Docker Installation.

Step 1:-From OS level requirement.

#yum install -y yum-utils device-mapper-persistent-data lvm2

#cat /etc/selinux/config |grep "SELINUX=" |grep -v "#"
SELINUX=disabled
#getenforce
Disabled

Step2 :- Add docker repo.

#yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
#yum-config-manager --enable docker-ce-nightly
#cd /etc/yum.repos.d && cat docker-ce.repo
#yum-config-manager --disable docker-ce-nightly → Optional to disable the repo.

Step3 :- Install Docker Engine, start/enable docker engine.

#yum install docker-ce docker-ce-cli containerd.io
systemctl start docker && systemctl enable docker && systemctl status docker

Docker Lab Practice.

@Check the version.
docker -v
Docker version 19.03.3, build a872fc2f86

#docker --help

@Check whether docker Images are available
: -What is image? Image is the form of all required binary bundles and setup, either it will be a copy of exiting image or kind of snapshot, which will be in rest mode and the copy of image if we run is called container and the multiple container can be run from then same image.

#docker images

```
[root@ansikube ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@ansikube ~]#
```

@Download image from Docker Forum.

: -This command will download the latest version.

#docker pull nginx

```
[root@ansikube ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
8d691f585fa8: Pull complete
047cb16c0ff6: Pull complete
b0bbed1a78ca: Pull complete
Digest: sha256:1b75cccb59e95f892790ed7fe0626196d4382155c906eb27bd7ecf595ad67ada
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[root@ansikube ~]#
```

#docker images

```
[root@ansikube ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              5a9061639d0a        3 hours ago         126MB
[root@ansikube ~]#
```

@Run the Docker image

Two types of modes are there, detached mode (-d) and Container interactive mode (-i). Detach mode the process will be detach from terminal and run it on back ground, but interactive mode the process will be open in terminal mode and when we interrupt it will get exit. For each container will have unique container ID.

#docker run -d nginx

```
[root@ansikube ~]# docker run -d nginx
4952324e7919a93573a38b980a10513b816b4547c64e260438f4492763ad6573
[root@ansikube ~]#
```

@Try to download and run lower version of nginx in detach mode

#docker run --name web1 -d nginx:1.17.0

```
[root@ansikube ~]# docker run --name web1 -d nginx:1.17.0
Unable to find image 'nginx:1.17.0' locally
1.17.0: Pulling from library/nginx
fc7181108d40: Pull complete
c4277fc40ec2: Pull complete
780053e98559: Pull complete
Digest: sha256:bdbf36b7f1f77ffe7bd2a32e59235dff6ecf131e3b6b5b96061c652f30685f3a
Status: Downloaded newer image for nginx:1.17.0
616fcee8177af62a83d5e2bd79cbab205f311bf5d02f22706b14c88150b3862d
[root@ansikube ~]#
```

@To check the status of container.

#docker ps

```
[root@ansikube ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
616fcee8177a        nginx:1.17.0        "nginx -g 'daemon of…"   56 seconds ago      Up 53 seconds       80/tcp              web1
4952324e7919        nginx               "nginx -g 'daemon of…"   3 minutes ago       Up 2 minutes        80/tcp              elated_burnell
[root@ansikube ~]#
```

@To stop Container (Stop the container with container ID)

docker stop 616fcee8177a

```
[root@ansikube ~]# docker stop 616fcee8177a
616fcee8177a
[root@ansikube ~]#
```

@To check detailed docker process

#docker ps -a

```
[root@ansikube ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS               NAMES
616fcee8177a        nginx:1.17.0        "nginx -g 'daemon of…"   3 minutes ago       Exited (0) About a minute ago                       web1
4952324e7919        nginx               "nginx -g 'daemon of…"   5 minutes ago       Up 5 minutes                    80/tcp              elated_burne
ll
```

@To Start container & Restart container.

#docker start web1

#docker restart web1

```
[root@ansikube ~]# docker start web1
web1
[root@ansikube ~]#
```

@Direct Run and Login into container with bash shell

Running the container from nginx image with interactive mode(-it) and get logged into container.

#docker run --name web3 -it nginx /bin/bash

```
[root@ansikube ~]# docker run --name web3 -it nginx /bin/bash
root@4b90d9658ddb:/# uname -a
Linux 4b90d9658ddb 3.10.0-1062.1.2.el7.x86_64 #1 SMP Mon Sep 30 14:19:46 UTC 2019 x86_64 GNU/Linux
root@4b90d9658ddb:/# date
Thu Oct 17 07:32:53 UTC 2019
root@4b90d9658ddb:/#
```

@To Remove Exited container

All exited container from terminal, will be in exited and persist on host, so to remove below command.

#docker ps -a

```
[root@ansikube ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS               NAMES
4b90d9658ddb        nginx               "/bin/bash"              About a minute ago  Exited (130) 1 second ago                           web3
616fcee8177a        nginx:1.17.0        "nginx -g 'daemon of…"   13 minutes ago      Up 8 minutes                    80/tcp              web1
4952324e7919        nginx               "nginx -g 'daemon of…"   15 minutes ago      Up 15 minutes                   80/tcp              elated_burnell
[root@ansikube ~]#
```

#docker rm 4b90d9658ddb

```
[root@ansikube ~]# docker rm 4b90d9658ddb
4b90d9658ddb
[root@ansikube ~]#
```

@Run new container in detach mode.

To run the container in detached mode (-d) from existing image.

#docker ps -a

#docker run --name web2 -d nginx

```
[root@ansikube ~]# docker run --name web2 -d nginx
e8525c185605d88175678eb100c9b51b83e67d081f7167ad6abde3d6a83299ec
```

#docker ps

```
[root@ansikube ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
e8525c185605        nginx               "nginx -g 'daemon of…"   2 minutes ago       Up 2 minutes        80/tcp              web2
```

Published ports

: - By default, when you create a container, it does not publish any of its ports to the outside world. To make a port available to services outside of Docker, or to Docker containers which are not connected to the container's network, use the --publish or -p flag. This creates a firewall rule which maps a container port to a port on the Docker host.

--publish or -p flag (we can specify the port)

#docker run --name web3 -p 80:80 -d nginx

#docker ps

```
[root@ansikube ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
70b238c25ec8        nginx               "nginx -g 'daemon of…"   23 seconds ago      Up 22 seconds       0.0.0.0:80->80/tcp     web3
```

: -Try to open from outside network with your base host ip or public ip, which will be access via 80 port.

Example :- http://34.68.63.0/

@Creating one more container from existing image with different port(8080).

#docker run --name web4 -p 8080:80 -d nginx

#docker ps

```
[root@ansikube ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
77a7fc5d2b0a        nginx               "nginx -g 'daemon of…"   22 seconds ago      Up 22 seconds       0.0.0.0:8080->80/tcp     web4
70b238c25ec8        nginx               "nginx -g 'daemon of…"   4 minutes ago       Up 4 minutes        0.0.0.0:80->80/tcp       web3
```

Example :- http://34.68.63.0:8080/

@Run the container from existing image with default port.

:- Flag -P will expose default available port to the container and same can be access from outside. It will randomly take the port from 32000.

docker run --name web5 -d -P nginx

#docker ps

```
[root@ansikube ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                     NAMES
fcb119ee2cb3        nginx               "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes        0.0.0.0:32769->80/tcp     web5
```

Example:- http://34.68.63.0:32769/

Dockerfile reference

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

@To Build image

:- To build an image by using docker file.

#cat Dockerfile

FROM centos:7.6.1810

LABEL maintainer=Deepan

docker build -t build1 .
:- (.) dot will instruct to pick up the file from existing folder.
:- (-f) path of the file.

```
[root@ansikube mainfest]# docker build -t build1 .
Sending build context to Docker daemon  434.7kB
Step 1/2 : FROM centos:7.6.1810
7.6.1810: Pulling from library/centos
ac9208207ada: Pull complete
Digest: sha256:62d9e1c2daa91166139b51577fe4f4f6b4cc41a3a2c7fc36bd895e2a17a3e4e6
Status: Downloaded newer image for centos:7.6.1810
 ---> f1cb7c7d58b7
Step 2/2 : LABEL maintainer=Deepan
 ---> Running in 83ce52974ca6
Removing intermediate container 83ce52974ca6
 ---> 9f9aa843ed04
Successfully built 9f9aa843ed04
Successfully tagged build1:latest
[root@ansikube mainfest]#
```

docker images

```
[root@ansikube mainfest]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
build1              latest              9f9aa843ed04        6 minutes ago       202MB
```

#docker run -it build1 /bin/bash

```
[root@f08906c1d96e /]# uname -a
Linux f08906c1d96e 3.10.0-1062.1.2.el7.x86_64 #1 SMP Mon Sep 30 14:19:46 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
[root@f08906c1d96e /]#
```

@Add multiple instruction in Docker conf file to build image.
:-Adding package install
#cat Dockerfile
FROM centos:7.6.1810
LABEL maintainer=Deepan
RUN yum update -y
RUN yum install httpd -y

#docker build -t build2 .
#docker images
#docker run -it build2 /bin/bash

```
[root@49e005d39a34 /]# yum list httpd
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: repos.forethought.net
 * extras: mirrors.cmich.edu
 * updates: mirror.grid.uchicago.edu
Installed Packages
httpd.x86_64                                          2.4.6-90.el7.centos
[root@49e005d39a34 /]#
```

:-Adding environment ENV
Note: - env parameter will be exist after the image build also.
#cat Dockerfile
FROM centos:7.6.1810
LABEL maintainer=Deepan
ENV DBHOST mydb
RUN yum update -y
RUN yum install httpd -y

#docker build -t build3 .
#docker images
#docker run -it build3 /bin/bash

:-Adding argument
Note:- ARG will be exist till the image build, where as ENV will be present after the build image.
#cat Dockerfile
FROM centos:7.6.1810
LABEL maintainer=Deepan
ENV DBHOST mydb
ARG HELLO=Deepan

#docker build -t build4 .
#docker images
#docker run -it build4 /bin/bash

:-Adding commands
Note:- To start main process in container during the build by using CMD, it will run whenever the new container run.

#cat Dockerfile
FROM centos:7.6.1810
LABEL maintainer=Deepan
CMD date

#docker build -t build6 .
#docker images
#docker run -it build6 /bin/bash
Note:- build6 will be exit because date is command in linux will run and exited immediately.

:-Overriding the command from the build image.
docker images

```
[root@ansikube mainfest]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
build2              latest              567d3affa79c        19 minutes ago      526MB
build1              latest              9f9aa843ed04        34 minutes ago      202MB
```

#docker run -d build2 uptime

```
[root@ansikube mainfest]# docker run -d build2 uptime
aab75a23ee07585154f94444f7841a29133f9dab2b5b8411cce72baa5043dc4d
```

#docker ps -a

```
[root@ansikube mainfest]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
aab75a23ee07        build2              "uptime"            14 seconds ago      Exited (0) 13 seconds ago                       vigorous_c
hatelet
```

docker logs -f aab75a23ee07585154f94444f7841a29133f9dab2b5b8411cce72baa5043dc4d:- logs -f
Note:- will be used to read the recent logs from the container.

```
[root@ansikube mainfest]# docker logs -f aab75a23ee07585154f94444f7841a29133f9dab2b5b8411cce72baa5043dc4d
 00:54:26 up 17:45,  0 users,  load average: 0.00, 0.01, 0.10
[root@ansikube mainfest]#
```

:-Adding ENTRYPOINT

Note:- When the ENTRYPOINT directive is used to build an image, then we cannot override with another command.

cat Dockerfile
FROM centos:7.6.1810
LABEL maintainer=Deepan
ENTRYPOINT date

#docker build -t build5 .
#docker images
#docker run -d build5
#docker logs -f 0ead39050c59c5d0deb31d54639f24078c7e7ecdd2de2ff835fdcf88cce6b131

```
#docker run -d build5
0ead39050c59c5d0deb31d54639f24078c7e7ecdd2de2ff835fdcf88cce6b131
#docker logs -f 0ead39050c59c5d0deb31d54639f24078c7e7ecdd2de2ff835fdcf88cce6b131
Sat Jun 15 16:08:37 UTC 2019
```

#docker run -d build5 uptime
#docker logs -f d6b52c2bd258c294c08fc1f3308b93b6fa5f8d984e6b3c5fd2c623dae06d4dc3

```
#docker run -d build5 uptime
d6b52c2bd258c294c08fc1f3308b93b6fa5f8d984e6b3c5fd2c623dae06d4dc3
#docker logs -f d6b52c2bd258c294c08fc1f3308b93b6fa5f8d984e6b3c5fd2c623dae06d4dc3
Sat Jun 15 16:09:20 UTC 2019
```

@To push/pull the docker images with help of docker hub repository.

:- Make sure to have docker logins created. (Example ID of mine :- deepu1986, kindly use your ID)
https://hub.docker.com/

Build the image by using dockerfile.

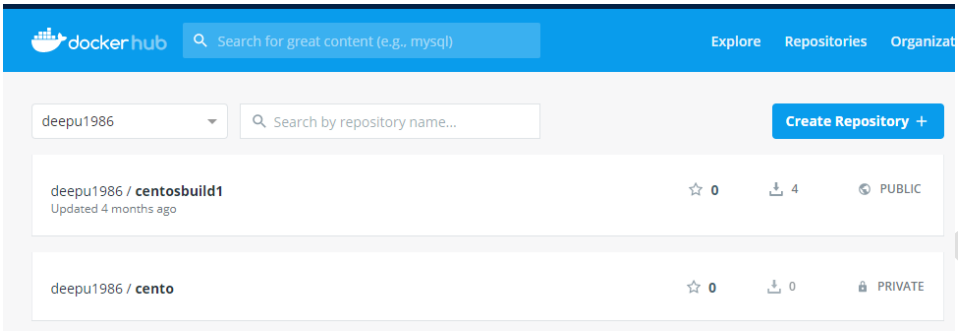
#docker build -t deepu1986/centosbuild1 .

```
#docker build -t deepu1986/centosbuild1 .

Sending build context to Docker daemon  38.91kB
Step 1/3 : FROM centos:7.6.1810
 ---> f1cb7c7d58b7
Step 2/3 : LABEL maintainer=Deepan
 ---> Using cache
 ---> 6cad46825c77
Step 3/3 : ENTRYPOINT date
 ---> Using cache
 ---> 671676f89b8a
Successfully built 671676f89b8a
Successfully tagged deepu1986/centosbuild1:latest
```

docker push deepu1986/centosbuild1

```
#docker push deepu1986/centosbuild1

The push refers to repository [docker.io/deepu1986/centosbuild1]
89169d87dbe2: Preparing
```

Sign in to this link https://hub.docker.com/ and verify the image which we have pushed into docker hub.



docker pull linuxkid/centosbuild1
docker images

@ Create Docker conf file for build with copy directive

COPY will do copy from local to container, however ADD also will do the copy and in addition it will download from remote local to container directly, exactly it will work like wget.

#cat Dockerfile

```
FROM centos:7.6.1810
LABEL maintainer=Deepan
COPY index.txt /tmp
```

:-create a file on current working directory.

#touch index.txt

#docker build -t build7 .

```
#docker build -t build7 .
Sending build context to Docker daemon  39.42kB
Step 1/3 : FROM centos:7.6.1810
 ---> f1cb7c7d58b7
Step 2/3 : LABEL maintainer=Deepan
 ---> Using cache
 ---> 6cad46825c77
Step 3/3 : COPY index.txt /tmp
 ---> 73dabe1bf198
Successfully built 73dabe1bf198
Successfully tagged build7:latest
```

#docker images

#docker run -it build7 /bin/bash

```
[root@7b8eee8c797d /]# ls -lrt /tmp
total 4
-rw------- 1 root root   0 Dec  4  2018 yum.log
-rwx------ 1 root root 836 Dec  4  2018 ks-script-6pKh_p
-rw-r--r-- 1 root root   0 Jun 15 16:48 index.txt
[root@7b8eee8c797d /]#
```

Container links

Establish the connectivity between the containers is called links.
The information in this section explains legacy container links within the Docker default bridge network which is created automatically when you install Docker.
Before the Docker networks feature, you could use the Docker link feature to allow containers to discover each other and securely transfer information about one container to another container. With the introduction of the Docker networks feature, you can still create links but they behave differently between default bridge network and user defined networks.

@ To create connectivity between two containers by using link
Ref: - with mysql and wordpress
https://hub.docker.com/_/mysql
https://hub.docker.com/_/wordpress

#docker images
#docker ps -a

```
[root@ansikube mainfest]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                     NAMES
fcb119ee2cb3        nginx               "nginx -g 'daemon of…"   2 hours ago         Up 2 hours          0.0.0.0:32769->80/tcp     web5
77a7fc5d2b0a        nginx               "nginx -g 'daemon of…"   2 hours ago         Up 2 hours          0.0.0.0:8080->80/tcp      web4
70b238c25ec8        nginx               "nginx -g 'daemon of…"   3 hours ago         Up 3 hours          0.0.0.0:80->80/tcp        web3
e8525c185605        nginx               "nginx -g 'daemon of…"   3 hours ago         Up 3 hours          80/tcp                    web2
616fcee8177a        nginx:1.17.0        "nginx -g 'daemon of…"   19 hours ago        Up 19 hours         80/tcp                    web1
4952324e7919        nginx               "nginx -g 'daemon of…"   19 hours ago        Up 19 hours         80/tcp                    elated_burnell
[root@ansikube mainfest]#
```

@Download the mysql image from docker hub (https://hub.docker.com/_/mysql)

#docker run --name robodb -d -e MYSQL_ROOT_PASSWORD=redhat -e MYSQL_DATABASE=wordpress -e MYSQL_USER=sqladmin -e MYSQL_PASSWORD=redhat123 -e MYSQL_DATABASE=wordpress mysql:5.7

```
[root@ansikube mainfest]# docker run --name robodb -d -e MYSQL_ROOT_PASSWORD=redhat -e MYSQL_DATABASE=wordpress -e MYSQL_USER=sqladmin -e MYSQL_PASSW
ORD=redhat123 -e MYSQL_DATABASE=wordpress mysql:5.7
Unable to find image 'mysql:5.7' locally
5.7: Pulling from library/mysql
80369df48736: Pull complete
e8f52315cb10: Pull complete
cf2189b391fc: Pull complete
cc98f645c682: Pull complete
27a27ac83f74: Pull complete
fa1f04453414: Pull complete
d45bf7d22d33: Pull complete
c7d49ffebc56: Pull complete
511a8052b204: Pull complete
5d5df4c12444: Pull complete
d482603a2922: Pull complete
Digest: sha256:44b33224e3c406bf50b5a2ee4286ed0d7f2c5aec1f7fdb70291f7f7c570284dd
Status: Downloaded newer image for mysql:5.7
3a1cbc6879602437129390c010e709969ae0e492c4acb857f5c73d1353326dfd5
[root@ansikube mainfest]#
```

@Run word press image
: - it will download the image and run with publish port to access from outside network.
docker run --name WP -p 8080:80 -d wordpress
#docker images

```
[root@ansikube mainfest]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
build2              latest              567d3affa79c        2 hours ago         526MB
build1              latest              9f9aa843ed04        3 hours ago         202MB
mysql               5.7                 cd3ed0dfff7e        22 hours ago        437MB
nginx               latest              5a9061639d0a        22 hours ago        126MB
wordpress           latest              fc466e7eaa8e        2 days ago          542MB
nginx               1.17.0              719cd2e3ed04        4 months ago        109MB
centos              7.6.1810            f1cb7c7d58b7        7 months ago        202MB
[root@ansikube mainfest]#
```

#docker ps -a

```
[root@ansikube mainfest]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
a2ba1f216d91        wordpress           "docker-entrypoint.s…"   3 minutes ago       Up 2 minutes        0.0.0.0:8080->80/tcp     WP
3a1cbc687960        mysql:5.7           "docker-entrypoint.s…"   31 minutes ago      Up 31 minutes       3306/tcp, 33060/tcp      robodb
[root@ansikube mainfest]#
```

: -Try to access from web

http://34.68.63.0:8080 →use your system ip → select language → click lets go→ then enter below details.

: - Enter the below info on webpage

Database Name wordpress

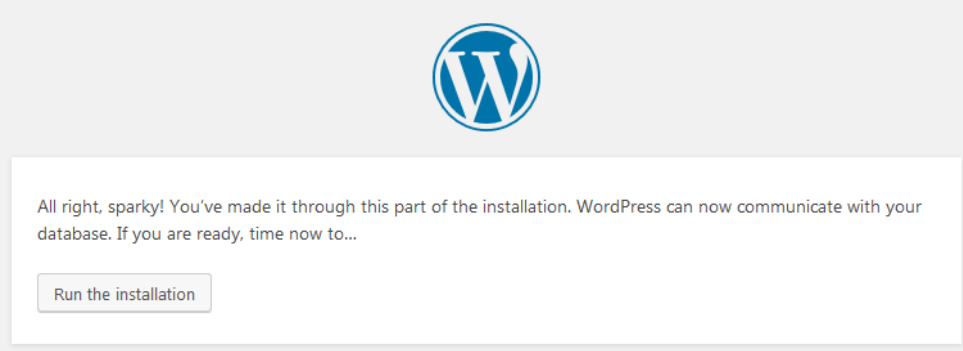
Username sqladmin

Password redhat123

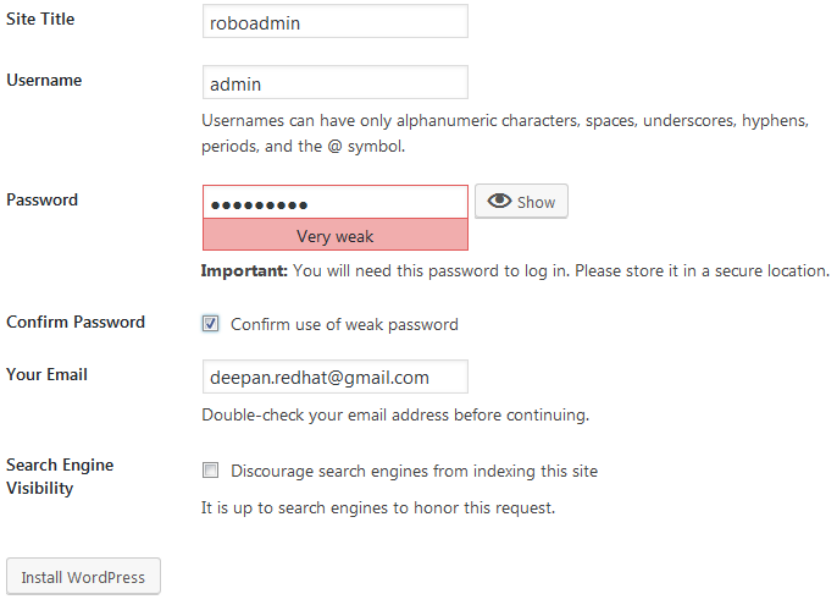
Database Host < 172.17.0.8> -->sql DB ip address

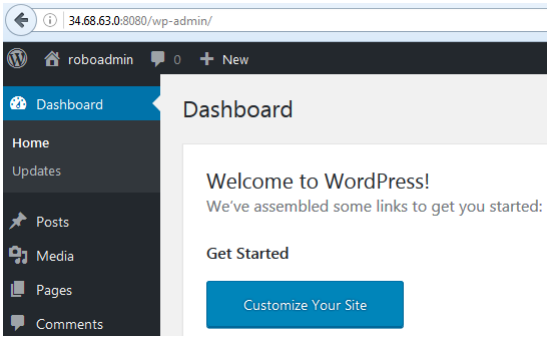
:- To get particular container ID info, like ip address etc..

#docker inspect <container ID>



:-Give below info according to your requirement.





Note: - above setup has done without link and there might be a chance for ip changes incase if container got restarted! so to avoid that, link will be helpful to manage the ip changes dynamically.

#docker ps -a

#docker stop <container id>

@run the sql and wordpress setup again.

#docker run --name robodb -d -e MYSQL_ROOT_PASSWORD=redhat -e MYSQL_DATABASE=wordpress -e MYSQL_USER=sqladmin -e MYSQL_PASSWORD=redhat123 -e MYSQL_DATABASE=wordpress mysql:5.7

docker run --name wp -p 8080:80 -d --link robodb:robodb2 -e WORDPRESS_DB_HOST=robodb2 -e WORDPRESS_DB_USER=sqladmin -e WORDPRESS_DB_PASSWORD=redhat123 -e WORDPRESS_DB_NAME=wordpress wordpress

#docker ps -a

```
[root@ansikube ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
9e20a0afff8b        wordpress           "docker-entrypoint.s…"   23 seconds ago      Up 22 seconds       0.0.0.0:8080->80/tcp     wp
c176ef740907        mysql:5.7           "docker-entrypoint.s…"   About a minute ago  Up About a minute   3306/tcp, 33060/tcp      robodb
[root@ansikube ~]#
```

#docker exec -it 9e20a0afff8b /bin/bash

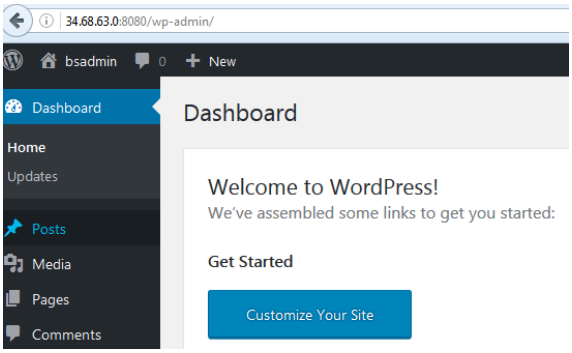
```
[root@ansikube ~]# docker exec -it 9e20a0afff8b /bin/bash
root@9e20a0afff8b:/var/www/html# env
ROBODB2_PORT_33060_TCP_PROTO=tcp
HOSTNAME=9e20a0afff8b
PHP_VERSION=7.3.10
APACHE_CONFDIR=/etc/apache2
PHP_MD5=
ROBODB2_ENV_MYSQL_DATABASE=wordpress
ROBODB2_PORT_3306_TCP=tcp://172.17.0.2:3306
```

#env

#cat /etc/hosts

172.17.0.2 robodb2 c176ef740907 robodb

@Try to access from web page
http://34.68.63.0:8080



Note: - After created link b/w sql & wordpress container, on webpage it won't ask for details to enter, reason because it's been linked and in case container ip got changed also, it will be managed by this link.

Manage data in Docker

By default, all files created inside a container are stored on a writable container layer. This means that:

- The data doesn't persist when that container no longer exists, and it can be difficult to get the data out of the container if another process needs it.
- A container's writable layer is tightly coupled to the host machine where the container is running. You can't easily move the data somewhere else.
- Writing into a container's writable layer requires a storage driver to manage the filesystem. The storage driver provides a union filesystem, using the Linux kernel. This extra abstraction reduces performance as compared to using data volumes, which write directly to the host filesystem.

Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops: volumes and bind mounts. If you're running Docker on Linux you can also use a tmpfs mount. If you're running Docker on Windows you can also use a named pipe.
Three type of mounts Bind mounts, volumes and tmpfs mounts.
@Create Bind mounts.
Bind mounts may be stored anywhere on the host system. Non-Docker processes on the Docker host or a Docker container can modify them at any time. Better to avoid to use this bind method, because container will be have full access and in case ifs deleted then data wont visible on local host.
: - Create a directory on base host and touch index.html.
#mkdir appdata && cd appdata/ && ls -lrt
#cat index.html

```
[root@ansikube appdata]# cat index.html
you are in confusion, come to the conclusion
[root@ansikube appdata]#
```

docker run --mount type=bind,source=/root/appdata,target=/usr/share/nginx/html/ -p 8080:80 -d nginx

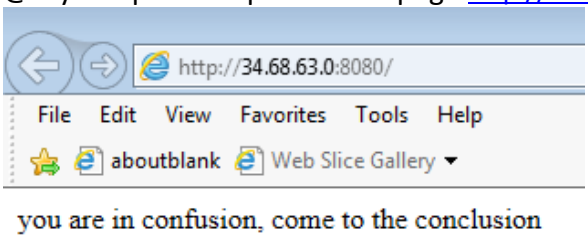
#docker ps -a

```
[root@ansikube appdata]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
e611ba8fe17d        nginx               "nginx -g 'daemon of…"   18 seconds ago      Up 16 seconds       0.0.0.0:8080->80/tcp     vigorous_hellman
[root@ansikube appdata]#
```

docker exec -it e611ba8fe17d /bin/bash

```
[root@ansikube appdata]# docker exec -it e611ba8fe17d /bin/bash
root@e611ba8fe17d:/# cd /usr/share/nginx/html
root@e611ba8fe17d:/usr/share/nginx/html# cat index.html
you are in confusion, come to the conclusion
root@e611ba8fe17d:/usr/share/nginx/html#
```

@Try to open the ip from webpage http://34.68.63.0:8080/



@Create volumes mounts.

Volumes are stored in a part of the host filesystem which is managed by Docker (/var/lib/docker/volumes/ on Linux). Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker. Same volume can be mounted on multiple container.

:- Create mount with volume option, which is advisable and default.

#docker volume create webdata

#docker volume ls

```
[root@ansikube appdata]# docker volume create webdata
webdata
[root@ansikube appdata]# docker volume ls
DRIVER              VOLUME NAME
local               4b0f9be4d2f13b7f9b1778b30767fe4ae45075888fa475ad6712aab6d494b658
local               76ca3645fd9db0726003f74e49dd13caf03c4fa8cf2188518c3bf8c0120c79c5
local               77d23dfeb06982a1591f81cb7f2bbad18cad3895337364bf15c981f27cef938a
local               86c05c04f67525674bd9efb82da18bd6cff844466cfdcc40018f23840e088e1a
local               919ceaffd7dab88dcdd531b307765fc71c41269a0d14dc95e5279a211be86e3b
local               19816f7184cbb030630328855b931ea8ee015b9f73fcc41030fa450320623c46
local               bf69fd3e4cc3cd11f09a1c987bbfc8953be4c809035ae1770dd8c55077c71d0a
local               ca6c6ee0558402025523bf0689fb43d83d802e1d4f3439c6ffea6b2d9fa68d82
local               eeec62452229befe347b944290a9dd5227e03e907050c527edccd943e8bfa921
local               webdata
[root@ansikube appdata]#
```

docker run --mount type=volume,source=webdata,target=/usr/share/nginx/html/ -p 8081:80 -d nginx

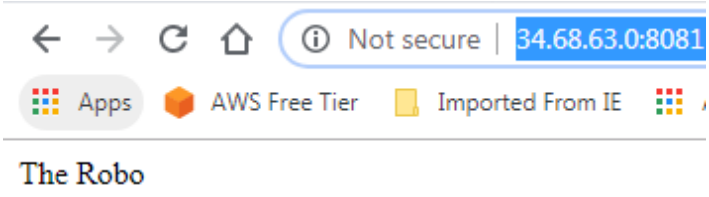
#docker ps -a

```
[root@ansikube appdata]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
a60d131dafd8        nginx               "nginx -g 'daemon of…"   13 seconds ago      Up 12 seconds       0.0.0.0:8081->80/tcp     romantic_colden
```

: -Modify the index.html.

```
[root@ansikube _data]# pwd
/var/lib/docker/volumes/webdata/_data
[root@ansikube _data]# cat index.html
The Robo
[root@ansikube _data]#
```

@Try to open the ip with port on webpage http://34.68.63.0:8081/



@Create one more container and point the same volume.
docker run --mount type=volume,source=webdata,target=/usr/share/nginx/html/ -p 8082:80 -d nginx
: -Try to open the ip with port on webpage http://34.68.63.0:8082/

#docker ps -a
#docker inspect <container ID>
:-Remove the container
for i in `docker ps -a |awk '{print $1}' |grep -v CONTAINER`; do docker stop $i; docker rm $i; done
#docker volume rm webdata
#docker volume ls
@tmpfs mounts.
tmpfs mounts are stored in the host system's memory only and are never written to the host system's filesystem.

Docker Networking

One of the reasons Docker containers and services are so powerful is that you can connect them together or connect them to non-Docker workloads. Docker containers and services do not even need to be aware that they are deployed on Docker, or whether their peers are also Docker workloads or not. Whether your Docker hosts run Linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.
Type of Network drivers (bridge, host, overlay, macvlan, none).

#docker network ls

```
[root@ansikube ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0e2ed481f842        bridge              bridge              local
d22bd9e6ece0        host                host                local
466031e18ac2        none                null                local
[root@ansikube ~]#
```

@bridge network.

The default network driver. If you don't specify a driver, this is the type of network you are creating. Bridge network will create and work on top of host network and

@Create Host network.

host: For standalone containers, remove network isolation between the container and the Docker host, and use the host's networking directly. host is only available for swarm services on Docker 17.06 and higher. bridge network will take the ip from base host only and so we cannot do port map.

#docker run --name web1 --net=host -d nginx

```
[root@ansikube ~]# docker exec -it 17c630a05443 /bin/bash
root@ansikube:/# uname -a
Linux ansikube 3.10.0-1062.1.2.el7.x86_64 #1 SMP Mon Sep 30 14:19:46 UTC 2019 x86_64 GNU/Linux
```

@Create None network.

none: For this container, disable all networking. Usually used in conjunction with a custom network driver. none is not available for swarm services

docker run --name web2 --net=none -d nginx

#docker ps -a

```
[root@ansikube ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
4db2b0bd33ca        nginx               "nginx -g 'daemon of…"   14 seconds ago      Up 13 seconds                           web2
17c630a05443        nginx               "nginx -g 'daemon of…"   4 minutes ago       Up 4 minutes                            web1
```

: - login to the container to verify network details

docker exec -it 4db2b0bd33ca /bin/bash

```
root@4db2b0bd33ca:/# ifconfig -a
bash: ifconfig: command not found
root@4db2b0bd33ca:/# ip a
bash: ip: command not found
root@4db2b0bd33ca:/# apt-get update
Err:1 http://security.debian.org/debian-security buster/updates InRelease
  Temporary failure resolving 'security.debian.org'
Err:2 http://deb.debian.org/debian buster InRelease
  Temporary failure resolving 'deb.debian.org'
Err:3 http://deb.debian.org/debian buster-updates InRelease
  Temporary failure resolving 'deb.debian.org'
Reading package lists... Done
W: Failed to fetch http://deb.debian.org/debian/dists/buster/InRelease  Temporary failure resolving 'deb.debian.org'
W: Failed to fetch http://security.debian.org/debian-security/dists/buster/updates/InRelease  Temporary failure resolving 'security.debian.org'
W: Failed to fetch http://deb.debian.org/debian/dists/buster-updates/InRelease  Temporary failure resolving 'deb.debian.org'
W: Some index files failed to download. They have been ignored, or old ones used instead.
root@4db2b0bd33ca:/#
```

@ Create an own bridge network.

docker network ls

```
[root@ansikube ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0e2ed481f842        bridge              bridge              local
d22bd9e6ece0        host                host                local
466031e18ac2        none                null                local
[root@ansikube ~]#
```

docker network create -d bridge --subnet 10.0.0.1/16 kee
docker network ls

```
[root@ansikube ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0e2ed481f842        bridge              bridge              local
d22bd9e6ece0        host                host                local
63e67b546bdb        kee                 bridge              local
466031e18ac2        none                null                local
[root@ansikube ~]#
```

docker run --name web4 --net=kee -d nginx
#docker ps -a

```
[root@ansikube ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
f5011a73fde8        nginx               "nginx -g 'daemon of…"   10 seconds ago      Up 9 seconds        80/tcp              web4
[root@ansikube ~]#
```

: -Login to the container and run below commands to see the ip details.
docker exec -it f5011a73fde8 /bin/bash

apt-get install net-tools && apt-get install net-tools && ifconfig -a

```
root@f5011a73fde8:/# apt-get update
Get:1 http://security-cdn.debian.org/debian-security buster/updates InRelease [39.1 kB]
Get:2 http://cdn-fastly.deb.debian.org/debian buster InRelease [122 kB]
Get:3 http://cdn-fastly.deb.debian.org/debian buster-updates InRelease [49.3 kB]
Get:4 http://security-cdn.debian.org/debian-security buster/updates/main amd64 Packages [99.2 kB]
Get:5 http://cdn-fastly.deb.debian.org/debian buster/main amd64 Packages [7899 kB]
Get:6 http://cdn-fastly.deb.debian.org/debian buster-updates/main amd64 Packages [5792 B]
Fetched 8214 kB in 2s (3830 kB/s)
Reading package lists... Done
root@f5011a73fde8:/# apt-get install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  net-tools
root@f5011a73fde8:/# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.255.0.0  broadcast 10.0.255.255
        ether 02:42:0a:00:00:02  txqueuelen 0  (Ethernet)
        RX packets 306  bytes 8489690 (8.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 174  bytes 13669 (13.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```