Kubernetes is the Greek word for helmsman or captain of a ship

Kubernetes

K8s

What is Kubernetes ?

container management (orchestration) tool

developed by Google lab (& later donated to CNCF)

open source

written on Golang

also called K8s

What is Container Management Tool ?

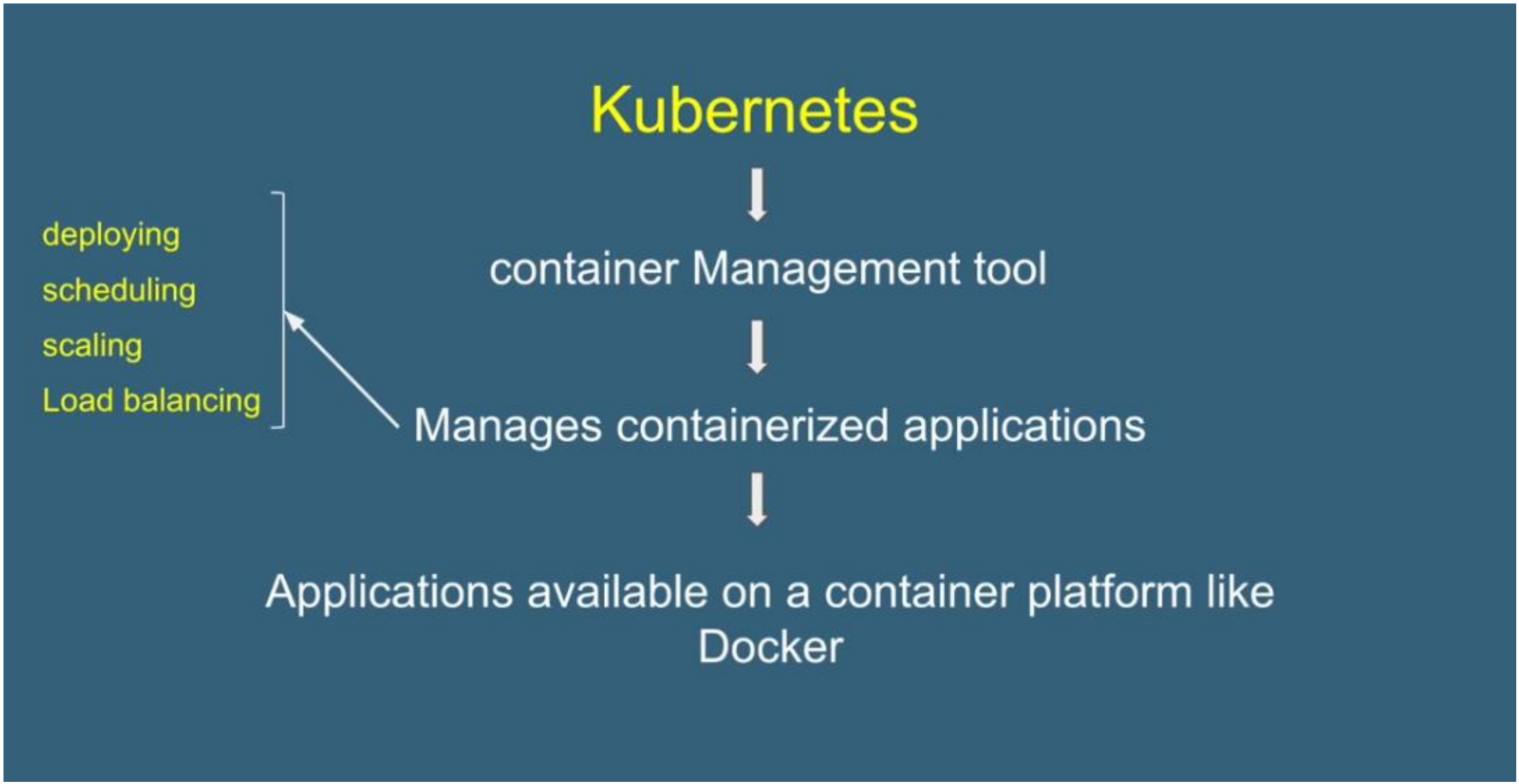
Container Management / Orchestration tool

Container Orchestration tool or engine automates deploying, scaling and managing containerized application on a group of servers

Kubernetes

Docker Swarm

Apache Mesos
Marathon


Kubernetes
container Management tool
Manages containerized applications
Applications available on a container platform like Docker
deploying
scheduling
scaling
Load balancing

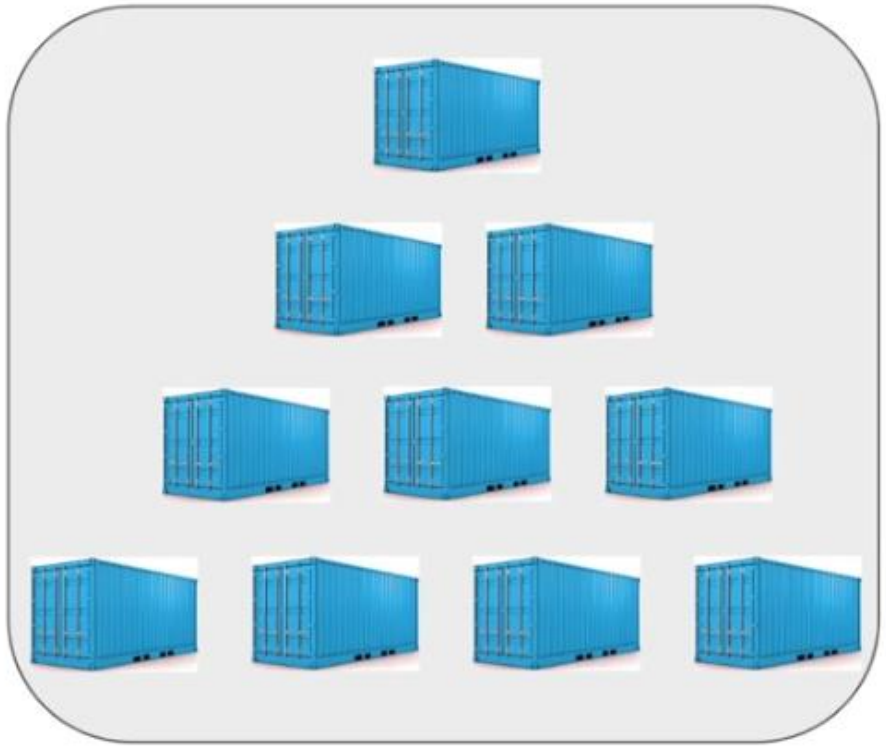
Container Management tools

Kubernetes

Docker Swarm

Apache Mesos
Marathon

- deploying
- scheduling
- scaling
- load balancing
- batch execution
- rollbacks
- monitoring



Kubernetes

Container Orchestration Engine

Manages containerized apps

deploying

scheduling

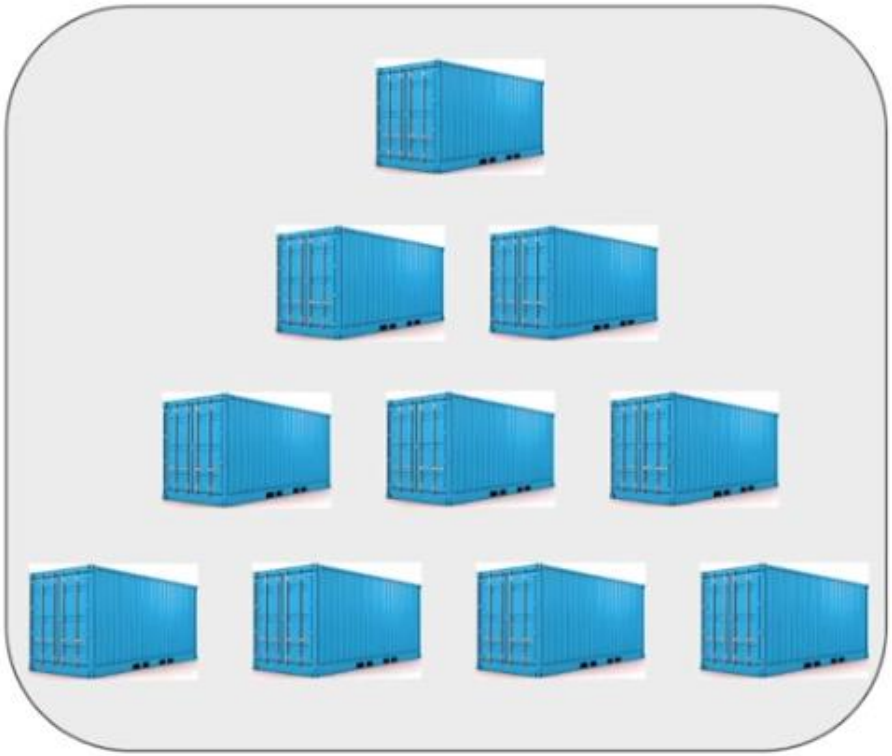
scaling

load balancing

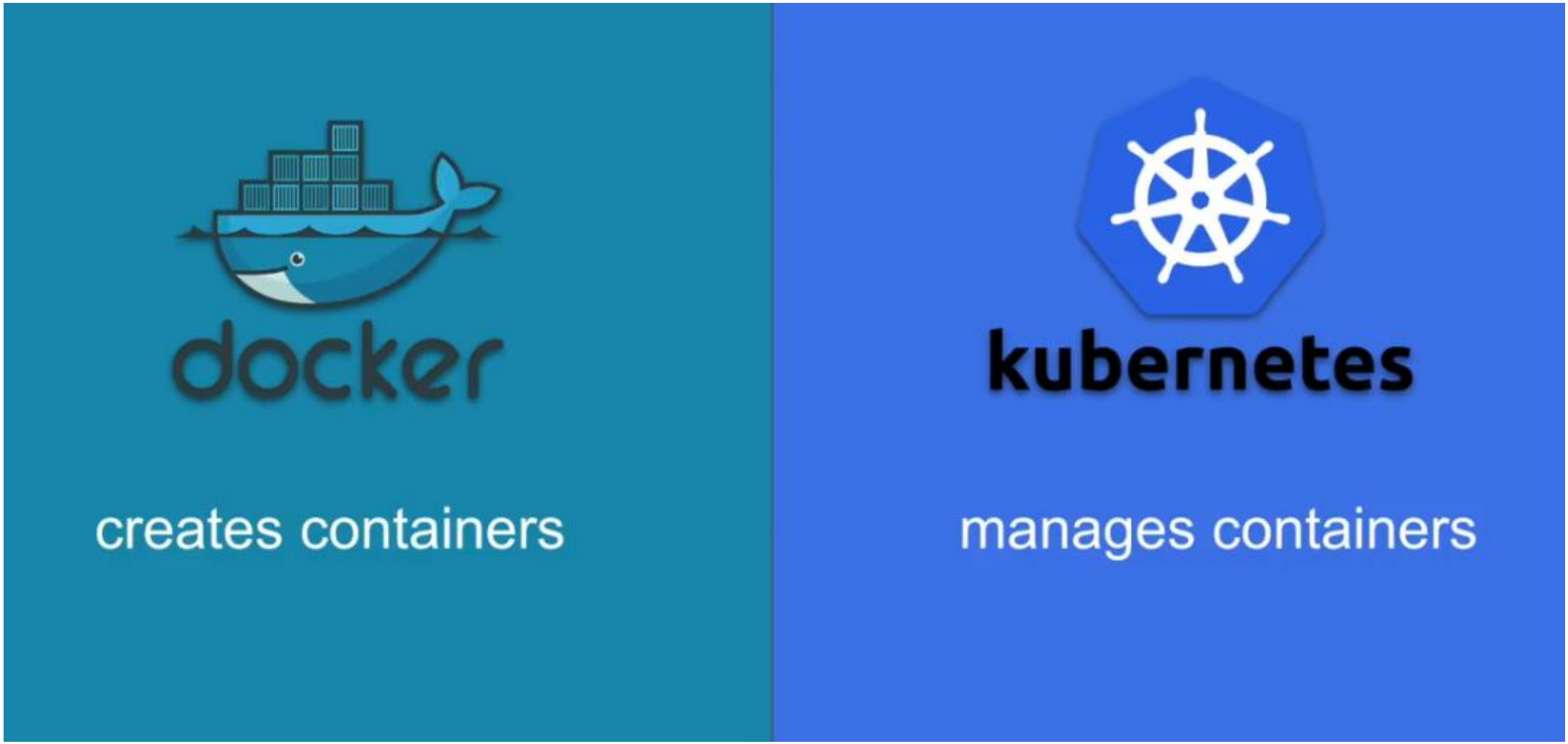
batch execution

rollbacks

monitoring



containerized apps


docker
creates containers
kubernetes
manages containers

What are the features of Kubernetes

- Automatic bin packing
- Service discovery & load balancing
- Storage orchestration
- Self-healing

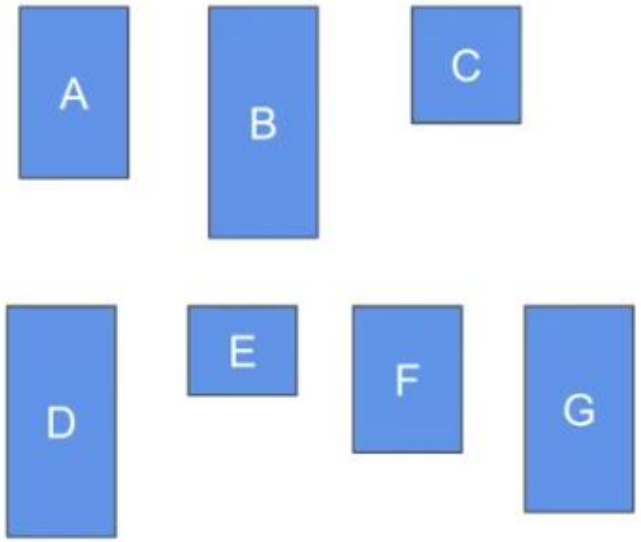
1

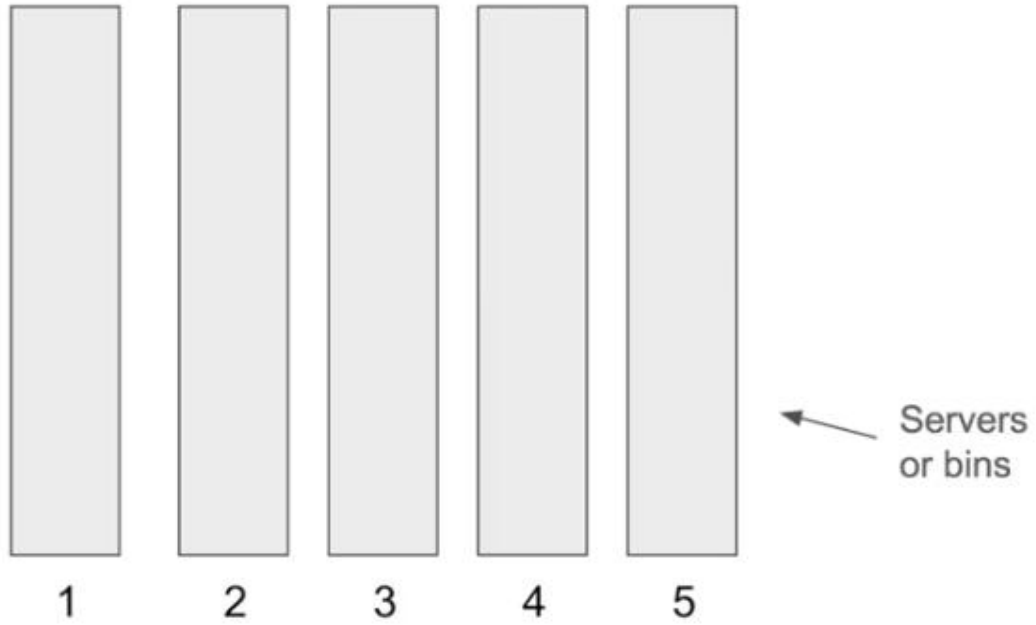
Automatic bin packing

We have 5 servers each having 10 GB of memory (RAM)

We have a list of jobs to run on these 5 servers

Every job has difference resource (memory) requirement





1

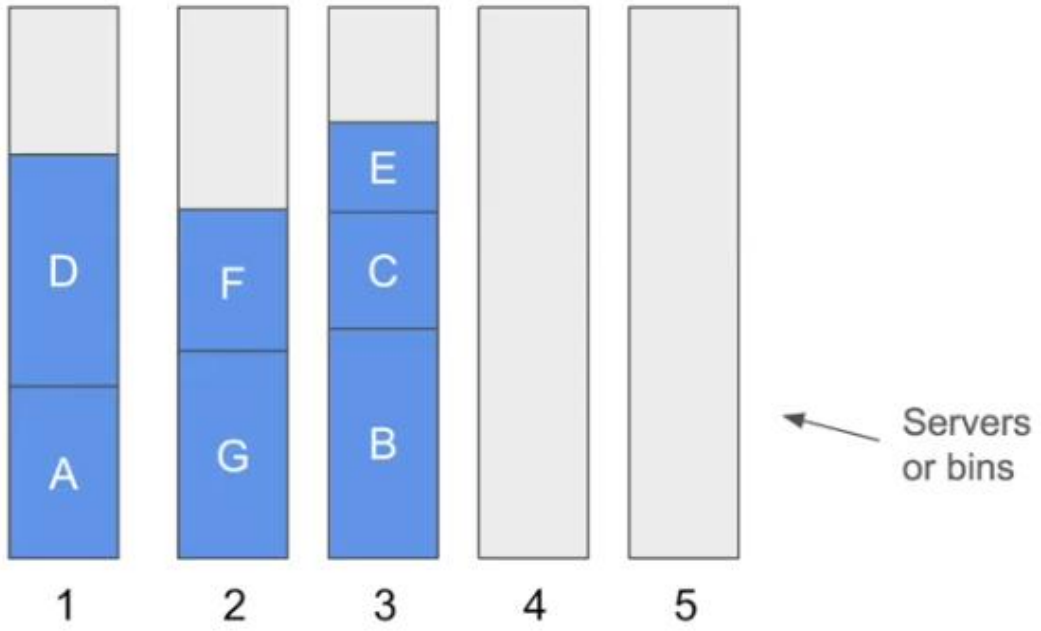
Automatic bin packing

We have 5 servers each having 10 GB of memory (RAM)

We have a list of jobs to run on these 5 servers

Every job has difference resource (memory) requirement

Kubernetes will take care of packaging these jobs (containers) in bins (servers) in the most efficient way

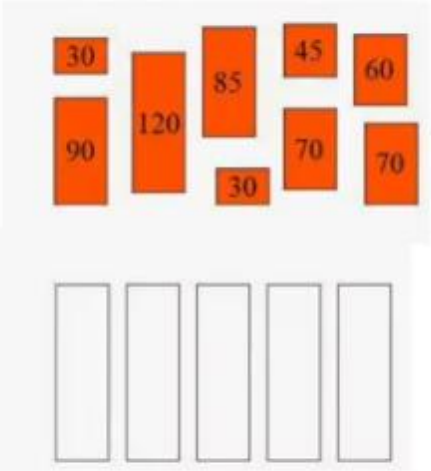


1

Automatic bin packing

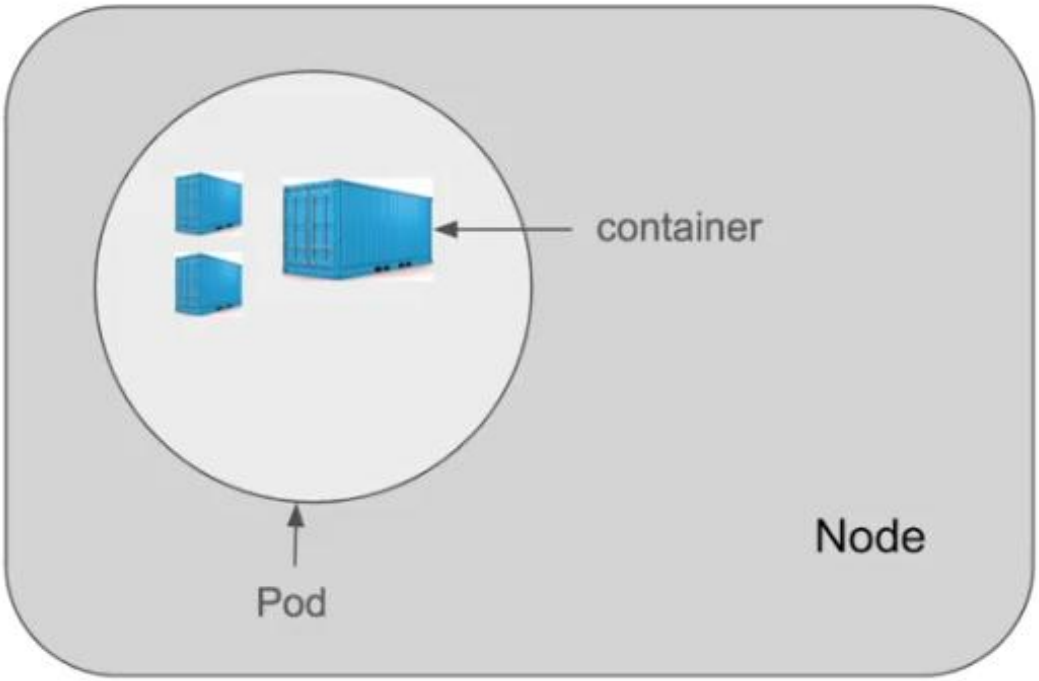
Kubernetes automatically packages your application and schedules the container based on the requirements and resources available

Pack all the pieces into as few bins as possible in the most efficient way



Automatically places containers based on their resource requirements like CPU & Memory (RAM), while not sacrificing availability
Saves resources

Pods & Nodes



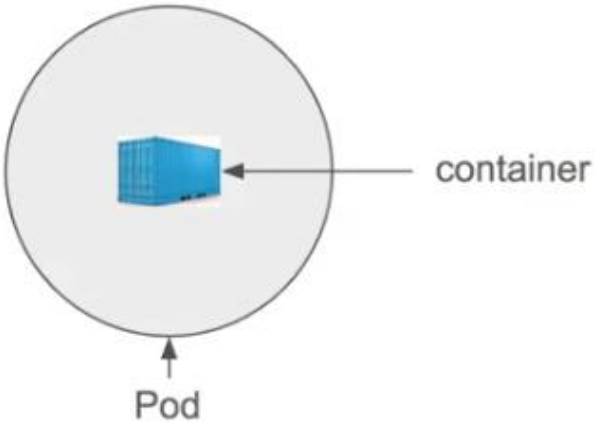
When you specify a Pod, you can optionally specify how much CPU and memory (RAM) each Container needs. When Containers have resource requests specified, the scheduler can make better decisions about which nodes to place Pods on

2

Service discovery & load balancing

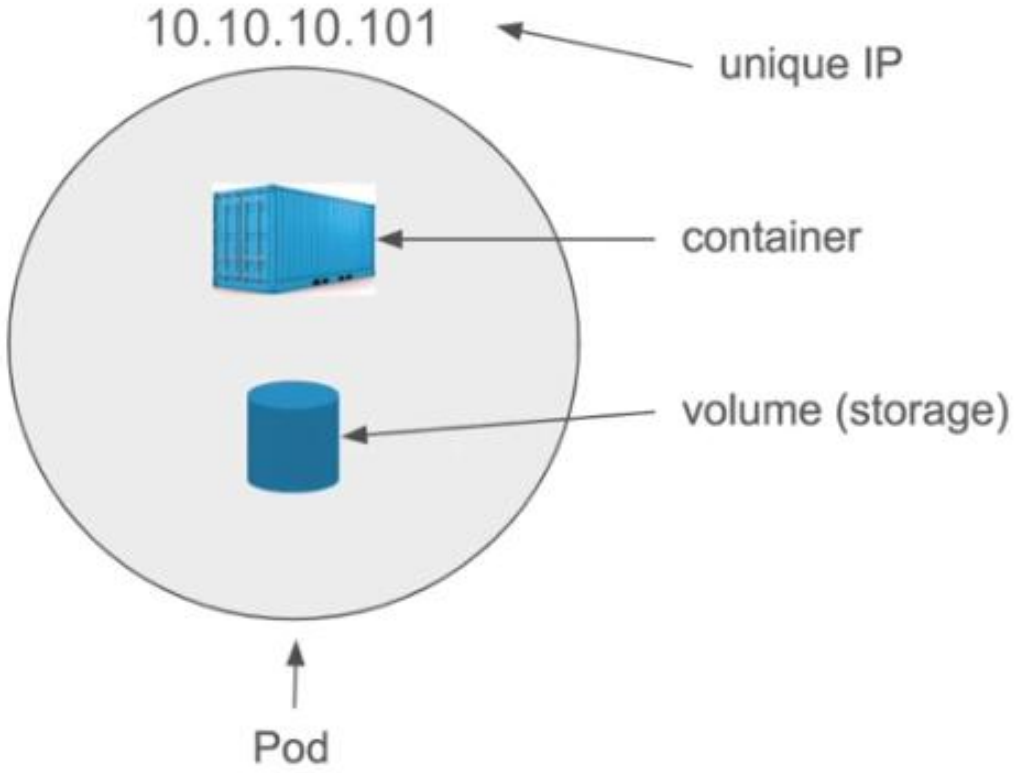
How Kubernetes organizes containers

Kubernetes doesn't run containers directly
instead it wraps one or more containers into a higher-level structure called a pod



A Pod contains

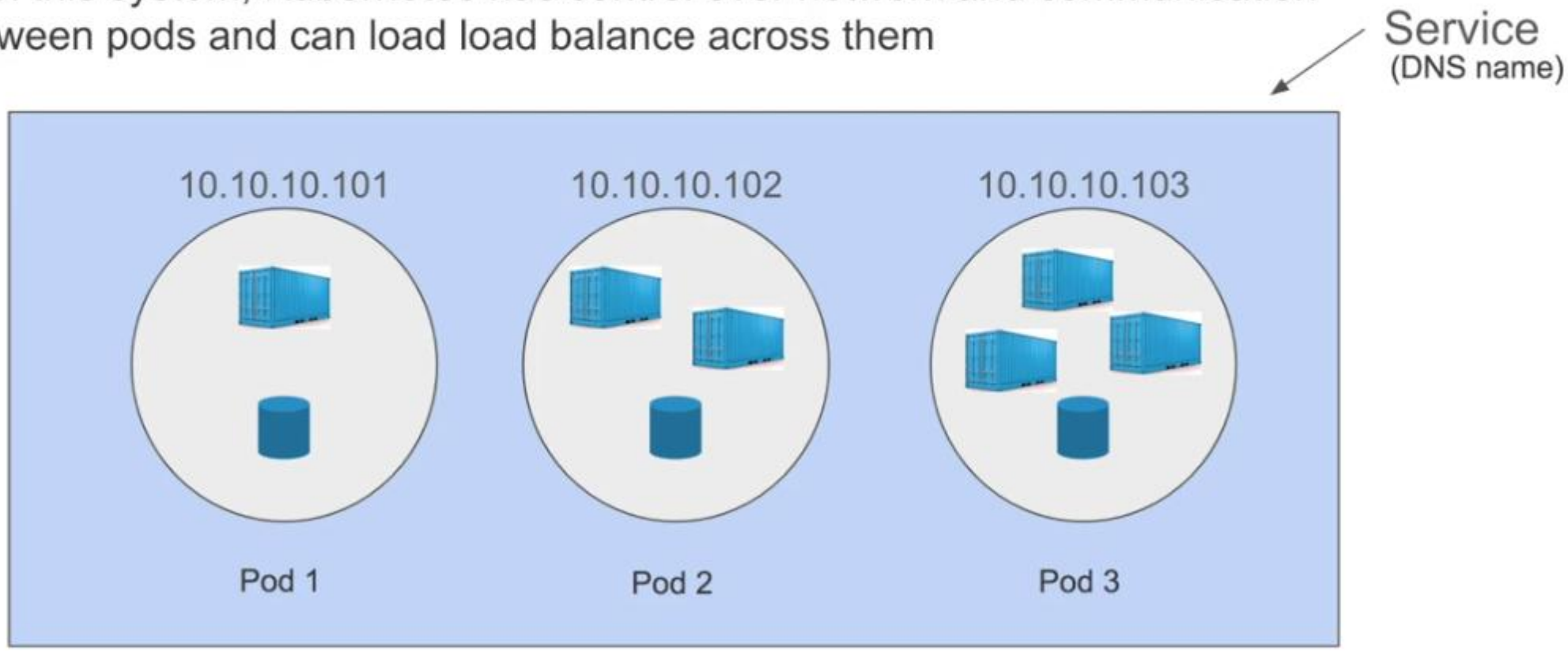
1. an application container (or, in some cases, multiple containers)
2. storage resources
3. a unique network IP



Pods that have the same set of functions are abstracted into sets, called services

Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them

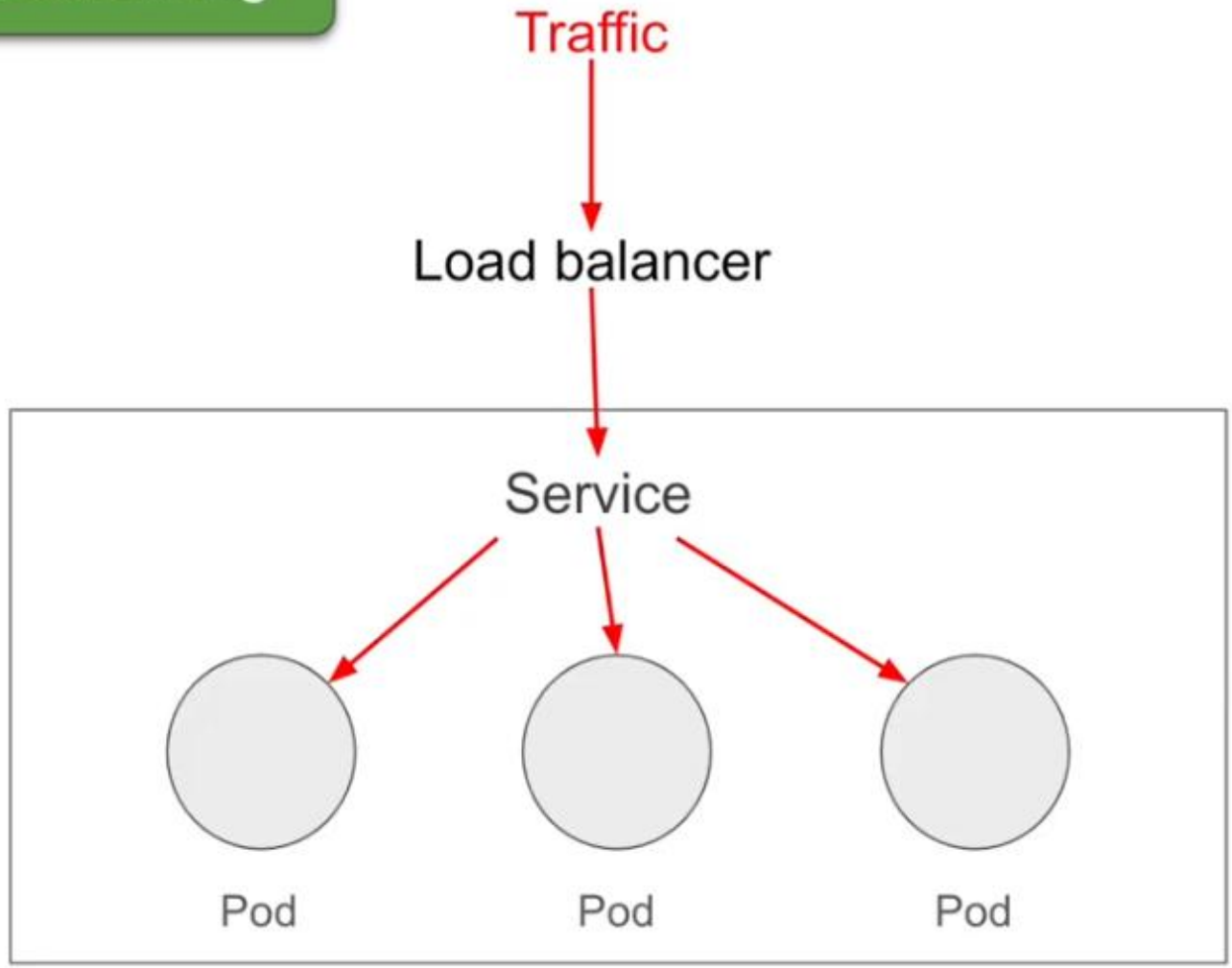
With this system, Kubernetes has control over network and communication between pods and can load load balance across them



2

Service discovery & load balancing

With this system, Kubernetes has control over network and communication between pods and can load load balance across them



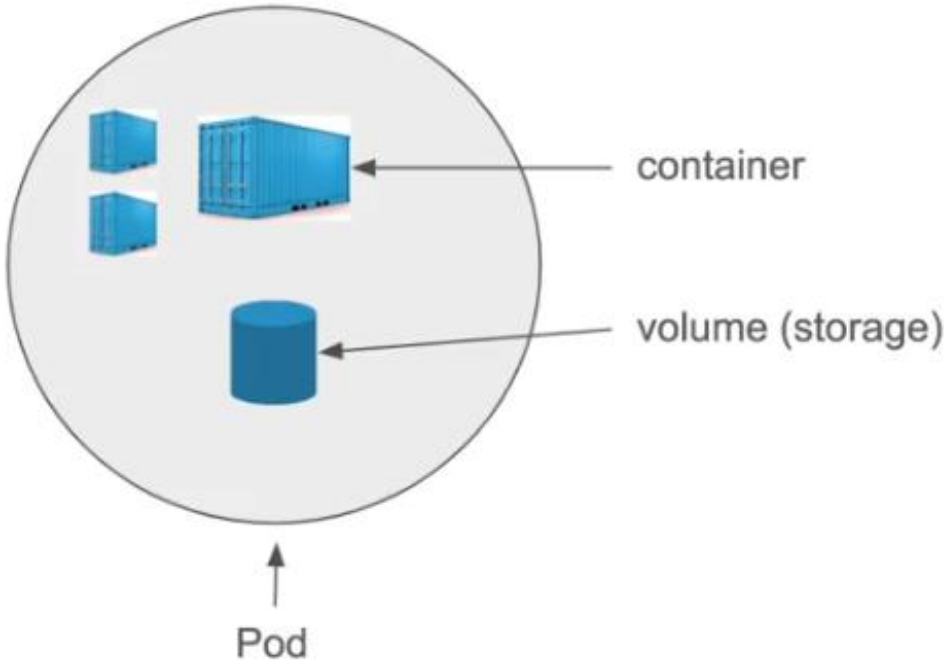
3

Storage Orchestration

Containers running inside a pod may need to store data

Pods can have a storage volumes

Usually a single volume is shared within all the containers in a pod



Kubernetes allows to mount the storage system of your choice

Local

Cloud (AWS)

Network (NFS)

4

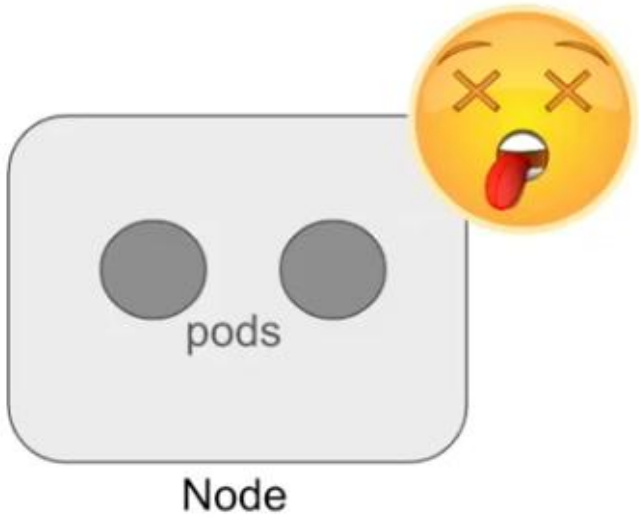
Self Healing

If container fails ⟶ restarts container

4

Self Healing

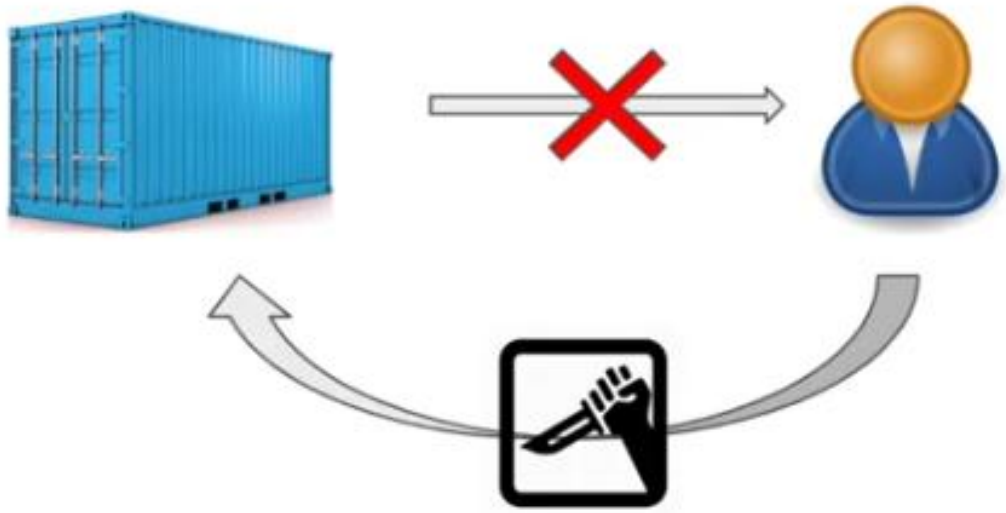
If node dies → replaces and reschedule containers on other nodes



4

Self Healing

If container does not respond to user defined health check → kills container



4

Self Healing

ReplicationController

If container fails → restarts container

If node dies → replaces and reschedule containers on other nodes

If container does not respond to user defined health check → kills container

1. Automatic bin packing
2. Service discovery & load balancing
3. Storage Orchestration
4. Self Healing

Features of Kubernetes

- Automated rollouts and rollbacks
- Secret & configuration management
- Batch execution
- Horizontal scaling

5

Automated rollouts & rollbacks

Rollout : deploy changes to the application or its configuration

Rollback : revert the changes & restore to previous state

Kubernetes ensures there is no downtime during this process

5

Automated rollouts & rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you.

Ensures there is no downtime during this process

Secret

In Kubernetes sensitive data like passwords, keys, tokens are handled using **Secrets**

Secret is a Kubernetes object

Created outside pods & containers

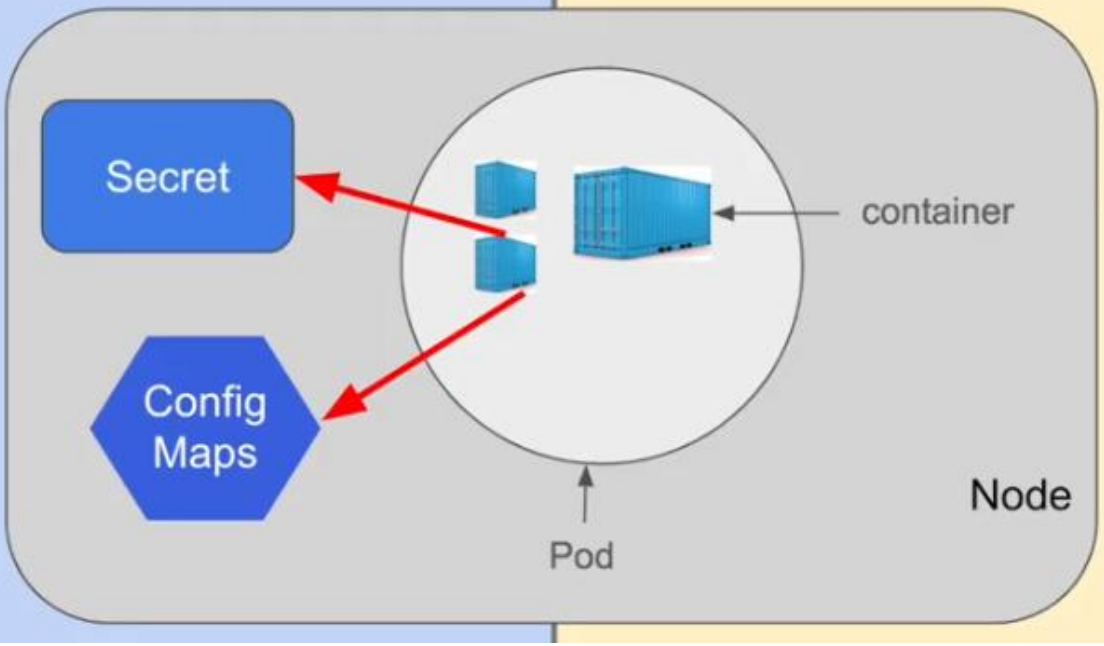
Config Maps

In Kubernetes configurations are handled using **Config Maps**

Config Map is a Kubernetes object

Created outside pods & containers

Secrets is a Kubernetes object that separates sensitive data from pods & containers



Config Map is a Kubernetes object that separates configurations from pods & containers

6

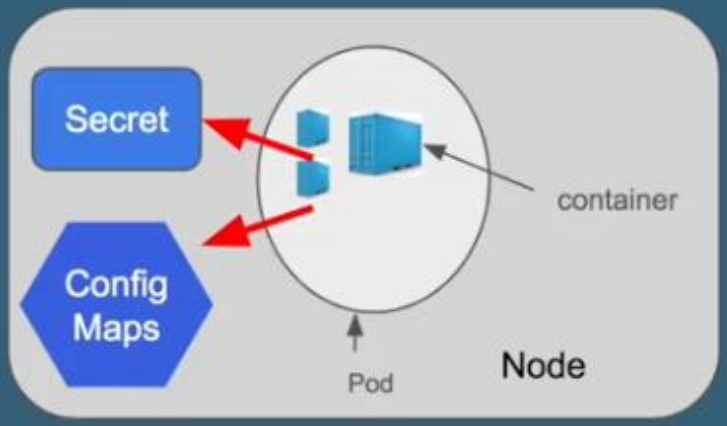
Secret & Configuration management

Kubernetes manages secrets and configuration details for an application separately from the container image,

Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

Interview FAQs

Secrets & Configurations are stored in ETCD
ETCD is a key-value datastore (database)



Interview Ques:

The max size limit for a secret is 1 MB

7

Batch execution

Batch jobs require an executable/process to be run to completion

In Kubernetes **run to completion** jobs are primarily used for batch processing

Each job creates one or more pods

7

Batch execution

Batch jobs require an executable/process to be run to completion

In Kubernetes **run to completion** jobs are primarily used for batch processing

Each job creates one or more pods

During job execution if any container or pods fails, **Job Controller** will reschedule the container, pods on another node

Can run multiple pods in parallel and can scale up if required

As the job is completed, the pods will move from running state to shut down state

7

Batch execution

Kubernetes supports batch execution, long-running jobs, and replaces failed containers.

8

Horizontal scaling

In Kubernetes, we can scale up or down the containers
- using commands
- from the dashboard (kubernetes ui)
- automatically based on CPU usage

8

Horizontal scaling

Replication Controller

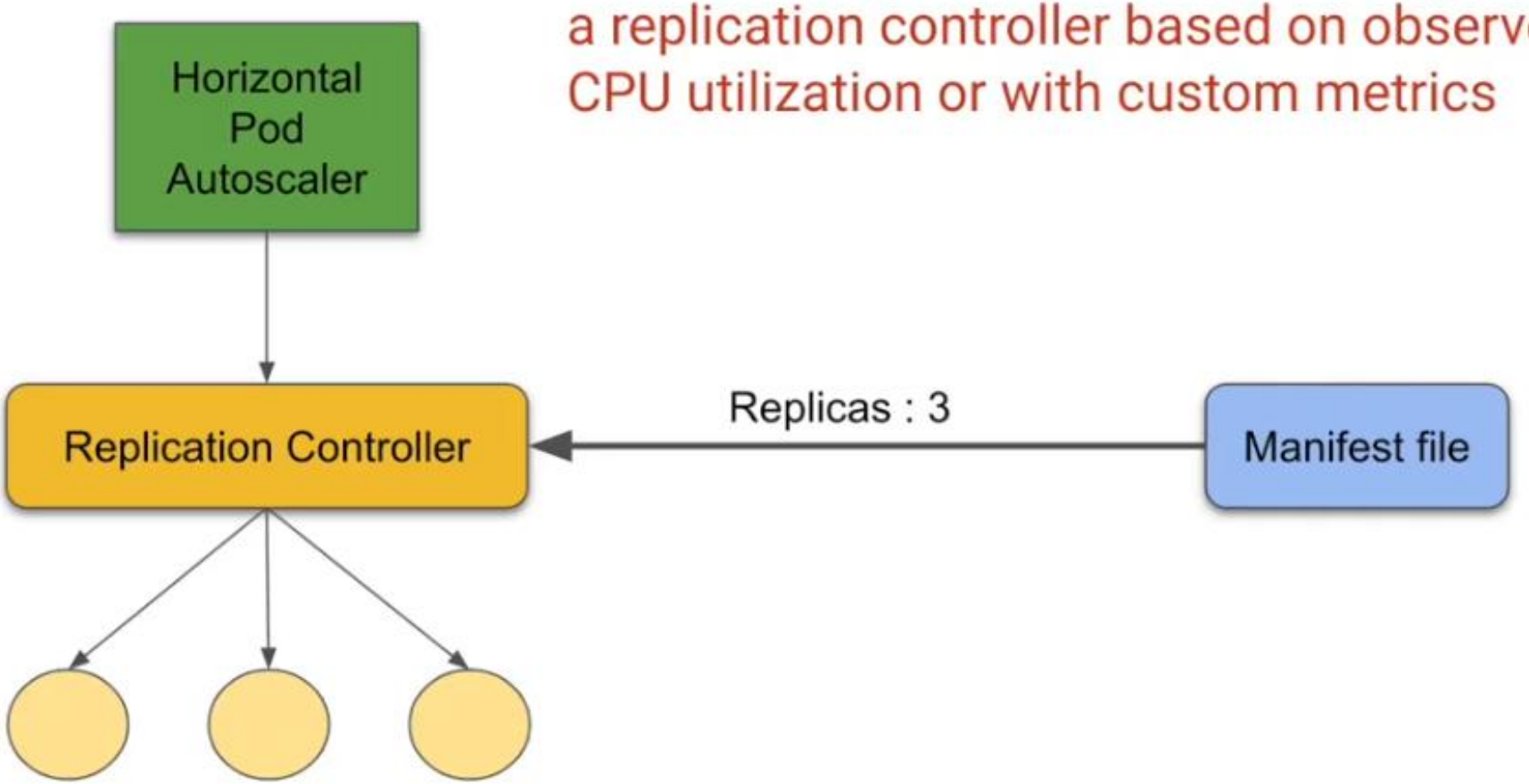
Manifest file

Horizontal Pod Autoscaler

Replication Controller is a structure that enables to create multiple pods, then make sure that that number of pods always exists.
If a pod crashes, the Replication Controller replaces it

Replication Controller gets the no of pods to run and make available at any time from the information provided in **manifest file**

automatically scales the number of pods in a replication controller based on observed CPU utilization or with custom metrics

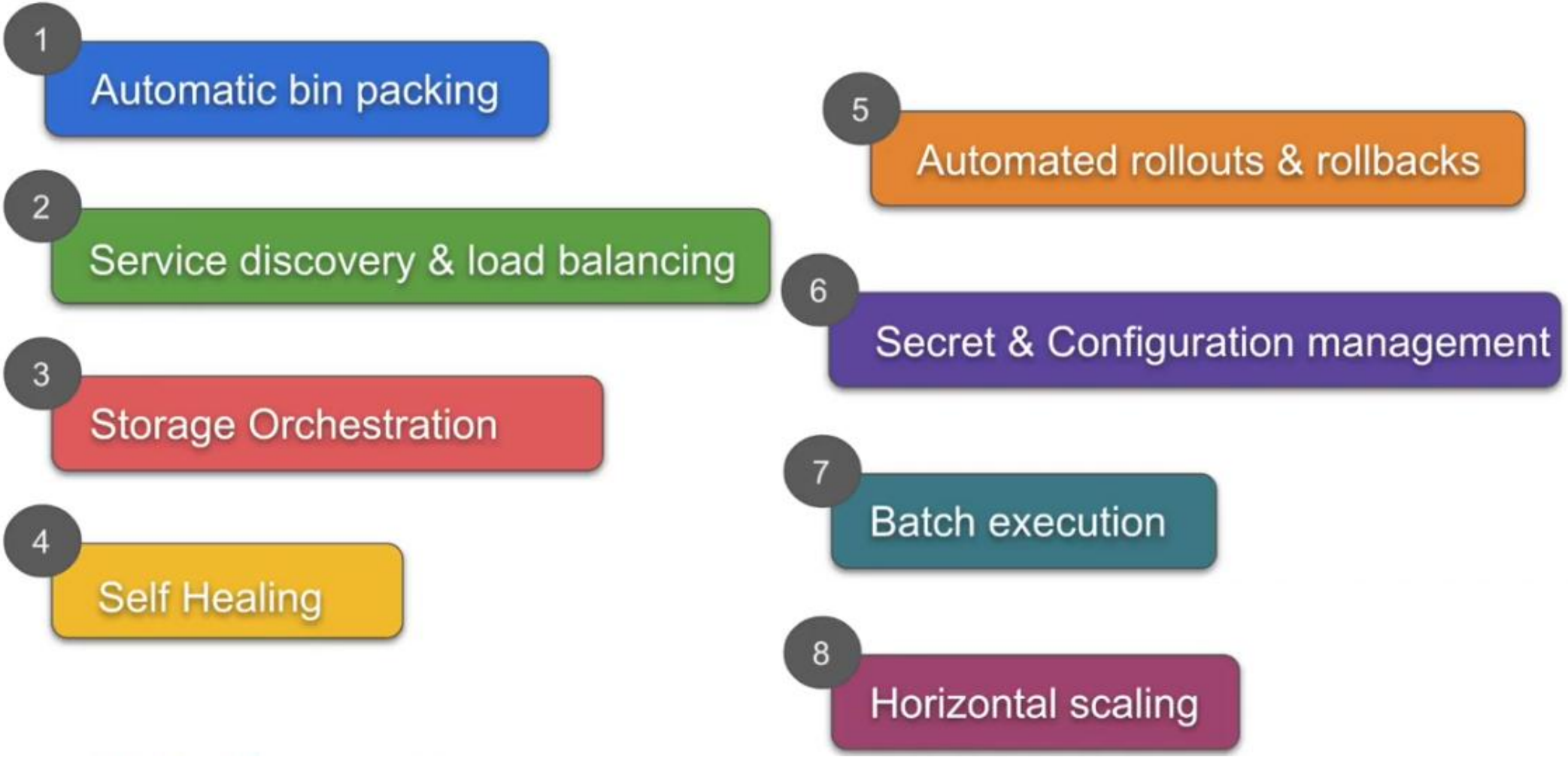


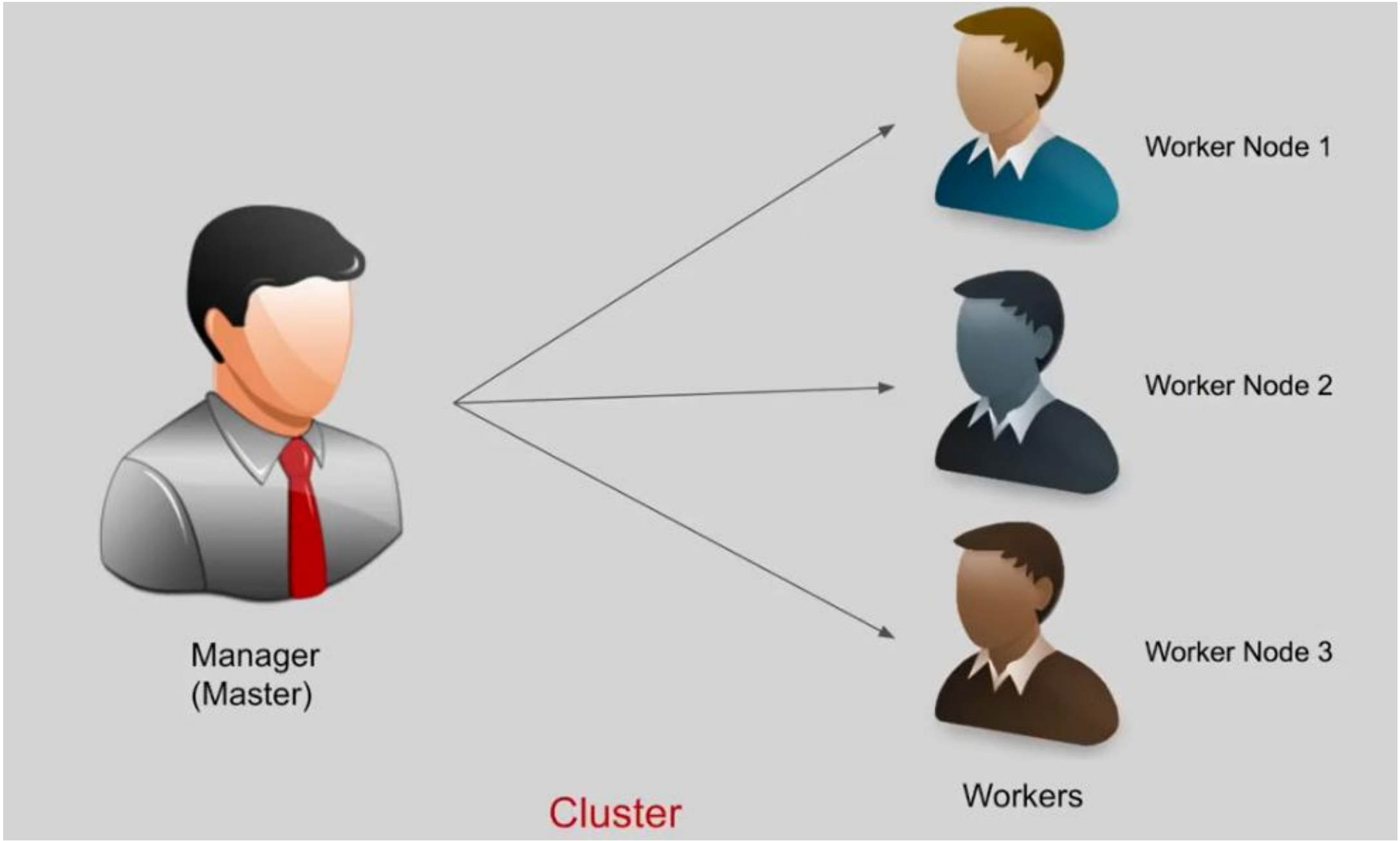
Replication Controller (**rc or rcs**) - ensures that the desired no of pods are maintained always

8

Horizontal scaling

In Kubernetes, we can scale up or down the containers
- using commands
- from the dashboard (kubernetes ui)
- automatically based on CPU usage

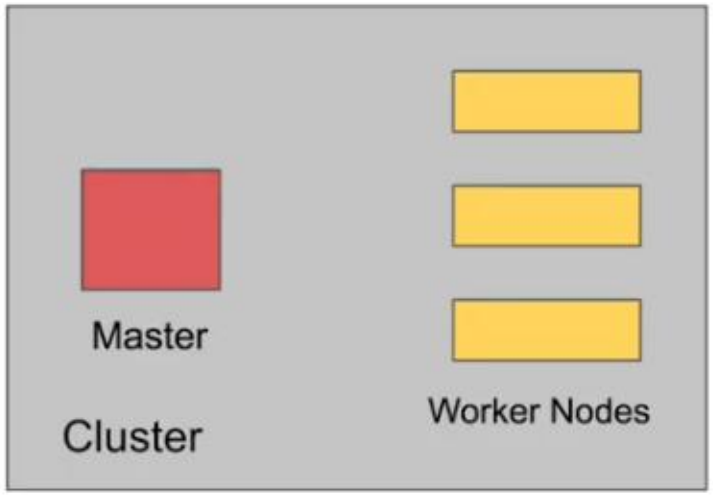

1
Automatic bin packing
2
Service discovery & load balancing
3
Storage Orchestration
4
Self Healing
5
Automated rollouts & rollbacks
6
Secret & Configuration management
7
Batch execution
8
Horizontal scaling


Worker Node 1
Worker Node 2
Worker Node 3
Manager
(Master)
Workers
Cluster

When you deploy Kubernetes, you get a **cluster**

A cluster is a set of machines, called **nodes**

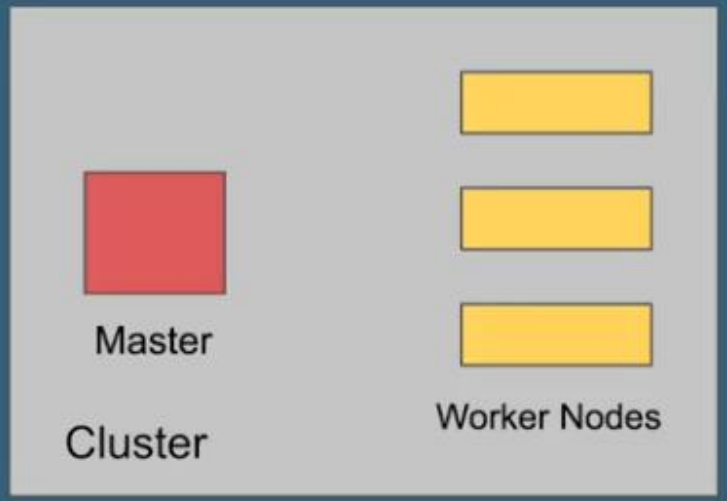
A cluster has at least one **worker node** and at least one **master node**

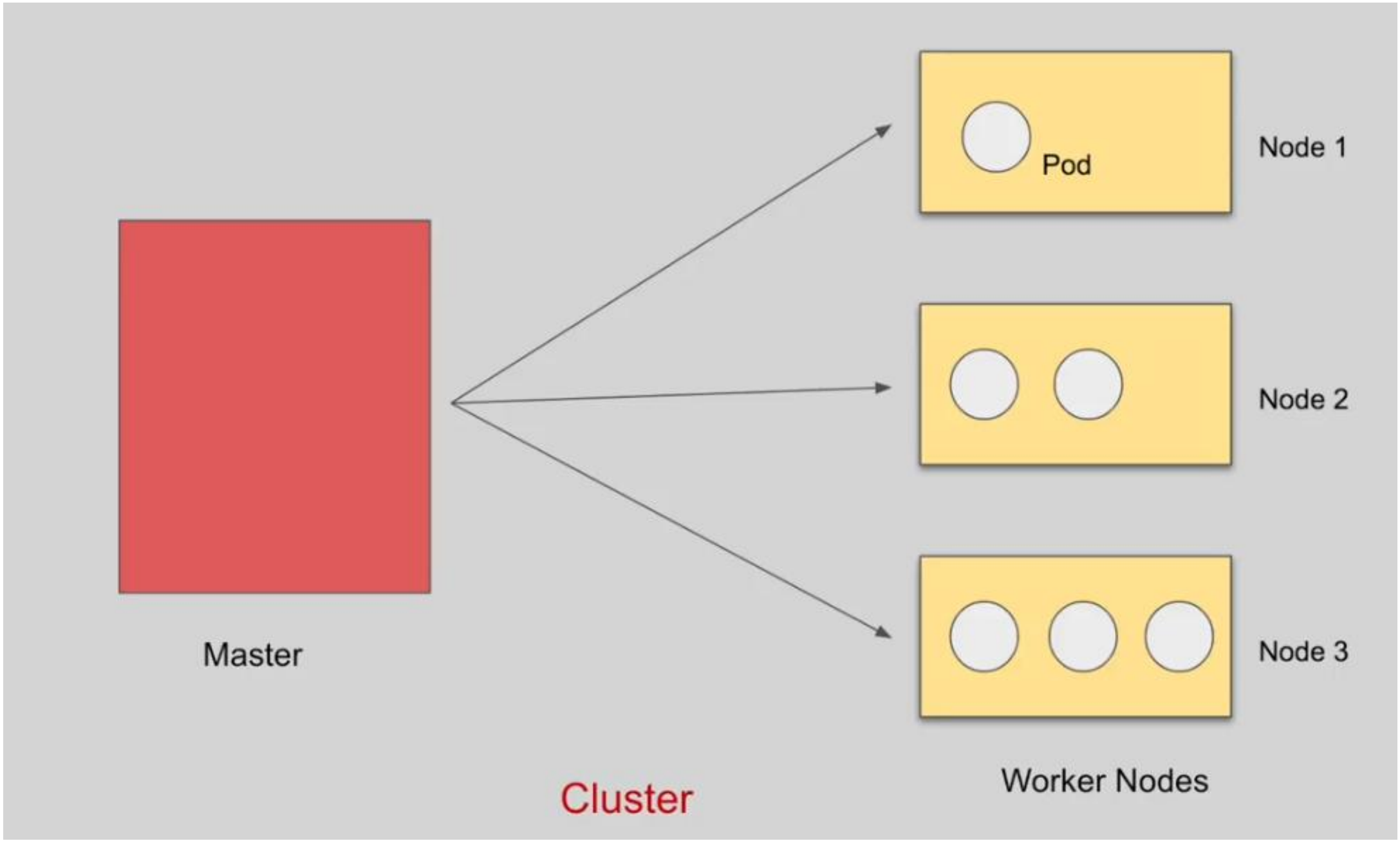


In Kubernetes, every cluster has at least one master node and at least one worker node

There can be more than one master nodes in a cluster to provide a cluster with failover and high availability

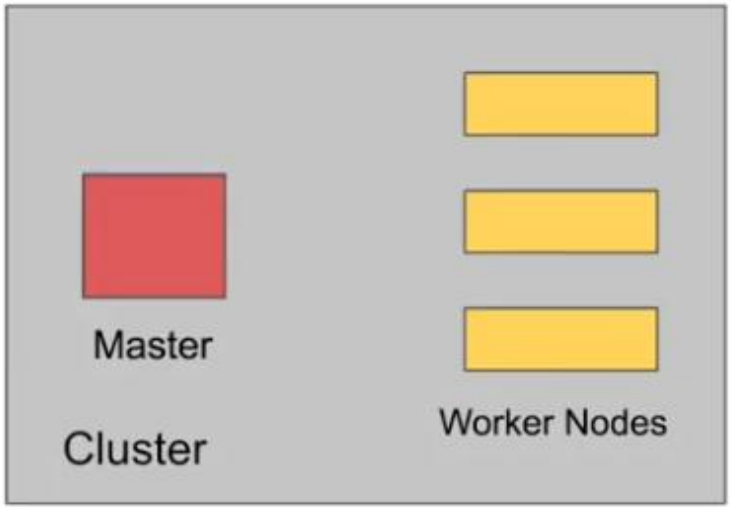
There can be multiple clusters in Kubernetes architecture



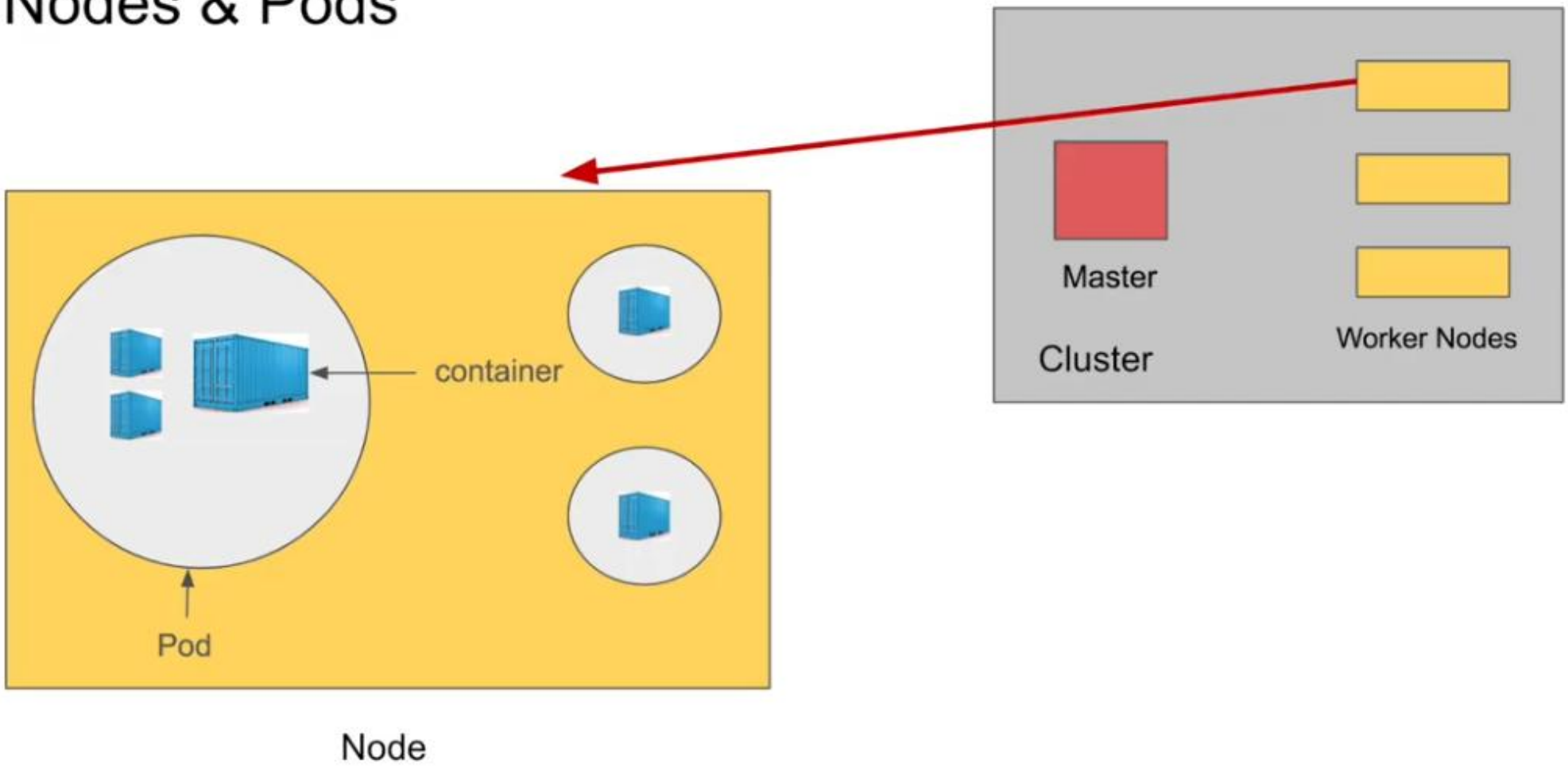

Pod
Node 1
Node 2
Node 3
Master
Cluster
Worker Nodes

Nodes & Pods

A node can be
- Physical machine
- Virtual machine
- VM on cloud

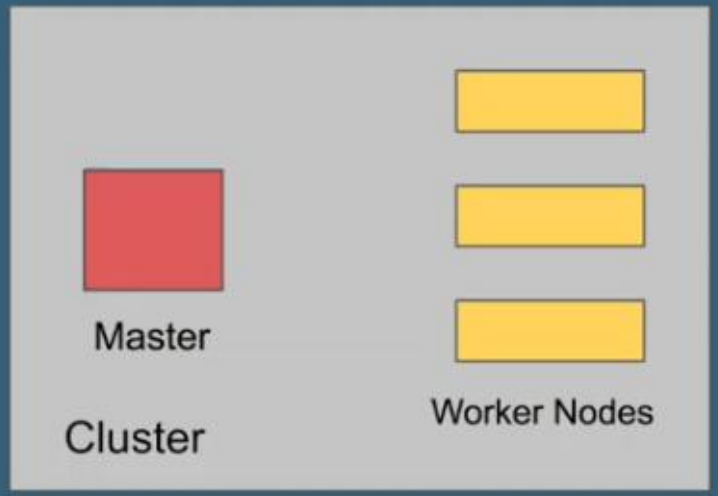


Nodes & Pods



The worker node(s) host the pods that are the components of the application

The master node(s) manages the worker nodes and the pods in the cluster

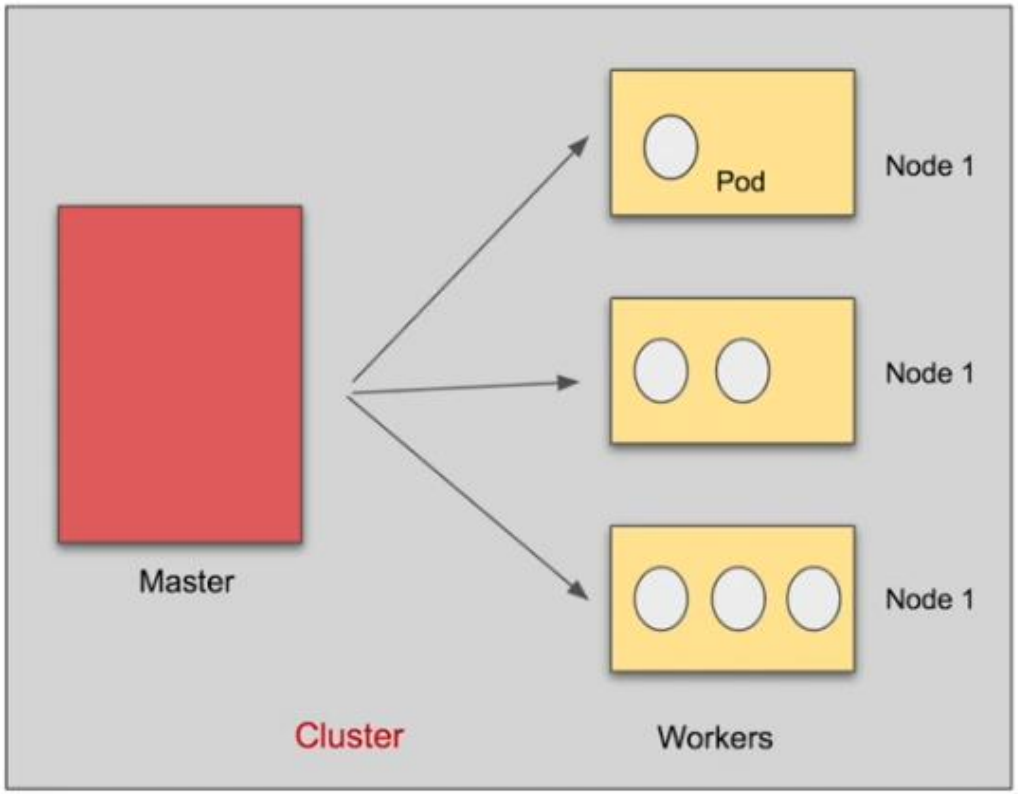


Cluster

Nodes

Pods

Master



Master node

responsible for managing the cluster

Monitors nodes & pods in a cluster

When a node fails, moves the workload of the failed node to another worker node

Master

4 Components of Master node

1. **API Server** - for all communications (JSON over HTTP API)
2. **Scheduler** - schedules pods on nodes
3. **Controller Manager** - runs controllers
4. **Etcd** - open source, distributed key-value database from CoreOS

API Server

APIs allow applications to communicate with one another

It is the front-end for the Kubernetes control plane

Exposes API for almost every operation

The users, management devices, and command line interfaces all talk to the API server to interact with the Kubernetes cluster

API Server

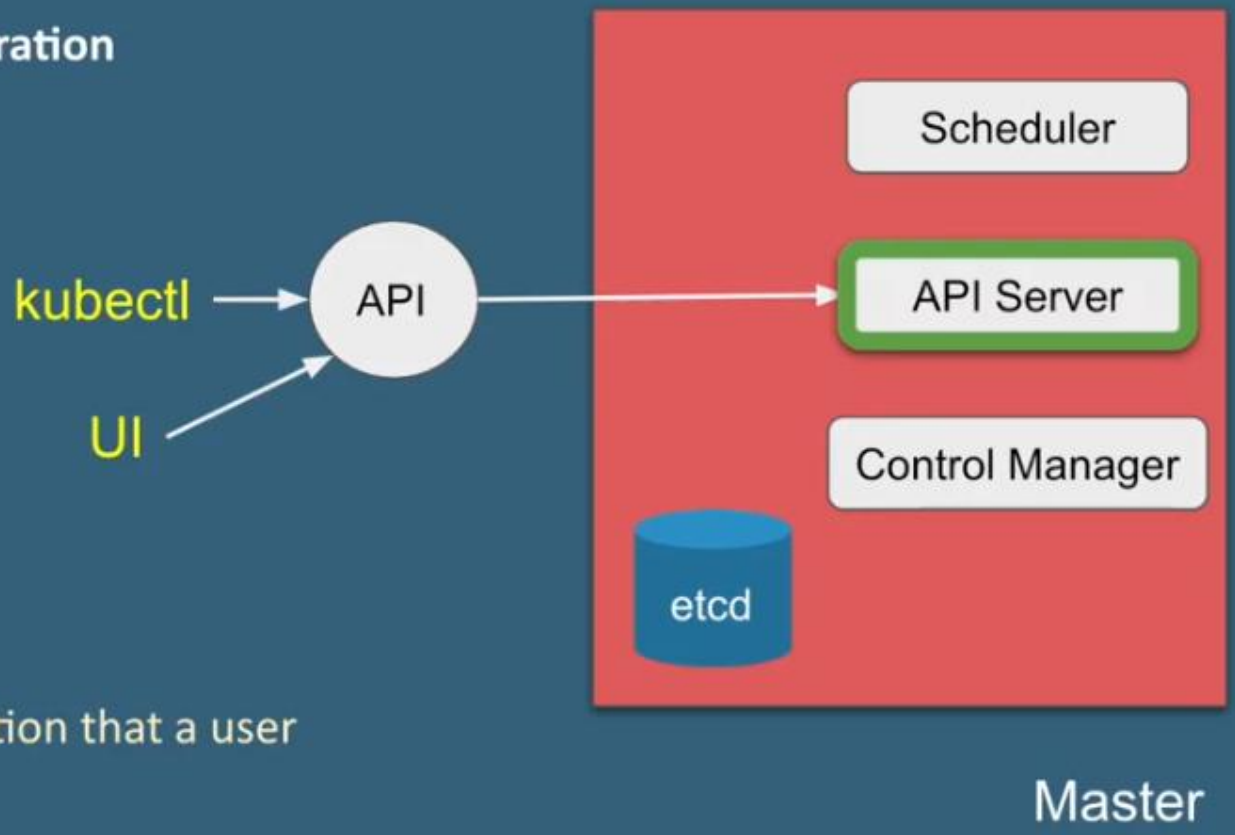
Exposes API for almost every operation

Users interact with the API using a tool called **kubectl**

Kubectl is the command line utility to interact with Kubernetes API

Kubectl is a Go Language binary

? Talks to the API to perform any action that a user issues from command line

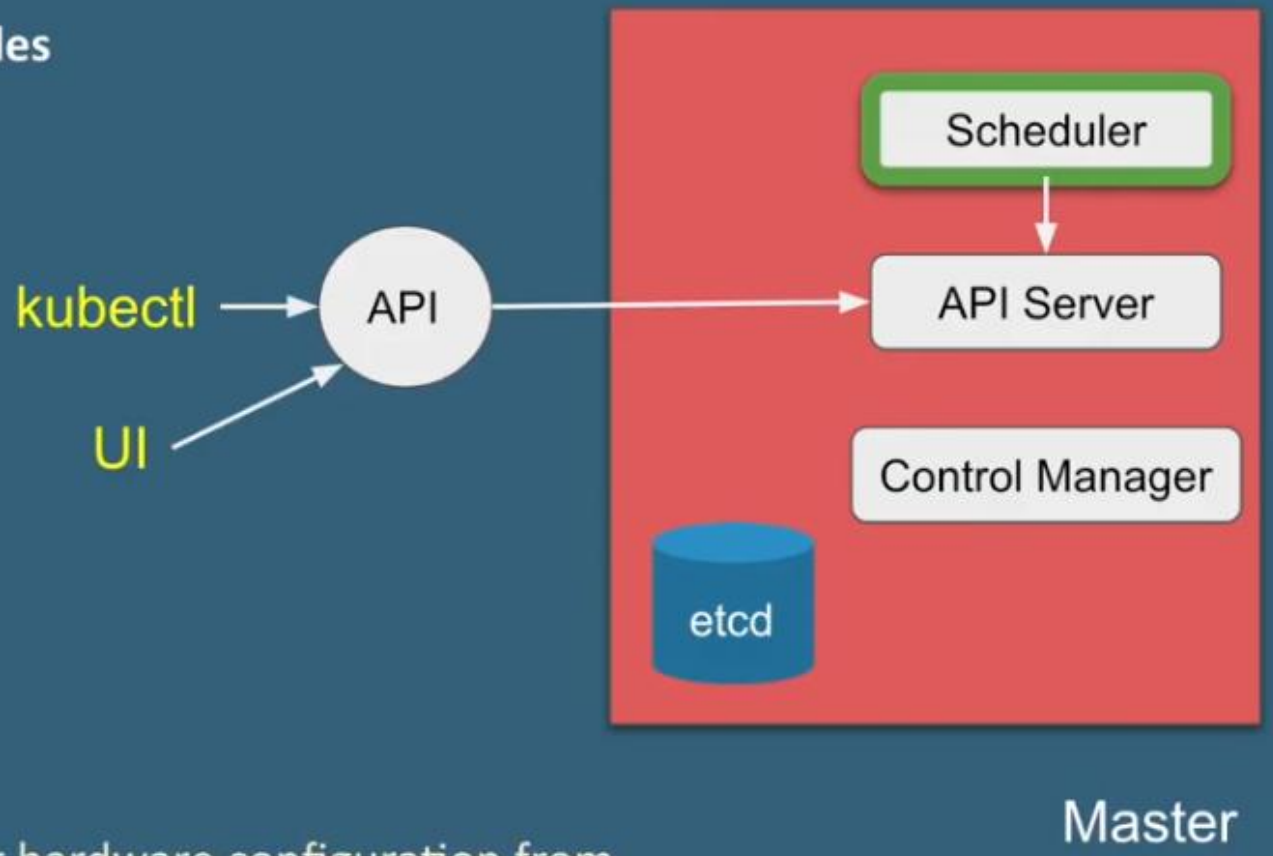


Scheduler

Schedules pods across multiple nodes

Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on

The scheduler obtains from etcd, via the API server, resource usage data for each worker node in the cluster



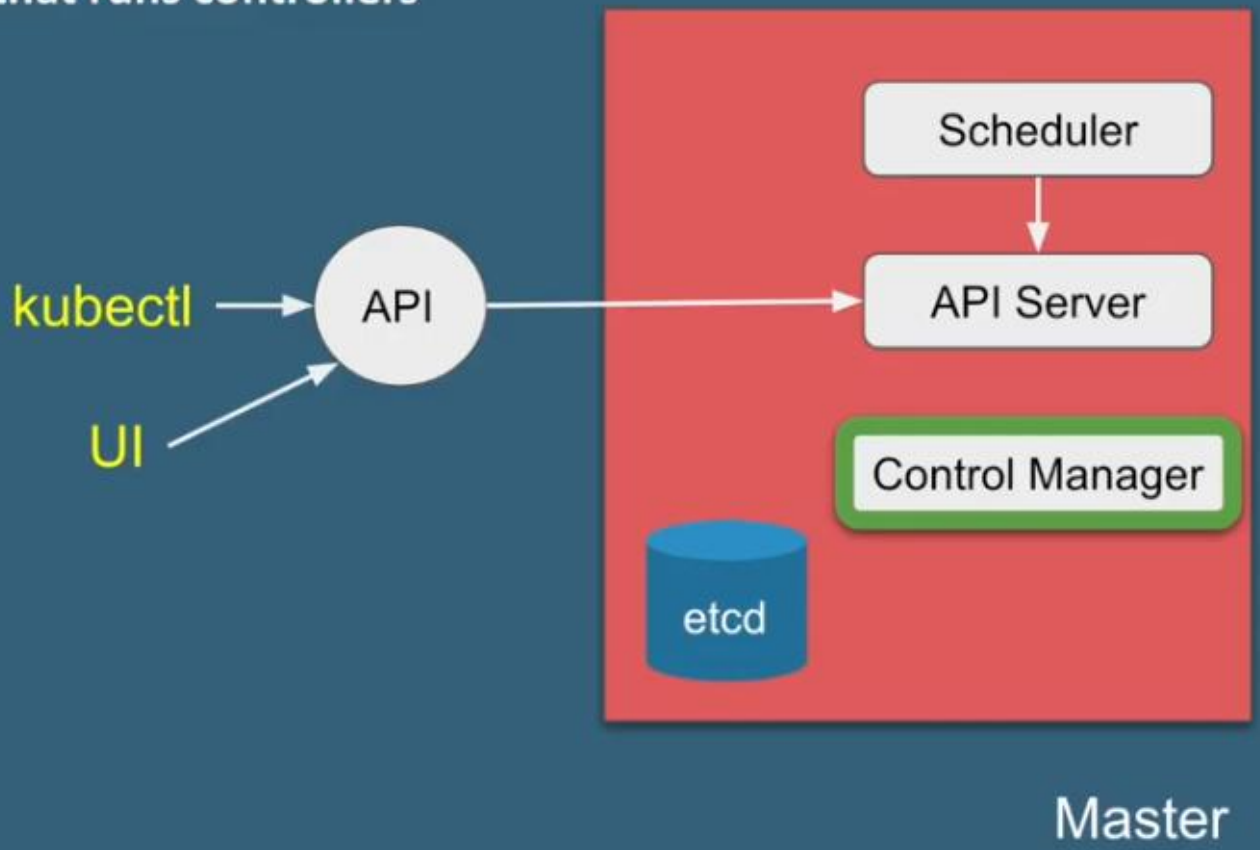
? Scheduler gets the information for hardware configuration from configuration file and schedules the pods on nodes accordingly

Control Manager

This is a component on the master that runs controllers

Kube-controller-manager
cloud-controller-manager

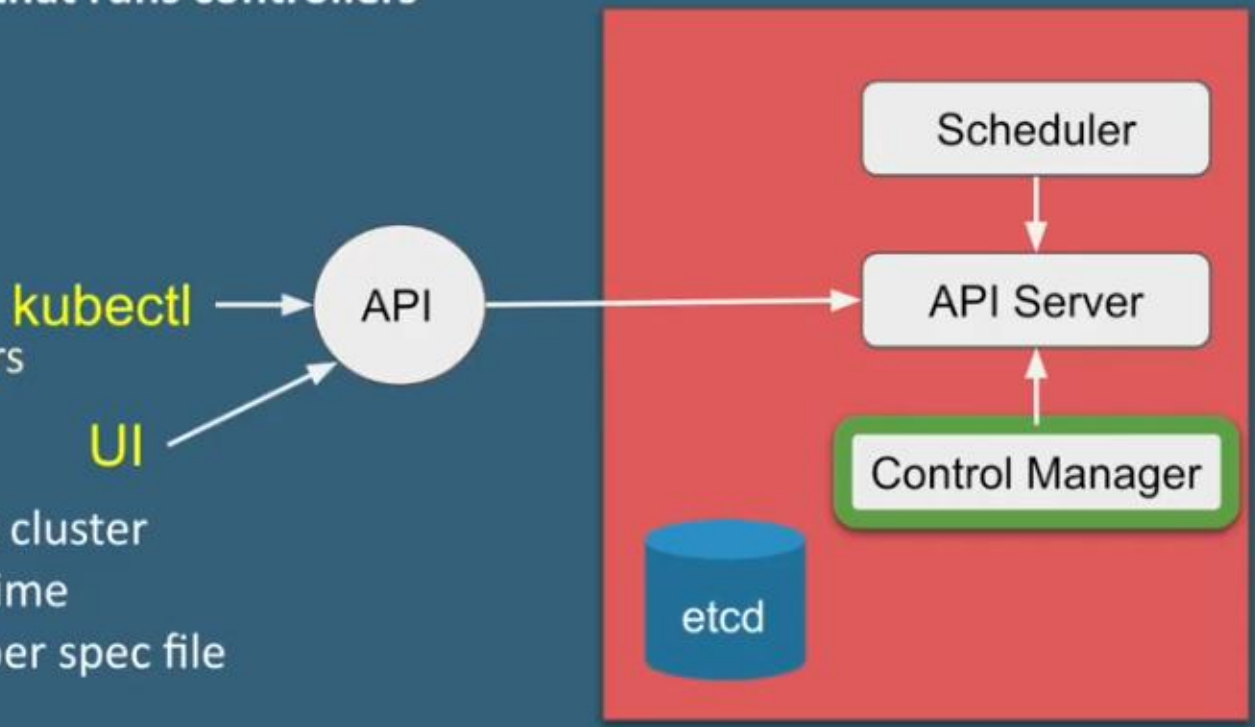
The **cloud-controller-manager** runs controllers responsible to interact with the underlying infrastructure of a cloud provider when nodes become unavailable, to manage storage volumes when provided by a cloud service, and to manage load balancing and routing



Control Manager

This is a component on the master that runs controllers

Kube-controller-manager
Node controller
Replication controller
Endpoints controller
Service account and token controllers



Responsible for overall health of the cluster
. Ensures nodes are running all the time
. Correct no of pods are running as per spec file

? Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process

Master

Control Manager

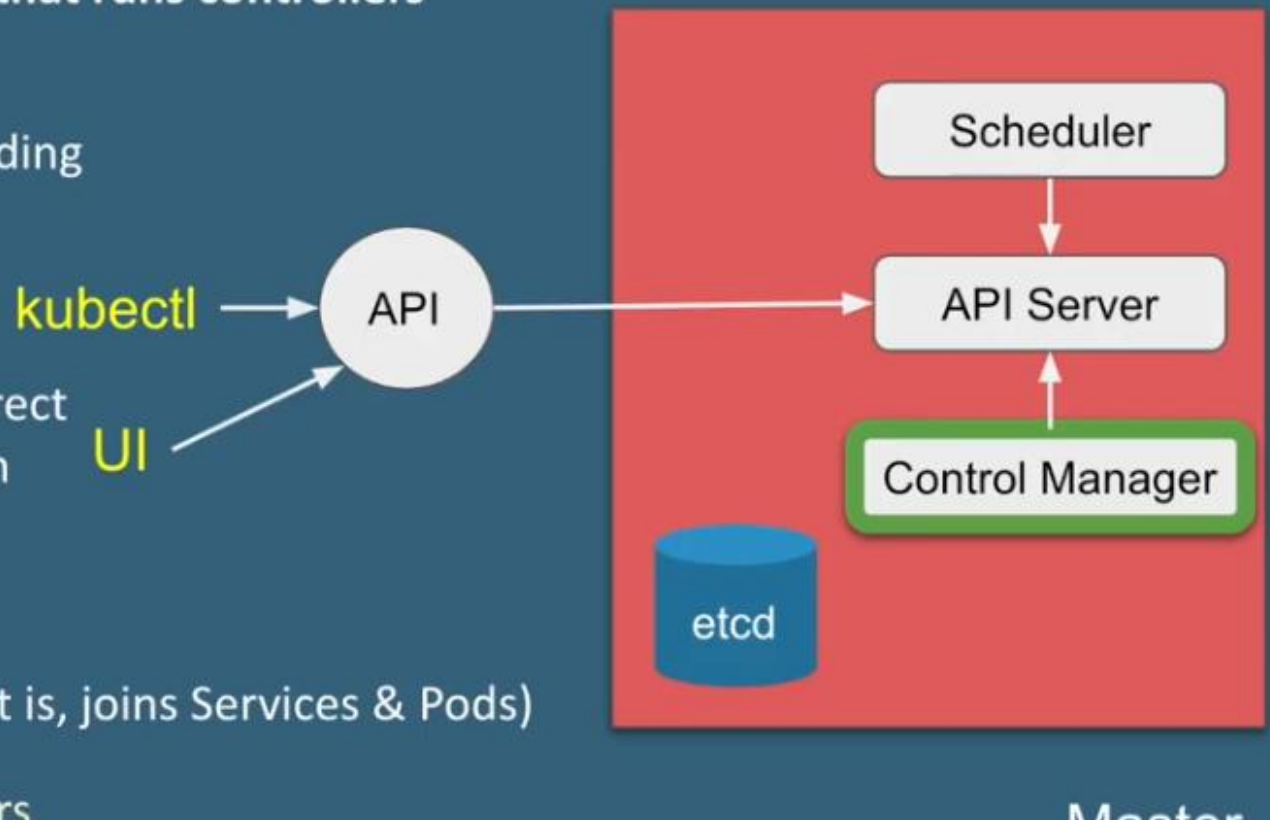
This is a component on the master that runs controllers

Node controller
Responsible for noticing and responding when nodes go down

Replication controller
Responsible for maintaining the correct number of pods for every replication controller object in the system

Endpoints controller
Populates the Endpoints object (that is, joins Services & Pods)

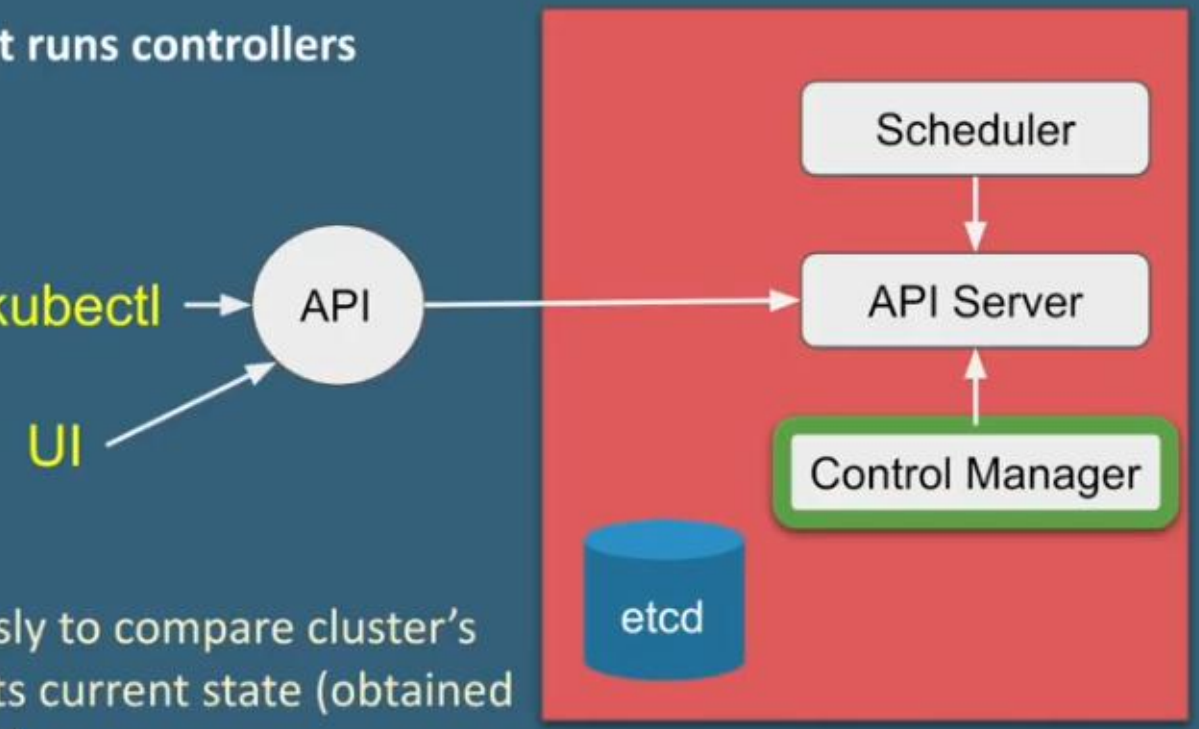
Service account and token controllers
Create default accounts and API access tokens for new namespaces



Control Manager

This is a component on the master that runs controllers

Node controller
Replication controller
Endpoints controller
Service account and token controllers

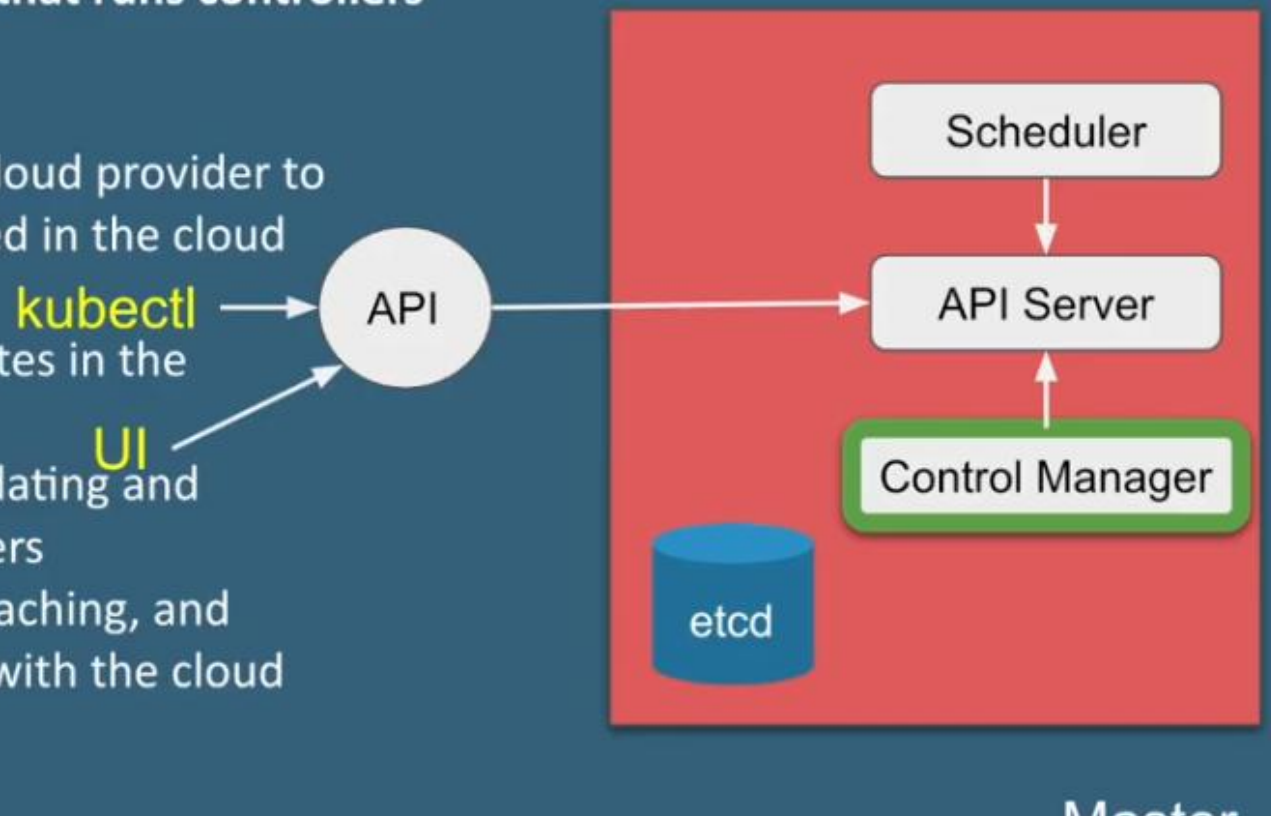


? Controller run watch-loops continuously to compare cluster's desired state (from configuration) to its current state (obtained from etcd data store via the API server)
In case of a mismatch corrective action is taken in the cluster until its current state matches the desired state

Control Manager

This is a component on the master that runs controllers

Cloud-controller-manager
Node Controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
Route Controller: For setting up routes in the underlying cloud infrastructure
Service Controller: For creating, updating and deleting cloud provider load balancers
Volume Controller: For creating, attaching, and mounting volumes, and interacting with the cloud provider to orchestrate volumes



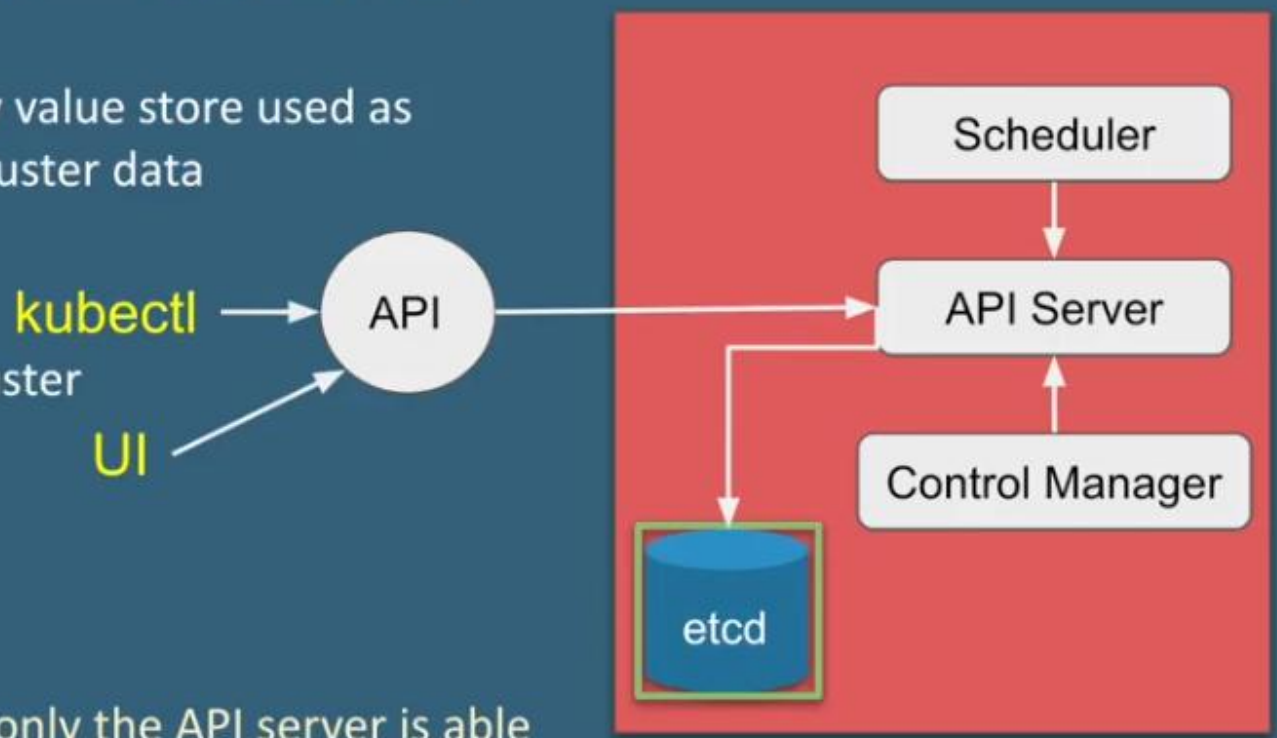
? You can disable the controller loops by setting the --cloud-provider flag to external when starting the kube-controller-manager

etcd

open source, distributed key-value database from CoreOS

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data

single source of truth for all components of the Kubernetes cluster

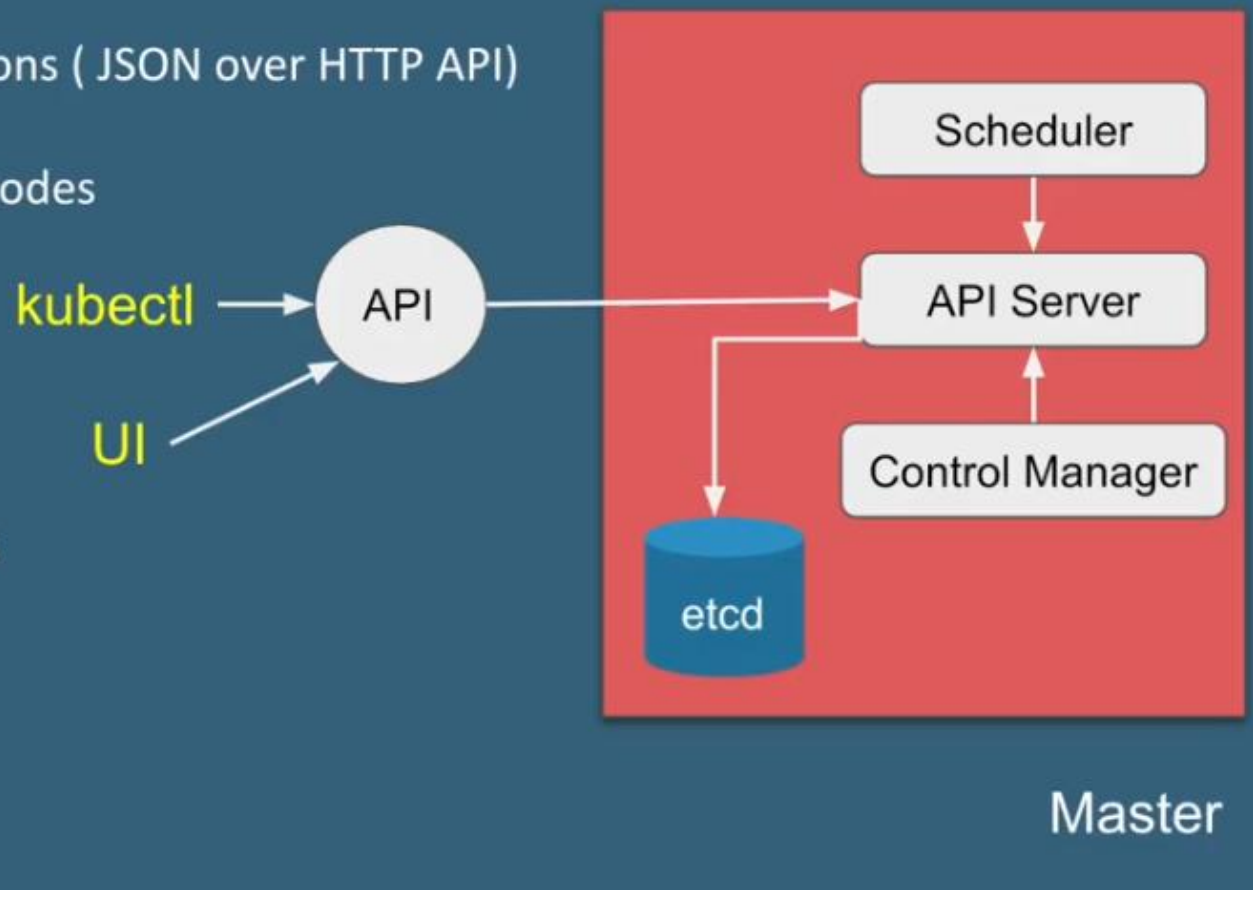


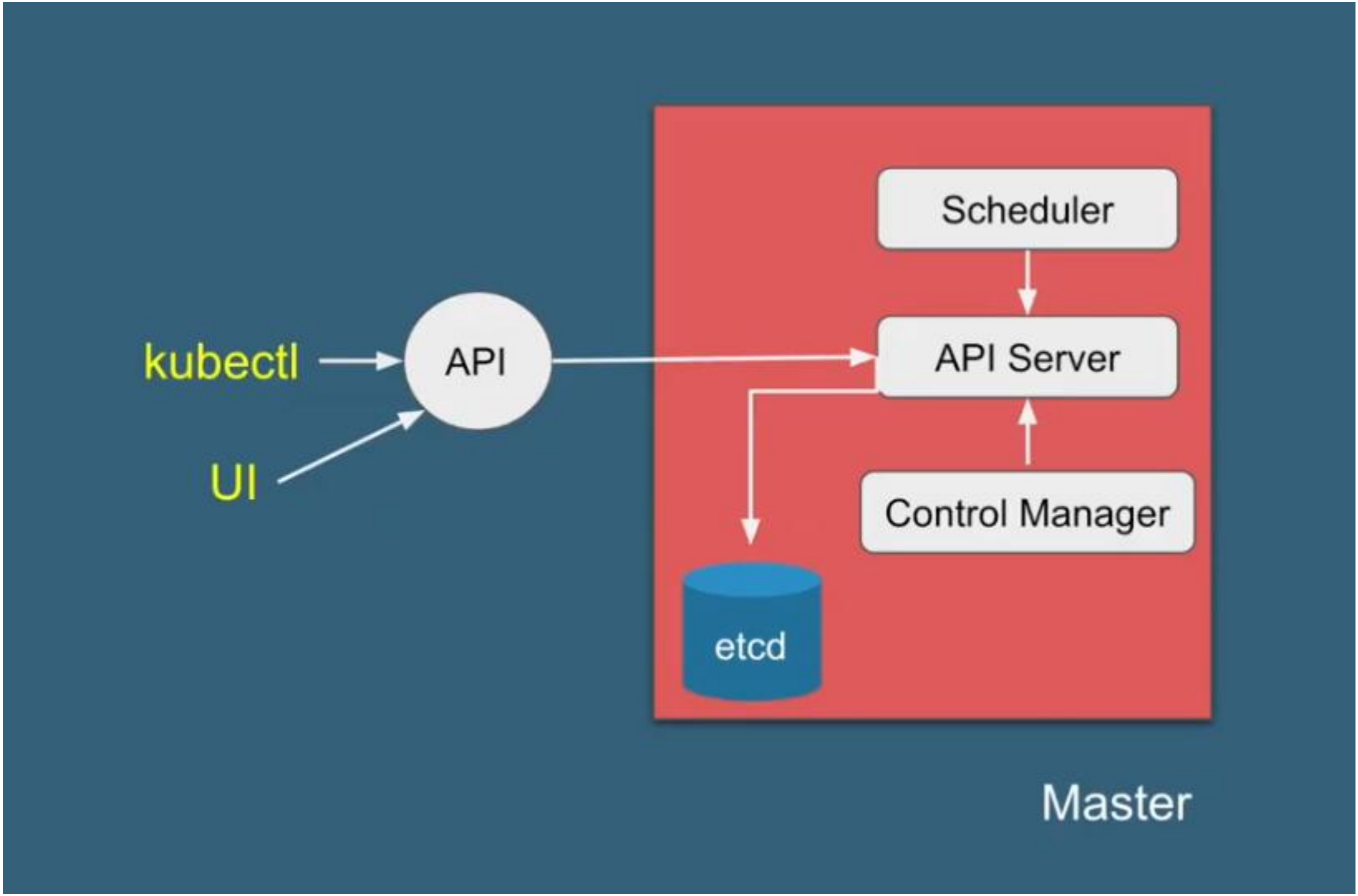
? Out of all the master components, only the API server is able to communicate with the etcd data store

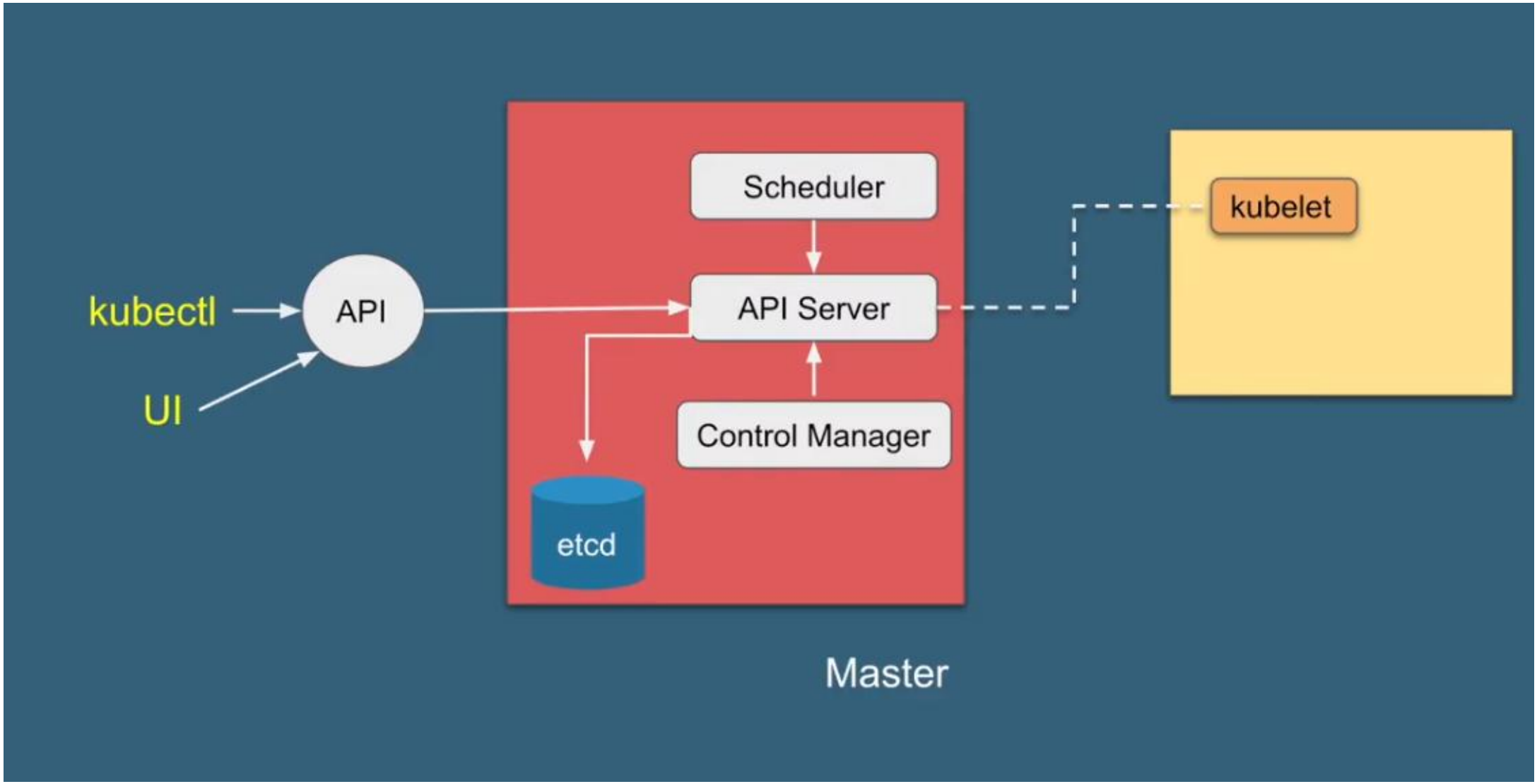
etcd can be part of the Kubernetes Master OR it can be configured externally

4 Components of Master node

1. **API Server** - for all communications (JSON over HTTP API)
2. **Scheduler** - schedules pods on nodes
3. **Controller Manager** - runs controllers
4. **Etcd** - open source, distributed key-value database from CoreOS




kubectl
UI
API
Scheduler
API Server
Control Manager
etcd
Master


kubectl
UI
API
Scheduler
API Server
Control Manager
etcd
kubelet
Master

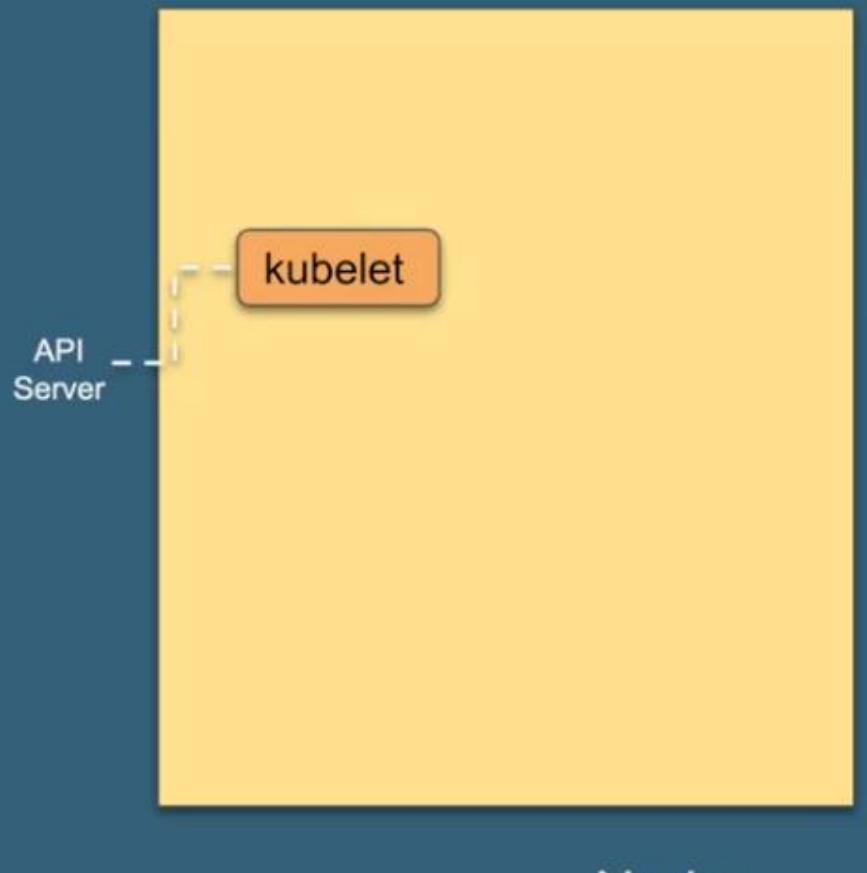
Worker node

Can be any physical or virtual machine where containers are deployed

Every node in a Kubernetes cluster must run a container runtime like Docker

Has following components:

1. kubelet
2. kube-proxy
3. Container runtime



? Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment

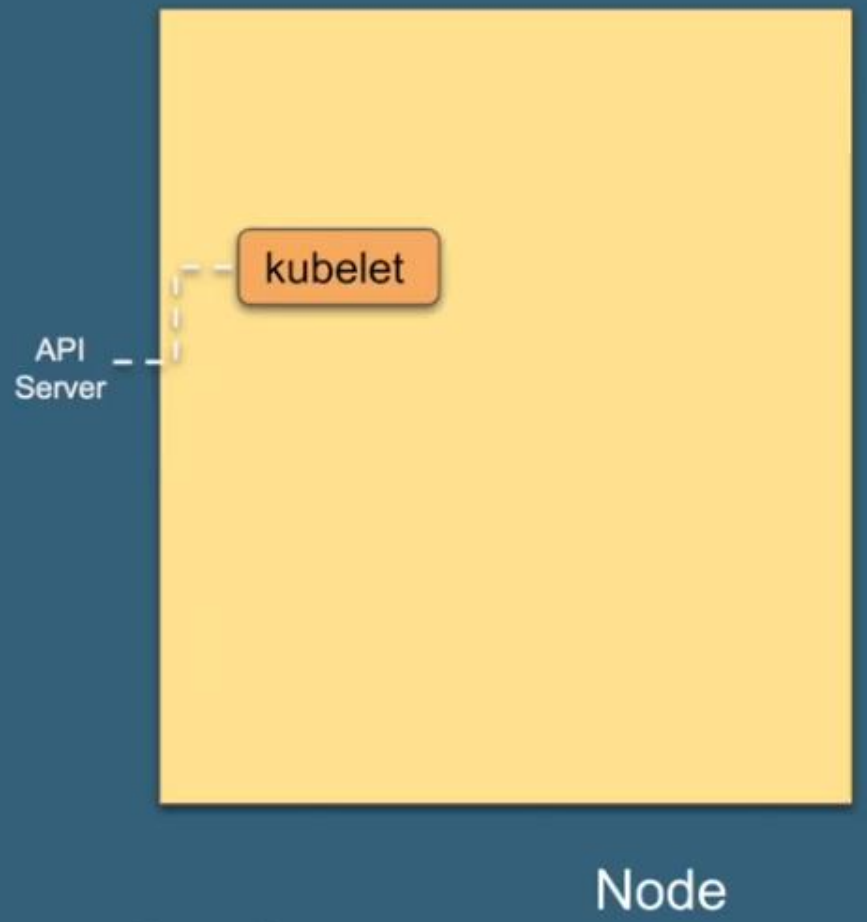
kubelet

kubelet is an agent running on each node
communicates with components from the master node

makes sure that containers are running in a pod

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy

In case any pod has any issue, kubelet tries to restart the pods on the same node or a different node



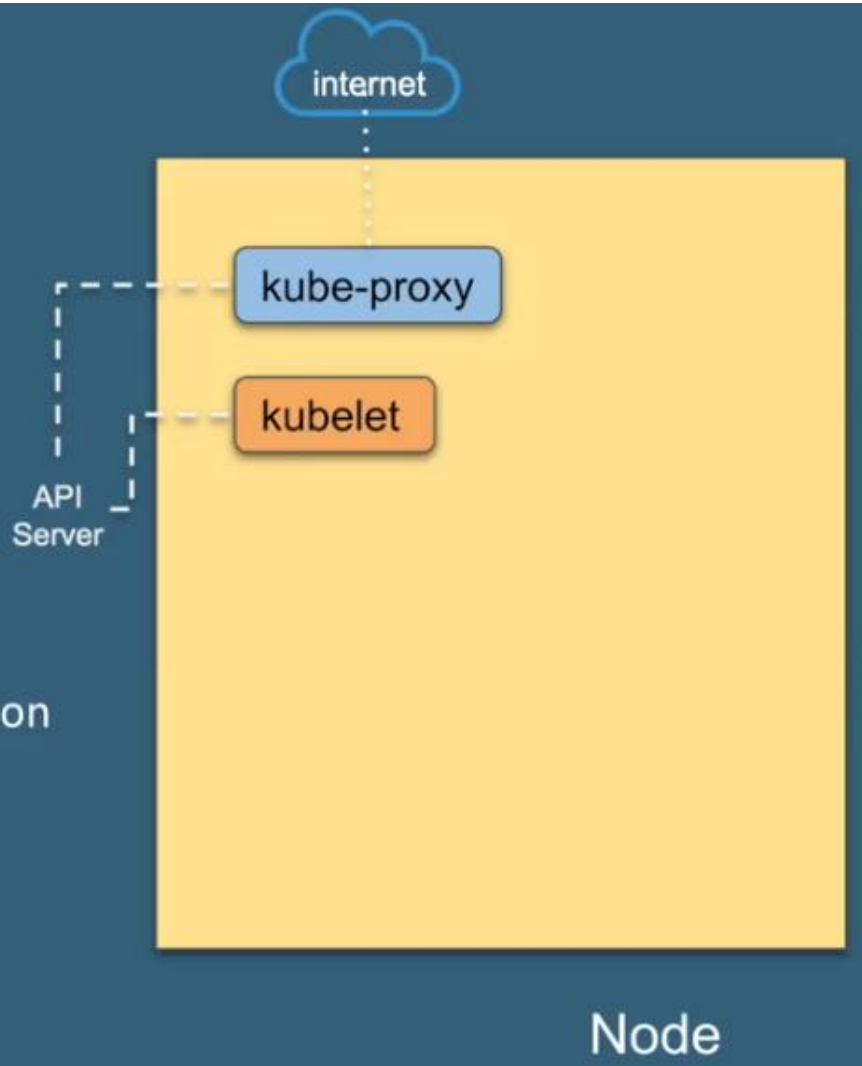
? The kubelet doesn't manage containers which were not created by Kubernetes

kube-proxy

A network agent which runs on each node responsible for maintaining network configuration & rules

Exposes services to the outside world

Core networking components in Kubernetes



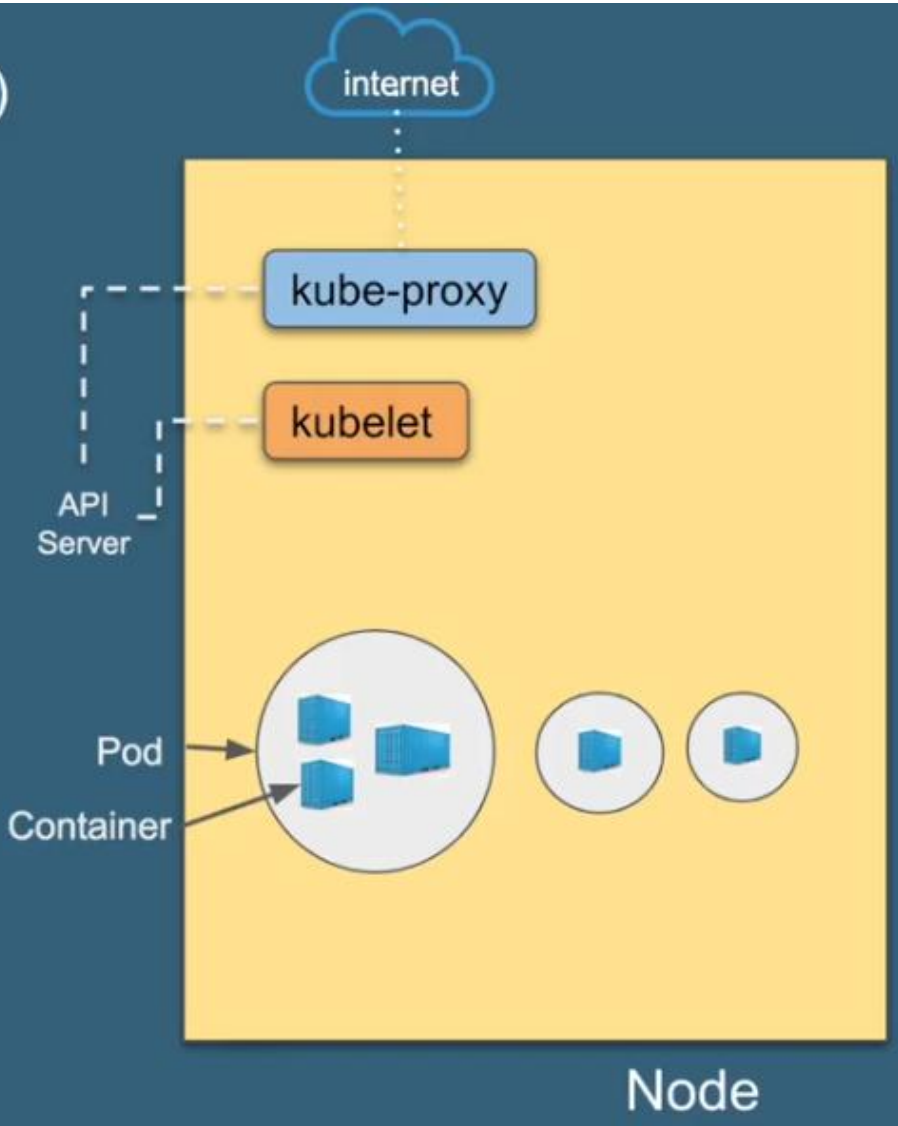
? All worker nodes run a daemon called kube-proxy, which watches the **API server on the master node** for the addition and removal of Services and endpoints

Container runtime (Pods & Containers)

The container runtime is the software that is responsible for running containers

Kubernetes supports several container runtimes:

Docker
Containerd
Cri-o
Rktlet
Kubernetes CRI (Container Runtime Interface).

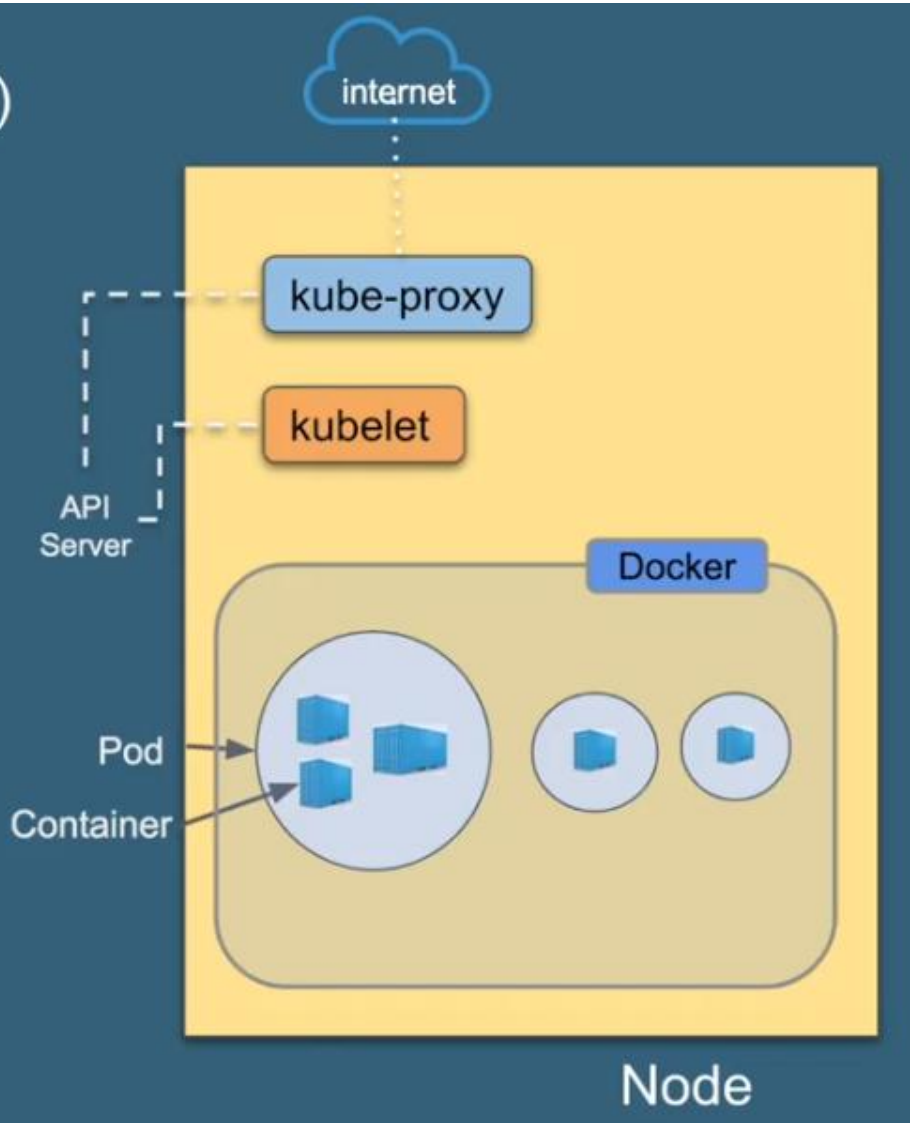


Container runtime (Pods & Containers)

?

Kubernetes does not have the capability to directly handle containers

In order to run and manage a container's lifecycle, Kubernetes requires a container runtime on the node where a Pod and its containers are to be scheduled



Addons for DNS, Dashboard, monitoring, logging etc

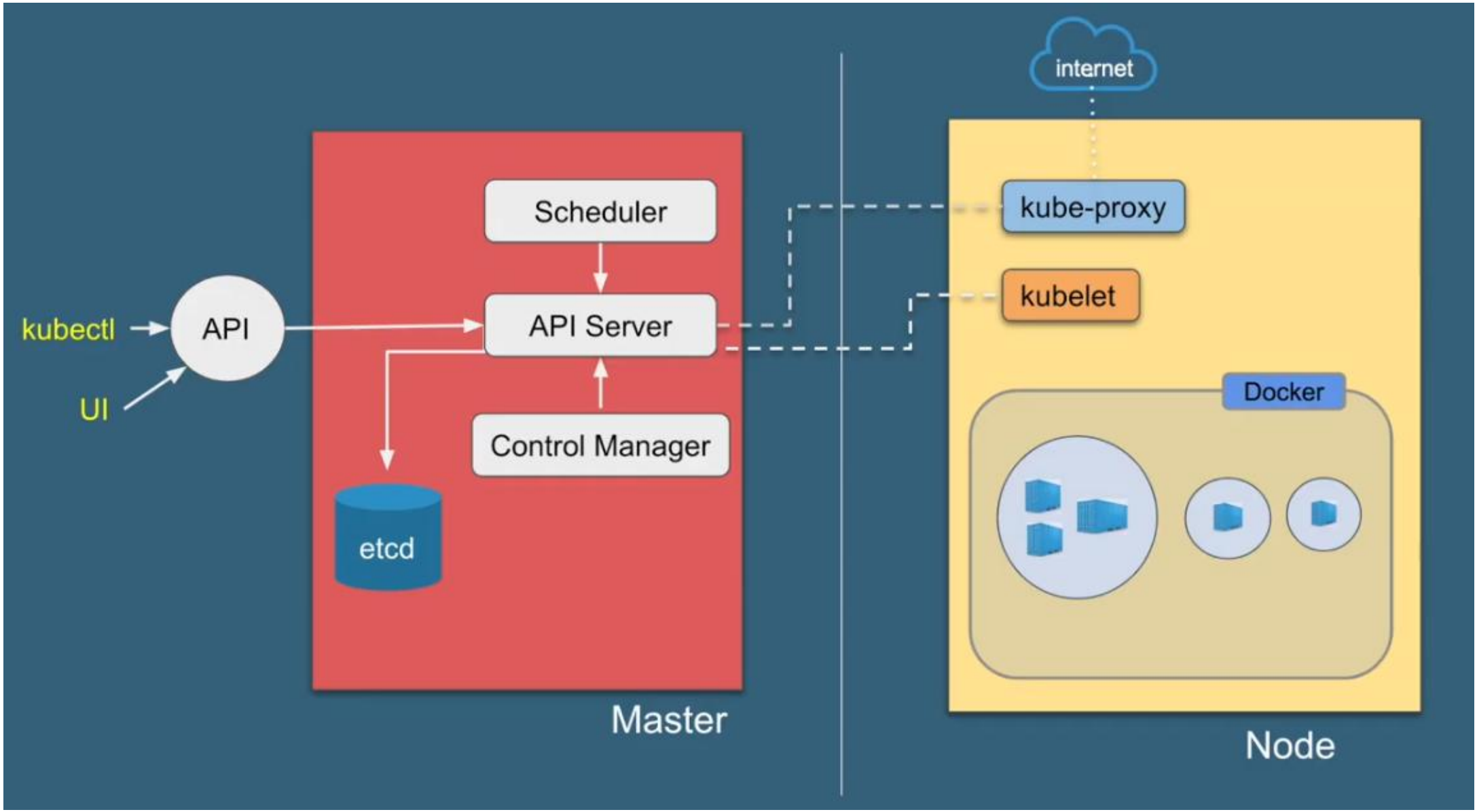
Add-ons extend the functionality of Kubernetes

Dashboard - a general purpose web-based user interface for cluster management

Monitoring - collects cluster-level container metrics and saves them to a central data store

Logging - collects cluster-level container logs and saves them to a central log store for analysis

DNS - cluster DNS is a DNS server required to assign DNS records to Kubernetes objects and resources


internet
Scheduler
kube-proxy
kubectl
API
API Server
kubelet
UI
Docker
Control Manager
etcd
Master
Node