**Command Line Docker**

**Note: Install docker community edition**

[**https://www.docker.com/community-edition**](https://www.docker.com/community-edition)

**https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#uninstall-old-versions**

Uninstall old versions

Older versions of Docker were called docker or docker-engine. If these are installed, uninstall them:

$ sudo apt-get remove docker docker-engine docker.io

If you need to use aufs

### Docker CE now uses the overlay2 storage driver by default, and it is recommended that you use it instead of aufs. If you need to use aufs, you will need to do additional preparation

### Install docker using the convenience script

curl -fsSL get.docker.com -o get-docker.sh

$ sudo sh get-docker.sh

The only note is that since Ubuntu comes with v4 Kernels, there is no need to install any more packages except docker itself. The commands below must be ran as root.

### TL;DR Docker With OverlayFS on Ubuntu 16.04:

wget https://get.docker.com -O - | sh

systemctl stop docker

CONFIGURATION\_FILE=$(systemctl show --property=FragmentPath docker | cut -f2 -d=)

cp $CONFIGURATION\_FILE /etc/systemd/system/docker.service

perl -pi -e 's/^(ExecStart=.+)$/$1 -s overlay/' /etc/systemd/system/docker.service

systemctl daemon-reload

systemctl start docker

### Details

The first step just installs docker, straight from docker.com.

wget https://get.docker.com -O - | sh

Awesome. The problem now is that it runs with the aufs driver, so we need to stop docker.

systemctl stop docker

The next step is to create a copy of the current configuration file from the system itself. We will override the configuration from the system by copying it as /etc/systemd/system/docker.service .

CONFIGURATION\_FILE=$(systemctl show --property=FragmentPath docker | cut -f2 -d=)

cp $CONFIGURATION\_FILE /etc/systemd/system/docker.service

Now, we change the ExecStart line by appending the OverlayFS setting:

perl -pi -e 's/^(ExecStart=.+)$/$1 -s overlay2/' /etc/systemd/system/docker.service

We now tell systemd that the configuration files have updated, and start docker again:

systemctl daemon-reload

systemctl start docker

Done. Enjoy.

#### **# INSTALL DOCKER CE**

1. Update the apt package index.
2. $ sudo apt-get update
3. Install the latest version of Docker CE, or go to the next step to install a specific version. Any existing installation of Docker is replaced.
4. $ sudo apt-get install docker-ce

**Got multiple Docker repositories?**

If you have multiple Docker repositories enabled, installing or updating without specifying a version in the apt-get install or apt-get update command will always install the highest possible version, which may not be appropriate for your stability needs.

1. On production systems, you should install a specific version of Docker CE instead of always using the latest. This output is truncated. List the available versions.
2. $ apt-cache madison docker-ce

docker-ce | 17.09.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

# docker pull centos

# docker images

# docker –help

# To remove Images

# docker rmi centos

# To run a container In daemon mode or detach mode

# docker run --name CENTOS -it -d centos

# To get inside the container

# docker attach CENTOS or id

OR

# docker exec -it CENTOS /bin/bash

# To Start a stopped container

# docker start CENTOS

# To Exit from a container with out stopping it.

# ctrl p+q

# Enter a command in a running container with out entering Note: Quotes are very important

# First run a container

# docker run --name CENTOS -it -d centos

# docker exec -it CENTOS /bin/bash -c "echo 'Hello World' > /QASIM.TXT"

# To remove a container first stop then

# docker rm CENTOS

# Enter in to a container, & when I exit remove the container as well

# docker run --name CENTOS --rm -it centos /bin/bash

# Enter in to a container, & when I exit do not remove the container as well

# docker run --name CENTOS -it centos /bin/bash

# Container Command history check

# docker history centos -------<base image actual name not a LABEL>

# BULK START, STOP, REMOVE

# docker rm $(docker ps -a -q)

# docker start $(docker ps -a -q)

# docker stop $(docker ps -a -q)

# Manual CREATION of a NEW IMAGE 2 STEPS & Restore <import>

# docker commit CENTOS qasim:v.1

# docker save -o /home/asim/Desktop/qasim.tar qasim

Tar to look inside

# tar -tf file.tar

# docker load -i /home/asim/Desktop/qasim.tar

# RUN a command without starting a container

# TRICK always START A CONTAINER# IF fails mean not working 🡪 MAKE IT RUNIING

# docker run --name CENTOS -d centos /bin/bash -c "ping 8.8.8.8 -c 30"

# docker top CENTOS <It shows PID of host> <To see PID of container, go inside and ps aux>

# Detailed information about container

# docker inspect CENTOS

# LAST CONTAINER RUN

# docker ps -l

# docker FORCEFULLY IS POWERFULL, You can remove a running container.

# docker rm -f CENTOS

# DOCKER INFORMATION lot of information.

# docker info

Containers: 0

Running: 0

Paused: 0

Stopped: 0

Images: 1

Server Version: 17.10.0-ce

Storage Driver: aufs

Root Dir: /var/lib/docker/aufs

Backing Filesystem: extfs

Dirs: 1

# docker logs of running container.

# docker logs CENTOS

#Best way to GET inside a docker machine, do not use ssh—>not recommended

#attach is not very best

# nsenter 🡪 Allow us to enter name spaces. 🡪 we need PID of container.

# docker inspect CENTOS | grep -i PID

# nsenter -m -u -n –i -p -t 12658 /bin/bash

<mount namespace> <uts namespace><network namespace><process namespace><ipc namespace> <target>

#RECOMMENDED WAY TO ENTER INTO A CONTAINER

# docker exec -it CENTOS /bin/bash

**DOCKER FILE**

**Is**

**Dockerfile**

# Centos Container

FROM centos:latest

MAINTAINER Muhammad Asim quickbooks@gmail.com +92 335 984 6593

RUN yum update -y && yum upgrade -y && yum install -y vim && yum install -y net-tools

CMD ["echo","Hello World"]

# docker build -t centos:v.1 .

#RUN for installation

* RUN executes command(s) in a new layer and creates a new image. E.g., it is often used for installing software packages.
* #CMD for service start stop, CMD sets default command and/or parameters, which can be overwritten from command line when docker container runs.
* #ENTRYPOINT ENTRYPOINT instruction allows you to configure a container that will run as an executable. It looks similar to CMD, because it also allows you to specify a command with parameters. The difference is ENTRYPOINT command and parameters are not ignored when Docker container runs with command line parameters. , which cannot be overwritten from command line when docker container runs.

(There is a way to ignore ENTTRYPOINT, but it is unlikely that you will do it.)

# Now simply spun a container 🡪 You will get the output

docker run centos:v.1

Hello World

#The command to show different layers, and their size, is

# docker history image:v.1

DDD

**DOCKER ONLINE REPOSITORIES**

[**https://hub.docker.com**](https://hub.docker.com)

[**https://hub.docker.com/r/chasim1982/ubuntu-16.04-asim-apache2/**](https://hub.docker.com/r/chasim1982/ubuntu-16.04-asim-apache2/)

# How push an image to docker public repo’s

Ans: First tag our image

NOTE: The image which is tagged will also be deleted.<In case of deletion>

# docker tag d123f4e55e12 chasim1982/centos:v.1

<imageID>

# . Before pushing, remember to log in from the command line to your docker hub account

docker login

# docker push chasim1982/centos:v.1

# docker pull chasim1982/ubuntu-16.04-asim-apache2:v1

OR Simply pull Image with all tags

# docker pull -a chasim1982/ubuntu-16.04-asim-apache2

#Note Repoinfo Tags 🡺 see tag from there.

**DEEP** **Dockerfile**

# Not a better way to write a Dockerfile

#Simple Web Server

FROM ubuntu:latest

RUN apt-get update

RUN apt-get install -y apache2

RUN apt-get install -y apache2-utils

RUN apt-get install -y vim

RUN apt-get clean

EXPOSE 80

CMD ["apache2ctl","-D","FOREGROUND"]

# Better way to write a Dockerfile <less layers, Total size of the Image is less>

#Simple Web Server

FROM ubuntu:latest

RUN apt-get update && apt-get install -y apache2 && apt-get install -y apache2-utils && apt-get install -y vim && apt-get clean \

&& rm -rf /var/lib/apt/lists/\* /tmp/\* /var/tmp/\*

EXPOSE 80

CMD ["apache2ctl","-D","FOREGROUND"]

#RUN Command 🡺 build time command, when container does not exist.

#RUN Command 🡺 Add layers to images.

#RUN Command 🡺Used to Install Apps to Images.

#CMD Command 🡺 run time command, when container does exist and running or launched.

#CMD Command🡺 Only One CMD is worked in Dockerfile, but if we add multiple CMD, only last one will be applicable.

#CMD 🡺 Can be executed in TWO forms. 🡺 SHELL FORM & 🡺 EXEC FORM.

# SHELL FORM 🡺 Commands are expressed in the same way as in SHELL.

# SHELL FORM 🡺 Commands gets prepended by “/bin/sh -c"

Prepend🡺 add (something) to the beginning of something else

# 🡺 EXEC FORM 🡺 JASON ARRAY STYLE 🡺 [“command1”,”command2”]

# Containers do not need a shell.

#🡺 EXEC FORM 🡺 Avoids string munging by the shell.

##🡺 EXEC FORM 🡺 downside 🡺 no shell features, no variable expansion, no special characters. ( &&, ||, >..)

mung 🡺 manipulate (data).

**ENTRYPOINT**

# ENTRYPOINT CAN NOT BE OVER WRITTEN, when you use docker command line generally,

But there are ways, you can override it. --entrypoint

# We can override CMD instructions at run time.

#ENTRYPOINT is just like positional parameter $1 or an argument like $1

#Example Entrypoint understanding

FROM centos:latest

RUN yum update -y && yum upgrade -y && yum install -y vim && yum install -y net-tools

ENTRYPOINT ["echo"]

# Just applying $1 or positional parameter or argument.

# docker run centos:entrypoint-v1 ASIM HERE FROM PAKISTAN

# Using containers acted like a binary, here containers acted like an echo binary

# Example Apache with entry point 🡺 you run in background

FROM ubuntu:latest

RUN apt-get update -y && apt-get upgrade -y && apt-get install -y vim && apt-get install -y net-tools apache2

ENTRYPOINT ["apache2"]

# docker build -t ubuntu:entrypoint-v2 .

# docker run ubuntu:entrypoint-v2 -d -p 80:80 ubuntu:entrypoint-v2 -D FOREGROUND

**NOTE: With ENTRYPOINT in Dockerfile, if you enter CMD 🡺 that would be consider as $1, positional parameters or arguments.**

**ENV INSTRUCTIONS**

ENV VARIABLENAME=value

# Note: These variables will be available, inside the running containers as normal environment variables.

# Note: Put ENV in same line, put them in different lines, you will get two images layers.

#ENV understanding

FROM centos:latest

RUN yum update -y && yum install -y net-tools

ENV FIRST=Muhammad LAST=Asim

docker run --name CENTOS-ENV -it centos:env-v1 /bin/bash

env

echo $FIRST $LAST

**# 2ND Example**

**#ENVunderstanding**

**FROM centos:latest**

**RUN yum update –y**

**VOLUME /DATA 🡺 We not perform host mount from Dockerfile**

**ENV PING=ping IP=8.8.8.8**

**CMD $PING $IP -c3**

**# docker run --name CENTOS-ENV -it -d centos:env-v1**

**# docker logs centos:env-v1**

**OR**

**# Follow the logs, just like the tail command**

**# docker logs -f centos:env-v1**

Docker images do not save running processes. Therefore, your RUN command executes only during docker build phase and stops after the build is completed. Instead, you need to specify the command when the container is started using the CMD or ENTRYPOINT commands like below:

CMD mysql start

Secondly, the docker container needs a process (last command) to keep running, otherwise the container will exit/stop. Therefore, the normal service mysql start command cannot be used directly in the Dockerfile.

## **Solution**

There are three typical ways to keep the process running:

* Using service command and append non-end command after that like tail -F
* CMD service mysql start && tail -F /var/log/mysql/error.log

This is often preferred when you have a single service running as it makes the outputted log accessible to docker.

* Or use foreground command to do this
* CMD /usr/bin/mysqld\_safe

This works only if there is a script like mysqld\_safe.

* Or wrap your scripts into start.sh and put this in end
* CMD /start.sh

This is best if the command must perform a series of steps, again, /start.sh should stay running.

In your Dockerfile, add at the last line

ENTRYPOINT service ssh restart && bash

**FROM ubuntu:latest**

**MAINTAINER Muhammad Asim quickbooks@gmail.com +92 335 984 6593**

**RUN apt-get update -y && apt-get upgrade -y && apt-get install -y vim && apt-get install -y net-tools && apt-get install -y apache2**

**CMD service apache2 restart && bash**

**OR**

**ENTRYPOINT service apache2 restart && bash**

**# CHECK PORT curl** [**http://localhost:5000**](http://localhost:5000)

**# keep in mind 2 IP’s**

**# docker port APACHE2**

**80/tcp -> 0.0.0.0:5000**

**192.168.1.8:5000 172.17.0.2:80**

**OR**

**FROM ubuntu:latest**

**MAINTAINER Muhammad Asim quickbooks@gmail.com +92 335 984 6593**

**RUN apt-get update -y && apt-get upgrade -y && apt-get install -y vim && apt-get install -y net-tools && apt-get install -y apache2**

**ENTRYPOINT ["apache2ctl"]**

**CMD ["-D","FOREGROUND"]**

***DOCKER VOLUMES***

# docker volume --help

Usage: docker volume COMMAND

Manage volumes

Options:

--help Print usage

Commands:

create Create a volume

inspect Display detailed information on one or more volumes

ls List volumes

prune Remove all unused volumes

rm Remove one or more volumes

# docker run --help

#STEP 1 🡺SIMPLE Mount which is not bind from host

docker run --name CENTOS -v /VOLUMES -it centos /bin/bash

ls

VOLUMES

# cd VOLUMES;echo “Hello World” > qasim.txt

docker inspect CENTOS | grep -i volume

"VolumeDriver": "",

"VolumesFrom": null,

"Type": "volume",

"Source": "/var/lib/docker/volumes/ed11c427b71f7af0659e991f0ddc9f6b97042afad06f112f5088fb742737a82d/\_data",

"Destination": "/VOLUMES",

"Volumes": {

"/VOLUMES": {}

# cd /var/lib/docker/volumes

# cat ed11c427b71f7af0659e991f0ddc9f6b97042afad06f112f5088fb742737a82d/\_data/QASIM.txt

Hello World Qasim Here

#NOTE: OTHER Containers can also share this data

docker run --name UBUNTU --volumes-from=CENTOS -it ubuntu:latest /bin/bash

ls

VOLUMES

cat VOLUMES/QASIM.txt

Hello World Qasim Here

Note: We can access this VOLUME, if the other container is stopped or deleted.

STEP 2 🡺SIMPLE Mount which is bind from host

docker run --name UBUNTU -v /home/asim/Desktop/VOLUMES:/VOLUMES -it ubuntu:latest /bin/bash

# mkdir /home/asim/Desktop/VOLUMES;cd VOLUMES;touch volumes\_test.txt

# To DELETE A VOLUME

# docker rm -v CONTAINERNAME

**# CREATION OF LOCAL VOLUME WITHOUT ABSOULTE PATH**

docker run --name UBUNTU -v TAHA:/VOLUMES -it ubuntu:latest /bin/bash

docker volume ls

DRIVER VOLUME NAME

local 1036233c6283d770c171c219d3fc1a4c06fa753d4dea5208ce932b18953fb86f

local TAHA

local ed11c427b71f7af0659e991f0ddc9f6b97042afad06f112f5088fb742737a82d

docker volume inspect TAHA

[

{

"CreatedAt": "2017-11-10T11:14:26-08:00",

"Driver": "local",

"Labels": null,

"Mountpoint": "/var/lib/docker/volumes/TAHA/\_data",

"Name": "TAHA",

"Options": {},

"Scope": "local"

}

]

**NOTE: If you delete a container, it will not delete a volume.**

**NOTE: You can also create multiple volume containers**

**docker run --name UBUNTU -v TAHA:/VOLUMES -v ZOHA:/DAUGHTER -it ubuntu:latest /bin/bash**

**root@b568aece1bb8:/# ls**

**DAUGHTER VOLUMES bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var**

**root@b568aece1bb8:/#**

**docker volume ls**

**DRIVER VOLUME NAME**

**local 1036233c6283d770c171c219d3fc1a4c06fa753d4dea5208ce932b18953fb86f**

**local TAHA**

**local ZOHA**

**DATA ONLY CONTAINER**

**# docker pull busybox**

#BUSYBOX FOR DATA ONLY CONTAINERS

docker create -v /home/asim/Desktop/WEB:/var/www/html --name BUSYBOX busybox

#WHMCS SETUP

docker run --name WHMCS --network="ASIM-86-A" --link MYSQL:db -p 8080:80 --volumes-from BUSYBOX -d wordpress

**# docker run --name DATA\_ONLY\_CONTAINER -v /home/asim/Desktop/VOLUMES:/VOLUMES busybox**

**# docker run --name CENTOS --volumes-from DATA\_ONLY\_CONTAINER -it -d centos**

**OR**

**docker run --name DATA\_ONLY\_CONTAINER -v /home/asim/Desktop/VOLUMES:/var/www busybox**

**docker run --name CENTOS --volumes-from DATA\_ONLY\_CONTAINER -it -d centos**

**NOTE: For To Make portable DATA CONTAINERS**

**# Copy all data in Busy Container make separate container and make DATA\_ONLY\_CONTAINER**

**# docker cp –help**

**# NOTE: To copy container must be running in a daemon mode.**

# docker run --name DATA\_ONLY\_CONTAINER -it -d busybox

# docker cp /home/asim/Desktop/VOLUMES/ DATA\_ONLY\_CONTAINER:/var/www/

# docker attach DATA\_ONLY\_CONTAINER

# ls

ls var/

spool/ www/

/ # ls var/www/VOLUMES/

power.txt volume\_test.txt

# docker commit DATA\_ONLY\_CONTAINER web\_data:latest

# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

web\_data latest 0e75b36cc8d9 54 seconds ago 1.13MB

**CREATE AN EMPTY VOLUME**

# docker volume create --name DATA\_ONLY\_CONTAINER

**MOUNT EMPTY VOLUME WITH DESIRED DIRECTORY**

# docker run --name WEB\_DATA -v DATA\_ONLY\_CONTAINER:/var -it -d web\_data

# docker stop WEB\_DATA

# docker run --name CENTOS --volumes-from WEB\_DATA -it centos

# ls var/www/VOLUMES/

# EVEN

docker rm WEB\_DATA

<Because data is now mounted in DATA\_ONLY\_CONTAINER 🡺 locally

docker volume inspect DATA\_ONLY\_CONTAINER

[

{

"CreatedAt": "2017-11-11T02:32:25-08:00",

"Driver": "local",

"Labels": {},

"Mountpoint": "/var/lib/docker/volumes/DATA\_ONLY\_CONTAINER/\_data",

"Name": "DATA\_ONLY\_CONTAINER",

"Options": {},

"Scope": "local"

}

]

cat /var/lib/docker/volumes/DATA\_ONLY\_CONTAINER/\_data/www/VOLUMES/volume\_test.txt

Hello World

Asim here

**IF YOU CREATE A DIRECTORY ON LOCAL MACHINE & DO MOUNT THIS WAY IT WILL NOT WORK.**

**BUT**

**YOU CAN COPY DATA FROM A CONTAINER LOCALLY AND THEN MOUNT SIMPLE**

**DONE**

**NETWORKING**

# docker port APACHE2

# On Main host must install bridge-utils

# docker inspect container

# docker containers meta data lives in /var/lib/docker/containers/

# docker run --dns=8.8.4.4 --name DNS\_TEST -it Ubuntu

# NOTE: By default no container have internet access

🡺 You need to run apt-update or yum update

# apt-get install iputils-ping

**#Note: If image is build with apt-get update then simply install ping package.**

Note: By default It is assumed that port mapping is TCP, but we can define port

UDP as well.

# docker run --name APACHE2\_V2 -p 6000:80/udp -dit apache2\_asim:v2

# ip -f inet a 🡺 To see ONLY IPV4 address.

# 🡺 -P expose all ports of container

***LINKING CONTAINERS***

# Note: docker use container-name for Linking containers.

# docker run --name=SOURCE -itd -p 80 ubuntu

# docker run --name=CLIENT -itd --link=SOURCE:ASIM\_SOURCE ubuntu /bin/bash

<source>:<alias>

# docker inspect CLIENT

"/SOURCE:/CLIENT/ASIM\_SOURCE"

# docker exec -it CLIENT bash

# root@5058234a7502:/# env | grep -i asim

ASIM\_SOURCE\_PORT=tcp://172.17.0.2:80

ASIM\_SOURCE\_PORT\_80\_TCP\_PORT=80

ASIM\_SOURCE\_PORT\_80\_TCP\_PROTO=tcp

ASIM\_SOURCE\_PORT\_80\_TCP\_ADDR=172.17.0.2

ASIM\_SOURCE\_NAME=/CLIENT/ASIM\_SOURCE

ASIM\_SOURCE\_PORT\_80\_TCP=tcp://172.17.0.2:80

# cat /etc/hosts

127.0.0.1 localhost

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

172.17.0.2 ASIM\_SOURCE 5f0bad3bedf1 SOURCE

172.17.0.3 5058234a7502

# LINKING CONTAINERS is more secure than exposing ports, only Client containers knows about

🡺 the Source container network configurations.

🡺 We can link multiple containers to a single SOURCE, also we can link multiple SOURCE to single CLIENT.

**DOCKER LOCAL AERA NETWORK SCRIPT**

#!/bin/bash

echo

NETWORK=`ip addr show | grep -iE "dynamic|static" | awk '{print $2}' | head -c 9`

NETWORKNAME=` docker network ls | grep -i ASIM | awk '{print $2}'`

#echo $NETWORK

if [ "$NETWORK" = "192.168.1" ]

then

echo -e "Your Network is $NETWORK"

echo

DOCKERNETWORK=`docker network create -d macvlan --subnet=192.168.1.0/24 --ip-range=192.168.1.255/26 --gateway 192.168.1.1 -o macvlan\_mode=bridge -o parent=ens33 ASIM-86-A`

echo $DOCKERNETWORK

echo $? > /dev/null

if [ "$?" = "0" ]

then

echo -e "\nYour docker local area network $NETWORKNAME is created\n"

fi

fi

.**DOCKER LOCAL AREA CONNECTIONS**

docker network create -d macvlan --subnet=192.168.1.0/24 --ip-range=192.168.1.255/24 --gateway 192.168.1.1 -o macvlan\_mode=bridge -o parent=ens33 MAC-V-LAN

467c103a782ca4ca562abd8b78b93de469cf1fb98f4e686096d4e2f1fb030ce3

root@ubuntu:/home/asim/Documents/Apache# docker network ls

NETWORK ID NAME DRIVER SCOPE

467c103a782c MAC-V-LAN macvlan local

5cc7d07e4146 bridge bridge local

67c39b007cea host host local

6f56f933779d none null local

<Note: We first create a Network, than we attach it, with New Cotainers>

2.To Avoid Overlapping

docker network create -d macvlan --subnet=192.168.1.0/24 --ip-range=192.168.1.255/26 --gateway 192.168.1.1 -o macvlan\_mode=bridge -o parent=ens33 MAC-V-LAN

**LEARNING SCRIPT FOR NETWORK AND LINKING**

#!/bin/bash

#MYSQL SETUP

docker run --name MYSQL -e MYSQL\_ROOT\_PASSWORD=abc -v /home/asim/Desktop/local:/var/lib/mysql --network="ASIM-86-A" -d mysql

#PHPMYADMIN SETUP

docker run --name PHPMYADMIN --network="ASIM-86-A" --link MYSQL:db -td phpmyadmin/phpmyadmin

#BUSYBOX FOR DATA ONLY CONTAINERS

docker create -v /home/asim/Desktop/web:/var/www/html --name BUSYBOX busybox

#WORDPRESS SETUP

docker run --name WORDPRESS --network="ASIM-86-A" --link MYSQL:db -p 80:80 --volumes-from BUSYBOX -itd wordpress

echo

echo -e "\nFollowing is the IP address of WORDPRESS Container\n"

docker inspect WORDPRESS | grep IPAddress | tail -n1

#END

***DOCKER TROUBLE SHOOTING***

# Docker Daemon Logging

https://docs.docker.com/engine/admin/#enable-debugging

There are two ways to enable debugging. The recommended approach is to set the debug key to true in the daemon.json file. This method works for every Docker platform.

1. Edit the daemon.json file, which is usually located in /etc/docker/. You may need to create this file, if it does not yet exist. On macOS or Windows, do not edit the file directly. Instead, go to **Preferences** / **Daemon** / **Advanced**.
2. If the file is empty, add the following:
3. {
4. "debug": true
5. }

If the file already contains JSON, just add the key "debug": true, being careful to add a comma to the end of the line if it is not the last line before the closing bracket. Also verify that if the log-level key is set, it is set to either info or debug. info is the default, and possible values are debug, info, warn, error, fatal

Send a HUP signal to the daemon to cause it to reload its configuration. On Linux hosts, use the following command.

$ sudo kill -SIGHUP $(pidof dockerd

* By running journalctl -u docker.service on Linux systems using systemctl
* /var/log/messages, /var/log/daemon.log, or /var/log/docker.log on older Linux systems

journalctl | grep -i docker

# docker info | grep –I debug 🡺 this has all docker logs

# Container Logging

🡺 docker logs –f container

🡺 Dokcer Image build from Dockerfile 🡺 first all do stuff in docker container first 🡺 run all commands in container, test if all is well, then make Dockerfile from that.

# Troubleshooting Images

🡺 Make sure create Dockerfile with testing container in seprate TAB.

# Network Troubleshooting

🡺 ip link del docker0

🡺 1.DOCKER LOCAL AREA CONNECTIONS

docker network create -d macvlan --subnet=192.168.1.0/24 --ip-range=192.168.1.255/24 --gateway 192.168.1.1 -o macvlan\_mode=bridge -o parent=ens33 MAC-V-LAN

467c103a782ca4ca562abd8b78b93de469cf1fb98f4e686096d4e2f1fb030ce3

root@ubuntu:/home/asim/Documents/Apache# docker network ls

NETWORK ID NAME DRIVER SCOPE

467c103a782c MAC-V-LAN macvlan local

5cc7d07e4146 bridge bridge local

67c39b007cea host host local

6f56f933779d none null local

<Note: We first create a Network, than we attach it, with New Cotainers>

2.To Avoid Overlapping

docker network create -d macvlan --subnet=192.168.1.0/24 --ip-range=192.168.1.255/26 --gateway 192.168.1.1 -o macvlan\_mode=bridge -o parent=ens33 MAC-V-LAN