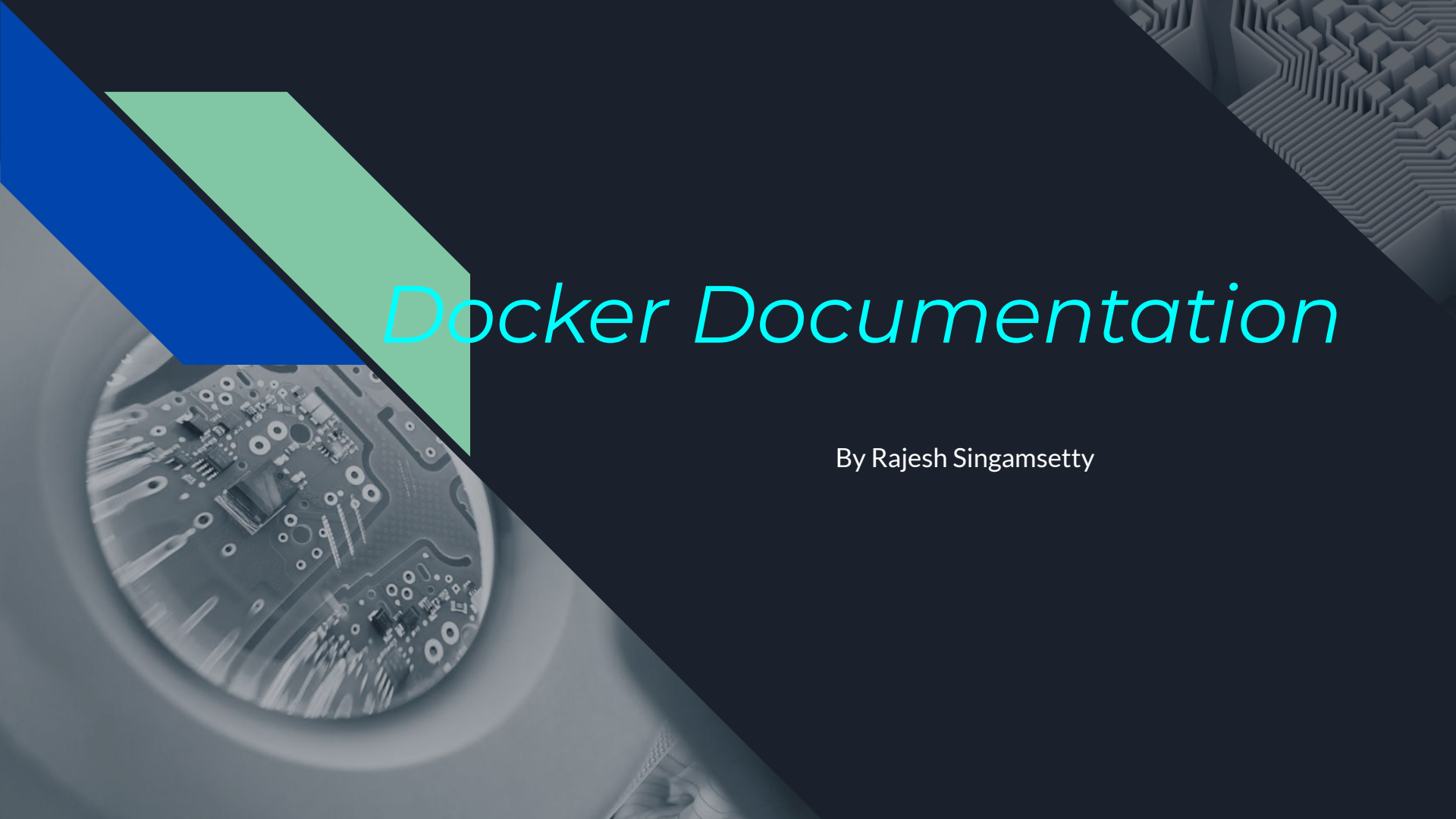

Docker Documentation

By Rajesh Singamsetty

Topics

Getting Requirements.

Docker Install

Container Basics

Image basics

Docker networking

Docker volumes

Docker compose

Orchestartion

Docker swarm

Kubernates

Swarm vs K8’s

Author By Rajesh Singamseety

Installation Of Docker

Docker Types:
Community Edition (CE)
Enterprise Edition(EE)

We are using community edition.

Installations:Rhel:

Reference link:
Go to console(putty or aws)
sudo su
cd

```
dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
dnf list docker-ce
  Wait until the result become:
```

Available Packages

```
docker-ce.x86_64              3:19.03.5-3.el7                 docker-ce-stable
```

```
Now we need to install latest docker
```

continue(Docker Compose)

```
dnf install docker-ce --nobest -y
```

Wait for status will be
Skipped:
docker-ce-3:19.03.11-3.el7.x86_64
Complete!

```
systemctl start docker
systemctl enable docker
docker --version
```

Test Docker:

```
docker run hello-world
```

If u see hello from docker.
Means its sucess

Install Docker Compose

```
sudo dnf install curl -y    (note below 1.26.0 is the now latest version u r time u can choose latest version)

sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)"
-o /usr/local/bin/docker-compose
```

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

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

```
sudo docker-compose --version
```

Docker Install On Ubuntu

```
sudo apt-get update
 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 apt-key fingerprint 0EBFCD88
 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
sudo apt-get update
 sudo apt-get install docker-ce docker-ce-cli containerd.io
 apt-cache madison docker-ce
sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-cli=5:18.09.6~3-0~ubuntu-bionic containerd.io
sudo apt-get install docker-ce=5:18.09.6~3-0~ubuntu-bionic docker-ce-cli=5:18.09.6~3-0~ubuntu-bionic containerd.io
```

Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose

 sudo chmod +x /usr/local/bin/docker-compose
 sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
 docker-compose --version
```

Nginx WebServer

From rhel install
sudo yum install nginx
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl status nginx

Reference link:https://www.cyberciti.biz/faq/how-to-install-and-use-nginx-on-centos-7-rhel-7/

From docker install nginx container follow below procedure

systemctl start docker
systemctl enable docker
docker container run --publish 80:80 nginx
Go to brower run ip u can see now welcome to nginx
Docker to run in background use below commnad
docker container run --publish 80:80 --detach nginx
To stop nginx serever.
docker container ls(will give the running container list)
docker container stop ffd94850f05c(image id)

Nginx continue

docker container run --publish 80:80 --detach --name webhost nginx

To check logs go to browser refresh 2 times and come to terminal type below command

docker container logs webhost
docker container top webhost
docker container --help

To remove multi contsiners at a time
docker container ls -a
docker container rm
docker container rm e37 ffd c66 3d8 1b6 e8b(first 3 digits of every container).
Note:we can’t remove running container
To remove running container use
rm -f (conteiner first 3 digits)

Topic completed

Container vs vm’s

docker run --name mongo -d mongo (note -d for running background)
docker top mongo
ps aux

U can stop the conatiner and u can observe procees did not exist

Again u will start a container it will show proceessind

Note:container will works on hosting processors

Manage multiple containers

docs.docker.com(reference link)

Quiz task we r doing now

1.docker container run -d -p 3306:3306 --name db -e MYSQL_RANDOM_ROOT_PASSWORD=yes mysql
To see the mysql password: docker container logs db (there u can see the password)
2.docker container run -d --name webserver -p 8080:80
3.docker container run -d --name proxy -p 80:80 nginx
To check nginix : curl localhost
curl localhos:8080
Docket stop ngix (to run container : docker run -d --name myhttpd -p 80:80 httpd)
docker stop c8b2f799dff1 dcae3acd6b48 bb52e7e5ee5d(conatiner ids) : docker ps -a

What going on container

For this we need to start our containers
docker container run -d --name nginx nginx
docker container run -d --name mysql -e MYSQL_RANDOM_ROOT_PASSWORD=true mysql
docker container top nginx
docker container top mysql

Note: we can use top command for how container is started

Inspect :

docker container inspect mysql
By above we can see that all json data format how conatiner is started

docker container stats (it will show how containers are working will give all information like memory etc……..)

Getting shell inside container(no need ssh)

In this proceed we have to follow
Run container
Execute container
Take diffrent linux distibutions containers

Let’s start

Note :docker ps && docker container ls (both r same output to see running containers)
docker container run -it --name proxy nginx bash
-i:interactive
-t: allowcates psudo-ttv
bash:it will give terminal inside the running conatiner

After running above command it will change from ip 2 container : root@ip-172-31-40-169 ~]# docker container run -it --name proxy nginx bash
root@e7edecf60fbe:/# -- (changed from ip to inside container root)
ls
Now here u can see one linux setup like bin var root tmp etc ……(means inside directory we have linux distribution)
exit (used to exit from container)

To be contuined …….

Ubuntu docker container

docker container run -it --name ubuntu ubuntu
apt-get update
apt-get install -y curl

exit (exit from the container)

Now i have again run the ubuntu conatiner now u can use the start command

docker container start -ai ubuntu(name of container)
Now we are in inside the container
Test u r in same container or not type curl google.com if result means u r in correct container
exit

docker exec command

docker container exec -it mysql bash
To use ps aux
U need tu update apt-get update and
apt-get install -y procps

Docker Network command (private&public

docker container run -p (to know port informtion)
docker conatiner port container
Start:
docker container run -p 80:80 --name webhost -d nginx
To check container ports
docker container port webhost
The result will be:80/tcp -> 0.0.0.0:80

To find docker container ip
docker container inspect --format "{{ .NetworkSettings.IPAddress}}" webhost

Network

docker network ls(to show networks)
docker network inspect (inspect network)
We can see how many bridges connected to that network.
docker network create --driver
docker network connect
docker network disconnect
continued

Create new network

docker network create my_app_net

Now we are running a container using a particular network

docker container run -d --name new_nginx --network my_app_net nginx(image name)
docker network inspect my_app_net
In last json data we can new_nginx server.

Connect
docker network connect 758d720e4cc2(networkid) 7f6fad993a93(existing conatiner id)

To check network added or not using

docker container inspect 7f6fad993a93 (docker existing wehost containerid)

Now we can see myappnet network is add
Disconnect:
docker network disconnect 758d720e4cc2(networkid) 7f6fad993a93(existing conatiner id)
docker container inspect 7f6fad993a93 (docker existing wehost containerid)
Now we have only one network only.

Docker Network Dns

Talking conatiners.
Start
docker container ls

Dns default names

docker run -d --name my_nginx --network my_app_net nginx

If conatiners having same network we can ping easily and we work easily

Need to focus more on dns concept.

Docker Images

Docker HUb : rajeshsingam
Create a docker hub account
Now we can find all the images in docker hub account here we are finding the all images what we are using in docker.
Here we can see how we can use that image in docker
ex:docker pull nginx

Image Layers

docker history nginx(image name) (steps included image layers how many times image changed).

Docker Image Inspect

It wil give all information about image in meta data format
Including version it will give

docker image inspect imagename

Image tagging and pusing docker hub

docker image tag --help
Tags are lables that had specific versions to pull or do some operations
If you not specified any tag docker will take automatically as the latest tag to the image
Create image with tag:
`docker image tag nginx  rajeshsingam/nginx`

Docker Image Push

Uploads changed layers to a image registry(default is HUB)

docker image push nginx (image name)
Before push u need to login to the docker
docker login
Give username and password
Now u can see login success message
docker push `rajeshsingam/nginx`
Now u can see my image in my docker hub repository

Docker File Basics

Docker file reference:
https://github.com/BretFisher/udemy-docker-mastery/blob/master/dockerfile-sample-1/Dockerfile
vi dockerfile (paste above url total code and save it)
Docker file is seems to be sheelscripting but its not.
docker build -f some-dockerfile
1.FROM debian:stretch-slim
From package name (container used package management system)
2.ENV NGINX_VERSION 1.13.6-1~stretch
Environmental variables.

3.RUN apt-get update \
To execute commnad run

```
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
        && ln -sf /dev/stderr /var/log/nginx/error.log
# forward request and error logs to docker log collector
EXPOSE 80 443
# expose these ports on the docker virtual network
# you still need to use -p or -P to open/forward these ports on host


CMD ["nginx", "-g", "daemon off;"]
# required: run this command when container is launched
```

Running Docker Buildsß

docker image build -t customnginx .
By using Dockerfile from name we are building one image

Building Images external official images

vi Dockerfile(paste the below code)
FROM nginx:latest
WORKDIR /usr/share/nginx/html
COPY index.html index.html

vi index.html(write some code).
After build image with dockerfile ur custom name
docker image build -t mynginximage .
*now run ur builded custom image(mynginximage).
docker container run -p 80:80 -rm mynginximage
Hit browser with port:80 Now u can see u r html page data.
docker image tag mynginximage:latest rajnew/mynginximage:latest(new image created with tag sameid)
docker push

Build Image using Docker file(node)

vi Dockerfile
FROM node:6-alpine
EXPOSE 3000
RUN apk add --update tini
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package.json package.json
RUN npm install && npm cache clean
COPY . .
CMD [“tini”, “--”, “node”, “./bin/www”]

Save

```
docker build -t testnode .
```

- Need to focus on more

Container Lifetime & persistent data

Containers usally immutable and ephemeral
“Iimutable infrastarcture” only re:deploy containers,never change
Docker gives features to ensure these “seperation pf concerns”
This is known as “persistance data”
Two Ways: volumes and bind mounts.

Data Volumes

docker pull mysql

```
docker container run -d --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD=True mysql
```

Run mysql

docker container inspect mysql(ruuning container location)


```
"Mounts": [
        {
            "Type": "volume",
            "Name": "b86d5de0ac023ac94d5eaa62c5537cfebcac6dd0b83dc4332aa153a7cec71867",
            "Source": "/var/lib/docker/volumes/b86d5de0ac023ac94d5eaa62c5537cfebcac6dd0b83dc4332aa153a7cec71867/_data",
            "Destination": "/var/lib/mysql",
            "Driver": "local",
            "Mode": "",
            "RW": true,
            "Propagation": ""
        }
    ],
```

Docker Volume Continue

docker volume ls

Named Volumes

docker container run -d --name mysql2 -e MYSQL_ALLOW_EMPTY_PASSWORD=True -v mysql-db:/var/lib/mysql mysql

docker volume ls

Now our docker volume is appear on below list

docker volume inspect mysql-db

Now i am going 2 remove my container

docker container rm -f mysql(containertid)

To check : docker conatainer ls

Different container will use the same volume

Docker volume create

docker volume create --help

Persistence Data: Bind Bounding

dsNote u shoyd have nginx image.
And u have to run: `docker container run -d --name nginx2 -p 80:80 -v $(pwd):/usr/share/nginx/html nginx`

After successfu;;y execution go and run container
`docker container exec -it nginx2 bash`

Here u can the the docker present working directory will moving into inside container
`cd /usr//share/nginx/html/`

Here ls -a
Here all the files of container will be shifted into the inside the container path it means the it will shifting from one location to other location volume this know as bind bounding
Now u can create any file in docket it automattocally created in docker inside container also
Beacause the both container are using same volume.

Docker compose and docker compose.yml

To save our docker container run settings in easy read file
Yaml file describes our solutions
1.containers
  Networks
  Volumes
2.cli tool docker-compose used for local dev/test automation with those yaml files.
3.compose yaml format has its own version ex:1,2,2.1,2.2
4.with docker directly in production with swarm (as of v1.13)
docker compose --help
docker-compose.yml is default filename,but any can be used with docker-compose -f
Sample file(referencelink:https://github.com/BretFisher/udemy-docker-mastery/blob/master/compose-sample-1/template.yml)

```
version: '3.1'  # if no version is specified then v1 is assumed. Recommend v2 minimum
services:  # containers. same as docker run
 servicename: # a friendly name. this is also DNS name inside network
  image: # Optional if you use build:
  command: # Optional, replace the default CMD specified by the image
  environment: # Optional, same as -e in docker run
  volumes: # Optional, same as -v in docker run
 servicename2:
volumes: # Optional, same as docker volume create
networks: # Optional, same as docker network create
```

Trying basic compose commands

Two most common commands are
docker-compose up(setup volumes/networks and start all conatainers)
docker-compose down(stop all containers and remove cont/vol/net)
If your project having a Dockerfile and docker-compose.yml then new developer onloading would be)
git clone github.com/some/software
docker-compose up

git clone https://github.com/BretFisher/udemy-docker-mastery.git

cd udemy-docker-mastery
cd compose-sample-2
docker-compose up
docker-compose up -d (to run in background)

Hit ur browser it will show it works.
docker-compose --help
All commands what we are running
docker-compose top

Basic compose file for drupal content management system

Use drupal image with postgres
use ports :8080
docker-compose.yml(below code is there)

```
version: '2'
services:
 drupal:
  image: drupal:8.8.2
  ports:
   - "8080:80"
  volumes:
   - drupal-modules:/var/www/html/modules
   - drupal-profiles:/var/www/html/profiles
   - drupal-sites:/var/www/html/sites
   - drupal-themes:/var/www/html/themes
 postgres:
  image: postgres:12.1
  environment:
   - POSTGRES_PASSWORD=mypasswd
volumes:
 drupal-modules:
 drupal-profiles:
 drupal-sites:
 drupal-themes:
```

Running A website using dockerfile & yml

Cd compose-sample-3

```
docker-compose  up
```