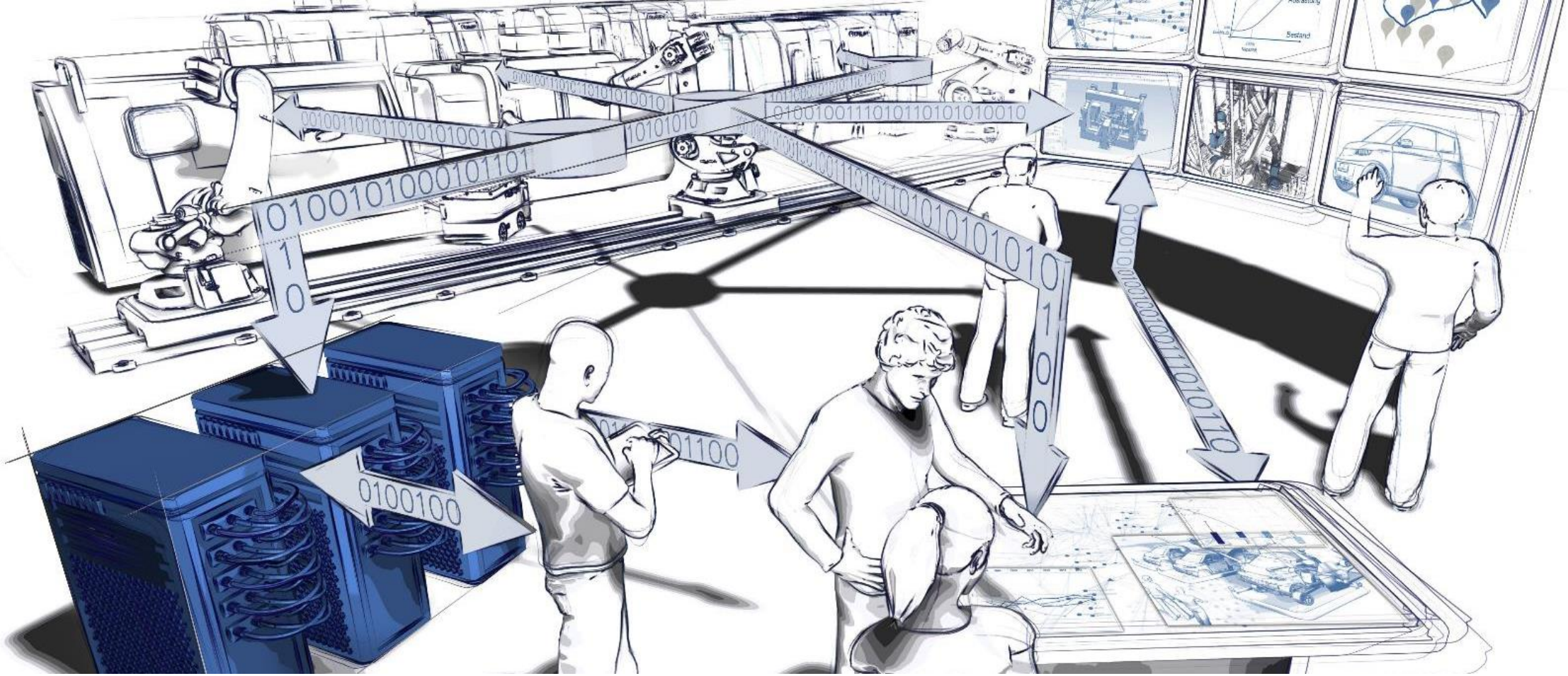


Make your Cluster App Cluster ready

Docker and Kubernetes

Overview

1. Docker, Kubernetes and us
2. Images and Container
3. Container Communication
4. Flask Demo
5. Docker Registry
6. GitLab Demo
7. Container Networks
8. Mongodb Demo
9. Integration in Kubernetes

A Cluster for a Cluster

Meet the Team

- Moritz Kröger (LLT)
- Michael Rath (IMA)
- Stefan Braun (I5)
- Manuela Dalibor (SE)
- Johannes Lipp (FIT)
- Iraklis Dimitriadis (FIT)

Goals

- Provide Infrastructure for your applications
- Provide basic functionality like databases or brokers (SQL, Kafka, MQTT …)

A Cluster for a Cluster

Common pain points

- Lack of access to Big Data infrastructure
- How to set up my database?
  - How to do analytics?
- Many institutes start setting up their own server hardware
  - But I need to store my data!

A Cluster for a Cluster

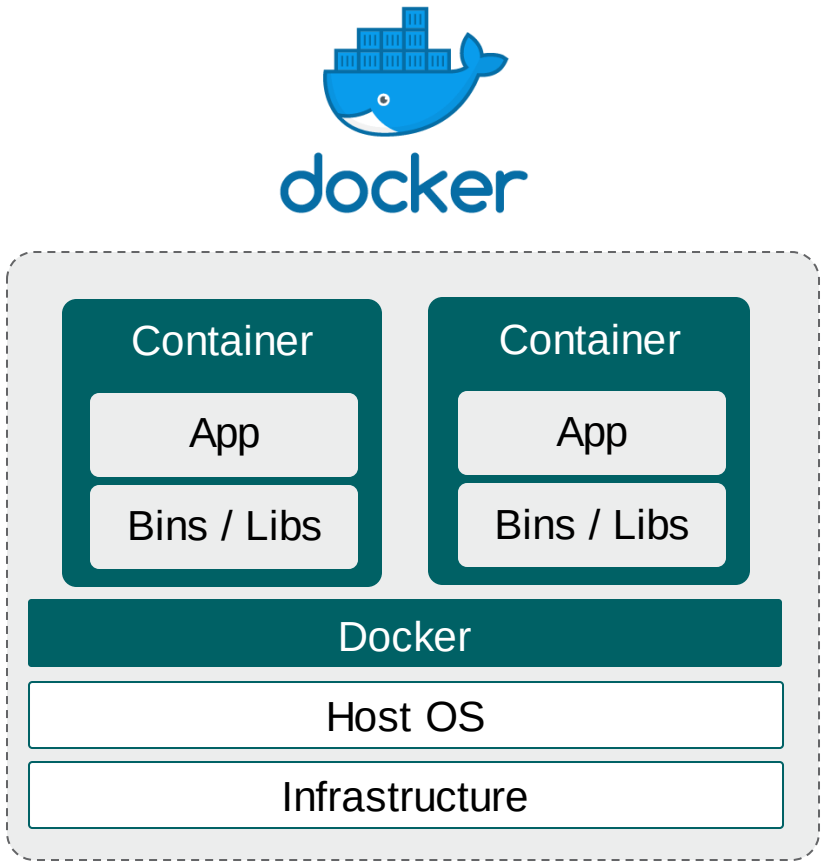
Why this training?

- How to turn your application cloud-native
- Make the transition to Kubernetes easy

Famous quotes

- *I just installed program A and now program B does not work anymore*
- *But it works on my machine!*
- *My student wrote this code years ago,*
  but it just runs on this one machine.
  Nobody except me is allowed to touch this computer!
- *I wrote this great analysis. It saves everything into these CSV Files.*
  I have about 4 million different files now. Only I know how everything works.

Managing Dependencies with Docker



- Containerization of software
- Build once, run everywhere
- All dependecies in one container
- Abstraction from hardware

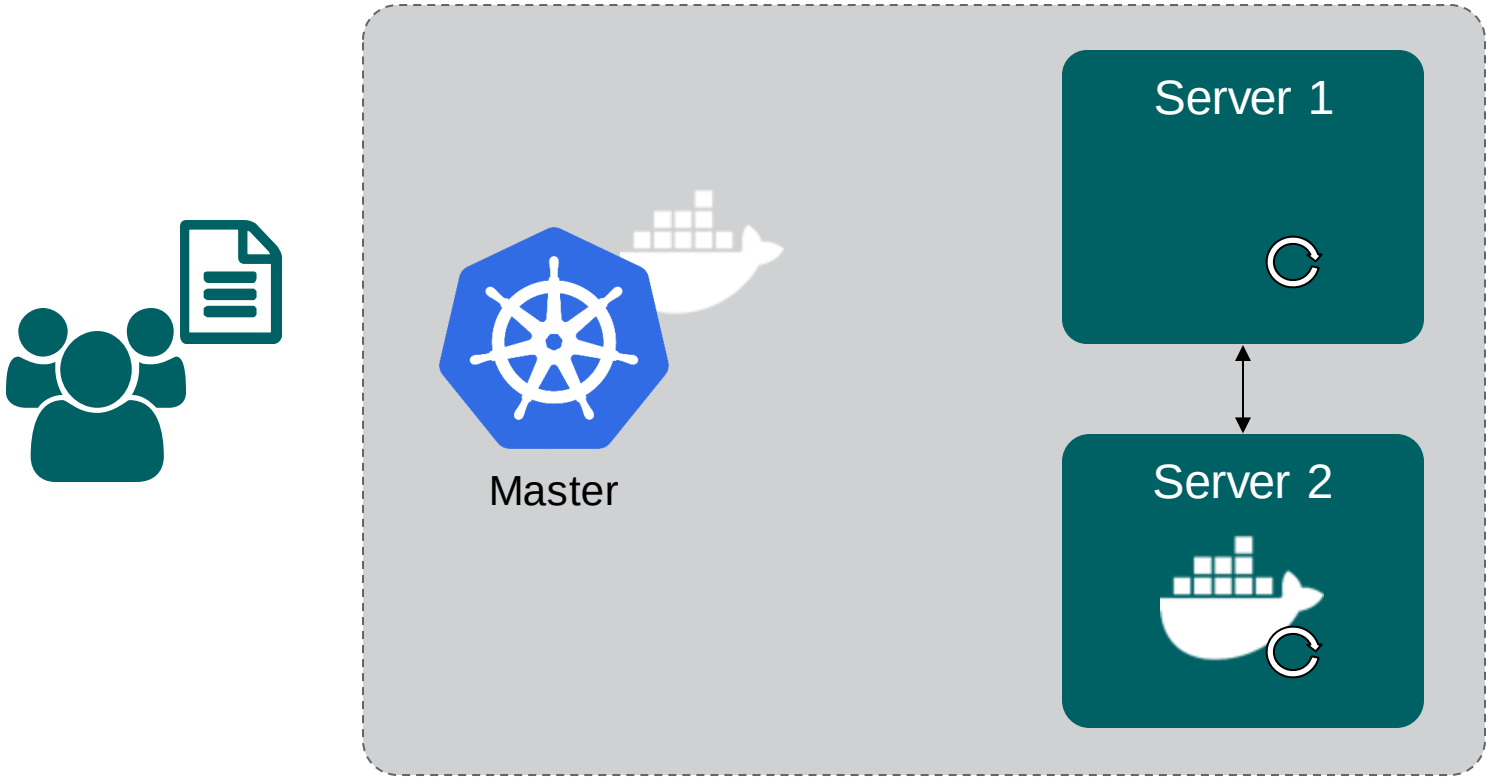
⇨ Cloud-native app design

Like Word-files but for apps

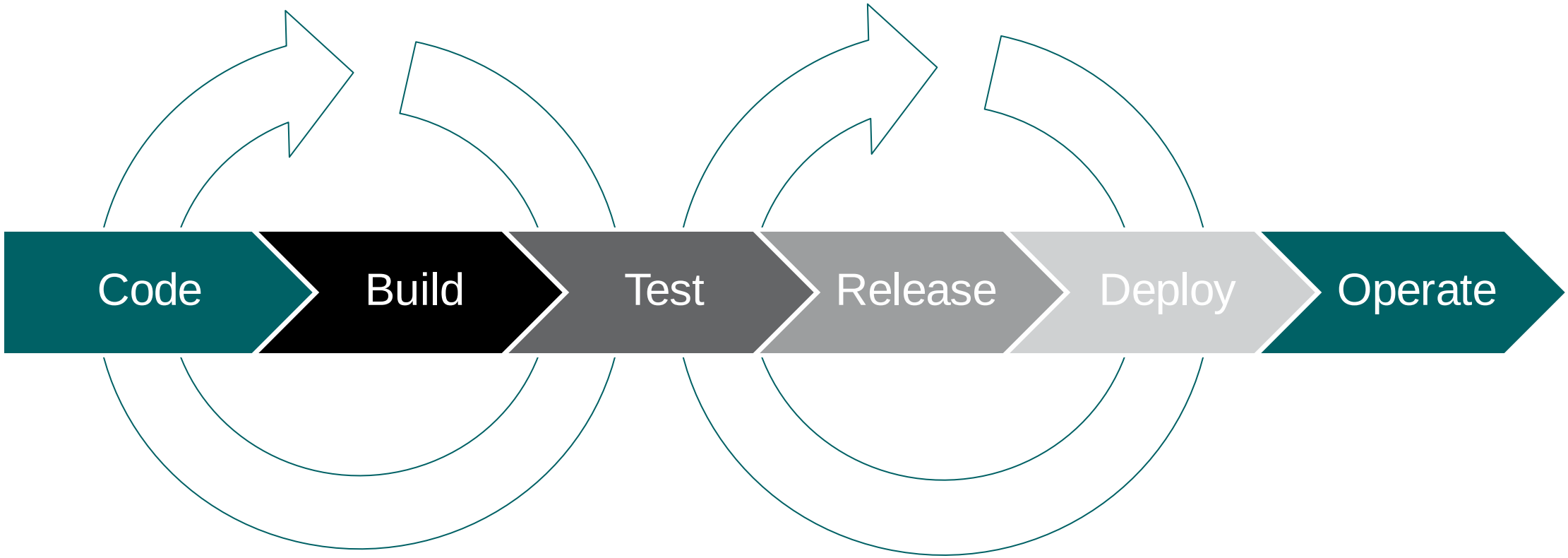
Managing Containers with Kubernetes

- Cluster/Server manager
- Container orchestration
- Offspring of Google's Borg
- Completely open source
- ~45.000 Contributors

Kubernetes in a Nutshell



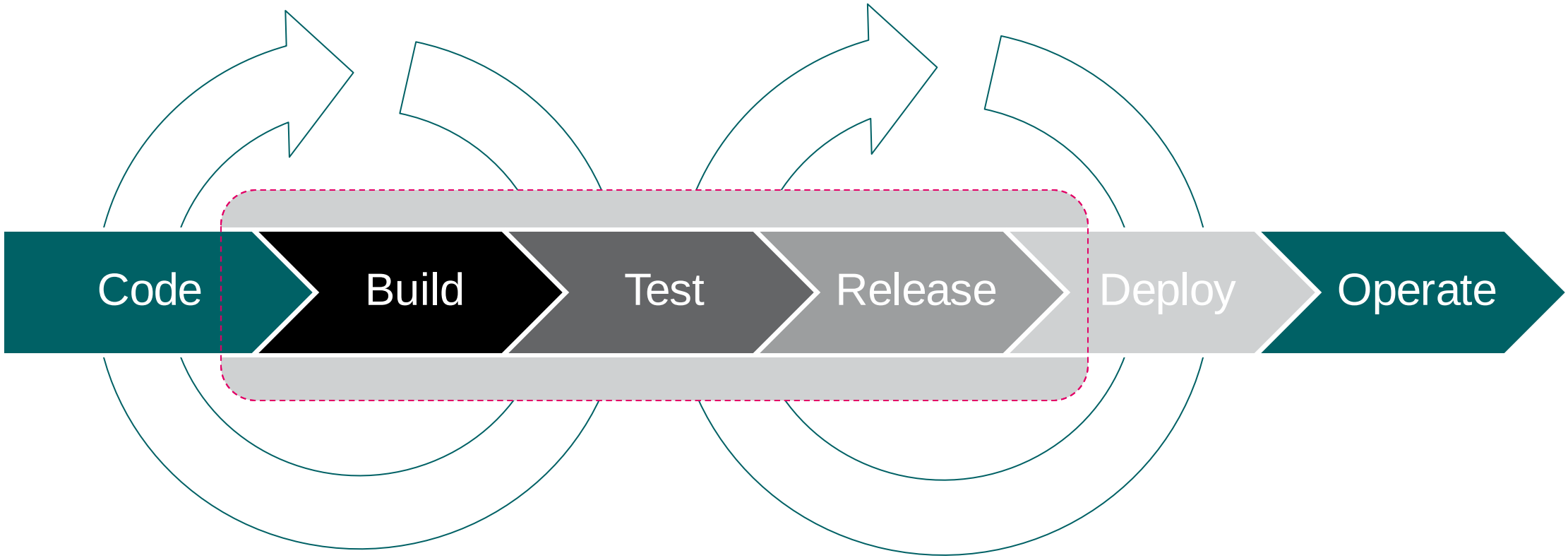
Continuous Integration / Deployment



Continuous Integration

Continuous Deployment

Continuous Integration / Deployment



Continuous Integration

Continuous Deployment

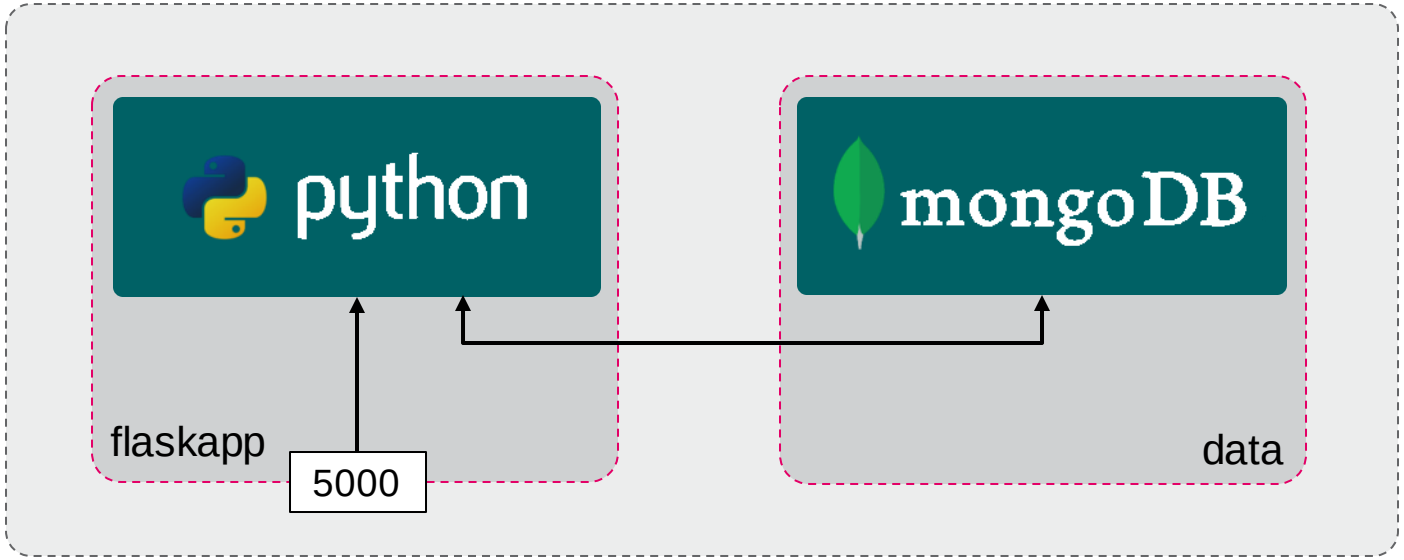
Let's Get Started

Code, Slides and Videopresentation can be found here:
https://github.com/rwth-iop/DockerMicrotraining

Please ask questions in the Zoom channel during the presentation.

Our experts will answer them!

Our App



Getters:

`localhost:5000/getID`	return ID of service
`localhost:5000/getJson`	return saved JSON Files

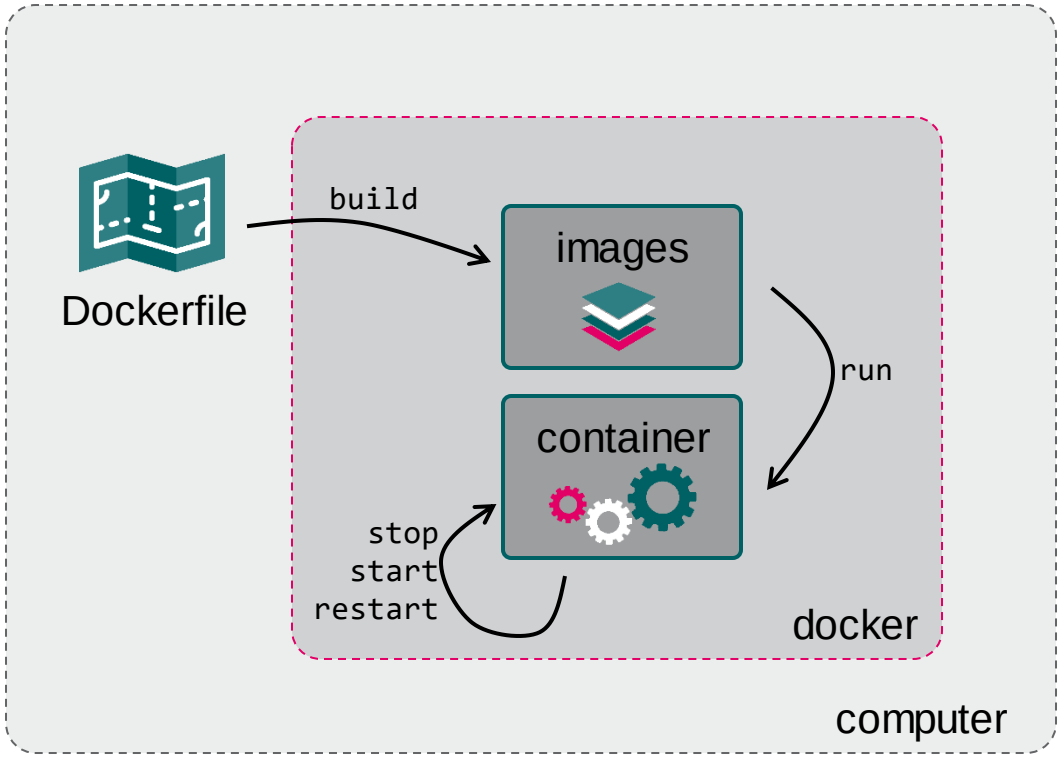
Putters:

`localhost:5000/saveJson`	save JSON File to database

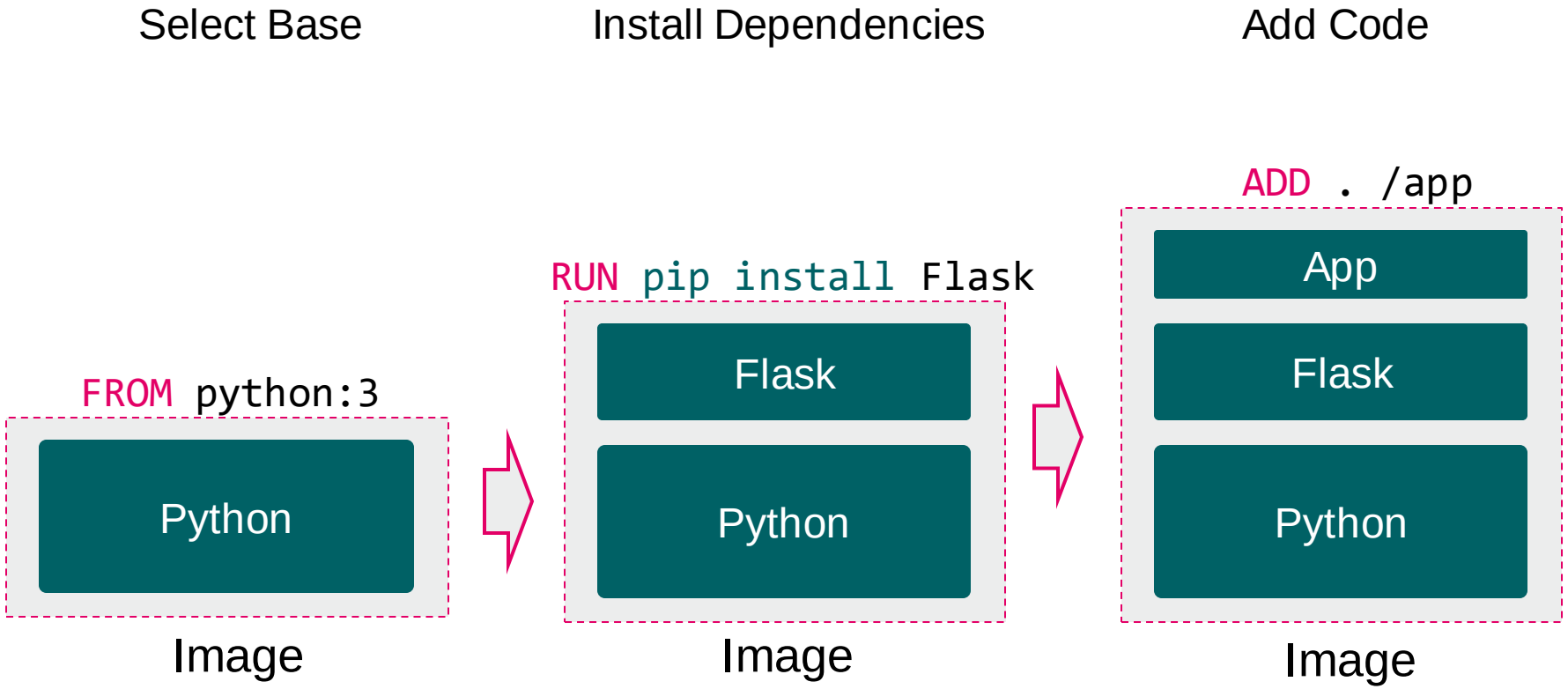
Our Flask Server

```
import random
import os
from flask import Flask
from flask import request
from pymongo import MongoClient

. . . Init Code

@app.route('/getID')
def getID():
    return str(ID)

@app.route('/saveJson', methods = ['POST'])
def saveJson():
    . . . Code save MongoDB

@app.route('/getJson', methods = ['GET'])
def getJson():
    . . . Code query MongoDB

if __name__ == "__main__":
    app.run(host='localhost', port=5000)
```

Docker Life Cycle



Assembling Images



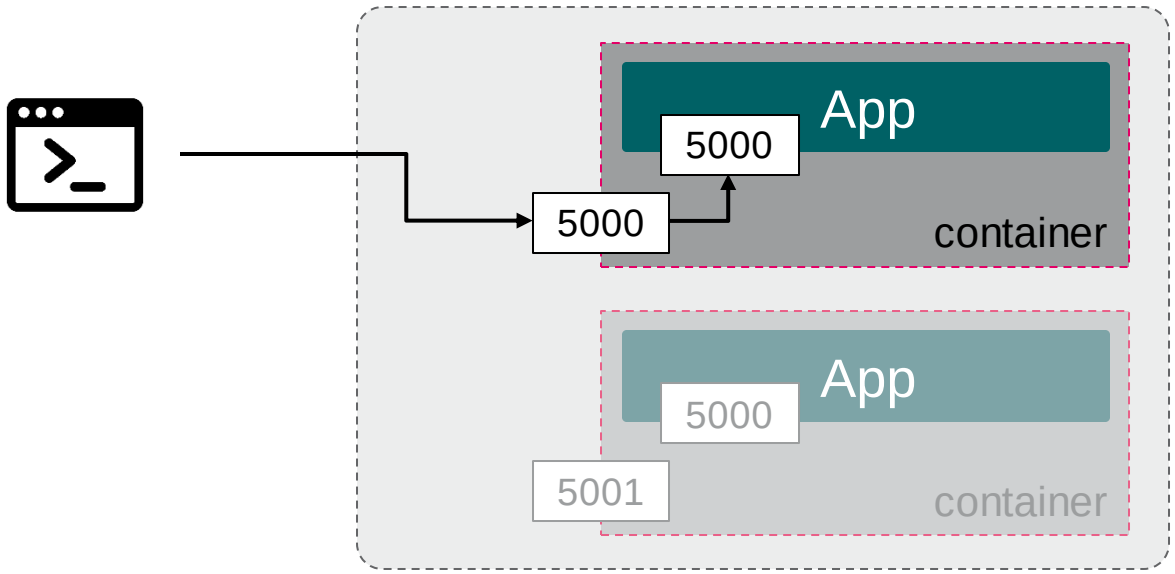
Dockerfile

```
1 FROM python:3
2 RUN pip install pymongo
3 RUN pip install Flask
4 ADD . /app
4 WORKDIR /app
5
6 CMD ["python", "main.py"]
```

```
docker build -t flaskserver:latest .
```

Run a Container

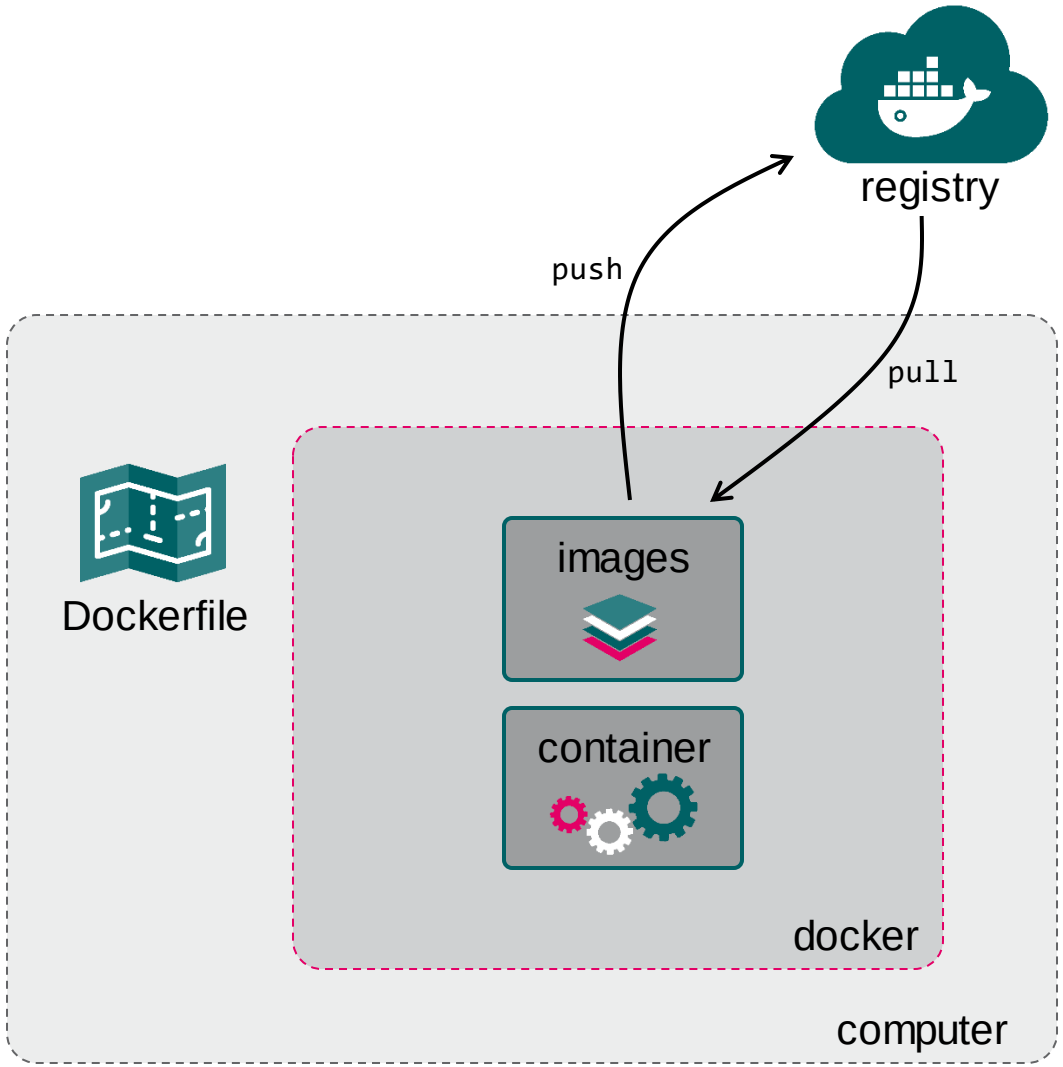
```
docker run -p 5000:5000
           flaskserver:latest
```



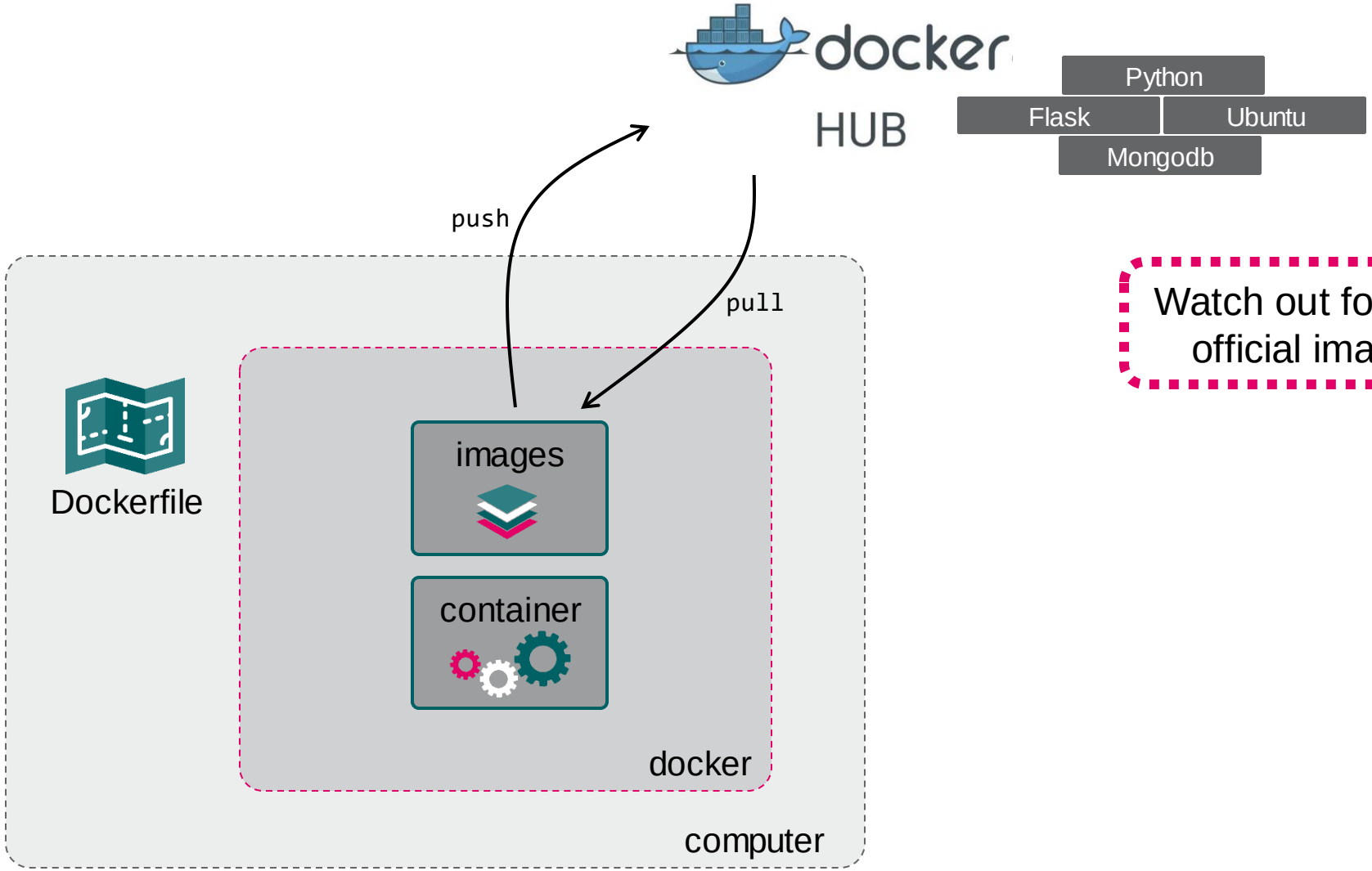
Flask Demo

```
docker build -t flaskserver:latest .
docker run -p 5000:5000 flaskserver:latest
```

Pulling and Pushing

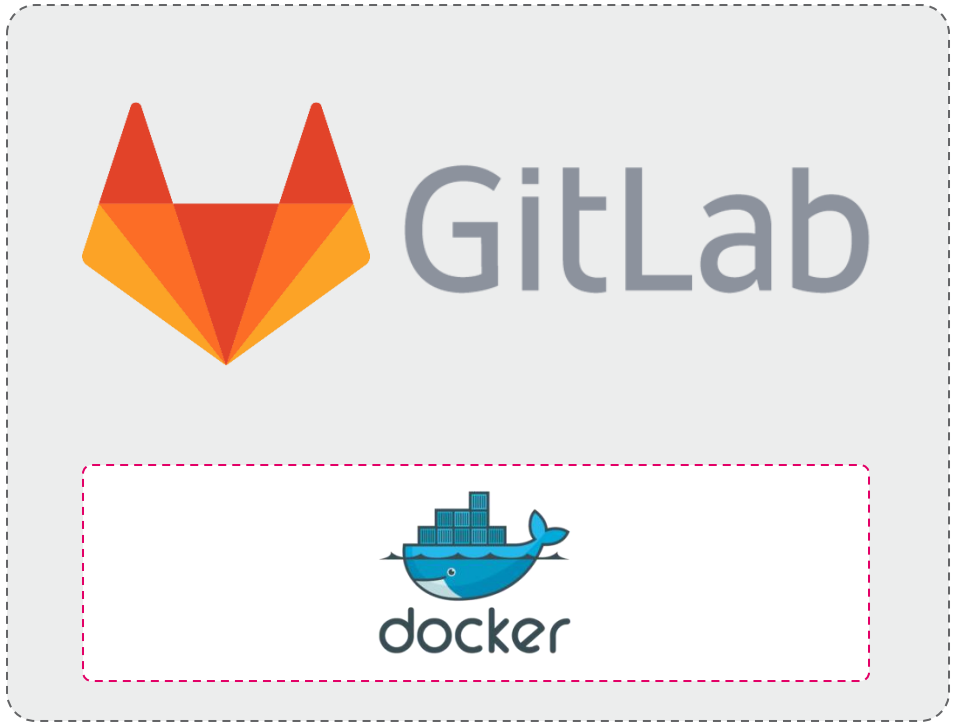


Docker Hub



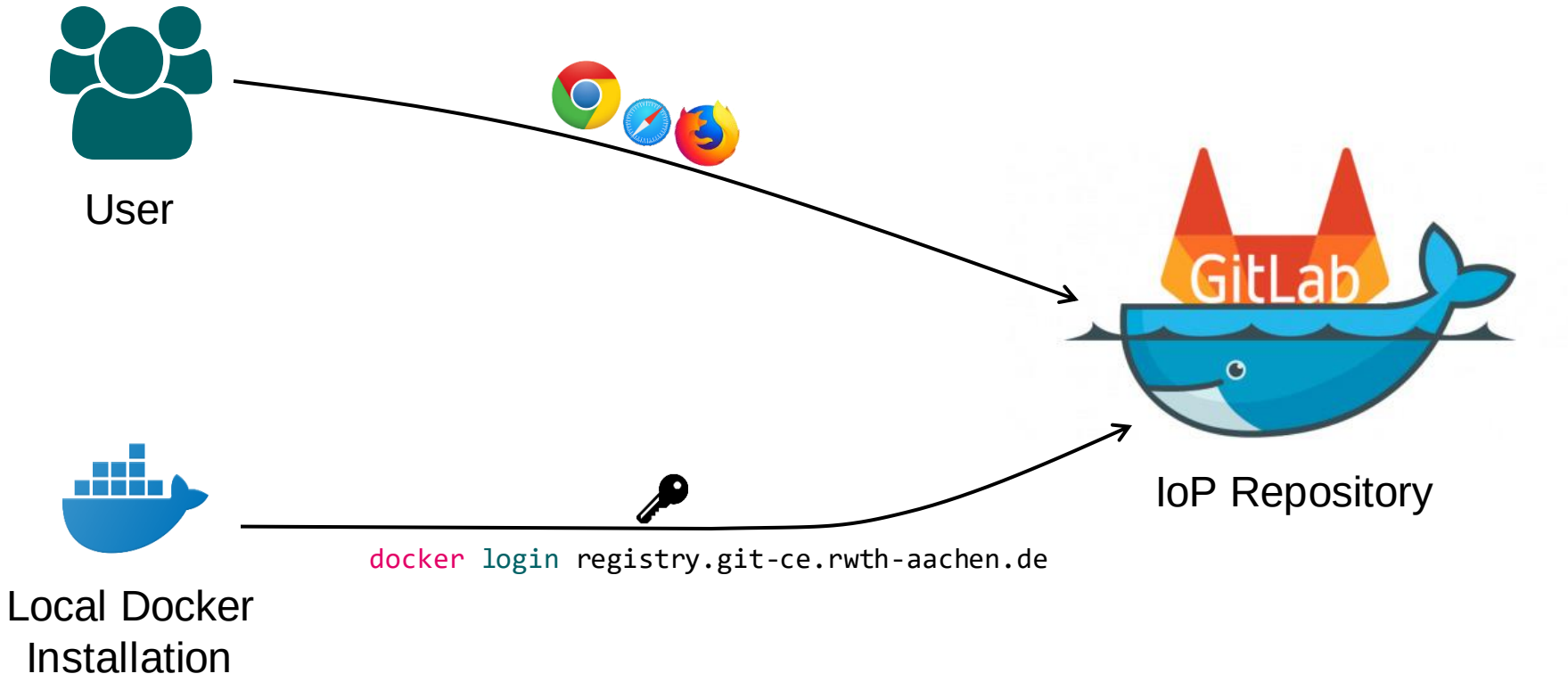
Watch out for non-official images

GitLab‘s Repository



- Docker Integration in GitLab
- Login using Single-Sign-On

Usage


User
GitLab
IoP Repository
Local Docker Installation
docker login registry.git-ce.rwth-aachen.de

GitLab Demo

```
export IOP=registry.git-ce.rwth-aachen.de
          /iop/infrastructure/dockerandk8smicrotraining/
           flaskserver:latest

docker login registry.git-ce.rwth-aachen.de

docker tag flaskserver:latest $IOP

docker push $IOP

docker rmi $IOP flaskserver:latest

docker pull $IOP
```

Our Flask server again

```
import random
import os
from flask import Flask
from flask import request
from pymongo import MongoClient
app = Flask(__name__)

MONGOHOST = os.getenv("MONGOHOST", "127.0.0.1")
MONGOPORT = os.getenv("MONGOPORT", "27017")

@app.route('/getID')
def getID():
    return str(ID)

@app.route('/saveJson', methods = ['POST'])
def saveJson():
    . . . Code save MongoDB

@app.route('/getJson', methods = ['GET'])
def getJson():
    . . . Code query MongoDB

if __name__ == "__main__":
    app.run(host='localhost', port=5000)
```

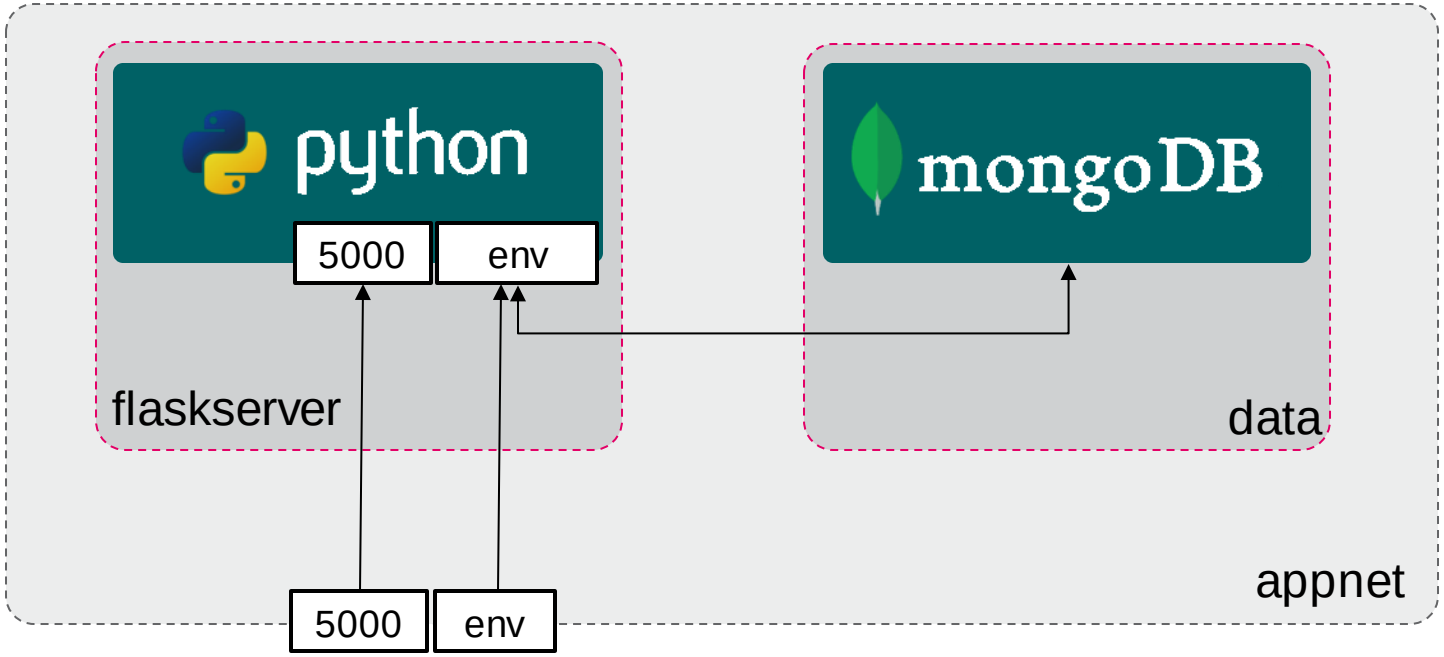
Add a network


```
docker network create appnet
```

Add a Mongodb


```
docker run --name data --network appnet mongo:latest
```

Run our Flask server




```
docker run -p 5000:5000
           --network appnet
           --env=MONGOHOST=data
           flaskserver:latest
```

Helpertools

Curl

- Send requests to a webserver
- `curl www.google.de`

Shell scripts

- Don‘t repeat yourself
- Write a shell script that types for you

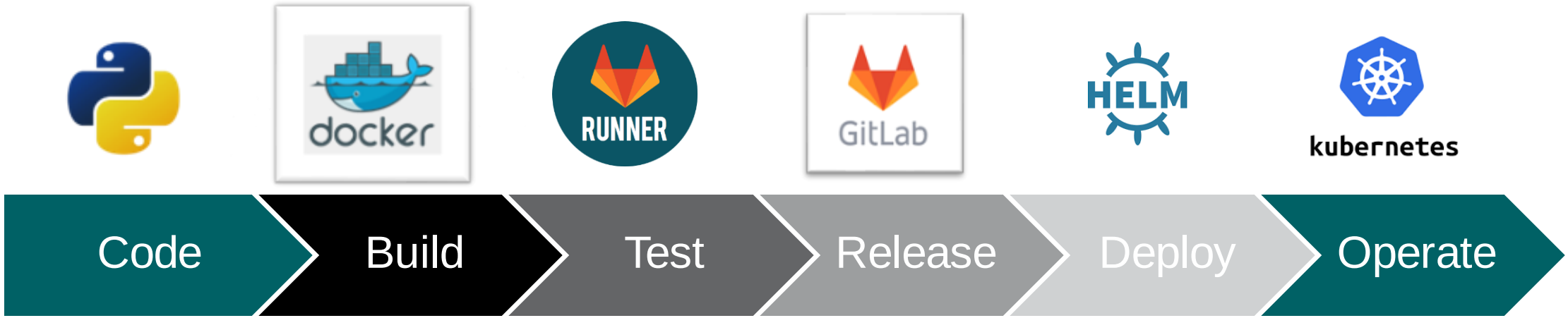
Mongodb Demo

```
docker network create appnet
```

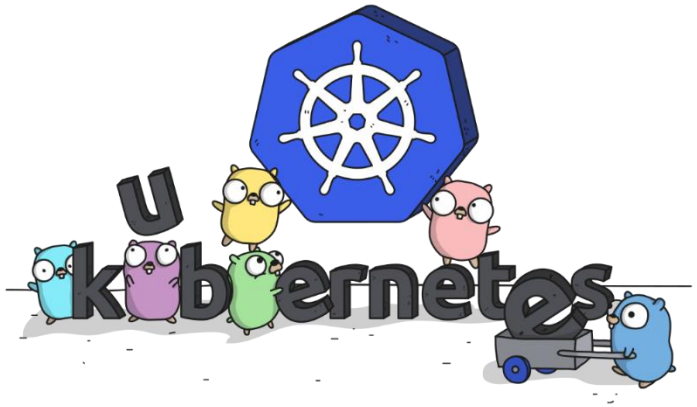
```
docker run --name data
           --network appnet
           mongo:latest
```

```
docker run -p 5000:5000
           --network appnet
           --env=MONGOHOST=data
           flaskserver:latest
```

Complete DevOps Cycle



Tutorial



Summary

- We built a small Cloud native webapp that basicly runs everywhere
- The Python application is stateless and thus easily scalable
- Possbility to switch between different databases (e.g local development database and production database)
- Decoupled configuration from code

⇨ **This is now a Kubernetes Cluster ready Application**

Thanks for listening

Code, Slides and Videopresentation can be found here:
https://github.com/rwth-iop/DockerMicrotraining

RocketChat: #docker-and-kubernetes