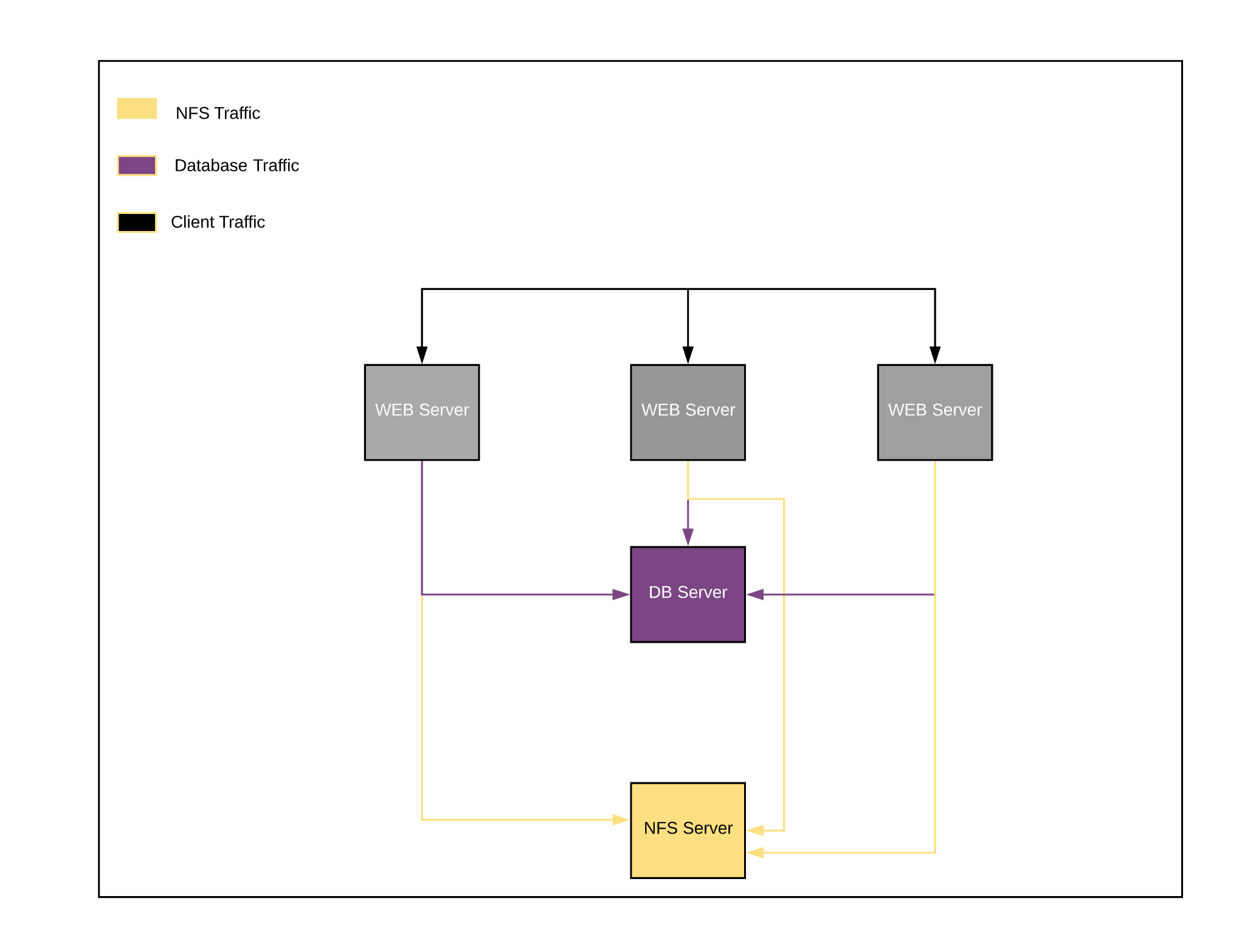
**DevOps Tooling Website Solution**

The objective of this project is to develop a means of implementing a tooling website solution which enables easy access to DevOps tools within the corporate infrastructure.

In this project, we will be looking at how the NFS server is used to serve files to the web servers.Also, a database server will be configured for the DevOps tooling website solution. A basic view of the whole setup is illustrated below;



PREREQUISITES:

* Access to an AWS account and five (5) virtual instances with an Ubuntu 20.04 server OS image on one and the rest having the RHEL8.0 Server OS Image installed.
* A laptop or PC to serve as a client.

The steps include the below:

**STEP 1**: Launch 4 new EC2 instances with RHEL Linux 8 OS and one with Ubuntu 20.04.

In order to get started, we will require the IP addresses of the five EC2 instances as mentioned above:

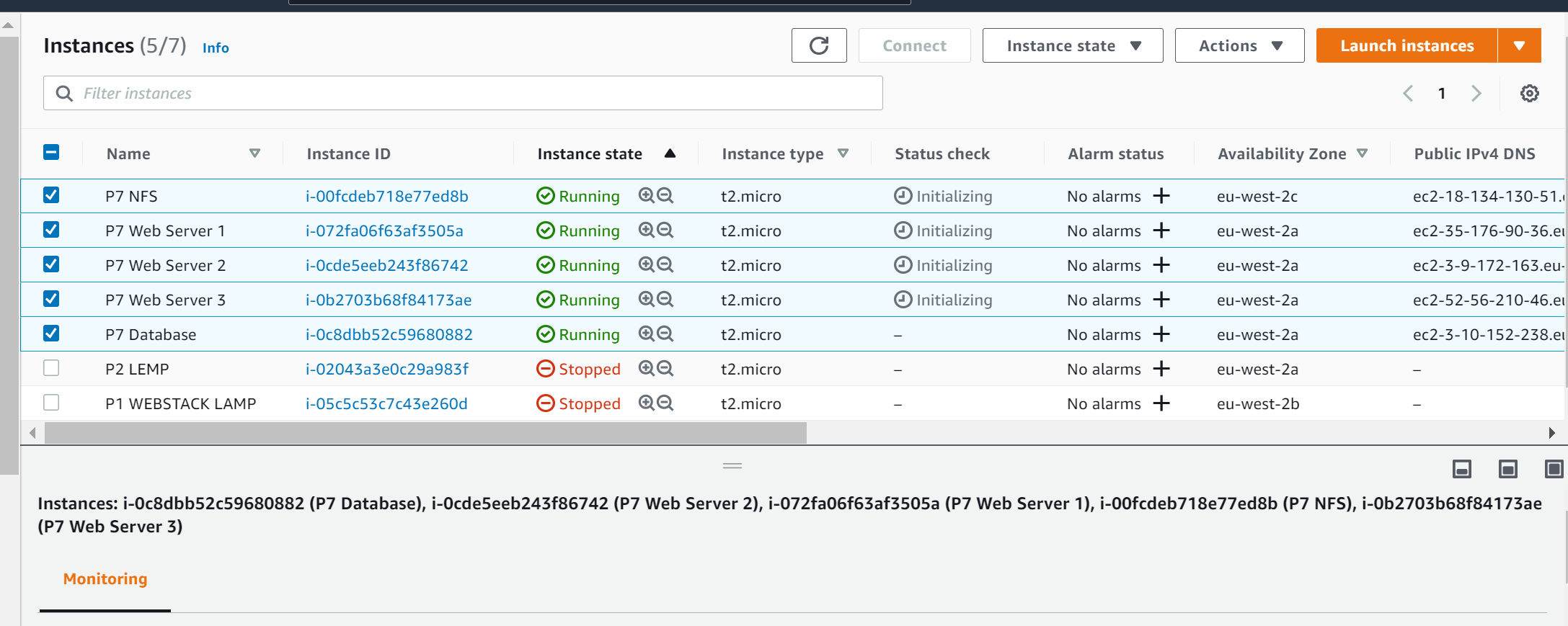
NFS Server - **172.31.4.52**

Database Server - **172.31.18.63**

Web Server 1 - **172.31.16.162**

Web server 2 - **172.31.16.41**

Web server 3 - **172.31.30.14**



**STEP 2: Preparing the NFS Server:**

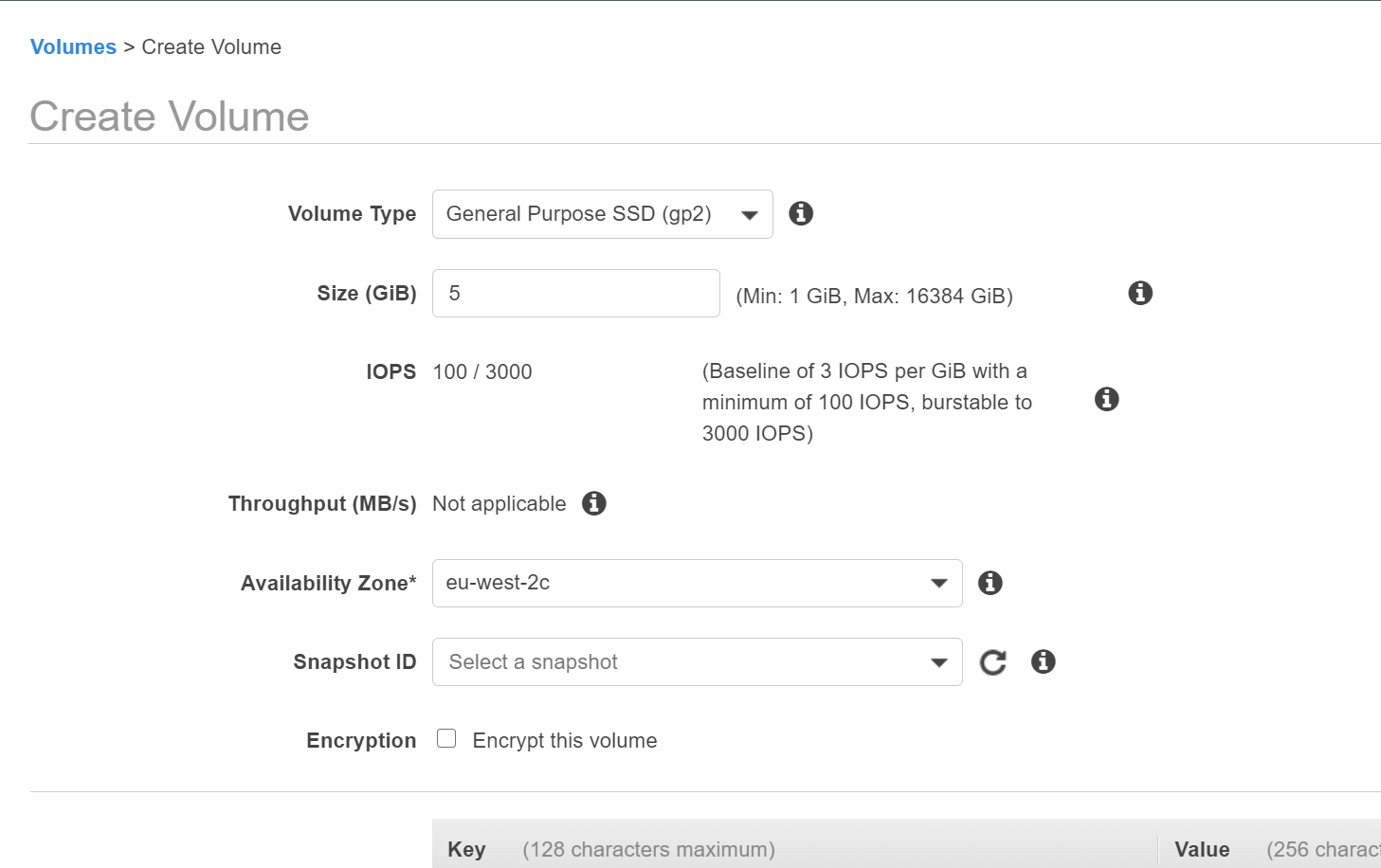
A [Network File System](https://en.wikipedia.org/wiki/Network_File_System) is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed. It definitely has advantages and dis-advantages. You can read more [here](https://searchenterprisedesktop.techtarget.com/definition/Network-File-System), but because of its shortcomings, it is not recommended for storing database files. Hence, we will only implement NFS for the website files.

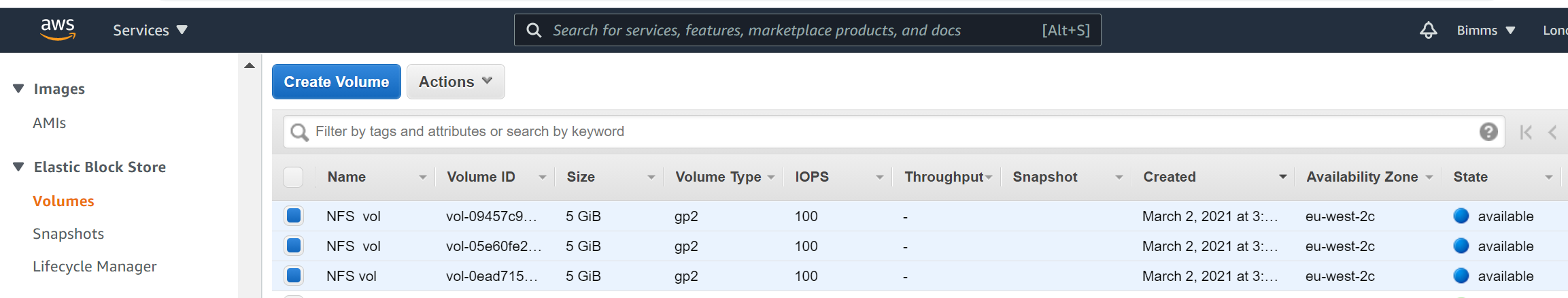
a. Logical Volume Configuration:

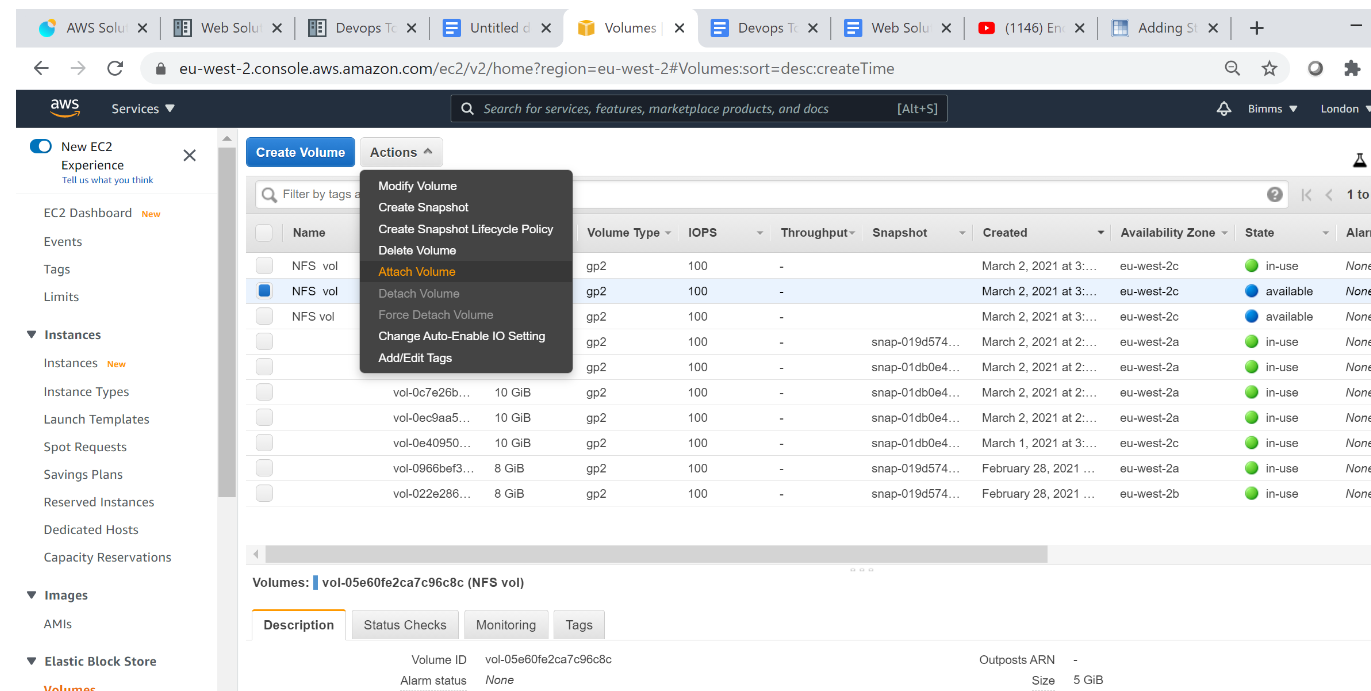
We shall be equipping the NFS server with Logical Volume Management configuration. Our NFS server will make use of three (3) of size 15GB each.

In order to choose the right storage system to implement the solution, we need to identify the suitable storage solution by checking: what data will be stored, in what format, how this data will be accessed, by whom, from where, how frequently, among others.

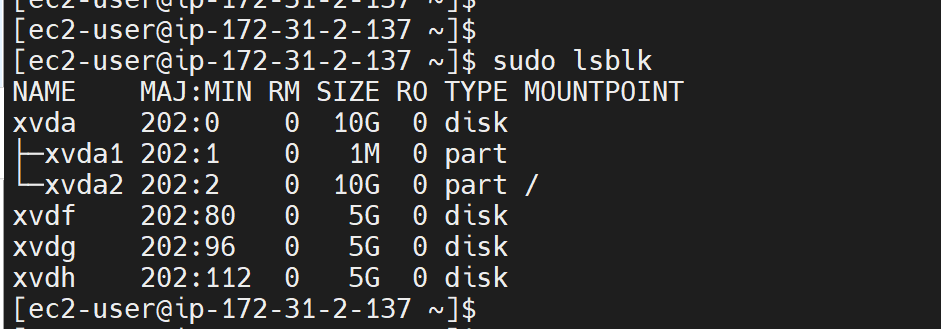
* Create and attach the 3 volumes to NFS instance,
* Locate the EBS volumes section and click on create volume
* Attach the volumes to the NFS server instance







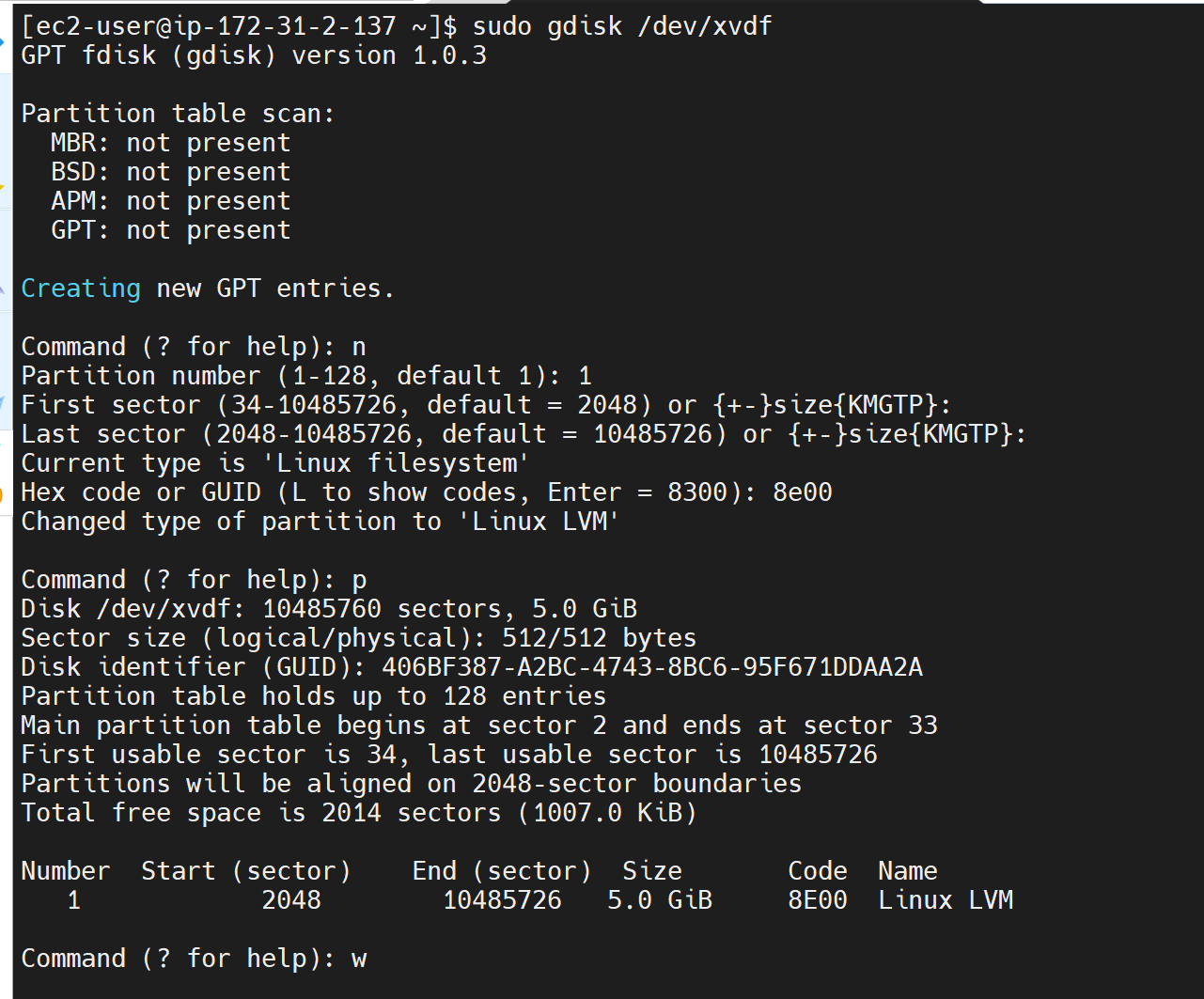
* Verify the disks are attached correctly: sudo lsblk

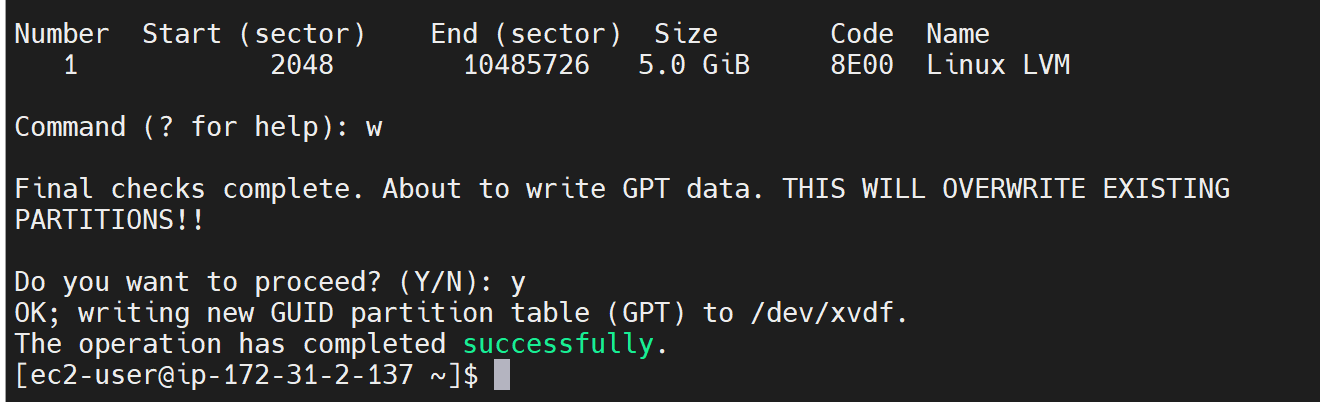


**Partitioning the disks:**

Use gdisk utility to create a single partition on each of the 3 disks

sudo gdisk /dev/xvdf



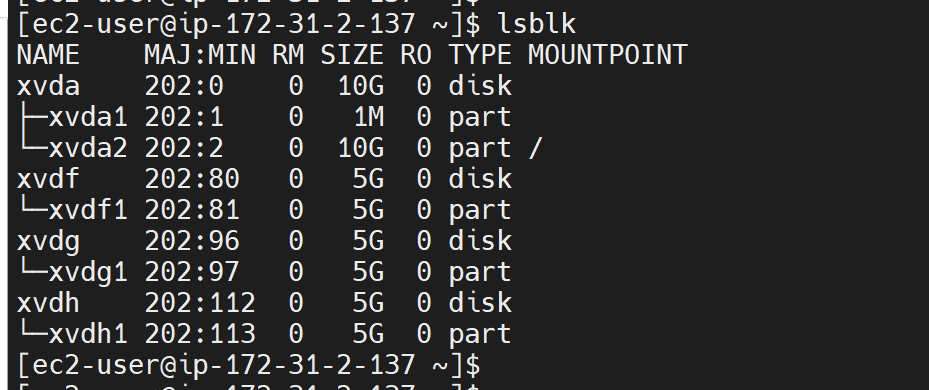


Repeat the same for the three disks.

sudo gdisk /dev/xvdg

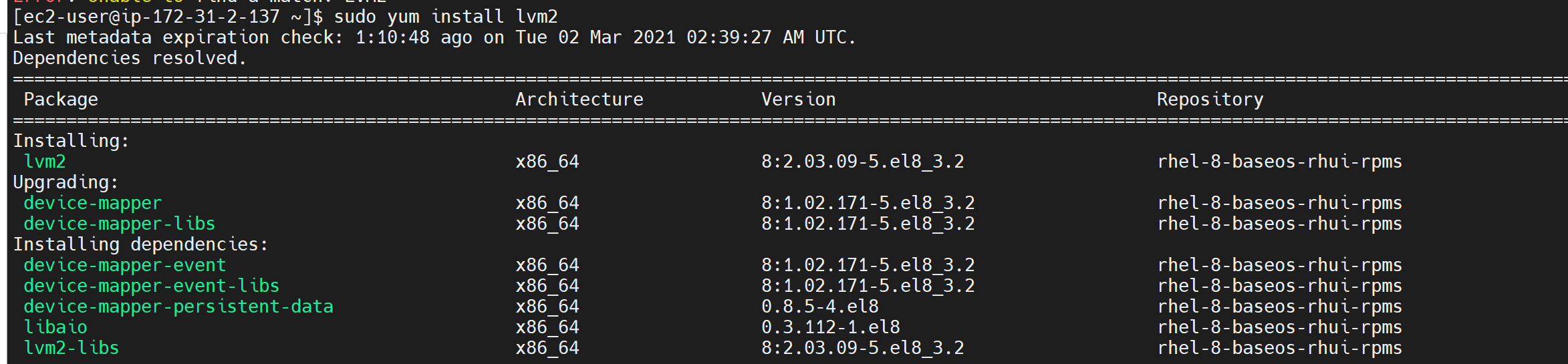
sudo gdisk /dev/xvdh

Run lsblk to view the newly configured partition on each of the 3 disks.



- **Creating LVM logical Volume**

- Install LVM package: sudo yum install lvm2

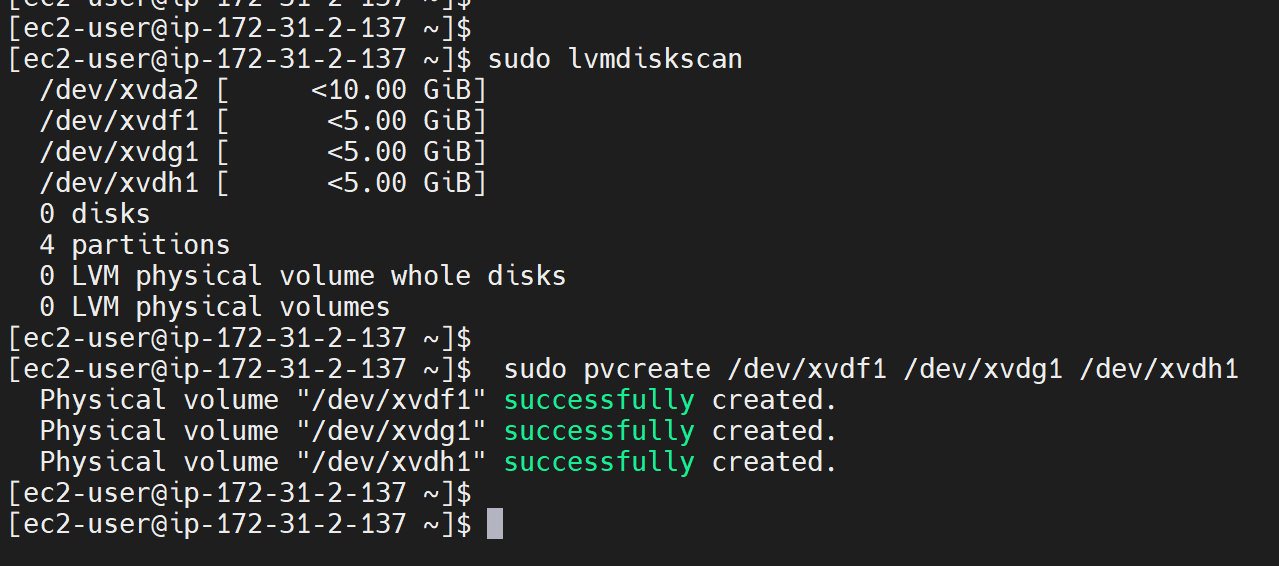


* Use lvmdiskscan utility to scan / check available storage for LVM. It shows the devices that are suitable to be turned in physical volumes for LVM.

sudo lvmdiskscan

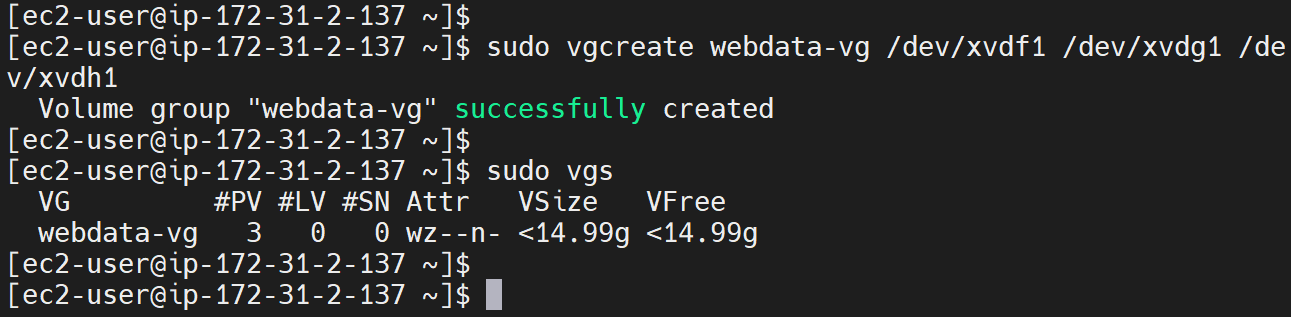
* Create and mark the newly created partitions of the raw storage device as LVM physical volumes using the command below;

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

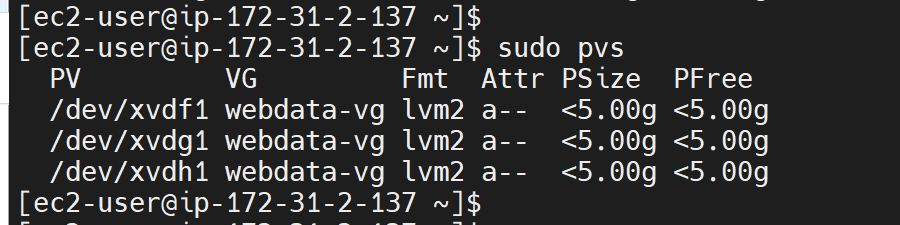


* Create a volume group named **‘webdata’** using the physical group already created.

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



Confirm the PVs / vgs: sudo pvs



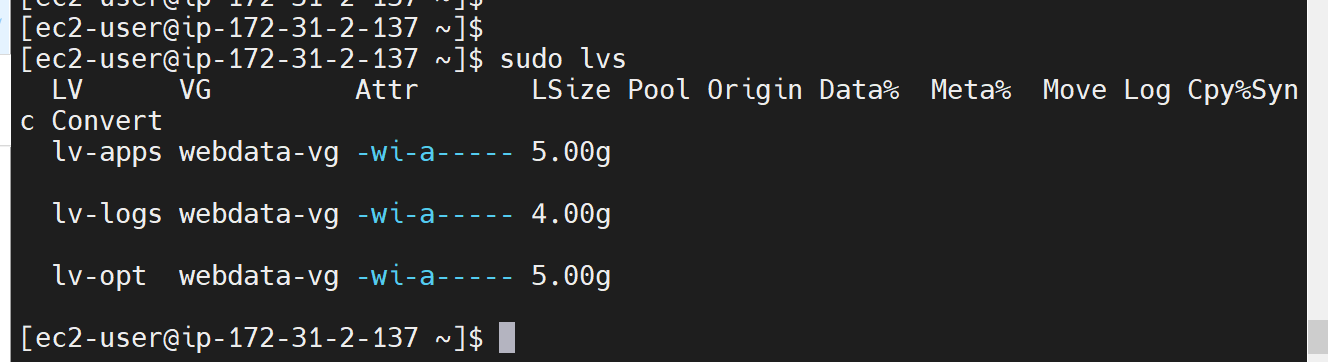
* Create three logical volumes, lv-opt lv-apps, and lv-logs:

sudo lvcreate -n lv-opt -L 5G webdata-vg

sudo lvcreate -n lv-apps -L 5G webdata-vg

sudo lvcreate -n lv-logs -L 4G webdata-vg

Confirm LVs have been created: sudo lvs



Please note - a reduced space was allocated to lv-logs because the volume group will have to use some of the space it has to store necessary information about itself.

Confirm all the set up with: sudo vgdisplay -v, pvs, lvs and sudo

lsblk

**Formatting the logical volumes with xfs file system**

We will use the mkfs utility to achieve this with the command lines below:

sudo mkfs.xfs /dev/webdata-vg/lv-opt

sudo mkfs.xfs /dev/webdata-vg/lv-apps

sudo mkfs.xfs /dev/webdata-vg/lv-logs

\*Create mount points on /mnt directory for the logical volumes as follows;

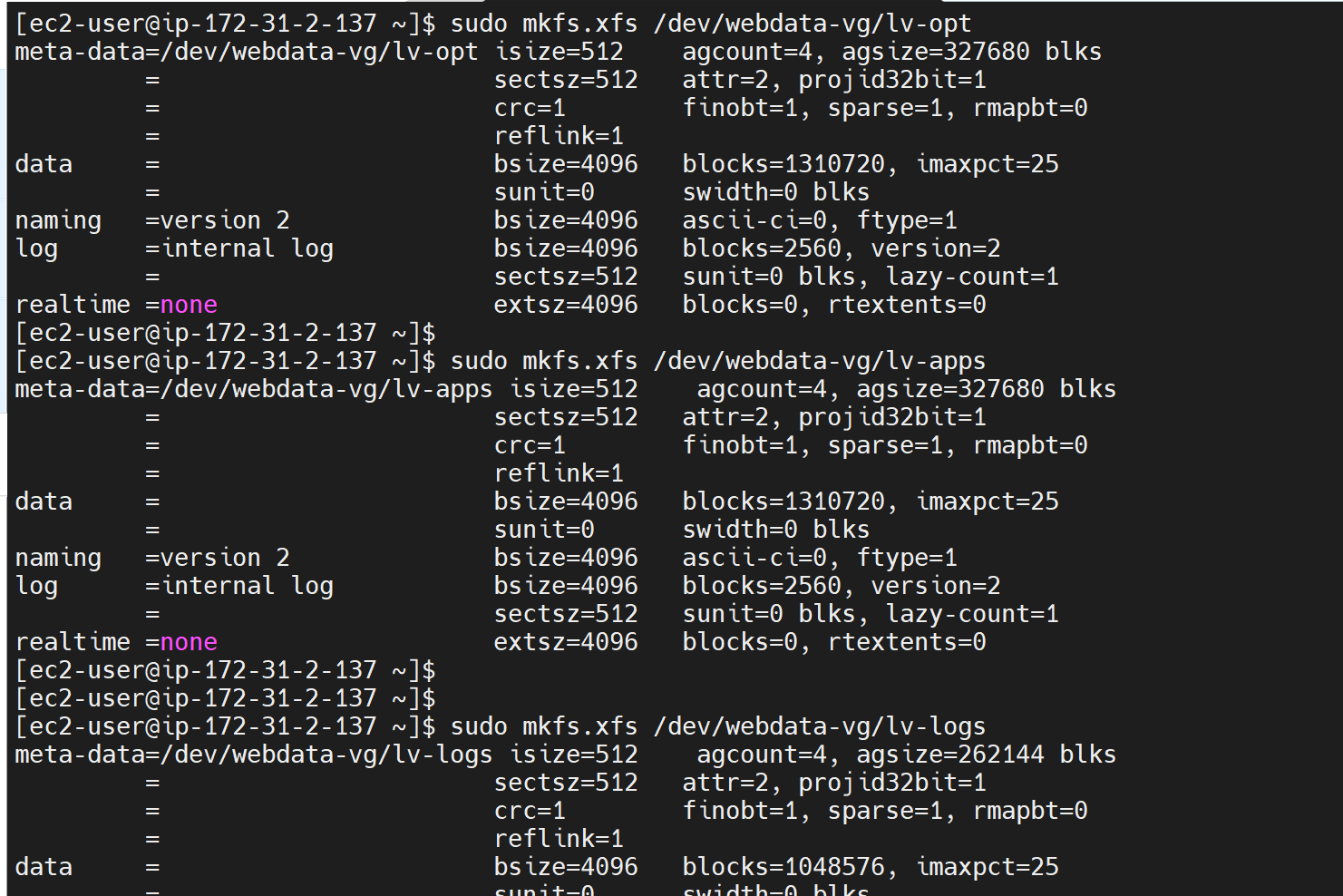
* /mnt/logs sudo mkdir /mnt/logs
* /mnt/opt sudo mkdir /mnt/opt
* /mnt/apps sudo mkdir /mnt/apps

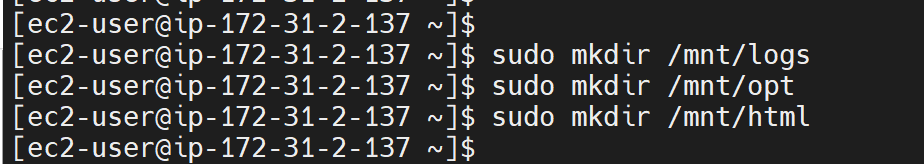
\*Mount lv-apps on /mnt/html - To be used by web servers Mount lv-logs on /mnt/logs - To be used by web server logs Mount lv-opt on /mnt/opt - To be used by Jenkins server in [Project 8](https://dareyio-pbl-progressive.readthedocs-hosted.com/en/latest/project8.html).

sudo mount /dev/webdata-vg/lv-opt /mnt/opt

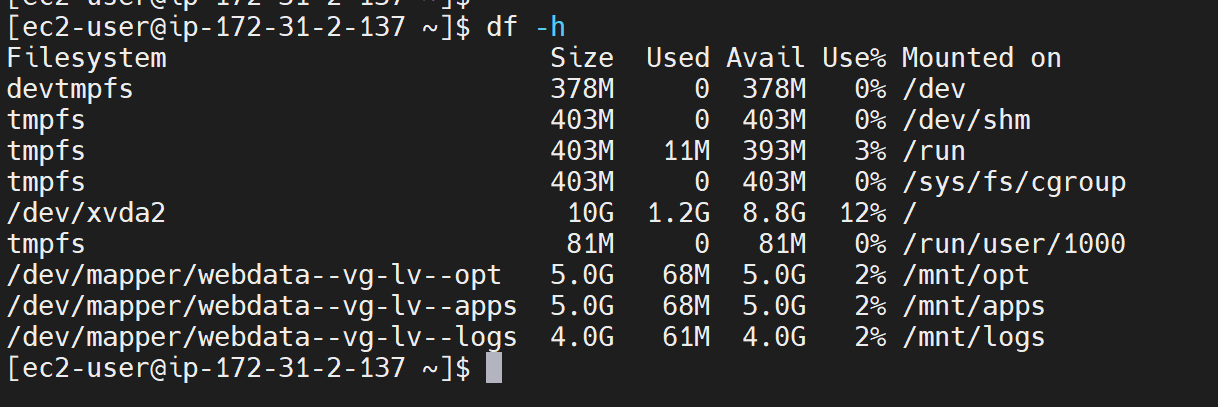
sudo mount /dev/webdata-vg/lv-apps /mnt/apps

sudo mount /dev/webdata-vg/lv-logs /mnt/logs





Confirm the mounts: df -h



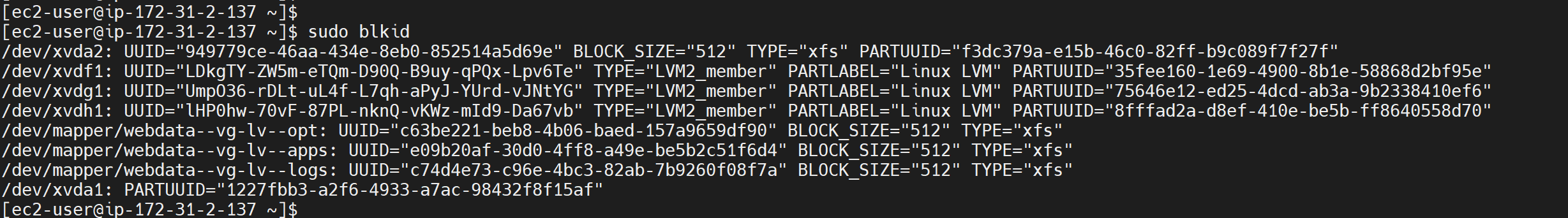
\*Update the ‘/etc/fstab’ file in order to enable the mount configuration to

persist upon restart of the server.

We will use the block id command to obtain the UUU code, which the system

device’s name.

sudo blkid



UUID=c63be221-beb8-4b06-baed-157a9659df90 /mnt/opt

UUID=e09b20af-30d0-4ff8-a49e-be5b2c51f6d4 /mnt/apps

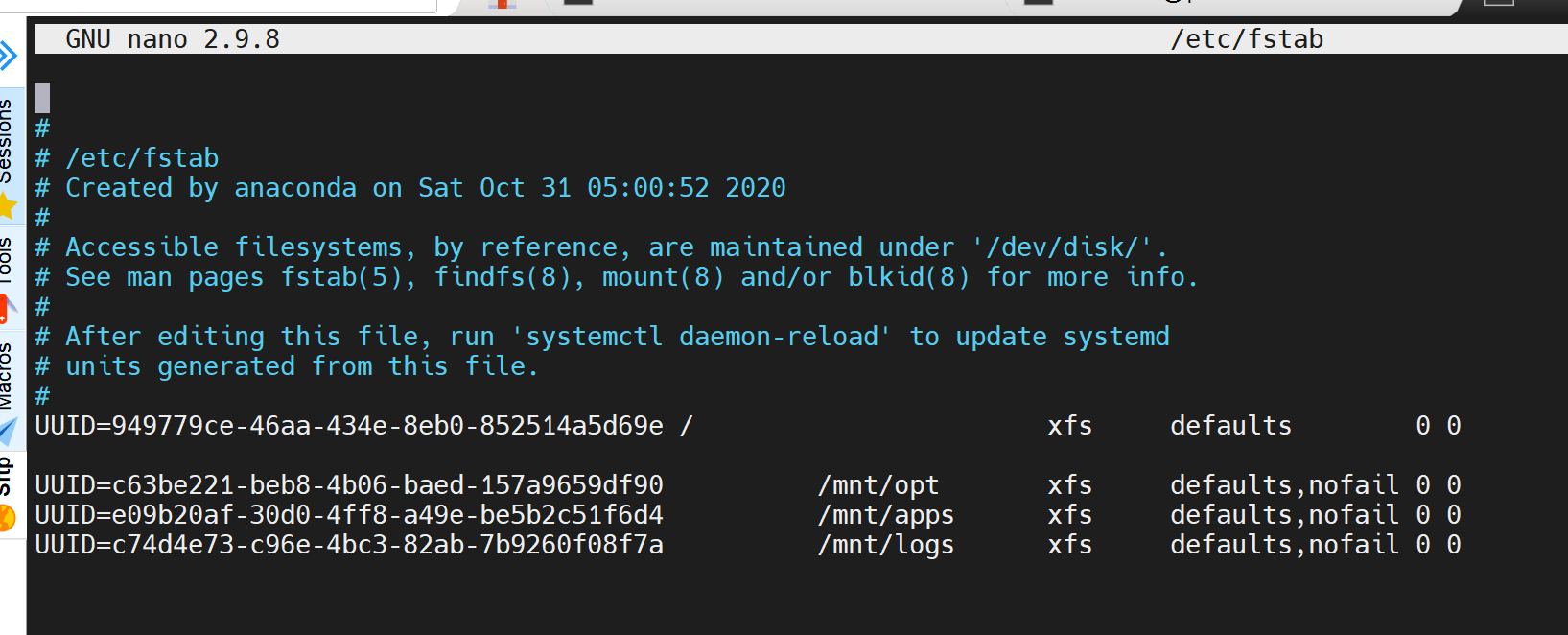
UUID=c74d4e73-c96e-4bc3-82ab-7b9260f08f7a /mnt/logs

**Go into the /etc/fstab/ file and paste the below in:**

UUID=c63be221-beb8-4b06-baed-157a9659df90 /mnt/opt xfs defaults,nofail, 0 0

UUID=e09b20af-30d0-4ff8-a49e-be5b2c51f6d4 /mnt/apps xfs defaults,nofail, 0 0

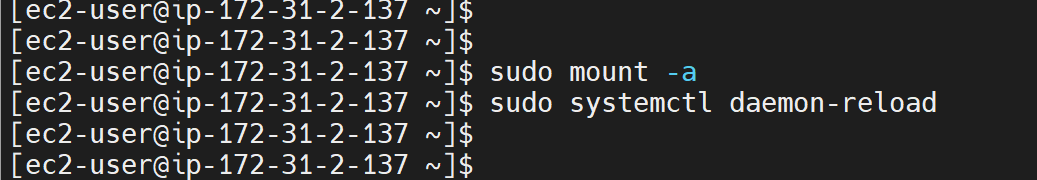
UUID=c74d4e73-c96e-4bc3-82ab-7b9260f08f7a /mnt/logs xfs defaults,nofail, 0 0



Save and close.

Check that the configuration in the etc/fstab is correct: sudo mount -a

Update all changes: sudo systemctl daemon-reload



### 

### 

### **Install and Configure the NFS Server.**

Firstly, install the NFS utils package

sudo yum -y update

sudo yum install nfs-utils -y

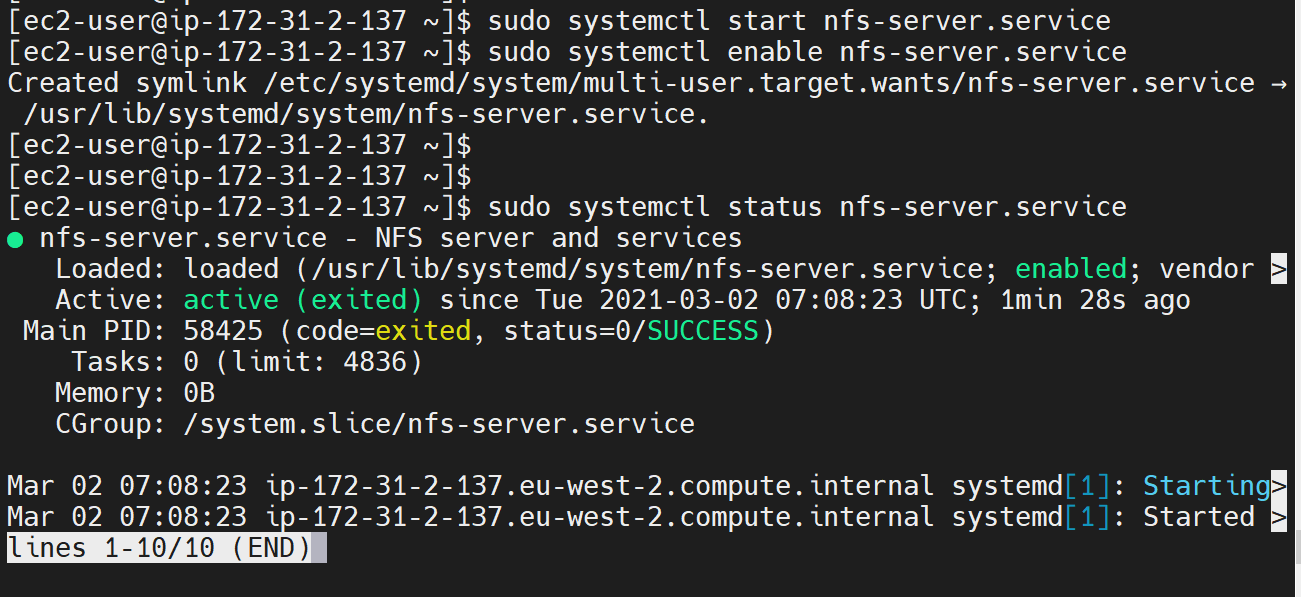
Start and enable the NFS service

sudo systemctl start nfs-server.service

sudo systemctl enable nfs-server.service

Check the NFS status

sudo systemctl status nfs-server.service

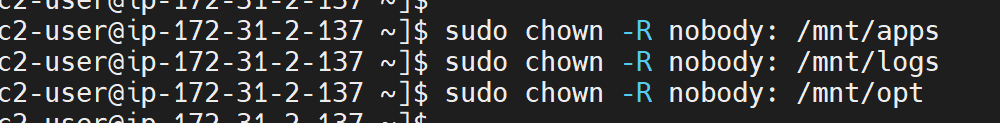


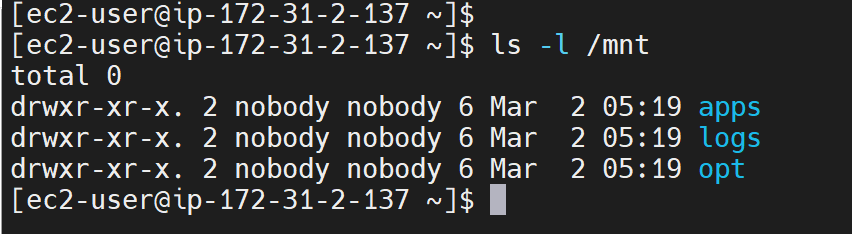
b. Change the permission of the directories to be exported. The ‘777’ will enable our Web servers to read, write and execute files on NFS. Also note that the ‘nobody’ implies that even though the permission has been given, only the servers that will be exported will have this access.

sudo chown -R nobody: /mnt/apps

sudo chown -R nobody: /mnt/logs

sudo chown -R nobody: /mnt/opt



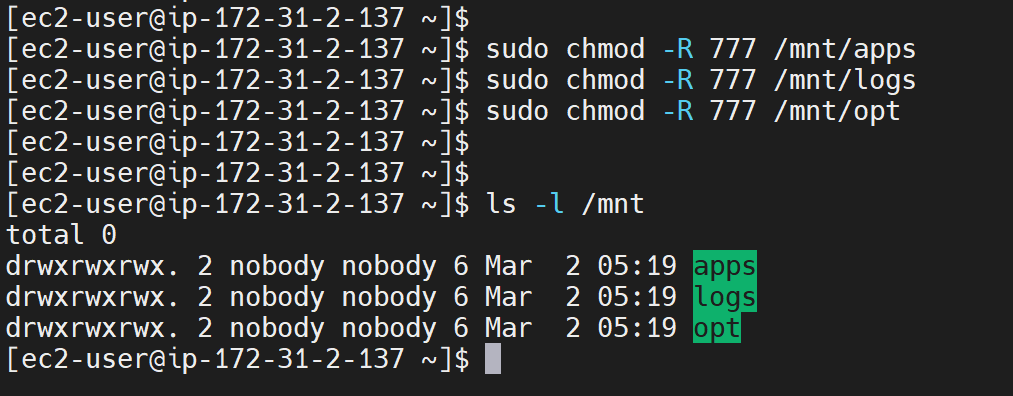


sudo chmod -R 777 /mnt/apps

sudo chmod -R 777 /mnt/logs

sudo chmod -R 777 /mnt/opt

OR sudo chmod 777 /mnt/logs /mnt/opt /mnt/apps



Restart the nfs-server to incorporate all the changes

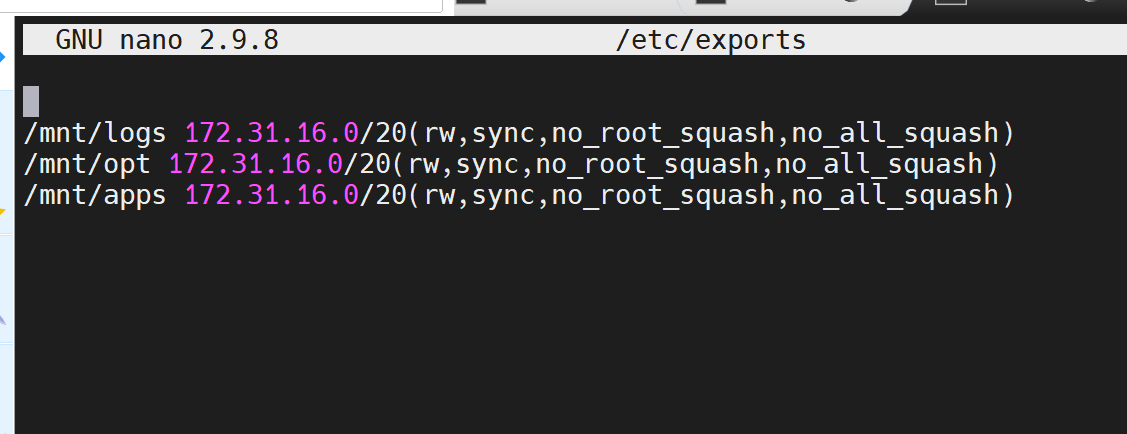
sudo systemctl restart nfs-server.service

Specify the configuration for the mount exports in the /etc/exports file:

sudo nano /etc/exports

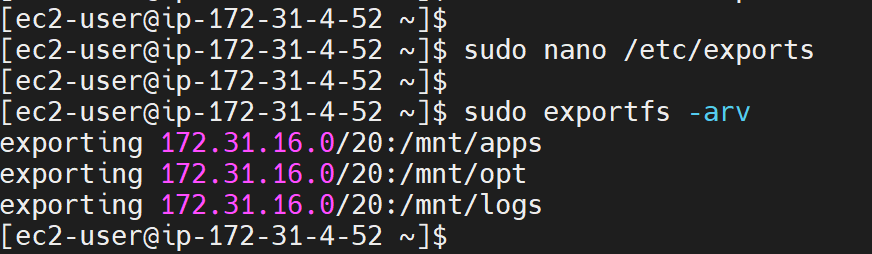
Subnet for NFS Server: 172.31.0.0/20

/mnt/logs 172.31.16.0/20(rw,sync,no\_root\_squash,no\_all\_squash)  
 /mnt/opt 172.31.16.0/20(rw,sync,no\_root\_squash,no\_all\_squash)  
 /mnt/apps 172.31.16.0/20(rw,sync,no\_root\_squash,no\_all\_squash)



Export the mounts for web servers' subnet cidr to connect as clients.:

sudo exportfs -arv



Restart NFS service: sudo systemctl enable nfs-server.service

**\*Blocker - At this point, my EC2 instance got corrupted and could not connect to the terminal.**

Actions taken to correct:

1. Detected what the issue was through the error in AWS.

Got to EC2 instance in aws

Action menu

Monitor and troubleshoot

Get system log

Noticed issues with fstab configuration - input ‘nofails’ instead of ‘nofail’

1. Stopped current instance
2. Detached the volumes
3. Created a new instance
4. Attached the root volume from the initial instance
5. Tried to resolve the issue via terminal before attaching back as /dev/sda1 but could not get the initial instance to initialise and run again.

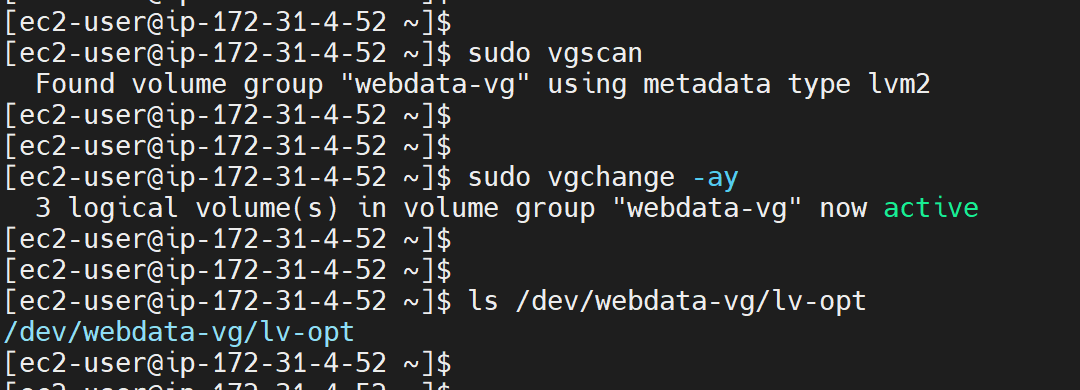
Next steps:

1. Created a new instance
2. Attached all the three volumes (already formatted and partitioned)

* Connected to the terminal
* Checked and ensured I could still access all the logical volumes
* Then activated the logical volumes on my new instance with the command

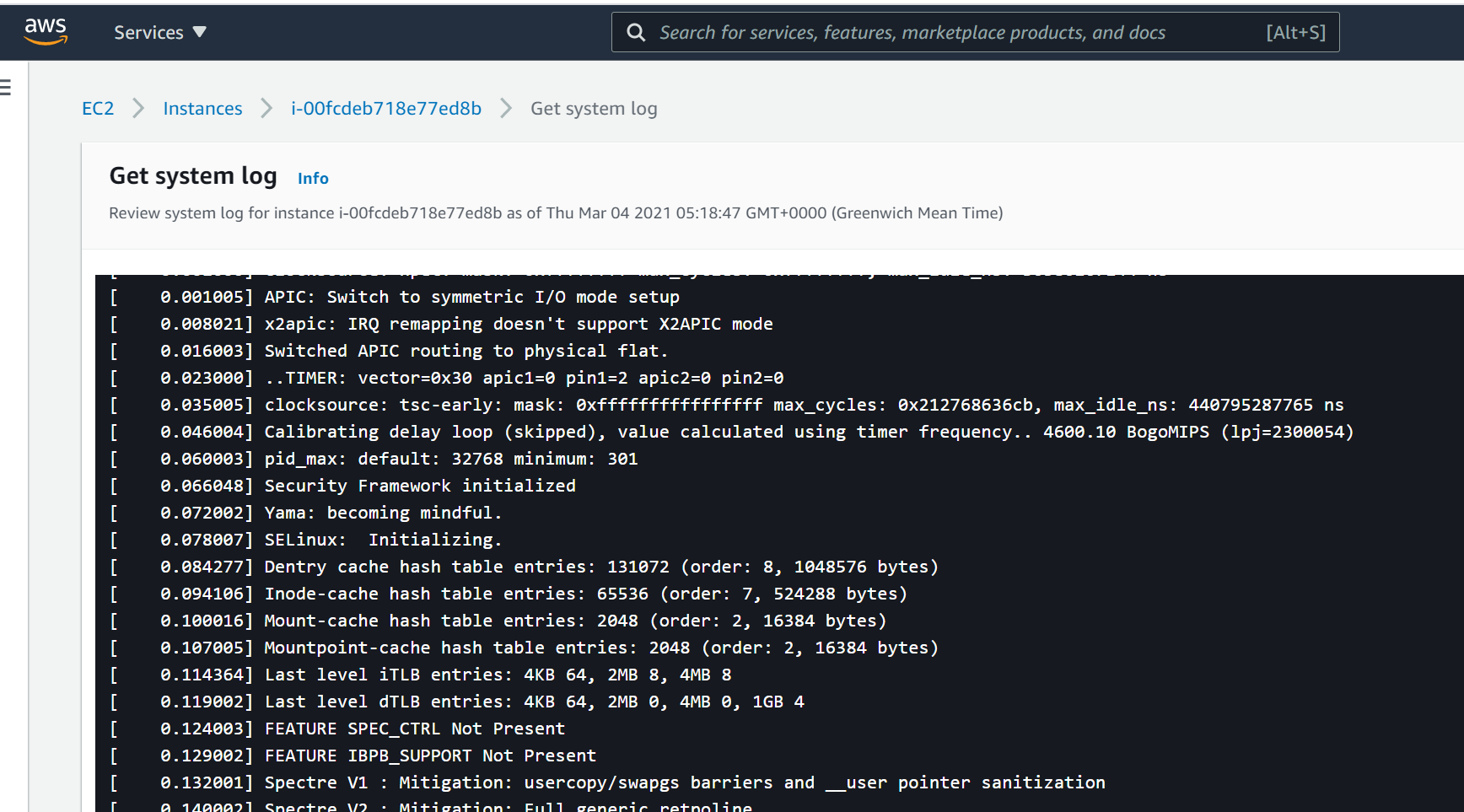
sudo vgscan

sudo vgchange -ay

****

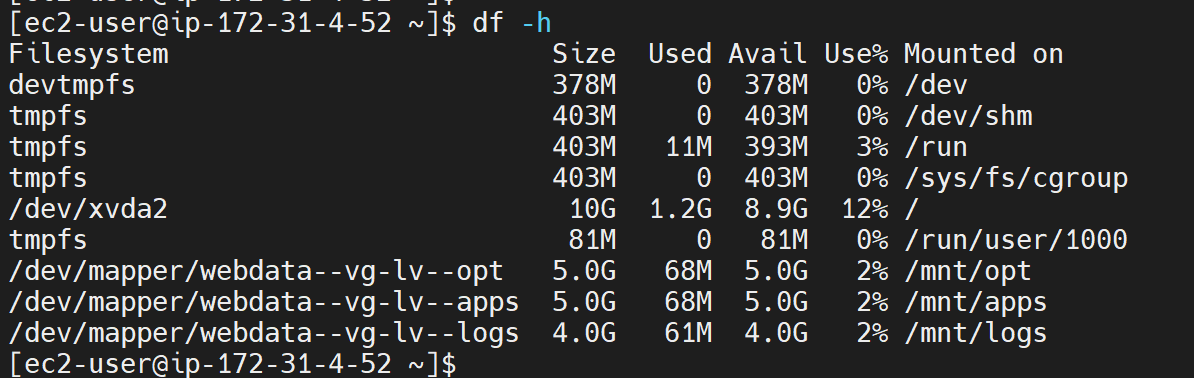
List with: sudo lvdisplay

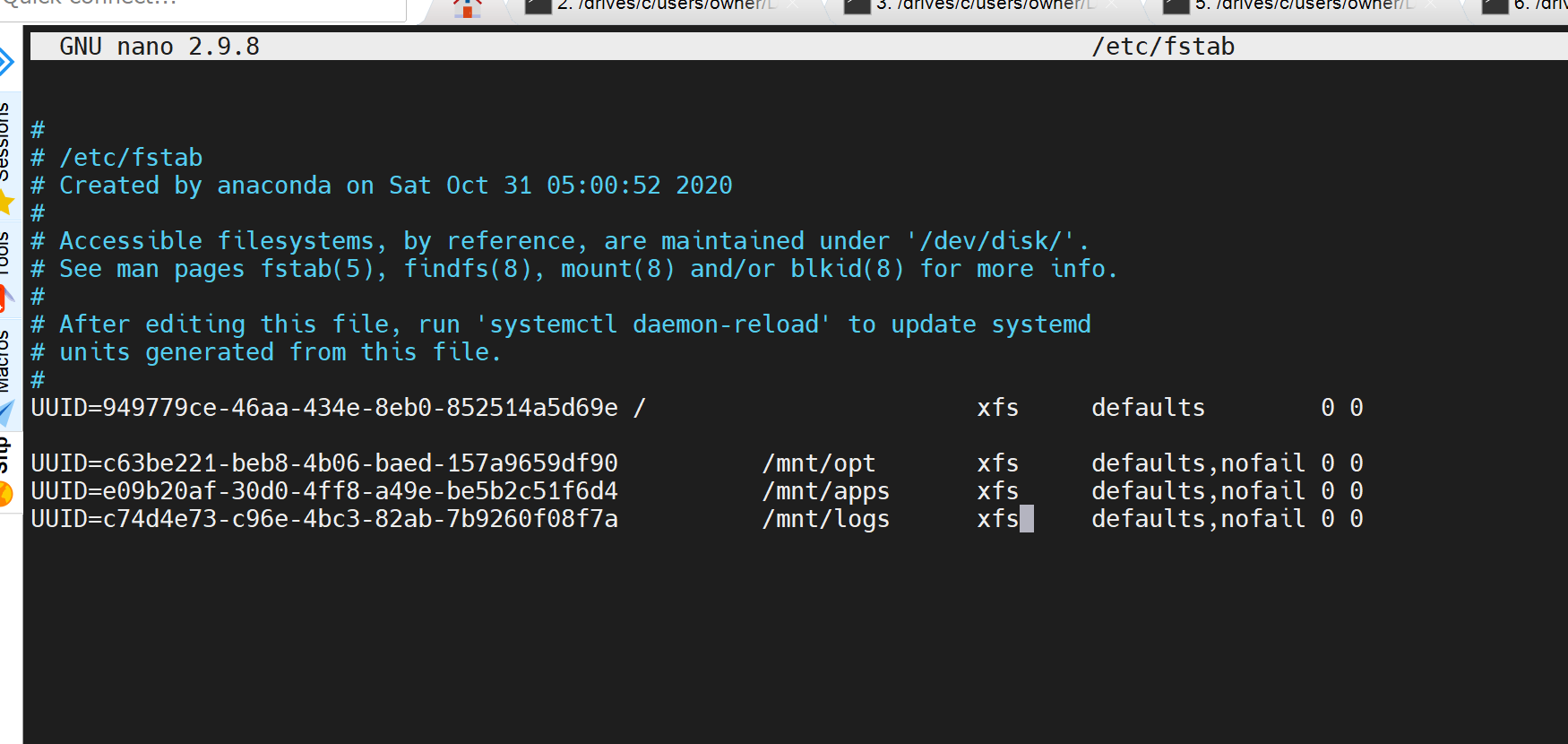
sudo lvs

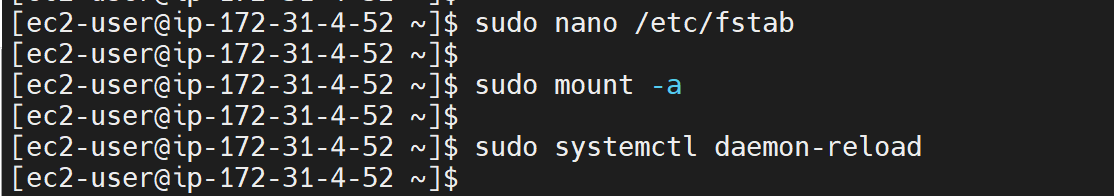
****

Then repeated the below steps:

* Formatting the logical volumes with xfs file system and
* Installing and configuring NFS Server.

****

****

****

**Set up Firewall rules for NFS Services.**

First install the firewall package: sudo yum install firewalld

sudo systemctl start firewalld

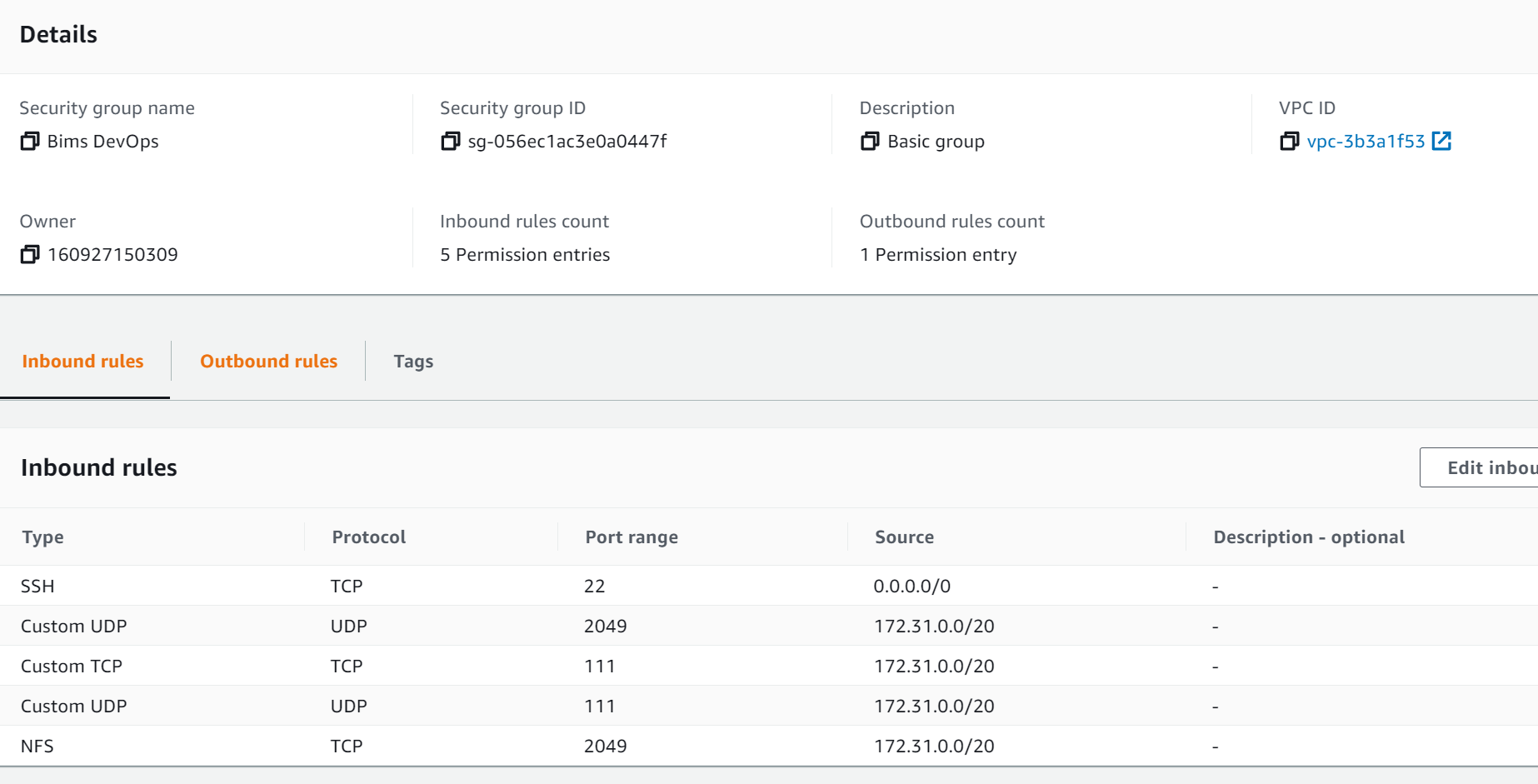
sudo systemctl enable firewalld

Check which port is used by NFS and open it using Security Groups (add new Inbound Rule)

rpcinfo -p | grep nfs



**Important note:** In order for NFS server to be accessible from your client, you must also open following ports: TCP 111, UDP 111, UDP 2049



Alternatively, this can also be done through the terminal command line:

sudo firewall-cmd --permanent --zone public --add-service mountd

sudo firewall-cmd --permanent --zone public --add-service rpc-bind

sudo firewall-cmd --permanent --zone public --add-service nfs --permanent

sudo firewall-cmd --permanent --zone public --add-source=172.31.0.0/20 --permanent

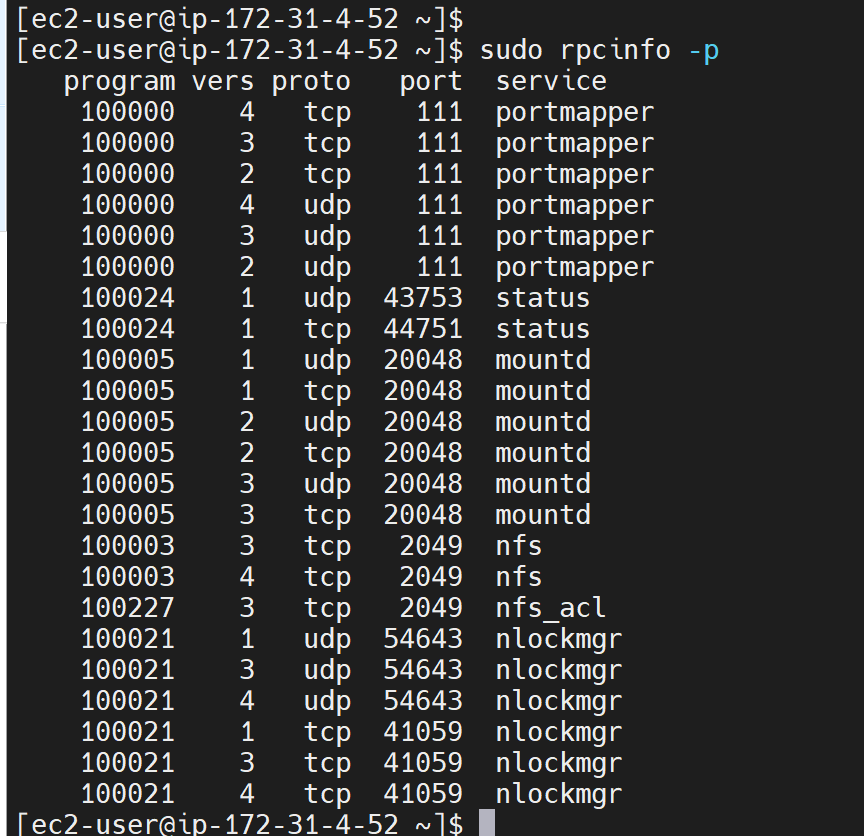
sudo firewall-cmd --permanent --zone public --add-port=2049/tcp

sudo firewall-cmd --permanent --zone public --add-port=2049/udp

sudo firewall-cmd --reload

Confirm that the ports have been opened.

sudo rpcinfo -p



This command sudo ‘rpcinfo -p’ will list out specific ports based on the **RPC** (Remote procedure calls extend the capabilities of conventional procedure calls across a network and are essential in the development of distributed systems.) tool. head over to the AWS security group and enable inbound connections on the following ports.

## **Step 3 — Configure the database server**

The MySQL Database simply put, is a storage place for data that will be sent in by application end users. It helps to deploy cloud-native applications.

The steps involved in installing and configuring database include the below:

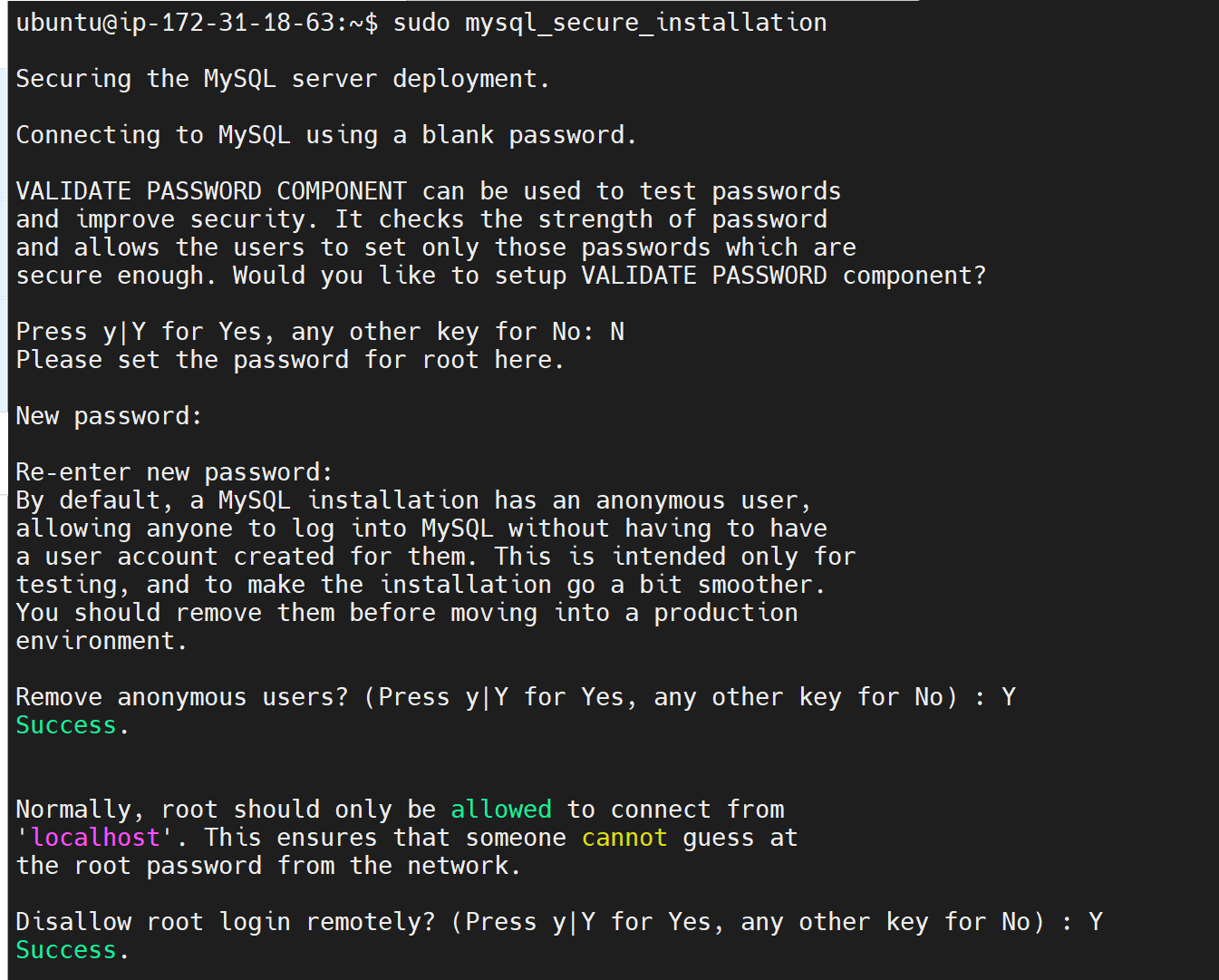
1. Launch an Ubuntu instance in AWS
2. Install MySQL server: sudo apt install mysql-server -y

sudo systemctl start mysql

sudo systemctl enable mysql

1. Run the security script: sudo mysql\_secure\_installation

This allows the removal of a broad range of insecure default settings and also performs lockdown access to your database system. This script comes pre-installed with MySQL.



You will be asked if you want to configure a VALIDATE PASSWORD PLUGIN database feature.

**Note**: This feature is something of a judgment call. Once enabled, passwords which do not match specified criteria will be rejected by MySQL with an error. It is safe to leave validation disabled; however, standard practice is to always use strong, unique passwords for database credentials.

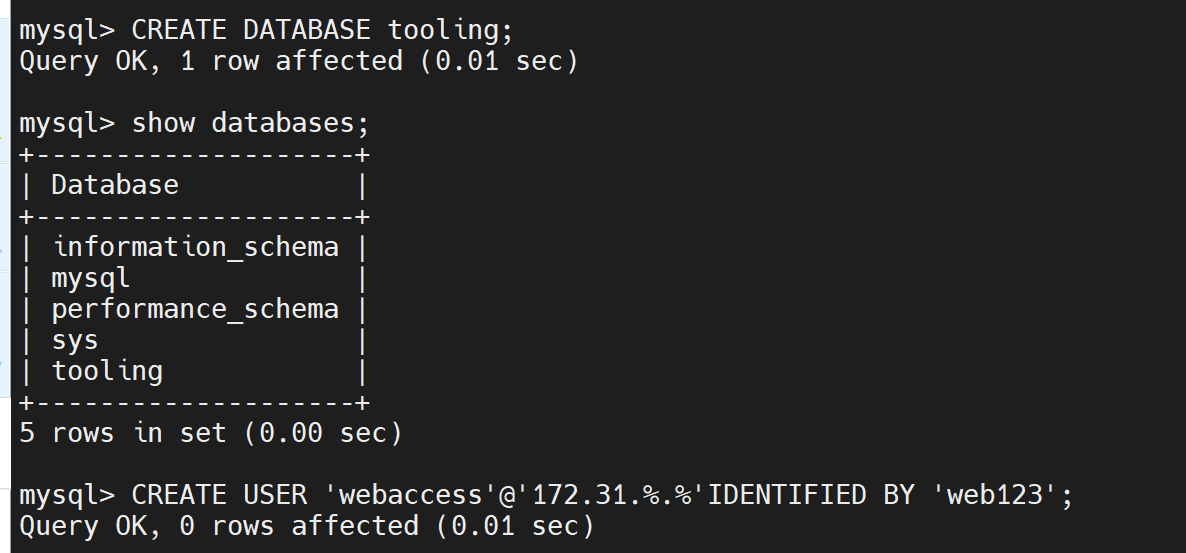
1. Create a database and name it tooling

mysql> CREATE DATABASE tooling;

msql> Show databases;

1. Create a new user account called ‘webaccess’ that will only connect to the remote host. On the database server, start the mysql console using the sudo mysql command then enter the following into the console;

mysql> CREATE USER 'webaccess'@'172.31.%.%'IDENTIFIED BY 'web123';



1. Confirm user has been created:

mysql> select user, host from mysql.user

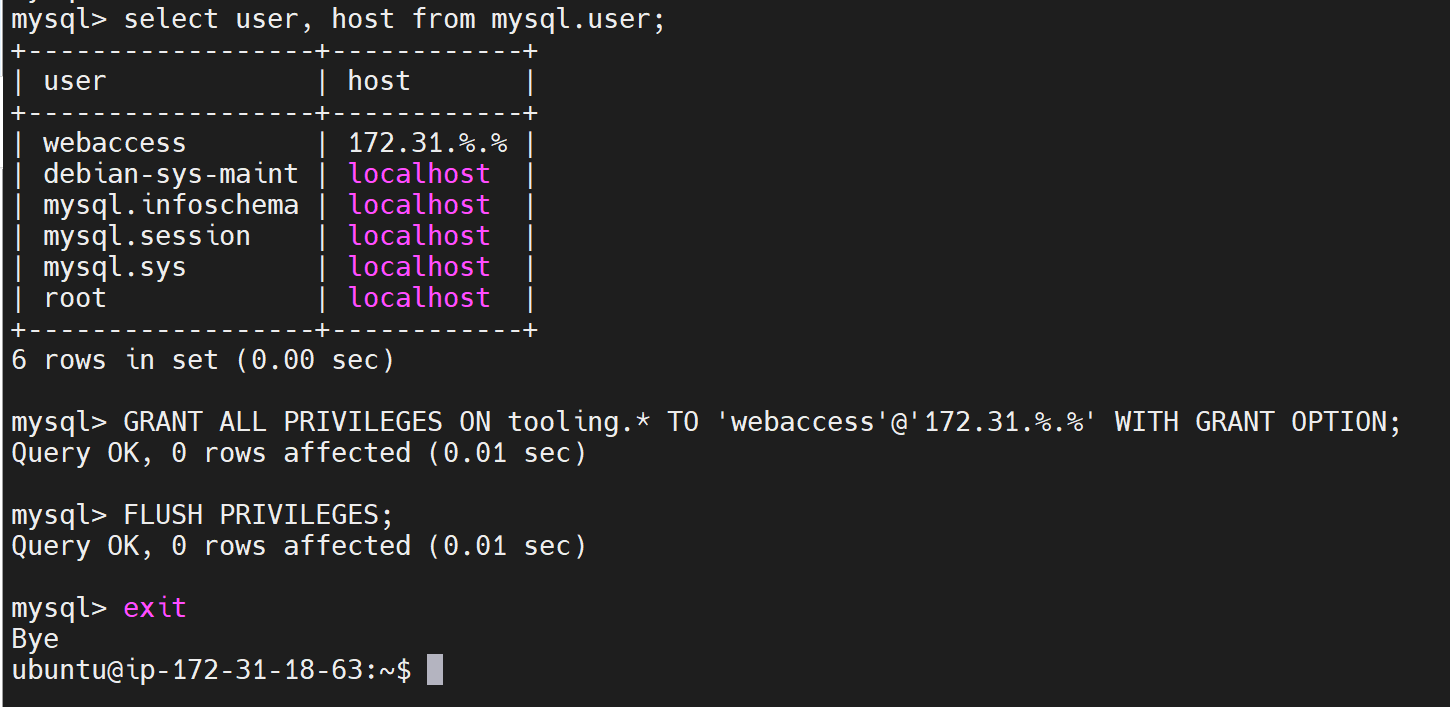
1. Grant privileges to ‘webacces’ user on ‘tooling’ database created;

GRANT ALL PRIVILEGES ON tooling.\* TO 'webaccess'@'172.31.%.%' WITH GRANT OPTION;

1. It is good practice to run the ‘FLUSH PRIVILEGES’ command:

mysql> FLUSH PRIVILEGES;

1. Exit



1. Configure Mysql file to listen to web servers' IP address and modify the bind address in order to enable access to the database.

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

The pre-configured 127.0.0.1, means that the server will only look

for local connections. Change this directive to reference an external

IP address by setting it to a wildcard IP address: either \*, ::, or

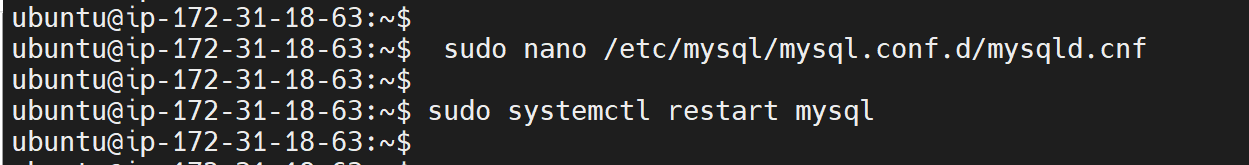
0.0.0.0:



Save and close: CTRLX; Y; ENTER.

k. Restart mysql service in order to make the changes effective@

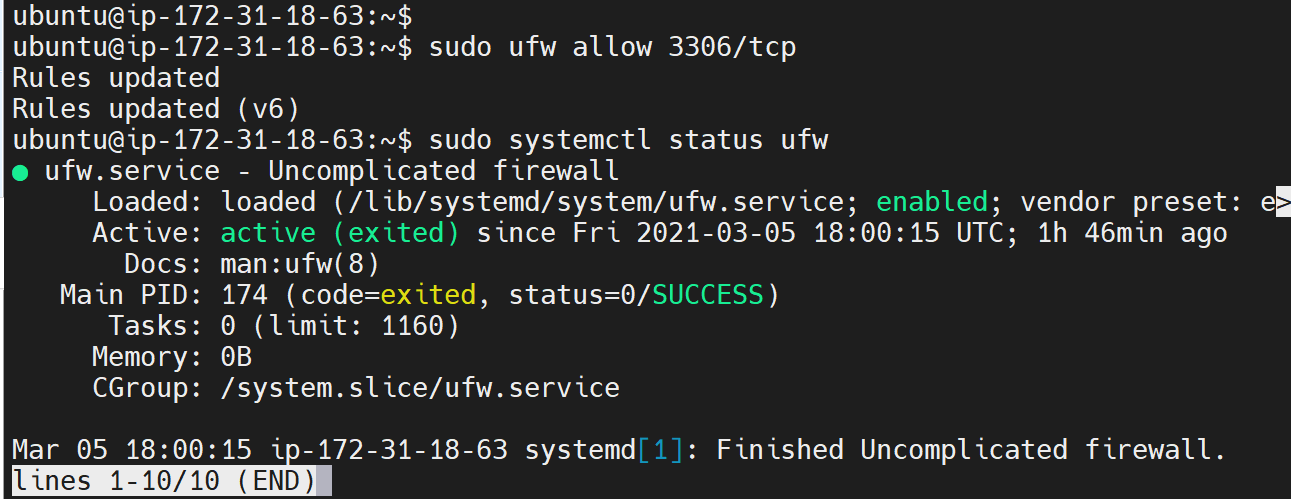
sudo systemctl restart mysql



l. Open firewall to llow incoming traffic on port 3306:

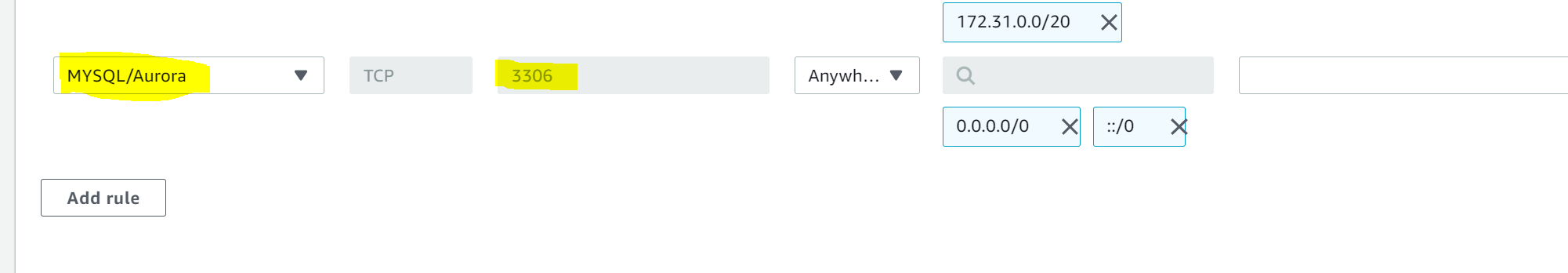
sudo ufw allow 3306/tcp

m. Check status: sudo systemctl status ufw



n. Add an inbound rule on port 3306 for our AWS security group since the

MySQL service listens on that port: MySQL/Aurora



## **Step 4 — Prepare the Web Servers**

IP addresses of the servers:

NFS Server - **172.31.4.52**

Database Server - **172.31.18.63**

Web Server 1 - **172.31.16.162**

Web server 2 - **172.31.16.41**

Web server 3 - **172.31.30.14**

We need to make sure that our Web Servers can serve the same content from shared storage solutions, in our case - NFS Server and MySQL database. You already know that one DB can be accessed for reads and writes by multiple clients. For storing shared files that our Web Servers will use - we will utilize NFS and mount previously created Logical Volume lv-apps to the folder where Apache stores files to be served to the users (/var/www).

This approach will make our Web Servers stateless, which means we will be able to add new ones or remove them whenever we need, and the integrity of the data (in the database and on NFS) will be preserved.

You can set up one or more Web Servers and point to the same NFS and connect to the same Database Server. During this step we will configure NFS client and deploy a tooling application to our Web Server(s).

**Steps:**

1. Launch a new EC2 instance with RHEL 8 Operating System
2. Install NFS client

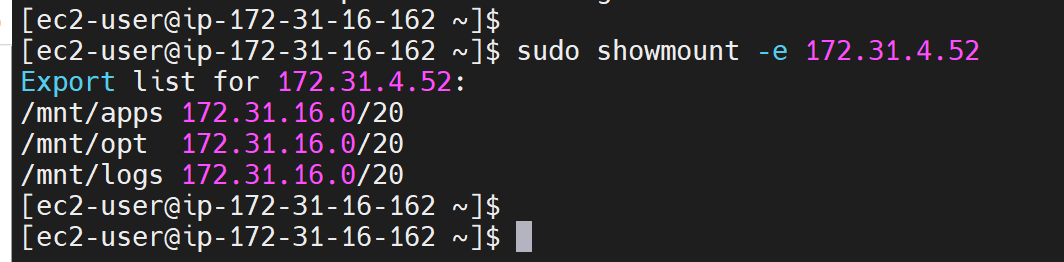
The web servers will make use of the NFS server as a backend storage. Based on this requirement, we shall configure the web servers as NFS clients. On instantiating the three (3) instances on the AWS console, we shall run the following commands on each of the servers.This will make necessary installation of the nfs client services.

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

sudo systemctl start nfs-utils

You can view the nfs share that is to be mounted:

sudo showmount -e 172.31.4.52

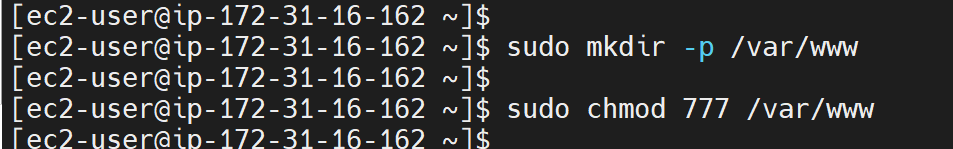


1. Create mount directory on the web server

sudo mkdir -p /var/www

1. Grant necessary permission on the web server

sudo chmod 777 /var/www



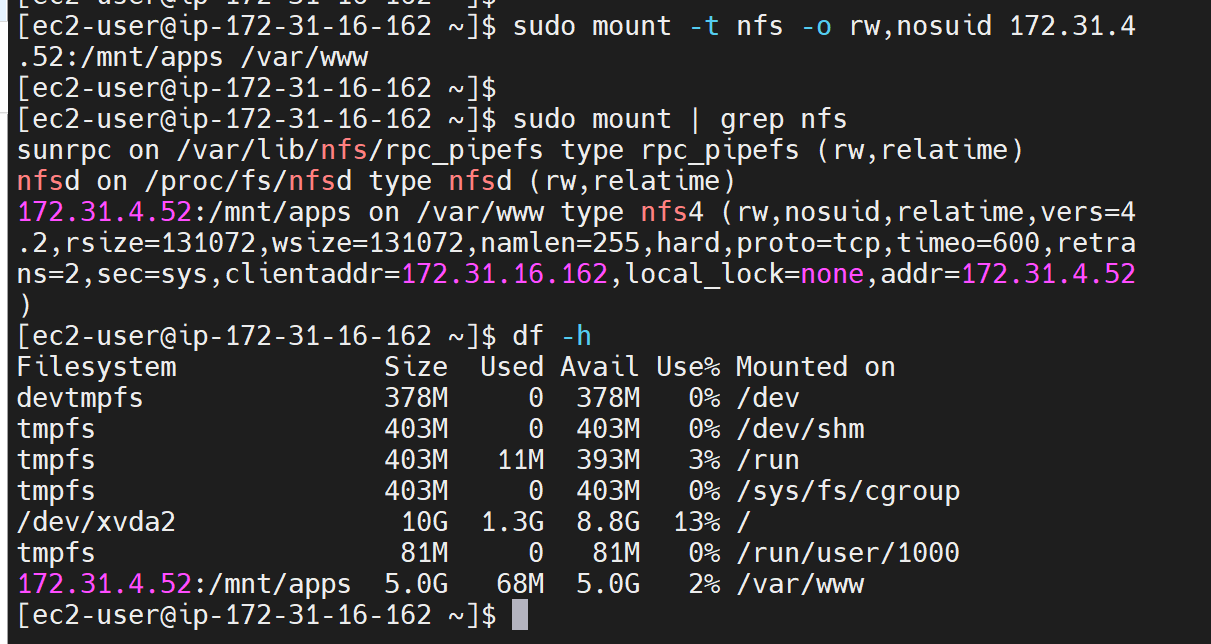
1. Mount the NFS Share on the /var/www directory

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

sudo mount -t nfs -o rw,nosuid 172.31.4.52:/mnt/apps /var/www

1. Verify successful mount

df -h OR sudo mount | grep nfs

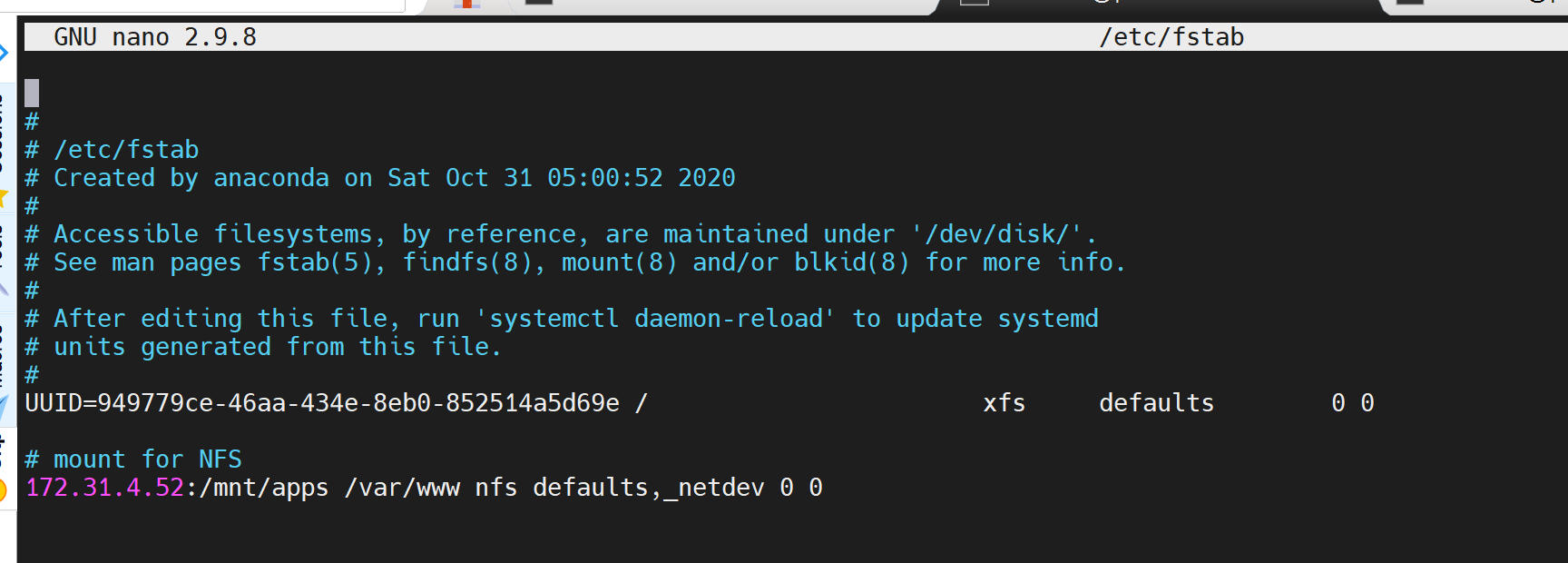


1. Make the mount persistent by updating the /etc/fstab file on all web servers:

sudo nano /etc/fstab

1. Add the below in the file:

172.31.4.52:/mnt/apps /var/www nfs defaults,\_netdev 0 0



From above, the ‘\_netdev’ option will help prevent the server from hanging during a scenario where the nfs share is not present and the web servers are booting up.

Save and close.

1. **Install Apache** on the web servers. This will help to serve the web content:

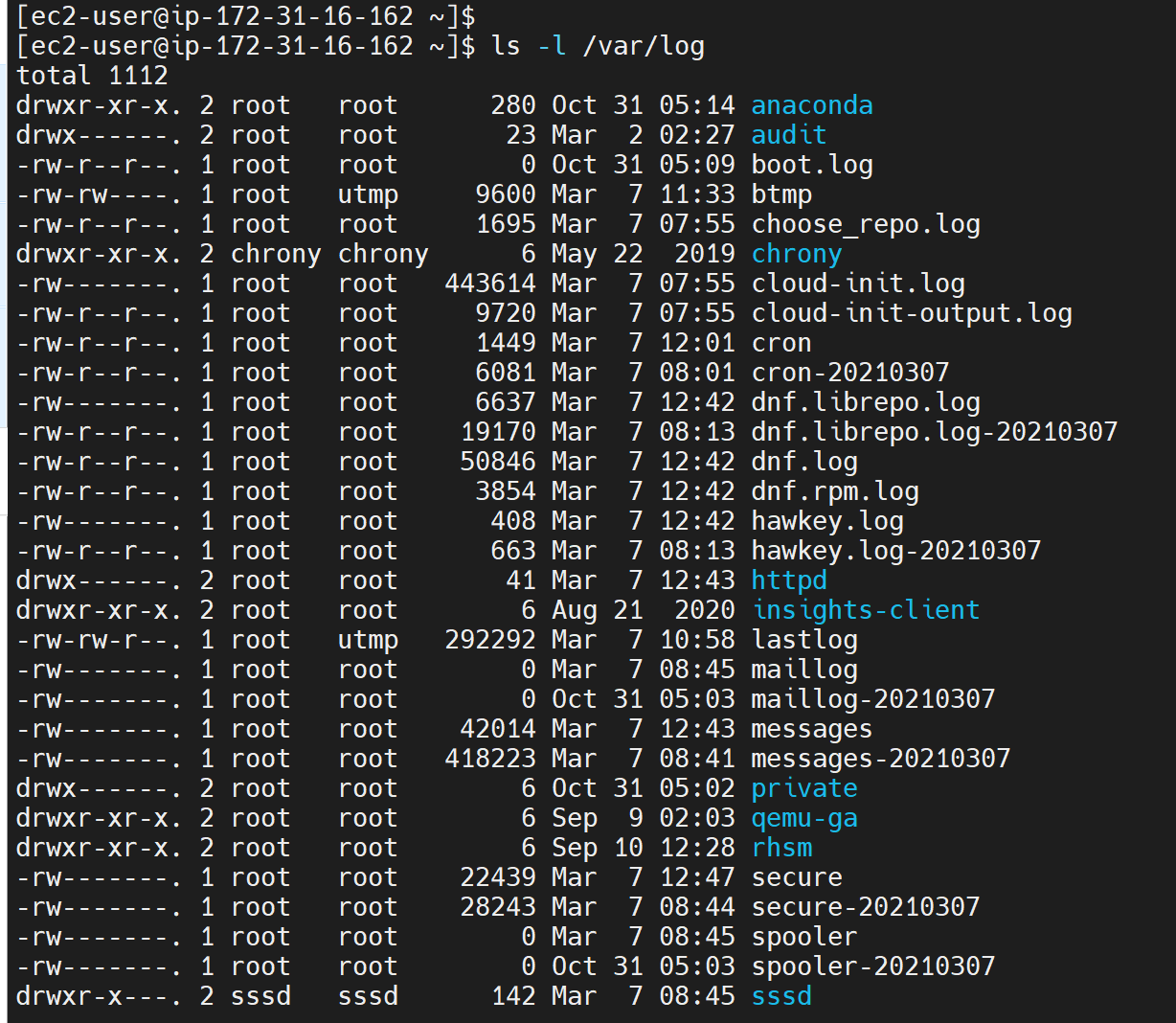
sudo yum install httpd -y

sudo systemctl start httpd

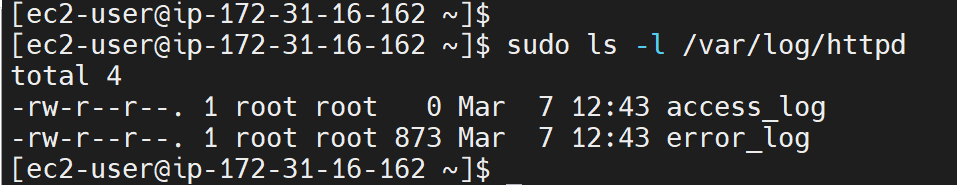
sudo systemctl enable httpd

1. Locate the log folder for Apache on the Web Server and mount it to NFS server’s export for logs.

ls -l /var/log



Log folder for apache: sudo ls -l /var/log/httpd



Create a backup file for log

sudo mkdir -p /home/log/backup

Copy the log file to the backup file

View the file: ls -l /var/log/httpd

Then copy:

sudo cp -R -v /var/log/httpd/access\_log /home/log/backup

sudo cp -R -v /var/log/httpd/error\_log /home/log/backup



\*Please note:Change ownership of the log file. If after mounting, the ownership is changed to apache and mode is changed to 777 as per nfs server, there will be a need to change ownership to 700 as a standard rule and also change back to root:root because httpd is meant for root only.

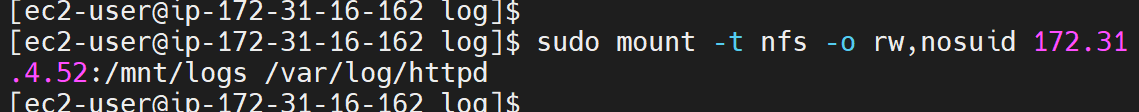
sudo chown -R root:root /var/log/httpd

sudo chmod 700 /var/log/httpd

Mount /var/log:

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

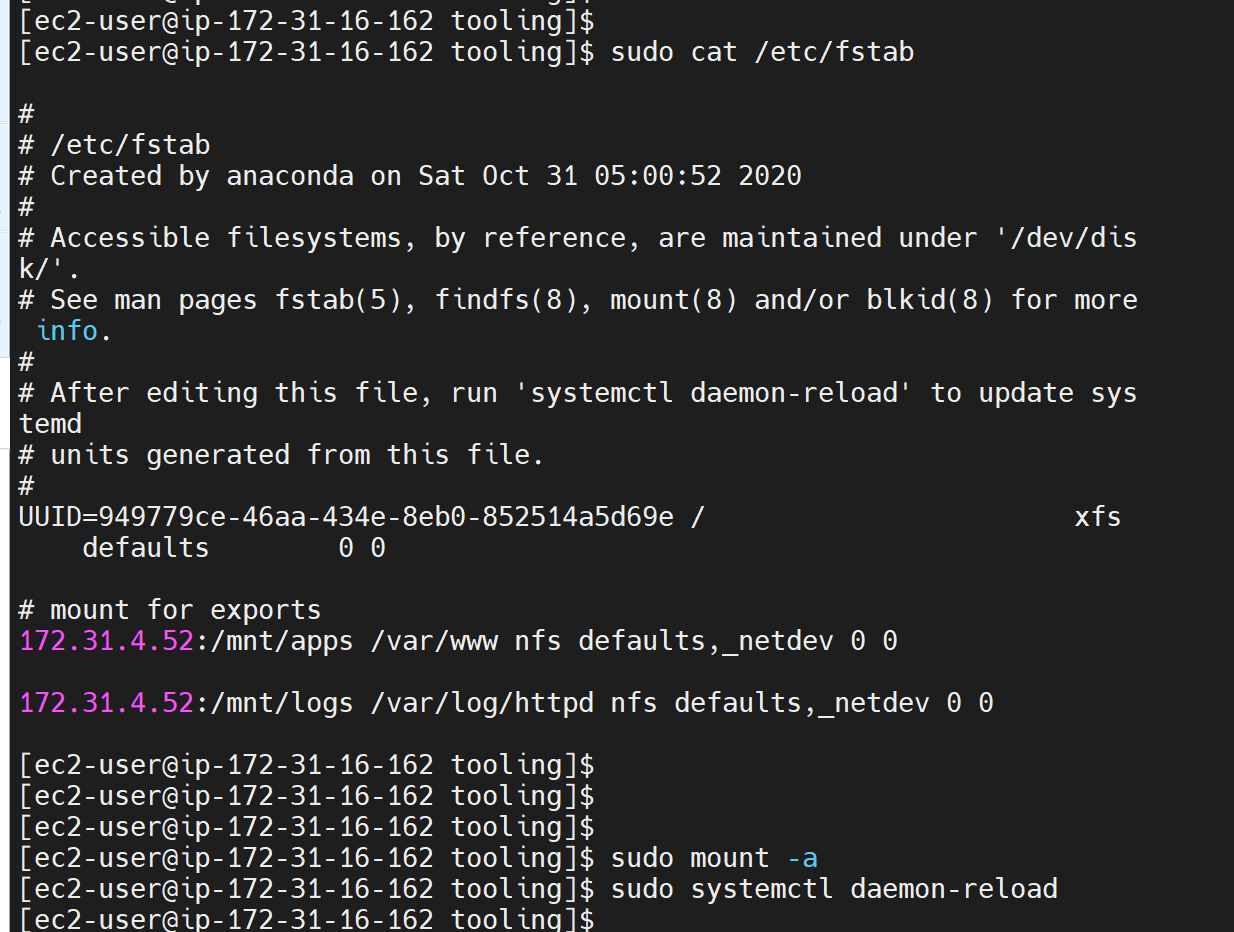
sudo mount -t nfs -o rw,nosuid 172.31.4.52:/mnt/logs /var/log/httpd



Make the mount persistent by updating the fstab file with the below:

172.31.4.52:/mnt/logs /var/log/httpd nfs defaults,\_netdev 0 0

sudo nano /etc/fstab

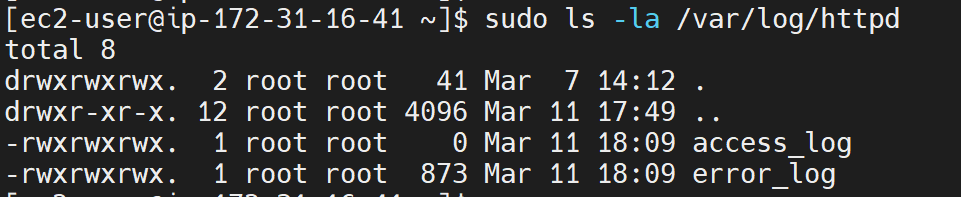


Then copy from the back up back into the /var/log/httpd file after the mount.

sudo cp -R -v /home/log/backup/. /var/log/httpd

View the log file

sudo ls -la /var/log/httpd



**Install PHP**

There is a need to install the latest PHP version. PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information and hand the processed content over to our web server to display.

Right off the bat, you need to enable the EPEL repository on your system. EPEL,

Short for Extra Packages for Enterprise Linux, is an effort from the Fedora team that provides a set of additional packages that are not present by default on RHEL &

CentOS.

* To install **PHP 7**, you have to install and enable **EPEL and Remi repository** on your CentOS 7 system with the commands below.

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

* install yum-utils, a collection of useful programs for managing yum repositories and packages. It has tools that basically extend yum’s default features.It can be used for managing (enabling or disabling) yum repositories as well as packages without any manual configuration and so much more.

sudo yum install yum-utils

* After the successful installation of yum-utils and Remi-packages, search for the PHP modules which are available for download by running the command

sudo dnf module list php

* The output indicates that the currently installed version of PHP is PHP 7.2. To install the newer release, PHP 7.4, reset the PHP modules.

sudo dnf module reset php

* Having reset the PHP modules, enable the PHP 7.4 module by running

sudo dnf module enable php:remi-7.4

* Finally, install **PHP, PHP-FPM** (FastCGI Process Manager) and associated PHP modules using the command.

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

* Verify the version installed

php -v

* Now PHP 7.4 has now been installed, we will now start and enable PHP-FPM on boot-up.

sudo systemctl start php-fpm

sudo systemctl enable php-fpm

* Check status

sudo systemctl status php-fpm

* double check the installed version of PHP on your system.

sudo php -v

* To instruct **SELinux** to allow **Apache** to execute the **PHP c**ode via PHP-FPM run

setsebool -P httpd\_execmem 1

sudo setenforce permissive

* Finally, restart Apache web server for PHP to work with Apache web server.

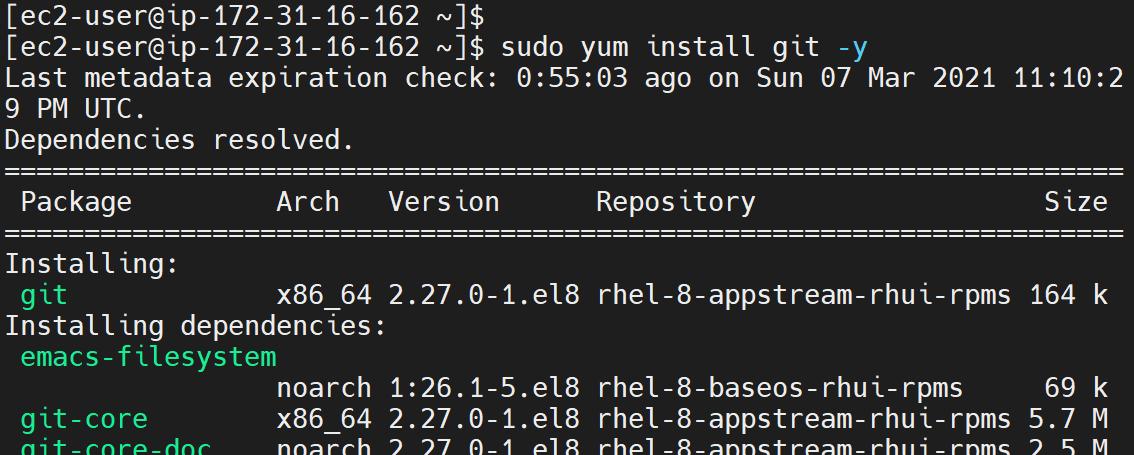
sudo systemctl restart httpd

Fork the tooling source code from [Darey.io Github Account](https://github.com/darey-io/tooling.git) to your Github account. Deploy the tooling website’s code to the Web Server. Ensure that the **html** folder from the repository is deployed to /var/www/html

**Install Git:**

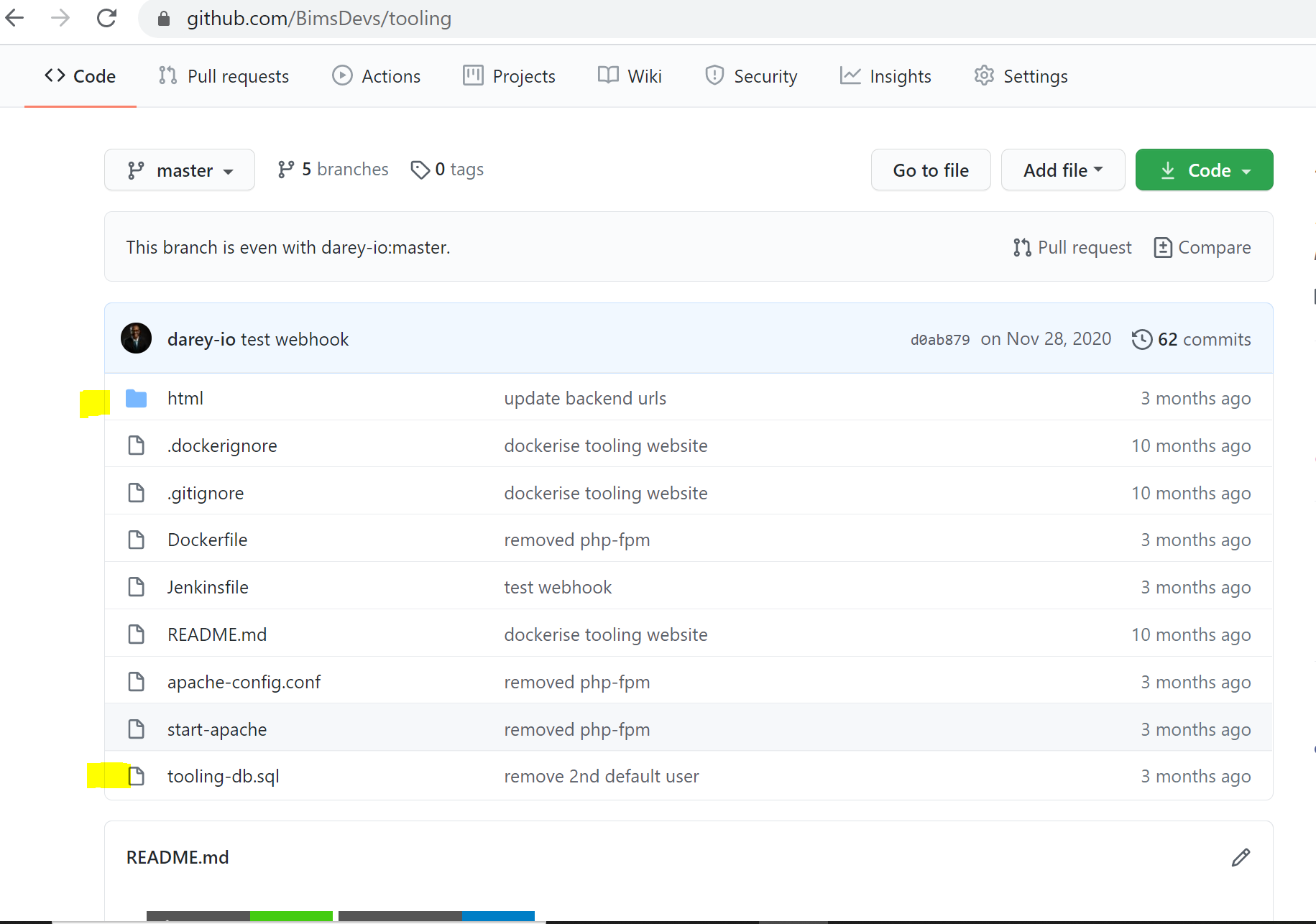
To get access to the codebase, we’ll need to install git. Run the following in the terminal to install git (this should be done on the nfs server)

sudo yum install git -y



Clone the repository for the tooling website;

git clone <https://github.com/BimsDevs/tooling.git>



cd into the tooling directory and recursively copy the contents of the html directory into /var/www/html

cd tooling ; sudo cp -R html/. /var/www/html

Verify successful copying:

sudo ls -la /var/www/html

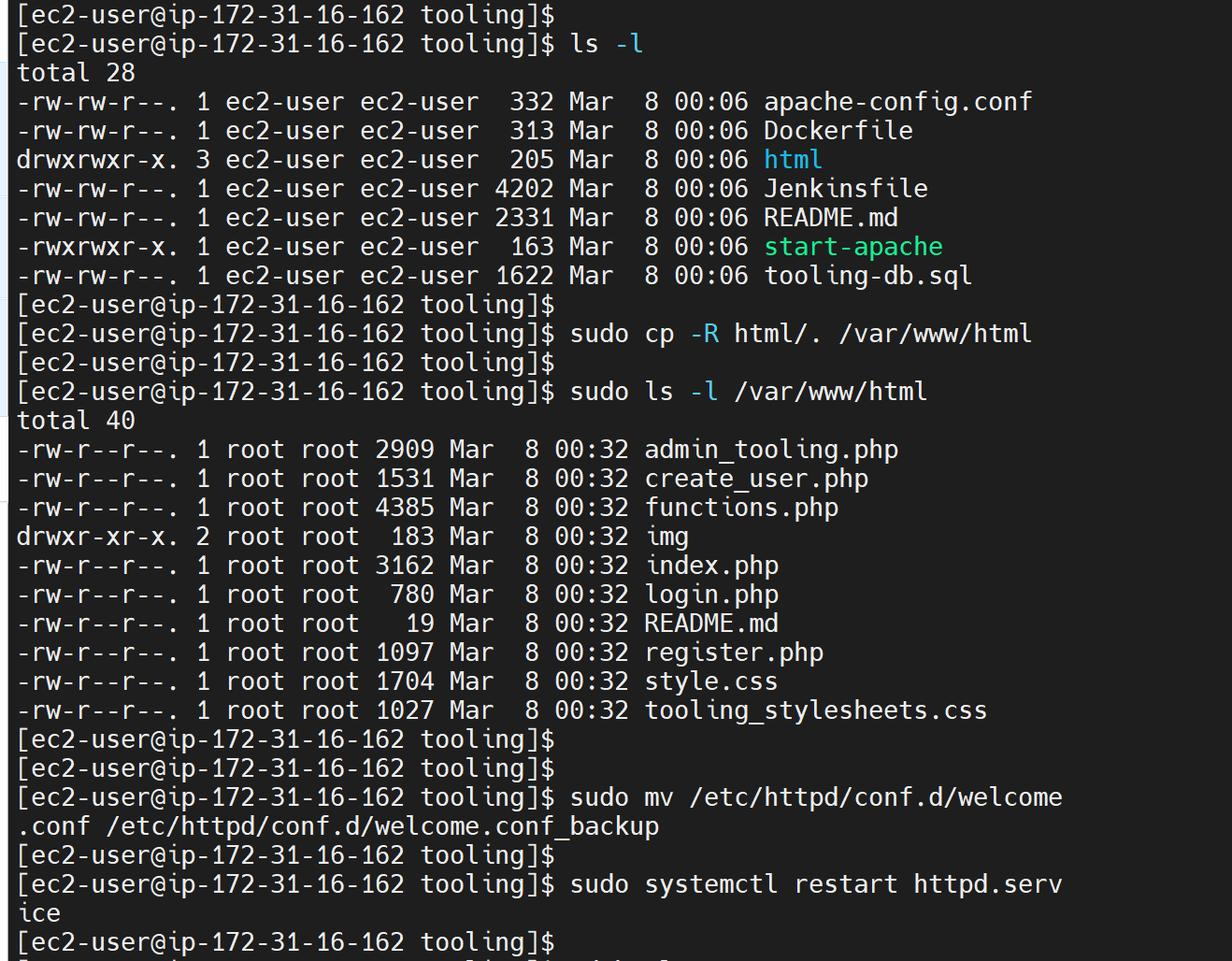
Change directory to /var/www/html

cd /var/www/html

Rename the Apache default page:

sudo mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf\_backup

Restart httpd.service: sudo systemctl restart httpd.service

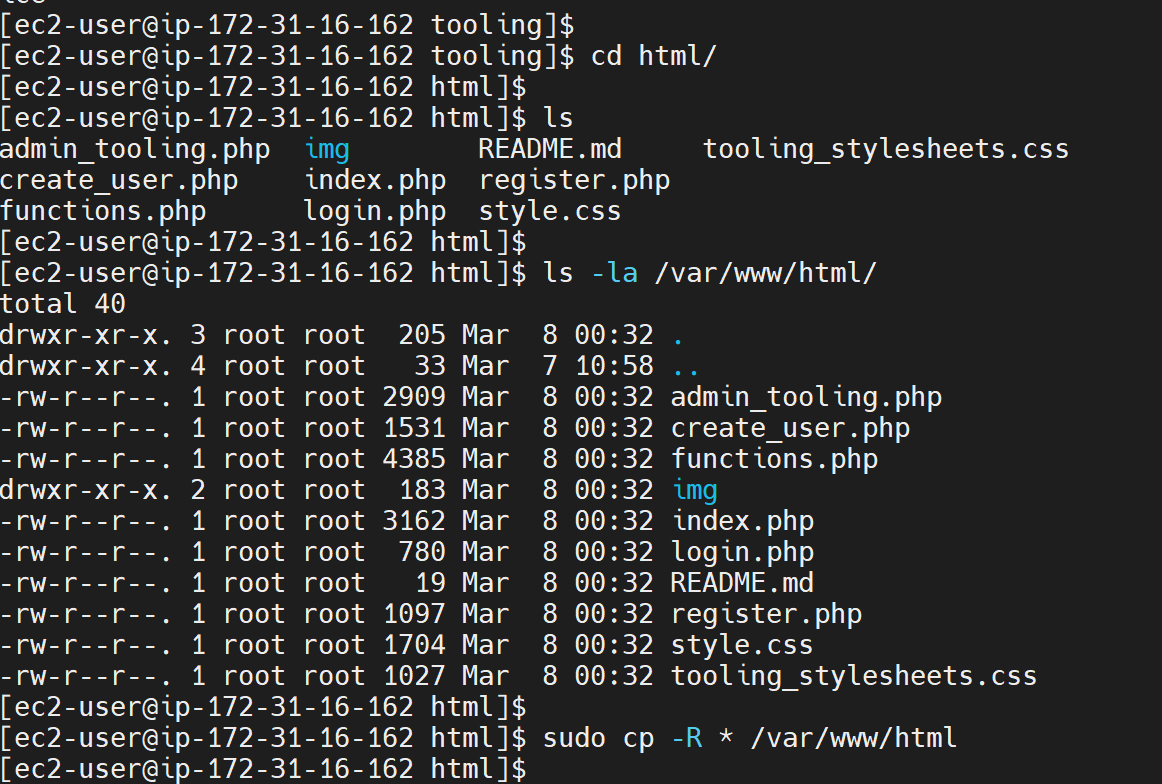


cd into html: cd /html

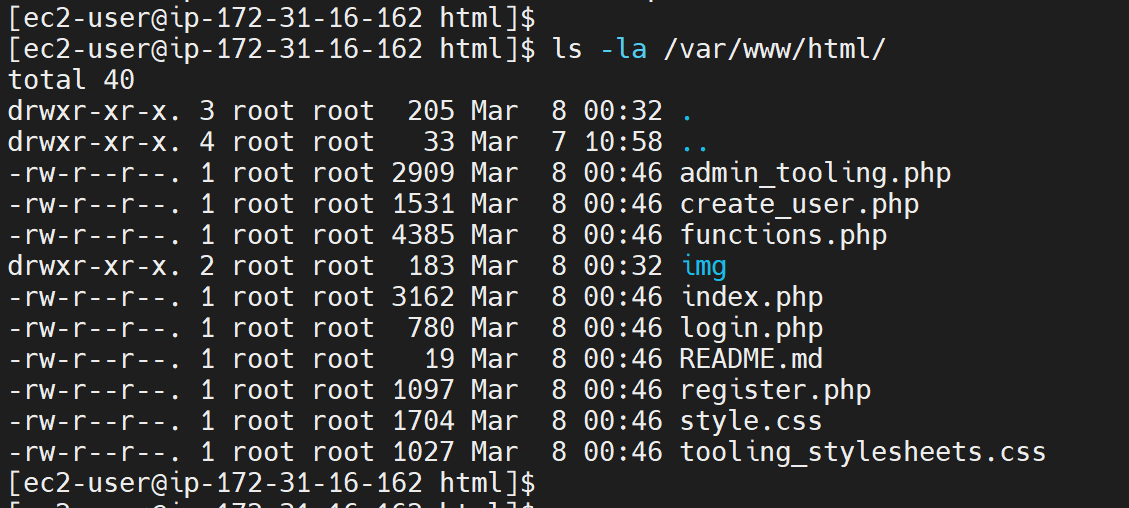
Deploy the entire html folder into /var/www/html;

cd into html: cd /html

sudo cp -R \* /var/www/html



List the files in the html folder to confirm its contents also has functions.php. ls -la



**Configure Firewall rules on all web servers:**

Open TCP port 80 on aws for all the Web Servers by selecting http in the security section of aws console /instance. We can also allow traffic from MySQL by selecting MySQL Aurora for port 3306. This will allow traffic from http and MySQL.

We can run the command below w=in the terminal.

sudo firewall-cmd --permanent --add-service=https

sudo firewall-cmd --permanent --add-service=http

sudo firewall-cmd --permanent --add-service=mysql

sudo firewall-cmd --reload

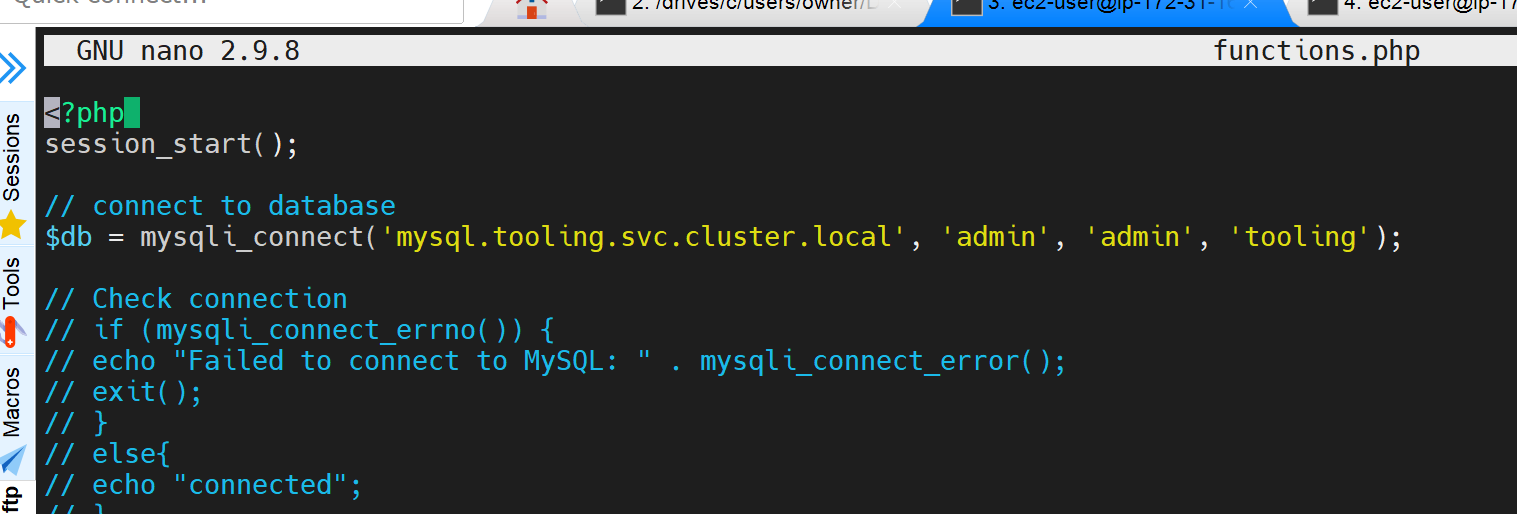
**Configure Apache / website to serve PHP.**

Update the website’s configuration to connect to the database (in functions.php file) and by applying the tooling-db.sql script.

As the Apache default page has already been renamed in the section above, and we have verified that the ‘functions.php’ file exists in the var/www/html directory, then we will need to update the tooling website credentials in the function.php folder.

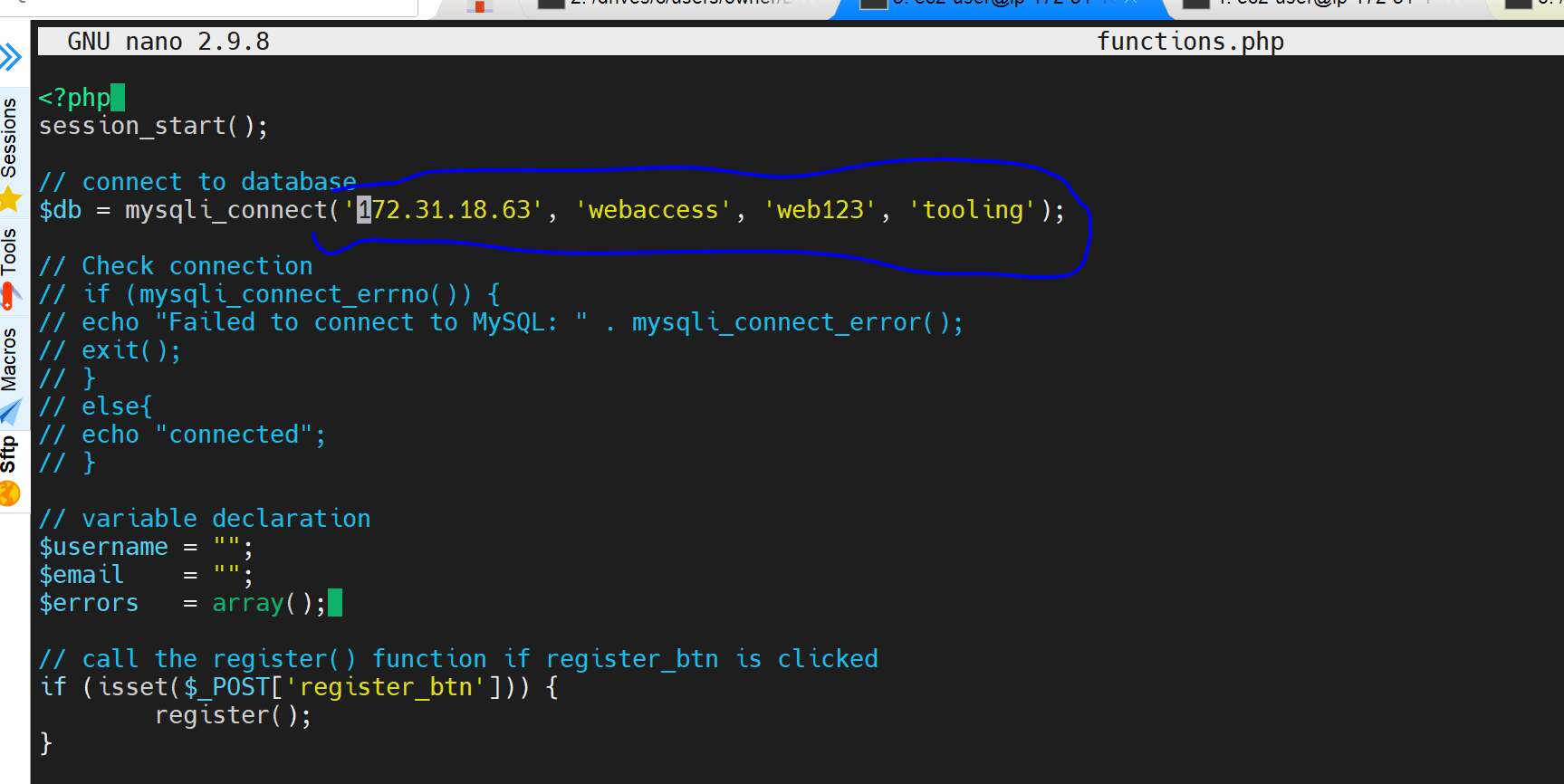
sudo nano functions.php

Pre-update:



Post update: with the database details and IP - This will set the database credentials to be used when connecting to the tooling database.

$db = mysqli\_connect('172.31.18.63', 'webaccess', 'web123', 'tooling');



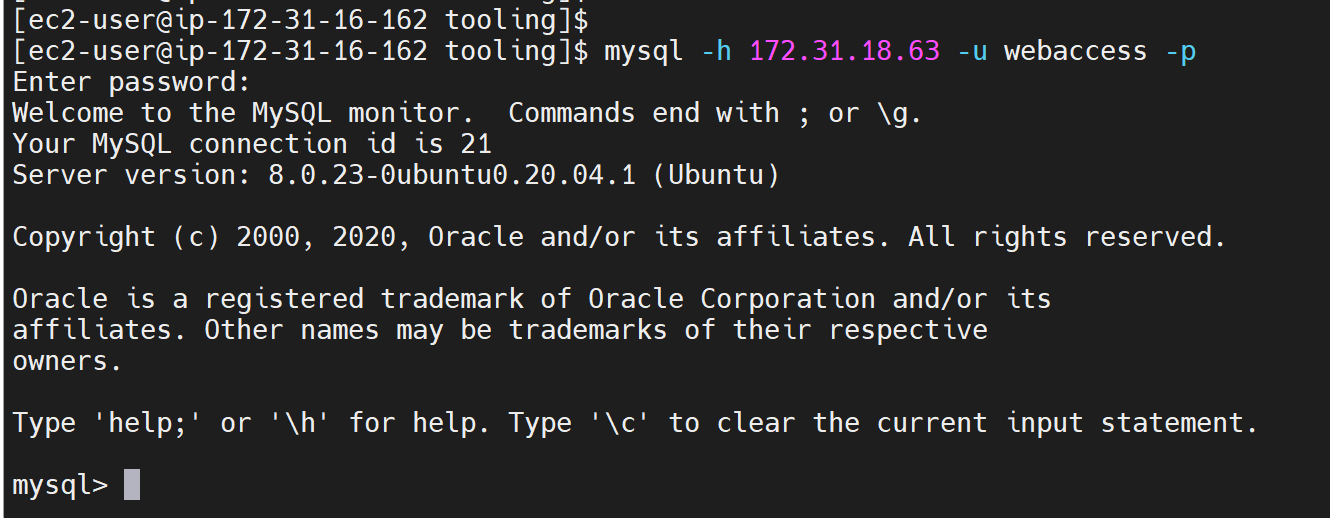
**Install MySQL on the web server**

sudo yum install mysql -y

mysql\_secure\_installation

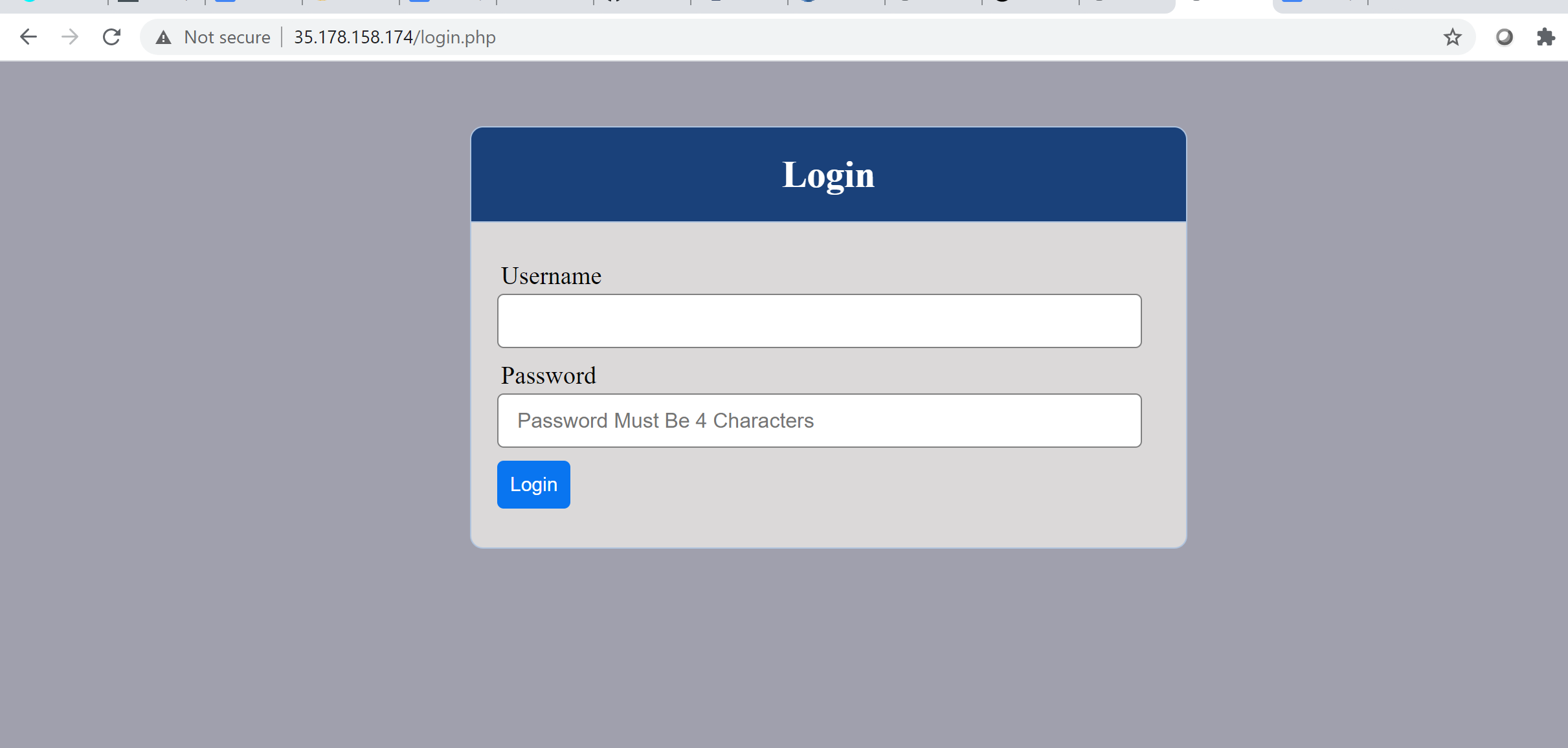
Verify that the web server can access the database:

mysql -h 172.31.18.63 -u webaccess -p



Restart the httpd service: sudo systemctl restart httpd.service

Then test the tooling website on the browser by typing the IP address of the web server on the browser: http://35.178.158.174/login.php



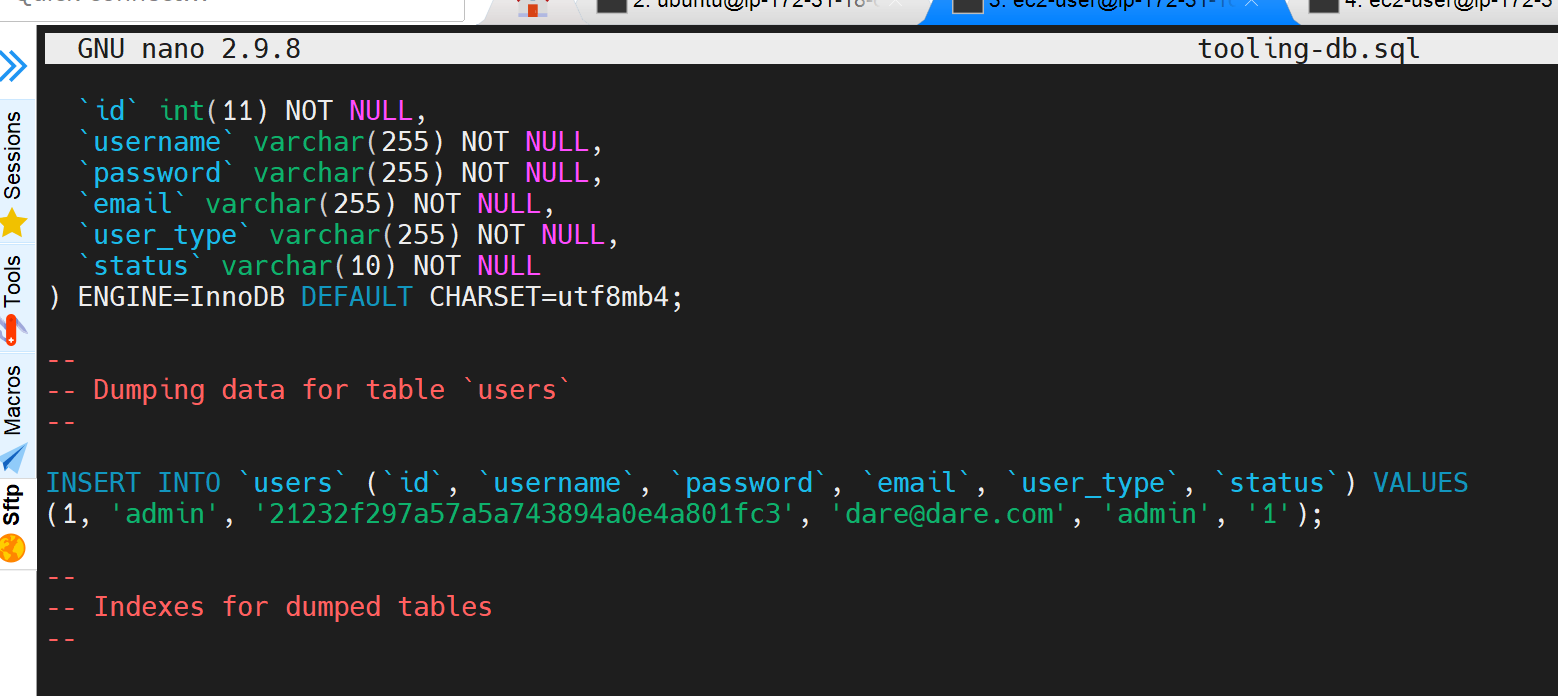
**Deploy the tooling-db.sql into the database.**

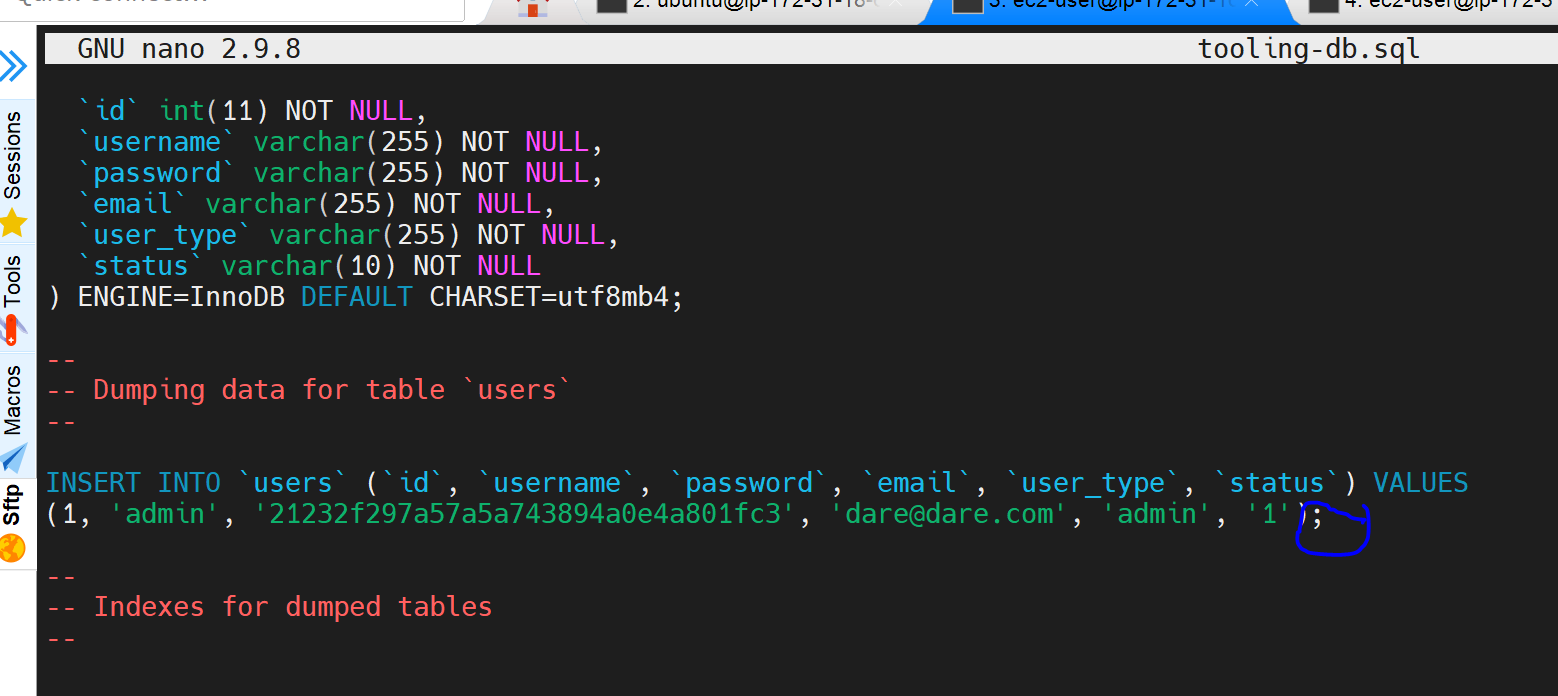
In order to apply the tooling-db.sql, run the below command:

cd tooling

sudo nano tooling-db.sql

Move into the file and add a column at the end of the ‘INSERT INTO’ line.



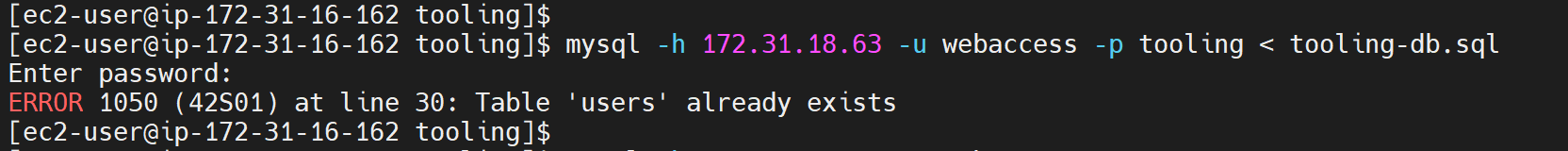


Then run below command so as to export tooling-db into our database:

$ mysql -h 172.31.18.63 -u webaccess -p tooling < tooling-db.sql

Input the tooling database password

Could not access as the user already exists.



Steps taken:

Logged in to MySQL of the database server

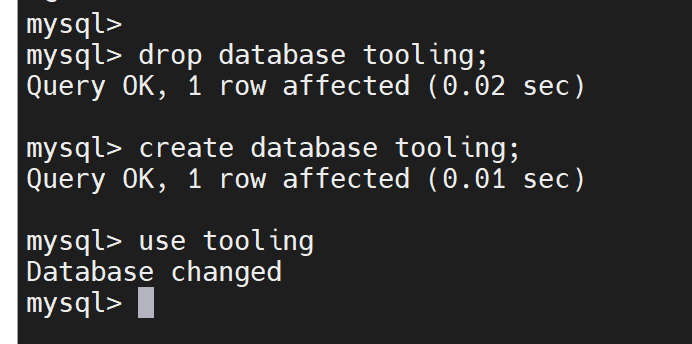
Dropped the existing database ‘tooling’ and then created a new one in

order to remove the conflicts.

mysql> drop database tooling;

mysql> create database tooling;

mysql> use tooling;



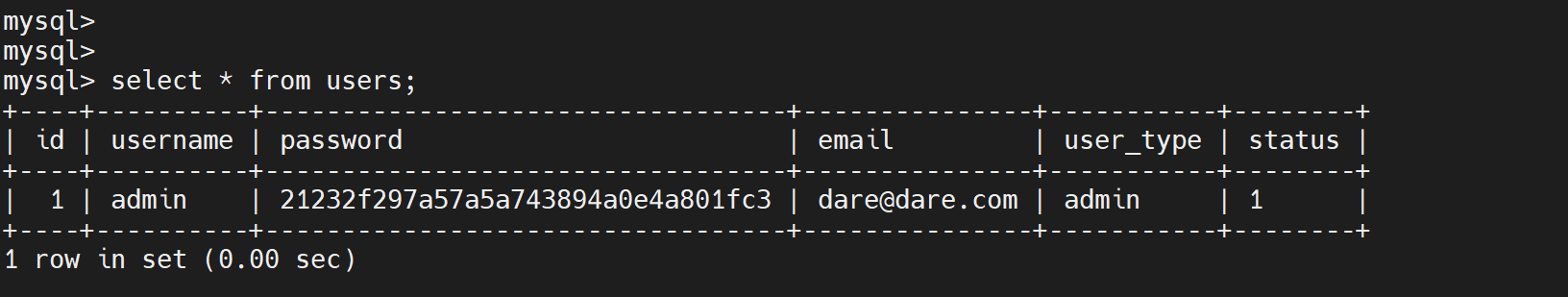
On webserver, try to deploy the tooling-db.sql again

$ mysql -h 172.31.18.63 -u webaccess -p tooling < tooling-db.sql



The above output confirms that the tooling-db.sql has been successfully deployed into the database.

mysql> select \* from users;



Finally, open the website in your browser

http://<Web-Server-Public-IP-Address-or-Public-DNS-Name>/index.php and make sure you can login into the website with admin user.

Web server 1 [http://35.177.62.107](http://35.177.62.107/)

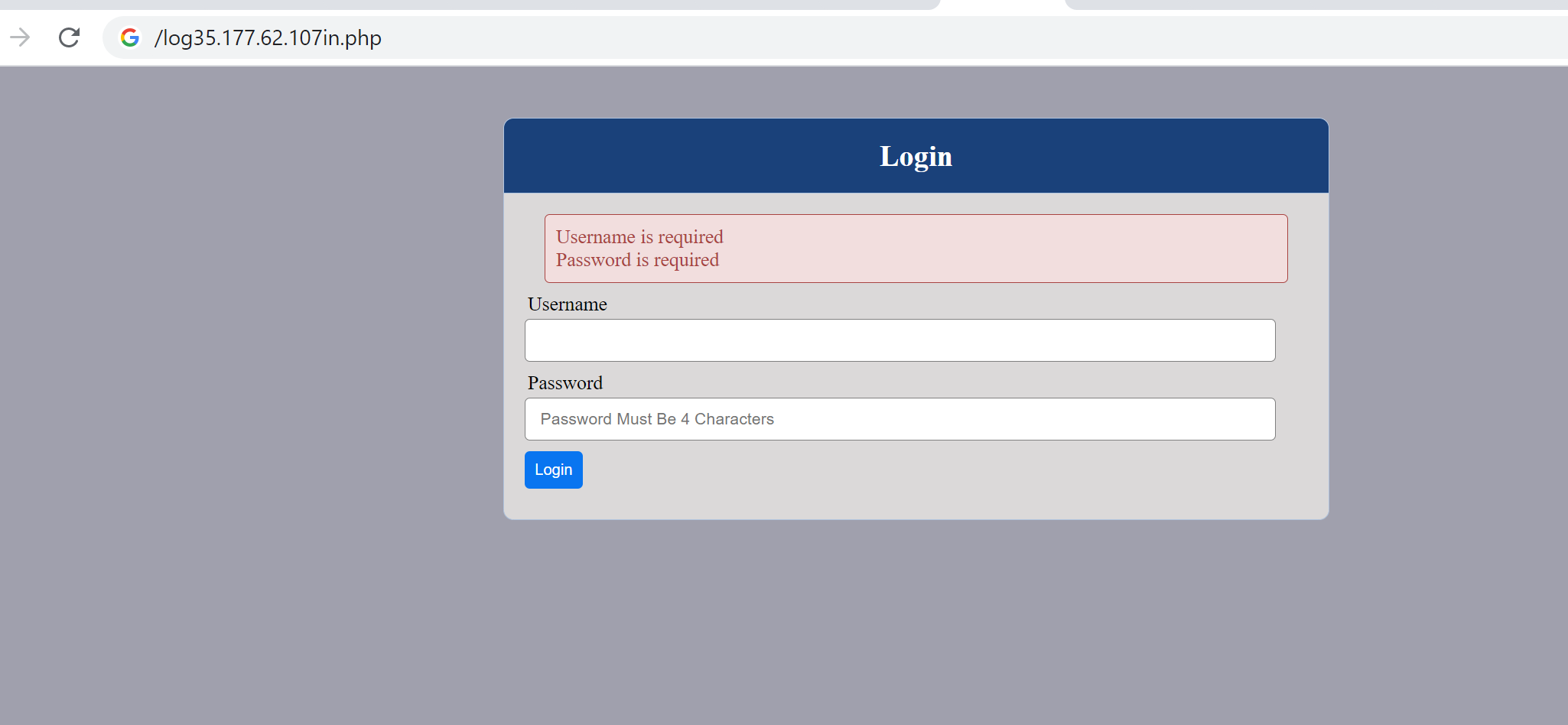
Web server 2 http://3.8.6.231/

Login with username: admin

Password: admin

Blocker:

Unable to log in with the username and password details. This was an issue with the PHP installation as an older version (PHP 7.2) was installed rather than a newer version- PHP 7.4.



**Steps taken to troubleshoot:**

Used error log command to identify where issue was:

sudo tail -f /var/log/httpd/error\_log

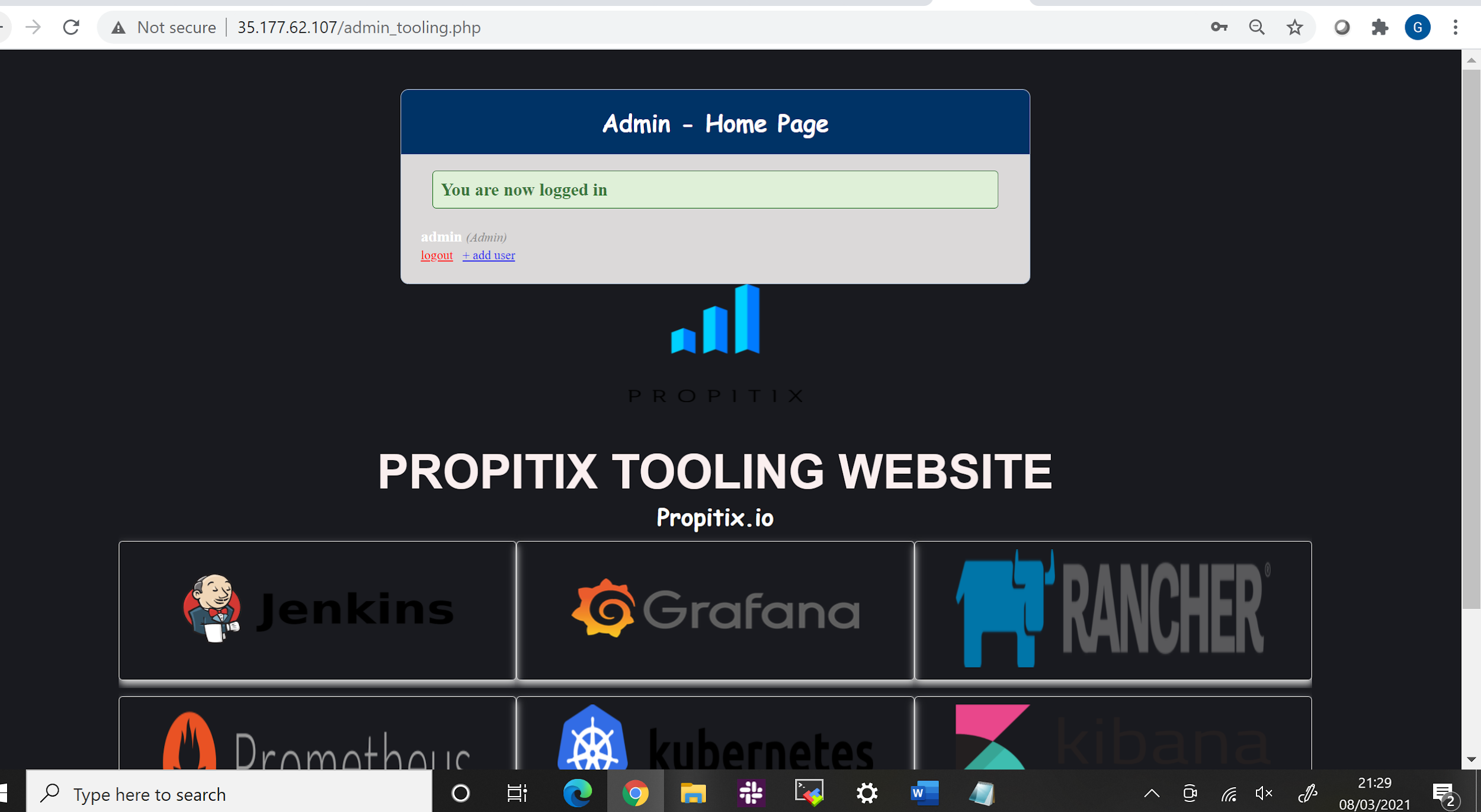
Uninstalled the older version by running the command: yum -y remove php\*

Then installed the latest version PHP 7.4 as done under section PHP above.

sudo dnf module install -y php:7.4

sudo dnf install -y php-{mysqlnd,xml,xmlrpc,curl,gd,mbstring,opcache,soap,zip}

sudo systemctl restart httpd



Credits:

Darey.io

Network-attached Storage

(NAS): <https://en.wikipedia.org/wiki/Network-attached_storage>

Storage Area Network (SAN): <https://en.wikipedia.org/wiki/Storage_area_network>

Block Level Storage: <https://en.wikipedia.org/wiki/Block-level_storage>

Object Storage: <https://en.wikipedia.org/wiki/Object_storage>

AWS Storage Services: <https://dzone.com/articles/confused-by-aws-storage-options-s3-ebs-amp-efs-explained>

PHP 7.4: <https://www.tecmint.com/install-lamp-on-centos-8>

NFS: <https://computingforgeeks.com/install-and-configure-nfs-server-on-centos-rhel/>

Subnet/CIDR:<https://www.digitalocean.com/community/tutorials/understanding-ip-addresses-subnets-and-cidr-notation-for-networking>

Firewall Man Page: <https://firewalld.org/documentation/man-pages/firewall-cmd.html>

NFS Server and Client: <https://www.linuxtechi.com/setup-nfs-server-on-centos-8-rhel-8/>

Subnet Calculator: <https://www.adminsub.net/ipv4-subnet-calculator/10.25.0.0/25>

Subnet Sheet: <https://www.aelius.com/njh/subnet_sheet.html>

NFS Mount: <https://www.poftut.com/how-to-mount-nfs-share-in-linux-and-windows/>

<https://www.codecademy.com/courses/deploy-a-website/lessons/github-pages/exercises/git-init>