Kubernetes

application orchestrator

Application Hosting Evolution

- Single Program Execution(Zuse Z1 - 1937)
- Multiprogramming(Leo III - 1963)
- Multitasking (MIT Multics - 1964)
- OS Virtualization (IBM LPAR - 1972)
- Filesystem Isolation chroot (AT&T Unix - 1979)
- Application Level Virtualization/Sandboxing (Citrix Metaframe - 1998)
- OS-Level Virtualization aka Container (BSD Jails - 1999)

OS-Level Virtualization Vorteile

- Isolierte Userspace Instanzen
- restriktiver Zugriff nur auf zugewiesene Systemresourcen.
- weniger overhead
- portabler durch weniger Deployment-Aufwand
- Skalierbarkeit

Was ist Kubernetes?

- Cluster zum Container-basierten Applikationsbetrieb bestehend aus:
  - Master Node(s)
  - Worker Node(s)
- Orchestrator für den Betrieb von Microservices

Kubernetes - Deklaratives Modell

- yaml-basierte Manifeste
- beschreibt den gewollten Zustand
- Was nicht wie
- Controller erledigen die eigentliche Implementierung

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
```

Kubernetes - yaml minimum

- API Version (der Kubernetes API)
- Kind (Typ des Objekts)
- metadata (Daten zur eindeutigen Identifierung eines Objekts: Name, uids, namespace)
- spec (Objekt-spezifische Felder)

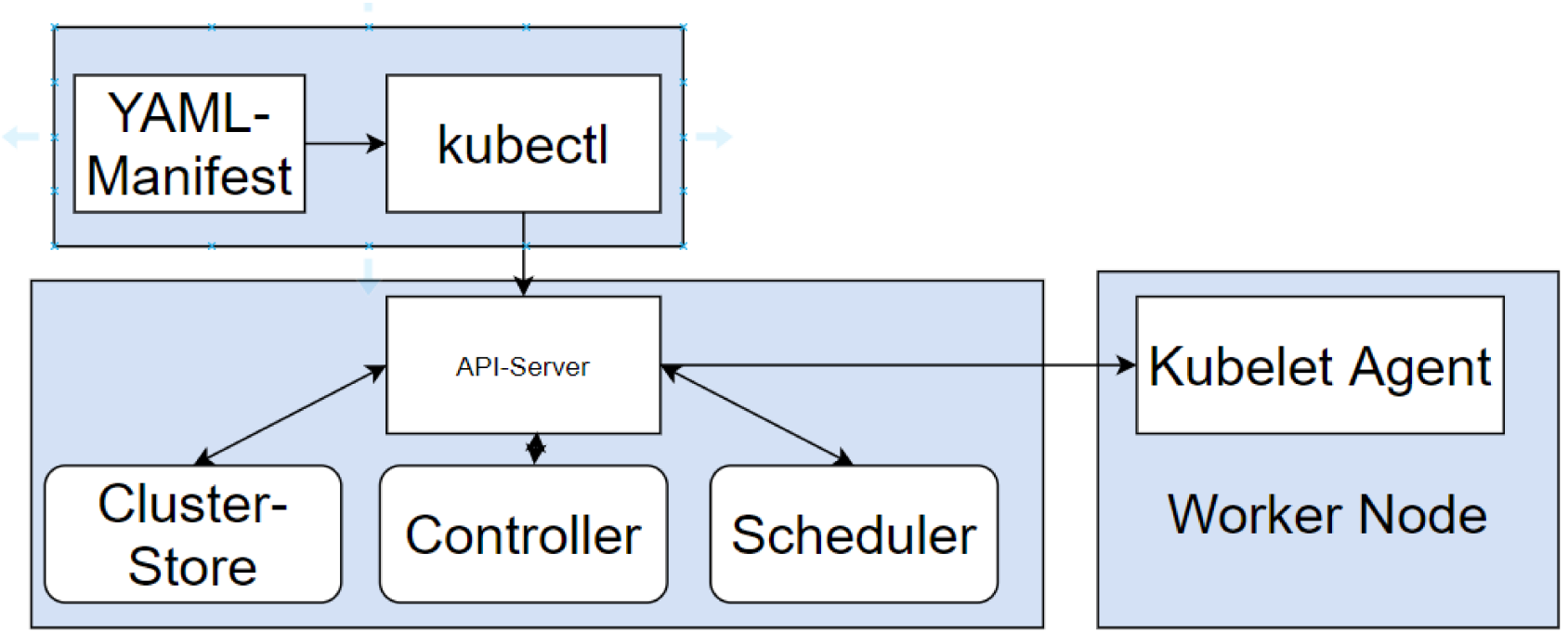
Kubernetes Cluster

- Master Node(control plane)
  - Bereitstellung der Kubernetes API
  - Scheduler
  - Controller
  - Persistent Store für State Information
  - Deployment
- Worker Nodes
  - führen Applikationen aus

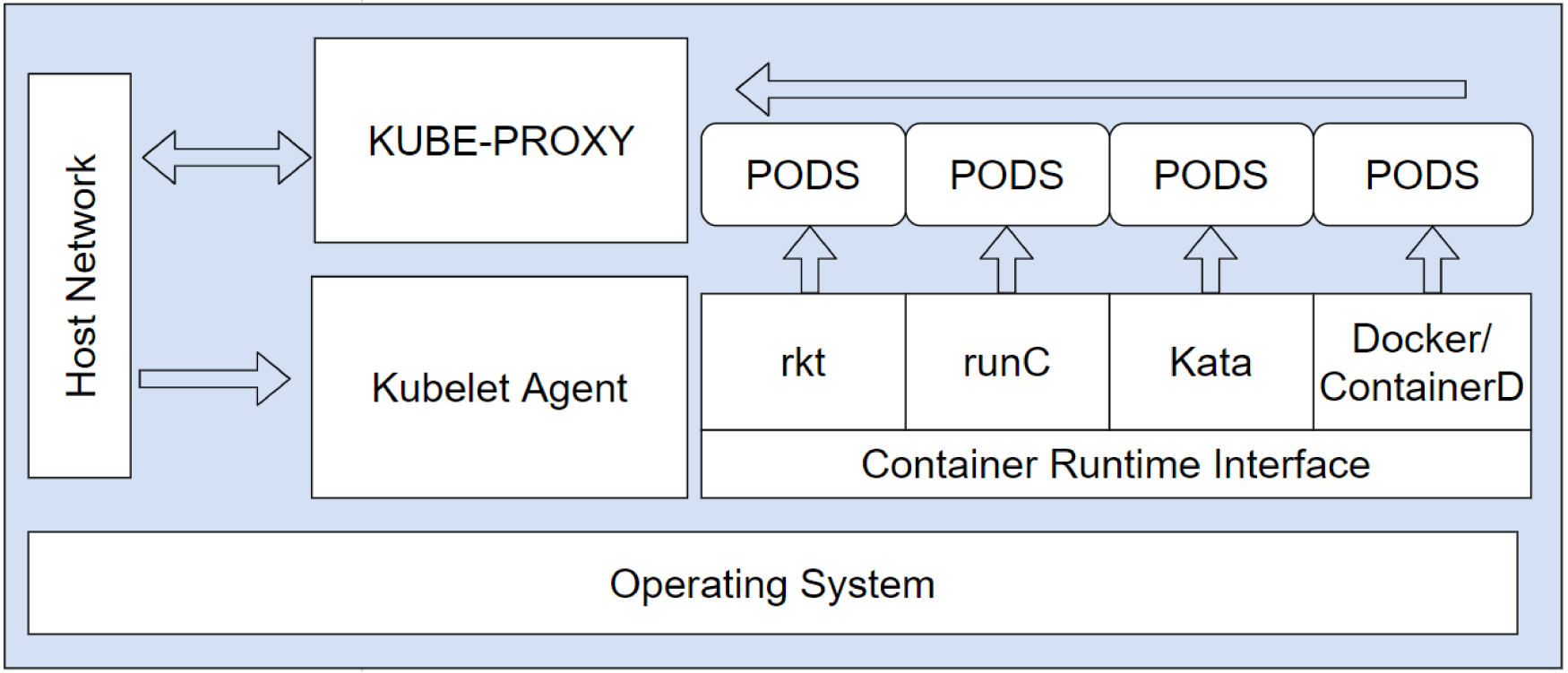
Master Node - Control Plane

- API Server (external Rest Interface, Reporting Interface, Kommunikation mit Worker Nodes)
- kube scheduler
  - weist Pods worker nodes zu
- kube controller
  - überwacht pod & cluster states
- state store (etcd)
  key/value based db zum speichern der Cluster Informationen/States

Kubernetes Workflow



Worker Nodes



Kubernetes - Pods

- Pods sind atomar und haben eine unique IP
- Pods können einen oder mehrere Container enthalten
- shared execution environment
- Pods können application centric oder infrastructure centric sein
- container innerhalb von Pods können unabhängig von einander skaliert und betreut werden

Kubernetes - Deployments

- aktueller Status (current)
- gewollter Status (desired)
  - deklaratives Model (recommended)
  beschreibt das gewünschte Endresultat
  - basiert auf Controlloops
  - Kubernetes versucht permanent den gewollten Status zu erreichen
- non-persistent

Kubernetes - Deployment

- Deployments sind Objekte innerhalb der Kubernetes API und werden im state store mit Revisionsnummer gespeichert
- Deployments beschreiben den gewünschten Status
- ein Deployment Object wird je Pod benötigt
- Deployments kontrollieren Replicasets
- Deployments können multiple Replicasets des selben Pods steuern

Kubernetes - Replicasets

- Replicasets bieten Skalierbarkeit und Selfhealing Capabilities
- Replicasets spawnen die gewünschte Anzahl Pods wie beschrieben (Skalierbarkeit)
- Replicasets ersetzen failed Pods(replace & start -> self-heal)
- erlauben Rolling Updates und Rollbacks

Kubernetes - Stateful Sets

- verhalten sich wie Deployments
- speichern Daten persistent
  zum Beispiel Datenbanken oder Fileserver
- Stateful Sets sind eigenständige Objekte
  innerhalb der Kubernetes API
- verwenden Sticky Volumes und Names
- benötigen ggfs. manuelle Eingriffe im
  Fehlerfall(Datacorruption, Split Brain
  Situationen)

Kubernetes - Stateful Sets

- Stateful Sets starten Pods seriell und verwenden nummerierte Pods
- Eine Start/Stop Aktion muss abgeschlossen sein bevor mit dem nächsten Pod fortgefahren wird(ordered creation/deletion)
- Scaling erfolgt in Order beginnend bei der kleinsten Nummer
- Downscaling beginnt immer bei der höchstnummerierten Instanz

Kubernetes - rolling updates

- wenn ein Pod-Template aktualisiert und zur Kubernetes API gesendet wird, wird ein neues Replicaset erzeugt. Dabei wird eine neue Revision des Deployments im Statestore angelegt und gespeichert.
- Wenn Pods aus dem neuen Replicaset gestartet werden wird die gleiche Anzahl Pods im alten Replicaset heruntergefahren (replace)

Kubernetes - rollbacks

- im Fehlerfall kann auf die alte Revision des Deployments zurückgegangen werden

- Wenn Pods aus dem alten Replicaset gestartet werden wird die gleiche Anzahl Pods im neuen Replicaset heruntergefahren
(replace - analog zum rolling update)

Kubernetes - Services(Network)

- Pods werden als unzuverlässig betrachtet.
  Sie können abstürzen, neugestartet oder heruntergefahren werden.
  Pods are unreliable(ephemeral)
- Neue Replicasets/Pods verwenden andere Netzwerk-Addressen (IPs)
- Services bieten zuverlässige Netzwerkkonnektivität für einen oder mehrere Pods

Kubernetes - Services(Network)

- Services verwenden Label und Label-Selektorenum Traffic zwischen einzelnen Pods zu verteilen
- Label-Selektoren enthalten eine Liste aller Pods mit entsprechendem Label
- Traffic wird nur an healthy Pods weitergeleitet.

Kubernetes - Services(Network)

- Services sind eigenständige Objekte innerhalb der Kubernetes API
- bieten zuverlässiges Naming und Addressing
- bieten Loadbalancing auf einem Node (IPVS)
- überwachen die zugehörigen Pod-States und aktualisieren ihre Informationen automatisch
- Support für TCP Streams und UDP Datagrams

Kubernetes - internal DNS

- Die DNS Service-IP des Clusters wird „hardcoded“ in jeden Pod injiziert(interne resolv.conf des containers)
- ggfs. definierte search domains werden ebenfalls in die container resolv.conf injiziert
- jeder neue Pods wird beim Deployment automatisch im internen DNS registriert
- basiert auf CoreDNS

Kubernetes - Service Discovery

- Service Discovery
  für Verbindungen zu anderen Pods/Services
  benötigt jeder service:

  - den Namen des Services
  - Eine Möglichkeit den Namen aufzulösen. (dns-resolve)

Kubernetes - Service Discovery

- Service Registration:
  - Kubernetes benutzt sein Cluster-DNS als service registry
  - jeder Service registriert sich in der internen DNS Registry(mit Name, IP, Port)
  - es werden nur Services registriert keine individuellen Pods

Kubernetes - Discovery und Namespaces

- Jeder Cluster hat einen Addressraum gebunden an die Cluster Domain
- Namespaces partitionieren den Cluster Addressraum
- Namespaces entsprechen Subdomains

  example
  default name space:
  cluster.local (default)
  partitioned space:
  prod.cluster.local
  def.cluster.local

Kubernetes - Discovery und Namespaces

- Objekt Namen müssen eindeutig innerhalb eines Namespace sein
- dürfen mehrfach verwendet werden über Namespace-Boundaries hinweg
- Pods…
  - dürfen shortnames innerhalb des gleichen Namespace verwenden
  - müssen FQDNs über Namespace-Grenzen hinweg verwenden

Kubernetes - Service Discovery

- Service Registration Steps:
  1. Post new Service to Kube API
  2. request wird authentifiziert, authorisiert und wenn entsprechend der Cluster Policies geprüft
  3. Dem Service wird eine virtuelle Cluster IP zugewiesen
  4. Ein Endpoint Objekt wird angelegt (enthält die Liste der Pods zu denen der Service Traffic weiterleitet
  5. das pod netzwerk wird entsprechend konfiguriert
  6. Service Name und IP werden im Cluster DNS registriert

  (abschliessend wird der neue Service im Cluster Store gespeichert)

Kubernetes - Storage

- Kubernetes unterstützt diverse Storageoptionen:
  - CIFS/SMB
  - NFS
  - iSCSI
  - CEPH/rbd
  - etc. pp.
- Storage wird als Volume im Cluster registriert
- Storageprovider werden via Container Storage Interface Plugin eingebunden

Kubernetes - Persistent Volume Subsystem

- Resourcen:
  - Persistent Volumes (PV)
  - Persistent Volume Claims (PVC)
    - Zugriffsanspruch auf ein Volume
    - Parallele Claims auf das gleiche Volume müssen den selben Mode haben.
  - Storage Classes (SC)
    - gruppieren Storage Volumes
    - erlauben die Definition verschiedenartiger Storagetypen

Kubernetes - Config Maps

- sind eigenständige Objekte der Kubernetes-API
- werden benutzt um bestimmte nicht-sensitive Konfigurationsinformationen ausserhalb von Pods zu speichern(server-configs, hostnamen, umgebungsvariablen etc)
- … und zur Laufzeit zu injizieren(via ConfigMap Volumes, EnvVars)
- key/value based
- Kubernetes Native Apps können die Config Map via Kubernetes API abfragen

Kubernetes - Application Rollout

- erstellen/besitzen einer Anwendung
- Containerisierung der Anwendung
- registrieren des Containers in der lokalen Registry
- Definieren des Kubernetes Pods unter Rückgriff auf den Container aus der Registry
- Deployment
- Expose