



Advanced Software Engineering (**LAB**)

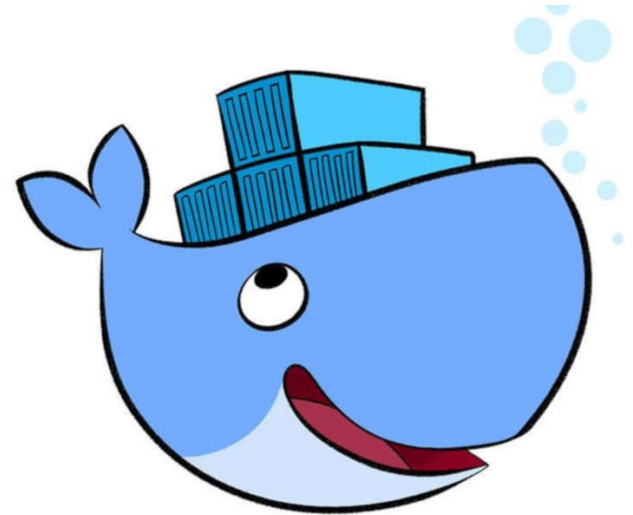
Stefano Forti

`name.surname@di.unipi.it`

Department of Computer Science @ University of Pisa

What will I do?

- Complete a multi-service application.
- Write `Dockerfiles` to create images to deploy your services.
- Use Docker Compose to run your multi-service application.



Instructions

PART ONE

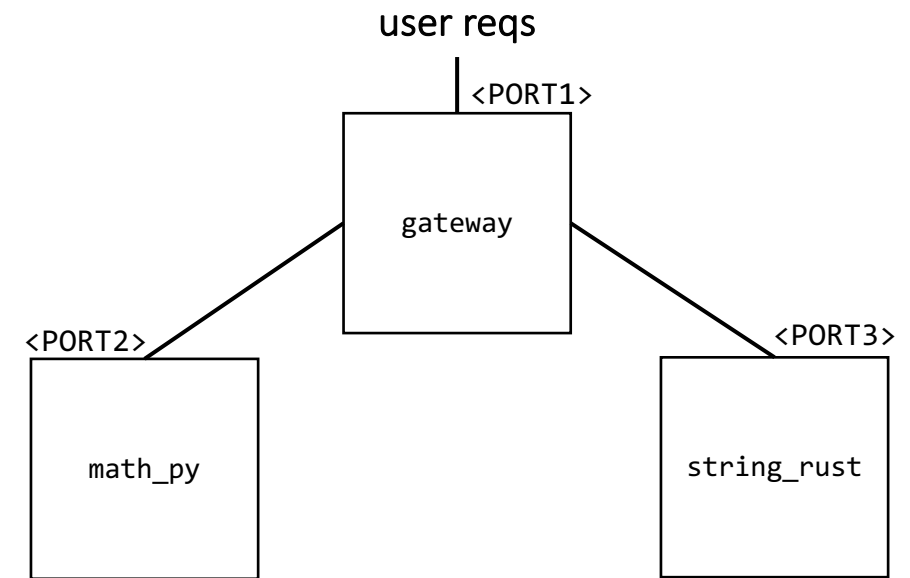
1. Download the code of the **microase** app from the Moodle.
2. Complete the **math** service code to feature the requested operations and set a port in the **.flaskenv** file.
3. Run the **math** service through the provided **Dockerfile** (instructions follow), after completing it with a suitable port.
4. Complete the **gateway/urls.py** file with suitable service name and ports.
5. Write **Dockerfile** for the **gateway** and **string_rust** services.

PART TWO

1. Write a **docker-compose.yml** file to run all three services.
2. Run the whole application through Docker Compose.

PART THREE (bonus)

1. Add a **random** service to feature operation related to randomness (e.g. choice, randint, ...).



Dockerfile

A **Dockerfile** is a text-file that contains a set of steps that can be used to create a Docker image.

Copy your current directory into the specified one in the container

```
1 FROM python:3.8-slim-buster
```

Define the base image

```
2
```

```
3 ADD . /gateway
```

```
4 WORKDIR /gateway
```

Move to the new directory

```
5
```

```
6 RUN pip3 install -r requirements.txt
```

Specify commands to be run while building the image

```
7
```

```
8 EXPOSE 5000
```

Container being build should listen on this port

```
9
```

Specify commands to be launched at startup.

```
10 CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

As easy as docker build .

The docker
command

Path to the
Dockerfile file

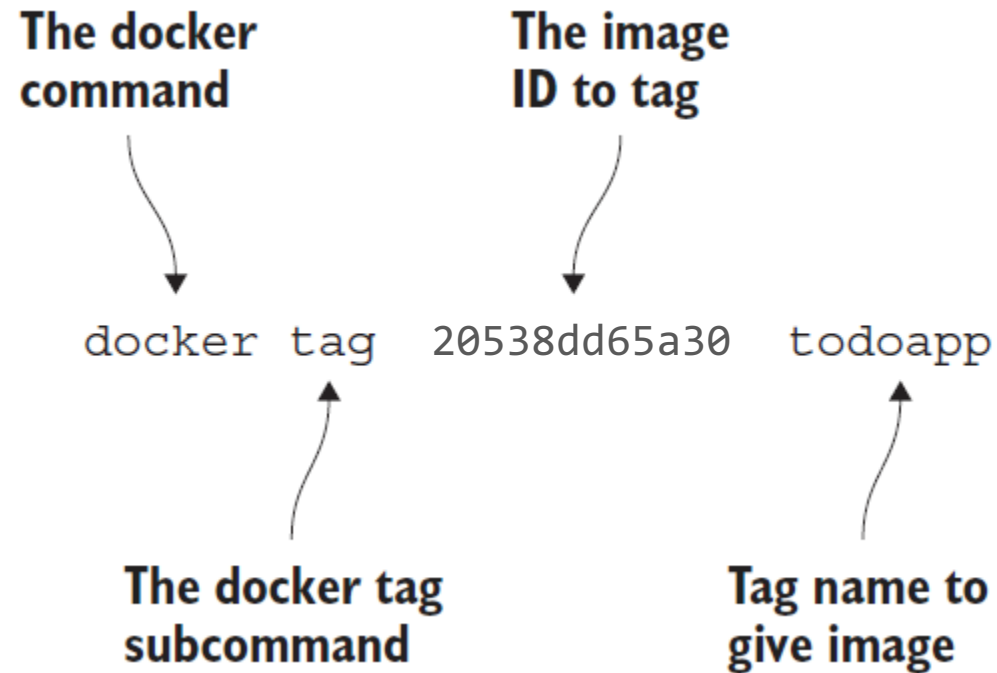
docker build .

The docker build
subcommand

```
(base) stefanoforti@MBPdiStefano4 math_py % docker build .
[+] Building 0.7s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.8-slim-buster
=> [internal] load build context
=> => transferring context: 151B
=> [1/4] FROM docker.io/library/python:3.8-slim-buster@sha256:03c12f7bbd977120133b73e4b3ef5c5707ca09be338156dc02306d41633db4c0
=> CACHED [2/4] ADD . /math_py
=> CACHED [3/4] WORKDIR /math_py
=> CACHED [4/4] RUN pip3 install -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:19f73b6c37fe020ce5a63dcf9d59eb2b294cddde819e9927c5a59398c8070385

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Tag an image



Run it!

```
docker run -p 5001:5000 <img_tag>
```

```
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

Try it:

```
http://127.0.0.1:5001/add?a=2&b=1
```

