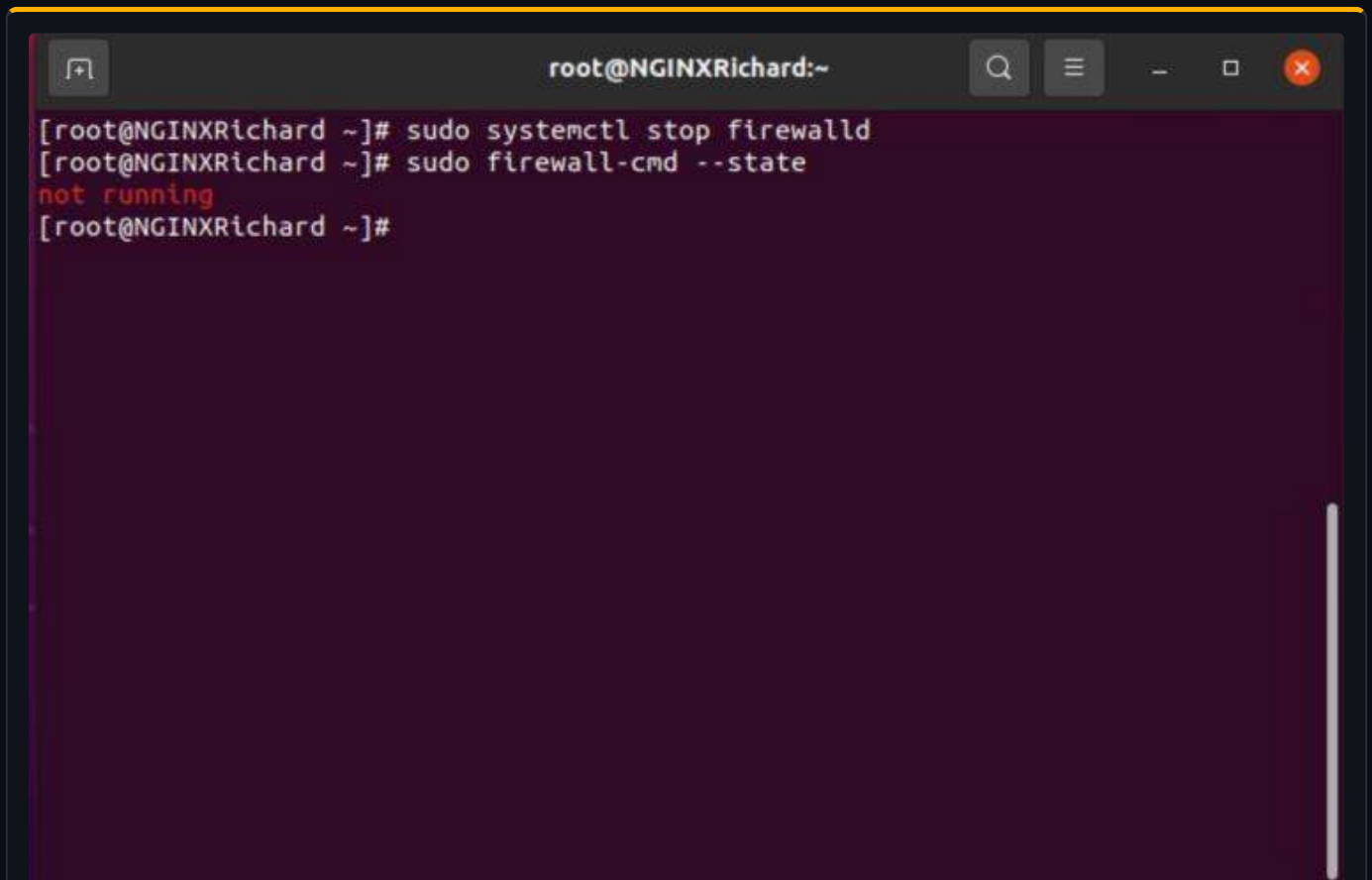


FIG 26 — sestatus confirming SELinux is disabled on the Nginx host after reboot.

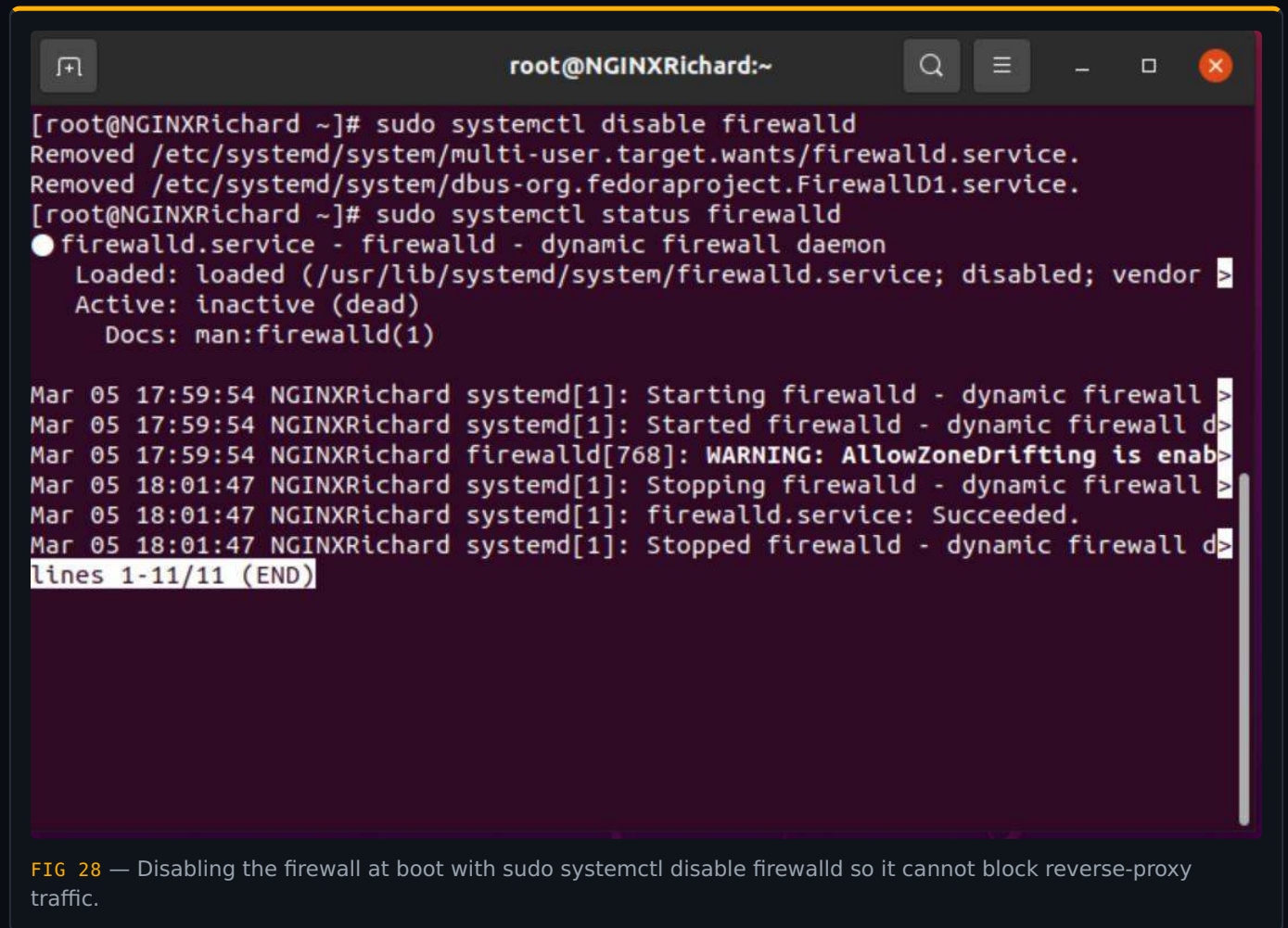
A terminal window titled 'root@NGINXRichard:~' with standard window controls. The terminal shows the following commands and output:

```
[root@NGINXRichard ~]# sudo systemctl stop firewalld
[root@NGINXRichard ~]# sudo firewall-cmd --state
not running
[root@NGINXRichard ~]#
```

FIG 27 — Stopping the firewall with `sudo systemctl stop firewalld` and checking `sudo firewall-cmd --state` to clear ports 80/443 for Nginx.

## PROXY HOST

## Disable firewalld and Install Nginx

A terminal window titled 'root@NGINXRichard:~' showing the execution of systemctl commands to disable the firewalld service. The output shows the service being disabled and then stopped. A warning message is also visible: 'WARNING: AllowZoneDrifting is enabled'. The terminal output is as follows:

```
[root@NGINXRichard ~]# sudo systemctl disable firewalld
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@NGINXRichard ~]# sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)

Mar 05 17:59:54 NGINXRichard systemd[1]: Starting firewalld - dynamic firewall daemon:
Mar 05 17:59:54 NGINXRichard systemd[1]: Started firewalld - dynamic firewall daemon:
Mar 05 17:59:54 NGINXRichard firewalld[768]: WARNING: AllowZoneDrifting is enabled
Mar 05 18:01:47 NGINXRichard systemd[1]: Stopping firewalld - dynamic firewall daemon:
Mar 05 18:01:47 NGINXRichard systemd[1]: firewalld.service: Succeeded.
Mar 05 18:01:47 NGINXRichard systemd[1]: Stopped firewalld - dynamic firewall daemon:
lines 1-11/11 (END)
```

FIG 28 — Disabling the firewall at boot with `sudo systemctl disable firewalld` so it cannot block reverse-proxy traffic.

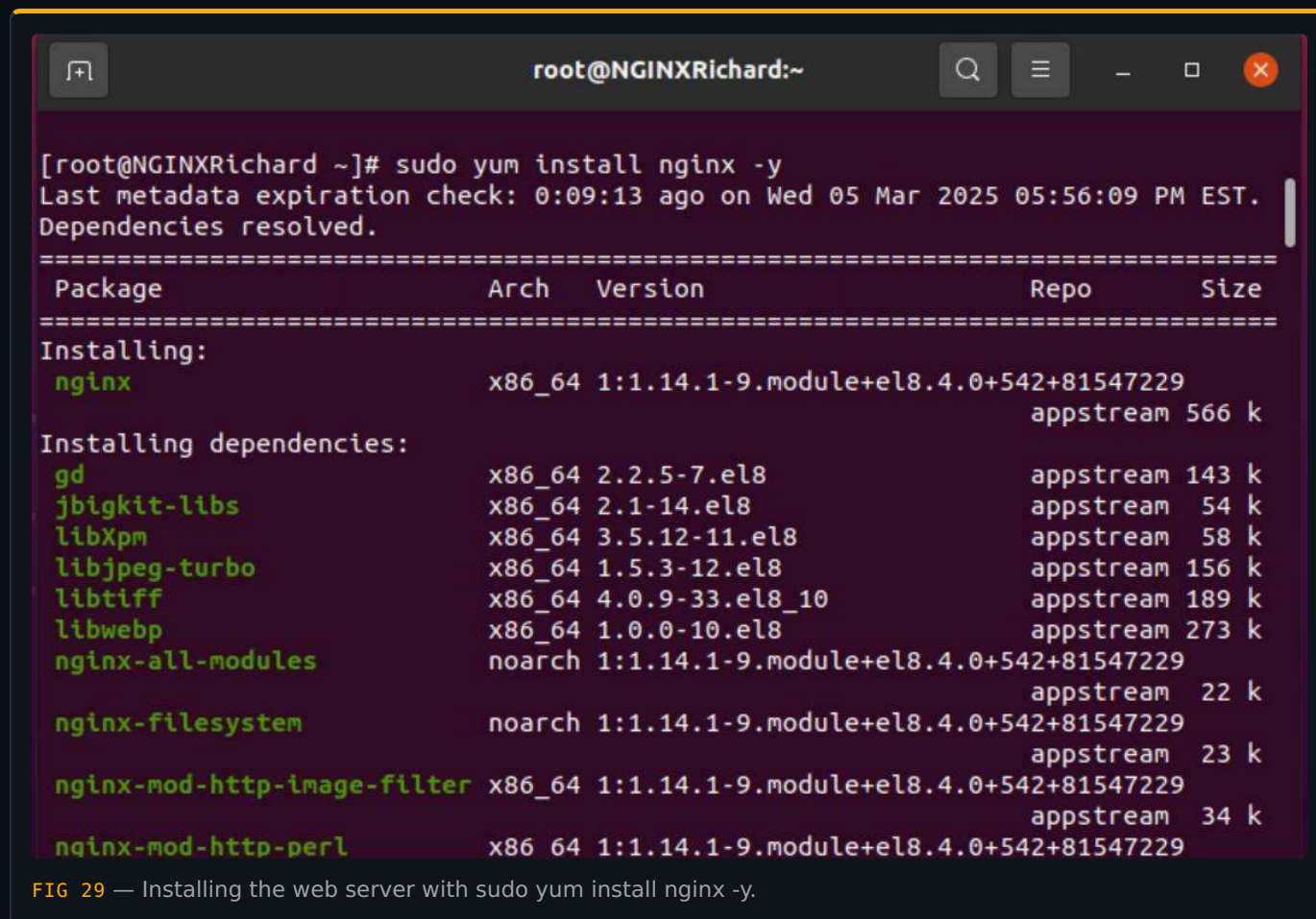
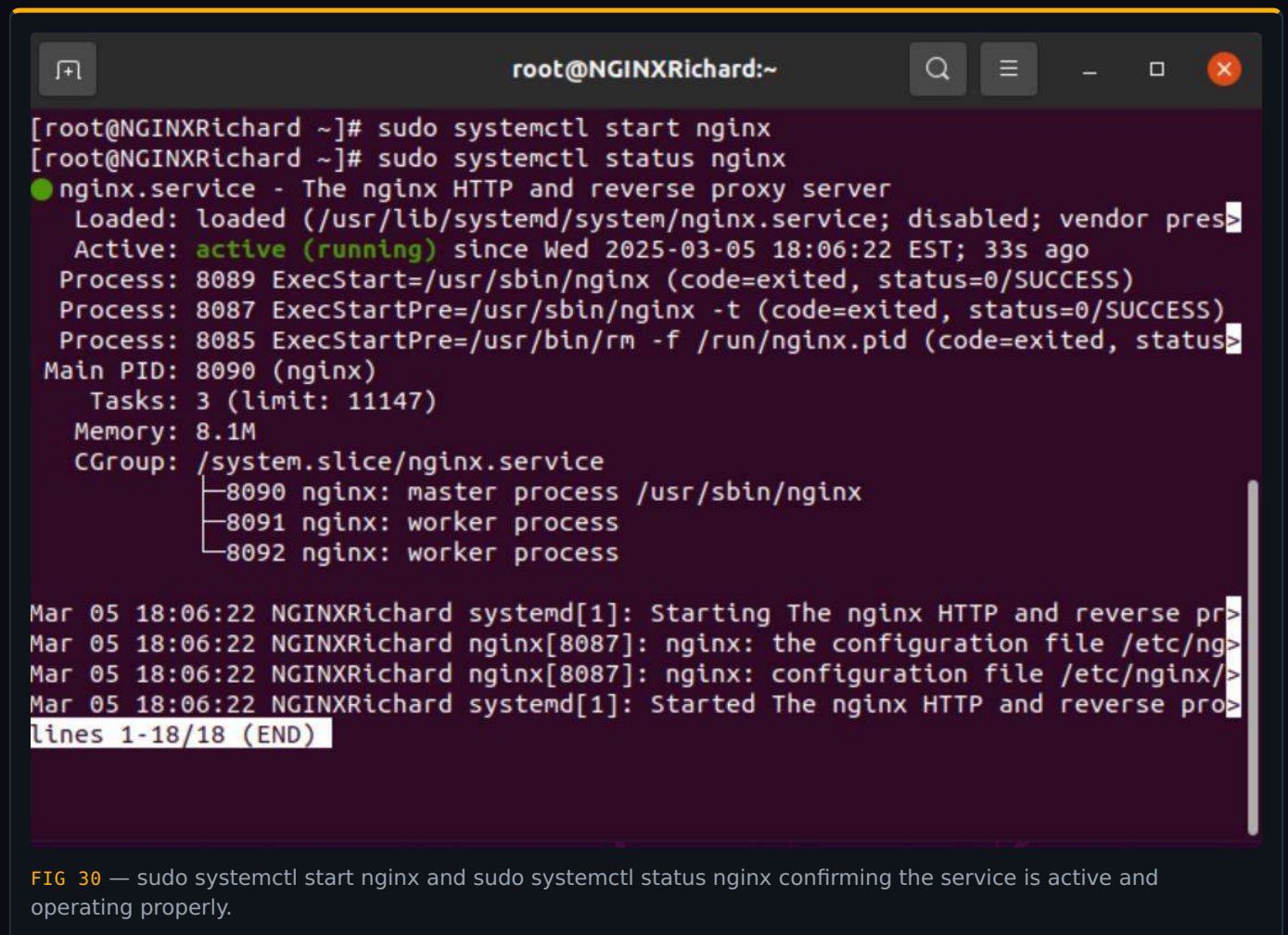


FIG 29 — Installing the web server with sudo yum install nginx -y.

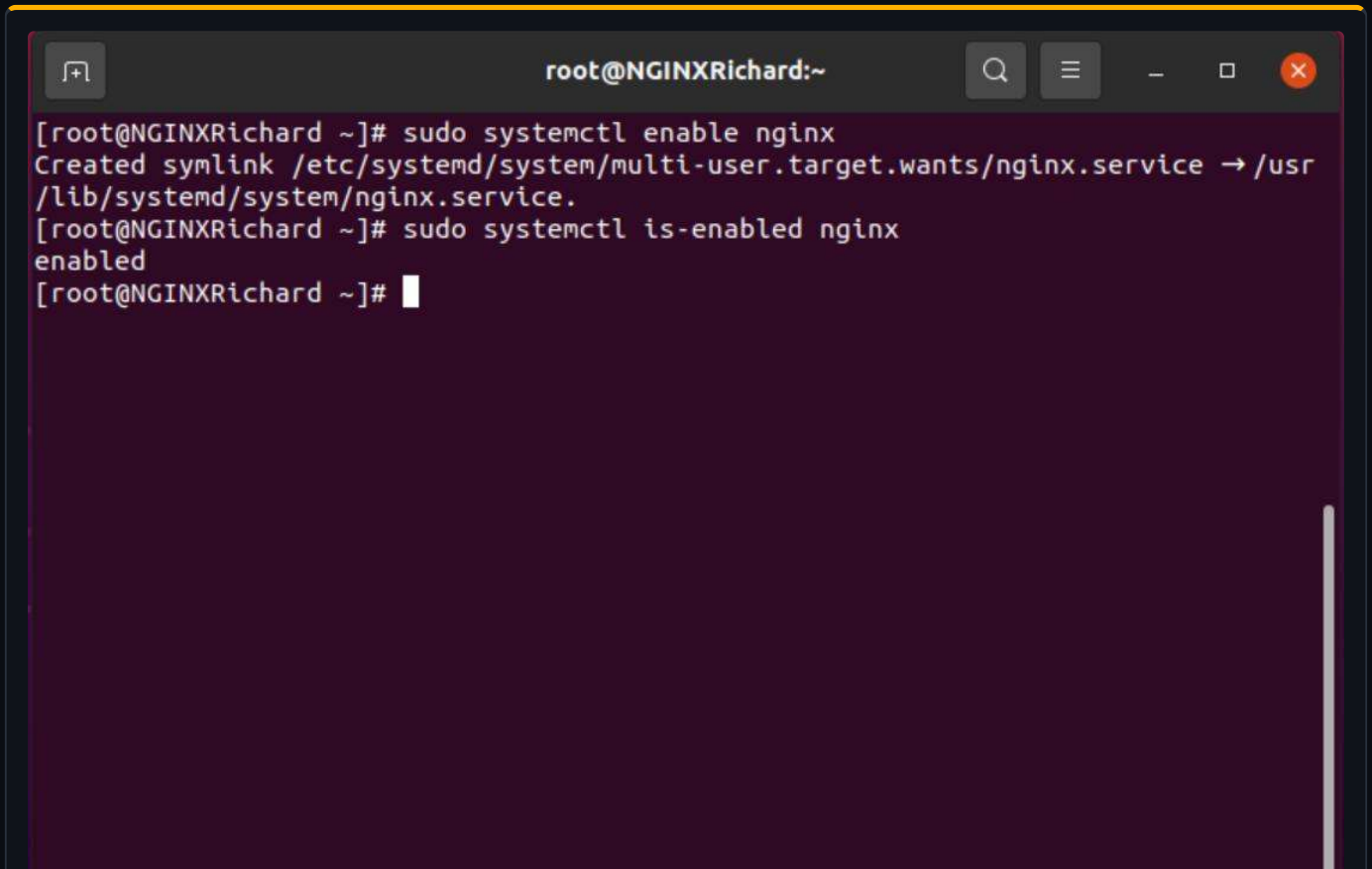
## PROXY HOST

## Start and Enable Nginx



```
root@NGINXRichard:~  
[root@NGINXRichard ~]# sudo systemctl start nginx  
[root@NGINXRichard ~]# sudo systemctl status nginx  
● nginx.service - The nginx HTTP and reverse proxy server  
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor pres  
   Active: active (running) since Wed 2025-03-05 18:06:22 EST; 33s ago  
     Process: 8089 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)  
     Process: 8087 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)  
     Process: 8085 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)  
    Main PID: 8090 (nginx)  
      Tasks: 3 (limit: 11147)  
     Memory: 8.1M  
    CGroup: /system.slice/nginx.service  
            └─8090 nginx: master process /usr/sbin/nginx  
              └─8091 nginx: worker process  
                └─8092 nginx: worker process  
  
Mar 05 18:06:22 NGINXRichard systemd[1]: Starting The nginx HTTP and reverse proxy service:   
Mar 05 18:06:22 NGINXRichard nginx[8087]: nginx: the configuration file /etc/nginx/nginx.conf is not  
Mar 05 18:06:22 NGINXRichard nginx[8087]: nginx: configuration file /etc/nginx/nginx.conf is not  
Mar 05 18:06:22 NGINXRichard systemd[1]: Started The nginx HTTP and reverse proxy service:   
lines 1-18/18 (END)
```

FIG 30 — sudo systemctl start nginx and sudo systemctl status nginx confirming the service is active and operating properly.



```
root@NGINXRichard:~  
[root@NGINXRichard ~]# sudo systemctl enable nginx  
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.  
[root@NGINXRichard ~]# sudo systemctl is-enabled nginx  
enabled  
[root@NGINXRichard ~]#
```

**FIG 31** — sudo systemctl enable nginx and sudo systemctl is-enabled nginx confirming Nginx is set to start at boot.

## REVERSE PROXY

## Confirm Status and Edit nginx.conf

Confirmed the service, then edited the Nginx config to forward /blog traffic to the Ghost backend.

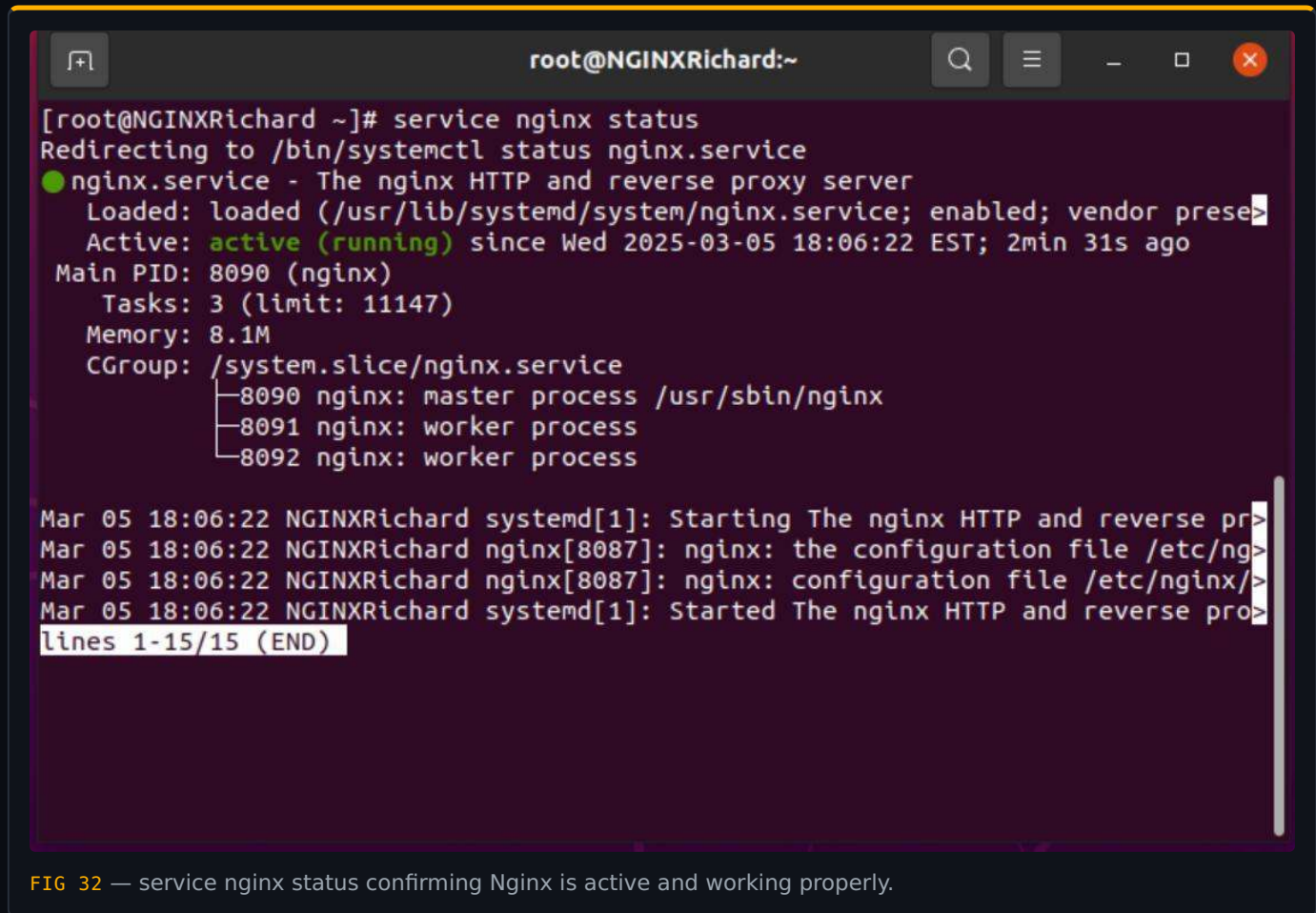


FIG 32 — service nginx status confirming Nginx is active and working properly.

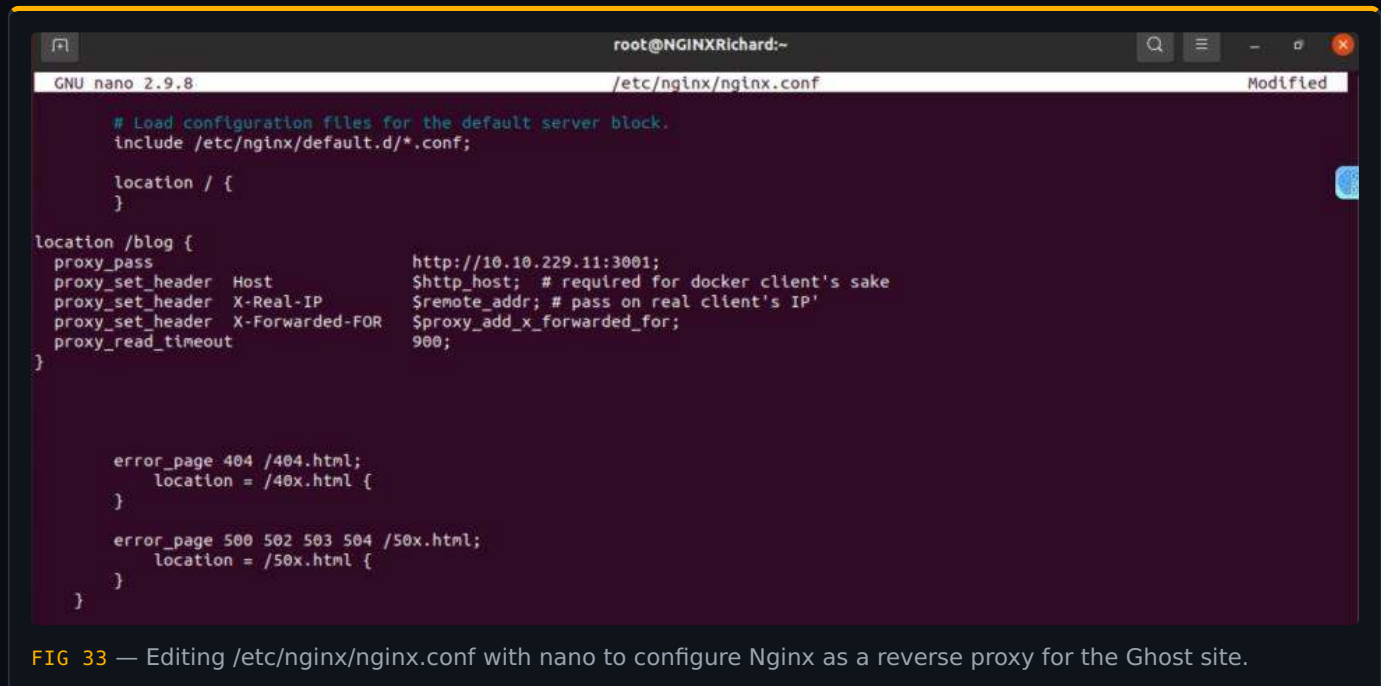
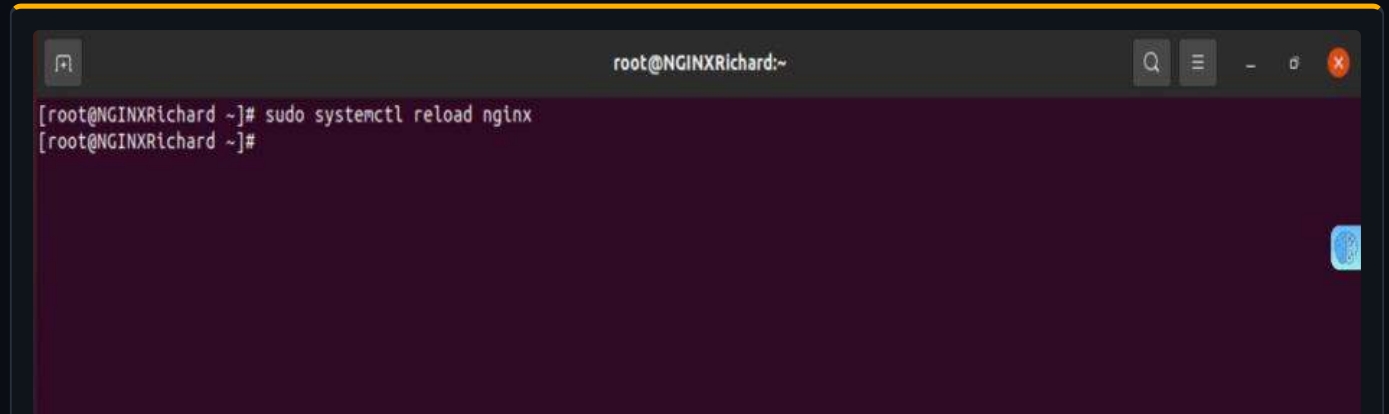


FIG 33 — Editing /etc/nginx/nginx.conf with nano to configure Nginx as a reverse proxy for the Ghost site.

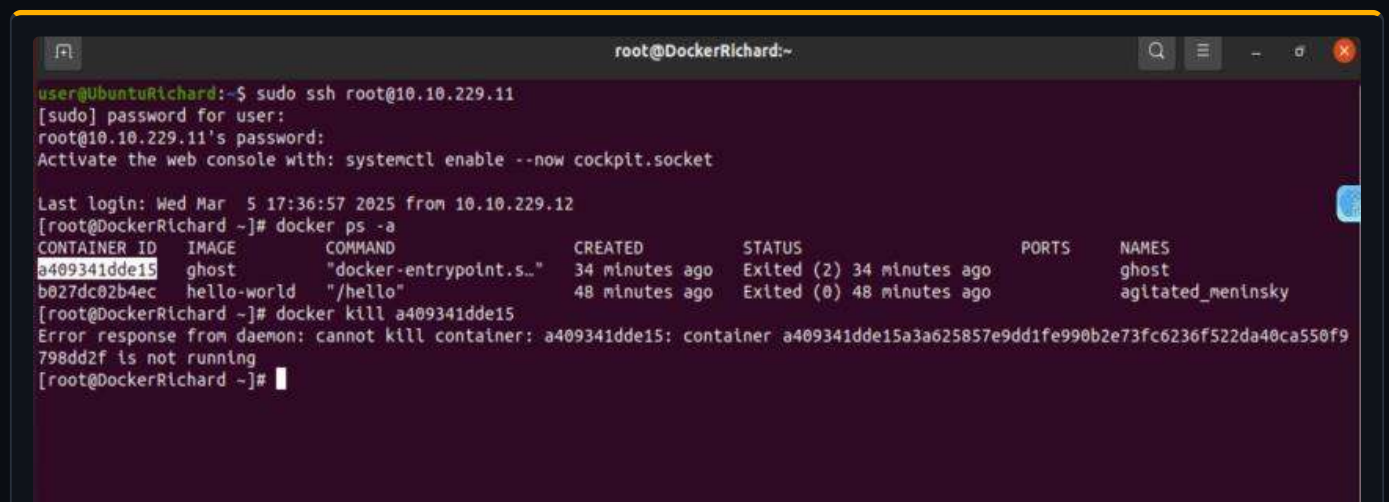
## REVERSE PROXY

## Reverse-Proxy Block and Reload



```
root@NGINXRichard:~  
[root@NGINXRichard ~]# sudo systemctl reload nginx  
[root@NGINXRichard ~]#
```

FIG 34 — The reverse-proxy location block in nginx.conf forwarding client requests to the Ghost container backend.



```
root@DockerRichard:~  
user@UbuntuRichard:~$ sudo ssh root@10.10.229.11  
[sudo] password for user:  
root@10.10.229.11's password:  
Activate the web console with: systemctl enable --now cockpit.socket  
  
Last login: Wed Mar  5 17:36:57 2025 from 10.10.229.12  
[root@DockerRichard ~]# docker ps -a  
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS          PORTS          NAMES  
a409341dde15   ghost         "docker-entrypoint.s..." 34 minutes ago Exited (2) 34 minutes ago ghost  
b027dc02b4ec   hello-world   "/hello"                 48 minutes ago Exited (0) 48 minutes ago agitated_meninsky  
[root@DockerRichard ~]# docker kill a409341dde15  
Error response from daemon: cannot kill container: a409341dde15a3a625857e9dd1fe990b2e73fc6236f522da40ca550f9798dd2f is not running  
[root@DockerRichard ~]#
```

FIG 35 — Applying the new configuration without a restart via `sudo systemctl reload nginx`.

## REVERSE PROXY

## Clear the Stale Ghost Container

Cleared the first, misconfigured Ghost container before rebuilding it against the proxy URL.

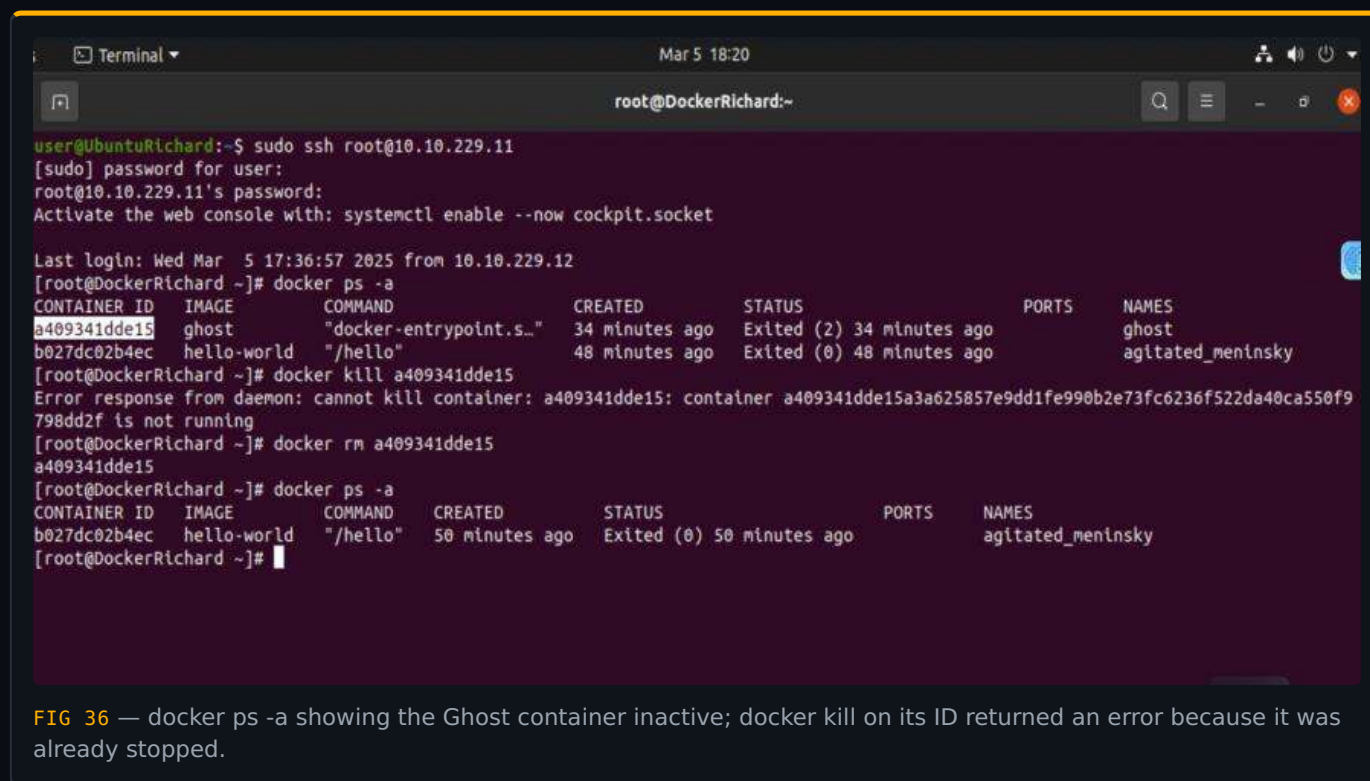


FIG 36 — docker ps -a showing the Ghost container inactive; docker kill on its ID returned an error because it was already stopped.

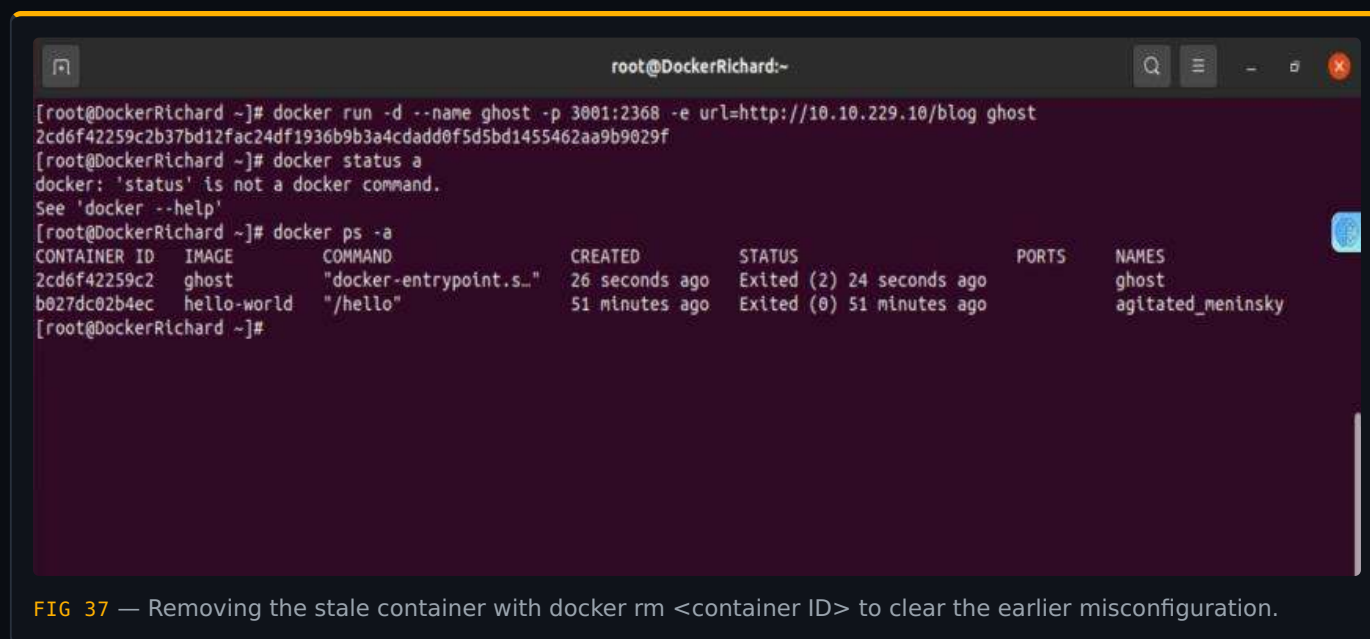


FIG 37 — Removing the stale container with docker rm <container ID> to clear the earlier misconfiguration.

REVERSE PROXY

## Recreate Ghost and Verify Through the Proxy

Recreated the container pointed at the proxy URL, then verified in Firefox that requests reached the reverse proxy.

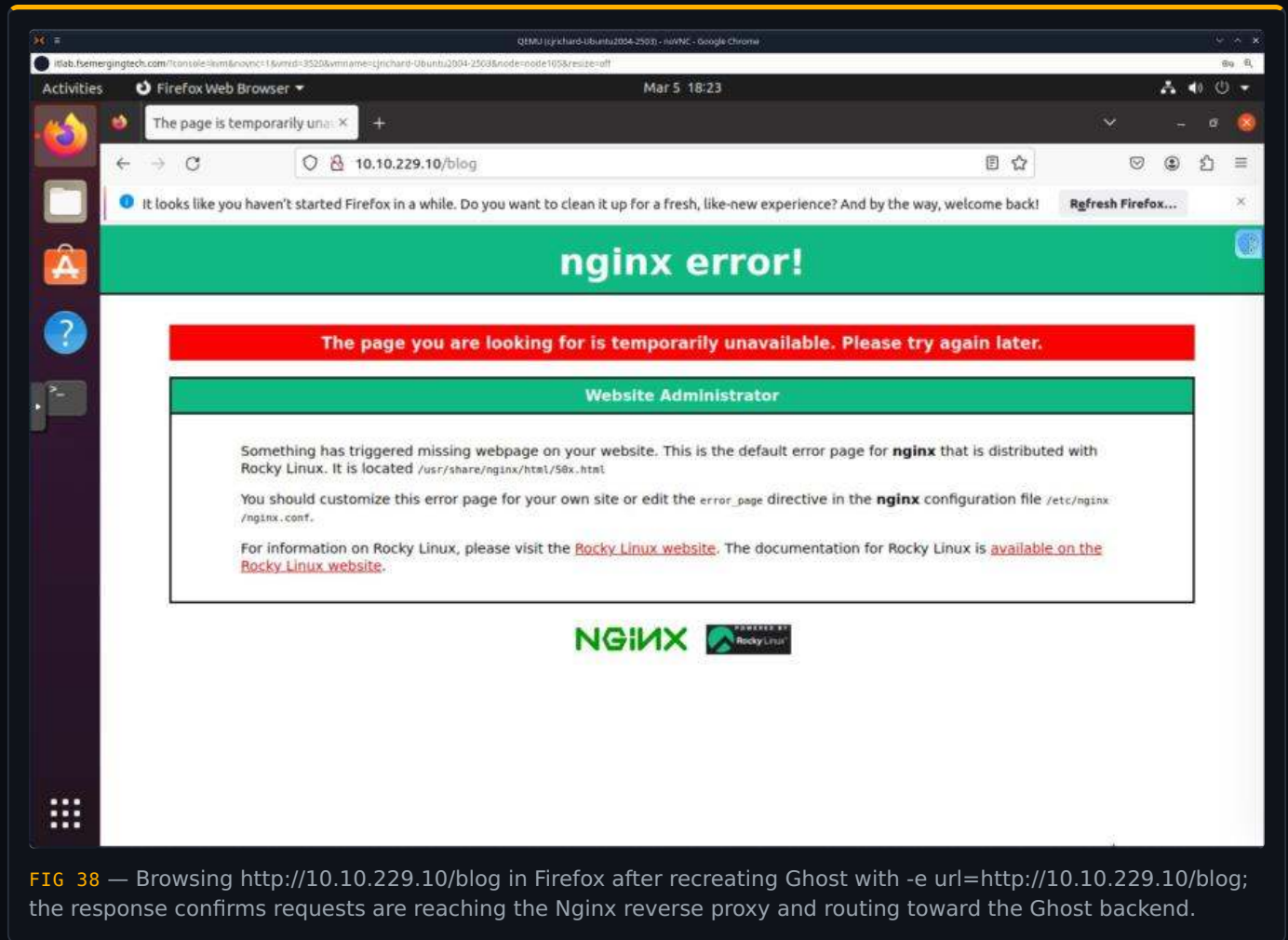


FIG 38 — Browsing `http://10.10.229.10/blog` in Firefox after recreating Ghost with `-e url=http://10.10.229.10/blog`; the response confirms requests are reaching the Nginx reverse proxy and routing toward the Ghost backend.

LAMP STACK

## Set Hostname and Update Ubuntu

Moved to the Ubuntu LAMP host: renamed to UbuntuXRichard and pulled the latest package lists.

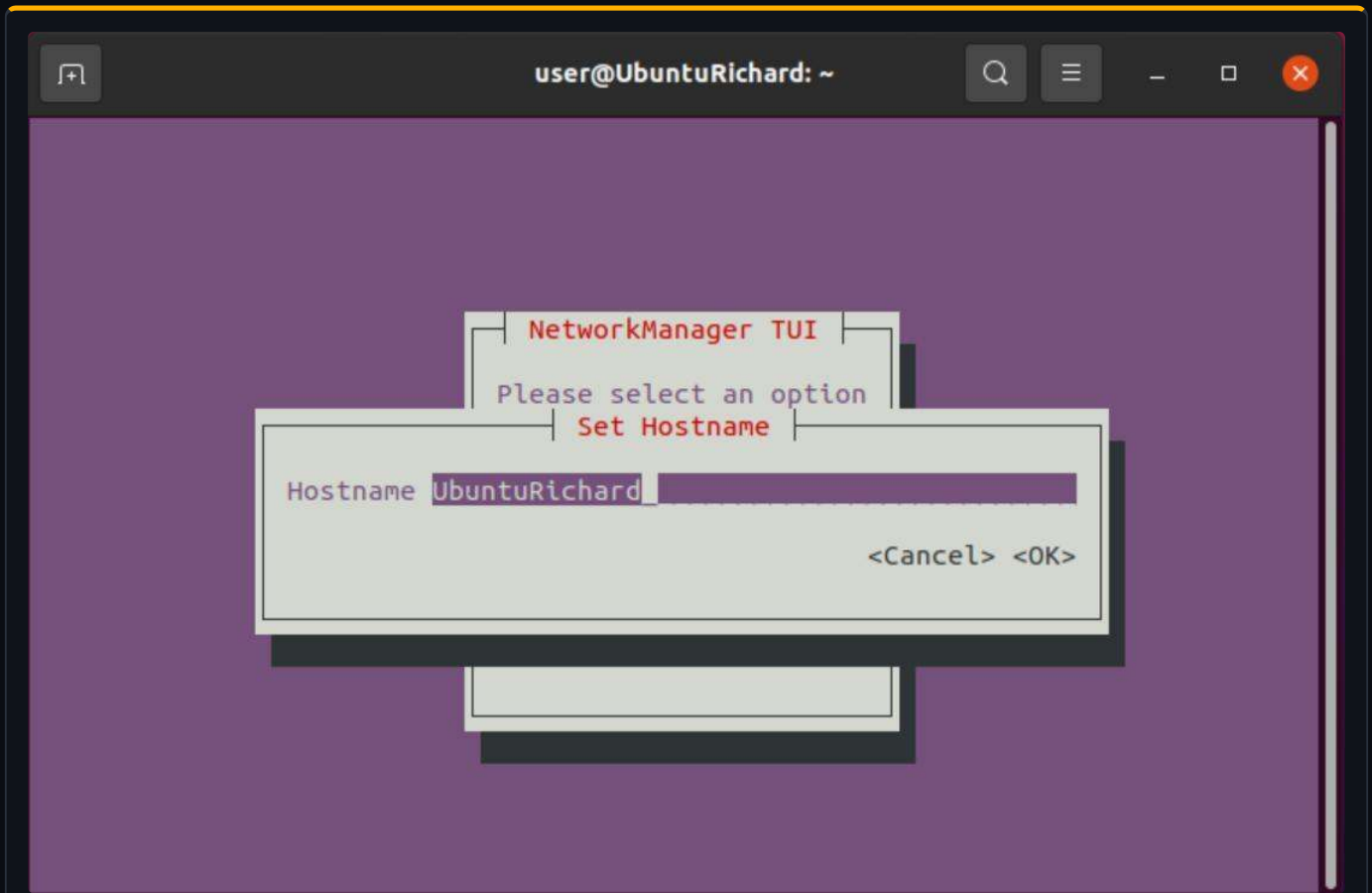
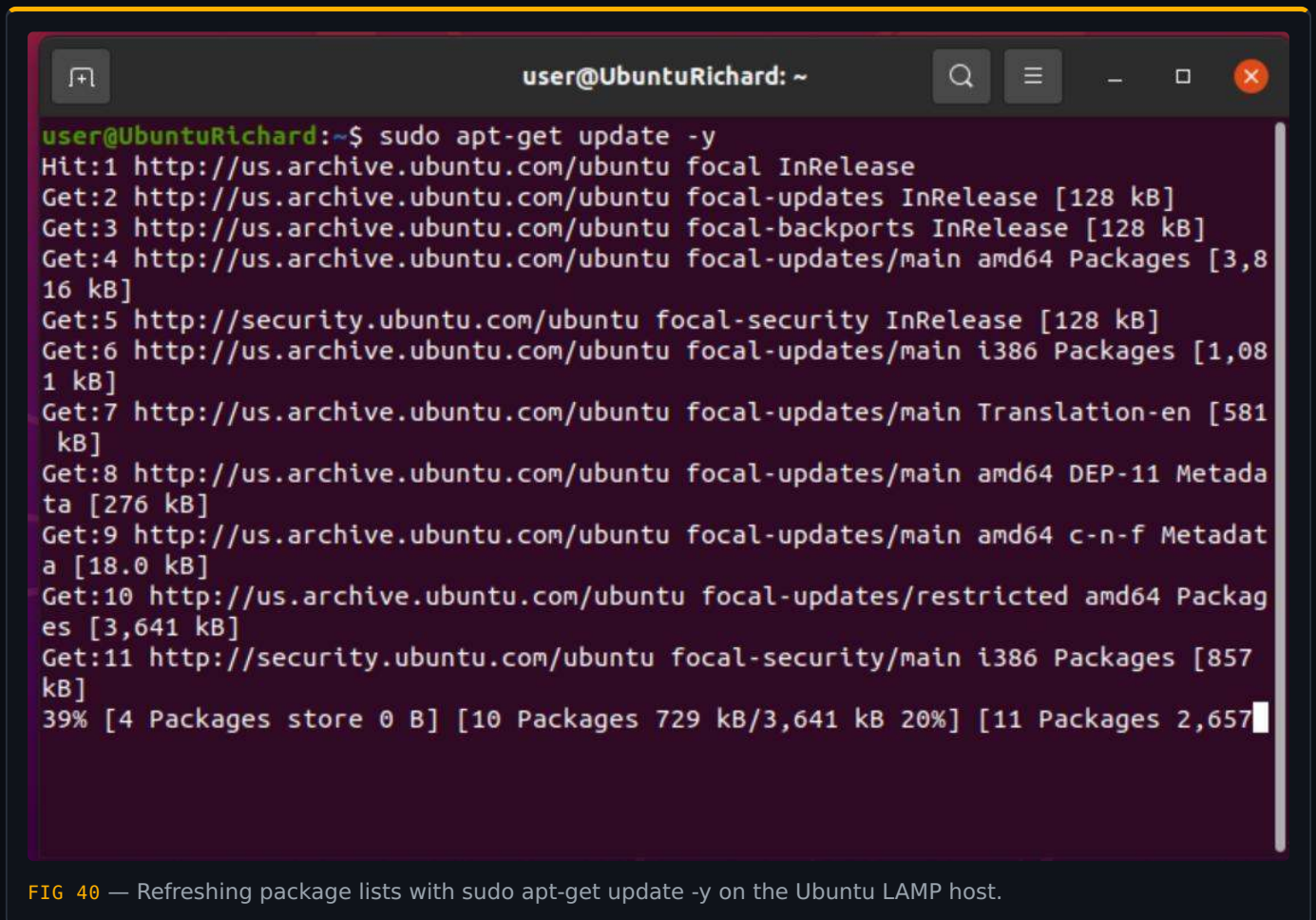


FIG 39 — Ubuntu host renamed to UbuntuXRichard via nmtui for clear identification across the VM fleet.

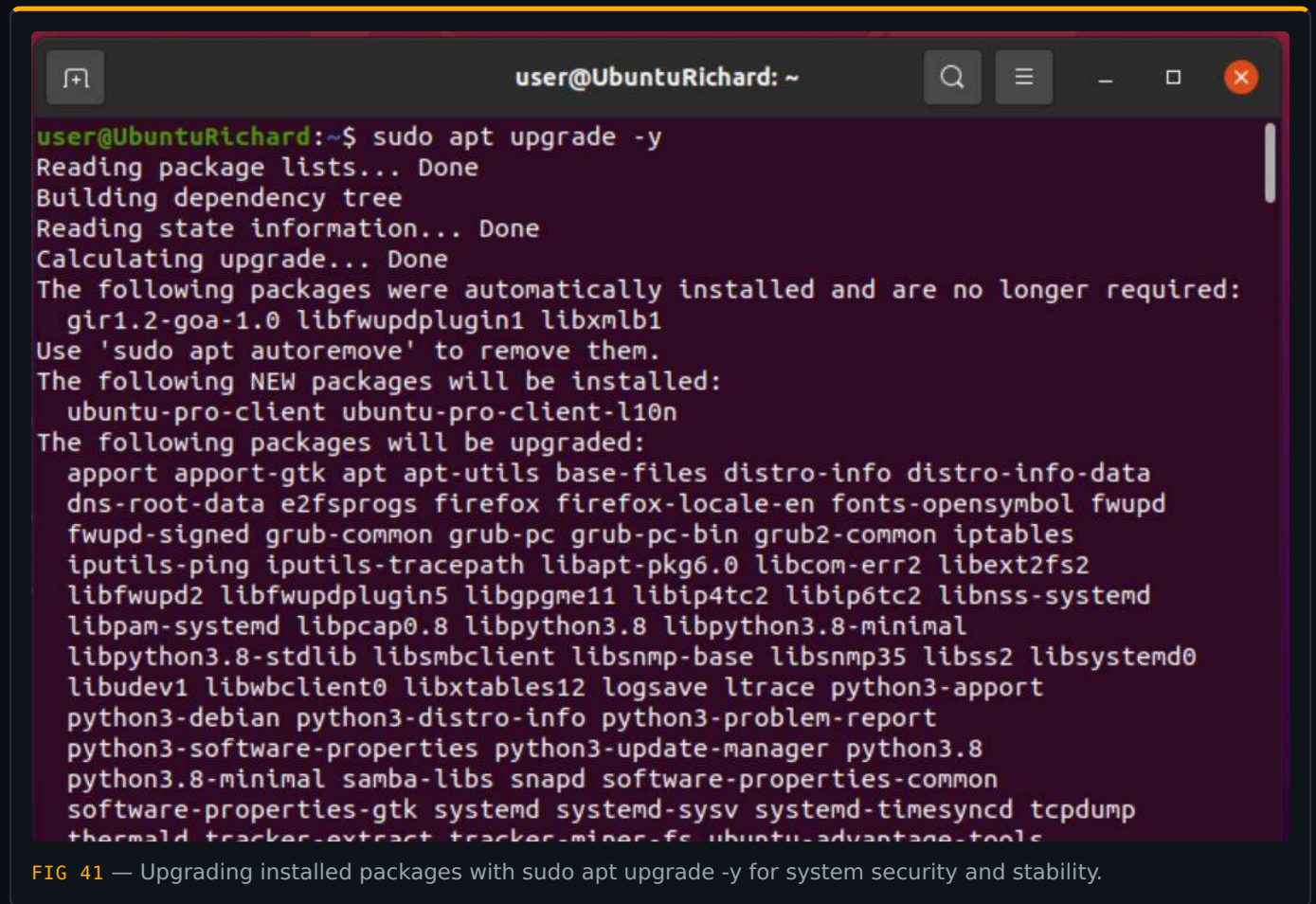
A terminal window titled 'user@UbuntuRichard: ~' with standard window controls. The terminal displays the output of the command 'sudo apt-get update -y'. The output shows the process of refreshing package lists from various sources, including the main Ubuntu archive and security updates. It lists several sources and the packages they contain, along with their sizes. The progress bar at the bottom indicates that 39% of the update process is complete, with 4 packages stored and 10 packages (729 kB) downloaded out of a total of 3,641 kB. The terminal text is as follows:

```
user@UbuntuRichard:~$ sudo apt-get update -y
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3,816 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [1,081 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [581 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [276 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [18.0 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3,641 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [857 kB]
39% [4 Packages store 0 B] [10 Packages 729 kB/3,641 kB 20%] [11 Packages 2,657
```

FIG 40 — Refreshing package lists with `sudo apt-get update -y` on the Ubuntu LAMP host.

## LAMP STACK

## Upgrade Packages and Install Nano



```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ sudo apt upgrade -y  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Calculating upgrade... Done  
The following packages were automatically installed and are no longer required:  
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1  
Use 'sudo apt autoremove' to remove them.  
The following NEW packages will be installed:  
  ubuntu-pro-client ubuntu-pro-client-l10n  
The following packages will be upgraded:  
  apport apport-gtk apt apt-utils base-files distro-info distro-info-data  
  dns-root-data e2fsprogs firefox firefox-locale-en fonts-opensymbol fwupd  
  fwupd-signed grub-common grub-pc grub-pc-bin grub2-common iptables  
  iputils-ping iputils-tracepath libapt-pkg6.0 libcom-err2 libext2fs2  
  libfwupd2 libfwupdplugin5 libgpgme11 libip4tc2 libip6tc2 libnss-systemd  
  libpam-systemd libpcap0.8 libpython3.8 libpython3.8-minimal  
  libpython3.8-stdlib libsmbclient libsnmp-base libsnmp35 libss2 libsystemd0  
  libudev1 libwbclient0 libxtables12 logsave ltrace python3-apport  
  python3-debian python3-distro-info python3-problem-report  
  python3-software-properties python3-update-manager python3.8  
  python3.8-minimal samba-libs snapd software-properties-common  
  software-properties-gtk systemd systemd-sysv systemd-timesyncd tcpdump  
  thermald tracker-extract tracker-miner-fs ubuntu-advantage-tools
```

FIG 41 — Upgrading installed packages with `sudo apt upgrade -y` for system security and stability.

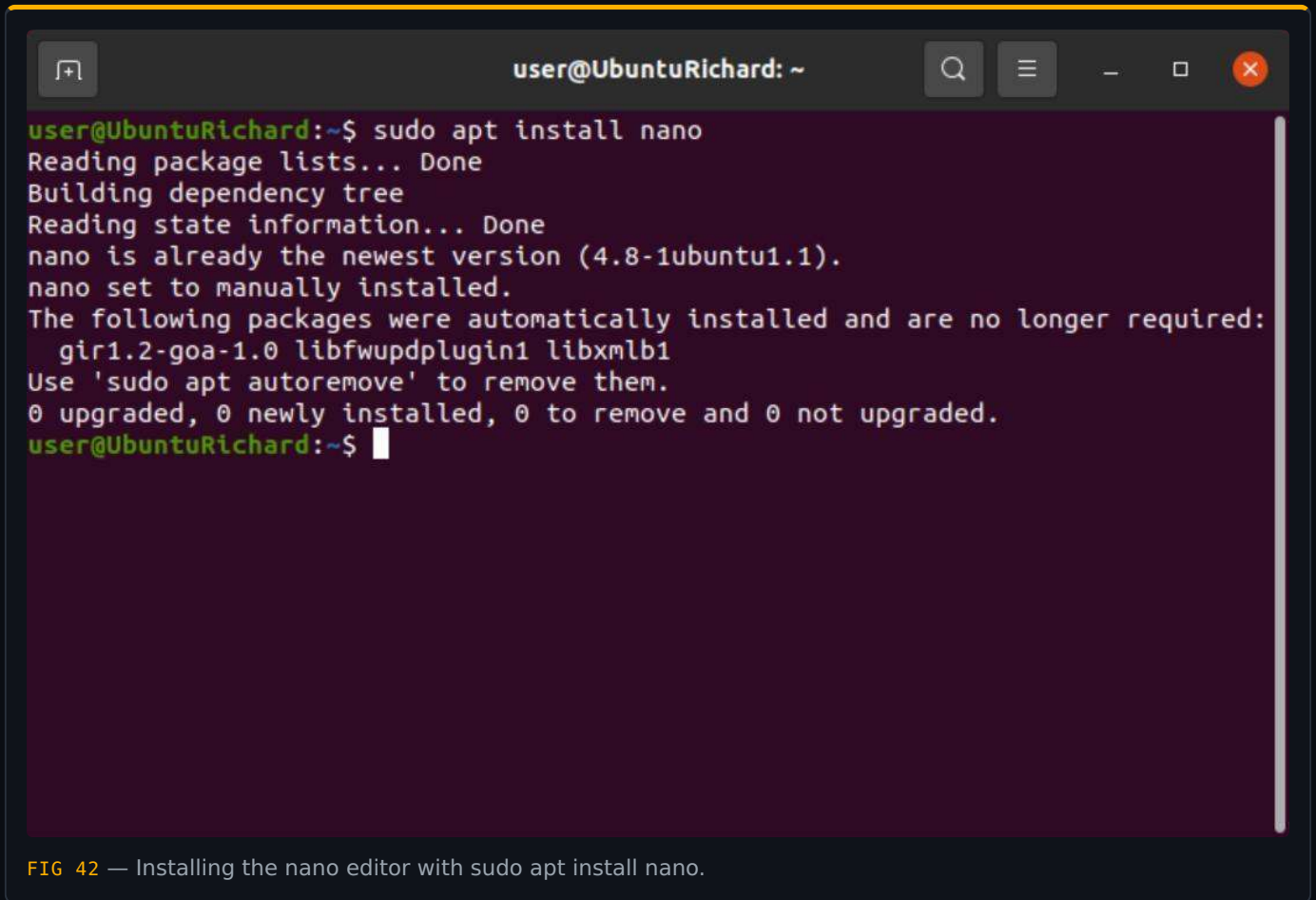
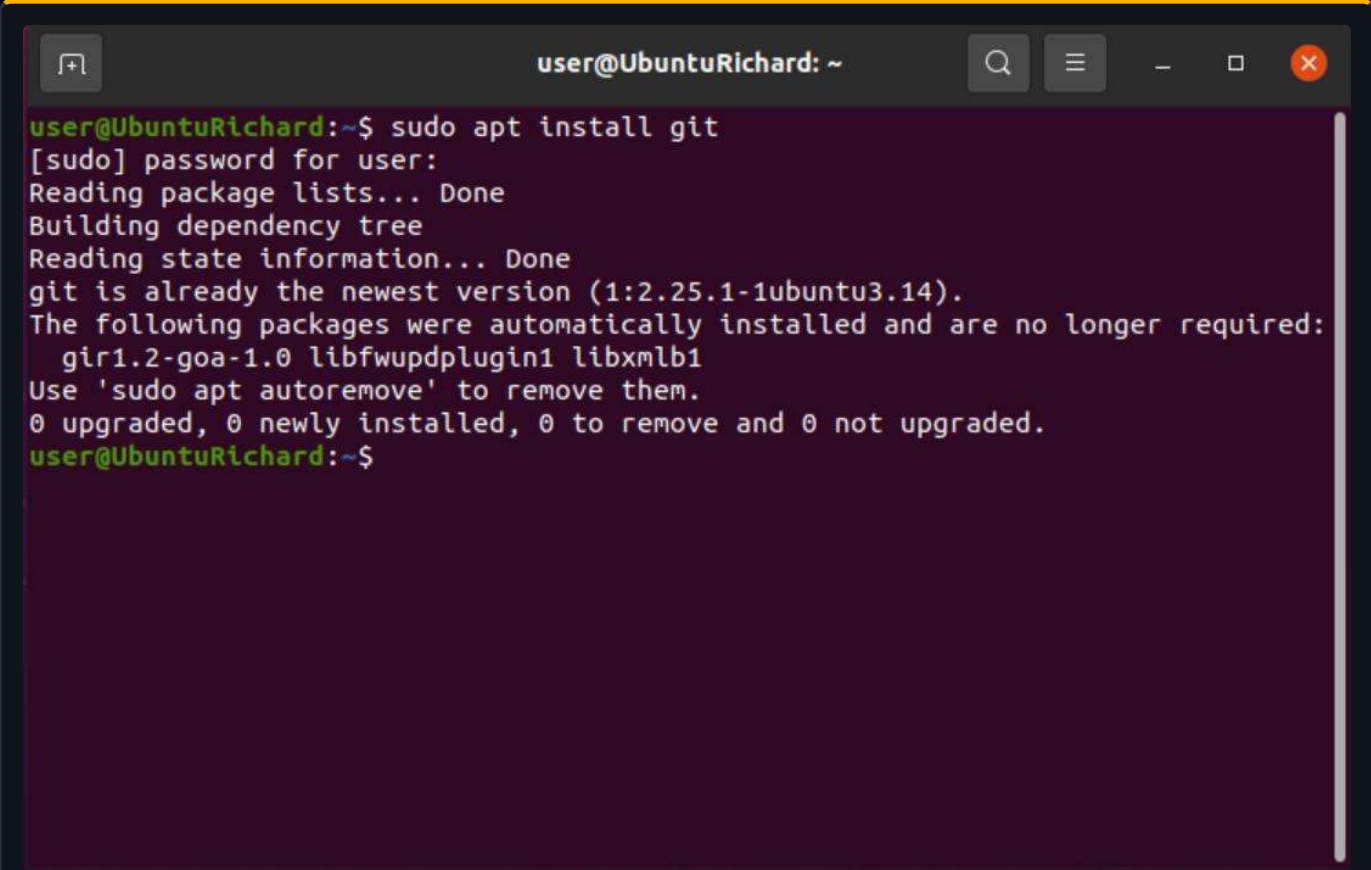


FIG 42 — Installing the nano editor with sudo apt install nano.

## LAMP STACK

## Install Git and Apache2

Installed Git (to clone WordPress later) and the Apache2 web server.



```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ sudo apt install git  
[sudo] password for user:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
git is already the newest version (1:2.25.1-1ubuntu3.14).  
The following packages were automatically installed and are no longer required:  
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
user@UbuntuRichard:~$
```

FIG 43 — Installing Git with `sudo apt install git` to enable cloning WordPress from GitHub.

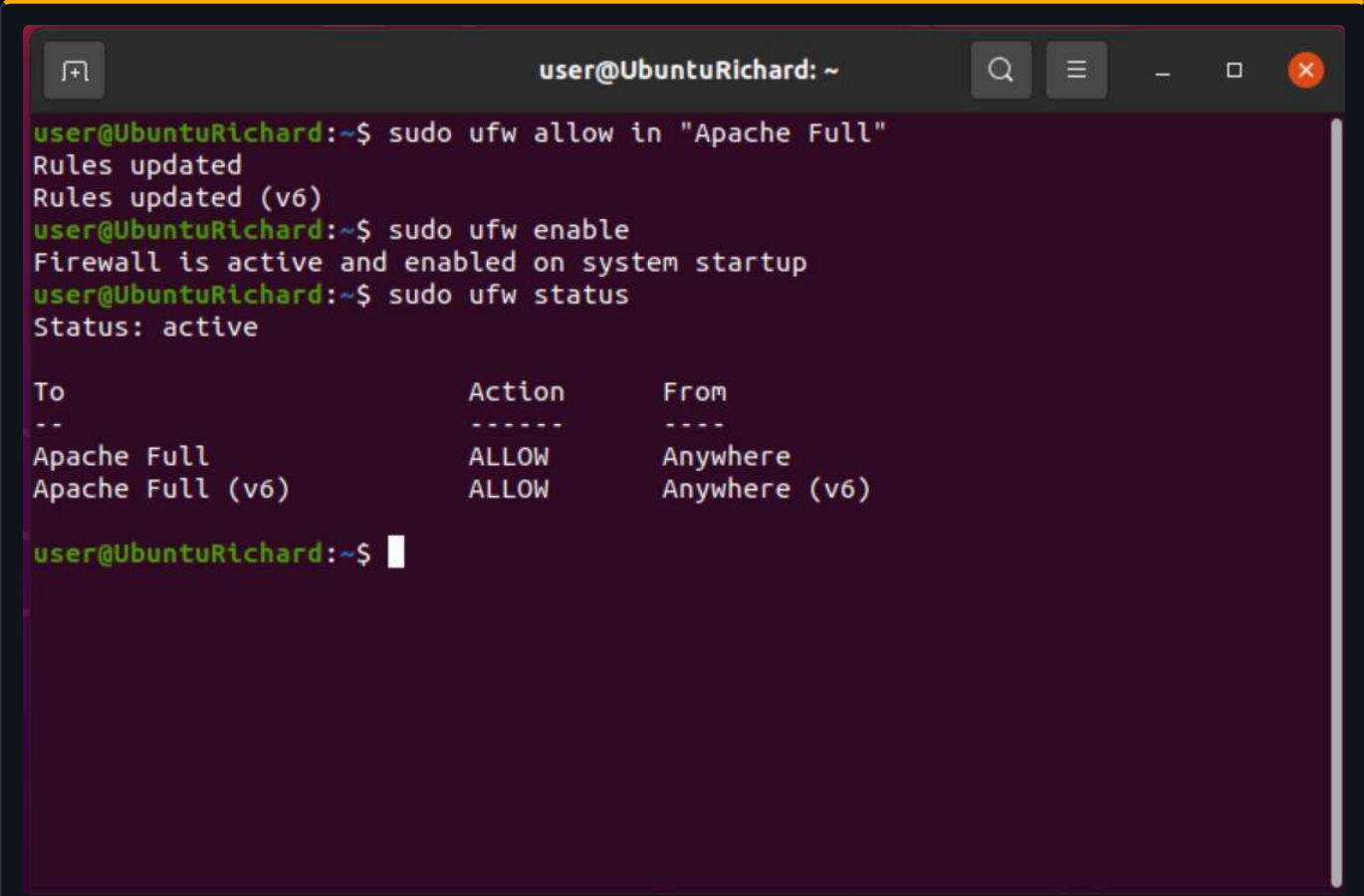
```
user@UbuntuRichard: ~
user@UbuntuRichard:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 liblua5.2-0
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 liblua5.2-0
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,064 kB of archives.
After this operation, 8,692 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libapr1 amd64
  1.6.5-1ubuntu1.1 [91.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libaprutil1 a
  md64 1.6.1-4ubuntu2.2 [85.1 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libaprutil1-d
  bd-sqlite3 amd64 1.6.1-4ubuntu2.2 [10.5 kB]
```

FIG 44 — Installing the web server with `sudo apt install apache2` to host the WordPress site.

## LAMP STACK

## Open Firewall Ports and Verify Apache

Allowed Apache through UFW on ports 80/443 and confirmed the default Apache page over the network.

A terminal window titled 'user@UbuntuRichard: ~' with search, menu, and window control icons. The terminal shows the following commands and output:

```
user@UbuntuRichard:~$ sudo ufw allow in "Apache Full"
Rules updated
Rules updated (v6)
user@UbuntuRichard:~$ sudo ufw enable
Firewall is active and enabled on system startup
user@UbuntuRichard:~$ sudo ufw status
Status: active

To Action From
-- -- --
Apache Full ALLOW Anywhere
Apache Full (v6) ALLOW Anywhere (v6)

user@UbuntuRichard:~$
```

**FIG 45** — `sudo ufw allow in 'Apache Full'`, `sudo ufw enable`, and `sudo ufw status` opening HTTP/HTTPS ports 80 and 443 and confirming the rules are active.

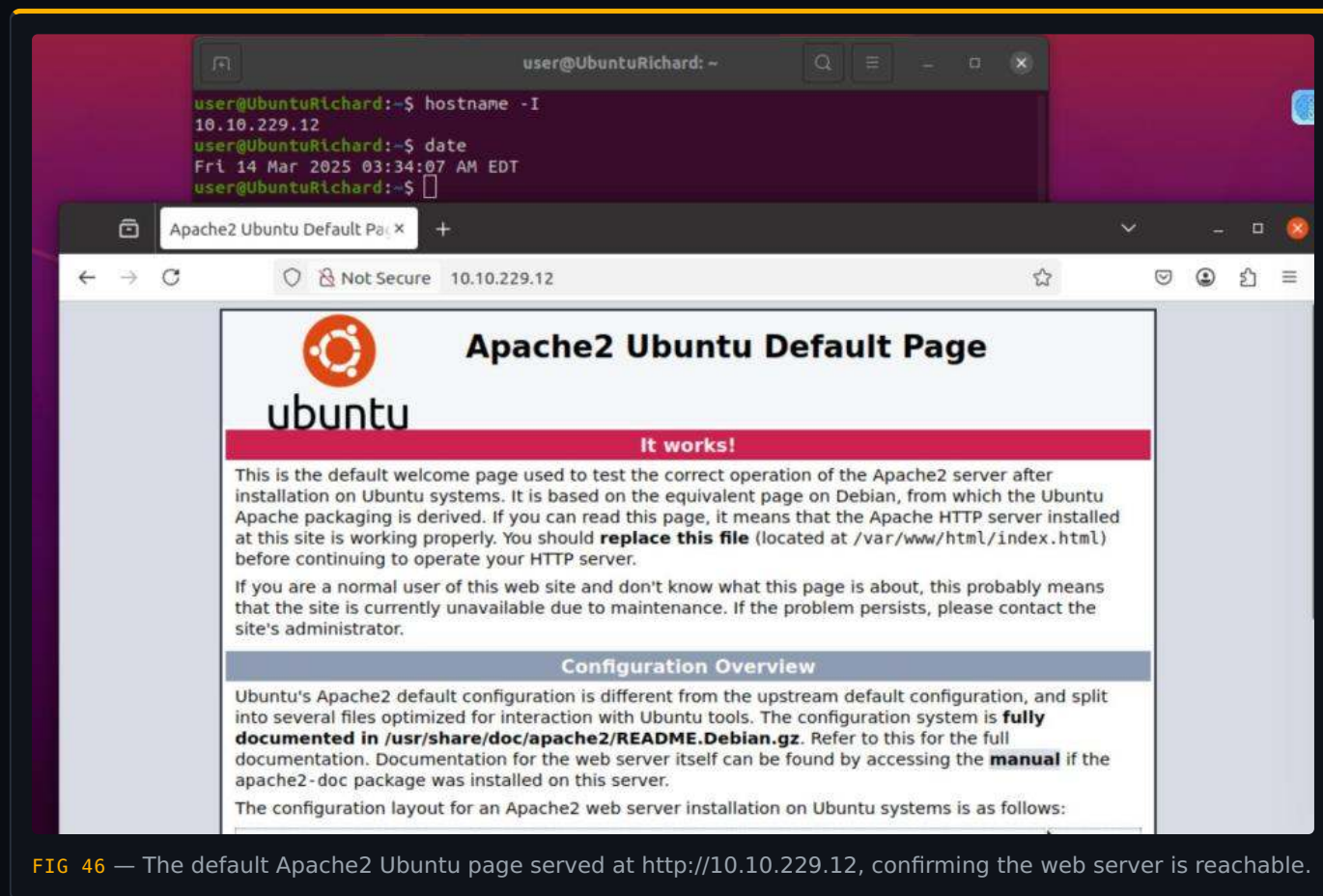
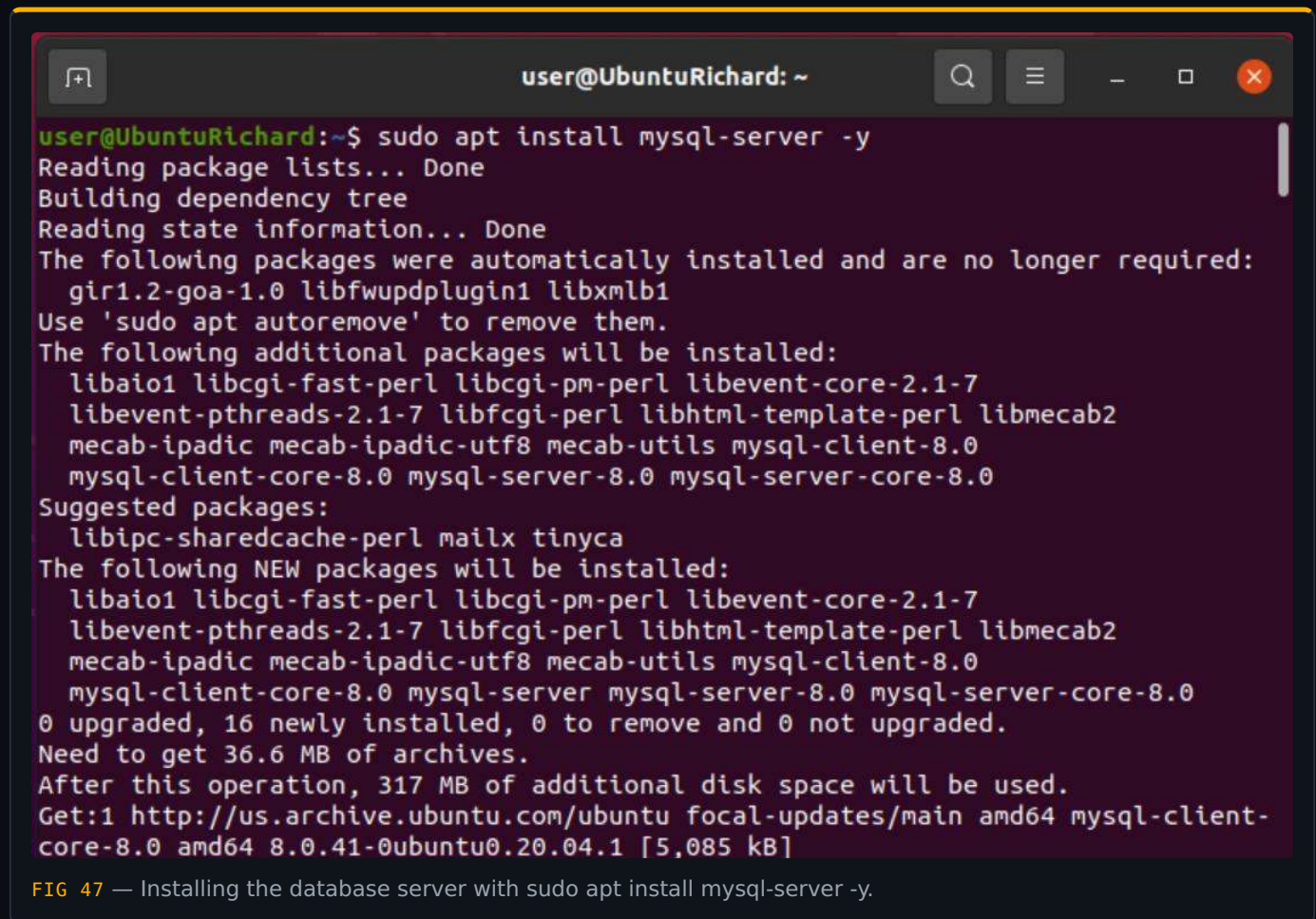


FIG 46 — The default Apache2 Ubuntu page served at `http://10.10.229.12`, confirming the web server is reachable.

## LAMP STACK

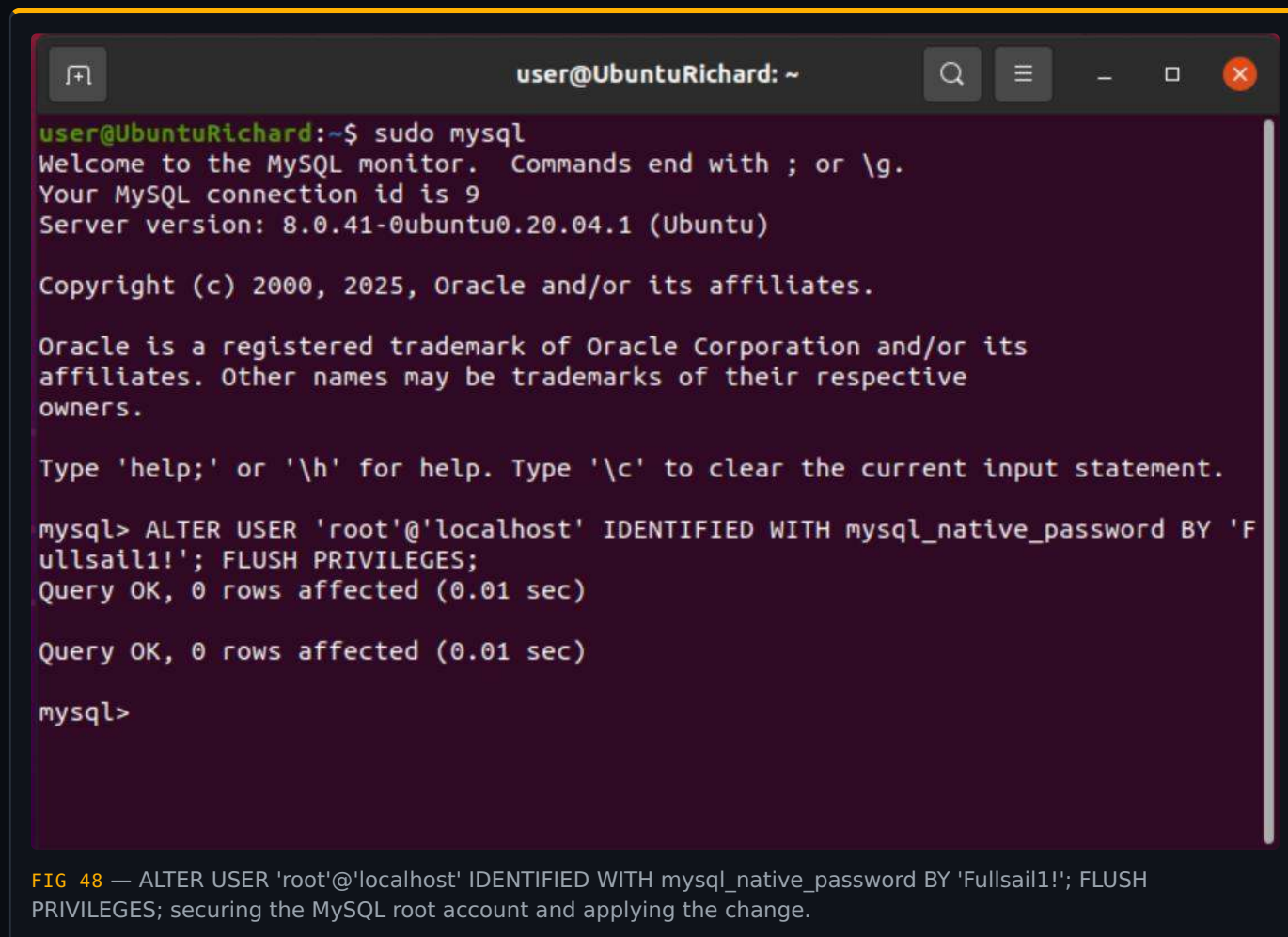
## Install MySQL and Secure root

Installed the MySQL server and set the root password using native-password authentication.

A terminal window titled 'user@UbuntuRichard: ~' showing the command 'sudo apt install mysql-server -y' and its output. The output details the installation process, including reading package lists, building a dependency tree, and listing packages to be installed. It also shows the disk space requirements and the source of the packages.

```
user@UbuntuRichard:~$ sudo apt install mysql-server -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-template-perl libmecab2
  mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0
  mysql-client-core-8.0 mysql-server-8.0 mysql-server-core-8.0
Suggested packages:
  libipc-sharedcache-perl mailx tinyca
The following NEW packages will be installed:
  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-template-perl libmecab2
  mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0
  mysql-client-core-8.0 mysql-server mysql-server-8.0 mysql-server-core-8.0
0 upgraded, 16 newly installed, 0 to remove and 0 not upgraded.
Need to get 36.6 MB of archives.
After this operation, 317 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 mysql-client-
core-8.0 amd64 8.0.41-0ubuntu0.20.04.1 [5,085 kB]
```

FIG 47 — Installing the database server with `sudo apt install mysql-server -y`.

A terminal window titled 'user@UbuntuRichard: ~' with standard window controls. The terminal shows the execution of 'sudo mysql' which opens the MySQL monitor. The user enters the command 'ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'Fullsail1!'; FLUSH PRIVILEGES;'. The terminal displays two 'Query OK, 0 rows affected' messages, indicating successful execution. The prompt returns to 'mysql>'.

```
user@UbuntuRichard:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Fullsail1!'; FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

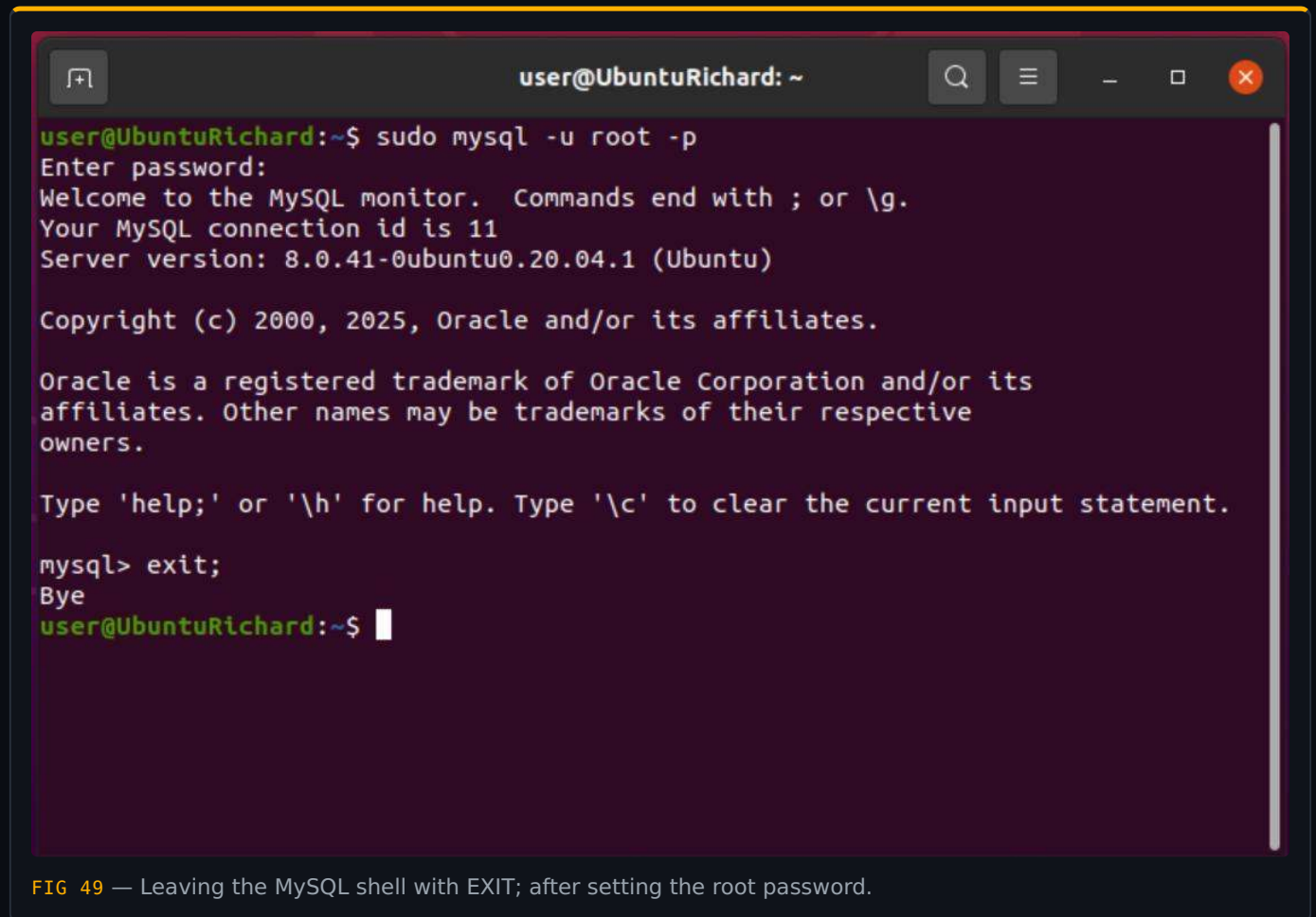
Query OK, 0 rows affected (0.01 sec)

mysql>
```

FIG 48 — ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'Fullsail1!'; FLUSH PRIVILEGES; securing the MySQL root account and applying the change.

## LAMP STACK

## Exit MySQL and Install PHP

A terminal window titled 'user@UbuntuRichard: ~' with search, menu, and window control icons. The terminal shows the execution of 'sudo mysql -u root -p', followed by a password prompt and MySQL welcome messages. The user enters 'exit;' and the terminal returns to the shell prompt.

```
user@UbuntuRichard:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)

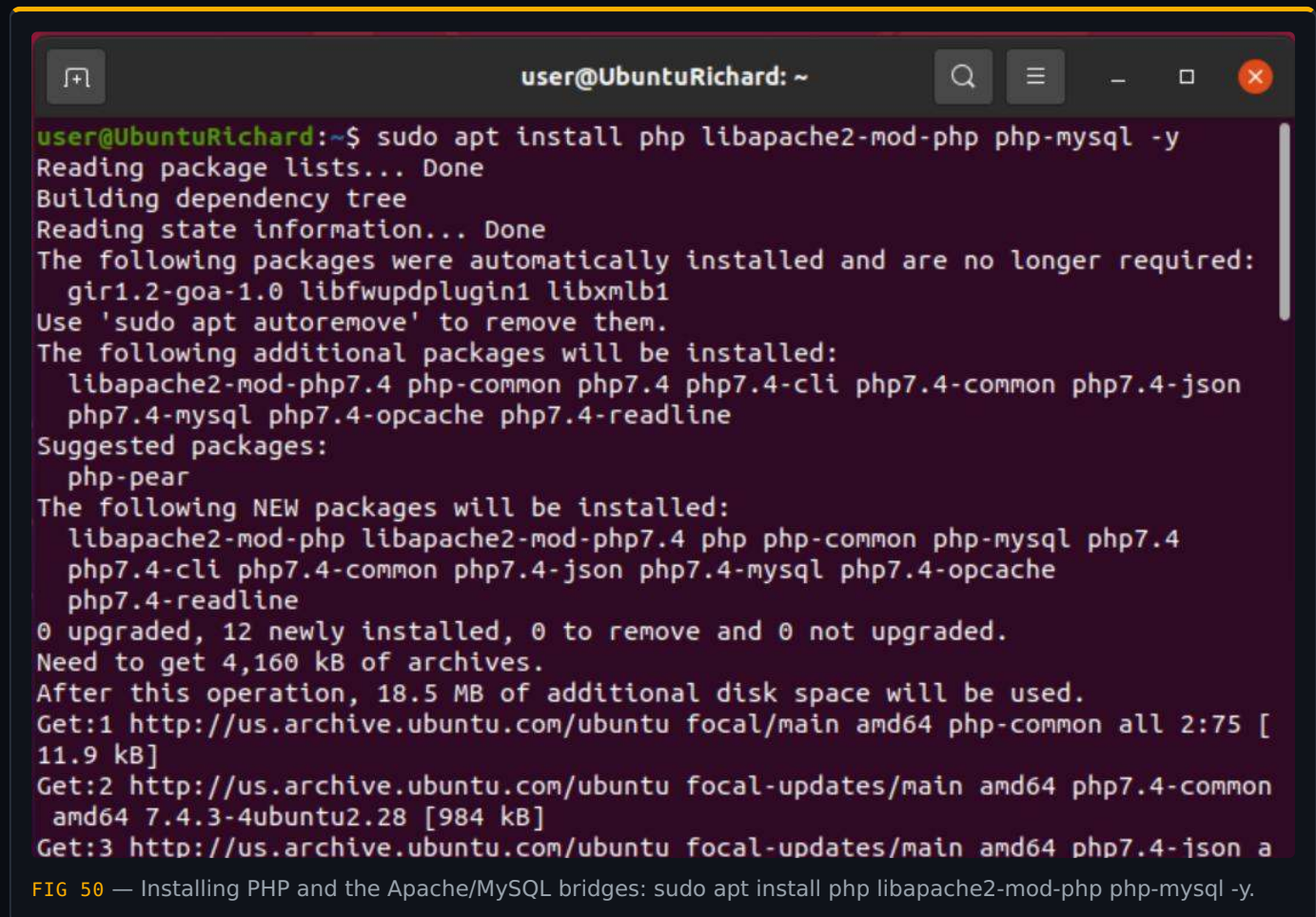
Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit;
Bye
user@UbuntuRichard:~$
```

FIG 49 — Leaving the MySQL shell with EXIT; after setting the root password.



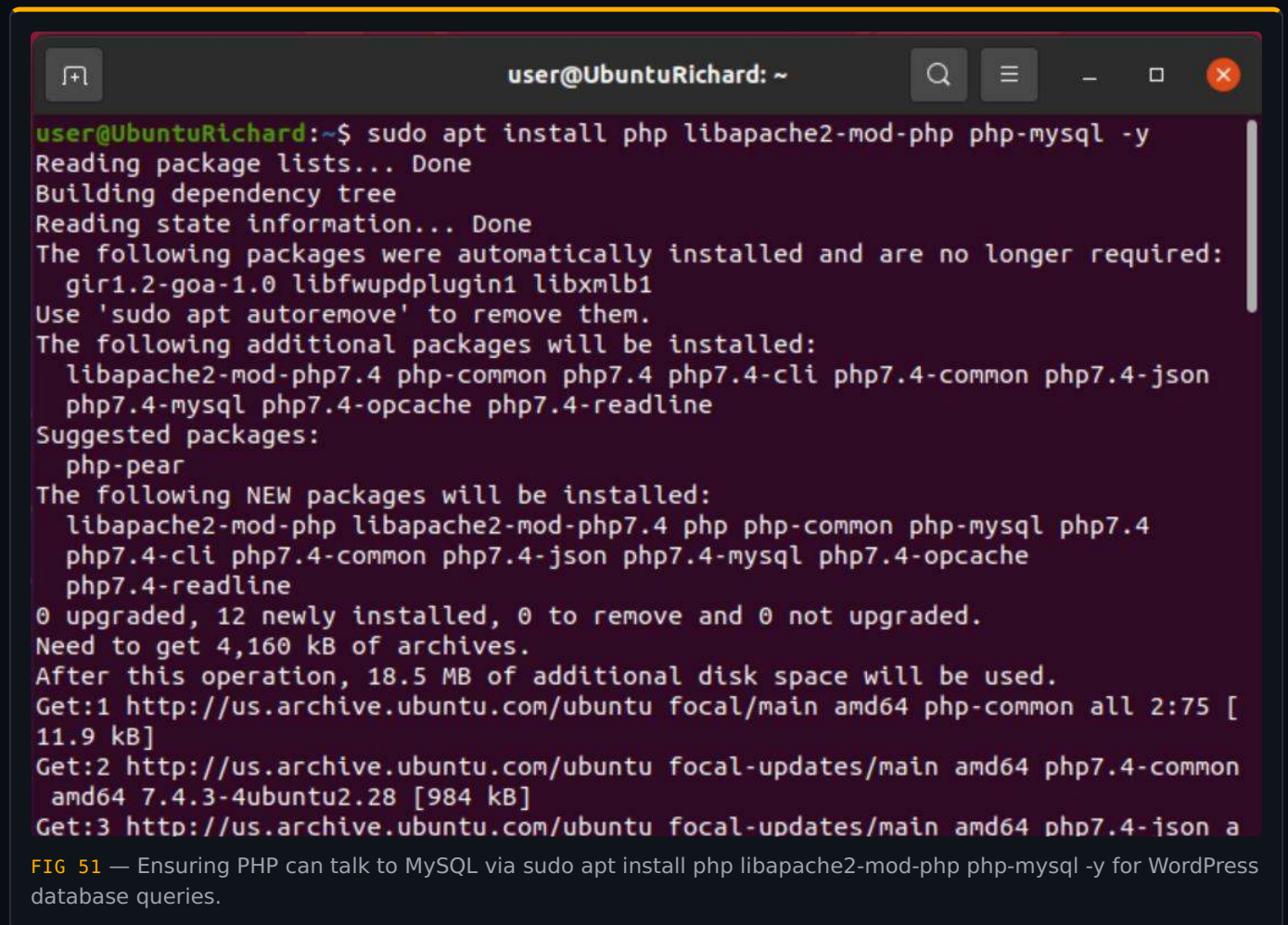
```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ sudo apt install php libapache2-mod-php php-mysql -y  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  libapache2-mod-php7.4 php-common php7.4 php7.4-cli php7.4-common php7.4-json  
  php7.4-mysql php7.4-opcache php7.4-readline  
Suggested packages:  
  php-pear  
The following NEW packages will be installed:  
  libapache2-mod-php libapache2-mod-php7.4 php php-common php-mysql php7.4  
  php7.4-cli php7.4-common php7.4-json php7.4-mysql php7.4-opcache  
  php7.4-readline  
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.  
Need to get 4,160 kB of archives.  
After this operation, 18.5 MB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu focal/main amd64 php-common all 2:75 [11.9 kB]  
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 php7.4-common amd64 7.4.3-4ubuntu2.28 [984 kB]  
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 php7.4-json a
```

FIG 50 — Installing PHP and the Apache/MySQL bridges: `sudo apt install php libapache2-mod-php php-mysql -y`.

## LAMP STACK

## MySQL PHP Libraries and mod\_rewrite

Confirmed the PHP/MySQL libraries and enabled Apache's rewrite module for clean WordPress URLs.

A terminal window titled 'user@UbuntuRichard: ~' showing the execution of the command 'sudo apt install php libapache2-mod-php php-mysql -y'. The output displays the package list, dependency tree, and the list of packages to be installed. The terminal text is as follows:

```
user@UbuntuRichard:~$ sudo apt install php libapache2-mod-php php-mysql -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libapache2-mod-php7.4 php-common php7.4 php7.4-cli php7.4-common php7.4-json
  php7.4-mysql php7.4-opcache php7.4-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php libapache2-mod-php7.4 php php-common php-mysql php7.4
  php7.4-cli php7.4-common php7.4-json php7.4-mysql php7.4-opcache
  php7.4-readline
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,160 kB of archives.
After this operation, 18.5 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal/main amd64 php-common all 2:75 [
11.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 php7.4-common
amd64 7.4.3-4ubuntu2.28 [984 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 php7.4-ison a
```

FIG 51 — Ensuring PHP can talk to MySQL via `sudo apt install php libapache2-mod-php php-mysql -y` for WordPress database queries.

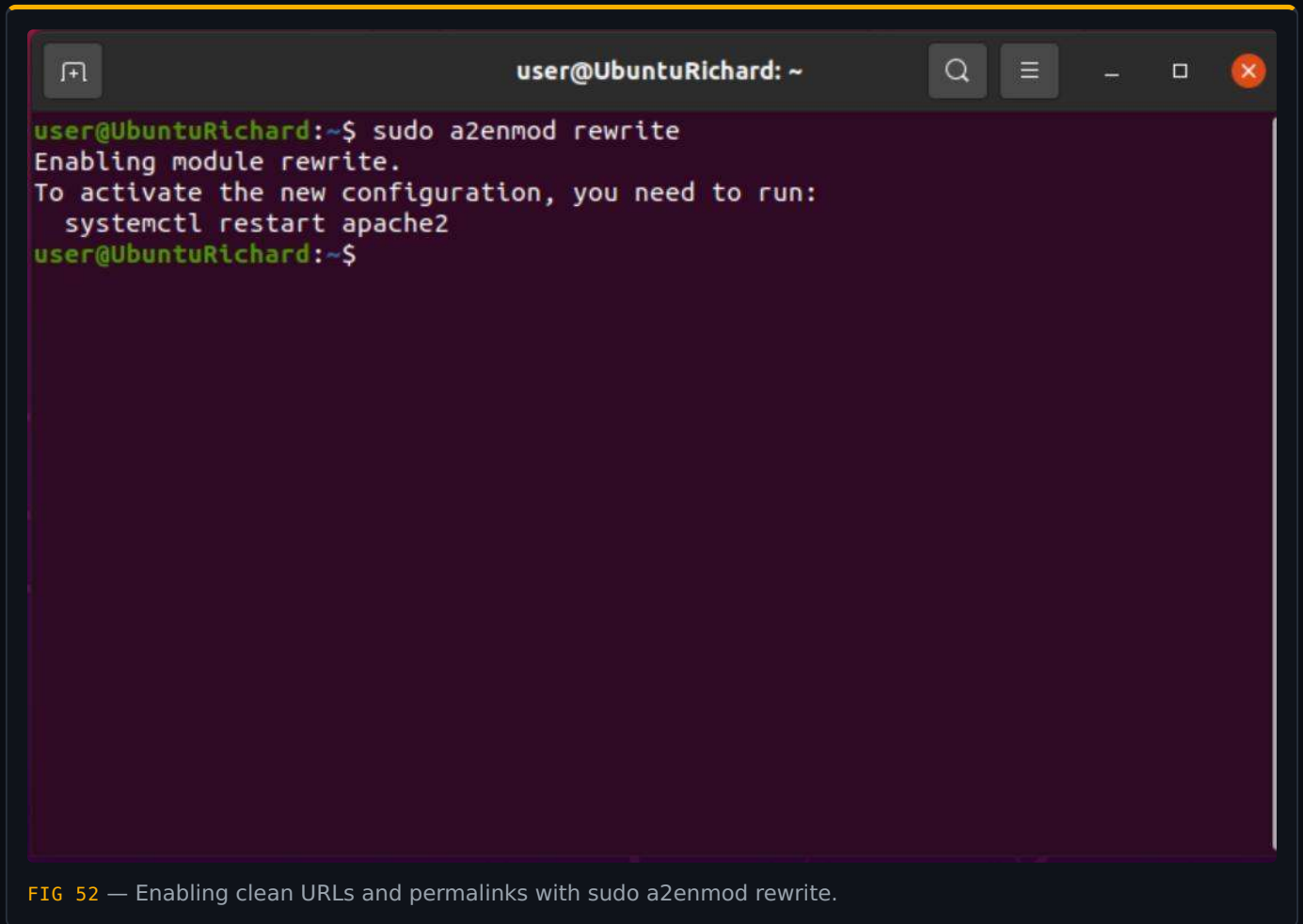
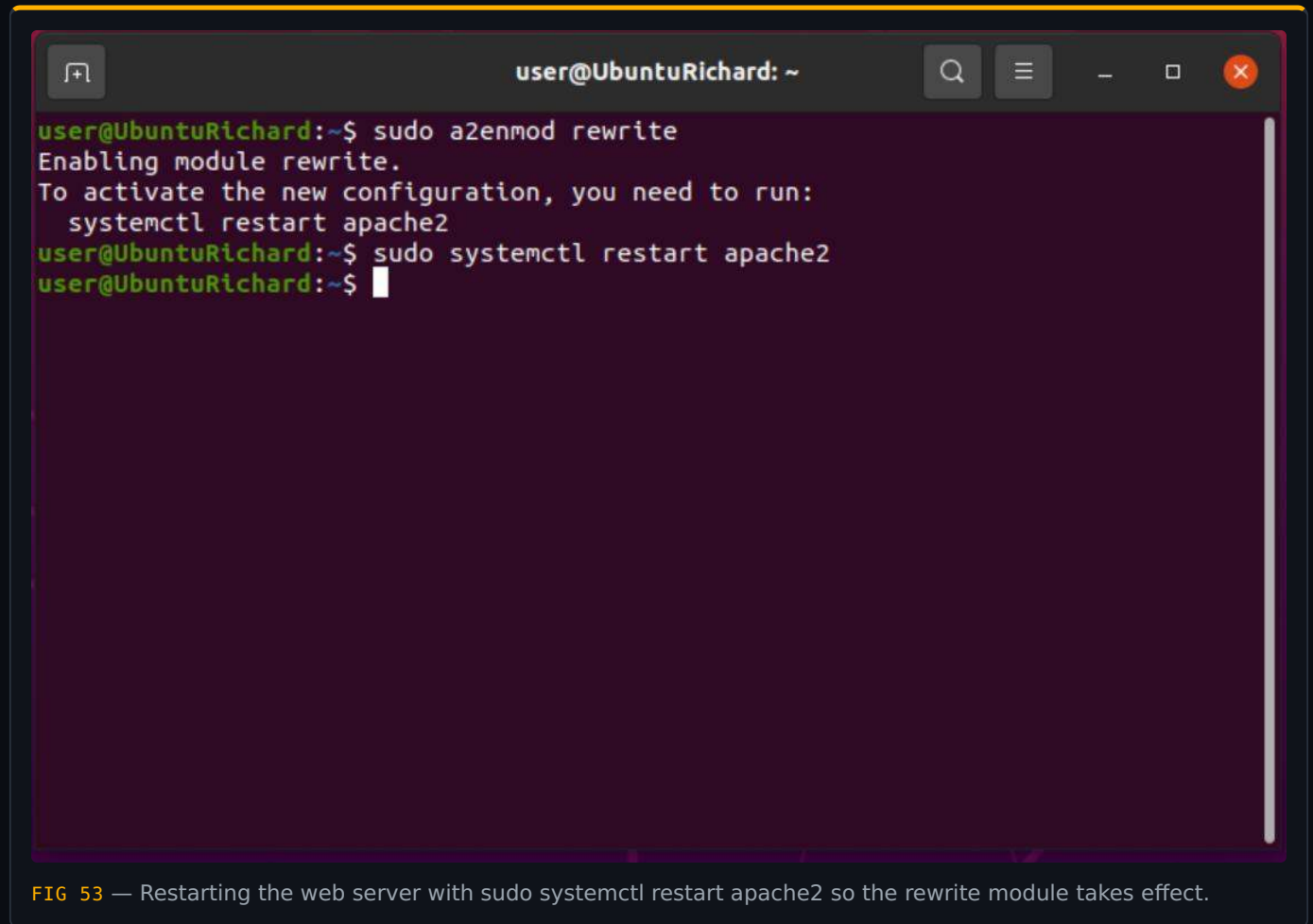


FIG 52 — Enabling clean URLs and permalinks with `sudo a2enmod rewrite`.

## LAMP STACK

## Restart Apache and Create test.php

Restarted Apache to load mod\_rewrite, then dropped a PHP probe page to confirm execution.

A terminal window titled 'user@UbuntuRichard: ~' with search, menu, and window control icons. The terminal shows the following commands and output:

```
user@UbuntuRichard:~$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
user@UbuntuRichard:~$ sudo systemctl restart apache2
user@UbuntuRichard:~$
```

FIG 53 — Restarting the web server with `sudo systemctl restart apache2` so the rewrite module takes effect.

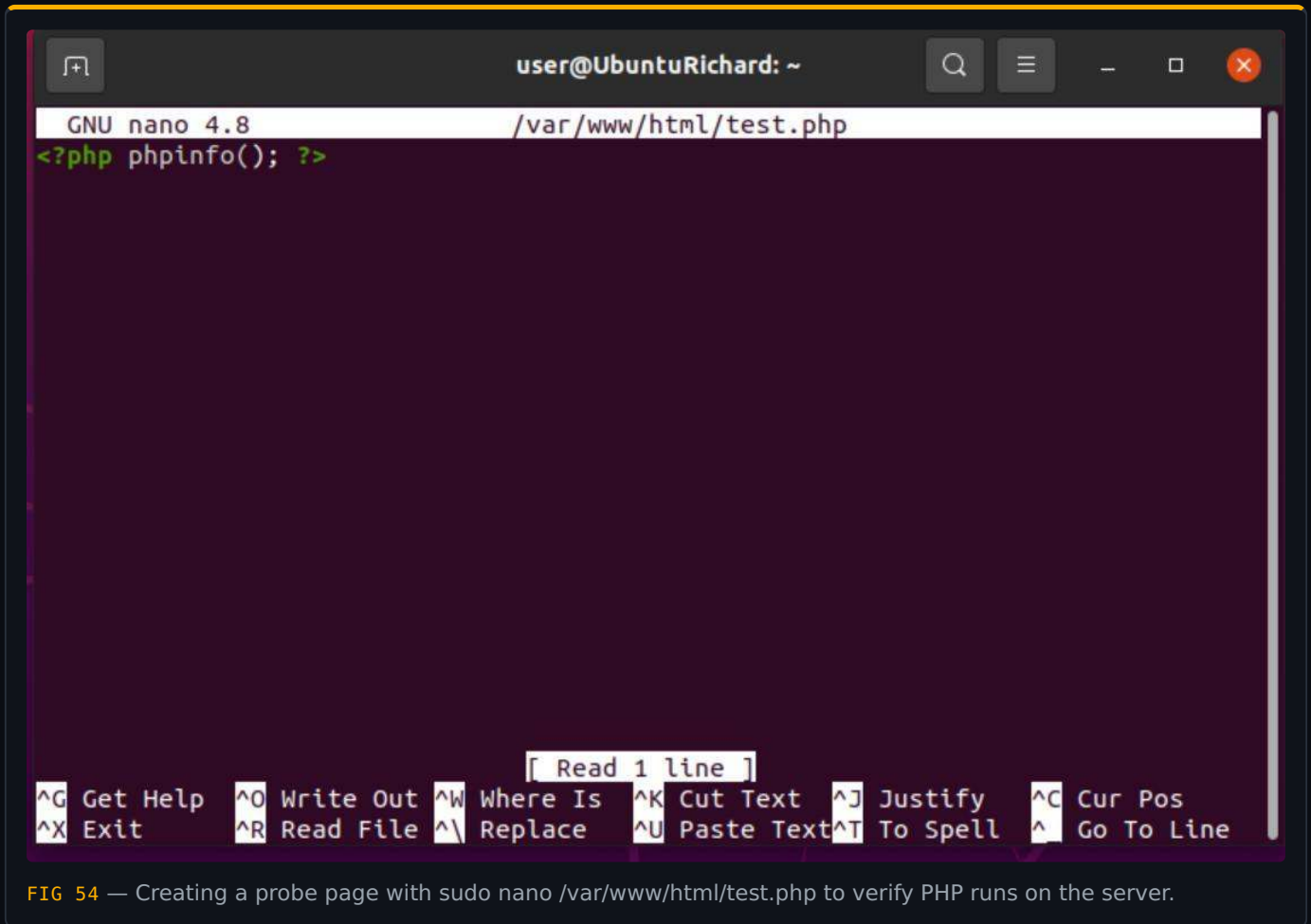


FIG 54 — Creating a probe page with sudo nano /var/www/html/test.php to verify PHP runs on the server.

LAMP STACK

# Verify PHP and Log Into MySQL

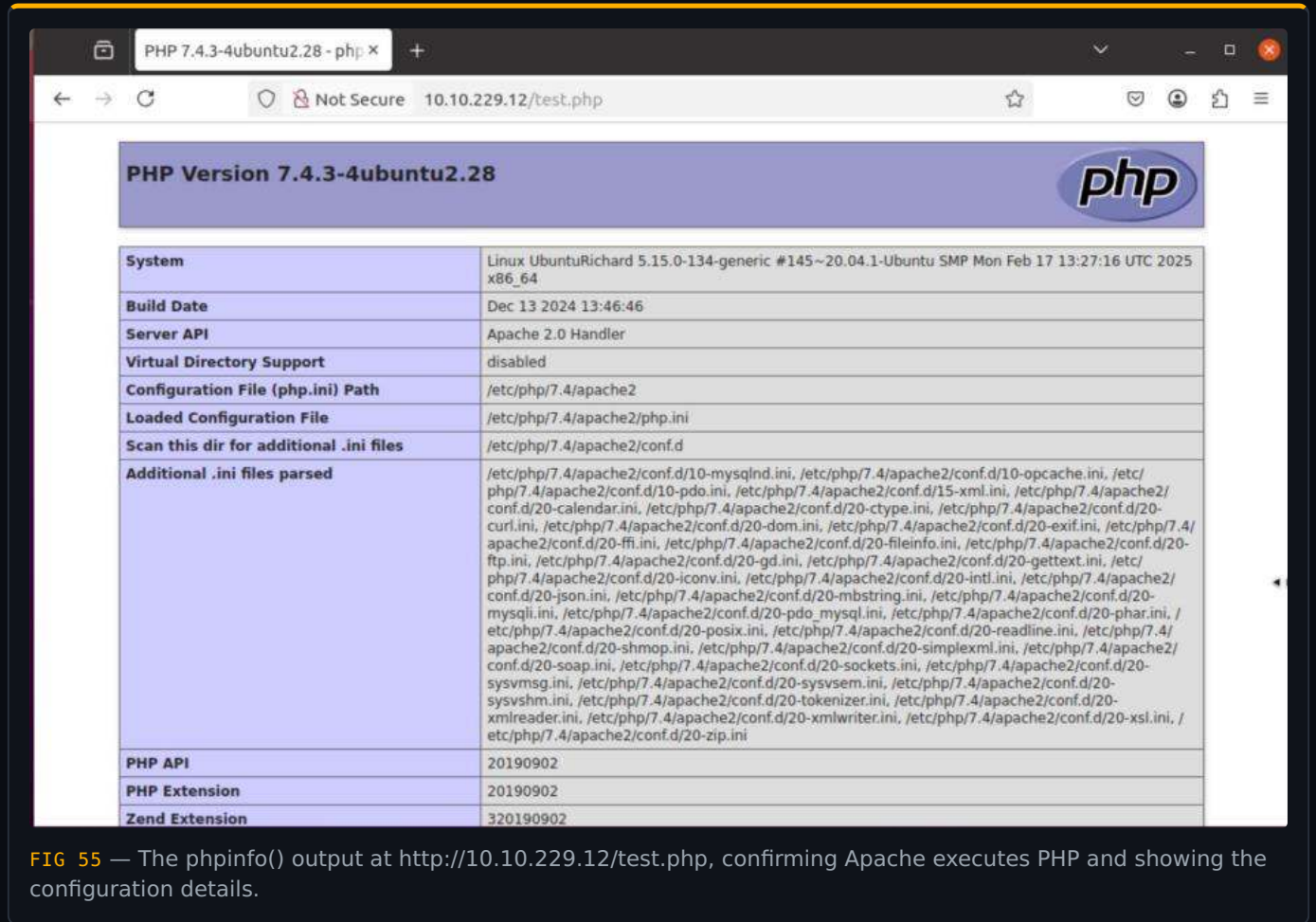


FIG 55 — The phpinfo() output at http://10.10.229.12/test.php, confirming Apache executes PHP and showing the configuration details.

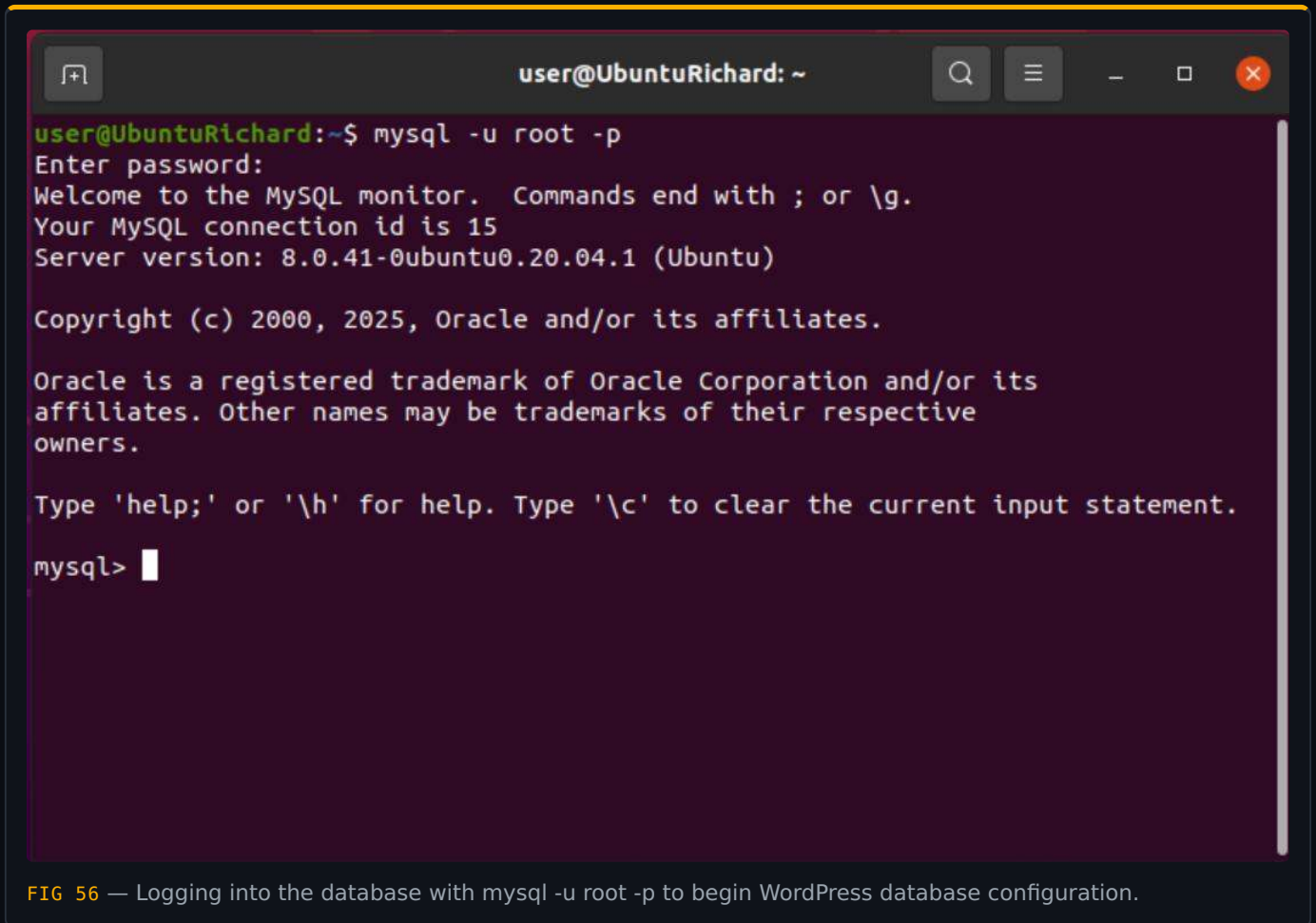
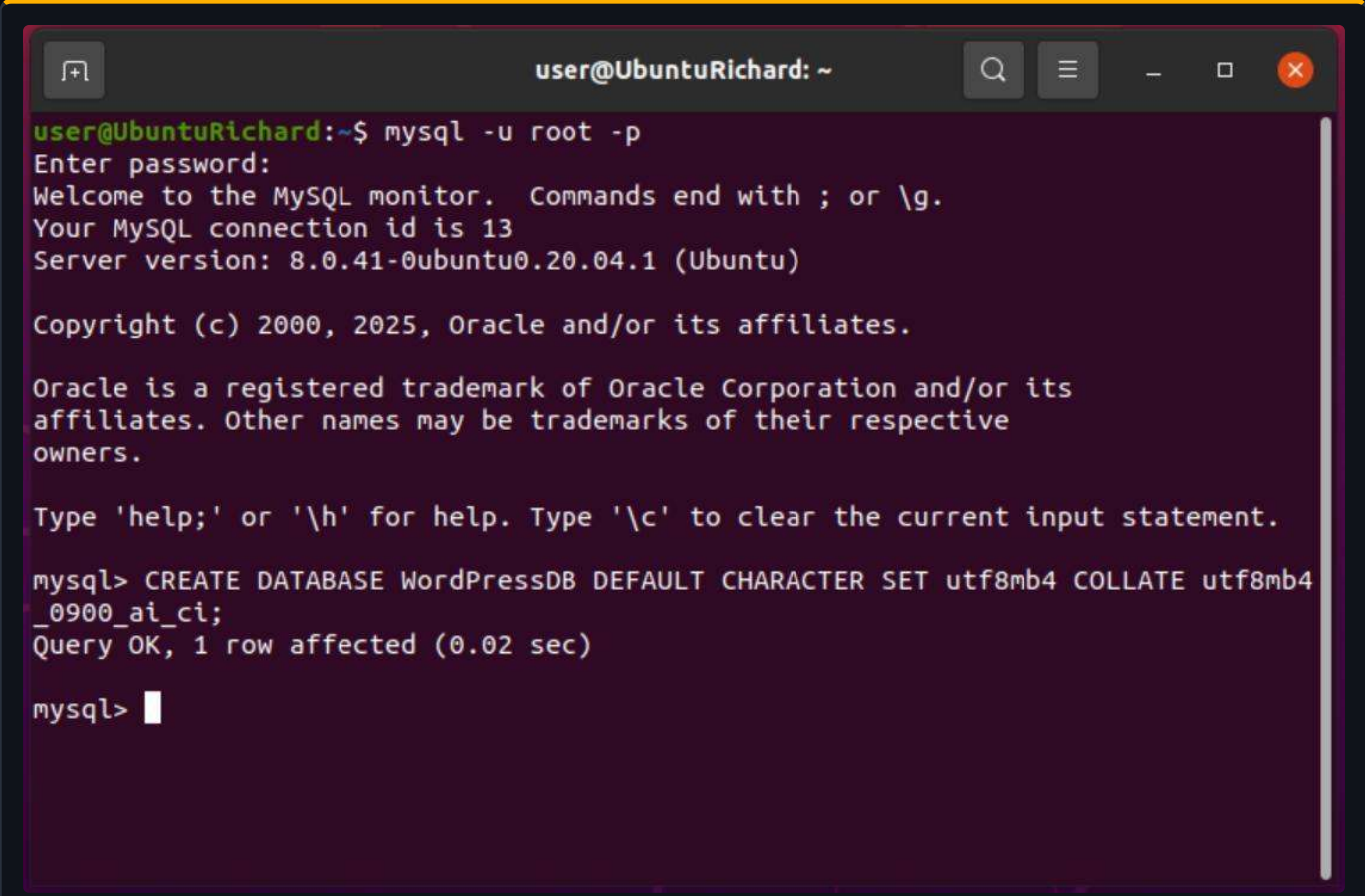


FIG 56 — Logging into the database with mysql -u root -p to begin WordPress database configuration.

## DATABASE

## Create the WordPress Database and User

Created the WordPressDB database and a dedicated WordPressUser account.



```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 13  
Server version: 8.0.41-0ubuntu0.20.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2025, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> CREATE DATABASE WordPressDB DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4  
_0900_ai_ci;  
Query OK, 1 row affected (0.02 sec)  
  
mysql> █
```

FIG 57 — CREATE DATABASE WordPressDB DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci; provisioning the WordPress database with UTF-8 encoding.

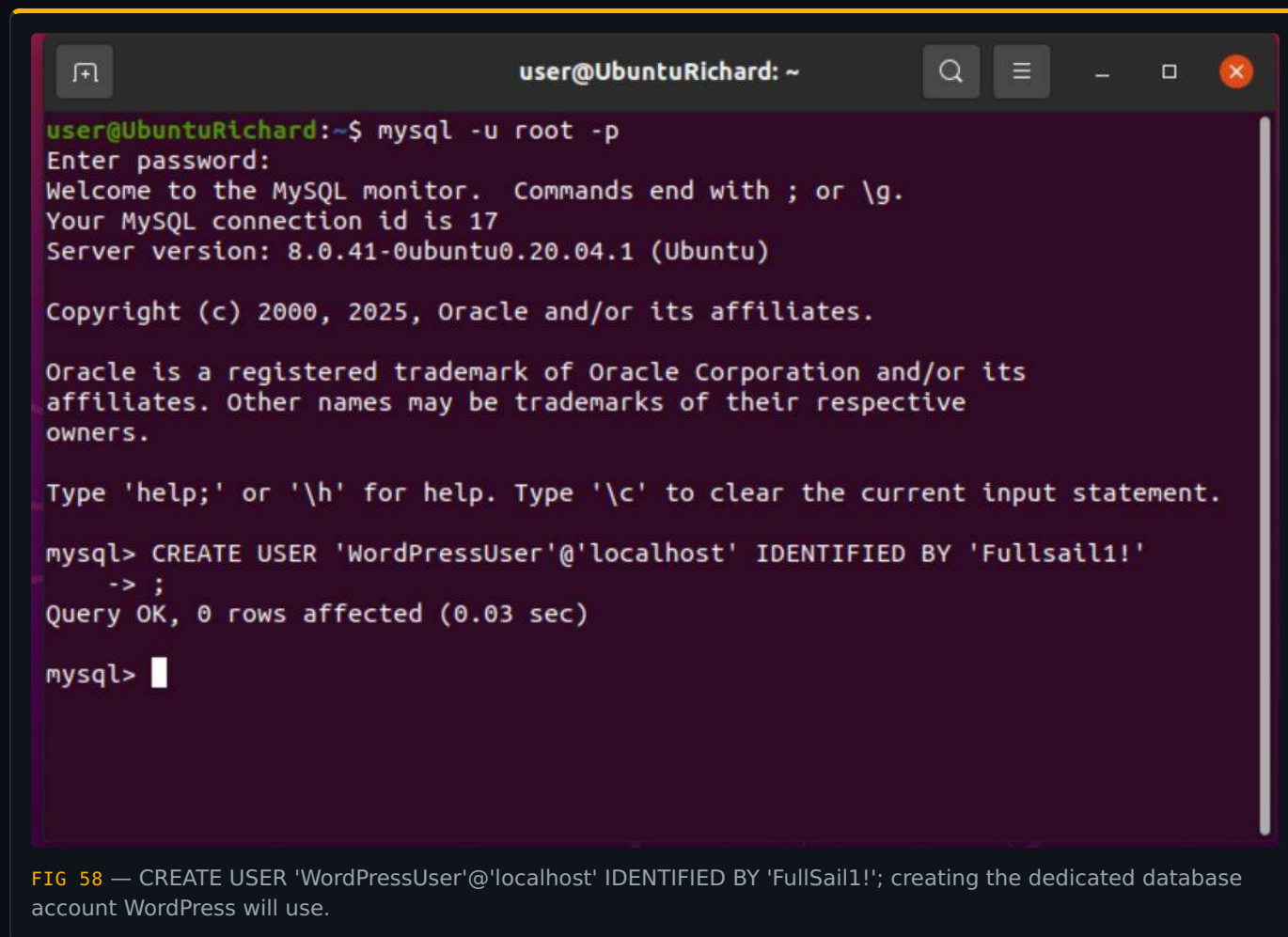
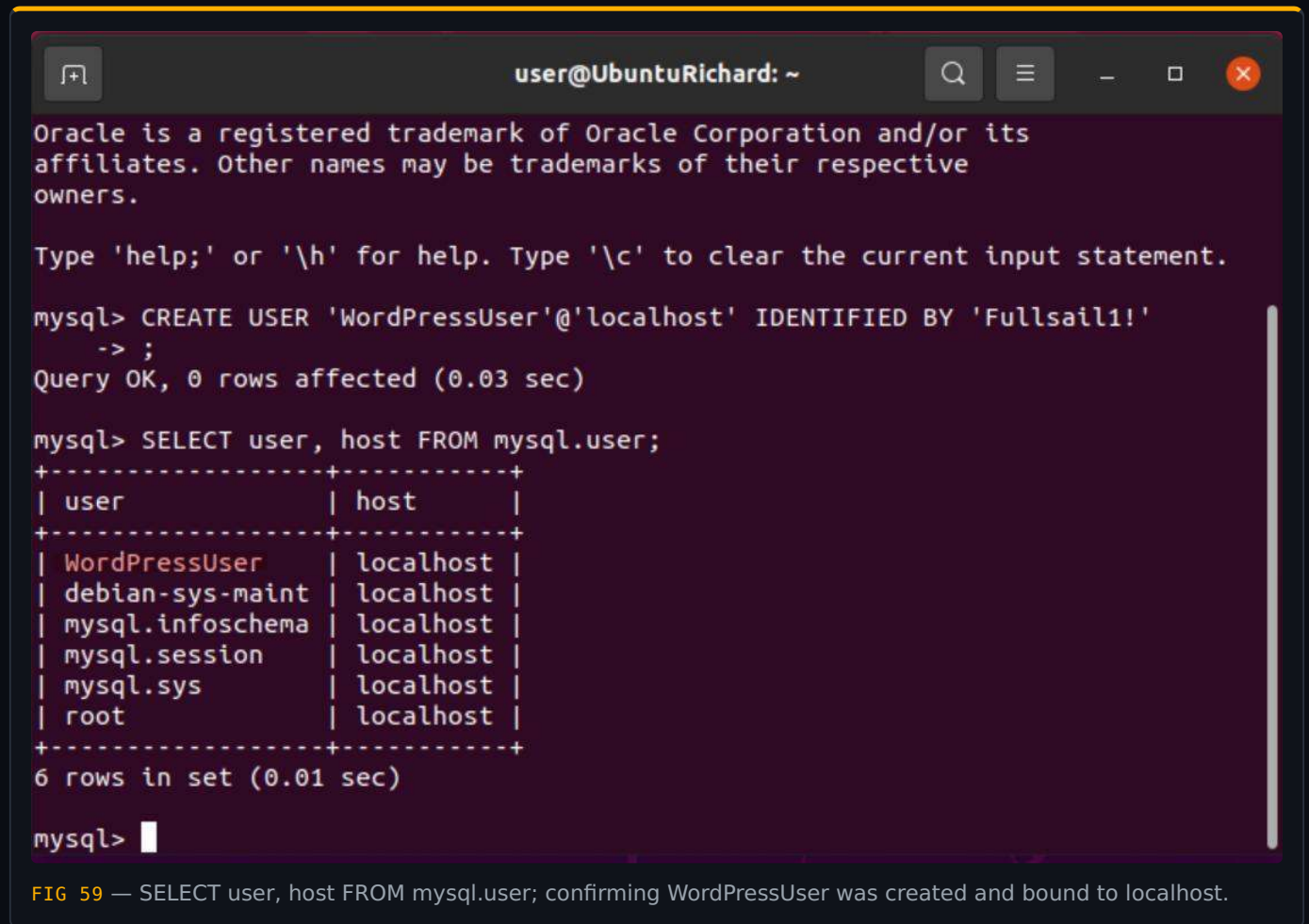


FIG 58 — CREATE USER 'WordPressUser'@'localhost' IDENTIFIED BY 'FullSail1!'; creating the dedicated database account WordPress will use.

## DATABASE

## Verify User and Grant Privileges

A terminal window titled 'user@UbuntuRichard: ~' showing a MySQL command-line session. The user creates a new user 'WordPressUser' with password 'Fullsail1!' and host 'localhost'. Then, they run a query to list all users, showing 'WordPressUser' is now listed with host 'localhost'.

```
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

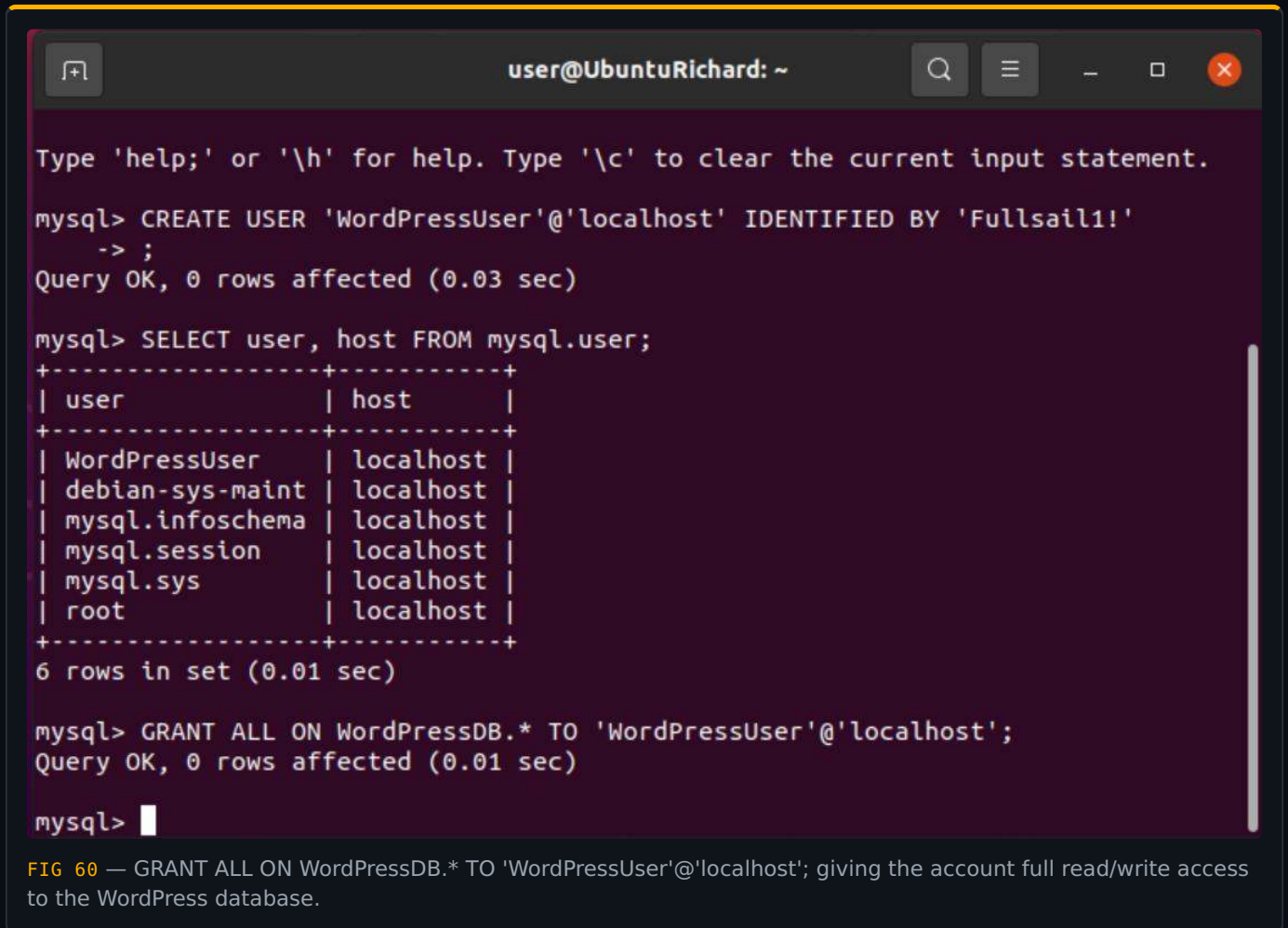
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE USER 'WordPressUser'@'localhost' IDENTIFIED BY 'Fullsail1!'
-> ;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT user, host FROM mysql.user;
+-----+-----+
| user          | host      |
+-----+-----+
| WordPressUser | localhost |
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session  | localhost |
| mysql.sys      | localhost |
| root           | localhost |
+-----+-----+
6 rows in set (0.01 sec)

mysql> █
```

FIG 59 — SELECT user, host FROM mysql.user; confirming WordPressUser was created and bound to localhost.

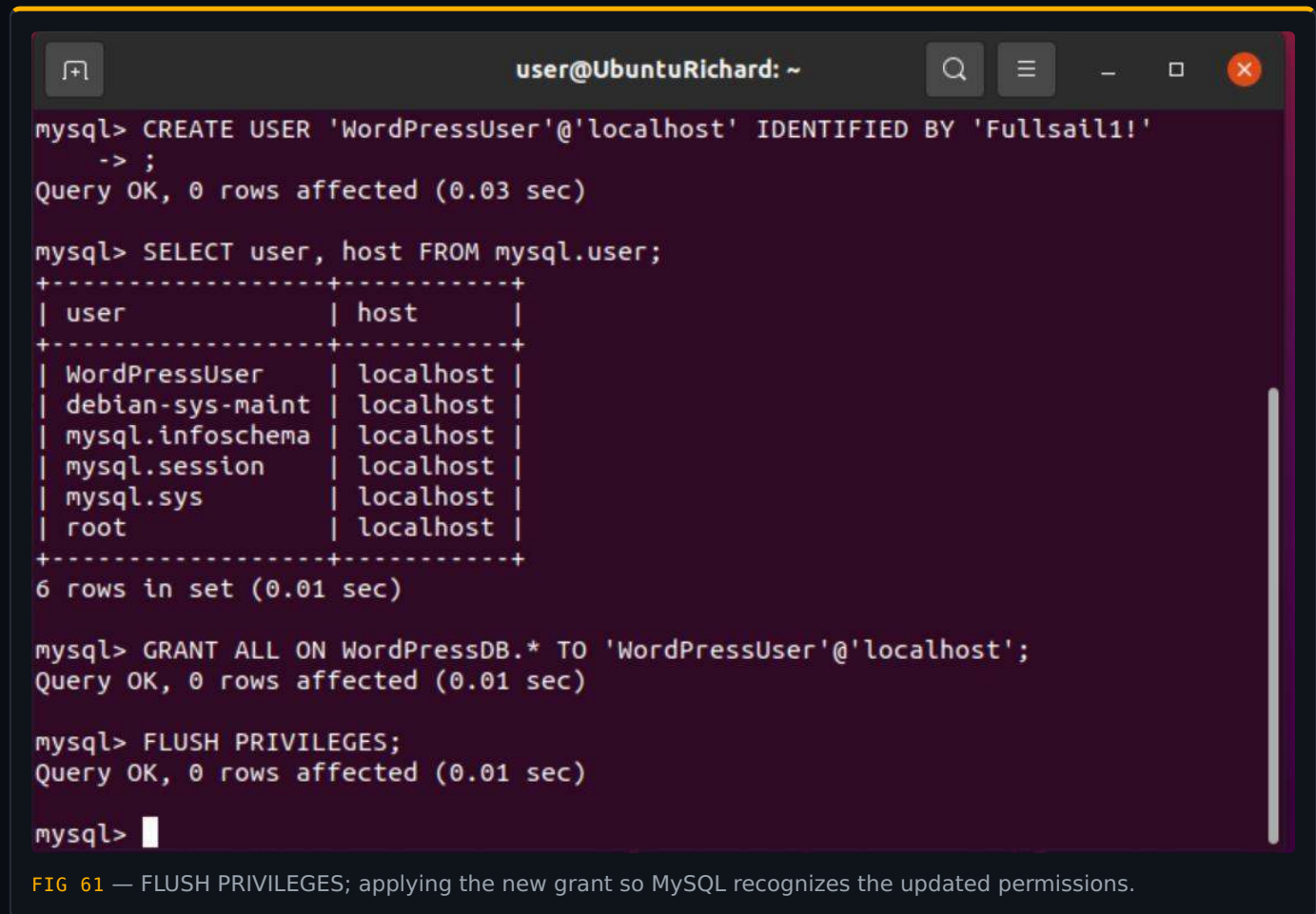


```
user@UbuntuRichard: ~  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> CREATE USER 'WordPressUser'@'localhost' IDENTIFIED BY 'Fullsail1!'  
-> ;  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> SELECT user, host FROM mysql.user;  
+-----+-----+  
| user          | host          |  
+-----+-----+  
| WordPressUser | localhost     |  
| debian-sys-maint | localhost     |  
| mysql.infoschema | localhost     |  
| mysql.session  | localhost     |  
| mysql.sys      | localhost     |  
| root           | localhost     |  
+-----+-----+  
6 rows in set (0.01 sec)  
  
mysql> GRANT ALL ON WordPressDB.* TO 'WordPressUser'@'localhost';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> █
```

**FIG 60** — GRANT ALL ON WordPressDB.\* TO 'WordPressUser'@'localhost'; giving the account full read/write access to the WordPress database.

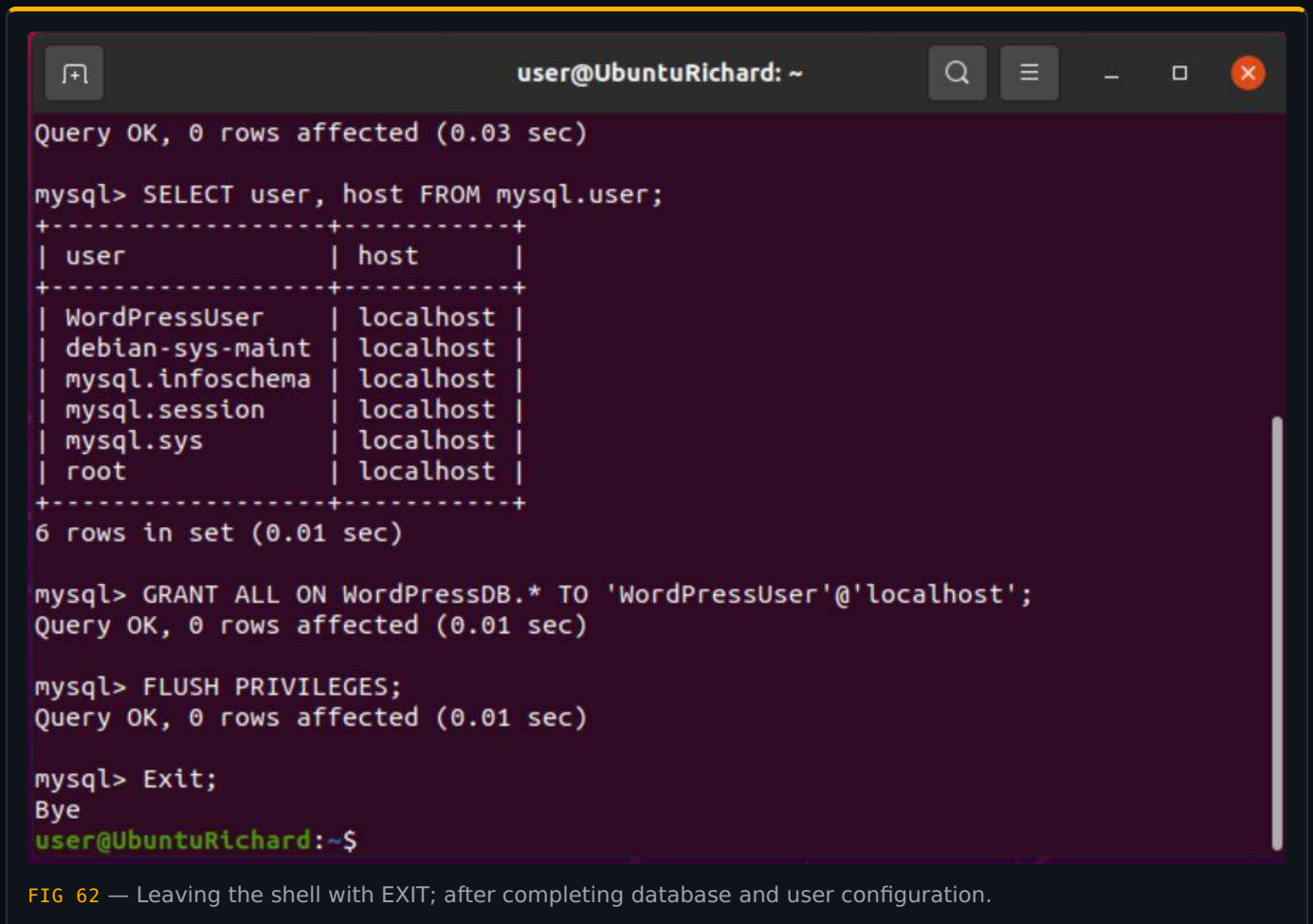
## DATABASE

## Flush Privileges and Exit



```
user@UbuntuRichard: ~  
mysql> CREATE USER 'WordPressUser'@'localhost' IDENTIFIED BY 'Fullsail1!'  
-> ;  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> SELECT user, host FROM mysql.user;  
+-----+-----+  
| user          | host      |  
+-----+-----+  
| WordPressUser | localhost |  
| debian-sys-maint | localhost |  
| mysql.infoschema | localhost |  
| mysql.session | localhost |  
| mysql.sys      | localhost |  
| root           | localhost |  
+-----+-----+  
6 rows in set (0.01 sec)  
  
mysql> GRANT ALL ON WordPressDB.* TO 'WordPressUser'@'localhost';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> █
```

FIG 61 — FLUSH PRIVILEGES; applying the new grant so MySQL recognizes the updated permissions.



```
user@UbuntuRichard: ~  
Query OK, 0 rows affected (0.03 sec)  
mysql> SELECT user, host FROM mysql.user;  
+-----+-----+  
| user          | host      |  
+-----+-----+  
| WordPressUser | localhost |  
| debian-sys-maint | localhost |  
| mysql.infoschema | localhost |  
| mysql.session  | localhost |  
| mysql.sys      | localhost |  
| root           | localhost |  
+-----+-----+  
6 rows in set (0.01 sec)  
  
mysql> GRANT ALL ON WordPressDB.* TO 'WordPressUser'@'localhost';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> Exit;  
Bye  
user@UbuntuRichard:~$
```

FIG 62 — Leaving the shell with EXIT; after completing database and user configuration.

## WORDPRESS

## Prepare the Web Root

Took ownership of `/var/www/html` and cleared it for a fresh WordPress install.

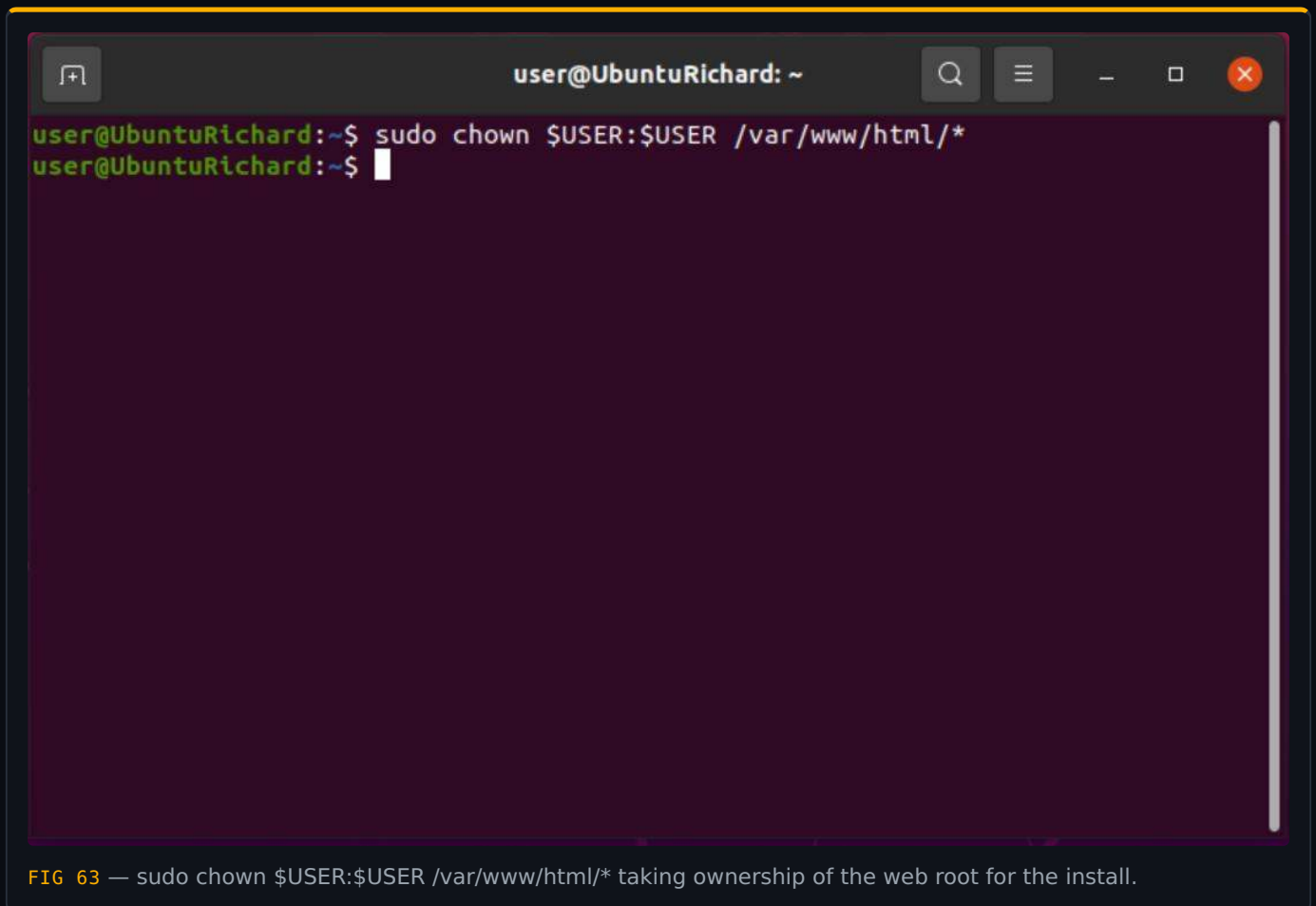


FIG 63 — `sudo chown $USER:$USER /var/www/html/*` taking ownership of the web root for the install.

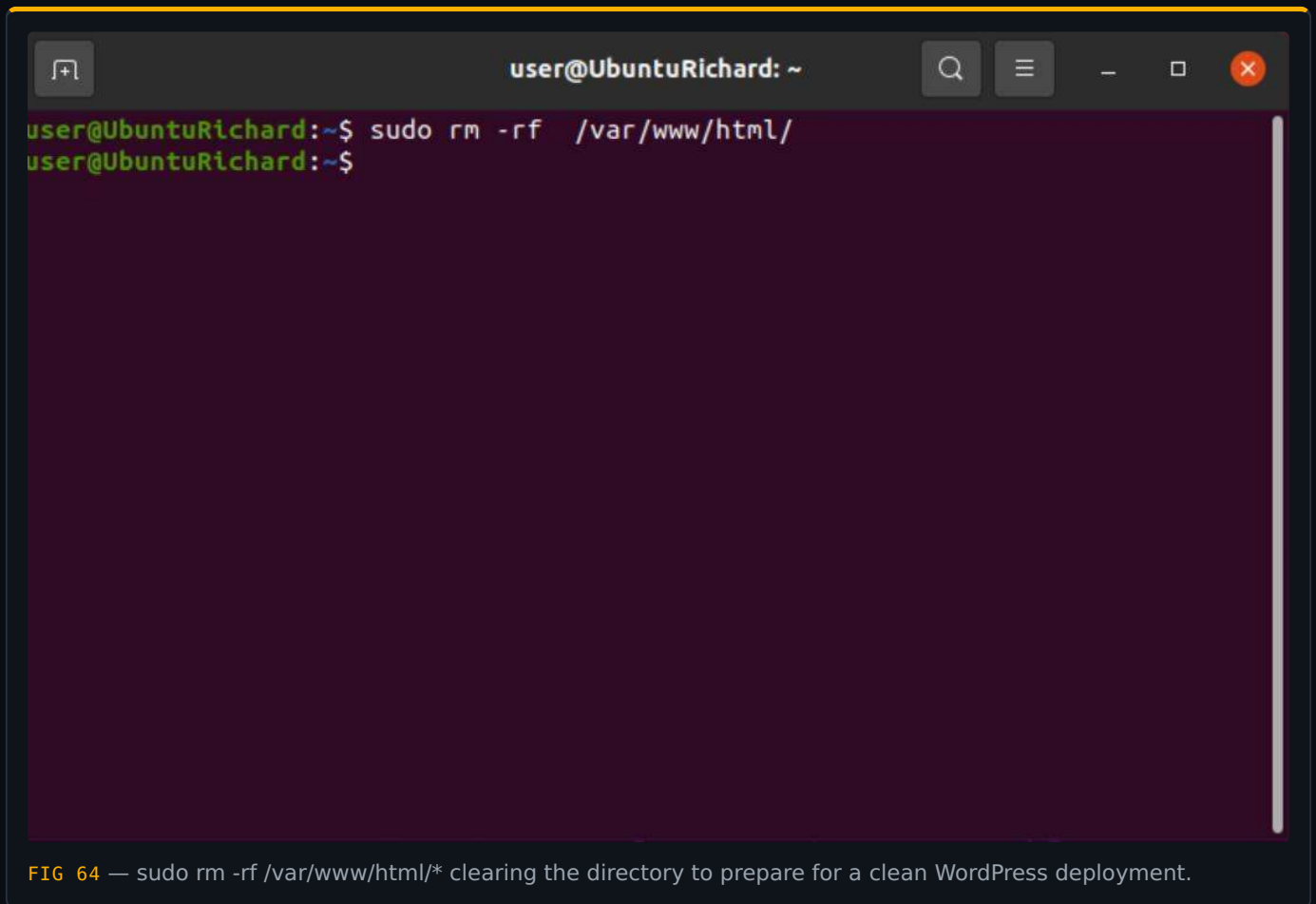


FIG 64 — `sudo rm -rf /var/www/html/*` clearing the directory to prepare for a clean WordPress deployment.

## WORDPRESS

### Confirm Empty Root and Clone WordPress

Verified the web root was empty, then cloned WordPress from GitHub into it.

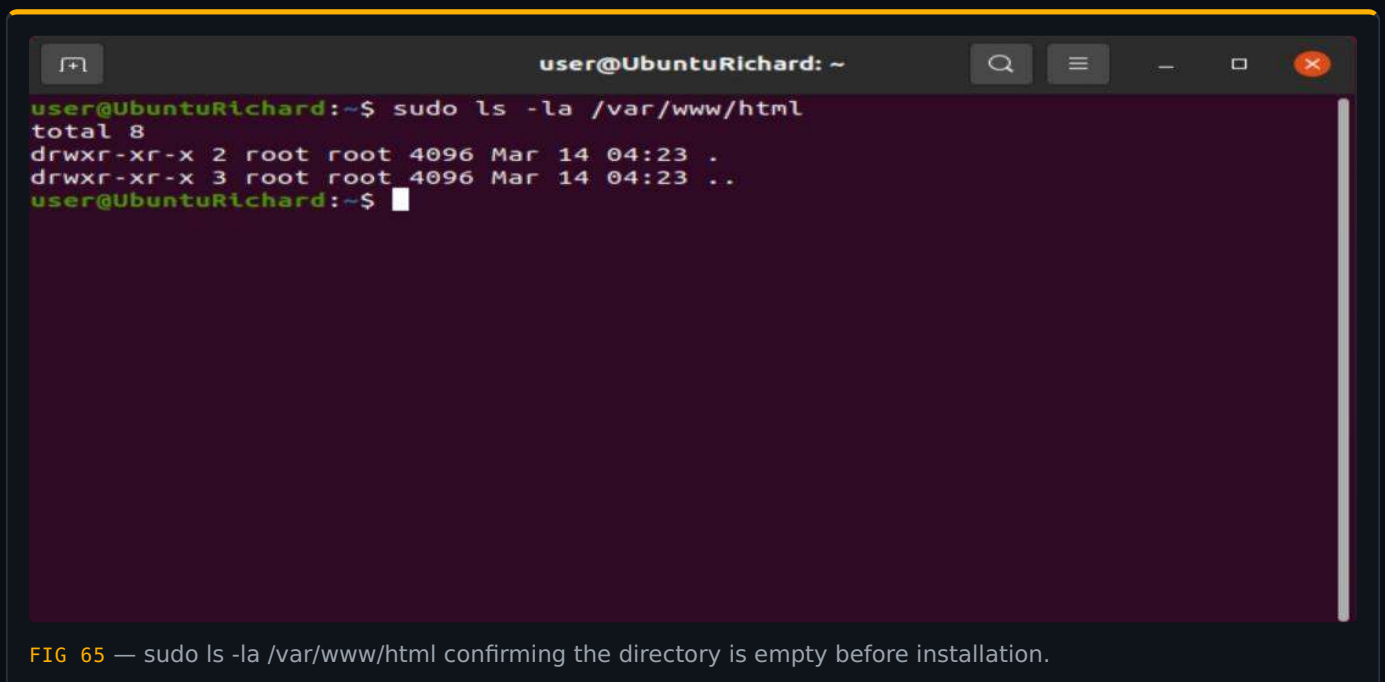


FIG 65 — `sudo ls -la /var/www/html` confirming the directory is empty before installation.

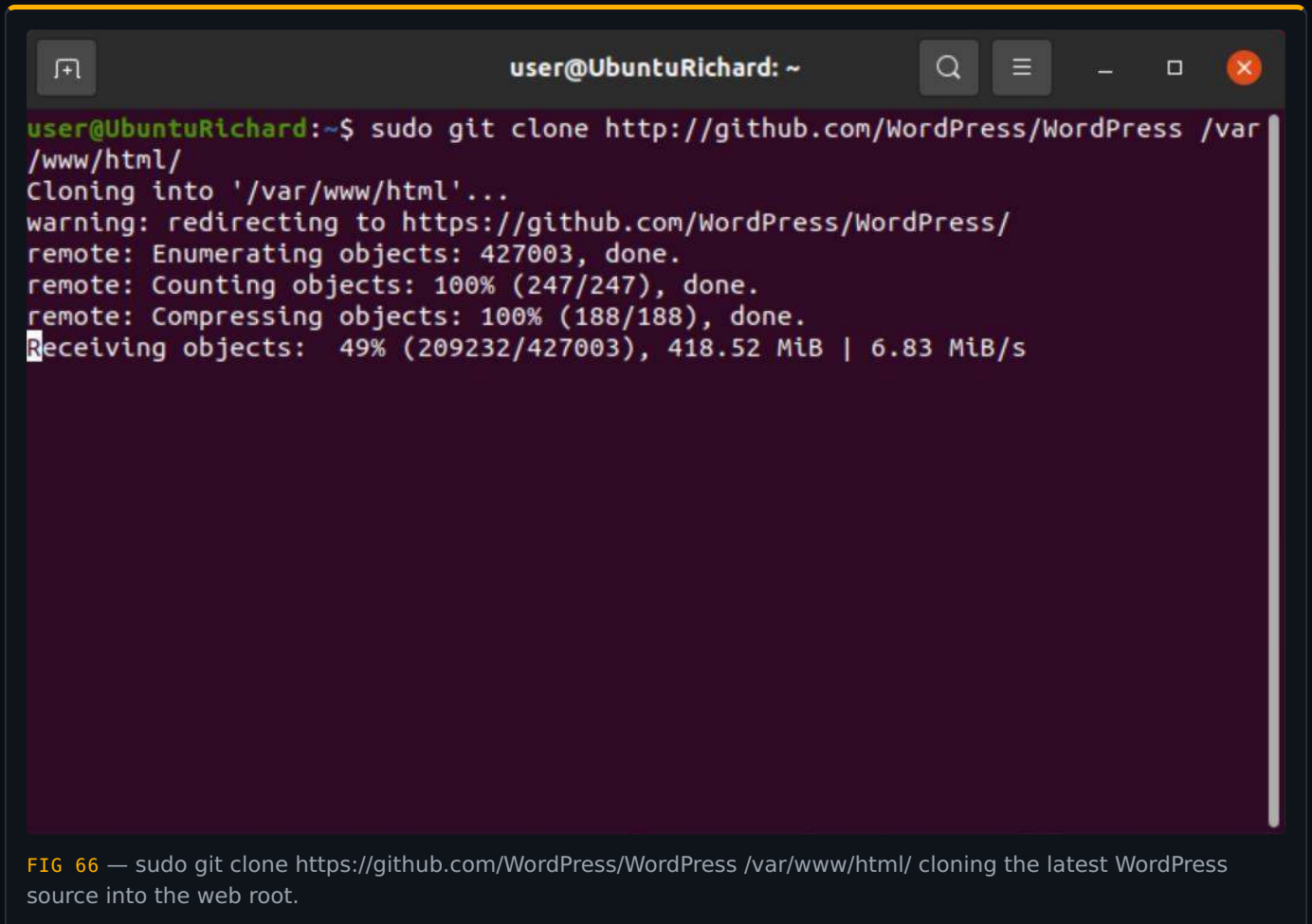
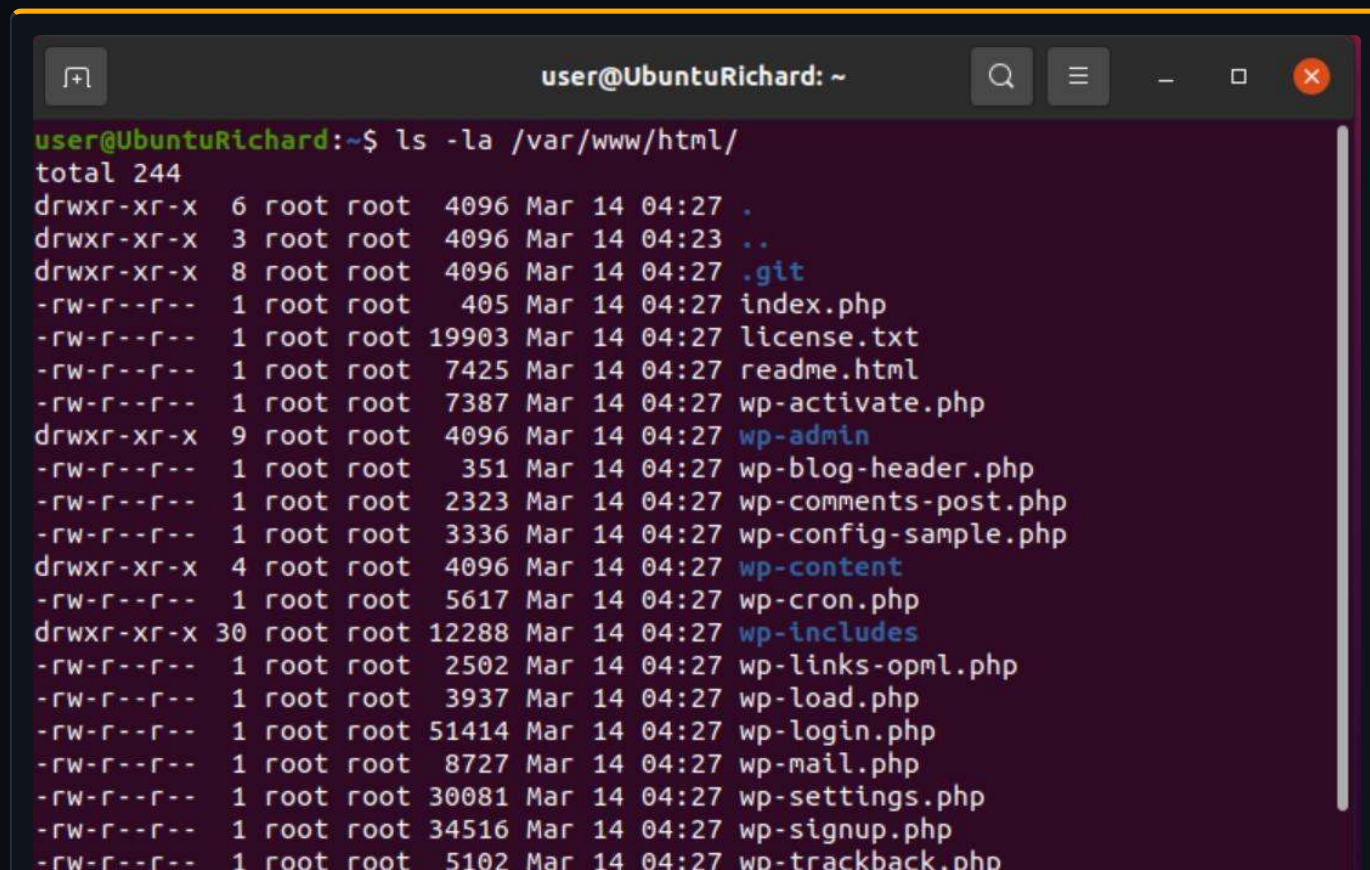


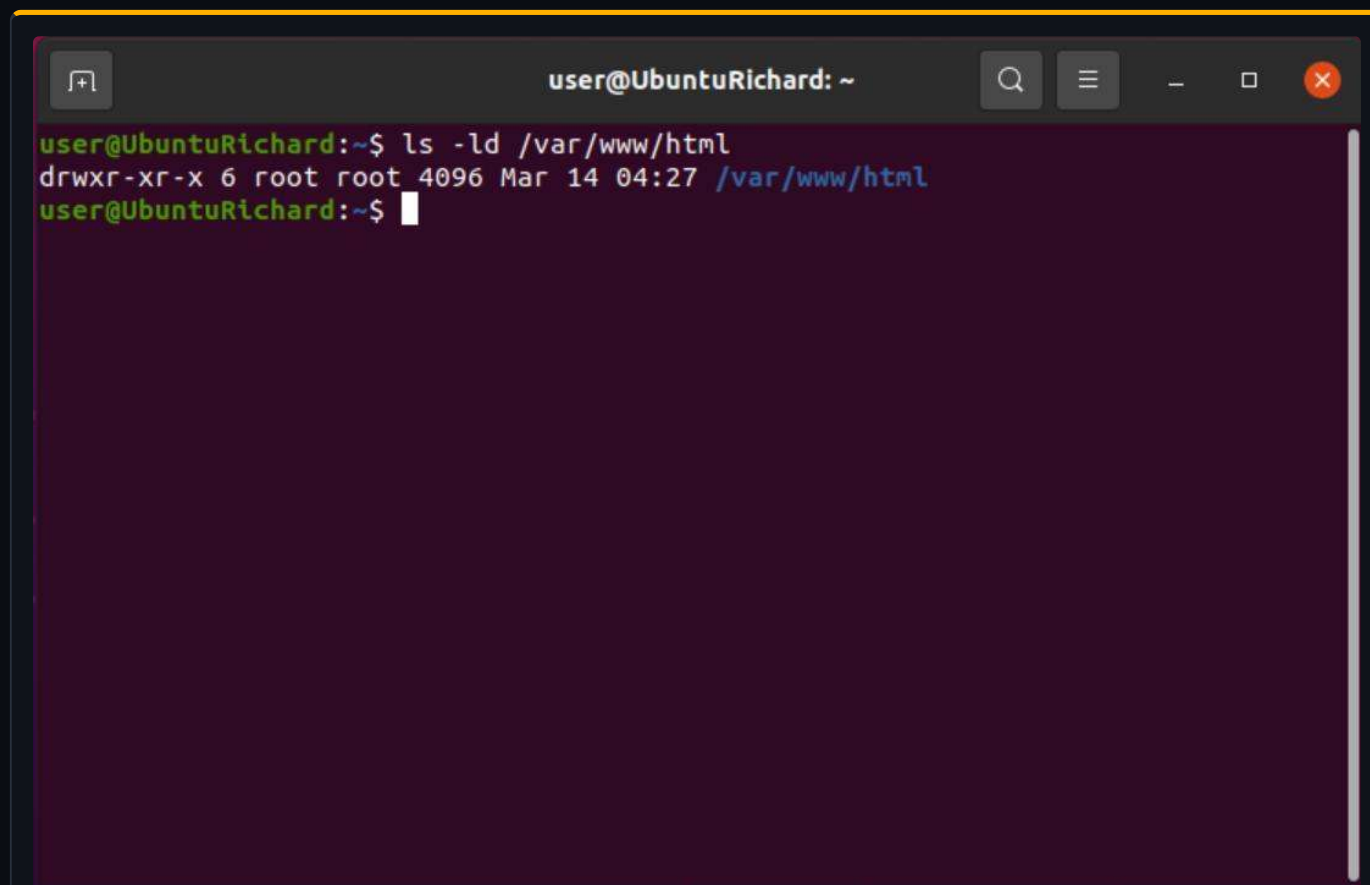
FIG 66 — `sudo git clone https://github.com/WordPress/WordPress /var/www/html/` cloning the latest WordPress source into the web root.

## WORDPRESS

## Verify Clone and Directory Permissions



```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ ls -la /var/www/html/  
total 244  
drwxr-xr-x 6 root root 4096 Mar 14 04:27 .  
drwxr-xr-x 3 root root 4096 Mar 14 04:23 ..  
drwxr-xr-x 8 root root 4096 Mar 14 04:27 .git  
-rw-r--r-- 1 root root 405 Mar 14 04:27 index.php  
-rw-r--r-- 1 root root 19903 Mar 14 04:27 license.txt  
-rw-r--r-- 1 root root 7425 Mar 14 04:27 readme.html  
-rw-r--r-- 1 root root 7387 Mar 14 04:27 wp-activate.php  
drwxr-xr-x 9 root root 4096 Mar 14 04:27 wp-admin  
-rw-r--r-- 1 root root 351 Mar 14 04:27 wp-blog-header.php  
-rw-r--r-- 1 root root 2323 Mar 14 04:27 wp-comments-post.php  
-rw-r--r-- 1 root root 3336 Mar 14 04:27 wp-config-sample.php  
drwxr-xr-x 4 root root 4096 Mar 14 04:27 wp-content  
-rw-r--r-- 1 root root 5617 Mar 14 04:27 wp-cron.php  
drwxr-xr-x 30 root root 12288 Mar 14 04:27 wp-includes  
-rw-r--r-- 1 root root 2502 Mar 14 04:27 wp-links-opml.php  
-rw-r--r-- 1 root root 3937 Mar 14 04:27 wp-load.php  
-rw-r--r-- 1 root root 51414 Mar 14 04:27 wp-login.php  
-rw-r--r-- 1 root root 8727 Mar 14 04:27 wp-mail.php  
-rw-r--r-- 1 root root 30081 Mar 14 04:27 wp-settings.php  
-rw-r--r-- 1 root root 34516 Mar 14 04:27 wp-signup.php  
-rw-r--r-- 1 root root 5102 Mar 14 04:27 wp-trackback.php
```

FIG 67 — `ls -la /var/www/html/` confirming the WordPress files were cloned successfully.

```
user@UbuntuRichard: ~  
user@UbuntuRichard:~$ ls -ld /var/www/html  
drwxr-xr-x 6 root root 4096 Mar 14 04:27 /var/www/html  
user@UbuntuRichard:~$
```

FIG 68 — `ls -ld /var/www/html/` checking the directory permissions for web-server accessibility.

## WORDPRESS

### Set Ownership and Edit apache2.conf

Handed the files to the www-data web-server user and opened the Apache config to allow .htaccess overrides.

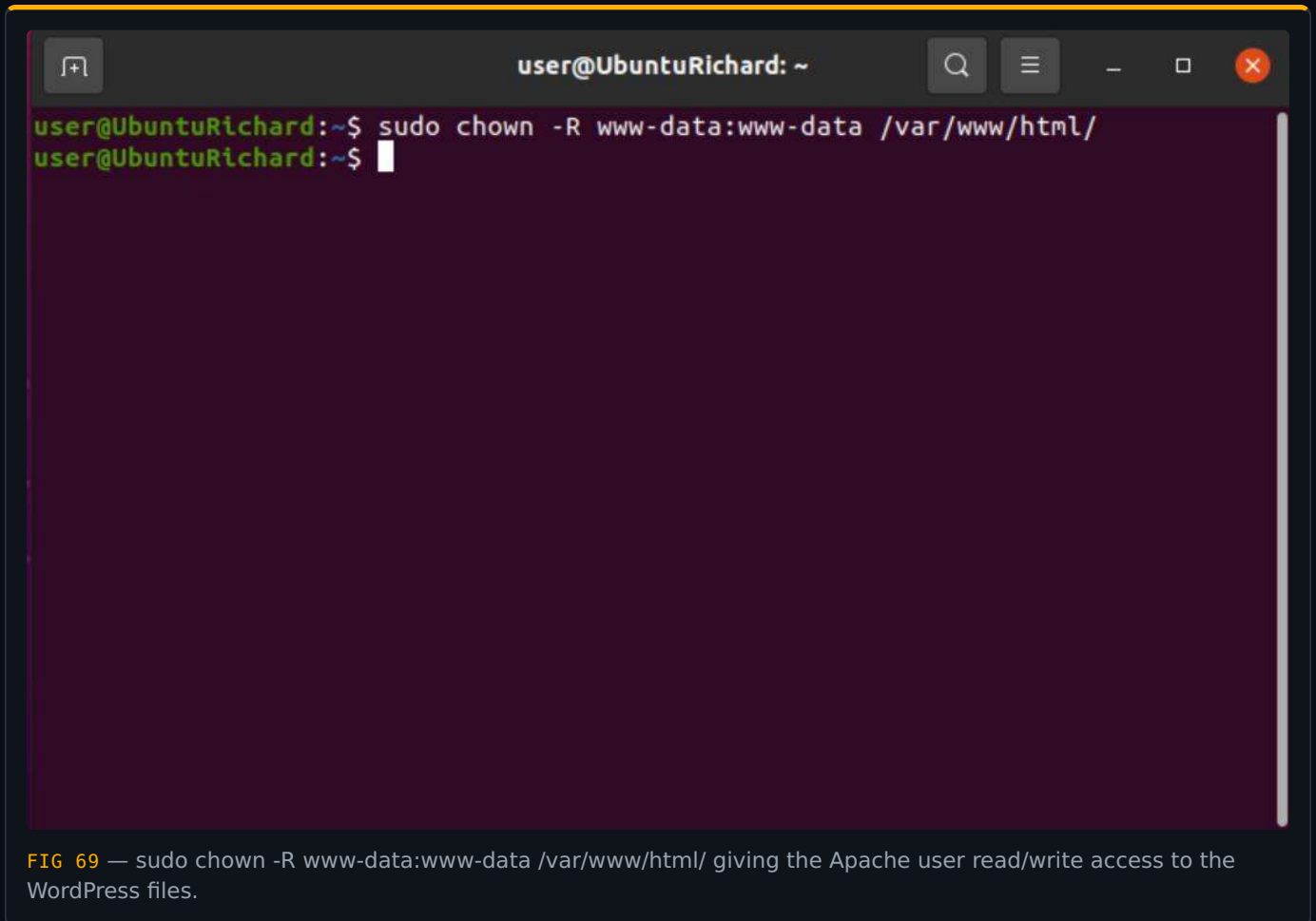


FIG 69 — sudo chown -R www-data:www-data /var/www/html/ giving the Apache user read/write access to the WordPress files.

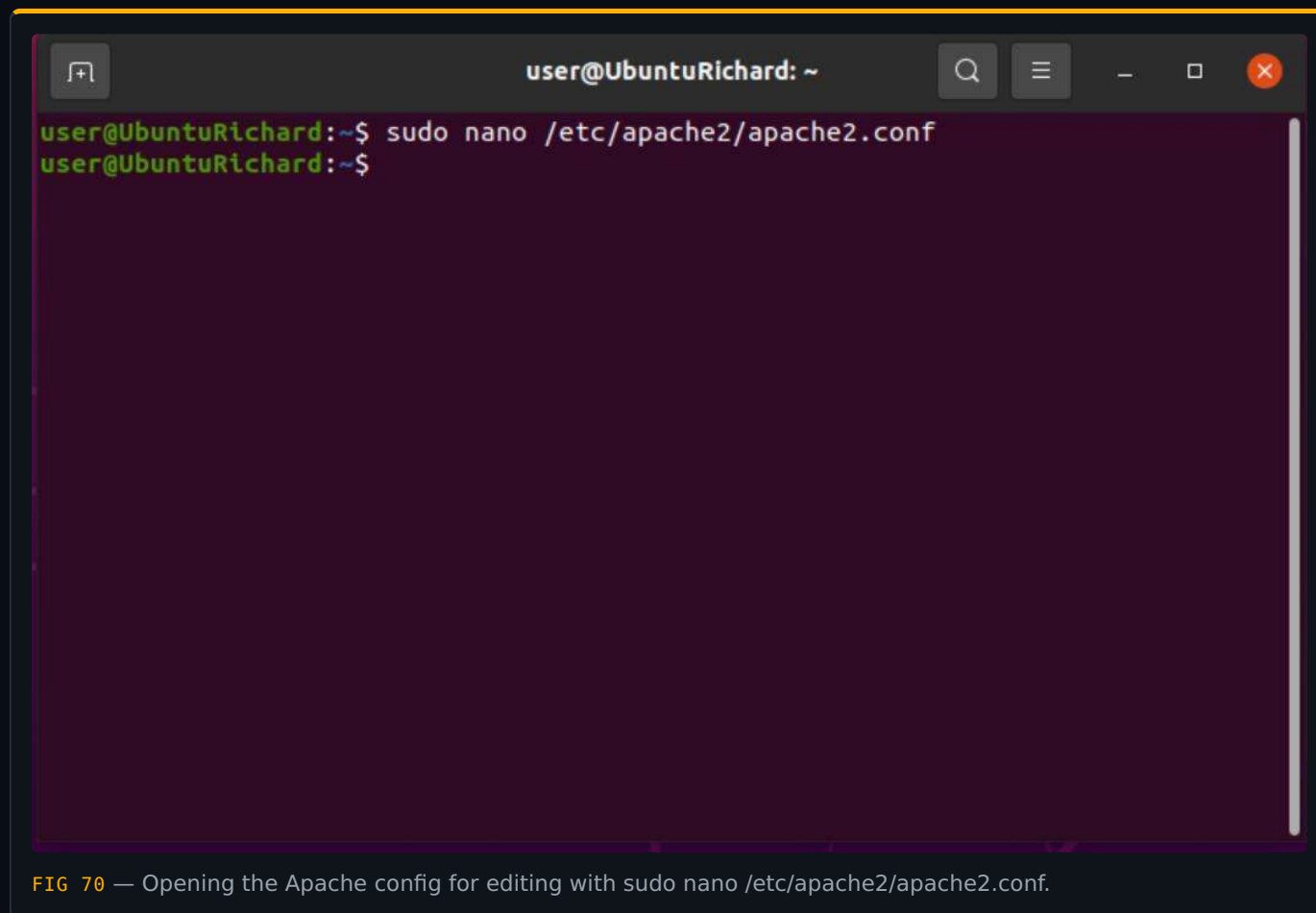


FIG 70 — Opening the Apache config for editing with sudo nano /etc/apache2/apache2.conf.

## WORDPRESS

### AllowOverride All and Protect .git

Set AllowOverride All for permalinks and added a protective .htaccess to block access to the cloned .git directory.

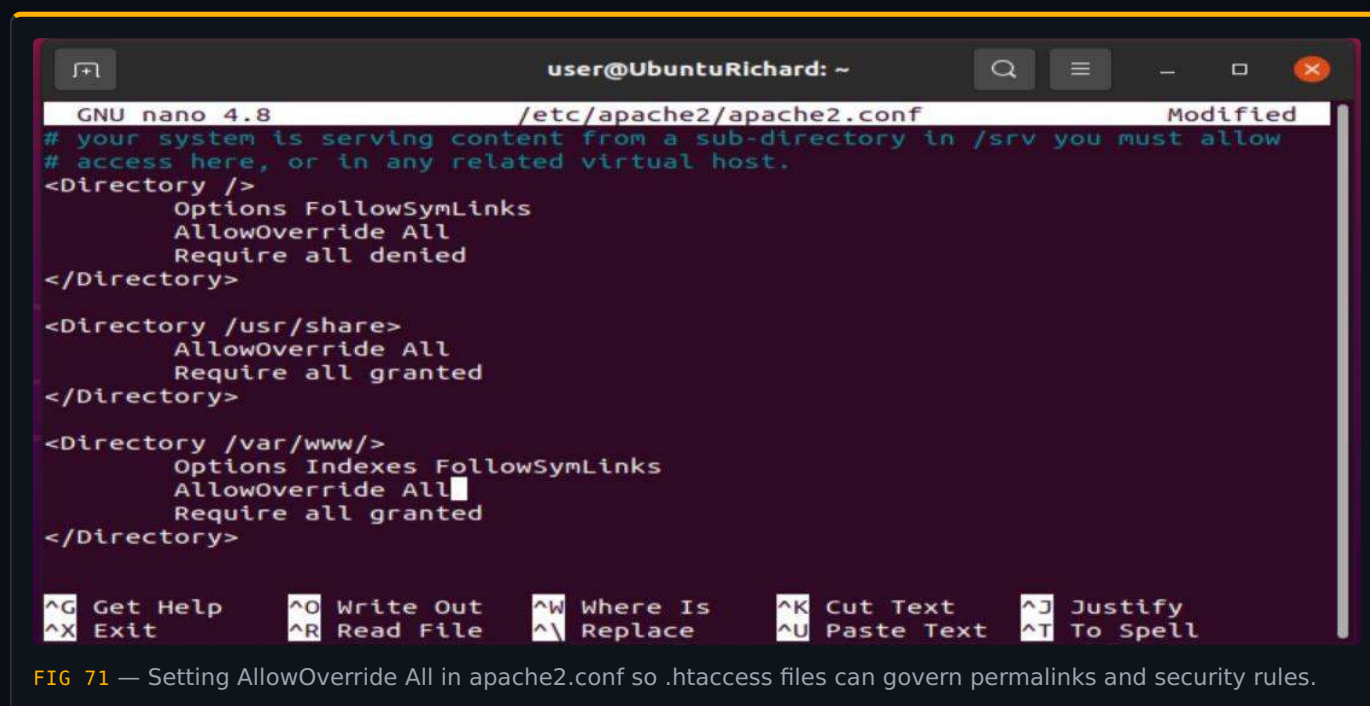


FIG 71 — Setting AllowOverride All in apache2.conf so .htaccess files can govern permalinks and security rules.

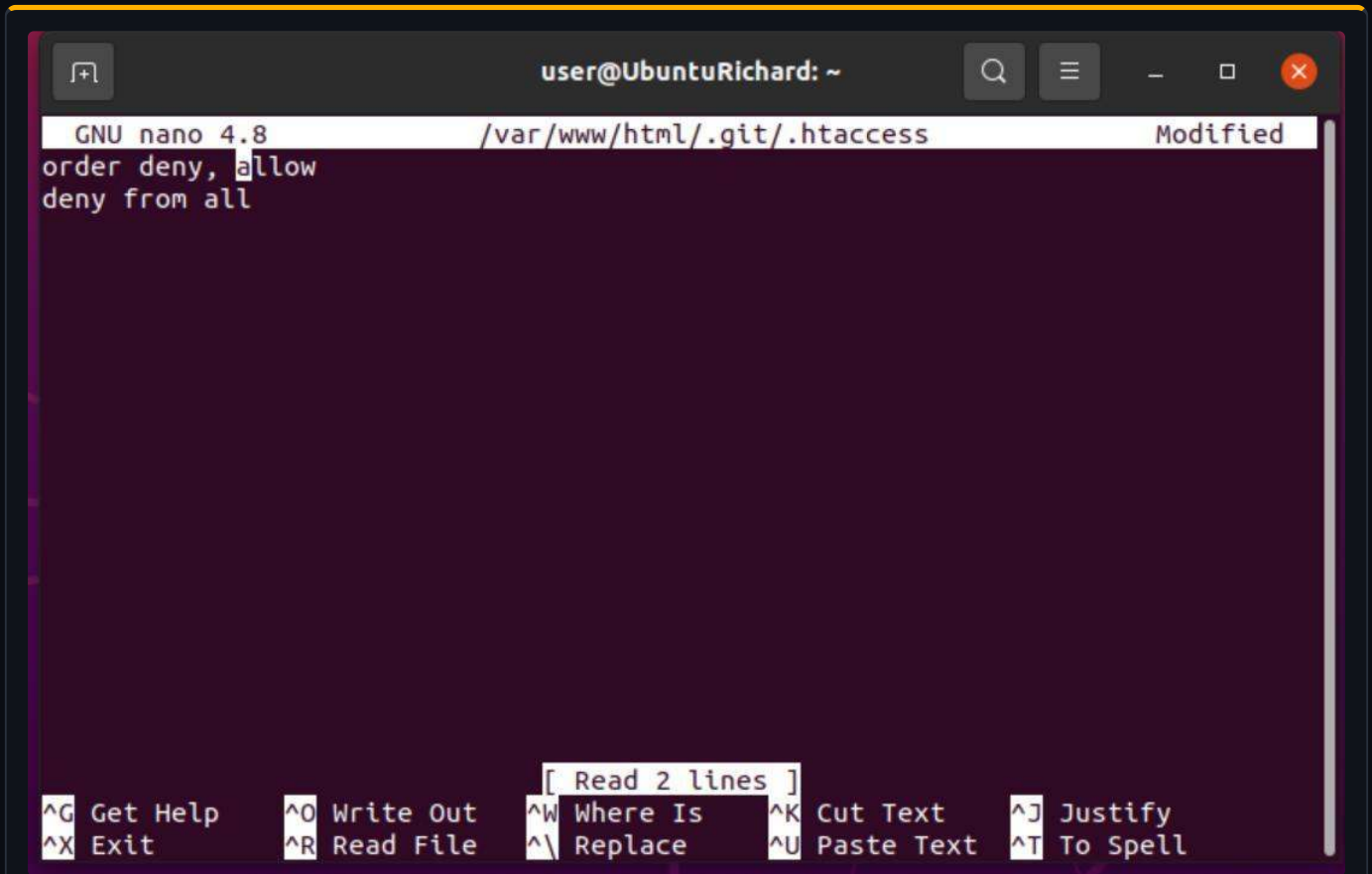


FIG 72 — Creating sudo nano /var/www/html/.git/.htaccess to block unauthorized access to the cloned .git directory.

## WORDPRESS

## Run the WordPress Installer

Restarted Apache, then ran the web installer against WordPressDB.

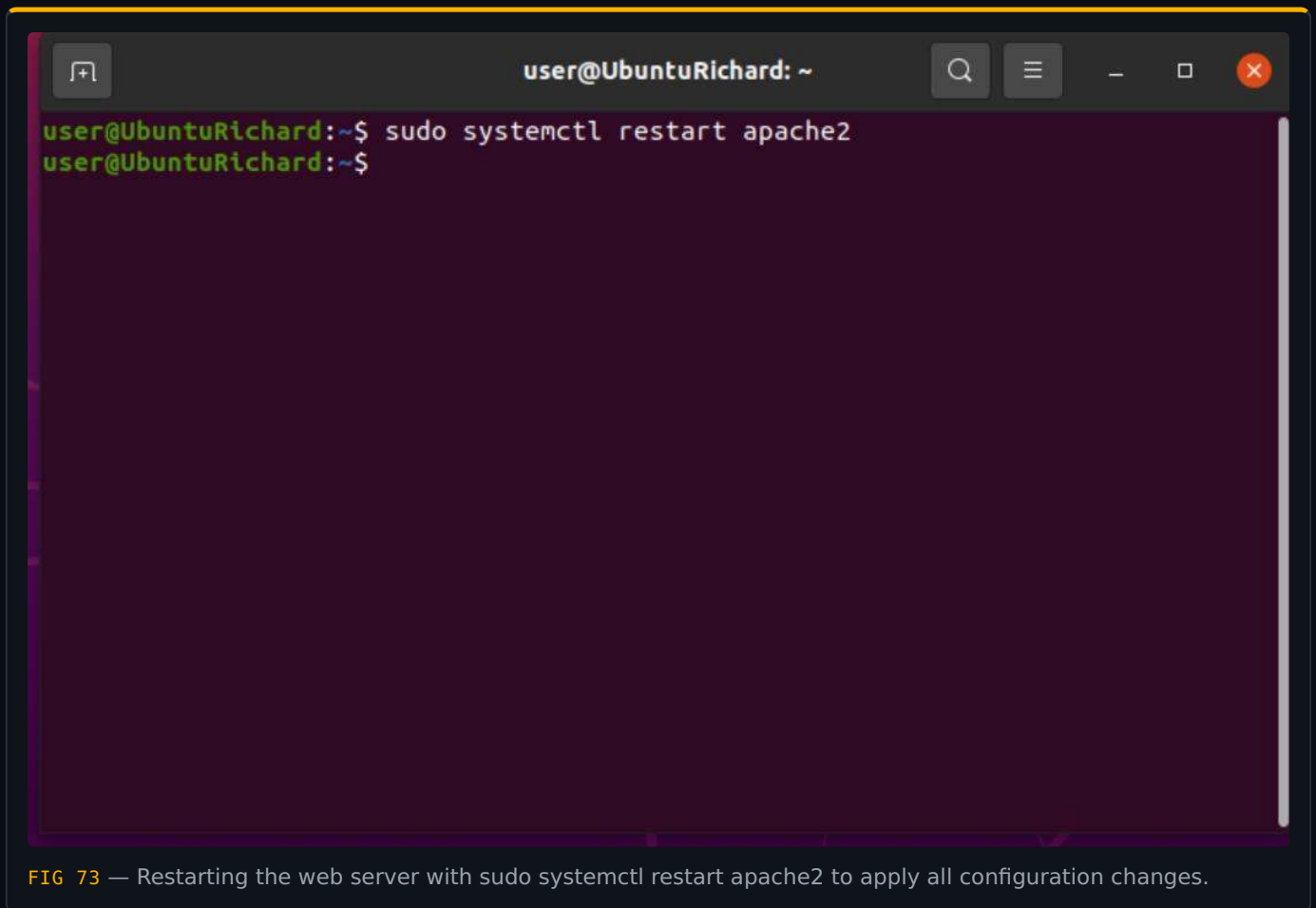


FIG 73 — Restarting the web server with `sudo systemctl restart apache2` to apply all configuration changes.

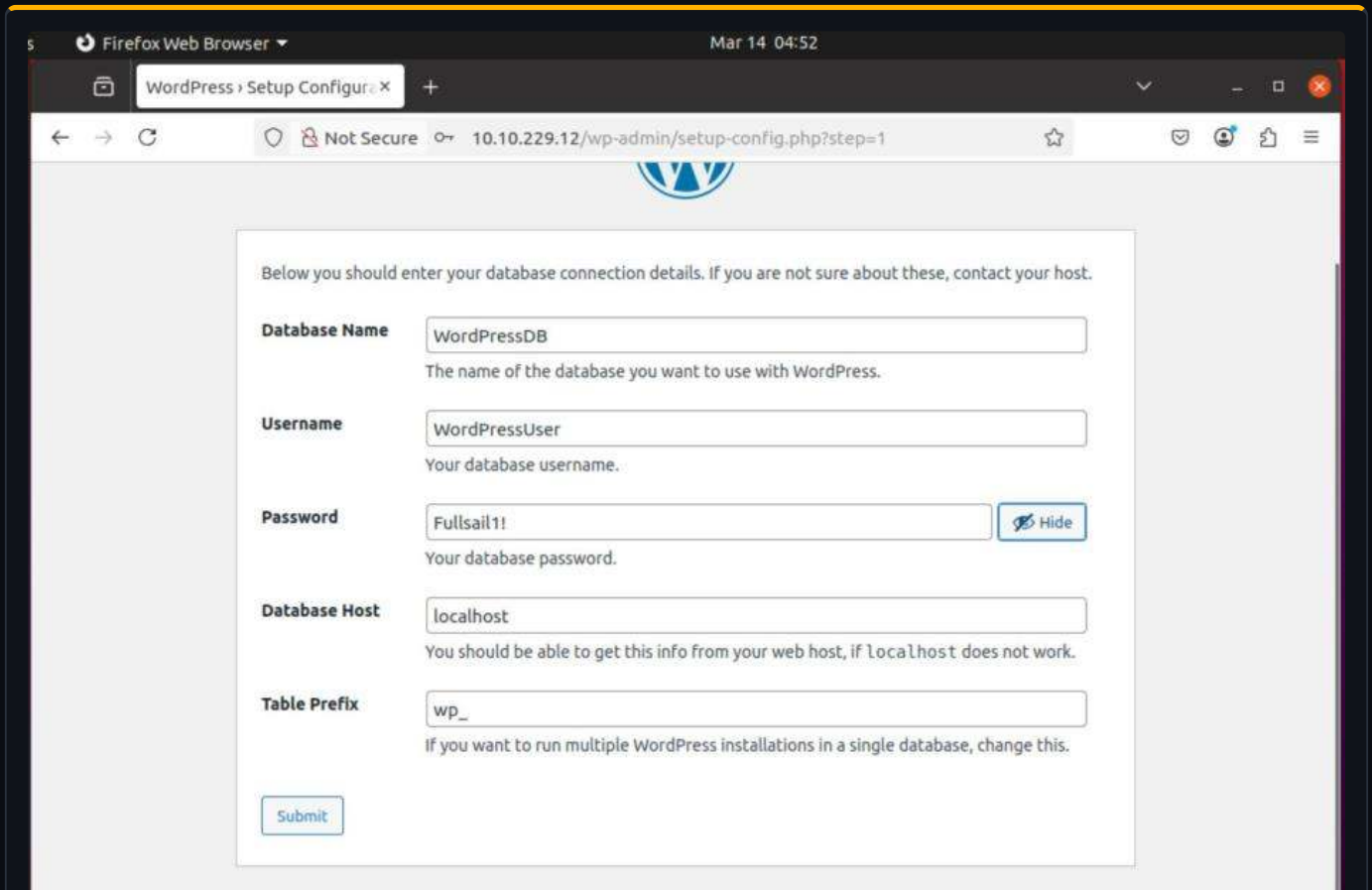


FIG 74 — The WordPress installer database step at <http://10.10.229.12>, entering WordPressDB, WordPressUser, and the password to connect WordPress to MySQL.

#### VALIDATION

### Installer Success and Live Site

Completed the install with an admin account, applied a theme, and published a post to prove the site is live.

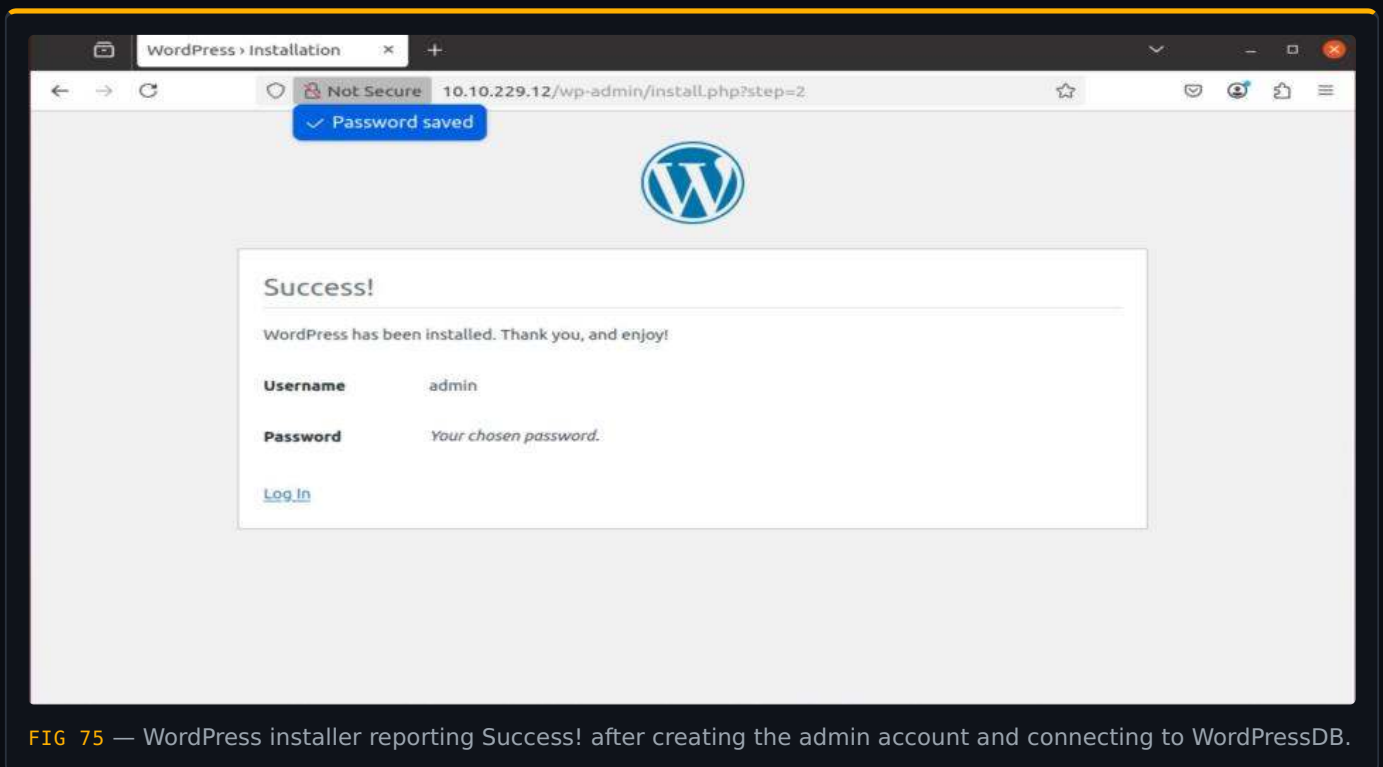


FIG 75 — WordPress installer reporting Success! after creating the admin account and connecting to WordPressDB.

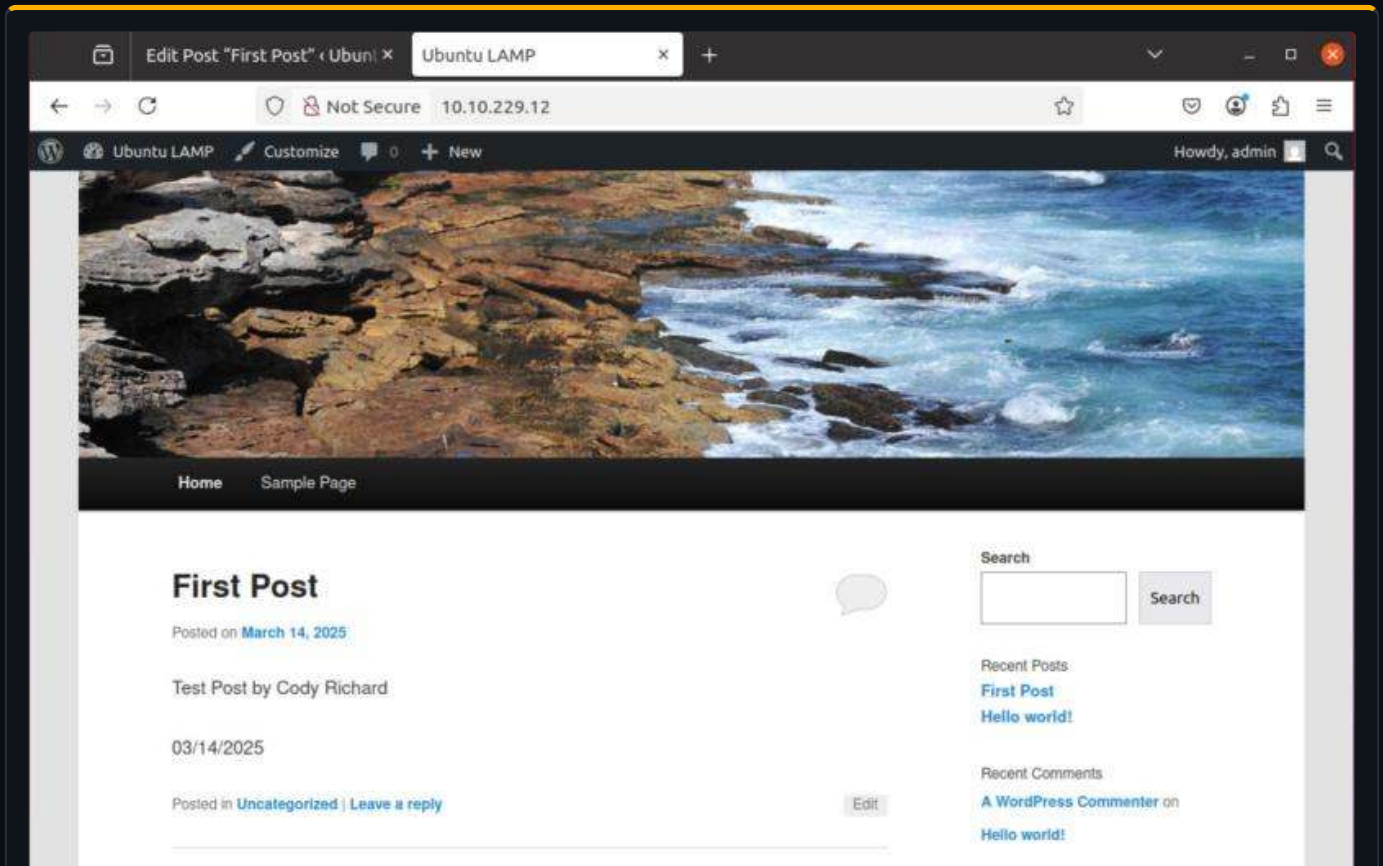
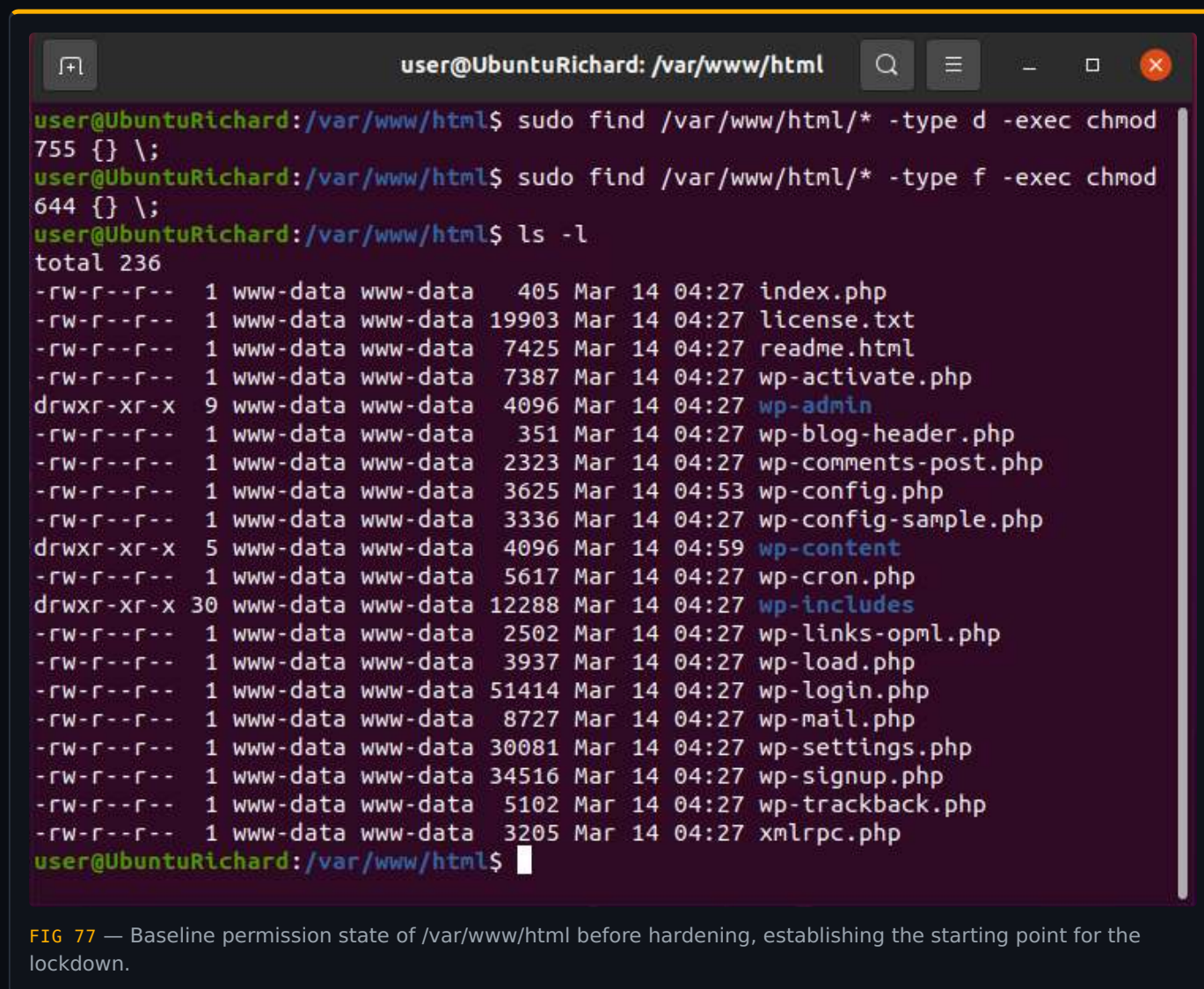


FIG 76 — The live WordPress site at <http://10.10.229.12> with a custom theme applied and a published post, confirming full end-to-end functionality.

## HARDENING

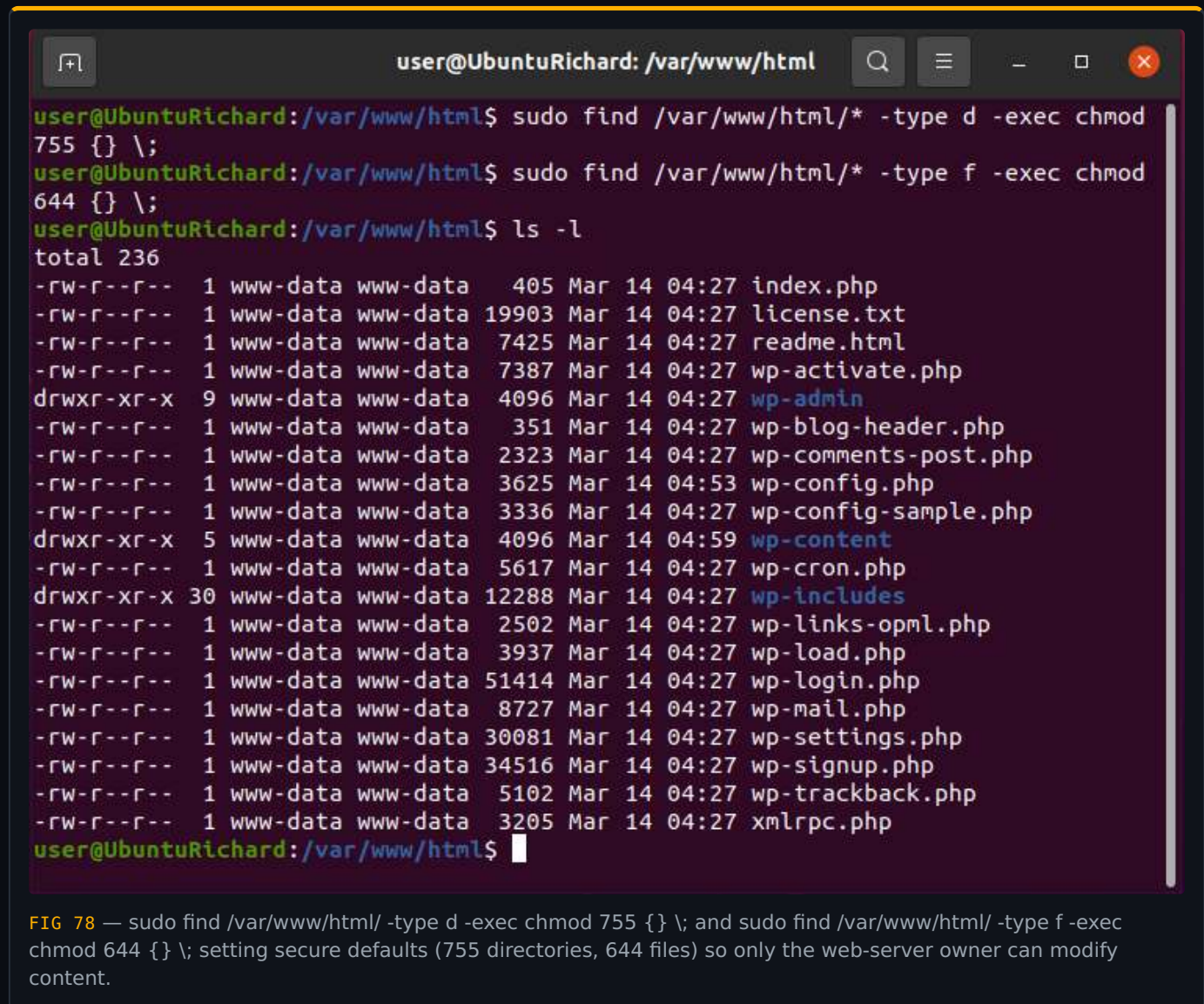
## File Permissions: Lockdown

Defense-in-depth, layer one: enforced 755 on directories and 644 on files across the web root.

A terminal window titled 'user@UbuntuRichard: /var/www/html' showing the execution of commands to set permissions on the web root. The user runs 'sudo find /var/www/html/\* -type d -exec chmod 755 {} \;' and 'sudo find /var/www/html/\* -type f -exec chmod 644 {} \;'. Then, they run 'ls -l' to display the current permissions for various files and directories. The output shows a list of files with permissions like '-rw-r--r--' for files and 'drwxr-xr-x' for directories, along with their owners, groups, sizes, dates, and names.

```
user@UbuntuRichard: /var/www/html$ sudo find /var/www/html/* -type d -exec chmod 755 {} \;
user@UbuntuRichard: /var/www/html$ sudo find /var/www/html/* -type f -exec chmod 644 {} \;
user@UbuntuRichard: /var/www/html$ ls -l
total 236
-rw-r--r--  1 www-data www-data  405 Mar 14 04:27 index.php
-rw-r--r--  1 www-data www-data 19903 Mar 14 04:27 license.txt
-rw-r--r--  1 www-data www-data  7425 Mar 14 04:27 readme.html
-rw-r--r--  1 www-data www-data  7387 Mar 14 04:27 wp-activate.php
drwxr-xr-x  9 www-data www-data  4096 Mar 14 04:27 wp-admin
-rw-r--r--  1 www-data www-data   351 Mar 14 04:27 wp-blog-header.php
-rw-r--r--  1 www-data www-data  2323 Mar 14 04:27 wp-comments-post.php
-rw-r--r--  1 www-data www-data  3625 Mar 14 04:53 wp-config.php
-rw-r--r--  1 www-data www-data  3336 Mar 14 04:27 wp-config-sample.php
drwxr-xr-x  5 www-data www-data  4096 Mar 14 04:59 wp-content
-rw-r--r--  1 www-data www-data  5617 Mar 14 04:27 wp-cron.php
drwxr-xr-x 30 www-data www-data 12288 Mar 14 04:27 wp-includes
-rw-r--r--  1 www-data www-data  2502 Mar 14 04:27 wp-links-opml.php
-rw-r--r--  1 www-data www-data  3937 Mar 14 04:27 wp-load.php
-rw-r--r--  1 www-data www-data 51414 Mar 14 04:27 wp-login.php
-rw-r--r--  1 www-data www-data  8727 Mar 14 04:27 wp-mail.php
-rw-r--r--  1 www-data www-data 30081 Mar 14 04:27 wp-settings.php
-rw-r--r--  1 www-data www-data 34516 Mar 14 04:27 wp-signup.php
-rw-r--r--  1 www-data www-data  5102 Mar 14 04:27 wp-trackback.php
-rw-r--r--  1 www-data www-data  3205 Mar 14 04:27 xmlrpc.php
user@UbuntuRichard: /var/www/html$
```

FIG 77 — Baseline permission state of /var/www/html before hardening, establishing the starting point for the lockdown.



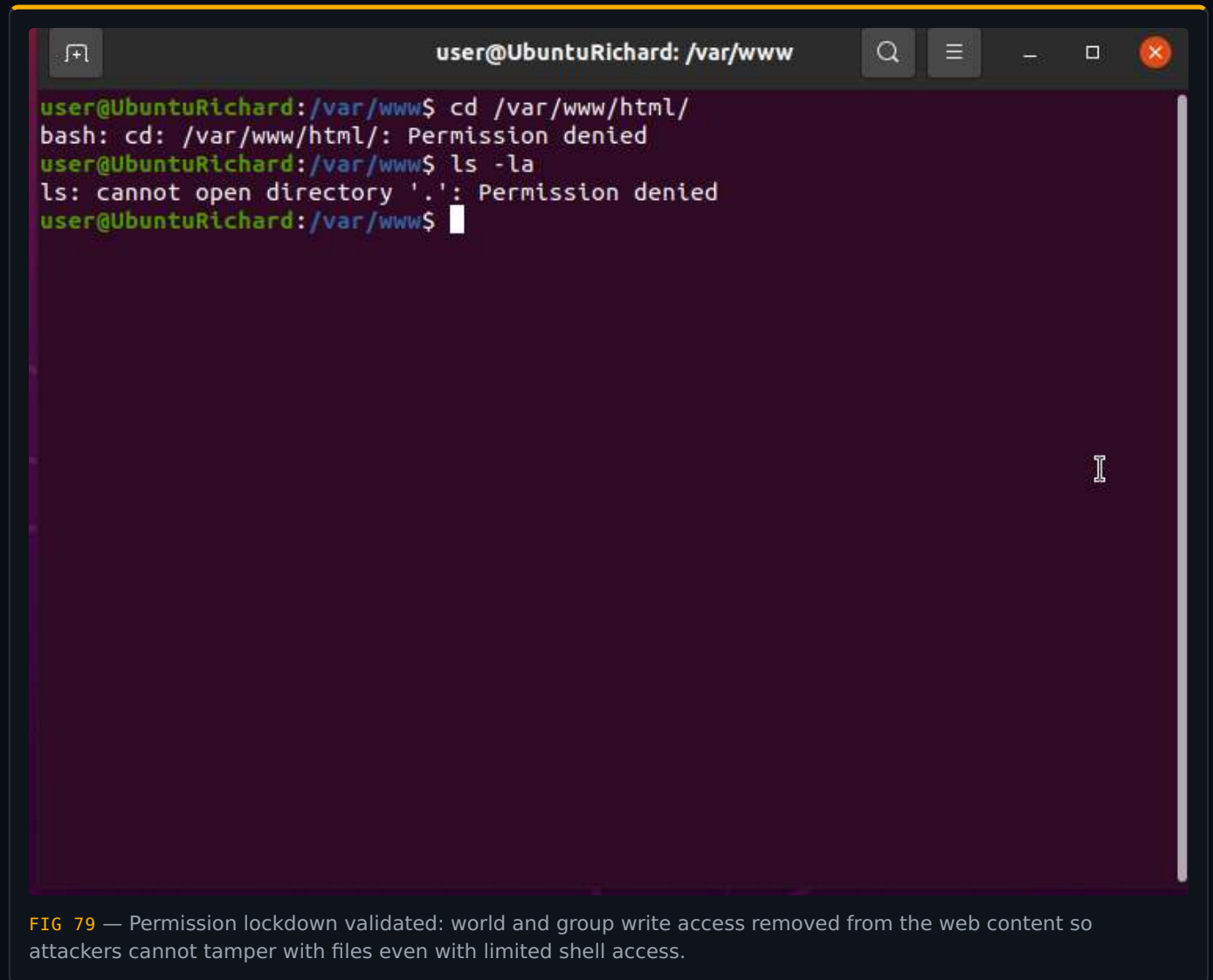
```
user@UbuntuRichard: /var/www/html
user@UbuntuRichard:/var/www/html$ sudo find /var/www/html/* -type d -exec chmod 755 {} \;
user@UbuntuRichard:/var/www/html$ sudo find /var/www/html/* -type f -exec chmod 644 {} \;
user@UbuntuRichard:/var/www/html$ ls -l
total 236
-rw-r--r--  1 www-data www-data  405 Mar 14 04:27 index.php
-rw-r--r--  1 www-data www-data 19903 Mar 14 04:27 license.txt
-rw-r--r--  1 www-data www-data  7425 Mar 14 04:27 readme.html
-rw-r--r--  1 www-data www-data  7387 Mar 14 04:27 wp-activate.php
drwxr-xr-x  9 www-data www-data  4096 Mar 14 04:27 wp-admin
-rw-r--r--  1 www-data www-data   351 Mar 14 04:27 wp-blog-header.php
-rw-r--r--  1 www-data www-data  2323 Mar 14 04:27 wp-comments-post.php
-rw-r--r--  1 www-data www-data  3625 Mar 14 04:53 wp-config.php
-rw-r--r--  1 www-data www-data  3336 Mar 14 04:27 wp-config-sample.php
drwxr-xr-x  5 www-data www-data  4096 Mar 14 04:59 wp-content
-rw-r--r--  1 www-data www-data  5617 Mar 14 04:27 wp-cron.php
drwxr-xr-x 30 www-data www-data 12288 Mar 14 04:27 wp-includes
-rw-r--r--  1 www-data www-data  2502 Mar 14 04:27 wp-links-opml.php
-rw-r--r--  1 www-data www-data  3937 Mar 14 04:27 wp-load.php
-rw-r--r--  1 www-data www-data 51414 Mar 14 04:27 wp-login.php
-rw-r--r--  1 www-data www-data  8727 Mar 14 04:27 wp-mail.php
-rw-r--r--  1 www-data www-data 30081 Mar 14 04:27 wp-settings.php
-rw-r--r--  1 www-data www-data 34516 Mar 14 04:27 wp-signup.php
-rw-r--r--  1 www-data www-data  5102 Mar 14 04:27 wp-trackback.php
-rw-r--r--  1 www-data www-data  3205 Mar 14 04:27 xmlrpc.php
user@UbuntuRichard:/var/www/html$
```

FIG 78 — `sudo find /var/www/html/ -type d -exec chmod 755 {} \;` and `sudo find /var/www/html/ -type f -exec chmod 644 {} \;`; setting secure defaults (755 directories, 644 files) so only the web-server owner can modify content.

## HARDENING

## Relocate and Lock wp-config.php

Layer two: moved wp-config.php above the web root and restricted it to owner-only read/write (600).



**FIG 79** — Permission lockdown validated: world and group write access removed from the web content so attackers cannot tamper with files even with limited shell access.

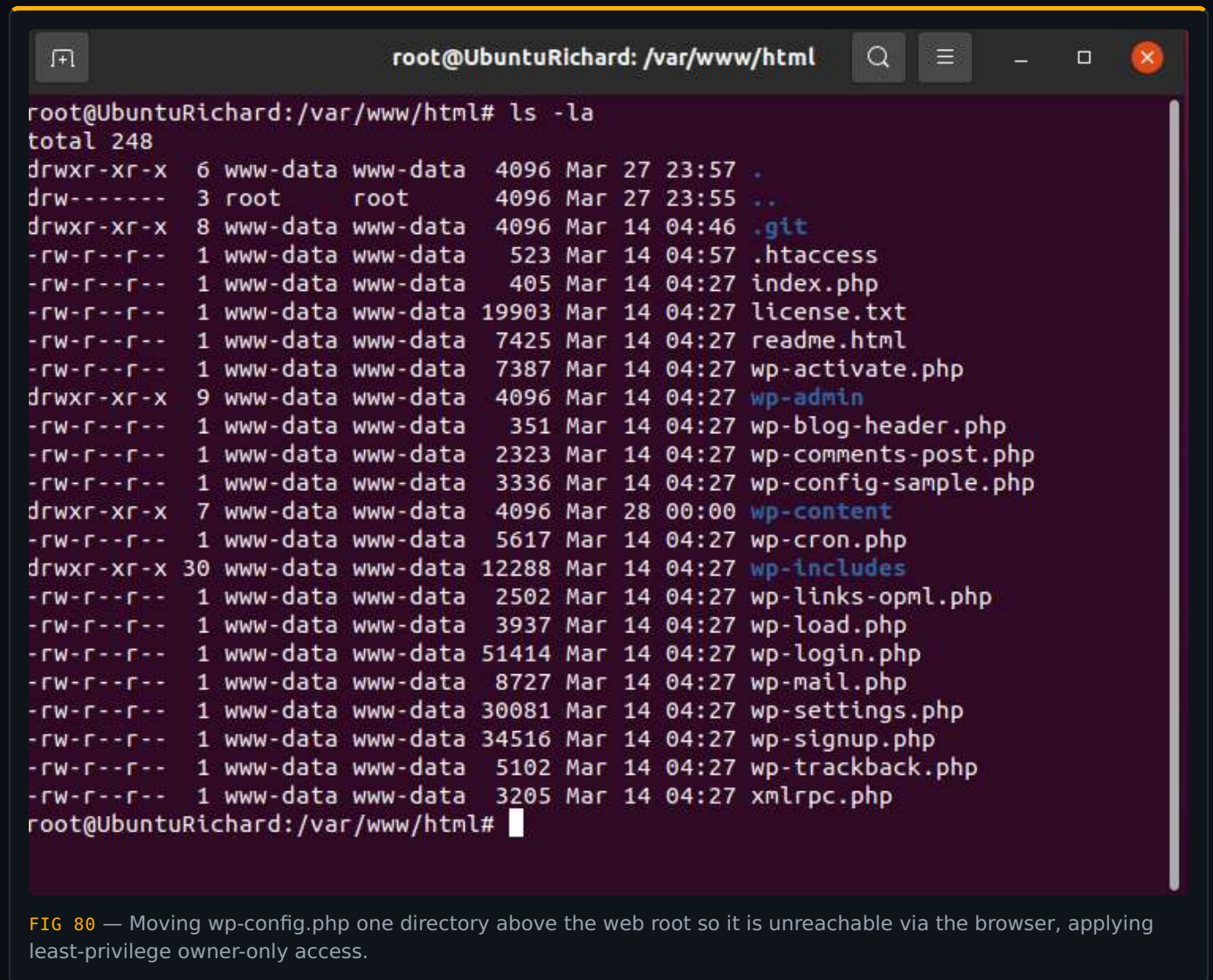
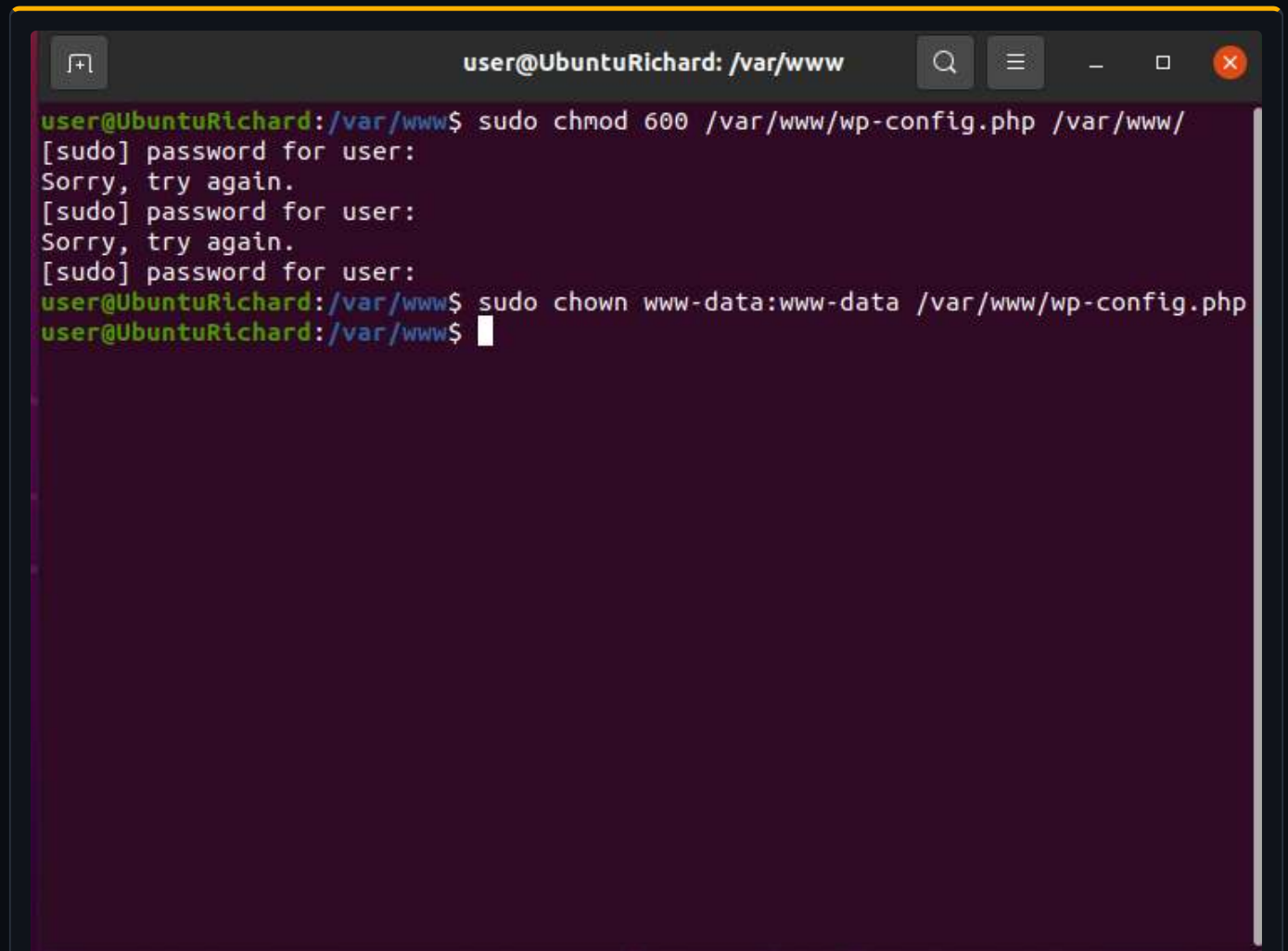


FIG 80 — Moving wp-config.php one directory above the web root so it is unreachable via the browser, applying least-privilege owner-only access.

## HARDENING

**wp-config.php: Configure and Validate**A terminal window titled 'user@UbuntuRichard: /var/www' showing a series of commands and their outputs. The user runs 'sudo chmod 600 /var/www/wp-config.php /var/www/'. The terminal shows three password prompts for 'user' and three 'Sorry, try again.' messages. Finally, the user runs 'sudo chown www-data:www-data /var/www/wp-config.php' and the terminal shows the command being executed. The terminal background is dark purple with light green text.

```
user@UbuntuRichard: /var/www$ sudo chmod 600 /var/www/wp-config.php /var/www/
[sudo] password for user:
Sorry, try again.
[sudo] password for user:
Sorry, try again.
[sudo] password for user:
user@UbuntuRichard: /var/www$ sudo chown www-data:www-data /var/www/wp-config.php
user@UbuntuRichard: /var/www$
```

**FIG 81** — Applying 600 (owner read/write only) permissions to the relocated wp-config.php to prevent database-credential exposure.

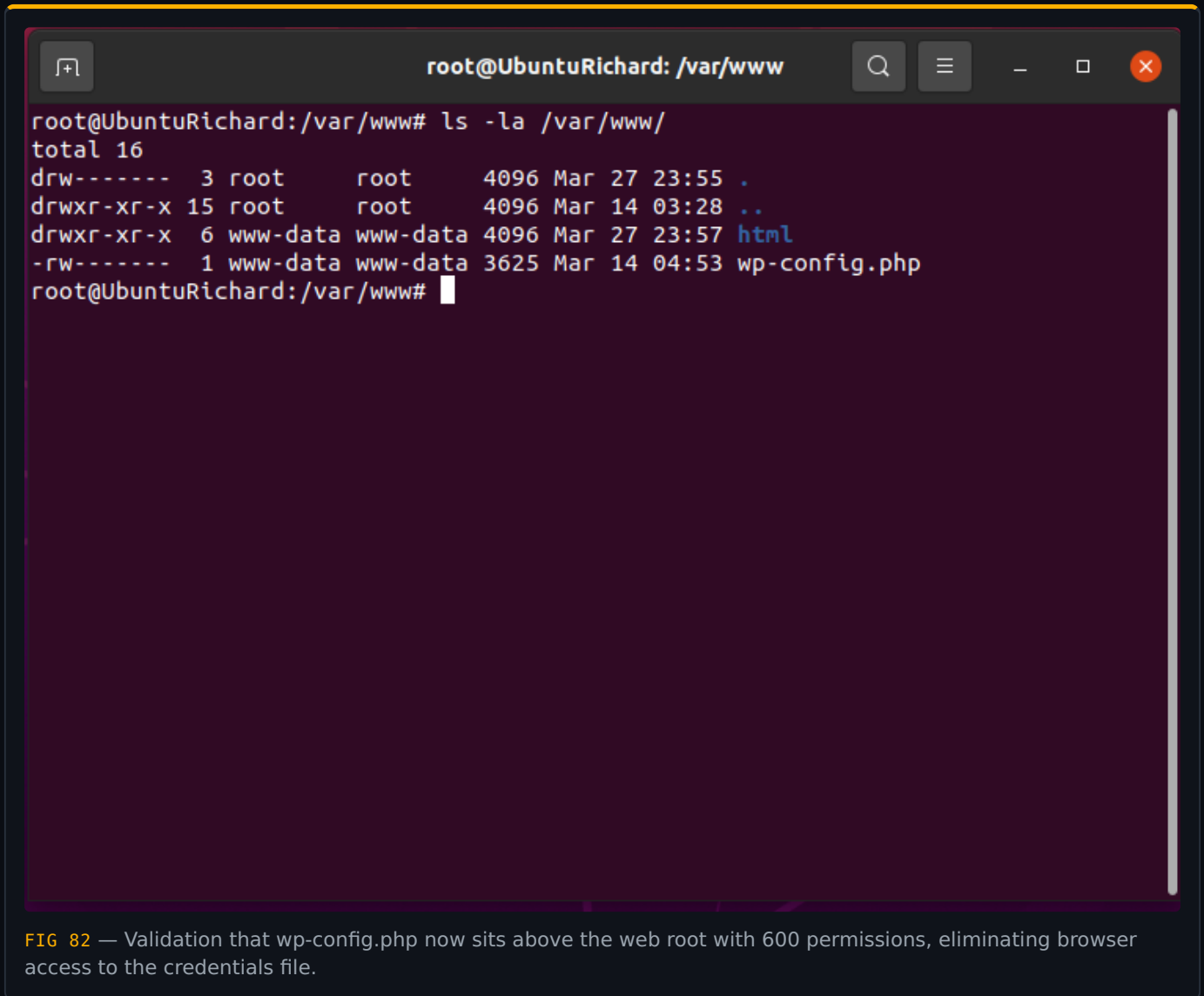


FIG 82 — Validation that wp-config.php now sits above the web root with 600 permissions, eliminating browser access to the credentials file.

HARDENING

# Wordfence Layer-7 WAF

Layer three: installed the Wordfence application firewall to inspect and filter HTTP(S) traffic at OSI Layer 7.

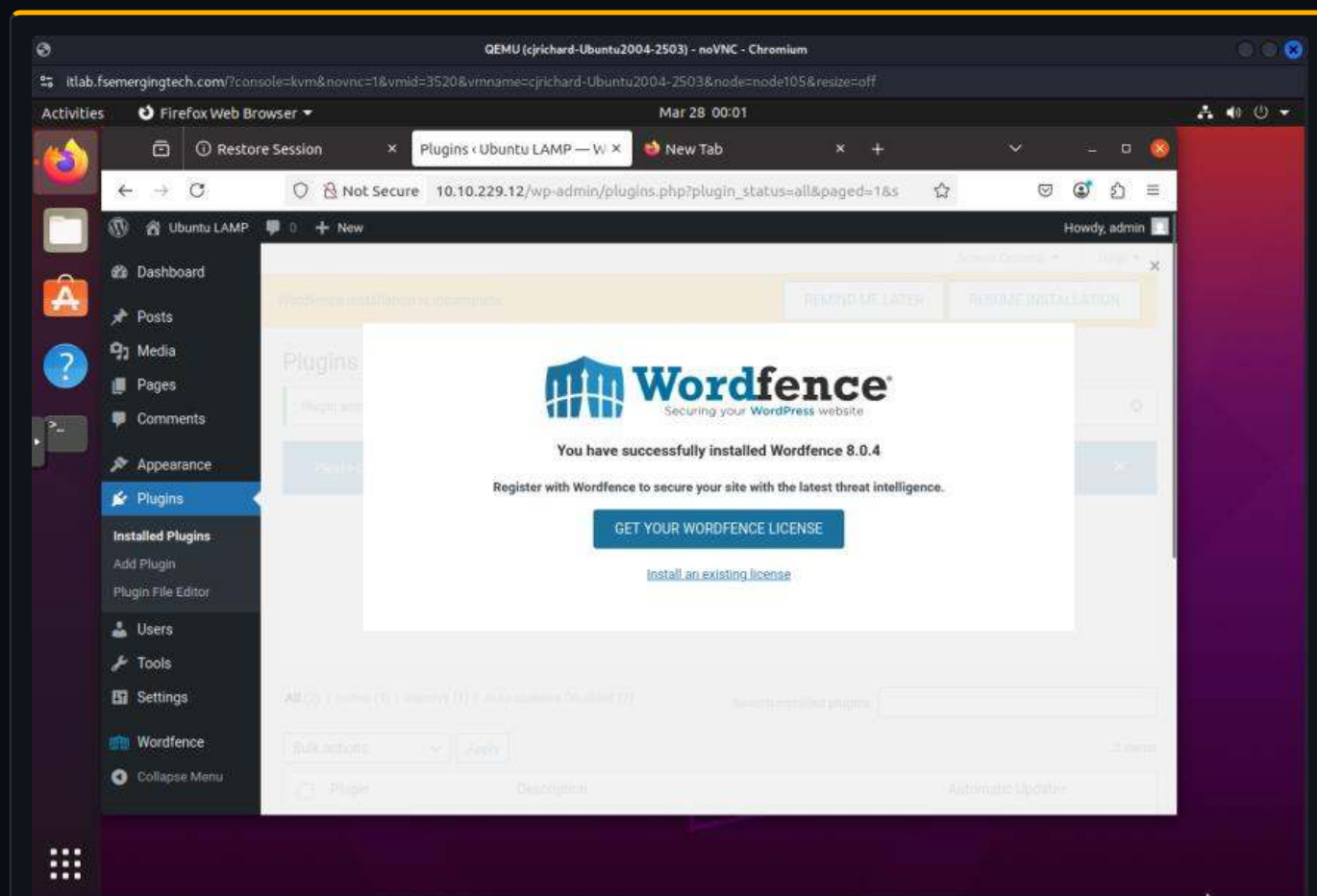


FIG 83 — Installing and configuring the Wordfence WAF plugin, adding Layer-7 inspection of HTTP(S) traffic for malicious requests.

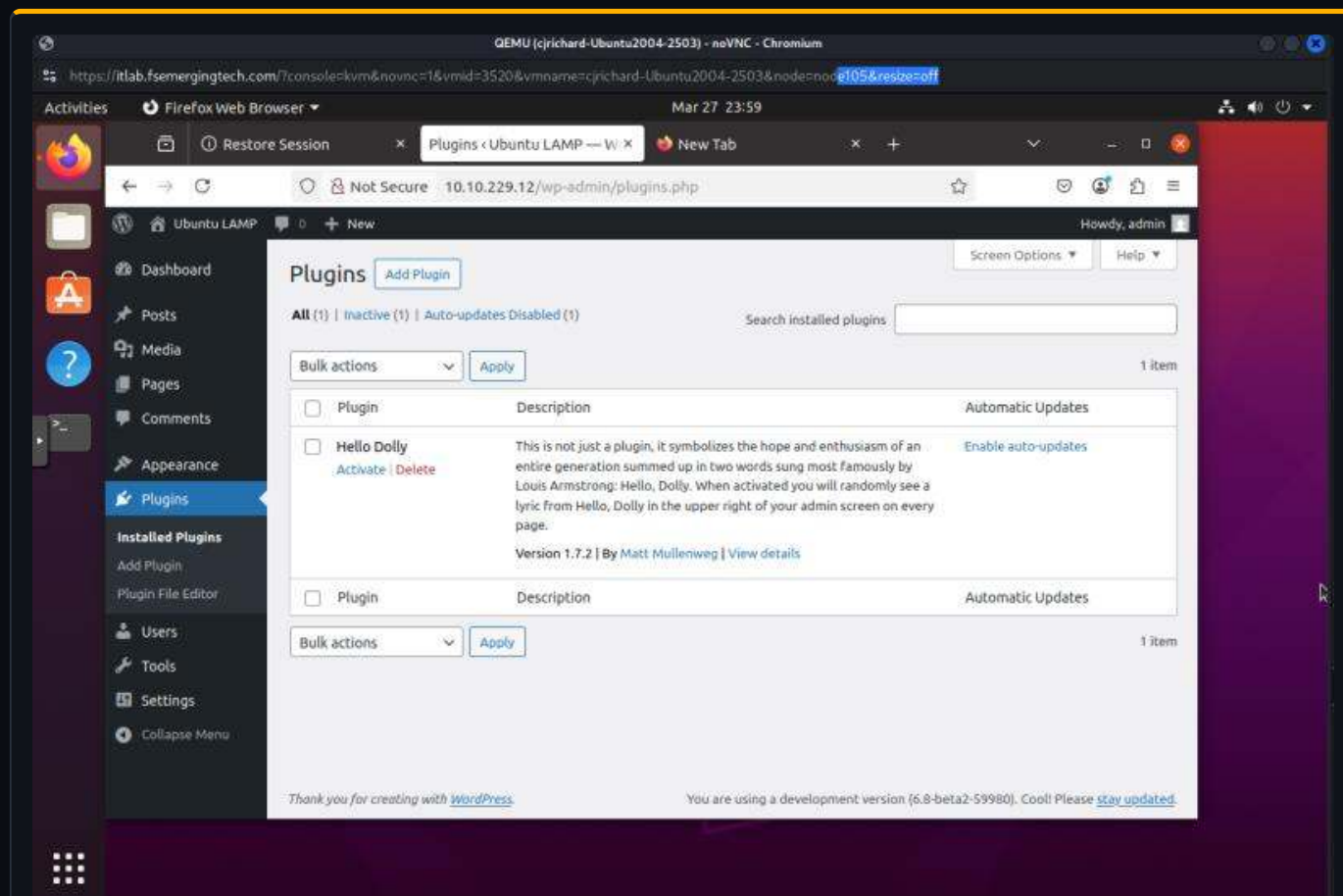


FIG 84 — Wordfence active and protecting against common application-layer attacks such as SQL injection, XSS, and brute-force logins.

## CONCLUSION

### Defense-in-Depth Summary

The conclusion ties the three layers together and documents resolving a 403 error and a redirect loop by correcting permissions and setting the site URL in wp-config.php.

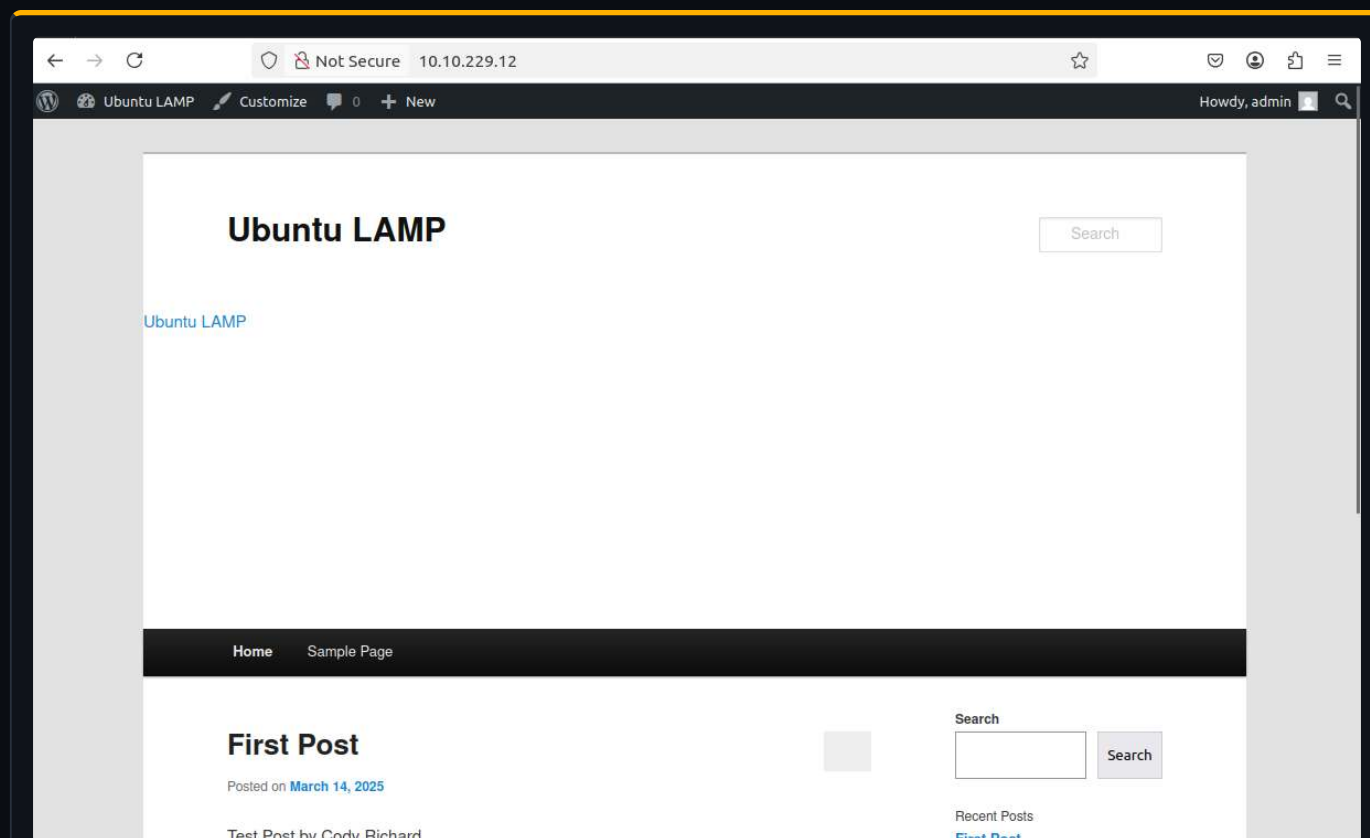


FIG 85 — Conclusion documenting the layered controls: 644/755 file and directory permissions, wp-config.php relocated to /var/www and set to 600, and Wordfence as a Layer-7 defense.

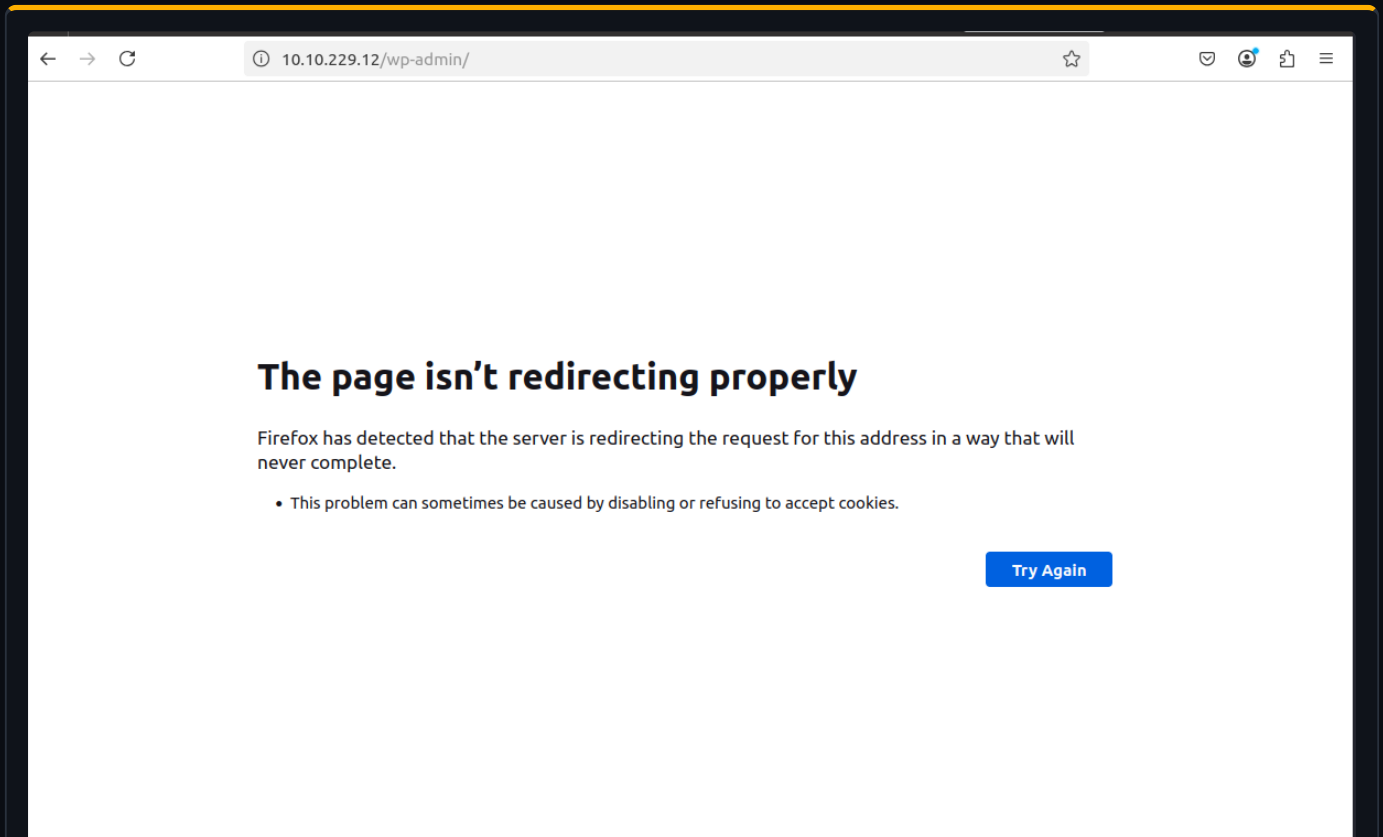
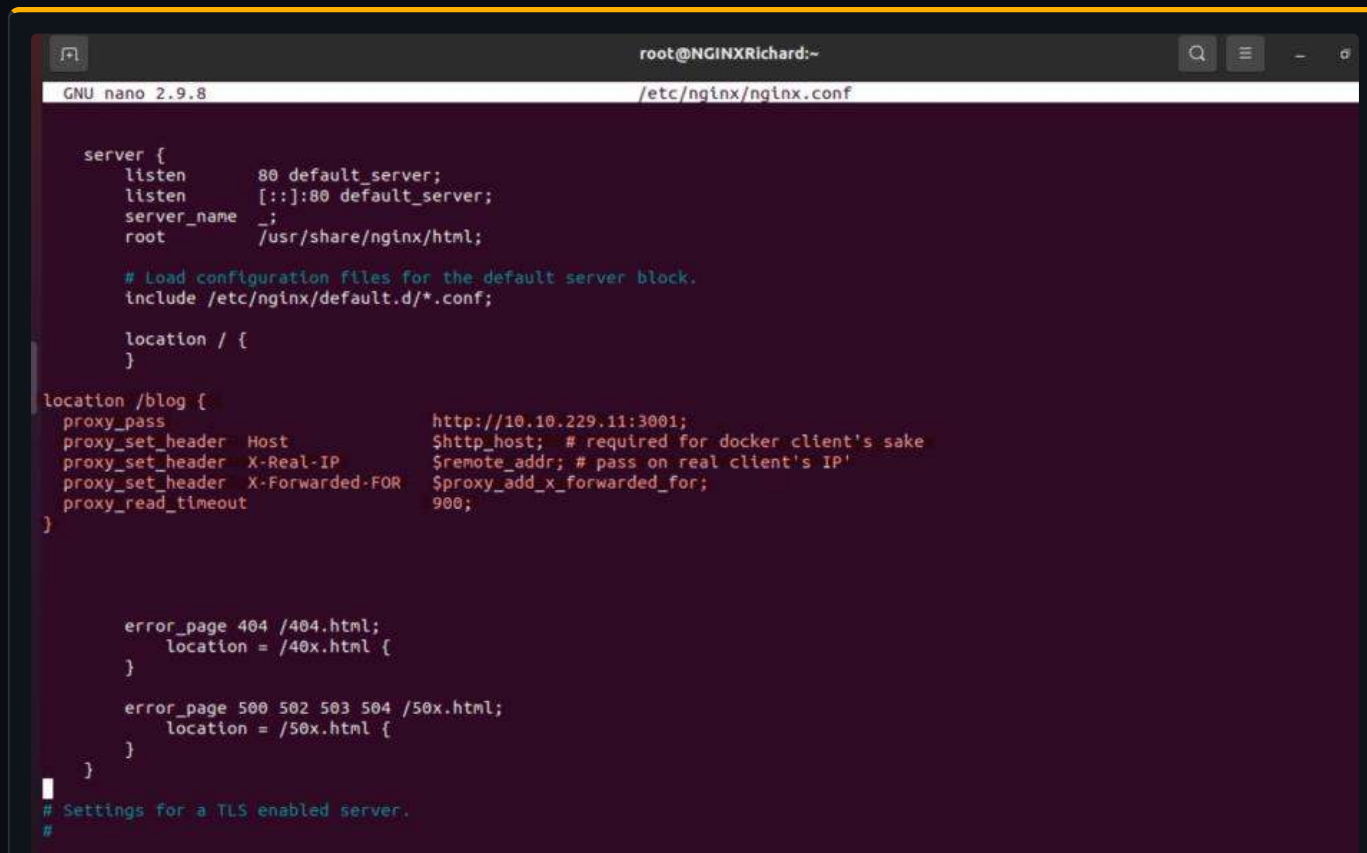


FIG 86 — Continued conclusion narrative on the defense-in-depth approach across file system, configuration, and application layers.

## APPENDIX A

## Nginx Config and Access Log

Reference appendices: the reverse-proxy configuration forwarding /blog to the Ghost backend, and the last 10 lines of the access log.



```
root@NGINXRichard:~
GNU nano 2.9.8 /etc/nginx/nginx.conf

server {
    listen      80 default_server;
    listen     [::]:80 default_server;
    server_name _;
    root       /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

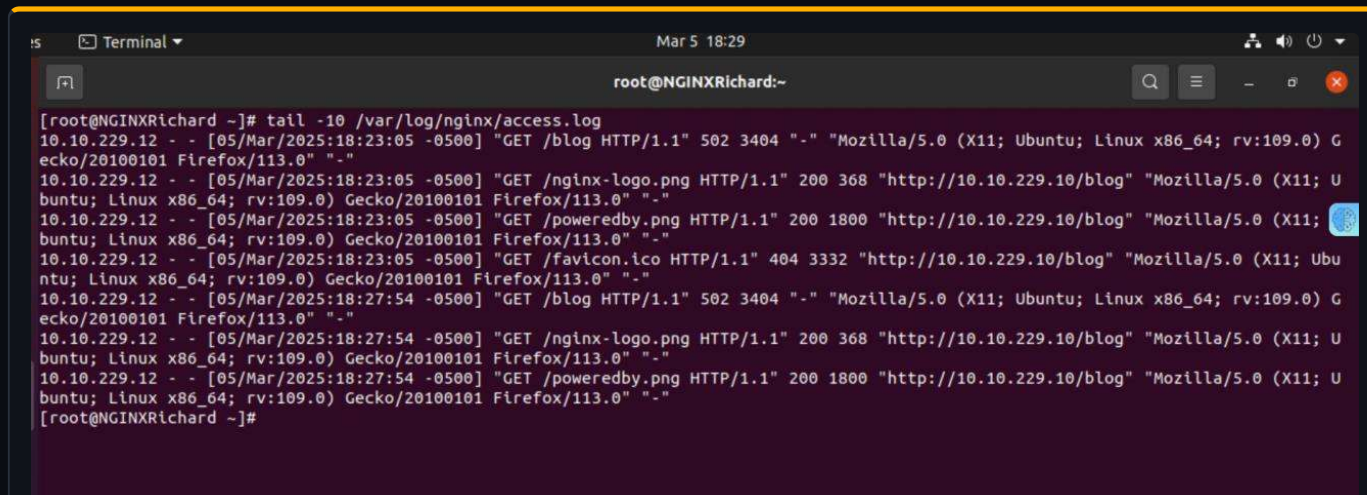
    location /blog {
        proxy_pass          http://10.10.229.11:3001;
        proxy_set_header    Host                $http_host; # required for docker client's sake
        proxy_set_header    X-Real-IP          $remote_addr; # pass on real client's IP
        proxy_set_header    X-Forwarded-For  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

    error_page 404 /404.html;
        location = /40x.html {
    }

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

# Settings for a TLS enabled server.
#
```

FIG 87 — Appendix A: /etc/nginx/nginx.conf showing the reverse proxy forwarding /blog requests to the Ghost backend.



```
is Terminal Mar 5 18:29
root@NGINXRichard:~

[root@NGINXRichard ~]# tail -10 /var/log/nginx/access.log
10.10.229.12 - - [05/Mar/2025:18:23:05 -0500] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:23:05 -0500] "GET /nginx-logo.png HTTP/1.1" 200 368 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:23:05 -0500] "GET /poweredby.png HTTP/1.1" 200 1800 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:23:05 -0500] "GET /favicon.ico HTTP/1.1" 404 3332 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:27:54 -0500] "GET /blog HTTP/1.1" 502 3404 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:27:54 -0500] "GET /nginx-logo.png HTTP/1.1" 200 368 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:27:54 -0500] "GET /poweredby.png HTTP/1.1" 200 1800 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
10.10.229.12 - - [05/Mar/2025:18:27:54 -0500] "GET /poweredby.png HTTP/1.1" 200 1800 "http://10.10.229.10/blog" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0" "-"
[root@NGINXRichard ~]#
```

FIG 88 — Appendix B: tail -10 /var/log/nginx/access.log showing the most recent incoming requests to the Nginx server.

## APPENDIX B

## Nginx Error Log

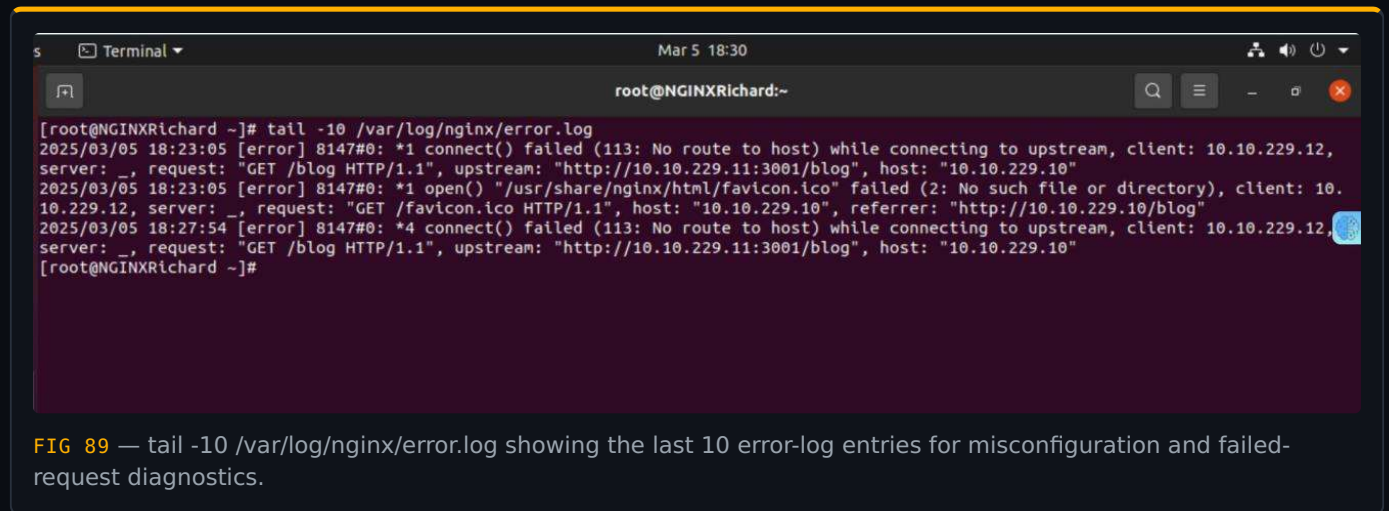


FIG 89 — tail -10 /var/log/nginx/error.log showing the last 10 error-log entries for misconfiguration and failed-request diagnostics.