DEVOPS TOOLING WEBSITE SOLUTION

In this project we will be implementing a solution that consists of the following components.

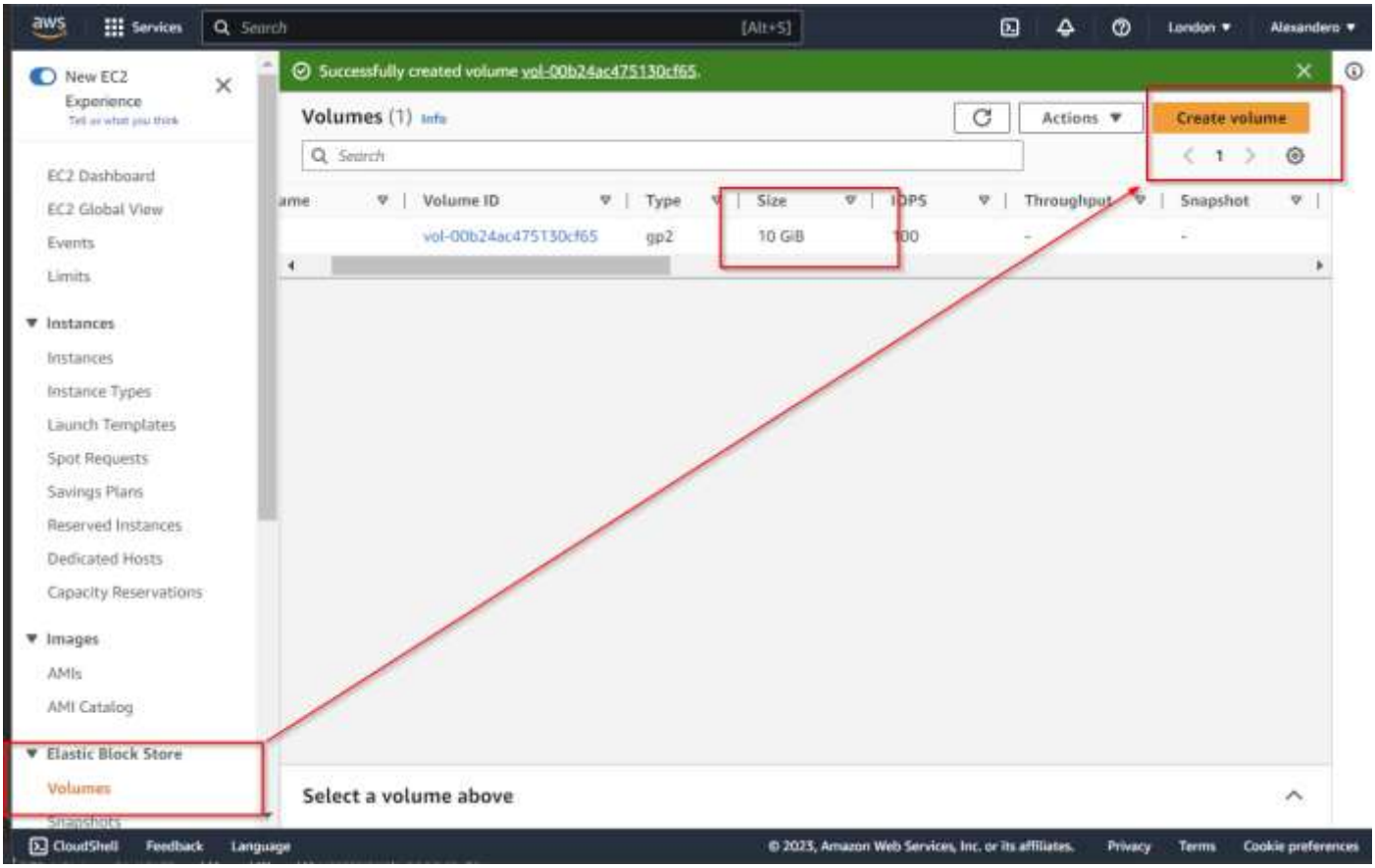
1. 1nfrastructure: **AWS**
2. Webserver Linux: **Red hat Enterprise Linux 8**
3. Database Server: **Ubuntu 20.04 + MySQL**
4. Storage Server: **Red Hat Enterprise Linux + NFS Server**
5. Programming Language: **PHP**
6. Code Repository: **GitHub**

3-Tier Web application architecture with a single database and an NFS server as a file storage

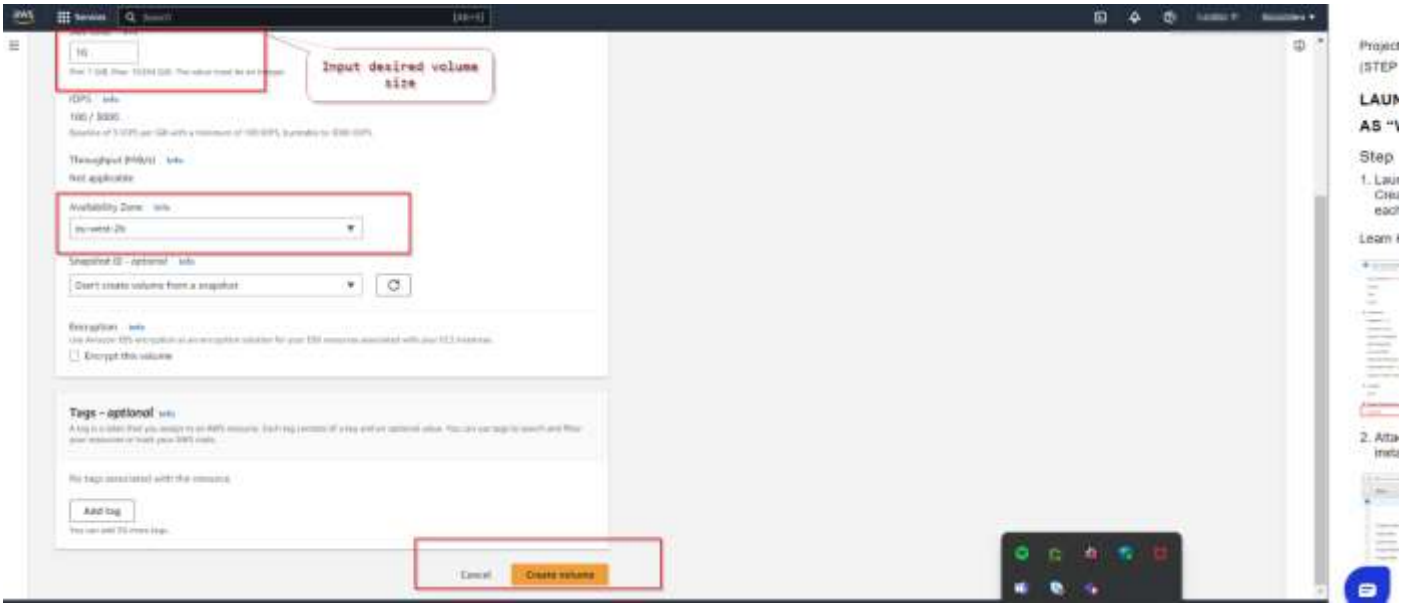
PREPARE THE NFS SERVER

Step 1.

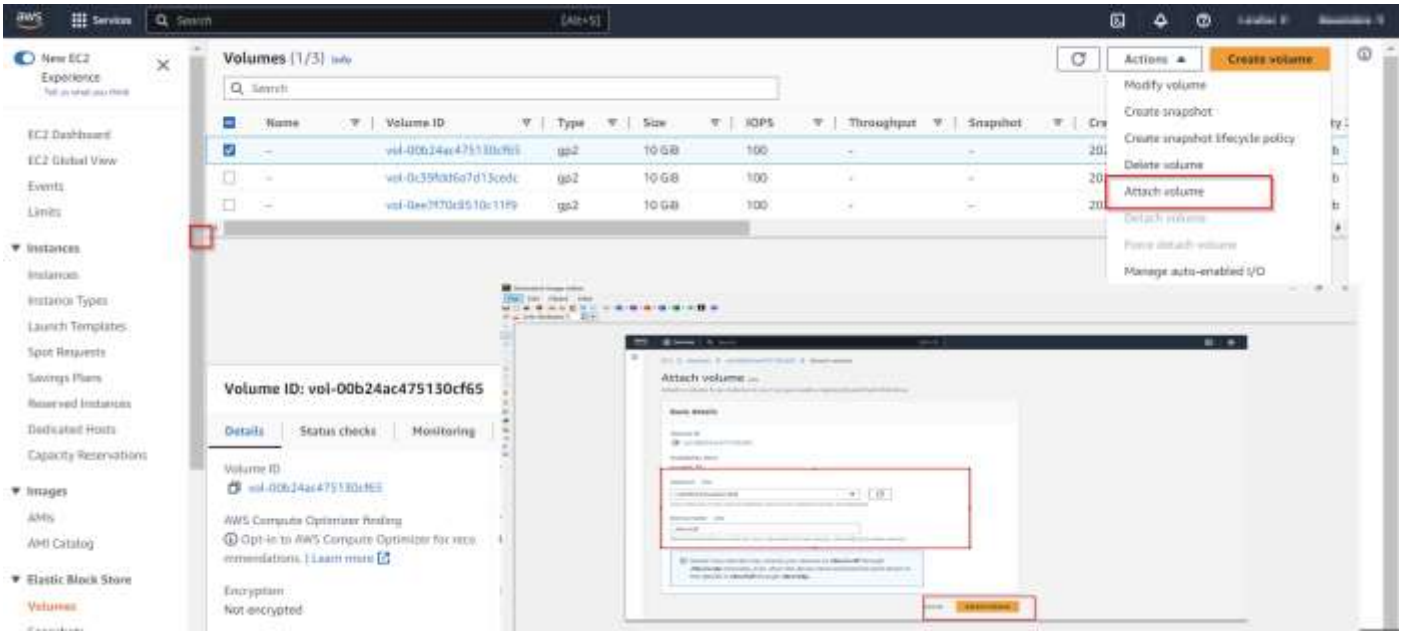
Create 3 volumes of 10GB size on AWS elastic block store (EBS), Lunch an EC2 instance on AWS that will serve as webserver and attach the 3 volumes to the instance.



Ensure the volumes are created on the same availability zone as the server instance.



Attach the volumes.



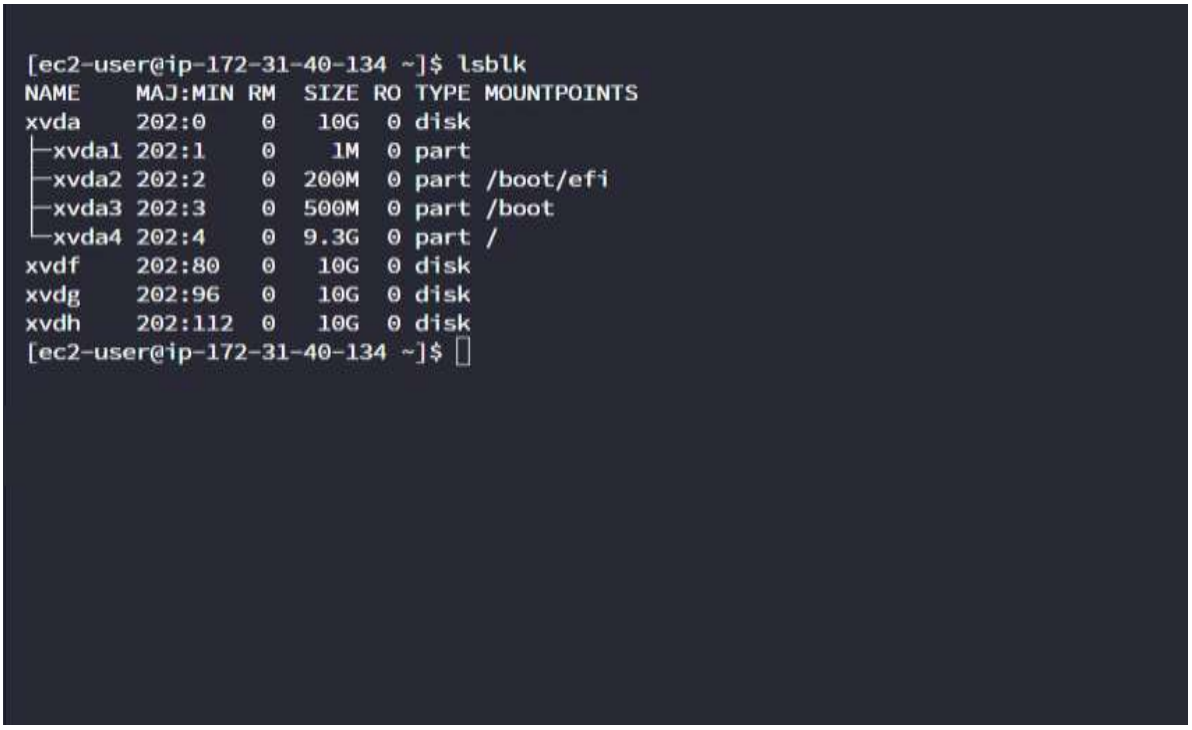
Step 2.

Connect the webserver instance to the terminal and begin configuration.

Firstly, check to confirm the names of the volume blocks that you have just attached to the webserver.

Use code.

$ lsblk



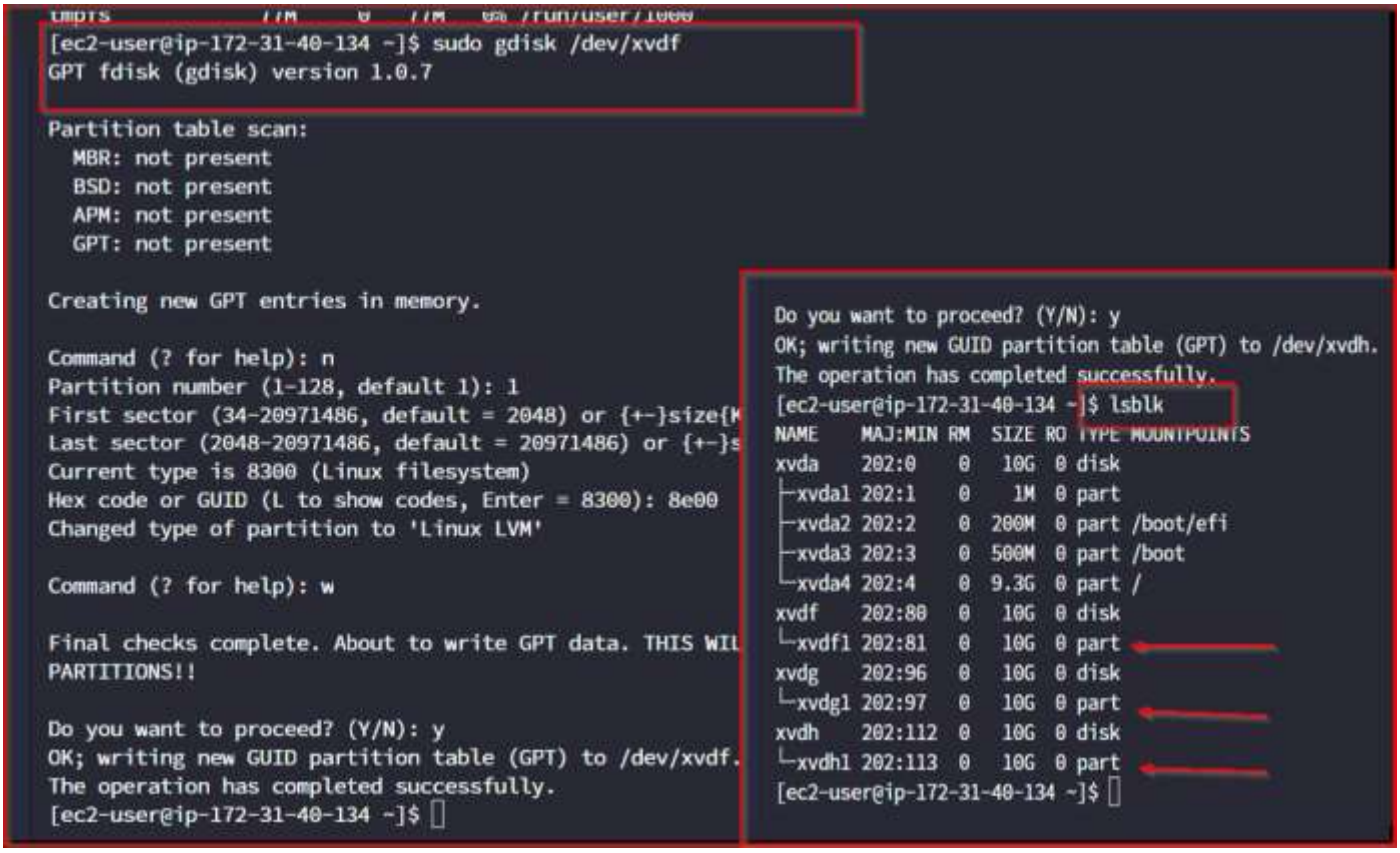
You would see names as xvdf, xvdg. xvdh. Note that these volume blocks (devices) reside in the /dev/ directory.

Step 3.

Create a single partition on each of the **3 disks** using the gdisk utility. Here you will require superuser privileges to use this utility.

$ sudo gdisk /dev/xvdf

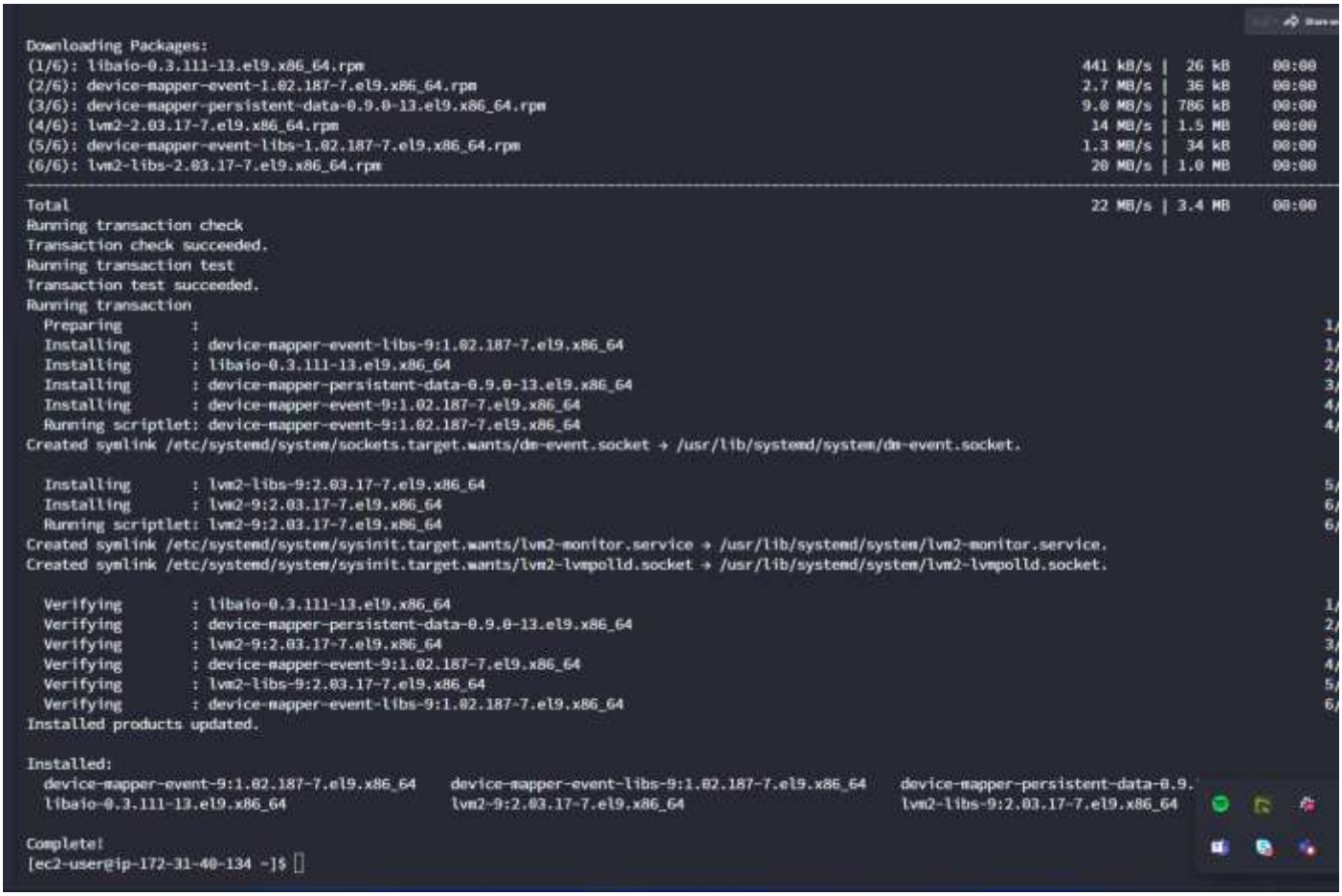
And use the **lsblk** command to view the newly configured partition on the 3 disks.



Step 4.

Install logical volume manager 2 (LVM2). This is a logical volume management system for Linux.

$ sudo yum install lvm2 -y



Make each of the 3 disks as physical volumes (PVs) to be used by the logical volume manager LVM. Use the pvcreate utility.

$sudo pvcreate /dev/xvdf1 /dev/xvdg1 /dev/xvdh1

Verify the physical volumes (PVs) created by using the **pvs** code.

```
[ec2-user@ip-172-31-40-134 ~]$ sudo pvs
  PV         VG Fmt  Attr PSize   PFree
  /dev/xvdf1    lvm2 ---  <10.00g <10.00g
  /dev/xvdg1    lvm2 ---  <10.00g <10.00g
  /dev/xvdh1    lvm2 ---  <10.00g <10.00g
[ec2-user@ip-172-31-40-134 ~]$
```

Create a volume group (VG) with vgcreate utility and name the volume group webdata-vg. This is needed in order to add the 3 physical volumes (PV) to the VG.

$ sudo vgcreate webdata-vg /dev/xvdf1 /dev/xvdg1 /dev/xvdh1

Verify the volumes group (VG) created by running the code **sudo vgs.**

```
[ec2-user@ip-172-31-40-134 ~]$ sudo vgcreate webdata-vg /dev/xvdf1 /dev/xvdg1 /dev/xvdh1
  Volume group "webdata-vg" successfully created
[ec2-user@ip-172-31-40-134 ~]$ sudo vgs
  VG         #PV #LV #SN Attr   VSize   VFree
  webdata-vg   3   0   0 wz--n- <29.99g <29.99g
[ec2-user@ip-172-31-40-134 ~]$
```

Create logical volumes

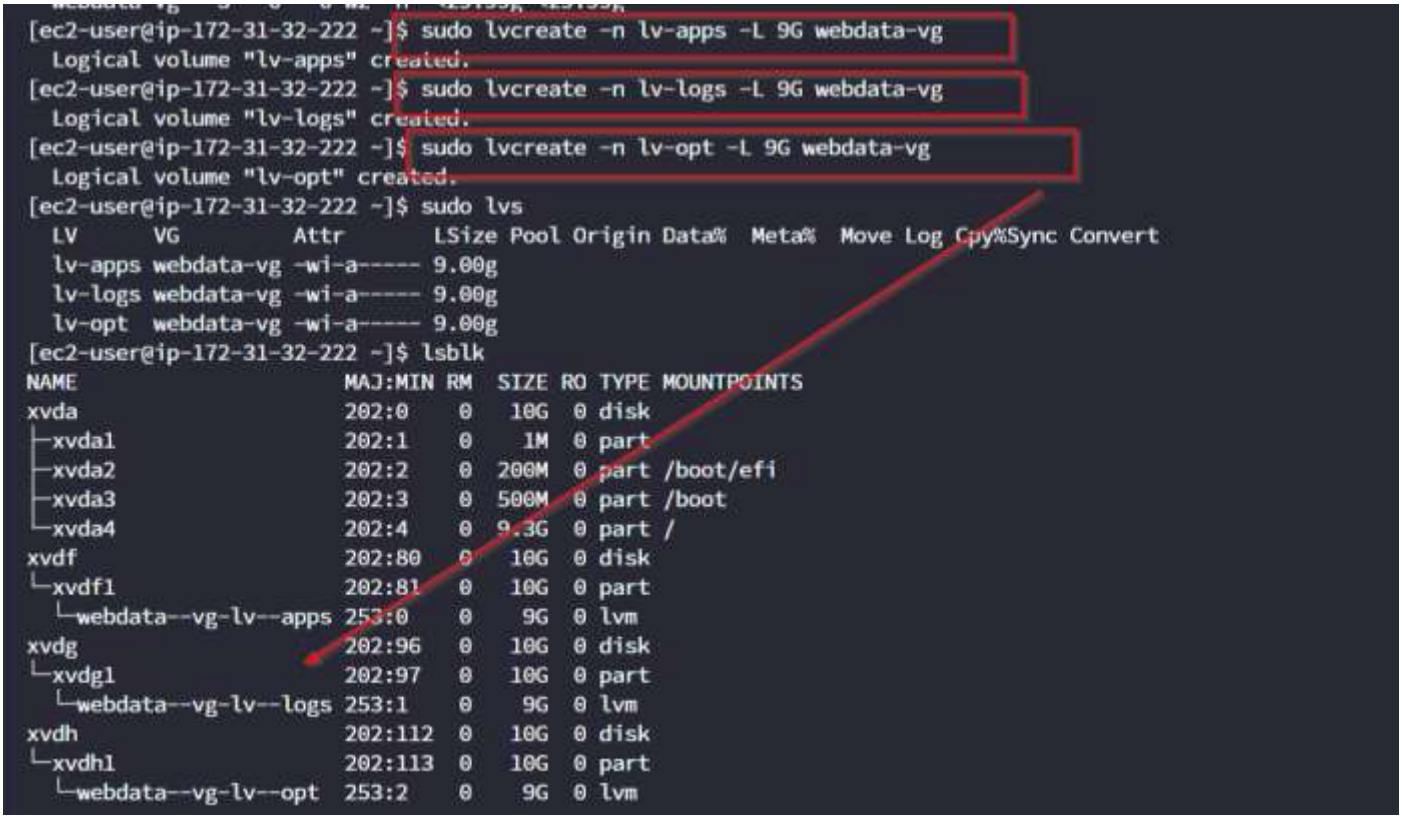
Now we will go a step further to create 3 logical volumes named **lv-apps, lv-logs and lv-opt.** In this case we would allocate the total volume group size to all logical volumes equally.

$ sudo lvcreate -n lv-apps -L 9G webdata-vg

$ sudo lvcreate -n lv-logs -L 9G webdata-vg

$ sudo lvcreate -n lv-opt -L 9G webdata-vg

Verify the logical volume groups (LV) created by running the code **sudo lvs** and check mount points.

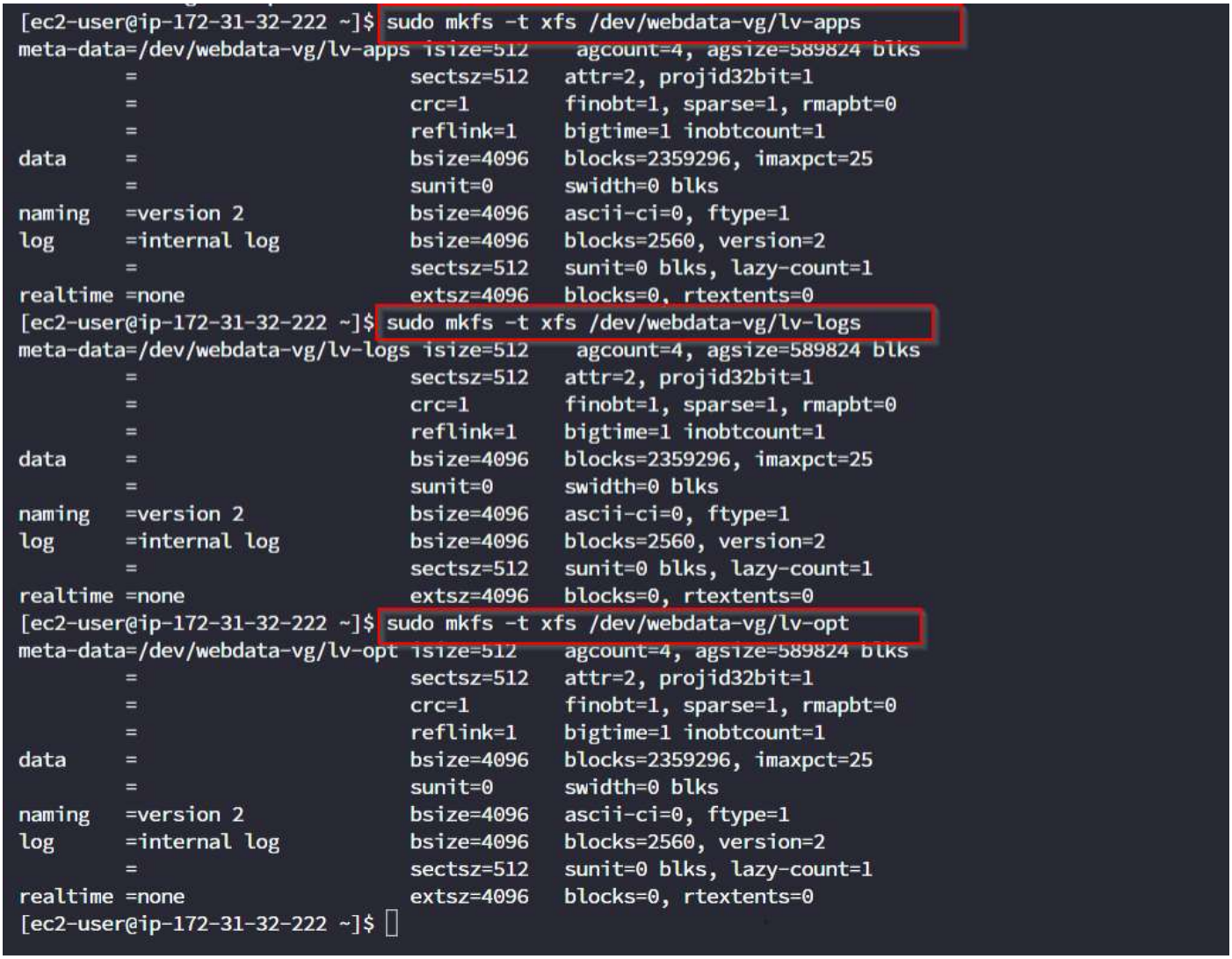


Format the disk (logical volumes) as xfs file system

$ sudo mkfs -t xfs /dev/webdata-vg/lv-apps

$ sudo mkfs -t xfs /dev/webdata-vg/lv-logs

$ sudo mkfs -t xfs /dev/webdata-vg/lv-opt



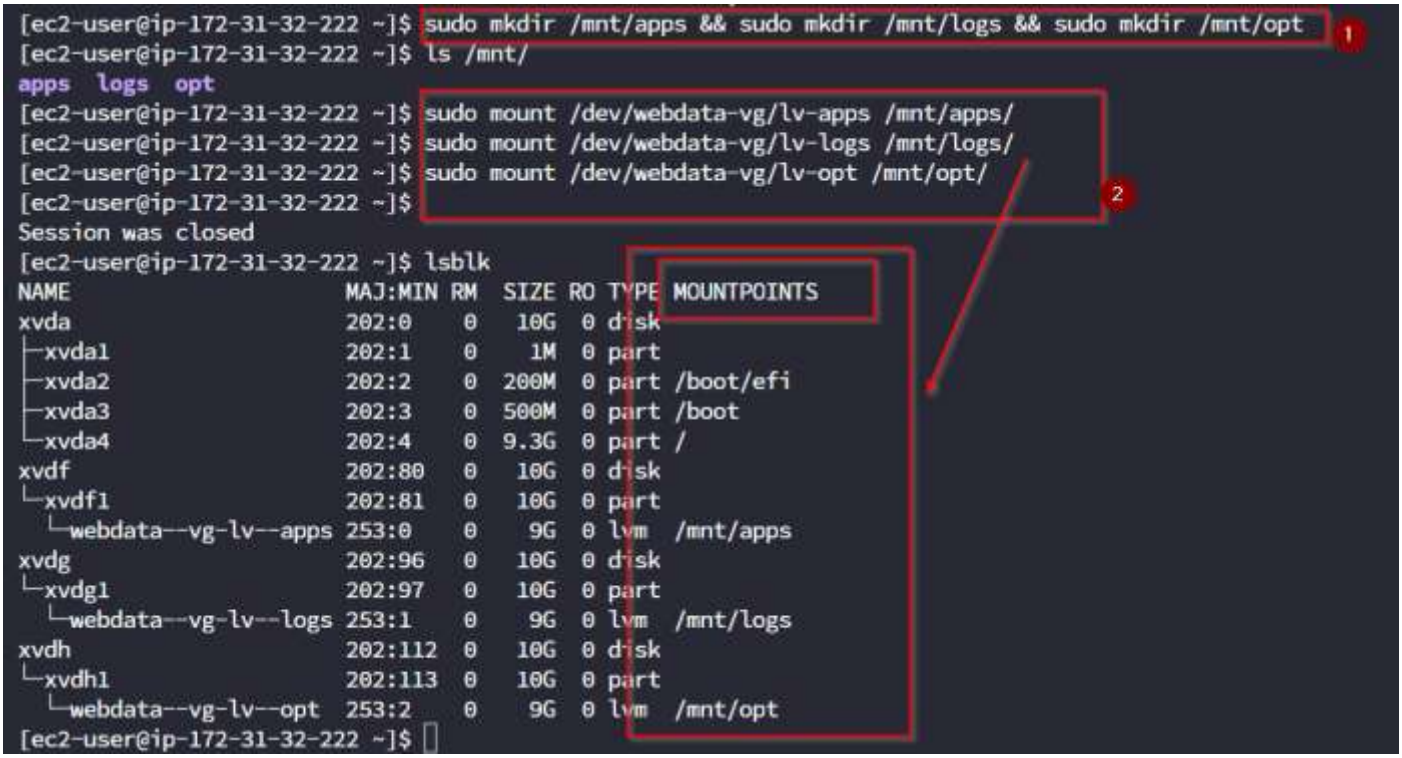
Create 3 mount points **/mnt/apps, /mnt/logs, /mnt/opt**

Mount the logical volumes to the respective mount points.

We will mount **lv-apps** on **/mnt/apps** (to be used by the webservers)

We will mount **lv-logs** on **/mnt/logs** (to be used by the webserver logs)

We will mount **lv-opt** on **/mnt/opt** (to be used by Jenkins server)



INSTALL NFS SERVER AND CONFIGURE IT TO START ON REBOOT AND MAKE SURE IT IS UP AND RUNNING

$ sudo yum update -y

$ sudo yum install nfs-utils -y

$ sudo systemctl start nfs server.service

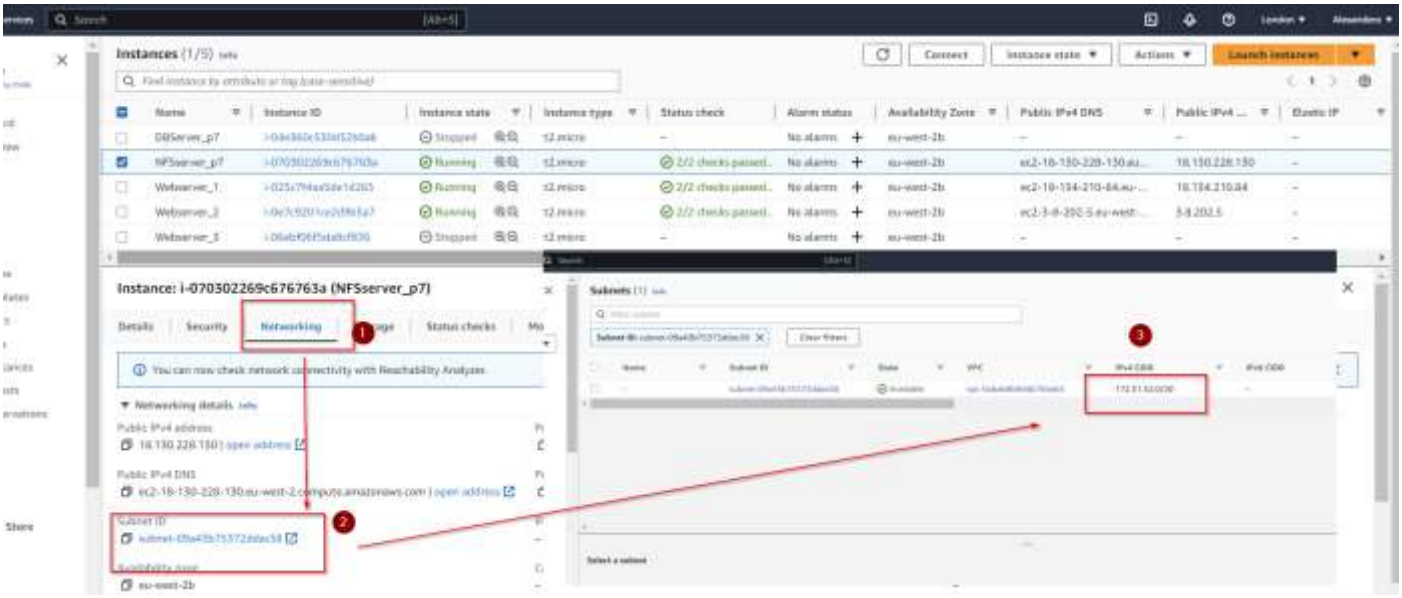
$ sudo systemctl enable nfs server.service

$ sudo systemctl status nfs server.service

```
  Installing       : sssd-nfs-idmap-2.8.2-2.el9.x86_64
  Running scriptlet: sssd-nfs-idmap-2.8.2-2.el9.x86_64
  Verifying        : quota-nls-1:4.06-6.el9.noarch
  Verifying        : libverto-libev-0.3.2-3.el9.x86_64
  Verifying        : libev-4.33-5.el9.x86_64
  Verifying        : quota-1:4.06-6.el9.x86_64
  Verifying        : rpcbind-1.2.6-5.el9.x86_64
  Verifying        : nfs-utils-1:2.5.4-18.el9.x86_64
  Verifying        : libtirpc-1.3.3-1.el9.x86_64
  Verifying        : sssd-nfs-idmap-2.8.2-2.el9.x86_64
  Verifying        : keyutils-1.6.3-1.el9.x86_64
  Verifying        : libnfsidmap-1:2.5.4-18.el9.x86_64
  Verifying        : gssproxy-0.8.4-5.el9_2.x86_64
Installed products updated.

Installed:
  gssproxy-0.8.4-5.el9_2.x86_64       keyutils-1.6.3-1.el9.x86_64       libev-4.33-5.el9.x86_64     libnfsidmap-1:2.5.4-
  libverto-libev-0.3.2-3.el9.x86_64   nfs-utils-1:2.5.4-18.el9.x86_64   quota-1:4.06-6.el9.x86_64   quota-nls-1:4.06-6.e
  sssd-nfs-idmap-2.8.2-2.el9.x86_64

Complete!
[ec2-user@ip-172-31-32-222 ~]$ sudo systemctl start nfs-server.service
[ec2-user@ip-172-31-32-222 ~]$ sudo systemctl enable nfs-server.service
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.serv
[ec2-user@ip-172-31-32-222 ~]$ sudo systemctl status nfs-server.service
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; preset: disabled)
     Active: active (exited) since Fri 2023-06-16 10:28:03 UTC; 28s ago
   Main PID: 13857 (code=exited, status=0/SUCCESS)
        CPU: 30ms

Jun 16 10:28:03 ip-172-31-32-222.eu-west-2.compute.internal systemd[1]: Starting NFS server and services...
Jun 16 10:28:03 ip-172-31-32-222.eu-west-2.compute.internal systemd[1]: Finished NFS server and services.
[ec2-user@ip-172-31-32-222 ~]$
```

Export mounts of webservers' **subnet cidr** to connect as clients.

See below diagram on how to locate your subnet cidr.



Now we must set permission that will allow our web servers to read and execute files on the NFS server.

$ sudo chown -R nobody: /mnt/apps

$ sudo chown -R nobody: /mnt/logs

$ sudo chown -R nobody: /mnt/opt

$ sudo chmod -R 777 /mnt/apps

$ sudo chmod -R 777 /mnt/logs

$ sudo chmod -R 777 /mnt/opt

Make sure to restart your NFS server after this permission setting.

$ sudo systemctl restart nfs-server.service

Configure access to NFS to allow clients within the same subnet CIDR (for example 172.31.32.0/20)

$ sudo vi /etc/exports

Paste the below: (note that you will have to input the subnet CIDR specific to your NFS server instance)

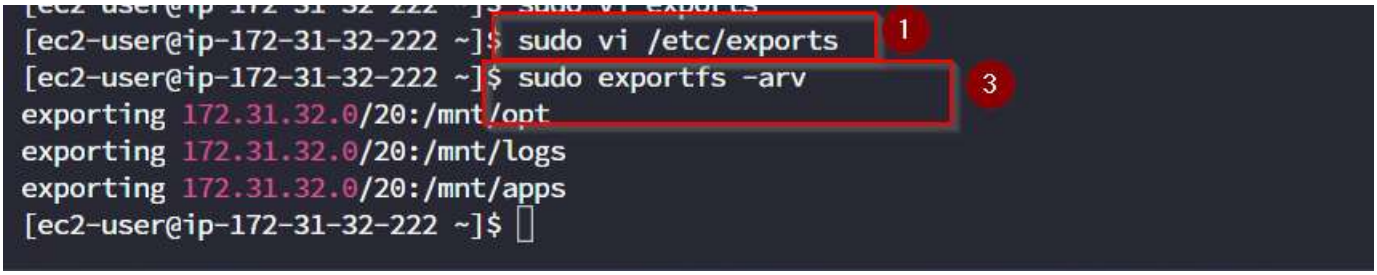
/mnt/apps 172.31.32.0/20(rw,sync,no_all_squash,no_root_squash)

/mnt/logs 172.31.32.0/20(rw,sync,no_all_squash,no_root_squash)

/mnt/opt 172.31.32.0/20(rw,sync,no_all_squash,no_root_squash)

Save and quit the editor **(esc + wq!)**

$ sudo exportfs -arv



Check which port NFS server is listening on using the **rpcinfo** utility and open it using the security groups and edit inbound rule.

We will open the following ports custom UDP 2049, custom TCP 111, NFS 2049, custom UDP 111

$ rpcinfo -p | grep nfs

Type	Protocol	Port range	Source	Description
Custom UDP	UDP	2049	172.31.32.0/20	–
SSH	TCP	22	0.0.0.0/0	–
Custom TCP	TCP	111	172.31.32.0/20	–
Custom UDP	UDP	111	172.31.32.0/20	–
NFS	TCP	2049	172.31.32.0/20	–
HTTP	TCP	80	0.0.0.0/0	–

STEP2.

CONFIGURE THE DATABASE SERVER

$ sudo apt install mysql-server -y

Get into the mysql environment as a root/super user

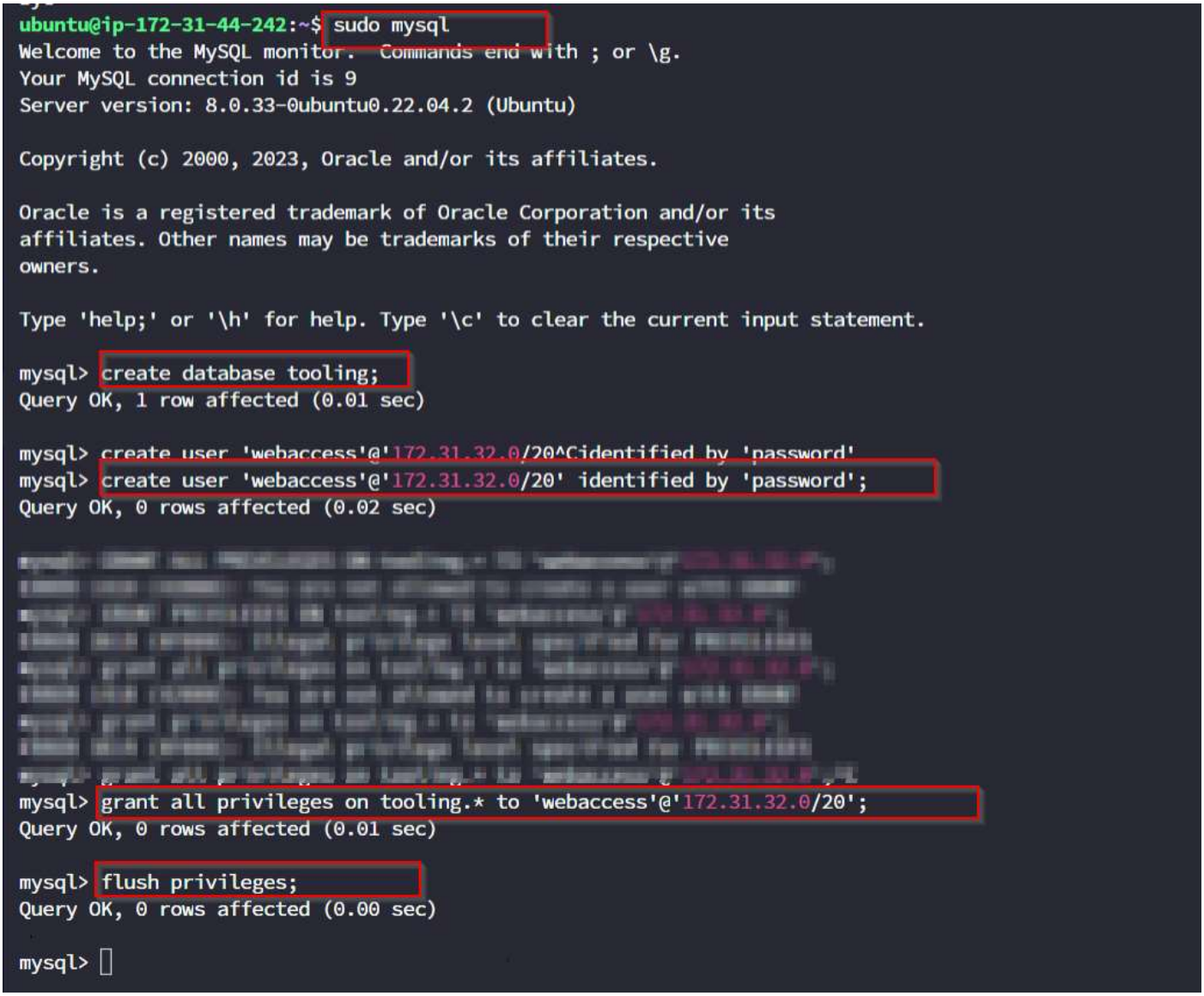
$ sudo mysql

$ create database tooling; (this is to create database called ‘tooling’)

$ create user ‘webaccess’@’172.31.32.0/20’ identified by ‘password’;

$ grant all privileges on tooling.* to ‘webaccess’@’172.31.32.0/20;
(Here, we are granting permission to the user called ‘webaccess’ on ‘tooling’ database to do anything only from the webservers ‘subnet cidr’)

$ flush privileges;



Ensure to locate the mysqld configuration file and set bind address to **0.0.0.0** to enable database server receive request from anywhere

```
# binlog_ignore_db      = include_database_name
ubuntu@ip-172-31-44-242:/etc/mysql/mysql.conf.d$ sudo vi mysqld.cnf
ubuntu@ip-172-31-44-242:/etc/mysql/mysql.conf.d$ sudo systemctl restart mysql-server
Failed to restart mysql-server.service: Unit mysql-server.service not found.
#
# The MySQL database server configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user            = mysql
# pid-file      = /var/run/mysqld/mysqld.pid
# socket        = /var/run/mysqld/mysqld.sock
# port          = 3306
# datadir       = /var/lib/mysql


# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir                = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
mysqlx-bind-address     = 0.0.0.0
#
# * Fine Tuning
#
key_buffer_size         = 16M
# max_allowed_packet    = 64M
# thread_stack          = 256K

# thread_cache_size       = -1
```

Prepare the 3 Webservers

In these next steps we will achieve the following:

Configure NFS client (on all three servers)

Deploy a Tooling application to our Web Servers into a shared NFS folder.

Configure the Web Servers to work with a single MySQL database.

1. Spin up 3 RHEL 8 webserver instances and connect to terminal.

2. Install NFS client

$ sudo yum install nfs-utils nfs4-acl-tools -y

Mount /var/www/ and target the NFS server’s export for apps.

$ sudo mkdir /var/www

$ sudo mount -t nfs -o rw,nosuid <NFS-Server-Private-IP-Address>:/mnt/apps /var/www

(for example, $ sudo mount -t nfs -o rw,nosuid 172.31.32.222:/mnt/apps /var/www)

```
  Verifying        : libnfsidmap-1:2.5.4-18.el9.x86_64
  Verifying        : gssproxy-0.8.4-5.el9_2.x86_64
Installed products updated.

Installed:
  gssproxy-0.8.4-5                    -1.6.3-1.el9.x86_64                libev-4.33-5.el9.x86_64
  libtirpc-1.3.3-1                    -libev-0.3.2-3.el9.x86_64          nfs-utils-1:2.5.4-18.el9.x86_64
  quota-1:4.06-6.e                    s-1:4.06-6.el9.noarch              rpcbind-1.2.6-5.el9.x86_64

Complete!
[ec2-user@ip-172-31-44-189 ~]$ sudo mkdir /var/www
[ec2-user@ip-172-31-44-189 ~]$ sudo mount -t nfs -o rw,nosuid 172.31.32.222:/mnt/apps /var/www
[ec2-user@ip-172-31-44-189 ~]$ df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  4.0M     0  4.0M   0% /dev
tmpfs                     385M     0  385M   0% /dev/shm
tmpfs                     154M  7.7M  147M   5% /run
/dev/xvda4                9.4G  1.3G  8.1G  14% /
/dev/xvda3                495M  153M  343M  31% /boot
/dev/xvda2                200M  8.0K  200M   1% /boot/efi
tmpfs                      77M     0   77M   0% /run/user/1000
172.31.32.222:/mnt/apps  9.0G   97M  8.9G   2% /var/www
[ec2-user@ip-172-31-44-189 ~]$
```

NFS client successfully installed

Verify the NFS was mounted successfully by running the command **df -h**

To make sure the changes will persist on the webserver after start up, run the command

$ sudo vi /etc/fstab

Add **172.31.32.222:/mnt/apps /var/www nfs defaults 0 0**

```
UUID=287d9c0b-0e0f-4e92-8534-45733aa3dc68       /       xfs     defaults        0       0
UUID=7bc24af7-289d-4bce-b17e-300c3aafe968       /boot   xfs     defaults        0       0
UUID=7B77-95E7  /boot/efi       vfat    defaults,uid=0,gid=0,umask=077,shortname=winnt  0       2

172.31.32.222:/mnt/apps /var/www nfs defaults 0 0
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

3. Install Remi's repository, Apache and PHP with the commands below:

$ sudo yum install httpd -y

$ sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

$ sudo dnf install dnf-utils http://rpms.remirepo.net/enterprise/remi-release-8.rpm

$ sudo dnf module reset php

$ sudo dnf module enable php

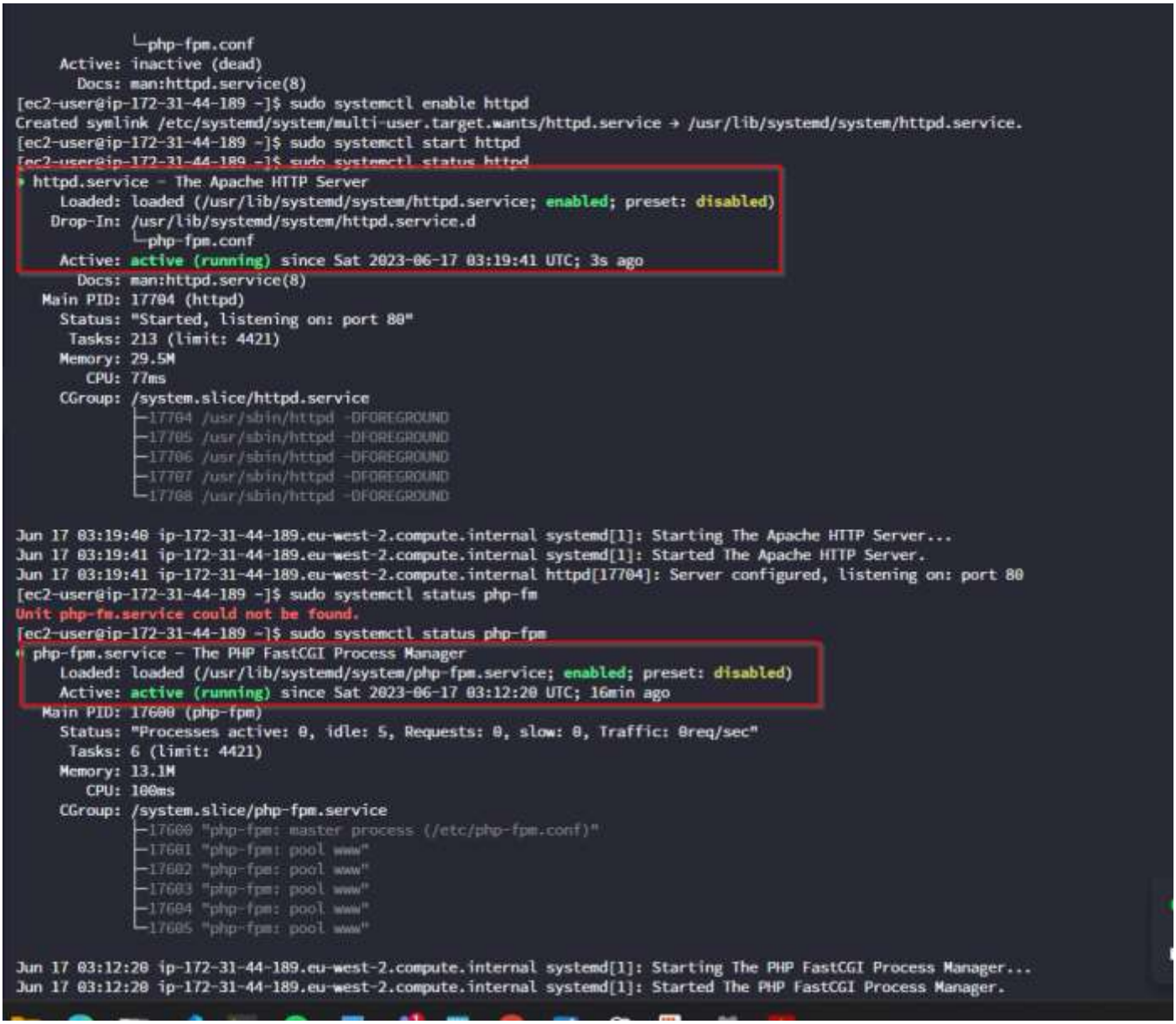
$ sudo dnf install php php-opcache php-gd php-curl php-mysqlnd

$ sudo systemctl start php-fpm

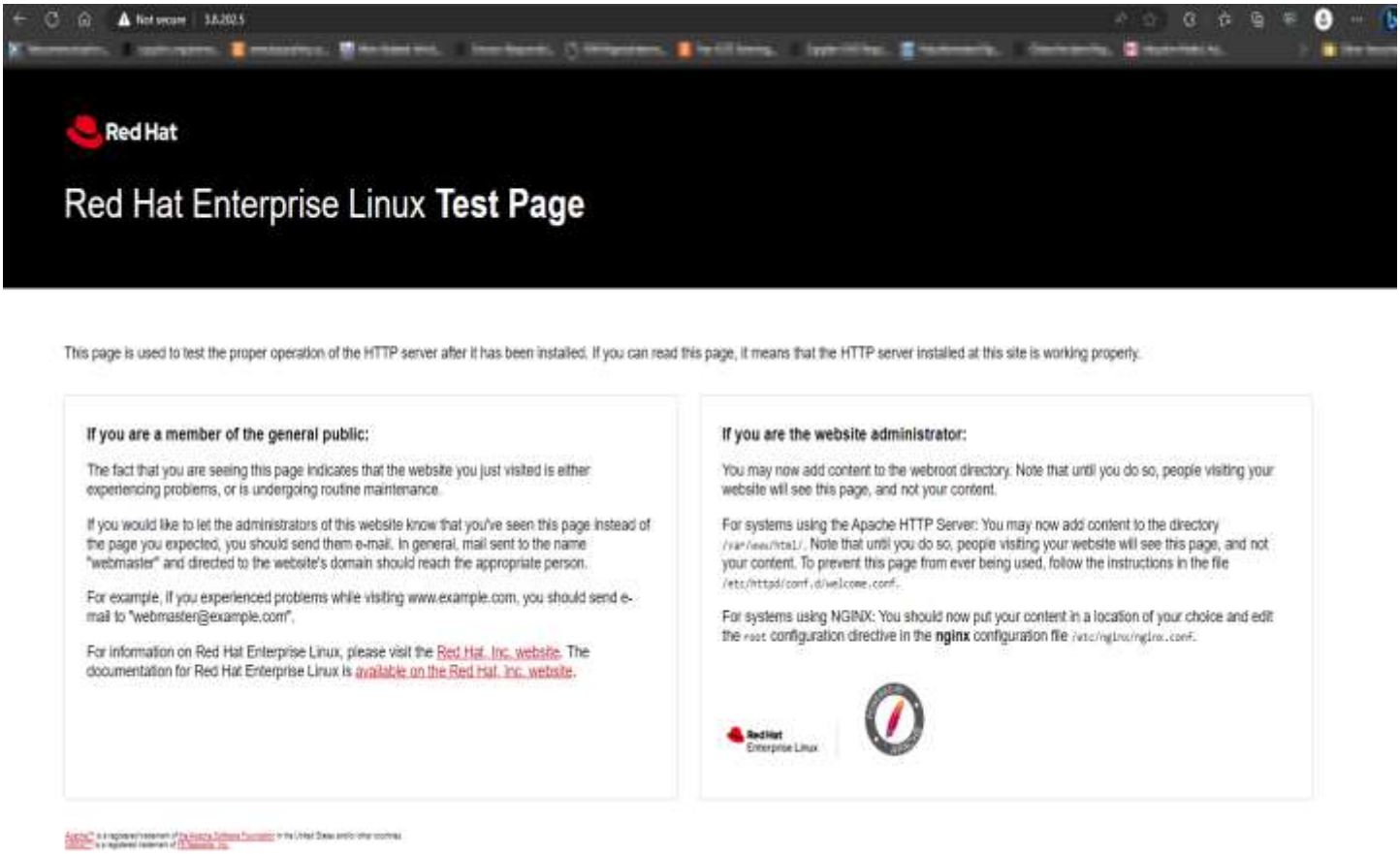
$ sudo systemctl enable php-fpm

$ setsebool -P httpd_execmem 1

Also enable and restart Apache (httpd) service.



4. Now use the webserver public IP to connect to a browser, you will find the screen below



This indicates that the Http server has been installed successfully.

Locate the log folder for Apache on the Web Server and mount it on the NFS server's export for logs to make sure the mount point will persist after reboot.

$ sudo mount -t nfs -o rw,nosuid 172.31.32.222:/mnt/logs /var/log/httpd

$ sudo vi /etc/fstab

```
UUID=287d9c0b-0e0f-4e92-8534-45733aa3dc68       /       xfs     defaults        0       0
UUID=7bc24af7-289d-4bce-b17e-300c3aafe968       /boot   xfs     defaults        0       0
UUID=7B77-95E7  /boot/efi       vfat    defaults,uid=0,gid=0,umask=077,shortname=winnt  0       2

172.31.32.222:/mnt/apps /var/www nfs defaults 0 0
172.31.32.222:/mnt/logs /var/log/httpd nfs defaults 0 0
```

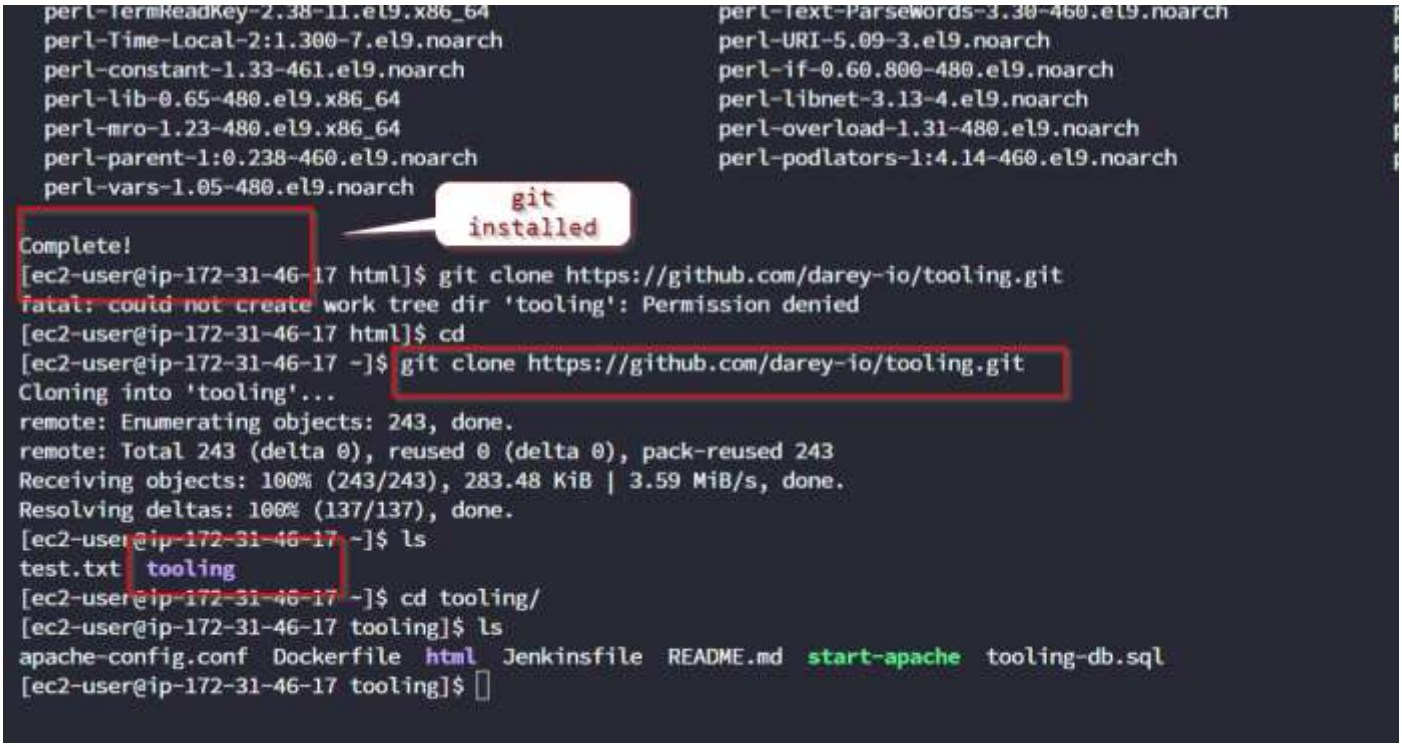
Now we must add contents to the /var/www/html directory.

We will fork the tooling source code from Darey.io Github Account to your Github account, deploy the tooling website's code to the Webserver and ensure that the html folder from the repository is deployed to /var/www/html

Firstly, we have to install git on our server in order to clone the code from the git repository.

$sudo yum install git -y

$ git clone https://github.com/darey-io/tooling.git



Deploy the tooling website's code to the Webserver. Ensure that the html folder from the repository is deployed to /var/www/html

$ cd /tooling/html

$ sudo cp -R . /var/www/html/

Ensure that you have TCP port 80 open on the webserver instance.

Update the website's configuration (in /var/www/html/functions.php file)to connect to the database. Input the private IP of the database , username and password.

You would notice that both servers are now synchronised on the updated websites's configuration.

```
<?php
session_start();

// connect to database
$db = mysqli_connect('172.31.44.242', 'webaccess', 'password', 'tooling');

// Check connection
// if (mysqli_connect_errno()) {
// echo "Failed to connect to MySQL: " . mysqli_connect_error();
// exit();
// }
// else{
// echo "connected";
// }

// variable declaration
$username = "";
$email    = "";
$errors   = array();

// call the register() function if register_btn is clicked
if (isset($_POST['register_btn'])) {
        register();
}

// REGISTER USER
function register(){
        // call these variables with the global keyword to make them available in function
        global $db, $errors, $username, $email;

        // receive all input values from the form. Call the e() function
    // defined below to escape form values
        $username    =  e($_POST['username']);
        $email       =  e($_POST['email']);
        $password_1  =  e($_POST['password_1']);
        $password_2  =  e($_POST['password_2']);

        // form validation: ensure that the form is correctly filled
```

Install mysql client on the webservers for them to be able to connect with the Database server.

$ sudo yum install mysql -y

Apply tooling-db.sql script to your database using this command mysql -h <databse-private-ip> -u <db-username> -p <db-pasword> tooling < tooling-db.sql

We will omit the password for security reasons.

Run the script below (make sure you are in the tooling directory).

$ sudo mysql -h 172.31.44.242 -u webaccess -p tooling < tooling-db.sql

Check the database server to ensure a table and user has been added.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| tooling            |
+--------------------+
5 rows in set (0.00 sec)

mysql> use tooling;
Database changed
mysql> show tables;
Empty set (0.01 sec)

mysql> show tables
    -> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the
s' at line 2
mysql> show tables;
+------------------+
| Tables_in_tooling |
+------------------+
| users            |
+------------------+
1 row in set (0.00 sec)

mysql>
```

```
+--------------------+
| Tables_in_tooling |
+--------------------+
| users              |
+--------------------+
1 row in set (0.00 sec)

mysql> select users;
ERROR 1054 (42S22): Unknown column 'users' in 'field list'
mysql> select * from users;
+----+----------+----------------------------------+---------------+-----------+--------+
| id | username | password                         | email         | user_type | status |
+----+----------+----------------------------------+---------------+-----------+--------+
|  1 | admin    | 21232f297a57a5a743894a0e4a801fc3 | dare@dare.com | admin     | 1      |
+----+----------+----------------------------------+---------------+-----------+--------+
1 row in set (0.00 sec)

mysql>
```

Disable SELinux sudo setenforce 0

```
[ec2-user@ip-172-31-44-189 ~]$ cd tooling/
[ec2-user@ip-172-31-44-189 tooling]$ sudo setenforce 0
[ec2-user@ip-172-31-44-189 tooling]$ sudo vi /etc/sysconfig/selinux
[ec2-user@ip-172-31-44-189 tooling]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-44-189 tooling]$
```

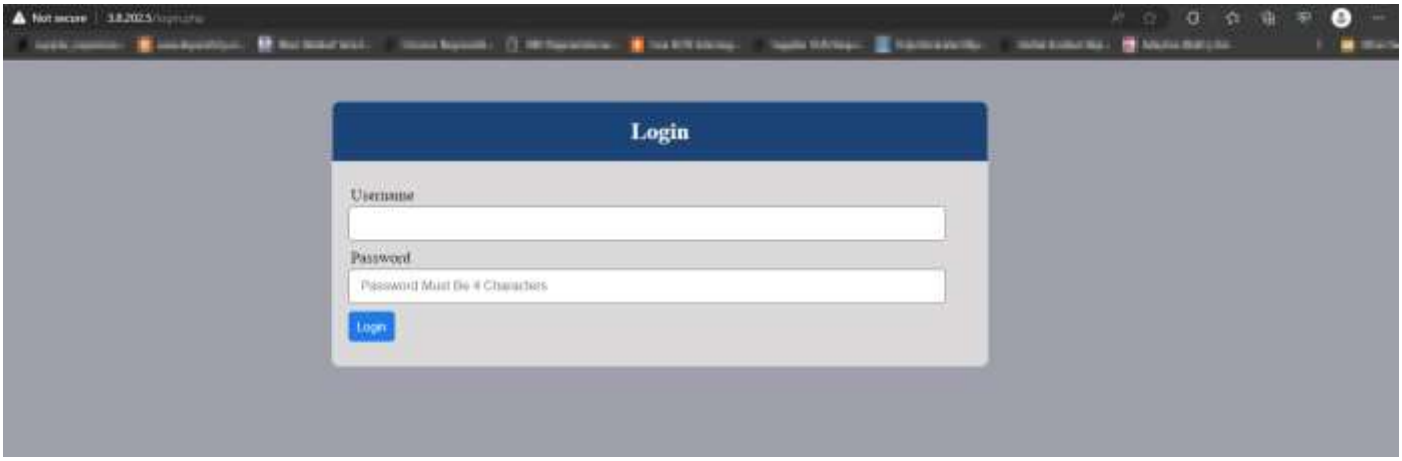
To make this change permanent – open following config file.

$ sudo vi /etc/sysconfig/selinux

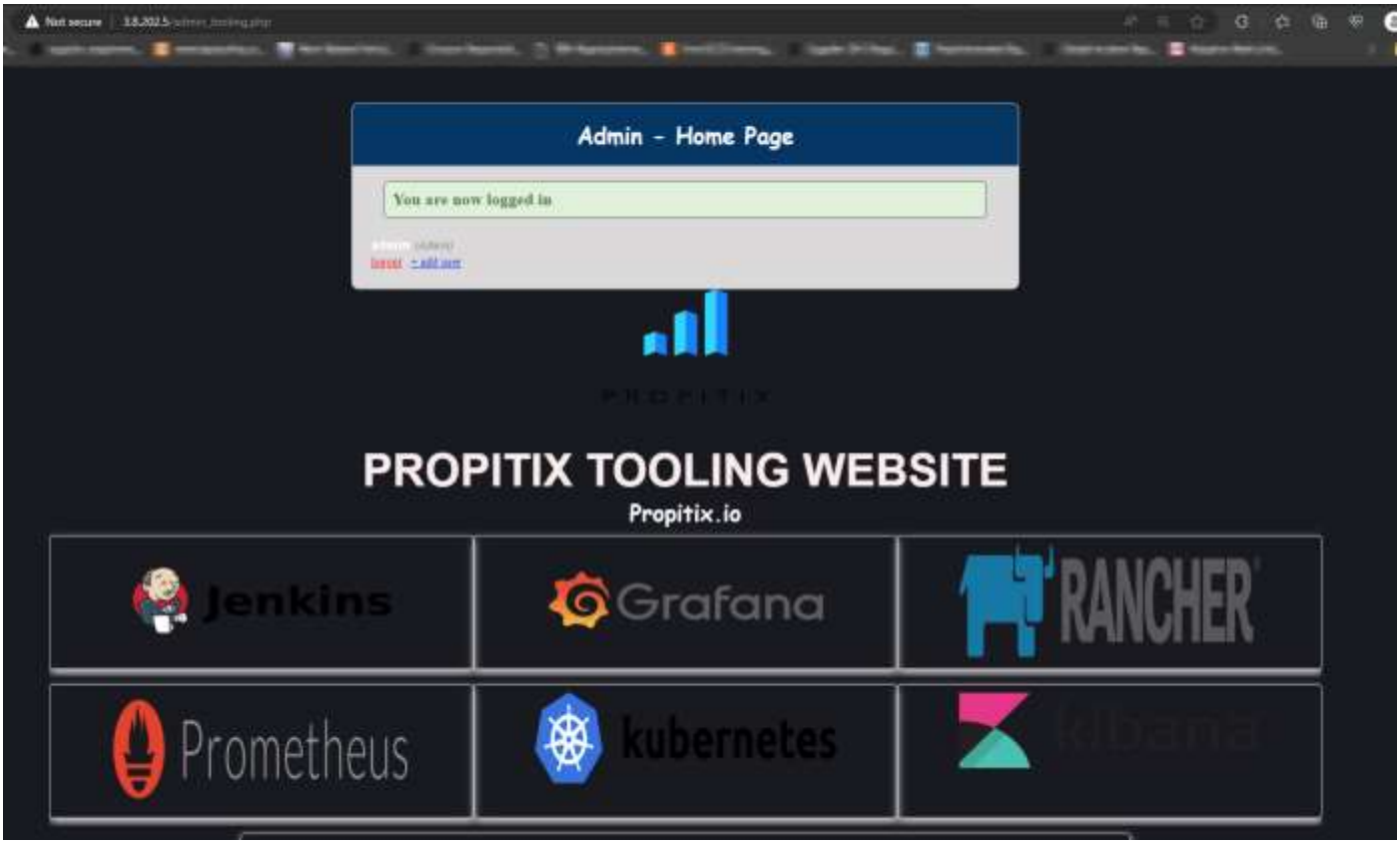
and set **SELINUX=disabled** then restrt httpd.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
# See also:
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/#getting-started-with-selinux-selinux-states-and-mod
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#    grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Reload the browswer



There you go! We now have our tooling website!



Congratulations and thank you for following through!