Kubernetes Internals

etcd • Control Plane • Kubelet • kube-proxy • Admission • Scheduler • API Machinery

Audience: Advanced / Intermediate (beginner-accessible)
Focus: Theory-first, component responsibilities, data/control flows

> Ziel: Tiefenverständnis der zentralen Kubernetes-Komponenten mit Fokus auf Datenflüsse, Verantwortlichkeiten und Interaktionen. Keine Hands-on-Demos; alle Beispiele sind konzeptionell.

Agenda

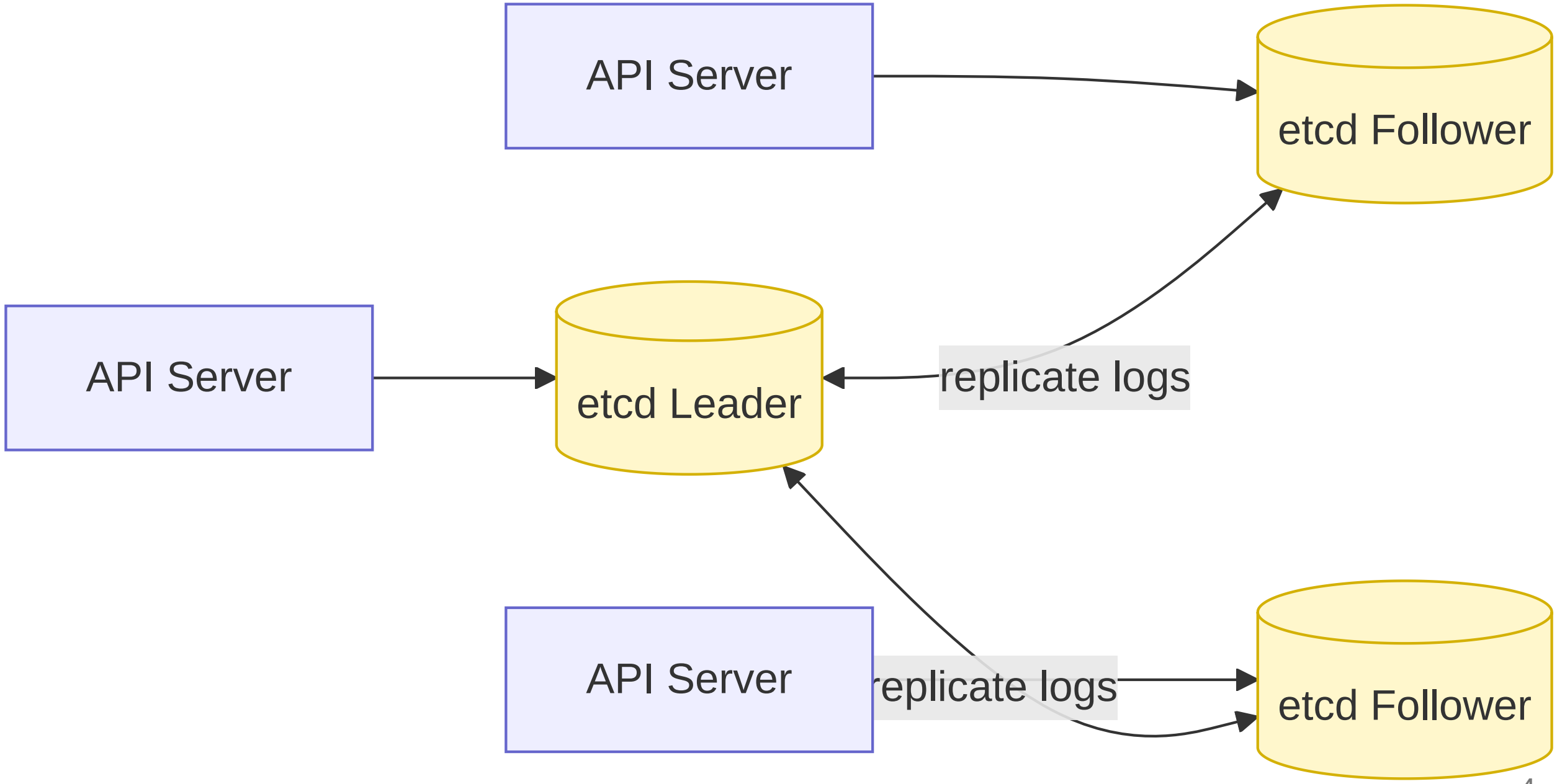
- etcd (key–value store)
- Control Plane Components (API Server, Controller Manager, Scheduler)
- Kubelet (node agent)
- kube-proxy & networking modes (iptables, IPVS)
- Admission Controllers
- Kubernetes Scheduler deep dive
- Kubernetes API Machinery

> Agenda-Überblick: Wir folgen dem Weg einer Anfrage von der API über Admission, Controller/Scheduler bis zum Node und zurück in etcd.

1 etcd — The Source of Truth

Role, Consistency & Topology

- **Role:** Strongly consistent, distributed key–value store that persists cluster desired & observed state.
- **Consistency:** Uses Raft for leader election and log replication; reads/writes via leader for linearizability.
- **Data model:** Hierarchical keys (e.g., /registry/pods/ns/name), versioned with revisions, watch streams for changes.
- **Topology:** Odd number of members (3/5); spread across failure domains.


API Server
etcd Follower
API Server
etcd Leader
replicate logs
API Server
replicate logs
etcd Follower

Ops essentials: Quorum, low latency, snapshots/defrag, encryption at rest (ETCD/Volume), regular backups & restore drills.

> etcd ist das Herz. Sicherstellen: Quorum (immer ungerade Anzahl), schnelle/zuverlässige Speicherschicht, regelmäßige Backups und Defragmentierung.

etcd Performance & Safety

- **Latency sensitive:** Co-locate API servers close to etcd; avoid slow disks & high network RTT.
- **Compaction/defrag:** Reduce DB bloat from watches; plan maintenance windows.
- **Watches:** Controllers/Scheduler rely on watch streams; use resourceVersion for efficient list/watch.

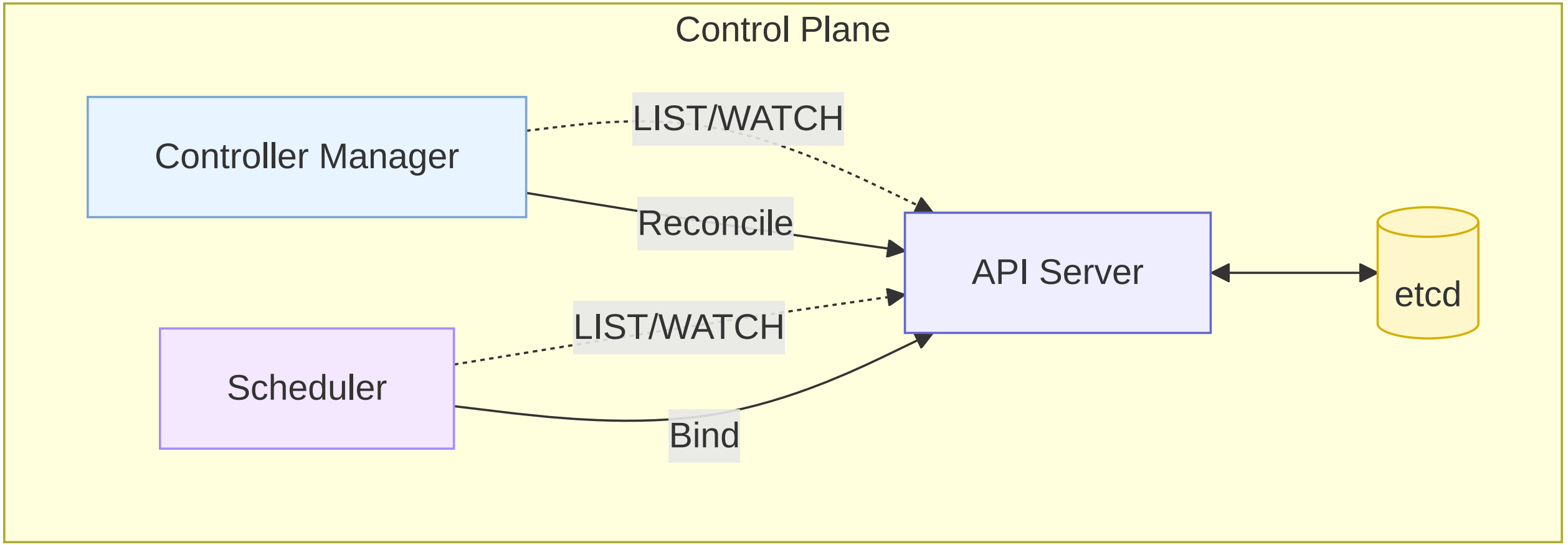
> Performance-Hebel: Nähe, schnelle SSD/NVMe, kompakte Datenbank. Watches sind der Motor für reaktive Controller.

2 Control Plane Components

API Server • Controller Manager • Scheduler

- **API Server:** Front door; authn/z, admission, validation, persistence to etcd, exposes watch APIs.
- **Controller Manager:** Built-in controllers (Deployment, RS, Job, Node, etc.) => level-based reconciliation.
- **Scheduler:** Assigns unscheduled Pods to Nodes based on constraints & scoring; writes Binding objects.

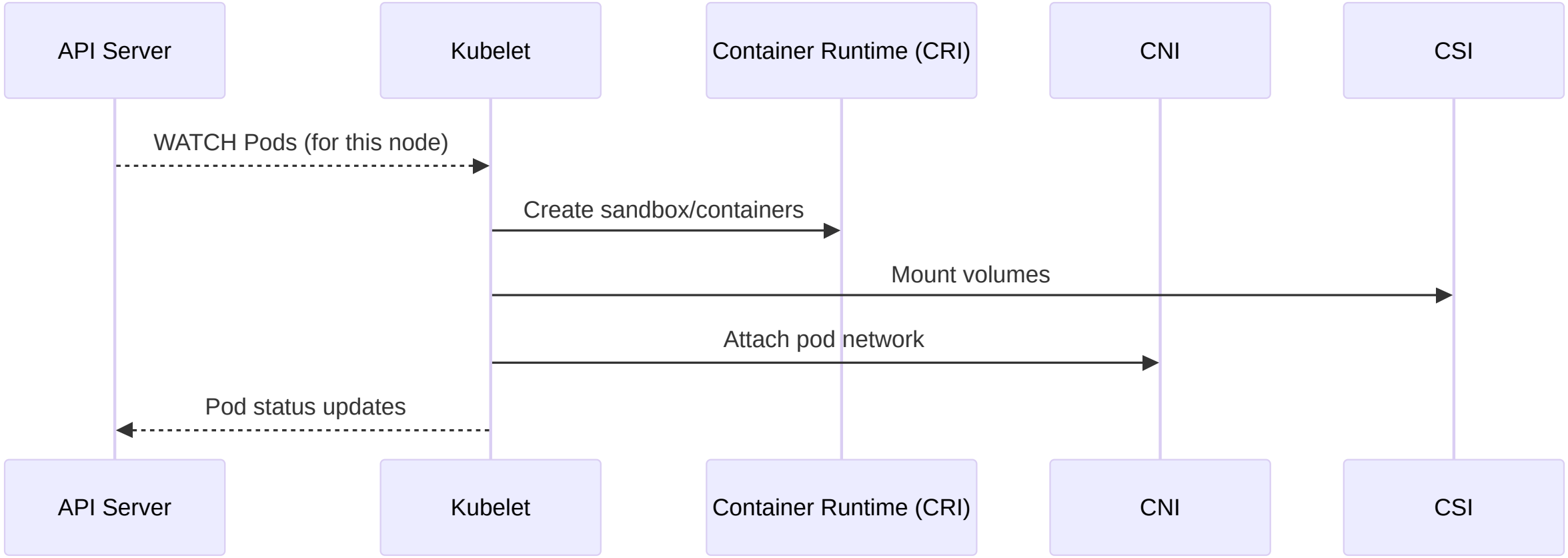
> API-Server ist die Schnittstelle und Gatekeeper. Controller sind zustandsbasiert (keine imperativen Skripte). Scheduler trifft Bindungsentscheidungen, nicht das Kubelet.


Control Plane
Controller Manager
LIST/WATCH
Reconcile
API Server
etcd
LIST/WATCH
Scheduler
Bind

3 Kubelet — The Node Agent

Responsibilities & Work Loop

- **Desired state consumer:** Watches PodSpecs assigned to its node (via API).
- **Execution:** Uses CRI runtime (containerd/CRI-O) to create sandboxes/containers; sets up volumes via CSI and networking via CNI.
- **Health:** Reports Pod/Node status, executes probes; enforces cgroups & eviction policies.
- **Auth to API:** Uses client certs; may use bootstrap/rotating certs.


API Server
Kubelet
Container Runtime (CRI)
CNI
CSI
WATCH Pods (for this node)
Create sandbox/containers
Mount volumes
Attach pod network
Pod status updates
API Server
Kubelet
Container Runtime (CRI)
CNI
CSI

Node lifecycle: Readiness/taints/cordon&drain; kubelet honors PDBs via eviction API.

> Kubelet setzt zugewiesene Pods um: Container, Netzwerk, Storage. Gleichzeitig sendet es Status zurück und respektiert Evictions/Drains.

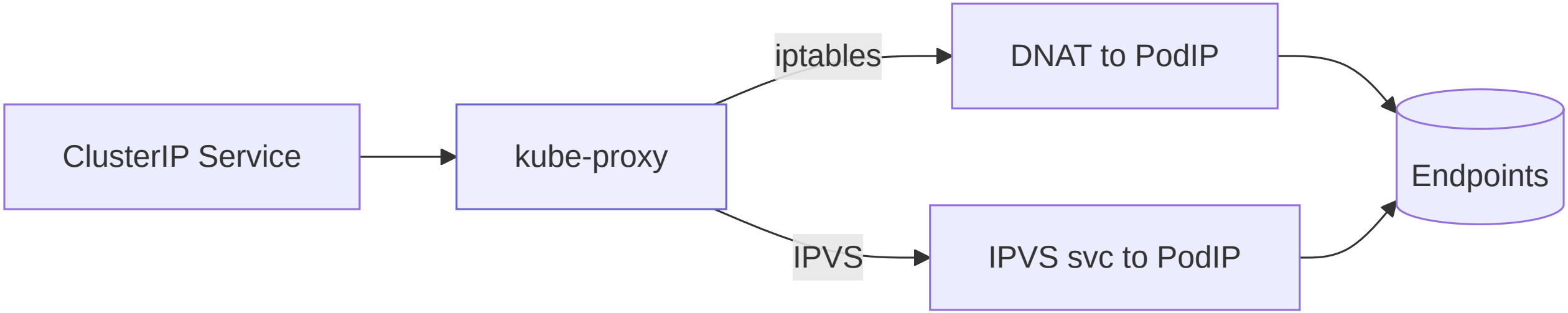
4 kube-proxy & Networking Modes

Service Virtual IPs & Data Paths

- **Purpose:** Implements Service abstraction (ClusterIP, NodePort, LoadBalancer) by programming node dataplane.

Modes:

- **iptables:** Rule chains for service → endpoint DNAT; simple, universal; scale limits with many rules.
- **IPVS:** Kernel IP Virtual Server; hash-based load balancing, faster updates & better scale.


ClusterIP Service
kube-proxy
iptables
IPVS
DNAT to PodIP
IPVS svc to PodIP
Endpoints

Alternatives: CNI project dataplanes (eBPF in Cilium) can bypass kube-proxy and handle services natively.

Health: EndpointSlice objects reduce churn; readiness gates endpoint inclusion.

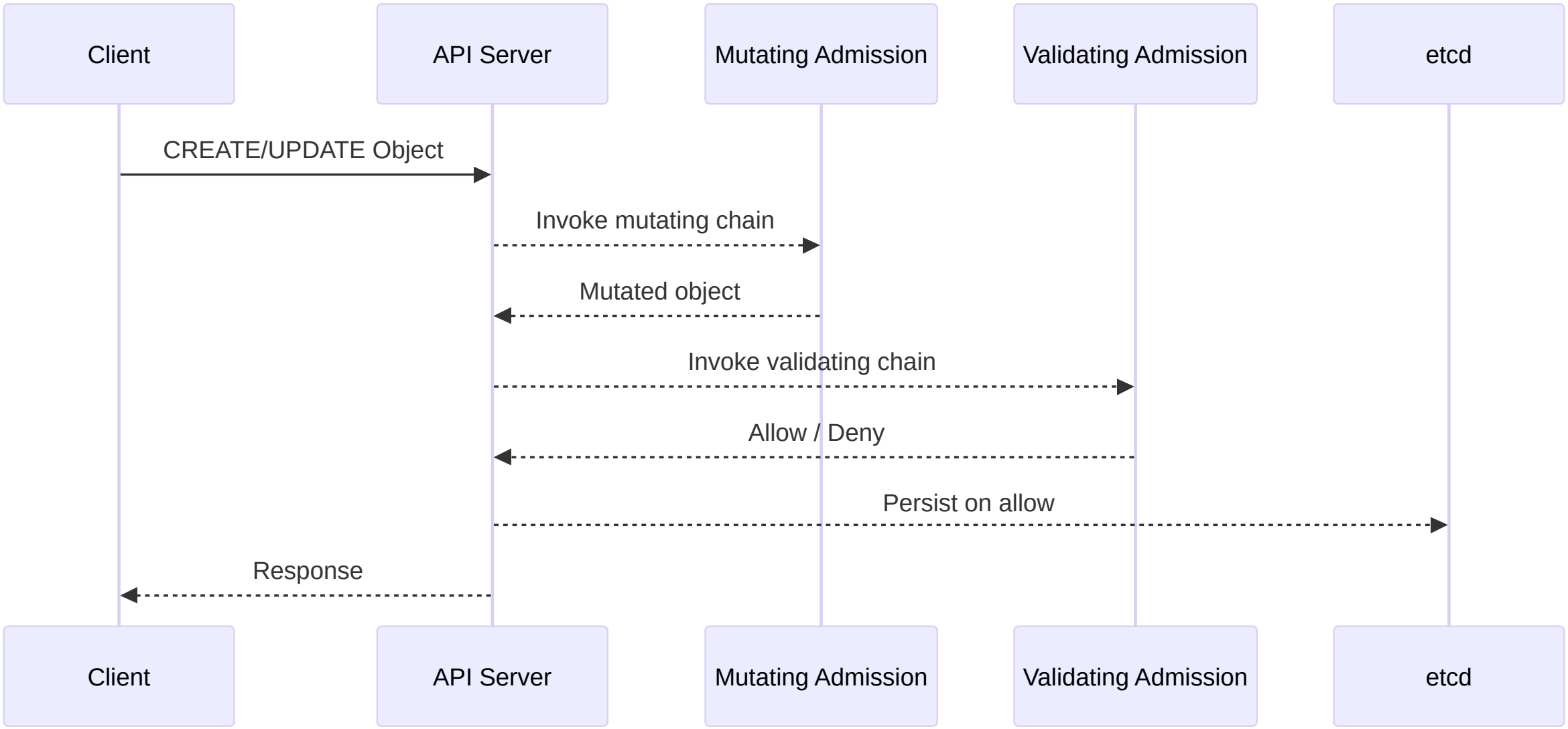
> iptables ist verbreitet, IPVS skaliert besser. Moderne CNIs (z. B. eBPF) übernehmen oft die Service-Weiterleitung ohne kube-proxy.

5) Admission Controllers

The Request Pipeline

Phases:
Authentication → Authorization (RBAC/ABAC) → Admission (mutating → validating) → Persistence

- **Mutating webhooks/controllers:** Modify incoming objects (defaults, sidecars).
- **Validating webhooks/controllers:** Enforce policy (deny/allow with reason).
- **Built-ins & Webhooks:** Mix of compiled-in and external webhooks; ordered chains.


Client
API Server
Mutating Admission
Validating Admission
etcd
CREATE/UPDATE Object
Invoke mutating chain
Mutated object
Invoke validating chain
Allow / Deny
Persist on allow
Response
Client
API Server
Mutating Admission
Validating Admission
etcd

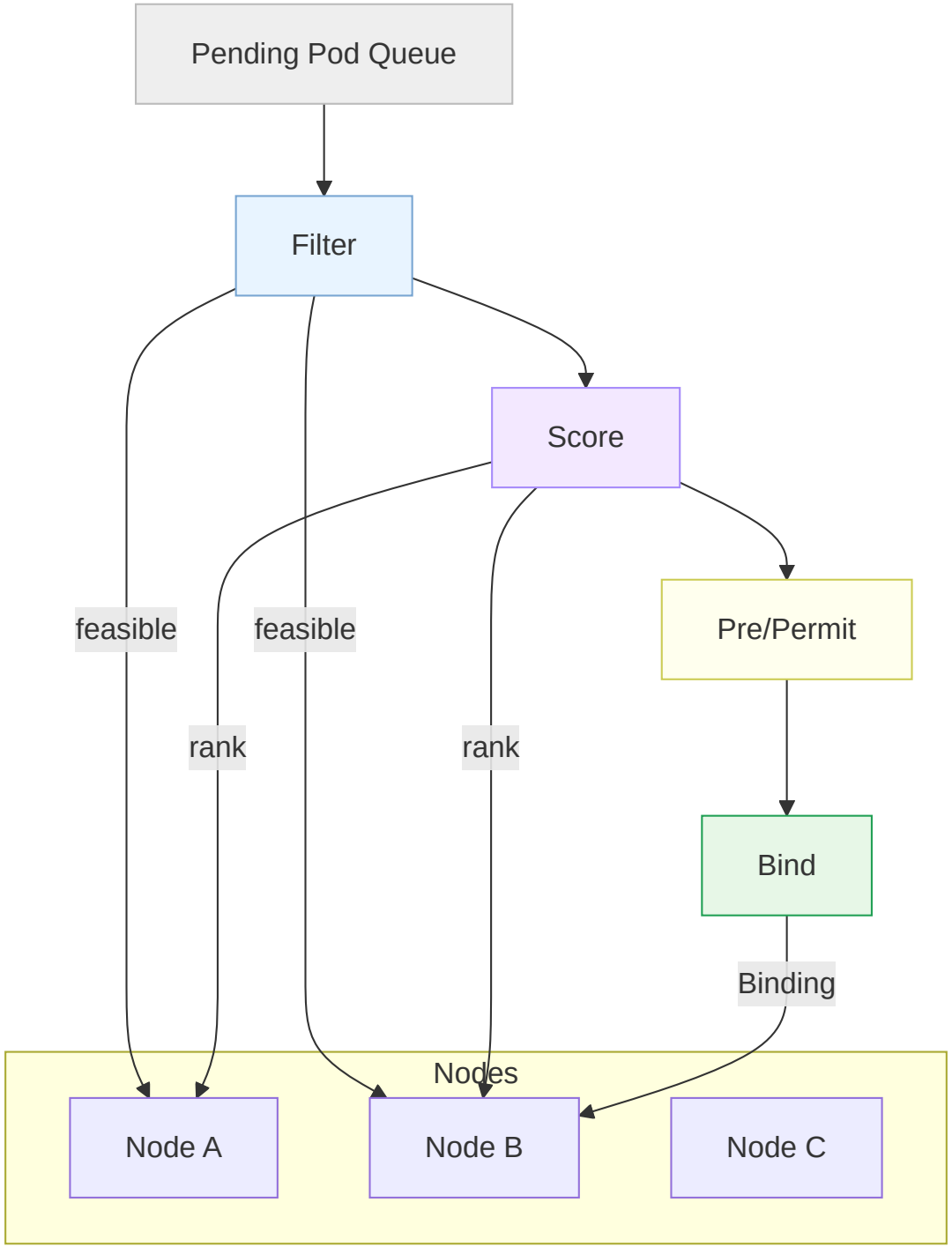
Design: Keep webhook latency/availability high; fail-open vs. fail-closed decisions matter for reliability vs. security.

> Admissions sind das Policy-Tor. Mutating zuerst (Defaults/Sidecars), dann Validating (Policy). Verfügbarkeit/Timeouts kritisch—Fehlermodus bewusst wählen.

6 Kubernetes Scheduler — Deep Dive

Scheduling Cycle & Framework

- **Objects:** Pods without nodeName are candidates.
- **Cycle:** Queue → Filter (Predicates) → Score → Reserve/Permit → Bind.
- **Inputs:** Node allocatable, taints/tolerations, affinities/anti-affinities, topology spread, resources, extenders/plugins.


Pending Pod Queue
Filter
Score
Pre/Permit
feasible
feasible
rank
rank
Bind
Binding
Nodes
Node A
Node B
Node C

- **Filters (examples):** NodeUnschedulable, PodFitsResources, PodToleratesTaints, NodeAffinity, VolumeRestrictions.
- **Scores (examples):** LeastAllocated, BalancedAllocation, ImageLocality, TopologySpread scoring.
- **Plugins:** Scheduling Framework allows extension points (preFilter, filter, postFilter, score, reserve, permit, preBind, bind, postBind).

> Planungszyklus verstehen: erst filtern (machbar), dann bewerten (Ranking), schließlich binden. Erweiterbar über Plugins; wichtig für Spezialanforderungen.

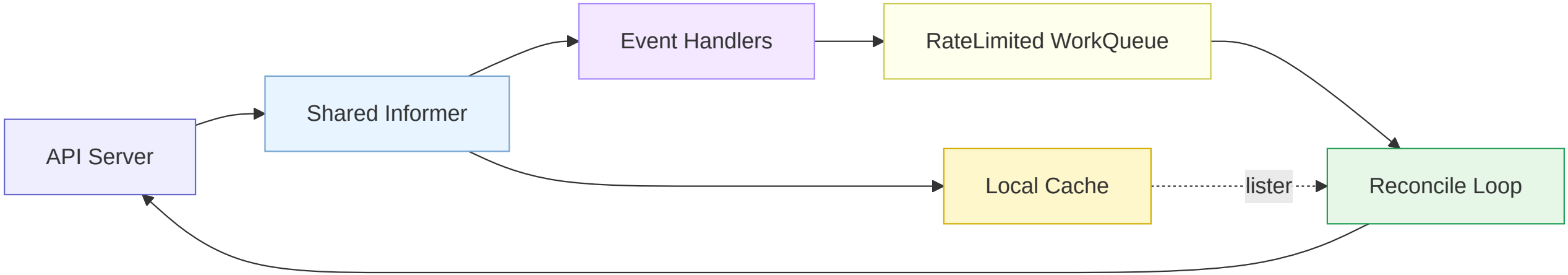
Preemption, Priorities & Fairness

- **PriorityClasses:** Higher priority pods may preempt lower ones to fit.
- **PDB-awareness:** Preemption respects PodDisruptionBudgets to preserve availability.
- **Queues:** Backoff & fairness to prevent starvation; multiple queues for different

7 Kubernetes API Machinery

Clients, Informers, Caches & Work Queues

- **client-go:** Typed clients for API groups; RESTMapper for discovery.
- **SharedInformers:** LIST+WATCH + local cache; deliver event handlers (add/update/delete) to controllers.
- **WorkQueue:** Rate-limited queue for reconciliation keys (usually namespace/name).
- **Controller Pattern:** Observe → Enqueue → Reconcile → Update Status; level-based & idempotent.


Event Handlers
RateLimited WorkQueue
Shared Informer
API Server
Local Cache
lister
Reconcile Loop

Guidelines:

- Reconcile on observed drift, not timers.
- Use finalizers for cleanup on delete.
- Status subresource for feedback; avoid writing spec during reconcile loops.

> API Machinery ermöglicht skalierbare Controller: Informer-Caches reduzieren Last, WorkQueues steuern Durchsatz, Finalizer sichern Aufräumlogik.

API Server: Storage & Versioning

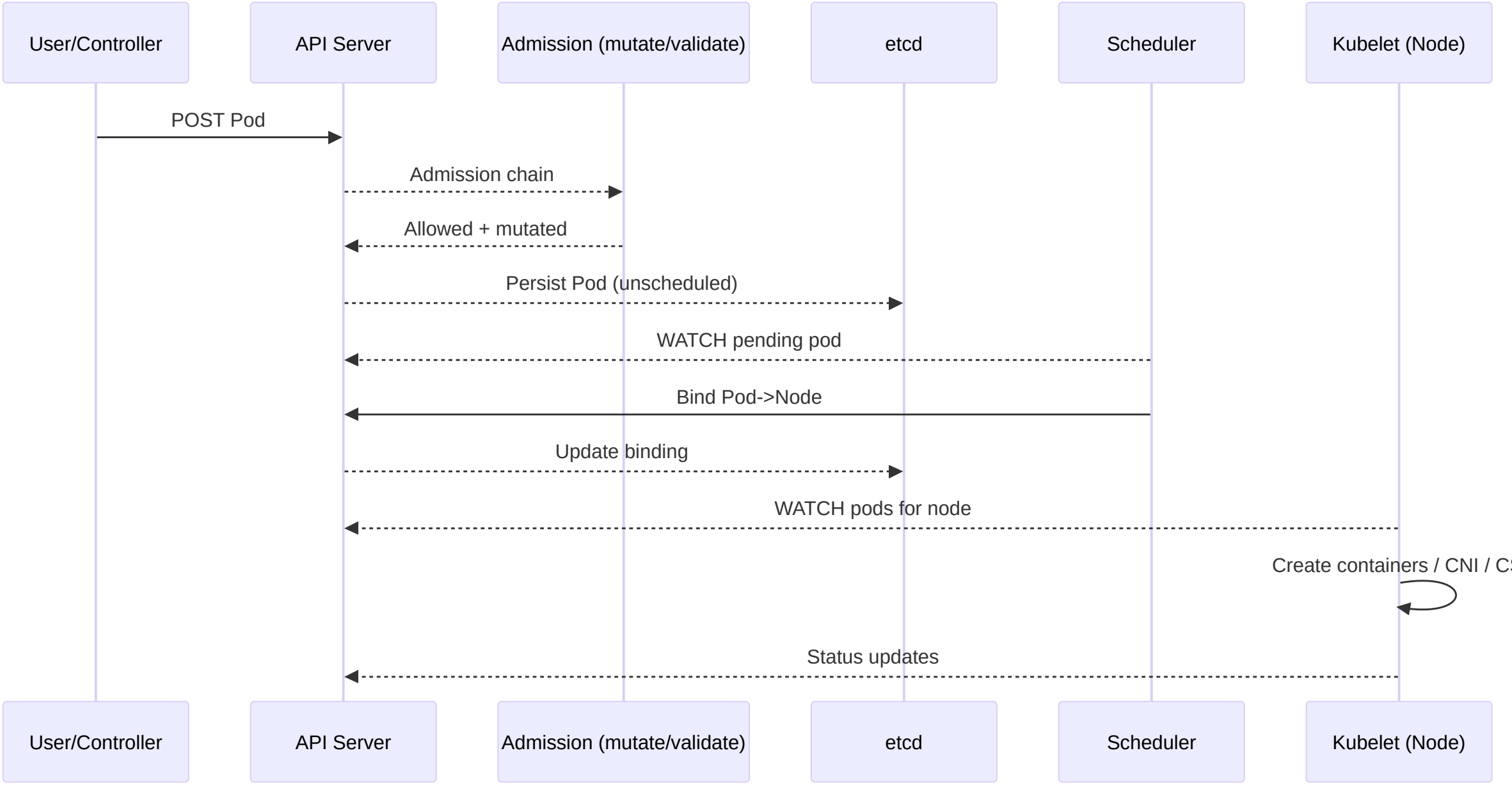
- **Storage:** Internal version in etcd; conversion webhooks for CRDs across versions.
- **Validation:** OpenAPI schemas; CEL/validations for CRDs; defaulting & pruning.
- **Discovery:** /apis and /api endpoints; dynamic clients use discovery to adapt.

> API-Versionierung & -Validierung sind Grundlage für Evolvierbarkeit. CRDs benötigen saubere Schemas und Version-Konvertierungen.

End-to-End Flow (Putting It Together)

Lifecycle: Desired state enters via API → policy checks → persisted → scheduled → realized by kubelet → observed via status.

> Gesamtablauf vom Erstellen bis zum Laufen: Admission sichert Policy, etcd speichert Zustand, Scheduler bindet, Kubelet setzt um, Status fließt zurück.


User/Controller
API Server
Admission (mutate/validate)
etcd
Scheduler
Kubelet (Node)
POST Pod
Admission chain
Allowed + mutated
Persist Pod (unscheduled)
WATCH pending pod
Bind Pod->Node
Update binding
WATCH pods for node
Create containers / CNI / CSI
Status updates
User/Controller
API Server
Admission (mutate/validate)
etcd
Scheduler
Kubelet (Node)

Key Takeaways

- etcd is the single source of truth; protect quorum, latency, and backups.
- API Server is the gatekeeper; Admission encodes platform policy.
- Controllers reconcile; Scheduler selects placement; Kubelet realizes workloads.
- kube-proxy / dataplanes implement Services; modern CNIs may replace proxy logic.
- API Machinery (informers, caches, queues) enables scalable, idempotent controllers.

> Merksätze: Datenkonsistenz (etcd), Governance (Admission), Zustandsabgleich (Controller/Scheduler/Kubelet), Netzpfad (kube-proxy/CNI), Skalierbare API-Mechanik.

References & Further Reading

- **Kubernetes Docs:** etcd, API Server, Controller Manager, Scheduler, Kubelet, kube-proxy
- **Controller pattern:** client-go, controller-runtime, Informers & WorkQueues
- **Scheduling Framework & plugins; EndpointSlice; Admission Webhooks best practices**

> Primärquellen für Vertiefung und produktionsreife Gestaltung. Besonderes Augenmerk auf etcd-Operations, Admission-Latenz/Fehlermodi, Scheduler-Plugins.