Kubernetes Controllers

The heart of the Control Plane

What Is a Controller in Kubernetes?

A **controller** in Kubernetes is a loop that watches the state of your cluster and makes changes to **match the desired state** defined by Kubernetes objects (like Deployments, Services, etc.).

It's part of the **control plane** and runs continuously, watching objects via the API server and making corrections when needed — this is called the **controller reconciliation loop**.

The Reconciliation Loop

Each controller follows a cycle:

1. **Observe**: Watch Kubernetes objects using informers (event-driven)
2. **Compare**: Desired state (spec) vs. actual state (status)
3. **Act**: Take actions to reconcile the two — e.g., create or delete Pods

Built-in Kubernetes Controllers (Core List)

Kubernetes comes with a variety of **built-in controllers**, most of which run as part of the **kube-controller-manager** process.

Here's a categorized list of the key controllers:

Core Workload Controllers

Controller	Purpose
ReplicaSet	Ensures a specified number of pod replicas exist
Deployment	Manages ReplicaSets, supports rollouts/rollbacks
StatefulSet	Manages stateful pods with stable identities
DaemonSet	Ensures a copy of a pod runs on every node
Job	Ensures a pod runs to completion (once or N times)
CronJob	Runs jobs on a schedule

Service Discovery & Networking Controllers

Controller	Purpose
Endpoints	Creates/updates Endpoints for Services
ServiceAccount	Manages default service accounts for namespaces

Resource Lifecycle Controllers

Controller	Purpose
Namespace	Handles finalizers, cleanup of namespaced resources
ServiceAccountToken	Manages token Secrets for ServiceAccounts
PVC Protection	Protects PersistentVolumeClaims from premature deletion
PV Binding	Binds PersistentVolumes to PersistentVolumeClaims
TTL After Finished	Cleans up finished Jobs automatically (optional)

Node & Pod Lifecycle Controllers

Controller	Purpose
Node	Detects node health and updates Node status
PodGarbageCollector	Cleans up orphaned pods or terminated ones
PodDisruptionBudget	Ensures PDBs are respected during disruptions
Eviction	Handles pod eviction when node is under pressure
TaintManager	Applies taint-based evictions

Horizontal Scaling & Autoscaling

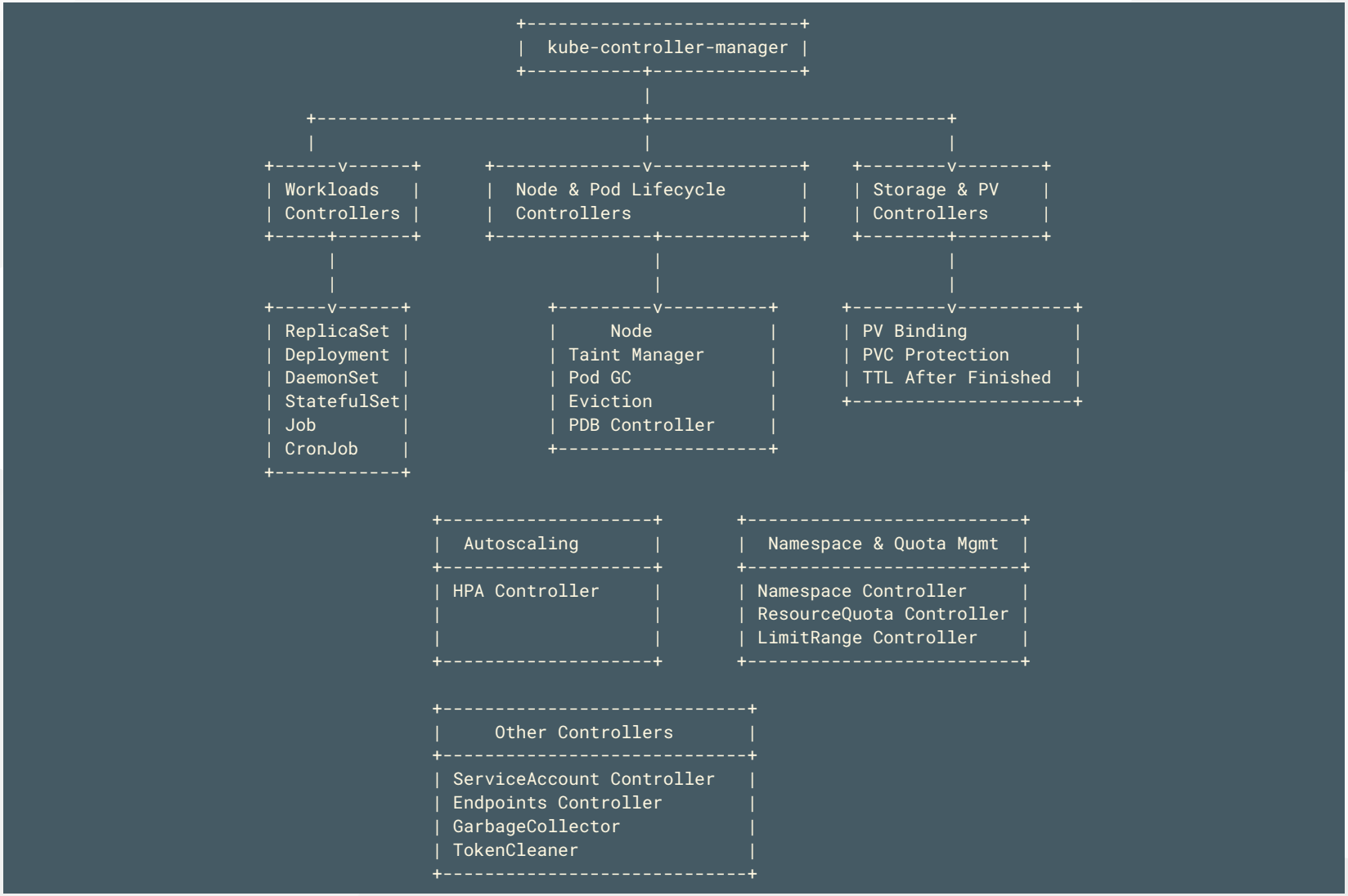
Controller	Purpose
HorizontalPodAutoscaler	Automatically scales pod replicas based on metrics

Other Important Built-ins

Controller	Purpose
ReplicationController	Legacy controller, predecessor to ReplicaSet
GarbageCollector	Handles cascading deletion via ownerRefs
ResourceQuota	Enforces resource limits per namespace
LimitRange	Applies default resource requests/limits
TokenCleaner	Cleans expired bootstrap tokens

Where They Run

- Most controllers run inside the `kube-controller-manager`
- Some run in other system components (e.g., kubelet, scheduler)
- Others are now optional (like TTL controller, enabled via feature gate)


kube-controller-manager
Workloads Controllers
ReplicaSet
Deployment
DaemonSet
StatefulSet
Job
CronJob
Node & Pod Lifecycle Controllers
Node
Taint Manager
Pod GC
Eviction
PDB Controller
Storage & PV Controllers
PV Binding
PVC Protection
TTL After Finished
Autoscaling
HPA Controller
Namespace & Quota Mgmt
Namespace Controller
ResourceQuota Controller
LimitRange Controller
Other Controllers
ServiceAccount Controller
Endpoints Controller
GarbageCollector
TokenCleaner