Docker

Table of Contents

[Tools 4](#_Toc27834888)

[Docker 4](#_Toc27834889)

[cmder 4](#_Toc27834890)

[Command Line 4](#_Toc27834891)

[Hello World 4](#_Toc27834892)

[Dockerfile 4](#_Toc27834893)

[Containers 4](#_Toc27834894)

[Sample Console App 4](#_Toc27834895)

[dockerfile 4](#_Toc27834896)

[Build 4](#_Toc27834897)

[Run 4](#_Toc27834898)

[Sample Web App with NGINX 5](#_Toc27834899)

[index.html 5](#_Toc27834900)

[nginx.conf 5](#_Toc27834901)

[dockerfile 5](#_Toc27834902)

[Build 5](#_Toc27834903)

[Run 5](#_Toc27834904)

[Test 5](#_Toc27834905)

[Access Container Shell 5](#_Toc27834906)

[PHP 6](#_Toc27834907)

[sample.php 6](#_Toc27834908)

[Build 6](#_Toc27834909)

[Run 6](#_Toc27834910)

[Text file 6](#_Toc27834911)

[dockerfile 6](#_Toc27834912)

[Build 6](#_Toc27834913)

[Run 6](#_Toc27834914)

[Result 6](#_Toc27834915)

[ubuntu 6](#_Toc27834916)

[dockerfile 6](#_Toc27834917)

[Build 6](#_Toc27834918)

[Run 7](#_Toc27834919)

[Result 7](#_Toc27834920)

[Console logs 7](#_Toc27834921)

[Run Apache HTTP Server 7](#_Toc27834922)

[Healthchecks 7](#_Toc27834923)

[add in dockerfile 7](#_Toc27834924)

[Linking Docker Containers 7](#_Toc27834925)

[MySQL 7](#_Toc27834926)

[PHP adminer 7](#_Toc27834927)

[Result 7](#_Toc27834928)

[Docker Compose 8](#_Toc27834929)

[Install docker compose 8](#_Toc27834930)

[docker-compose.yml 8](#_Toc27834931)

[Run 8](#_Toc27834932)

[Network 8](#_Toc27834933)

[Network in docker-compose.yml 8](#_Toc27834934)

[Connect container to network 9](#_Toc27834935)

[Scaling 9](#_Toc27834936)

[Volumes 9](#_Toc27834937)

[Store mysql files on hosting machine 9](#_Toc27834938)

[Folder in host 9](#_Toc27834939)

[Folder in container 9](#_Toc27834940)

[docker-compose.yml 9](#_Toc27834941)

[Docker Hub 10](#_Toc27834942)

[Other registry 10](#_Toc27834943)

[Docker Swarm 10](#_Toc27834944)

[Create manager 10](#_Toc27834945)

[Create node 10](#_Toc27834946)

[Docker Services 10](#_Toc27834947)

# Tools

## Docker

<https://docs.docker.com/docker-for-windows/install/>

## cmder

<https://cmder.net>

# Command Line

cd .. // one folder up

ls // list files

# Hello World

docker run hello-world

<https://hub.docker.com/_/hello-world>

## Dockerfile

FROM scratch

COPY hello /

CMD ["/hello"]

# Containers

docker -v // version

docker ps // actively running containers

docker ps -a // install containers on machine

docker rm // remove container

docker images // show built images

docker stop container-name // stop a container

docker remove container-name // stop and remove a container

$(docker ps -q) // list all containers IDs

docker stop $(docker ps -q) // stop all containers

docker remove $(docker ps -a -q) // stop and remove all containers

docker inspect container-name // json with data about the container

docker events // show events

# Sample Console App

## dockerfile

FROM ubuntu:[latest]

CMD echo “hello world”

## Build

docker build . -t container-name

. for current directory

-t for tags

## Run

docker run container-name

# Sample Web App with NGINX

## index.html

<html>

<body>

<h1>First Web App</h1>

</body>

</html>

## nginx.conf

config file for web server

server {

root /www;

}

## dockerfile

FROM nginx

ADD index.html /www/ // copy html file to container

ADD nginx.conf /etc/nginx/conf.d/default.conf

// copy config file for web server

EXPOSE 80

EXPOSE 443

CMD nginx -g “daemon off”;

## Build

docker build . -t web-app-sample

## Run

docker run web-app-sample

docker run -d -p 84:80 web-app-sample

-d run as a background process and release the terminal

-p 84:80 map a port 80 from nginx to 84

## Test

<http://dockerIP:84/index.html>

# Access Container Shell

docker run -it ubuntu /bin/bash // run a container with Shell access

exit

docker exec -it ubuntu /bin/bash // execute a running container

# PHP

## sample.php

<?php

Print(“Sample PHP”);

?>

## Build

docker run -it -v “$(pwn)/sample.php:/home/sample.php” php /bin/bash

-v specify the volume for the file

## Run

php sample.php

# Text file

## dockerfile

FROM ubuntu

RUN echo “sample text” > /home/text.txt

// create a file with “sample text” in home folder

RUN echo “sample text” > /home/text.txt \

&& echo “bla” > /home/bla.txt

// run two commands in same build step

## Build

docker build . -t ubuntu-text

## Run

docker run -it ubuntu-text /bin/bash

## Result

ls // show a new file text.txt in home folder

cat /home/text.txt // show the file content

# ubuntu

## dockerfile

FROM ubuntu

RUN apt-get update // update ubuntu

RUN apt-get install -y –-no-install-recommends vim

// -y force install if there are questions

// --no-install-recommends discard recommendations

// vim editor to install

RUN apt-get clean // clean temp files

## Build

docker build . -t ubuntu-vim

## Run

docker run -it ubuntu-vim /bin/bash

## Result

Open the vim editor

# Console logs

docker logs container-name // show log

docker logs -f container-name // continuous log

docker logs –tail 10 container-name // show last 10 log

# Run Apache HTTP Server

docker run -d -p httpd

# Healthchecks

curl -I -f “http://localhost:32777”

## add in dockerfile

HEALTHCHECK --interval=15s --retries=5 --timeout=30s –-start-period=30s CMD curl -I -f “http://localhost:80” || exit 1

--interval // interval between checks

--retries // how many retries before restart

--timeout // timeout before fail

--start-period // start time required for container to start

--port // port inside the container

# Linking Docker Containers

## MySQL

docker run --name our-mysql -e MYSQL\_ROOT\_PASSWORD=password -d mysql

--name // custom name

-e // environment variables

## PHP adminer

docker run --link “out-mysql:db” -p 8080:8080 adminer

## Result

The php apache will show a login page to the mysql admin web application

Username: root

Password: password

# Docker Compose

## Install docker compose

<https://docs.docker.com/compose/install/>

## docker-compose.yml

version: “3.1”

services:

db:

image: mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: password

adminer:

image: adminer

restart: always

ports:

* 8080:8080

## Run

Creates a default network

docker-compose -f filename up -d // start

-f set a specific filename or use docker-compose.yaml

docker-compose down // stop and remove

docker-compose pull // pull the images to the host

docker-compose restart // restart all the containers

docker-compose pause // pause all the containers

docker-compose unpause // unpause all the containers

docker-compose stop // stop

docker-compose start // start

docker-compose top // some info

docker-compose logs // show logs

docker-compose events // show events

# Network

docker-compose created a default internal network

docker network ls // list networks

docker network create network-name // manual create a network

docker network prune // delete all network not used

docker network inspect network-name // info on network

## Network in docker-compose.yml

version: “3.1”

services:

db:

image: mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: password

network:

* backend

adminer:

image: adminer

restart: always

ports:

* 8080:8080

network:

* frontend

network:

default:

external:

name: network-name // default name is container-name\_default

frontend:

driver: bridge

backend:

driver: bridge

## Connect container to network

docker networks connect network-name container-name

docker networks disconnect network-name container-name

# Scaling

docker-compose up -d --scale nginx=4

--scale // starts 4 instances of this image

# Volumes

docker volume ls // list volumes

docker volume prune // delete all volumes not used

docker volume inspect network-name // info on volume

## Store mysql files on hosting machine

MySQL container will store the file on the hosting machine by mapping folder

### Folder in host

/var/lib/docker/volumes/container-name/\_data/

### Folder in container

/usr/lib/mysql/

### docker-compose.yml

version: “3.1”

services:

db:

image: mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: password

volumes:

* dbdata:/usr/lib/mysql

adminer:

image: adminer

restart: always

ports:

* 8080:8080

volumes:

dbdata:

driver: local

# Docker Hub

docker login

docker build image-name

docker push image-name

## Other registry

* Quay.io
* Azure Container Registry
* Amazon Elastic Container Registry
* Google Container Registry
* Private Docker Registry
* Artifactory

# Docker Swarm

## Create manager

docker swarm init

## Create node

docker-machine ls

docker-machine create

# Docker Services

docker service ls

docker service create nginx

docker service update --replicas 6 container-name