Amirkabir University of Technology
(Tehran Polytechnic)

Introduction to Kubernetes

Saman Hoseini

Fall 2021

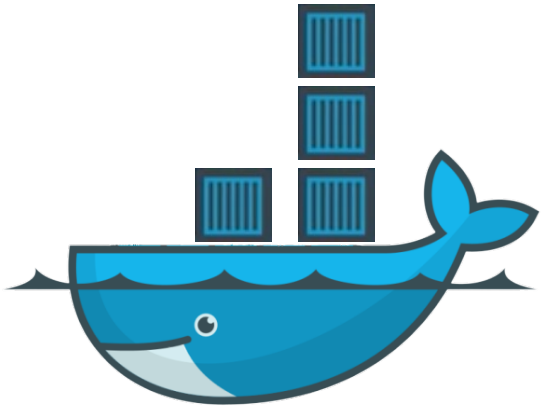
Watch Videos Offline → Ask Questions

- Lecture by Mr. Hoseini (part of your final exam)
  - https://blue.aut.ac.ir/playback/presentation/2.3/f4901a6812d33b53b0850a7617177631478d2520-1634988525036
- Workshops by Mr. Hoseini (critical for HW2 but not part of exam)
  - https://blue.aut.ac.ir/playback/presentation/2.3/f4901a6812d33b53b0850a7617177631478d2520-1635161348272
  - https://blue.aut.ac.ir/playback/presentation/2.3/f4901a6812d33b53b0850a7617177631478d2520-1635405800789

Monolithic applications

- A collection of **tightly coupled** components that have to be **developed, deployed and managed** as **one entity**.
- Scaling up is easy but scaling out requires **major code changes**.
- If a **single part** of an application is **unscalable** then **the whole application becomes unscalable**.

docker run my-image



Microservice Architecture

- These and other problems have forced us to start splitting complex monolithic applications into **smaller independently deployable** components called **microservices**.

  Microservices **communicate** through:

  1. Synchronous protocols such as **REST API** or **gRPC**
  2. Asynchronous protocols such as **AMQP** & **MQTT**

Microservice drawbacks

➢ Service Discovery

- Microservices perform their work together as a team, so they need to **find** and **talk** to each other.
- With **increasing numbers of microservices**, this becomes **tedious** and **error-prone**.

Microservice drawbacks (cont.)

- **Component Management**
  - Increase in number of components:
    - deployment-related decisions become increasingly **difficult**
    - the number of **deployment combinations increase**
    - the number of **inter-dependencies** between the components **increases** by an even greater factor

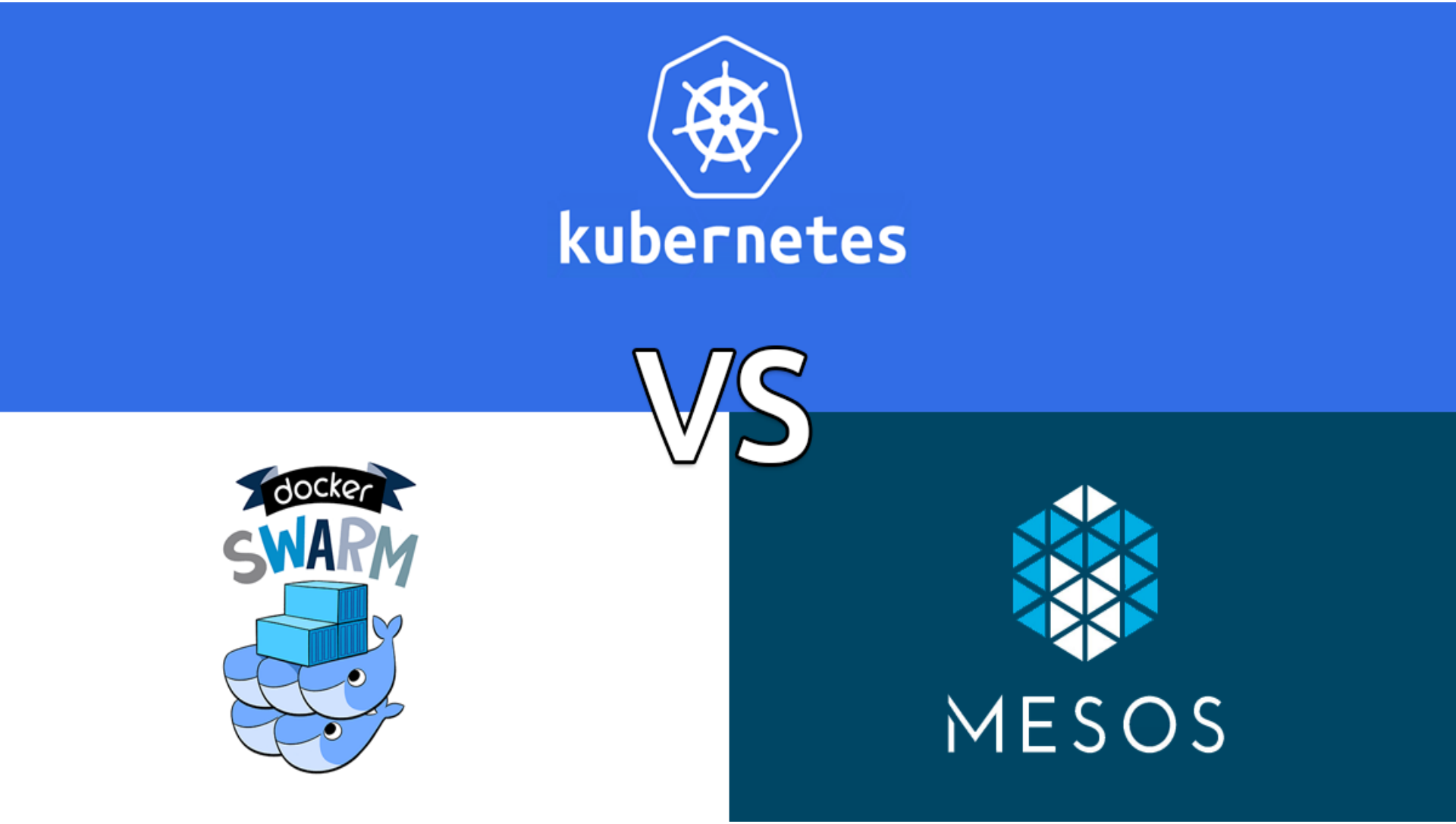
Microservice drawbacks (cont.)

- **Library Conflicts**
  - When each component has its own development team, nothing impedes each team from **using** different **libraries** and **replacing** them whenever the need arises.

Solution:
Container Orchestration

Container orchestration advantages

1. Scaling according to load of the system
2. Advanced network between containers in different hosts
3. Sharing storage between the hosts
4. Configuration managements
5. Clusters security


kubernetes
VS
docker
SWARM
MESOS

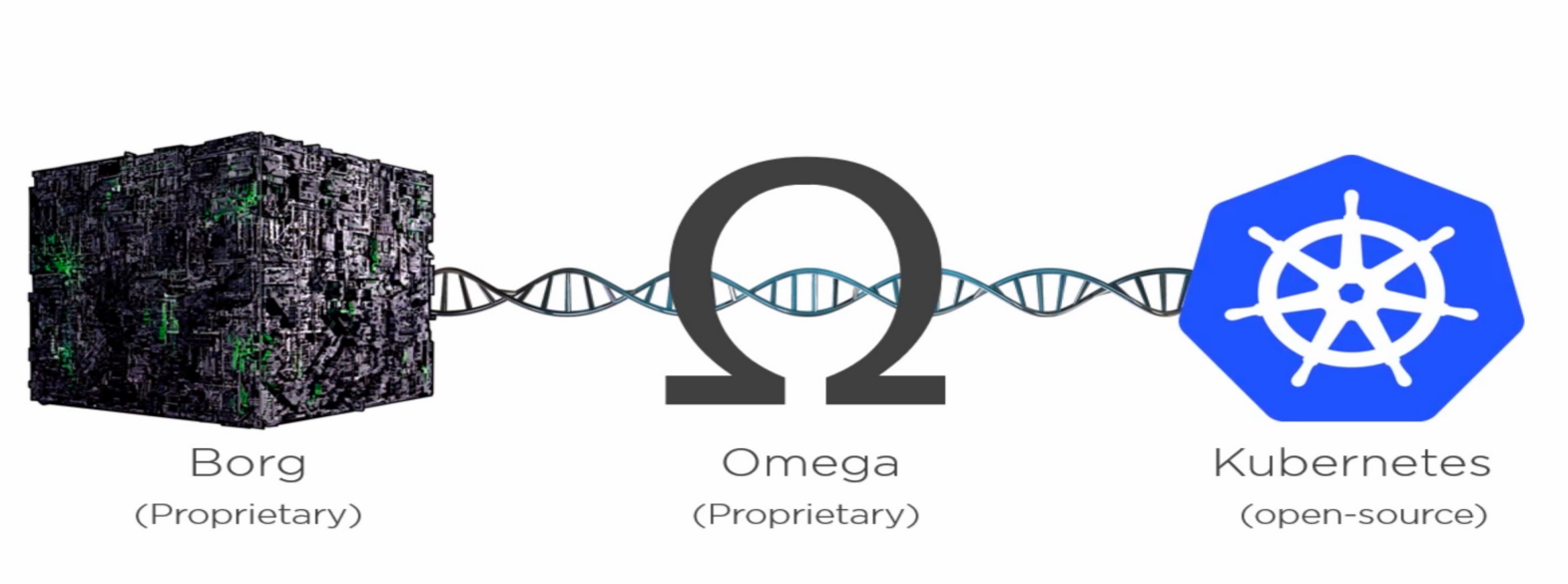
kubernetes

Where it came from?

- Born in Google
- Donated to CNCF in 2014 (open source)
- Written in Go/Golang
- https://github.com/kubernetes/kubernetes
- Often shortened to k8s

DNA

- **Borg**: a cluster manager used by Google
- **Omega**: an offspring of Borg

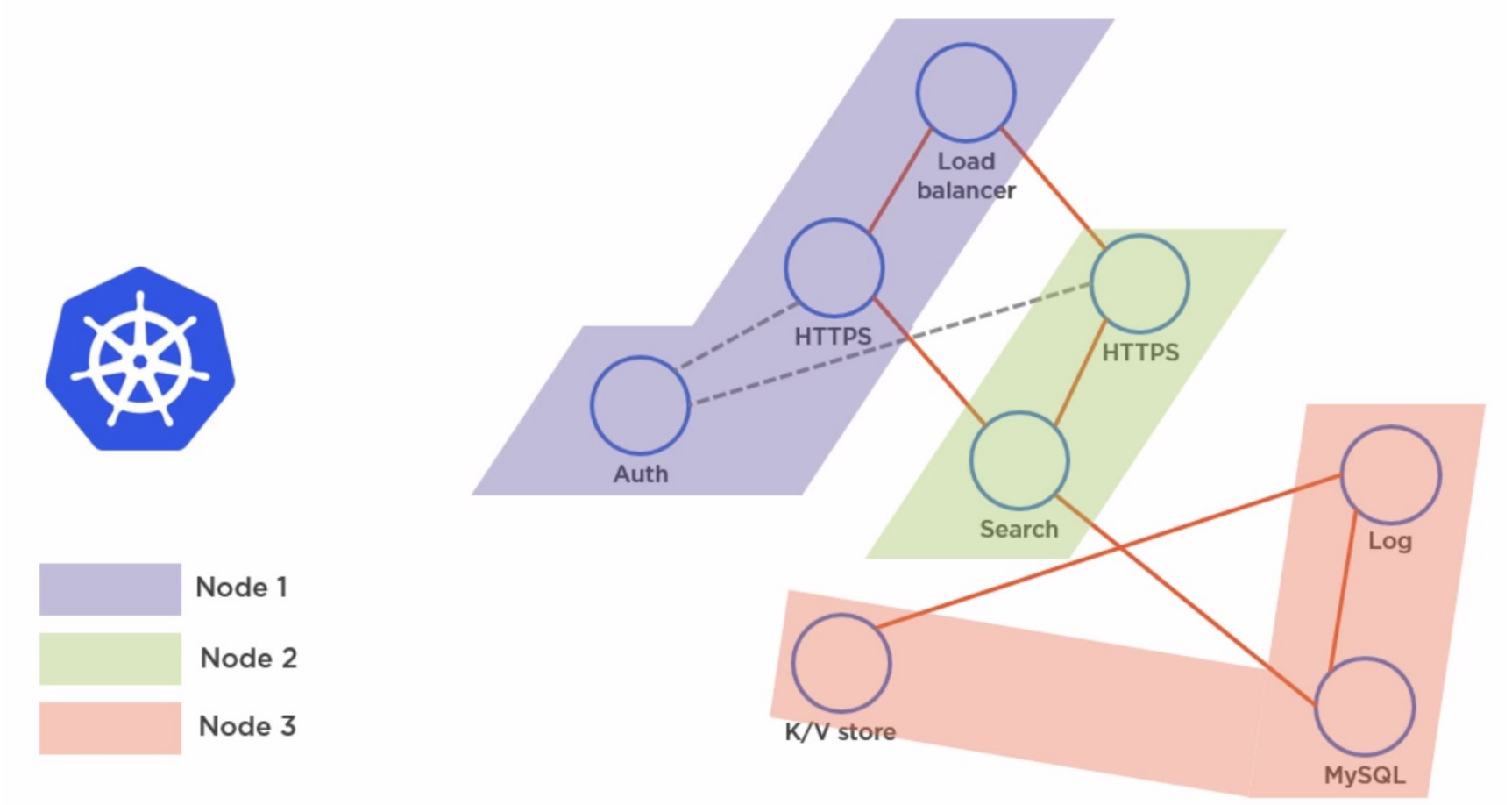


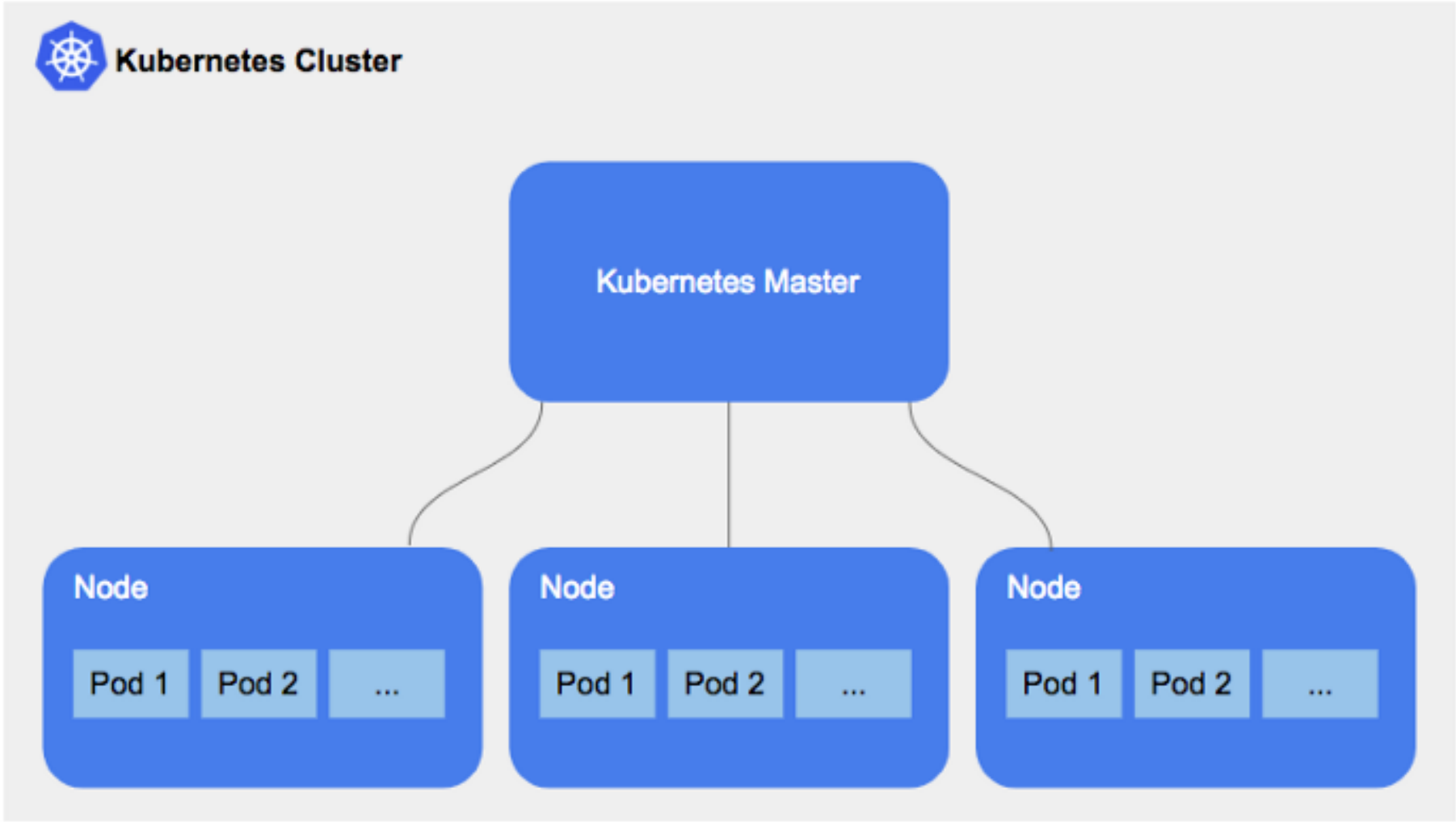
Goal

- Scale containers
- Storage orchestration
- Load balancer
- Update containers without bringing down the application
- Eliminate single points of failure – Self-healing

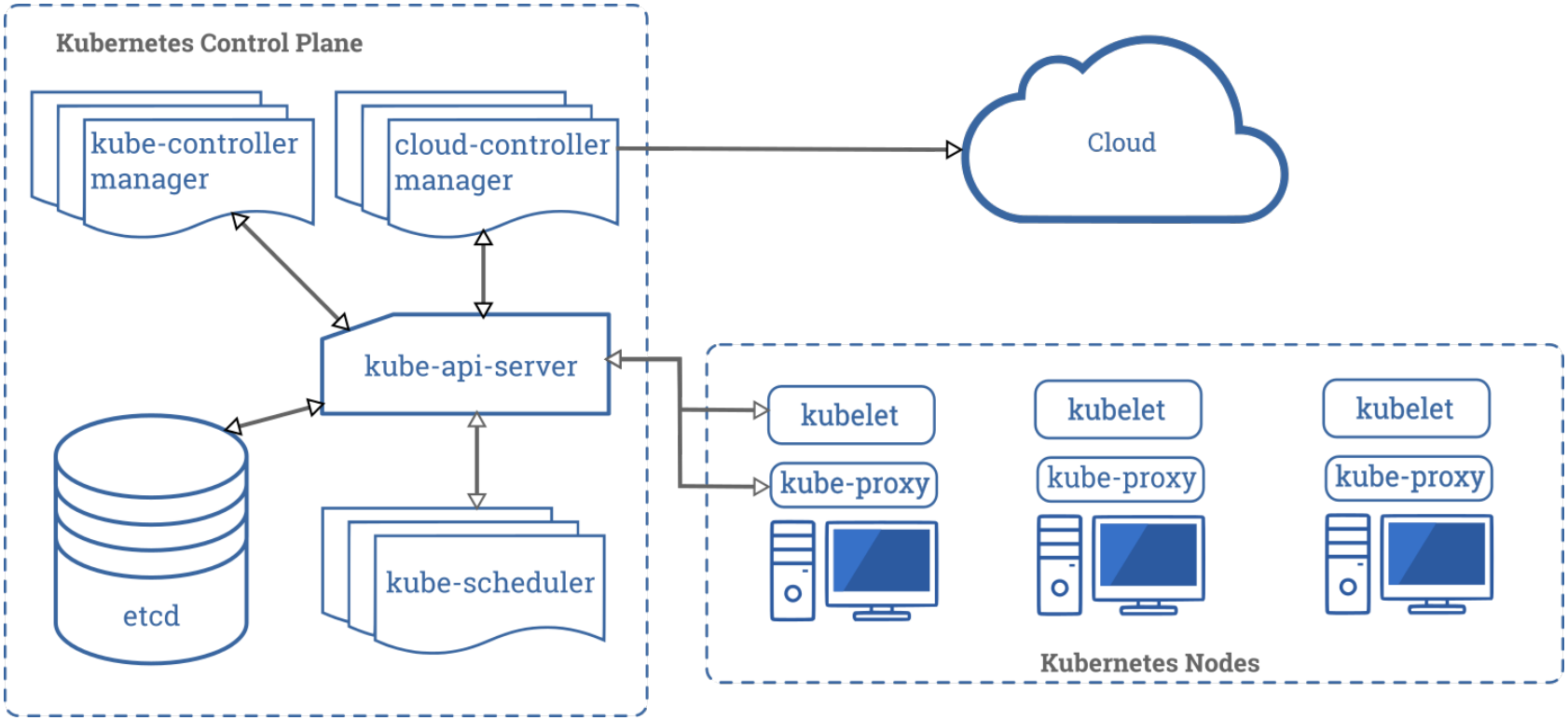
Big picture view

An orchestrator for microservice apps



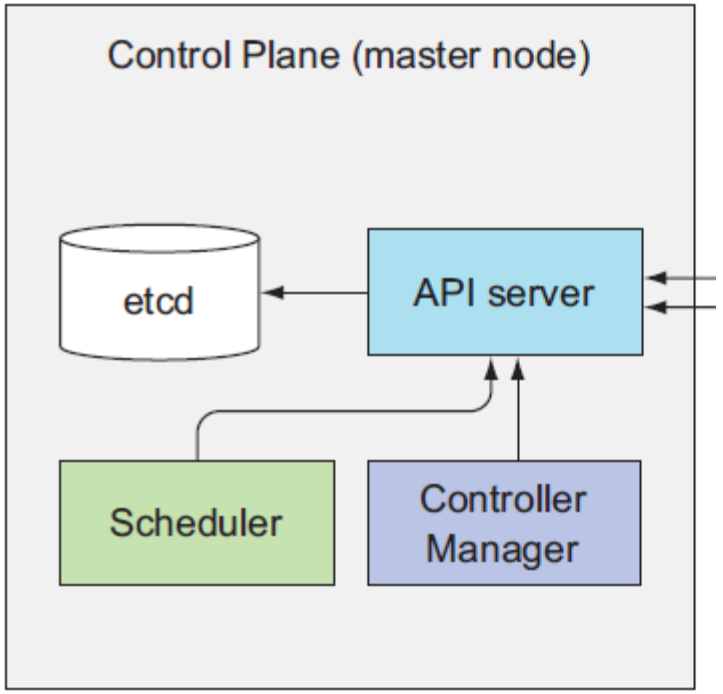

Kubernetes Cluster
Kubernetes Master
Node
Pod 1
Pod 2
...
Node
Pod 1
Pod 2
...
Node
Pod 1
Pod 2
...

Kubernetes control plane (master node)



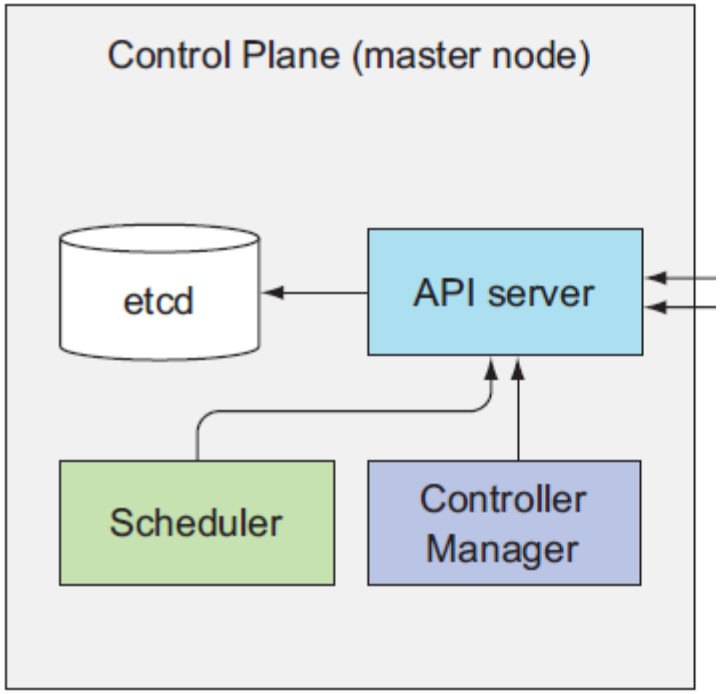
api-server

- Front-end to the control plane
- Exposes the API (REST)
- Consumes JSON



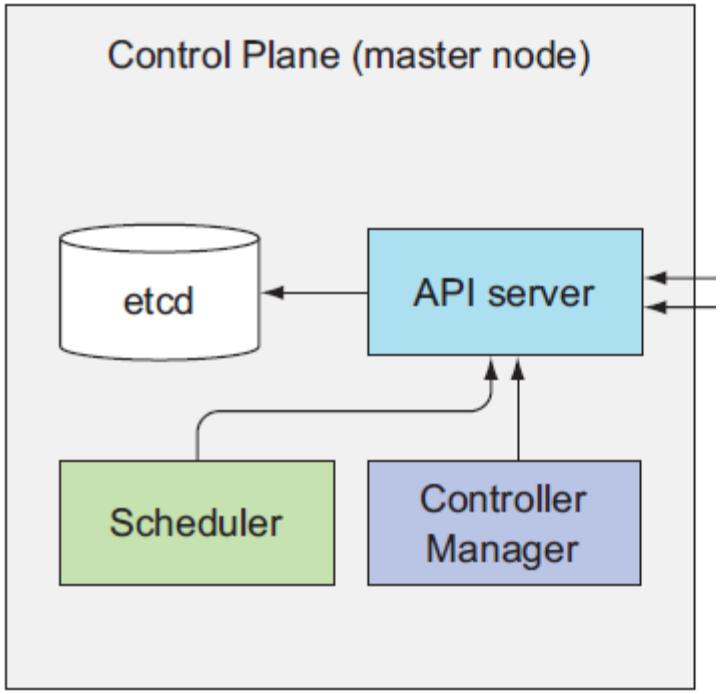
Cluster store

- Persistent storage
- Cluster state and config
- It uses etcd
  - etcd is A distributed, reliable key-value store for the most critical data of a distributed system



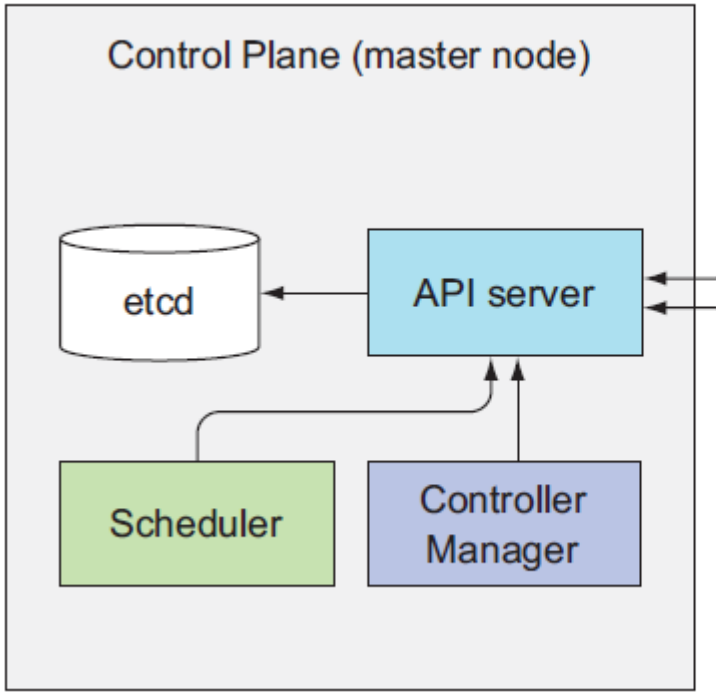
controller-manager

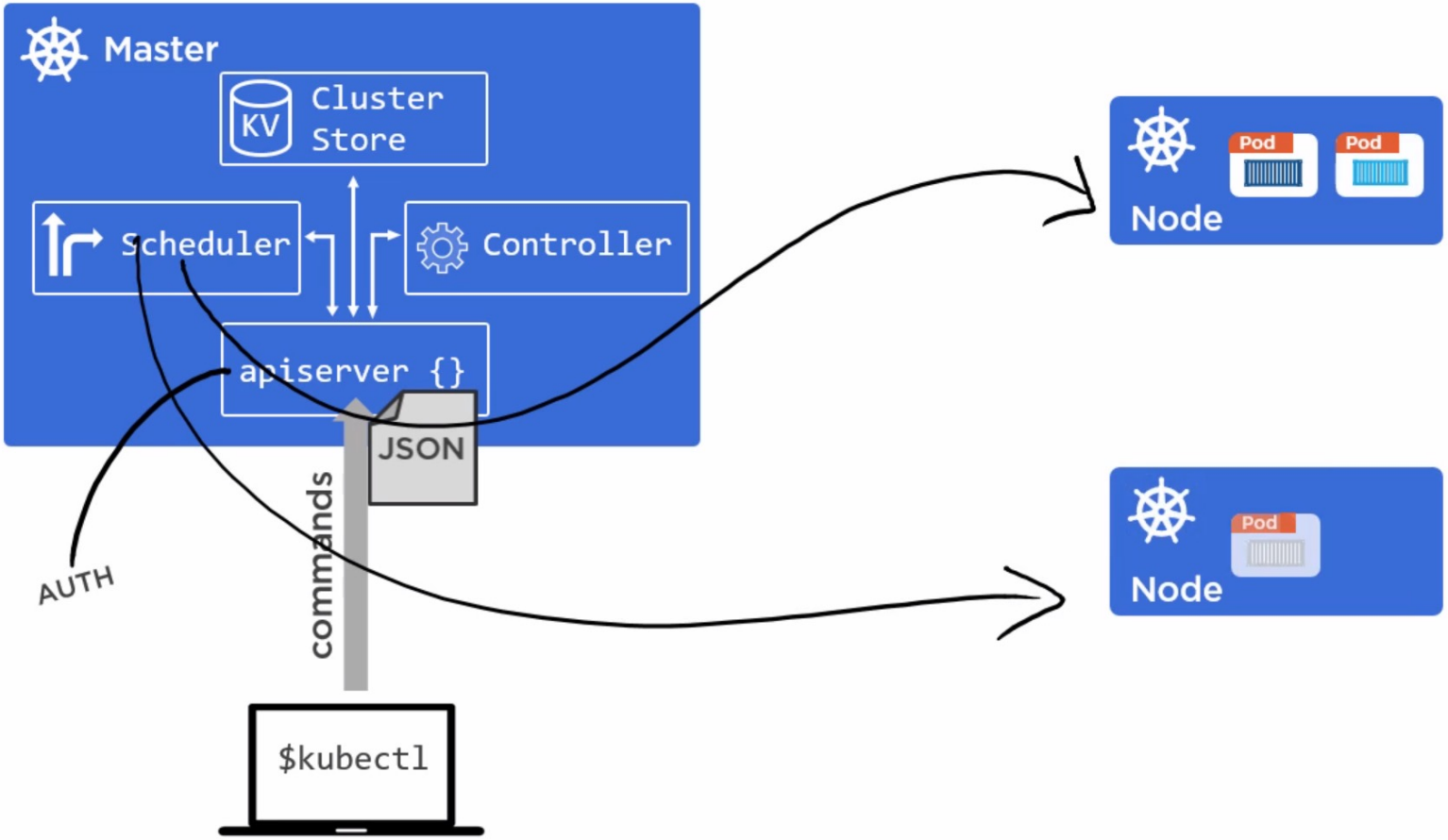
- Controller of controllers:
  - Node controller
  - Endpoints controller
  - Namespace controller
  - …
- Watches for changes
- Helps maintain desired state



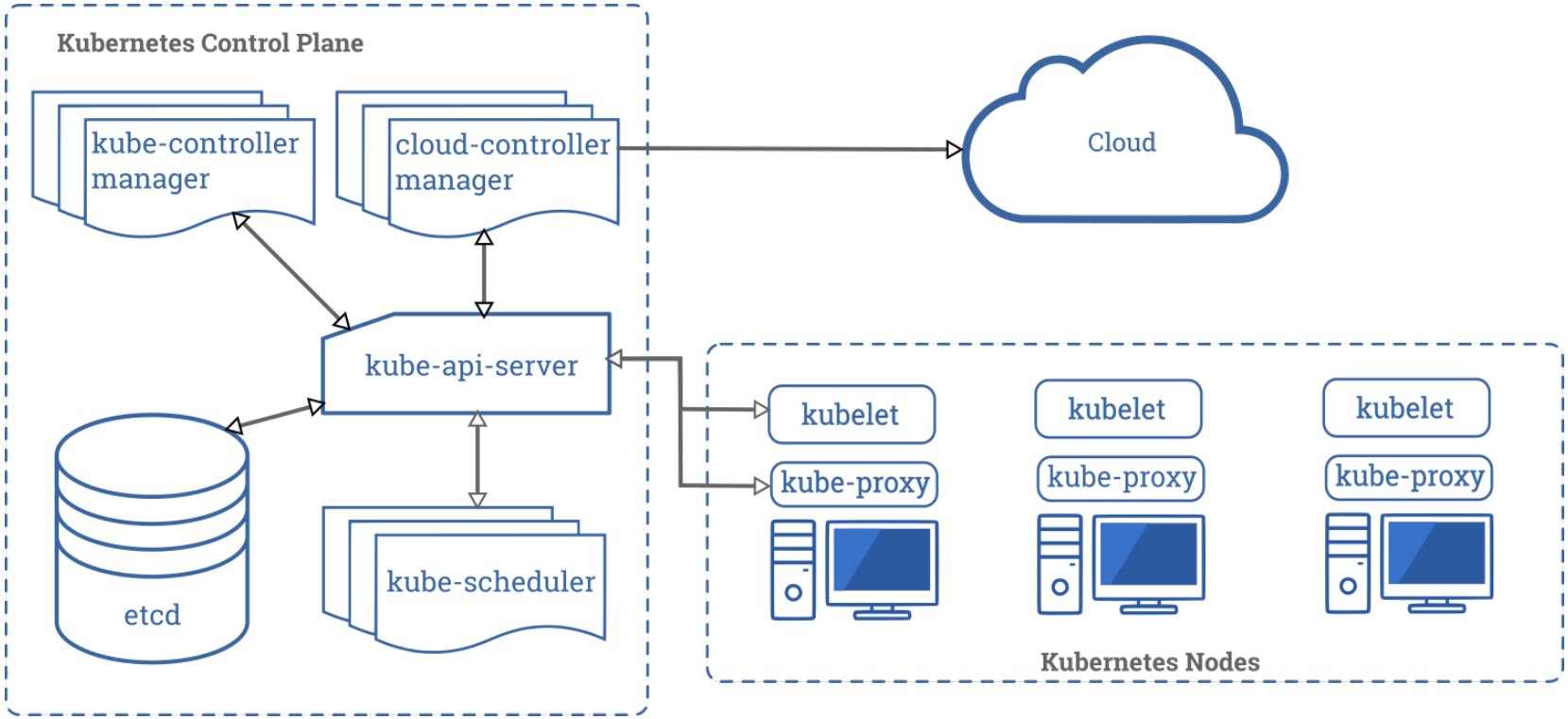
Scheduler

- Watches api-server for new pods
- Assign work to nodes



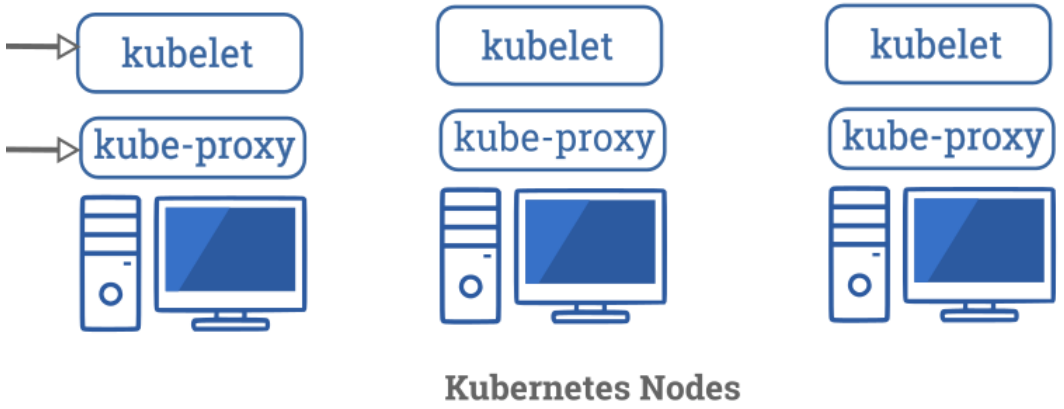

Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}
JSON
commands
AUTH
$kubectl
Pod
Pod
Node
Pod
Node

Worker node


Kubernetes Control Plane
kube-controller manager
cloud-controller manager
Cloud
kube-api-server
etcd
kube-scheduler
kubelet
kube-proxy
kubelet
kube-proxy
kubelet
kube-proxy
Kubernetes Nodes

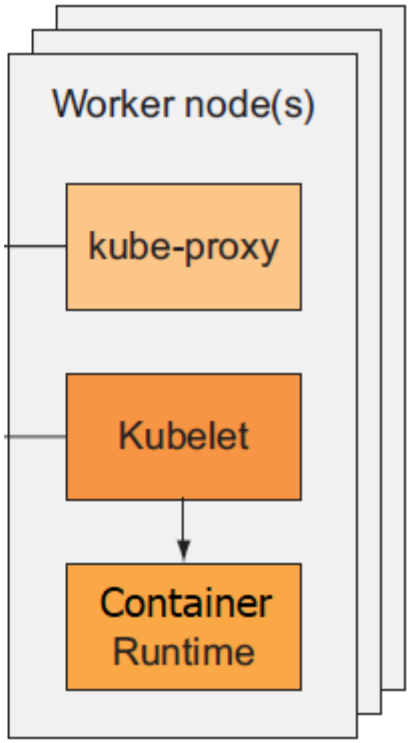
Kubelet

- The main Kubernetes agent
- Registers node with cluster
- Watches api-server
- Instantiates pods
- Reports back to master



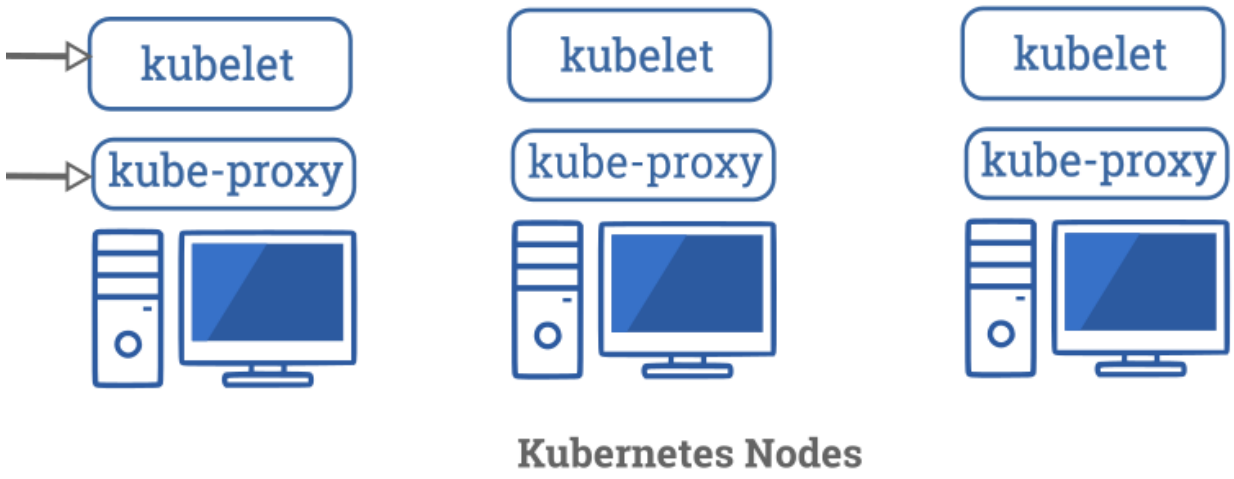
Container Runtime

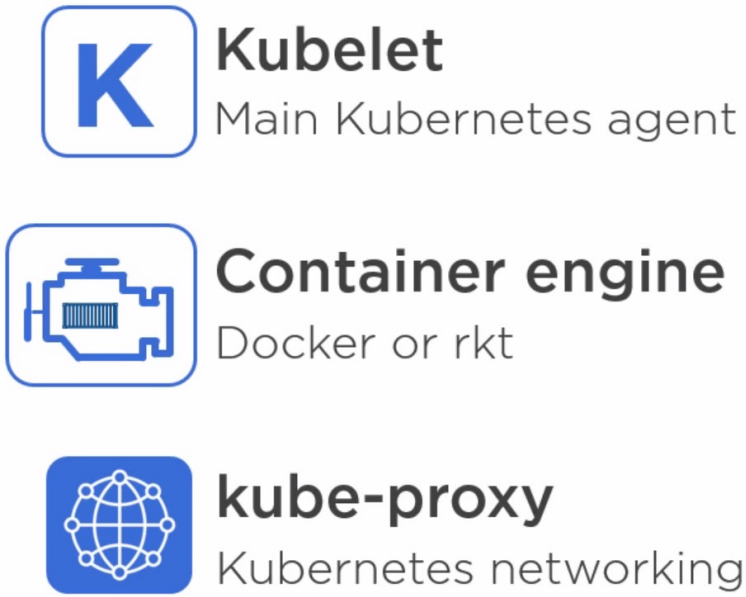
- Does container management:
  - Pulling images
  - starting/stopping containers
- Pluggable:
  - Usually Docker
  - Can be rkt

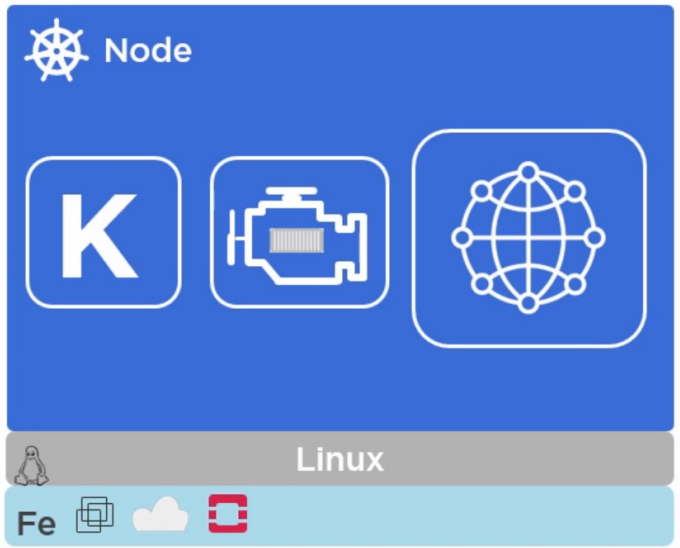


kube-proxy

- Kubernetes networking:
  - Pod IP addresses
  - Load balances of pods

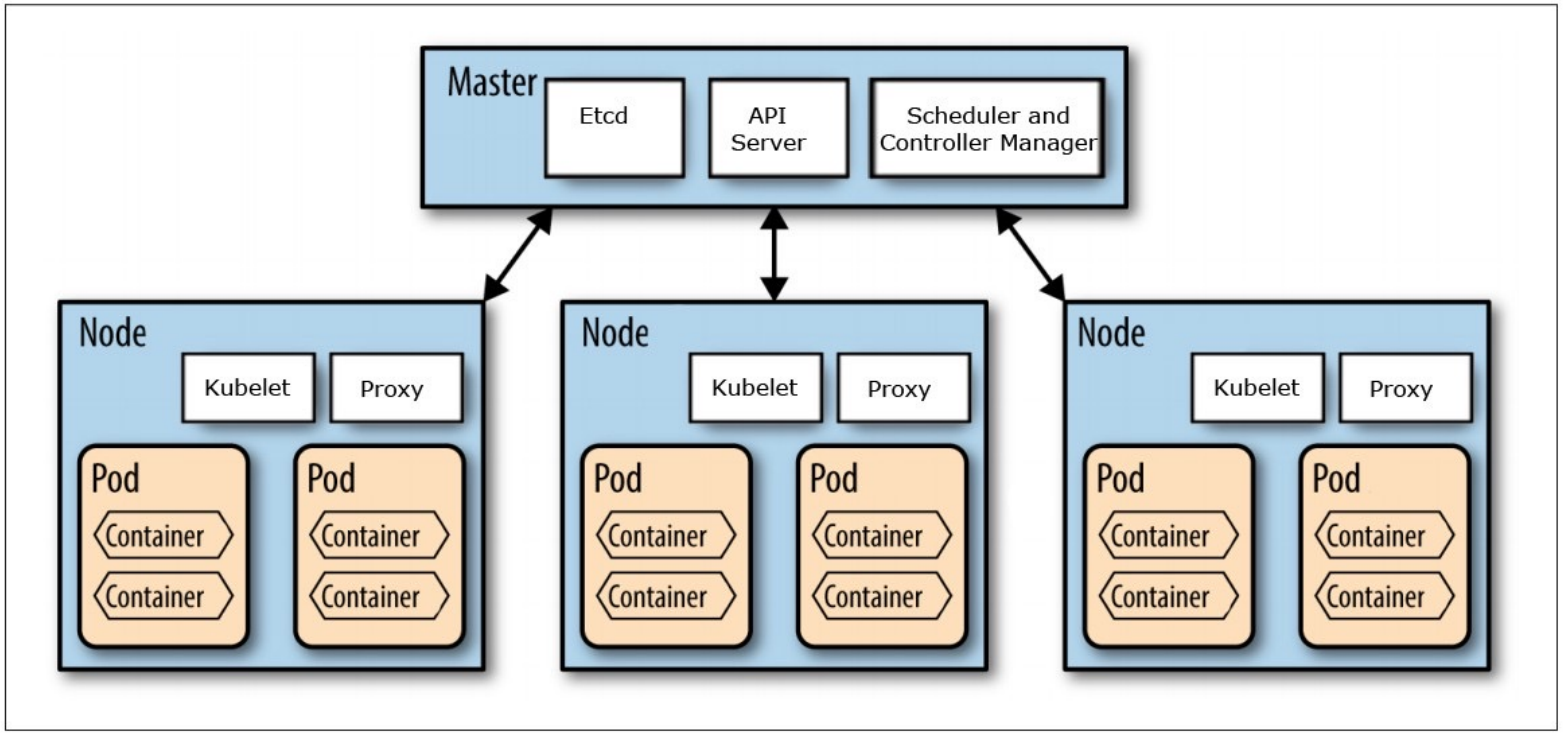



Kubelet
Main Kubernetes agent
Container engine
Docker or rkt
kube-proxy
Kubernetes networking


Node
Linux
Fe

Kubernetes pods

- Ring-fenced environment
  - Network stack
  - Kernel namespaces
  - …
- n containers
- All containers in pod share the environment


Master
Etcd
API Server
Scheduler and Controller Manager
Node
Kubelet
Proxy
Pod
Container
Container
Pod
Container
Container
Node
Kubelet
Proxy
Pod
Container
Container
Pod
Container
Container
Node
Kubelet
Proxy
Pod
Container
Container
Pod
Container
Container

Any Questions?

saman2000hoseini@gmail.com