Kubernetes - Workshop

Essentials

Eine kompakte Einführung in die Container-Orchestrierung: Architektur, Konzepte & Praxis

Marc Simon
Q4 2025

Agenda:

- Warum eigentlich Kubernetes
- Architektur Überblick
- kubectl
- Kern-Komponenten
- Praktische Beispiele
- Fragen und Antworten

Warum eigentlich Kubernetes?

Das Problem

Manuelles Management von Containern auf mehreren Servern ist fehleranfällig und skaliert nicht. Was passiert, wenn ein Server ausfällt? Wer startet den Container neu?

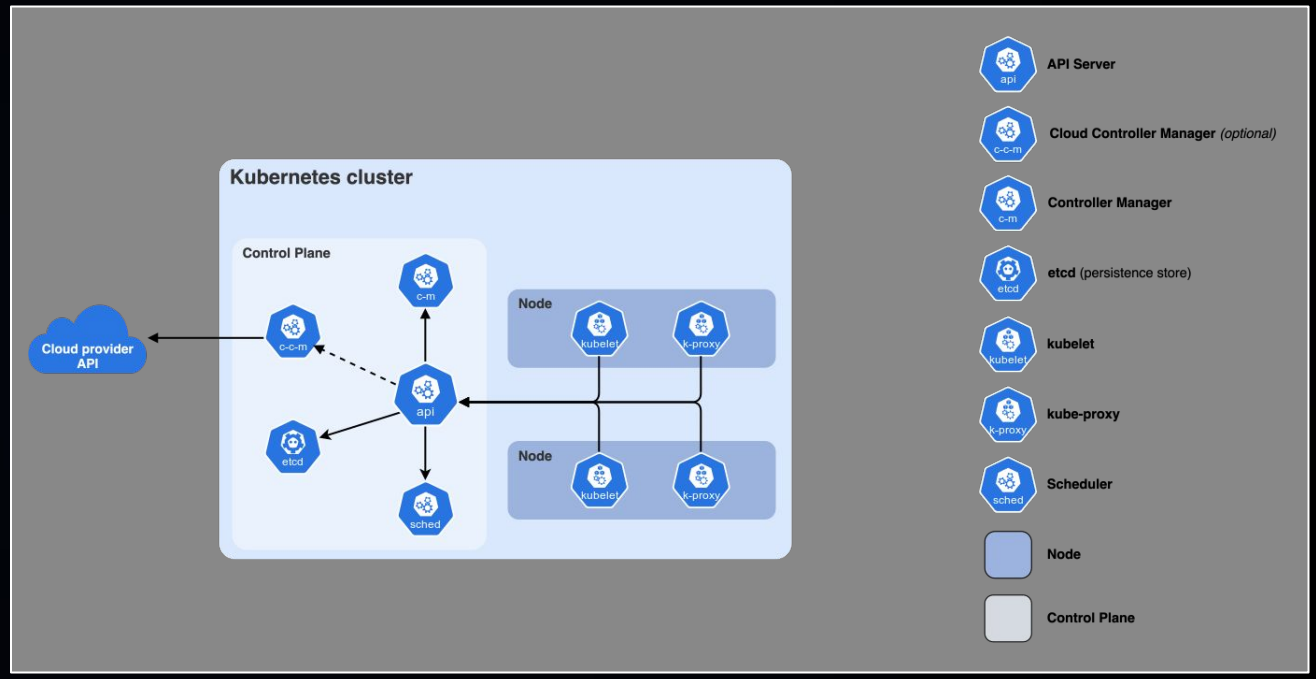
Die Lösung

Kubernetes (K8s) automatisiert Deployment, Skalierung und Verwaltung. Es ist ein **Orchestrierungs-Tool**, das für Ausfallsicherheit (Self-Healing) und Effizienz sorgt.

Die Architektur

Ein K8s-Cluster besteht aus zwei Hauptbereichen:

- **Control Plane (Master):** Das "Gehirn", das Entscheidungen trifft und den Zustand verwaltet.
- **Worker Nodes:** Die "Muskeln", auf denen die eigentlichen Anwendungen (Container) laufen.



Control Plane (Das Gehirn)

API Server

Die zentrale Schnittstelle. Alle Befehle (kubectl, interne Prozesse) laufen hier auf. Der "Torwächter" des Clusters.

etcd

Der Speicher (Source of Truth). Ein konsistenter Key-Value Store, der den gesamten Cluster-Zustand speichert.

Scheduler

Entscheidet, auf welchem Node ein neuer Pod platziert wird, basierend auf Ressourcen und Regeln.

Worker Node (Die Arbeiter)

Kubelet

Der Agent auf jedem Node. Er spricht mit dem API Server und sorgt dafür, dass die Container wie gewünscht laufen.

Kube-proxy

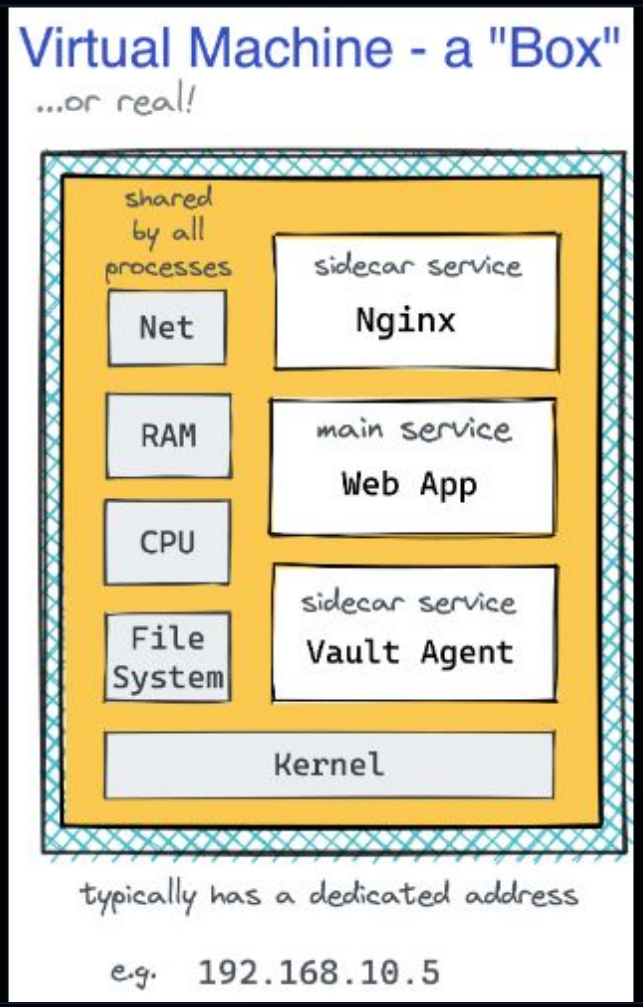
Verwaltet die Netzwerkregeln auf dem Node. Ermöglicht die Kommunikation zwischen Pods und Services.

Runtime

Die Software (z.B. Containerd, Docker), die die Container technisch ausführt.

Der Pod: Das Atom

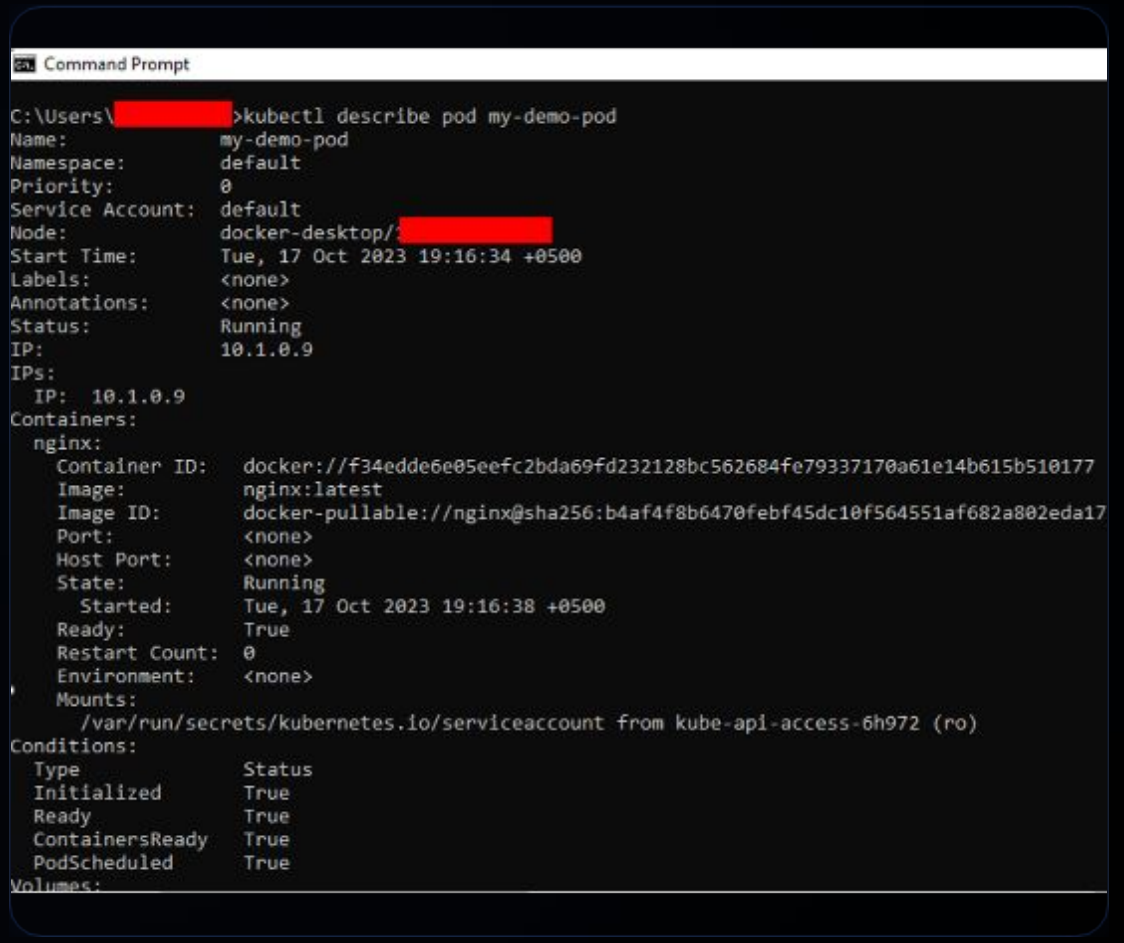
- In Kubernetes starten wir keine "Container" direkt, sondern **Pods**.
- Ein Pod ist die kleinste deploybare Einheit.
- Er umschließt einen oder mehrere Container (meistens einen).
- Container in einem Pod teilen sich IP-Adresse, Speicher und Netzwerk-Namespace ("localhost").



Das Werkzeug: kubectl

Das CLI-Tool zur Steuerung des Clusters. Die wichtigsten Befehle:

`kubectl get pods`	Zeigt laufende Pods an
`kubectl describe ...`	Zeigt Details & Events (Fehlersuche!)
`kubectl logs -f ...`	Streamt Container-Logs
`kubectl apply -f ...`	Wendet YAML-Konfiguration an



Metadaten: Der Ausweis

Identität

Macht Objekte eindeutig auffindbar.

- **name:** Eindeutiger Name innerhalb eines Namespaces (z.B. my-app-pod).
- **namespace:** Virtueller Cluster zur Isolierung (z.B. prod vs dev).

Organisation

Für Gruppierung und Auswahl.

- **labels:** Für K8s. Selektoren verbinden Services mit Pods (z.B. app: nginx).
- **annotations:** Für Tools. Infos wie Build-Version oder Maintainer-Kontakt.

Workloads: Deployment vs StatefulSet

Deployment (Stateless)

Ideal für Webserver & APIs. Pods sind austauschbar ("Cattle").

- Zufällige Hash-Namen (web-7f8b9c-xyz)
- Keine feste Identität
- Paralleles Starten/Stoppen möglich
- Rolling Updates ohne Downtime

StatefulSet (Stateful)

Ideal für Datenbanken. Pods sind einzigartig ("Pets").

- Feste Namen (db-0, db-1, db-2)
- Stabile Netzwerk-Identität
- Geordnetes Starten/Beenden (0 -> 1 -> 2)
- Persistenter Speicher bleibt erhalten

Konfiguration: ConfigMap vs Secret

ConfigMap

Für nicht-sensible Konfigurationsdaten.

- Umgebungsvariablen (DB_HOST, DEBUG_LEVEL)
- Konfigurationsdateien (nginx.conf, settings.json)
- Entkoppelt Konfiguration vom Container-Image

Secret

Für sensible Daten.

- Passwörter, API-Tokens, SSH-Keys, Zertifikate
- Base64-kodiert gespeichert (Achtung: Nicht verschlüsselt!)
- Wird als Datei gemountet oder als Env-Var injiziert

Networking: Service vs Ingress

Service (Layer 4)

Die interne Abstraktion.

- Bietet stabile Cluster-IP für Pod-Gruppen
- Loadbalancing über alle Pods (Round Robin)
- Typen: ClusterIP (intern), NodePort (Port am Node), LoadBalancer (Cloud-IP)
- Versteht nur TCP/UDP, keine URLs

Ingress (Layer 7)

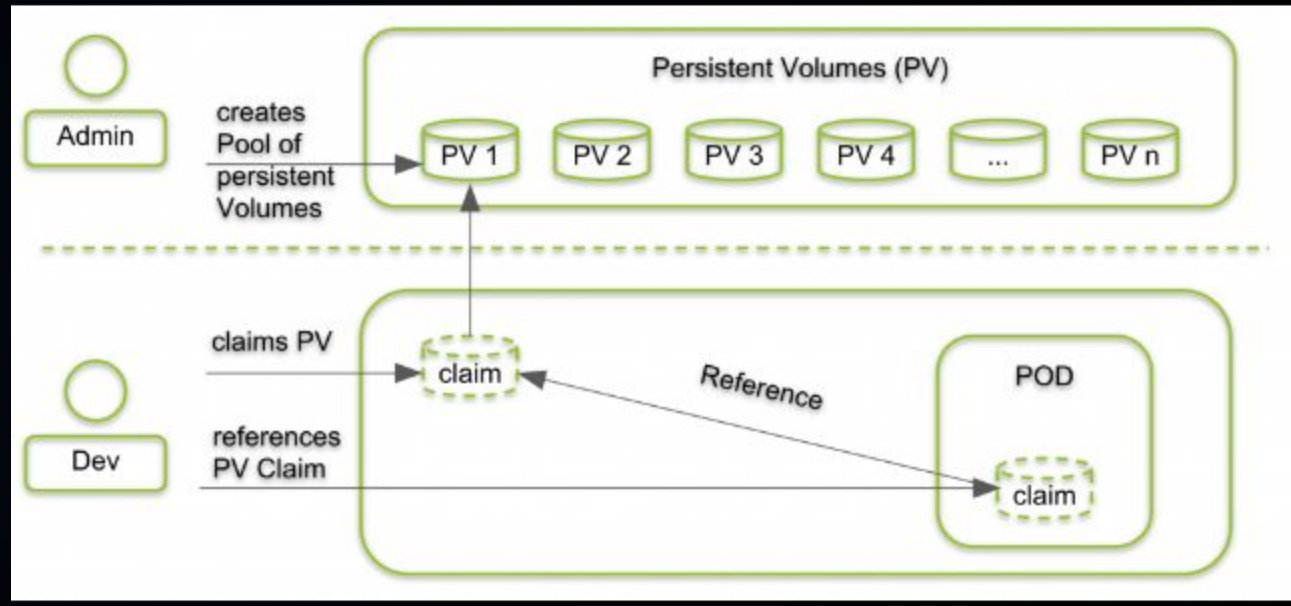
Der externe Zugang.

- Routet HTTP/HTTPS Traffic von außen
- Versteht Domains (Host) und Pfade (/app)
- Terminiert SSL/TLS Zertifikate
- Leitet Traffic an den passenden **Service** weiter

Persistenz: PV & PVC

Container sind flüchtig – Daten gehen beim Neustart verloren. Wir brauchen externen Speicher.

- **PV (PersistentVolume):** Die "Steckdose" an der Wand. Die eigentliche Ressource (z.B. 10GB NFS Speicher), bereitgestellt vom Admin.
- **PVC (PersistentVolumeClaim):** Der "Stecker". Die Anforderung des Entwicklers ("Ich brauche 5GB").



Workloads: Job & CronJob

Job (Einmalig)

"Run-to-Completion". Führt eine definierte Aufgabe aus und beendet sich danach.

- Einsatz: DB-Migrationen, Batch-Processing
- Policy: RestartPolicy: OnFailure
- Garantiert einmalige erfolgreiche Ausführung

CronJob (Zeitgesteuert)

Erstellt Jobs automatisch basierend auf einem Zeitplan (Cron-Format).

- Einsatz: Backups, nächtliche Reports
- Syntax: */5 * * * * (alle 5 Min)
- Verwaltet Historie (Successful / Failed Jobs)

Fragen & Antworten

Vielen Dank für Eure Aufmerksamkeit!

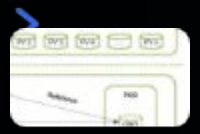
Image Sources

https://kubernetes.io/images/docs/components-of-kubernetes.svg

Source: kubernetes.io

https://labs.iximiuz.com/content/files/tutorials/containers-vs-pods/__static__/vm-min.png

Source: labs.iximiuz.com



https://miro.medium.com/0*v7-cw-1KYxQHGjVa.png

Source: medium.com

https://refine.ams3.cdn.digitaloceanspaces.com/blog/2023-10-19-kubecti-exec/image-back-pull.png

Source: refine.dev