CLOUD NATIVE COMPUTING

CLOUD-NATIVE

Unit:

Container-Orchestrierung

(3) Kubernetes (Teil I: Architektur und Ressourcen)

PROF. DR.
NANE KRATZKE

creative commons

CLOUD NATIVE COMPUTING

Urheberrechtshinweise

Diese Folien werden zum Zwecke einer praktikablen und pragmatischen Nutzbarkeit im Rahmen der **CC0 1.0 Lizenz** bereitgestellt.

Sie dürfen die Inhalte also kopieren, verändern, verbreiten, mit eigenen Inhalten mixen, auch zu kommerziellen Zwecken, und ohne um weitere Erlaubnis bitten zu müssen.

Eine Nennung des Autors ist nicht erforderlich (aber natürlich gern gesehen, wenn problemlos möglich).

Diese Folien sind insb. für die Lehre an Hochschulen konzipiert und machen daher vom **§51 UrhG (Zitate)** Gebrauch.

Die CC0 Lizenz überträgt sich nicht auf zitierte Quellen. Hier sind bei der Nutzung natürlich die Bedingungen der entsprechenden Quellen zu beachten.

Die Quellenangaben finden sich auf den entsprechenden Folien.

KAPITEL 9

Container-Plattformen

9.1 Scheduling

- Heterogenität von Workloads
- Scheduling-Algorithmen
- Scheduling-Architekturen

9.2 Orchestrierung

- Definition von Betriebszuständen
- Regelkreis: Desired vs Current State

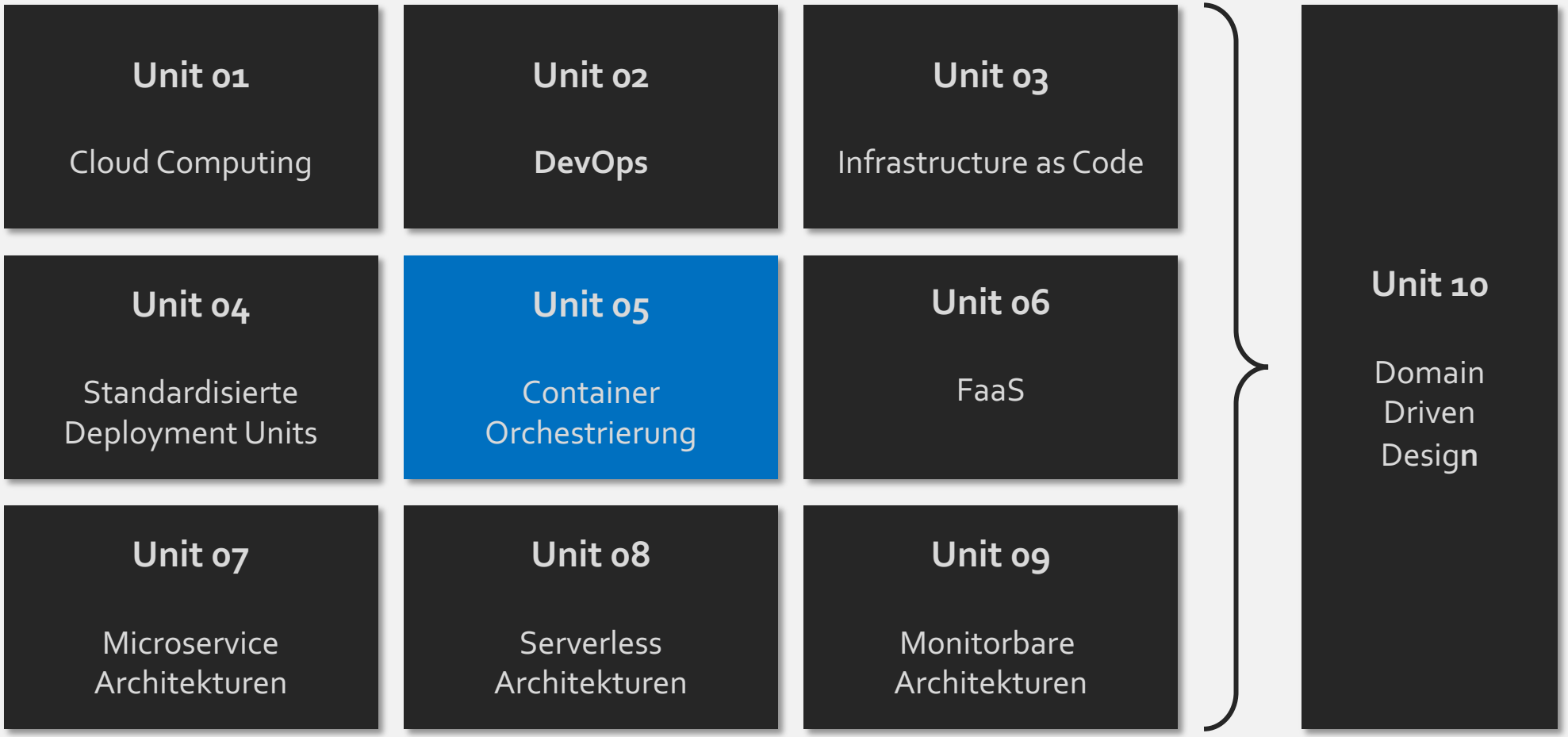
9.3 Inside Kubernetes

- Kubernetes-Architektur und Ressourcen
- Workloadarten
- Scheduling Constraints
- Automatische Skalierung von Workloads
- Exponierung von Services
- Health Checking
- Persistenz
- Isolation von Workloads

9.4 Zusammenfassung

INHALTSVERZEICHNIS

Überblick über Units und Themen dieses Moduls



INHALTE

Scheduling

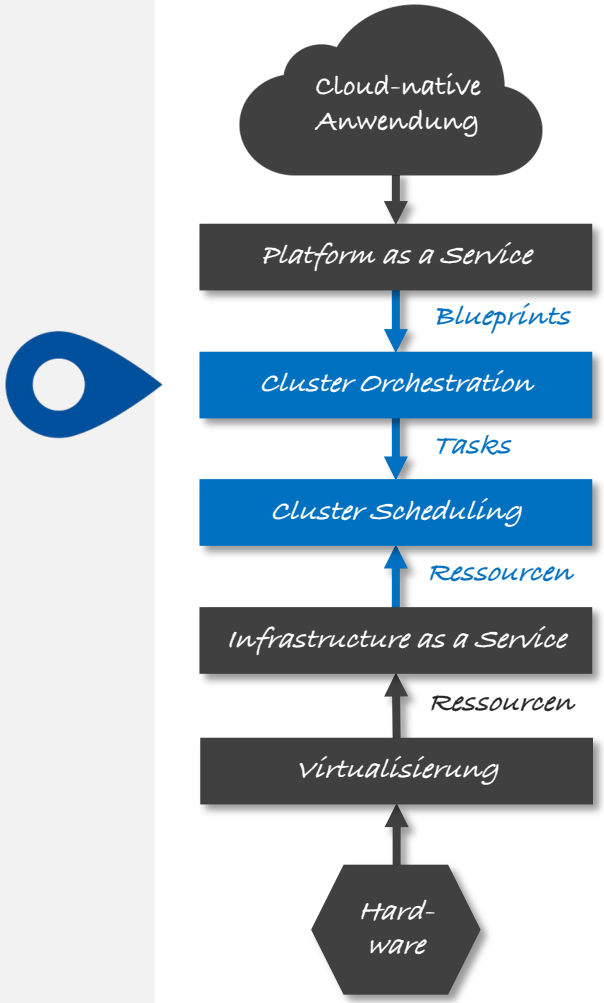
- Scheduling Problem Definition
- Scheduling Algorithmen
- Scheduler Architekturen
- Beispiele von Cluster Schedulern: Mesos, Swarm

Orchestrierung

- Was ist Orchestrierung (in Abgrenzung zum Scheduling)?
- Was sind Blueprints?
- Überblick über bestehende Orchestrierungslösungen

Inside Kubernetes (Typ-Vertreter)

- K8S-Architektur
- K8S-Ressourcen
- Workloads
- Persistenz
- Isolation und Exponieren von Services



KUBERNETES

De-facto Standard für Container-Orchestierung

- Cluster-Orchestrierer für OCI-Container, der eine Reihe an Kern-Abstraktionen für den Betrieb von Anwendungen in einem großen Cluster einführt.
- Applikations-Blueprints werden über YAML- oder JSON-Dateien (sogenannte Manifests) definiert.
- Open-Source Projekt, das von Google initiiert wurde. Google will damit die jahrelange Erfahrung im Betrieb großer Cluster der Öffentlichkeit zugänglich machen und damit auch Synergien mit dem eigenen Cloud-Geschäft heben.
- Seit Juli 2015 in Version 1.0. Skaliert auf bis zu 1000 Nodes großen Clustern.
- Beiträge zur Codebasis von vielen Firmen neben Google – u.a. Mesosphere, Microsoft, Pivotal, RedHat, u.v.m.
- Zur Standardisierung der Cluster-Orchestrierung wurde die Cloud Native Computing Foundation (https://cncf.io) gegründet.
- Kubernetes ist das Lead-Projekt der CNCF.

KUBERNETES

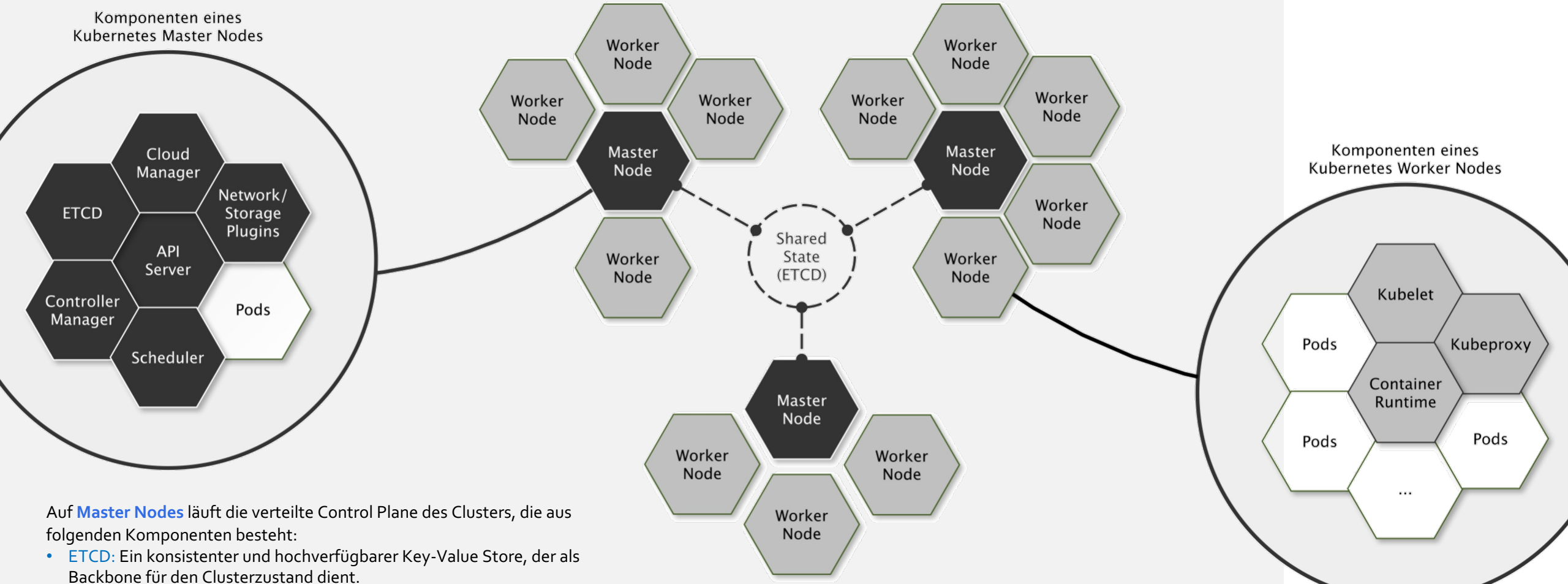
Node-, Netzwerk- und Storage-Virtualisierung

Kubernetes
(Orchestrierung von Blueprints)

Overlay Network	Cluster Scheduler	Persistent Volume
Netzwerk-Virtualisierungen	**Node-Virtualisierung**	**Storage Virtualisierungen**
Gängige Overlay Networks • OpenVSwitch • Flannel • Weave • Calico Diese können Add-on bringen. Z.B. transparente Container-to-Container Verschlüsselung (weave) und Netzwerkisolation (Calico).	*Siehe Cluster-Scheduling im ersten Teil dieser Unit*	Gängige Anbindungen • GCE/AWS Block-Store • NFS • iSCSI • Ceph oder GlusterFS Über diese können Persistent Volumes mittels Cloud-Storages oder Storage Cluster für statusbehaftete Container bereitgestellt werden.

Neben einem Cluster-Scheduler setzt Kubernetes auch noch auf Netzwerk- und Storage-Virtualisierungen auf.

KUBERNETES PLATTFORM



Auf **Master Nodes** läuft die verteilte Control Plane des Clusters, die aus folgenden Komponenten besteht:

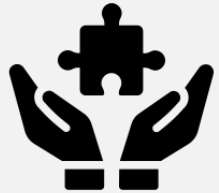
- ETCD: Ein konsistenter und hochverfügbarer Key-Value Store, der als Backbone für den Clusterzustand dient.
- API-Server: Dient als Schnittstelle für die Steuerung und Verwaltung des Clusters.
- Scheduler: Weist Pods den Worker Nodes zu.
- Controller Manager: Überwacht und steuert Ressourcen.
- Cloud Manager: Integriert Cloud-spezifische Funktionen.

Worker Nodes dienen zur Ausführung von Applikationen (Pods). Die Interaktion mit der Control Plane erfolgt dabei über folgende Komponenten:

- Kubelet: Registriert den Node bei dem API-Server und überwacht die auf ihm laufenden Pods.
- Kube-Proxy: Arbeitet als Netzwerkproxy und Load Balancer.
- Container Runtime Environment, wie bspw. Docker, CRI-O oder containerd.

KUBERNETES ADDONS

Funktionen, die oft als ergänzende Cluster-Ressourcen bereitgestellt werden



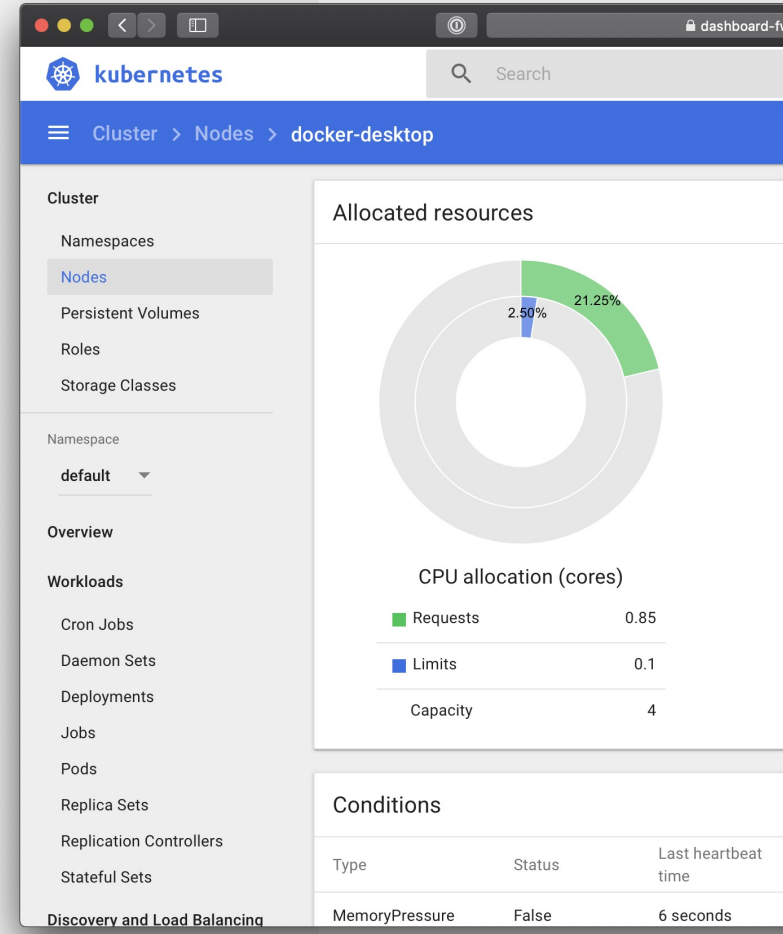
CLOUD NATIVE COMPUTING

DNS

- Ein obligatorischer DNS-Server muss DNS-Einträge für Kubernetes-Dienste bereitstellen.
- Container, die von Kubernetes gestartet werden, beziehen diesen DNS-Server automatisch in ihre DNS-Suche ein.

Web UI (Dashboard)

- Dashboard ist eine allgemeine, webbasierte Benutzeroberfläche für Kubernetes-Cluster.
- Es ermöglicht Benutzern die Verwaltung und Fehlerbehebung von Anwendungen, die im Cluster laufen, sowie des Clusters selbst.
- Mittlerweile gibt es auch Cluster-externe IDEs (wie bspw. Lens)



Container Resource Monitoring

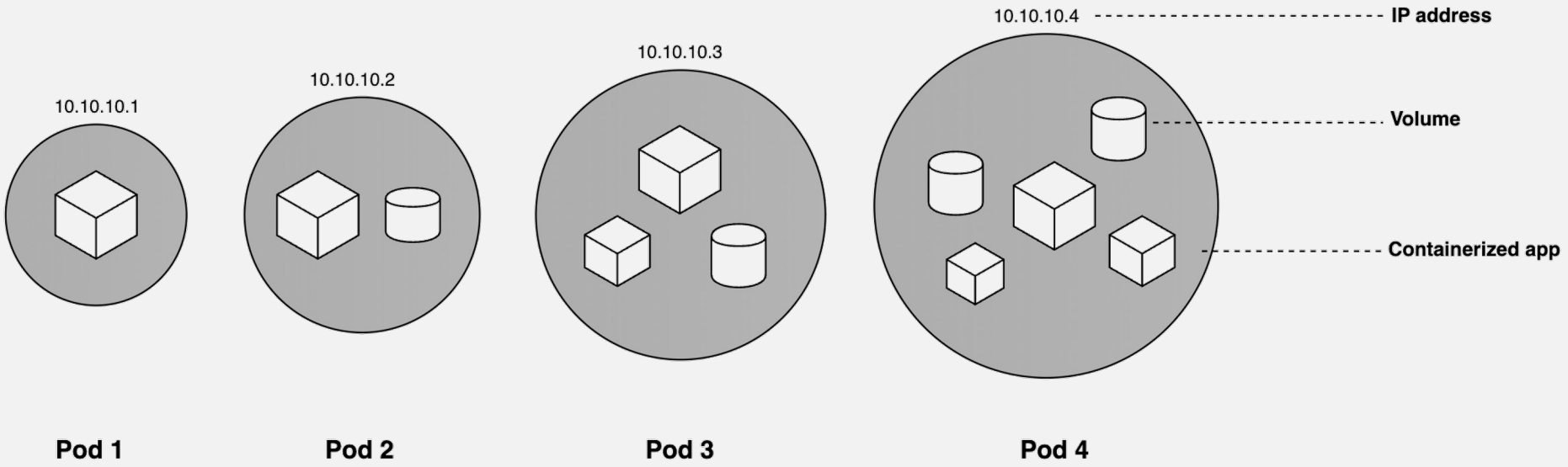
- Ein Container Resource Monitoring zeichnet generische Zeitreihenmetriken in einer zentralen Metrik-Datenbank auf.
- Die Metrik-Datenbank bietet oft eine Analyse-/Browsing-Schnittstelle
- Bspw.: Prometheus / Grafana oder MetricBeat / ElasticSearch / Kibana

Cluster-level Logging

- Ein Protokollierungsmechanismus auf Clusterebene ist für die Weiterleitung von Containerprotokollen an eine zentralen Log-Datenbank zuständig.
- Die Log-Datenbank bietet oft eine Such-/Browsing-Schnittstelle
- Bspw.: FileBeat / ElasticSearch / Kibana

KUBERNETES

Der Pod – die grundlegende Ausführungseinheit



- Jeder Pod hat eine einzigartige IP-Adresse im Kubernetes-Cluster.
- Jeder Pod wird in der Regel für einen spezifischen Workload genutzt.
- Ein Pod kann einen oder mehrere Container umfassen, die gemeinsam gescheduled werden.
- Container innerhalb eines Pods teilen sich Netzwerkressourcen und Volumes.
- Alle Container eines Pods werden gemeinsam erstellt, gestartet, gestoppt und als eine Einheit repliziert.

BLUEPRINTS

YAML-basierte Manifest Dateien

Kubernetes Manifeste beschreiben den gewünschten Zustand einer Ressource. Der Kubernetes-Controller sorgt dafür, dass der aktuelle Zustand des Clusters diesem gewünschten Zustand entspricht. Wenn der aktuelle Zustand abweicht, unternimmt Kubernetes Schritte, um die Diskrepanzen zu korrigieren.

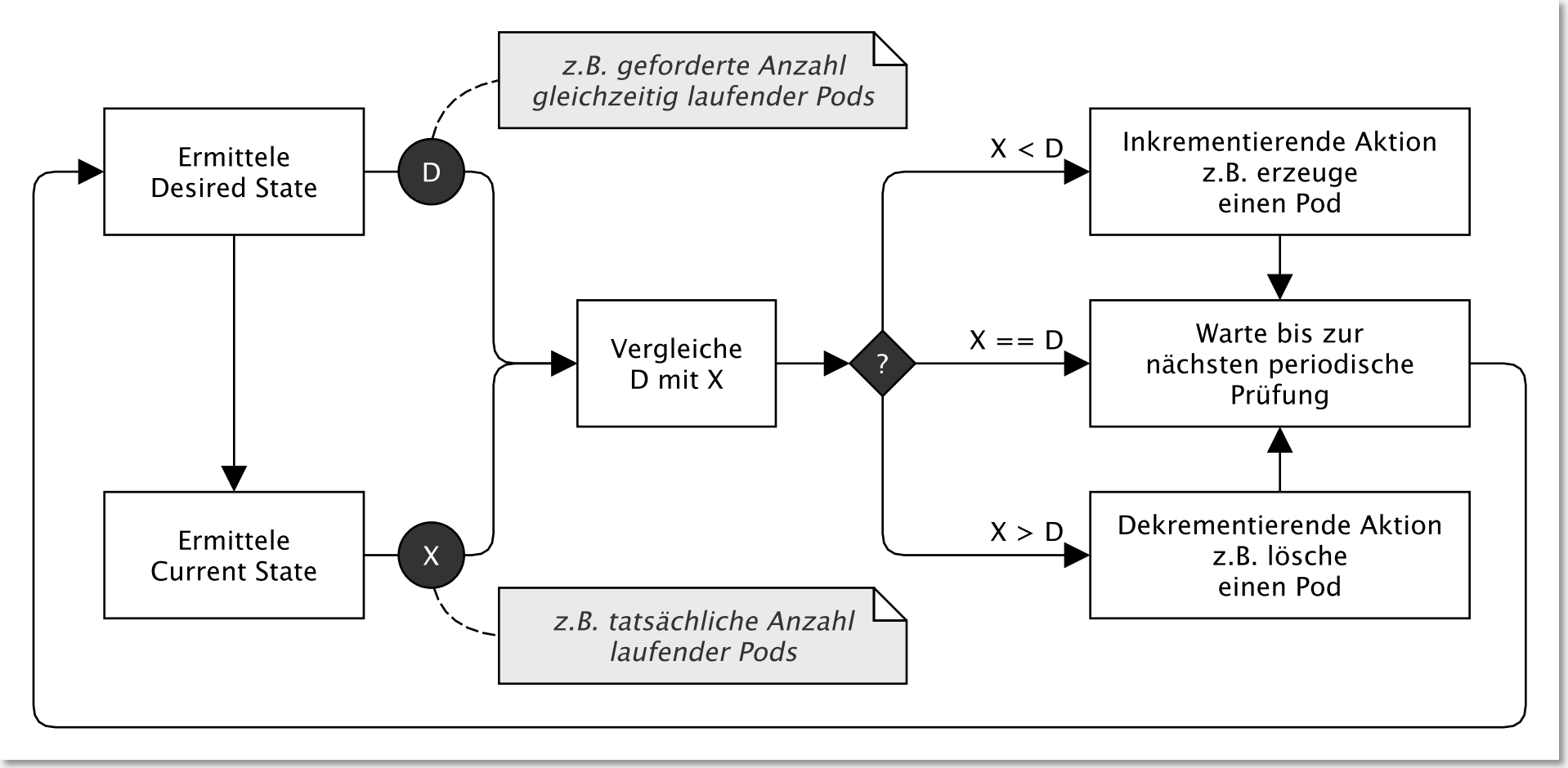
Ein Kubernetes Manifest enthält spezifische Anweisungen für den Kubernetes API-Server über Folgendes:

- Die **Art der Ressource** wie zum Beispiel Pod, Service, Deployment, ReplicaSet etc.
- **Metadaten der Ressource** wie Name, Namespace, Labels und andere Identifikatoren
- Die **Spezifikation der Ressource** umfasst die spezifischen Eigenschaften und den gewünschten Betriebszustand
- Der aktuelle **Status der Ressource** wird von Kubernetes selbst ausgefüllt und beschreibt den aktuellen Zustand der Ressource (nicht dargestellt).

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

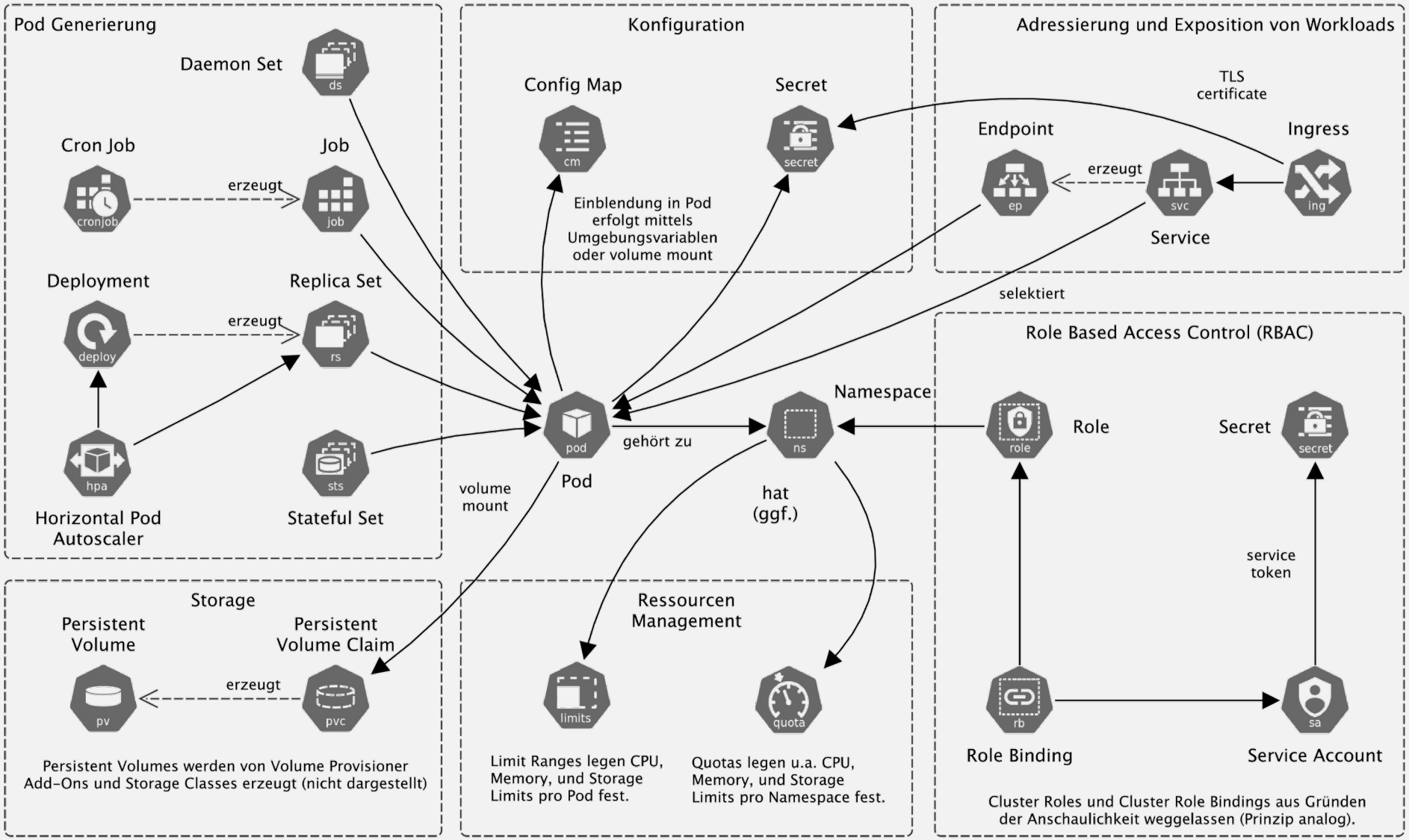
KUBERNETES CONTROLLER

Regelkreis: Desired vs. Current State



HÄUFIG GENUTZTE K8S-RESSOURCEN

Und ihre Zusammenhänge



CLOUD NATIVE COMPUTING

Von Kubernetes verwaltete Ressourcen

- Pod
- Namespace
- Deployment
- Horizontal Pod Autoscaler
- Replica Set
- Stateful Set
- Daemon Set
- (Cron -) Job
- ConfigMap
- Secret
- Persistent Volume
- Persistent Volume Claim
- ServiceAccount
- (Cluster-Role Binding
- (Cluster-)Role
- Service
- Endpoint
- Ingress
- Limit Ranges
- Quota

AUSBLICK

Überblick über die wichtigsten von Controllern überwachten Kubernetes-Ressourcen

Ausführbare Workloads	Exponieren von Anwendungen und Diensten	Persistenz	Isolieren von Anwendungen mittels Namespaces
 **Deployments**	Primär Intra-Cluster  **Services**	 **Persistent Volume Claim**	  **Quotas und Limit Ranges**
 **(Cron-)Jobs**			
 **Daemon Sets**	Primär Extra-Cluster  **Ingress**	 **Persistent Volume**	 **RBAC**  **Network Policies**
 **Stateful Sets**			

KONTAKT

Disclaimer

Nane Kratzke

+49 451 300-5549

nane.kratzke@th-luebeck.de

kratzke.mylab.th-luebeck.de