Contents

[1. Software Requirement 3](#_30j0zll)

[2. Hardware Requirement 3](#_1fob9te)

[3. Conventions 3](#_3znysh7)

[4. Install docker compose 3](#_2et92p0)

[5. Setup SSH 3](#_tyjcwt)

[6. Enable Networking in Ubuntu 3](#_3dy6vkm)

[7. Commands for installing Docker on Ubuntu 14.0.4 LTS 4](#_1t3h5sf)

[8. Commands for installing Docker on CentOS 6.x 4](#_4d34og8)

[9. Configure Docker to run by non root user 4](#_2s8eyo1)

[10. Ubuntu: configuring docker to communicate over network 5](#_17dp8vu)

[11. CentOS: configuring docker to communicate over network 5](#_3rdcrjn)

[12. Reset to use Docker on local system 5](#_26in1rg)

[13. Docker Example – Run CentOS Container 5](#_lnxbz9)

[14. From Docker Host Access files created in Docker Container 6](#_35nkun2)

[15. Connect to existing container 6](#_1ksv4uv)

[16. Commands for Docker Images 6](#_44sinio)

[17. Docker Containers Commands 6](#_2jxsxqh)

[18. Docker registries and Repo 7](#_z337ya)

[19. A closer look to images & containers 7](#_3j2qqm3)

[20. Move docker container from one docker host to another 7](#_1y810tw)

[21. Focusing on containers 8](#_4i7ojhp)

[22. Container Management 8](#_2xcytpi)

[23. Look Inside Containers 9](#_1ci93xb)

[24. Getting a shell in a container. Helps easy entry to Docker container 9](#_3whwml4)

[25. Building from docker file 9](#_2bn6wsx)

[26. Inspecting a docker file from docker hub 10](#_qsh70q)

[27. Docker Registries 10](#_3as4poj)

[28. Docker private registries. 10](#_1pxezwc)

[29. DIVING Deeper into docker file 11](#_49x2ik5)

[30. Dockerfile and image layers 11](#_2p2csry)

[31. Build A Web Server 12](#_147n2zr)

[32. Reducing the number of layers 12](#_3o7alnk)

[33. Docker Networking 13](#_23ckvvd)

[34. Linking containers 16](#_ihv636)

## Software Requirement

* Oracle VM Virual Box
* Ubuntu 14.0.4 – Take VDI from https://osboxes.org
* Docker
* SSH enable on Ubuntu
* Docker Compose
* MobaXterm or Putty

## Hardware Requirement

Any CPU with

* 8 GB+ RAM
* Quod Code+ Processor
* 200 GB Space

## Conventions

* Use SSH to connect to Ububtu Server
* Create a directory called – DockerPractice in Home Directory

## Install docker compose

curl -L "https://github.com/docker/compose/releases/download/1.23.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

docker-compose --version

## Setup SSH

**Reference:**

http://ubuntuhandbook.org/index.php/2014/04/enable-ssh-ubuntu-14-04-trusty-tahr/

**Instructions:**

sudo apt-get install openssh-server

sudo service ssh start

sudo /etc/init.d/ssh restart

## Enable Networking in Ubuntu

sudo ip link set dev eth0 down

sudo dhclient eth0

These commands configure your interface but these changes will not survive a reboot, since the information is not stored anyhwere. This is where the interfaces file comes in handy. To configure a interface permanently you'll need to edit the interfaces file, /etc/network/interfaces.

sudo vi /etc/network/interfaces

#To configure a dynamic IP address

auto eth0

iface eth0 inet dhcp

## Commands for installing Docker on Ubuntu 14.0.4 LTS

# https://docs.docker.com/install/linux/docker-ce/ubuntu/

# Uninstall older versions

sudo apt-get remove docker docker-engine docker.io containerd runc

sudo apt-get update

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

docker run hello-world

## Commands for installing Docker on CentOS 6.x

yum install -y docker

systemctl status docker.service

systemctl start docker.service

docker -v

docker version

docker info

## Configure Docker to run by non root user

sudo su

cat /etc/group

sudo gpasswd -a <username> docker

logout

// Now Login using normal user which is added to Docker group

docker run -it ubuntu /bin/bash

**Multiple Ports Expose:** docker run -it -p 3000-4000:3000-4000 -p 80:80 -p 8080:8080 -p 23:23 -p 21:21 -p 81:81 -p 8888:8888 atin\_resume\_builder\_2 /bin/bash

#exit

## Ubuntu: configuring docker to communicate over network

netstat -tlp

service docker stop

docker -H 192.168.56.50:2375 -d &

netstat -tlp

docker info

## CentOS: configuring docker to communicate over network

export DOCKER\_HOST="tcp://192.168.56.50|2375"

docker version

export DOCKER\_HOST=

## Reset to use Docker on local system

nohup docker daemon -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock &

## Docker Example – Run CentOS Container

docker run -it centos /bin/bash

# ping 8.8.8.8

# ps -elf

# cat /etc/hosts

# yum install -y net-tools

# yum install -y iproute

# ip a

# uname -a

# cat /etc/redhat-release

# apt-get update //it will not work

# yum check-update

# yum install -y vim

# vim /tmp/tempfile.txt

# exit

docker ps -a

## From Docker Host Access files created in Docker Container

sudo su

ls -l /var/lib/docker/overlay2/diff

ls -l /var/lib/docker/overlay2/<containerid>/tmp

cat /var/lib/docker/overlay2/<containerid>/tmp/tempfile.txt

exit

## Connect to existing container

docker start <container id>

docker attach <containerid>

# cat /tmp/testfile.txt

# exit

## Commands for Docker Images

docker run -it fedora /bin/bash

#exit

docker pull -a fedora // will pull all images from fedora library

docker images fedora

//images are stored locally under - /var/lib/docker/overlay2

sudo ls -al /var/lib/docker/overlay2

## Docker Containers Commands

docker ps

docker attach <containerid>

// Exit container without killing it by <CTRL + P + Q>

docker ps

docker ps -a

## Docker registries and Repo

//Visit hub.docker.com

//It has 2 kind of repositories, Official and User Repo

## Move docker container from one docker host to another

docker images

docker run ubuntu /bin/bash -c "echo 'cool content' > /tmp/cool-file"

docker ps -a

docker commit <container id> image-name:v

docker images

docker history fridge

docker save -o /tmp/fridge.tar fridge

ls -lh /tmp/fridge.tar

//Copy the above tar file to another Docker Host

//On another Docker host run below commands

tar -tf /tmp/fridge.tar

docker load -i /tmp/fridge.tar

docker images

docker run -it fridge /bin/bash

# cat /tmp/cool-file

## Focusing on containers

docker run -d ubuntu /bin/bash -c "apt-get update && apt-get install -y iputils-ping && ping 8.8.8.8 -c 3000"

docker ps

docker top <container id>

docker ps

docker ps -a

docker run -d ubuntu /bin/bash -c "apt-get update && apt-get install -y iputils-ping && ping 8.8.8.8 -c 3000"

docker ps

docker inspect <container id> or <image id>

docker attach <container id/name>

//Press CTRL+C

## Container Management

docker run -it ubuntu:14.04 /bin/bash

//Hit <CTRL + P + Q>

docker ps

docker stop <container id>

docker ps

docker ps -l // Last container ran

docker start <container id>

docker attach <container id>

docker restart <container id>

docker info

docker rm <containerid> //Don't allow to remove running container

docker stop <containerid>

docker rm <containerid>

docker info

docker rm -f <containerid> //Use -f to remove running container

alias dps = "docker ps" // Creating alias of “docker ps”

## Look Inside Containers

docker ps

docker top <containerid>

docker attach <containerid>

#ps -ef

<CTRL + P + Q>

docker logs <containerid>

docker inspect <containerid>

ls -l /var/lib/docker/containers/<containerid>/

cat /var/lib/docker/containers/<containerid>/config.v2.json

## Getting a shell in a container. Helps easy entry to Docker container

docker-enter <containerid>

npm i -g docker-enter # if required

docker exec -it <containerid> /bin/bash

## Building from docker file

// Move out to base directory

mkdir df1

cd df1

vim Dockerfile

FROM ubuntu:14.04

MAINTAINER atingupta2005@gmail.com

RUN apt-get update

RUN apt-get install -y nginx

RUN apt-get install -y golang

CMD ["echo", "HELLO WORLD"]

ls -l

docker build -t helloworld:0.1 .

docker images

docker history <imageid>

docker run helloworld:0.1

## Inspecting a docker file from docker hub

//Visit Docker Huib and inspect docker file from ubuntu image

## Docker Registries

//Create account on dockerhub

//Create repository on dockerhub

docker images

docker tag <imageid> <dockerhub username>/<repository name>:<any tag>

docker login

docker push <dockerhub username>/<repository name>:<any tag>

now on another computer

docker rm <cid> <cid> <cid> <cid>

docker rmi <imgid> <imgid> <imgid> <imgid> <imgid>

docker images

docker pull <repo name>

docker images

## Docker private registries.

docker run -d -p 5000:5000 --restart always --name registry registry:2

docker images

docker ps

// now visit at browser -> http://<server host name>:5000

docker images

//Push image to private registry

docker tag <image id> localhost:5000/<imagename>

docker push localhost:5000/<imagename>

## Diving Deeper into docker file

//Build Cache

docker info

docker images

cd df1

docker build -t="build1" .

//below command will take lesser time due to reuse of build cache

docker build -t="build2" .

## Dockerfile and image layers

docker images

docker info

// Move out to base directory

mkdir df2

cd df2

vim Dockerfile

FROM ubuntu:14.04

MAINTAINER atingupta2005@gmail.com

RUN apt-get update

RUN apt-get install -y nginx

RUN apt-get install -y golang

CMD ["echo", "HELLO WORLD"]

docker build -t="build3" .

docker info // 6 images are increased. Each for single Dockerimage instruction

docker history <img id>

## Build A Web Server

// Move out to base directory

mkdir web

cd web

vim Dockerfile

FROM ubuntu:14.04

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 -y

EXPOSE 80

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

docker build -t="webserver" .

docker images

docker run -d -p 80:80 webserver // Check on browser now

docker ps

docker stop <container name> // Check on browser now

## Reducing the number of layers

docker history webserver

vim Dockerfile

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

apache2 \

apache2-utils \

vim && \

apt-get clean -y\

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

EXPOSE 80

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

docker build -t="webserver-small" .

docker history webserver-small

## Docker Networking

ip a // Notice docker0 port

apt-get install bridge-utils

brctl show docker0

docker images

// Move out to base directory

mkdir df4

cd df4

vim Dockerfile

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

iputils-ping\

traceroute

ENTRYPOINT ["/bin/bash"]

docker build -t="net-img" .

docker run -it --name=net1 net-img

<CTRL+P+Q>

brctl show

docker run -it --name=net2 net-img

<CTRL+P+Q>

brctl show

docker attach net1

#ip a

#ping 8.8.8.8

#traceroute 8.8.8.8

<CTRL+P+Q>

docker run -it --name=net3 net-img

<CTRL+P+Q>

docker inspect net2 // Inspect network settings

ip a

docker ps

ls -l /var/lib/docker/containers/<containerid> // host and resolve.conf are copied from docker host

//Use below command to overwrite default settings

docker run --dns=8.8.4.4 --name=dnstest net-img

docker inspect dnstest

// Move out to base directory

mkdir df5

cd df5

# Exposing Ports

vim Dockerfile

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

iputils-ping \

traceroute \

apache2

EXPOSE 80

ENTRYPOINT ["apache2ctl"]

CMD ["-D","FOREGROUND"]

docker build -t="apache-img" .

docker run -d -p 5001:80 --name=web1 apache-img

docker ps

//Check in web browser

docker run -d -p <hostip>:5003:80 --name=web3 apache-img

docker port web3

// Move out to base directory

mkdir df6

cd df6

vim Dockerfile

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

iputils-ping\

traceroute\

apache2

EXPOSE 80 500 600 700 800 900

ENTRYPOINT ["apache2ctl"]

CMD ["-D","FOREGROUND"]

docker build -t="throw-away" .

docker run -d -P --name=throw throw-away

docker port throw

## Linking containers

docker images

cat Dockerfile // Web image

docker run --name=src -d apache-img

docker ps

docker run --name=rcvr --link=src:ali-src -it ubuntu:14.04 /bin/bash

<CTRL+P+Q>

docker ps

docker inspect rcvr

docker inspect src | grep Links

docker attach rcvr

#env

#env | grep ALI

#cat /etc/hosts