# Docker for Data Scientist

Yes, you need it…

## Prerequisites

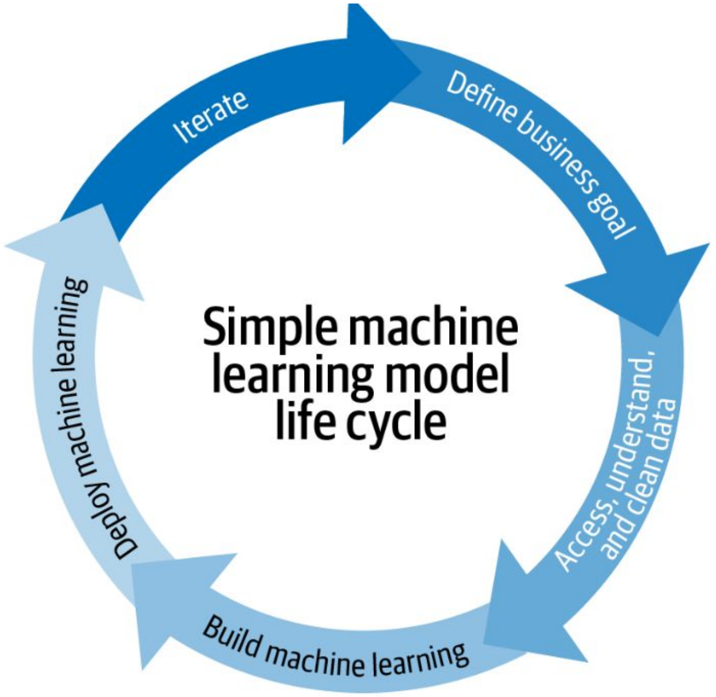
## Python Environments: your small sandbox to play

- different projects mean different libraries
- sometimes even with different versions
- virtual environments  keep these dependencies in separate “sandboxes” so you can switch between both applications easily and get them running

# Lab 1: Anaconda + Conda environments

1. Install Anaconda [doc]
2. Getting Started with Conda Environments [article]
   a. Create a new conda environment: `conda create --name your_env_name python=3.7 -y`
   b. Activate your environment; `conda activate your_env_name`
   c. Deactivate your environment: `conda deactivate`
   d. Delete your new environment: `conda remove --name your_env_name --all`
3. Clone the repo https://github.com/enriquemoraayala/agiledatascience_labs

# Running a ML project


Simple machine learning model life cycle
Define business goal
Access, understand, and clean data
Build machine learning
Deploy machine learning
Iterate

## Steps in a ML project

1. Set up the environment to work
   a. Install language runtime: Python? R? Julia?... in the proper version
   b. Install needed packages (and dependencies): numpy, pytorch, pandas, …. in the proper version
2. Build the modeling code in the selected language
3. Run the training script and measure the result
4. Run the inference part

Are you able to share it? Are you able to reproduce it in a near future? sure?

## Lab 2: try to run the following …

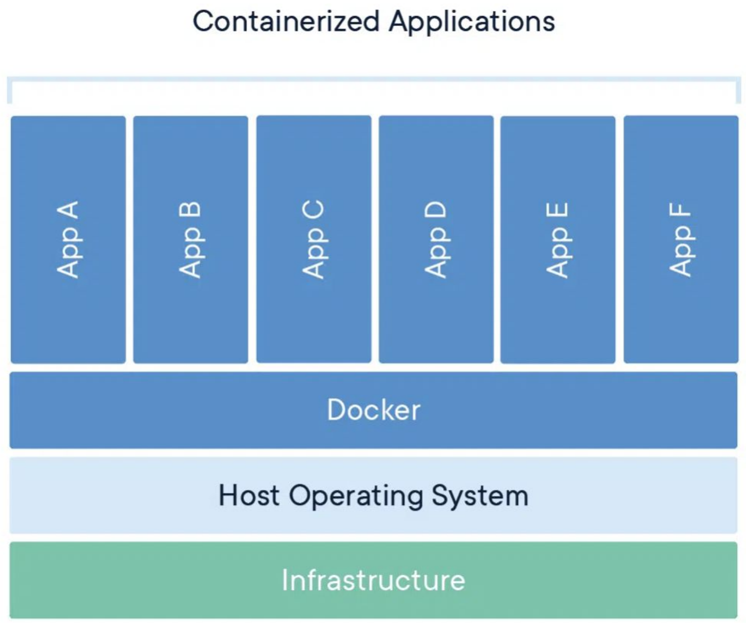
1. Create a new environment with python 3.8
2. Follow the Logistic Regression with SciKit Learn [Article] (use the notebook *digits_toy.ipynb* in the github)
3. Let’s talk about it…
    a. Which are the steps if you want to share your code / model with your partner?
    b. What would happen if you want to run everything again next month to retrain the model?
    c. And in a year?
    d. And if you need more computer power? or more memory?

## Dockers

## What is Docker?

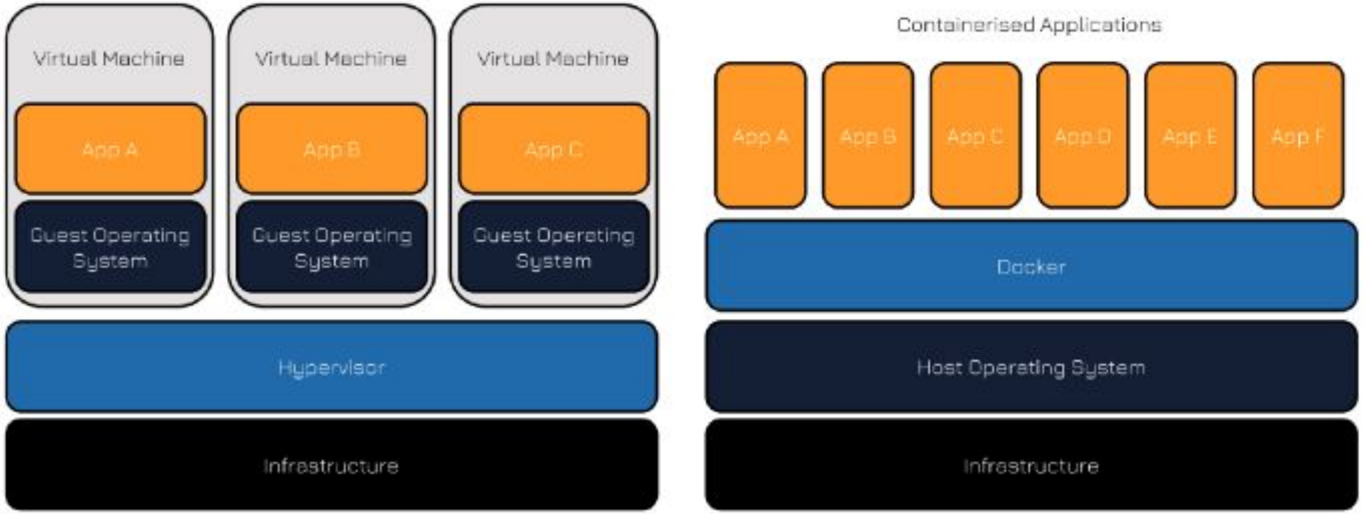
Docker is a **free software platform** that allows you to **package and isolate** your software.

That is, with Docker you can create **an element** (called Image) that **will include everything you need** to run your application.



The Docker handbook

## Dockers vs Virtual Machines



On the left, we have three virtual machines (VMs) running apps, each with its own guest operating system (OS), on a single host supported by a hypervisor. On the right we have six containerised apps, running on Docker, sharing the host OS (images by author).

# Docker Container

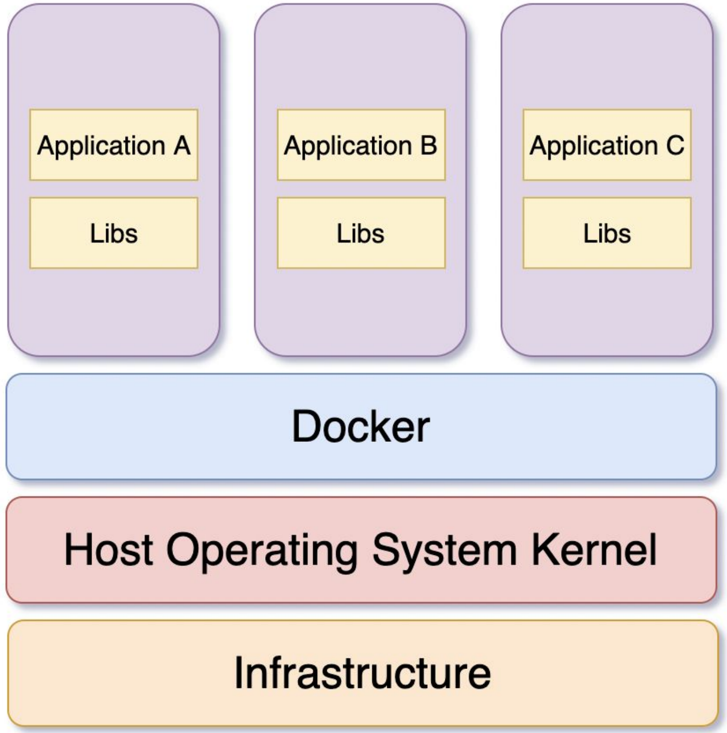
containers are completely isolated environments from the host system as well as from each other

Containers

Instead of having a complete guest operating system inside a container, it just utilizes the host operating system via the container runtime while maintaining isolation

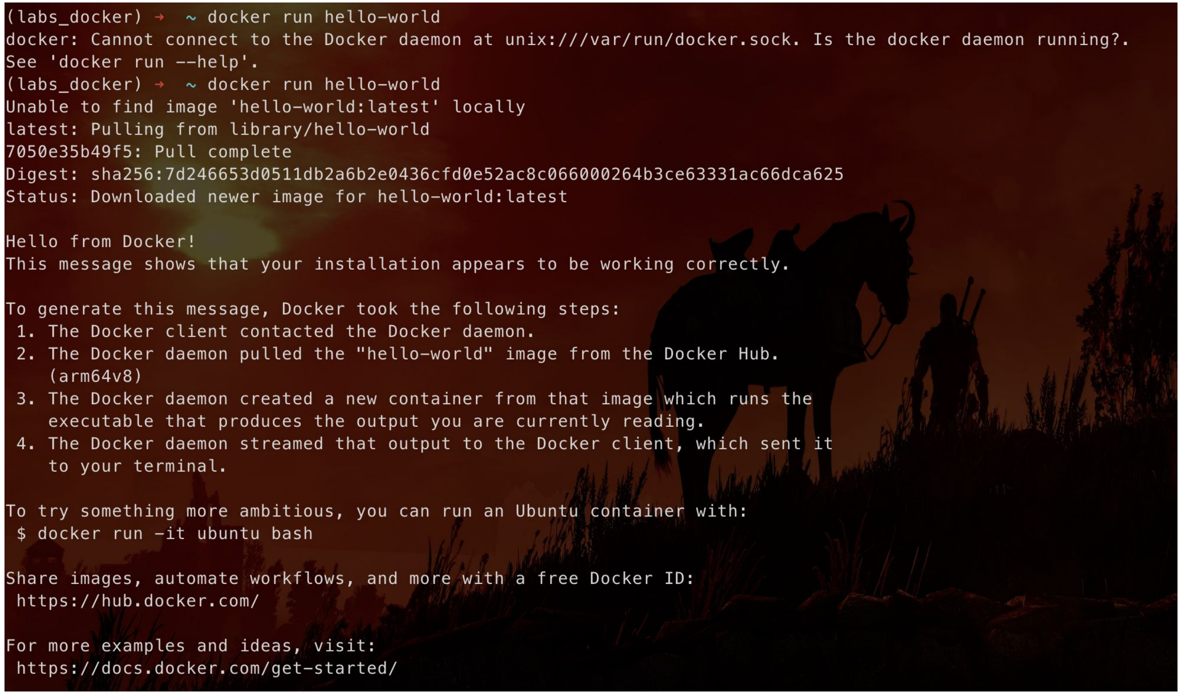
Windows, Linux, Mac

Hard disk, memory, network card, …



## Lab 3: Install Docker

1. Install Docker [link]
2. Test if it is working
   a. `docker run hello-world`



```
(labs_docker) ➜  ~ docker run hello-world
docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?.
See 'docker run --help'.
(labs_docker) ➜  ~ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

## Docker Image

Images are multi-layered self-contained files that act as the **template for creating containers.**

## Docker Registry

An image registry is **a centralized place where you can upload your images** and can also download images created by others. Docker Hub is the default public registry for Docker.

## …so, at this moment I know:

1. I cannot work alone for ever, so I have to share my code and make it reproducible
2. It could be really difficult since everyone is working in their own computer
3. I can create environments in python to make the libraries versions under control (they will change during time)
4. I can create mini-server with everything inside called Docker Containers
5. I can create such Containers using templates or Images

## Dockerfile

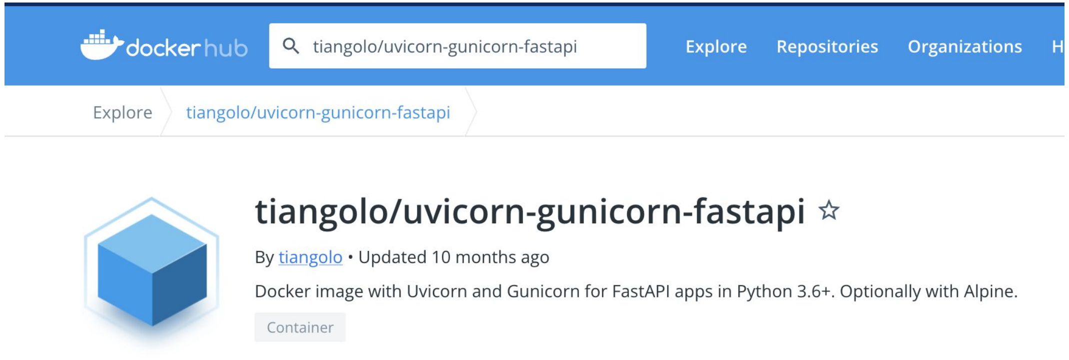
To build Docker Images (and run them in the Containers) we need to write a **Dockerfile**.

*“Dockerfile is a file that indicates Docker the instructions to follow to include everything necessary in the image: the application, the programs you need, the installation of the libraries, etc.*

*For this, Docker has several commands that allow us to know what elements to include in our image. The most typical instructions are **`FROM`**, **`COPY`**, **`RUN`**, **`CMD`** and **`ENTRYPOINT`**”*

## Dockerfile. FROM

- This verb indicates the base image from where we are going to start, since you can find out there pre-created images ready to fine-tune for your project
- To find these images, the most common thing to do is to search for the image in the Docker image repository


docker hub
tiangolo/uvicorn-gunicorn-fastapi
Explore
Repositories
Organizations
Explore
tiangolo/uvicorn-gunicorn-fastapi

tiangolo/uvicorn-gunicorn-fastapi

By tiangolo • Updated 10 months ago

Docker image with Uvicorn and Gunicorn for FastAPI apps in Python 3.6+. Optionally with Alpine.

Container

```
FROM tiangolo / uvicorn-gunicorn-fastapi
```

## Dockerfile. RUN

The RUN instruction will execute a command in the console. This will help us to install new software that we need but that the image we started from does not include.

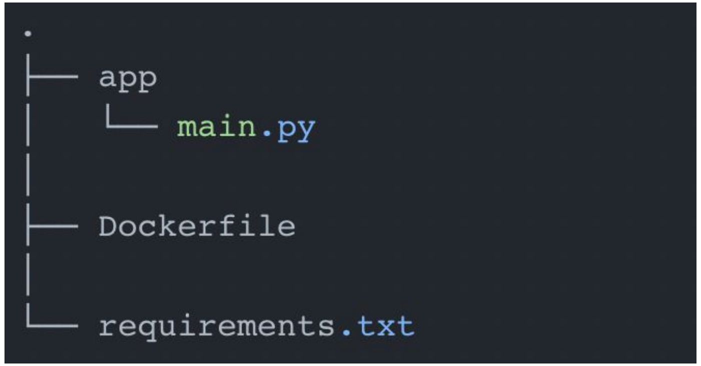
```
RUN apt-get update
RUN apt-get install -y libpq-dev
```

```
RUN pip install -r requirements.txt
RUN mkdir -p app
```

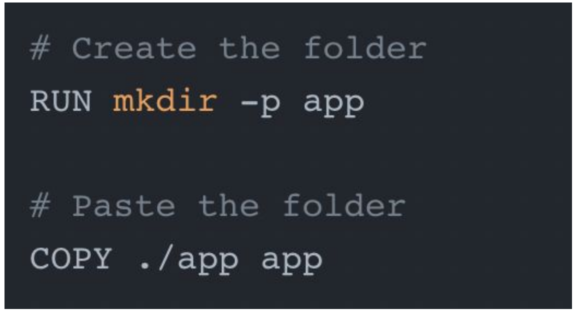
# Dockerfile. COPY, ADD and WORKDIR

While COPY only copies local files, with ADD you can add files from a URL.

Besides, if you add a local compressed file (like a tar.gz file), ADD will unzip it into a new folder



```
.
├── app
│   └── main.py
│
├── Dockerfile
│
└── requirements.txt
```



```
# Create the folder
RUN mkdir -p app

# Paste the folder
COPY ./app app
```

## Dockerfile. **EXPOSE, ENTRYPOINT and CMD**

If your code will be listening on a port, `EXPOSE` allows you to define the port on which your application will be listening.

On the other hand, `ENTRYPOINT` allows you to define the command to be executed when the Docker image is launched. For its part, `CMD` allows you to define the arguments that `ENTRYPOINT` is going to use.

```
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8080"]
```

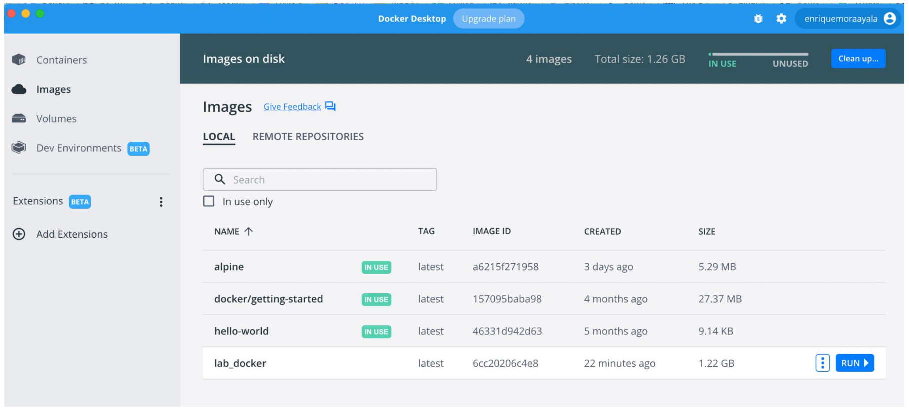
## …ok, let’s do it together… step by step

1. Create a new folder
2. Activate the python environment
3. Write a Dockerfile like this

```
FROM tiangolo/uvicorn-gunicorn-fastapi
COPY requirements.txt .
RUN pip install -r requirements.txt
RUN mkdir -p app
COPY ./app app
EXPOSE 80
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "80"]
```

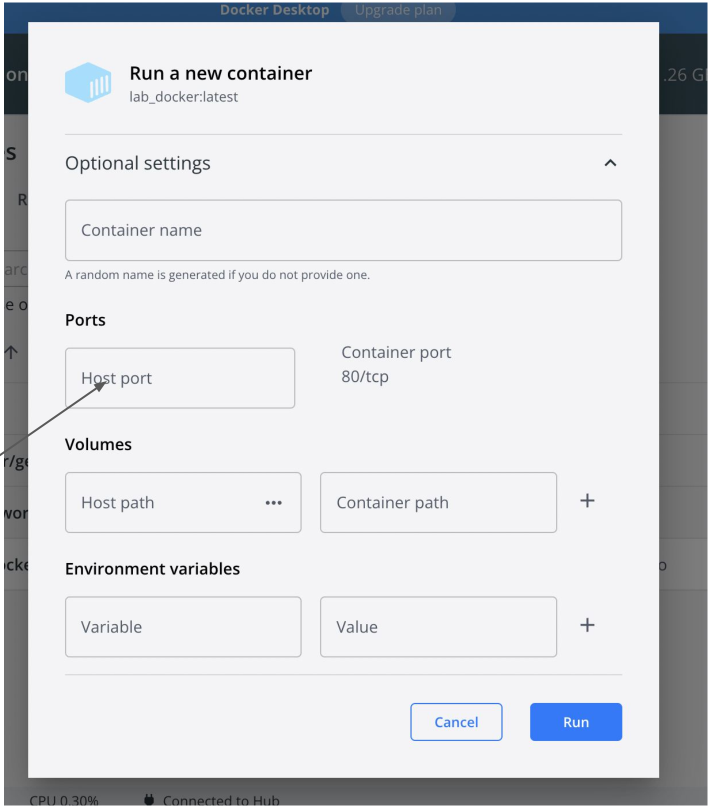
4. Copy the “app_test” folder from the course github
5. Generate a Docker Image - `docker build . -t <image_name>`
6. Run the Image in the Docker Container

# Run the Docker Image



Images on disk — 4 images — Total size: 1.26 GB

| NAME | | TAG | IMAGE ID | CREATED | SIZE |
|---|---|---|---|---|---|
| alpine | IN USE | latest | a6215f271958 | 3 days ago | 5.29 MB |
| docker/getting-started | IN USE | latest | 157095baba98 | 4 months ago | 27.37 MB |
| hello-world | IN USE | latest | 46331d942d63 | 5 months ago | 9.14 KB |
| lab_docker | | latest | 6cc20206c4e8 | 22 minutes ago | 1.22 GB |



Run a new container
lab_docker:latest

Optional settings

Container name

A random name is generated if you do not provide one.

Ports

Host port — Container port 80/tcp

Volumes

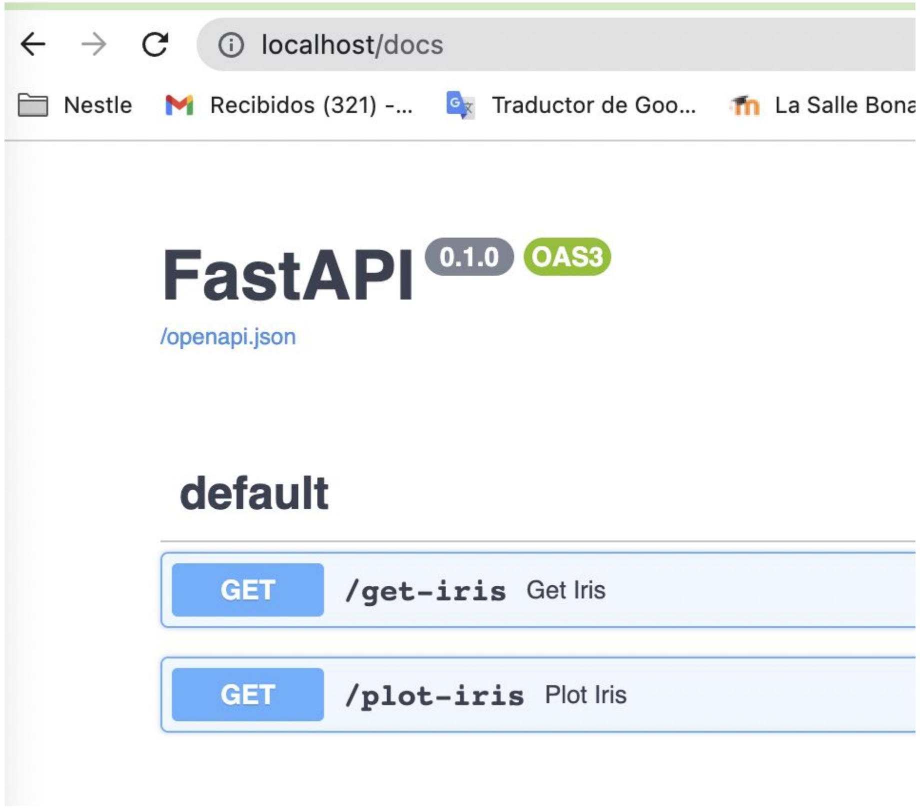
Host path — Container path

Environment variables

Variable — Value

Host port: 80

## Request the API inside the app


localhost/docs

Nestle Recibidos (321) -... Traductor de Goo... La Salle Bona

FastAPI 0.1.0 OAS3

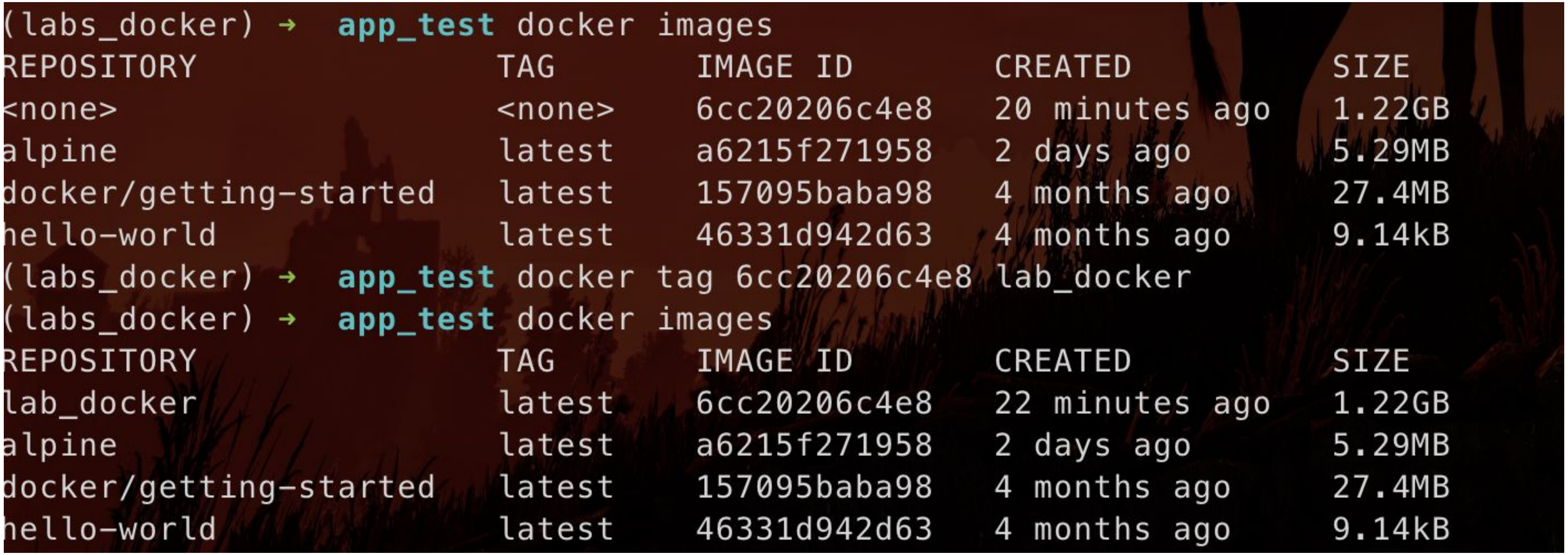
/openapi.json

default

GET /get-iris Get Iris

GET /plot-iris Plot Iris

## Result



```
(labs_docker) ➜  app_test docker images
REPOSITORY               TAG       IMAGE ID       CREATED          SIZE
<none>                   <none>    6cc20206c4e8   20 minutes ago   1.22GB
alpine                   latest    a6215f271958   2 days ago       5.29MB
docker/getting-started   latest    157095baba98   4 months ago     27.4MB
hello-world              latest    46331d942d63   4 months ago     9.14kB
(labs_docker) ➜  app_test docker tag 6cc20206c4e8 lab_docker
(labs_docker) ➜  app_test docker images
REPOSITORY               TAG       IMAGE ID       CREATED          SIZE
lab_docker               latest    6cc20206c4e8   22 minutes ago   1.22GB
alpine                   latest    a6215f271958   2 days ago       5.29MB
docker/getting-started   latest    157095baba98   4 months ago     27.4MB
hello-world              latest    46331d942d63   4 months ago     9.14kB
```

## Homework: Build a Docker Image

Build a Docker Image with as a Jupyter Notebook lab for your experiments

[Extra] Mount a local from your computer in the Docker Container to be able to develop locally but execute in the Docker Container

[NOTE] You can debug apps inside a docker container. See this example