Kubernetes in Action

AH SHAH KHAN

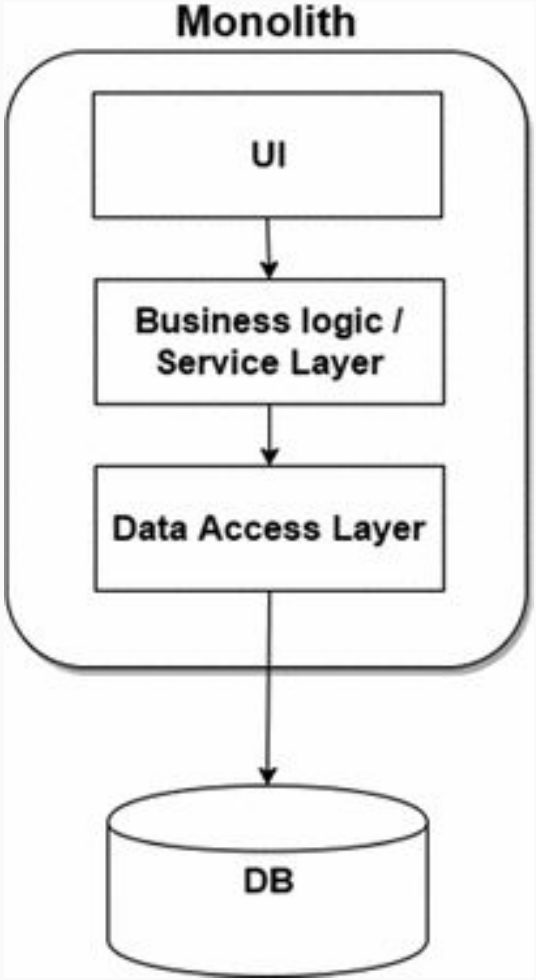
Agenda

1. Monolithic Architecture
2. Microservices
3. Complexities of Microservices
4. What is Kubernetes?
5. The Origin of Kubernetes
6. A Broader Perspective
6. The Purpose of K8s
7. Kubernetes Architecture
8. Kubernetes Master
9. Worker Nodes
10. Hitting a Moving Target
11. Benefits of K8s
12. Required Tools

Monolithic Architecture

Monolithic Architecture

- Designed to be self-contained
- Has all that is needed in itself



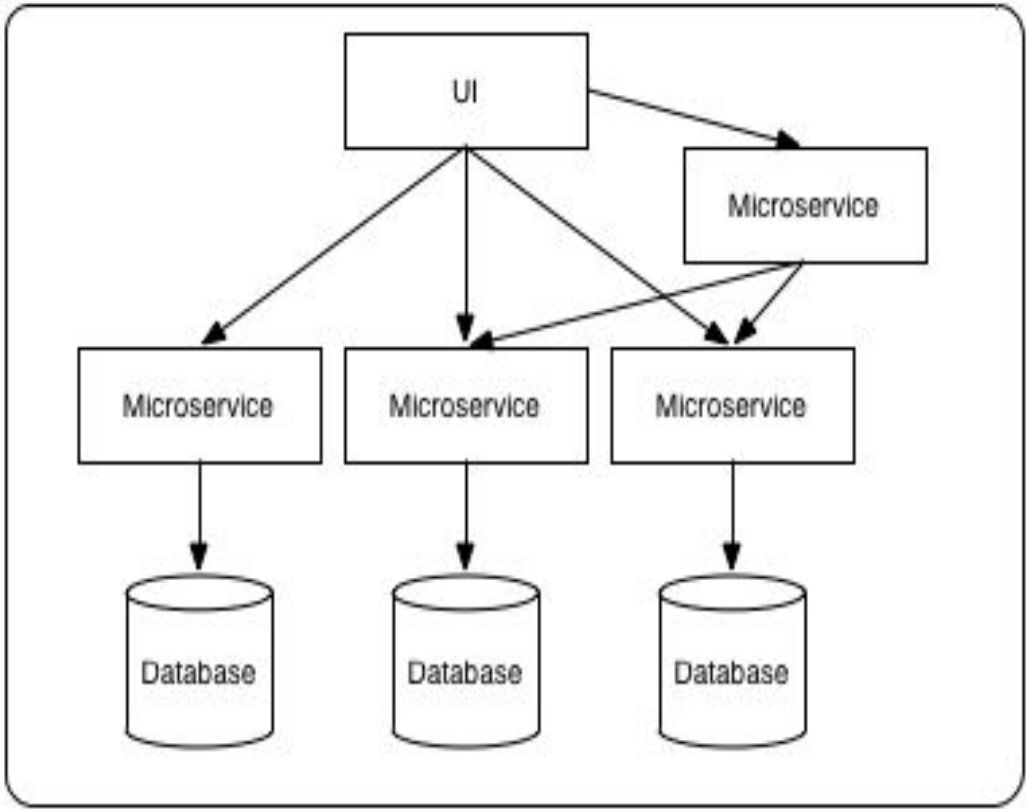
Issues with Monolithic Architecture

- Slow release cycles and relatively infrequent updates
- Once new release gets ready, developers package up the whole system and hand it over to the ops team, who then deploys and monitors it
- In case of hardware failures, the ops team manually migrates it to the remaining healthy servers

Microservice Architecture

Microservice Architecture

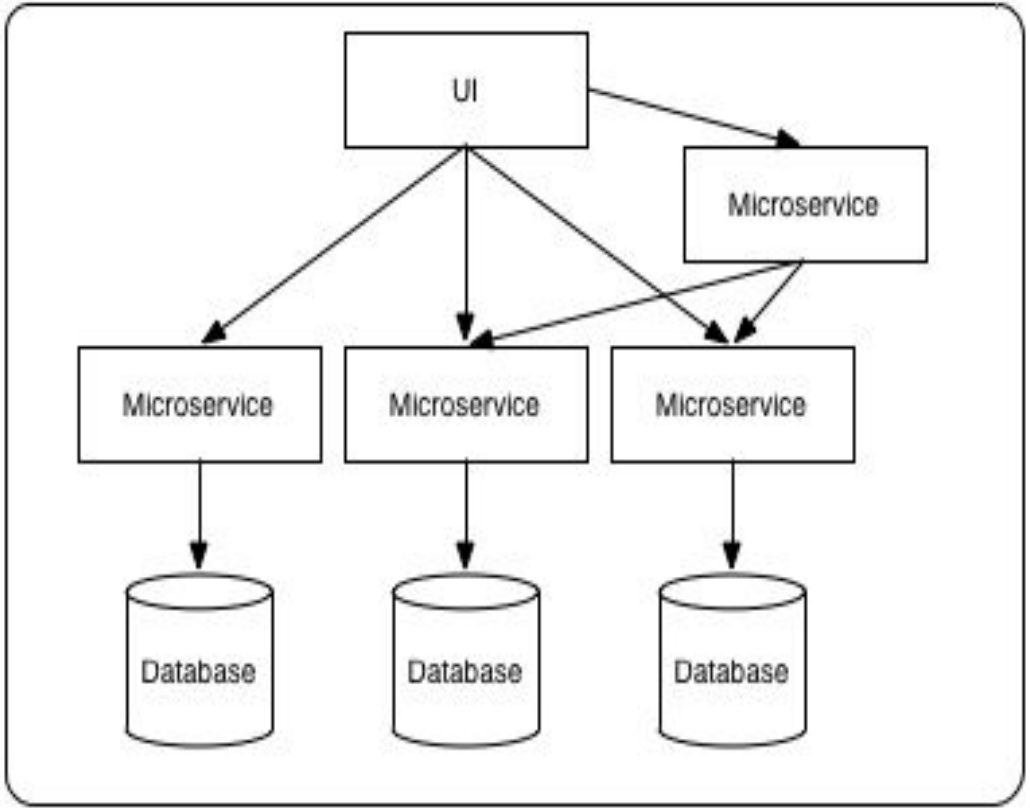
- Breaking down big applications into smaller components
- Microservices are decoupled from each other



Microservices Architecture

Microservice Architecture

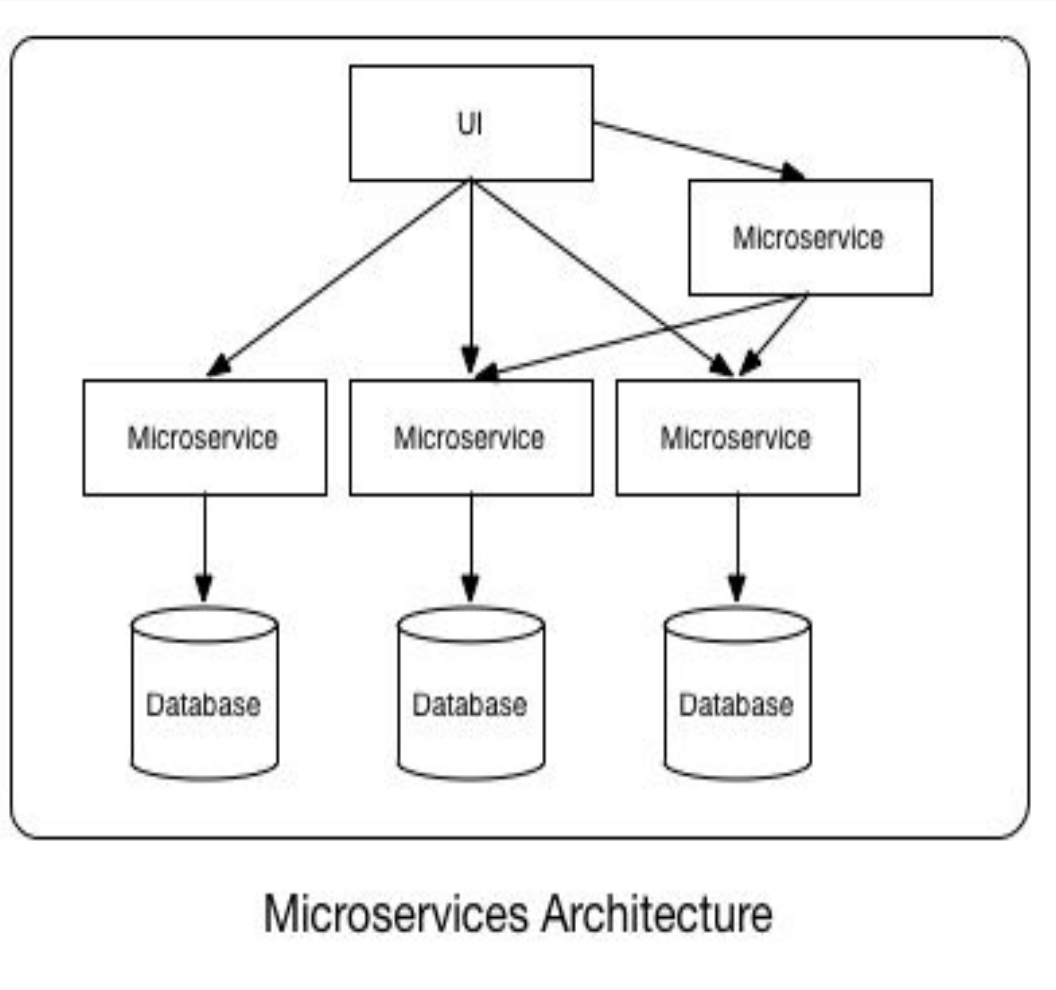
- They can be developed, deployed, updated, and scaled individually



Microservices Architecture

Microservice Architecture

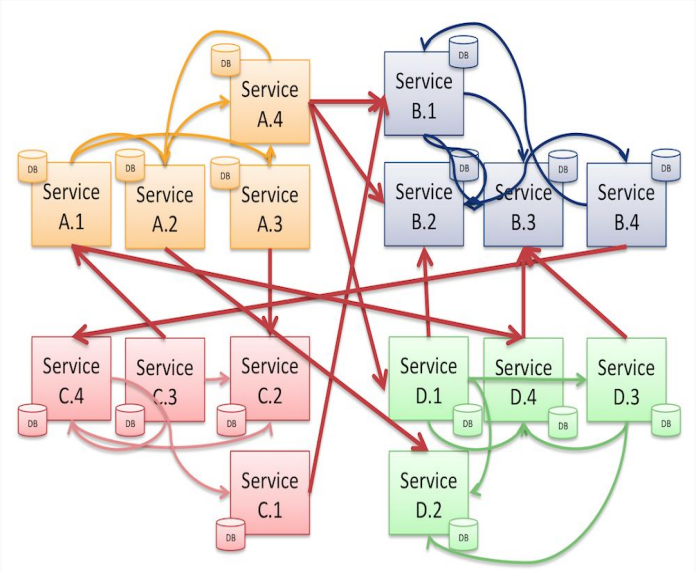
- Making a change to any component is convenient and new technology can be adopt where needed



Microservices Architecture

Complexities of Microservice Architecture

Complexities of Microservice Architecture



- Configuring, managing, and keeping the whole system running smoothly
- Achieving max hardware resource utilization
- Making sure all services are working
- Scaling services

Automated:

- **Scheduling**
- **Configuration**
- **Supervision**
- **Failure-handling**

What is Kubernetes?

What is Kubernetes?

- Developers can deploy applications as often as they want, without assistance from the operations team
- It also helps the ops team by automatically monitoring and rescheduling apps in the event of hardware failure
- Exposes your whole data center as a single enormous computational resource

What is Kubernetes?

- When you have multiple servers and you are deploying a multi-component application through Kubernetes,
  - Selects a server for each component
  - Deploys components
  - Enables accessibility and communication easy among all application components

Magical Powers of Kubernetes

- Easy component management makes Kubernetes great for most on-premises data centers & cloud providers
- Kubernetes allows cloud providers to offer developers a simple platform for deploying and running any type of application, while not requiring the cloud provider's own sysadmins to know anything about the tens of thousands of apps running on their hardware.

The Origin of Kubernetes

The Origin of Kubernetes

- Google created a system for there internal use with the name “Borg”
- Later they changed the name of Borg with “Omega”
- The Purpose of Borg/Omega was to help both application developers and system administrators **manage google’s applications and services**

The Origin of Kubernetes

- Helped them achieve a much **higher utilization of their infrastructure**, which is important when your organization is that large.
- A decade after the creation of Borg/Omega, **Google introduced Kubernetes in 2014**, an **open-source** system based on the experience gained through Borg, Omega, and other internal Google systems.

A Broader Perspective of Kubernetes

kubernetes by Google

Manage a cluster of Linux containers as a single system to accelerate Dev and simplify Ops.

A Broader Perspective of Kubernetes

- Kubernetes enables you to run your software applications on thousands of computer nodes **as if all those nodes were a single computer**
- It abstracts away the underlying infrastructure and, by doing so, **simplifies development, deployment, and management**

A Broader Perspective of Kubernetes

- **Deploying applications through Kubernetes is always the same**, whether your cluster contains only a couple of nodes or thousands of them
- The **size of the cluster makes no difference** at all

The Purpose of Kubernetes

An OS for the Cluster

The Purpose of Kubernetes

- Application developers can focus on their application instead of infrastructure-related services into their apps:
  - Service discovery
  - Maximize resource utilization
  - Scaling / Load-balancing
  - Self-healing

Service Discovery

- Kubernetes will:
  - run your containerized app somewhere in the cluster
  - provide information to its components on how to find each other
  - keep components running

Maximize Resource Utilization

- Kubernetes can **relocate the app** at any time
- By **mixing and matching apps**, Kubernetes can achieve far better resource utilization than is possible with manual scheduling

Scaling / Load Balancing

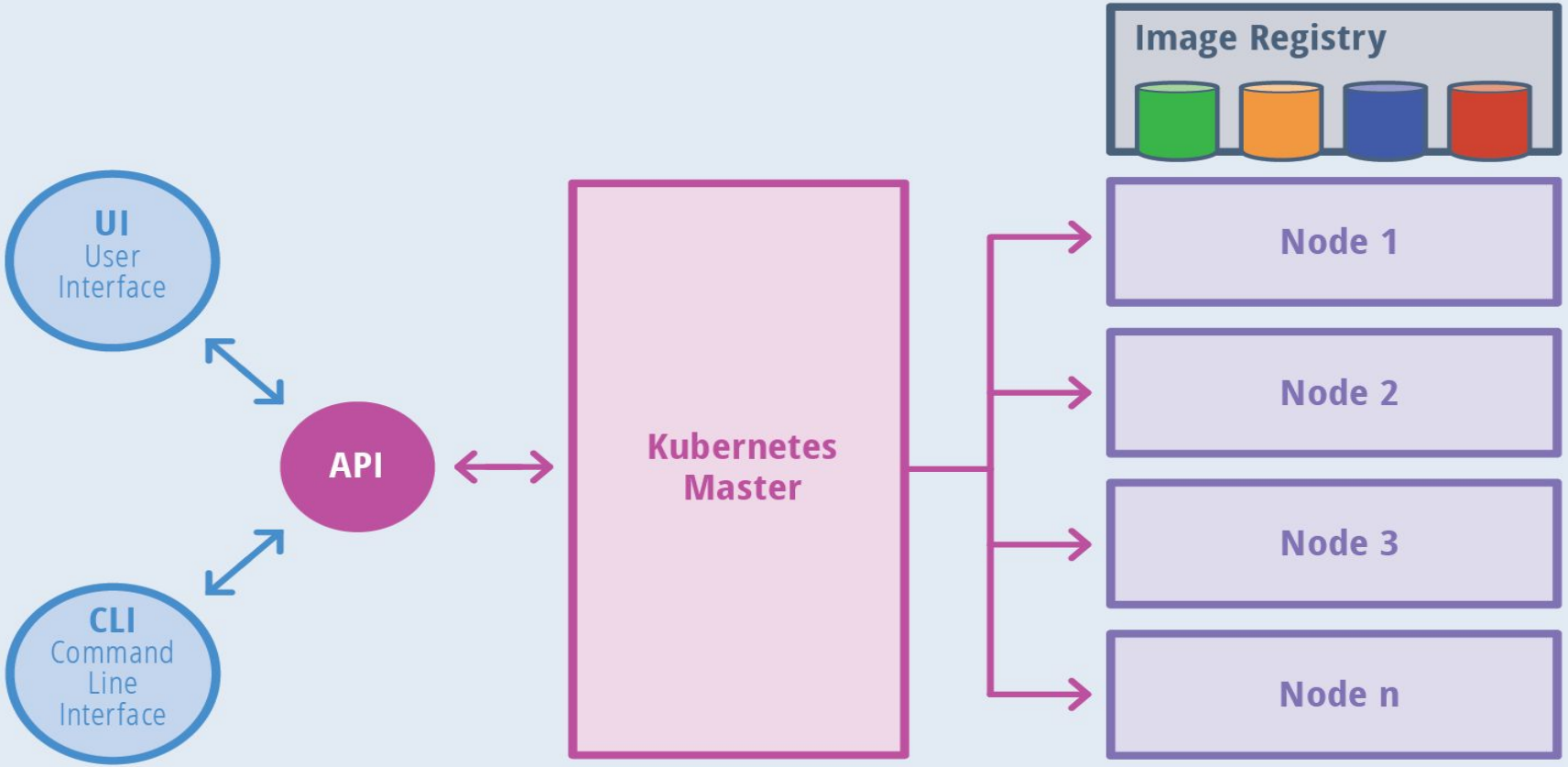
- Easily add or remove app replica manually and kubernetes then will **auto divert traffic** to each for load balancing
- Kubernetes will **add or remove copies (replicas) of your application based on real-time metrics**, such as CPU load, memory consumption, queries per second, or any other metric your app exposes

Self Healing

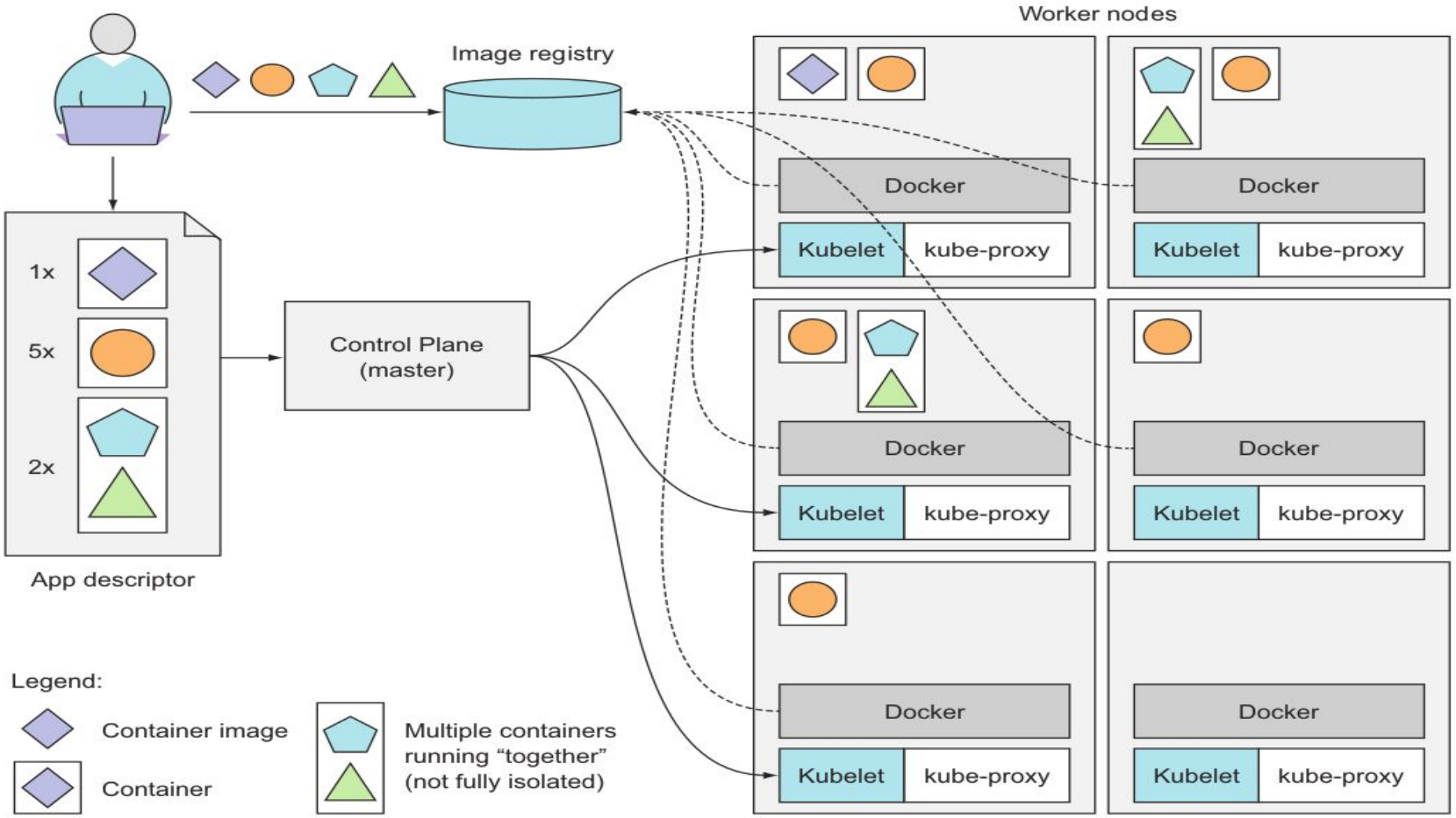
- Kubernetes **checks our application's** health on a described period and it also **does the needful** in case there is no response from the containerized app
- In case of worker node failure Kubernetes immediately starts up another worker node

Kubernetes Architecture

Kubernetes Architecture

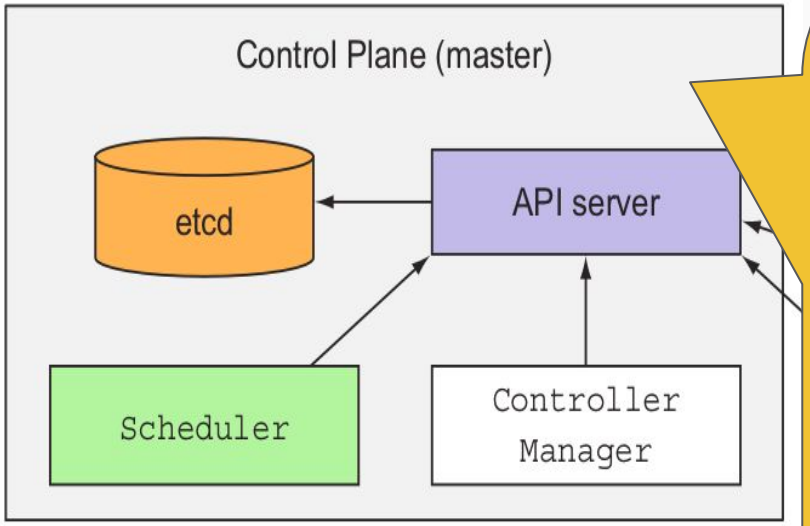


Source: Janakiram MSV


Worker nodes
Image registry
Docker
Kubelet
kube-proxy
1x
5x
2x
App descriptor
Control Plane
(master)
Legend:
Container image
Container
Multiple containers
running "together"
(not fully isolated)

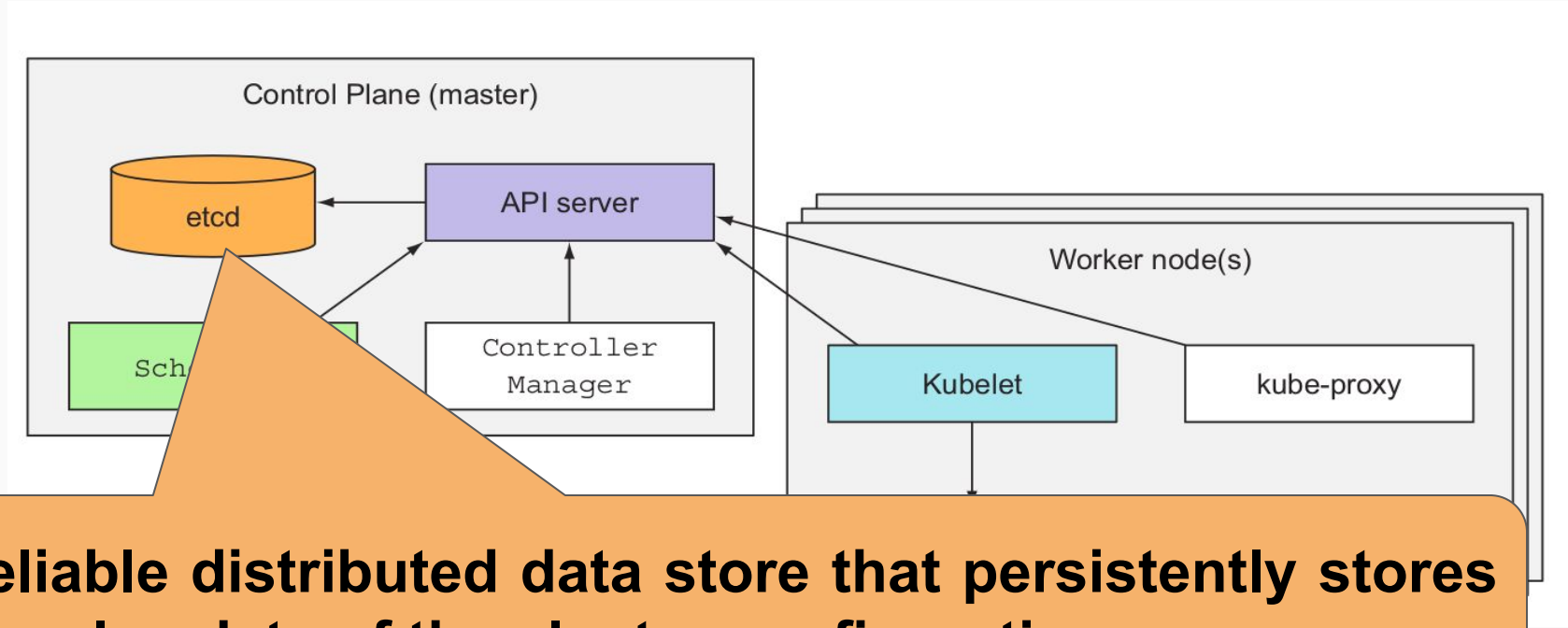
Kubernetes Master

Kubernetes Master - The Control Plane (Master)



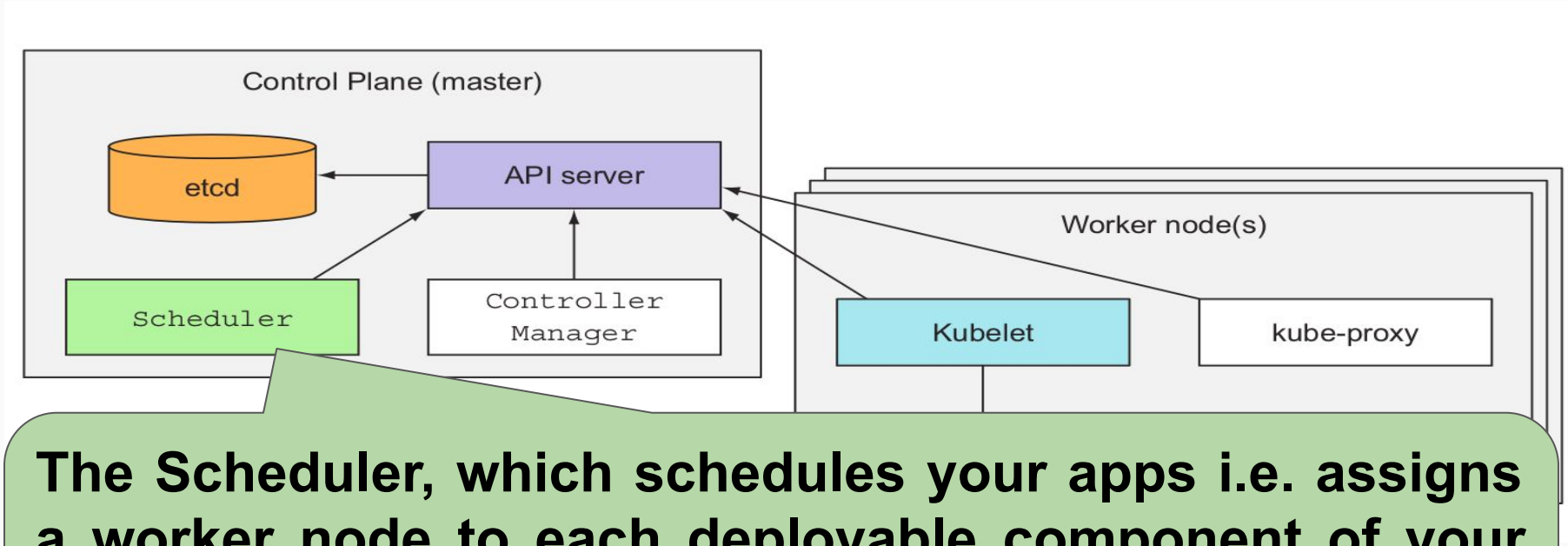
- **The control plane (master) is what controls the cluster and makes it function.**
- **The control plane consists of multiple components.**
- **It could be on a single master node or be split across multiple nodes and replicated to ensure high availability.**

Kubernetes Master - etcd



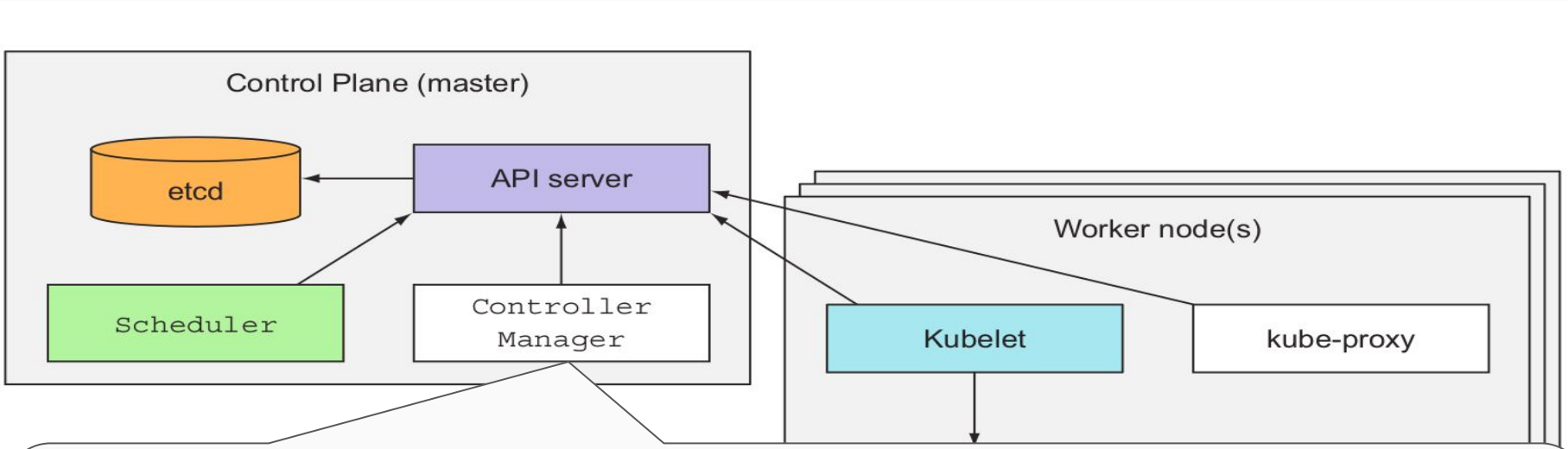
a reliable distributed data store that persistently stores key-value data of the cluster configuration

Kubernetes Master - Scheduler



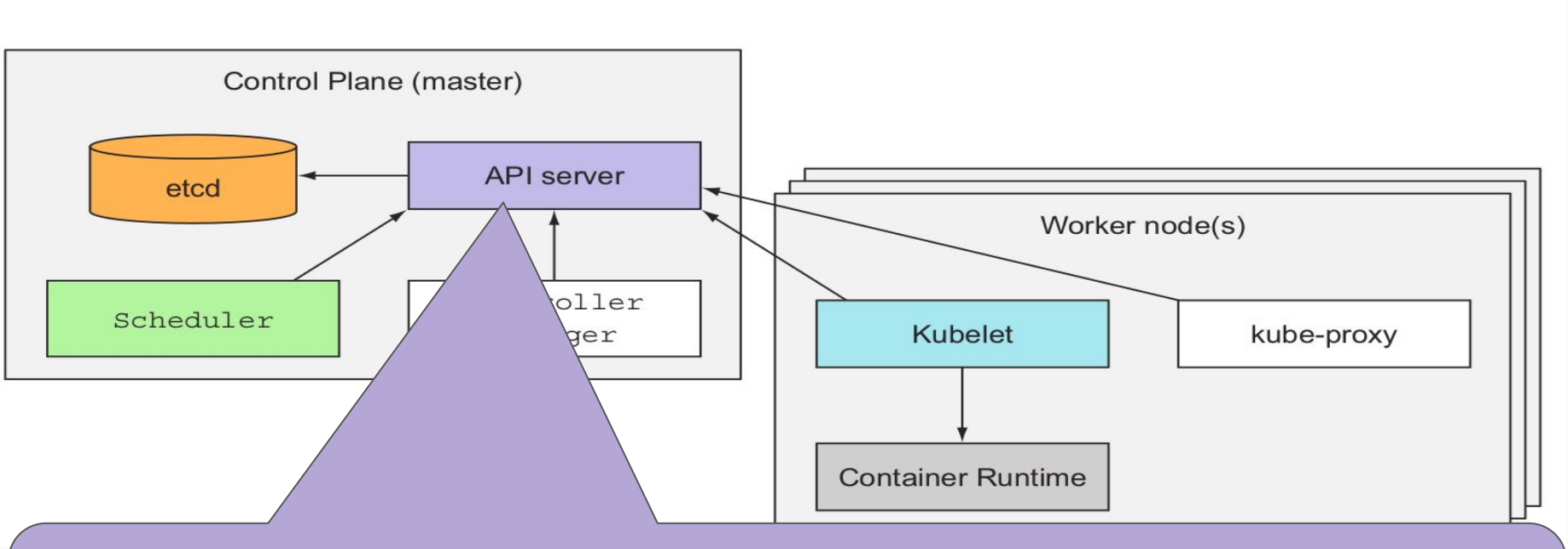
The Scheduler, which schedules your apps i.e. assigns a worker node to each deployable component of your application depending on resource requirement by app and resource availability of the node.

Kubernetes Master - Controller Manager



The Controller Manager, which performs cluster-level functions, such as replicating components, keeping track of worker nodes, handling node failures, and so on

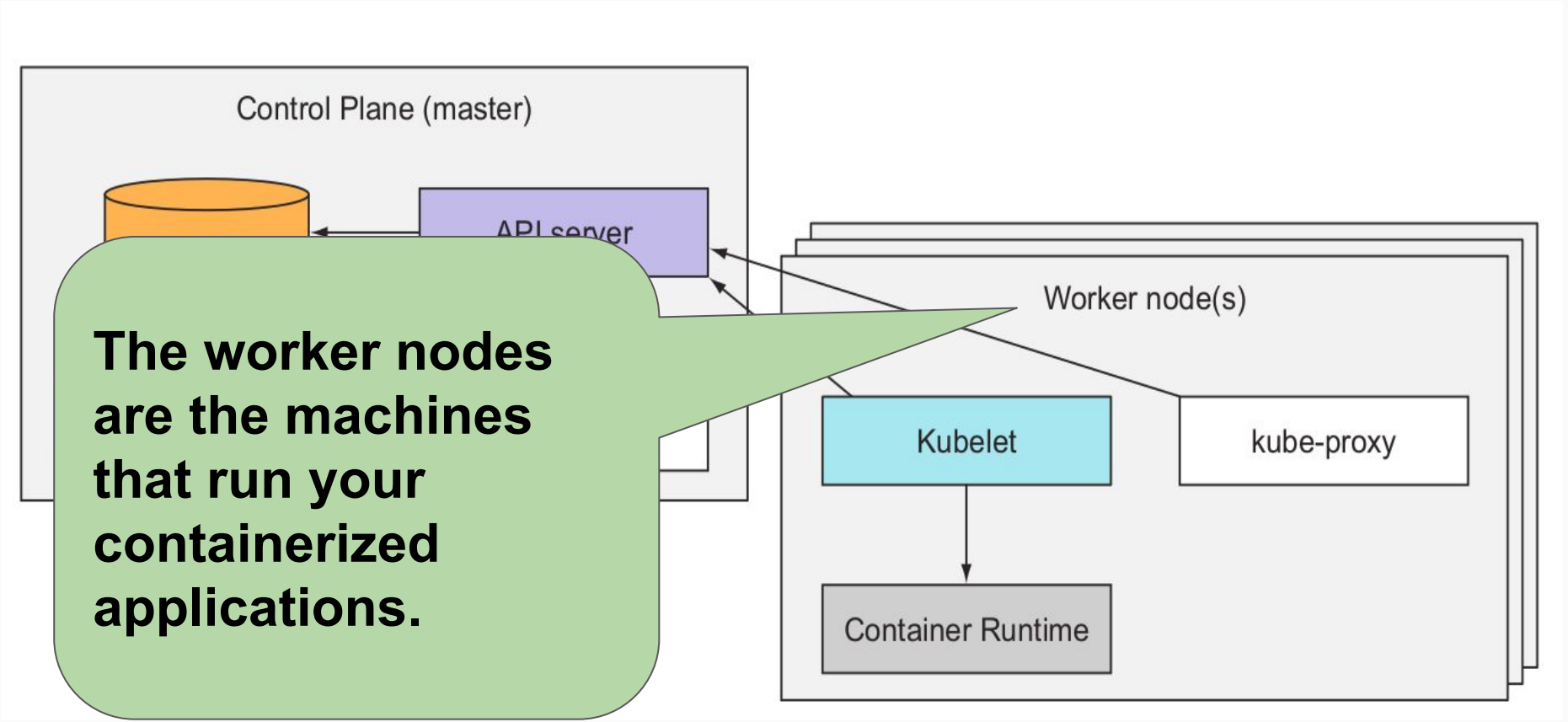
Kubernetes Master - API Server



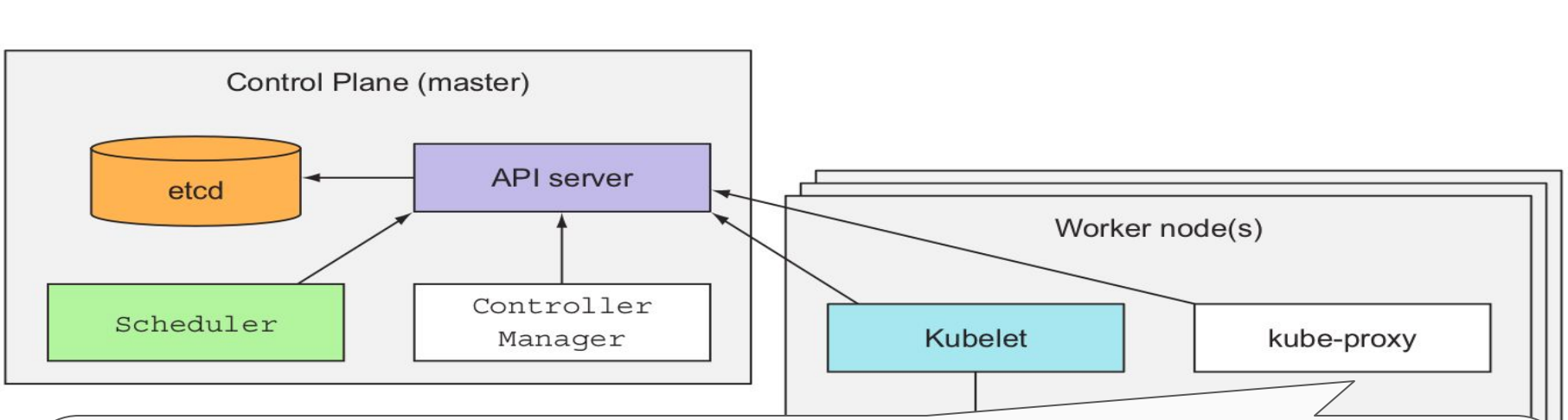
The Kubernetes API Server, which you (user), other Control Plane and worker node components communicate with

Kubernetes Worker Nodes

Kubernetes Master - Worker Nodes

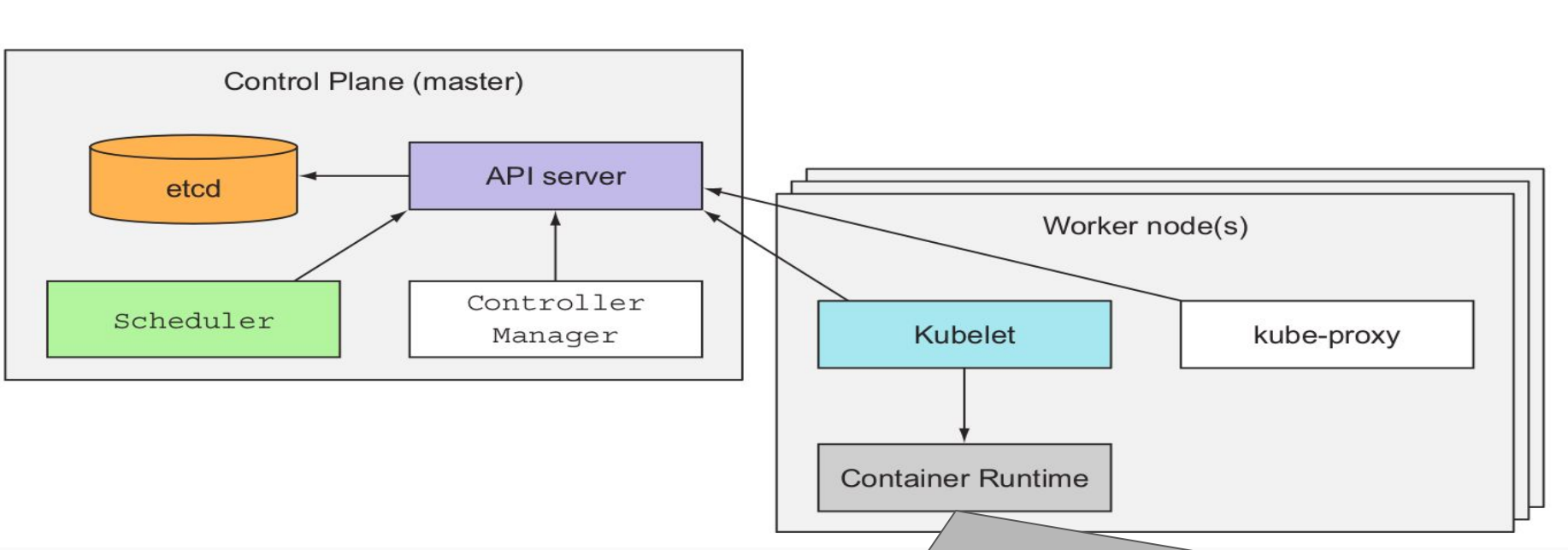


Kubernetes Master - kube-proxy



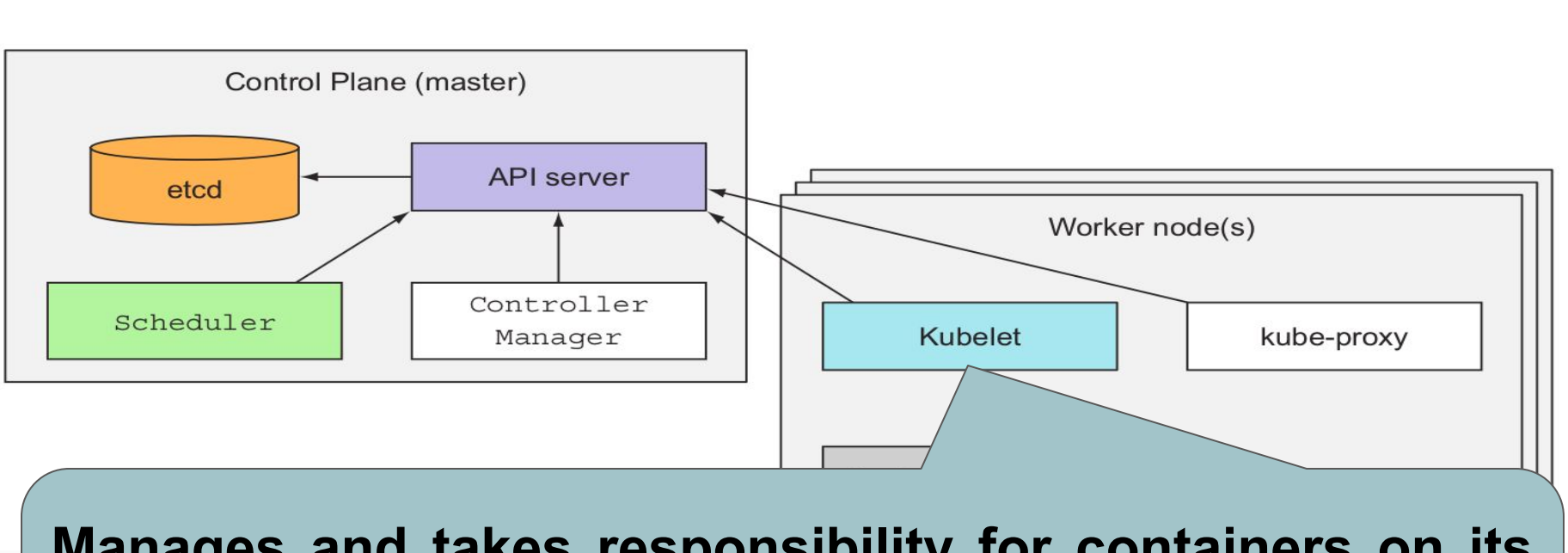
A networking proxy that runs on each worker node, enables pod to pod, pod-to-service communication
The Kubernetes Service Proxy (kube-proxy), which also load-balances network traffic between application components.

Kubernetes Master - Container Runtime



Docker, rkt, or another container runtime, which runs your containers

Kubernetes Master - Container Runtime



Manages and takes responsibility for containers on its node. It ensures that the containers described in those PodSpecs are running and healthy

Hitting a Moving Target

Hitting a Moving Target

- Kubernetes may need to move containers around the cluster
- This can occur in any of the follow case
  - The node they were running on has failed
  - Because they were evicted from a node to make room for other containers

Hitting a Moving Target

- If the container is providing a service to external clients or other containers running in the cluster,
  - how can they use the container properly if it's constantly moving around the cluster?
  - how can clients connect to containers providing a service when those containers are replicated and spread across the whole cluster?

Hitting a Moving Target

- Kubernetes keeps track of all the containers it manages
- If multiple containers provide the same service then you can group them at a single static IP address
- Kubernetes then expose that address to all applications running in the cluster or to the outside world

Hitting a Moving Target

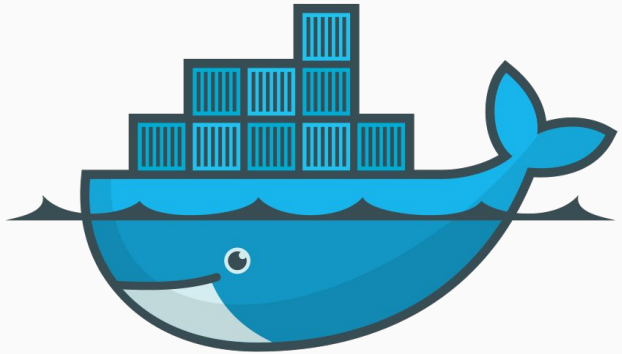
- The kube-proxy will make sure connections to the service are load balanced across all the containers that provide that same service
- The IP address of the service stays constant, so clients can always connect to its containers, even when they’re moved around the cluster

Benefits of Kubernetes

Benefits of Kubernetes

- The ops team doesn’t need to deal with deploying your apps anymore because a containerized application already contains all it needs to run, the system administrators don’t need to install anything to deploy and run the app.
- On a node where Kubernetes is deployed, K8s can run the app immediately without help from sysadmins

Tools Required for Running Kubernetes



Docker - http://docs.docker.com/engine/installation/

Minikube - http://github.com/kubernetes/minikube

Kubernetes Client (KUBECTL) - https://kubernetes.io/docs/tasks/tools/install-kubectl/


CERTIFIED
kubernetes
ADMINISTRATOR


CERTIFIED
kubernetes
APPLICATION
DEVELOPER

Certified Kubernetes
Application Developer (CKAD)

13% – Core Concepts

18% – Configuration

10% – Multi-Container Pods

18% – Observability

20% – Pod Design

13% – Services & Networking

8% – State Persistence

Minikube

Minikube - http://github.com/kubernetes/minikube

Minikube

- The Kubernetes architecture consists of the master and worker nodes
- Minikube is a tool that makes it easy to run Kubernetes locally
- Minikube runs a single-node Kubernetes cluster on your laptop to use kubernetes for practice or development

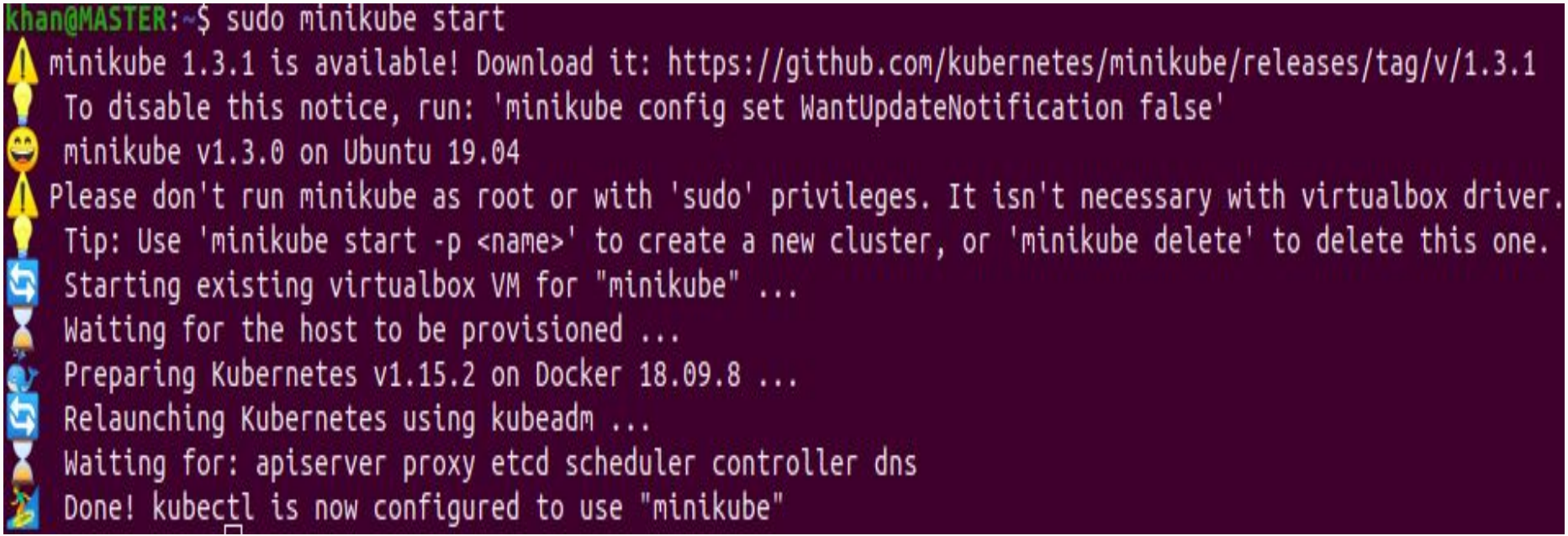
minikube start

To start your local Kubernetes cluster using minikube

Run the command

Minikube start or Sudo minikube start

Wait until it is done, configured and ready to use

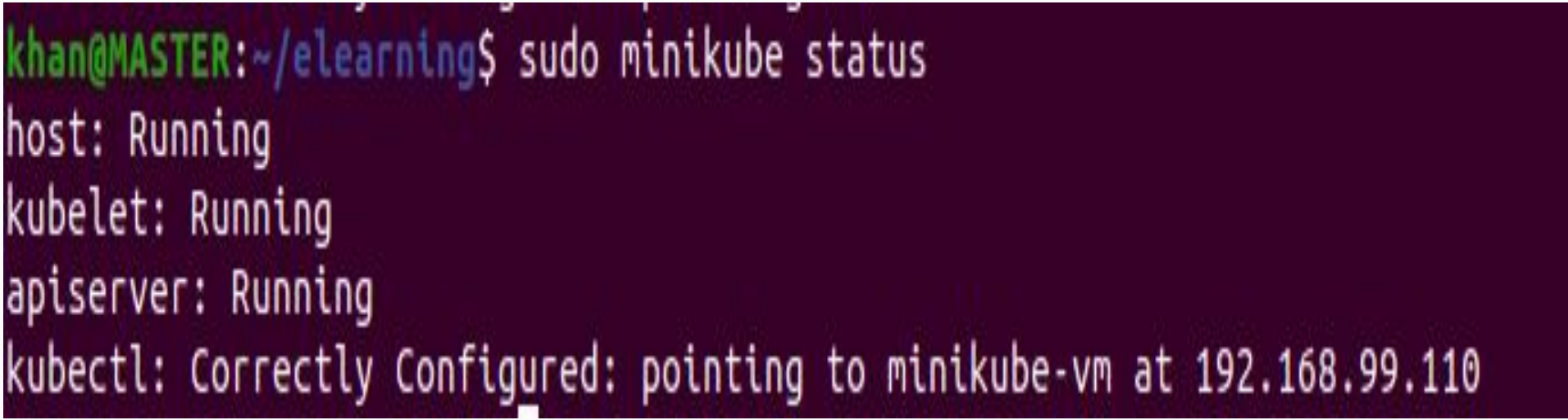


minikube status

To find the current state of minikube

Run the command

Minikube status or Sudo minikube status



kubectl cluster-info

To find the state of the kubernetes cluster in minikube

Run the command

sudo kubectl cluster-info or kubectl cluster-info

```
khan@MASTER:~/elearning$ sudo kubectl cluster-info
Kubernetes master is running at https://192.168.99.110:8443
KubeDNS is running at https://192.168.99.110:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
khan@MASTER:~/elearning$
```

Nodes

Nodes

- A group of servers is called a cluster
- A K8s cluster consists of master and worker nodes
- The master node acts as a manager
- Worker nodes are servers or systems on which K8s deploys and runs applications

kubectl get nodes

To list nodes in a kubernetes cluster

```
IOSs-MacBook-Pro:~ adil$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   12m   v1.15.0
IOSs-MacBook-Pro:~ adil$
IOSs-MacBook-Pro:~ adil$
IOSs-MacBook-Pro:~ adil$ kubectl get no
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   12m   v1.15.0
IOSs-MacBook-Pro:~ adil$ _
```

kubectl describe node <name>

To get a description of a kubernetes node

```
IOSs-MacBook-Pro:~ adil$ kubectl describe node minikube
Name:               minikube
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=minikube
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/master=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 30 Jul 2019 04:54:22 +0500
Taints:             <none>
Unschedulable:      false
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime
    Reason                  Message
  ----             ------  -----------------                 ------------------
    ------                  -------
  MemoryPressure   False   Tue, 30 Jul 2019 05:08:23 +0500   Tue, 30 Jul 2019 04:54:16 +05
00   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Tue, 30 Jul 2019 05:08:23 +0500   Tue, 30 Jul 2019 04:54:16 +05
00   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Tue, 30 Jul 2019 05:08:23 +0500   Tue, 30 Jul 2019 04:54:16 +05
00   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Tue, 30 Jul 2019 05:08:23 +0500   Tue, 30 Jul 2019 04:54:16 +05
```

Alias

Alias

- The linux alias command enables us to shorten lengthy and frequently used commands with an alias.

alias <alias>=<command>

To use an alias to shorten lengthy and frequently used kubectl commands

```
IOSs-MacBook-Pro:~ adil$ alias kgn="kubectl get no"
IOSs-MacBook-Pro:~ adil$ kgn
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   20m   v1.15.0
IOSs-MacBook-Pro:~ adil$
IOSs-MacBook-Pro:~ adil$
IOSs-MacBook-Pro:~ adil$ alias kdn="kubectl describe node"
IOSs-MacBook-Pro:~ adil$ kdn minikube
Name:               minikube
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=minikube
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/master=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 30 Jul 2019 04:54:22 +0500
Taints:             <none>
Unschedulable:      false
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime
     Reason                         Message
  ----             ------  -----------------                 ------------------
     ------                         -------
```

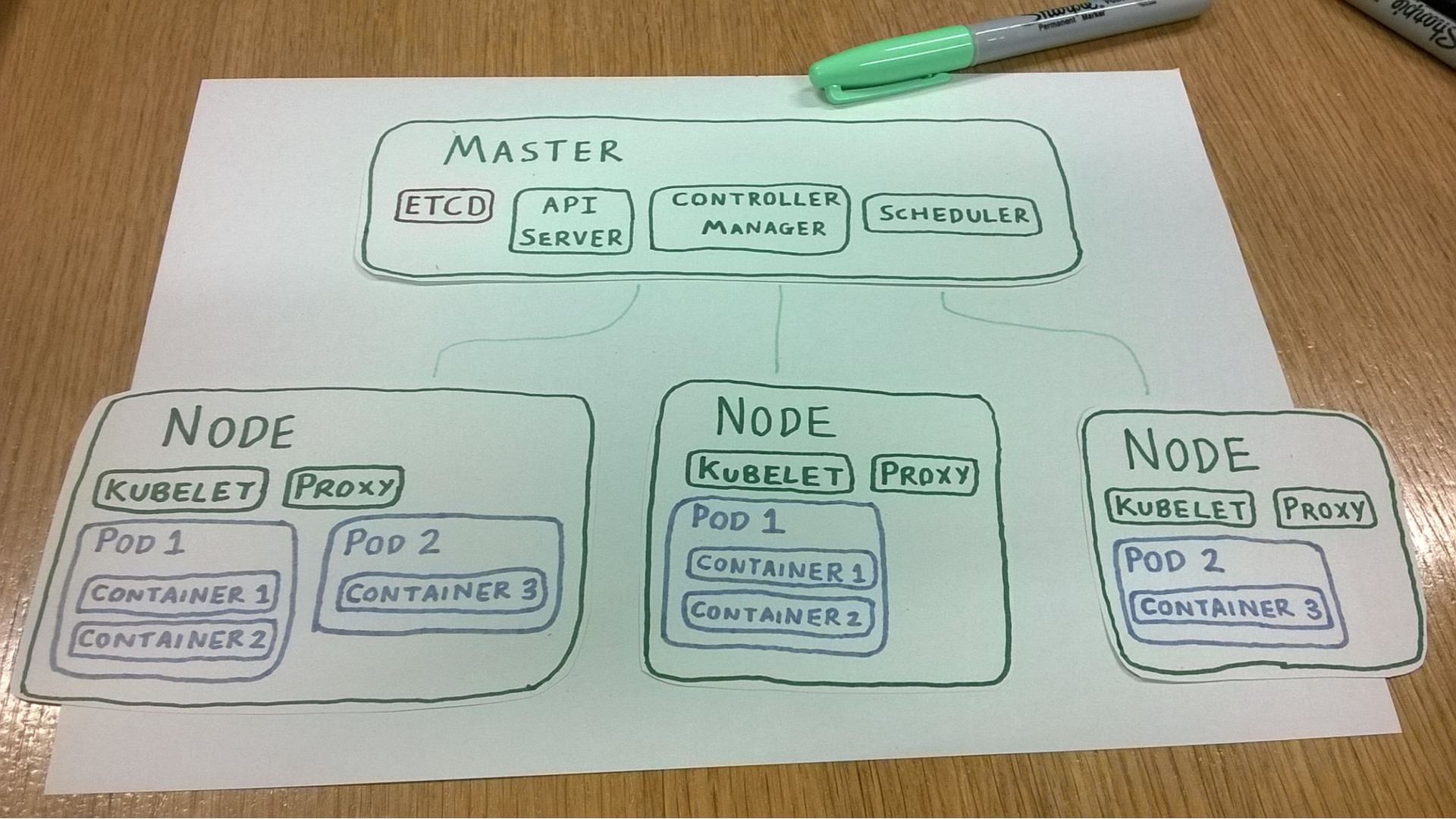
Pods

Pods

- To run an application on K8s, the most basic condition is the **application must be containerized**
- K8s does not deploy the application's container directly like Docker
- K8s wraps the application container or group of containers together into a **Pod**

Pods

- Each pod is like a separate logical machine with its own IP, hostname, processes, and so on, running a single application.


MASTER
ETCD
API SERVER
CONTROLLER MANAGER
SCHEDULER
NODE
KUBELET
PROXY
POD 1
CONTAINER 1
CONTAINER 2
POD 2
CONTAINER 3
NODE
KUBELET
PROXY
POD 1
CONTAINER 1
CONTAINER 2
NODE
KUBELET
PROXY
POD 2
CONTAINER 3

Pods

- All the containers in a pod will appear to be running on the same logical machine.
- A Pod with multiple containers will always run on the same worker node.

Why Pods?

-