# But it works on my machine!

Leon Gjinovci, gjle1011@h-ka.de, 2025

# Developers Law

“any code that works perfectly on your machine will inevitably fail on everyone else’s”

(made it up btw)

# Warum Docker

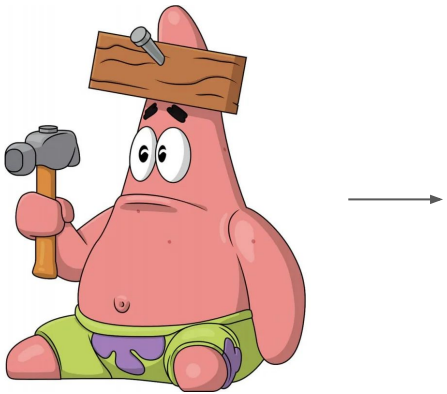
weitestgehend verwendet von “professionellen” Teams

-> wird gebraucht -> geld

einfaches development + deployment

# Was macht Docker so gut?

Wir nehmen unseren Code und schieben ihn woanders hin.



# Was ist also in dem Container?

Unser Code

Code Dependencies (Frameworks/Libs/Datenbanken)

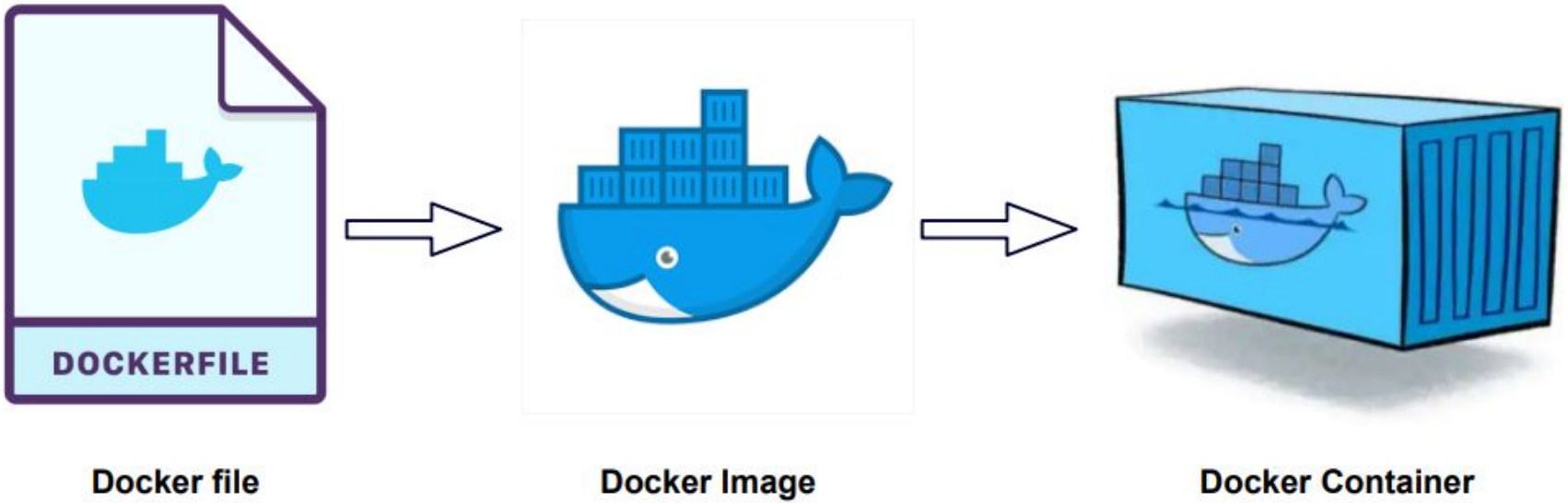
Environment settings

Alles andere um unseren Code laufen zu lassen

# Was kann ich mit einem Container machen?

Klonen und Verteilen der Inhalte → Konsistenz und Vereinfachung

# Wie macht Docker das alles?



# Docker File

“Kochrezept” für das Image

Enthält:

Technologien die verwendet werden

Runtimes

Werkzeuge/Instruktionen um den Code zum laufen zu bringen

# Docker Image

Read Only “Template”

Standardisiertes Paket, das die in der (eigenen) Dockerfile spezifizierten Eigenschaften enthält.

Es können theoretisch unendlich viele Container basierend auf einem einzelnen Image erstellt werden

Wenn Open Source: Wird üblicherweise veröffentlicht, z.B. Python Runtime Images

# Docker Container

Ausführungsumgebung basierend auf dem vorgegebenen Image

Self-contained -> Wenn richtig aufgesetzt enthalten sie alles was wichtig für den Betrieb ist

Isolated -> Von anderen Container und dem System getrennte Umgebung

Independent -> Ein Docker Container kann unabhängig von anderen funktionieren

Portable -> Funktioniert überall identisch

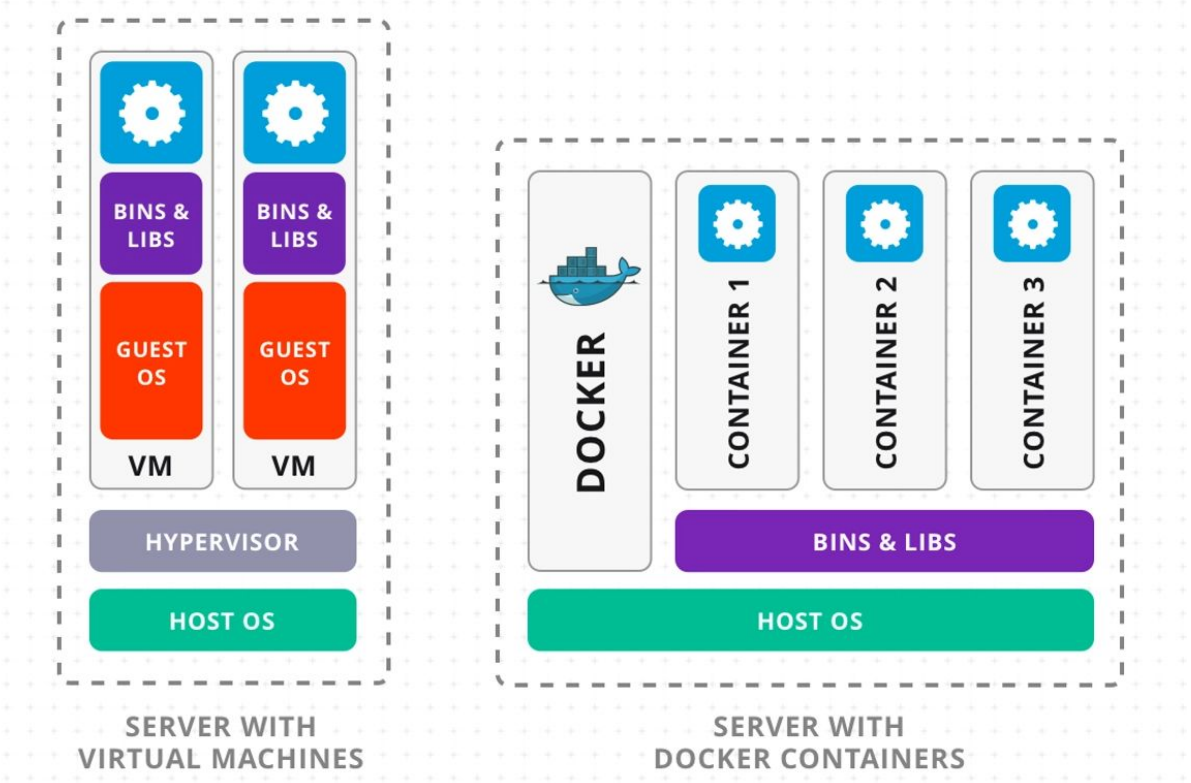
# Was haben wir also gemacht?

Anstatt viele unterschiedliche Software auf mehreren Geräten installieren zu müssen und deren konsistenz zu wahren müssen wir nur einen Docker Container starten, basierend auf unserem benötigten Image:

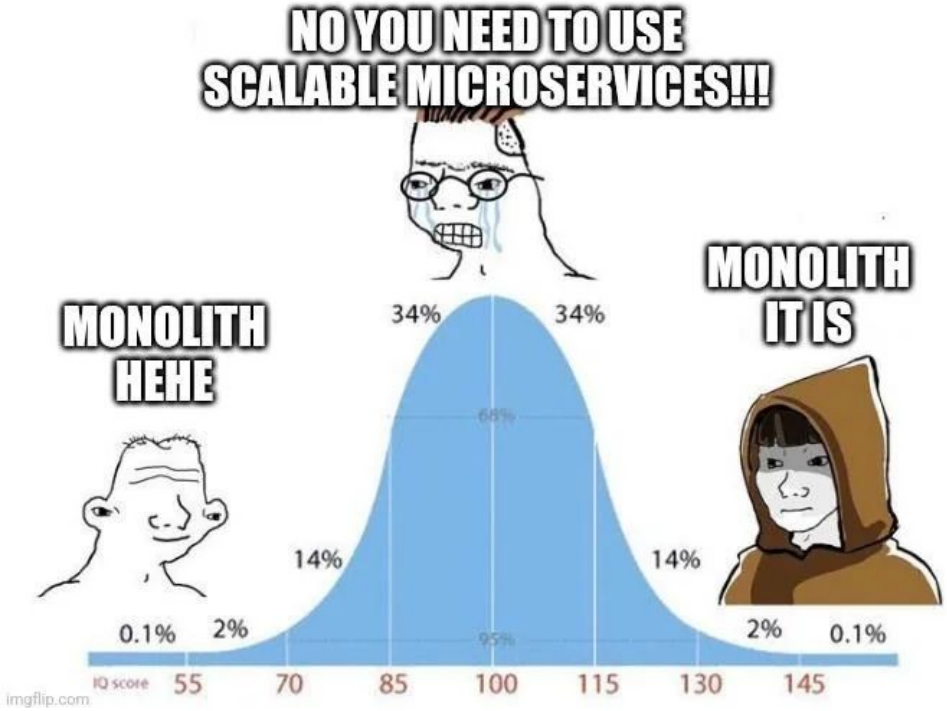
“docker run mein_container_aus_der_hoelle”

Und unsere Software kann verwendet werden :)

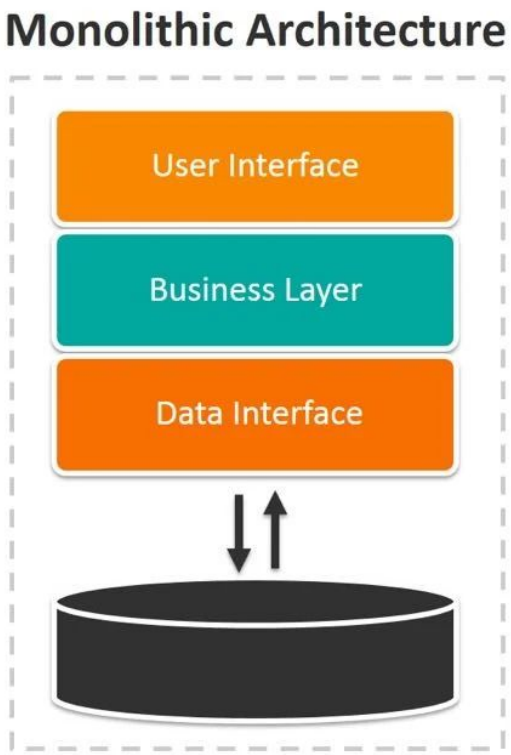
# UnD wARum NiChT eiNe vM?

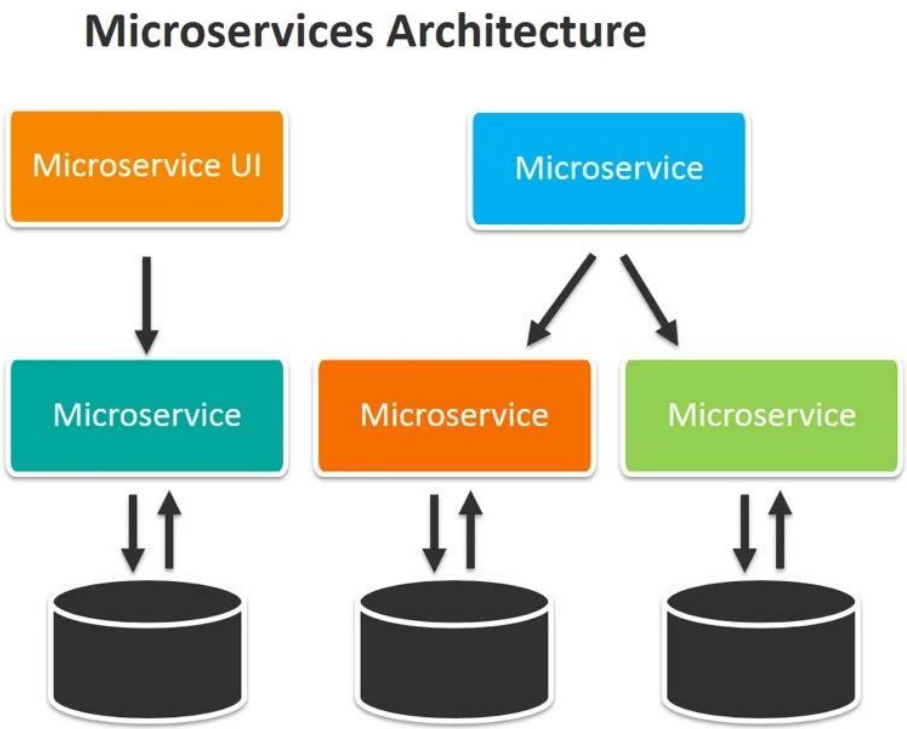


# Ok, was ist Kubernetes (K8s)?



# Software Architektur



# K8s Control Plane

Cloud Provider API: Software Interface zur Kommunikation mit dem Cloud Server anbieter (optional)

Cloud Controller Manager: Steuerungsebene für Cloud spezifische Logik (optional)

etcd: Verteiltes Key-Value Speichersystem, enthält API Serverdaten

kube-api-server: Kernverbindungstück, stellt Kubernetes HTTP API zur Verfügung

kube-scheduler: Teilt Pods zu Nodes zu

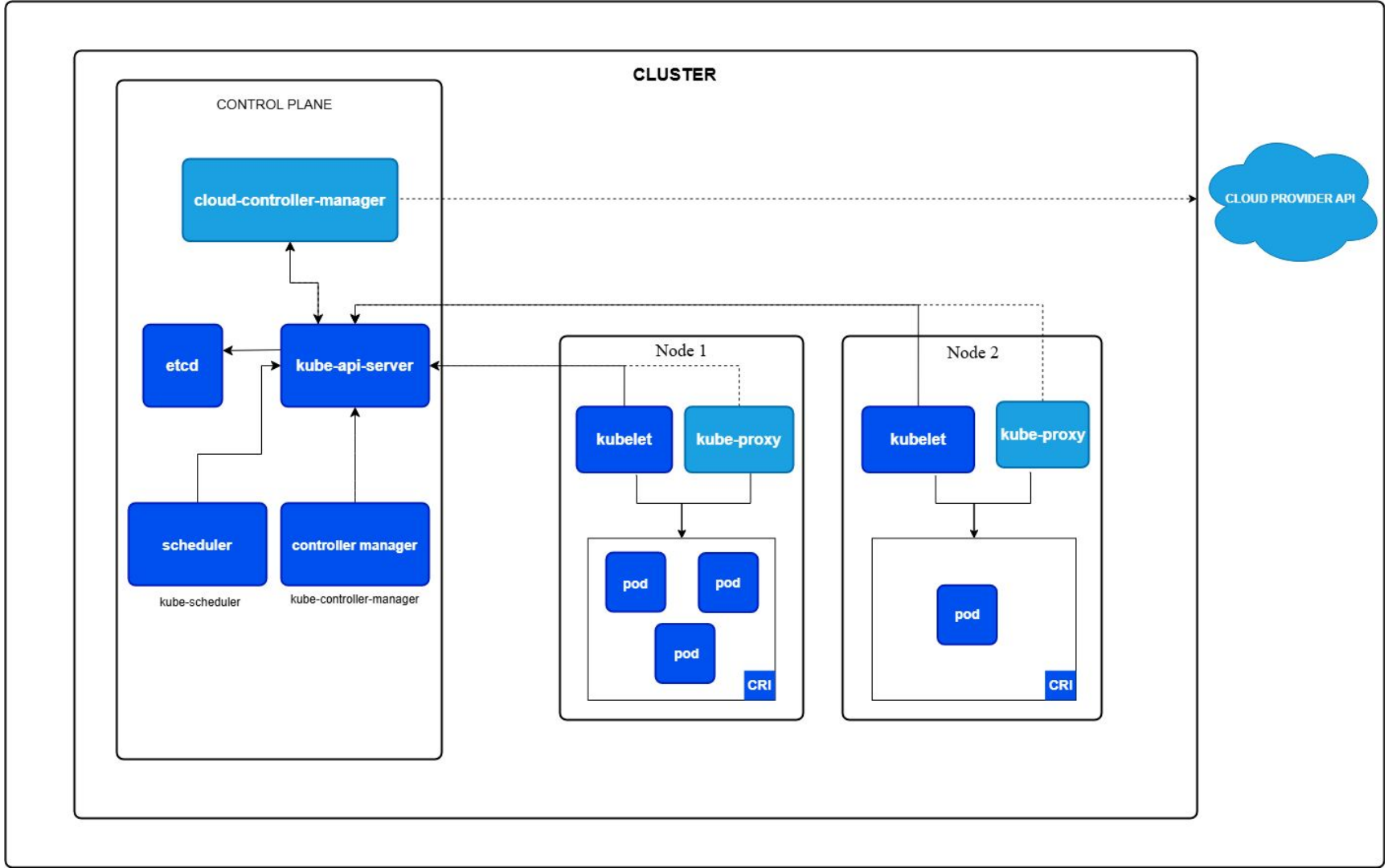
# K8s Node Komponenten

kubelet: Sorgt dafür und Kontrolliert das Pods laufen

kube-proxy: Enthält Netzwerkregeln innerhalb von Nodes (optional)

container-runtime: Software Verantwortlich für die Laufzeit von Containern

pod: (Sammlung an) Container(n), wird ausgeführt


CLUSTER
CONTROL PLANE
cloud-controller-manager
CLOUD PROVIDER API
etcd
kube-api-server
scheduler
controller manager
kube-scheduler
kube-controller-manager
Node 1
kubelet
kube-proxy
pod
pod
pod
CRI
Node 2
kubelet
kube-proxy
pod
CRI