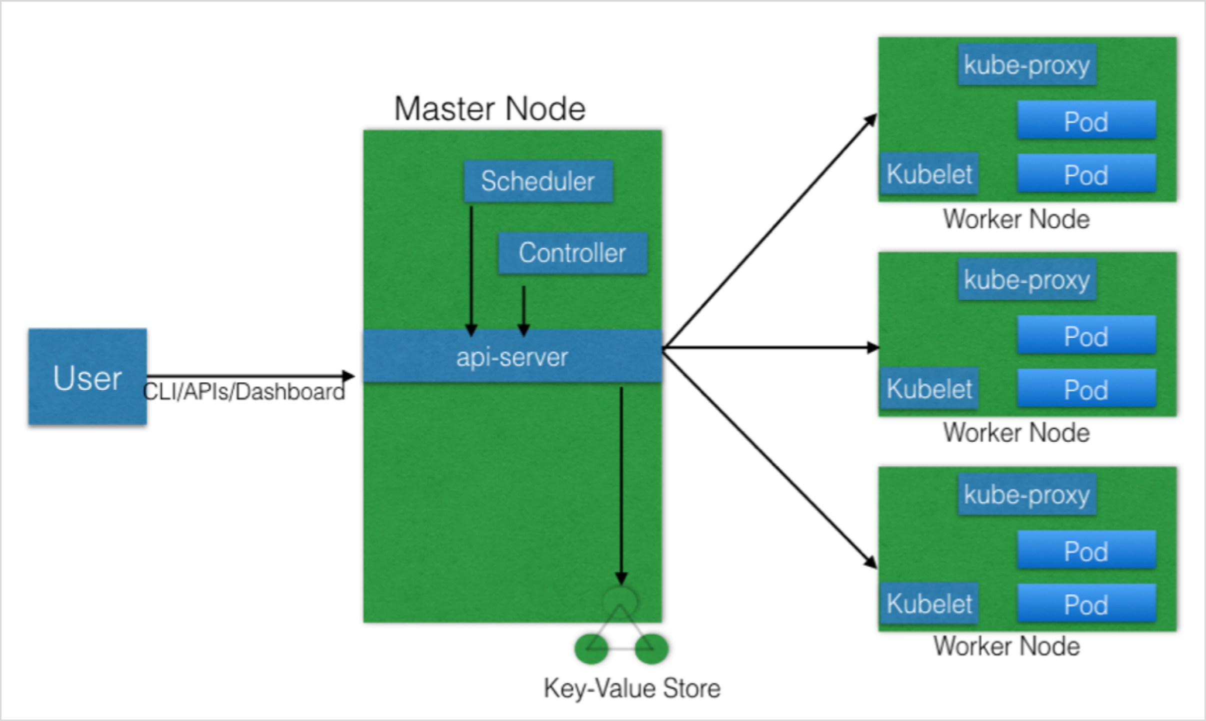
Chapter 4 Kubernetes Architecture

Master node

Worker Node

Networking Challenges



Master node

- Provides a running environment for the control plane
  - Manages the state of a Kubernetes cluster
  - Brain behind all operations inside the cluster
- User interacts with master node
- Replicas exist in the cluster with sync'd control plane components
  - Minimize downtime

API Server (kube-apiserver)

- Intercepts, validates, and processes RESTful calls
- talks to etcd data store (fetches and changes state)
- Custom API servers
  - Primary API server becomes a proxy to all secondary custom ones
  - Routes the calls to them based on custom defined rules

Scheduler (kube-scheduler)

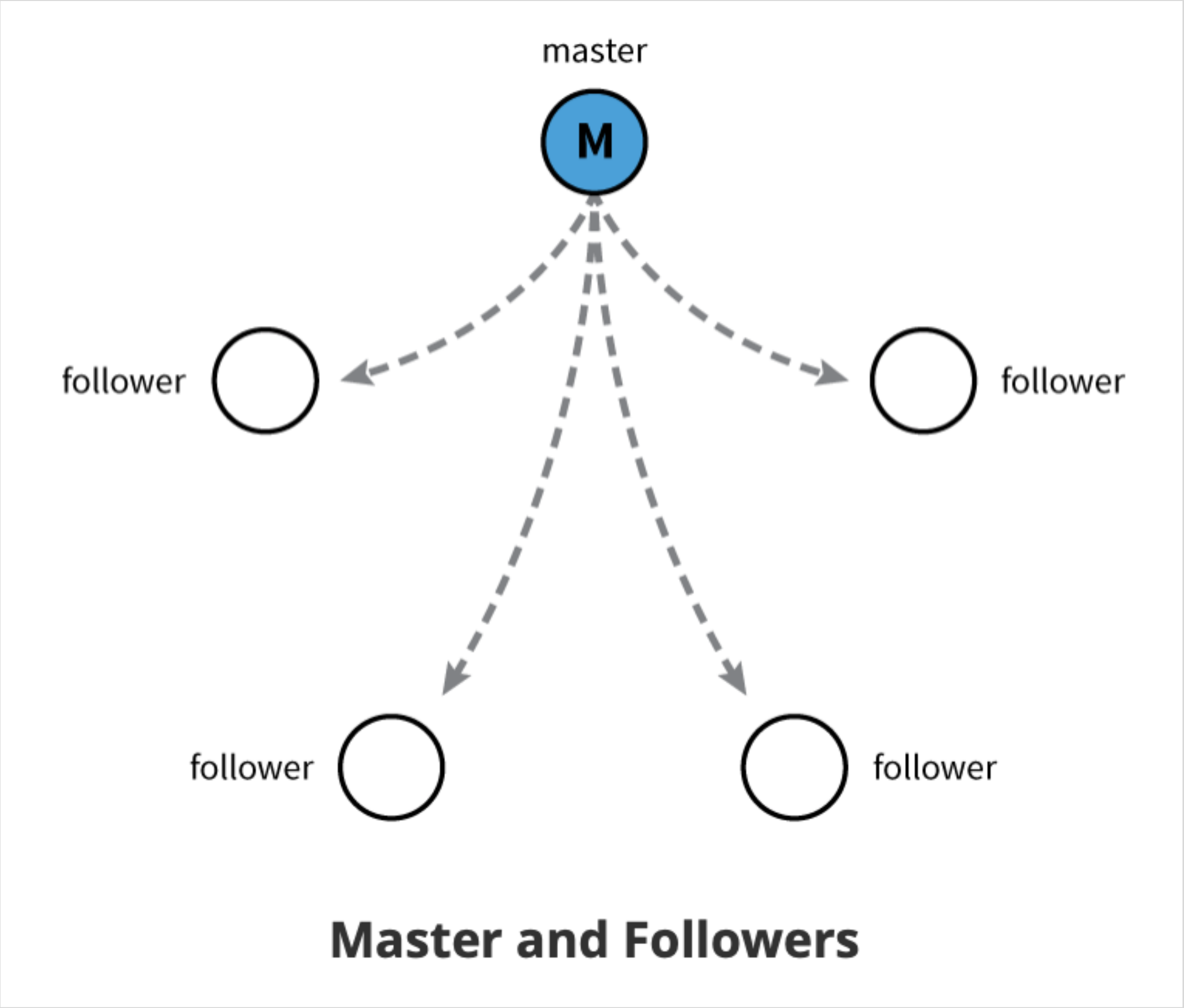
- Assigns new objects to nodes
- Talks to API server to learn about the new state in etcd

Controller managers

- Regulates the state of cluster
- Kube-controller-manager checks nodes’
    - Availability
    - Pod count
    - Endpoints
    - Service accounts
    - API access tokens
- Cloud-controller-manager
    - Interact with cloud provider when nodes become unavailable
    - Manage storage volumes provided by cloud service
    - Manage LB and routing

Etcd: Distributed Key-Value **State Store**

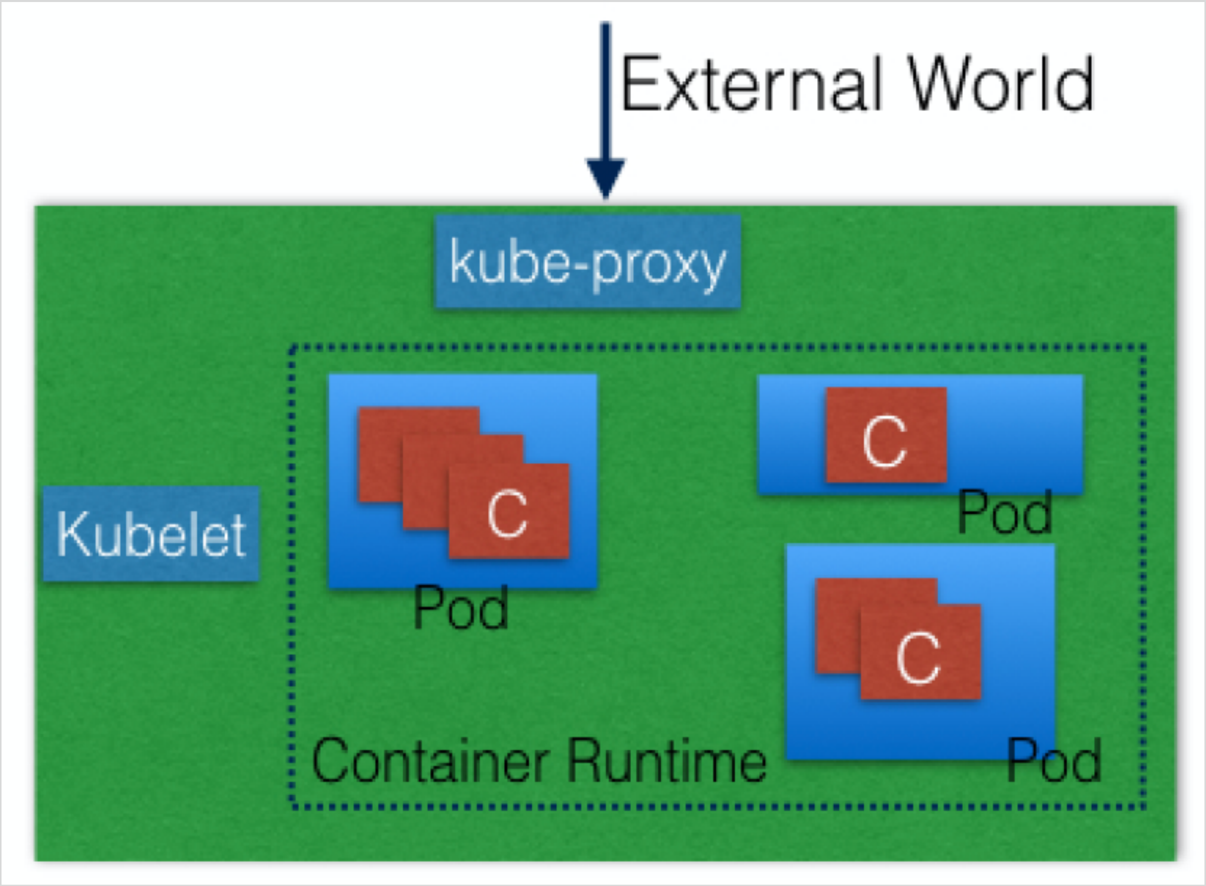
- Holds cluster state related data, not client workload data
- Stacked: on master node
- External: on dedicated host
- New data is appended, never replaced
- Obsolete data is compacted periodically
- Very important to replicate data stores in HA mode
- Raft Consensus Algorithm



  - Failure protection
  - Any one node as master, others as followers
- Written in Go
- Stores subnets, ConfigMaps, Secrets, etc.

Worker Node

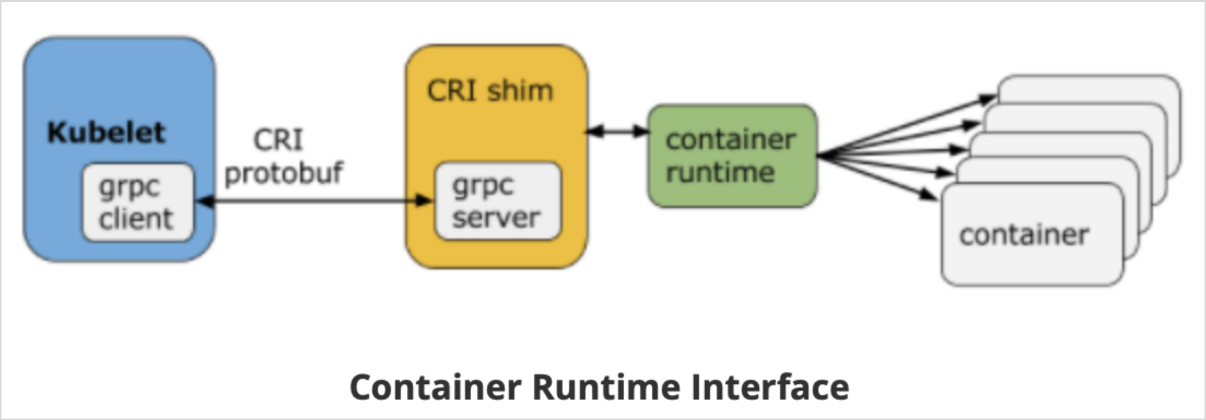
back to top



- Environment for Client applications
- Apps encapsulated in pods
  - Controlled by the cluster control plane agents (on MN)
- Can talk to each other + outside world
- Pod: smallest scheduling unit

Kubelet

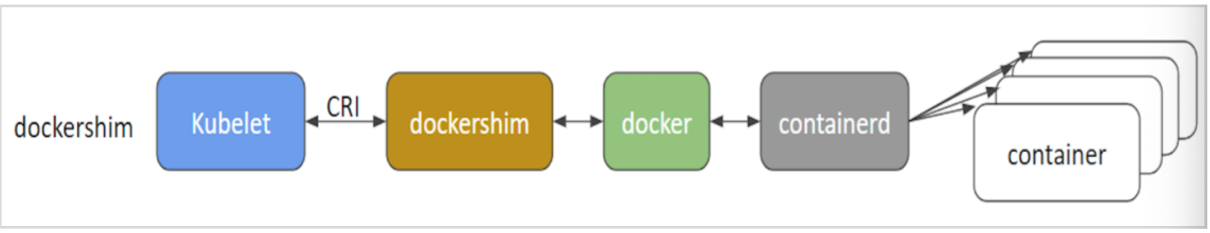
- Agent that communicates with the control plane components
- Receives pod definitions
- Interacts with the container runtime (CR) to run containers associated with the Pod
- Monitors health of pods' containers
- Connects to CR using Container Runtime Interface (CRI)



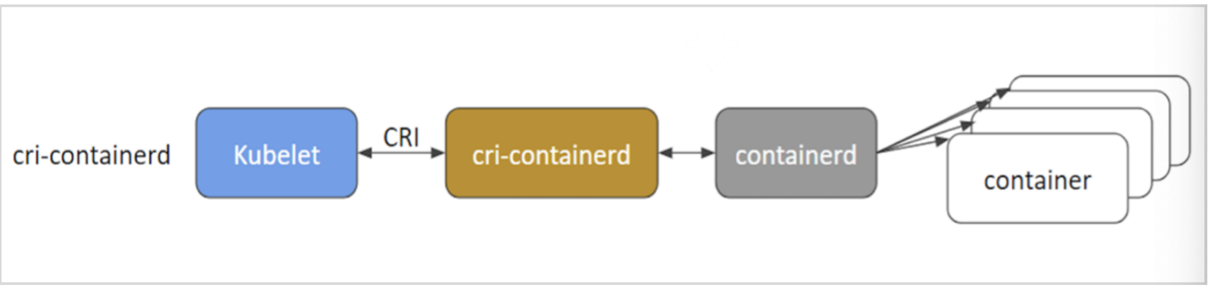
- kubelet acts as grpc client
- connects to CRI shim activating as grpc server
- CRI implements two services:
  - ImageService: image-related operations
  - RuntimeService: pod and container-related operations

CRI Shims

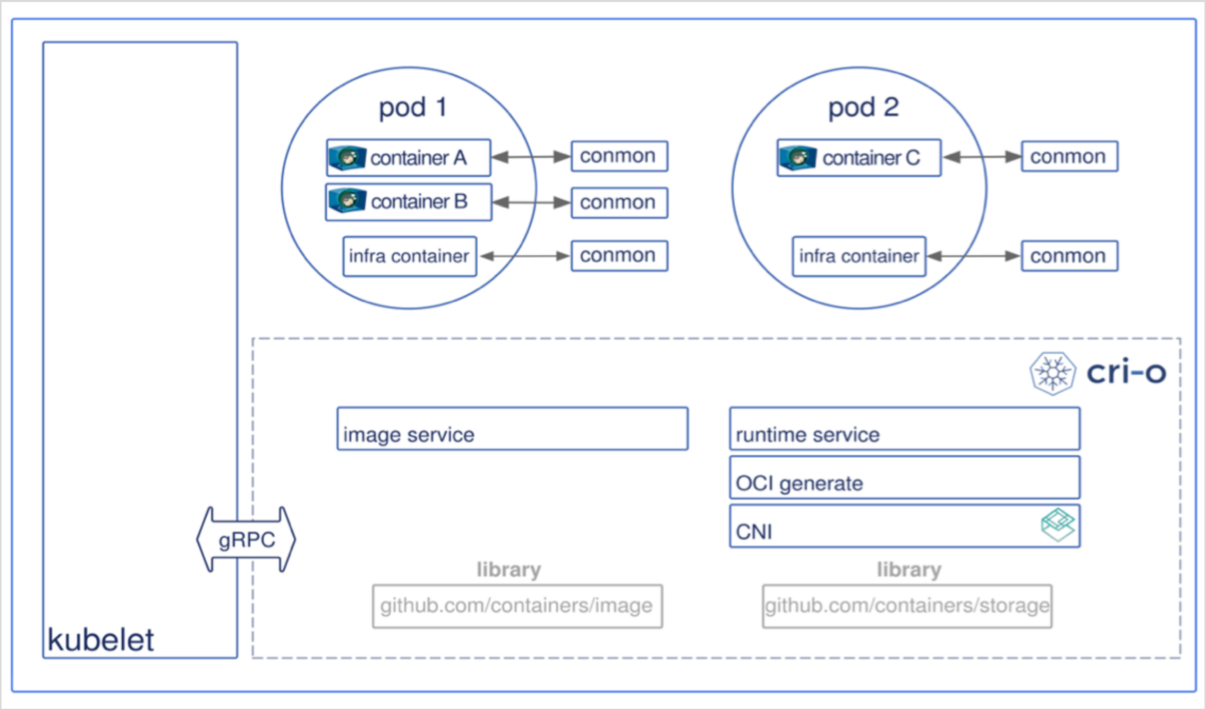
- Dockershim



  - containers to create and manage containers
- Cri-containerd



  - Directly use Docker’s smaller offspring containerd
- CRI-O



- enables using any Open Container Initiative (OCI) compatible runtimes with Kubernetes

Kube-Proxy

- Network agent on each node
- Dynamic updates and maintenance of networking rules
- Forwards connection requests to pods

Add-ons

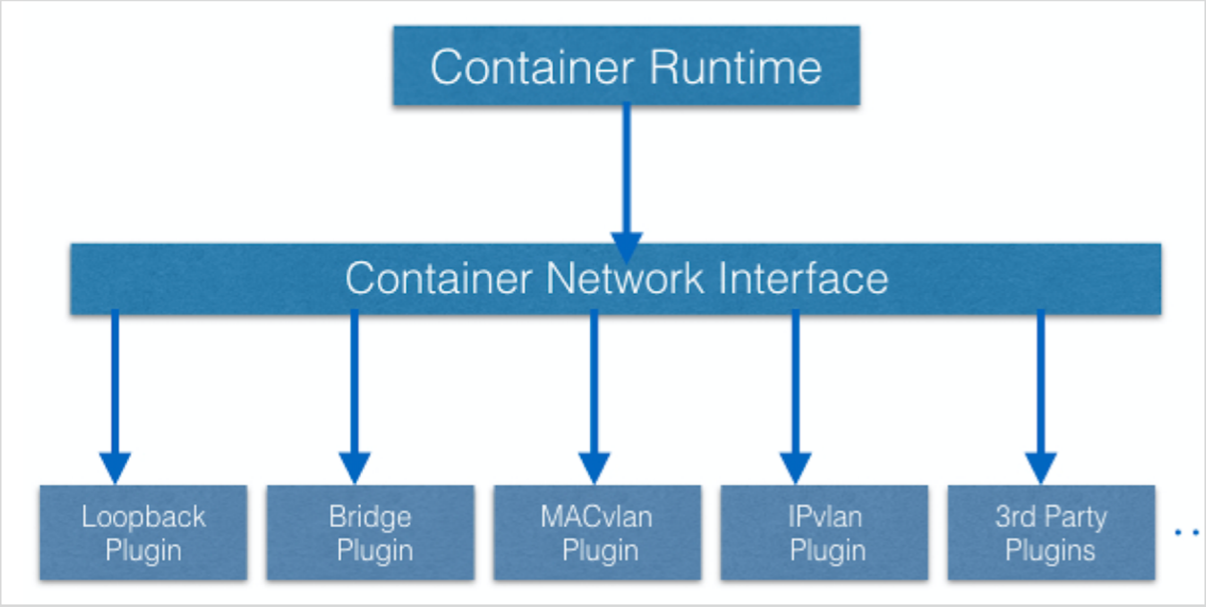
- 3rd party pods and services
- Cluster DNS, Dashboard, Monitoring, Logging

Networking Challenges

Back to top

- Container-to-Container inside Pods
  - Network Namespace
    - Created when a pod is started by container runtime
    - Shared across containers on the pod
      - Talk to each other via localhost
- Pod-to-Pod Communication Across Nodes
  - IP-per-pod: similar to VMs on a network
  - Containers inside pods coordinate port assignments
  - Container Network Interface (CNI)

- set of specification and libraries
- allows plugins to configure the networking for containers
- 3rd-party Software Defined Networking (SDN) solutions



- CR offloads the IP assignment to CNI
- CNI connects to the plugins and get the IP address
- CNI forwards the IPs to CR

- Pod-to-External World communication
  - Services
    - Enable external accessibility
    - Encapsulate networking rules definition on cluster nodes
  - Kube-proxy
    - Exposes services to the external world
    - Applications accessible by outside world over a virtual IP