DevOps - Project 02

github: - github.com/manish-g0u74m Linkedin : linkedin.com/in/manish-g0u74m

Deploying WordPress on Docker with Data Persistence and Separate Networks also using NFS for High Data availability

Objective:

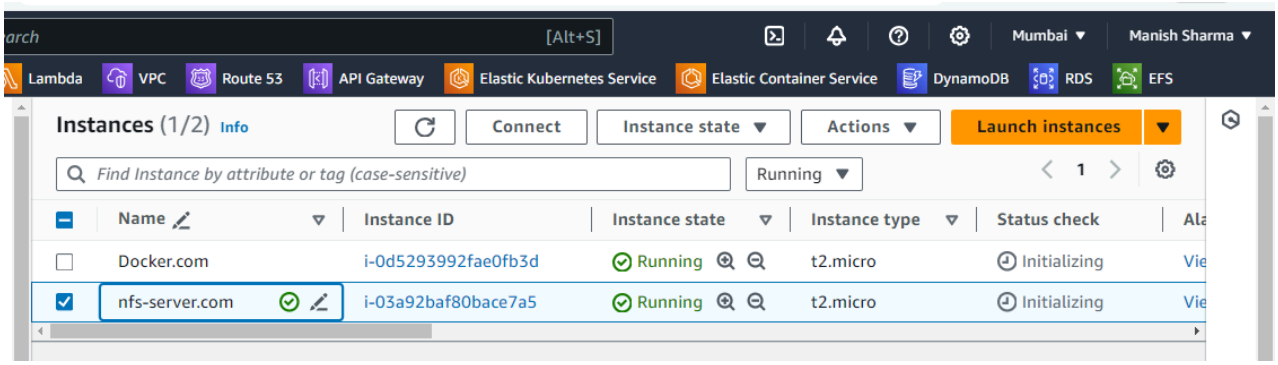
The objective of this Project is to understand how to deploy a WordPress website using Docker, ensuring data persistence using Docker Volumes and isolating the database and WordPress containers on separate networks.

Prerequisites:

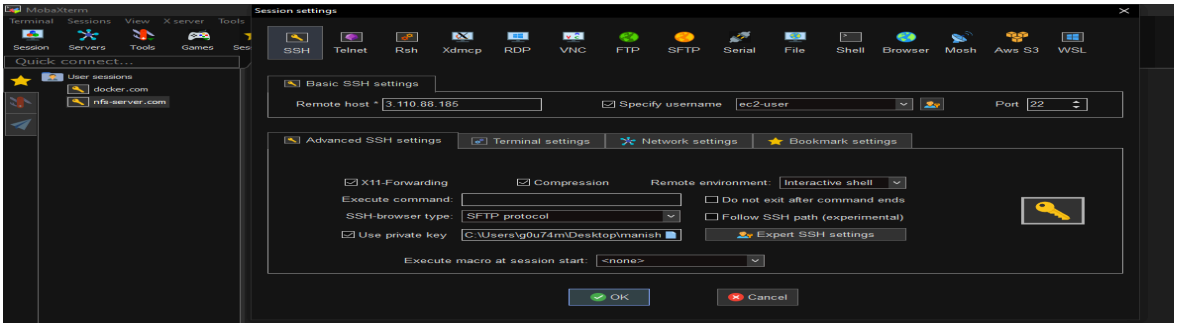
1. Basic knowledge of Docker.
2. Docker installed on your machine.
3. Basic understanding of WordPress and MySQL.
4. Basic Knowledge of NFS

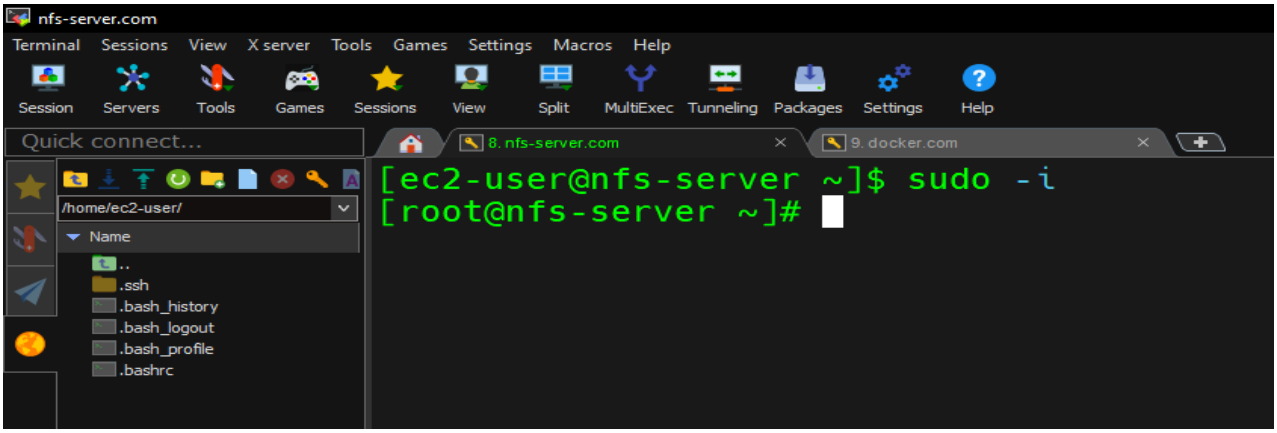
Steps to Complete the Project:

Create 2 vm on aws. 1 vm for nfs server hostname should be **nfs-server.com** and 2nd vm for docker host hostname should be **docker.com**.



Connecting Both Vm's the first one **nfs-server.com** and second one **docker.com** to the MobaXterm (we can use other platforms for connecting like powershell, putty etc.)

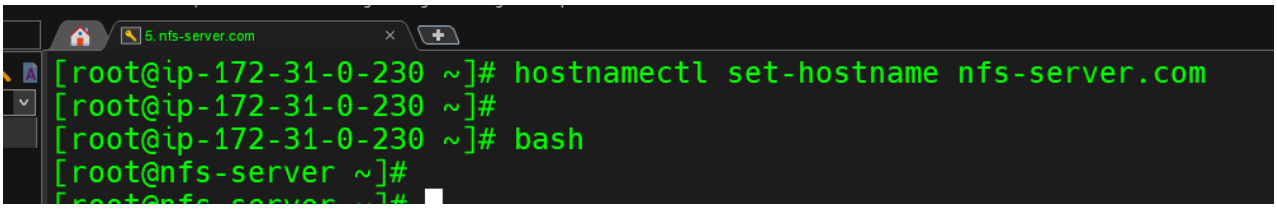




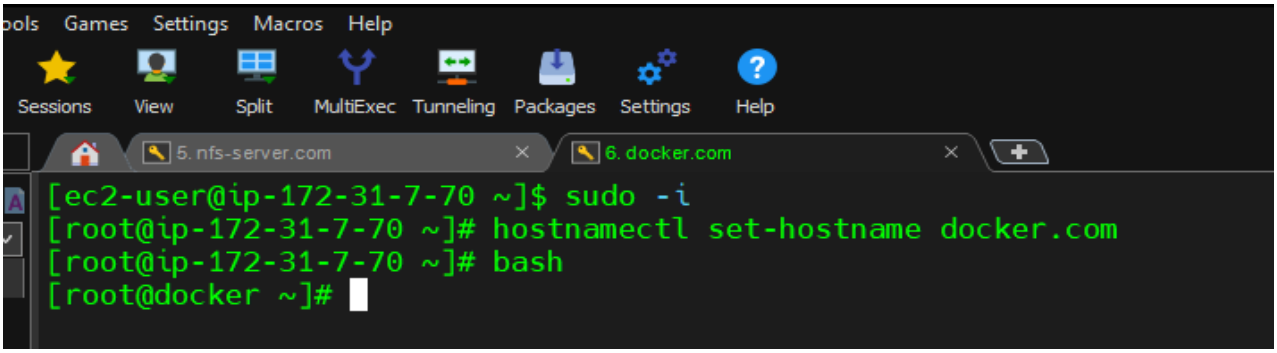
and Switch to the Root User via **sudo -i** command.

Configuring Hostname On Both VM's Via **hostnamectl** command →

1. VM-1 nfs-server.com



2. VM-2 docker.com



Configure NFS server.

1. **Install NFS Utilities:**

   Installing the required NFS packages.

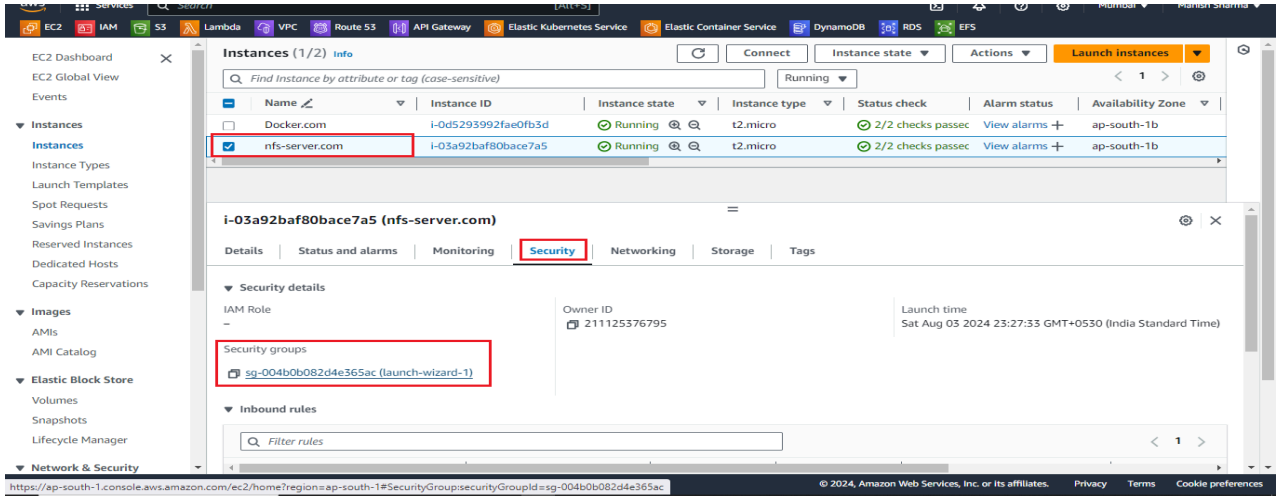
```
[root@nfs-server ~]# yum install nfs-utils -y
Last metadata expiration check: 0:14:58 ago on Sat Aug  3 17:58:24 2024.
Package nfs-utils-1:2.5.4-2.rc3.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@nfs-server ~]#
```

2. **Start & enable nfs service and open 2049/tcp port in security group.**

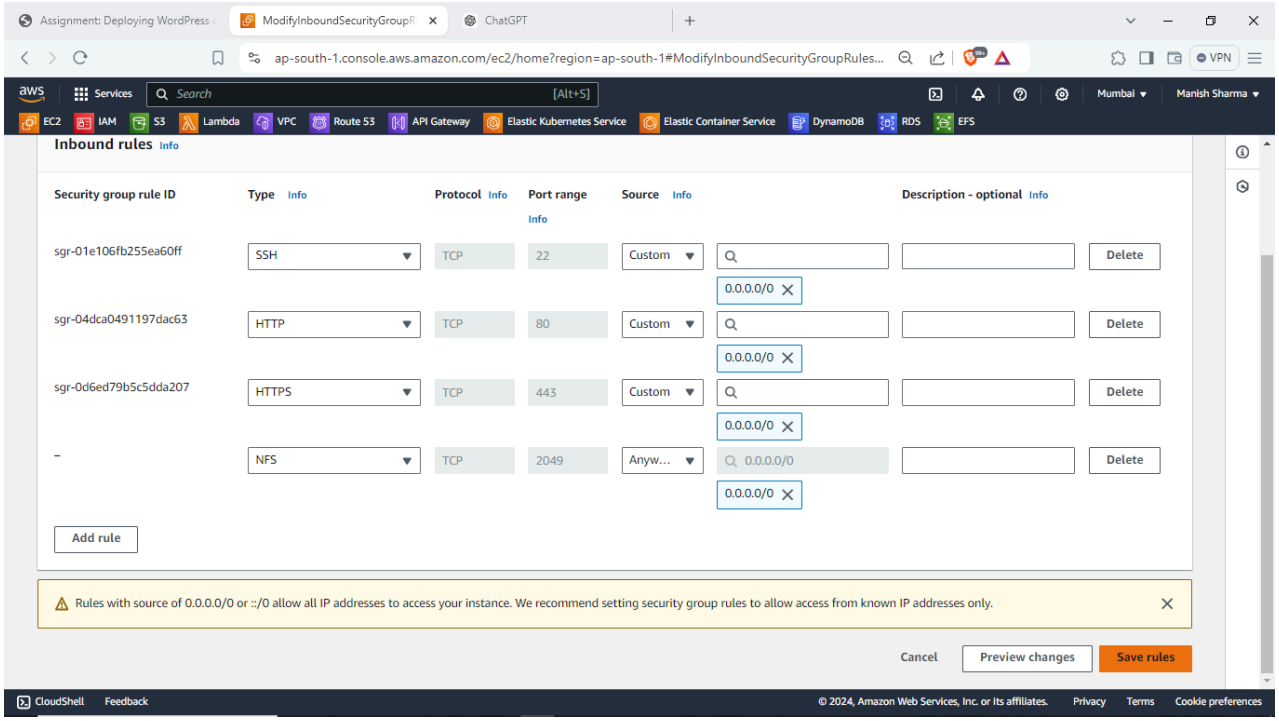
```
[root@nfs-server ~]# systemctl enable --now nfs-server.service
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service
 → /usr/lib/systemd/system/nfs-server.service.
[root@nfs-server ~]# systemctl start nfs-server.service
[root@nfs-server ~]# systemctl status nfs-server.service
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; pre>
     Active: active (exited) since Sat 2024-08-03 18:23:49 UTC; 19s ago
    Process: 3454 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/S>
    Process: 3455 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
    Process: 3473 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; th>
   Main PID: 3473 (code=exited, status=0/SUCCESS)
```

Now Open 2049/tcp port in Security group —>

Check mark on NFS server Vm and click on Security Group and than Open Security groups



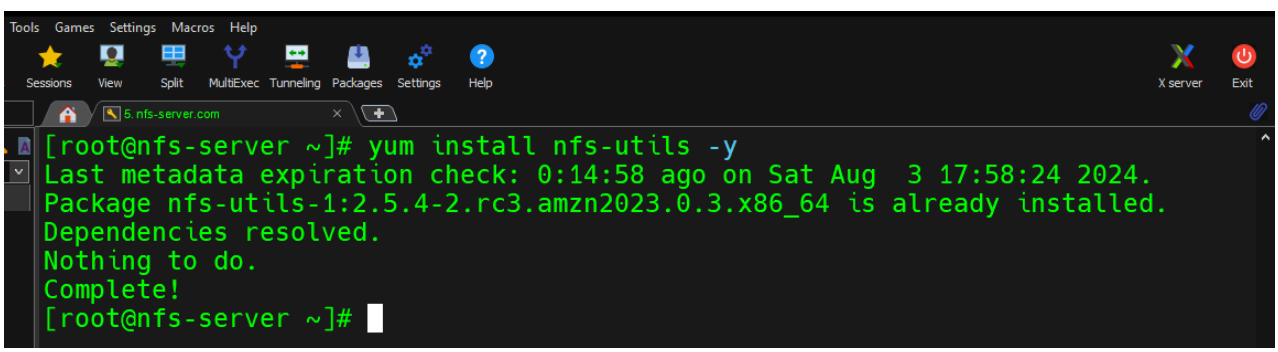
and Add 2049/tcp port for nfs

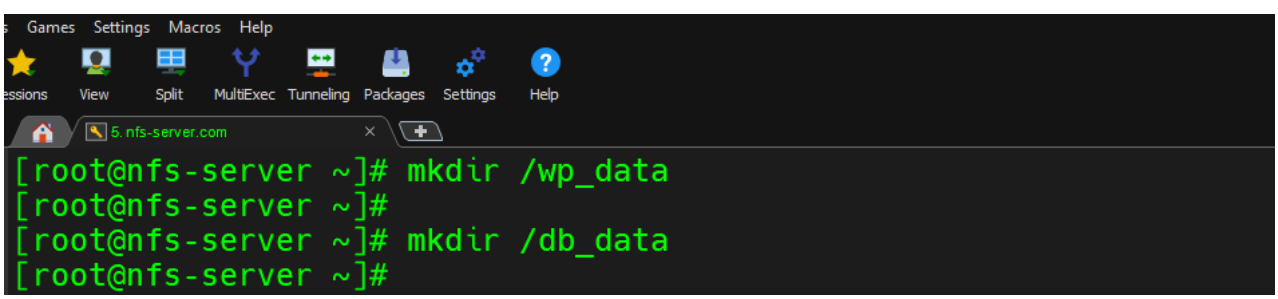


3. **Create a Shared Directory:**

   Create a directory `/wp_data` **(to store wordpress data)with 777 permission** to share over NFS

   Create a directory `/db_data` **(to store mysql data)with 777 permission** to share over NFS.





4. **Edit the Exports File:**
   Configure the `/etc/exports` file to define the shared directory and set the

access permissions with (rw,sync,no_root_squash) only for docker host.
Open /etc/exports file via vim editor & than add data

```
[root@nfs-server ~]# vim /etc/exports
```

```
/wp_data        172.31.7.207(rw,sync,no_root_squash)
/db_data        172.31.7.207(rw,sync,no_root_squash)

~
```

:wq Save And Exit the file and See the exports directory via **showmount -e** command

```
[root@nfs-server ~]# showmount -e
Export list for nfs-server.com:
/db_data 172.31.7.70
/wp_data 172.31.7.70
[root@nfs-server ~]#
```

5. **Restart & enable nfs service.**

```
[root@nfs-server ~]# systemctl enable --now nfs-server.service
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service
 → /usr/lib/systemd/system/nfs-server.service.
[root@nfs-server ~]# systemctl start nfs-server.service
[root@nfs-server ~]# systemctl status nfs-server.service
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; pre>
     Active: active (exited) since Sat 2024-08-03 18:23:49 UTC; 19s ago
    Process: 3454 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/S>
    Process: 3455 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
    Process: 3473 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; th>
   Main PID: 3473 (code=exited, status=0/SUCCESS)
```

Configure Wordpress on Docker.

→ Installing Docker on VM-2 docker machine

```
5. nfs-server.com    6. docker.com
[root@docker ~]# yum install -y docker
Last metadata expiration check: 1:08:05 ago on Sat Aug  3 17:58:14 2024.
Dependencies resolved.
================================================================================
 Package                    Arch      Version                    Repository      Size
================================================================================
Installing:
 docker                     x86_64    25.0.6-1.amzn2023.0.1      amazonlinux     44 M
Installing dependencies:
 containerd                 x86_64    1.7.11-1.amzn2023.0.1      amazonlinux     35 M
 iptables-libs              x86_64    1.8.8-3.amzn2023.0.2       amazonlinux    401 k
 iptables-nft               x86_64    1.8.8-3.amzn2023.0.2       amazonlinux    183 k
 libcgroup                  x86_64    3.0-1.amzn2023.0.1         amazonlinux     75 k
 libnetfilter_conntrack     x86_64    1.0.8-2.amzn2023.0.2       amazonlinux     58 k
 libnfnetlink               x86_64    1.0.1-19.amzn2023.0.2      amazonlinux     30 k
 libnftnl                   x86_64    1.2.2-2.amzn2023.0.2       amazonlinux     84 k
 pigz                       x86_64    2.5-1.amzn2023.0.3         amazonlinux     83 k
 runc                       x86_64    1.1.11-1.amzn2023.0.1      amazonlinux    3.0 M
```

Enable and Start the Docker Service and check status

```
5. nfs-server.com    6. docker.com
[root@docker ~]# systemctl enable --now docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/li
b/systemd/system/docker.service.
[root@docker ~]# systemctl start  docker
[root@docker ~]# systemctl status  docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disab>
     Active: active (running) since Sat 2024-08-03 19:08:30 UTC; 15s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Process: 28298 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SU>
    Process: 28299 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=>
   Main PID: 28300 (dockerd)
      Tasks: 7
     Memory: 29.6M
        CPU: 319ms
     CGroup: /system.slice/docker.service
             └─28300 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contain>
```

—> Create Docker volumes

For wordpress container -> **`wp_data`**
For mysql container -> **`db_data`**

```
[root@docker ~]# docker volume create --name wp_data
wp_data
[root@docker ~]# docker volume create --name db_data
db_data
[root@docker ~]#
```

1. **Mount the NFS Share:**

   Mount the NFS share to the created volumes (*/wp_data to wp_vol, /db_data to db_vol*).
   NOTE-> You have to find the path of volumes before mount.

```
[root@docker ~]# mount -t nfs 172.31.0.230:/wp_data /var/lib/docker/volumes/wp_data/
_data
[root@docker ~]#
[root@docker ~]# mount -t nfs 172.31.0.230:/db_data /var/lib/docker/volumes/db_data/
_data
[root@docker ~]#
```

2. **Create Docker Networks:**

   Create two Docker networks, one for the database (**`db_network`**) and one for the WordPress (**`wp_network`**) application.

```
[root@docker ~]# docker network create db_network
3fe248c5341130a1791342a2582d6474f2a69840a0a76e8dbb619e40faf320f7
[root@docker ~]# docker network create wp_network
3eb2e57f939415a1dc7a4df217d0e548b191e5beab209f1c9aa26f809f12b832
[root@docker ~]# docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
a580267d602d   bridge       bridge    local
3fe248c53411   db_network   bridge    local
3100110be886   host         host      local
b1f7717da6c5   none         null      local
3eb2e57f9394   wp_network   bridge    local
[root@docker ~]#
```

3. **Pull mysql & wordpress images from docker registry to docker host.**

```
[root@docker ~]# docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
efc2b5ad9eec: Pull complete
a6a83fa76a2b: Pull complete
efb3cd9e6b42: Extracting  21.73MB/104.3MB
f41714dd6e6a: Download complete
e362d14d0b88: Download complete
d1b475c73fa4: Download complete
```

```
[root@docker ~]# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
d9a40b27c30f: Extracting  24.58MB/48.99MB
fe4b01031aab: Download complete
aa72c34c4347: Download complete
473ade985fa2: Download complete
cc168a9482de: Download complete
3ca3786815dd: Download complete
3e3fac98ea83: Download complete
```

We can see docker images via **docker ps** command.

```
[root@docker ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
wordpress     latest    f610b9395a59   11 days ago   686MB
mysql         latest    7ce93a845a8a   11 days ago   586MB
[root@docker ~]#
```

4. **Run the MySQL Container:**

   Start a MySQL container connected to both the network **`wp_network`** **and** **`db_network`.**

   Container name should be **`mysql`.**

   Mount **`db_vol`** volume to **`/var/lib/mysql`** on wordpress container.

   Container should be in running state.

   Give environment variable mysq_root_password as **redhat.**

```
[root@docker ~]# docker run -d --network db_network --network wp_network --restart always -e MYSQL_
ROOT_PASSWORD=redhat -v db_data:/var/lib/mysql --name mysql  mysql:latest
b71e17e513c8054c128dbb3993c1e5c07c83a69c9753bcc12ef3a0f7967c4320
[root@docker ~]#
```

Check running Container via **docker ps** comm

5. **Switch on mysql container and create database & user for wordpress.**

Username should be wp_user for all host.

Database name should be wp_db and grant all privileges to wp_user.

Now logining in the mysql container

```
[root@docker ~]#
[root@docker ~]# docker exec -it mysql bash
bash-5.1#
bash-5.1#
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.0.1 MySQL Community Server - GPL
```

and creating **wp_db** database —>

```
mysql> create database wp_db;
Query OK, 1 row affected (0.01 sec)

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

creating wp_user with full permissions so we can perform crud operation on **wp_db** database →

```
mysql> create user wp_user'@'%' identified by 'redhat';
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> grant all privileges on wp_db.* to 'wp_user'@'%';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

6. **Run the WordPress Container:**

   Start a WordPress container connected to both the **`web_network`** **and** **`db_network`.**

   Container name should be **`wordpress`**

   Map host machine 80 port to container 80 post.

   Mount *wp_vol* volume to /var/www/html on wordpress container.

```
[root@docker ~]# docker run -d --name wordpress --network wp_network \
  --restart always \
  -e WORDPRESS_DB_HOST=mysql \
  -e WORDPRESS_DB_USER=wp_user \
  -e WORDPRESS_DB_PASSWORD=redhat \
  -e WORDPRESS_DB_NAME=wp_db \
  -v wp_data:/var/www/html \
  -p 80:80 \
  wordpress:latest
fb96112281917ff4f1fc4001f442440a2c0a0c28800f8c64a20311988d507741
[root@docker ~]# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          ST
ATUS          PORTS                                  NAMES
fb9611228191   wordpress:latest   "docker-entrypoint.s…"   34 seconds ago   Up
 33 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   wordpress
b71e17e513c8   mysql:latest       "docker-entrypoint.s…"   4 hours ago      Up
 4 hours      3306/tcp, 33060/tcp                    mysql
[root@docker ~]#
```

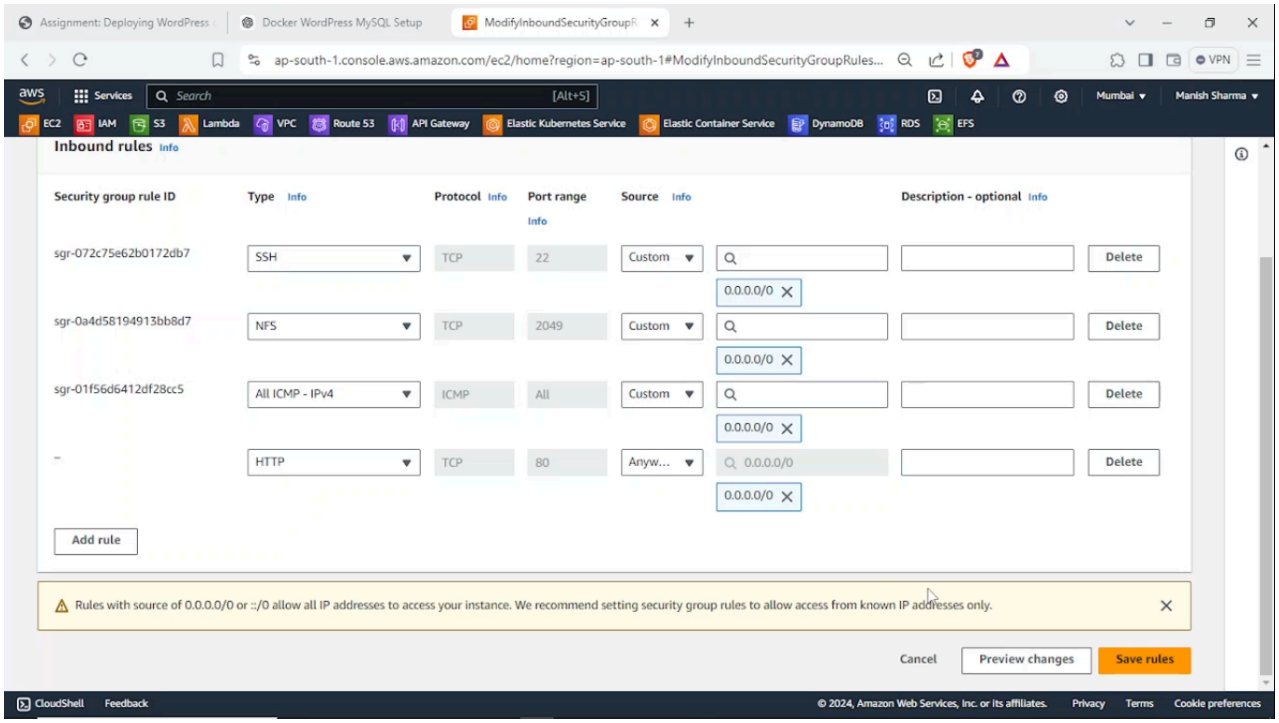
7. **Switch on mysql container and Modify `wp-config.php` file** such as database details**.**

   Copy **`wp-config-sample.php`** file as **`wp-config.php`**
   Update the following lines with your database information:

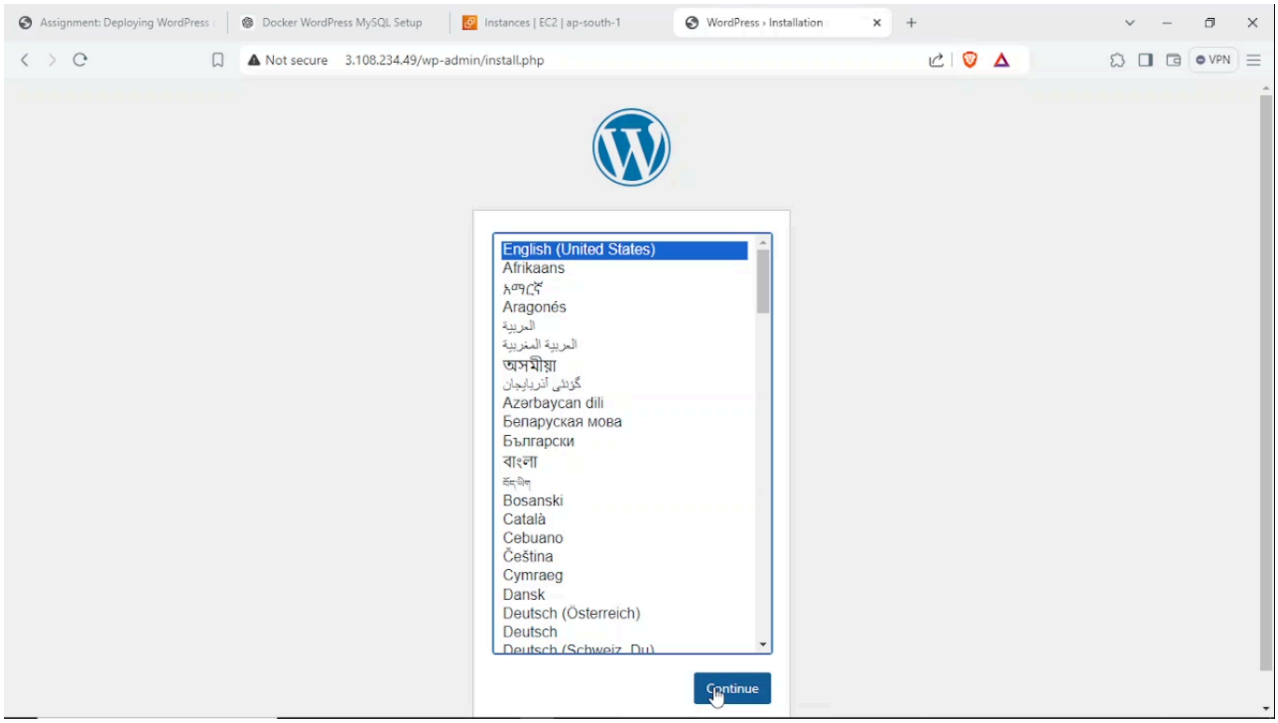
   -> database name
   -> database user
   -> database user password
   -> database host

   i am not doing this step because i already specify all environmental veriable at the time of starting the wordpress container

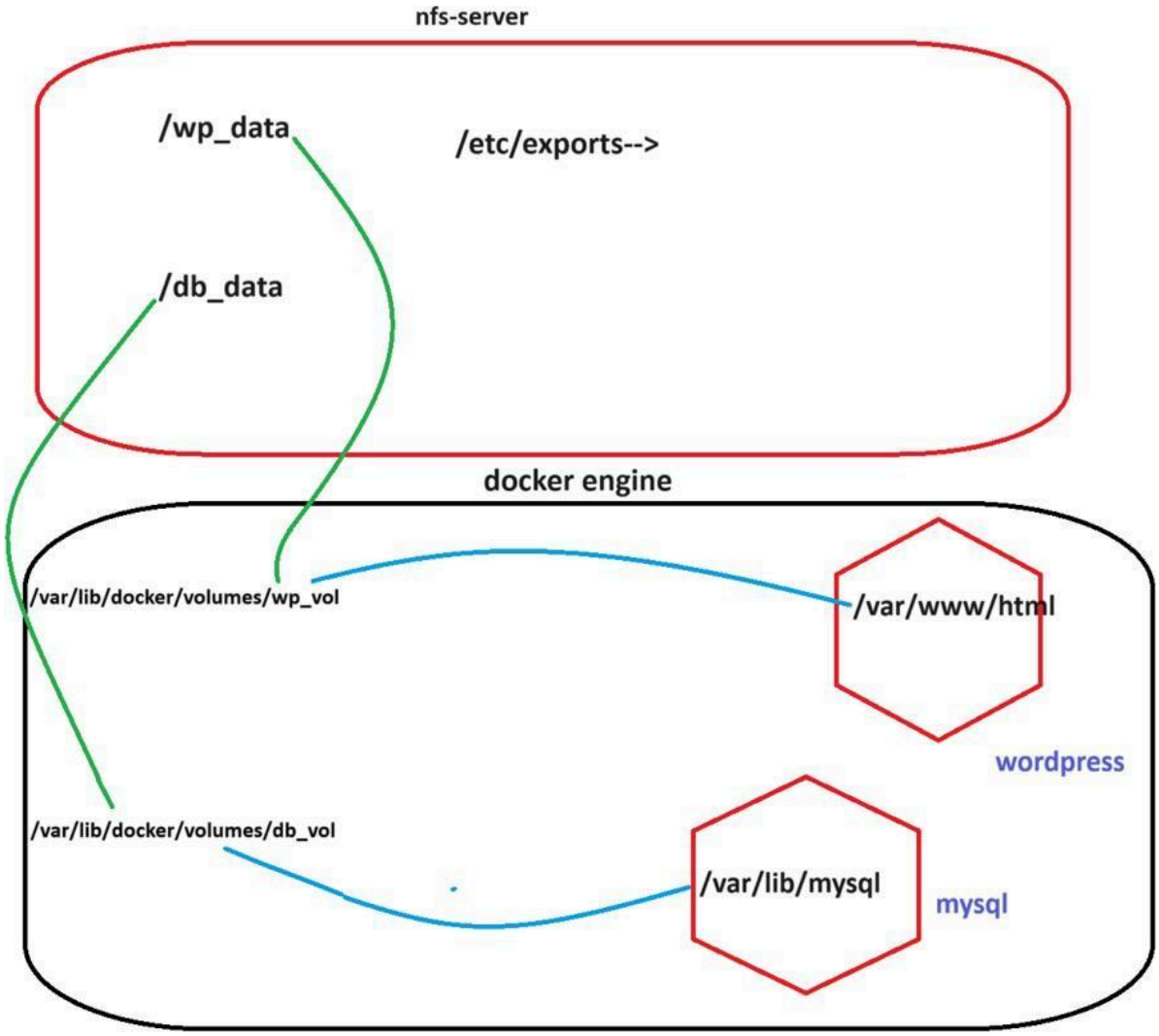
8. Open 80/tcp port in security Group
   Adding 80/tcp port in security group



9. Now open the browser and http://docker-host-public-IP



BOOM💥 Wordpress Successfully Deployed on Docker with Data Persistence and Separate Networks also using NFS for High Data availability.


nfs-server
/wp_data
/etc/exports-->
/db_data
docker engine
/var/lib/docker/volumes/wp_vol
/var/www/html
wordpress
/var/lib/docker/volumes/db_vol
/var/lib/mysql
mysql