Container-Orchestrierung mit Kubernetes

Prof. Dr. Wolfgang Blochinger

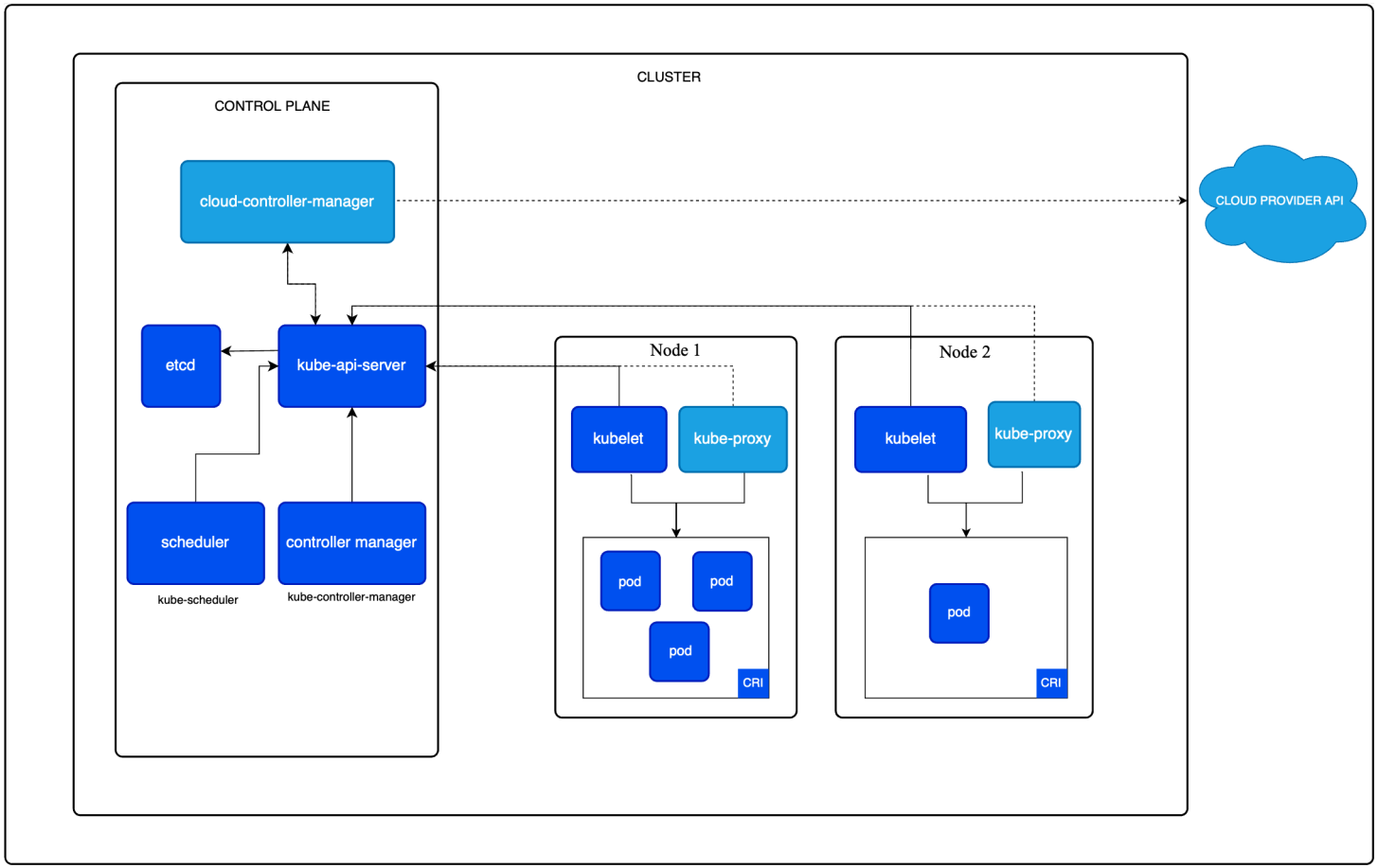
Kubernetes

- Der Name „Kubernetes“ (abgekürzt K8S) bedeutet „Steuermann“ (altgriechisch).
- Von Google entwickeltes System zur Container-Orchestrierung.
  - 2014 als Nachfolger von „Borg“ entwickelt.
- Wurde 2015 der Cloud Native Computing Foundation gespendet.
  - Open-Source Software unter Apache-Lizenz 2.0
- Wird von den meisten Cloud-Anbietern als Plattform-Service zur Verfügung gestellt.
  - Hat sich als Provider unabhängige, standardisierte Ausführungsumgebung für Container etabliert.
  - Vereinfacht stark Cloud-Migration und Multi-Cloud-Szenarien.

Container Orchestrierung

- Ausführung einer großen Zahl von Containern in Clustern
- Planung und Zuweisung von Cluster-Ressourcen
- Überwachung des Zustands von Containern und Cluster-Ressourcen
- Sicherstellung der Verfügbarkeit von Containern
- (Automatische) Skalierung von Containern zur Realisierung elastischer Anwendungen
- Update- und Rollback-Prozesse für Container
- Externe Bereitstellung von Diensten, die in (mehreren) Containern ausgeführt werden (inklusive Lastverteilung).
- Migration von Containern
- …

Kubernetes Cluster-Architektur



Quelle: https://kubernetes.io/docs/concepts/architecture/

Kubernetes Cluster-Architektur

- **Worker Nodes:** Führen Container in Form von Pods (siehe unten) aus.
  - **Kubelet:** Steuert und überwacht die Ausführung der Container/Pods.
  - **Kube-Proxy:** Stellt externe Netzwerk-Konnektivität für Pods her.
  - **Kubernetes Container Runtime Interface (CRI):** Standardisierte Schnittstelle zu einer lokalen Container-Runtime.
- **Control Plane:** Globale Steuerung des Clusters.
  - **kube-api-server:** Frontend für die Komponenten der Control Plane.
  - **etcd** (Hochverfügbarer Key-Value Store): Speichert alle für den Cluster-Zustand relevanten Daten.
  - **kube-scheduler:** Bestimmt Zuordnung der Pods zu Worker Nodes.
  - **kube-controller-manager:** Überwachung des Clusters
  - **cloud-controller-manager:** Anbindung an externes Cloud-Management

Pod

- Ein Pod ist verantwortlich für die Verwaltung und Ausführung von einem oder mehreren (zusammengehörenden) Containern.
  - Stellt für Container gemeinsame Ressourcen zur Verfügung (Netzwerk, Storage, Konfigurationen).
- Pods sind die elementaren Replikations- und Skalierungseinheiten von Kubernetes.
- Typisches Setup:
  - Ein Container pro Pod oder
  - ein Container für Hauptfunktionalität plus Helfercontainer („Sidecar“).

Replica Set

- Replica Set spezifiziert ein Template für Pods.
- Erzeugt (mehrere) Pods durch Instanziierung des Templates.
- Überwacht die erzeugten Pods und stellt sicher, dass stets eine definierte Anzahl von Pods existiert.
- Ermöglicht zustandslose, horizontale Skalierung von Pods.

Deployment

- Deployments erstellen, überwachen und löschen Replica Sets.
- Ermöglichen kontrollierte Update und Rollback Prozesse für Pods.
- Ablauf Update:
  - Neues Replica Set erzeugen.
  - Pods im neuen Replica Set hochfahren und entsprechend im alten Replica Set herunterfahren.
    - Recreate Update Strategie: Zunächst alle alten Pods herunterfahren, anschließend die neue Pods hochfahren
    - Rolling Update Strategie: Abwechselndes Herunter- und Hochfahren alter bzw. neuer Pods.
  - Alte Replica Sets bleiben bestehen und können für Rollbacks verwendet werden.

Service

- Ein Kubernetes Service ermöglicht (externen) Zugriff auf Pods in Form eines festen Zugriffspunktes (feste IP-Adresse oder URL).
- Weitere Funktionen eines Service:
  - Weiterleitung: Service kennt alle Pods eines Replica Sets und leitet eingehenden Datenverkehr zu den Pods weiter.
  - Load Balancing: Service verteilt Datenverkehr auf alle verfügbaren Pods eines Replica Sets.

Stateful Set

- Funktionalität ähnlich wie bei Deployments/Replica Sets.
- Hauptunterschied: Jeder Pod in einem Stateful Set erhält eine beständige Identität mittels eines Ordinal-Index (beginnend bei 0).
  - Bleibt bei Pod-Neustart/Neuerstellung erhalten.
- Stateful Sets unterstützen zustandsbehaftete verteilte Anwendungen:
  - Dauerhafte Zuordnung von Funktionalität zu Ressourcen.
    - Wichtige für verteilte Datenspeicherung und –verarbeitung.
  - Definierte, reproduzierbare Reihenfolge bei Update-Prozessen.