

corewire

Docker & Kubernetes Troubleshooting

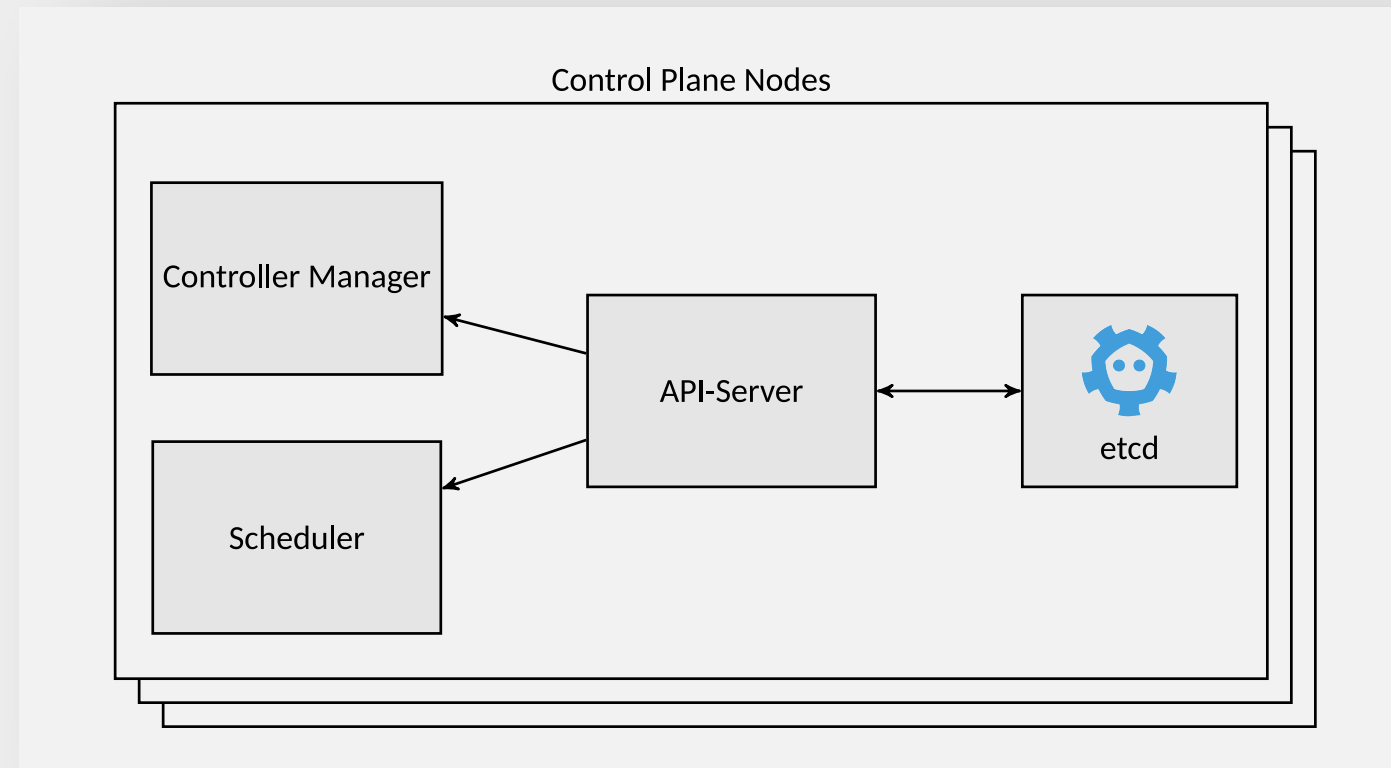
Kubernetes Admin Troubleshooting

Folien-Hinweis

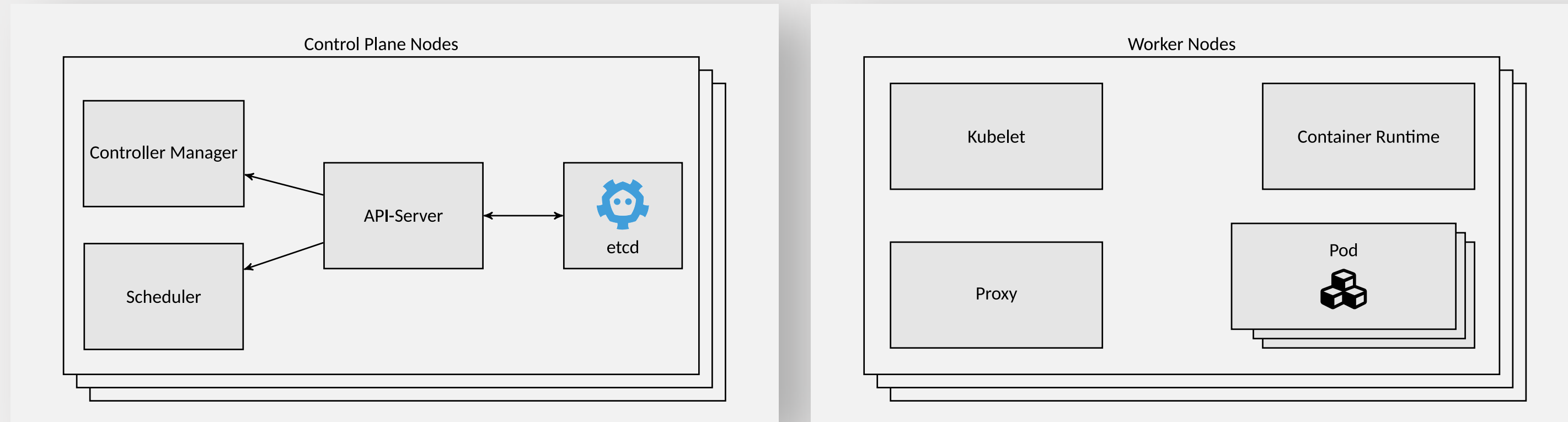
- `Space`, `Page down`: Nächste Folie
- `Page up`: Vorherige Folie
- `ESC`, `o`: Übersicht

[Zur Kapitelübersicht](#)

Kubernetes-Komponenten



Kubernetes-Komponenten

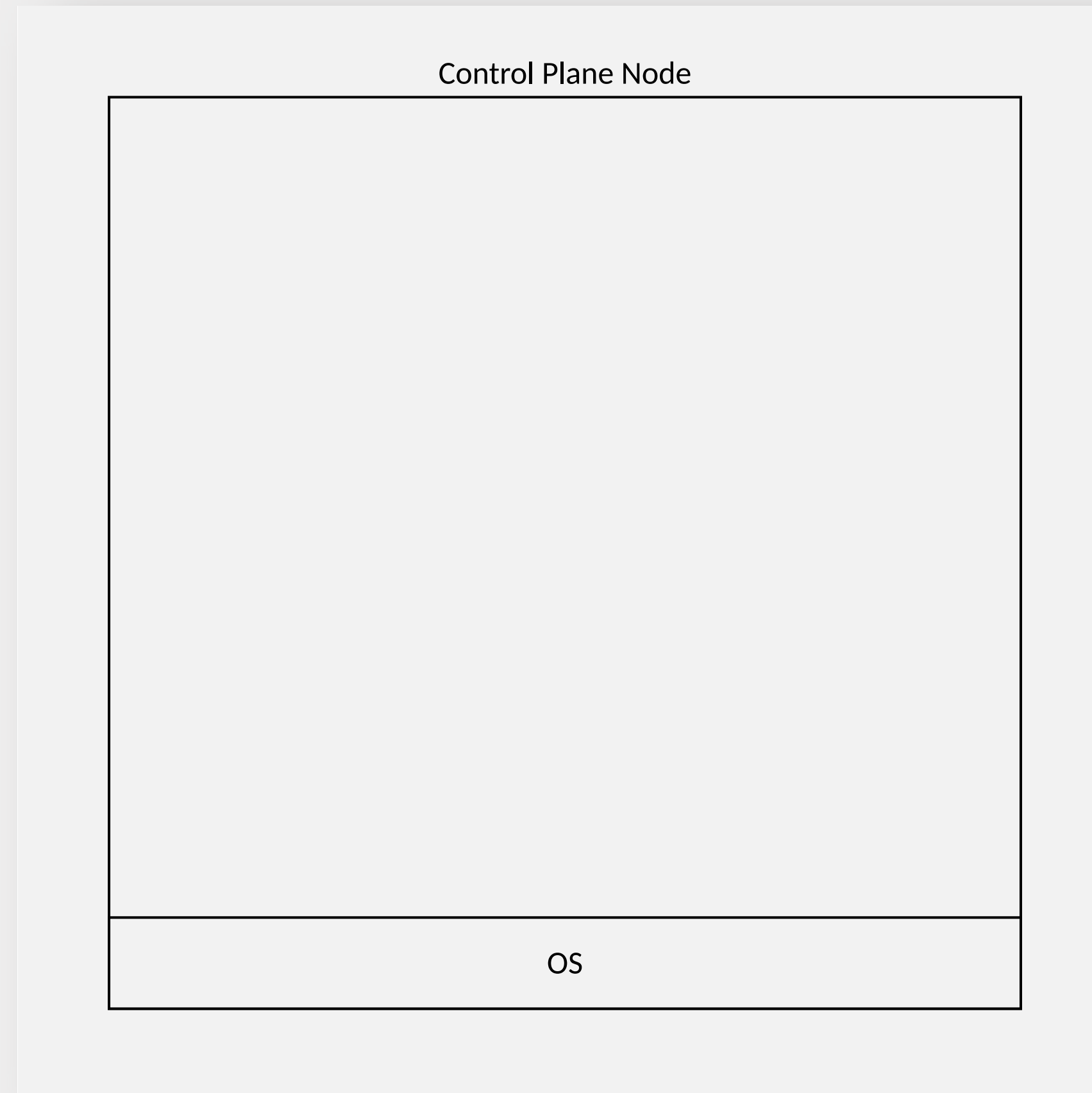


Kubernetes-Komponenten

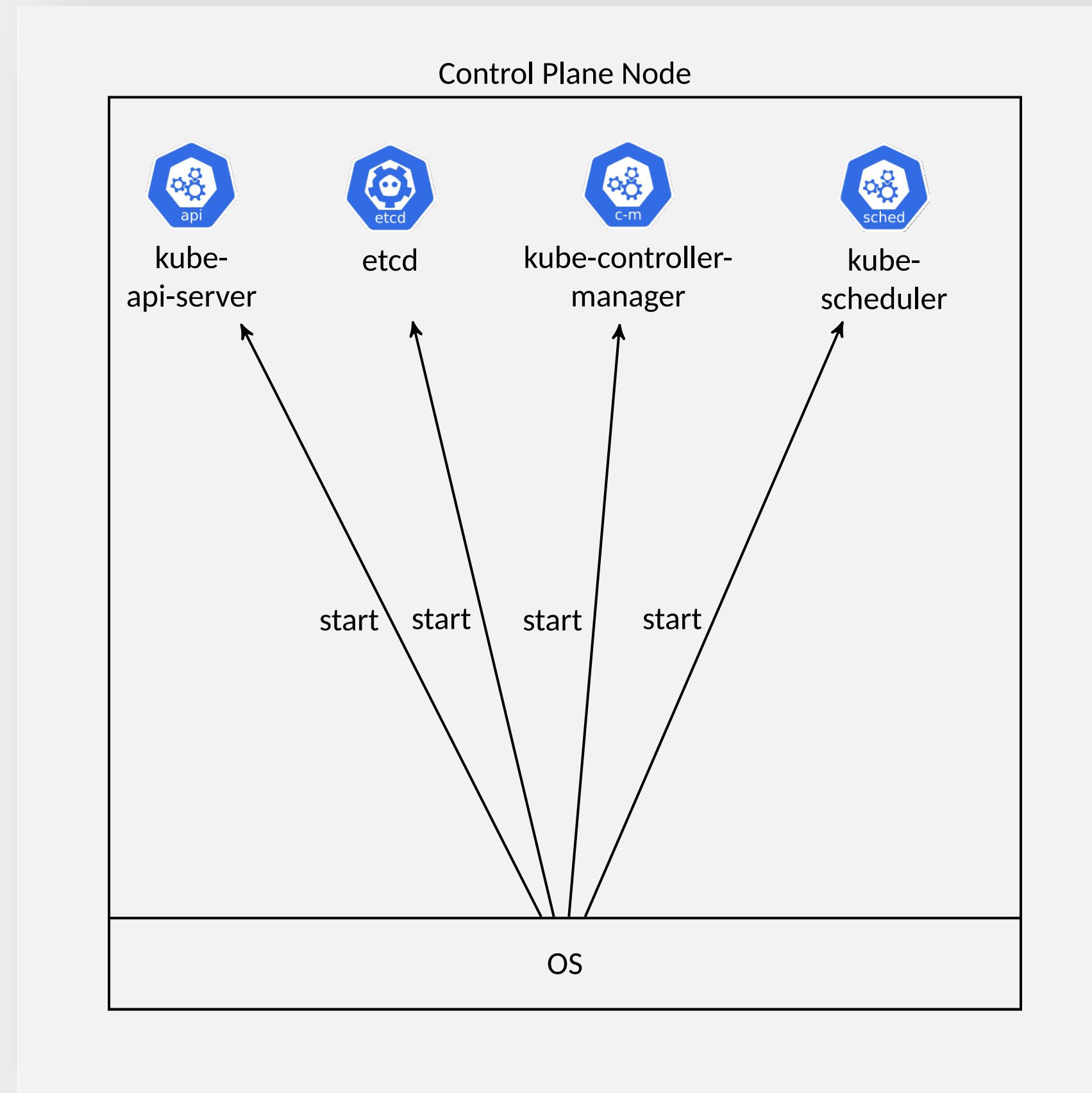
- die Control Plane verwaltet und steuert
- die Worker führen Anwendungen und Aufgaben aus

Wie starten wir die Komponenten?

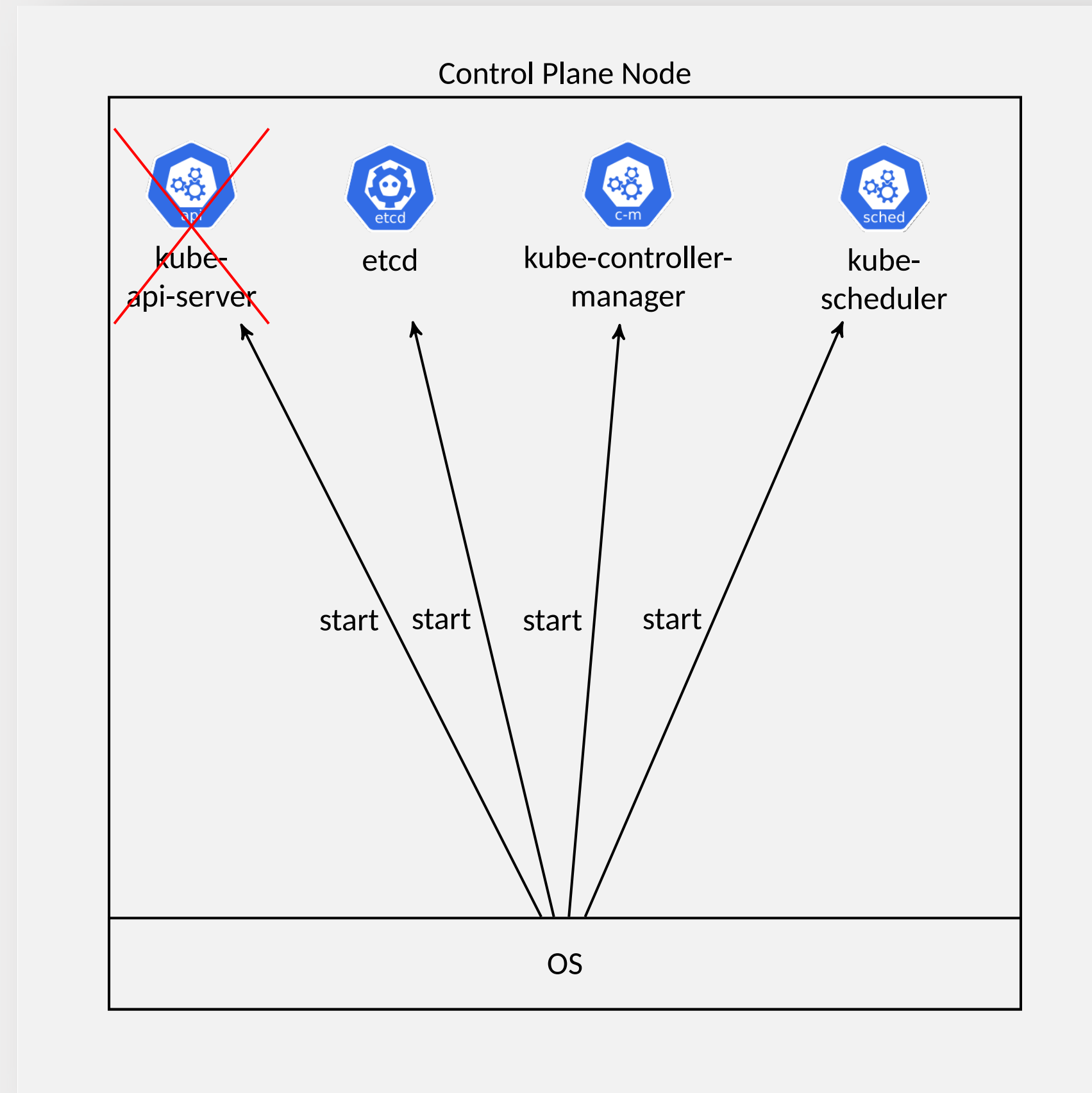
Manuell aufgesetzte Controlplane



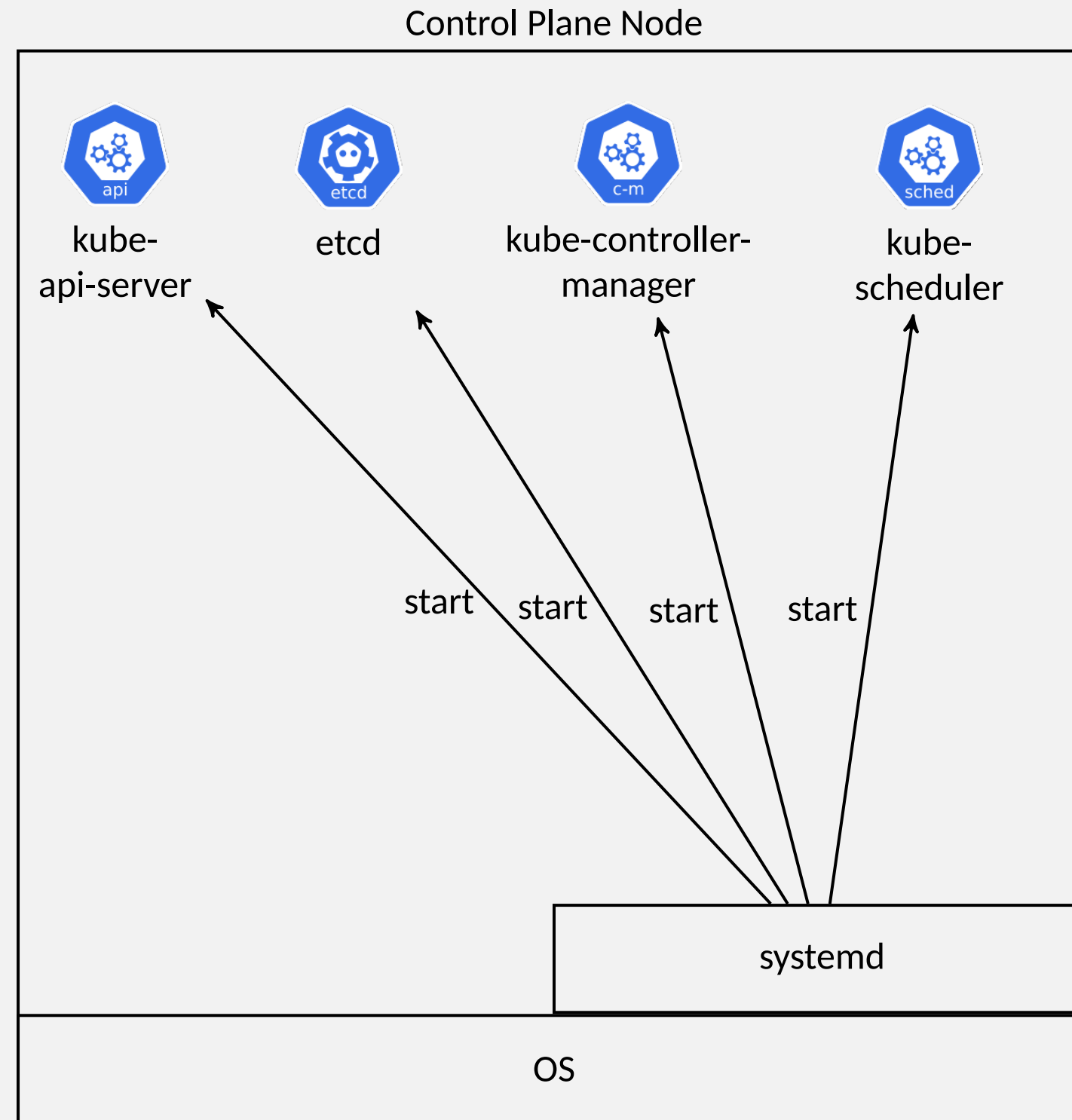
Manuell aufgesetzte Controlplane



Manuell aufgesetzte Controlplane



Manuell aufgesetzte Controlplane



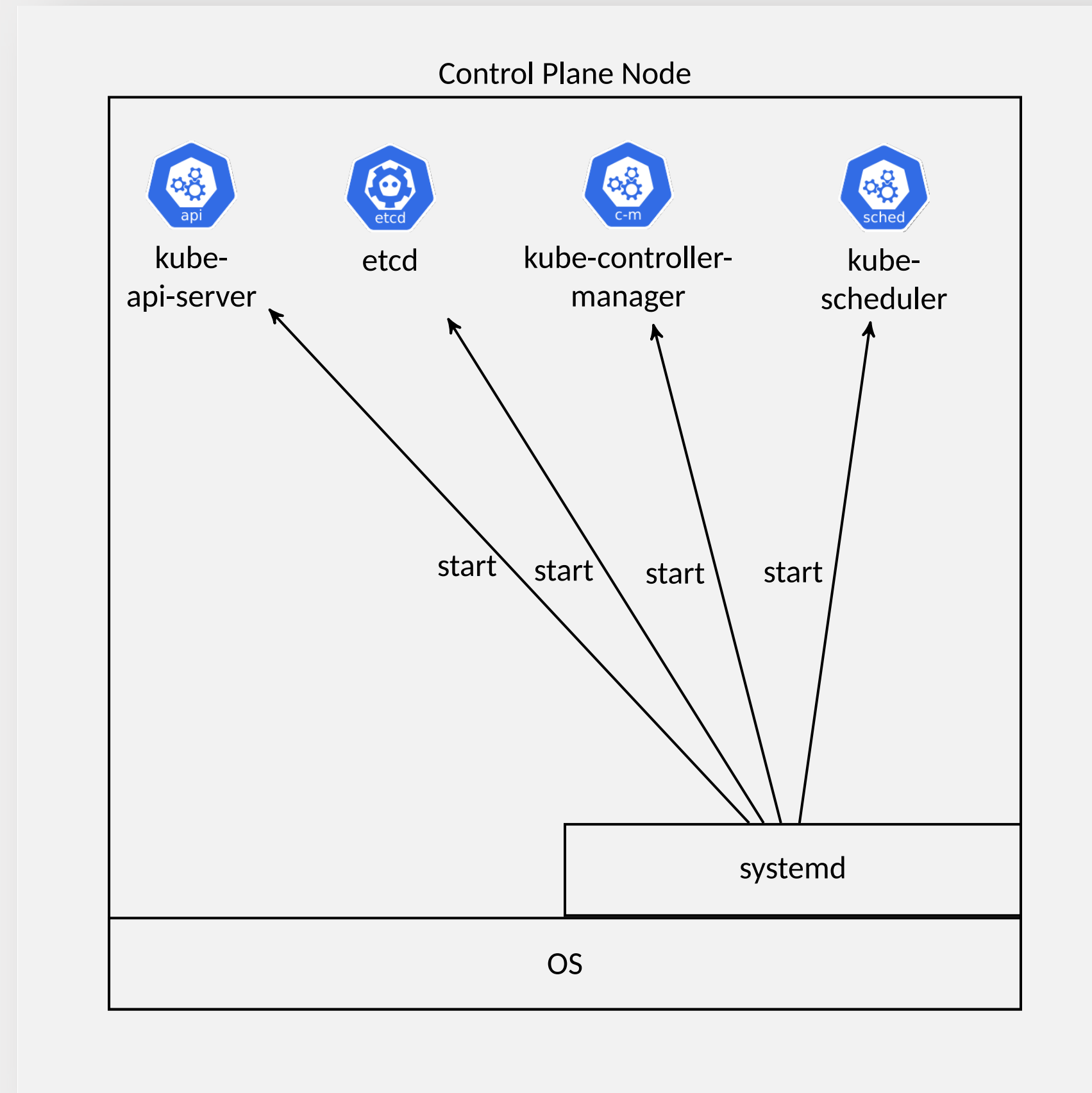
Systemd

- Überwacht Prozesse
- Keine Abhängigkeit zur Container-Runtime

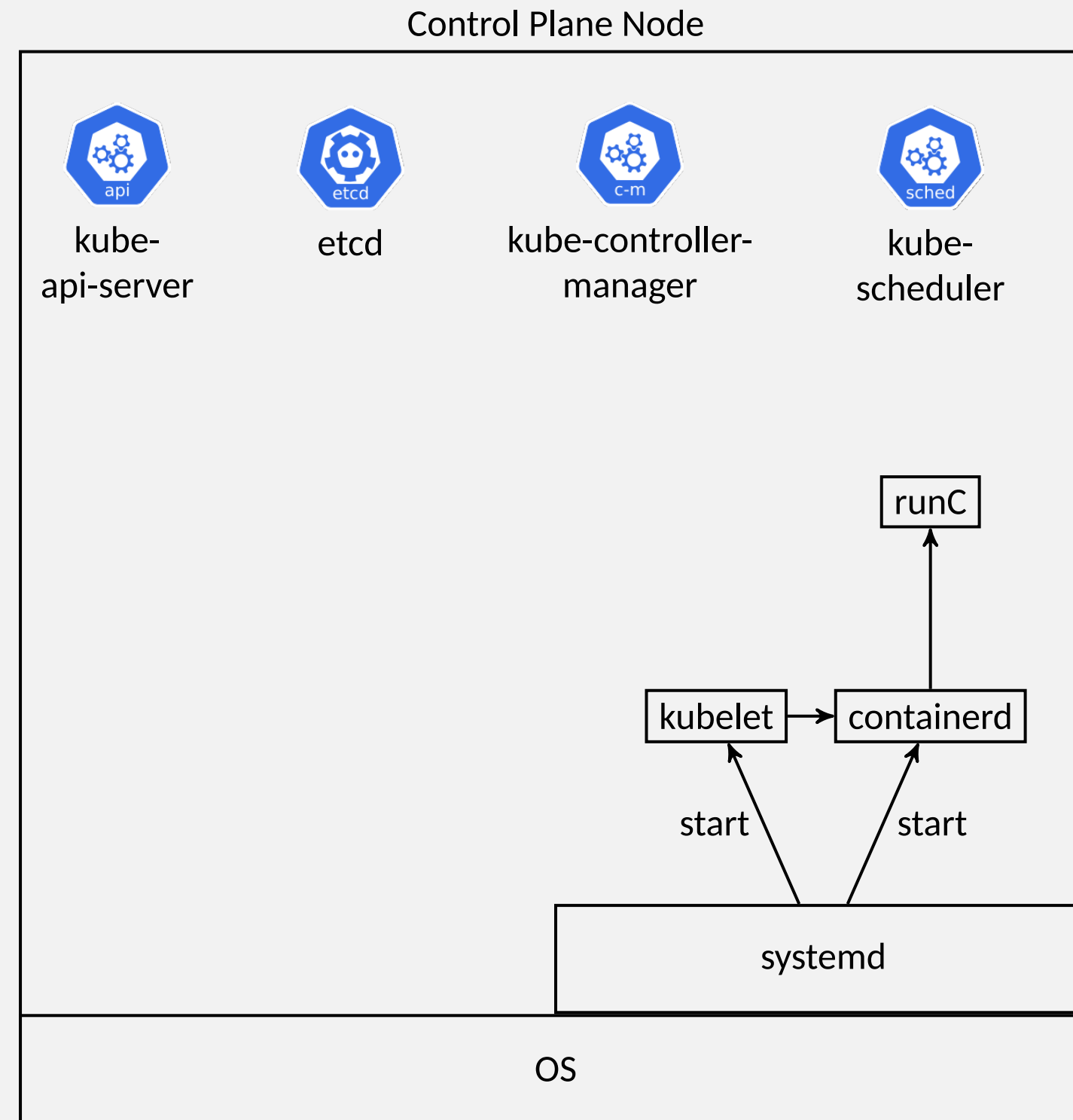
Systemd - Nachteile

- Separates Tooling

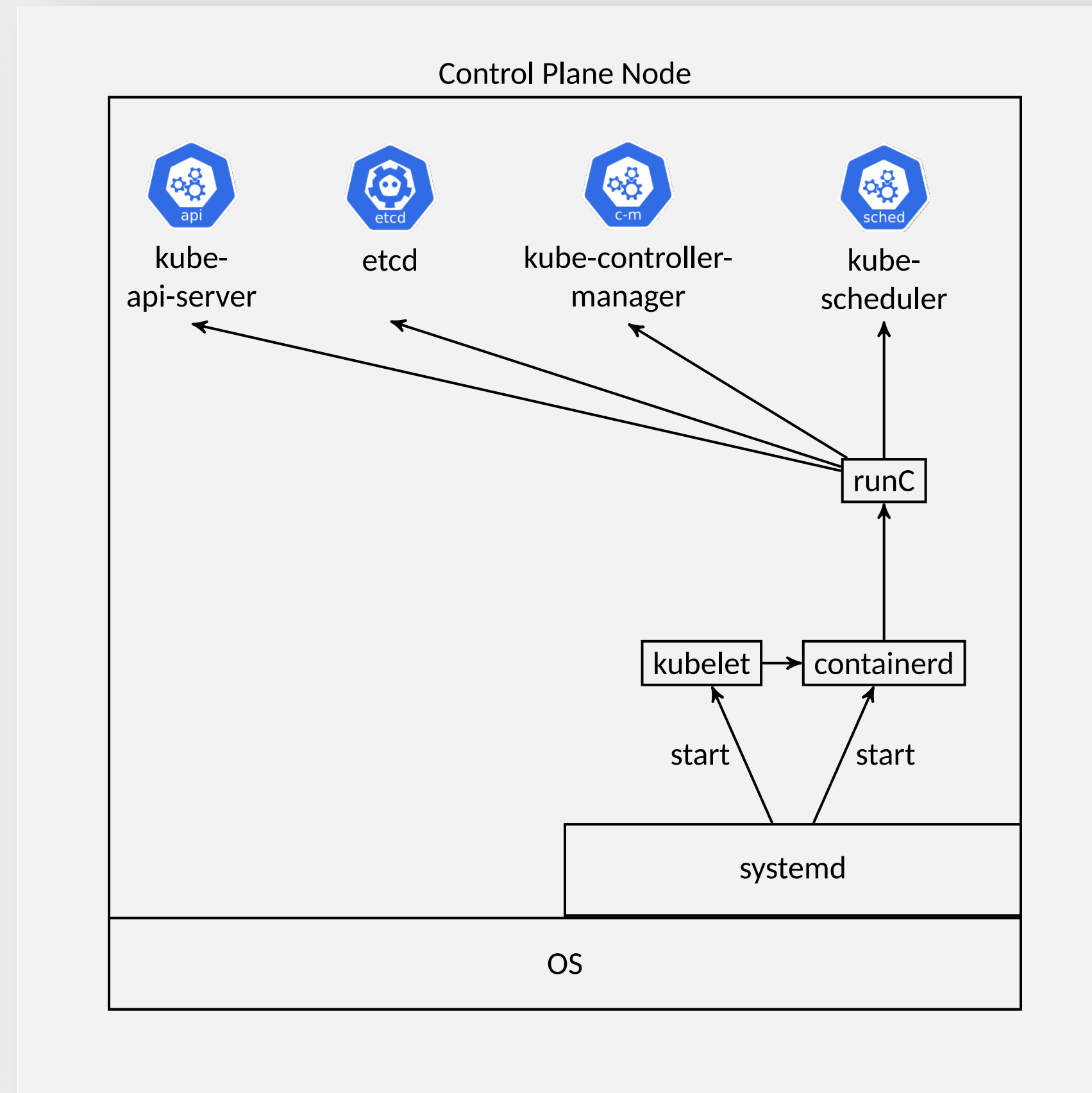
Controlplane als spezialisierter Worker



Controlplane als spezialisierter Worker



Controlplane als spezialisierter Worker



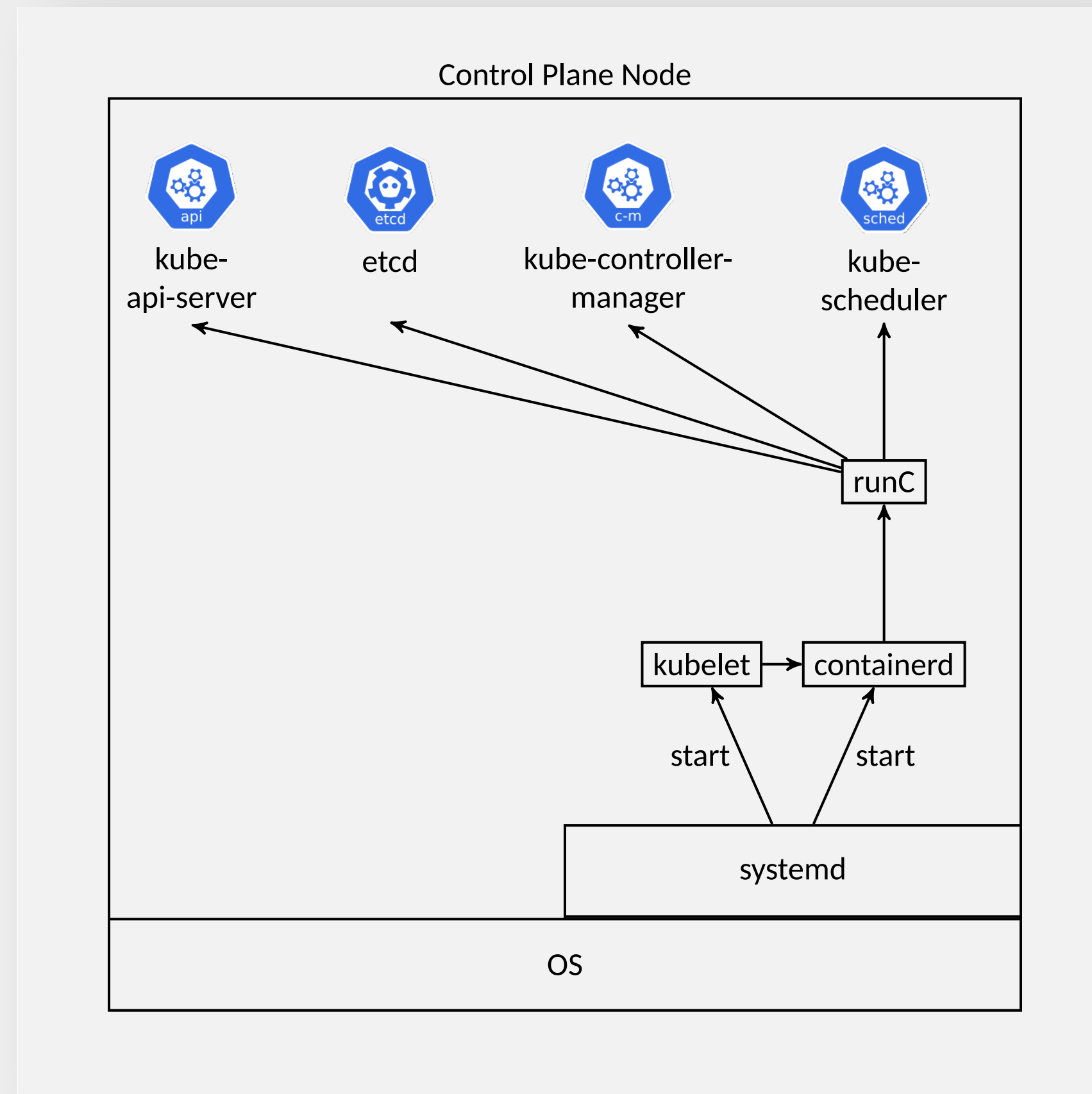
Controlplane als spezialisierter Worker

- Von kubelet verwaltet
- Einheitliches Tooling:
 - Logging/Monitoring
 - Healthchecks
 - Rollouts/Deployments

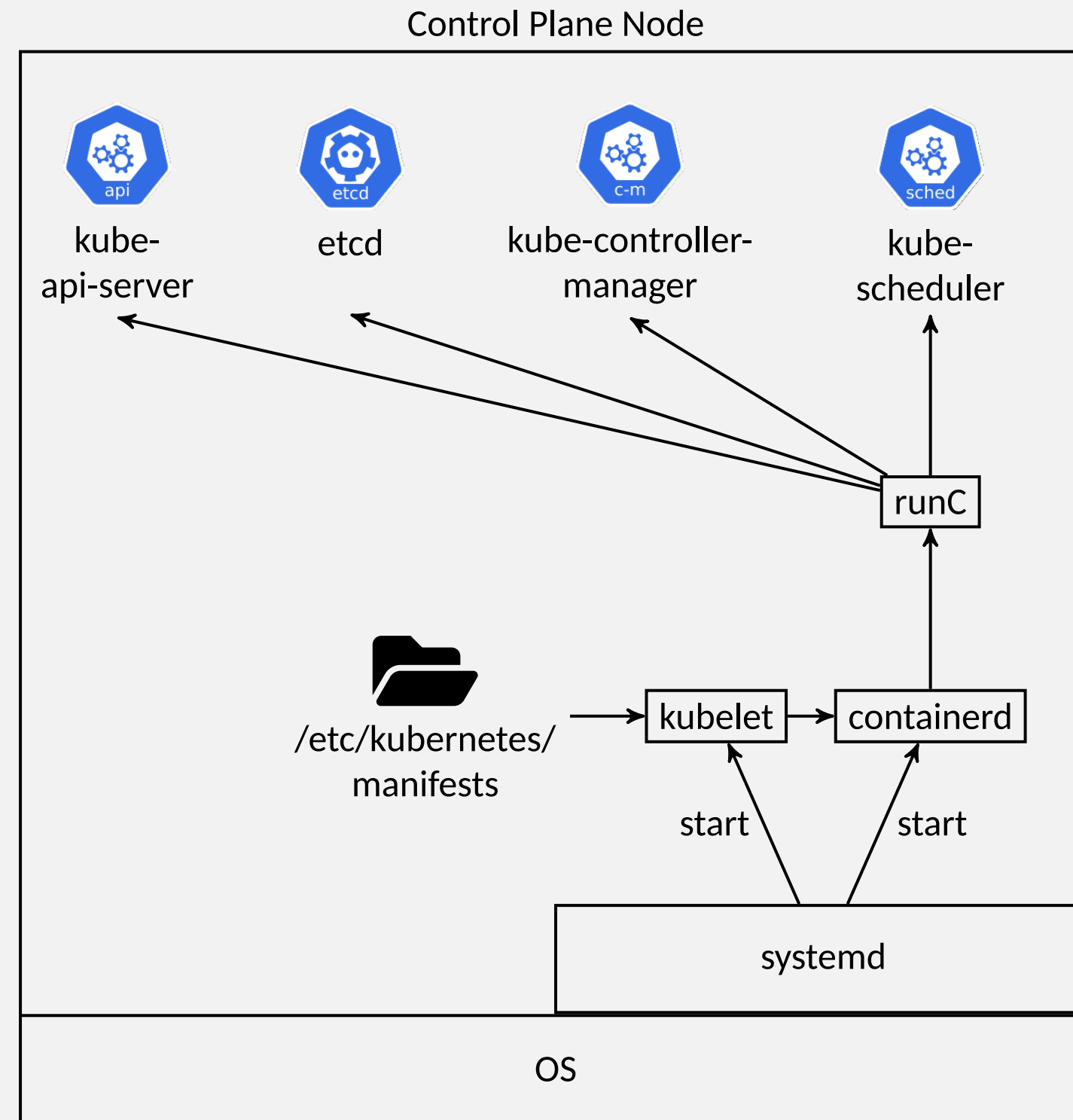
Controlplane als spezialisierter Worker

Wo liegt die PodSpec, wenn Services als Pod betrieben wird?

Controlplane als spezialisierter Worker



Controlplane als spezialisierter Worker



Controlplane als spezialisierter Worker

- Controlplane Knoten als Teil des Clusters
- Kubelet überwacht Prozesse
- Ermöglicht zusätzliche Pods/Workload auf der Controlplane

Static Pods

- PodSpec in `/etc/kubernetes/manifests`
- Werden lokal auf dem Knoten ausgeführt
- Umgehen Api-Server / Scheduler
- Read-only Mirror-Pod für Api-Server

Statische Pods debuggen

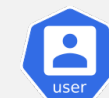
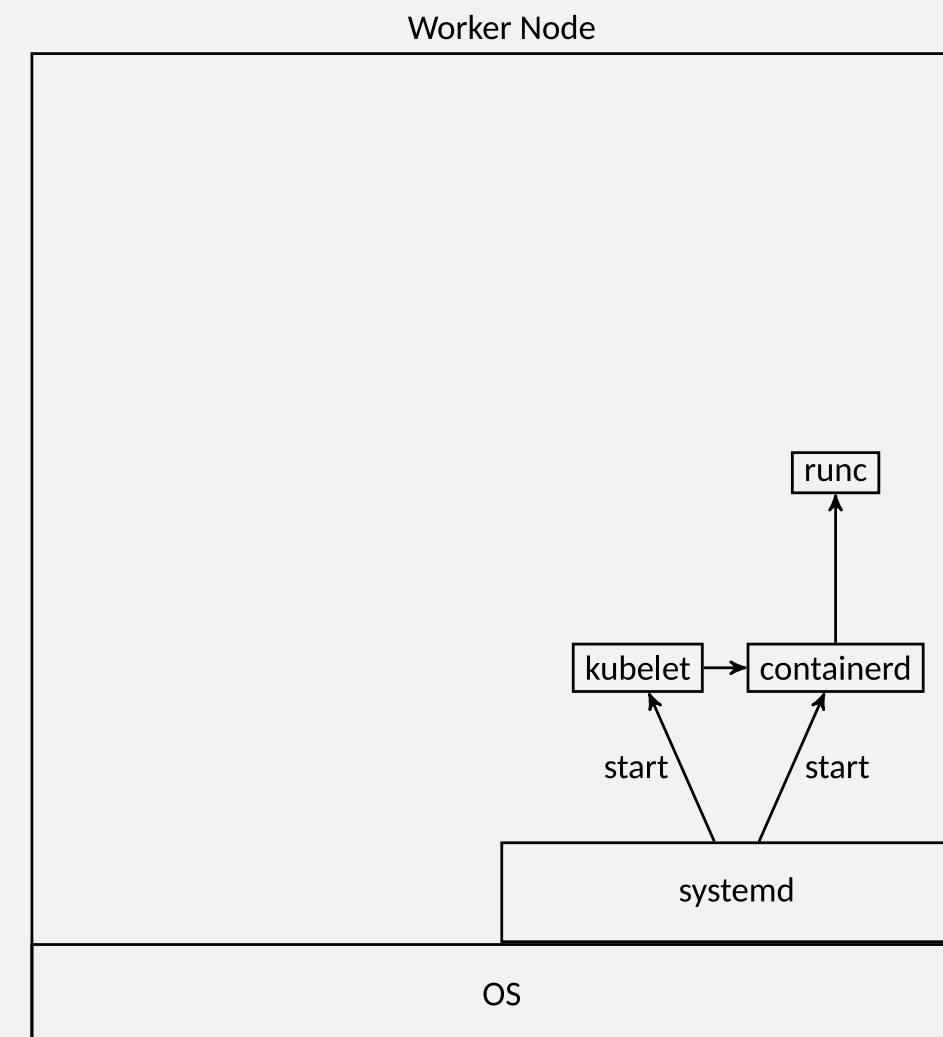
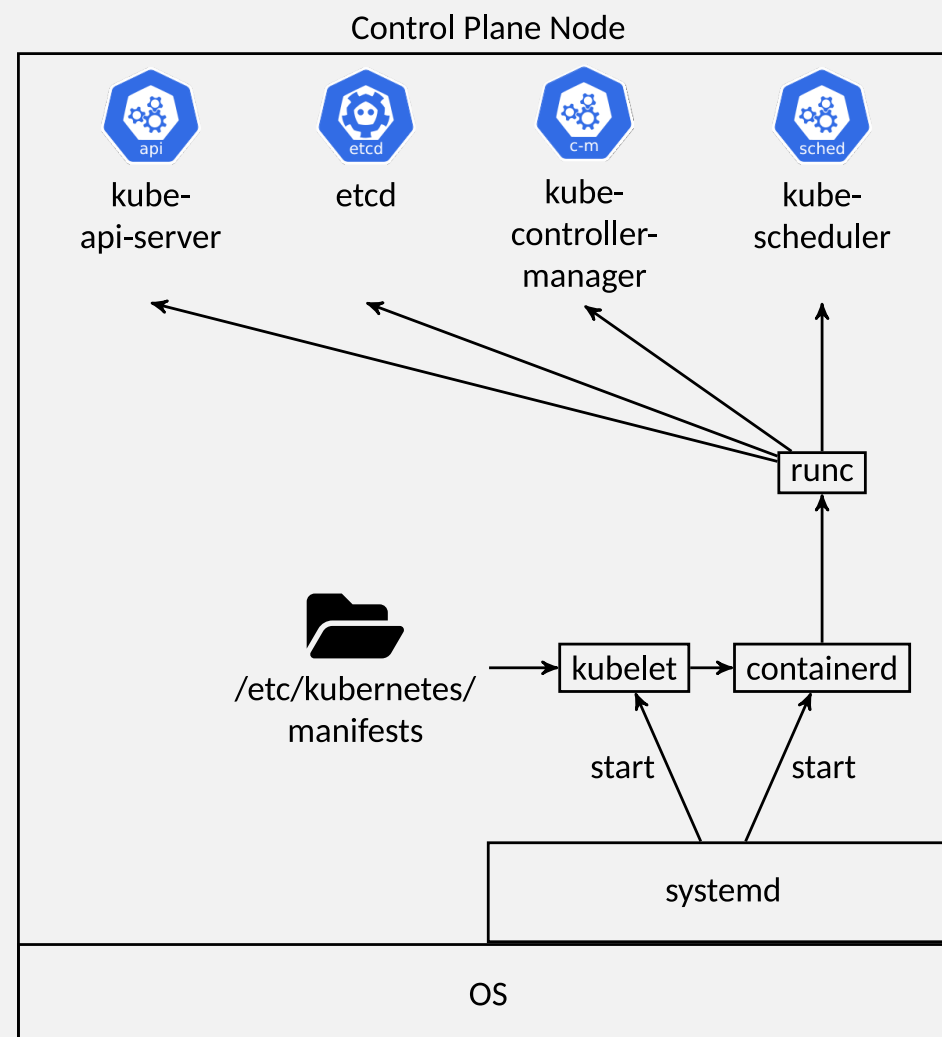
- `kubectl get pods -n kube-system`
- `crictl ps -a` auf der Controlplane

Wenn der API-Server nicht läuft, muss `crictl` verwendet werden, da `kubectl` auf den API-Server angewiesen ist.

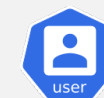
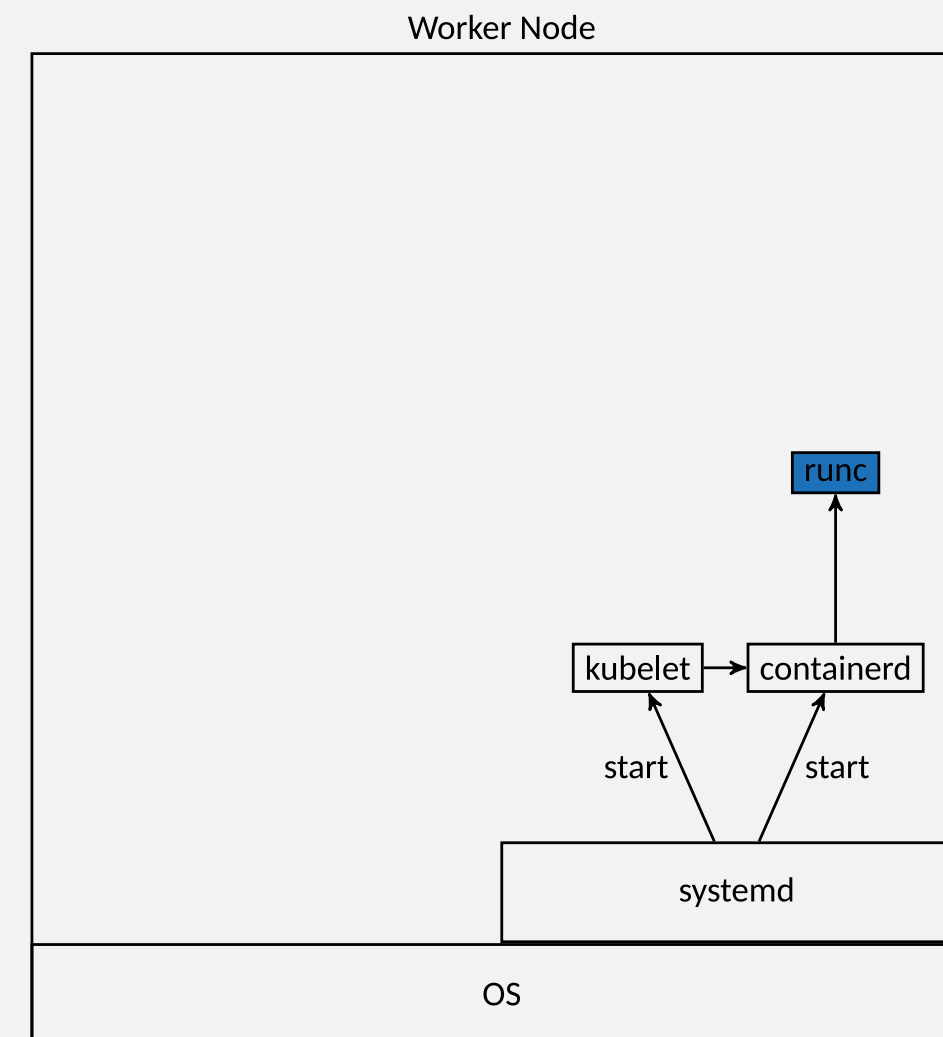
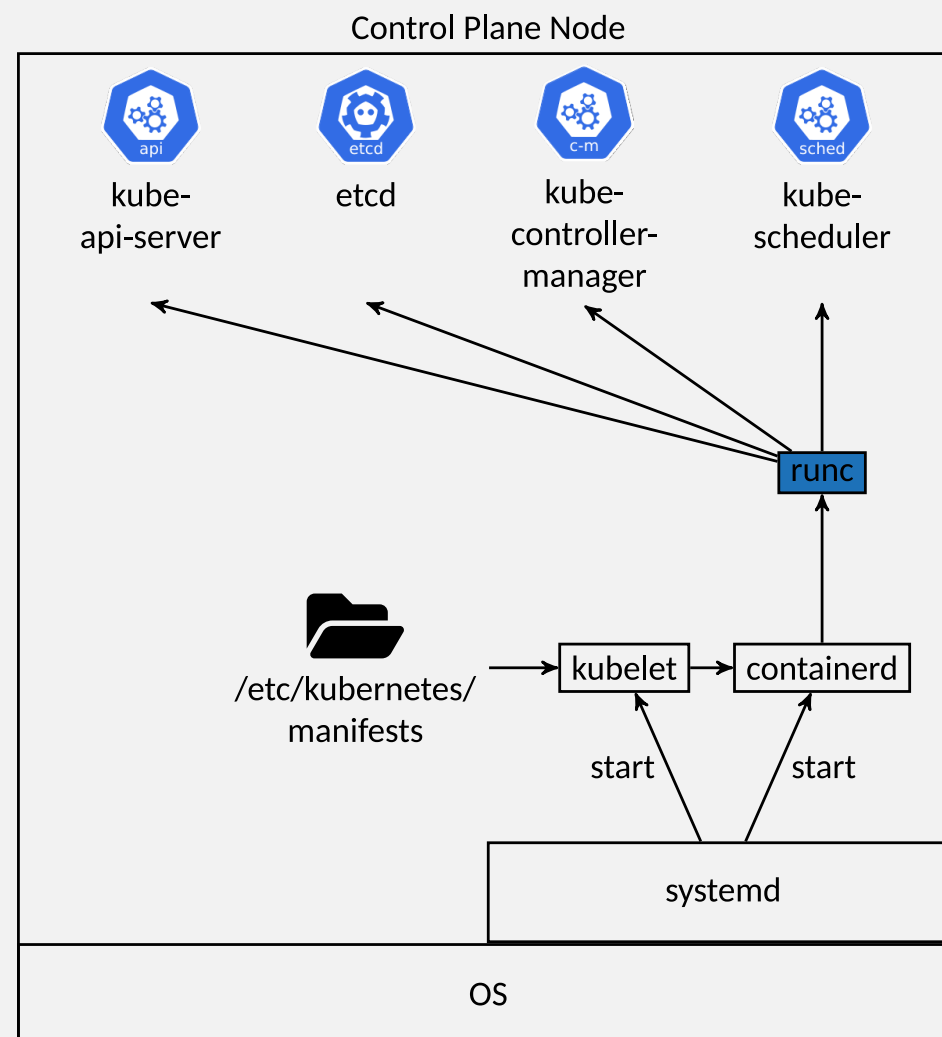
crictl

- CLI-Tool zur Containerverwaltung
- Syntax ähnlich zu `docker`
- Für die Fehlersuche bei statischen Pods oder der Container Runtime
- Unterstützt verschiedene Container-Runtimes

runc



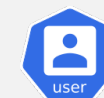
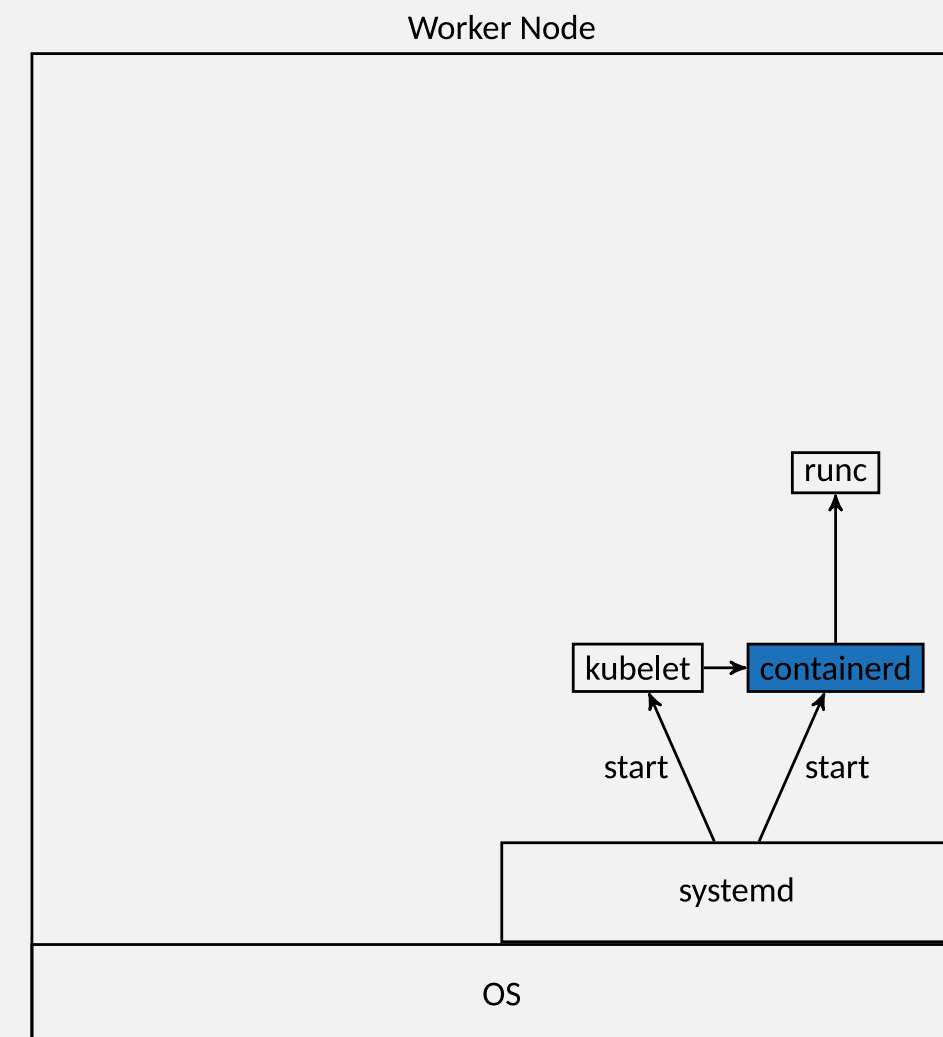
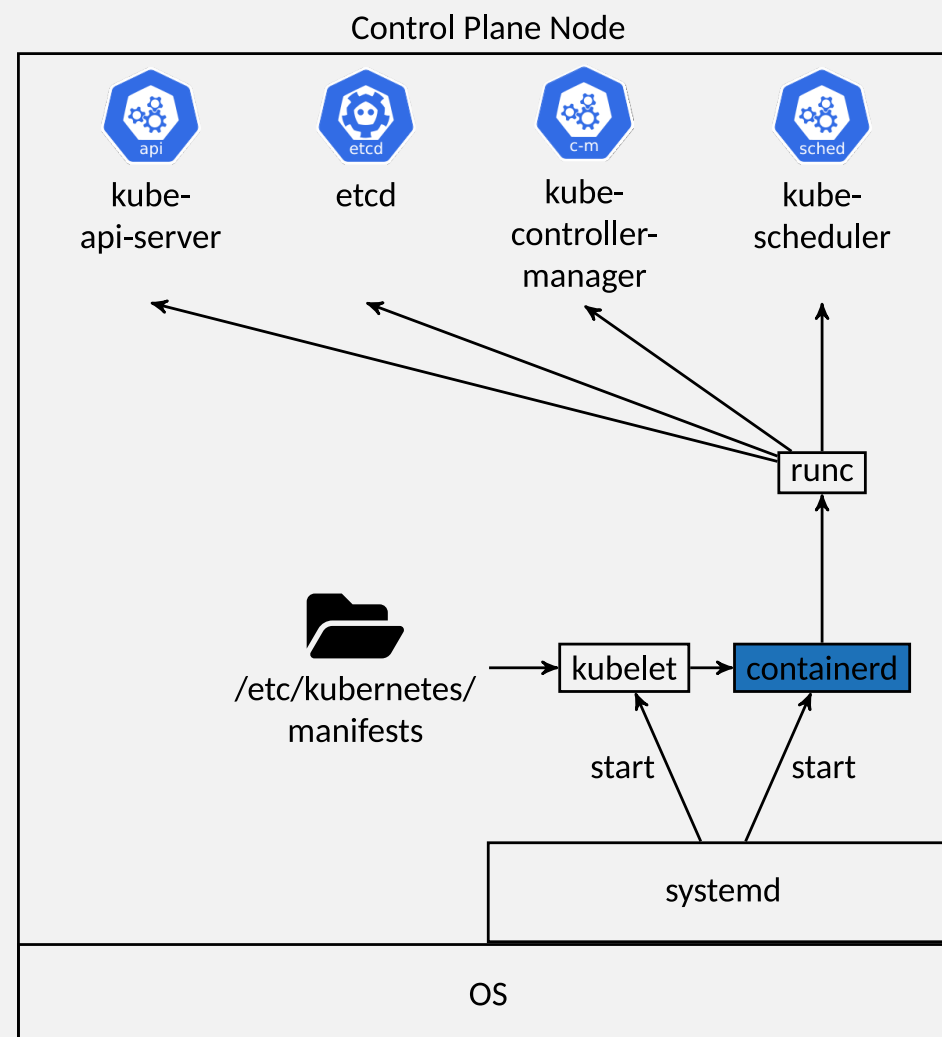
runc



runc

- CLI-Tool zum Starten und Ausführen von Containern
- Implementiert die Open Container Initiative (OCI) Spezifikation
- Wird von containerd als Standard-Runtime verwendet

containerd



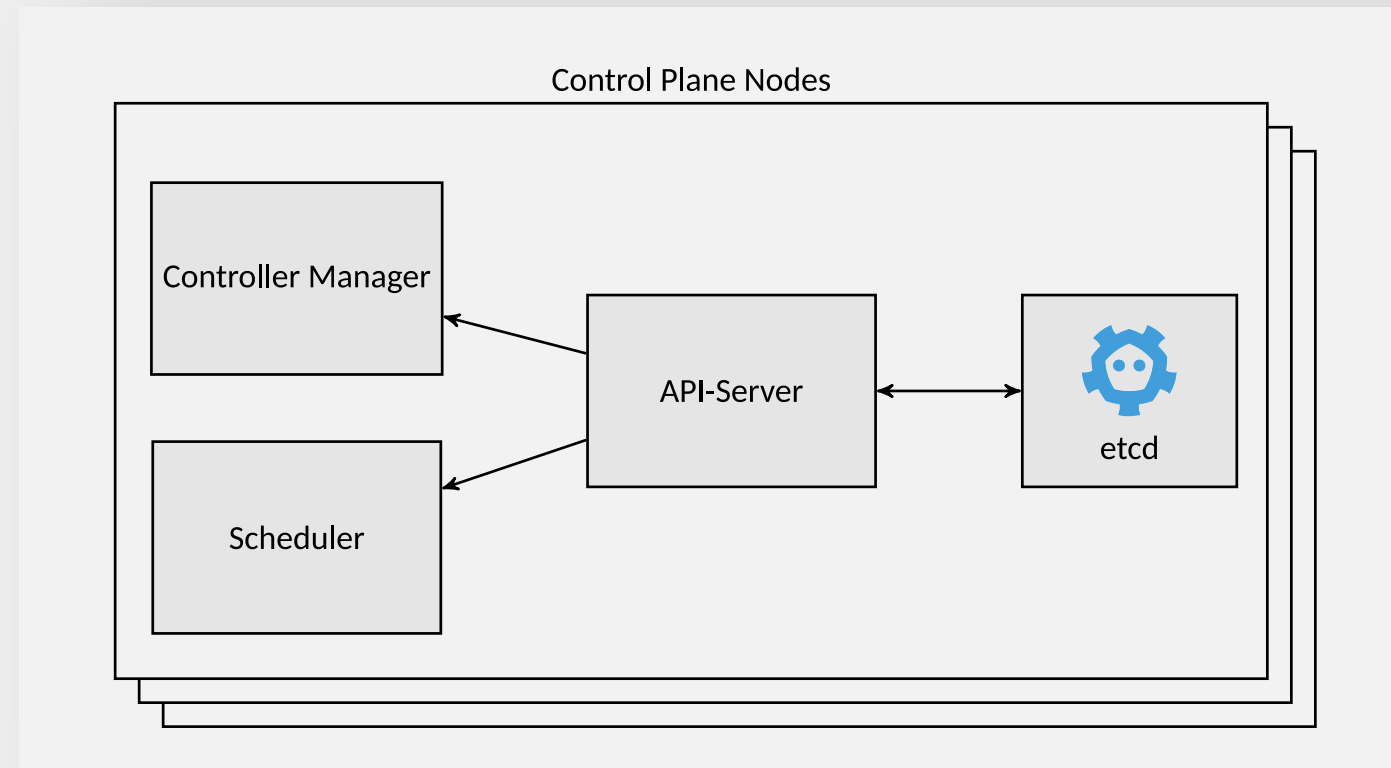
containerd

- Container Runtime
- Verwaltet den Lebenszyklus von Containern
 - Image Transfer & Storage
 - Container Execution & Supervision
 - Storage & Network Interface Management
- Nutzt Low-Level Runtimes (z.B. runc, gvisor, crun)
- Kubernetes als default Container Runtime

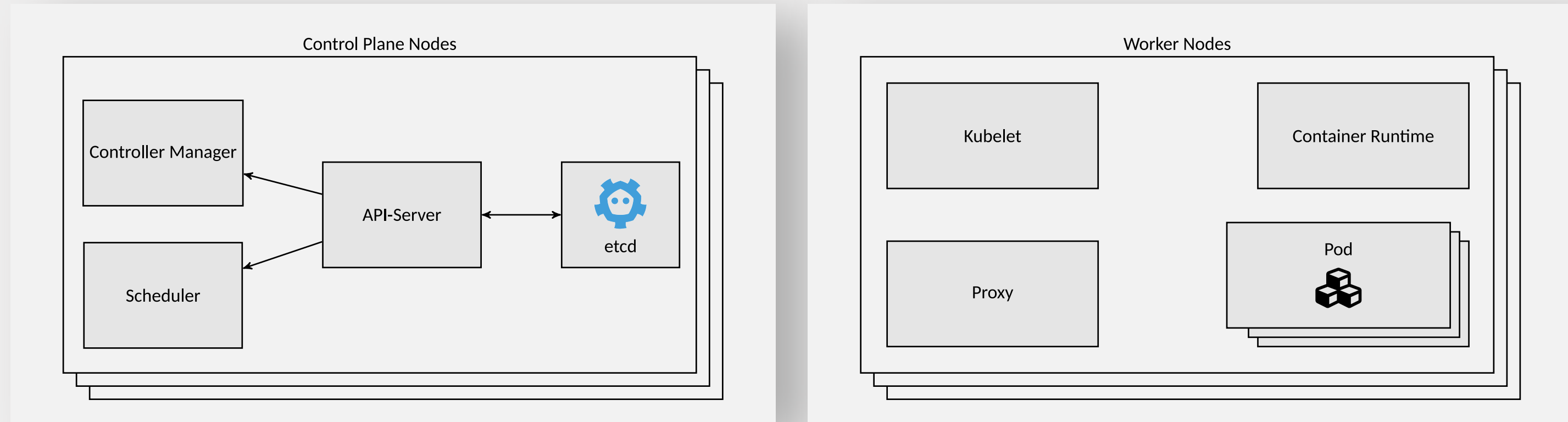
Network Plugins

- Container Network Interface (CNI) Plugins
- Ermöglichen Netzwerkfunktionen für Container
- Basis Plugins für lokale Netzwerke

Kubernetes-Komponenten



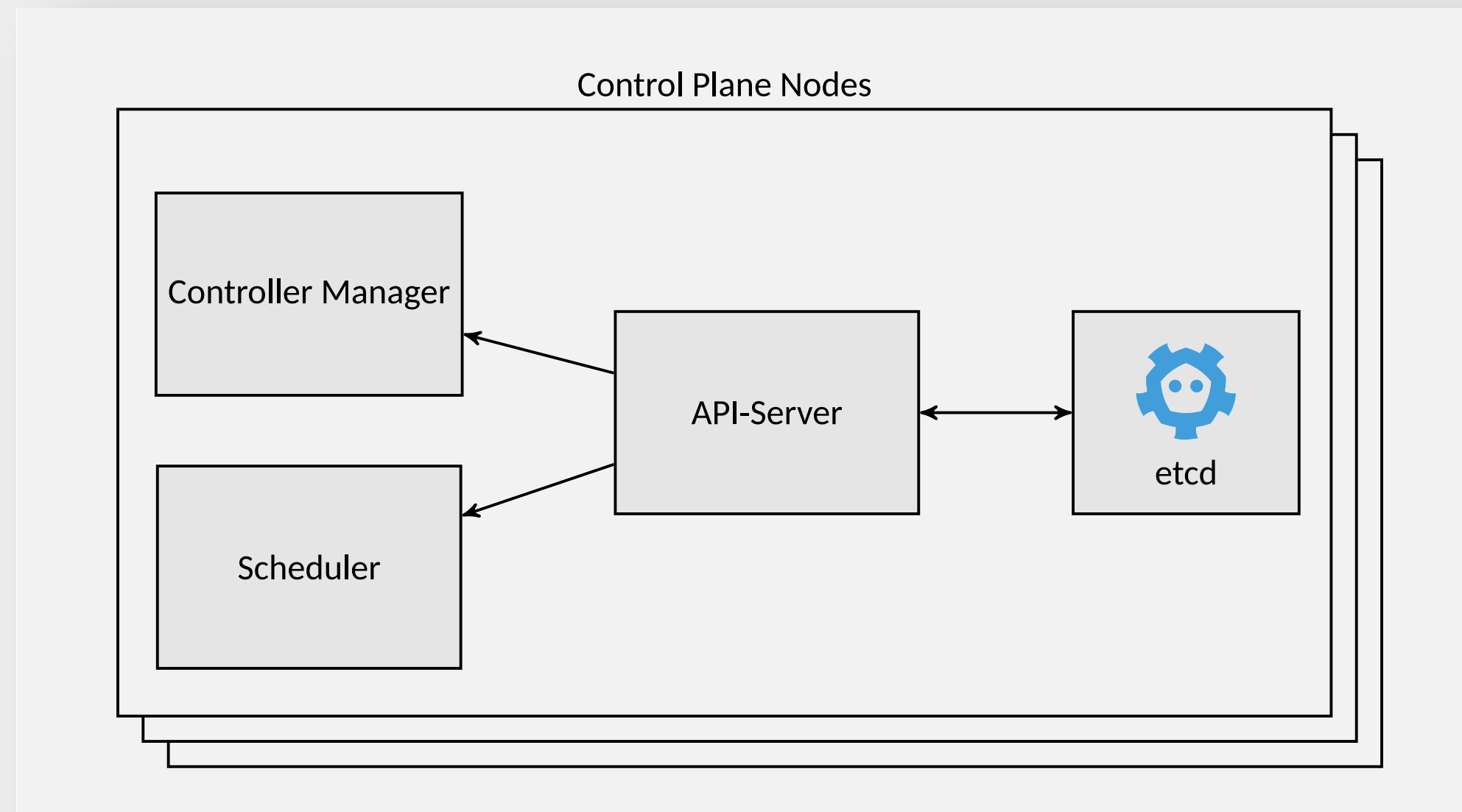
Kubernetes-Komponenten



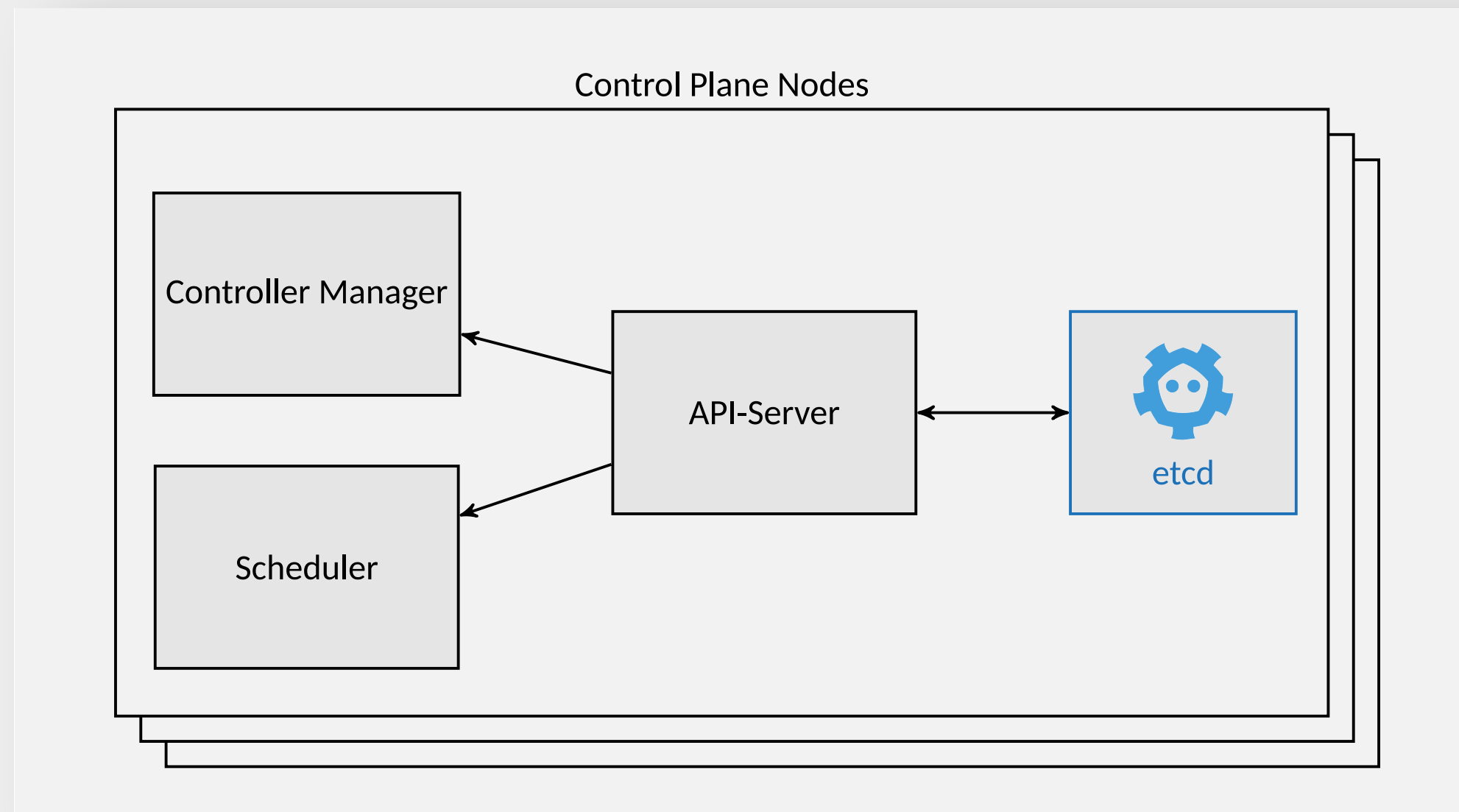
Kubernetes-Komponenten

- die Control Plane verwaltet und steuert
- die Worker führen Anwendungen und Aufgaben aus

Control Plane



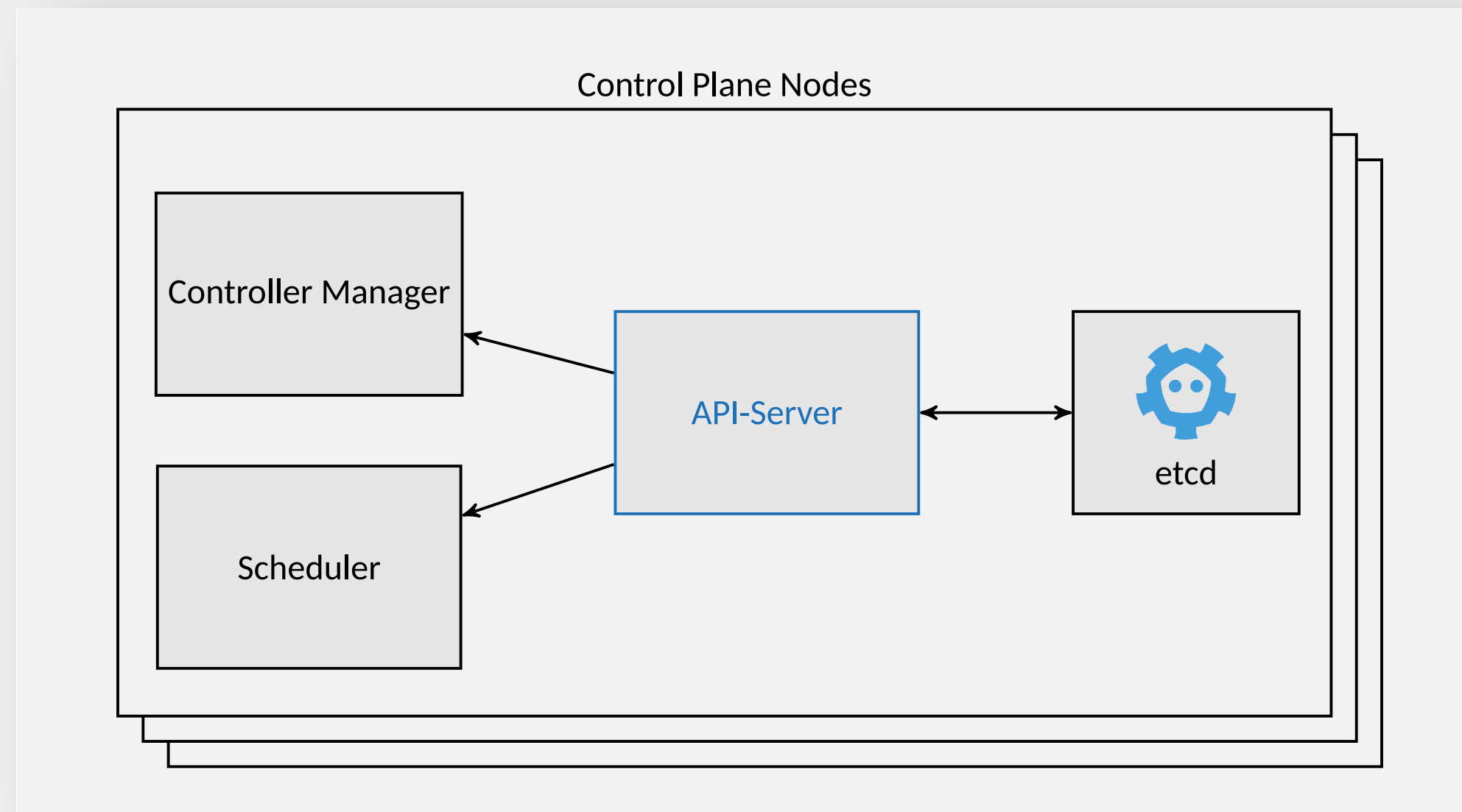
Control Plane - API-Server



API-Server

- stellt die Kubernetes-API bereit
- wird von kubectl über HTTP angesprochen
- kommuniziert mit Kubelets auch über HTTP
- schreibt die Zielkonfiguration, Secrets etc. in den etcd

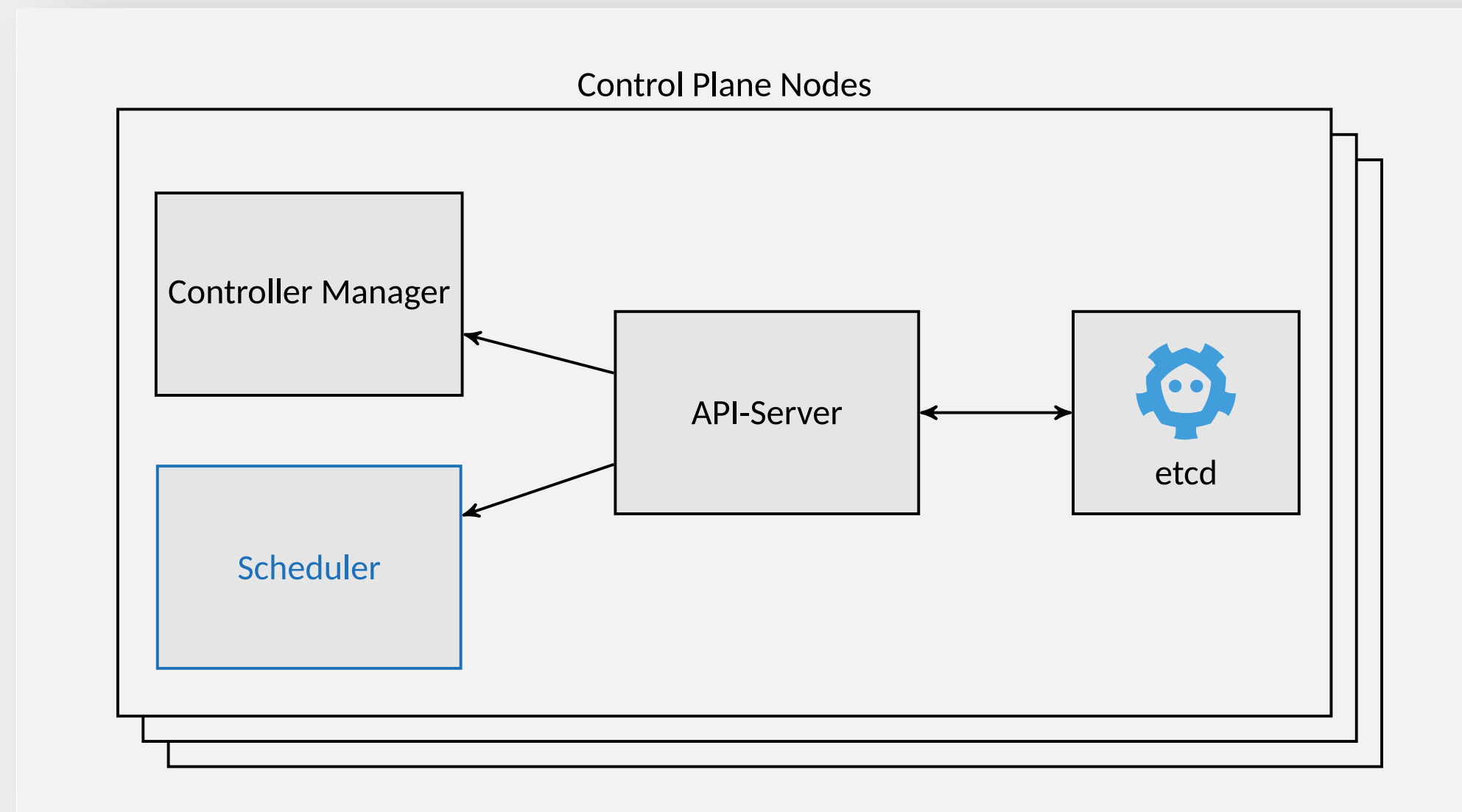
Control Plane - etcd



etcd

- verteilter Key-Value-Speicher
- Kubernetes speichert hier den Zustand des Clusters
- vollständig repliziert
- Ausfallsicher: mehr als die Hälfte der Instanzen muss verfügbar sein

Control Plane - Scheduler

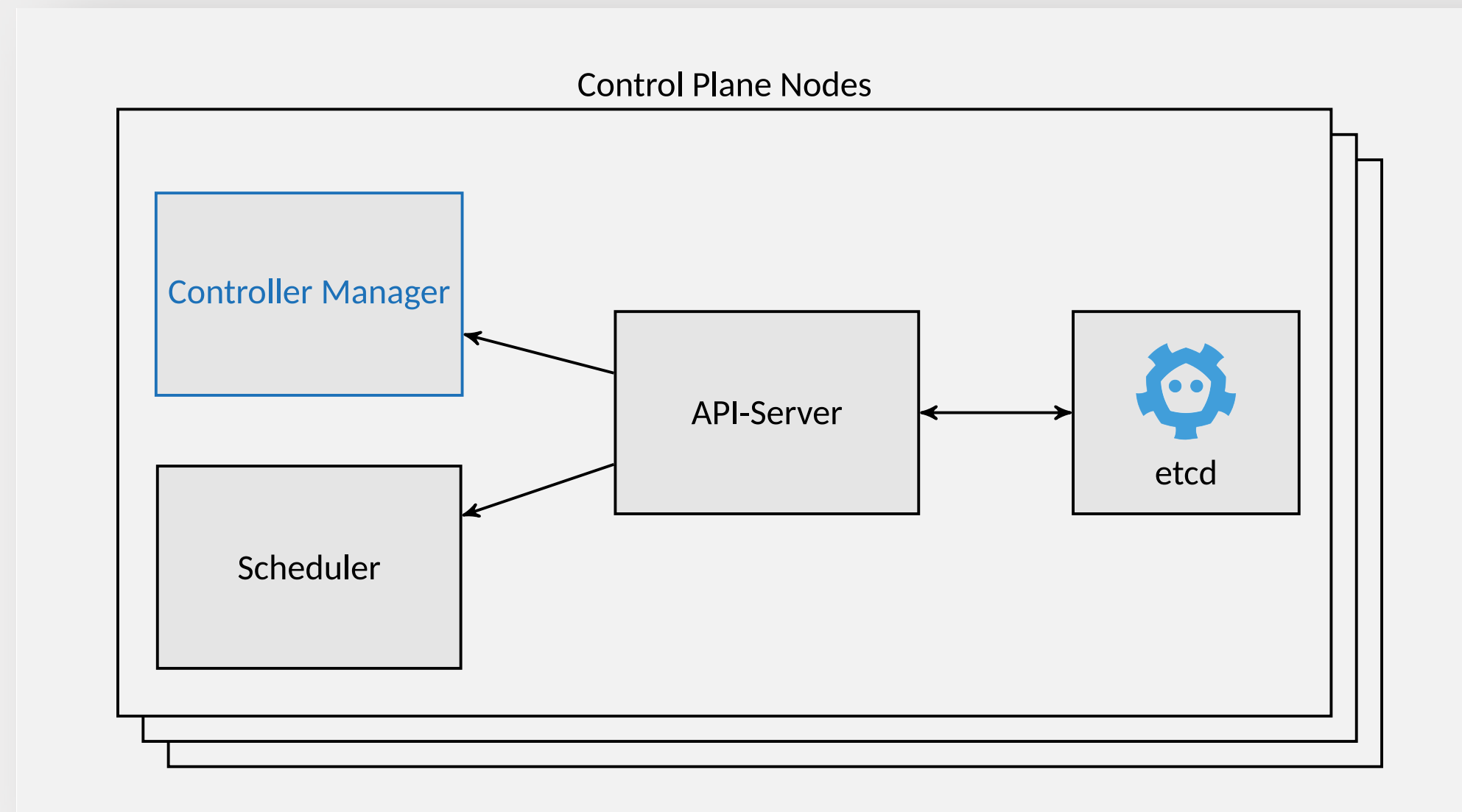


Scheduler

Verteilt Workloads / Pods auf die einzelnen Nodes anhand von:

- Ressourcenanforderungen
- Node-Selektoren
- Affinität und Anti-Affinität
- Taints und Tolerations
- ...

Control Plane - Controller Manager

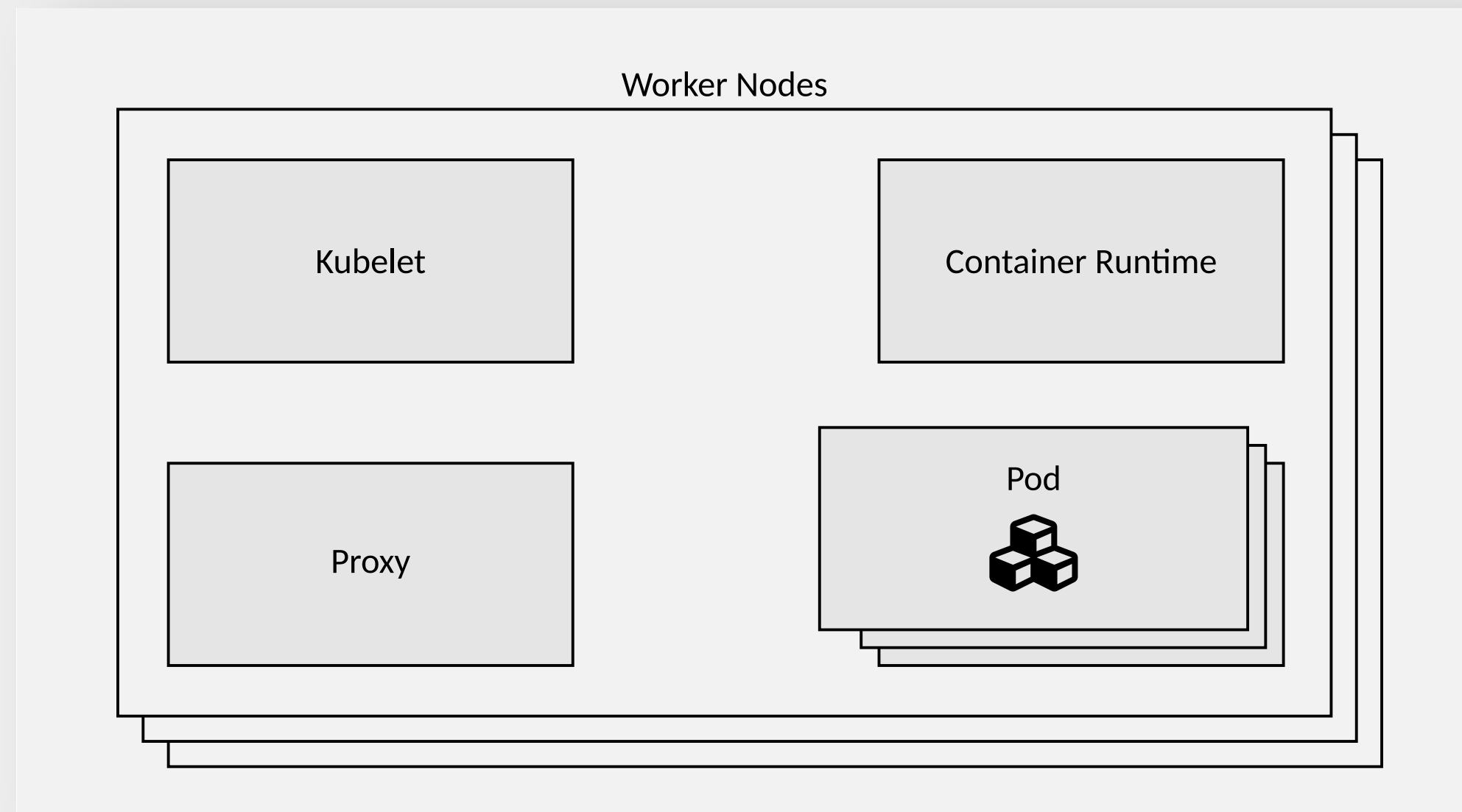


Controller Manager

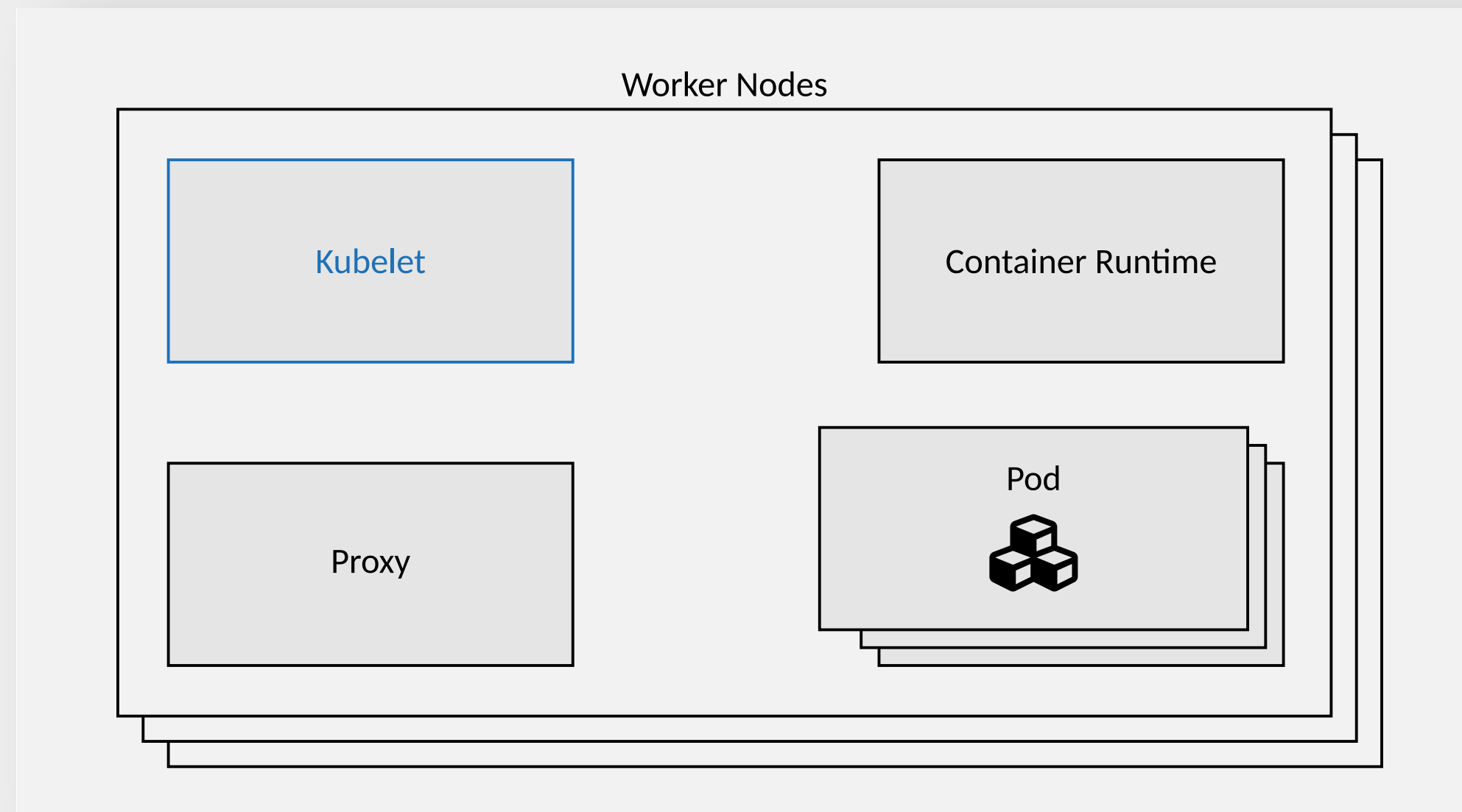
Daemon, der die wesentlichen Regelkreise von K8s verwaltet, z.B.
für

- Replicas
- Deployments
- Garbage Collection
- Daemon-Sets

Worker



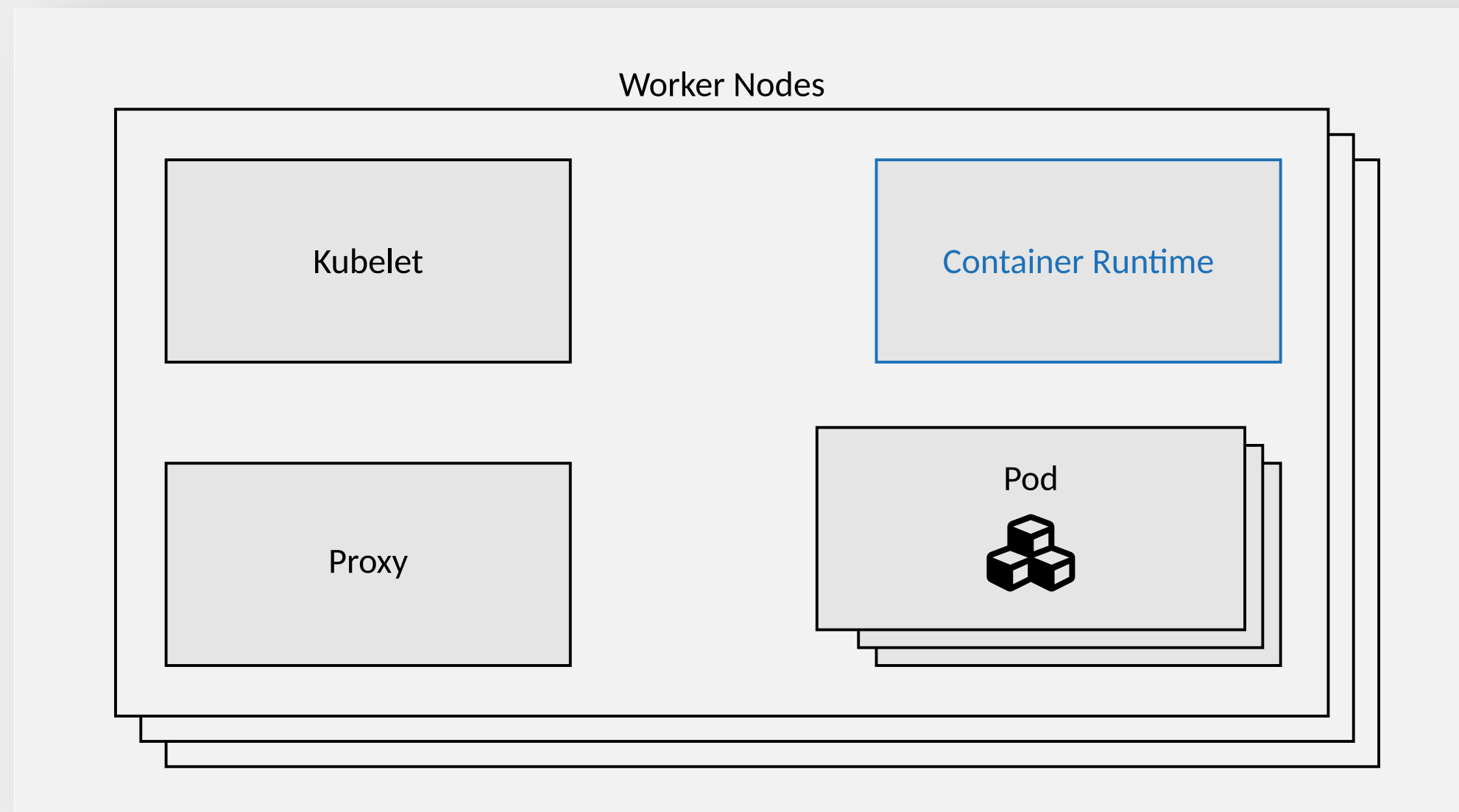
Worker - Kubelet



Kubelet

- läuft auf jedem Worker Knoten
- empfängt eine Vielzahl von Pod-Beschreibungen (Hauptsächlich vom API-Server)
- stellt sicher, dass die Pods auf dem Knoten gemäß den PodSpecs laufen -> nutzt dafür die Container Runtime

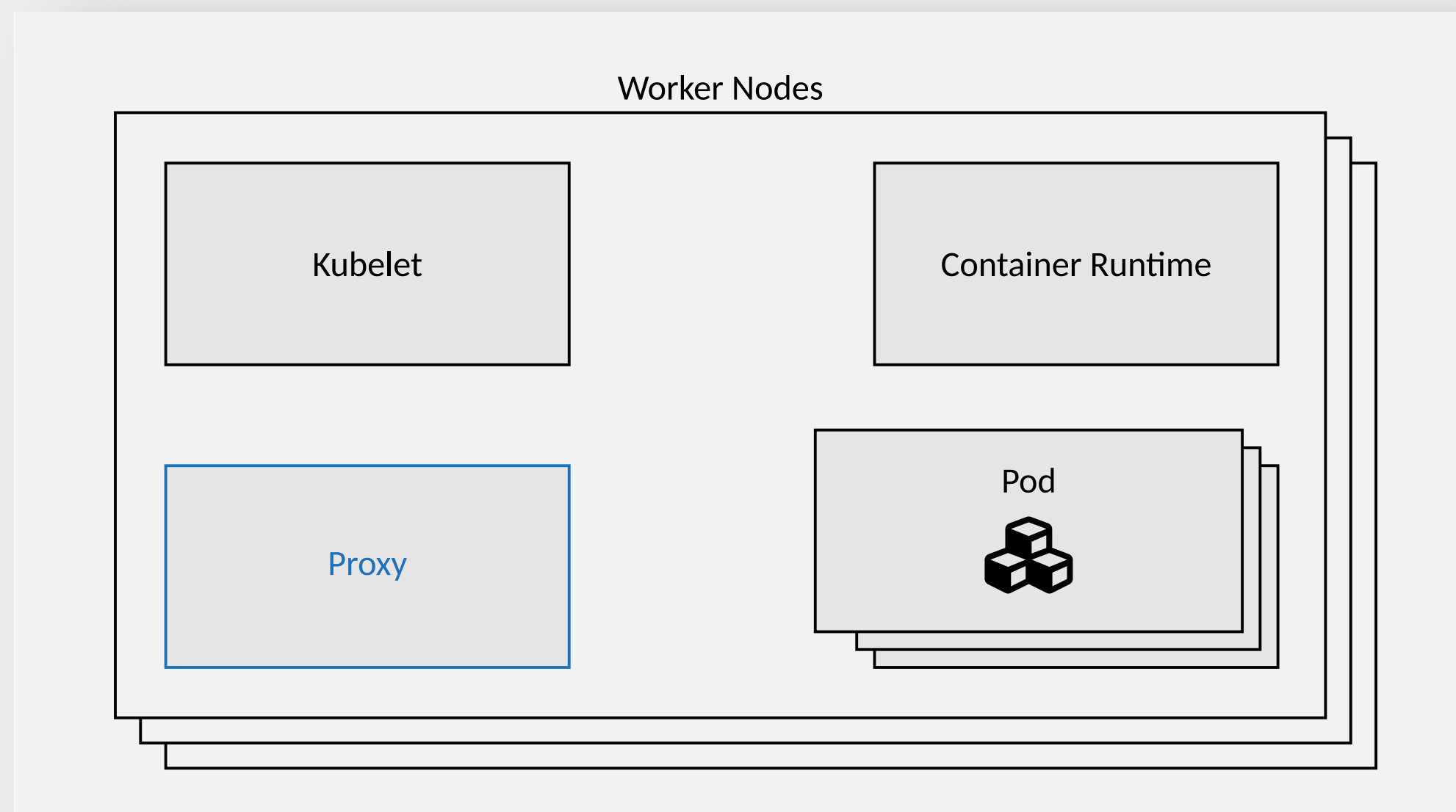
Worker - Container Runtime



Container Runtime

Wird von Kubelet zur Ausführung von Containern verwendet

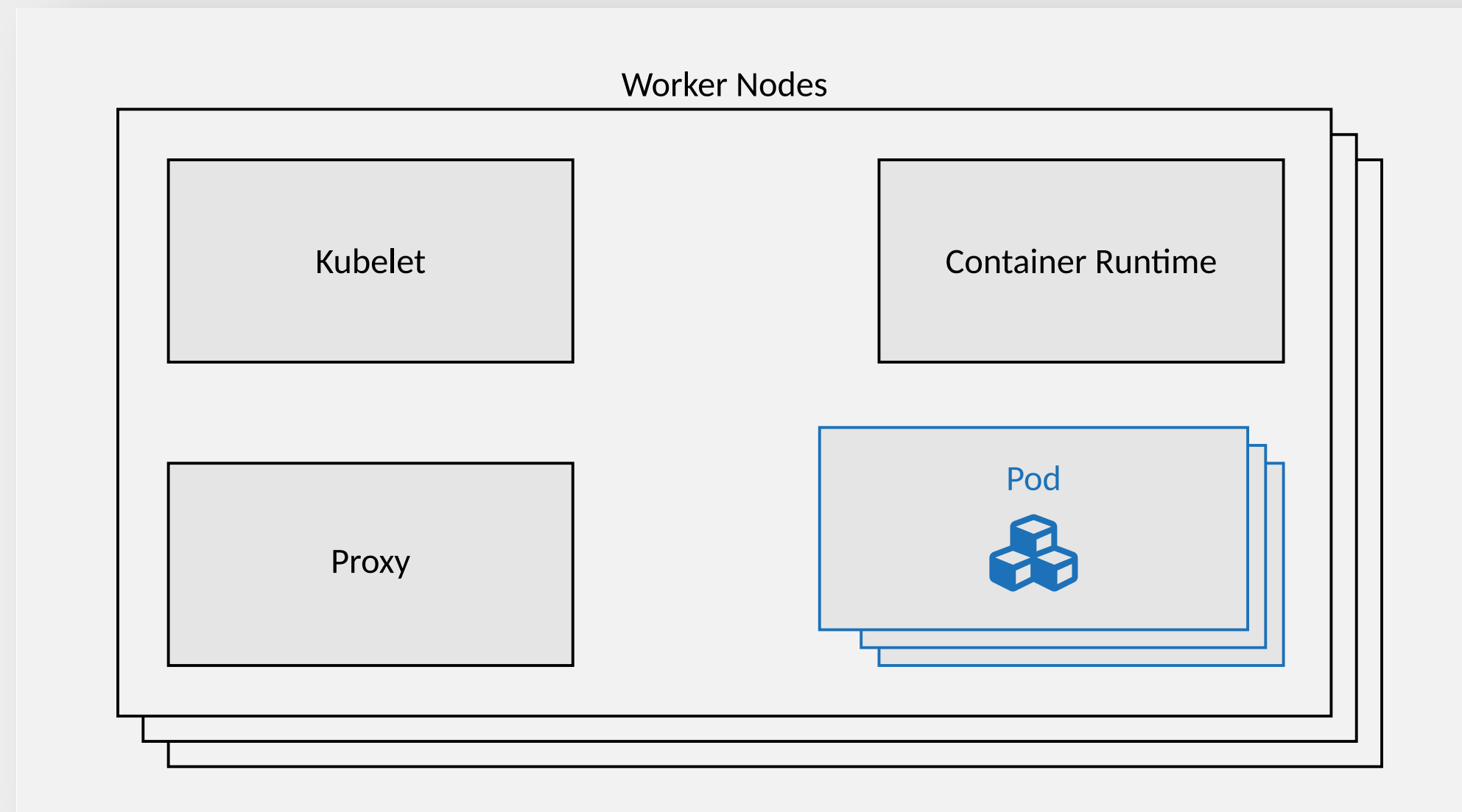
Worker - Proxy



Proxy

- läuft auf jedem Knoten
- leitet den für den Service bestimmten Verkehr an diesen

Worker - Pods



Pods

Lassen die Anwendung containerisiert laufen

Networking (CNI)

Ziel 1: Jeder Pod kann jeden anderen Pod ohne NAT erreichen Ziel

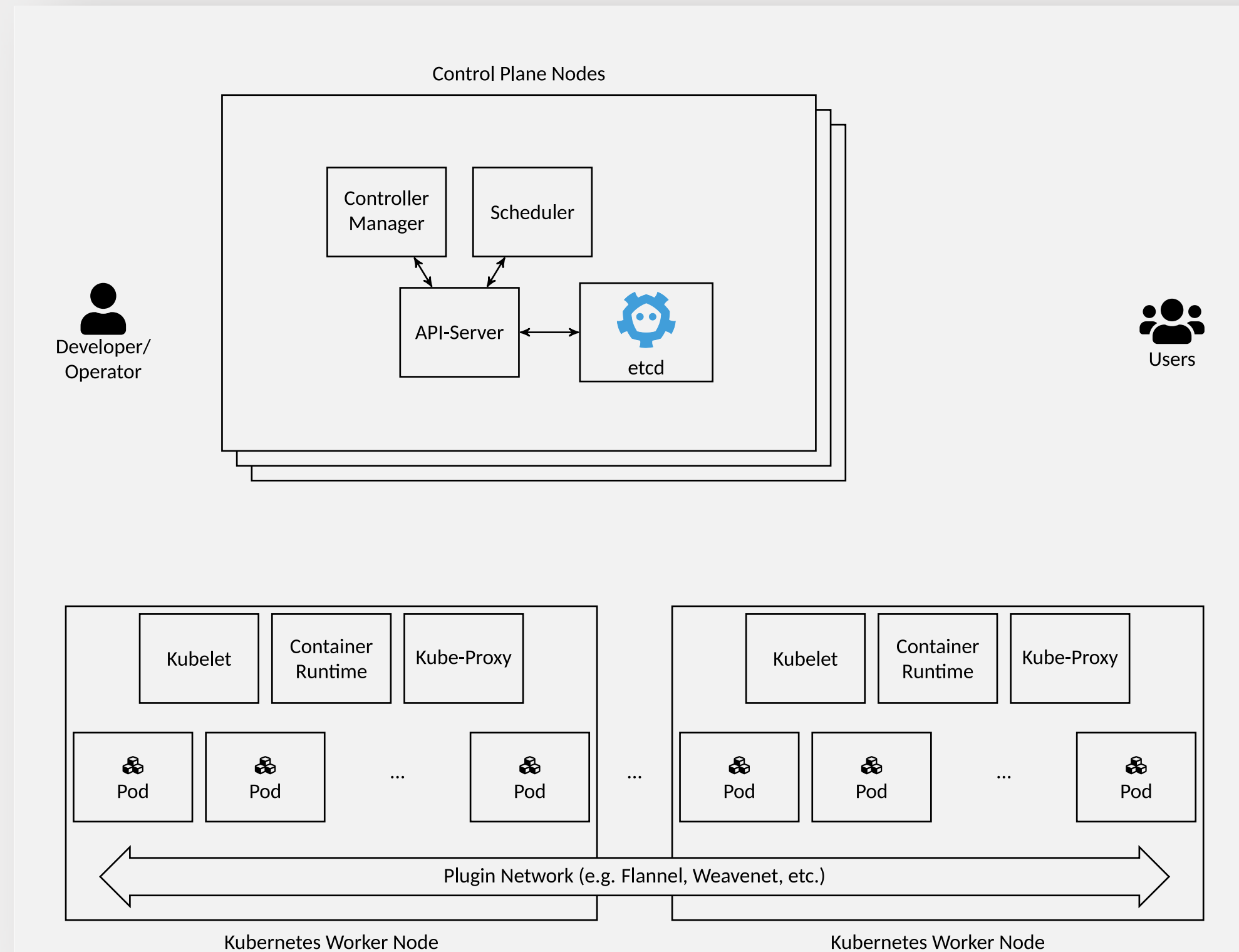
2: Alle Services auf einem Knoten können mit allen Pods auf dem Knoten kommunizieren

Lösung: Container Networking Interface (CNI) auf jedem Knoten

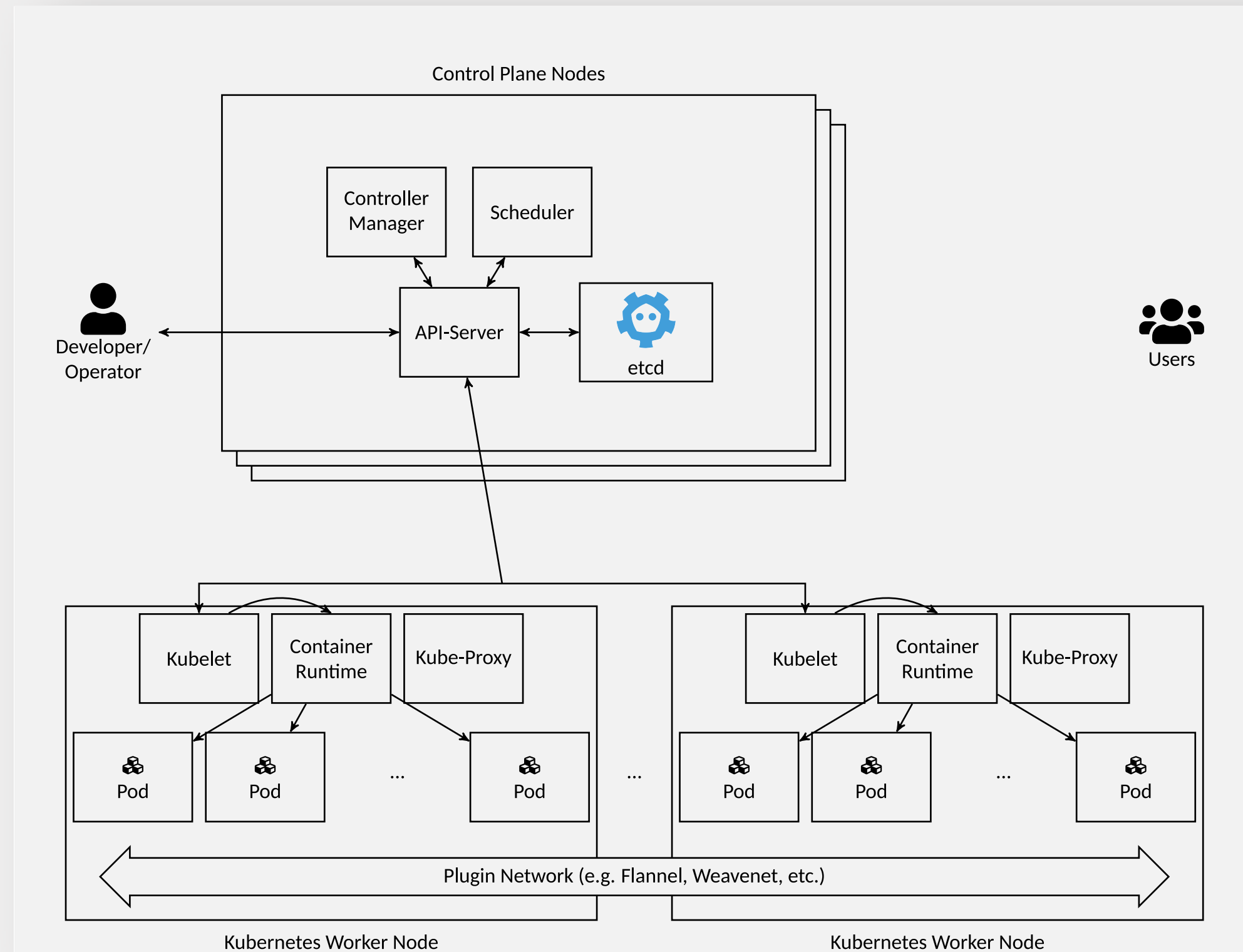
Add-On: DNS

- optional
- läuft nicht direkt auf dem Knoten sondern in der container engine
- Services per `servicename.namespace-name.svc.cluster.local` statt IP ansprechbar

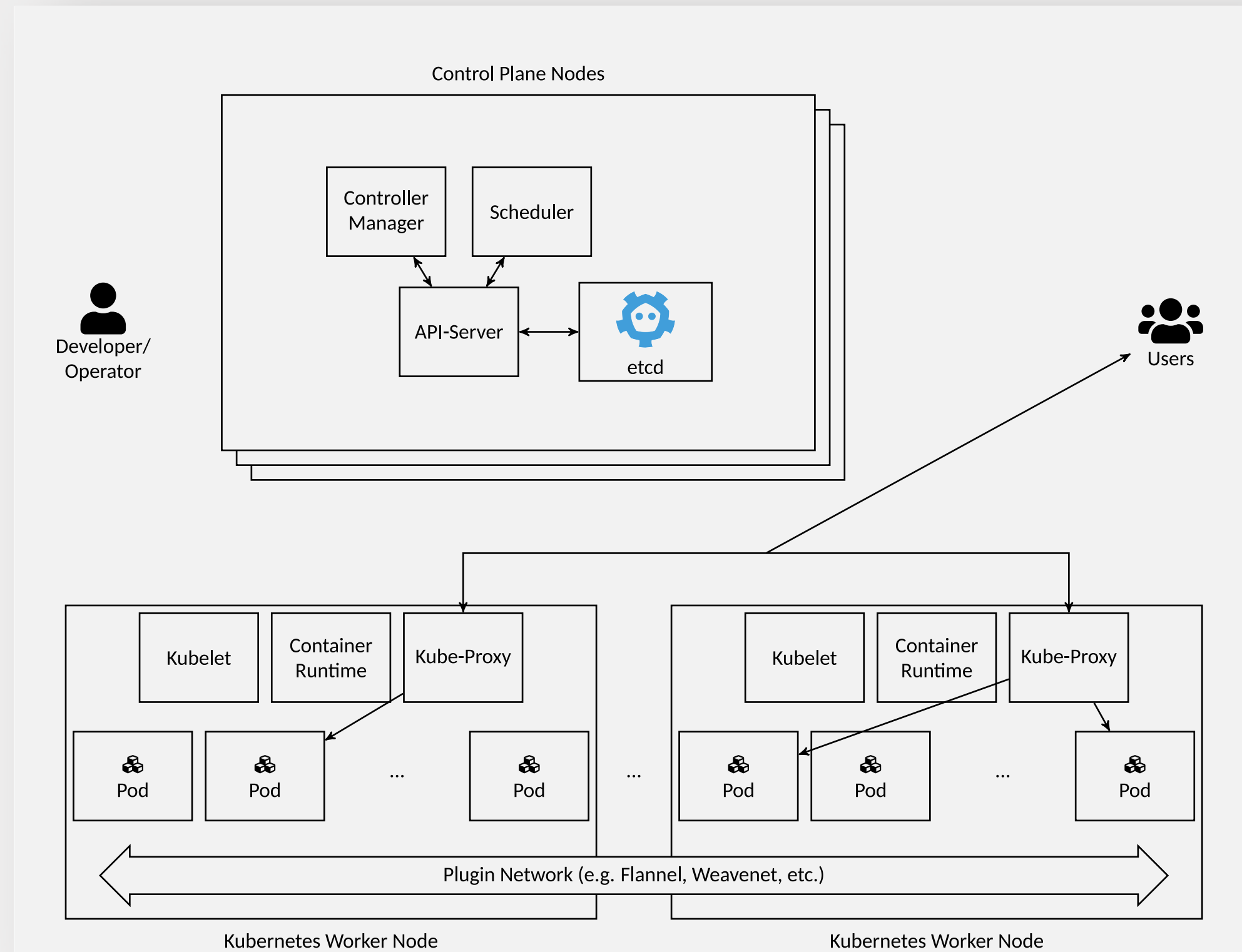
Zusammenspiel der Komponenten



Zusammenspiel der Komponenten



Zusammenspiel der Komponenten



Systematische Troubleshooting- Strategie

Troubleshooting-Methodology

1. Symptom identifizieren

- Was funktioniert nicht?
- Wann trat das Problem auf?
- Wurde etwas kürzlich geändert?

Troubleshooting-Methodology

2. Datensammlung

- Logs sammeln (journalctl, kubectl logs)
- Status prüfen (kubectl get, describe)
- Events analysieren

Troubleshooting-Methodology

3. Hypothese aufstellen

- Mögliche Ursachen eingrenzen
- Von außen nach innen arbeiten

Troubleshooting-Methodology

4. Testen und Verifizieren

- Eine Änderung nach der anderen
- Ergebnis dokumentieren

1. Symptome

Symptom

Node ist `NotReady`

Meist: Kubelet oder CNI-Problem

Symptom

Neue Pods bleiben in `Pending`

Meist: Scheduler oder Ressourcen-Problem

Symptom

`kubectl` Befehle schlagen fehl

Meist: Api-Server oder Etcd Problem

Symptom

Services sind nicht oder nur unzuverlässig erreichbar

Meist: kube-proxy, CNI oder DNS-Problem

Symptom

Pods starten nicht oder bleiben hängen

Meist: Kubelet oder Ressourcen-Problem

Symptom

Pod Status wird nicht aktualisiert

Meist: Kubelet Problem

Symptom

Deployments/ReplicaSets gleichen den Sollzustand nicht mehr an

Meist: Controller Manager Problem

Symptom

`kubectl logs`, `exec` oder `port-forward` auf einer Node
funktionieren nicht mehr

Meist: Kubelet Problem

Symptom

Jobs/CronJobs laufen nicht sauber

Meist: Controller Manager Problem

Symptom

Namensauflösung in Pods funktioniert nicht

Meist: CoreDNS oder CNI-Problem

Symptom

Cluster-Zustand wirkt eingefroren oder inkonsistent

Meist: Etcd Problem

Symptom

Ressourcen hängen in `Terminating`

Meist: Controller Manager (Garbage Collector) Problem

Symptom

Container beenden sich sofort oder lassen sich nicht ausführen

Meist: Container-Runtime Problem

2. Datensammlung

Log-Werkzeuge Übersicht

System-Level:

- `journalctl` - Systemd-Logs
- `systemctl` - Service-Status
- `ps`, `top`, `htop` - Prozess-Monitoring
- `netstat`, `ss` - Netzwerk-Verbindungen

Kubernetes-Level:

- `kubectl` - Cluster-Interaktion
- `crictl` - Container-Runtime Interface
- `etcdctl` - etcd-Cluster-Management

journalctl - Systemdienste analysieren

Wichtigste journalctl-Befehle für Kubernetes:

```
# kubelet Logs anzeigen
journalctl -u kubelet

# Letzte 100 Zeilen mit Follow-Mode
journalctl -u kubelet -n 100 -f

# Logs seit bestimmter Zeit
journalctl -u kubelet --since "2024-04-23 10:00:00"

# Nur Fehler und kritische Nachrichten
journalctl -u kubelet -p err

# Container Runtime Logs (containerd)
journalctl -u containerd
```

kubectl - Cluster-Logs analysieren

Pod- und Container-Logs:

```
# Pod Logs anzeigen
kubectl logs pod-name

# Container-spezifische Logs (Multi-Container Pod)
kubectl logs pod-name -c container-name

# Logs mit Follow-Mode
kubectl logs pod-name -f

# Vorherige Container-Instanz
kubectl logs pod-name --previous

# Logs aller Pods eines Deployments
kubectl logs deployment/my-app
```

kubectl - System-Komponenten Logs

Control Plane Komponenten:

```
# API Server Logs
kubectl logs -n kube-system kube-apiserver-control-plane

# Controller Manager Logs
kubectl logs -n kube-system kube-controller-manager-control-plane
...
```

kubectl - Events und Describe

Cluster-Events anzeigen:

```
# Alle Events chronologisch
kubectl get events --sort-by=.metadata.creationTimestamp

# Events für spezifisches Objekt
kubectl describe pod pod-name
kubectl describe node node-name

# Events in allen Namespaces
kubectl get events --all-namespaces

# Events filtern nach Typ
kubectl get events --field-selector type=Warning
```

Node-Level Diagnose

Node-Ressourcen analysieren

```
# Node-Ressourcenverbrauch
kubectl top nodes

# Detaillierte Ressourcen-Aufteilung
kubectl describe node node-name | grep -A 10 "Allocated resources"

# System-Ressourcen auf Node (SSH erforderlich)
free -h          # Memory
df -h            # Disk Space
lscpu            # CPU Info
uptime           # Load Average
```

kubelet und Systemdienste Diagnose

kubelet-Status prüfen

```
# kubelet Service-Status  
systemctl status kubelet  
  
# kubelet-Konfiguration anzeigen  
cat /var/lib/kubelet/config.yaml  
  
# kubelet-Version  
kubelet --version
```

kubelet-Logs analysieren

```
# Aktuelle kubelet-Logs  
journalctl -u kubelet -f  
  
# kubelet-Fehler suchen  
journalctl -u kubelet -p err
```

Häufige kubelet-Probleme:

- CNI-Plugin nicht verfügbar
- Container-Runtime nicht erreichbar
- Zertifikat-Probleme
- Node-Registrierung fehlgeschlagen

Container-Runtime Diagnose

containerd diagnostizieren:

```
# containerd Status  
systemctl status containerd
```

```
# containerd-Logs  
journalctl -u containerd
```

```
# Container Runtime Info  
crictl info
```

```
# Laufende Container  
crictl ps
```

```
# Container-Images  
crictl images
```

```
# Container-Logs  
crictl logs container-id
```

Control Plane Komponenten

Diagnose

Static Pods Status prüfen

```
# Static Pods anzeigen
kubectl get pods -n kube-system -o wide

# Static Pod Manifests prüfen
ls -la /etc/kubernetes/manifests/

# Static Pod Events
kubectl describe pod kube-apiserver-control-plane -n kube-system

# Container-Status direkt prüfen
crictl ps | grep -E "(etcd|apiserver|controller|scheduler)"
```

API-Server Diagnose

```
# API-Server Health Check
curl -k https://localhost:6443/healthz

# API-Server Listening Ports
netstat -tlnp | grep 6443

# API-Server Manifest prüfen
cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

Häufige API-Server-Probleme:

- Konfigurationsfehler
- Zertifikat-Fehler
- etcd-Verbindungsprobleme

etcd Diagnose

```
# etcd-Health-Check
ETCDCTL_API=3 etcdctl endpoint health \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key

# etcd-Member-Status
ETCDCTL_API=3 etcdctl member list \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key
```

Netzwerk- und CNI-Probleme

DNS-Probleme diagnostizieren

```
# CoreDNS Status
kubectl get pods -n kube-system -l k8s-app=kube-dns

# CoreDNS Logs
kubectl logs -n kube-system -l k8s-app=kube-dns

# DNS-Resolution testen
kubectl run debug-pod --image=busybox -it --rm --restart=Never -- sh
nslookup kubernetes.default
nslookup mein-service

# DNS-Service prüfen
kubectl get svc -n kube-system kube-dns

# CoreDNS-Konfiguration
kubectl get configmap coredns -n kube-system -o yaml
```

Pod-zu-Pod Konnektivität

```
# Test-Pods erstellen
kubectl run test-pod-1 --image=busybox --restart=Never -- sleep 3600
kubectl run test-pod-2 --image=busybox --restart=Never -- sleep 3600

# Pod-IPs ermitteln
kubectl get pods -o wide

# Ping-Test zwischen Pods
kubectl exec test-pod-1 -- ping <test-pod-2-ip>
```

Service und Ingress Diagnose

```
# Service-Endpoints prüfen
kubectl get endpoints service-name

# kube-proxy Logs
kubectl logs -n kube-system -l component=kube-proxy

# Service-Connectivity aus Pod testen
kubectl exec test-pod -- wget -O- service-name:port
```

Zertifikats- und API- Zugriffsprobleme

Zertifikat-Validierung

```
kubeadm certs check-expiration
```

kubeconfig-Probleme

```
# kubeconfig-Syntax prüfen
kubectl config view

# Aktueller Context
kubectl config current-context

# Cluster-Connectivity testen
kubectl cluster-info

# Manuelle API-Verbindung testen
curl -k --cert /etc/kubernetes/pki/apiserver-kubelet-client.crt \
      --key /etc/kubernetes/pki/apiserver-kubelet-client.key \
      https://localhost:6443/api/v1/nodes
```

Zur Kapitelübersicht

- Vorheriges Kapitel: [Kubernetes Developer Troubleshooting](#)