Docker

Kurzgesagt - In a Nutshell

Intro

Goals of today's lecture

- You can classify Docker
- You know the basic concepts of Docker
- You can apply those concepts
- You know the basic concepts Docker Compose
- You can apply those concepts
- You know about Docker Multi-stage-builds
- You know about VSCode Devcontainers

What are Containers?

What are Containers?

A container is a ***standard unit of software*** that ***packages up code and all its dependencies*** so the application runs quickly and reliably from one computing environment to another.

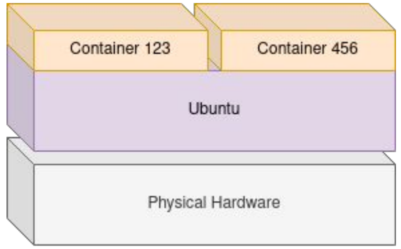
- Wait.. that just sounds like an ordinary software package?
  - Yes but it is so much more

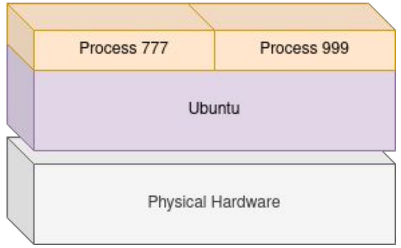
OS-level ***virtualization*** is an operating system (***OS***) paradigm in which the ***kernel*** allows the existence of ***multiple isolated*** user space ***instances, called containers***..

- So it is a virtual machine?
  - Yes, in a way

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





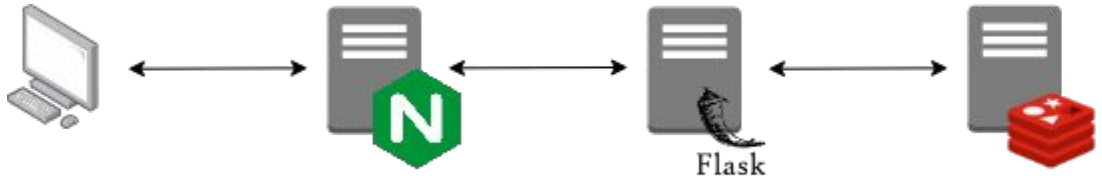
What are Containers?

Project Setup

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

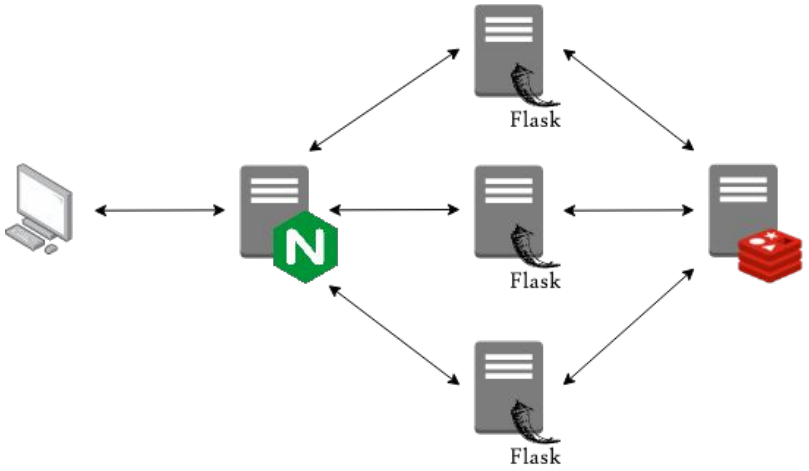
Constraints

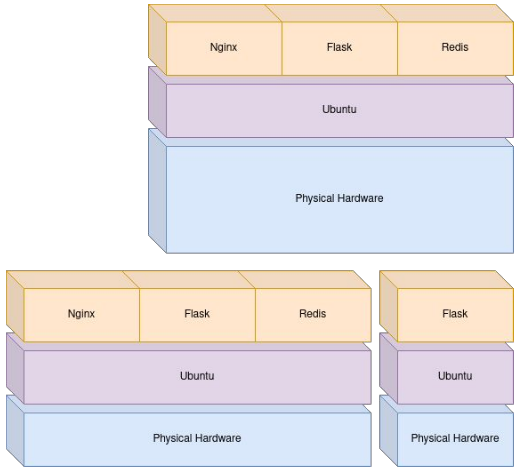
- 99% workload is on Flask Service



What are Containers?

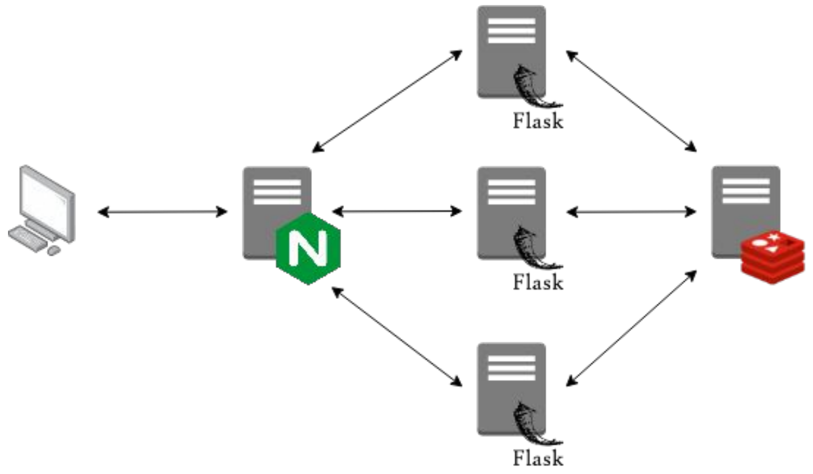
Bare-Metal-Deployment

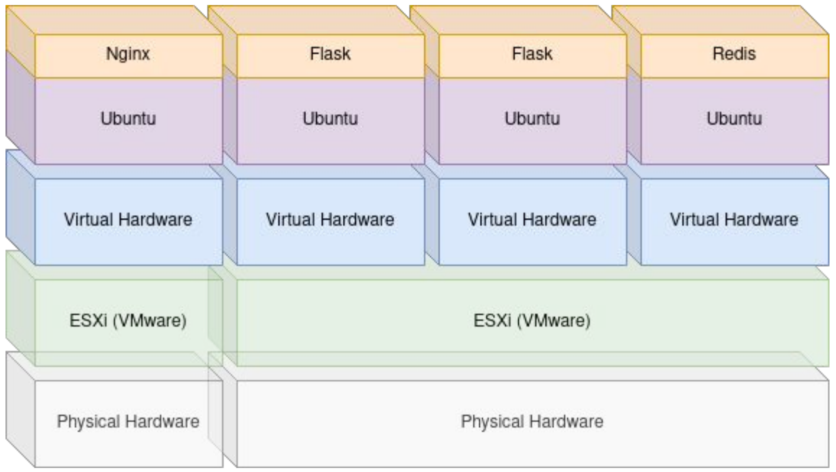




What are Containers?

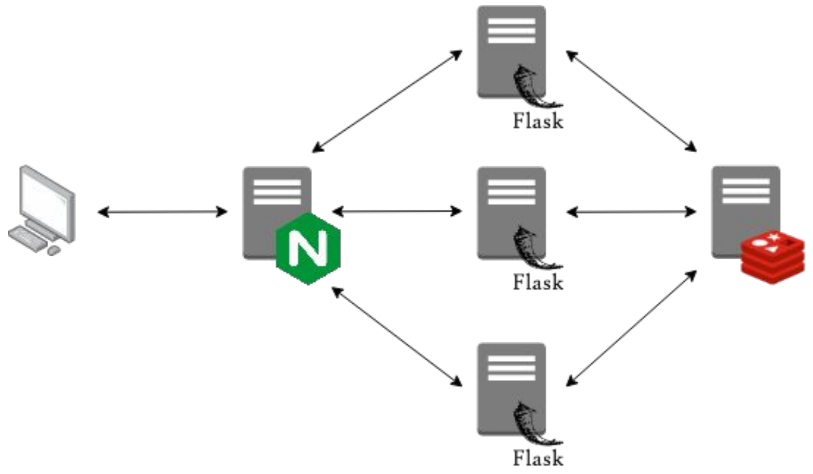
“Traditional” Virtual Machine

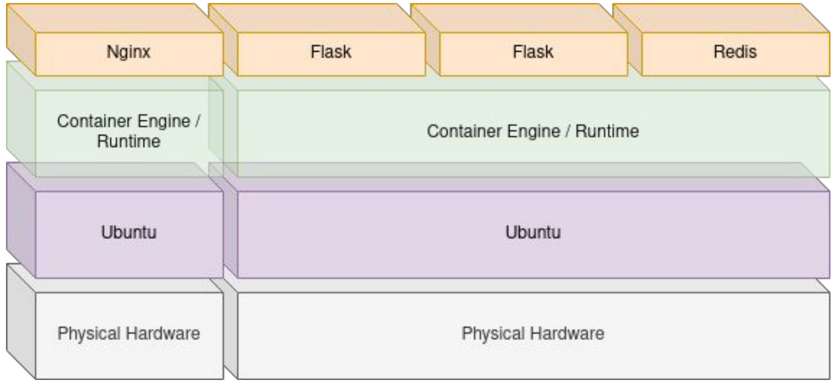




What are Containers?

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

Break

If there are any questions, feel free to approach me

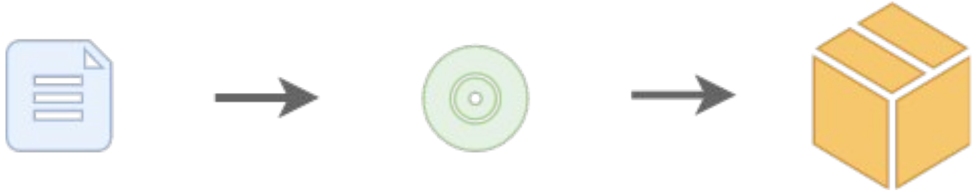
Docker Intro

- 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

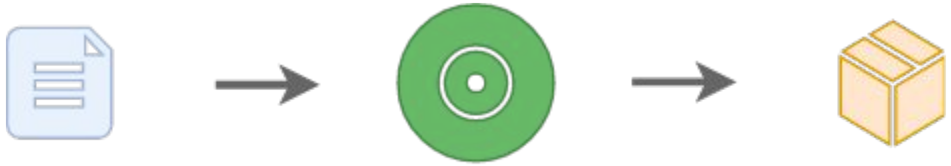


Docker Intro

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

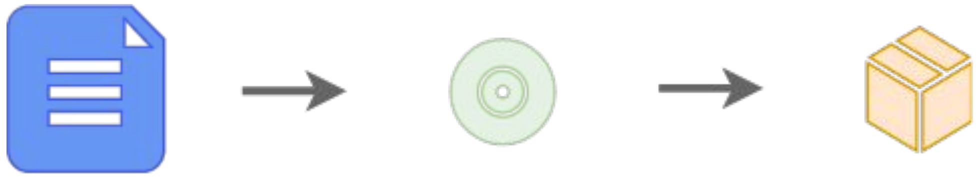


Docker Intro

Dockerfile

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

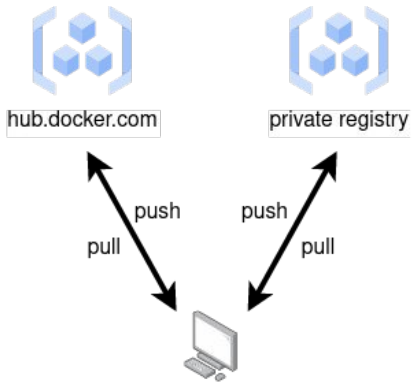
- Instructions for Docker to build Image
- Declares how the Docker Image looks like
- A human readable representation of the Docker Image



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



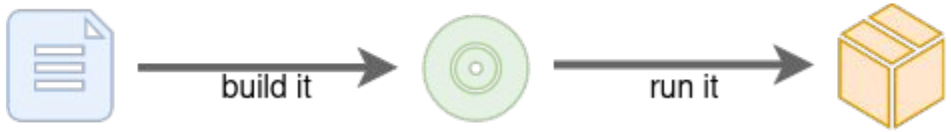
Docker Intro

Demo

Docker Intro

Process

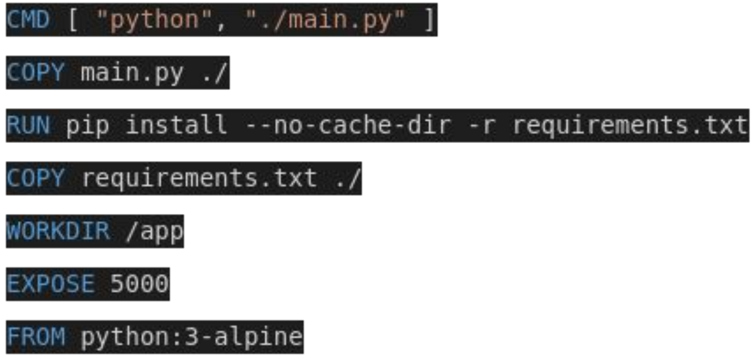
- Write Dockerfile
- Build Image from it
- Instantiate Image to run Container
- Push Image to Registry if desired



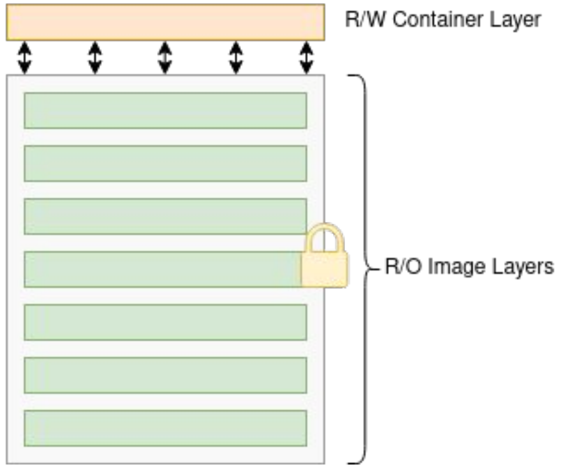
Docker Intro

Layer Concept

- Image consists of ReadOnly Layers
- Container ReadWrite Layer represents Container State



```
CMD [ "python", "./main.py" ]
COPY main.py ./
RUN pip install --no-cache-dir -r requirements.txt
COPY requirements.txt ./
WORKDIR /app
EXPOSE 5000
FROM python:3-alpine
```



Docker Intro

What not to do

- Treat a Container like a Virtual Machine
- Upgrade Containers
  - 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 a 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?

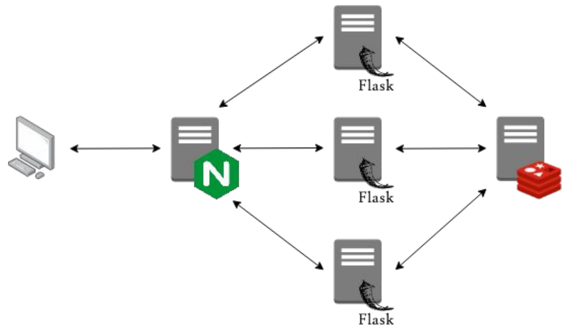
Docker Compose Intro

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 vs Docker-Compose
  - Plugin
  - Decorator Script around Docker CLI

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

Demo

Docker Multi-Stage-Build

Problem

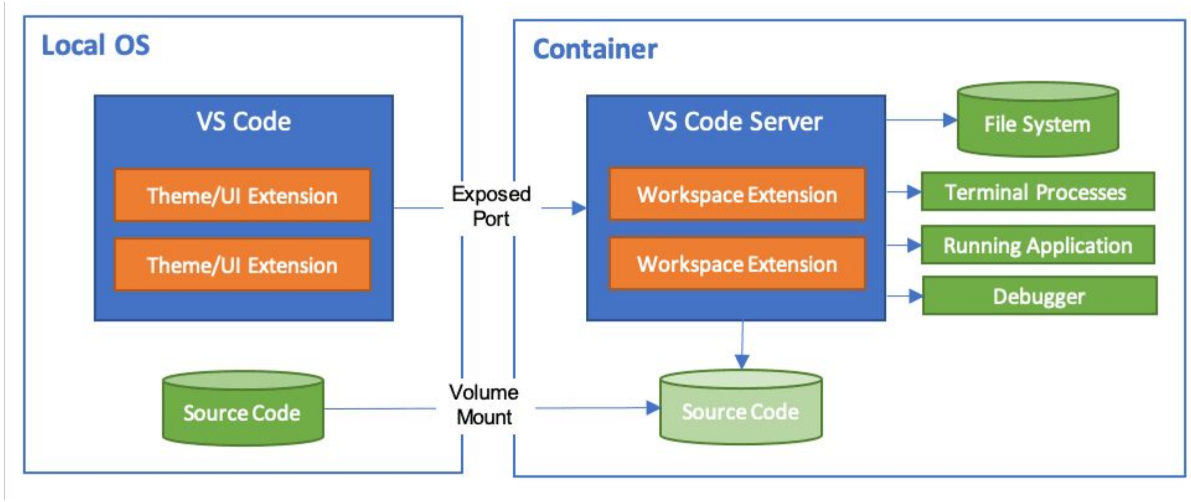
- Things needed to build application are often not needed in Prod Image
- Docker Images can become big

Solution

- Separate Dev & Prod Dockerfile
  - Known as “Builder Pattern”
- Docker Multi-Stage-Build

VSCode Devcontainer

- Visual Studio Code Plugin
- Toolchain wrapped into Docker Image



Q&A