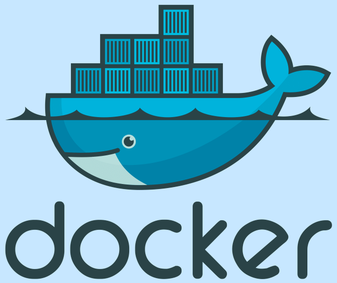


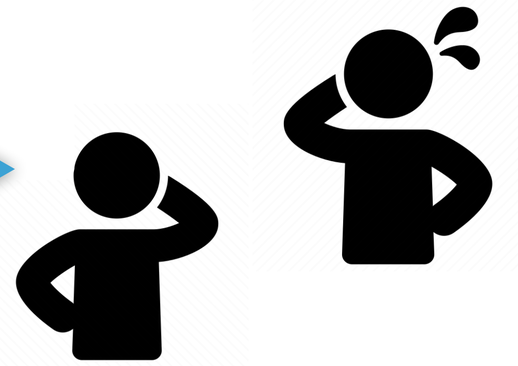
Docker : Introduction and basics

Amine FERDJAOUI
amine.ferdjaoui@sogeti.com

Année universitaire 2024/2025

Why Docker ?

The application works fine on my computer but not on another ?
The application is not rendered the same in your computer and mine ?



- **Some reasons :**
  - Missing/Mismatching libraries
  - Mismatching programming langage versions

Working locally

When you start a project locally you should always work with **venv (or conda)**.

To initialise a virtual environment :

```
$ python -m venv venv
```

To start it :

Unix (mac or linux)

```
$ source venv/bin/activate
```

On windows

```
$ .\venv\Scripts\activate
```

To exit :

```
(venv) $ deactivate
```

Why Docker ?

When you start a project locally you should always work with **venv (or conda)**.

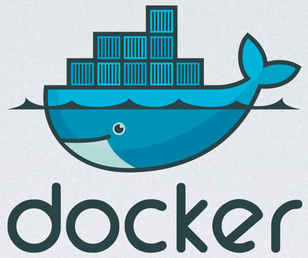
But in the cloud how can you manage and push your solution(s) ?

Application

Python 3.8
TF 2.0.1
Sklearn 1.1
Coclust 0.2

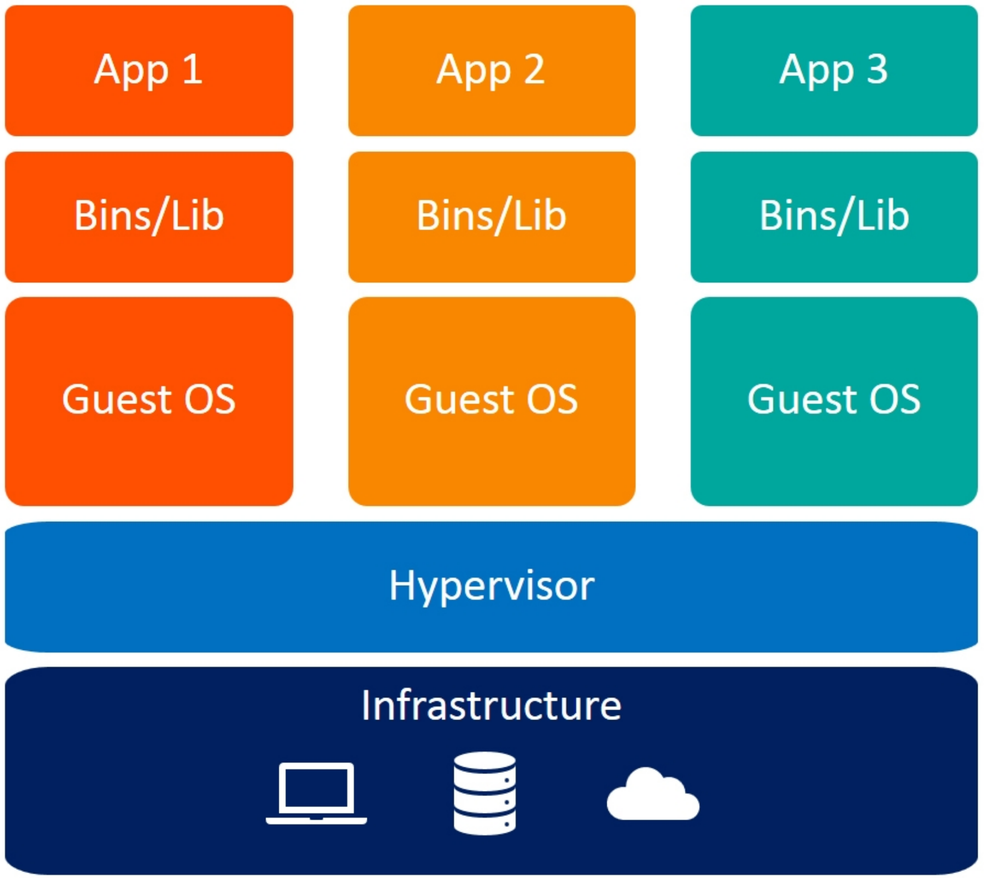
Application 2

Python 3.6
TF 1.5
Sklearn 0.8
Coclust 0.2

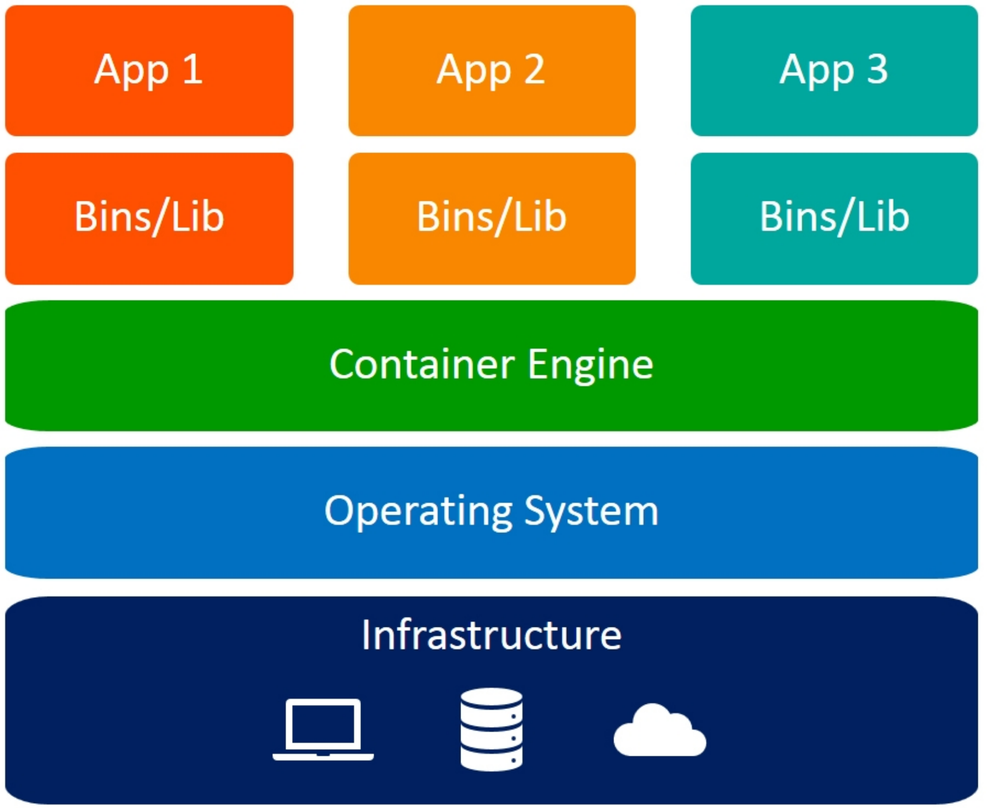


- Docker is a platform for Building, running and shipping apps.
- It is the easiest solution to deploy applications (ML models) to production.
- Each application runs in an isolated environment (no need for any installation).
- We can remove all dependencies with one click.

Docker vs Virtual machines



Virtual Machines



(or docker)

Containers

Docker vs Virtual machines

VMs

Need a separated OS

Very slow

Need lot of ressources

Docker containers

Use host's OS

Light and fast

Can run multiple isolated applications

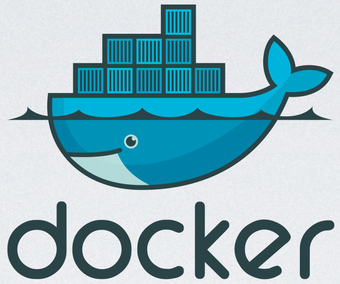
Don't consume lot of memory

Deployment and testing (once it works you can run it anywhere)

Installation

- Go to docs.docker.com/get-docker and create an account and ***download from Docker Hub.***
- Make sure the docker logo appears in status bar each time you want to use docker.
- Once docker installed and launched, test it using :

```
$ docker --version
```



DOCKER IMAGES AND CONTAINERS

How does it work ?

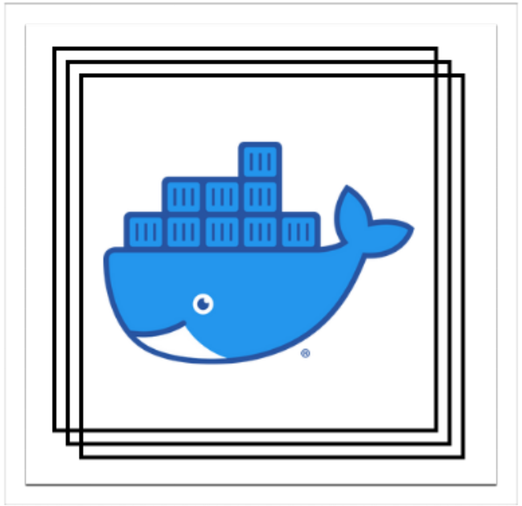
Application

YOLO V8
Flask

Dockerfile

Build

Docker image
(Snapshot)



- Development environment (for exp python)
- Copy of the files
- Libraries
- Commande(s) to be executed

How does it work ?

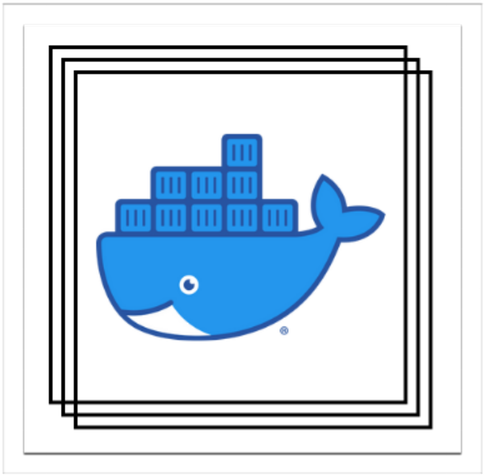
Application

YOLO V8
Flask

Dockerfile

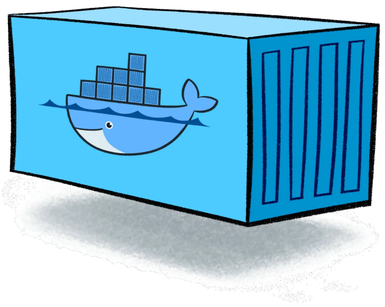
Build

Docker image

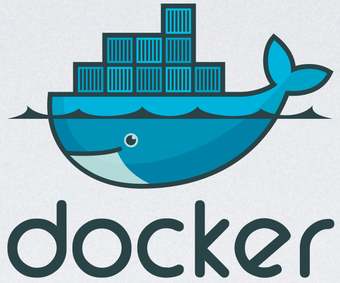


Run

docker container
(Isolated environment)



Runs its own software, binaries, and configurations



DOCKERFILE

https://docs.docker.com/engine/reference/builder/

Dockerfile

- Start from a base image with the command : **FROM**
- Base images can be Linux, or langages (installed on top of a linux) published on docker hub (See hub.docker.com) based on different versions of linux. Alpine (or slim) is preferable for docker images (Cf. https://www.alpinelinux.org/about/)
- Copy files into the image using the command **COPY**
- Execute a command using the command **CMD**

Dockerfile

https://github.com/AmineFrj/flask-backend

Dockerfile

```
1 FROM python:3.8-alpine #or lts-alpine or latest-alpine or slim
2 COPY . /app
3 WORKDIR /app #the commands will be executed within this directory (like cd command)
4 RUN pip install -r requirements.txt
5 CMD python app.py
```

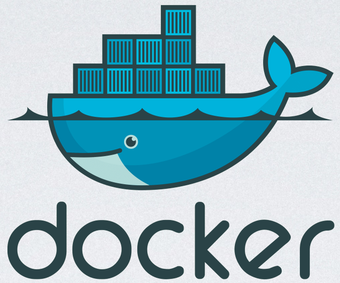
Nb. Try to always specify the version of the base image inside Dockerfile

Build and run an image

```
$ docker build -t <image_name>:<tag> . # -t flag is used to tag the image
$ docker run --name <container_name> <image_name>:<tag> # --name flag is used to name the container
```

```
$ docker images #or docker image ls
$ docker ps # to list the running containers
$ docker ps -a    # to list all containers
$ docker ps --help
```

```
$ docker build -t mon-premier-docker .
$ docker images
$ docker ps -a
$ docker run -p 5000:5000 mon-premier-docker
```



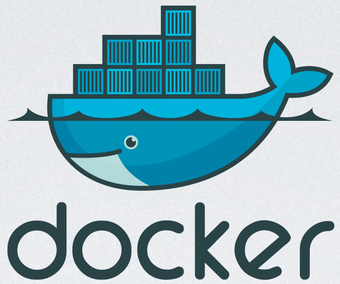
REGISTRY

DOCKER HUB

Docker registry

- Go to Docker Hub and search for an image : https://hub.docker.com/
- Pull the image locally https://hub.docker.com/r/jcdemo/flaskapp

```
$ docker pull jcdemo/flaskapp
$ docker run -p 5000:5000 jcdemo/flaskapp:latest #Latest by default if not specified
```

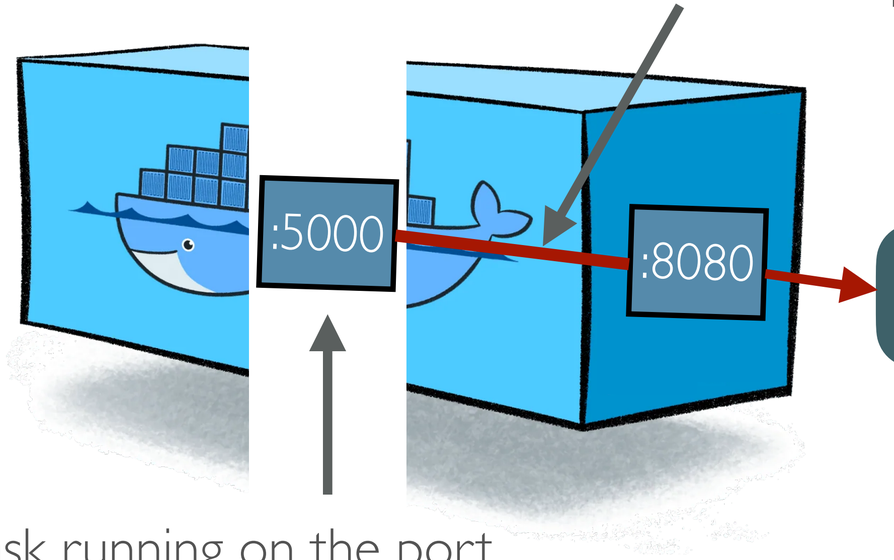


EXPOSE PORTS

Expose port

Container - Flask application

Map the host (public) 8080 demandes to the port 5000 internally (container)




```
$ docker run -d -p <host>:<container> <image>
```

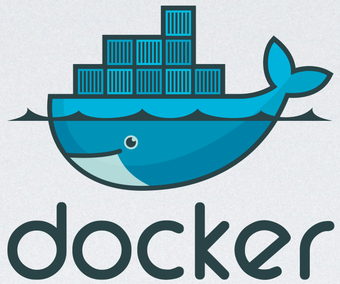
Flask running on the port 5000 (of the container)

Ps. The EXPOSE command (inside Dockerfile) opens a port to another container only, whereas publish flag opens a port to both public and other containers.

Expose port

```
$ docker stop <container_id>
```

```
$ docker stop 3c4e61706806
$ docker run -d -p 8080:5000 jcdemo/flaskapp:latest # -d means in detached mode
```



MANIPULATE CONTAINERS

HTTPS://DOCS.DOCKER.COM/ENGINE/REFERENCE/RUN/

Start and stop container

- You can start and stop containers by their name (runs latest configuration)

```
$ docker start <container_name> #or <container_id>
$ docker stop <container_name> #or <container_id>
```

```
$ docker start dazzling_moore
$ docker stop dazzling_moore
```

Rename and delete containers

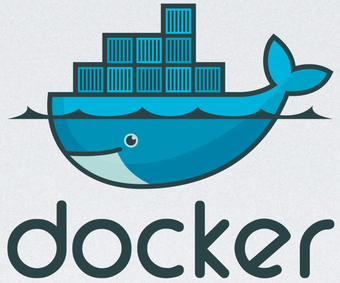
```
$ docker rename <old_name> <new_name>
$ docker rm <container_id>
```

```
$ docker start dazzling_moore
$ docker stop dazzling_moore
```

- Nb. The container must be stopped before deletion (or use -f to force deletion)

Container logs

```
$ docker logs <container-id>
$ docker logs -f <container-id> #follow
```



SOME DOCKER USEFUL CONCEPTS

Cache and layers

- Docker processes in layers and uses cache for commands already seen.

  ie. If you change the i-th line of Dockerfile, all commands after i-th line will be **recomputed** (docker cannot know what has been changed since).

- You have to take **advantage** of caching (for example : copy the files that can be changed in the futur right after installation).

Jump into the container

```
$ docker exec -it <container-id> bash #interactive
```

Nb. In case bash in not used in the container check for the commande used to execute the CMD

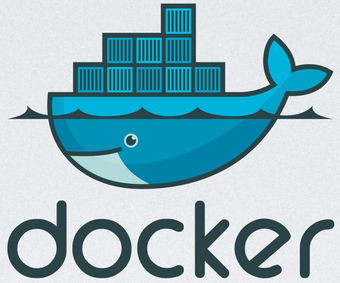
```
$ docker inspect <container-id>
$ docker inspect 8047f8e5c5ab | grep -A 1 Cmd #search for Cmd
$ docker exec -it <container-id> sh
```

.dockerignore

- To ignore files from being copied inside container

.dockerignore

```
1  Dockerfile
2  .git
3
4
5
6
7
8
9
```



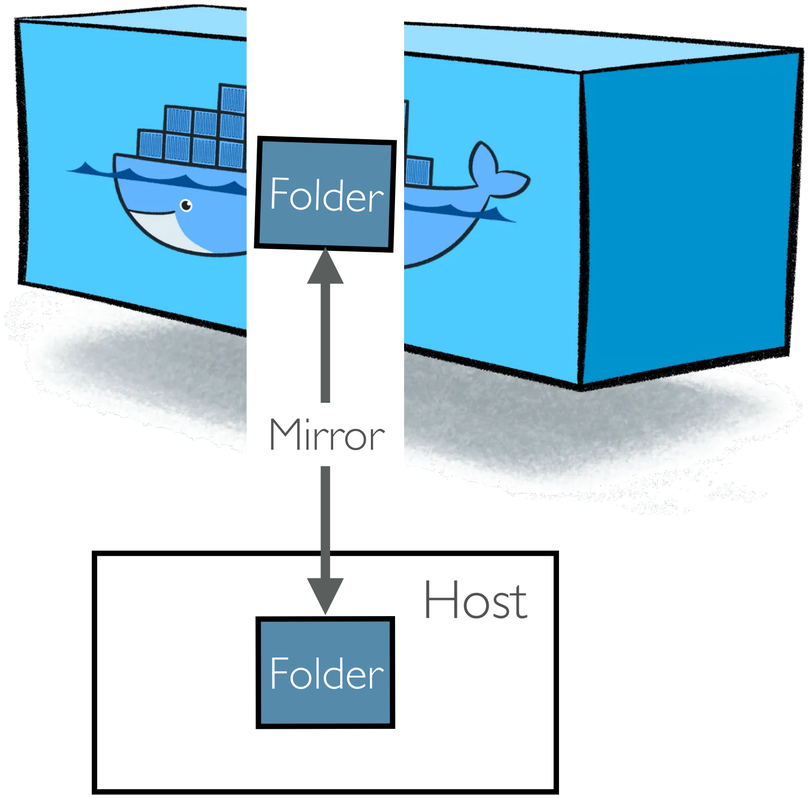
VOLUMES

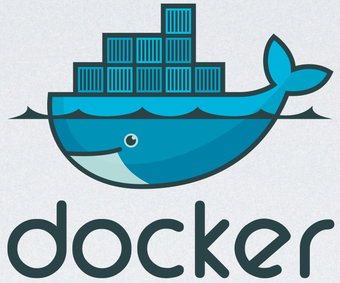
Volumes

- You can mirror some files/folder in the host inside the docker using -v flag

```
$ docker run -v <host>:<container> <image>
$ docker run -v <host>:<container>:ro <image> #for read-only
```

Ps. Use absolute paths

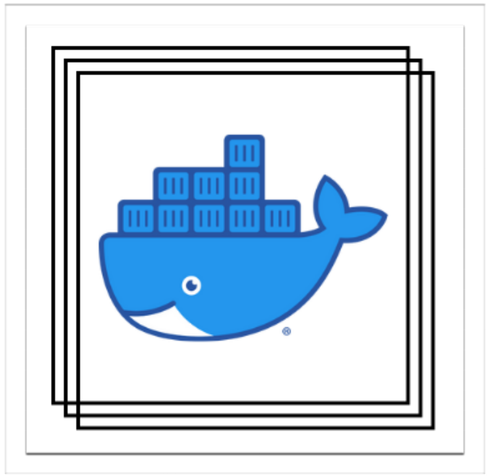




DOCKER REGISTRY

How does it work ?

Docker image



Push

Docker registry

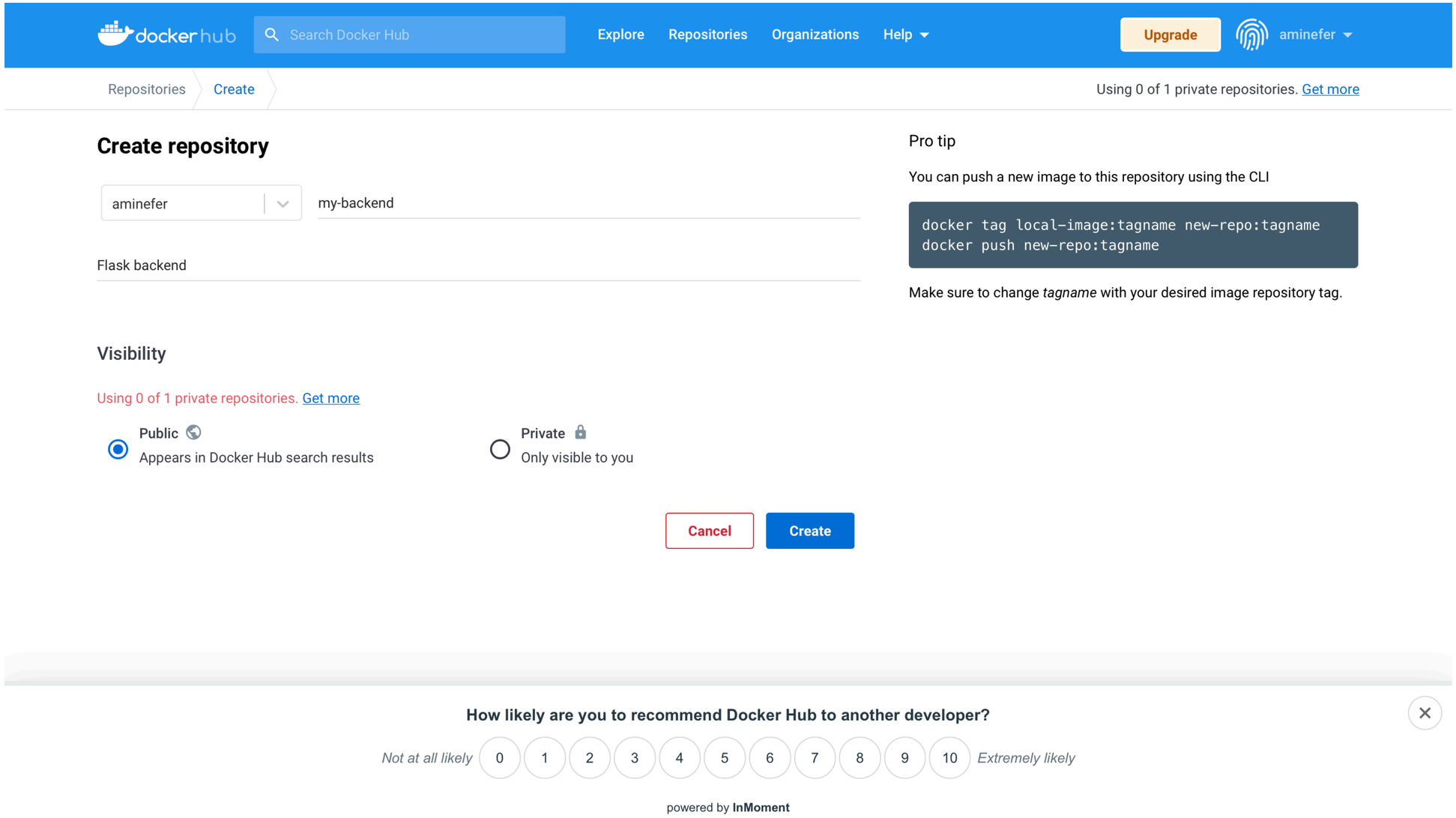
Local

Docker hub

Push to Docker Hub

- Register in Docker hub : http://hub.docker.com
- Go to repositories > Create repository

Push to Docker Hub



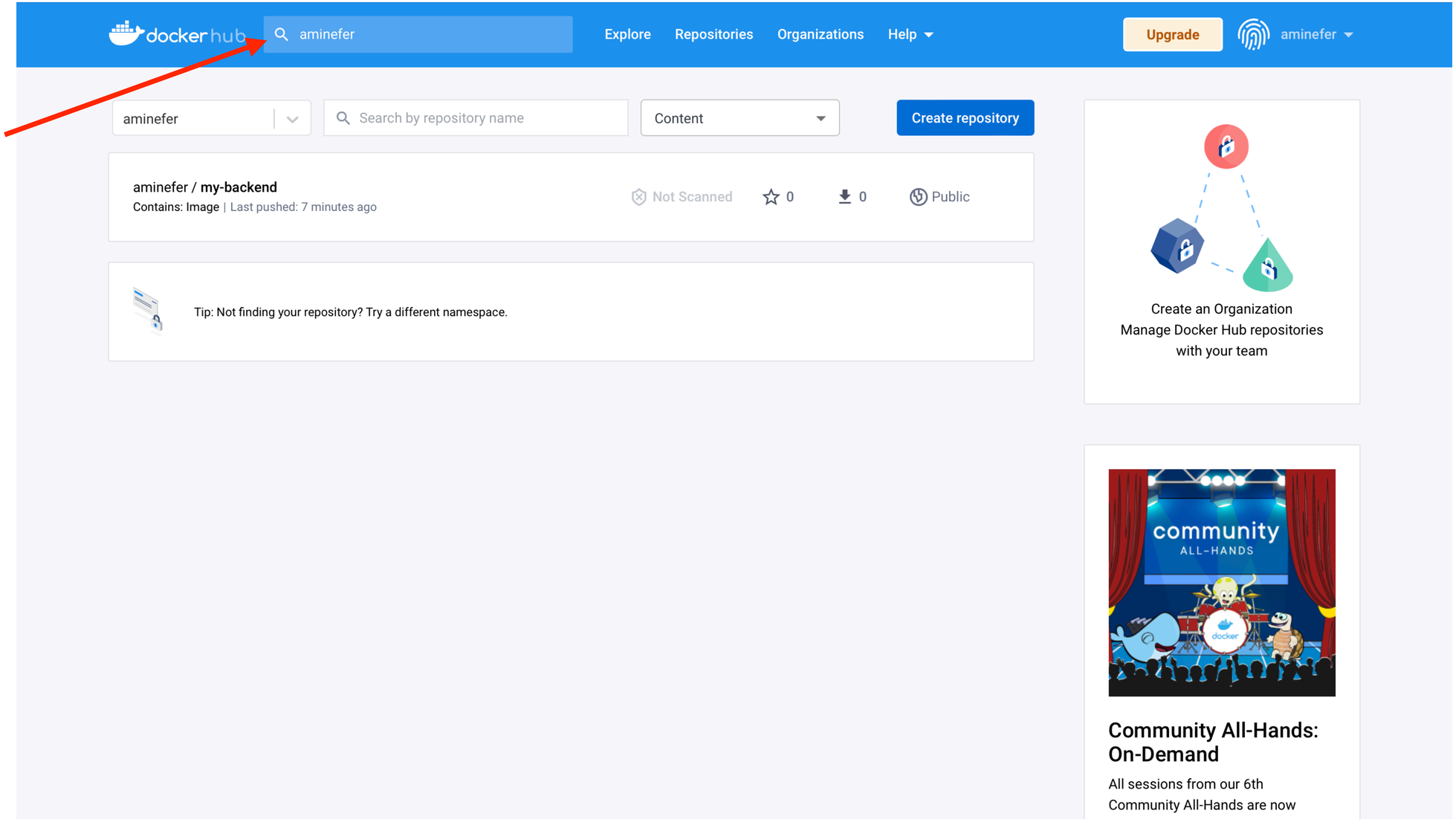
Push to Docker Hub

- Go to the terminal and login using the command :

```
$ docker login
```

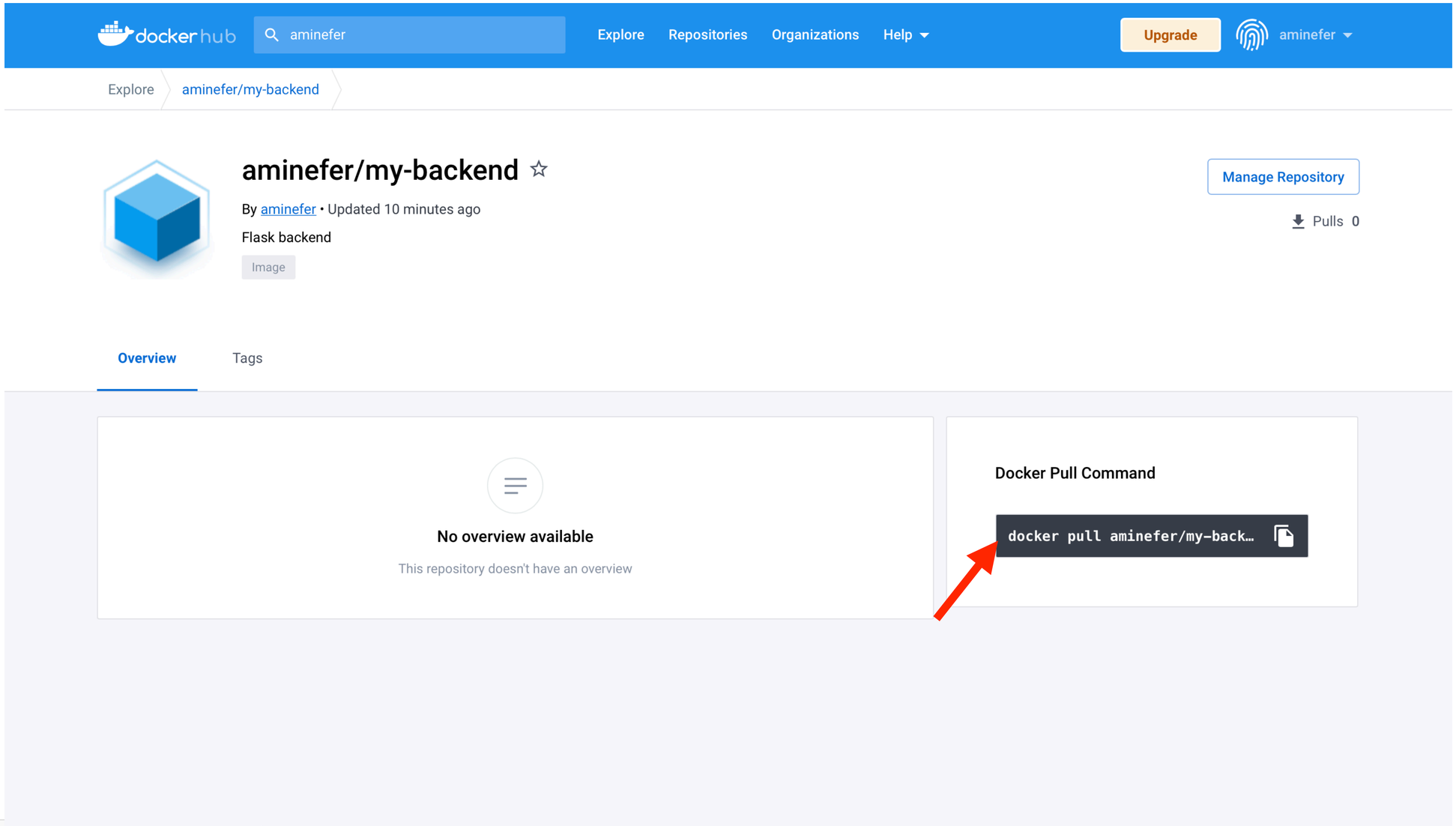
Search for your image

Search for your image



Pull it !

https://labs.play-with-docker.com/



Good practices

- Create bash scripts as shortcuts (for example to jump into a docker)
- Always write version inside dockerfile and libraries versions inside requirements.txt

Container orchestration system

kubernetes

In production we usually have hundreds of containers and we need to manage the communication between them and bugs that occur etc.

So we use a container orchestration system, the most known one is *Kubernetes*