Control plane

Control plane

- Control plane keeps track of Kubernetes Objects and desired state
- Kubernetes Master
  - Responsible for maintaining the desired state for your cluster
  - Collection of processes managing state
- Kubernetes Kubelet

- Manages everything as discussed in ‘Managing workload’

Master Components

- Kube-apiserver
  - Runs the K8s API. Frontend for control plane
- Etcd
  - HA key-value store for all cluster data
- Kube-scheduler
  - Watches newly created pods and assigns them to a node
- Kube-controller-manager
  - Node Controller, manages up-down of nodes in cluster
  - Replication Controller, manages correct number of pods
  - Endpoints Controller, joins services and pods
  - Service Account & Token Controllers
- Cloud-controller-manager

Node components

- Kubelet
  - Manages running containers described in pods (PodSpecs)
- Kube-proxy
  - Network proxy making sure Service concept works
  - Manages network rules on the node
- Container Runtime

Running Containers

- Kubernetes CRI
- Open Container Initiative containers

- Docker
- Cri-o https://cri-o.io/
- Containerd

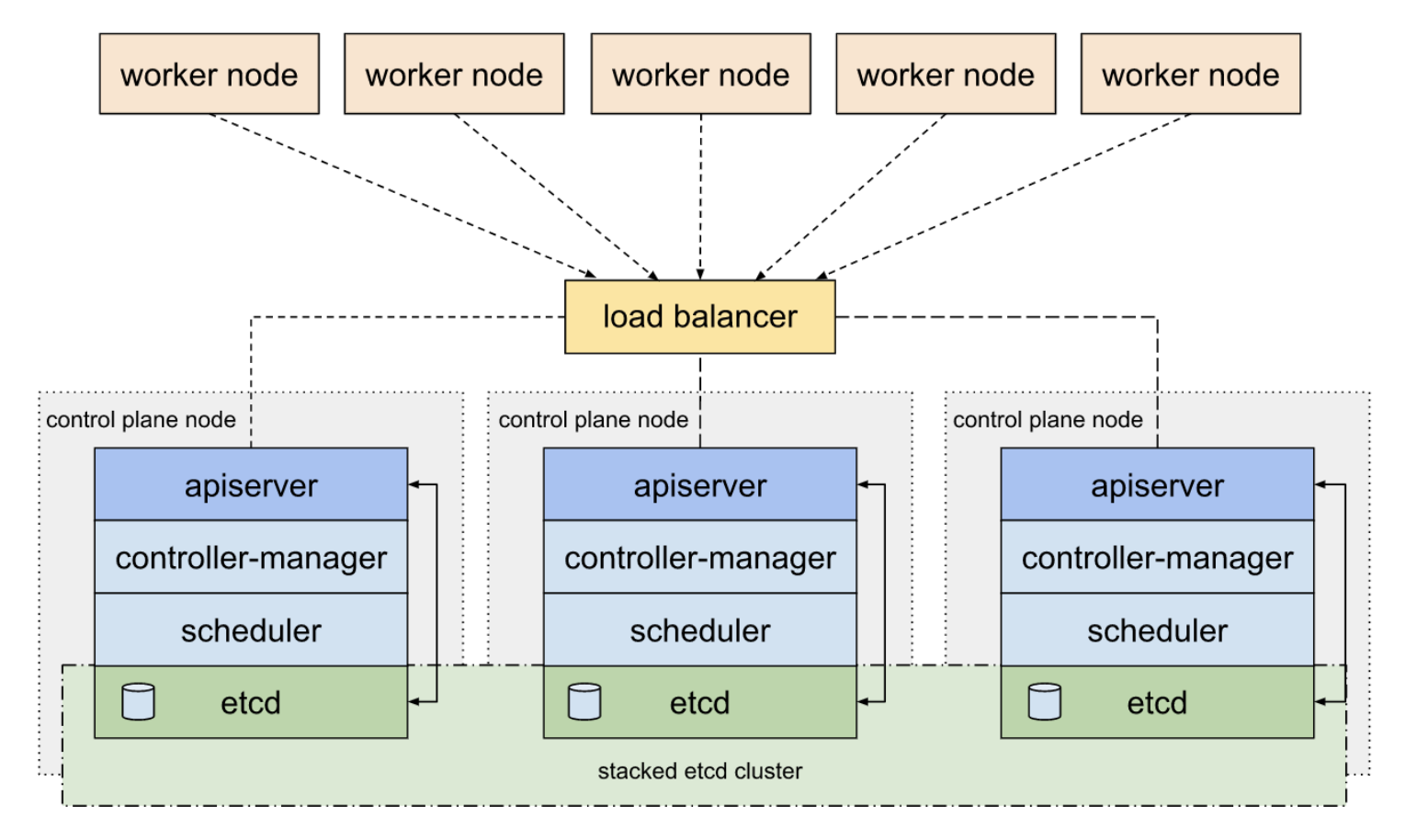
Control plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

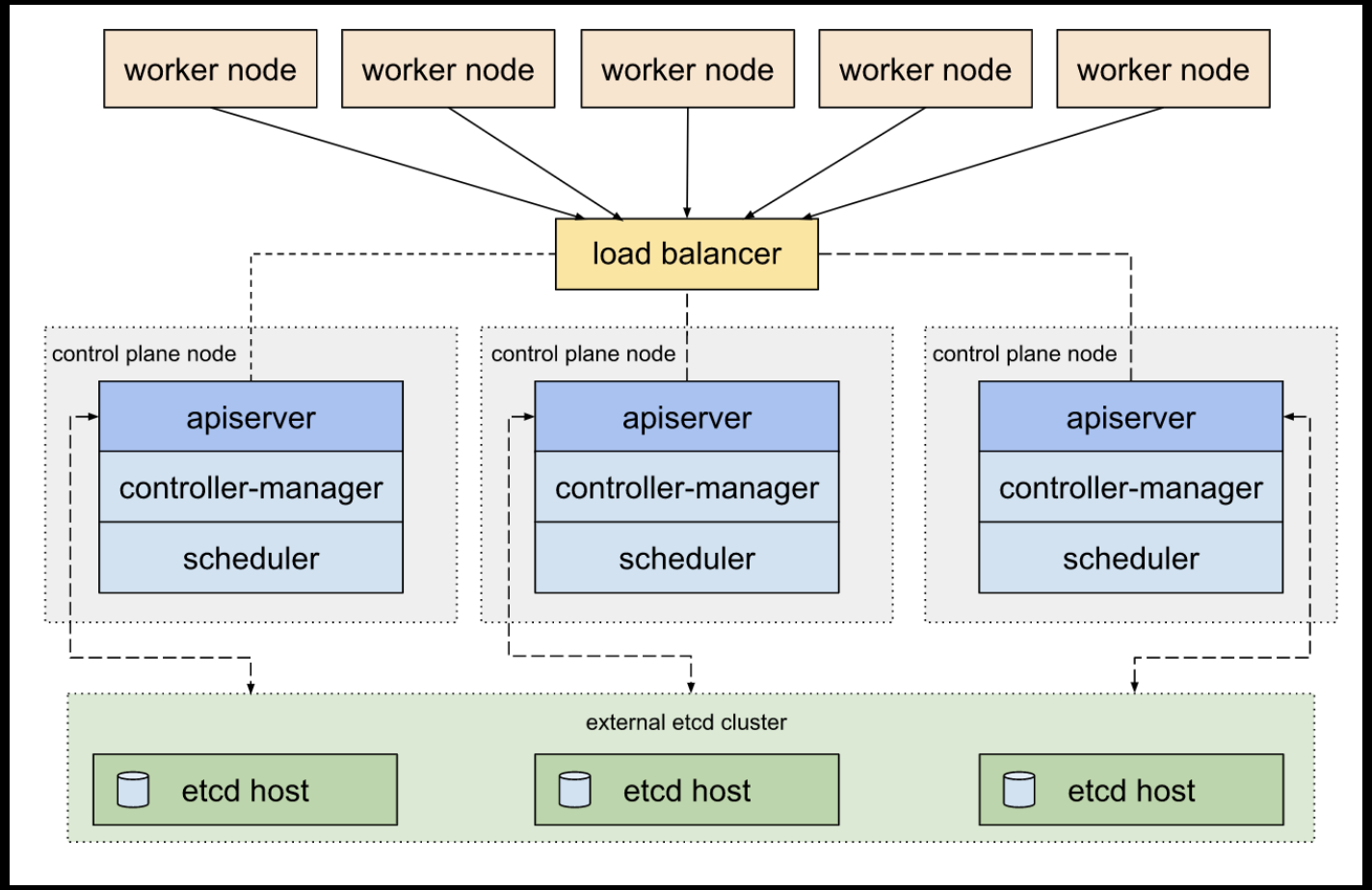
Etcd

- Stores all object definitions for Kubernetes
- Etcd is defined as a distributed, reliable key-value store for the most critical data of a distributed system
- Etcd is deployed as a cluster
  - Communication is handled by a protocol called RAFT
  - Needs at least 3 nodes for production

Etcd


worker node
worker node
worker node
worker node
worker node
load balancer
control plane node
control plane node
control plane node
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
stacked etcd cluster

K8s with external Etcd cluster



Etcd

- Backup data

```
ETCDCTL_API=3 etcdctl --endpoints $ENDPOINT snapshot save snapshotdb
```

- Restore (on all members in the cluster!)

```
ETCDCTL_API=3 etcdctl snapshot restore snapshot.db \
  --name m1 \
  --initial-cluster
m1=http://host1:2380,m2=http://host2:2380,m3=http://host3:2380 \
  --initial-cluster-token etcd-cluster-1 \
  --initial-advertise-peer-urls http://host1:2380
```