KUBERNETES-MANAGEMENT MIT CLUSTER API (CAPI) AUF OPENSTACK

Kevin Honka | @piratehonk@norden.social

Rabbithole Meetup Berlin | 2026-03-06

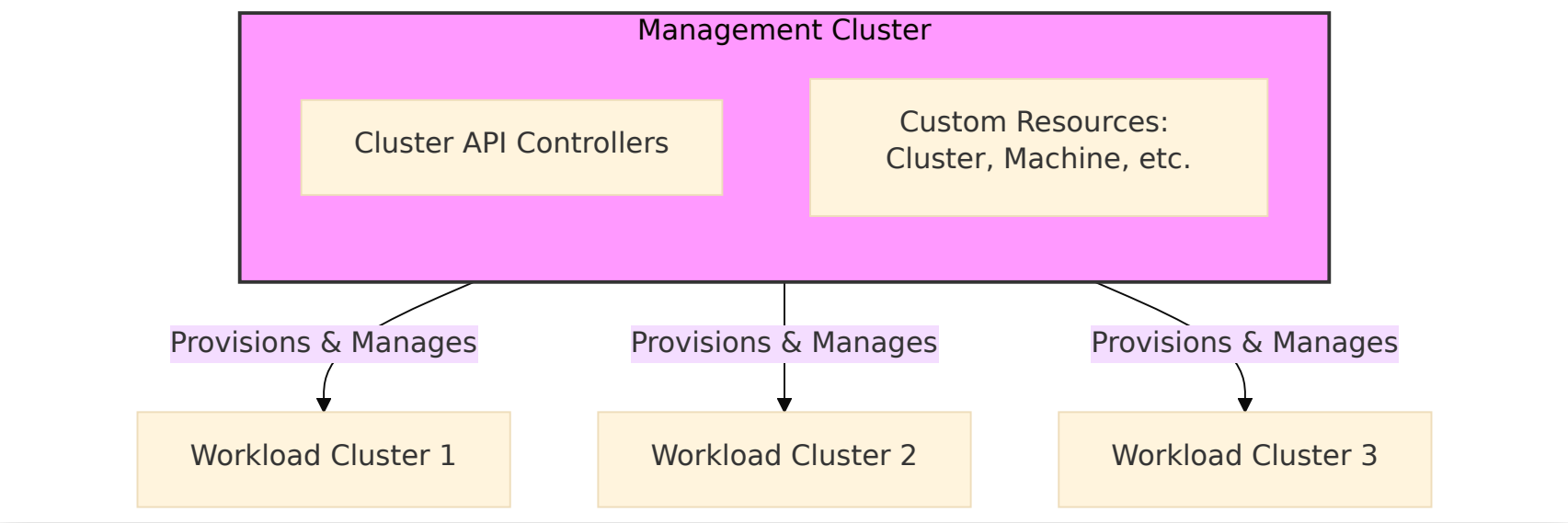
IT SYSTEMS

WAS IST CLUSTER API?

- Ein Kubernetes-Subprojekt
- Nutzt **Deklarative APIs** zur Verwaltung von Kubernetes-Clustern
- Kubernetes-in-Kubernetes (Management Cluster verwaltet Workload Cluster)
- Erlaubt konsistente Abläufe über verschiedene Provider hinweg (AWS, Azure, vSphere, OpenStack...)

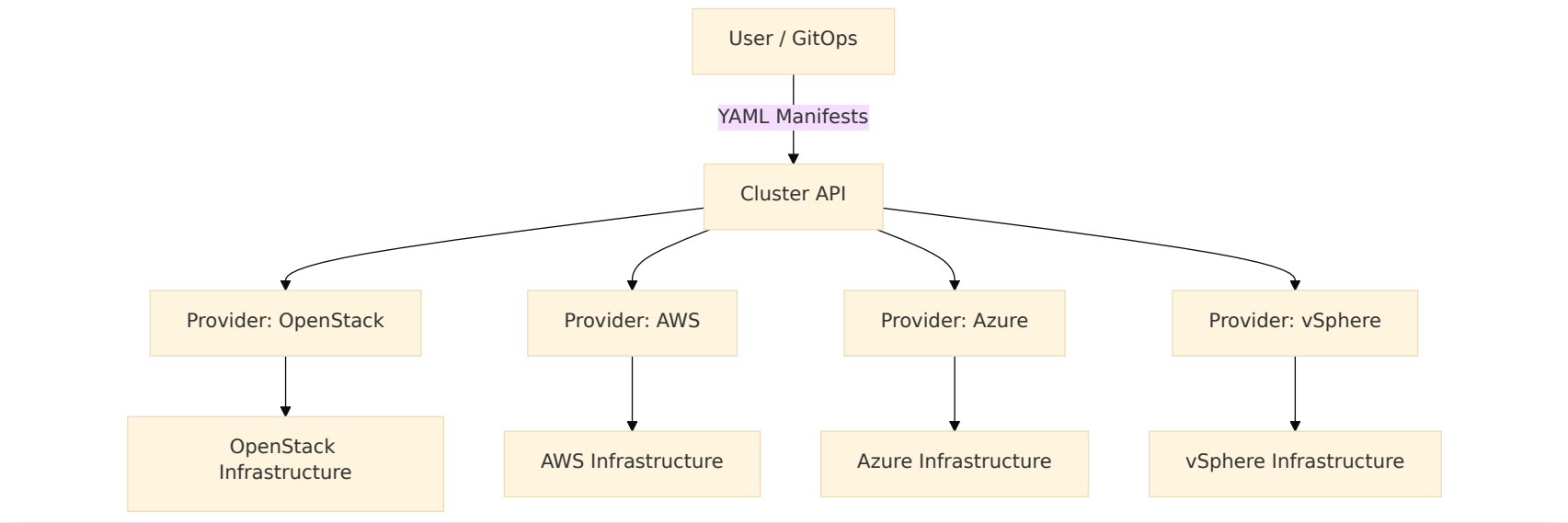
IT SYSTEMS

MANAGEMENT VS WORKLOAD CLUSTER



IT SYSTEMS

MULTI-PROVIDER ABSTRAKTION



IT SYSTEMS

WARUM CLUSTER API?

- **Infrastruktur als Code (IaC)** für den gesamten Cluster-Lifecycle
- Automatisierte Skalierung und Upgrades
- Selbstheilung (Self-healing) der Nodes
- Reduzierung von manuellem Konfigurationsaufwand

IT SYSTEMS

UNSERE CAPI-ARCHITEKTUR

- **Provider:** OpenStack (CAPO - Cluster API Provider OpenStack)
- **Bootstrap-Prozess:** Kind (Kubernetes in Docker) als temporärer Management-Cluster
- **Self-Management:** Pivot-Prozess verschiebt CAPI-Ressourcen in den Workload-Cluster

IT SYSTEMS

KOMPONENTEN

- **ORC (OpenStack Resource Controller):** Erlaubt die deklarative Verwaltung von OpenStack-Ressourcen direkt via K8s
- **Cilium:** Als CNI (Container Network Interface)
- **OpenStack CCM & CSI:** Für Cloud-Integration (LoadBalancer, Storage)

IT SYSTEMS

BOOTSTRAPPING-PROZESS

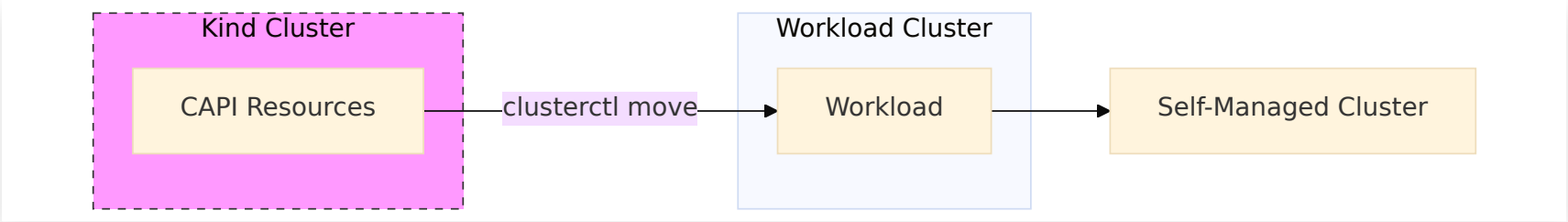
1. **Kind Cluster erstellen:** Dient als initialer Ankerpunkt
2. **CAPI initialisieren:** Installation der Controller (CAPI Core, Bootstrap Kubeadm, Infrastructure OpenStack)
3. **Workload Cluster definieren:** YAML-Manifeste für Control Plane und Worker Nodes

IT SYSTEMS

DER PIVOT (SELF-MANAGEMENT)

- Sobald der Workload-Cluster läuft, wird er selbst zum Management-Cluster
- `clusterctl move` verschiebt alle Objekte
- Kind-Cluster wird gelöscht -> Infrastruktur ist nun autark

IT SYSTEMS


Kind Cluster
CAPI Resources
clusterctl move
Workload Cluster
Workload
Self-Managed Cluster

IT SYSTEMS

AUTOMATISIERUNG

Wir nutzen Terraform zur Orchestrierung der CAPI-Manifeste:

- Definierte Templates für Control-Plane und Worker
- Einfache Konfiguration von Flavors und Images
- Integration von Addons (Ingress, Monitoring, RBAC)

IT SYSTEMS

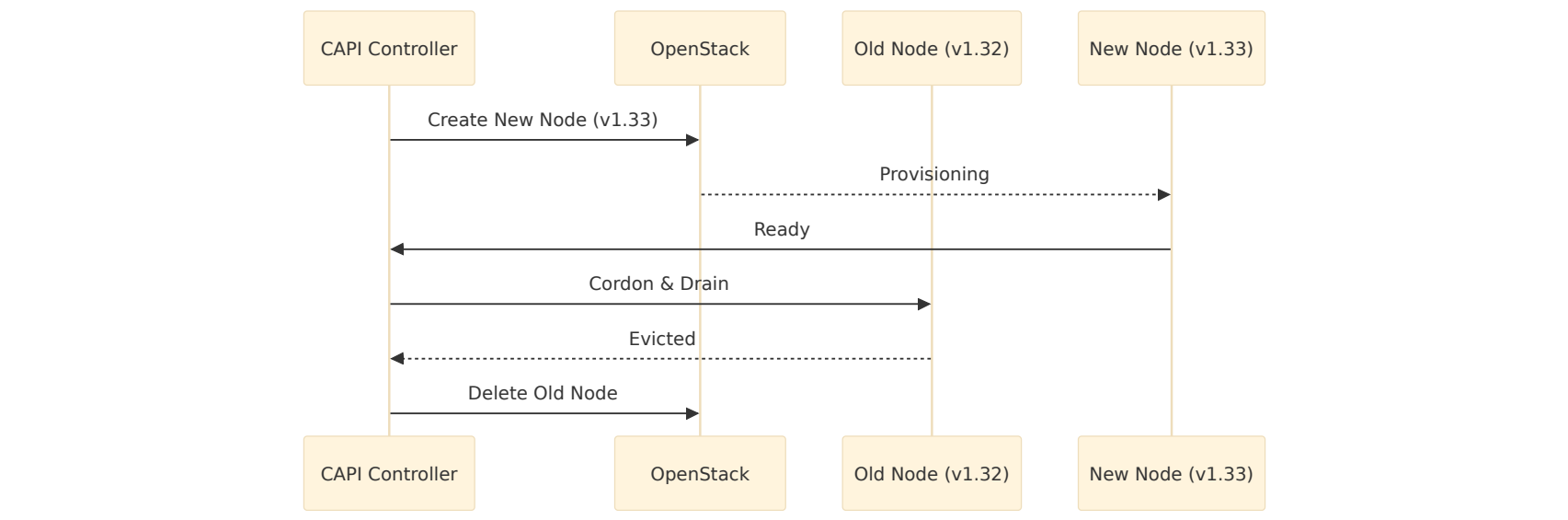
BEISPIEL: UPGRADE-STRATEGIE

Rolling Upgrades durch Änderung des Templates:

```
worker_templates = {
  "v1" = { flavor = "gp2a.16", image_name = "kube-v1.32.4" },
  "v2" = { flavor = "gp2a.16", image_name = "kube-v1.33.2" }
}
active_worker_template = "v2"
kubernetes_version     = "v1.33.2"
```

- CAPI erkennt die Änderung
- Erstellt neue Nodes, drained alte Nodes
- Zero-Downtime Upgrades der Infrastruktur

IT SYSTEMS


CAPI Controller
OpenStack
Old Node (v1.32)
New Node (v1.33)
Create New Node (v1.33)
Provisioning
Ready
Cordon & Drain
Evicted
Delete Old Node
CAPI Controller
OpenStack
Old Node (v1.32)
New Node (v1.33)

IT SYSTEMS

VORTEILE IM BETRIEB

- **Reproduzierbarkeit:** Jeder Cluster sieht identisch aus
- **Skalierbarkeit:** MachineDeployments erlauben einfaches Hochskalieren
- **Automatische Skalierung:** Worker können anhand von Metriken automatisch skaliert werden
- **Sicherheit:** Schnelles Einspielen von neuen Images (Security Patches) durch Rolling Updates
- **Standardisierung:** Gleiche Tools für Applikation und Infrastruktur (`kubectl`, GitOps)

IT SYSTEMS

FAZIT & AUSBLICK

- Cluster API macht Kubernetes-Infrastruktur "langweilig" (im positiven Sinne)
- Fokus verschiebt sich von "Wie baue ich einen Cluster" zu "Wie nutze ich ihn sinnvoll"
- Nächste Schritte: Integration von Cluster Autoscaler

IT SYSTEMS