EDEM

Docker

September 2021

Agenda

1. Microservice Architecture
2. Containers
3. Docker
4. Docker Compose
5. Kubernetes

1

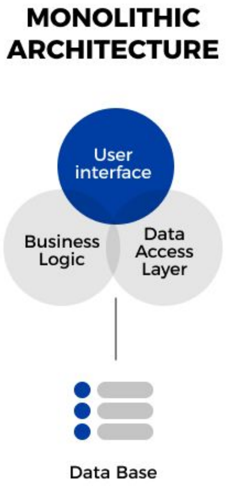
Microservice Architecture

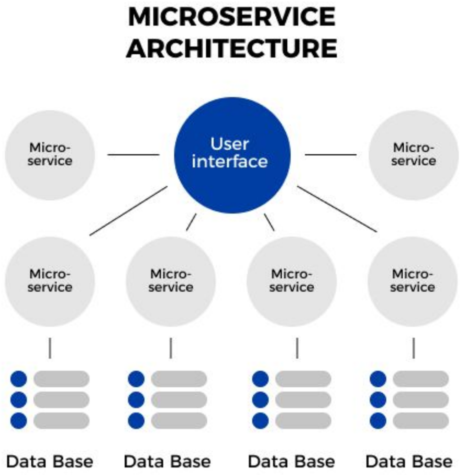
Microservices

- Microservice Architecture is an architectural style that structures an application as a collection of services that are
    - Highly maintainable and testable
    - Loosely coupled
    - Independently deployable
    - Organized around business capabilities
    - Owned by a small team

- The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications
    - It also enables an organization to evolve its technology stack

Microservice Architecture


MONOLITHIC ARCHITECTURE
User interface
Business Logic
Data Access Layer
Data Base


MICROSERVICE ARCHITECTURE
Micro-service
User interface
Micro-service
Micro-service
Micro-service
Micro-service
Micro-service
Data Base
Data Base
Data Base
Data Base

- Developer issues:
  - Minor code changes require full re-compile and re-test
  - Application becomes single point of failure
  - Application is difficult to scale

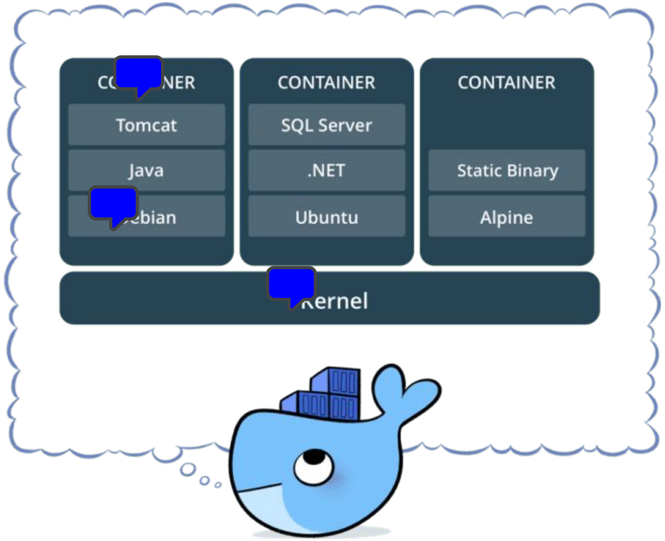
- Microservice:
  - Break application into separate operations
  - Make the app independently, scalable, stateless, highly available by design

2

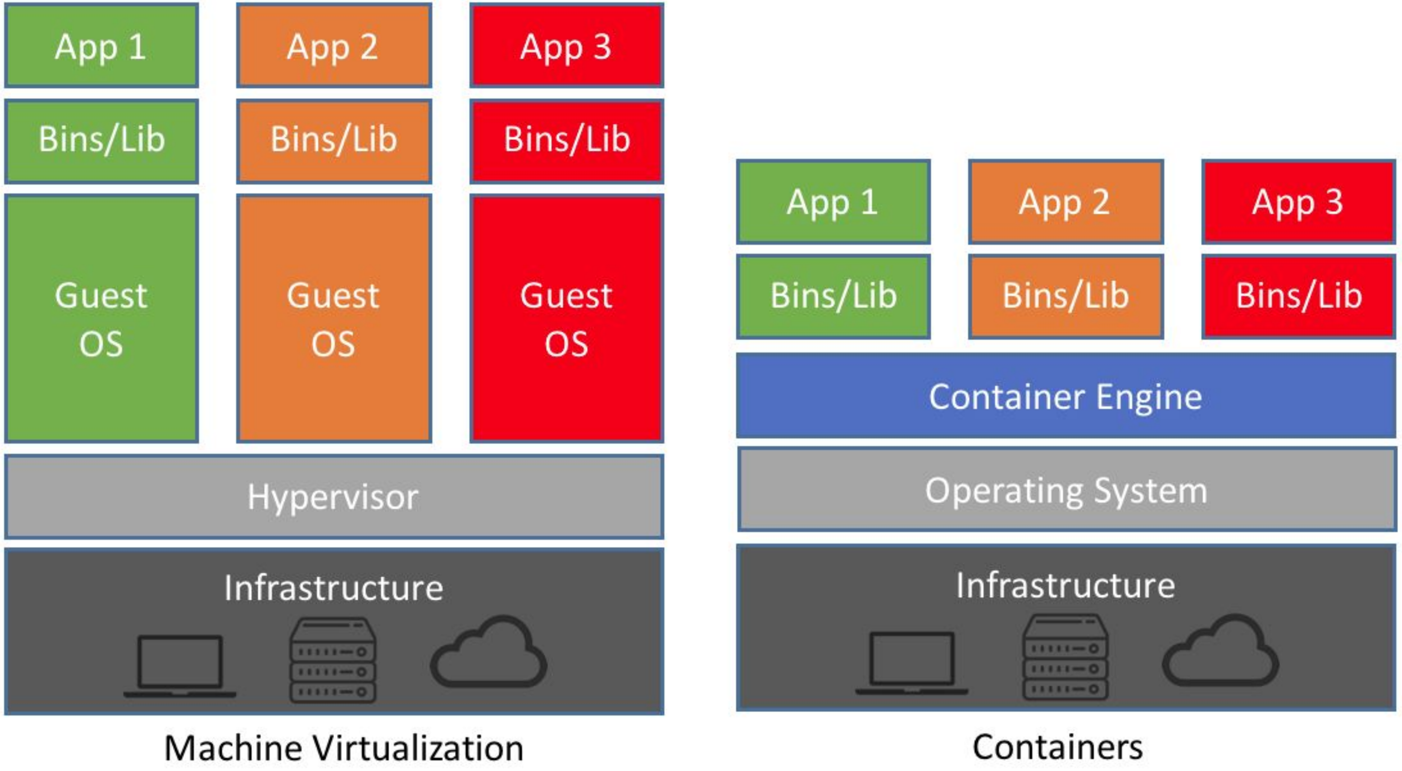
Containers

What is a Container?

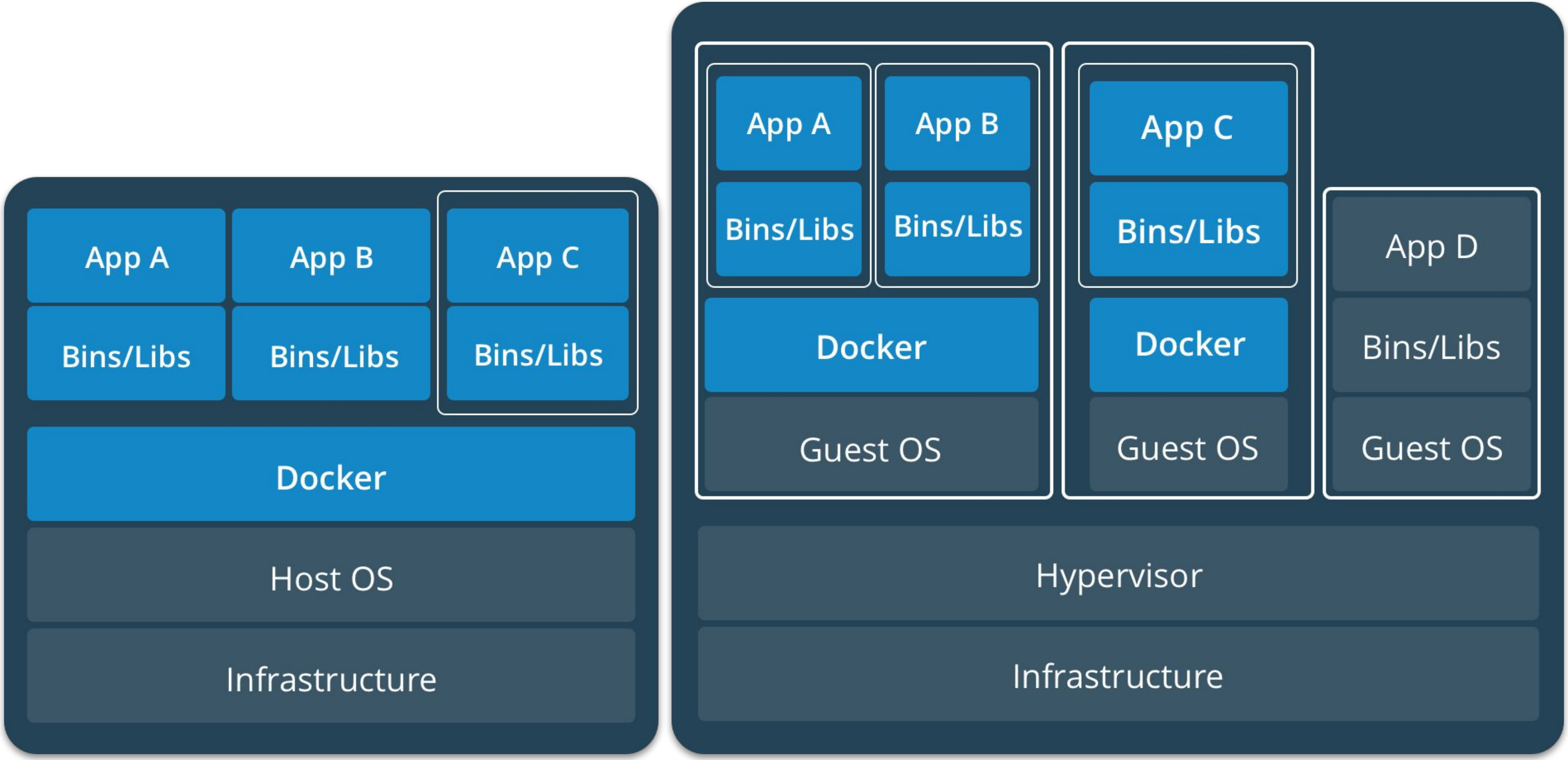
- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS Kernel
- Works with all major Linux and Windows Server


CC NER
Tomcat
Java
ebian
CONTAINER
SQL Server
.NET
Ubuntu
CONTAINER
Static Binary
Alpine
ernel

Containers vs. VMs


App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
Machine Virtualization
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure
Containers

Containers & VMs together


App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Docker
Guest OS
Guest OS
App D
Bins/Libs
Guest OS
Hypervisor
Infrastructure

Containers & VMs together

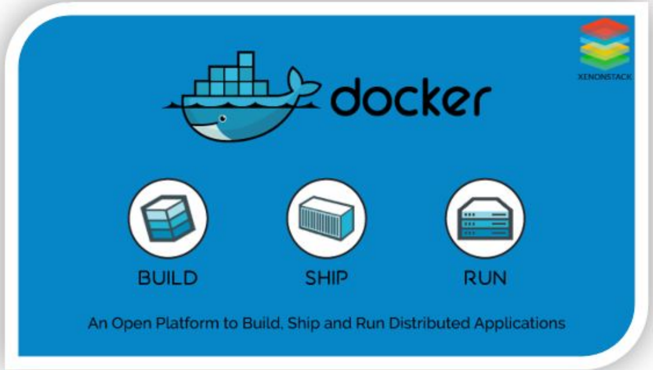
APP
NW
OS Win
IDG.tv

3

Docker

Docker

- Docker is an open platform for developing, shipping, and running containerized applications
- With Docker, you can manage your infrastructure in the same way you manage your applications
- No OS to boot → Applications online in seconds


docker
XENONSTACK
BUILD
SHIP
RUN
An Open Platform to Build, Ship and Run Distributed Applications

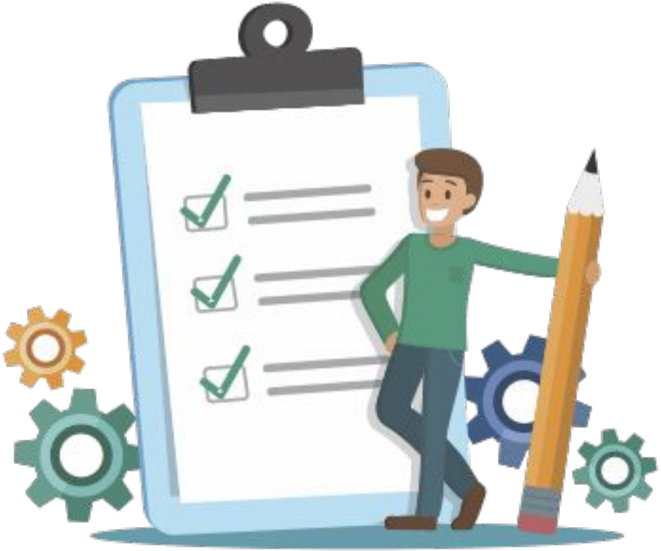
Hands-on

- Docker in your laptop
  - Windows Users (Windows 10 Enterprise & pro & home):

    https://docs.docker.com/desktop/windows/install/
  - Mac Users

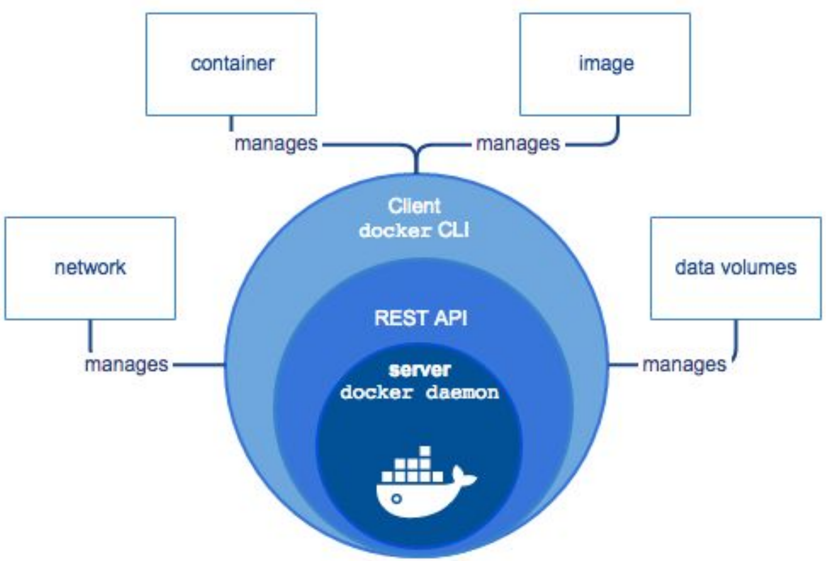
    https://docs.docker.com/desktop/mac/install/
  - Ubuntu Users

    https://docs.docker.com/engine/install/ubuntu/



Docker Engine

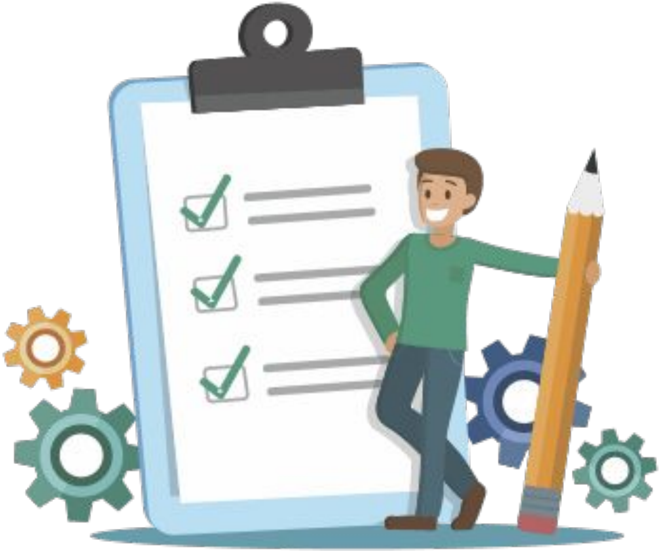
- Docker Engine is a client-server application with these major components:
  - Server
  - REST API
  - Command Line Interface (CLI)


container
image
manages
manages
Client
docker CLI
network
data volumes
REST API
server
docker daemon
manages
manages

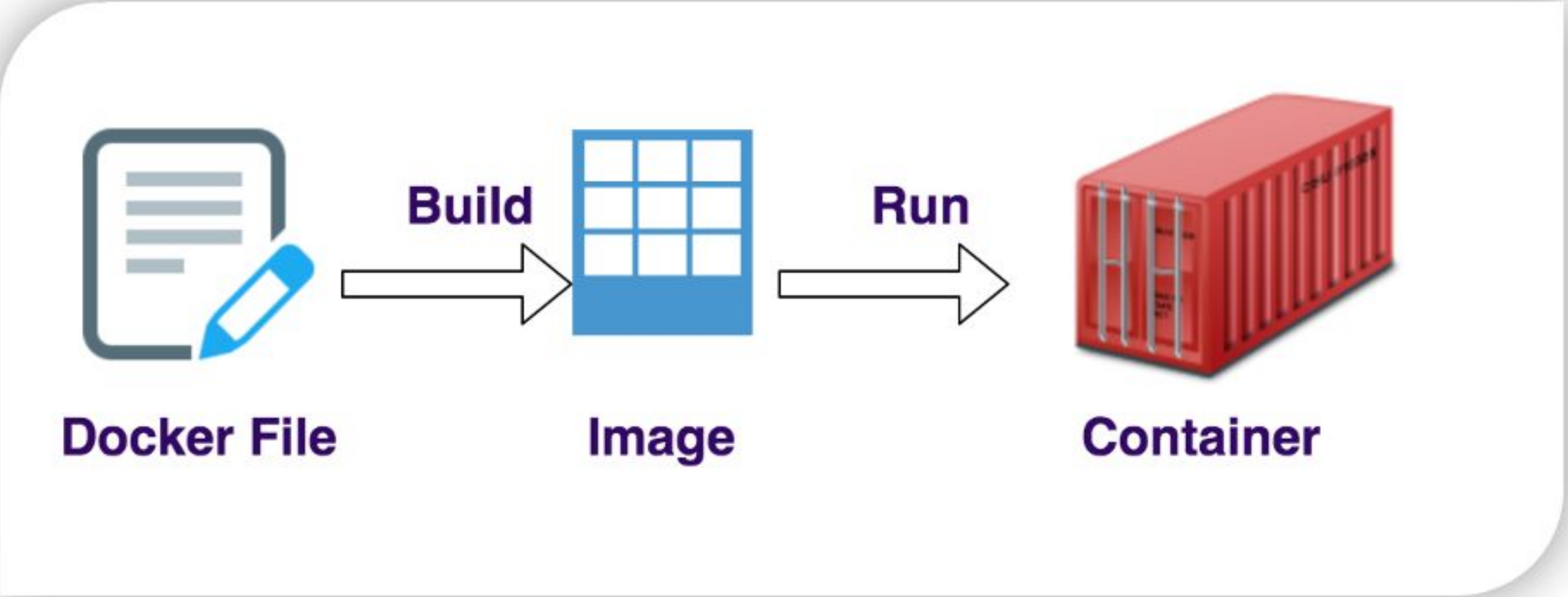
Hands-on

- Docker Google Cloud:
    - Create an **VM instance** with the following features:
        - Zone: us-central1-a
        - Machine: e2-micro
        - OS: Container optimized OS
        - Allow HTTP/HTTPS
    - $ docker version
    - $ docker run -dp 80:80 docker/getting-started
    - Your browser: http://[VM-IP]

- Execute the same instructions in your local machine (Optional)



Docker Concepts

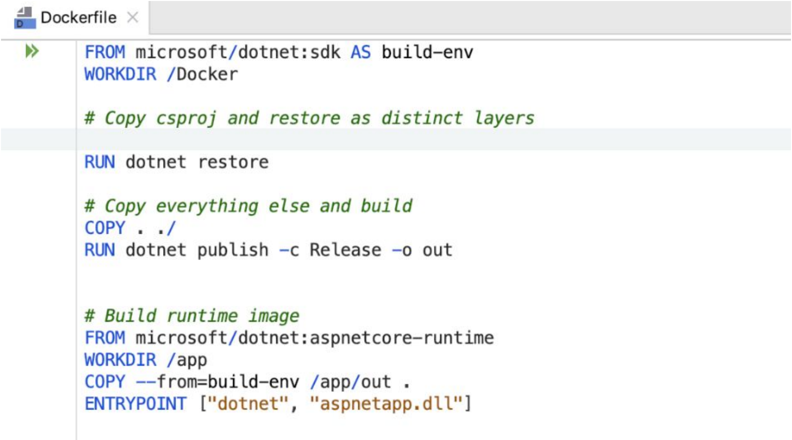

Build
Run
Docker File
Image
Container

Docker File

- A DockerFile is a text document that contains all the commands a user could call on the command line to assemble an image
    - You can consider DockerFile as blueprint of Docker Image

- DockerFile as a sequential set of instruction for Docker Engine
    - Order of sequence is important!!
    - Each instruction creates a layer
    - A stack of such sequenced layers managed by a filesystem becomes a docker image
    - Layers can be cached and reused by Docker

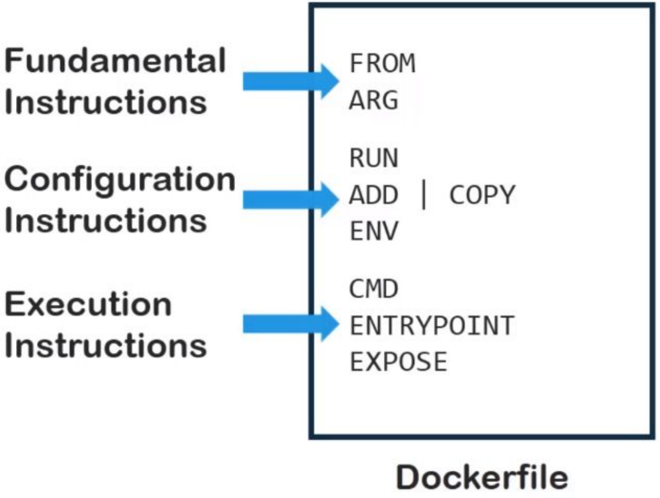
- Primary way to interacting with Docker



```
FROM microsoft/dotnet:sdk AS build-env
WORKDIR /Docker

# Copy csproj and restore as distinct layers

RUN dotnet restore

# Copy everything else and build
COPY . ./
RUN dotnet publish -c Release -o out


# Build runtime image
FROM microsoft/dotnet:aspnetcore-runtime
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "aspnetapp.dll"]
```

Docker File Structure

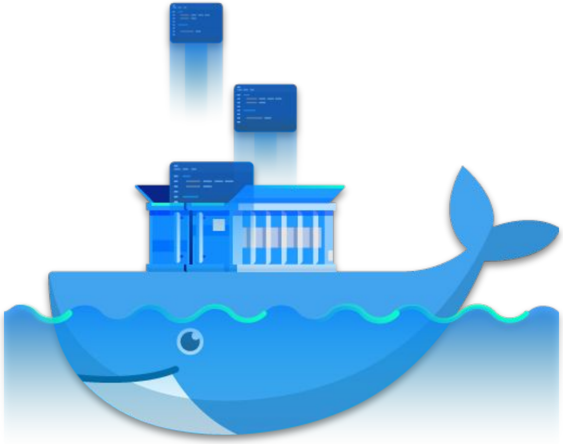
- It’s a file with no extension called “Dockerfile”
- The instructions can be generally divided into three categories:
  - Fundamental
  - Configuration
  - Execution


Fundamental Instructions
FROM
ARG
Configuration Instructions
RUN
ADD | COPY
ENV
Execution Instructions
CMD
ENTRYPOINT
EXPOSE
Dockerfile

EDEM

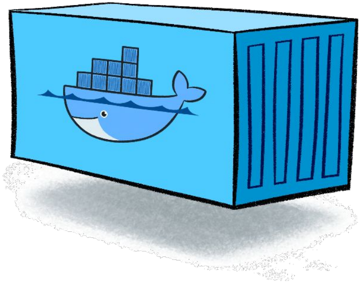
Docker Image

- A stack of multiple layers created from DockerFile instructions
- Recognized by name or Image ID
- The are pushed to and can be pulled from Docker Hub



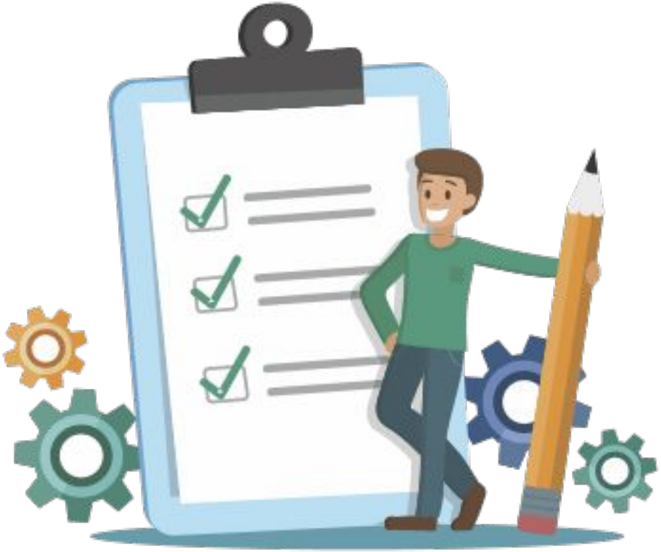
Docker Container

- Running instance of a Docker Image
- Provides similar isolation to VMs but lighter!
- Adds writable layer on top of image layers and works on it
- Can talk to other containers like processes in Linux
- Provide resources to an image



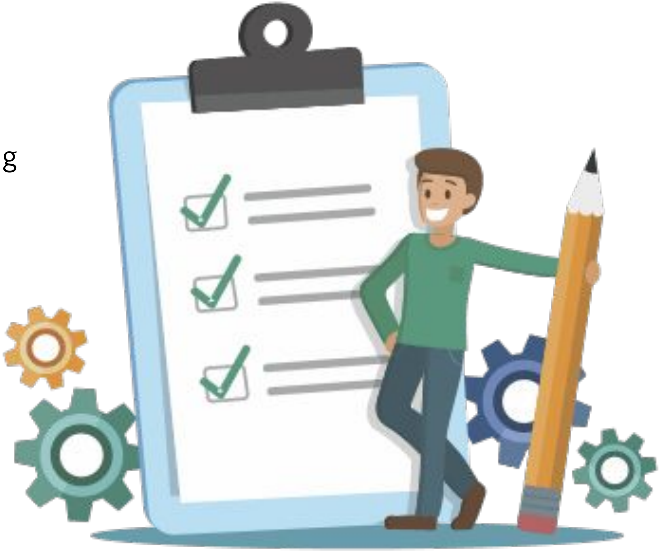
Hands-on

- **Exercise 1**
    - Create a new DockerFile
    - `$ docker build -t first_edem_img .`
    - `$ docker images`



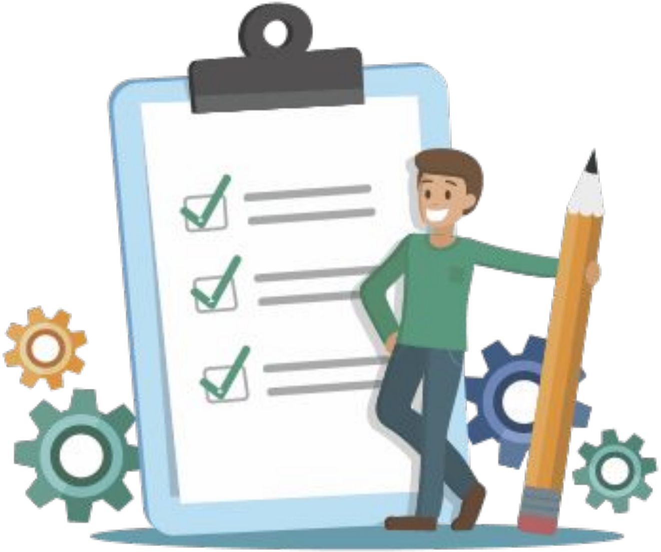
Hands-on

- **Exercise 2**
  - Create a new DockerFile
  - `$ docker build -t second_edem_img .`
  - `$ docker images`
  - `$ docker run -itd --name cont_second_edem second_edem_img`
  - `$ docker ps -a`
  - `$ docker exec -it cont_second_edem bash`



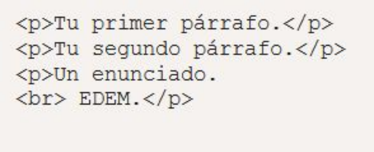
Hands-on

- **Exercise 3**
  - Create a new DockerFile
  - `$ docker build -t third_edem_img .`
  - `$ docker images`
  - `$ docker run -itd --name cont_third_edem -p 8080:80 third_edem_img`
  - `$ docker ps -a`
  - Using your browser, go to this URL: http://localhost:8080

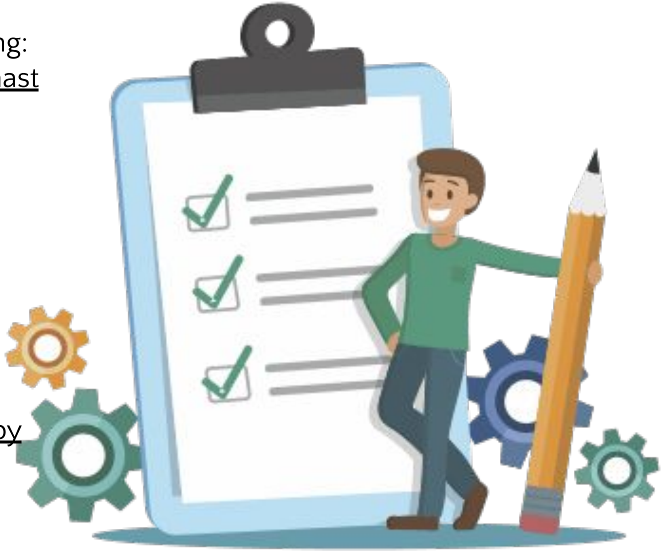


Hands-on

- **Exercise 4**
    - Create a new file called “index.html” which contains the following:
        - https://raw.githubusercontent.com/masfworld/edem/master/index.html



```
<p>Tu primer párrafo.</p>
<p>Tu segundo párrafo.</p>
<p>Un enunciado.
<br> EDEM.</p>
```

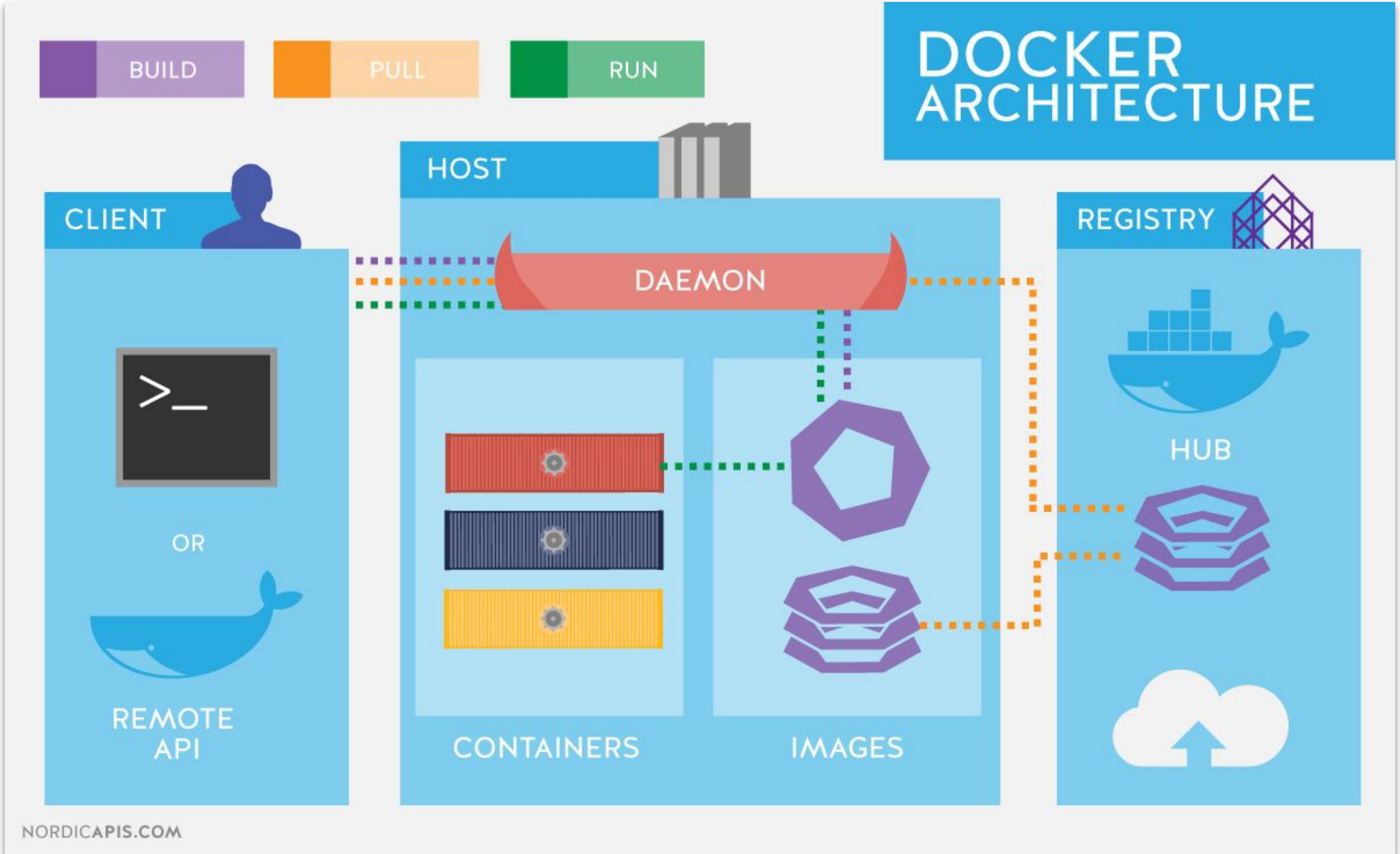


    - Modify Dockerfile from Demo 3 to COPY file “index.html” into “/var/www/html”
        - https://docs.docker.com/engine/reference/builder/#copy
    - Dockerfile → https://raw.githubusercontent.com/masfworld/edem/master/DockerFile
    - Generate a container with the previous web page

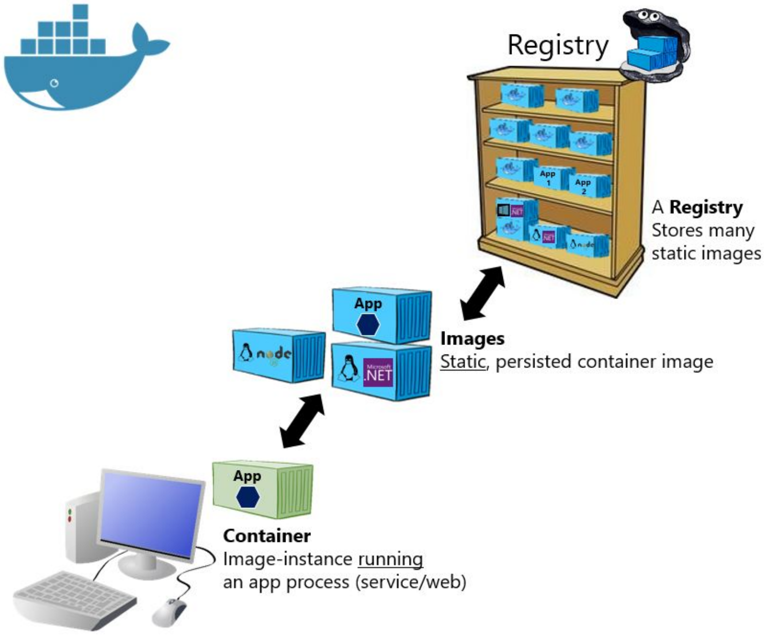
Docker Registry

- The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images
  - Fully own your images distribution pipeline
  - Locally or using Docker Hub

docker
REGISTRY


BUILD
PULL
RUN
DOCKER ARCHITECTURE
HOST
CLIENT
REGISTRY
DAEMON
>_
OR
REMOTE API
CONTAINERS
IMAGES
HUB
NORDICAPIS.COM

Basic taxonomy in Docker



A **Registry**
Stores many
static images

Images
Static, persisted container image

Container
Image-instance running
an app process (service/web)

On-premises
('n' private organizations)

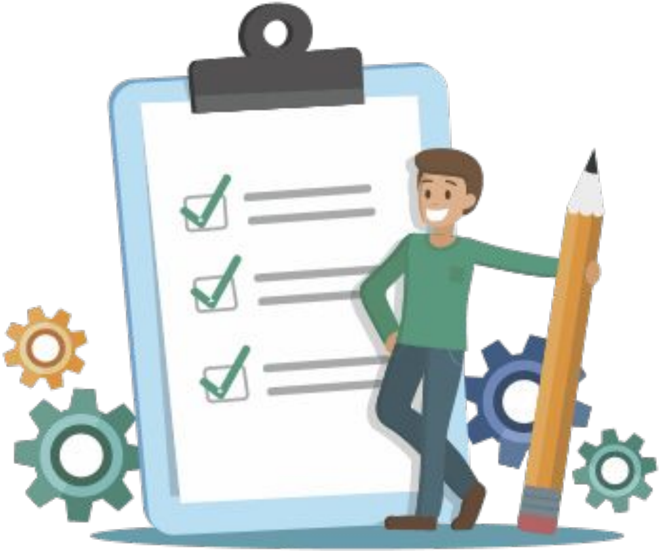
- **Hosted Docker Registry**
- **Docker Trusted Registry on-prem.**

Public Cloud
(specific vendors)

- **Docker Hub Registry**
- Docker Trusted Registry on-cloud
- **Azure Container Registry**
- AWS Container Registry
- Google Container Registry
- Quay Registry
- Other Cloud

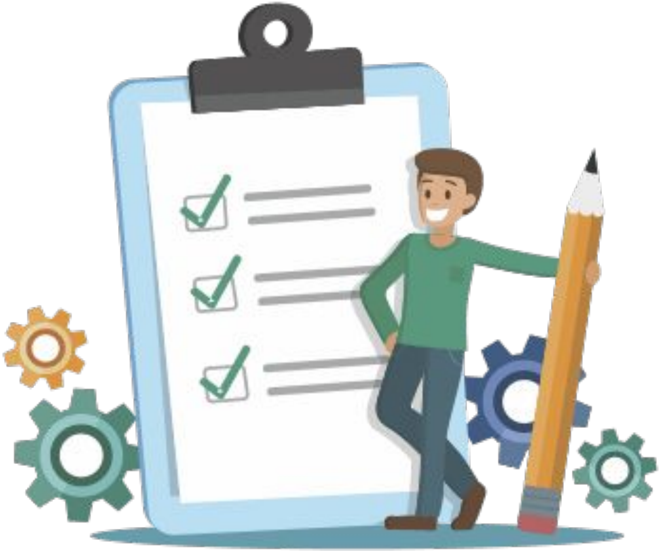
Hands-on

- **Exercise 5**
  - Stop all containers
  - Remove second last image
    - $ docker image rm third_edem_img
    - Any issue?
    - Remove all containers and test again
  - Remove all images except *fourth_edem_img*



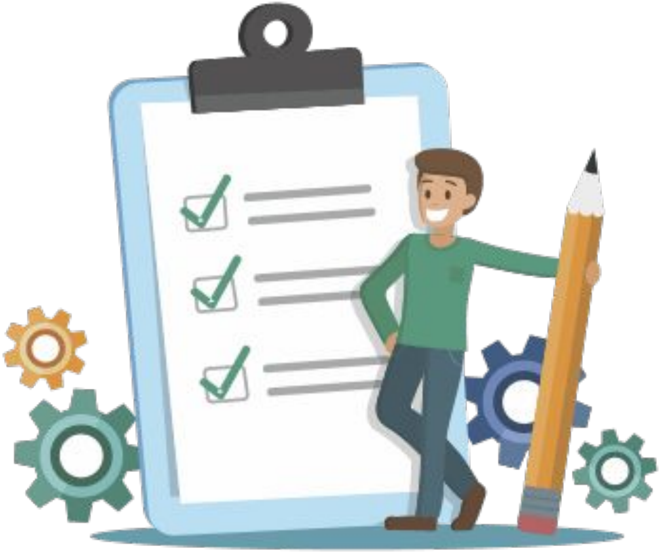
Hands-on

- **Exercise 6**
  - Pull wordpress image from Docker Hub https://hub.docker.com/_/wordpress
  - Run a container with Wordpress.
    - We want to access port 8080



Hands-on

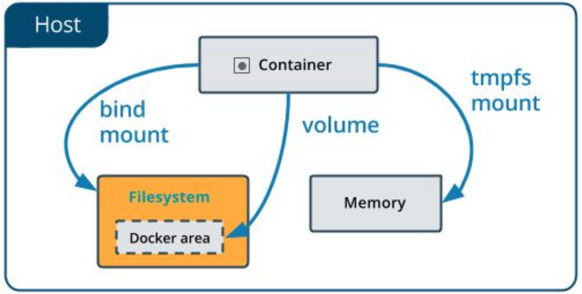
- **Exercise 7**
  - Convert web page container into image
    - Execute Container from Exercise 4
    - The name of the container should be something like "cont_exercise_7"
  - $ docker ps -a
  - Add new line in "/var/www/html/index.html"
    - $ docker exec -it cont_exercise_7 bash
    - $ cd /var/www/html
    - $ echo "<p>my_name</p>" >> ./index.html
    - $ cat index.html
  - $ docker commit cont_exercise_7 cont_fourth_edem_img_newline:latest
  - $ docker login --username=[Dockerid]
  - $ docker tag [image_id] [Dockerid]/myfirstimage:latest
  - $ docker push [Dockerid]/myfirstimage



Docker Volume

- What happens to the data if a container crash o removed?
    - Data could be lost!!!

- Docker has two options for containers to store files in the host machine:
    - Volumes
    - Bind mounts

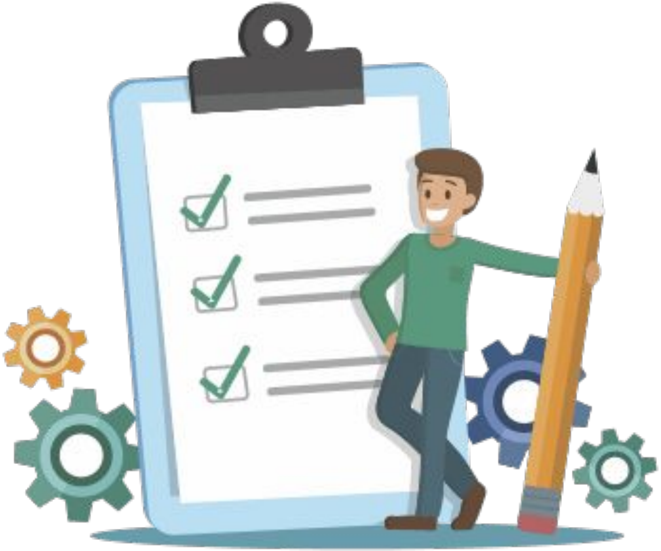

Host
Container
bind mount
volume
tmpfs mount
Filesystem
Docker area
Memory

- Volumes have the following advantages:
    - Easier to back up or migrate
    - Managed using Docker CLI
    - More safely shared among multiple containers
    - Isolated from the host file system

EDEM

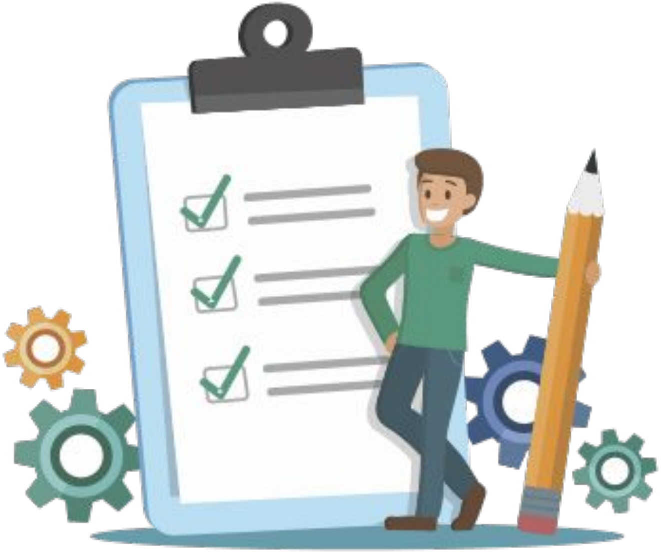
Hands-on

- **Exercise 8**
    - $ docker volume create my-vol
    - $ docker volume ls
    - $ docker volume inspect my-vol
    - $ docker run -d --name volume_test --mount source=my-vol,target=/app [DockerId]/myfirstimage
    - $ docker exec -it volume_test bash



Hands-on

- **Exercise 9**
  - Remove all containers
  - Remove all images except wordpress and exercise 4 image



EDEM

4

Docker Compose

Docker Compose

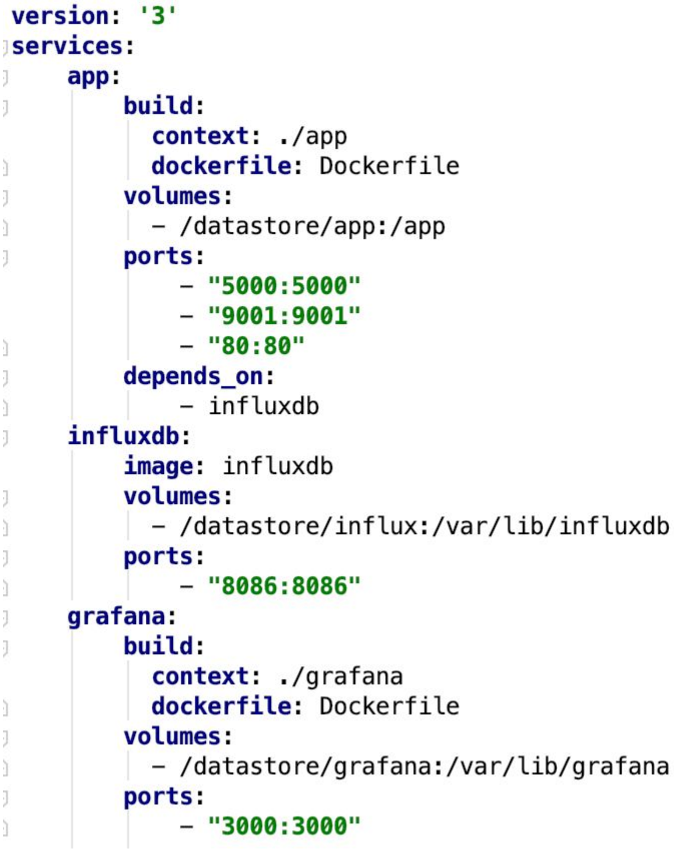
- Compose is a tool for defining and running complex applications with docker

- Without Docker Compose, multiple DockerFiles will be needed for a full or complex application
    - Separate files for front-end, back-end…

- With Docker Compose, you can define a multi-container application in a single file

- Usually the file is called “docker-compose.yml”

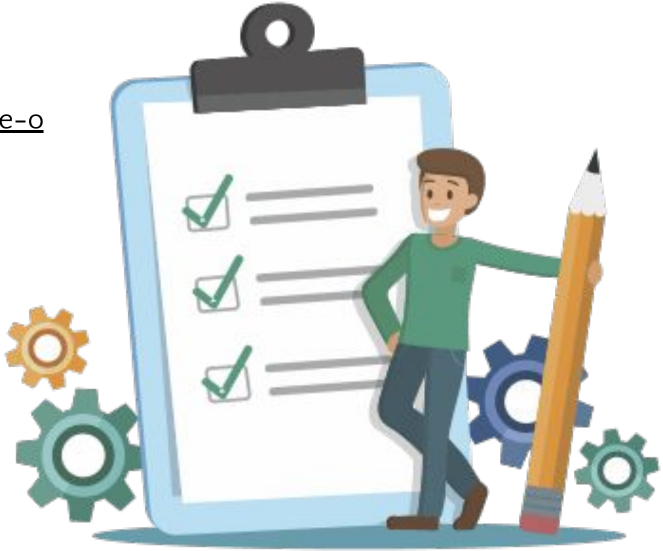
Docker Compose



```yaml
version: '3'
services:
    app:
        build:
          context: ./app
          dockerfile: Dockerfile
        volumes:
          - /datastore/app:/app
        ports:
            - "5000:5000"
            - "9001:9001"
            - "80:80"
        depends_on:
            - influxdb
    influxdb:
        image: influxdb
        volumes:
          - /datastore/influx:/var/lib/influxdb
        ports:
            - "8086:8086"
    grafana:
        build:
          context: ./grafana
          dockerfile: Dockerfile
        volumes:
          - /datastore/grafana:/var/lib/grafana
        ports:
            - "3000:3000"
```

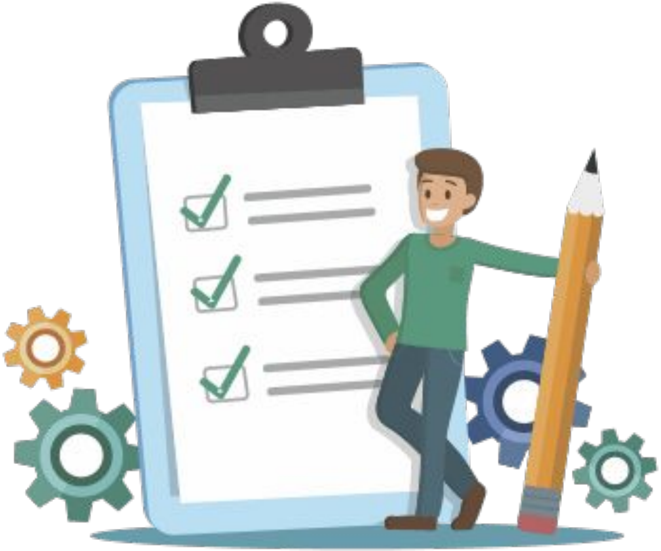
Hands-on

- **Install Docker compose in GCP VM**
    - https://cloud.google.com/community/tutorials/docker-compose-on-container-optimized-os



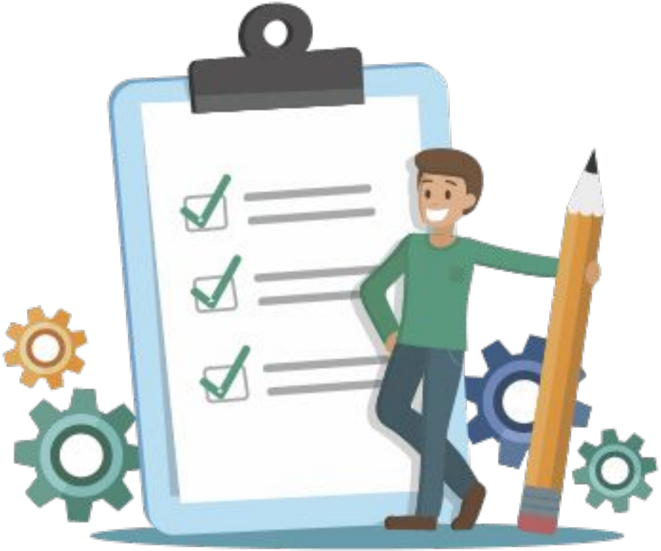
Hands-on

- **Exercise 10**
  - Create a docker compose file
    - Use this file:
    - https://docs.docker.com/compose/wordpress/
  - $ docker compose up -d



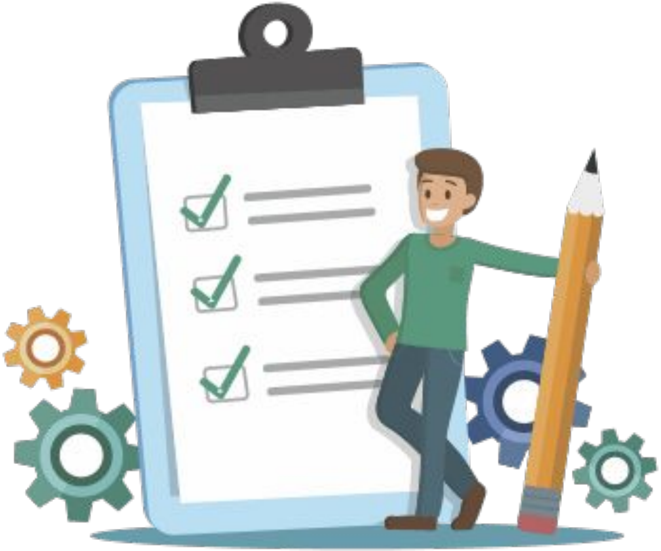
Hands-on

- **Exercise 11**
    - Add Ubuntu con Nginx from exercise 4 into the previous docker-compose file
    - Execute this docker-compose again with new changes



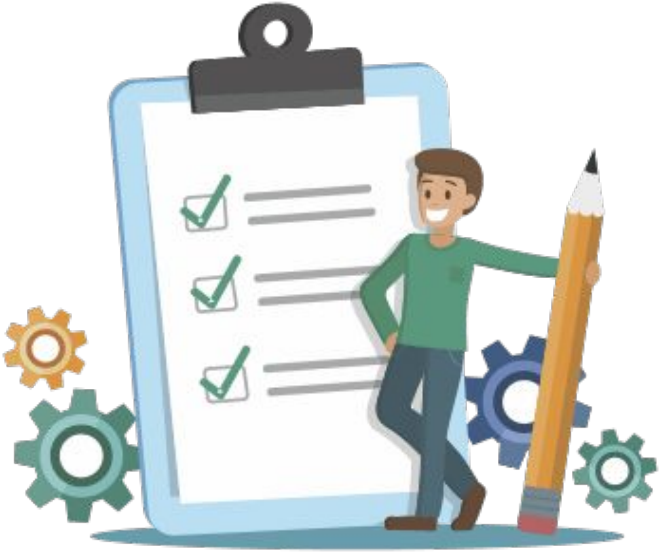
Hands-on

- **Exercise 12**
    - Remove all containers
    - Remove all images
    - Remove all volumes



Hands-on

- **Exercise 13**
  - Install Jupyter using Docker



5

Kubernetes

Kubernetes

- Large and small software companies deploying thousands of container instances daily
  - How can we manage this complexity?

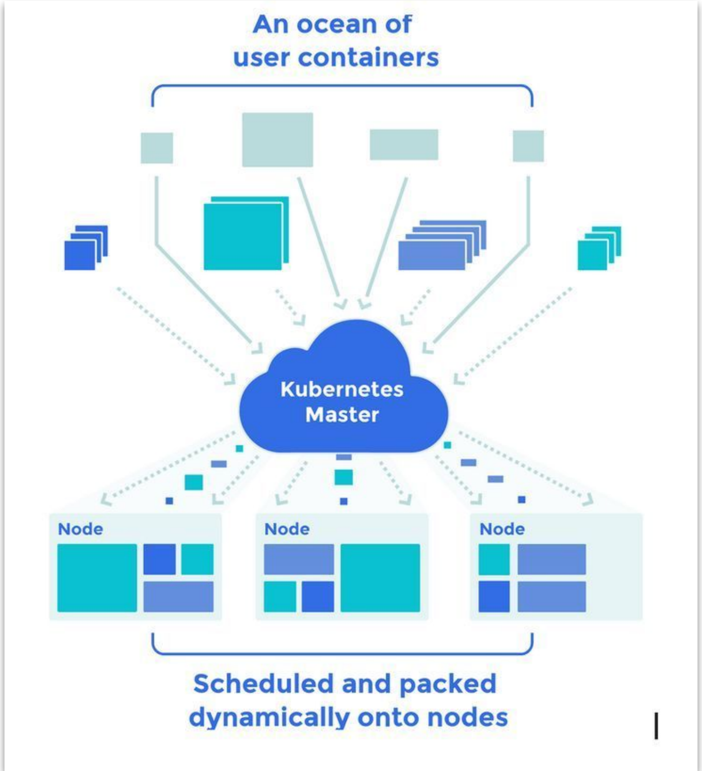
- Originally developed by Google.

- Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications

- Kubernetes makes it easy to deploy and operate applications in a microsevice architecture

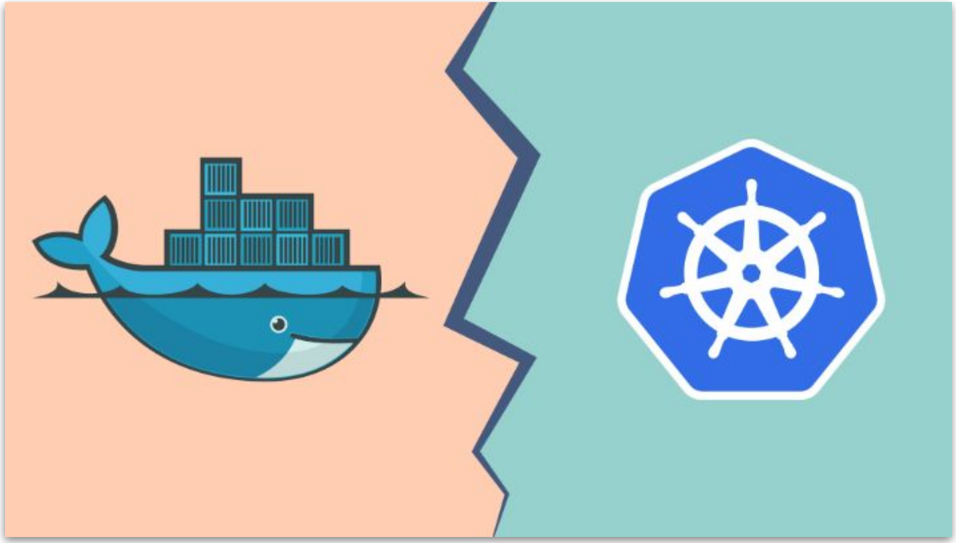
Kubernetes

- Features:
    - Controlling resource consumption by application or team
    - Evenly spreading application load across a host infrastructure
    - Automatically load balancing requests across the different instances of an application
    - Monitoring resource consumption and resource limits
    - Moving an application instance from one host to another
    - Automatically leveraging additional resources made available when a new host is added

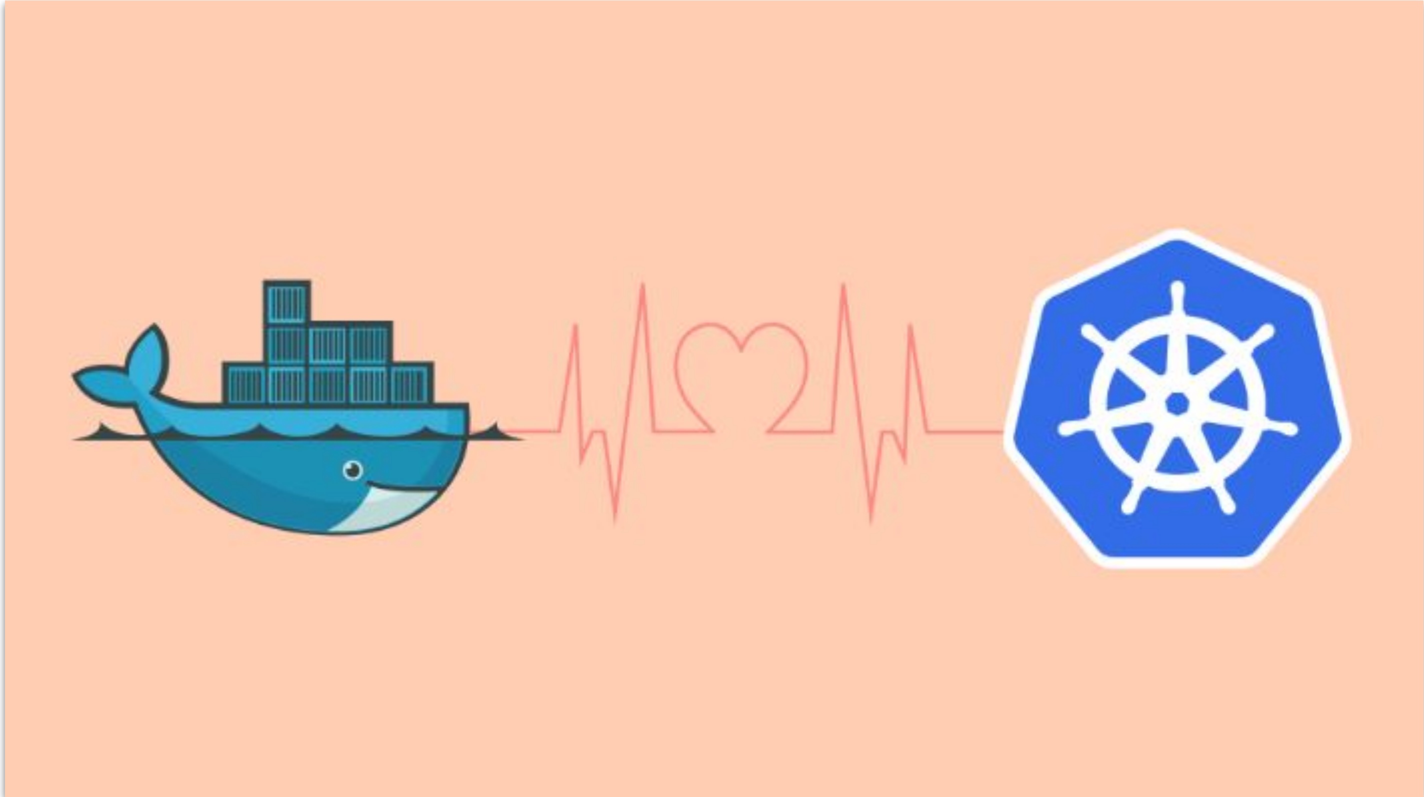

An ocean of user containers
Kubernetes Master
Node
Node
Node
Scheduled and packed dynamically onto nodes

Kubernetes - Docker

- Docker is used to isolate your application into containers
- Kubernetes, on the other hand, is a container scheduler. It's used to deploy and scale your application



Kubernetes - Docker



Thanks!

Any questions ?

You can find me at

- https://www.linkedin.com/in/miguelsotomayorf/
- @masfworld
- miguel.sotomayor@sidesna.es