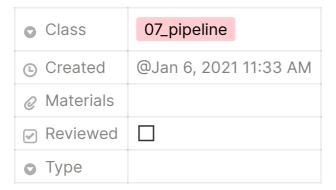
# **Docker-Compose**



## **Docker Compose**

## What is Docker useful for?

 Allows you to develop and run code/applications from/on different operating systems and servers (hardware) without depended problems on os and hardware

## 0) Warmups

 Check if Docker Compose is installed (should be the case if you are using Docker Desktop)

```
docker-compose --version
```

If it is not installed, please install it. Follow the instructions on <a href="https://docs.docker.com/compose/install/">https://docs.docker.com/compose/install/</a>

## 1) What is Docker-Compose

• It is a tool for defining and running multi-container Docker applications

Docker-Compose 1

- Why do we want to run multi-container Docker applications (Micorservices architecture)? → Has advantages over monolithic architecture
  - Highly observable → through logging one can track what each service is doing
  - Loosely coupled → each service can perform their task without being overly dependent on other services
  - **Highly testable** → Easier to test services independently
  - Highly maintainable → easier to structure, document, build and change each service
  - Easily replaced or reimplemented
  - Scalability → Easier to scale up one service on its own than the whole application
  - → adheres to the Unix philosophy: "do one thing well"

Are there disadvantages to the microservices architecture:

- · Communication overhead
- Overhead to learning and applying the tools of microservices architecture

#### 2) How do we use Docker-Compose

- Docker compose is set up from different Docker containers
- The whole configuration of your Docker-Compose application will be in the YAML file. This is central to Docker-Compose
- In the yaml file you define all the services/containers of your application
- When debugging during the week and you have no idea as to why your code is failing, first start by looking at your docker\_compose.yml

#### Sidenote:

Docker-Compose 2

When do we have to rebuild (docker-compose build our pipeline and when can we just use docker-compose up after changing something?

- Scenario 1: we did not create a logical volume (volumes: or -v)
  In this case, you will always have to rebuild after changing something
- Scenario 2: we did create a logical volume
  In this case, you will not have to rebuild if you change the app.py file (or any python file); you will have to rebuild if you changed the pockerfile the requirements.txt file or the docker-compose.yml

## 4) Show

- Talk about volumes
- Mention pymongo
- docker-compose ps
- docker-compose logs

Docker-Compose 3