

Docker

A vibrant, cartoon-style illustration of a landscape. In the background, there are snow-capped mountains under a blue sky with white clouds. A penguin stands on a patch of snow in the middle ground. To the left, an octopus with a smiling face and large eyes is holding a small blue box. In the foreground, a blue, blob-like creature with large eyes is lying on a green mat on a grassy hill, next to a small blue cup. To the right, a large blue whale is partially visible. The scene is framed by a large black circle, and a smaller black circle highlights a specific area in the middle ground.

14 May 2018



Hello!

I Am Cahya Sulianto Wibawa

You can contact me at
Email : inbox@cahsul.com
Wa : 0899 6120 713

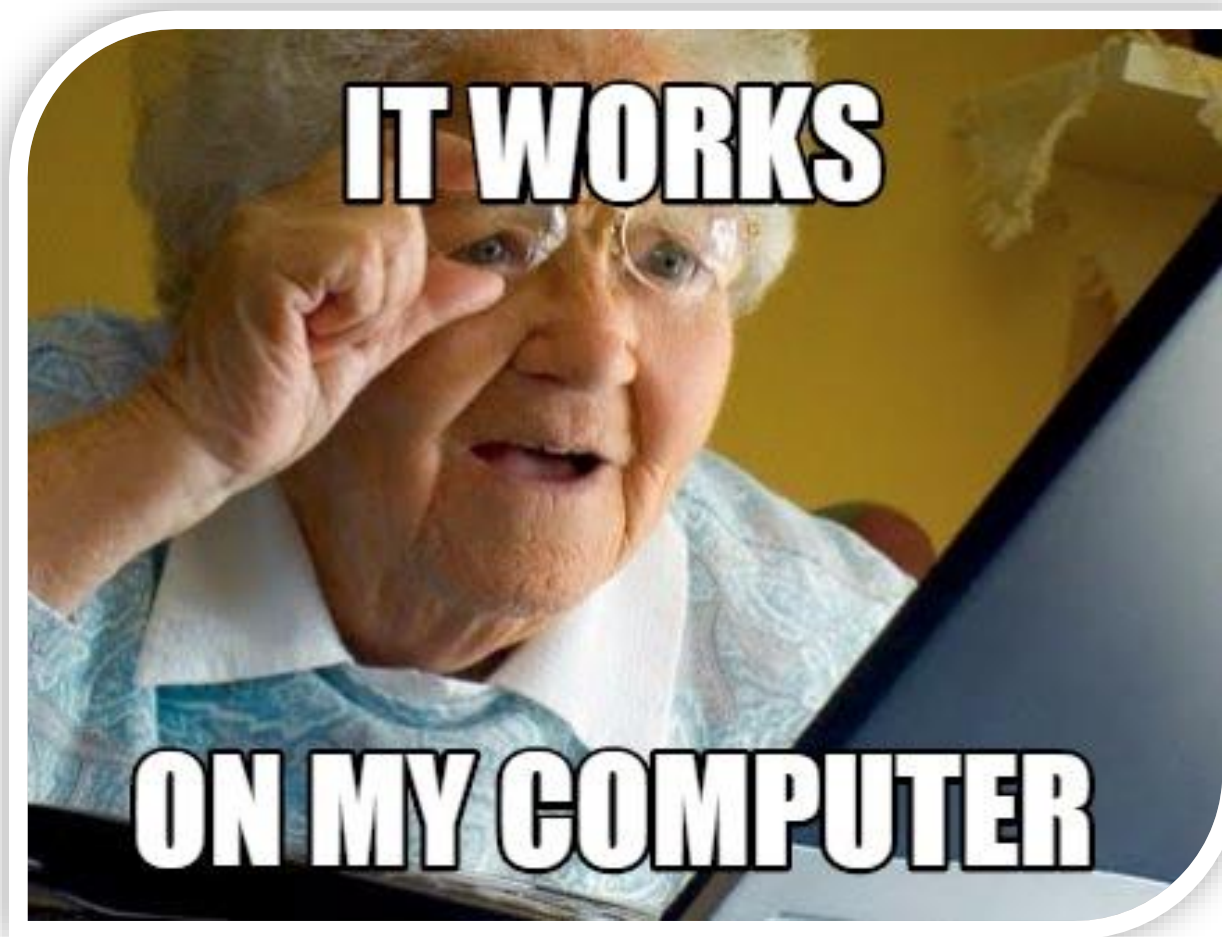
A scenic landscape featuring a deep, rocky valley with green patches of vegetation. In the distance, numerous hot air balloons of various colors (yellow, red, blue, white) are floating in the sky, suggesting a festival or competition. The sky is a soft, hazy blue, and the overall atmosphere is peaceful and expansive.

“

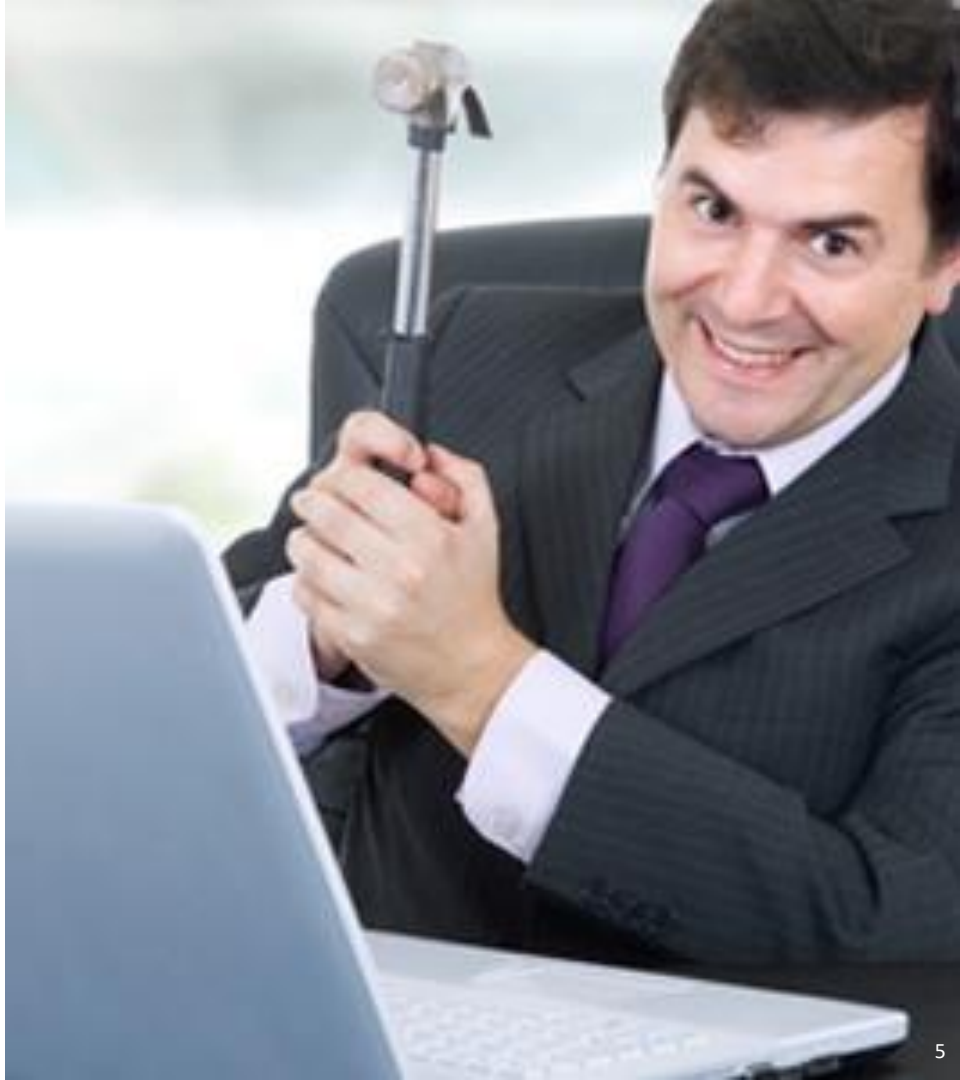
Diberikan **ilmu – ilmu** yang **bermanfaat**

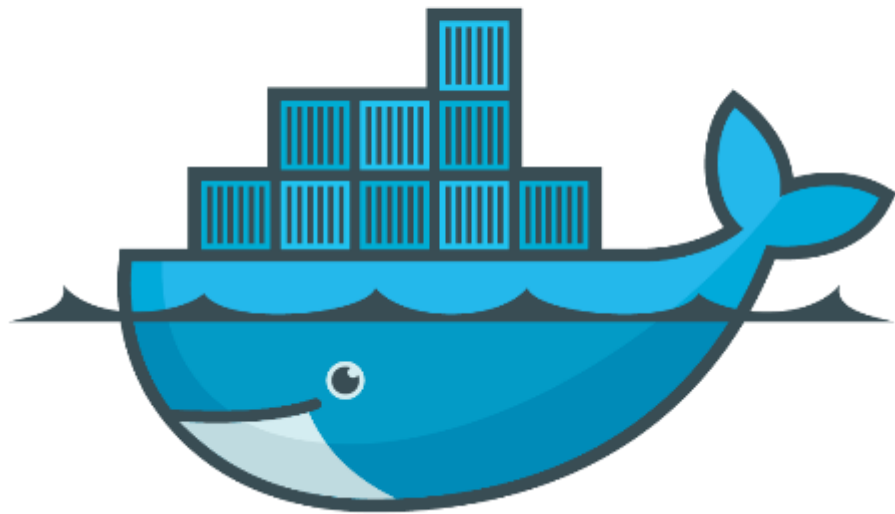
Dijauhkan dari **ilmu – ilmu** yang **tidak bermanfaat**

Diberikan **pemahaman ilmu** yang **mendalam**



has no uninstaller





docker

WHAT IS
DOCKER

What is Docker ?

Docker is a [computer program](#) that performs [operating-system-level virtualization](#) also known as [containerization](#).^[6]

It is developed by [Docker, Inc.](#)^[7]

Docker is primarily developed for [Linux](#), where it uses the resource isolation features of the [Linux kernel](#) such as [cgroups](#) and kernel [namespaces](#), and a [union-capable file system](#) such as [OverlayFS](#) and others^[8] to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining [virtual machines](#) (VMs).^[9]

The Linux kernel's support for namespaces mostly^[10] isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU.^[11]

Since version 0.9, Docker includes the **libcontainer library** as its own way to directly use virtualization facilities provided by the Linux kernel, in addition to using abstracted virtualization interfaces via [libvirt](#), [LXC](#) and [systemd-nspawn](#).

What is Docker ?

Open platform for developers and sysadmins to build, ship and run distributed applications

Can run on popular 64-bit Linux distributions with kernel 3.8 or later

Supported by several cloud platforms including Amazon EC2, Google Compute Engine, and Rackspace.

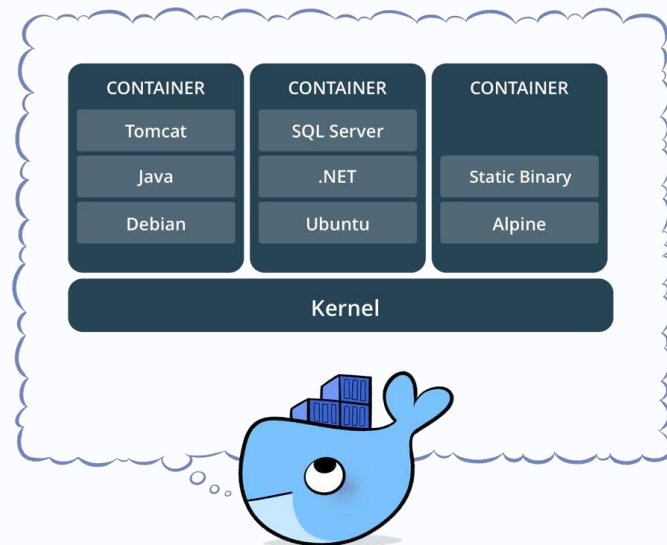
Standardized packaging for software and dependencies

Isolate apps from each other

Share the same OS kernel

Works for all major Linux distributions

Containers native to Windows Server 2016



Why Docker ?

Speed. Speed. Speed.

Making dev->test->prod easier and faster

"make my code run the same way on two different machines"

Reduce complexity of developing code for distributed systems

Reduce complexity of deploying code to the cloud

Reduce complexity of updating code in the cloud

Docker Features

Light-Weight

Minimal overhead (cpu/io/network)

Based on Linux containers

Uses layered filesystem to save space (AUFS/LVM)

Uses a copy-on-write filesystem to track changes

Portable

Can run on any OS.

Raspberry pi support.

Self-sufficient

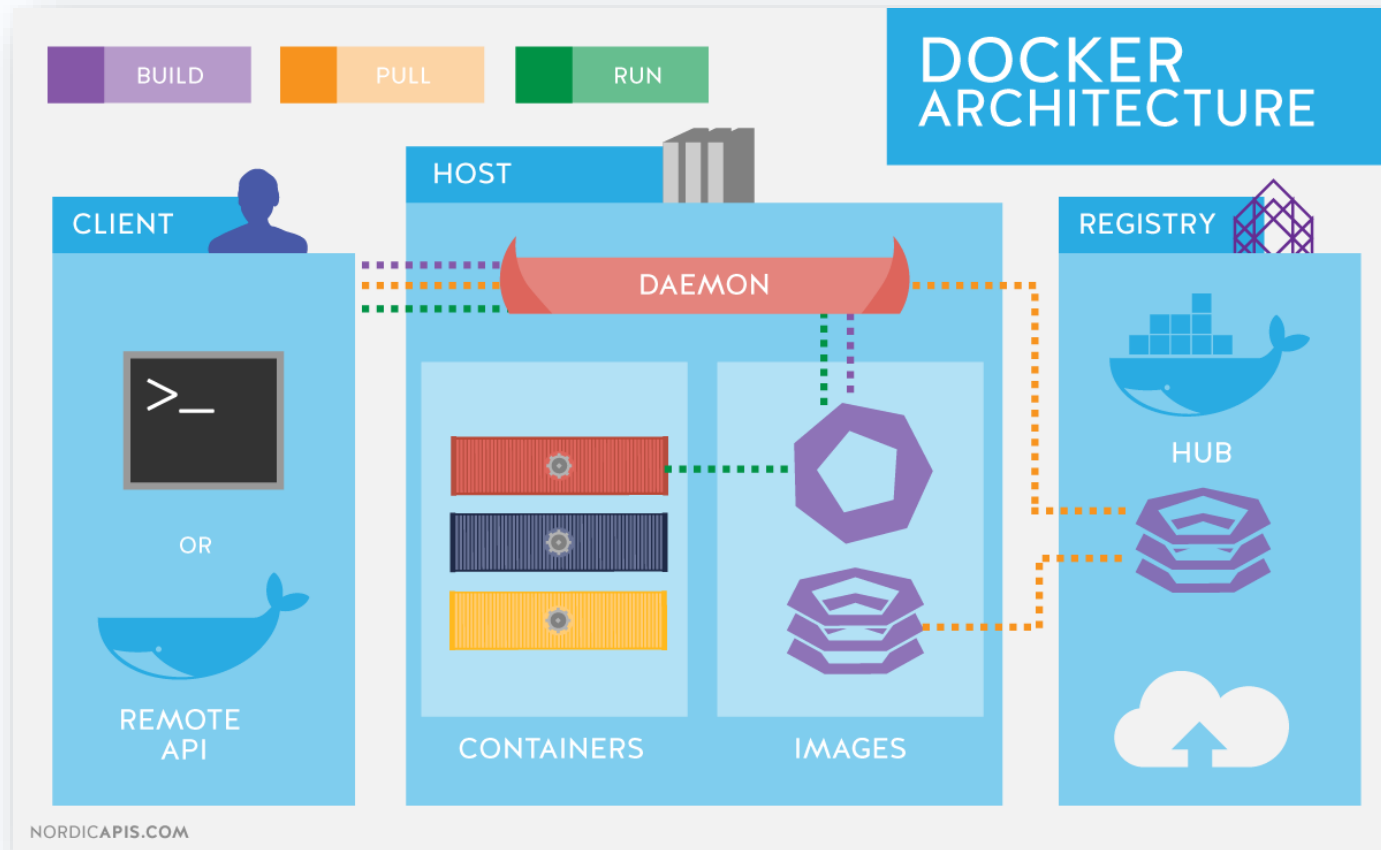
A Docker container contains everything it needs to run

Minimal Base OS

Libraries and frameworks

Application code

A docker container should be able to run anywhere that Docker can run.



Docker VS VM

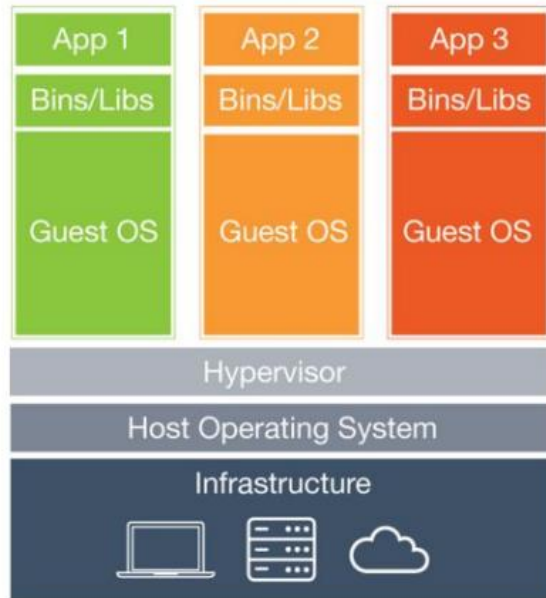
Containers aren't Mini-VM's

They are just processes

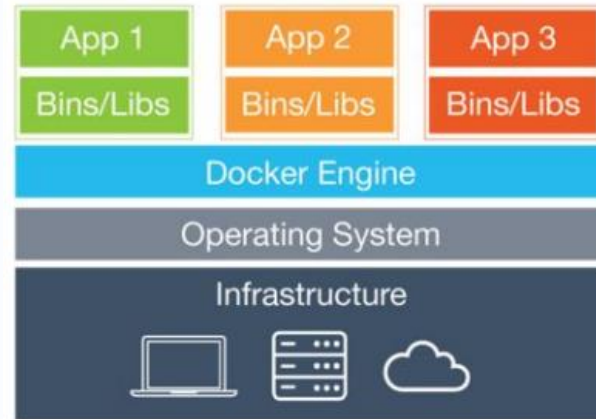
Limited to what resources they can access (file paths, network devices, running processes)

Exit when process stops

Docker VS VM



Virtual Machines



Containers



Service Providers



Dev Tools



Official Repositories



Operating Systems



Configuration Management



Big Data



Service Discovery



Orchestration



System Integrators



I'll Show You A Thing Or Two About...



Image & Container

- Images vs Container
- Docker Hub
- Official Image
- Volume
- Tagging / Pushing
- Dockerfile



Install

Windows, MAC, LINUX



Docker Compose

- yaml files
- Compose commands
- Building images



1

Install Docker

Docker (Windows)

Win 10 Pro / Ent Only

Win7/8/8.1 or Win10 Home should use **Docker Toolbox**

Windows Server 2016 also supports Windows Containers

Use Docker for Windows, from **store.docker.com**

More features then just a Linux VM

Uses **Hyper-V** with **tiny Linux VM** for Linux Containers

Docker (MAC)

Use Docker for Mac, from **store.docker.com**

More features than just a Linux VM

Don't use homebrew (brew install docker), it's docker CLI only

Requires Yosemite 10.10.3 (2014 release)

Yosemite works with **2007-2008** Mac's and newer

For Snow Leopard, Lion, Mountain Lion (10.6-10.8) use **Docker Toolbox**

Docker (Linux)

Easiest install/setup, **best** native experience

Three main ways to install: script, store, or docker-machine

get.docker.com script (latest Edge release)

store.docker.com has instructions for each distro

RHEL officially only supports Docker EE (**paid**), but CentOS will work

Installing in a VM, Cloud Instance, all are the same process

May **not work** for unlisted distros (Amazon Linux, Linode Linux, etc.)

Don't use pre-installed setups (Digital Ocean, Linode, etc.)



2

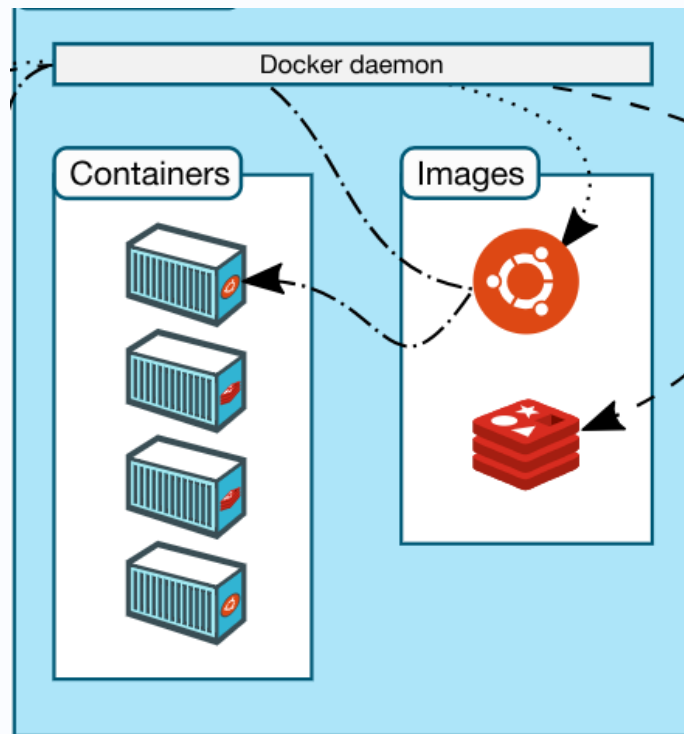
Image & Container

Image Vs Container


An **Image** is the application we want to run

A **Container** is an **instance** of that image running as a **process**

You can have **many containers** running off the **same image**








```
01 <?php
02
03 class MyClass
04 {
05     public $prop1 = "I'm a class property!";
06
07     public function setProperty($newval)
08     {
09         $this->prop1 = $newval;
10     }
11
12     public function getProperty()
13     {
14         return $this->prop1 . "<br />";
15     }
16 }
17
18 // Create two objects
19 $obj = new MyClass;
20 $obj2 = new MyClass;
21
22 // Get the value of $prop1 from both objects
23 echo $obj->getProperty();
24 echo $obj2->getProperty();
25
26 // Set new values for both objects
27 $obj->setProperty("I'm a new property value!");
28 $obj2->setProperty("I belong to the second instance!");
29
30 // Output both objects' $prop1 value
31 echo $obj->getProperty();
32 echo $obj2->getProperty();
33
34 ?>
```



DashboardExploreOrganizationsCreatecahsul

Explore Official Repositories

 alpine official	3.5K STARS	10M+ PULLS	> DETAILS
 nginx official	8.5K STARS	10M+ PULLS	> DETAILS
 httpd official	1.7K STARS	10M+ PULLS	> DETAILS
 busybox official	1.2K STARS	10M+ PULLS	> DETAILS
 redis	5.1K	10M+	>

Basic Docker Commands

docker version => verified it's working

docker info => most config values

docker <command> <sub-command> (options)=> docker command line structure


```
docker run
  --rm remove container automatically after it exits
  -it connect the container to terminal
  --name web name the container
  -p 5000:80 expose port 5000 externally and map to port 80
  -v ~/dev:/code create a host mapped volume inside the container
  alpine:3.4 the image from which the container is instantiated
  /bin/sh the command to run inside the container
```

Stop a running container through SIGTERM
docker stop web

Stop a running container through SIGKILL
docker kill web

Create an overlay network and specify a subnet
docker network create --subnet 10.1.0.0/24
--gateway 10.1.0.1 -d overlay mynet

List the networks
docker network ls

List the running containers
docker ps

Delete all running and stopped containers
docker rm -f \$(docker ps -aq)

Create a new bash process inside the container and connect it to the terminal
docker exec -it web bash

Print the last 100 lines of a container's logs
docker logs --tail 100 web

Pull an image from a registry
docker pull alpine:3.4

Retag a local image with a new image name and tag
docker tag alpine:3.4 myrepo/myalpine:3.4

Log in to a registry (the Docker Hub by default)
docker login my.registry.com:8000

Push an image to a registry
docker push myrepo/myalpine:3.4

Build an image from the Dockerfile in the current directory and tag the image

```
docker build -t myapp:1.0 .
```

List all images that are locally stored with the Docker engine

```
docker images
```

Delete an image from the local image store

```
docker rmi alpine:3.4
```

Assignment - 1

Use docker pull to get image **ubuntu 16.04**

Create a **container** ubuntu 16.04

Check the container to always up

Access the container console

run apt-get update

Cleanup the container

Volume

Data volumes are designed to **persist data**
independent of the container's lifecycle.

Docker volume ...

`docker run -v /path/on/host:/path/in/container ...`

Assignment – 2 - Volume

Run a **php 7.2** with **apache**

Use volume

Apache should listen port **8080:80**

Cleaup

Run a **Mysql**

Use volume

Cleaup

Docker File

A **Dockerfile** is a **text** file that **defines** a Docker **image**. You'll use a Dockerfile to **create** your **own** custom Docker **image**.

It supports a **simple** set of **commands** that you need to use in your Dockerfile

There are several commands supported like **FROM**, **CMD**, **ENTRYPOINT**, **VOLUME**, **ENV** and more

Assignment – 3 - Dockerfile

FROM nginx:latest

WORKDIR /usr/share/nginx/html

COPY index.html index.html



3

Docker Compose



Docker Compose

configure relationships between containers

save our docker container run settings in easy-to-read file

create one-liner developer environment startups

Compose YAML format has it's own versions: 1, 2, 2.1, 3, 3.1

YAML file can be used with docker-compose command for local docker automation or 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

Docker Compose

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

Assignment – 4 – Docker Compose

Build a basic compose file for a Drupal content management system website

Use the drupal image along with the postgres image

Use ports to expose Drupal on 8080 so you can localhost:8080

Be sure to set POSTGRES_PASSWORD for postgres

Walk through Drupal setup via browser

Tip: Drupal assumes DB is localhost, but it's service name

A large, irregular blue watercolor splash shape is centered on a white background. The splash has a gradient from light blue to dark blue. Below the splash, there are scattered blue and teal dots and small leaf-like shapes.

“

“A person who never made a
mistake never tried anything
new”