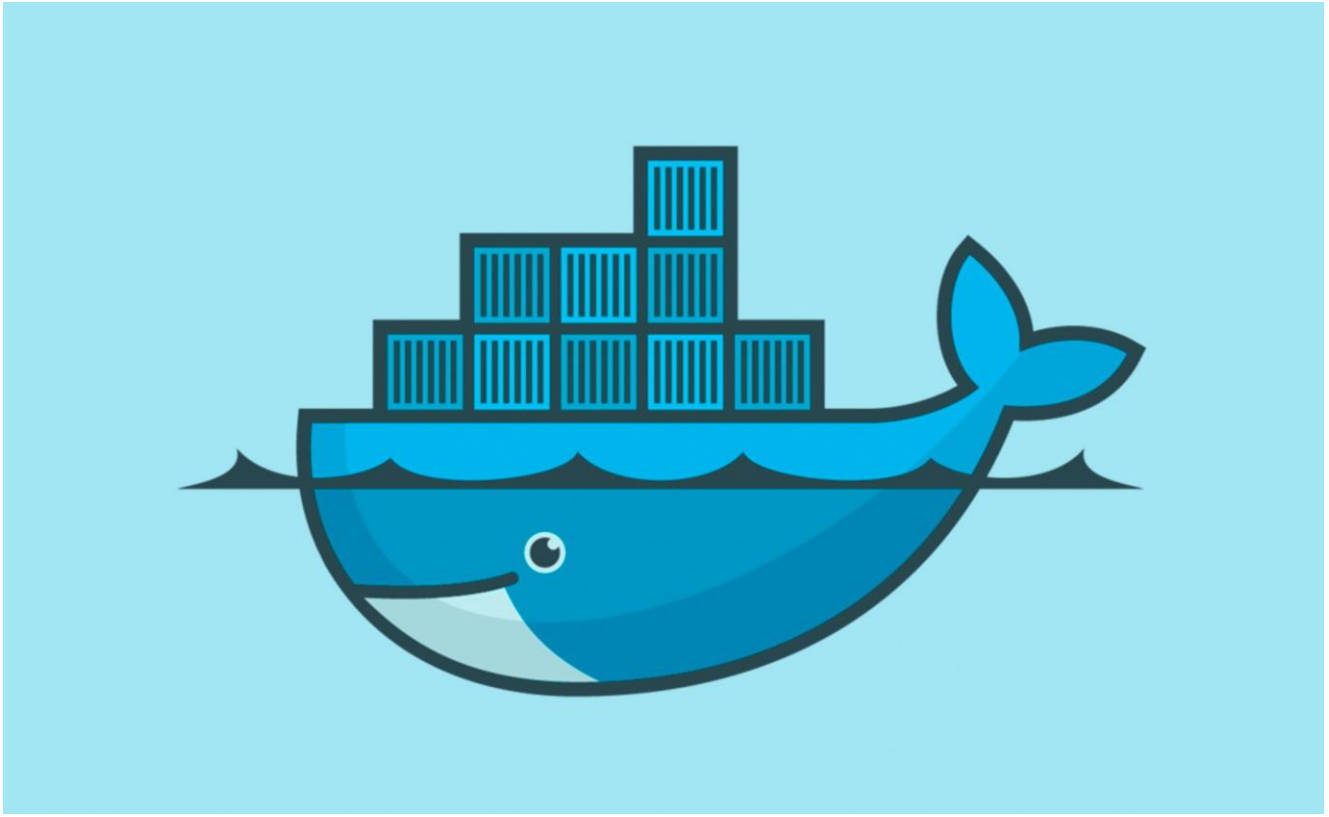
TAC-POT #1

DOCKER



Einführung in die Theorie

1. **Warum Docker?**
2. Docker vs. VM
3. Docker Workflow
4. Die wichtigsten Docker Befehle
5. Docker Compose

Hands-on: Deployment einer Angular-Anwendung

schlechte Erfahrungen mit dem **Deployment** von „nexHR“ bei Xenium

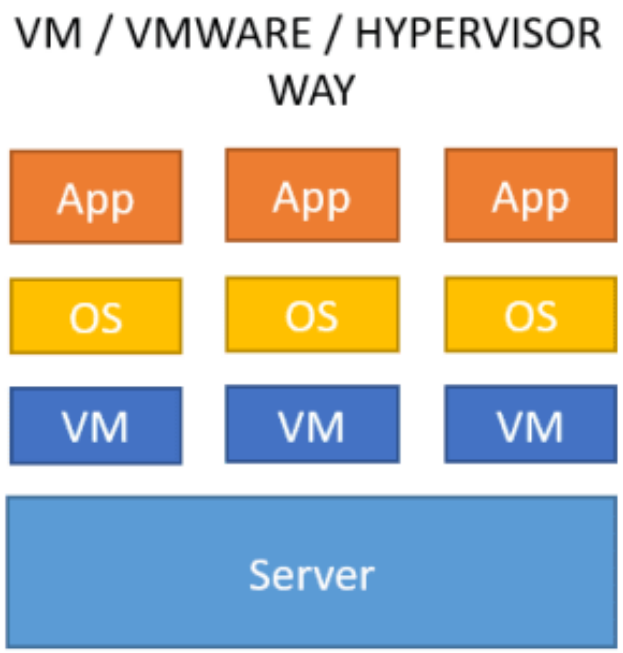
Überblick

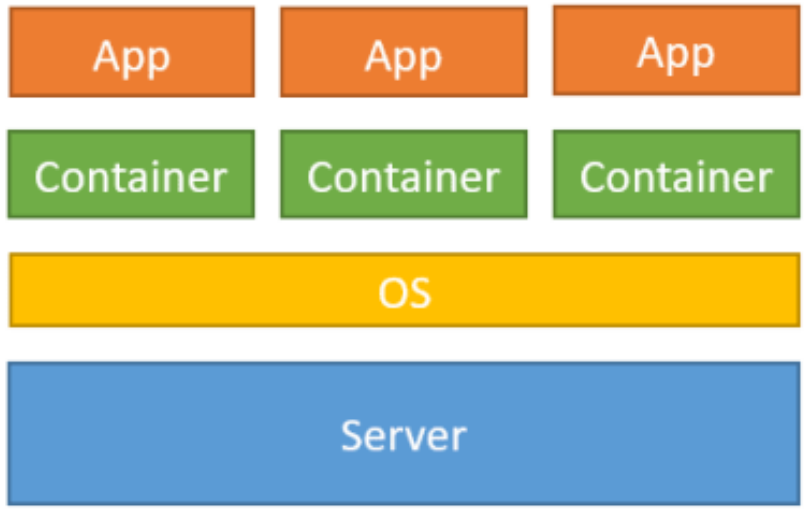
Einführung in die Theorie

1. Warum Docker?
2. **Docker vs. VM**
3. Docker Workflow
4. Die wichtigsten Docker Befehle
5. Docker Compose

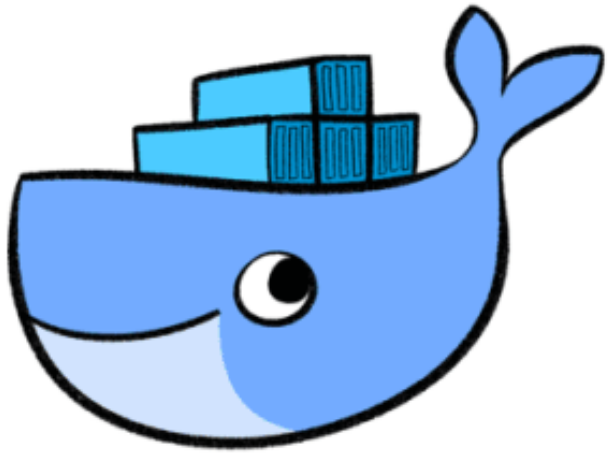
Hands-on: Deployment einer Angular-Anwendung

Docker vs. VM


VM / VMWARE / HYPERVISOR WAY
App
App
App
OS
OS
OS
VM
VM
VM
Server


DOCKER / CONTAINERS WAY
App
App
App
Container
Container
Container
OS
Server


vmware



Virtualisierung

- Abstraktion von IT-Ressourcen durch zusätzliche Ebene zw. Hardware & Anwendung
- ermöglicht verschiedene IT-Komponenten zu emulieren und virtuell bereitzustellen
- kein Unterschied für Anwender in der Nutzung

- IT-Komponenten, die sich virtualisieren lassen:
  - **Betriebssysteme**
  - **Applikationen**
  - Server
  - Speicher
  - Netzwerke
  - Desktopsysteme

Virtuelle Maschine (VM)
= in sich abgeschlossene Softwareeinheit auf einem Host, die gegenüber dem Betriebssystem eine echte Hardwareumgebung simuliert

Hypervisor / Virtual Machine Monitor
= abstrahierende Schicht zwischen realem Host-Rechner und der VM
= übernimmt oft die Aufgabe der Virtualisierungssoftware

z.B.: VirtualBox

Vorteile VM

+ verschiedene OS gleichzeitig auf der selben physischen Maschine

+ bessere Nutzung von Ressourcen (Bsp. Rechenzentrum)

+ Sicherheit durch Kapselung

+ günstigerer und vereinfachter Betrieb

Nachteile VM

- Effizienzverlust

- Overhead

- gegenseitige Beeinflussung gleichzeitig betriebener virtueller Maschinen

Docker
= freie Software zur Isolierung von Anwendungen mit Containervirtualisierung

Container enthalten alle notwendigen Pakete (dependencies, libraries, … = Image), um Applikationen zu installieren und transportieren

Vorteile

+ Booten deutlich schneller als VM (Sekunden anstatt Minuten)
+ verschafft Anwendungen Unabhängigkeit (leichter Transport, …)
+ geringeres Overhead (teilen sich OS)
+ Performant
+ Stabil

Nachteile

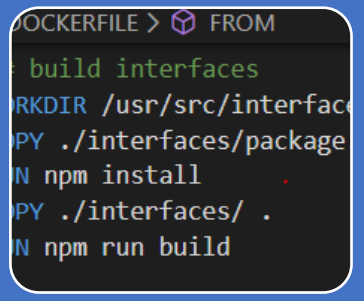
- Sicherheit (minimal schlechter als VM)

DOCKER

Baut auf dem Linux Kernel auf

Alle Container teilen sich gleiches OS, aber komplett getrennte Prozesse

→ Sicher aber auch ressourcensparend & schnell



DOCKERFILE

Einfache Textdatei mit Befehlen wie Programm aufgebaut wird

→ Bei Ausführung entsteht ein **Image**

CONTAINER

In diesen laufen die Images

Docker-Netzwerk → Kommunikation zw. Containern

Container können beliebig skaliert werden

Überblick

Einführung in die Theorie

1. Warum Docker?
2. Docker vs. VM
3. **Docker Workflow**
4. Die wichtigsten Docker Befehle
5. Docker Compose

Hands-on: Deployment einer Angular-Anwendung

Docker Workflow

„Normale Entwicklung“ mit Live Servern mit automatischer Aktualisierung

Schreiben von Docker Files für jeden Service

& Docker Compose File

Deployment mit **nur einem!** Docker Befehl

Einführung in die Theorie

1. Warum Docker?
2. Docker vs. VM
3. Docker Workflow
4. **Die wichtigsten Docker Befehle**
5. Docker Compose

Hands-on: Deployment einer Angular-Anwendung

Docker build –tag <name:version> <pathToDockerFile>
→ Erstellt ein Image, welches z.B. über DockerHub geteilt werden kann

Docker start/stop/restart <containerId | name>
→ Starte/Stoppt/Restarts einen bereits existierenden Container

Docker run –d –p <hostPort:containerPort> **--name** <containerName>
→ Startet den Container (create & start)

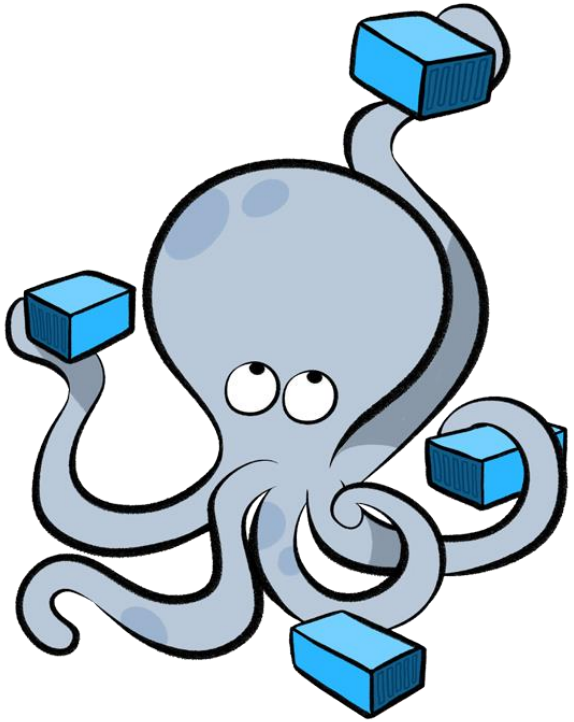
Docker ps –a
→ Zeigt alle Container

Docker rm <containerId |name> **--force**
→ Entfernt einen Container, mit force auch laufende

Einführung in die Theorie

1. Warum Docker?
2. Docker vs. VM
3. Docker Workflow
4. Die wichtigsten Docker Befehle
5. **Docker Compose**

Hands-on: Deployment einer Angular-Anwendung



Kombiniert verschiedene DockerFiles, definiert Umgebungsvariabeln wie Paswörter und führt diese gesammelt aus → Ein Befehl die gesamte App wird gebaut!

Befehle

Docker compose up/down *(--build <service>) –d*

Überblick

Warum Docker?

Einführung in die Theorie

Hands-on: Deployment einer Angular-Anwendung + node.Js Server