# Docker

Kurzgesagt - In a Nutshell

# About Me

- Apprenticeship System Admin/Engineer
- CS Degree at BFH
- Consulting Years
  - Public Cloud Provider, Telco Provider, Medtech
  - DevOps / Automation Engineer, Software Engineer
  - .NET Core, Java Spring Boot and a lot of Tooling
- Securiton
  - Intrusion Alarm System
  - Software Engineer
  - Go, Rust, and a lot of Tooling

Christian Nydegger
LinkedIn

# Intro

Goals of today's lecture

- You can classify Docker
- You know the basic concepts of Docker
- You can apply those concepts
- You know about Docker-Compose

# What are Containers?

# What are Containers?

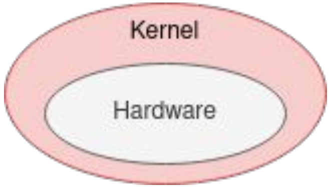
Terminology

- Kernel
- Operating System
- Process

# What are Containers?

Kernel

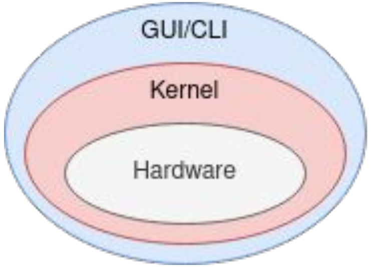
- Abstracts Hardware
- Offers Well-Defined Interface
- Linux Kernel, Unix Kernel, NT Kernel

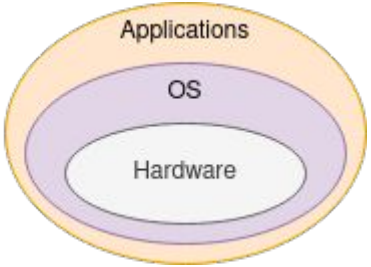


# What are Containers?

Operating System

- Extended User Interface (GUI and or CLI)
  - Bash, PowerShell, Tcsh
- Software Services and Utilities
  - Top, Task Manager, Activity Monitor

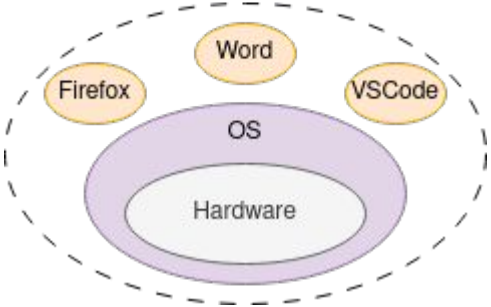




# What are Containers?

Process

- A running program
- Keeps things separate
    - Isolated memory space



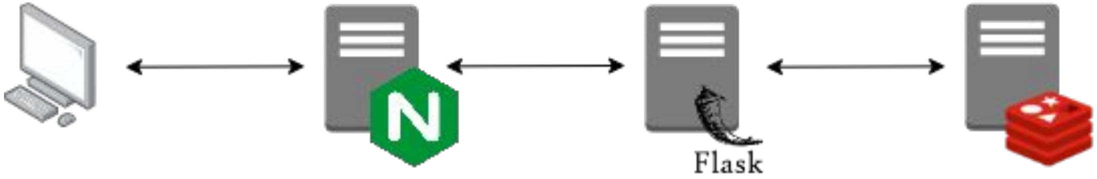
# What are Containers?

**Processes** are managed by an **operating system**, which uses its **kernel** to abstract hardware complexity and efficiently run these processes on the physical **hardware**.

# What are Containers?

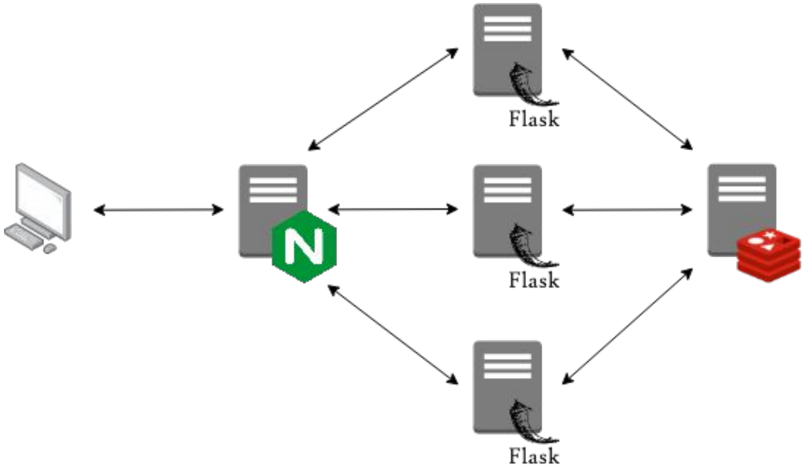
Project Setup

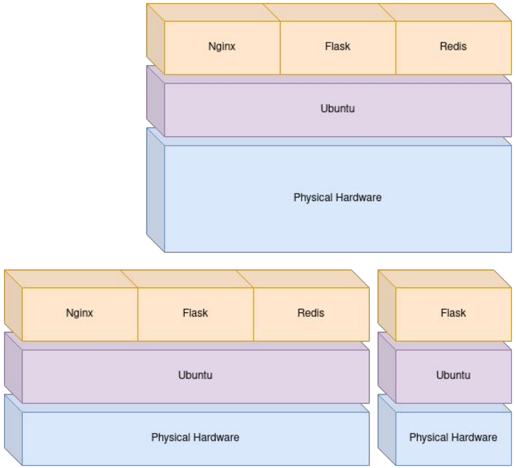
- Simple Web Application Setup
- Nginx as Load Balancer
- Flask to implement Rest Service
- Redis as Persistence Layer



# What are Containers?

## Bare-Metal-Deployment

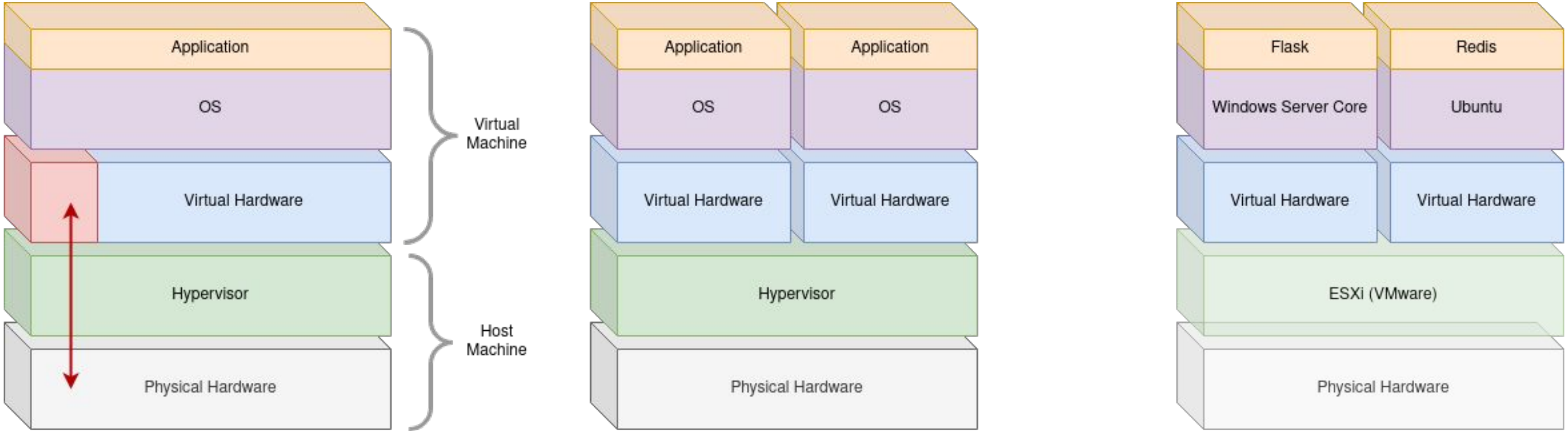




# What are Containers?

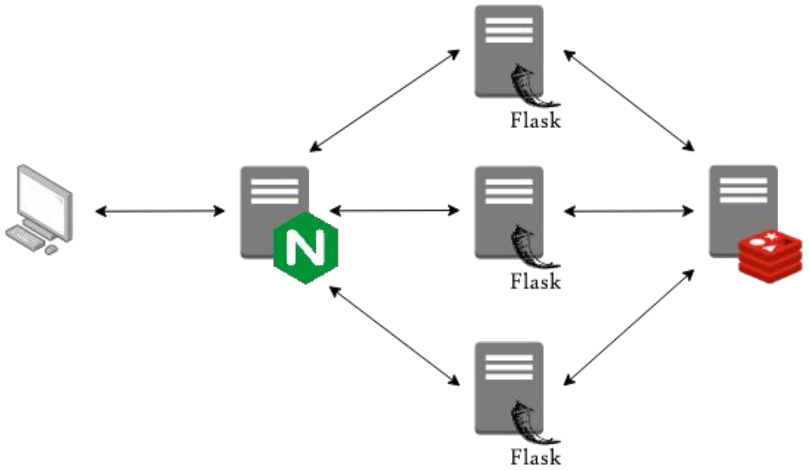
"Traditional" Virtual Machine

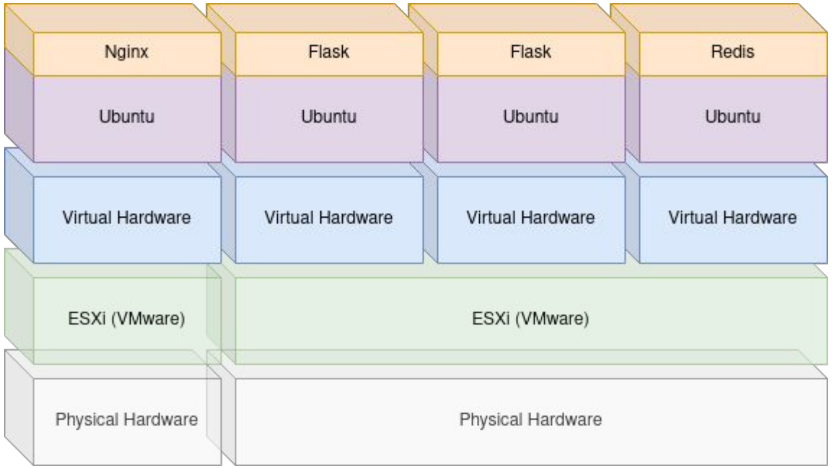
- Isolated Instance with its own Hardware and OS



# What are Containers?

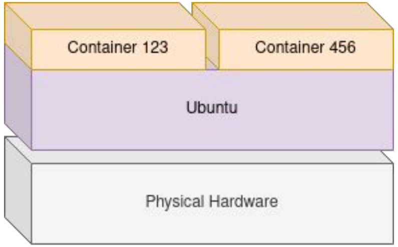
## “Traditional” Virtual Machine

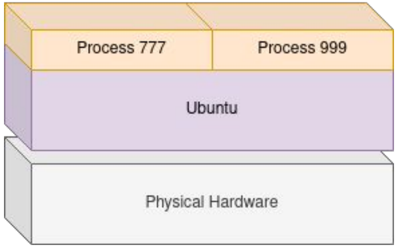




# What are Containers?

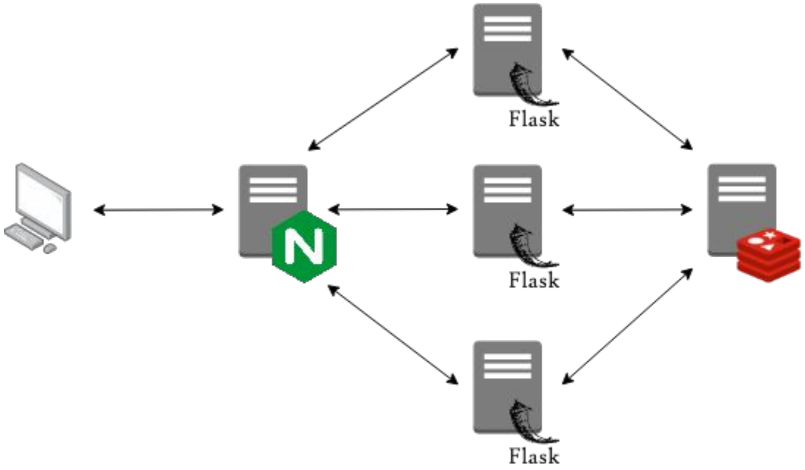
- Linux Process
  - A running program
  - Isolated memory space and restricted privileges
- Linux Container
  - A process or group of processes
  - Further isolated by private root-fs, process namespace etc.
  - Enabled by kernel features like cgroups or namespaces
  - OS-Level or Kernel-Level virtualization

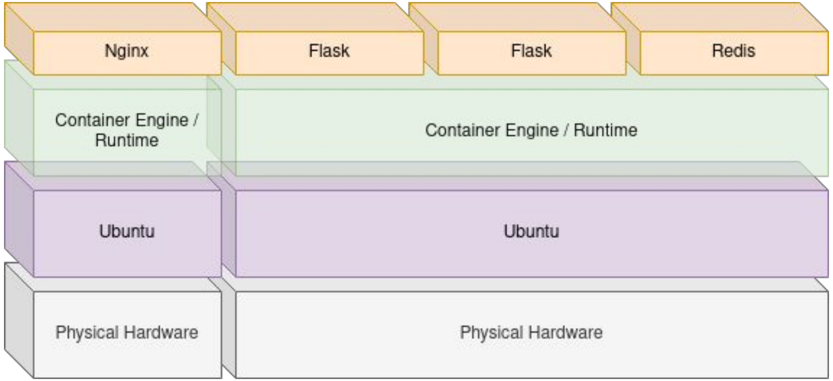




# What are Containers?

## Container





# What are Containers?

## VM vs. Bare Metal Machine

- Improved Resource Economy
- Horizontal vs. Vertical Scalability
  - Memory slots are limited :)
- Virtualization Overhead
  - Loss of performance
  - Lots of duplications

## Container vs. VM

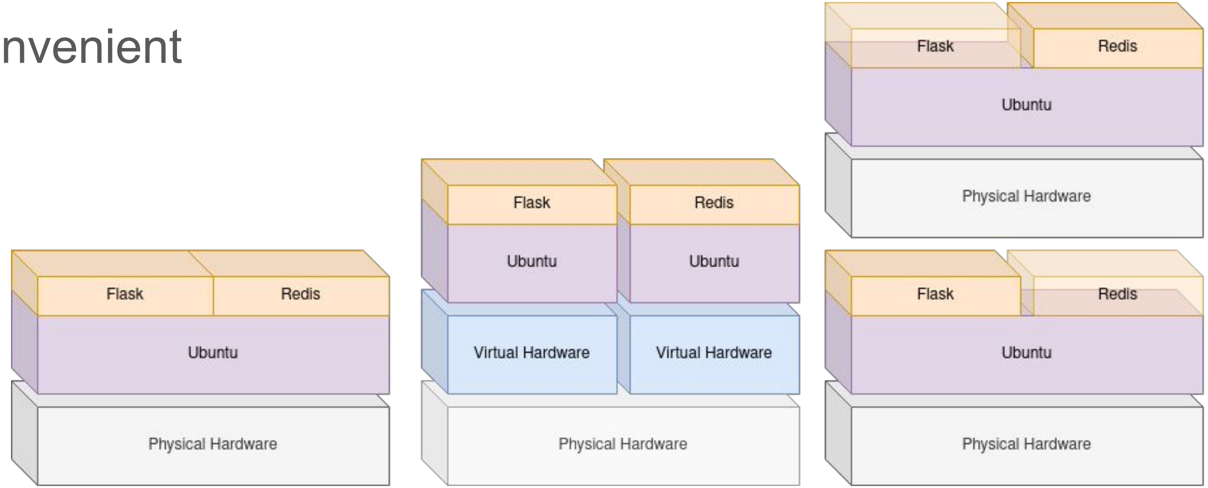
- Less overhead
  - Isolation not by virtualized hardware
- Increased Performance*
  - Direct hardware access
- Smaller Footprint*

*there is a tendency

# What are Containers?

## Bare Metal vs. Virtual Machine vs. Container

- More flexibility
- More efficient
- More convenient



# Break

If there are any questions, feel free to approach me

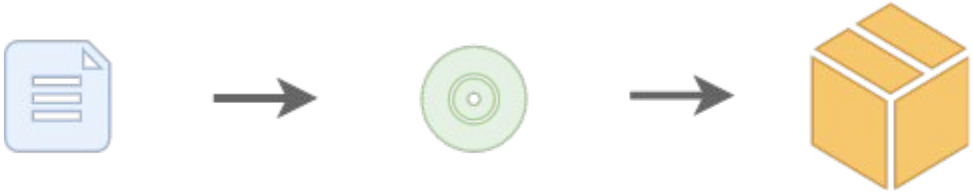
# Docker Intro

- Set of Tools to work with Containers
- Alternatives
  - Podman
  - LXC
- Why Docker?
  - Well established
  - Big Community
- Terminology
  - Container
  - Image
  - Dockerfile
  - Registry

# Docker Intro

Container

- Runtime instance of a Docker Image
- Can be compared to an Object

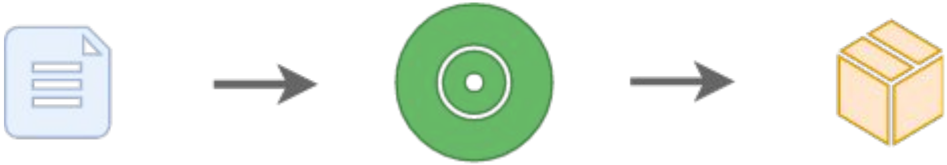


# Docker Intro

## Image

Docker images are the basis of containers. An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a union of layered filesystems stacked on top of each other. An image ***does not have state and it never changes***.

- Blueprint to instantiate Containers from
- Can be compared to a Class

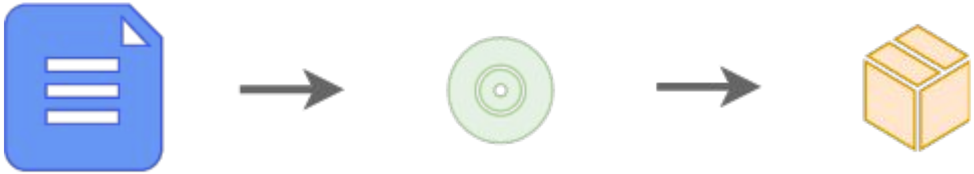


# Docker Intro

## Dockerfile

A Dockerfile is a text document that contains all the commands you would normally execute manually in order to build a Docker image.

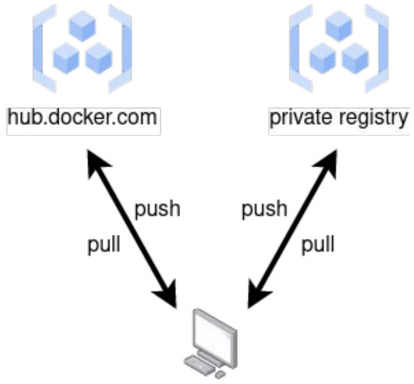
- Instructions for Docker to build Image
- Declares how the Docker Image looks like
- A human readable representation of the Docker Image



# Docker Intro

## Registry

- Hosts Docker Images
  - Can be searched by *docker search*
- Default is hub.docker.com
  - Can be accessed by browser
- Private registry can be setup
  - Available as an Image itself



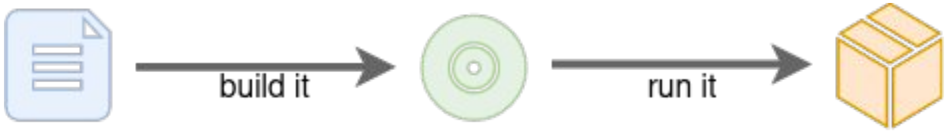
# Docker Intro

Demo

# Docker Intro

Process

- Write Dockerfile
- Build Image from it
- Instantiate Image to run Container
- Push Image to Registry if desired



# Docker Intro

## What not to do

- Treat a Container like a Virtual Machine
- Upgrade Containers
    - internals
    - Upgrade Dockerfile and rebuild Image instead
- Reuse Containers
    - Run a new container instead
    - If a container is gone, let it rest
- Run multiple Services in on Container
    - Run a container for each service instead

# Break

If there are any questions, feel free to approach me

What problem might occur with Docker?

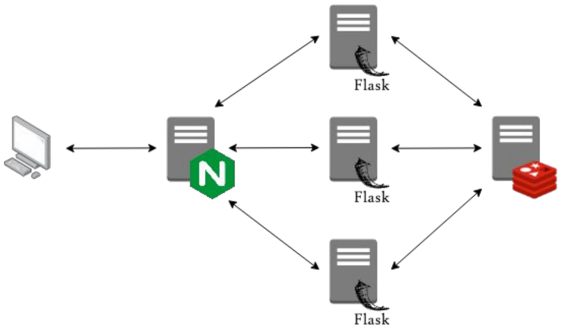
# Docker-Compose Intro

*Compose* is a tool for defining and running multi-container *Docker* applications.

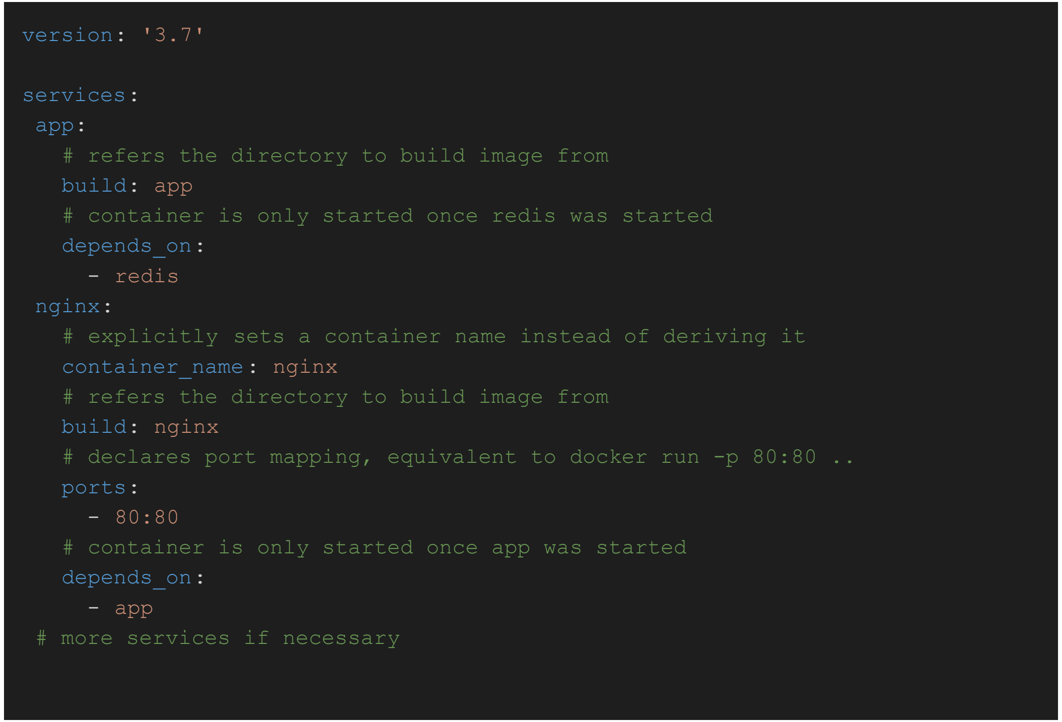
- Compose File
  - Instructions for Compose to configure and run individual Services
- Similar command set as Docker
  - Application level:
    - Up, Down, Build, ..
  - Container level:
    - Start, Stop, Run, ..

# Docker-Compose Intro

- CLI
  - Instantiate individual Containers with *docker run*
  - Very inconvenient and error prone
- Script
  - Essentially wrap individual commands in a bash script
  - Technically possible
  - Scripting vs. declaring
- Compose File
  - Declare your multi container application

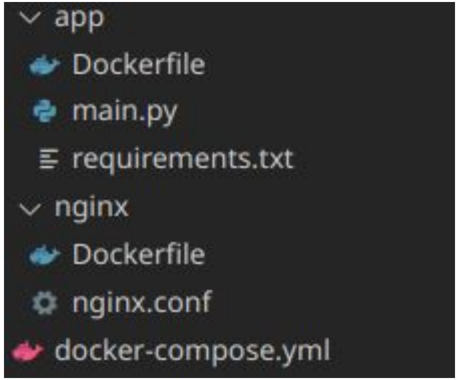


# Docker-Compose Intro



```
version: '3.7'

services:
 app:
   # refers the directory to build image from
   build: app
   # container is only started once redis was started
   depends_on:
     - redis
 nginx:
   # explicitly sets a container name instead of deriving it
   container_name: nginx
   # refers the directory to build image from
   build: nginx
   # declares port mapping, equivalent to docker run -p 80:80 ..
   ports:
     - 80:80
   # container is only started once app was started
   depends_on:
     - app
 # more services if necessary
```



```
∨ app
   Dockerfile
   main.py
   requirements.txt
∨ nginx
   Dockerfile
   nginx.conf
  docker-compose.yml
```

# Docker-Compose Intro

Demo

# Your Task

## Dockerize a small web application

The goal is to implement a tiny web service similar to the examples discussed during the lecture. It can be a simple ping or something a bit more sophisticated. The only requirement for the service is that the persistence layer is used. The example discussed during the lecture implemented a simple hit count stored in a redis store.

Other requirements are:

- The rest service and all its dependencies **must** be packed in a Docker Image.
- The redis store **must** be run as a container
- The application **can** be managed with docker-compose
- It is **recommended** to us redis and flask but **not a must**

Deliverables:

- Create one GIT-Repository per group and hand-in at least one solution

# Q&A