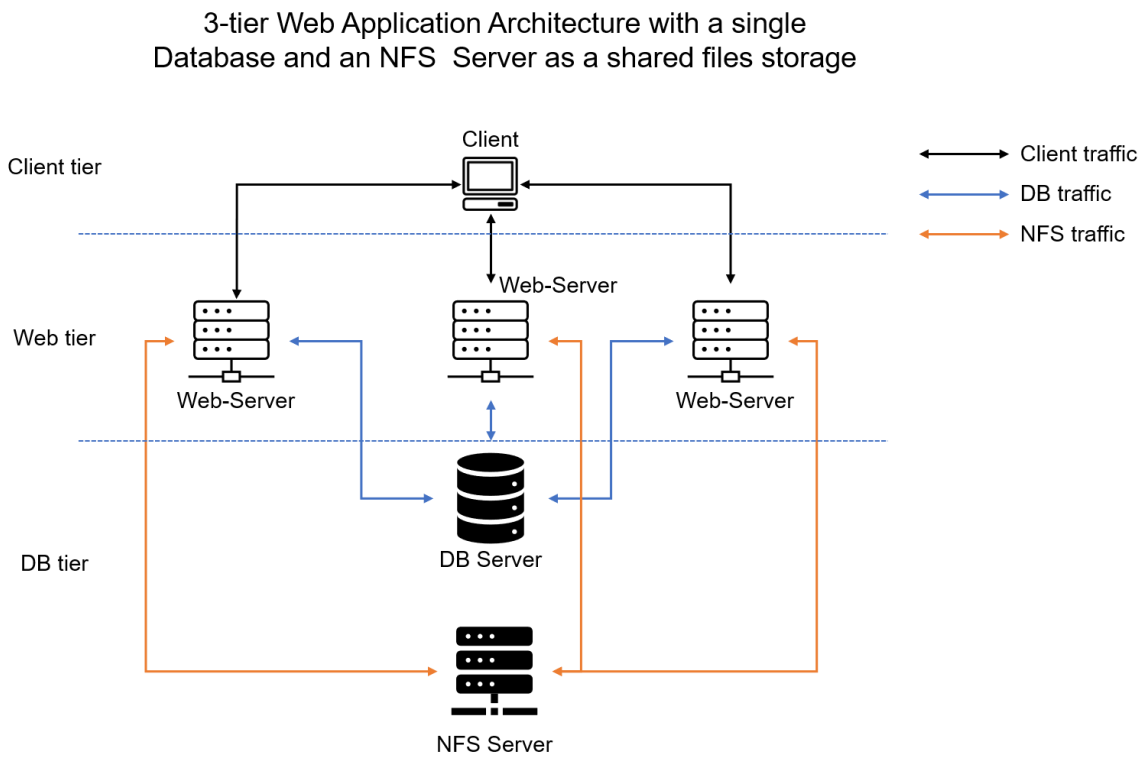
DEVOPS TOOLING WEBSITE SOLUTION

In this follow-up tutorial, we'd introducing a single DevOps Tooling Solution that will consist of the following:

- Jenkins - free and open source automation server used to build CI/CD pipelines.
- Kubernetes - an open-source container-orchestration system for automating computer application deployment, scaling, and management.
- Jfrog Artifactory - Universal Repository Manager supporting all major packaging formats, build tools and CI servers. Artifactory.
- Rancher - an open source software platform that enables organizations to run and manage Docker and Kubernetes in production.
- Grafana - a multi-platform open source analytics and interactive visualization web application.
- Prometheus - An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.
- Kibana - Kibana is a free and open user interface that lets you visualize your Elasticsearch data and navigate the Elastic Stack.

In this project you will implement a solution that consists of following components:

- ✓ Infrastructure: AWS.
- ✓ Webserver Linux: Red Hat Enterprise Linux 8.
- ✓ Database Server: Ubuntu 20.04 + MySQL.
- ✓ Storage Server: Red Hat Enterprise Linux 8 + NFS Server.
- ✓ Programming Language: PHP.
- ✓ Code Repository: GitHub



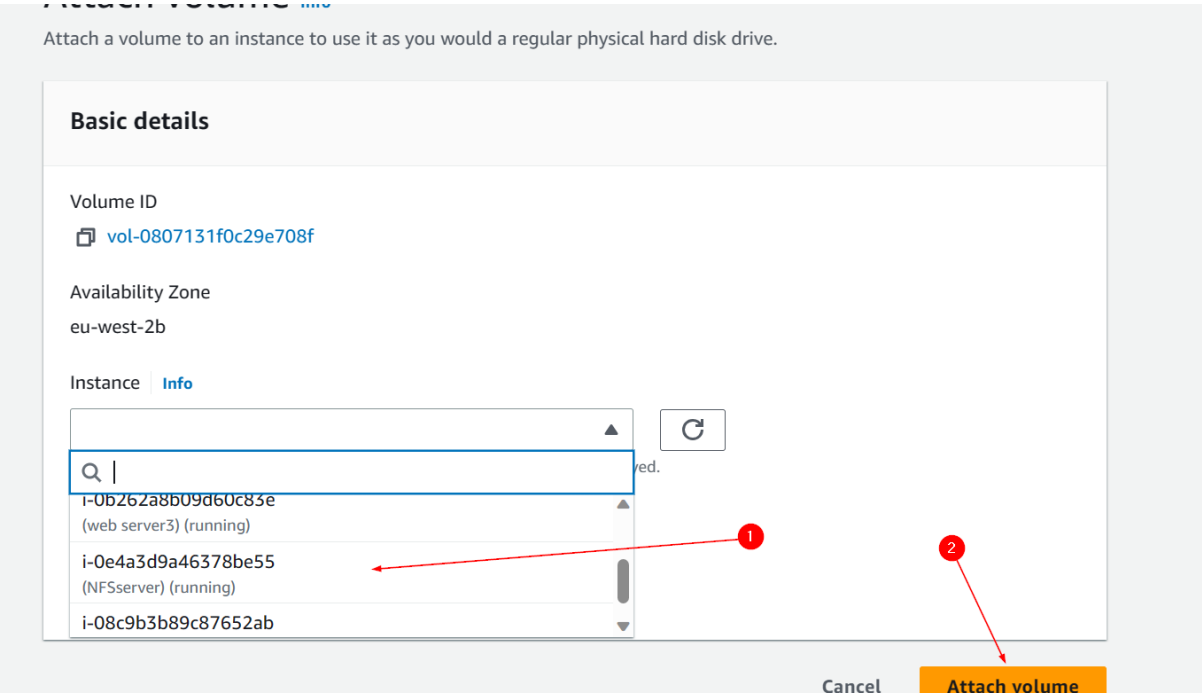
Step 1 - Prepare the NFS Server

1. Spin up a new EC2 instance with RHEL Linux 8 Operating System.

2. Configure LVM on the Server (Follow the following steps to carry out the LVM config):

- Create volumes in the same availability zone as your instance and attach them.

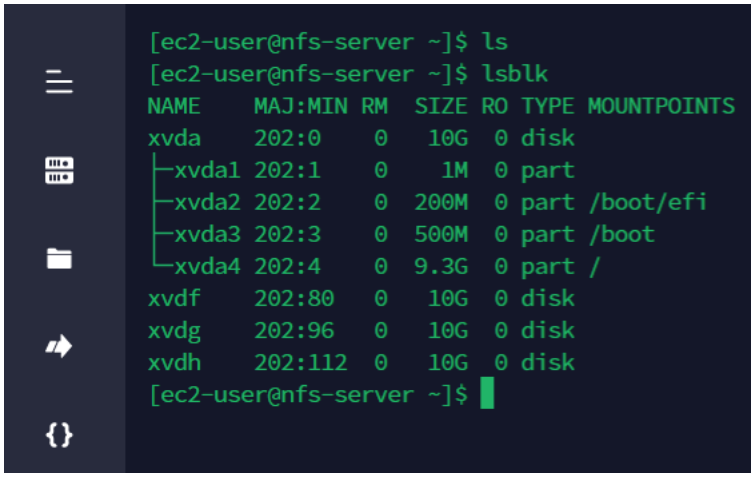
Like so:



nfs-vol1	vol-079fe51c46984388d	gp2	10 GiB	100
nfs-vol2	vol-057abd4d55f4d7449	gp2	10 GiB	100
nfs-vol3	vol-0807131f0c29e708f	gp2	10 GiB	100

To verify, run:

```
sudo lsblk
```



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

```
sudo gdisk /dev/xvdh
```

Output:

```
GPT fdisk (gdisk) version 1.0.7

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries in memory.

Command (? for help): n
Partition number (1-128, default 1):
First sector (34-20971486, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-20971486, default = 20971486) or {+-}size{KMGTP}:
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): 8e00
Changed type of partition to 'Linux LVM'

Command (? for help): p
Disk /dev/xvdf: 20971520 sectors, 10.0 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): F0137738-696E-4817-921D-E9049F6B5997
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 20971486
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048        20971486   10.0 GiB    8E00  Linux LVM

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/xvdf.
The operation has completed successfully.
[ec2-user@nfs-server ~]$
```

verify by running: `lsblk`

```
[ec2-user@nfs-server ~]$ lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
xvda    202:0    0   10G  0 disk
├xvda1 202:1    0    1M  0 part
├xvda2 202:2    0  200M  0 part /boot/efi
├xvda3 202:3    0  500M  0 part /boot
└xvda4 202:4    0  9.3G  0 part /
xvdf    202:80   0   10G  0 disk
└xvdf1 202:81   0   10G  0 part
xvdg    202:96   0   10G  0 disk
└xvdg1 202:97   0   10G  0 part
xvdh    202:112  0   10G  0 disk
└xvdh1 202:113  0   10G  0 part
[ec2-user@nfs-server ~]$
```

3. Install lvm2 package using:

```
sudo yum install lvm2 -y
```

Run `sudo lvmdiskscan` command to check for available partitions

4.Use pvcreate utility to mark each of 3 disks as physical volumes (PVs) to be used by LVM

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

Output:

```
[ec2-user@nfs-server ~]$ sudo pvcreate /dev/xvdf1 /dev/xvdg1 /dev/xvdh1
  Physical volume "/dev/xvdf1" successfully created.
  Physical volume "/dev/xvdg1" successfully created.
  Physical volume "/dev/xvdh1" successfully created.
  Creating devices file /etc/lvm/devices/system.devices
[ec2-user@nfs-server ~]$
```

Verify that your Physical volume has been created successfully by running : `sudo pvs`

Output:

```
[ec2-user@nfs-server ~]$ 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@nfs-server ~]$
```

4. Use vgcreate utility to add all 3 PVs (physical volumes) to a volume group (VG). Name the VG webdata-vg:

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

Verify that your VG has been created successfully by running: `sudo vgs`

Output:

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

5. Use lvcreate utility to create 3 logical volumes:lv-opt lv-apps, and lv-logs

```
sudo lvcreate -n lv-opt -L 9G webdata-vg
sudo lvcreate -n lv-apps -L 9G webdata-vg
sudo lvcreate -n lv-logs -L 9G webdata-vg
```

Output:

```
[ec2-user@nfs-server ~]$ sudo lvcreate -n lv-apps -L 9G webdata-vg
  Logical volume "lv-apps" created.
[ec2-user@nfs-server ~]$ sudo lvcreate -n lv-logs -L 9G webdata-vg
  Logical volume "lv-logs" created.
[ec2-user@nfs-server ~]$ sudo lvcreate -n lv-opt -L 9G webdata-vg
  Logical volume "lv-opt" created.
[ec2-user@nfs-server ~]$
```

run `sudo lvs`

Output:

```
[ec2-user@nfs-server ~]$ sudo lvs
  LV      VG         Attr       LSize Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv-apps webdata-vg -wi-a----- 9.00g
  lv-logs webdata-vg -wi-a----- 9.00g
  lv-opt  webdata-vg -wi-a----- 9.00g
[ec2-user@nfs-server ~]$
```

Format the disks as xfs

```
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
sudo lsblk
```

Output:

```
[ec2-user@nfs-server ~]$ sudo lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
xvda                    202:0    0   10G  0 disk
├xvda1                  202:1    0    1M  0 part
├xvda2                  202:2    0  200M  0 part /boot/efi
├xvda3                  202:3    0  500M  0 part /boot
└xvda4                  202:4    0  9.3G  0 part /
xvdf                    202:80   0   10G  0 disk
└xvdf1                  202:81   0   10G  0 part
  └webdata--vg-lv--opt  253:2    0    9G  0 lvm
xvdg                    202:96   0   10G  0 disk
└xvdg1                  202:97   0   10G  0 part
  └webdata--vg-lv--logs 253:1    0    9G  0 lvm
xvdh                    202:112  0   10G  0 disk
└xvdh1                  202:113  0   10G  0 part
  └webdata--vg-lv--apps 253:0    0    9G  0 lvm
[ec2-user@nfs-server ~]$
```

6. Create mount points on /mnt directory for the logical volumes as follow:

- Mount lv-apps on /mnt/apps – To be used by webservers
- Mount lv-logs on /mnt/logs – To be used by webserver logs
- Mount lv-opt on /mnt/opt – To be used by Jenkins server

```
sudo mkdir /mnt/apps
sudo mkdir /mnt/logs
sudo mkdir /mnt/opt

sudo mount /dev/webdata-vg/lv-apps /mnt/apps
sudo mount /dev/webdata-vg/lv-logs /mnt/logs
sudo mount /dev/webdata-vg/lv-opt /mnt/opt
```

Once mount is completed run:

`sudo blkid` to get the UUID, edit the fstab file accordingly

```
sudo vi /etc/fstab
```

Verify the mount points

```
sudo mount -a
sudo systemctl daemon-reload
```

7. Install NFS server, configure it to start on reboot and make sure it is up and running

```
sudo yum -y update
sudo yum install nfs-utils -y
sudo systemctl start nfs-server.service
sudo systemctl enable nfs-server.service
sudo systemctl status nfs-server.service
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/syst
emd/system/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 01:52:38 UTC; 567ms ago
   Main PID: 15753 (code=exited, status=0/SUCCESS)
        CPU: 31ms
```

8. Set the mount point directory to allow read and write permissions to our webserver

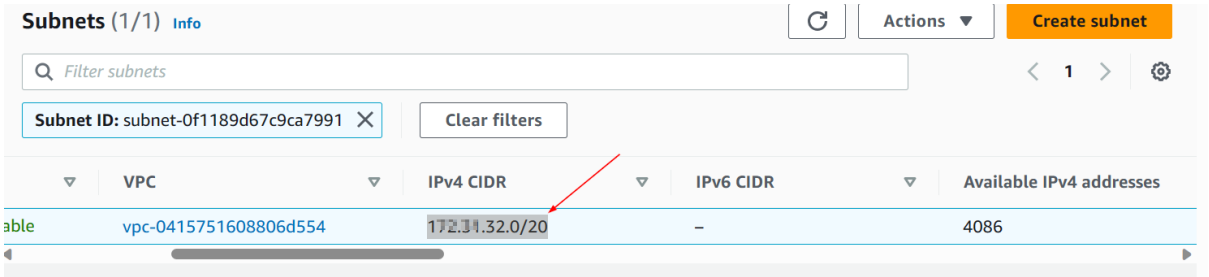
```
sudo chown -R nobody: /mnt/apps
sudo chown -R nobody: /mnt/logs
sudo chown -R nobody: /mnt/opt

sudo chmod -R 777 /mnt/apps
sudo chmod -R 777 /mnt/logs
sudo chmod -R 777 /mnt/opt
```

```
sudo systemctl restart nfs-server.service
```

Note: In this project, we will be creating our NFS-server, web-servers and database-server all in the same subnet

9. Next we configure NFS to interact with clients present in the same subnet.

We can find the subnet ID and CIDR in the Networking tab of our instances




```
sudo vi /etc/exports

/mnt/apps <Subnet-CIDR>(rw,sync,no_all_squash,no_root_squash)
/mnt/logs <Subnet-CIDR>(rw,sync,no_all_squash,no_root_squash)
/mnt/opt <Subnet-CIDR>(rw,sync,no_all_squash,no_root_squash)

Esc + :wq!

sudo exportfs -arv
```

Output:

```
[ec2-user@nfs-server ~]$ sudo exportfs -arv
exporting 172.31.32.0/20:/mnt/opt
exporting 172.31.32.0/20:/mnt/logs
exporting 172.31.32.0/20:/mnt/apps
```

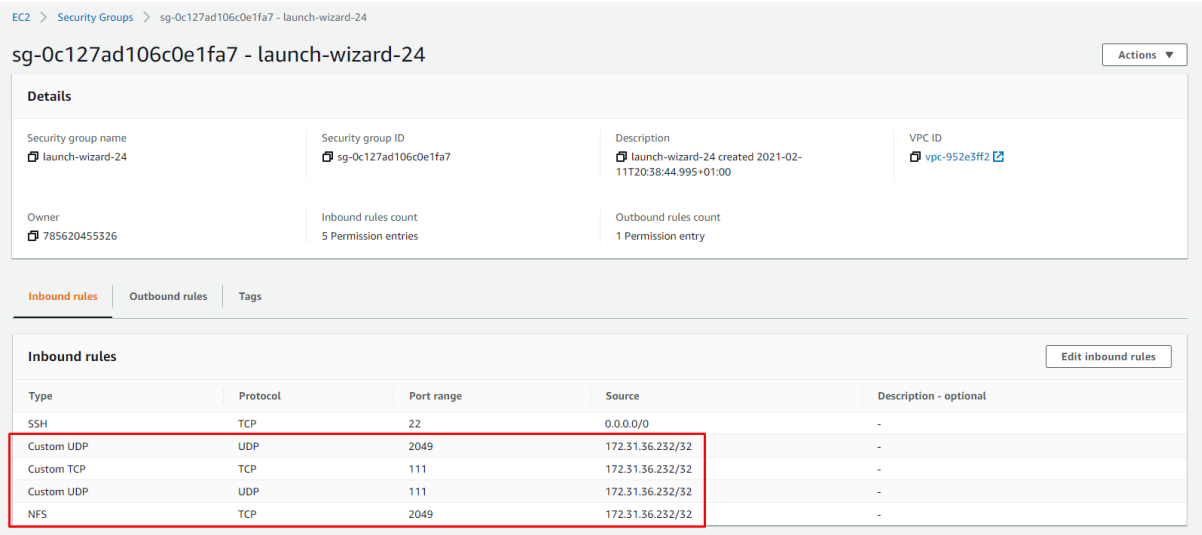
10. Check which port is used by NFS and open it using Security Groups (add new Inbound Rule)

```
rpcinfo -p | grep nfs
```

Output:

```
[ec2-user@nfs-server ~]$ rpcinfo -p | grep nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049  nfs_acl
```

In order for NFS server to be accessible from your client, you must also open following ports:



Step 2 -Configure the Database Server

Create an Ubuntu Server on AWS which will serve as our Database. **Ensure its in the same subnet as the NFS-Server**

Install mysql-server

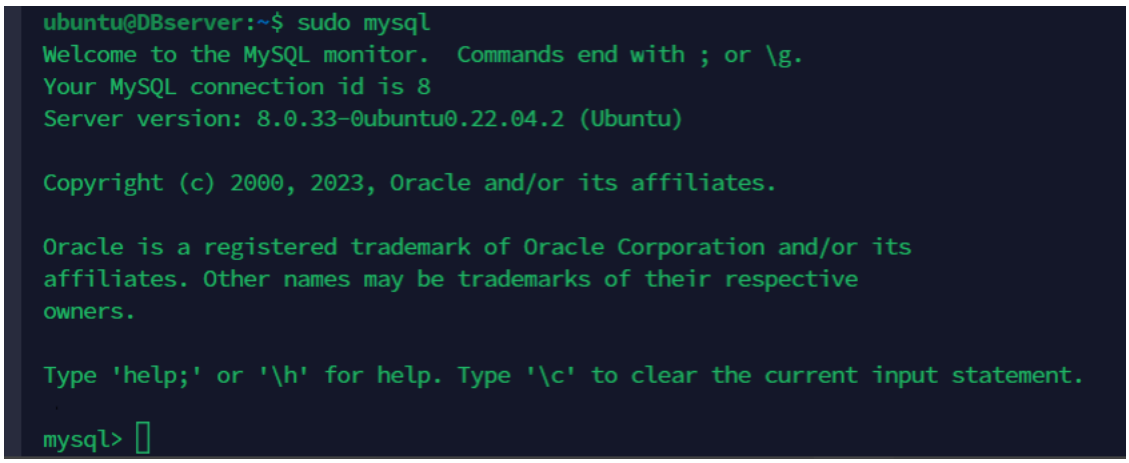
```
sudo apt update
```

```
sudo apt upgrade
```

```
sudo apt install mysql-server
```

```
mysql –version
```

```
sudo mysql
```



- Create a database user with name webaccess and grant permission to the user on tooling db to be able to do anything only from the webservers subnet cidr

```
create database tooling;
```

```
create user 'webaccess'@'172.31.32.0/20' identified by 'password';
grant all privileges on tooling.* to 'webaccess'@'172.31.32.0/20';
flush privileges;
show databases;
```

```
Database changed
mysql> select user, host from user;
+------------------+----------------+
| user             | host           |
+------------------+----------------+
| webaccess        | 172.31.32.0/20 |
| debian-sys-maint | localhost      |
| mysql.infoschema | localhost      |
| mysql.session    | localhost      |
| mysql.sys        | localhost      |
| root             | localhost      |
+------------------+----------------+
6 rows in set (0.00 sec)

mysql>
```

The ip address is the webserver's IPv4 CIDR

Step 3 -Preparing Web Servers

Create a RHEL EC2 instance on AWS which serves as our web server. Also remember to have in it in same subnet

A couple of configurations will be done on the web servers:

- configuring NFS client
- deploying tooling website application
- configure servers to work with database

1. Installing 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
```

Verify that NFS was mounted successfully by running `df -h`

```
[ec2-user@webserver-2 ~]$ 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  4.4M  150M   3% /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
172.31.36.107:/mnt/apps 9.0G   97M  8.9G   2% /var/www
[ec2-user@webserver-2 ~]$
```

You can test the mount by creating a file on the web server and check to see if it's on the nfs server

Make sure that the changes will persist on Web Server after reboot:

```
sudo vi /etc/fstab
```

add following line:

```
<NFS-Server-Private-IP-Address>:/mnt/apps /var/www nfs defaults 0 0
```

2. Installing Apache and Php

```
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:remi-7.4

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

sudo systemctl start php-fpm

sudo systemctl enable php-fpm

sudo setsebool -P httpd_execmem 1
```

```
ec2-13-41-188-21.eu-west-2.compute.amazonaws.com

● php-fpm.service - The PHP FastCGI Process Manager
     Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; enabled; preset: disabled)
     Active: active (running) since Fri 2023-06-16 22:11:57 UTC; 2min 12s ago
   Main PID: 15553 (php-fpm)
     Status: "Processes active: 0, idle: 5, Requests: 0, slow: 0, Traffic: 0req/sec"
      Tasks: 6 (limit: 4421)
     Memory: 13.1M
        CPU: 63ms
     CGroup: /system.slice/php-fpm.service
             ├─15553 "php-fpm: master process (/etc/php-fpm.conf)"
             ├─15554 "php-fpm: pool www"
             ├─15555 "php-fpm: pool www"
             ├─15556 "php-fpm: pool www"
             ├─15557 "php-fpm: pool www"
             └─15558 "php-fpm: pool www"
lines 1-15
```

We can see that both /var/www and /mnt/apps contains same content. This shows that both mount points are connected via NFS.

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

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

```
sudo vi /etc/fstab
```

4. Fork the tooling source code from [Darey.io Github Account]() to your Github account. (Learn how to fork a repo [here]()

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

```
sudo yum install git -y
```

```
[ec2-user@webserver-2 ~]$ git clone https://github.com/opeoba30/tooling.git
Cloning into 'tooling'...
remote: Enumerating objects: 234, done.
remote: Total 234 (delta 0), reused 0 (delta 0), pack-reused 234
Receiving objects: 100% (234/234), 282.72 KiB | 3.04 MiB/s, done.
Resolving deltas: 100% (130/130), done.
[ec2-user@webserver-2 ~]$ ls
tooling
[ec2-user@webserver-2 ~]$ cd tooling/
[ec2-user@webserver-2 tooling]$ ls
apache-config.conf  Dockerfile  html  Jenkinsfile  README.md  start-apache  tooling-db.sql
[ec2-user@webserver-2 tooling]$
```

```
[ec2-user@webserver-1 tooling]$ ls
apache-config.conf  Dockerfile  html  Jenkinsfile  README.md  start-apache  tooling-db.sql
[ec2-user@webserver-1 tooling]$ cd html/
[ec2-user@webserver-1 html]$ ls
admin_tooling.php  functions.php  index.php  README.md     style.css
create_user.php    img            login.php  register.php  tooling_stylesheets.css
[ec2-user@webserver-1 html]$ sudo cp -R . /var/www/html
[ec2-user@webserver-1 html]$ ls
admin_tooling.php  functions.php  index.php  README.md     style.css
create_user.php    img            login.php  register.php  tooling_stylesheets.css
[ec2-user@webserver-1 html]$ cd
[ec2-user@webserver-1 ~]$ cd /var/www/html/
[ec2-user@webserver-1 html]$ ls
admin_tooling.php  functions.php  index.php  README.md     style.css
create_user.php    img            login.php  register.php  tooling_stylesheets.css
[ec2-user@webserver-1 html]$
```

Do not forget to open TCP port 80 on the Web Server.

If you encounter 403 Error – check permissions to your /var/www/html folder and also disable `SELinux sudo setenforce 0`

To make this change permanent – open following config file

```
sudo vi /etc/sysconfig/selinux
```

and set **SELINUX=disabled,** then restart httpd;

```
sudo systemctl start httpd
```

```
sudo systemctl status httpd
```

6. Update the website's configuration to connect to the database (in /var/www/html/functions.php file). Apply tooling-db.sql script to your database using this command mysql -h <databse-private-ip> -u <db-username> -p <db-pasword> < tooling-db.sql

edit the database private ip address, database username and database name. save and quit

```
ec2-18-133-247-53.eu-west-2.compute.amazonaws.com

<?php
session_start();

// connect to database
$db = mysqli_connect('172.31.41.137', 'webaccess', 'password', 'tooling');

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

"/var/www/html/functions.php" 178L, 4374B
```

7. Apply tooling-db.sql script to your database using this command

```
mysql -h <databse-private-ip> -u <db-username> -p <db-pasword> <
tooling-db.sql
```

To do this, you have to follow the following steps:

Install mysql on web server

```
sudo yum install mysql -y
```

Open port 3306 on database server

You'd also need to configure MySQL server to allow connections from remote hosts.

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
#
# 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_tmp
dir
# 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
"mysqld.cnf" 78L, 2216B                                          32,29-33      Top
```

Restart mysql

```
sudo systemctl restart mysql
```

```
sudo systemctl status mysql
```

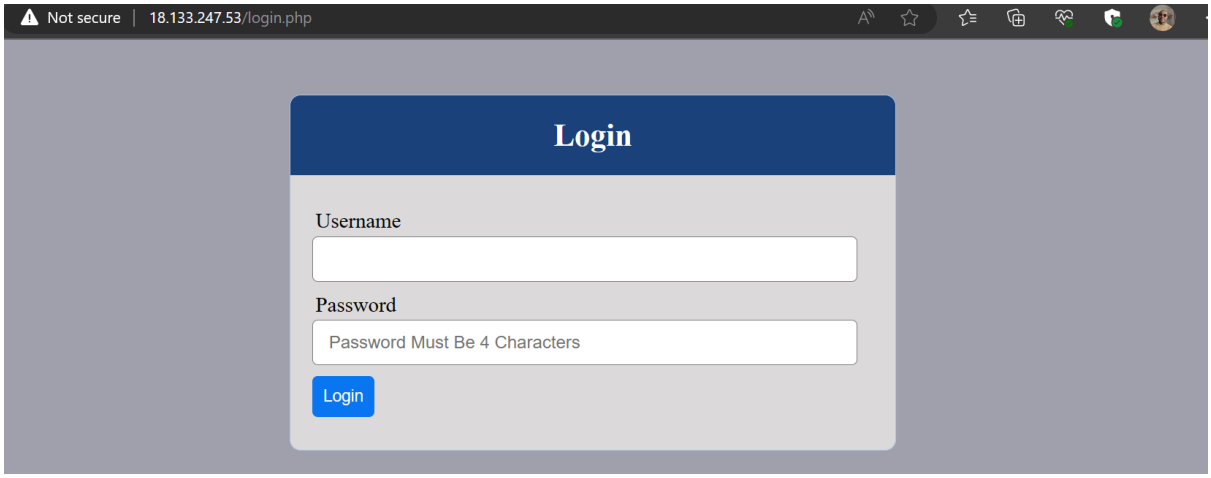
From the webserver, apply tooling-db.sql script to your database

```
mysql -h <databse-private-ip> -u <db-username> -p <dbname> <
tooling-db.sql
```

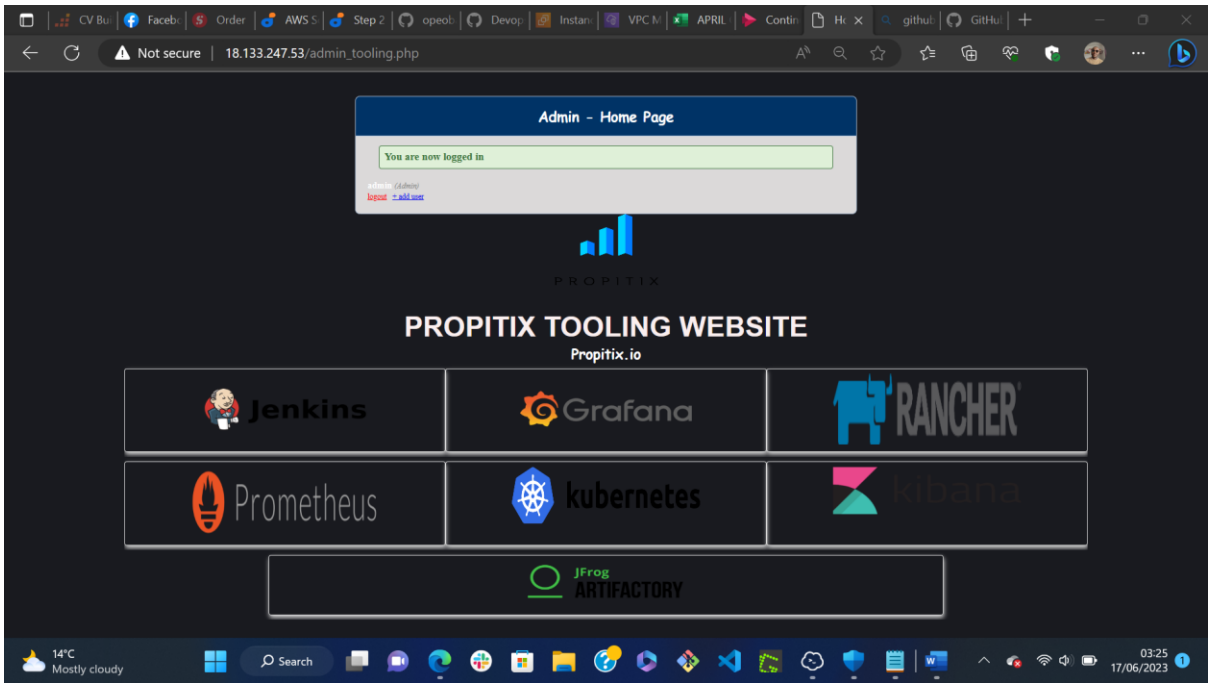
```
[ec2-user@webserver-1 ~]$ mysql -h 172.31.41.137 -u webaccess -p tooling < tooling-db.sql
bash: tooling-db.sql: No such file or directory
[ec2-user@webserver-1 ~]$ sudo setenforce 0
[ec2-user@webserver-1 ~]$ cd tooling/
[ec2-user@webserver-1 tooling]$ sudo vi /etc/sysconfig/selinux
[ec2-user@webserver-1 tooling]$ sudo systemctl restart httpd
[ec2-user@webserver-1 tooling]$
```

8. If it returns no error, create in MySQL a new admin user with username: myuser

9. 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 myuser user.



I will be logging in using user name "admin" and corresponding password



Congratulations!

You have just implemented a web solution for a DevOps team using LAMP stack with remote Database and NFS servers.