## LINKS

Alle Code-Beispiele findet ihr unter:

- https://github.com/Lhdang88/cloud_computing_ws

## HELLO WORLD IN NODEJS

## HELLO WORLD IN NODEJS

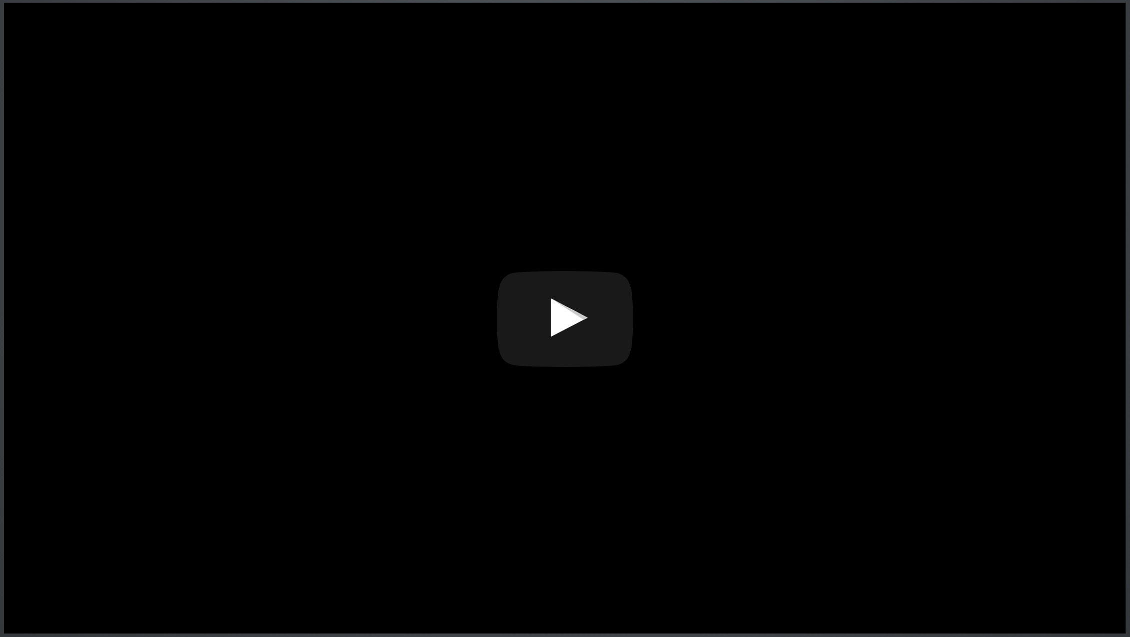
in your Shell / Terminal (Ubuntu: CTRL + Alt + T)

```
cd into-your-directory
git clone https://github.com/Lhdang88/cloud_computing_ws.git
cd cloud_computing_ws
git checkout hello_world
```

## HELLO WORLD IN NODEJS

```
//install dependencies
npm install
//start the application with
npm start
//stop the application with CTRL+C
```

## WHAT IS DOCKER



## DOCKER DEMO

## PULL & RUN SOMEONE'S DOCKER IMAGE

https://hub.docker.com/_/mongo/

```
# pull and run mongo-db a NoSQL DB
docker run --name my-mongodb-name -p 27017:27017 -itd mongo
# confirm with
docker ps -a
# try to connect with a MongoDB Client
```

## KONZEPT: APP-CONTAINERISIERUNG

## DOCKERIZE YOUR APPLICATION

In your app-root-directory, create a file named 'Dockerfile' Dockerize your app: Put this content in your Dockerfile

```
# Base OS from Dockerhub
FROM alpine:latest
# update software repository and install nodejs
RUN apk add --update nodejs
# copy local data into image
ADD . /myApp/
# install packages
RUN cd /myApp && npm install
# set working directory
WORKDIR /myApp
# start command
CMD npm start
```

## BUILD & RUN YOUR-OWN IMAGE

API docu:

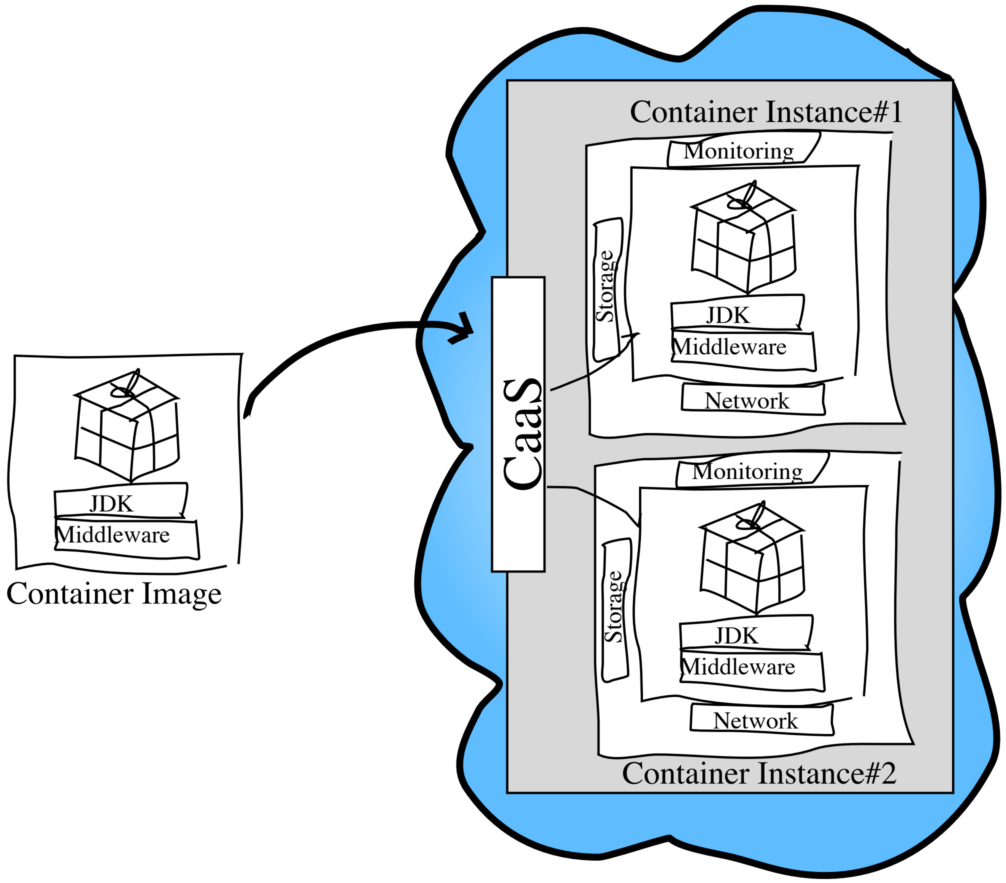
https://docs.docker.com/engine/reference/commandline/

```
cd into-your-app-directory
# build the docker image with docker build
# if you need help: execute 'docker [Command] help'
docker build -t your-image-name:your-tag-name .
#confirm with
docker images
# run the docker image with docker run
docker run --name your-container-name -p 80:3000 -itd your-image-
# confirm with
docker ps -a
```

## PUSH IMAGE TO A REPOSITORY

```
# re-tag it, to reference my docker-hub repository
docker tag your-image:your-tag lhdang88/dhbw:hello_world
# push the image to my repository on 'Docker-Hub'
docker push lhdang88/dhbw:hello_world
```

## BRING-YOUR-OWN-CONTAINER



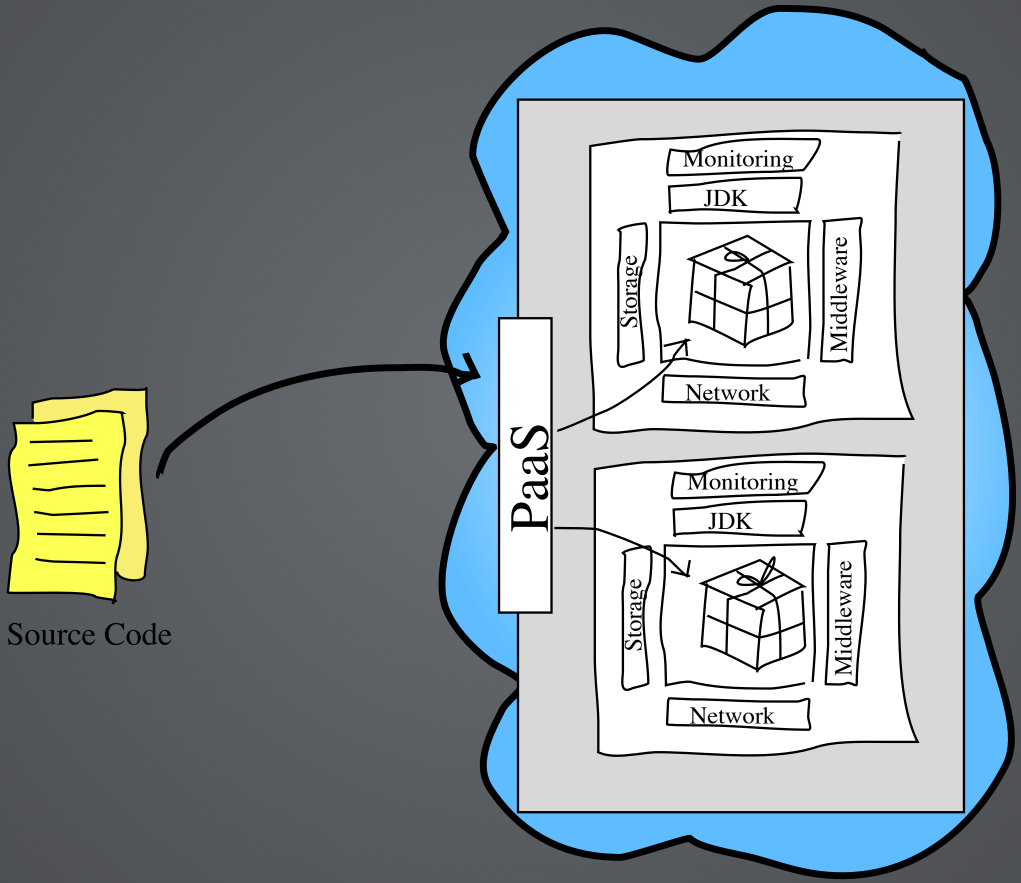
## HANDS-ON: MICROSOFT AZURE + KUBERNETES

Container-as-a-Service (CaaS)

## CLOUD FOUNDRY

- "Bring-Your-Own-Code"-Prinzip: Container-Image wird durch ein "Buildpack"-script gebaut
- Container-Updates werden durch die Plattform automatisch durchgeführt, bspw. Security-Patching, Runtime-Updates (siehe Verantwortlichkeit PaaS)
- Plattform startet/pausiert/rebootet/terminiert Container eigenständig
- Container = "Cell"

## "BRING-YOUR-OWN-CODE"



## Ein bisschen PaaS-Magic!

## PUSH YOUR OWN CLOUD FOUNDRY APP

create "manifest.yaml" in your app-root-directory with this content:

```
---
applications:
- name: dhbw-hello-world
  memory: 512M
  disk_quota: 512MB
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack
  command: node index.js
```

## PUSH YOUR OWN CLOUD FOUNDRY APP

```
cd into-your-app
# log into cloud Foundry, with the provided credentials
cf login -a the-cloud-foundry-endpoint
# push your app
cf push -f manifest.yaml
```