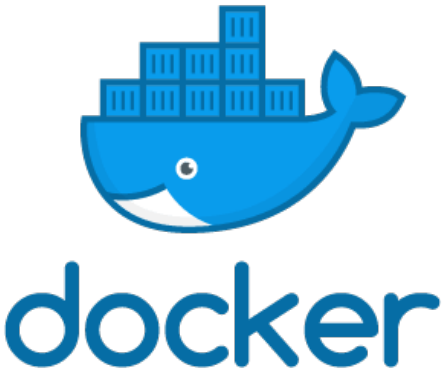
# CONTAINERS AND DOCKER EXPRESS COURSE




Presented by :

Kaulitz Guimarães - Software developer at IBM

# Agenda

- Part I
  - ( Concept content)
- Part II
  - (Practical part)

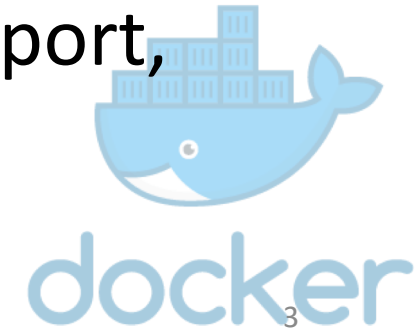
- Content :
  - Docker Context
  - Docker Infrastructure
  - Docker images
  - Docker containers
  - DockerHub

# Kaulitz Guimarães

- Student at Unicamp ( 8th semester) .

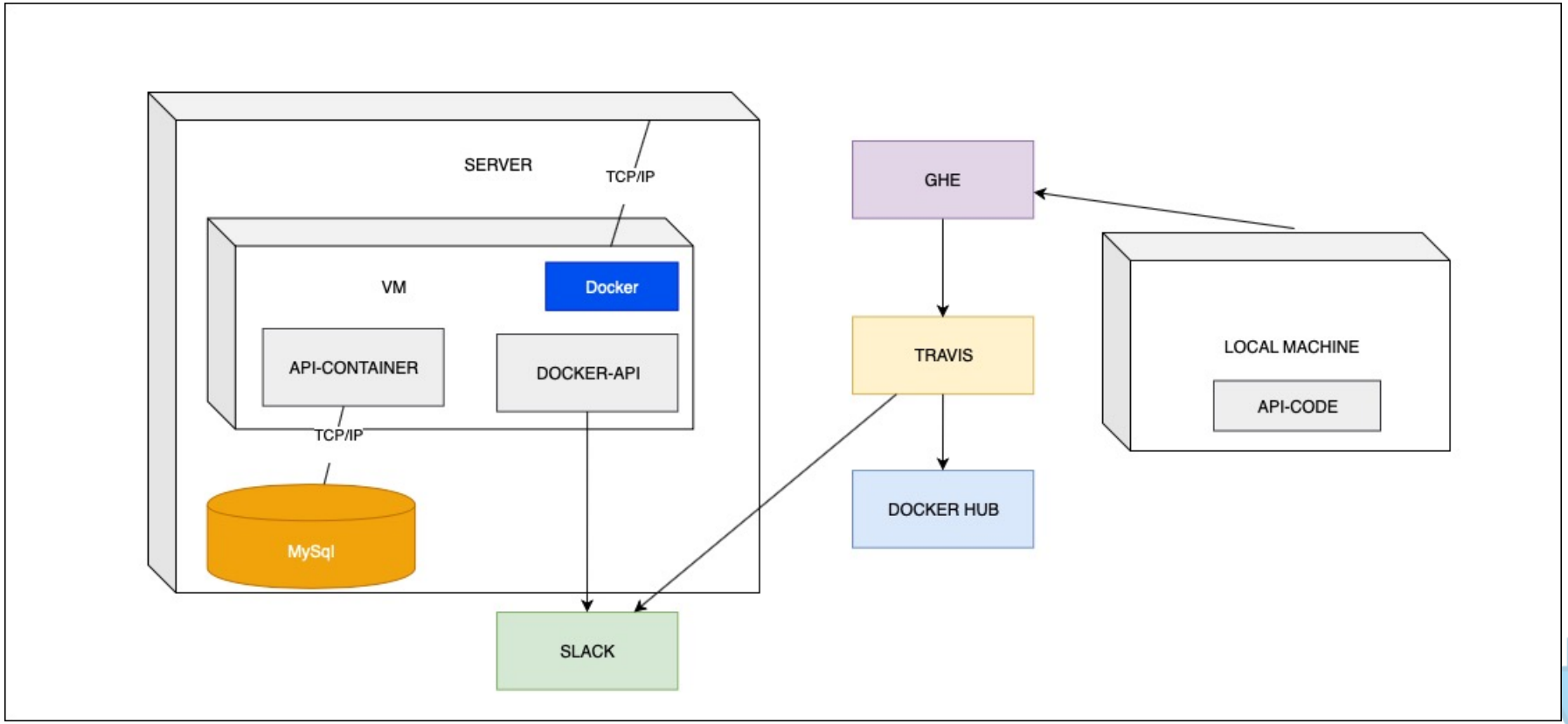
- Software Developer at IBM ( Internal Compliance  Services) - DevOps, automation, Kubernetes, Docker, API's and Dashboard).

- IBM parallel projects - AI-A-THON, Call for code, Passport, patents related to machine learning.

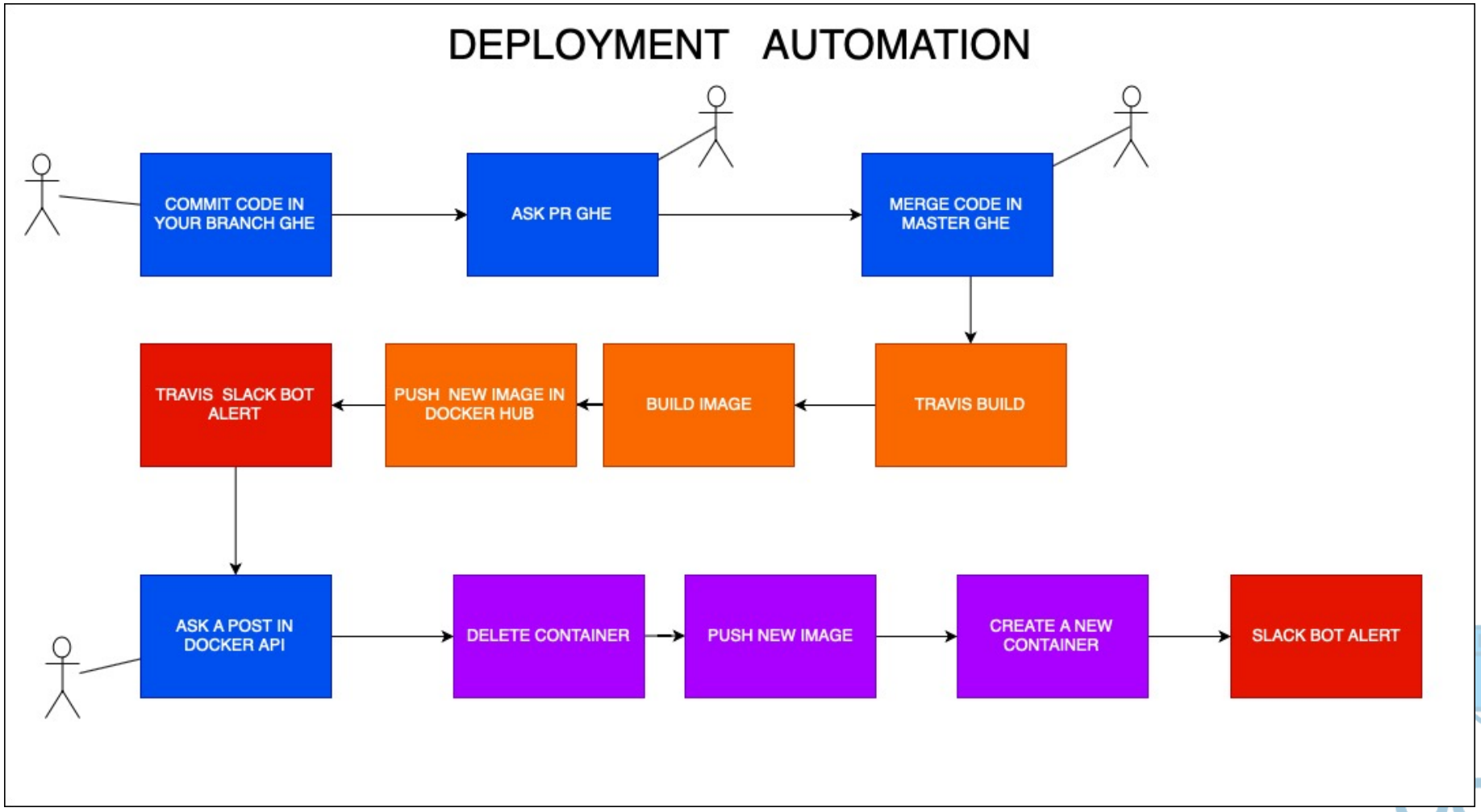


# Why I learned Docker ?

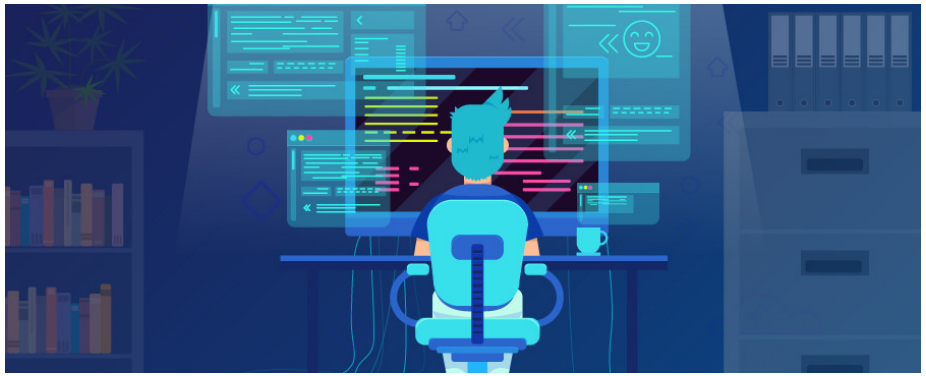
# Passport Project



# Passport Project



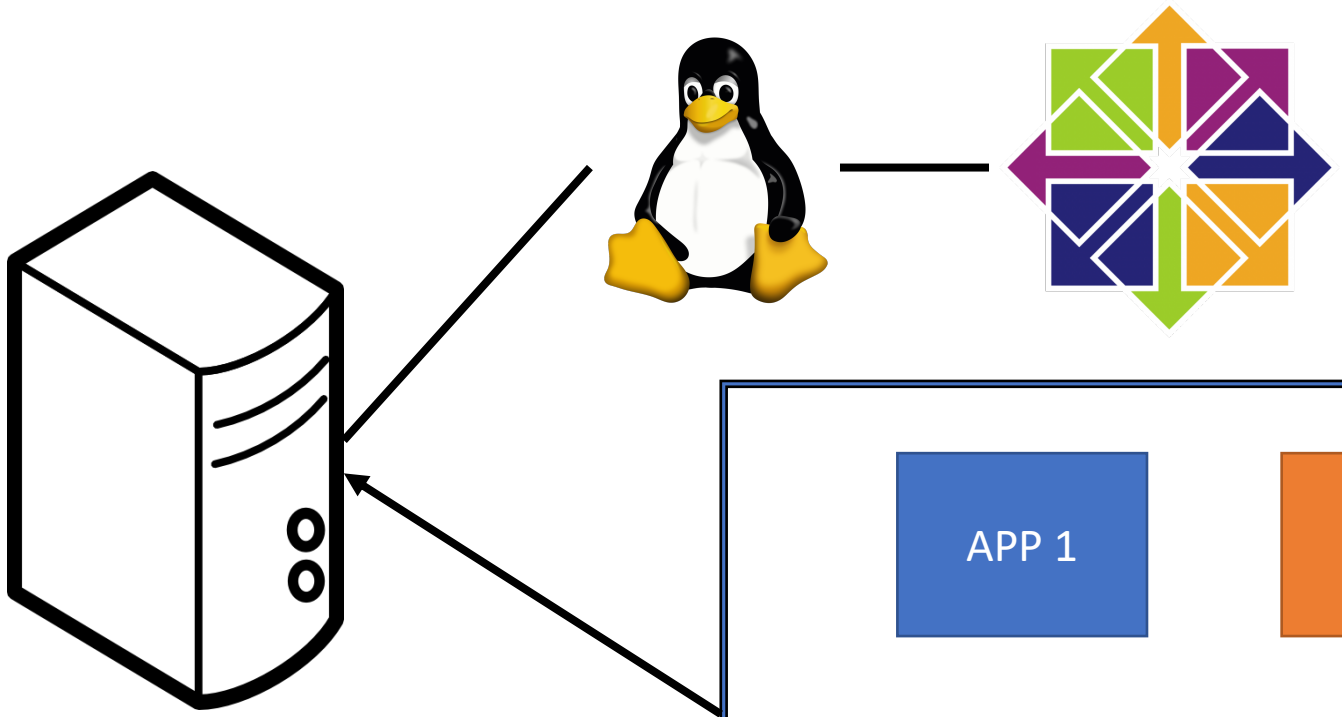
# A developer issue ….



?




# A developer issue ….




**Client requirements :**

- Python 3.5 (backend)
- Nginx ( frontend)
- MYSQL 5.6 ( database)

# A developer issue ….

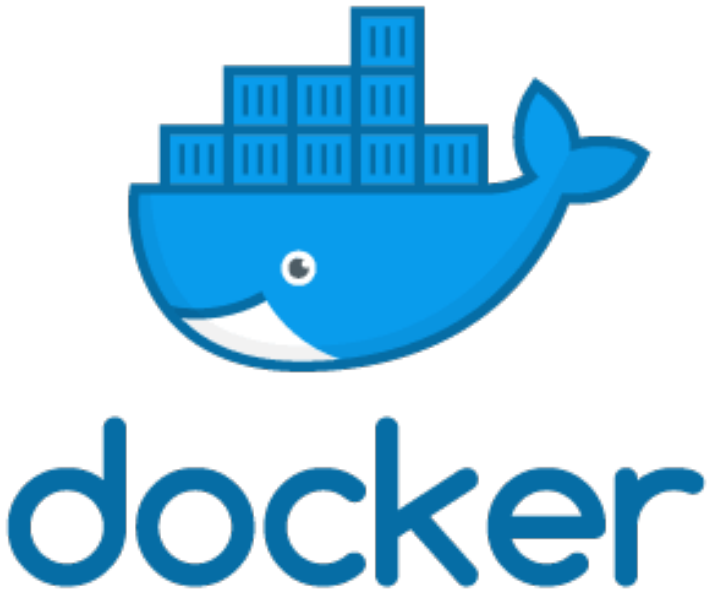
**ISSUES :**

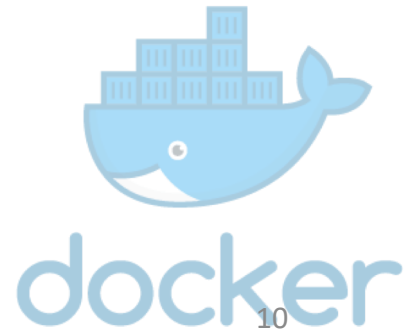
- Install python and MySQL
  - Possible Issue : You  need to use a different version of python that is installed in the server.
- Install NGINX
  - Possible Issue : It can have conflict between your OS and the NGINX version that you are using.

This will cost a lot of time to developer solve it

Or not …

# The developer solution ….



# What is Docker ?

- Historical Facts :
  - Docker start as a internal project in dotCloud company, as a **PaaS**.
  - It was released in 2013 as an **open source Tool**.

- Docker is a  container **management** software.
- The main idea of docker is for developers to **easily** deploy their applications without worry with common infrastructure issues, **as O.S version**, current **language version**  and the **infrastructure dependencies** itself .

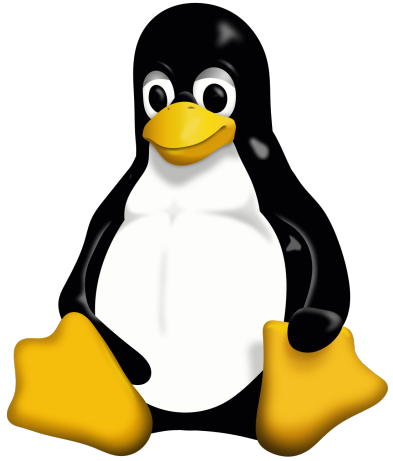
- Docker documentation : https://docs.docker.com

# VMs vs Containers

- **Virtual machines** : share same **hardware**, but they have they own kernel and is mounted by a default OS image (ISO), they are heavy and the need a lot of hardware usage.

- **Containers** : share the same **kernel**, they have an OS as a base, but only with the necessary resources, which means they are lighter and scalable.

# OS’s  that support Docker

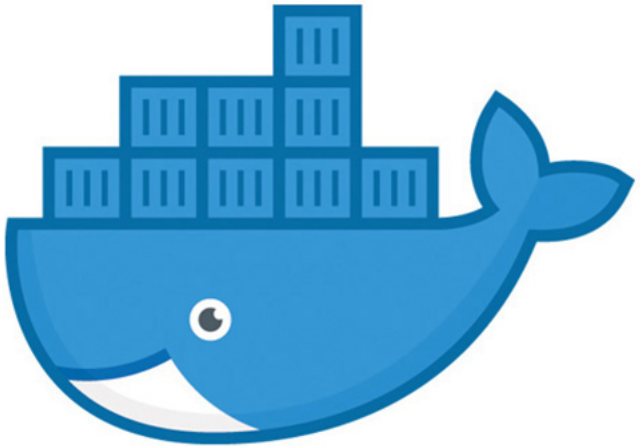


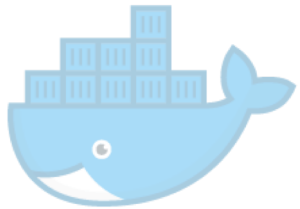
# Where Can We Download Docker ?

- Mac and Windows :
  - https://www.docker.com/products/docker-desktop
- Linux Distribution :
  - https://runnable.com/docker/install-docker-on-linux

# Docker Container

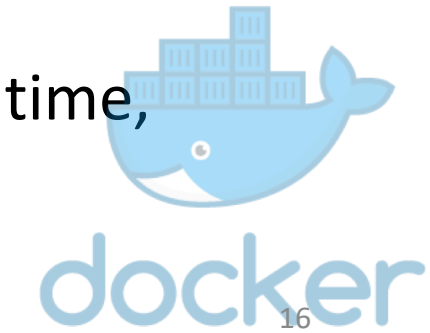
- It is a “pack” with code and its dependencies runs quickly and reliably from one computing environment to another.
- Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.



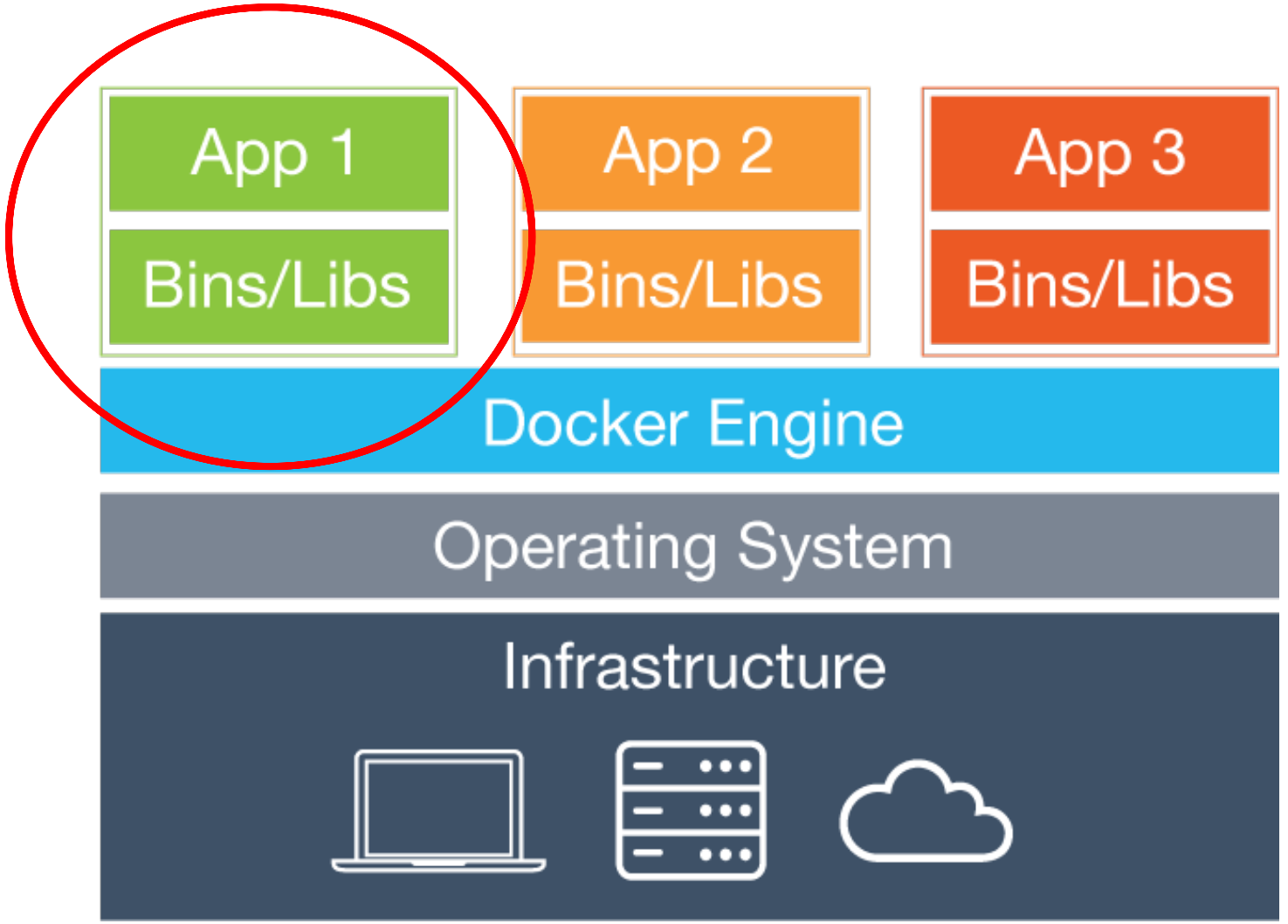


# Docker advantages

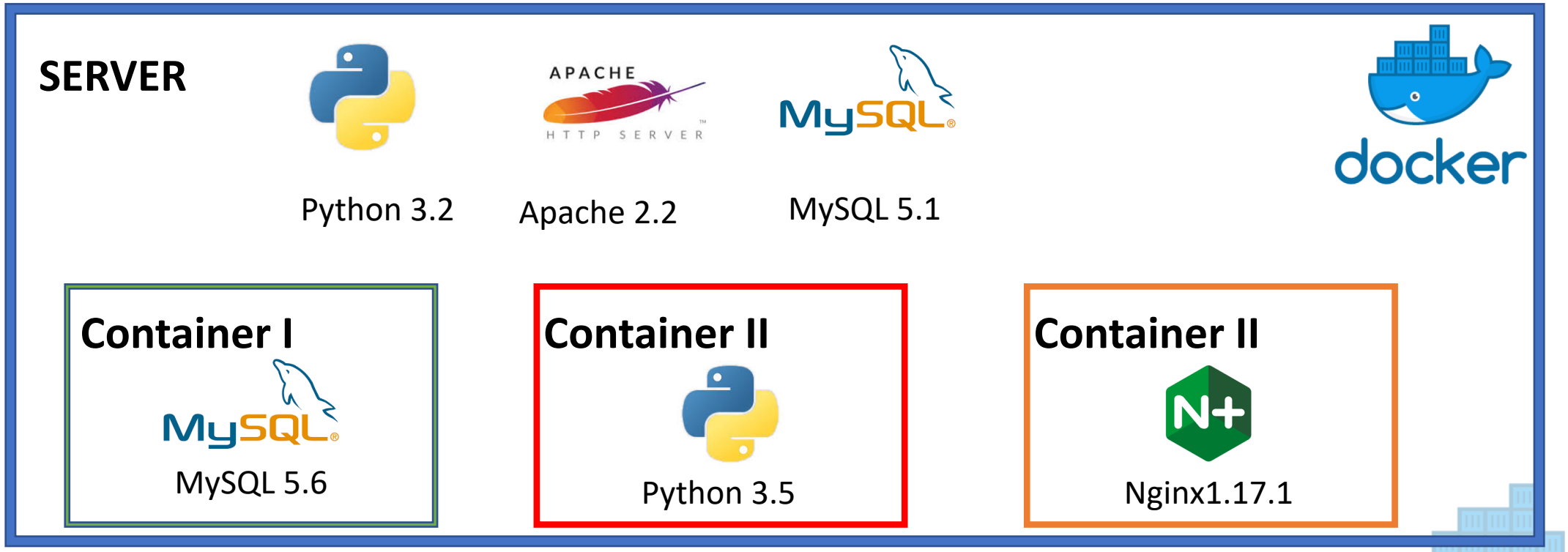
- Target Users : Developers
- Docker **reduces time** configuring infrastructure as servers and VM's to support the **application.**
- Docker **prevents** that the server **stays "dirty"** due application **dependencies**.
- You can deploy Docker containers **anywhere**, on any **physical** and **virtual** machines and even on the **cloud**.
- Since Docker containers are pretty **lightweight**, they are very easily **scalable**.
- Docker containers are made to be **destroyed** and **created** all the time, otherwise servers and VM's.



# Docker Infrastructure

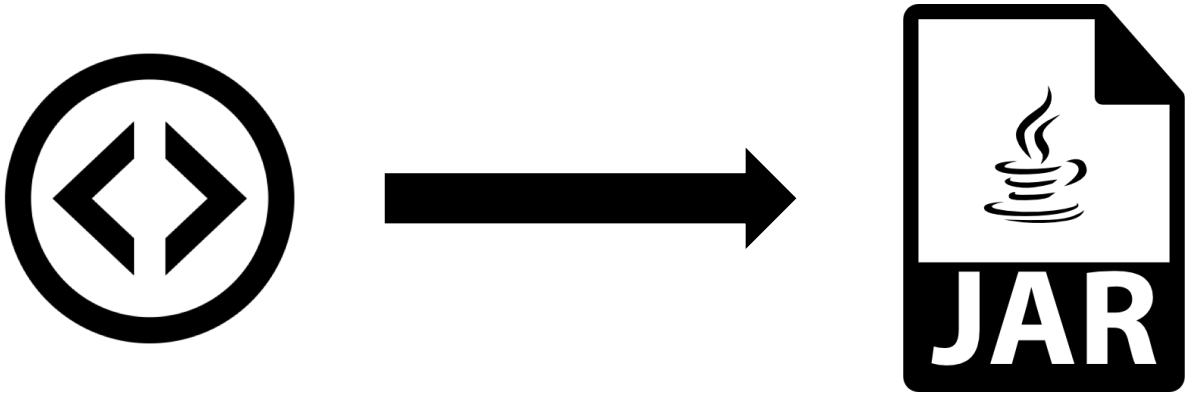


# Docker Container



# How a container is created ?


## DOCKER IMAGE!!!!!

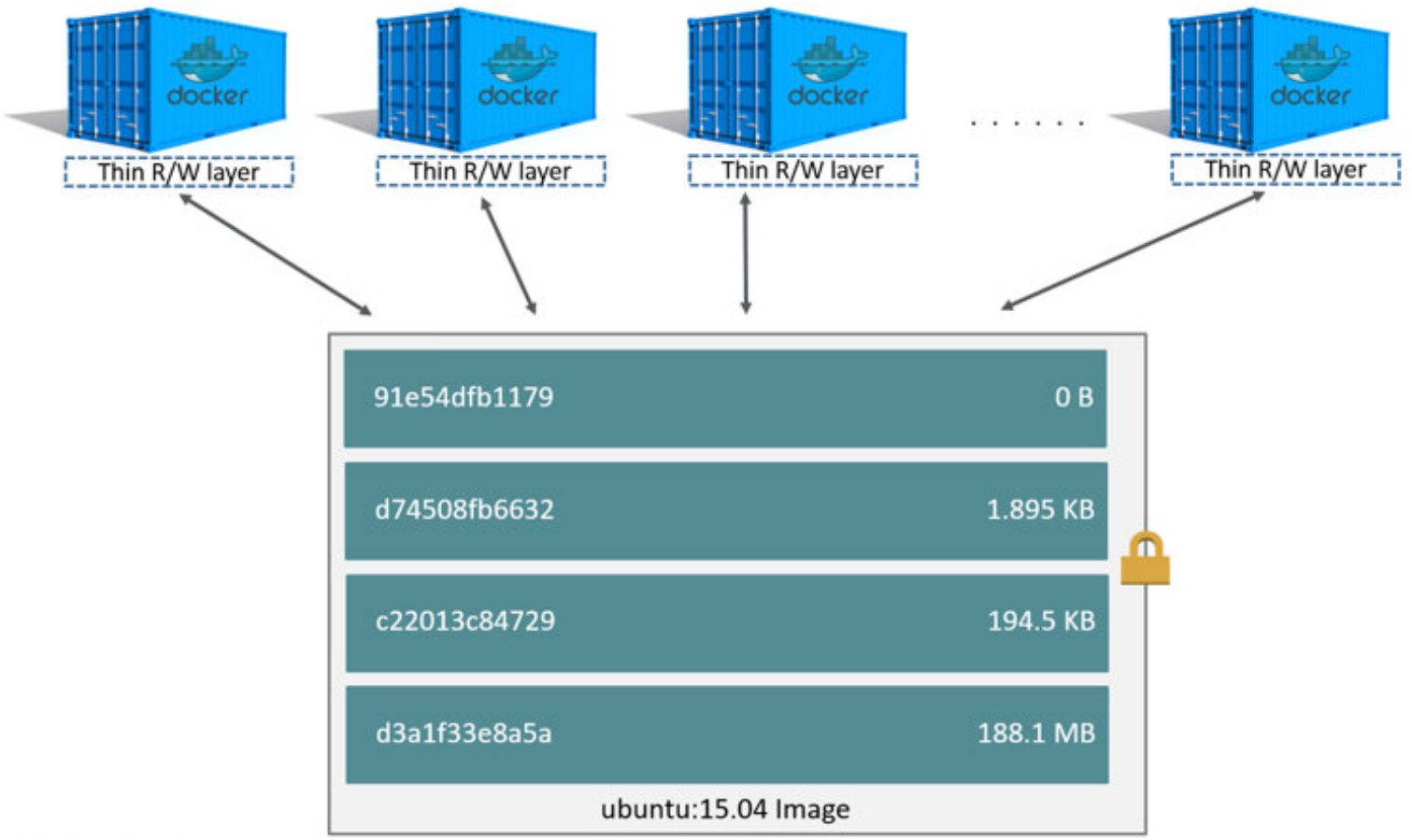


# Docker Image

- Docker images are made by commands with a really simple syntax that will be use to build the containers, like a class that becomes an objects.

- Each command in the image creates layer/digest. This way docker minimize processing cost when we update them.

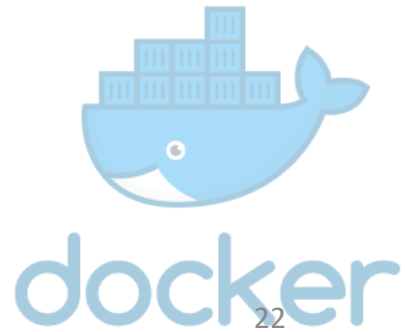
# Docker Image



# Dockerfile

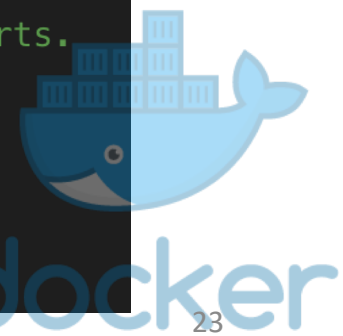
- Dockerfiles contains the commands to build an image.

- By pattern the first command is always **FROM (image name)**. This command tells to docker that this image will be build above the one that you specified. You need at least an OS ( a Linux distribution)  as a base image.

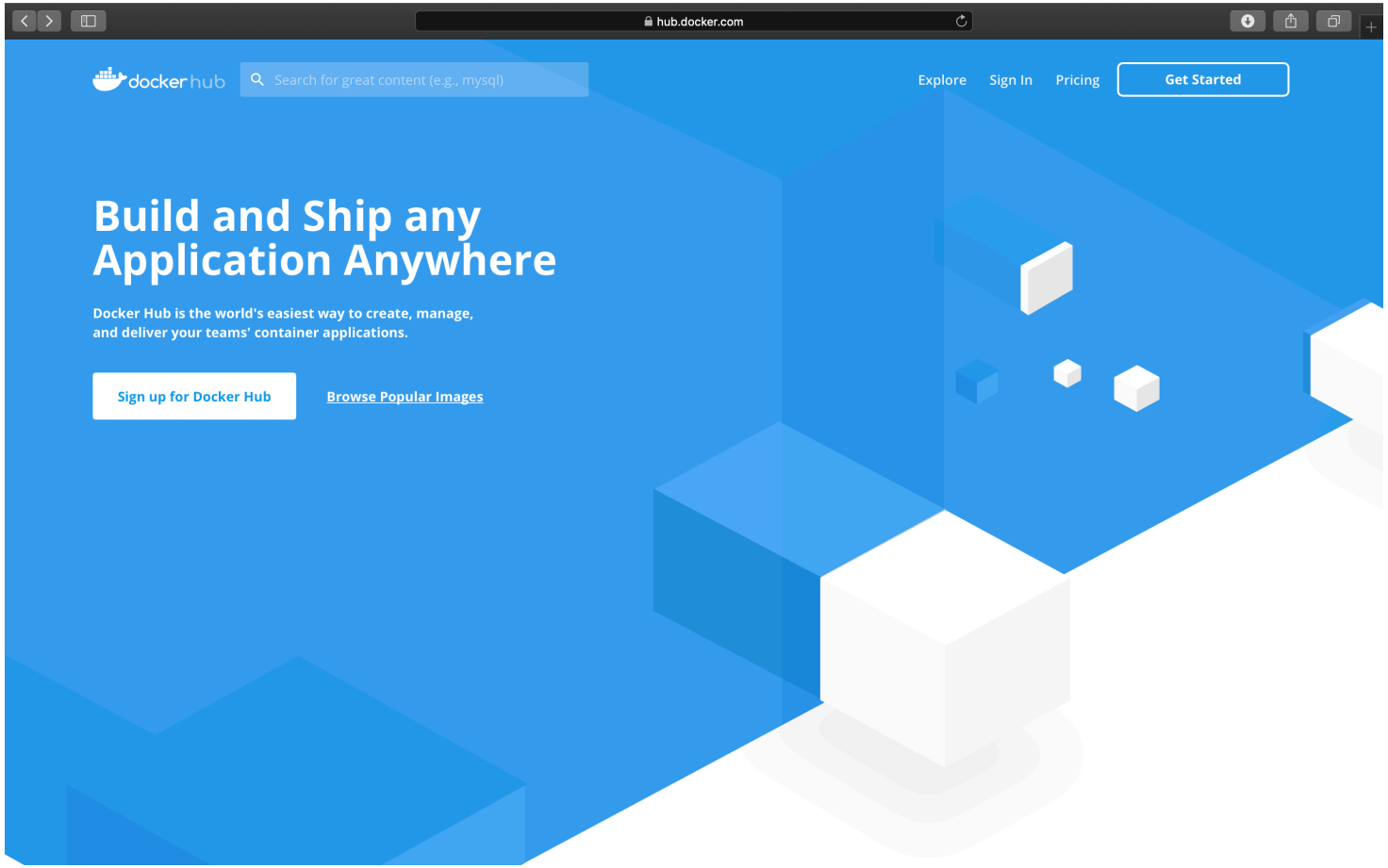


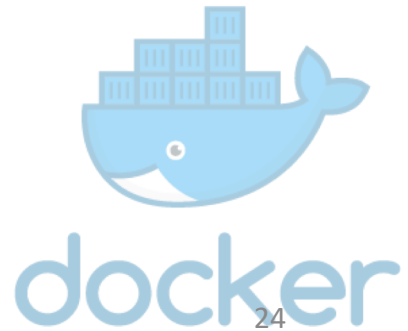
# Dockerfile

```
# OS as base image
FROM alpine:3.4
# WORKDIR : Switch to directory specified
WORKDIR files
# RUN : Run linux commands  (Same as we run in linux bash)
RUN apk update
RUN apk add python
RUN apk add vim
RUN apk add git
RUN mkdir files
RUN touch ola-mundo.py
RUN echo "print 'ola mundo'" > ola-mundo.py
# COPY : Copy files to current directory
COPY hello-world.py .
# The ENTRYPOINT specifies a command that will always be executed when the container starts.
# The CMD specifies arguments that will be fed to the ENTRYPOINT.
ENTRYPOINT ["python"]
CMD ["hello-world.py"]
```



# DockerHub

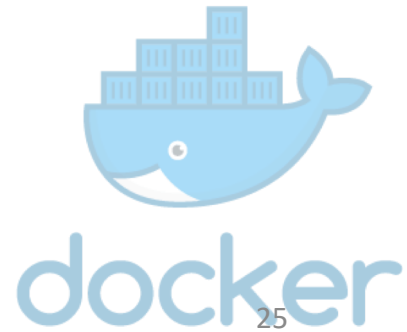

hub.docker.com
dockerhub
Search for great content (e.g., mysql)
Explore
Sign In
Pricing
Get Started
Build and Ship any Application Anywhere
Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications.
Sign up for Docker Hub
Browse Popular Images


docker

# DockerHub

- DockerHub is like a GitHub for containers.
- There are official images that you can use directly or as a image base in your Dockerfile.
- You can also upload your own image (public and one private as free).

https://hub.docker.com



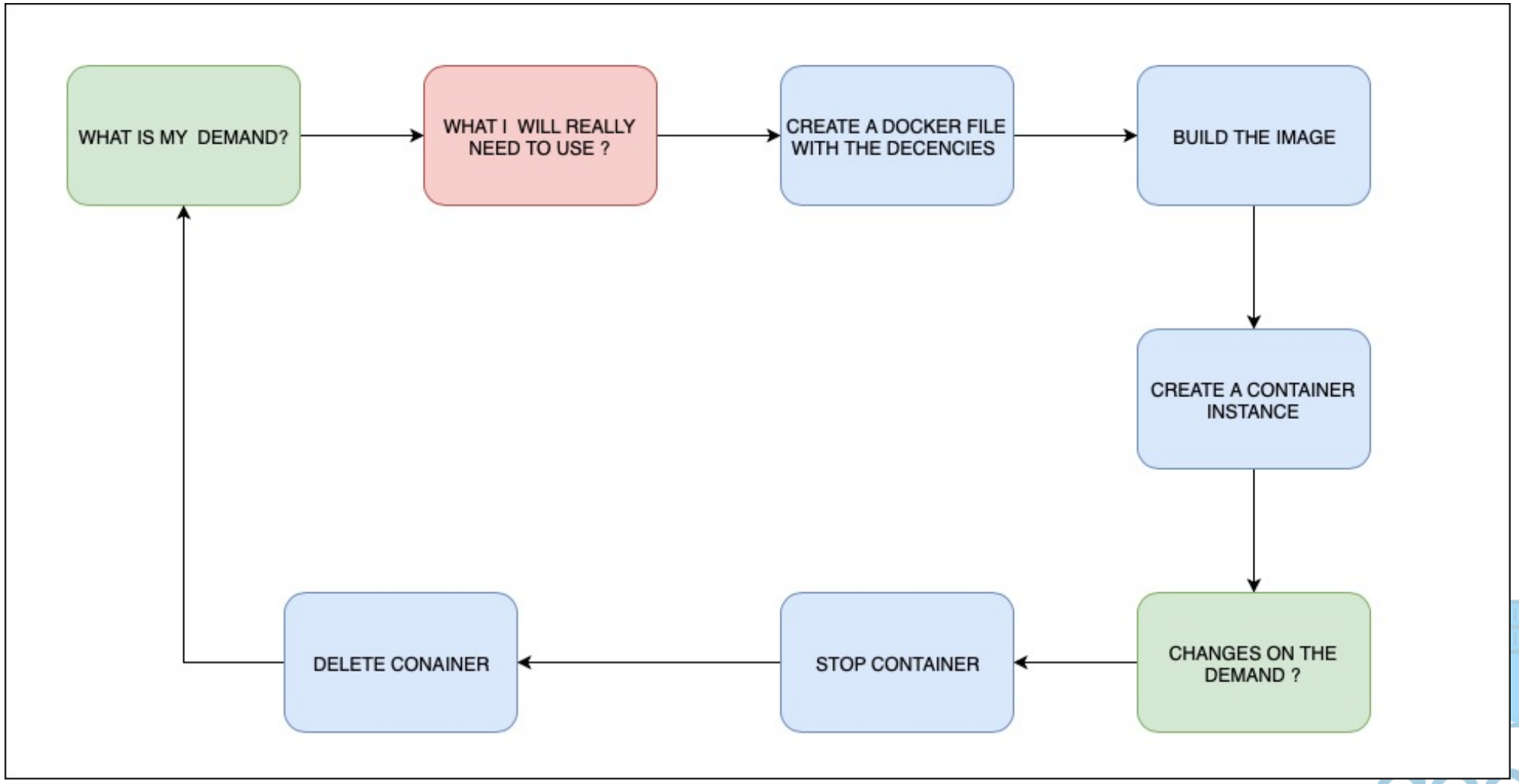
# Pause

See you in 10 min ☺

# Docker in practice

- Create an account  in https://hub.docker.com
- Access this link : https://labs.play-with-docker.com
- Add an instance.

# Docker in practice



# Docker in practice

- How to pull an image with docker command :
  - docker pull (image-name)
- How create a container by image name:
  - docker run --name (container-name) -it [-d] [-p] [portFromContainer:portToExpose] (image-name) --env  [ENV variable]
- How to build an image :
  - docker build -t (image_name) .
- How to list containers/images
  - docker (image/container) ls
- How to stop/start a container
  - docker start/stop (container-name or container-id)
- How to delete a container/image
  - docker (image/container) rm (container-name/image-name)

# Exercise I

**A)**

**1.** Test docker creating an instance of hello-world image (not detached)

**2.** Delete container.

**B)**

**1.** Test docker creating an instance of hello-world image (detached)

**2.** Delete container.

**A)**

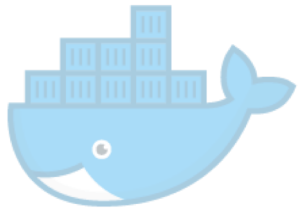
**1.** docker pull hello-world / docker run -it --name my_first_container hello-world

**2.** docker stop my_first_container / docker container rm my_first_container

**B)**

**1.** docker pull hello-word / docker run -it --name my_first_container hello-word

**2 .** docker stop my_first_container / docker container rm my_first_container



# Exercise II

A) Create a Dockerfile based on alpine OS, run the command “uname”.

B) Create a Dockerfile based on ubuntu OS, run the command “uname” .

A)
touch Dockerfile

vi Dockerfile

```
FROM alpine:latest
CMD [ "uname" ]
```

B)
touch Dockerfile

vi Dockerfile

```
FROM ubuntu:latest
CMD [ "uname" ]
```

# Exercise II

C) Create an image based on  alpine  docker file.

D) Create an image based on  ubuntu  docker file.

C) Docker build -t my_alpine .

D) Docker build -t my_ubuntu .

# Exercise II

E) Create an alpine container (not detached).

F) Create an ubuntu container (detached).

E) Docker run -it --name alpine_container  my_alpine

F) Docker run -it  -d --name ubuntu_container my_ ubuntu

# Docker in practice

- How to get in a container :
  - docker exec -it container-name or container-id) (command  e.g., bash)
- How to get a container log :
  - docker logs stop (container-name or container-id)

# Exercise II

G) Catch ubuntu container log

H) Get into alpine container then run the  command “cat /etc/os-release”

G) docker logs ubuntu_container

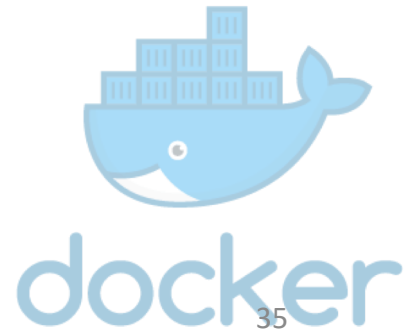
H)
Way I :

docker  exec -it alpine_container bash

-> cat /etc/os-release

Way II :

docker  exec -it alpine_container cat /etc/os-release



# Exercise III

A) Choose one Dockerfile from exercise before, then add the uname command as an entry point. Then add a command with the flag “-r”.

B) Rebuild the image using same command as the exercise before.

A)

```
FROM ubuntu:latest
ENTRYPOINT [ "uname" ]
CMD [ "-r" ]
```

B)

```
Docker build -t my_ubuntu .
```

# Exercise III

C) Delete the old container from exercise II.

D) Create a new container with same name as exercise II ( detached).

C)
```
docker stop ubuntu_container
docker  container rm ubuntu_container
```

D)
```
docker run -it  -d  --name ubuntu_container my_ubuntu
```

# Exercise III

E) Catch container logs

F) Delete the container

E)
docker logs ubuntu_container

F)
docker stop ubuntu_container
docker  container rm ubuntu_container

# Exercise III

G) Create another container with same name as before, but overriding the “-f” default flag  by using the “-a” one (detached).

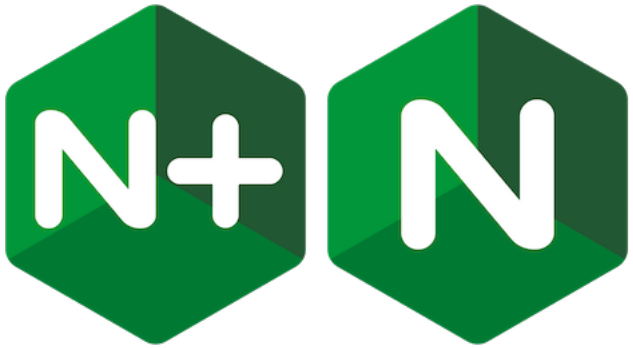
H) Catch ubuntu container log

G)
docker run -it -d --name ubuntu_container -a

H)
Docker logs ubuntu_container

# Front-End with NGINX , HTML5, CSS3 and Js

Now we'll deploy a web page, developed in HTML5, CSS3a and Js that shows the current Dollar price based on Real.
We will use Docker container along with NGINX to do that



NGINX is a HTTP server, like APACHE.
We'll use it to deploy a HTML webpage.

# Exercise IV

**A)** Create a NGINX container

**A)**

docker pull nginx:1.17 / docker run -it -d --name ngnixContainer -p 8080:80 nginx:1.17

# Exercise V

A)
Run this command on your bash :
git clone https://github.com/KaulitzGuimaraes/DockerCourse

B) Go to the Dollar Price folder by running the command

cd DockerCourse/PraticalExercises/DollarPrice

C)

Create a Dockerfile, then :

- Create a new direct called **dollarprice**
- Copy the current code into the folder
- Switch to this directory
- Move the current content to this path **/usr/share/nginx/html**
- Expose port 80

C)

```
FROM nginx:1.17
VOLUME dollarprice
COPY . / dollarprice
WORKDIR dollarprice
COPY . /usr/share/nginx/html
EXPOSE 80
```

# Exercise V

D)

Build the image with the name dollarprice

E)

Create a container with the name dollarprice_container and port 8080

D)
docker build -t dollarprice .

E)
docker run -it -d --name dollarprice_container -p 8080:80 dollarprice

# Exercise V

F)

Delete the container

G)

Update the application code by running the command

git pull

H) Rebuild the image

I) Create another container with same name and port

F)
docker stop dollarprice
docker container rm dollarprice
H)
docker build -t dollarprice .
I)
docker run -it -d --name dollarprice_container -p 8080:80 dollarprice

# Exercise VI

A)

Download the code from GitHub link in your machine. Unzip it and open in a editor, you can use a local one or download sublime.

GitHub : https://github.com/KaulitzGuimaraes/DockerCourse

Sublime (Windows only) : https://www.sublimetext.com/3

B) Now change the application appearance , you can change the HTML and the CSS. In case you feel comfortable to change the Js code, you can also switch the coins conversion.

API documentation : https://exchangeratesapi.io

# Exercise V

C)

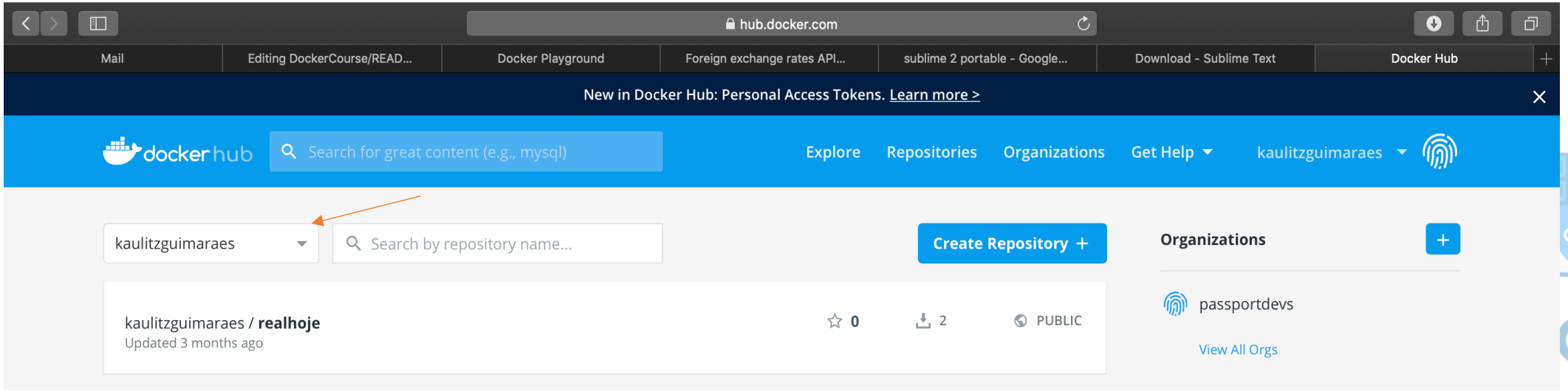
Copy the content for each file that you have changed in the docker player .

D)

Now open your dockerhub account accessing the link below :

https://hub.docker.com

Check your name profile :



# Docker in practice

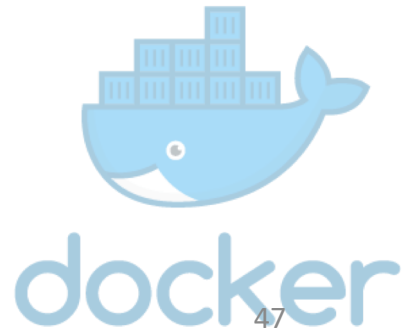
- Login with docker account

docker login

- Upload an image at docker hub

docker push (imagename)

P.S. : Your image name MUST follow this pattern : “your_usrname”/image_name



# Exercise VI

C)

Build the a new image with the same Dockerfile, but using “your_user”/dollarprice as the image name

D)

Login with your DockerHub login

E)

Push your image to DockerHub repository

F) Reload your DockerHub page , check if your image is there.

C)
docker build -t “your_user”/dollarprice .
D)
docker login
E)
docker push “your_user”/dollarprice

# Exercise VI

F)

Add a new instance in your docker playground .

G)

Check the images available.

H)

Push your image from DockerHub to your local machine.

I) Create a container with your username and port 80.

G)
docker image ls
H)
docker pull “your_user”/dollarprice
I)
docker run -it -d --name “your_user” -p 80:80 “your_user”/dollarprice

# MYSQL in Container

Instead of waist time installing databases locally, you can just create a container with a MySQL image by one command, then you can delete whenever you want just removing the container.


# Exercise VII

A) Pull the MySQL version 5.6 image

B) Create a container called mysql52 with the following parameters :
   - "MYSQL_ROOT_PASSWORD=123456"

C) Get in the container called mysql-server by using the follow command :
   - mysql -p

D) Run the command , inside of the container :
   - show databases;

A) docker pull mysql:5.6

B) docker run -it -d --name mysql-server --env "MYSQL_ROOT_PASSWORD=123456” -p 3306:3306 mysql:5.6

C) docker exec -it mysql-server mysql -p

D) -> show databases;

# MYSQL in Container

What if I have a local database and I want to switch to a container ?

# Exercise VIII

A) Go the the mysql_ex folder.

B) Create a new Dockerfile based on the latest image from mysql, and then add a command to copy the “d.sql” file to the image current directory.

C) Build the image with the name mysql-server and create a new container with name mysql_container .

B)

```
FROM mysql:latest
COPY d.sql .
```

C) docker build -t mysql_content .

docker run -it -d --name mysql-server --env "MYSQL_ROOT_PASSWORD=123456” -p 3306:3306 mysql_content

# Exercise VIII

D) Get into the container bash, then run the following command :

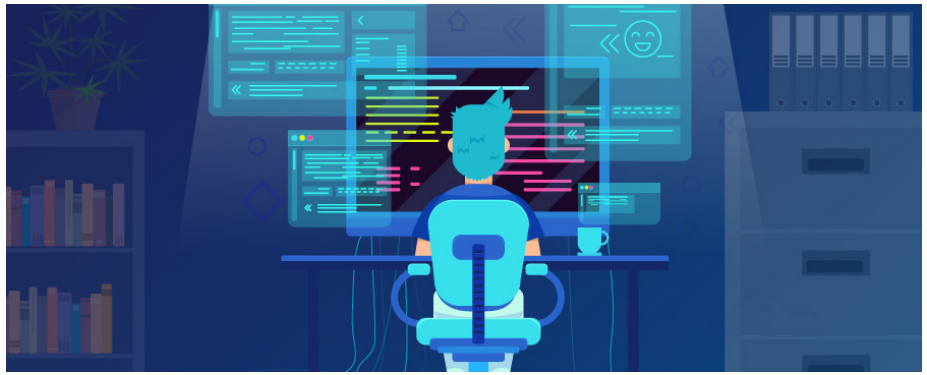
- mysql -u root -p < d.sql

E) Execute the command “show databases;”

D)
docker exec -it mysql_content bash
-> mysql -u root -p < d.sql

E)
-> mysql -p
-> show databases;

# A full Application with Docker



?




# Some tips

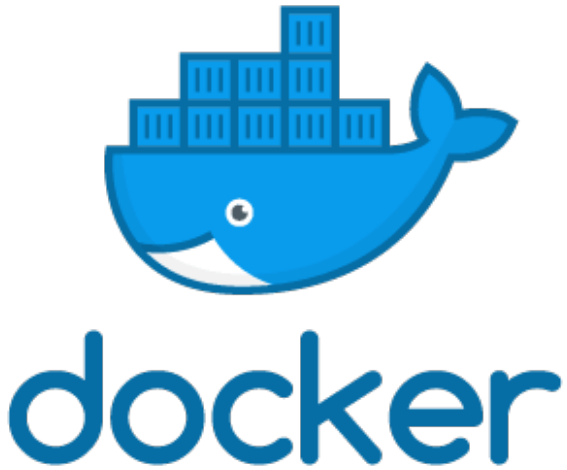
- Docker Essentials (cognitiveclass.ai) :
  - https://cognitiveclass.ai/courses/docker-essentials

- Technologies that you MUST have to know :
  - API
  - Github
  - React/Angulat
  - Docker/Kubernetes

# Conclusion

- What is a container
- Why use  Docker
- How do I containerize my applications

# THANK YOU !!!!!




E-mail :

kaulitzguimaraes@hotmail.com

GitHub :

KaulitzGuimaraes