Create a Custom Bridge Docker Network:

```
sudo docker network create –d bridge \
      ––subnet 192.168.0.0/24 \
      ––gateway 192.168.0.1 \
      is_network;
```

Confirm it works:

```
# Start the Alpine container, but this time we'll use our custom network.
docker run \
       ––rm –it \
       ––name Alpine_20 \
       ––network is_network \
       alpine sh;

# Ping the custom IP address we set up.
ping –c 3 192.168.0.2;
exit
```

Now we have a legit implementation of connecting to your Docker host over a custom network with a static IP address. Nice!

Create MySQL container:

```
# Create dir for database data
sudo mkdir –p /opt/mysql

# Create MySQL container

sudo mkdir –p /home/Data/mysql_100
sudo docker run –d \
   ––name MySQL_100 \
   ––network is_network \
   ––restart always \
   –e MYSQL_ROOT_PASSWORD= "Administrator2" \
   –v /home/Data/mysql_100:/var/lib/mysql \
   –p 8301:3306 \
   mysql;
mysql –u root –pAdministrator2 –h 192.168.1.20 –P 8301

sudo mkdir –p /home/Data/mysql_200
sudo docker run –d \
   ––name MySQL_200 \
   ––network is_network \
   ––restart always \
   –e MYSQL_ROOT_PASSWORD= "Administrator2" \
   –v /home/Data/mysql_200:/var/lib/mysql \
   –p 8302:3306 \
   mysql;

mysql –u root –pAdministrator2 –h 192.168.1.20 –P 8302
```

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Administrator2';
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY 'Administrator2';

SELECT host, user, plugin, authentication_string FROM user ORDER BY user;

CREATE USER 'is_derayo'@'localhost' IDENTIFIED BY 'Administrator2';
GRANT ALL PRIVILEGES ON *.* TO 'is_derayo'@'localhost' WITH GRANT OPTION;
CREATE USER 'is_derayo'@'%' IDENTIFIED BY 'Administrator2';
GRANT ALL PRIVILEGES ON *.* TO 'is_derayo'@'%' WITH GRANT OPTION;

ALTER USER 'is_derayo'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Administrator2';
ALTER USER 'is_derayo'@'%' IDENTIFIED WITH mysql_native_password BY 'Administrator2';

SELECT host, user, plugin, authentication_string FROM user ORDER BY user;

Create phpMyAdmin container:

```
# PMA_HOST is the IP or domain of the MySQL server,
# so we can use the MySQL container name as the domain
# cause the Docker network create the route as a DNS server.
sudo docker run -d \
    --name phpmyadmin_100 \
    --network is_network \
    --restart always \
    -e PMA_HOST=MySQL_100 \
    -p 8303:80 \
    phpmyadmin/phpmyadmin;

sudo docker run -d \
    --name phpmyadmin_200 \
    --network is_network \
    --restart always \
    -e PMA_HOST=MySQL_200 \
    -p 8304:80 \
    phpmyadmin/phpmyadmin;

192.168.1.20:8303
192.168.1.20:8304
http://192.168.1.101:8303

sudo docker update --restart always MySQL_200;
sudo docker update --restart always phpmyadmin_200;

mysql -u root -pAdministrator2 -h 192.168.1.101 -P 8301
sudo -u postgres psql -d postgres -h 192.168.1.101 -p 8313;
```

Remove images from docker:

```
# - List images - #
sudo docker images;
sudo docker images -a;

# - Remove images - #
sudo docker rmi name1 name2;

# - List unused images - #
sudo docker images -f dangling=true

# - Remove unused images - #
sudo docker images purge;
```

Remove volumes from docker:

```
# - List volumes - #
sudo docker volume ls;

# - List unused volumes - #
sudo docker volume ls -f dangling=true;

# - Remove unused volumes - #
sudo docker volume prune;

# - Remove container with its volume - #
sudo docker rm -v ContainerName;

# - Remove unused data - #
sudo docker system prune;
# - Very important - #
sudo docker system prune -a --volumes;

# - Show docker disk usage - #
sudo docker system df;

# - Show system-wide information - #
sudo docker system info;
sudo docker --version;
sudo docker version;
sudo docker info;



# - Get real time events from the server - #
sudo docker system events;
```

Remove container from docker

```
# - List container - #
sudo docker ps;
sudo docker ps -a;

# - Stop container - #
sudo docker stop ContainerName;

# - Remove container - #
sudo docker rm ContainerName ContainerName;
```

Ubuntu Server Container

```
# - Run Ubuntu image container - #
sudo docker run -d \
    --name UbuntuImg \
    --network is_network \
    --restart always \
    -it --entrypoint /bin/sh \
    -v /home/Data/UbuntuImg/Workspaces:/src \
    -p 8320:80 \
    ubuntu;
```

```
docker run \
  --name ubuntu \
  -e HOST_IP=$(ifconfig en0 | awk '/ *inet /{print $2}') \
  -v /home/Data/UbuntuImg/Workspaces:/src \
  -t -i \
  ubuntu /bin/bash;
```

```
sudo docker stop UbuntuImg;
sudo docker rm UbuntuImg;

sudo docker ps;
sudo docker ps -a;

sudo docker images;
sudo docker system df;
sudo docker system df -v;
```

```
# - Exec UbuntuImg console- #

apt update;
apt-get install -y curl;
apt-get update && apt-get install -y ubuntu-server;
```

 # - How to create Docker Images with a Dockerfile: Nginx webserver - #

```
mkdir -p /home/Data/Docker/ImageBuild
cd /home/Data/Docker/ImageBuild
touch Dockerfile
vim Dockerfile
vim default
vim supervisord.conf
vim start.sh
chmod +x start.sh
sudo docker build -t nginx_image .
sudo docker images;

mkdir -p /home/Data/Docker/ImageBuild/Nginx/webroot


sudo docker run -d \
    --name hakase \
    --network is_network \
    --restart always \
    -v /home/Data/Docker/ImageBuild/Nginx/webroot:/var/www/html \
    -p 8322:80 \
    nginx_image;

sudo docker run -d \
    --name nWebServer1 \
    --network is_network \
    --restart always \
    -v /home/Data/Docker/ImageBuild/Nginx/webroot:/var/www/html \
    -p 8322:80 \
    nginx_image;




sudo docker ps;

echo '<h1>Nginx and PHP-FPM 7 inside Docker Container</h1>' > webroot/index.html

curl 192.168.1.101:8322
curl -I 192.168.1.101:8322

echo '<?php phpinfo(); ?>' > webroot/info.php

http://192.168.1.101:8322/
http://192.168.1.101:8322/info.php

http://192.168.1.101:8322/info.php


sudo docker stop hakase;
sudo docker rm hakase;
```

 # - Ubuntu Server Container - #

 # - Dockerfile for Ubuntu Server - #

```
cd /home/Data/Docker/ImageBuild/uServer
vim /home/Data/Docker/ImageBuild/uServer/Dockerfile


FROM ubuntu:16.04
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update && apt-get install -y ubuntu-server

sudo docker build -t ubuntu-server .
```

 # - Run Ubuntu image container - #

```
sudo docker run -d \
    --name uServer \
    --network is_network \
    --restart always \
    -it --entrypoint /bin/sh \
    -v /home/Data/UbuntuImg/Workspaces:/src \
    -p 8323:80 \
    ubuntu-server;
```

 # - Run Ubuntu image container - #

```
sudo docker login;
sudo docker tag ubuntu-server isderayo/userver:v.0.01
sudo docker push isderayo/userver:v.0.01

sudo docker pull isderayo/userver:v.0.01

sudo docker save isderayo/userver > userver.tar
```

 # - Stop/remove container - #

```
sudo docker stop uServer;
sudo docker rm uServer;
```

 # - Remove images - #

```
sudo docker rmi ubuntu-server;
sudo docker rmi isderayo/userver:v.0.01;
sudo docker rmi isderayo/get-started:part1;
```

 # - Pull my uServer image - #

```
sudo docker pull isderayo/userver:v.0.01
```

```
# - Run my uServer image - #
sudo docker run -d \
    --name uServer \
    --network is_network \
    --restart always \
    -it --entrypoint /bin/sh \
    -v /home/Data/UbuntuImg/Workspaces:/src \
    -p 8323:80 \
    isderayo/userver:v.0.01;



# - Configuring uServer image - #
apt install openssh-server;
passwd root;
apt-get install vsftpd;

vim /etc/vsftpd.conf
write_enable=YES
local_enable=YES

vim /etc/ftpusers
is_derayo

service vsftpd restart;

vim /etc/ssh/sshd_config
PermitRootLogin yes
PasswordAuthentication yes

service ssh restart;

# - Images list - #
sudo docker ps;




# - Edit webpage - #
vim /home/Data/Docker/uWebServer1/docker.html

# - Paste - #
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Learn Docker at www.isdevelopment.us</title>
</head>
<body>
    <h1>Learn Docker With Us</h1>
</body>
</html>

# - Firefox - #
http://192.168.1.101:8324/
```

```
# - Setting up an Apache Container - #

sudo docker run -dit \
    --name aWebServer1 \
    --network is_network \
    --restart always \
    -v /home/Data/Docker/uWebServer1/:/usr/local/apache2/htdocs/ \
    -p 8324:80 \
    httpd:2.4;



# - Configuring Nginx WebServer image - #


sudo docker run -d --name docker-nginx -p 8385:80 nginx

sudo docker run -d \
     --name nWebServer2 \
     --network is_network \
     --restart always \
     -p 8385:80 \
     -v /home/Data/Docker/nWebServer2/html:/usr/share/nginx/html \
     nginx


sudo docker run -d \
     --name nWebServer2 \
     --network is_network \
     --restart always \
     -p 8385:80 \
     -v /home/Data/Docker/nWebServer2/:/usr/share/nginx/html \
     nginx;



# - Using your own Nginx configuration file - #

cd /home/Data/Docker/nWebServer2/
sudo docker cp nWebServer2:/etc/nginx/conf.d/default.conf default.conf

sudo docker stop nWebServer2;
sudo docker rm nWebServer2;

sudo docker run -d \
     --name nWebServer2 \
     --network is_network \
     --restart always \
     -p 8385:80 \
     -v /home/Data/Docker/nWebServer2/html:/usr/share/nginx/html \
     -v /home/Data/Docker/nWebServer2/default.conf:/etc/nginx/conf.d/default.conf \
     nginx;

# - Restart Nginx container after modifying default.conf file - #
sudo docker restart nWebServer2;
```