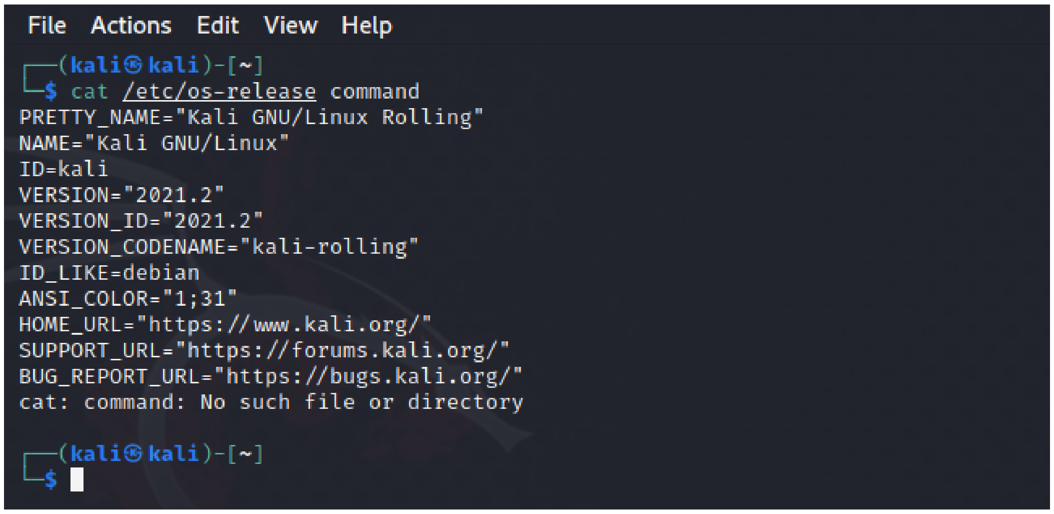
Docker-Compose Exercise

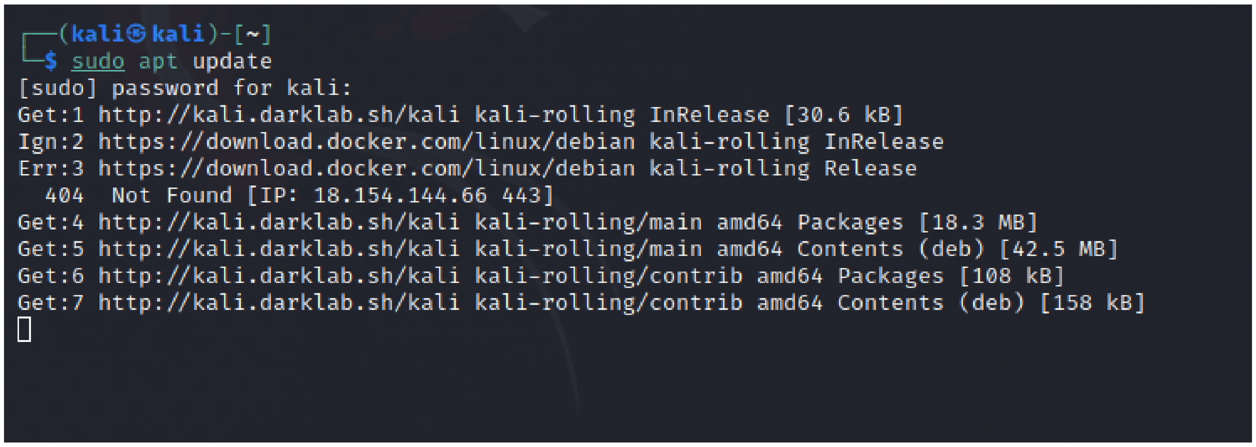
In this exercise we will create images using docker and docker compose to quickly initiate the startup of networked environments using a newly created .yaml file from inside a Kali Linux virtual machine hosted on VMware.

Performed on the following system (below)



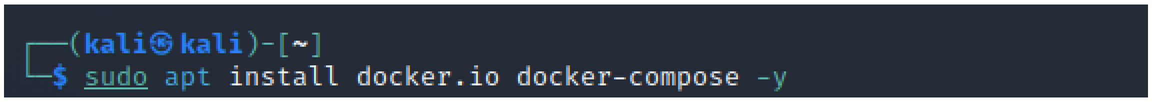
```
┌──(kali㉿kali)-[~]
└─$ cat /etc/os-release command
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
ID=kali
VERSION="2021.2"
VERSION_ID="2021.2"
VERSION_CODENAME="kali-rolling"
ID_LIKE=debian
ANSI_COLOR="1;31"
HOME_URL="https://www.kali.org/"
SUPPORT_URL="https://forums.kali.org/"
BUG_REPORT_URL="https://bugs.kali.org/"
cat: command: No such file or directory

┌──(kali㉿kali)-[~]
└─$ 
```

1. Update files with
   sudo apt update



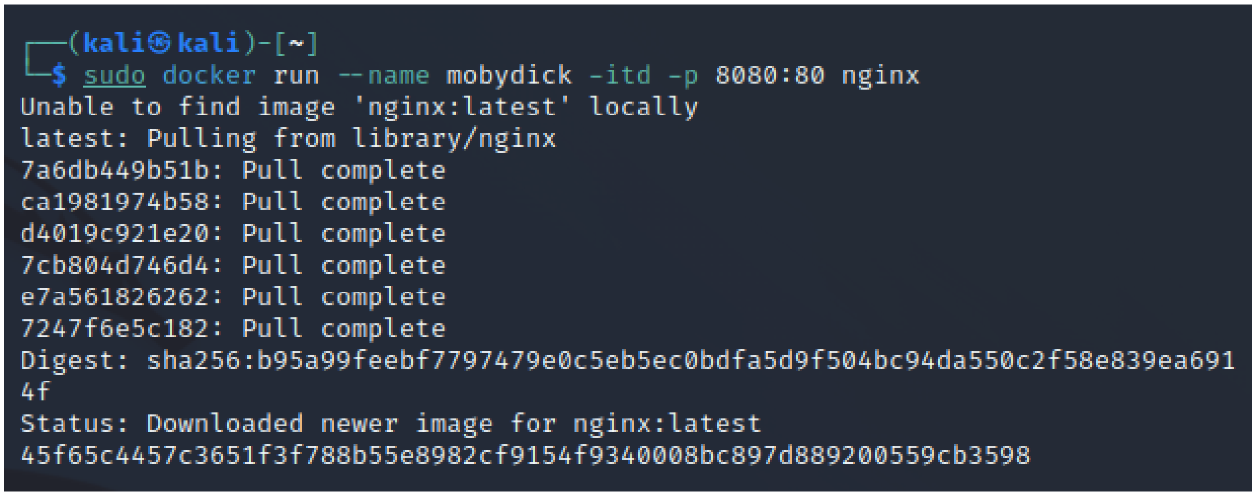
```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
[sudo] password for kali:
Get:1 http://kali.darklab.sh/kali kali-rolling InRelease [30.6 kB]
Ign:2 https://download.docker.com/linux/debian kali-rolling InRelease
Err:3 https://download.docker.com/linux/debian kali-rolling Release
  404  Not Found [IP: 18.154.144.66 443]
Get:4 http://kali.darklab.sh/kali kali-rolling/main amd64 Packages [18.3 MB]
Get:5 http://kali.darklab.sh/kali kali-rolling/main amd64 Contents (deb) [42.5 MB]
Get:6 http://kali.darklab.sh/kali kali-rolling/contrib amd64 Packages [108 kB]
Get:7 http://kali.darklab.sh/kali kali-rolling/contrib amd64 Contents (deb) [158 kB]
```

2. **sudo apt install docker.io docker-compose -y** (this will install docker compose)



```
┌──(kali㉿kali)-[~]
└─$ sudo apt install docker.io docker-compose -y
```

3. **sudo docker run --name mobydick -itd -p 8080:80 nginx**
   (this will startup a single docker image for comparison)

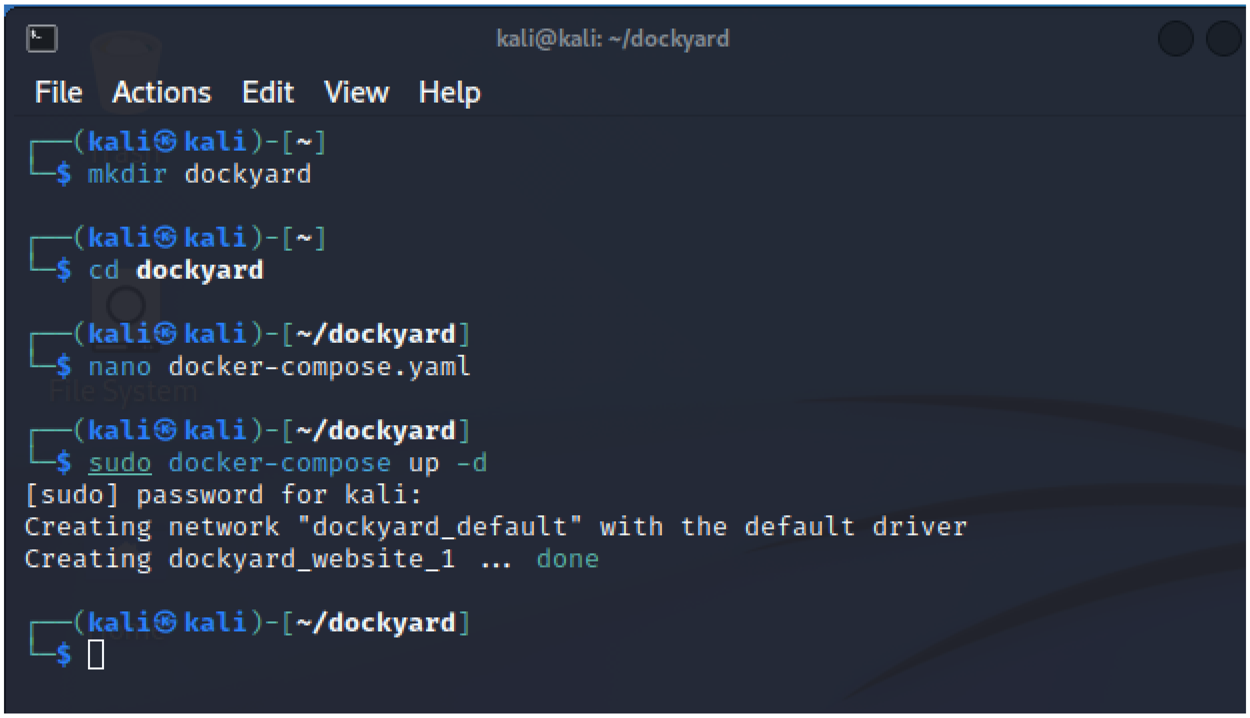


```
┌──(kali㉿kali)-[~]
└─$ sudo docker run --name mobydick -itd -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
7a6db449b51b: Pull complete
ca1981974b58: Pull complete
d4019c921e20: Pull complete
7cb804d746d4: Pull complete
e7a561826262: Pull complete
7247f6e5c182: Pull complete
Digest: sha256:b95a99feebf7797479e0c5eb5ec0bdfa5d9f504bc94da550c2f58e839ea691
4f
Status: Downloaded newer image for nginx:latest
45f65c4457c3651f3f788b55e8982cf9154f9340008bc897d889200559cb3598
```

4. Create new directory to hold a created .yaml file
   mkdir dockyard

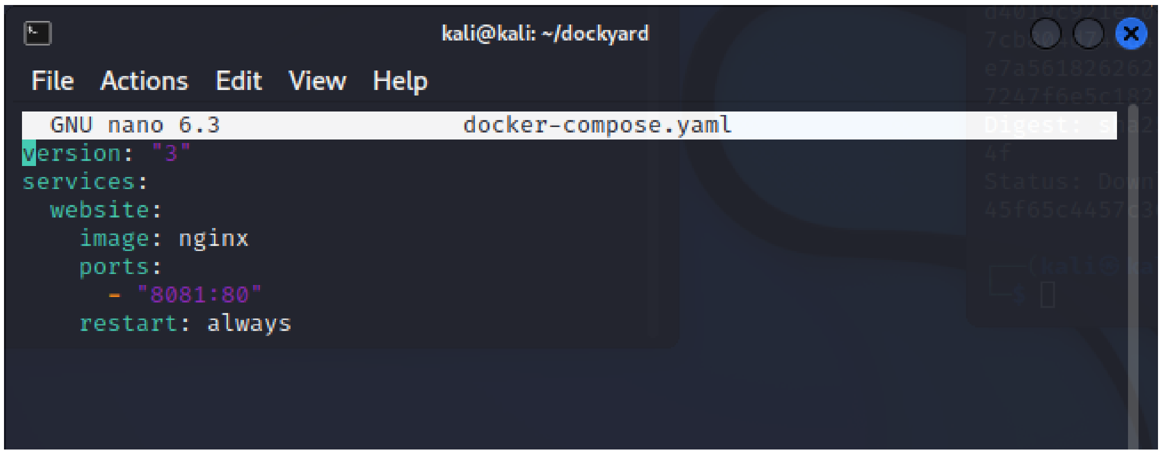
5. Switch to that directory
   cd /dockyard

6. Create new file
   nano docker-compose.yaml



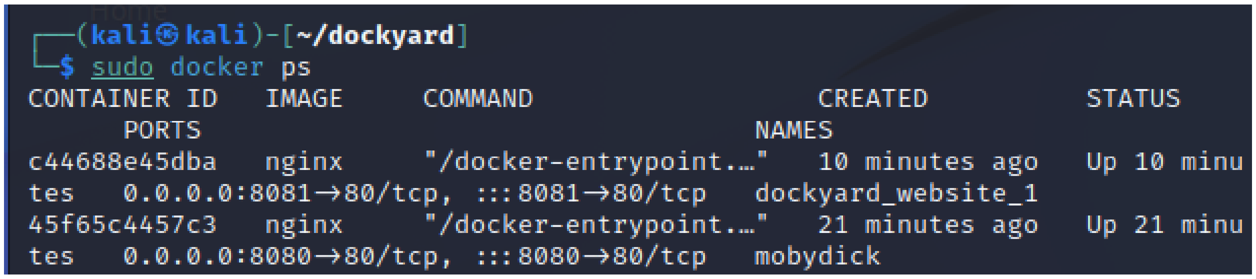
```
┌──(kali㉿kali)-[~]
└─$ mkdir dockyard

┌──(kali㉿kali)-[~]
└─$ cd dockyard

┌──(kali㉿kali)-[~/dockyard]
└─$ nano docker-compose.yaml

┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker-compose up -d
[sudo] password for kali:
Creating network "dockyard_default" with the default driver
Creating dockyard_website_1 ... done

┌──(kali㉿kali)-[~/dockyard]
└─$ 
```

7. Setup .yaml with the commands shown below



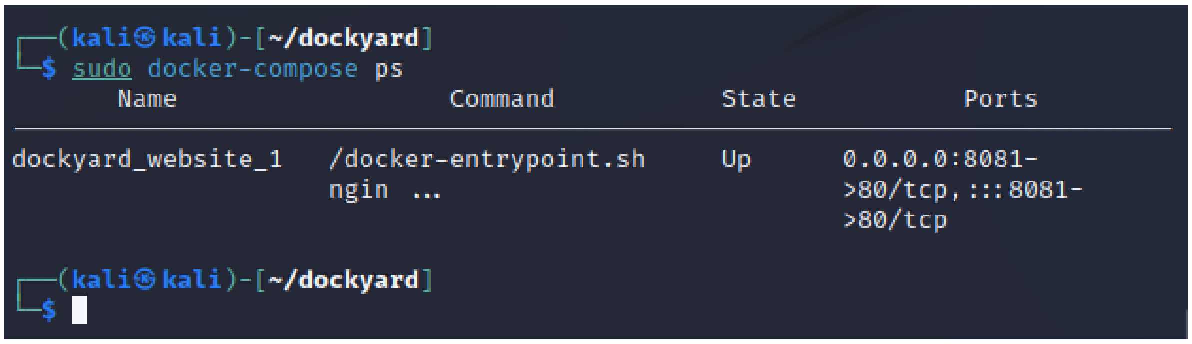
```
 GNU nano 6.3                    docker-compose.yaml
version: "3"
services:
  website:
    image: nginx
    ports:
      - "8081:80"
    restart: always
```

8. Verify the website works with
   sudo docker ps
   (this shows the docker image created outside of docker compose named “mobydick” and the auto created “dockyard_website_1”)



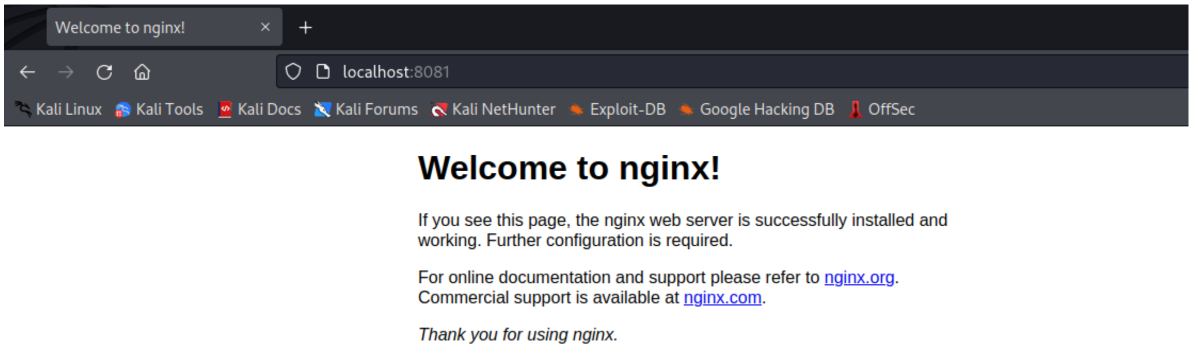
```
┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
      PORTS                                   NAMES
c44688e45dba   nginx     "/docker-entrypoint.…"   10 minutes ago   Up 10 minu
tes   0.0.0.0:8081->80/tcp, :::8081->80/tcp   dockyard_website_1
45f65c4457c3   nginx     "/docker-entrypoint.…"   21 minutes ago   Up 21 minu
tes   0.0.0.0:8080->80/tcp, :::8080->80/tcp   mobydick
```

9. By typing
   sudo docker-compose ps
   (we can then see only the containers within the environment it created)



```
┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker-compose ps
      Name                   Command              State            Ports
-----------------------------------------------------------------------------
dockyard_website_1   /docker-entrypoint.sh        Up      0.0.0.0:8081-
                     ngin ...                             >80/tcp,:::8081-
                                                          >80/tcp

┌──(kali㉿kali)-[~/dockyard]
└─$ 
```

10. Verify website by typing by typing in a web browser
    localhost:8081



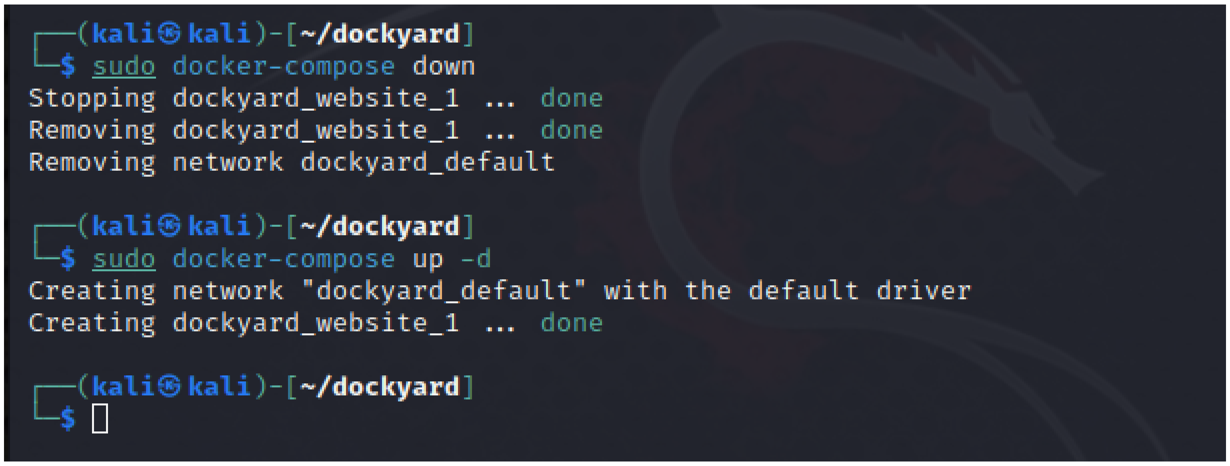
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

11. Take down entire container network command or start it up
    sudo docker-compose down (or) sudo docker-compose up -d (to spin up the network)

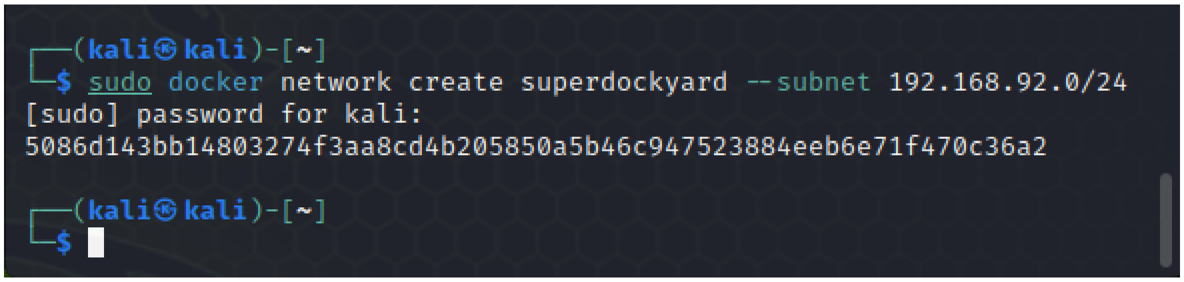


```
┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker-compose down
Stopping dockyard_website_1 ... done
Removing dockyard_website_1 ... done
Removing network dockyard_default

┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker-compose up -d
Creating network "dockyard_default" with the default driver
Creating dockyard_website_1 ... done

┌──(kali㉿kali)-[~/dockyard]
└─$ 
```

Creating separate networks

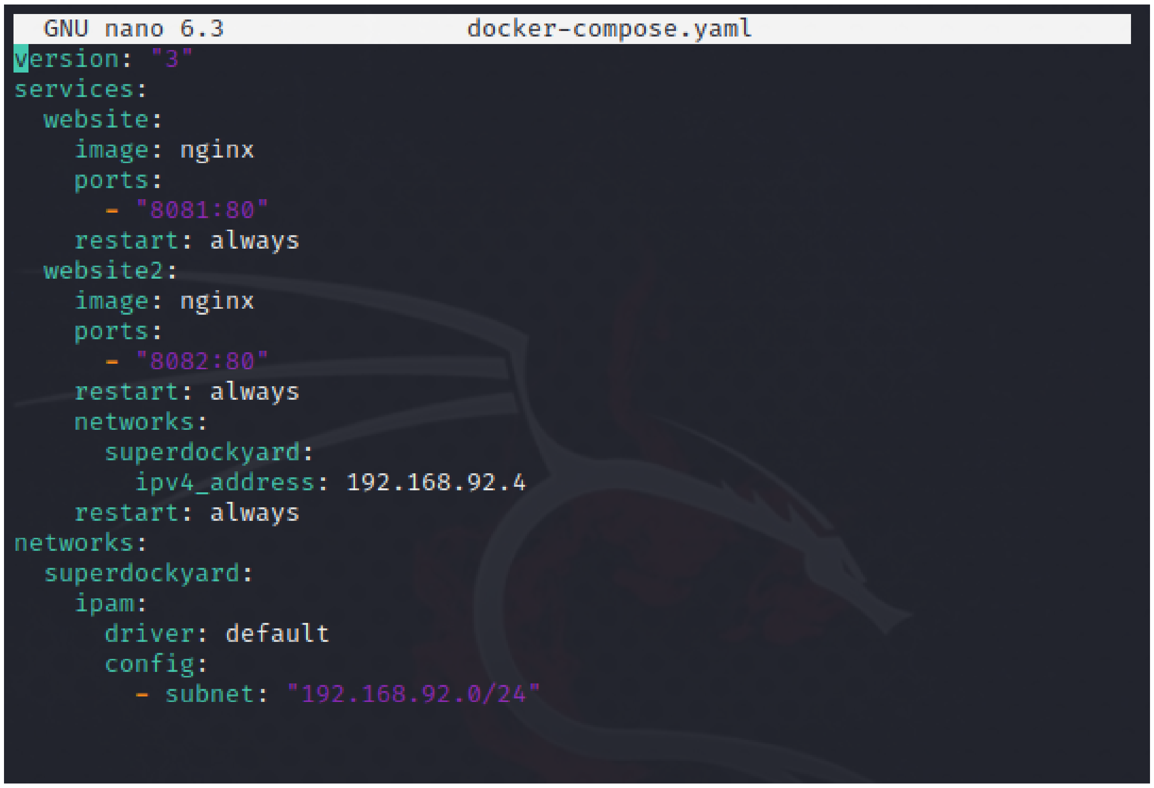
The steps shown below show how additional images may be added to a created network.

1. Docker simple method with singular command resulting in a single step being performed
   After viewing this method, run
   docker network ls
   to list all the docker networks running currently on your machine. you should be able to remove it with the following command
   docker network rm my_network (where my_network is the one you have created initially)



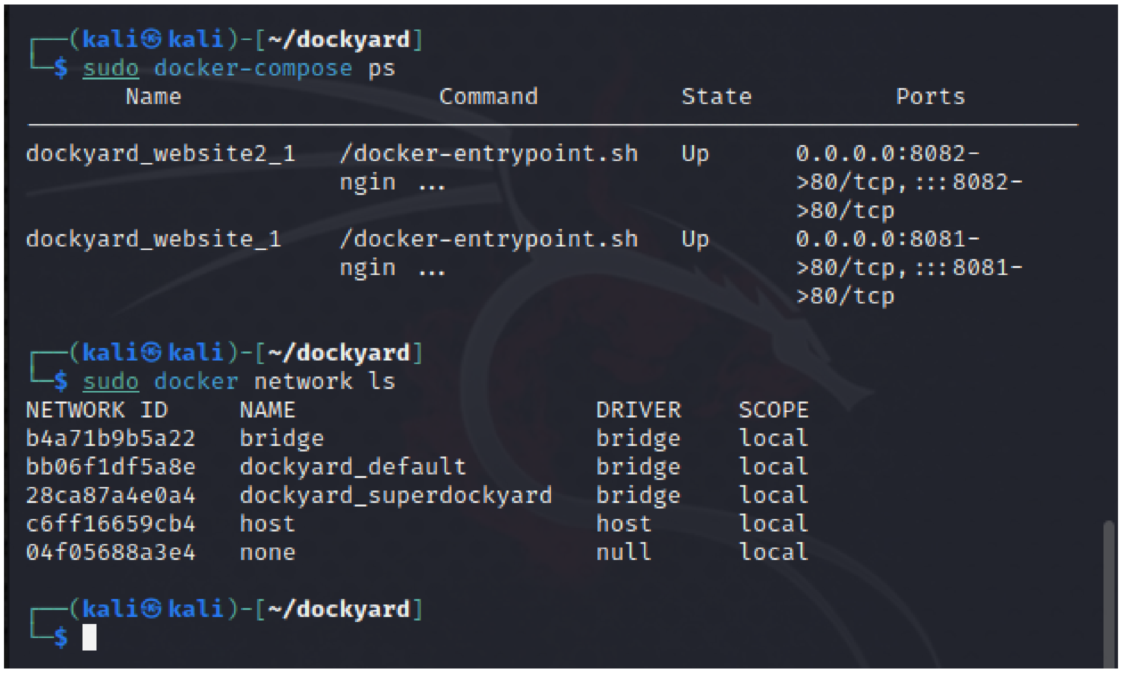
```
┌──(kali㉿kali)-[~]
└─$ sudo docker network create superdockyard --subnet 192.168.92.0/24
[sudo] password for kali:
5086d143bb14803274f3aa8cd4b205850a5b46c947523884eeb6e71f470c36a2

┌──(kali㉿kali)-[~]
└─$ 
```

2. An example of changes made to the docker-compose.yaml file include the following below We have now added a separate network named “superdockyard” and assigned “website2 to it on port 8082:80”

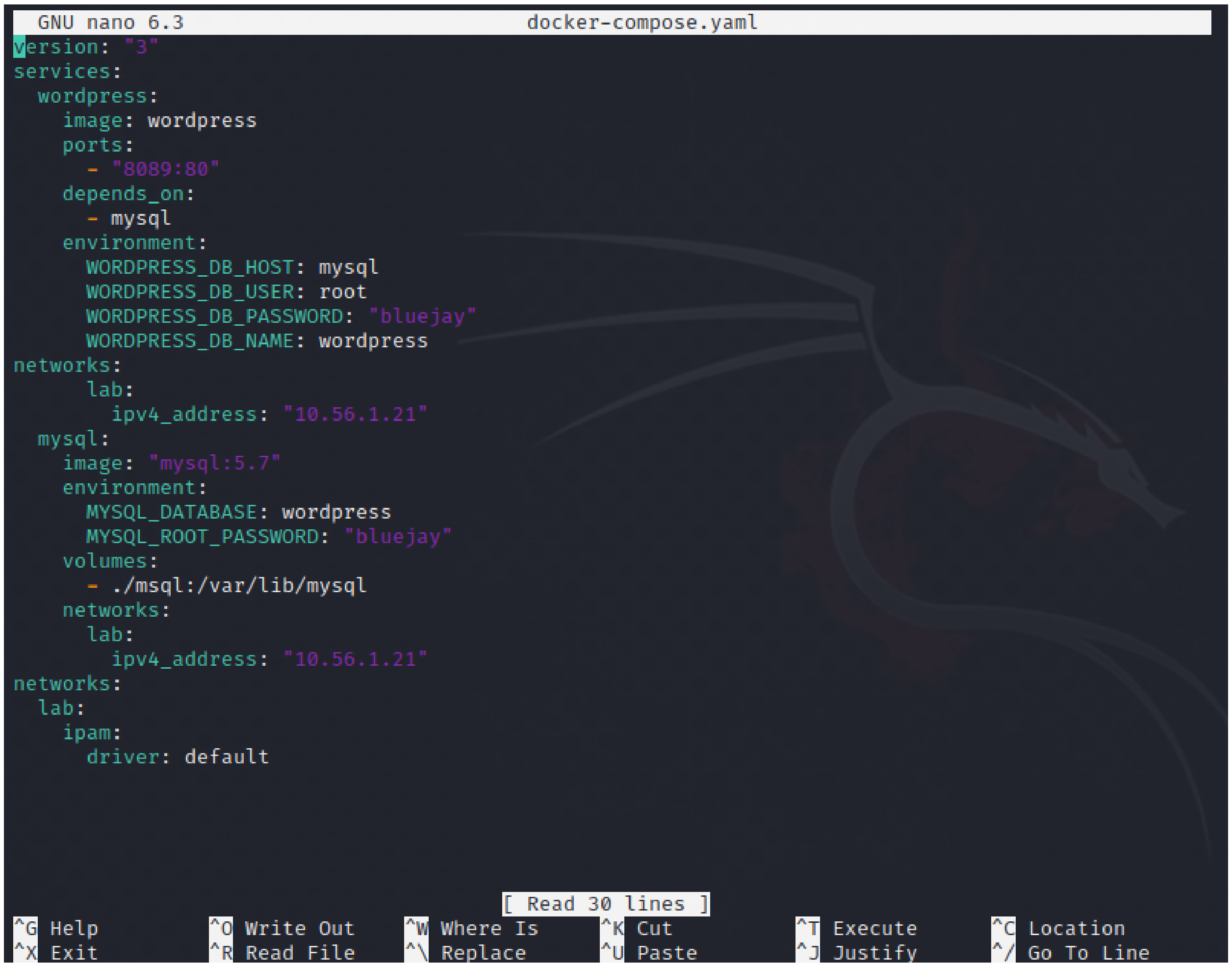


```
  GNU nano 6.3                    docker-compose.yaml
version: "3"
services:
  website:
    image: nginx
    ports:
      - "8081:80"
    restart: always
  website2:
    image: nginx
    ports:
      - "8082:80"
    restart: always
    networks:
      superdockyard:
        ipv4_address: 192.168.92.4
    restart: always
networks:
  superdockyard:
    ipam:
      driver: default
      config:
        - subnet: "192.168.92.0/24"
```

3. We can view the newly created networks with separate websites assigned to them after running one command to execute the .yaml   Also the command
   sudo docker network ls
   will provide us with a list of running networks on the machine.
   sudo docker-compose ps
   will provide a list of machines with status.



```
┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker-compose ps
        Name                      Command               State               Ports
─────────────────────────────────────────────────────────────────────────────────────
dockyard_website2_1    /docker-entrypoint.sh    Up      0.0.0.0:8082-
                       ngin ...                         >80/tcp,:::8082-
                                                        >80/tcp
dockyard_website_1     /docker-entrypoint.sh    Up      0.0.0.0:8081-
                       ngin ...                         >80/tcp,:::8081-
                                                        >80/tcp

┌──(kali㉿kali)-[~/dockyard]
└─$ sudo docker network ls
NETWORK ID     NAME                     DRIVER    SCOPE
b4a71b9b5a22   bridge                   bridge    local
bb06f1df5a8e   dockyard_default         bridge    local
28ca87a4e0a4   dockyard_superdockyard   bridge    local
c6ff16659cb4   host                     host      local
04f05688a3e4   none                     null      local

┌──(kali㉿kali)-[~/dockyard]
└─$ 
```



```
  GNU nano 6.3                          docker-compose.yaml
version: "3"
services:
  wordpress:
    image: wordpress
    ports:
      - "8089:80"
    depends_on:
      - mysql
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: "bluejay"
      WORDPRESS_DB_NAME: wordpress
networks:
      lab:
        ipv4_address: "10.56.1.21"
  mysql:
    image: "mysql:5.7"
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_ROOT_PASSWORD: "bluejay"
    volumes:
      - ./msql:/var/lib/mysql
    networks:
      lab:
        ipv4_address: "10.56.1.21"
networks:
  lab:
    ipam:
      driver: default




                                        [ Read 30 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line
```