KUBERNETES – INTERNALS

Github repo..

https://github.com/vsaini44/KubernetesRepo.git

Understanding the Architecture

A Kubernetes cluster is split into two parts:

- **The Kubernetes Control Plane**
- **The (worker) nodes**

Control Plane Components

The Control Plane is what controls and makes the whole cluster function.

The components that make up the Control Plane are

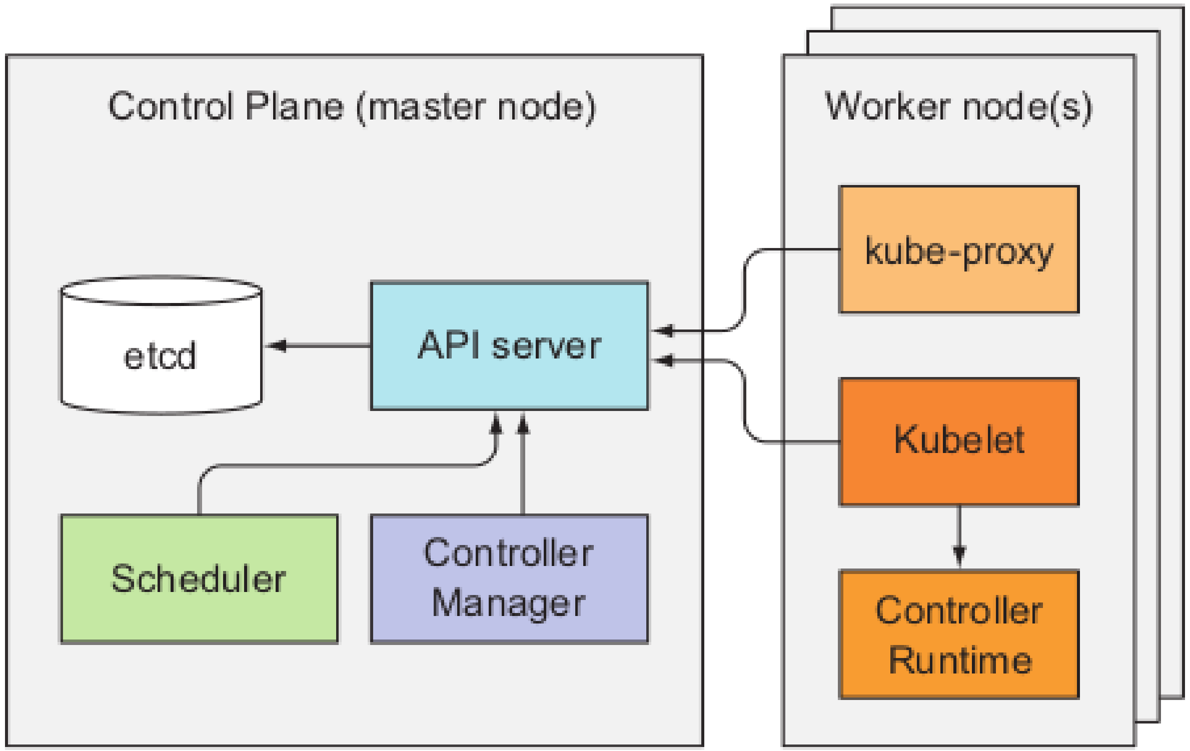
- **The etcd distributed persistent storage**
- **The API server**
- **The Scheduler**
- **The Controller Manager**

Worker Node Components

The task of running your containers is up to the components running on each worker node:

- **The Kubelet**
- **The Kubernetes Service Proxy (kube-proxy)**
- **The Container Runtime (Docker, rkt, or others)**

Architecture ..



How kubernetes uses ETCD

Kubernetes stores all the objects in etcd which is is a fast, distributed, and consistent key-value store.

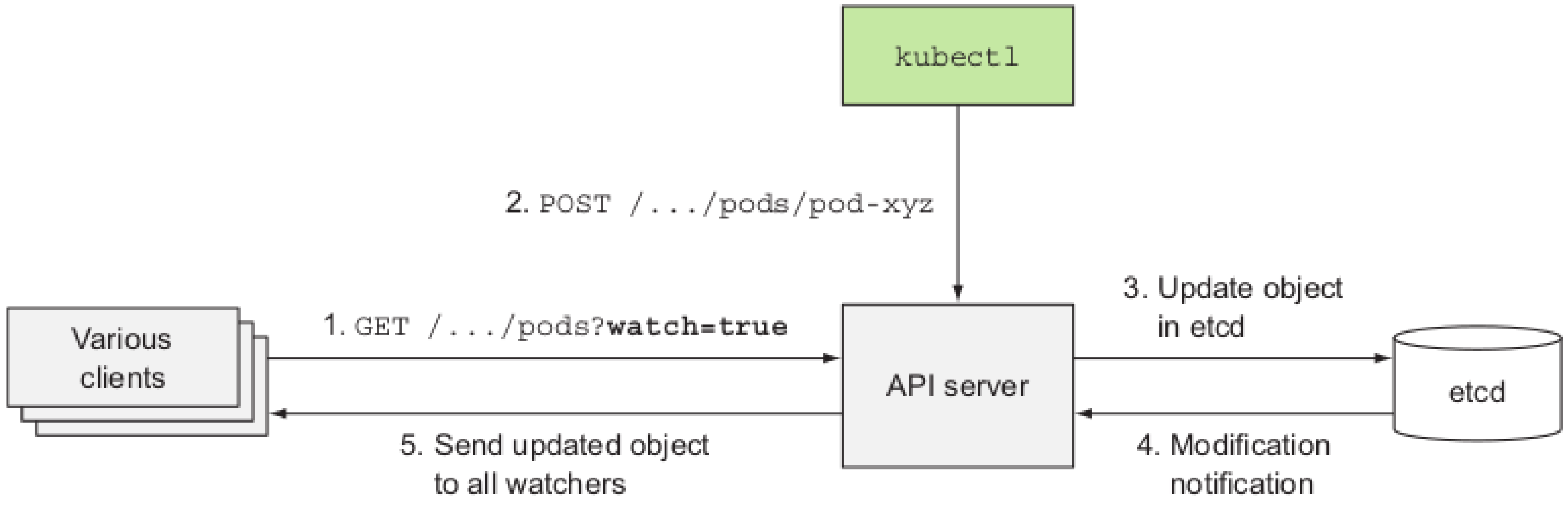
Because it’s distributed, you can run more than one etcd instance to provide both high availability and better performance.

How does API Server works?

The Kubernetes API server is the central component used by all other components and by clients, such as kubectl .

It provides a CRUD (Create, Read, Update, Delete) interface for querying and modifying the cluster state over a RESTful API. It stores that state in etcd.

Api Service



kubectl
2. POST /.../pods/pod-xyz
3. Update object in etcd
Various clients
1. GET /.../pods?watch=true
API server
etcd
5. Send updated object to all watchers
4. Modification notification

Schedular Working

We don’t usually specify which cluster node a pod should run on. This is left to the Scheduler. From afar, the operation of the Scheduler looks simple.

All it does is wait for newly created pods through the API server’s watch mechanism and assign a node to each new pod that doesn’t already have the node set.

Schedular Working

Hardware check: Can the node fulfill the pod’s requests for hardware resources

Network check: If the pod requests to be bound to a specific host port

Storage check: The volume be mounted for this pod on this node,

Affinity, Anti affinity , Node selector etc.

Controller Manager Working

The Scheduler only assigns a node to the pod, so you need other active components to make sure the actual state of the system converges toward the desired state, as specified in the resources deployed through the API server. This work is done by controllers running inside the Controller Manager

Controller Manager Working

The list of these controllers includes the:

Replication Manager (a controller for ReplicationController resources)

ReplicaSet, DaemonSet, and Job controllers

Deployment controller

Node controller

Service controller

Endpoints controller

Namespace controller

PersistentVolume controller

Kubelet Working

The Kubelet is the component responsible for everything running on a worker node. Its initial job is to register the node it’s running on by creating a Node resource in the API server.

Then it needs to continuously monitor the API server for Pods that have been scheduled to the node, and start the pod’s containers.

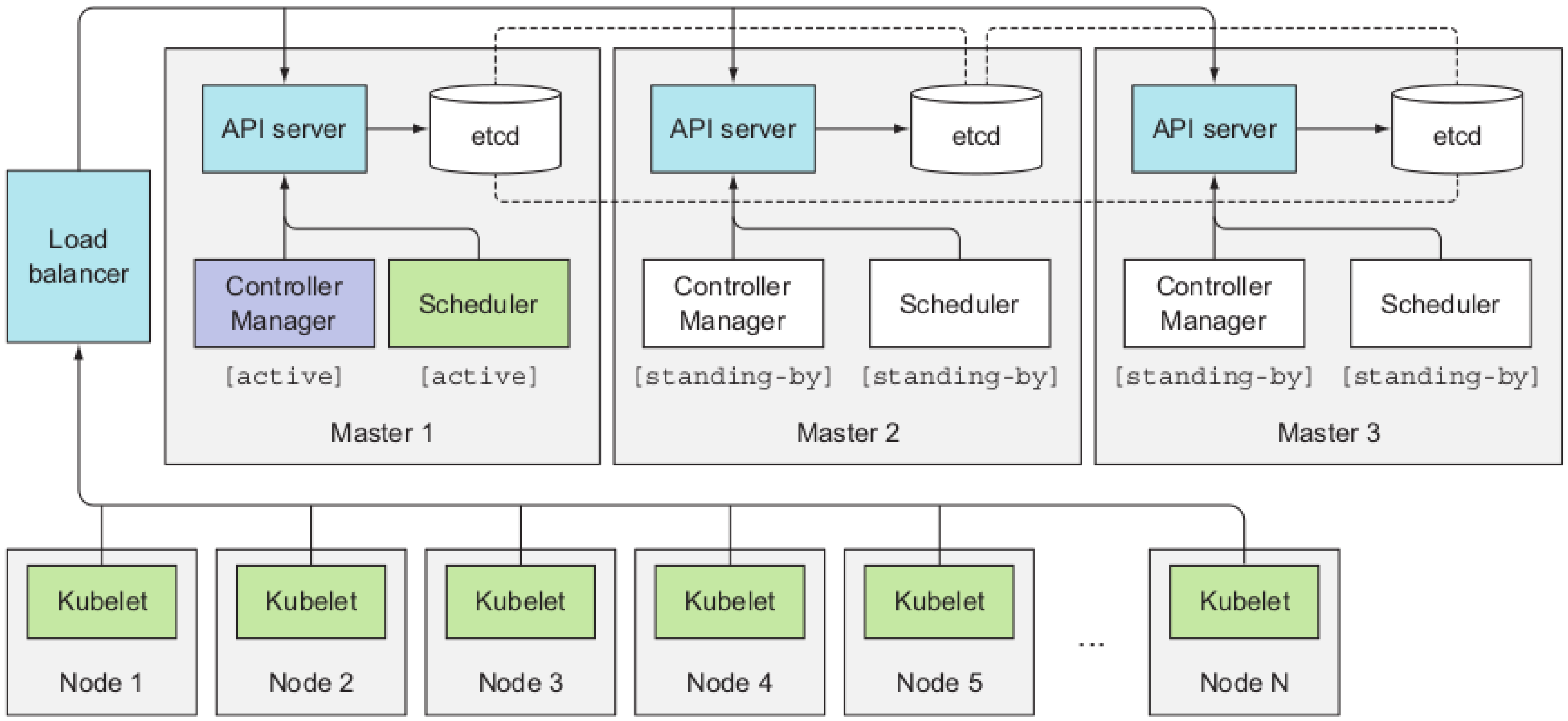
Kube-Proxy Working

Beside the Kubelet, every worker node also runs the kube-proxy, whose purpose is to make sure clients can connect to the services you define through the Kubernetes API.

The kube-proxy makes sure connections to the service IP and port end up at one of the pods backing that service (or other, non-pod service endpoints).

When a service is backed by more than one pod, the proxy performs load balancing across those pods.

High Available Cluster Architecture


Load balancer
API server
etcd
Controller Manager
Scheduler
[active]
[active]
Master 1
API server
etcd
Controller Manager
Scheduler
[standing-by]
[standing-by]
Master 2
API server
etcd
Controller Manager
Scheduler
[standing-by]
[standing-by]
Master 3
Kubelet
Node 1
Kubelet
Node 2
Kubelet
Node 3
Kubelet
Node 4
Kubelet
Node 5
...
Kubelet
Node N