

02

CTI1301 - VCO (VIRTUALIZATION, CLOUD, AND OPERATIONS)

Containerized WordPress Deployment on Rocky Linux with Docker and Docker Compose

Installing Docker Engine on a Proxmox-hosted Rocky Linux VM and standing up a two-container WordPress + MySQL stack with port-mapped access

COURSE

CTI1301 - VCO (Virtualization, Cloud, and Operations)

DATE

Section C202411; lab executed Sun Nov 17, 2024 (19:06 EST per terminal date output)

ENVIRONMENT

Proxmox VE 8.1.4 virtualization cluster (Full Sail IT lab, node IT127)

PAGES (REPORT)

19

STUDENT

Cody Richard

ORGANIZATION

Full Sail University

Containerized WordPress Deployment on Rocky Linux with Docker and Docker Compose

For this final Docker lab I installed Docker Engine from source on a Rocky Linux VM running on the Full Sail IT lab Proxmox VE 8.1.4 cluster (VM 289 "CodyRichard-Rocky," 4 GB RAM, 4 cores, accessed via noVNC at itlab.fsemergingtech.com), then used Docker Compose to deploy a complete WordPress application stack. I added the official Docker CE repository, installed and enabled the docker service, wrote a docker-compose.yml defining WordPress and MySQL 8.0 containers, and ran the stack with `docker compose up -d`. I verified both containers (`wordpress-wordpress-1` and `wordpress-db-1`) were running via `docker ps`, confirmed the port mapping `0.0.0.0:8080->80/tcp`, and inspected the host and Docker bridge networking (`ens18` at `10.1.102.52/22`, `docker0` at `172.17.0.1/16`). The deployment was validated end to end by reaching the WordPress installation and admin portal from a separate Windows 10 Pro lab VM through the host:port URL `10.1.102.52:8080`, then cleanly powering down all VMs.

► Objectives

- Install Docker Engine (Docker CE) on a Rocky Linux host using the official Docker repository
- Enable and verify the docker systemd service so containers persist across reboots
- Author and execute a docker-compose.yml to orchestrate a multi-container WordPress + MySQL stack
- Use Docker port mapping to expose the containerized web app to the lab network
- Validate the running deployment via `docker ps`, host networking, and a browser-based install/admin walkthrough
- Demonstrate full container lifecycle including a clean shutdown of all lab VMs

► Environment

```
Proxmox VE 8.1.4 virtualization cluster (Full Sail IT lab, node IT127)
```

```
Rocky Linux guest VM 289 'CodyRichard-Rocky'  
- 4 GB RAM, 4 sockets/4 cores (SandyBridge),  
VMXNET3/virtio NIC on vmbr0
```

```
Console access via noVNC over  
itlab.fsemergingtech.com (QEMU/KVM)
```

```
Host networking: ens18 10.1.102.52/22; Docker  
bridges docker0 172.17.0.1/16 and  
br-4dd4f7bb5c55 172.18.0.1/16
```

```
Separate Windows 10 Pro VM (VM 795, node  
IT122) used as the client to reach the site  
via Microsoft Edge
```

```
WordPress container published on host port  
8080 mapped to container port 80
```

WALKTHROUGH & EVIDENCE

This lab walks through standing up a containerized WordPress stack on a Rocky Linux VM running on the Full Sail Proxmox VE 8.1.4 cluster. I install Docker Engine from the official Docker CE repository, author a docker-compose.yml to orchestrate WordPress alongside MySQL 8.0, then publish the app on host port 8080 and validate it end to end from a separate Windows 10 client. Every step is captured from the noVNC console and the client browser, from repo setup through a clean shutdown of all VMs.

ENVIRONMENT

Rocky Linux VM on Proxmox

The lab host is VM 289 "CodyRichard-Rocky" provisioned on the Full Sail Proxmox VE 8.1.4 cluster with 4 GB RAM and 4 cores, accessed over a noVNC console.

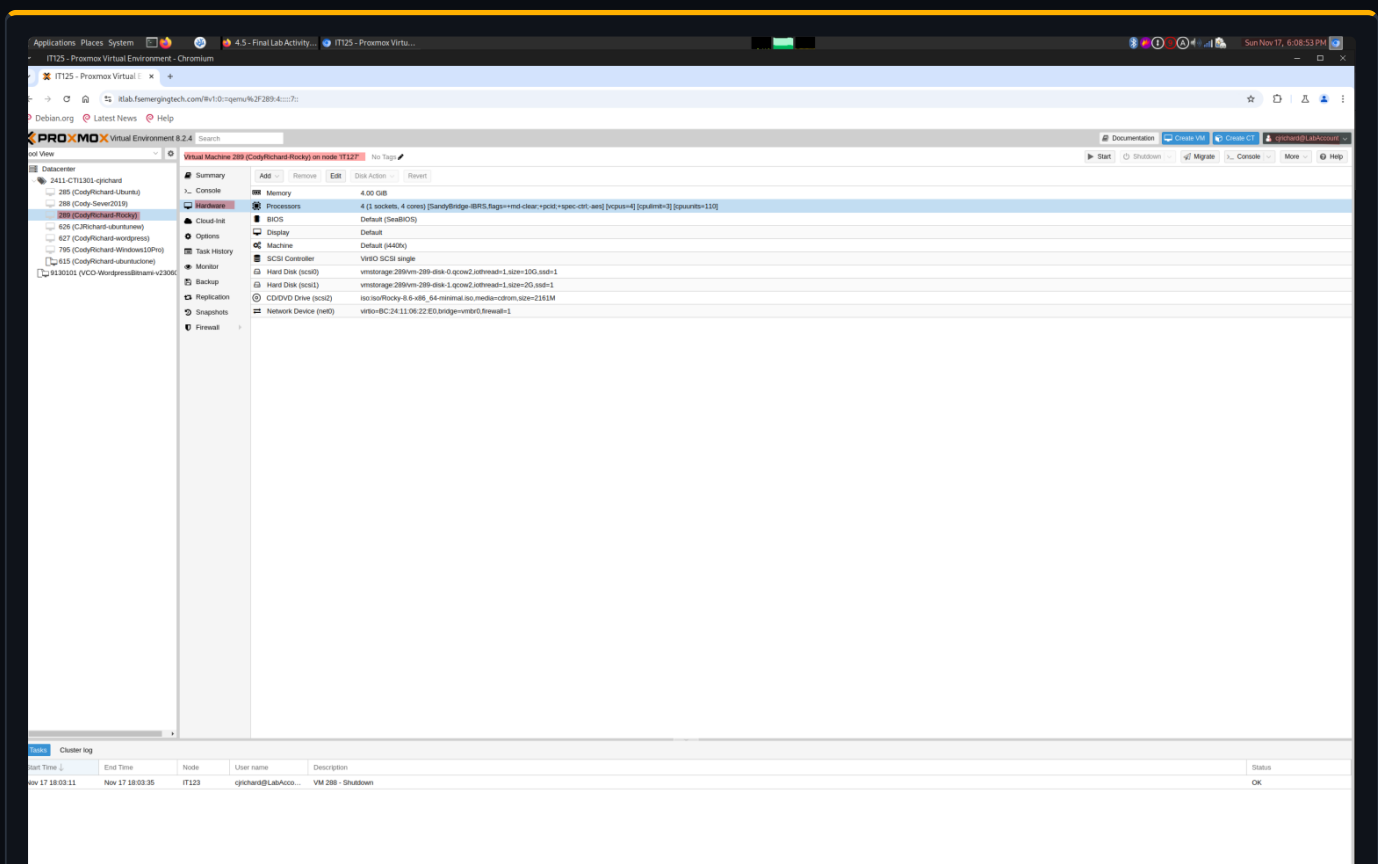


FIG 01 — Proxmox VE hardware view for VM 289 CodyRichard-Rocky, showing the Rocky Linux guest configured with 4 GB RAM and 4 sockets/cores on the Full Sail IT lab cluster.

SETUP

Repository Tooling

Before adding the Docker repository, I install yum-utils to provide yum-config-manager.

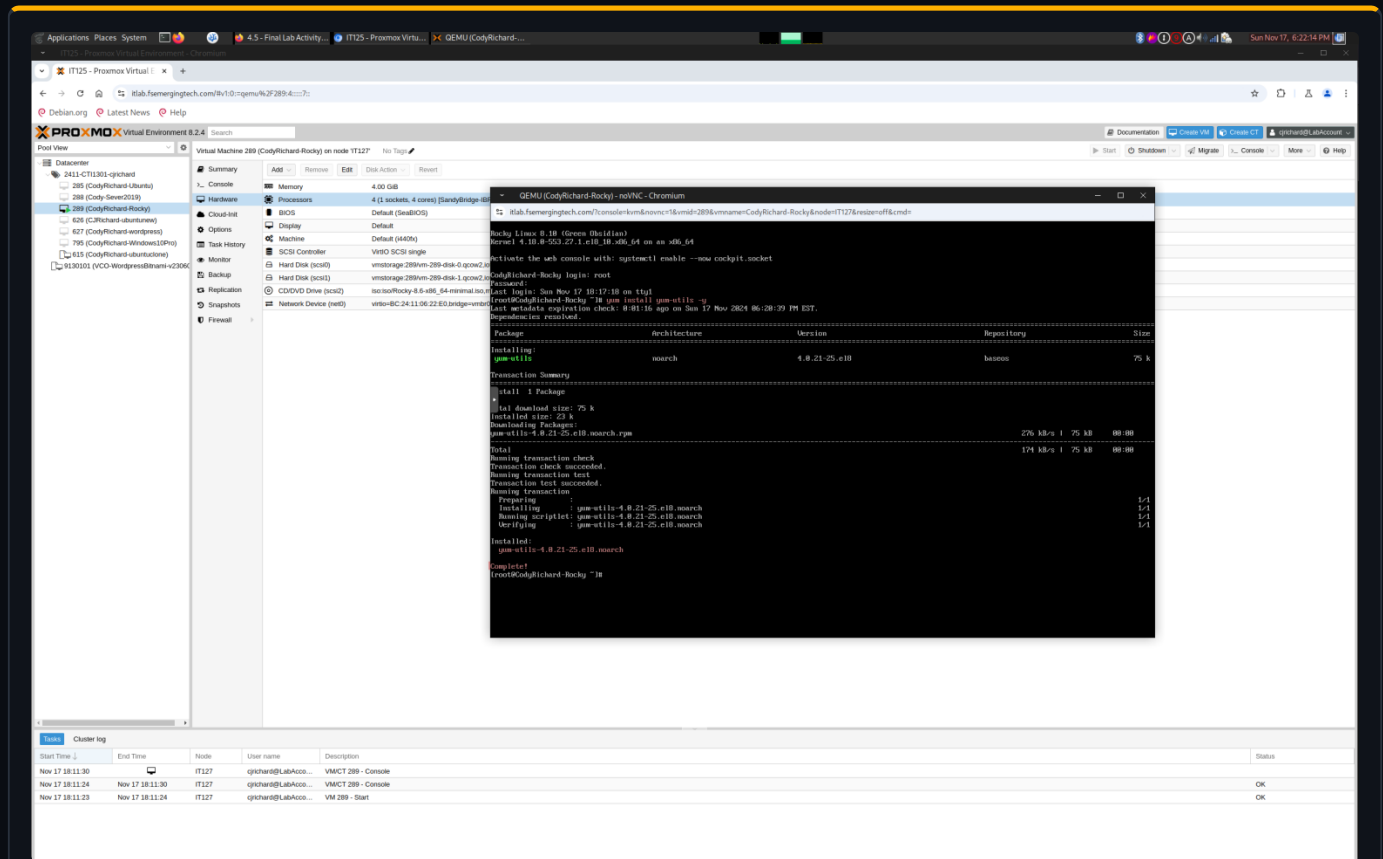


FIG 02 — Installing the yum-utils package, which supplies the yum-config-manager tooling used to register the Docker CE repository.

SETUP

Adding the Docker CE Repository

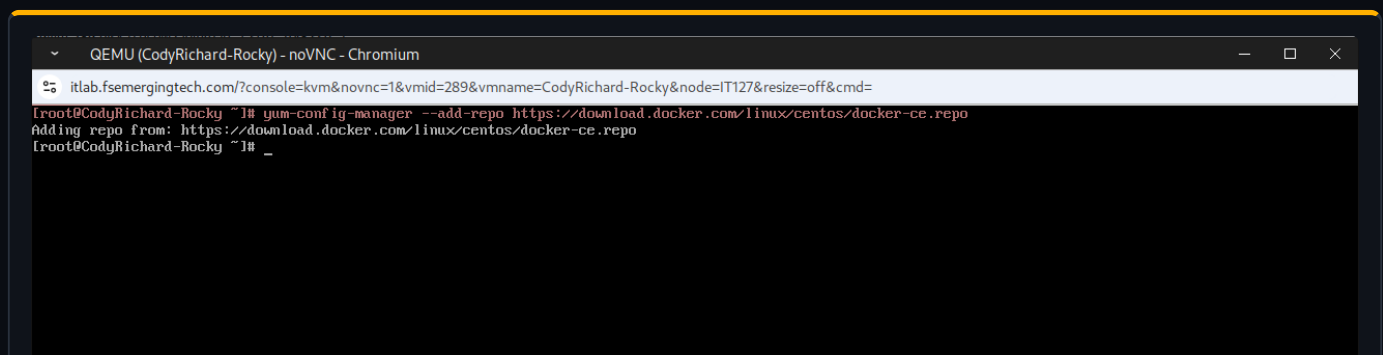


FIG 03 — Registering the official Docker repository: yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo.

INSTALL

Installing Docker Engine

```
QEMU (CodyRichard-Rocky) - noVNC - Chromium
itlab.fsmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
root@CodyRichard-Rocky ~]# yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

FIG 04 — Issuing the Docker Engine install command via the package manager to pull docker-ce and its dependencies from the newly added repository.

```
QEMU (CodyRichard-Rocky) - noVNC - Chromium
itlab.fsmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
In order to upgrade the database, you must run SSSD.
Removing cache files in /var/lib/sss/db should fix the issue, but note that removing cache files will also remove all of your cached credentials.
Could not open available domains
[sss_cache] [sysdb_domain_cache_connect] (0x0010): DB version too old [0.23], expected [0.24] for domain implicit_files!
Higher version of database is expected!
In order to upgrade the database, you must run SSSD.
Removing cache files in /var/lib/sss/db should fix the issue, but note that removing cache files will also remove all of your cached credentials.
Could not open available domains

Installing      : libcgroupp-0.41-19.e18.x86_64                                7/14
Running scriptlet: libcgroupp-0.41-19.e18.x86_64                            7/14
Installing      : fuse-common-3.3.0-19.e18.x86_64                          8/14
Installing      : fuse3-3.3.0-19.e18.x86_64                               9/14
Installing      : fuse-overlappfs-1.13-1.module+e18.10.0+1880+0e896d1b.x86_64 10/14
Running scriptlet: fuse-overlappfs-1.13-1.module+e18.10.0+1880+0e896d1b.x86_64 10/14
[ 577.279199] fuse: init (API version 7.34)
Installing      : libslirp-4.4.0-2.module+e18.10.0+1880+0e896d1b.x86_64      11/14
Installing      : slirp4netns-1.2.3-1.module+e18.10.0+1880+0e896d1b.x86_64  12/14
Installing      : docker-ce-rootless-extras-26.1.3-1.e18.x86_64            13/14
Running scriptlet: docker-ce-rootless-extras-26.1.3-1.e18.x86_64           13/14
Installing      : docker-ce-3:26.1.3-1.e18.x86_64                          14/14
Running scriptlet: docker-ce-3:26.1.3-1.e18.x86_64                          14/14
Running scriptlet: container-selinux-2:2.229.0-2.module+e18.10.0+1880+0e896d1b.noarch 14/14
Running scriptlet: docker-ce-3:26.1.3-1.e18.x86_64                          14/14
Verifying       : container-selinux-2:2.229.0-2.module+e18.10.0+1880+0e896d1b.noarch 1/14
Verifying       : fuse-overlappfs-1.13-1.module+e18.10.0+1880+0e896d1b.x86_64  2/14
Verifying       : libslirp-4.4.0-2.module+e18.10.0+1880+0e896d1b.x86_64      3/14
Verifying       : slirp4netns-1.2.3-1.module+e18.10.0+1880+0e896d1b.x86_64  4/14
Verifying       : fuse-common-3.3.0-19.e18.x86_64                             5/14
Verifying       : fuse3-3.3.0-19.e18.x86_64                                   6/14
Verifying       : fuse3-libs-3.3.0-19.e18.x86_64                              7/14
Verifying       : libcgroupp-0.41-19.e18.x86_64                               8/14
Verifying       : containerd.io-1.6.32-3.1.e18.x86_64                         9/14
Verifying       : docker-buildx-plugin-0.14.0-1.e18.x86_64                    10/14
Verifying       : docker-ce-3:26.1.3-1.e18.x86_64                            11/14
Verifying       : docker-ce-cli-1:26.1.3-1.e18.x86_64                         12/14
Verifying       : docker-ce-rootless-extras-26.1.3-1.e18.x86_64              13/14
Verifying       : docker-compose-plugin-2.27.0-1.e18.x86_64                  14/14

Installed:
container-selinux-2:2.229.0-2.module+e18.10.0+1880+0e896d1b.noarch          containerd.io-1.6.32-3.1.e18.x86_64
docker-buildx-plugin-0.14.0-1.e18.x86_64                                     docker-ce-3:26.1.3-1.e18.x86_64
docker-ce-cli-1:26.1.3-1.e18.x86_64                                       docker-ce-rootless-extras-26.1.3-1.e18.x86_64
docker-compose-plugin-2.27.0-1.e18.x86_64                                   fuse-common-3.3.0-19.e18.x86_64
fuse-overlappfs-1.13-1.module+e18.10.0+1880+0e896d1b.x86_64               fuse3-3.3.0-19.e18.x86_64
fuse3-libs-3.3.0-19.e18.x86_64                                              libcgroupp-0.41-19.e18.x86_64
libslirp-4.4.0-2.module+e18.10.0+1880+0e896d1b.x86_64                     slirp4netns-1.2.3-1.module+e18.10.0+1880+0e896d1b.x86_64

Complete!
root@CodyRichard-Rocky ~]#
```

FIG 05 — Tail of the Docker install output confirming the package installation completed successfully on the Rocky Linux host.

SERVICE

Enabling the Docker Service

With the engine installed, I enable the systemd unit so containers persist across reboots, then confirm the daemon is active.

```
QEMU (CodyRichard-Rocky) - noVNC - Chromium
itlab.fsmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
root@CodyRichard-Rocky ~]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service + /usr/lib/systemd/system/docker.service.
root@CodyRichard-Rocky ~]#
```

FIG 06 — Enabling the service with systemctl enable docker, which creates the multi-user.target symlink to /usr/lib/systemd/system/docker.service.

```
QEMU (CodyRichard-Rocky) - noVNC - Chromium
itlab.fsemmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
root@CodyRichard-Rocky ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2024-11-17 18:35:49 EST; 7min ago
     Docs: https://docs.docker.com
   Main PID: 1179 (dockerd)
    Tasks: 10
   Memory: 143.9M
   CGroup: /system.slice/docker.service
           └─1179 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 17 18:35:47 CodyRichard-Rocky.localdomain systemd[1]: Starting Docker Application Container Engine...
Nov 17 18:35:47 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:47.718764934-05:00" level=info msg="Starting up"
Nov 17 18:35:47 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:47.899295466-05:00" level=info msg="Loading containers: start."
Nov 17 18:35:49 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:49.165493118-05:00" level=info msg="Firewalld: interface docker0 already pa
Nov 17 18:35:49 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:49.333168364-05:00" level=info msg="Loading containers: done."
Nov 17 18:35:49 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:49.367014429-05:00" level=info msg="Docker daemon" commit=8e96db1 container
Nov 17 18:35:49 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:49.367279862-05:00" level=info msg="Daemon has completed initialization"
Nov 17 18:35:49 CodyRichard-Rocky.localdomain dockerd[1179]: time="2024-11-17T18:35:49.446646670-05:00" level=info msg="API listen on /run/docker.sock"
Nov 17 18:35:49 CodyRichard-Rocky.localdomain systemd[1]: Started Docker Application Container Engine.
root@CodyRichard-Rocky ~]# docker -v
Docker version 26.1.3, build b72abbb
root@CodyRichard-Rocky ~]# date
Sun Nov 17 18:43:18 EST 2024
root@CodyRichard-Rocky ~]# _
```

FIG 07 — Docker confirmed active, with the version output and date command verifying the running daemon on the host.

DEPLOYMENT

Preparing the Compose Stack

I install wget, then author the docker-compose.yml that defines the WordPress and MySQL 8.0 services.

```

itlab.fsmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
root@CodyRichard-Rocky ~# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.8G  0  1.8G   0% /dev
tmpfs           1.8G  0  1.8G   0% /dev/shm
tmpfs           1.8G  0  1.8G   0% /dev/pts
tmpfs           1.8G  0  1.8G   0% /sys/fs/cgroup
/dev/mapper/rl-root  8.0G  3.3G  4.8G  41% /
/dev/sdb1       2.0G  47M  2.0G   3% /codyrichard-data
/dev/sda1      1014M  324M  691M  32% /boot
tmpfs          367M  0  367M   0% /run/user/0
root@CodyRichard-Rocky ~# cd /codyrichard-data
root@CodyRichard-Rocky codyrichard-data# mkdir wordpress
root@CodyRichard-Rocky codyrichard-data# cd wordpress
root@CodyRichard-Rocky wordpress# yum install wget -y
Last metadata expiration check: 0:16:18 ago on Sun 17 Nov 2024 06:29:42 PM EST.
Dependencies resolved.
=====
Package                Architecture      Version           Size
=====
Installing:
wget                   x86_64           1.19.5-12.e18_10 733 k
Installing dependencies:
libmetalink            x86_64           0.1.3-7.e18      31 k
=====
Transaction Summary
=====
Install 2 Packages

Total download size: 764 k
Installed size: 2.8 M
Downloading Packages:
(1/2): wget-1.19.5-12.e18_10.x86_64.rpm          1.4 MB/s | 733 kB  00:00
(2/2): libmetalink-0.1.3-7.e18.x86_64.rpm        3.4 kB/s | 31 kB  00:09
-----
Total
-----
79 kB/s | 764 kB  00:09
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Installing    : libmetalink-0.1.3-7.e18.x86_64 1/1
  Installing    : wget-1.19.5-12.e18_10.x86_64 1/2
  Running scriptlet: wget-1.19.5-12.e18_10.x86_64 2/2
  Verifying     : wget-1.19.5-12.e18_10.x86_64 2/2
  Verifying     : libmetalink-0.1.3-7.e18.x86_64 1/2
  Verifying     : libmetalink-0.1.3-7.e18.x86_64 2/2

Installed:
  libmetalink-0.1.3-7.e18.x86_64
  wget-1.19.5-12.e18_10.x86_64

Complete!
root@CodyRichard-Rocky wordpress#

```

FIG 08 — Installing wget on the Rocky Linux host in preparation for staging the Compose project files.

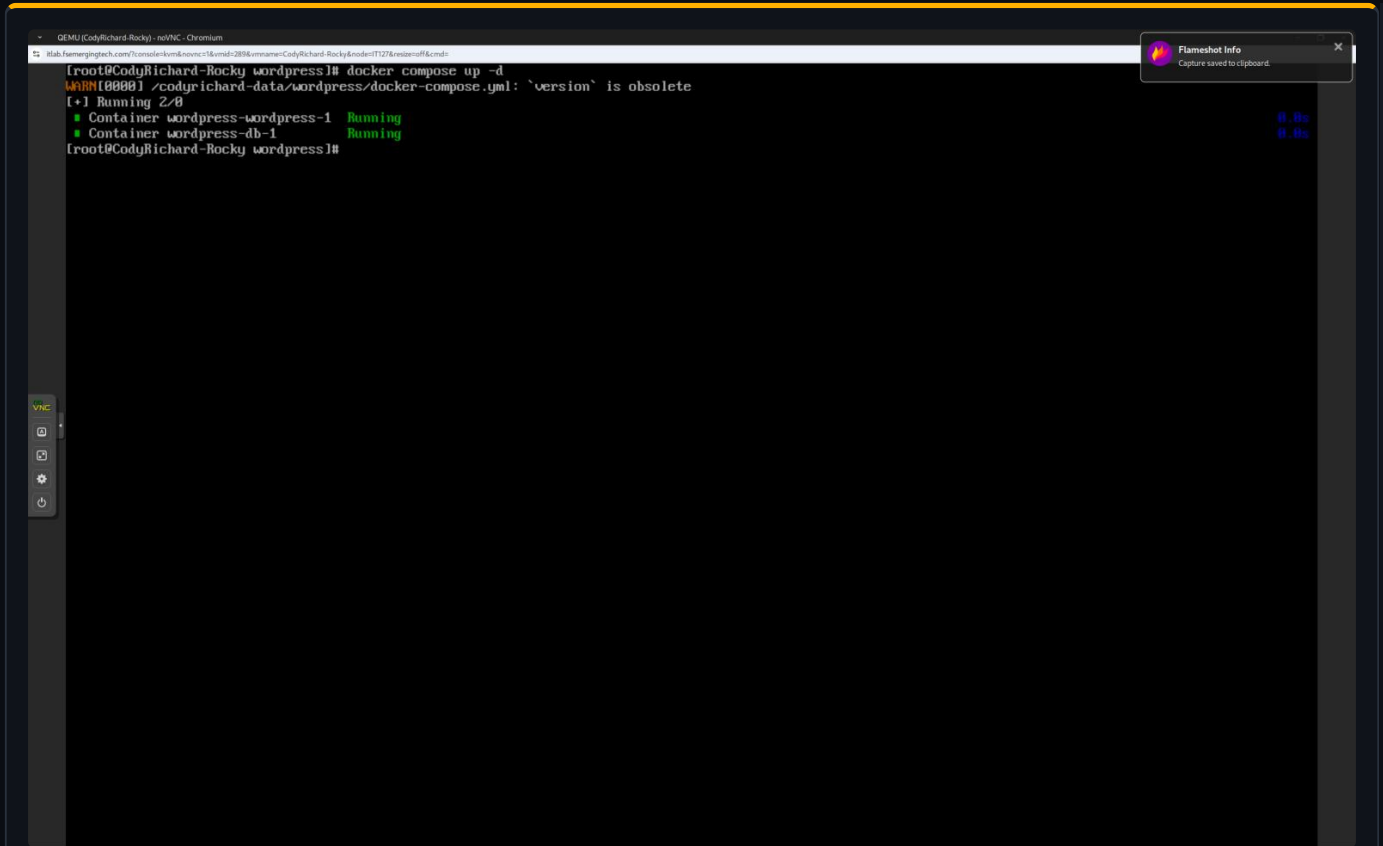


FIG 09 — The docker-compose.yml under /codyrichard-data/wordpress/ defining the WordPress and MySQL 8.0 services, deployed with docker compose up -d to bring up wordpress-wordpress-1 and wordpress-db-1.

VERIFICATION

Host State and Disk

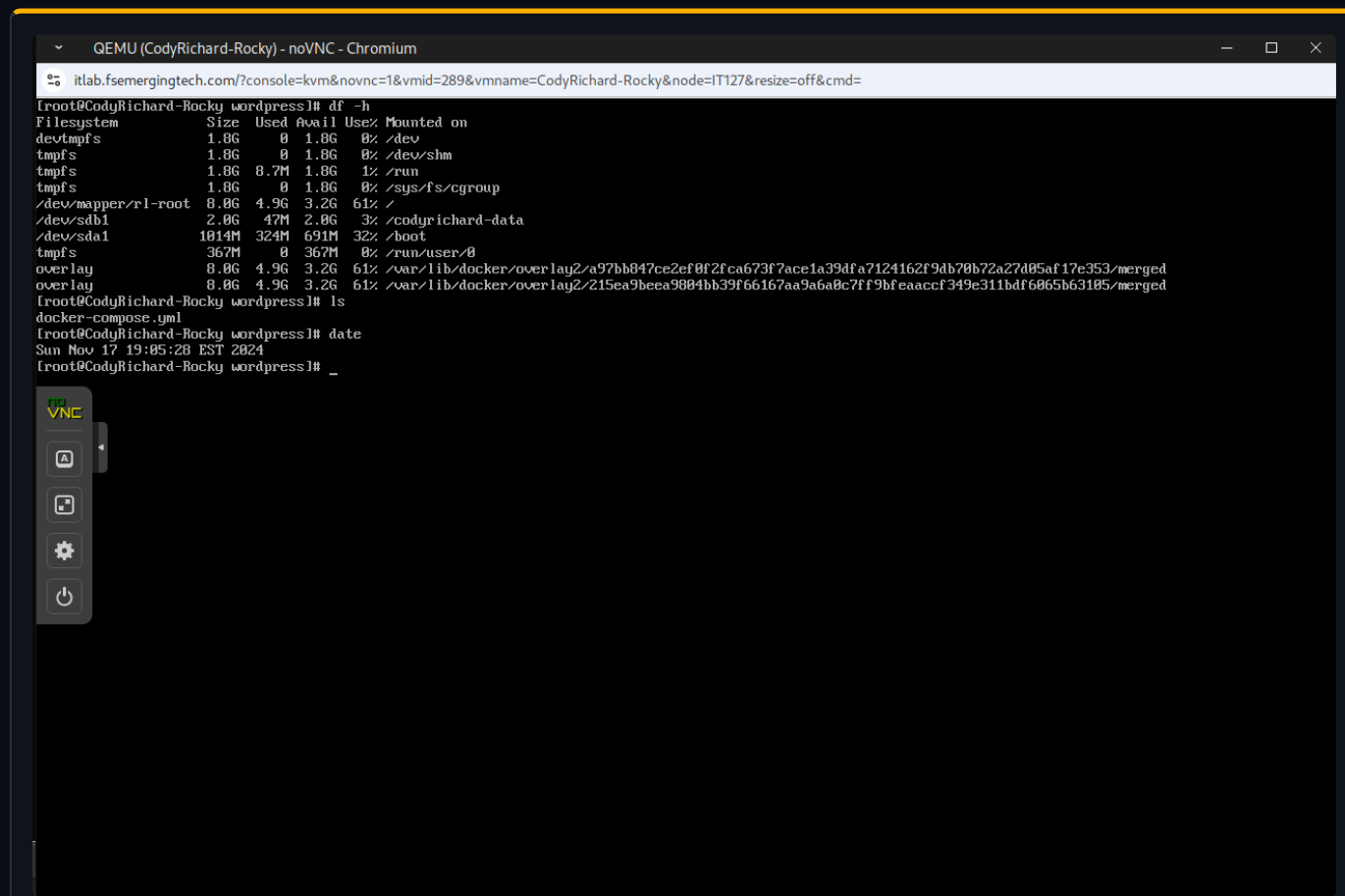


FIG 10 — Output of df -h, ls, and date, showing disk utilization and the working directory state on the host after deployment.

VERIFICATION

Containers and Networking

```

QEMU (CodyRichard-Rocky) - noVNC - Chromium
itlab.fsemmergingtech.com/?console=kvm&novnc=1&vmid=289&vmname=CodyRichard-Rocky&node=IT127&resize=off&cmd=
root@CodyRichard-Rocky wordpress1# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
ff05ceead28   wordpress    "docker-entrypoint.s..." 9 minutes ago Up 8 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp   wordpress-1
a0f57152c8e5   mysql:8.0    "docker-entrypoint.s..." 9 minutes ago Up 8 minutes    3306/tcp, 33060/tcp   wordpress-db-1
root@CodyRichard-Rocky wordpress1# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether bc:24:11:06:22:e8 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.1.102.52/22 brd 10.1.103.255 scope global dynamic noprefixroute ens18
        valid_lft 3895sec preferred_lft 3895sec
    inet6 fe80::be24:11ff:fe06:22e0/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
vnc docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:1a:a7:8a:3f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet4 4f7bb5c55: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:b4:14:86 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-4dd4f7bb5c55
        valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:feb4:1486/64 scope link
        valid_lft forever preferred_lft forever
    inet4 10d7fce80if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-4dd4f7bb5c55 state UP group default
    link/ether ea:15:a0:00:ab:b6 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::e815:a0ff:fe00:abb6/64 scope link
        valid_lft forever preferred_lft forever
16: veth46e9f650if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-4dd4f7bb5c55 state UP group default
    link/ether 0e:88:4e:23:59:a5 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::c08:4eff:fe23:59a5/64 scope link
        valid_lft forever preferred_lft forever
root@CodyRichard-Rocky wordpress1# date
Sun Nov 17 19:06:56 EST 2024
root@CodyRichard-Rocky wordpress1# _

```

FIG 11 — docker ps with ip addr and date output, confirming both containers Running, the port mapping 0.0.0.0:8080->80/tcp, MySQL on 3306/33060, and host ens18 at 10.1.102.52/22 alongside the docker0 bridge at 172.17.0.1/16.

VALIDATION

Reaching the WordPress Installer

From a separate Windows 10 Pro client, I hit the host:port URL to confirm the published container is reachable across the lab network.

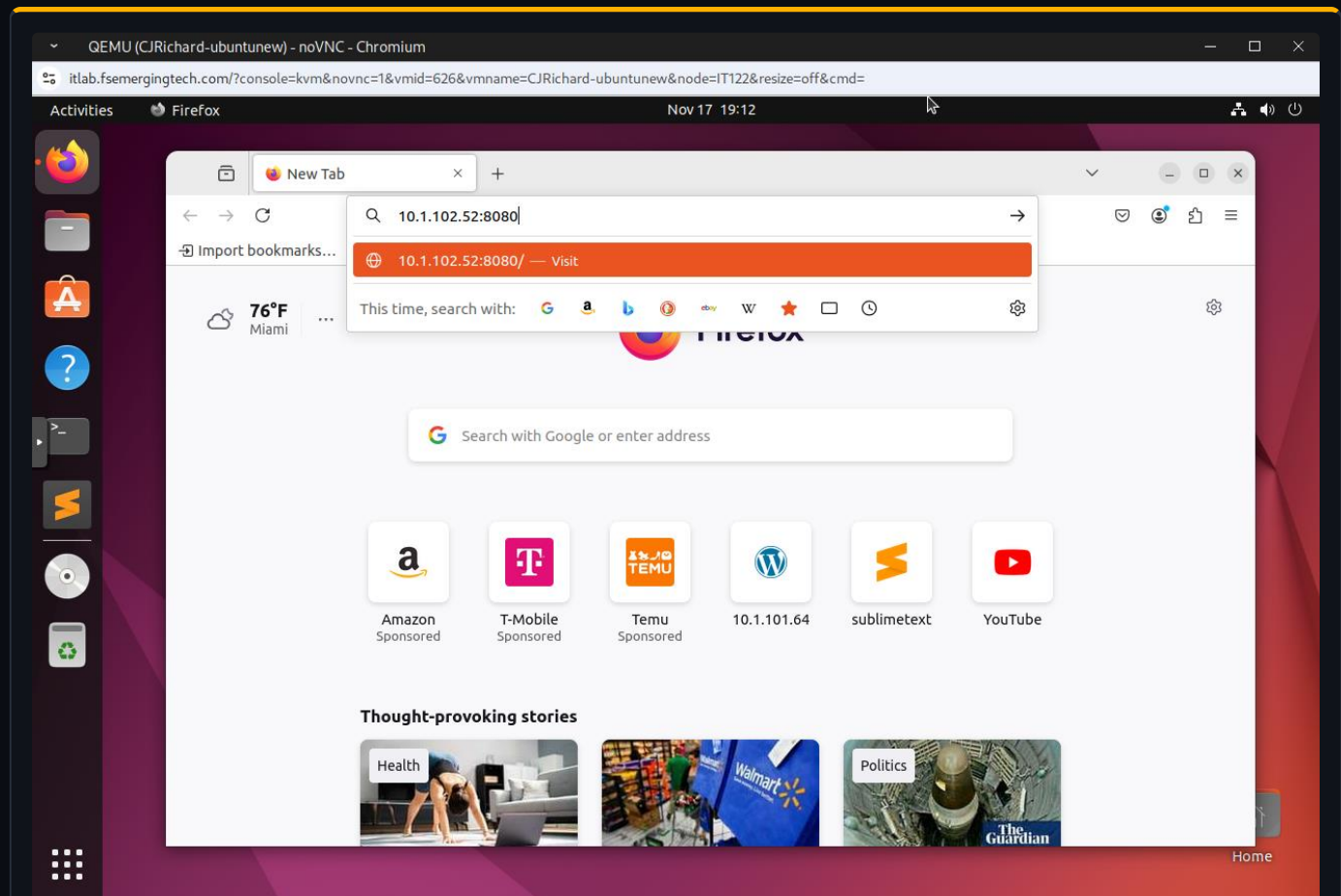


FIG 12 — Using the port mapping to load the WordPress installation page in a browser at <http://10.1.102.52:8080>, served by the containerized stack.

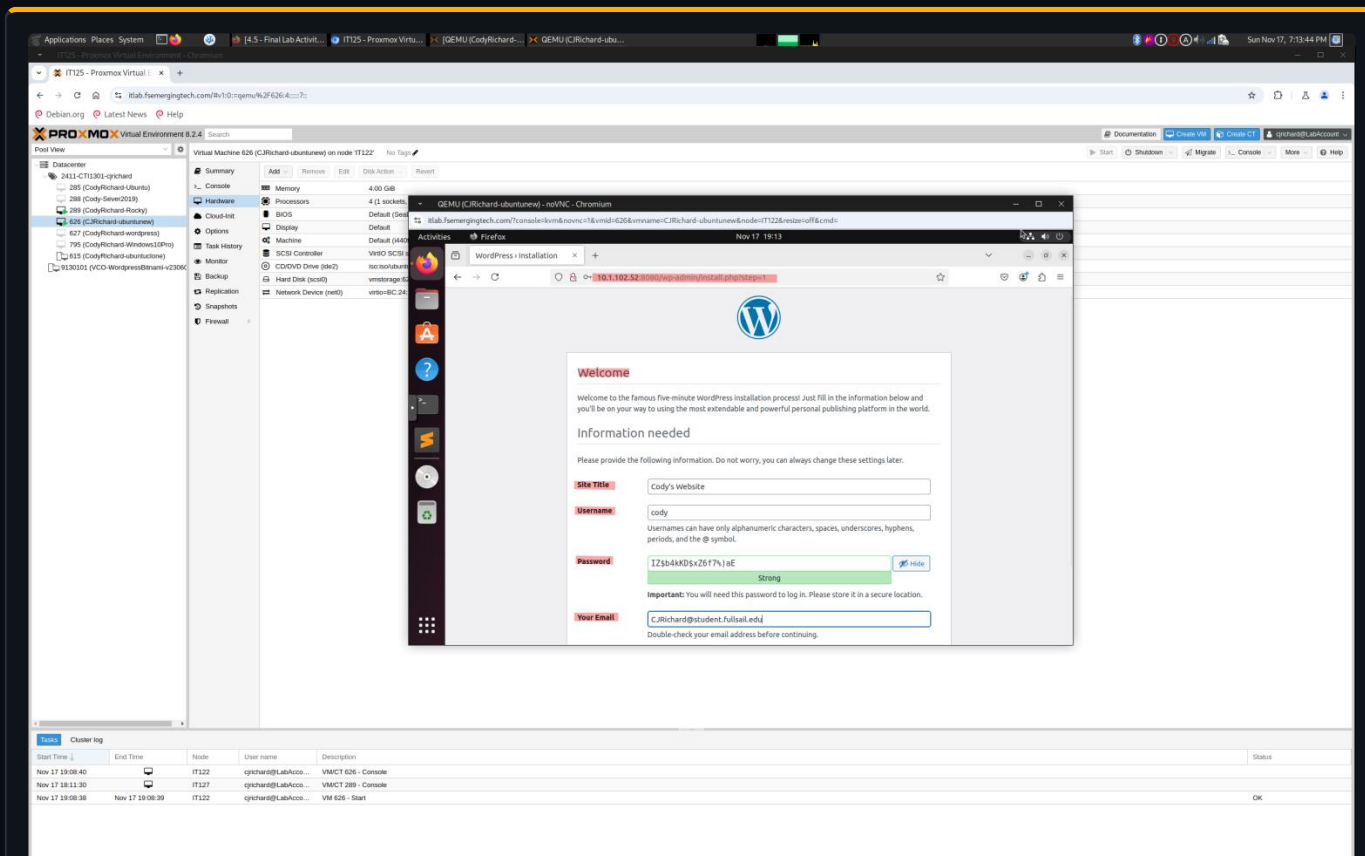


FIG 13 — The WordPress installation wizard collecting site title and admin credentials over the published port mapping.

VALIDATION

Installation Complete

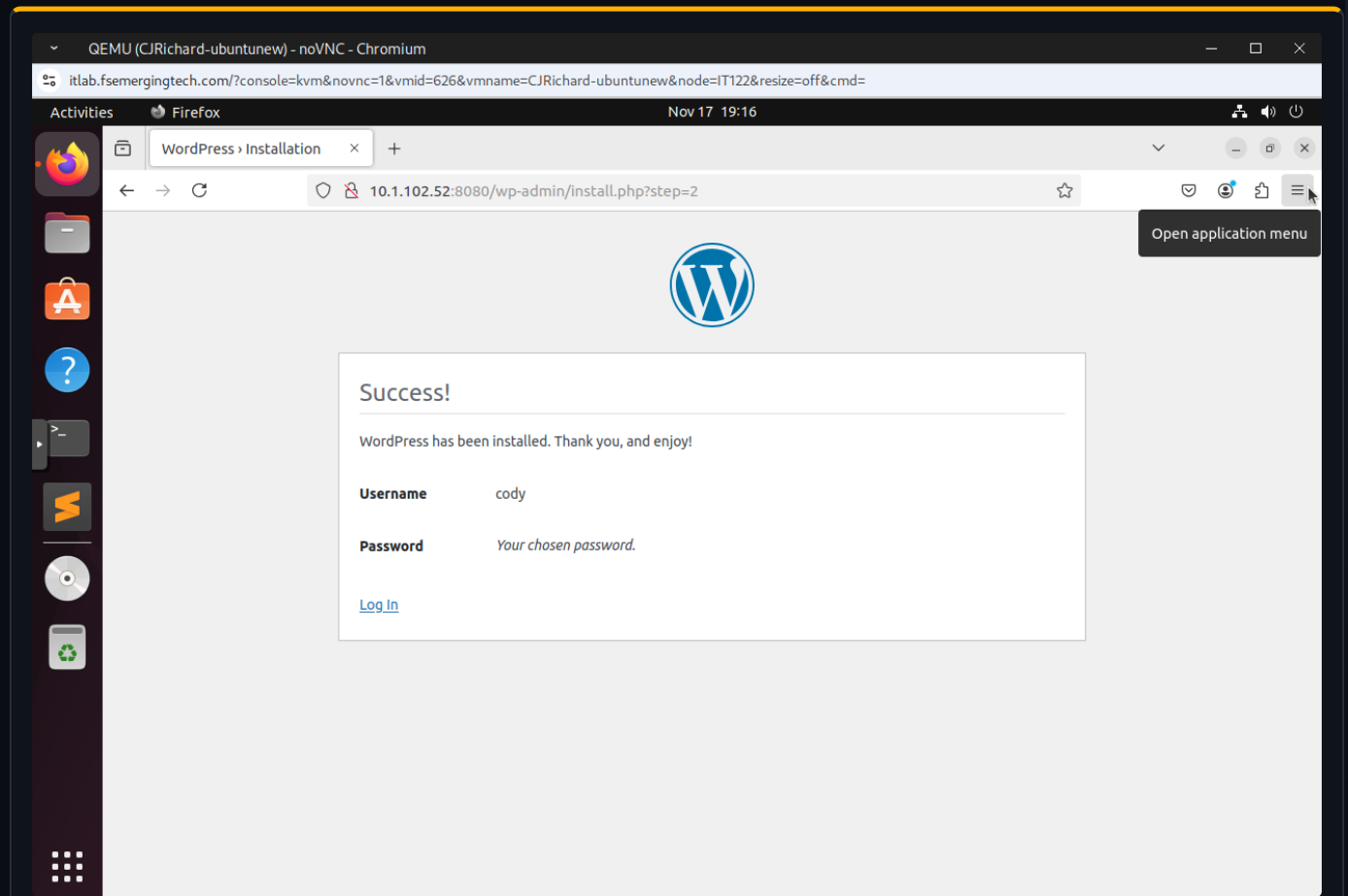


FIG 14 — WordPress reporting a successful installation, confirming the application and its MySQL 8.0 backend are wired correctly.

VALIDATION

Admin Portal

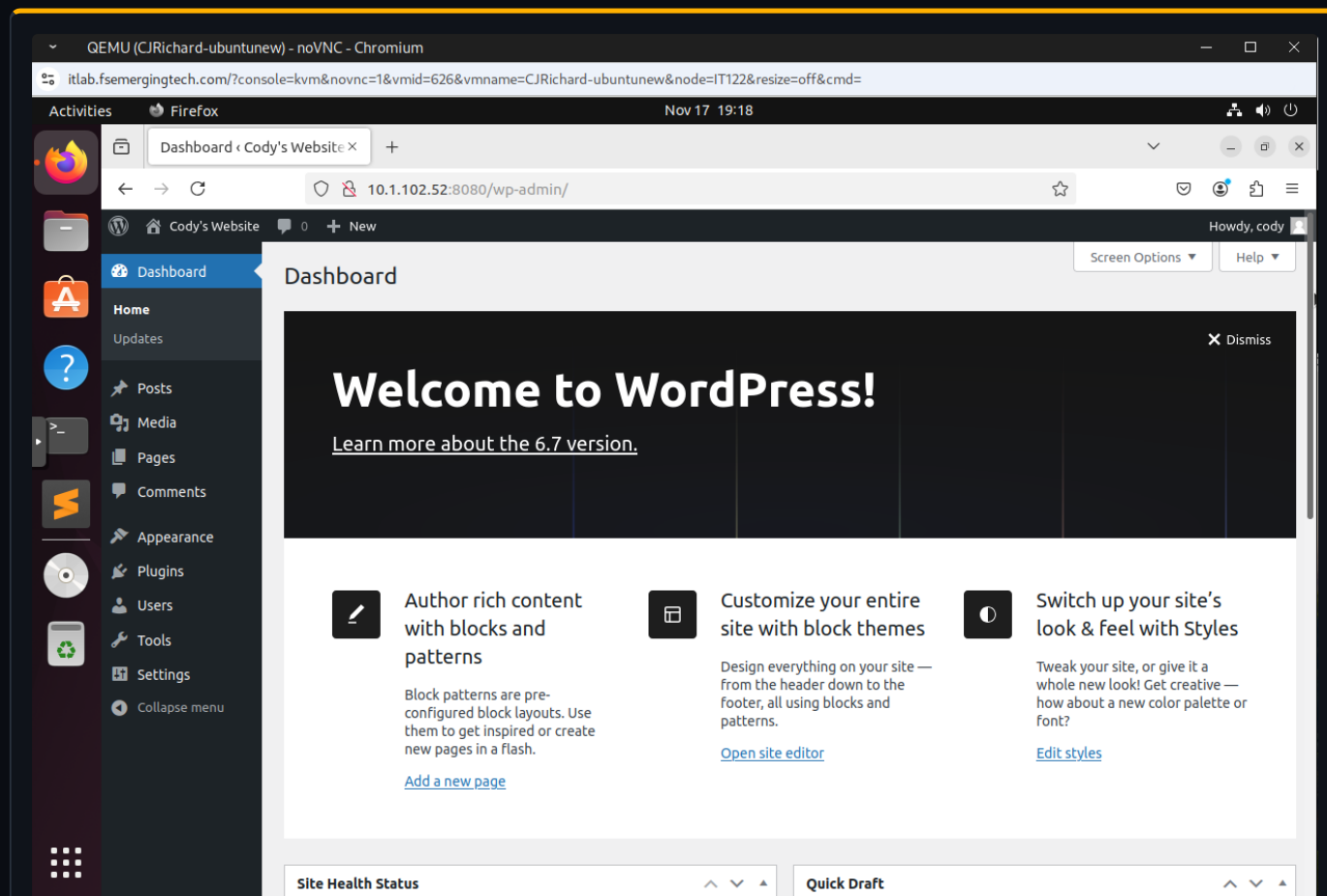


FIG 15 — The WordPress admin dashboard reached from the client, validating an authenticated session against the containerized site.

VALIDATION

Front-End Access

Final client-side checks confirm the public site loads through the same host:port mapping.

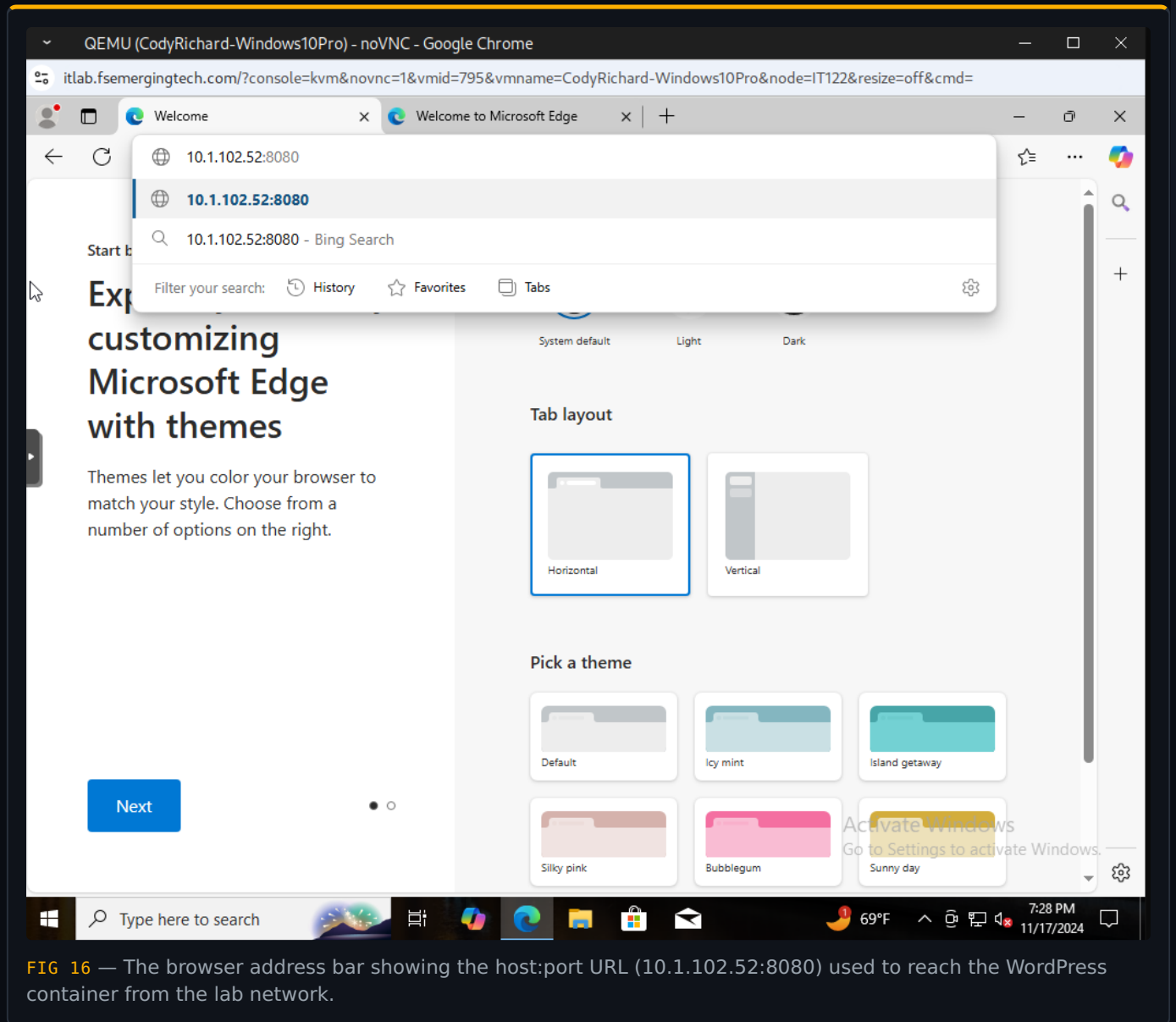


FIG 16 — The browser address bar showing the host:port URL (10.1.102.52:8080) used to reach the WordPress container from the lab network.

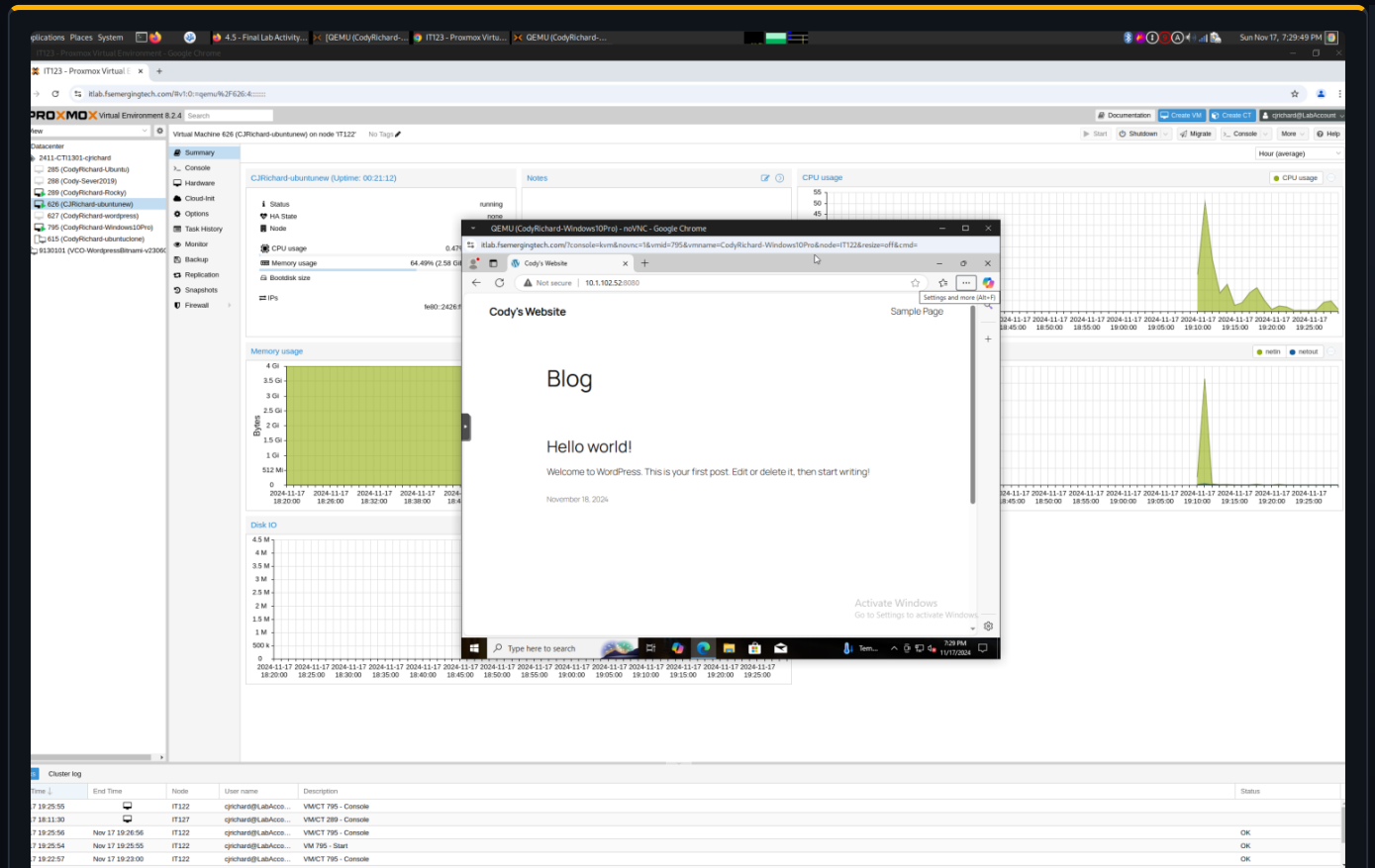


FIG 17 — The rendered WordPress front-end site loading successfully from the client, confirming end-to-end delivery from the published container.

SHUTDOWN

Clean Lab Teardown

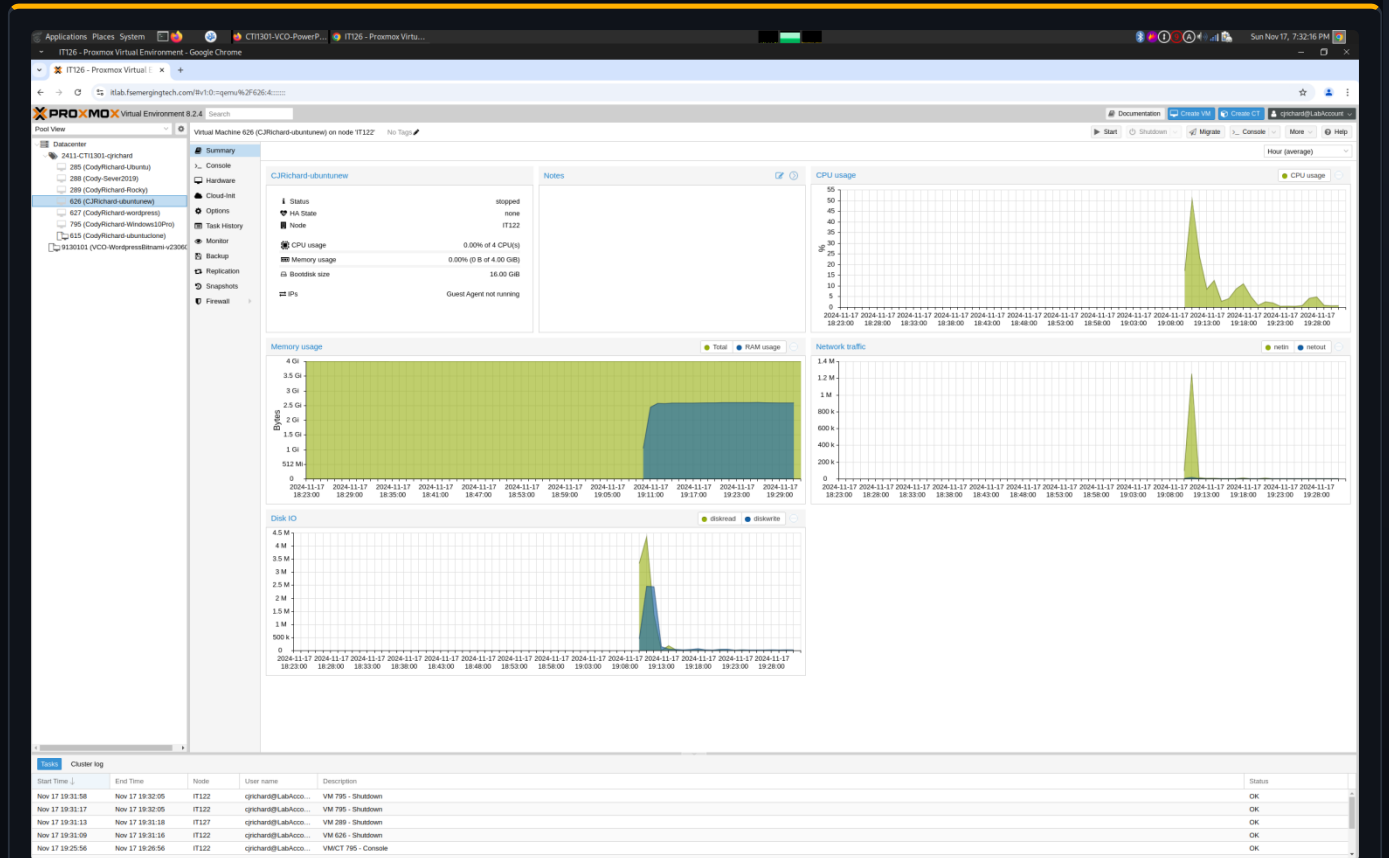


FIG 18 — Proxmox showing all lab VMs powered down, closing out the deployment with a clean shutdown.