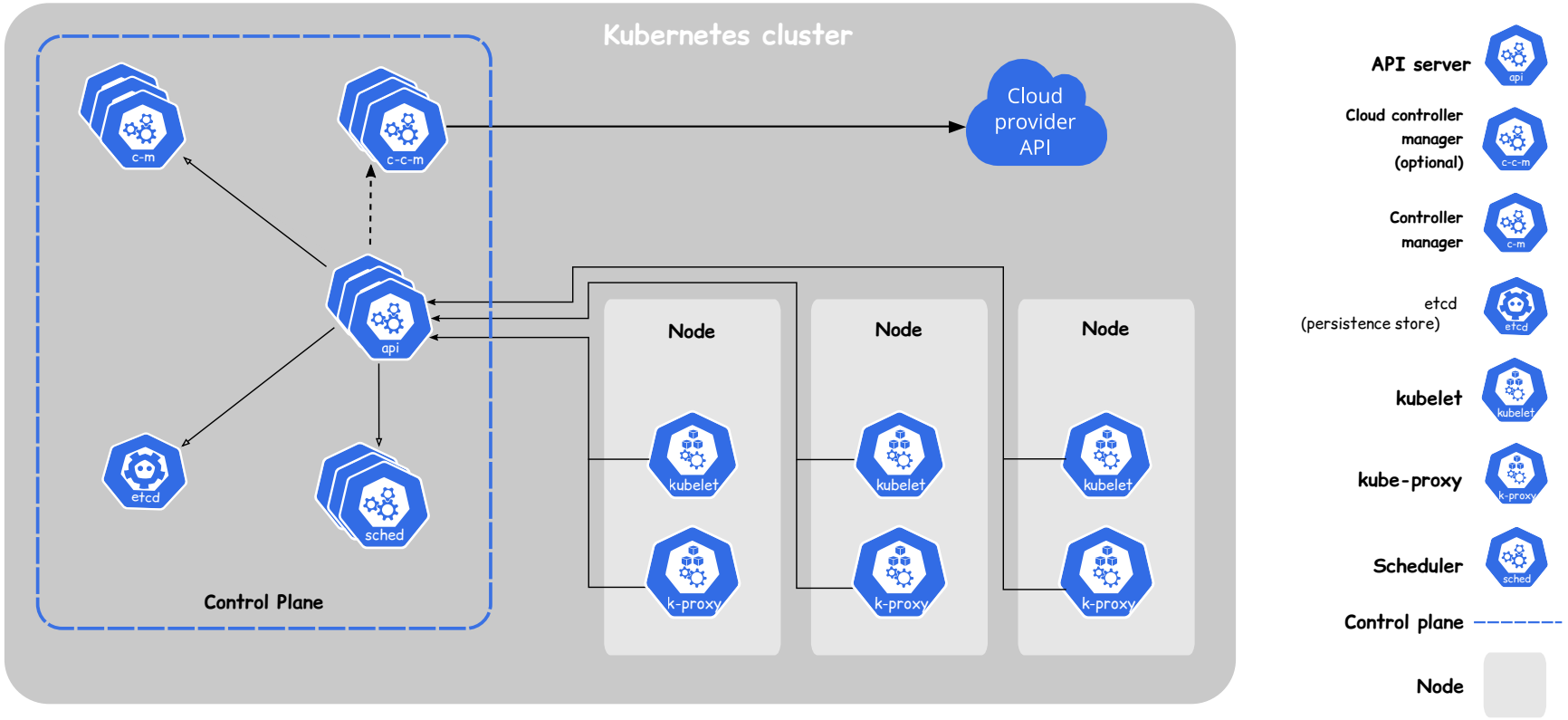
Kubernetes Control Plane in Depth

Ziping Sun

2025-06-05

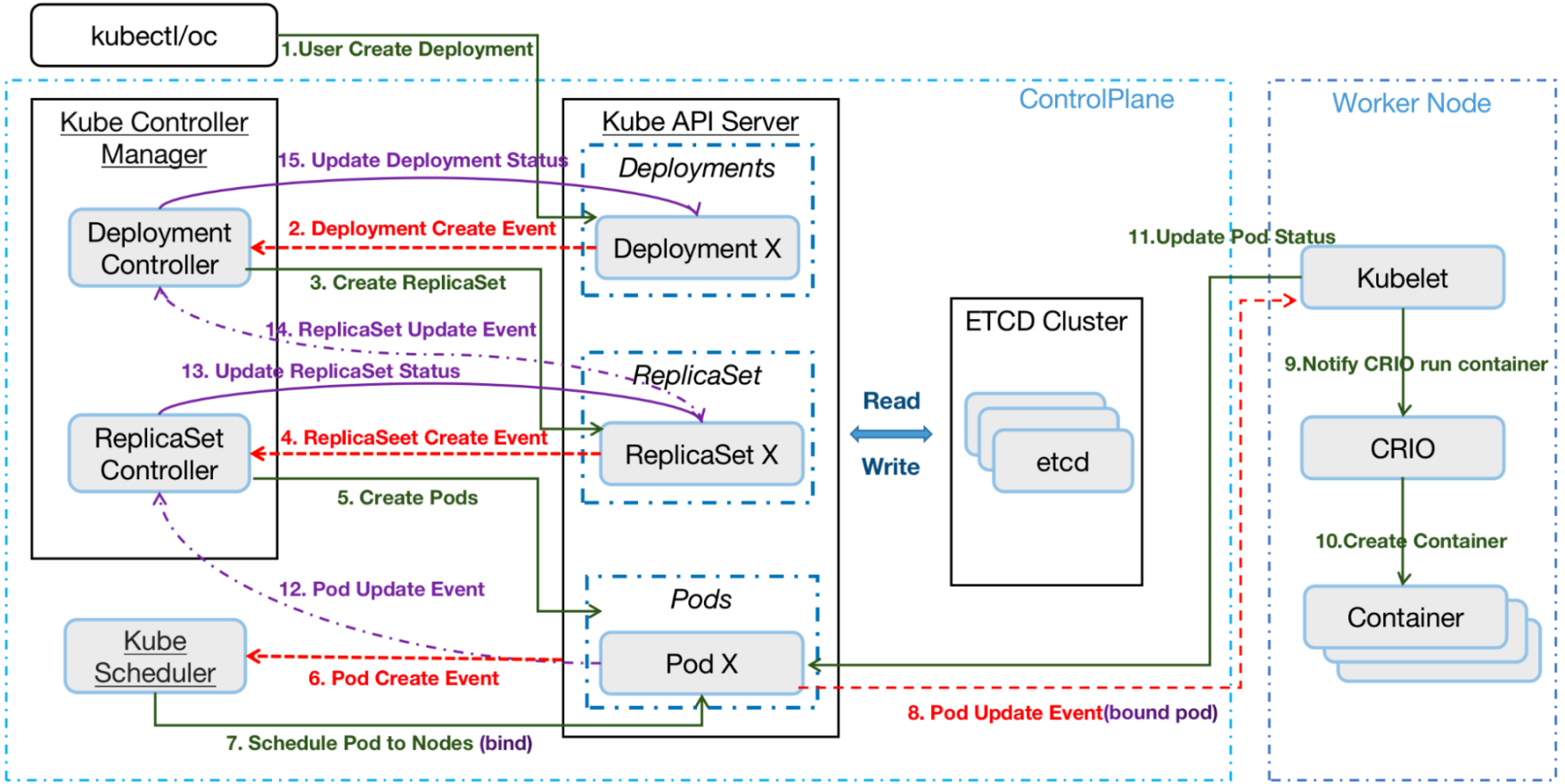
Outline

Overview: Components



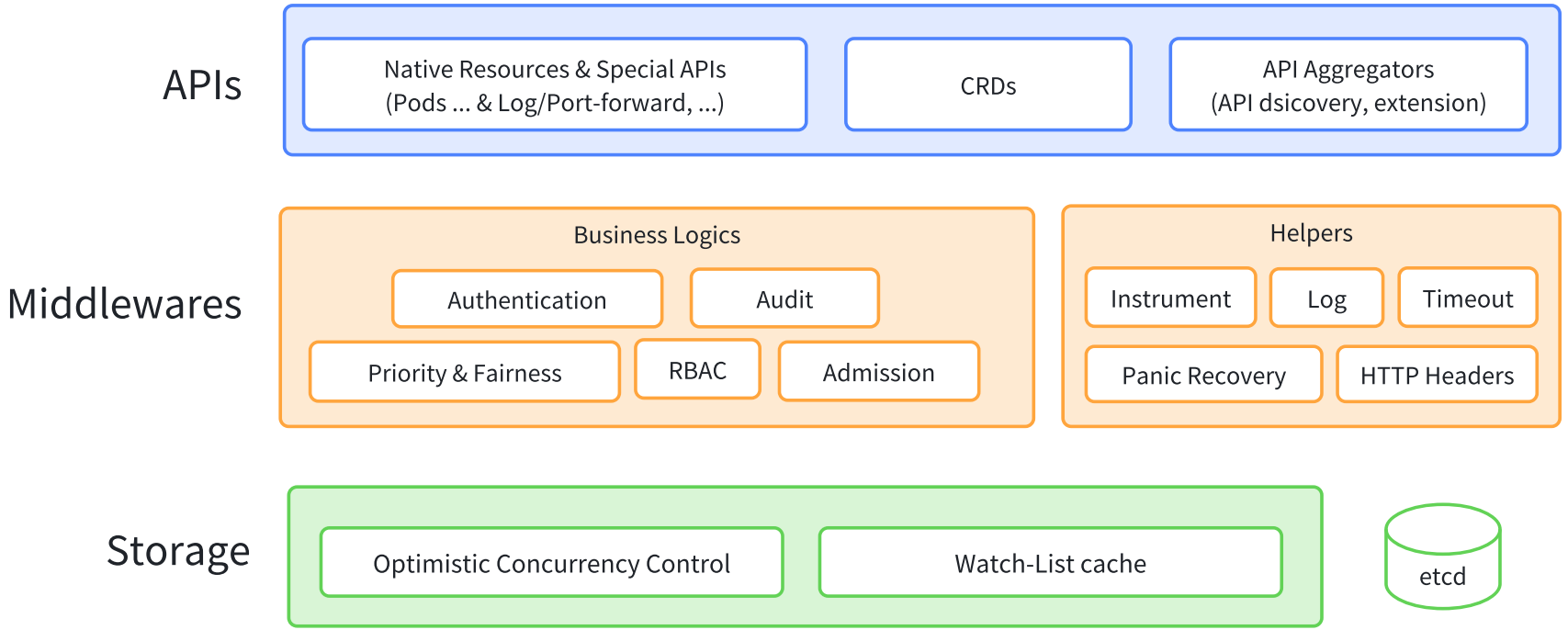
Control Plane: API Server, Controller Manager, Scheduler, etcd

Overview: Hub-and-Spoke Pattern



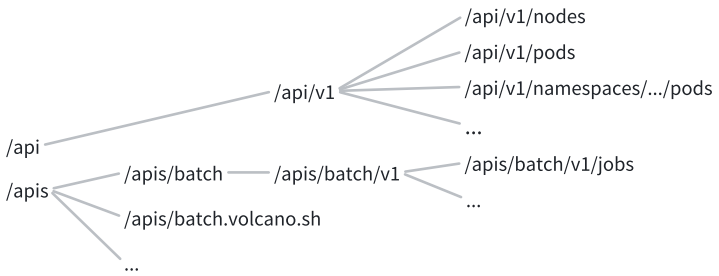
Outline

API Server: Overview



Request Flow: Middlewares (Authn, Audit, Flow Control, RBAC, Admission ⋯) → APIs → Storage

API Server: APIs



API Categories

- **Native** APIs
- CRDs: **batch.volcano.sh**
- Aggregated APIs: **metrics.k8s.io**

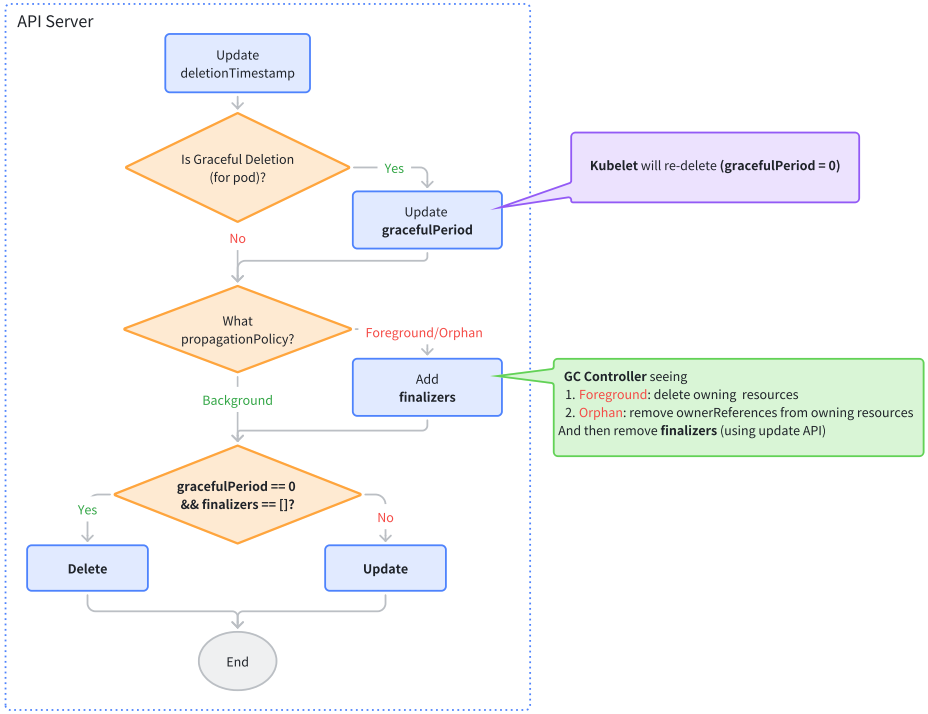
What makes native resources special?

- Special subresources
  - pods **resize/ephemeralcontainers/eviction/binding** …
  - uncommon "control → data" flow
    - pods **exec/attach/port-forward**
    - service/pod/node **proxy**
- Change API Server behavior: **CRD, API Services** …
- Virtual resources: **TokenReview** …

CRUD + Watch Consistency

- Per-resource **linearizability**
- No guarantee for watch

API Sever: APIs (deletion flow example)[1]



- Deletion might be asynchronous
- Cascade deletion is implemented by **GC Controller**
- Typical usage
  - **graceful period** Pod
  - **finalizers** PV/PVC

[2] Based on `k8s.io/apiserver/pkg/registry/generic/registry/store.go`

API Server: Middlewares[1]

Authentication

- Various methods: **X.509**, **Service Account**, **OIDC** ⋯

Audit

- Configured with static Audit Policy

Priority and Fairness v1.29

- Controlled by **FrowSchema**

Authorization

- Two methods:
  RBAC
  node (hardcoded for kubelet)

Admission

- Two forms:
  In-tree Plugin PodSecurity, NamespaceLifecycle ⋯
  Webhooks

[2] Based on `k8s.io/apiserver/pkg/server/config.go`

API Server: Storage

Optimistic concurrency control

- **ResourceVersion**

Watch Cache

- consistency read

Outline