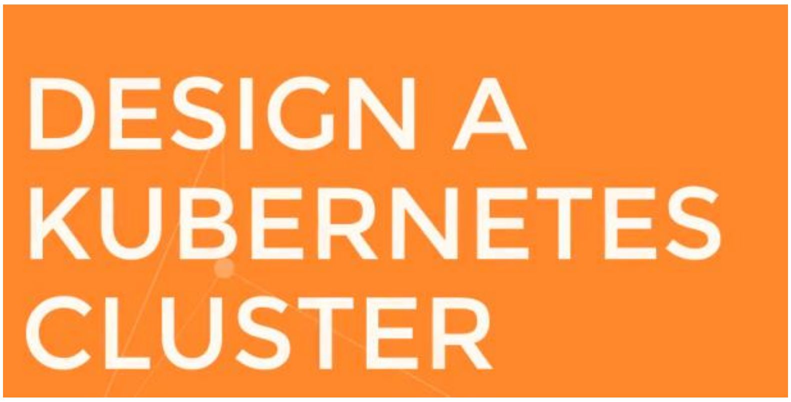
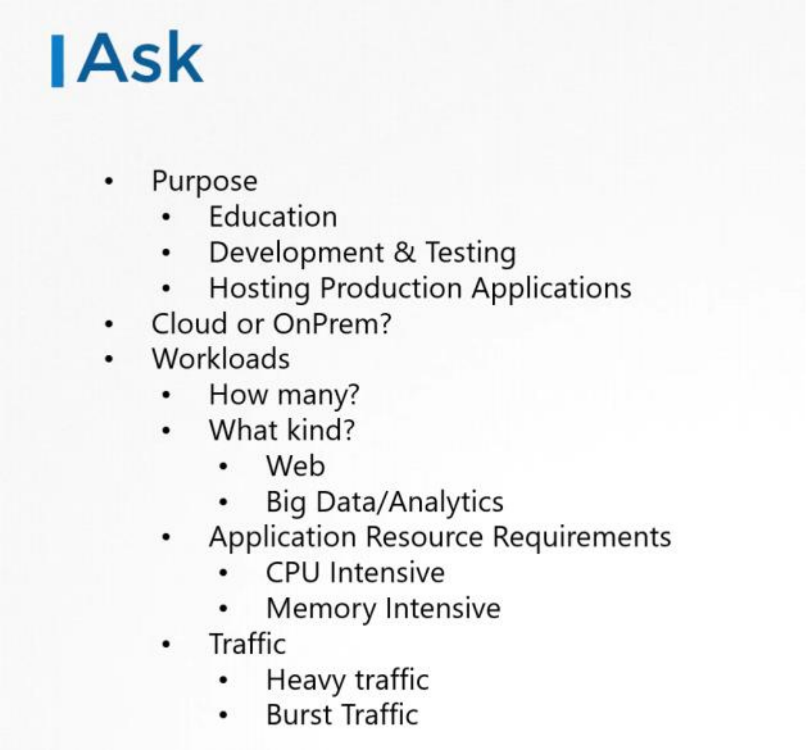
Sunday, 13 November 2022 12.59

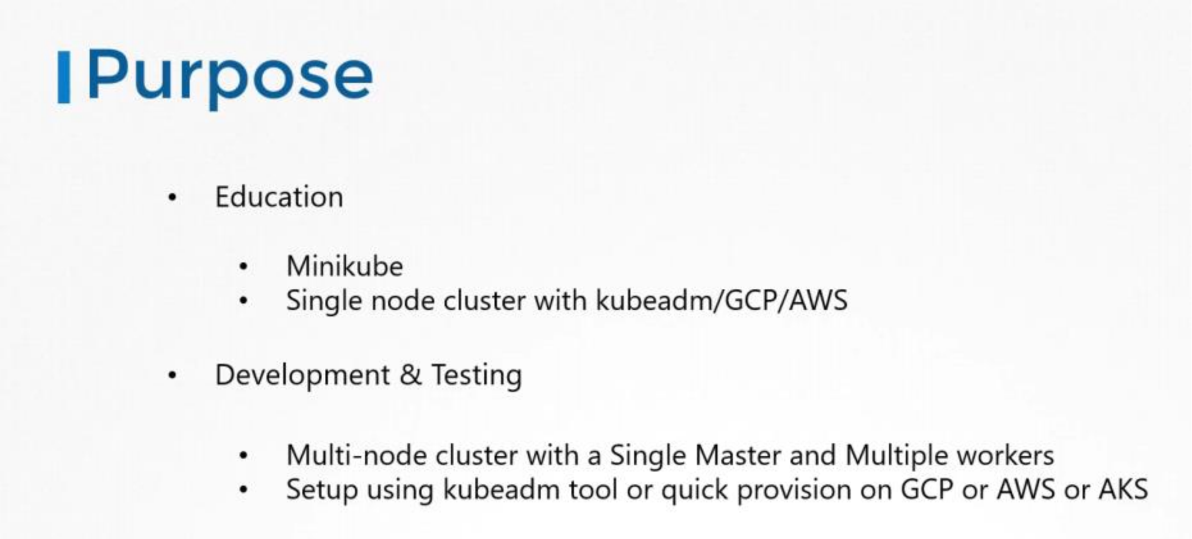


DESIGN A KUBERNETES CLUSTER



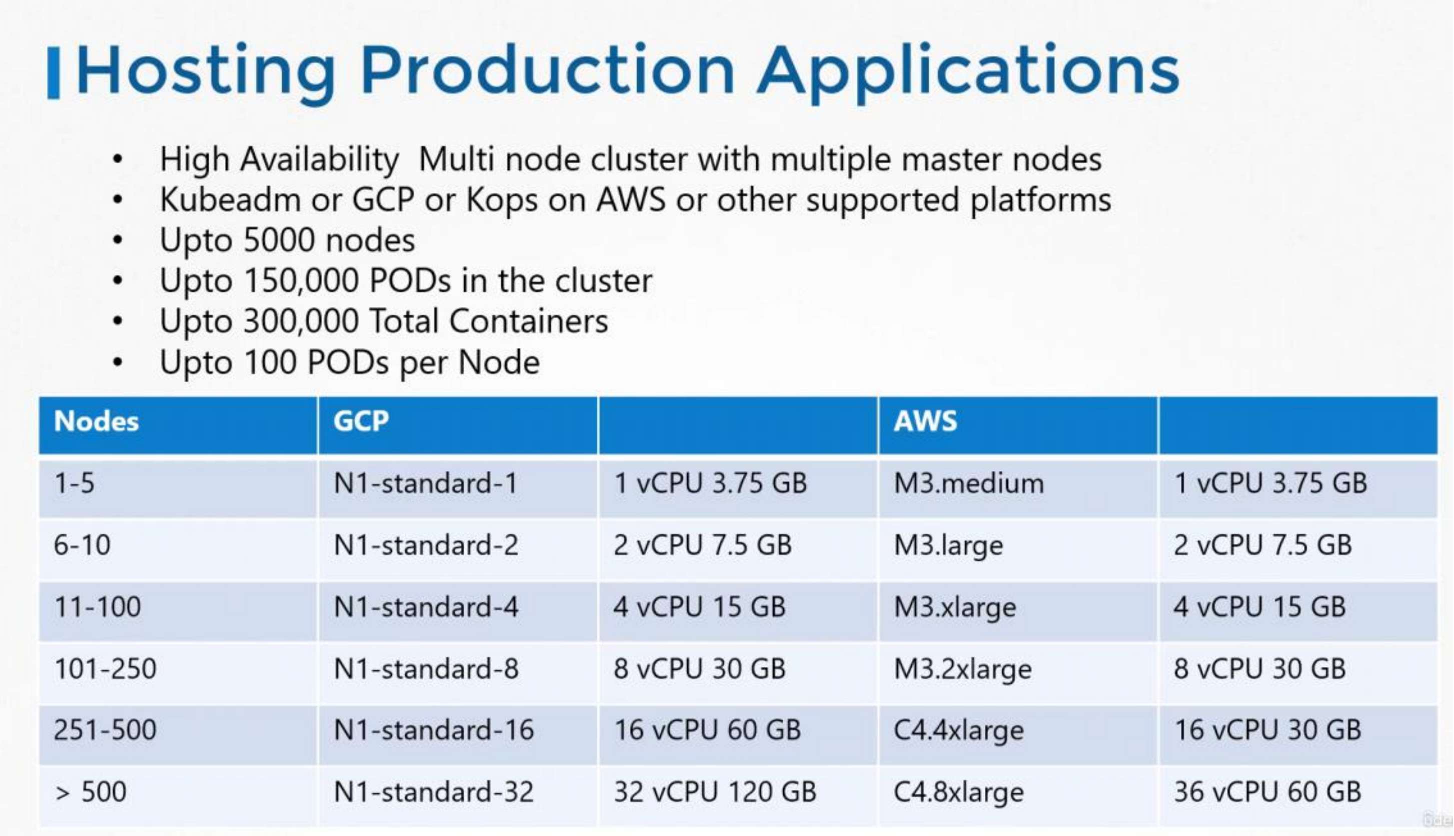
Ask

- Purpose
  - Education
  - Development & Testing
  - Hosting Production Applications
- Cloud or OnPrem?
- Workloads
  - How many?
  - What kind?
    - Web
    - Big Data/Analytics
  - Application Resource Requirements
    - CPU Intensive
    - Memory Intensive
  - Traffic
    - Heavy traffic
    - Burst Traffic



Purpose

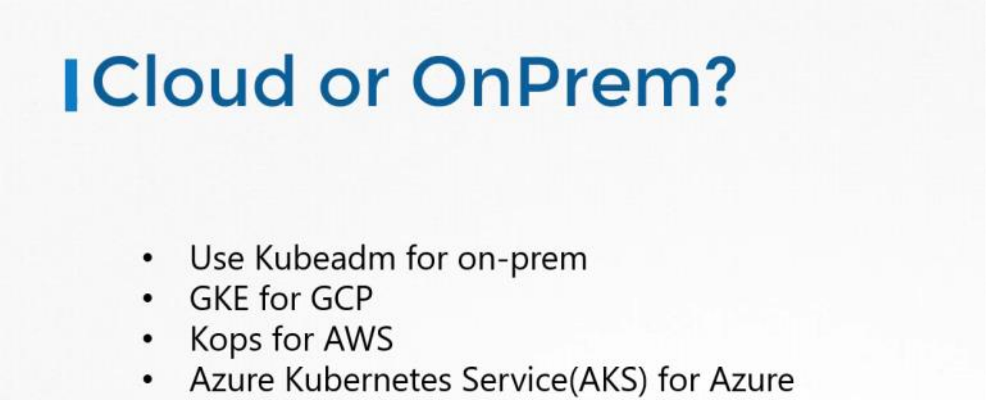
- Education
  - Minikube
  - Single node cluster with kubeadm/GCP/AWS
- Development & Testing
  - Multi-node cluster with a Single Master and Multiple workers
  - Setup using kubeadm tool or quick provision on GCP or AWS or AKS



Hosting Production Applications

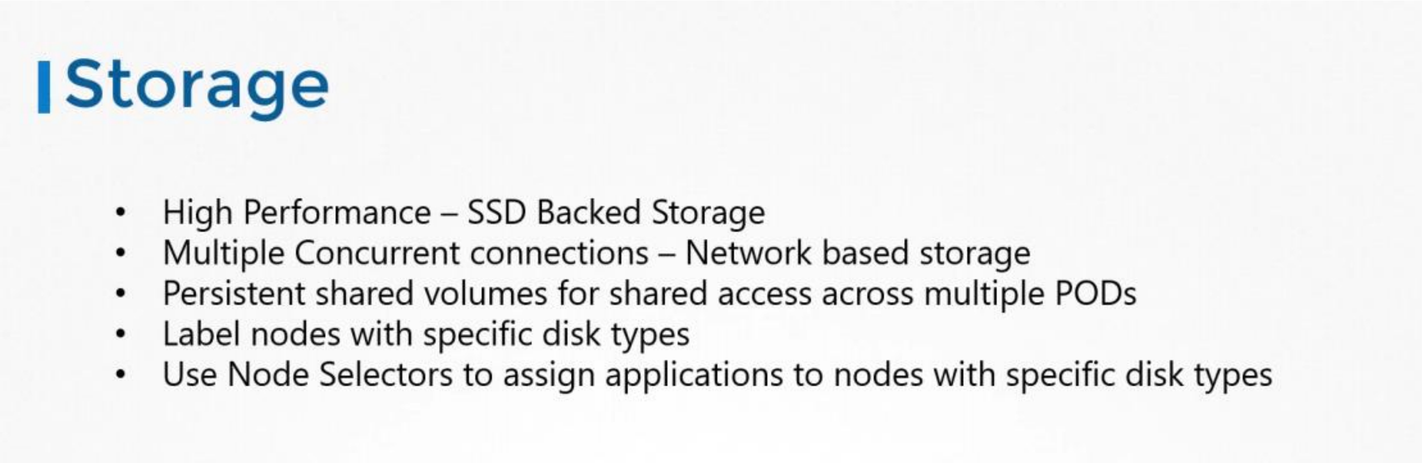
- High Availability Multi node cluster with multiple master nodes
- Kubeadm or GCP or Kops on AWS or other supported platforms
- Upto 5000 nodes
- Upto 150,000 PODs in the cluster
- Upto 300,000 Total Containers
- Upto 100 PODs per Node

Nodes	GCP		AWS	
1-5	N1-standard-1	1 vCPU 3.75 GB	M3.medium	1 vCPU 3.75 GB
6-10	N1-standard-2	2 vCPU 7.5 GB	M3.large	2 vCPU 7.5 GB
11-100	N1-standard-4	4 vCPU 15 GB	M3.xlarge	4 vCPU 15 GB
101-250	N1-standard-8	8 vCPU 30 GB	M3.2xlarge	8 vCPU 30 GB
251-500	N1-standard-16	16 vCPU 60 GB	C4.4xlarge	16 vCPU 30 GB
> 500	N1-standard-32	32 vCPU 120 GB	C4.8xlarge	36 vCPU 60 GB



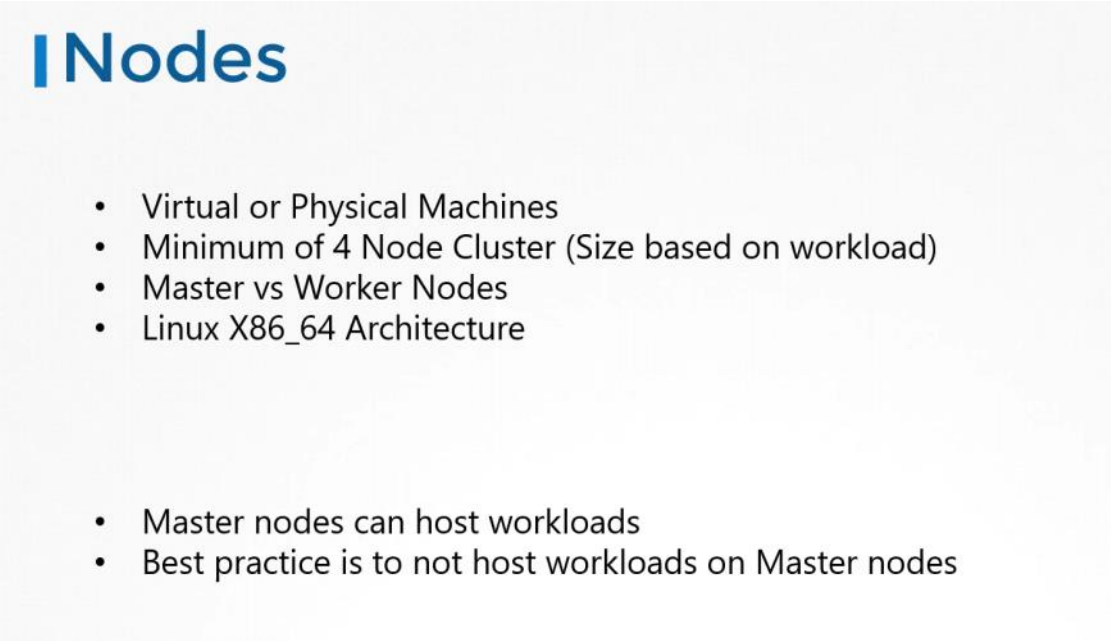
Cloud or OnPrem?

- Use Kubeadm for on-prem
- GKE for GCP
- Kops for AWS
- Azure Kubernetes Service(AKS) for Azure



Storage

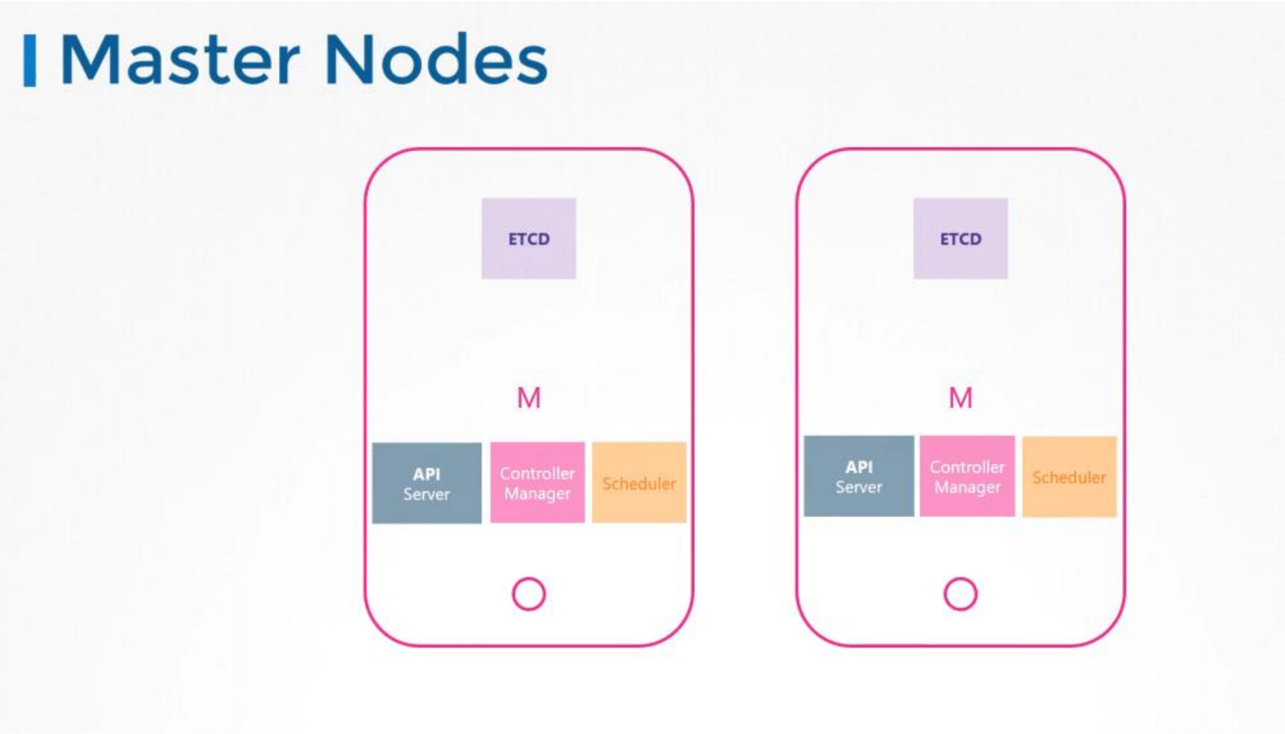
- High Performance – SSD Backed Storage
- Multiple Concurrent connections – Network based storage
- Persistent shared volumes for shared access across multiple PODs
- Label nodes with specific disk types
- Use Node Selectors to assign applications to nodes with specific disk types



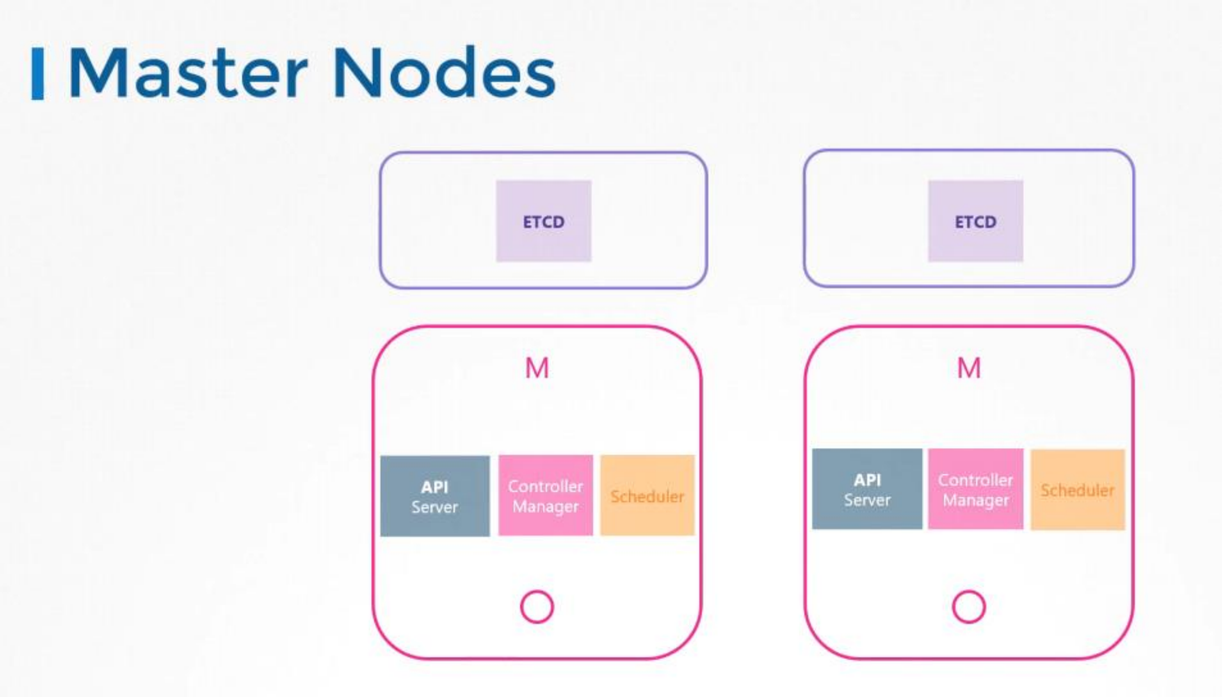
Nodes

- Virtual or Physical Machines
- Minimum of 4 Node Cluster (Size based on workload)
- Master vs Worker Nodes
- Linux X86_64 Architecture

- Master nodes can host workloads
- Best practice is to not host workloads on Master nodes

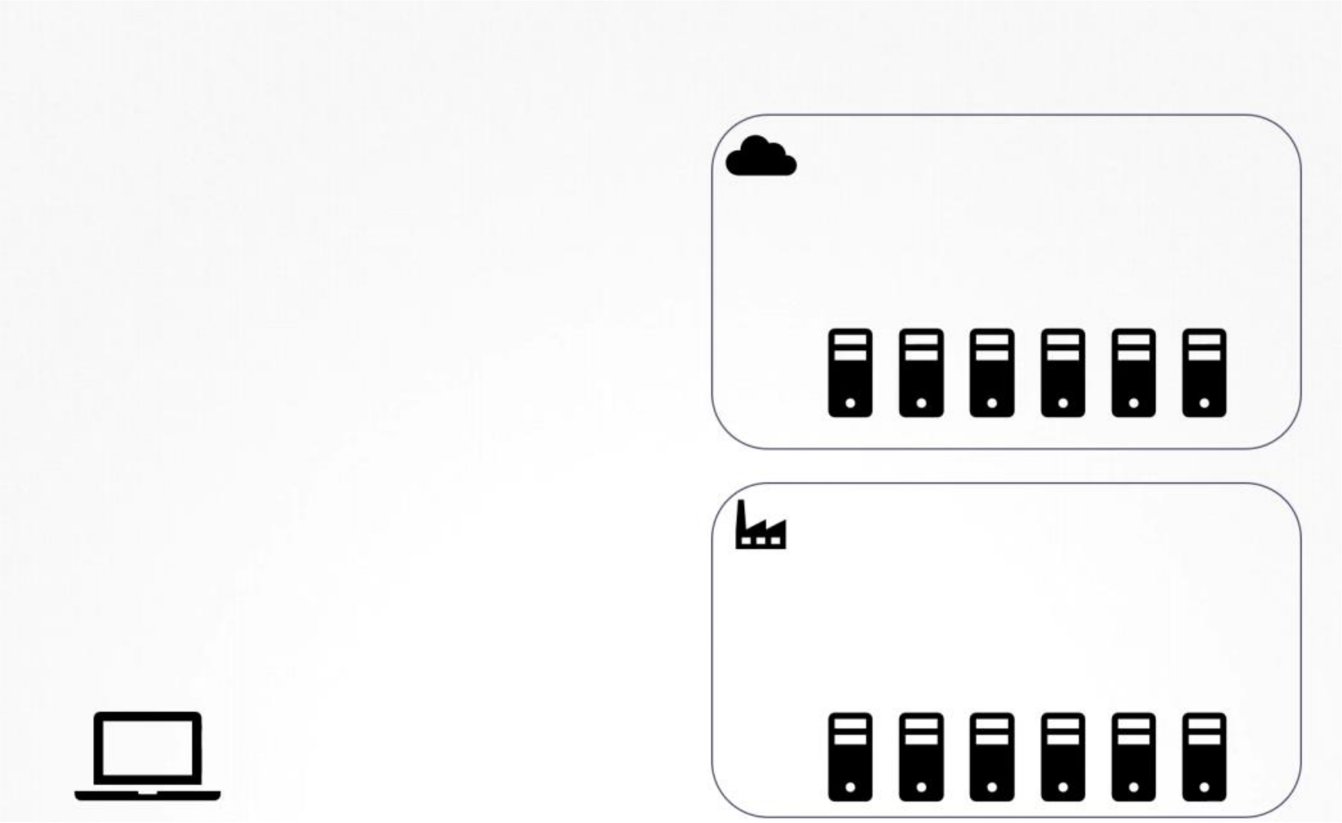


Master Nodes

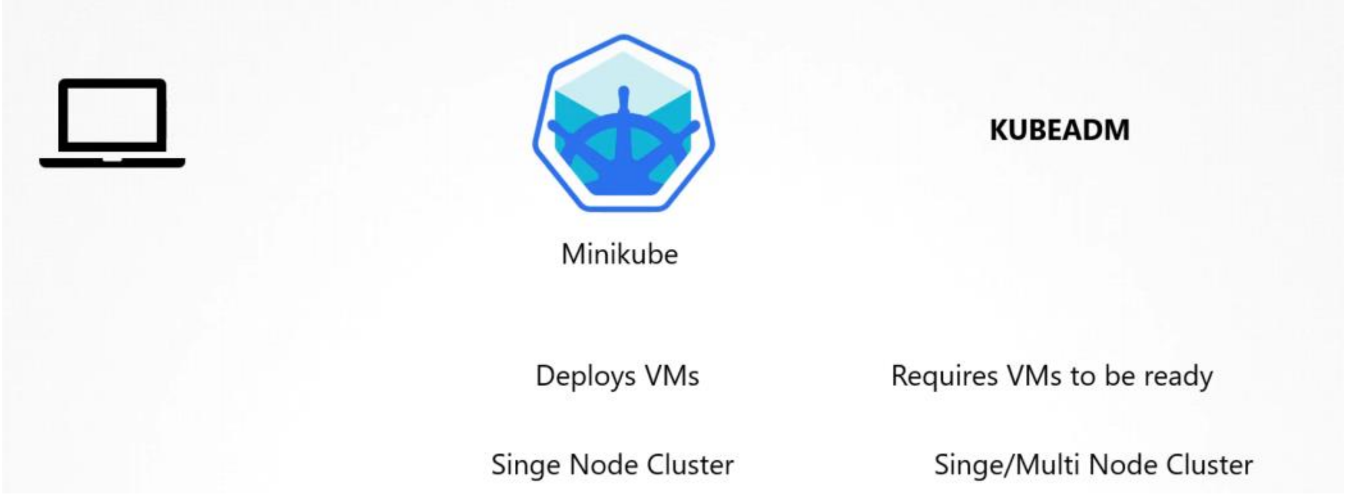


Master Nodes

Choosing Kubernetes Infrastructure



Linux | Windows



Minikube

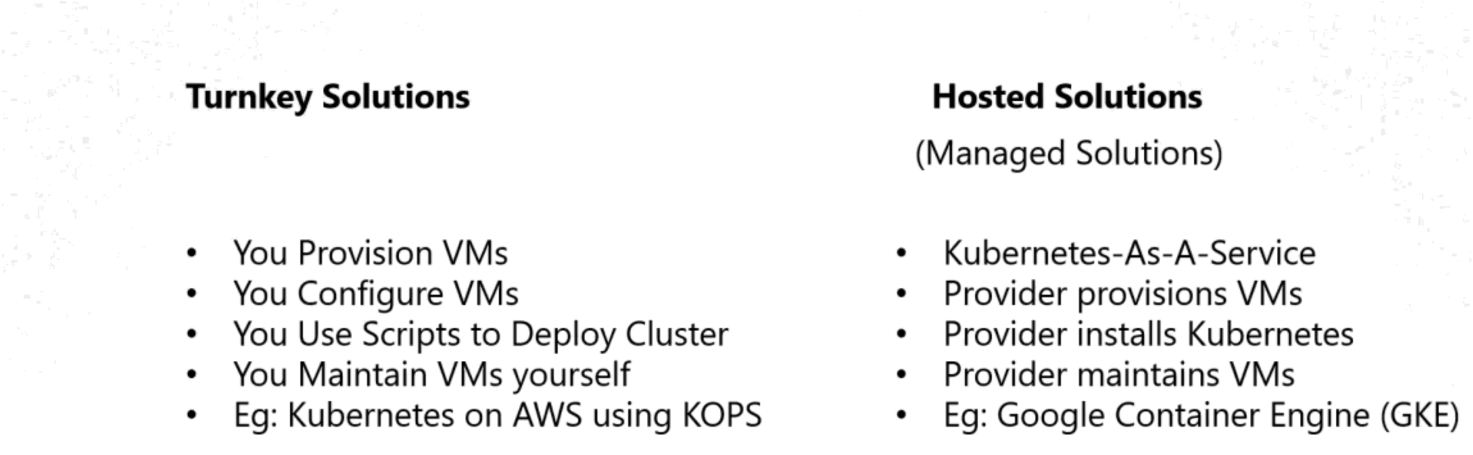
KUBEADM

Deploys VMs

Requires VMs to be ready

Singe Node Cluster

Singe/Multi Node Cluster



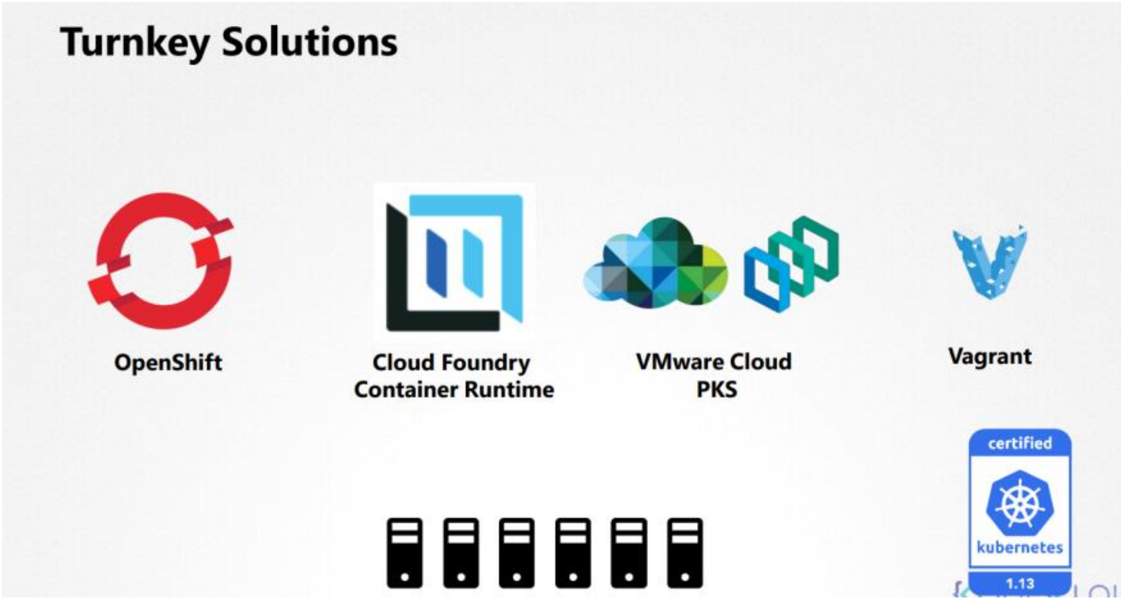
Turnkey Solutions

- You Provision VMs
- You Configure VMs
- You Use Scripts to Deploy Cluster
- You Maintain VMs yourself
- Eg: Kubernetes on AWS using KOPS

Hosted Solutions

(Managed Solutions)

- Kubernetes-As-A-Service
- Provider provisions VMs
- Provider installs Kubernetes
- Provider maintains VMs
- Eg: Google Container Engine (GKE)



Turnkey Solutions

OpenShift

Cloud Foundry Container Runtime

VMware Cloud PKS

Vagrant

certified kubernetes 1.13

Hosted Solutions

Google Container Engine **(GKE)**

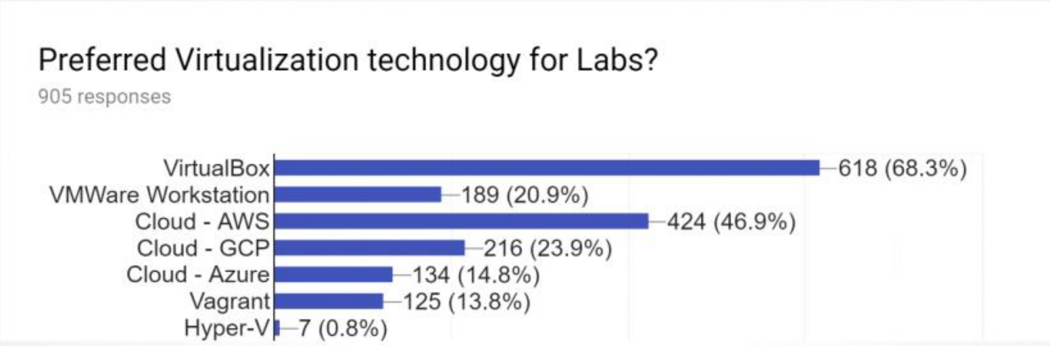
OpenShift Online

Azure Kubernetes Service

aws

Amazon Elastic Container Service for Kubernetes (EKS)

Our Choice


Preferred Virtualization technology for Labs?

905 responses

Chose a Networking Solution

cisco

weaveworks

flannel

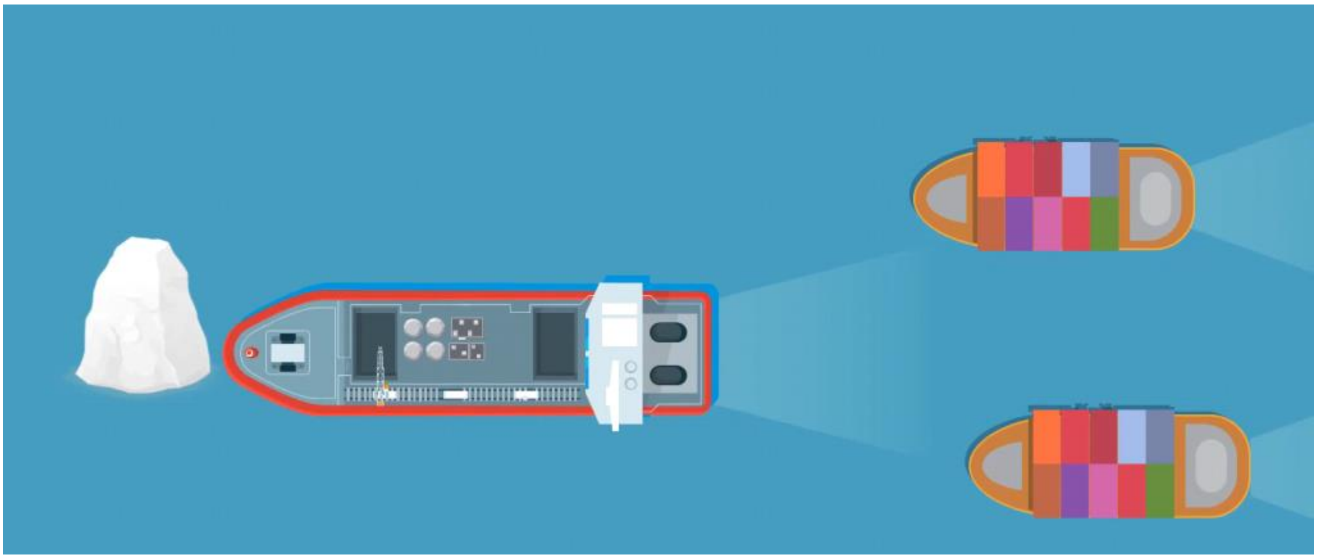
cilium

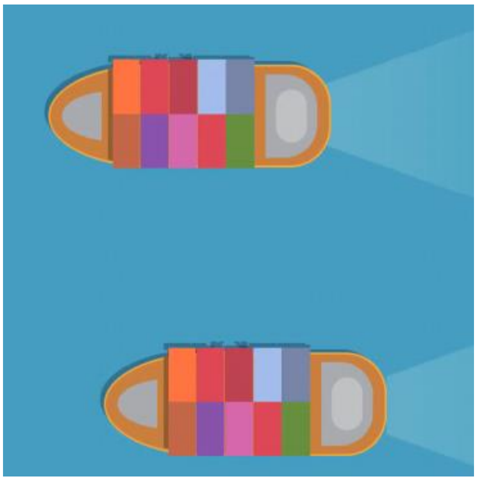
vmware NSX

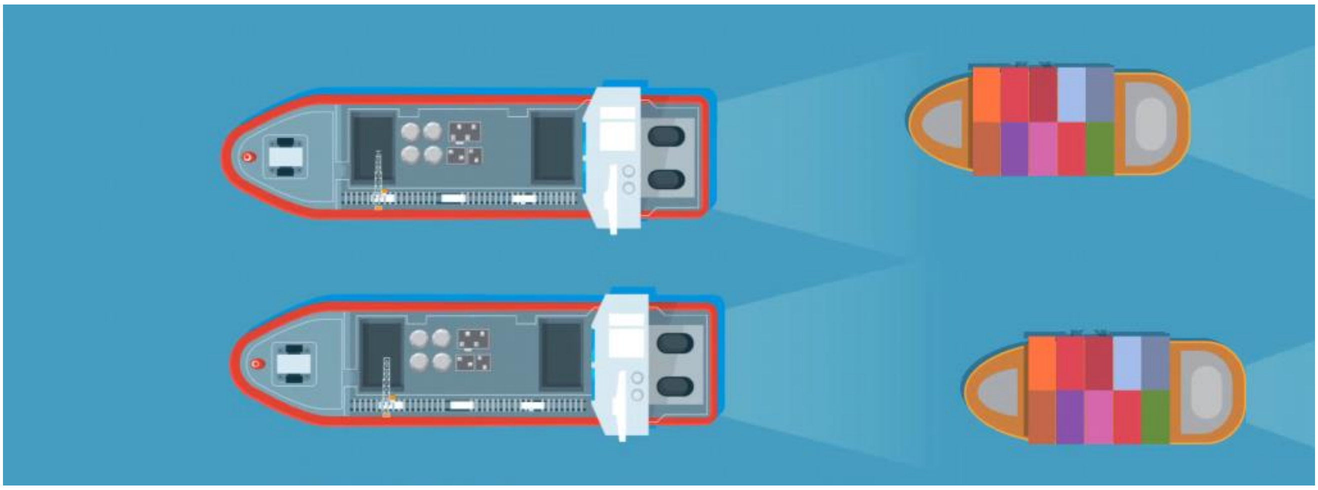
GCE

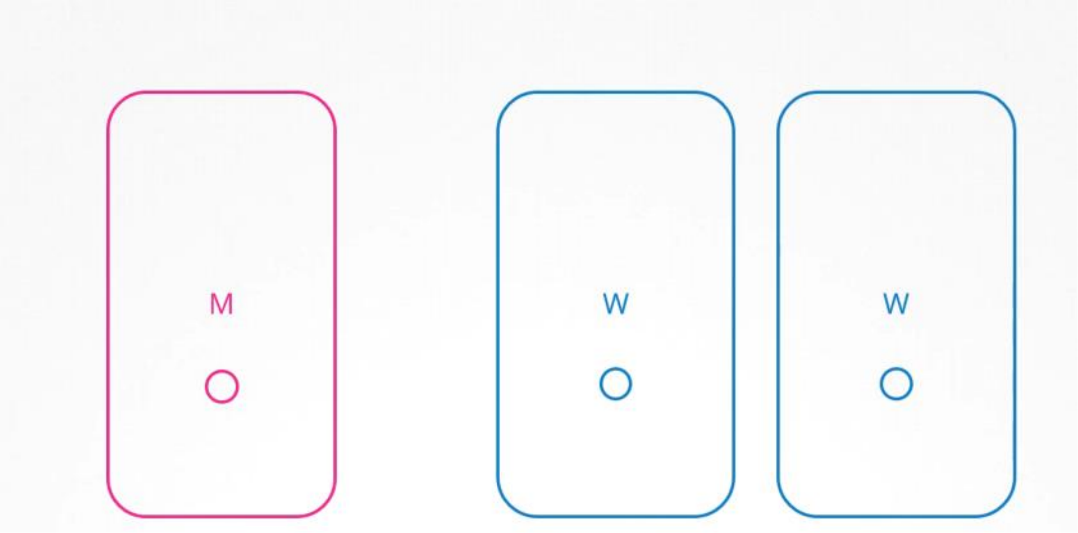
nuagenetworks

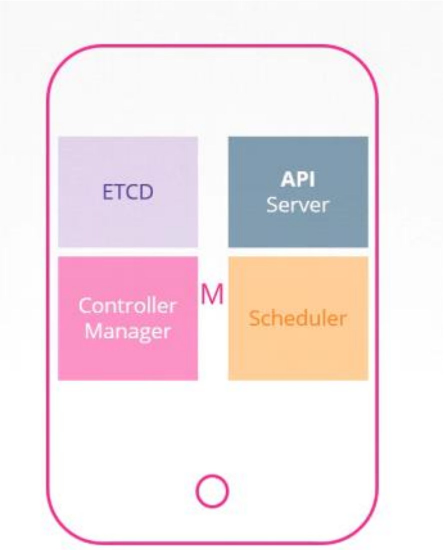
HA Kubernetes Cluster

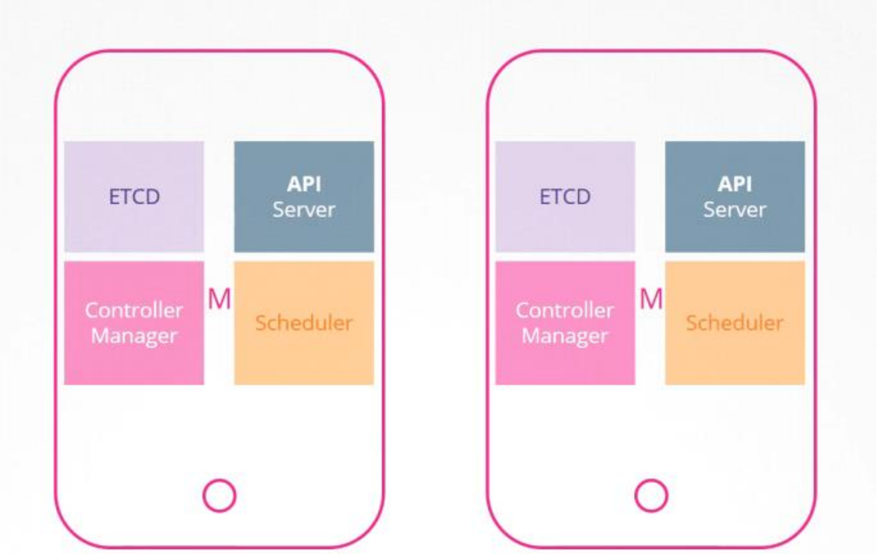


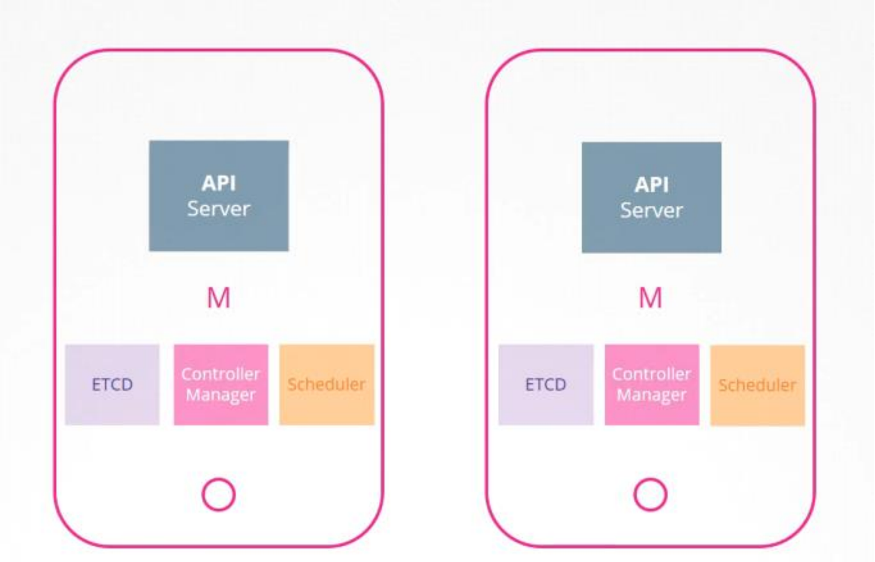


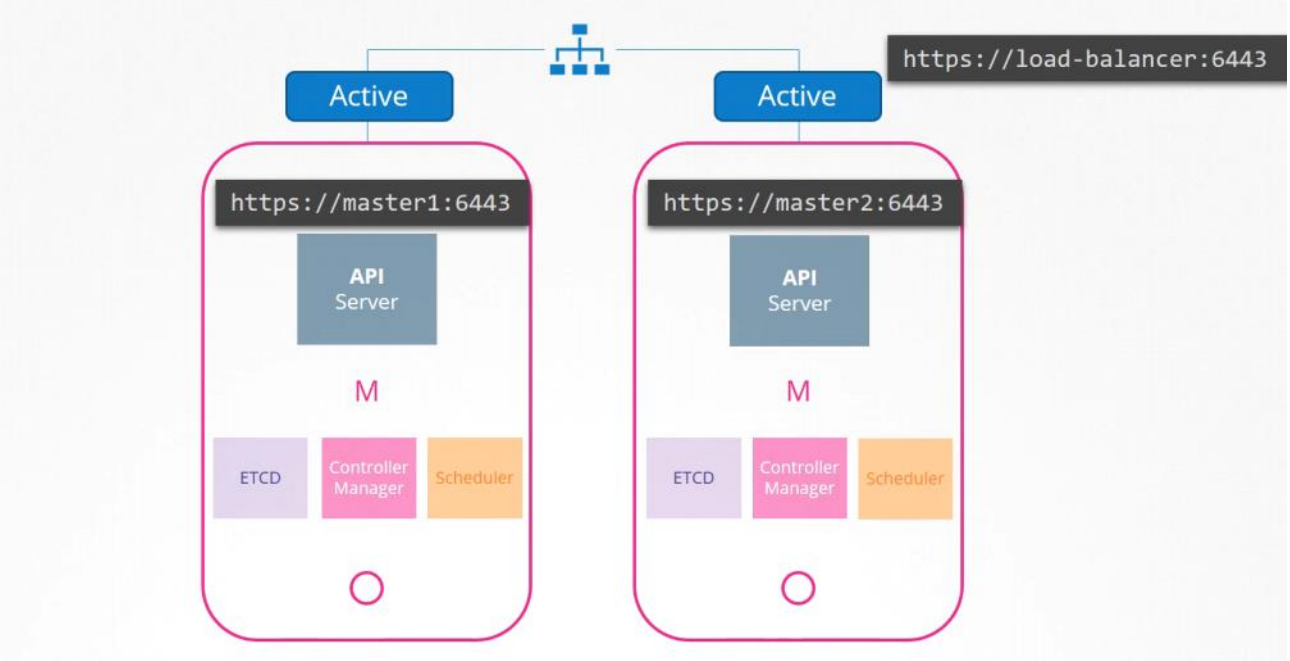


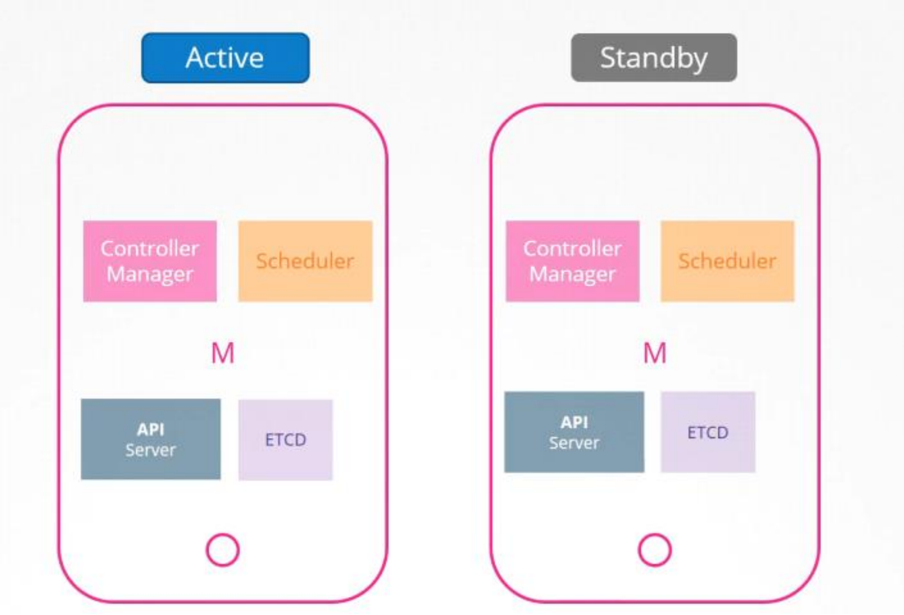

M
W
W

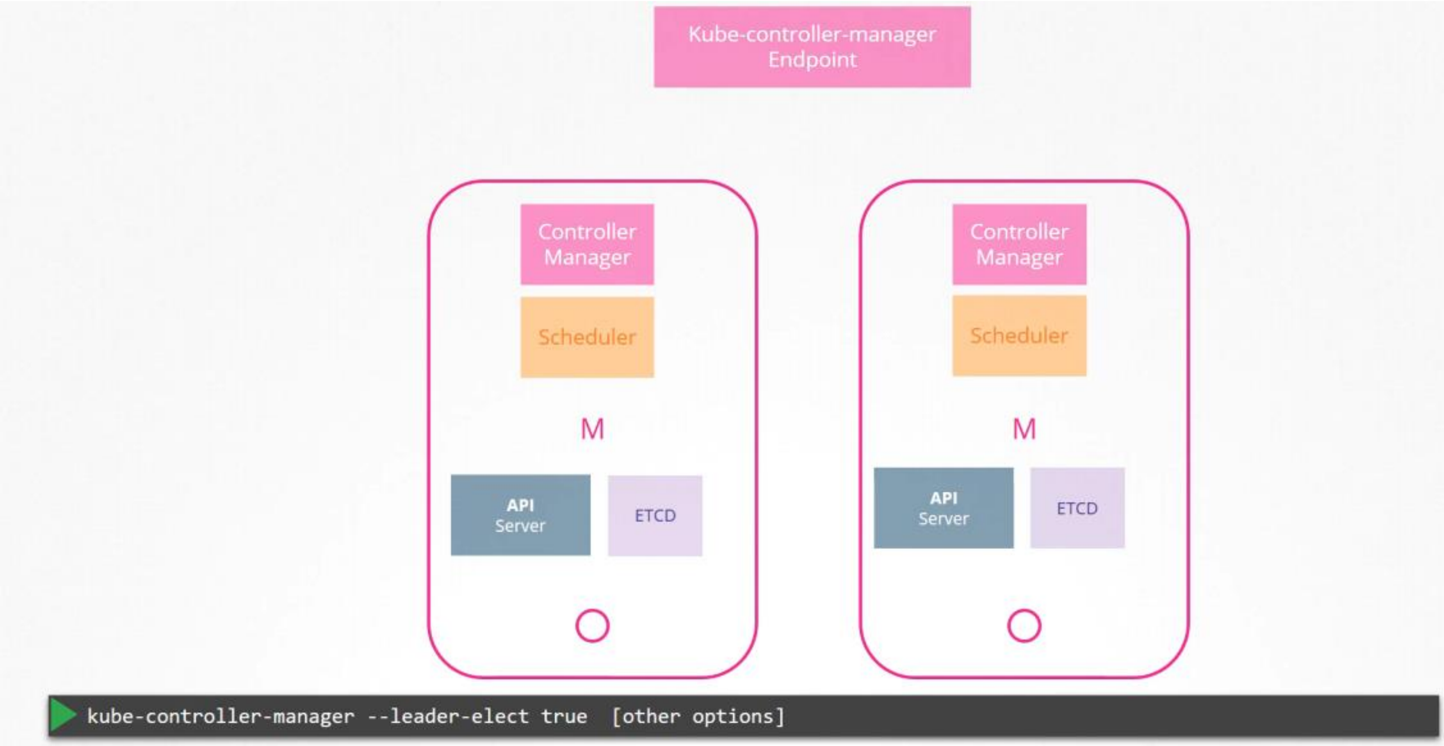

ETCD
API Server
Controller Manager
M
Scheduler


ETCD
API Server
Controller Manager
M
Scheduler
ETCD
API Server
Controller Manager
M
Scheduler

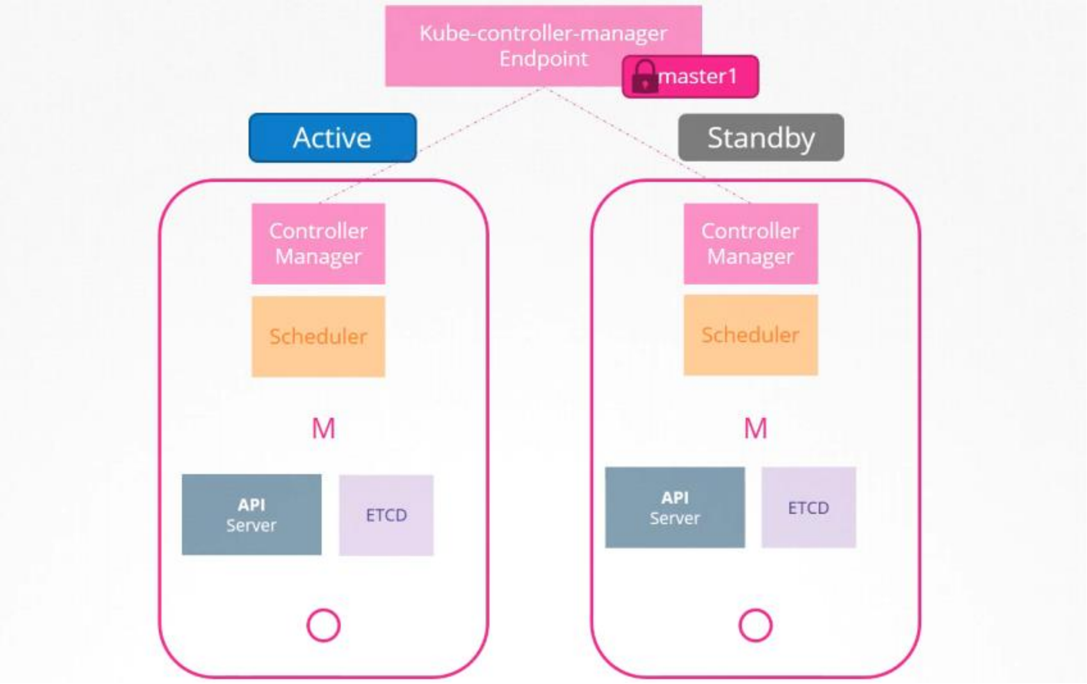

API Server
M
ETCD
Controller Manager
Scheduler
API Server
M
ETCD
Controller Manager
Scheduler

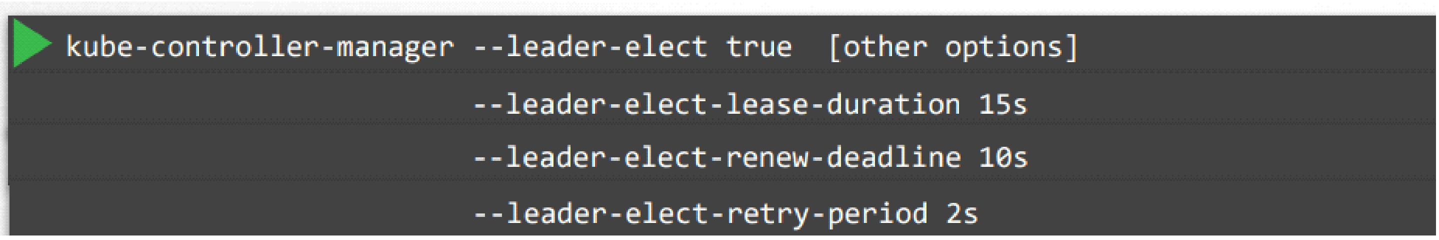

https://load-balancer:6443
Active
Active
https://master1:6443
API Server
M
ETCD
Controller Manager
Scheduler
https://master2:6443
API Server
M
ETCD
Controller Manager
Scheduler


Active
Standby
Controller Manager
Scheduler
M
API Server
ETCD
Controller Manager
Scheduler
M
API Server
ETCD

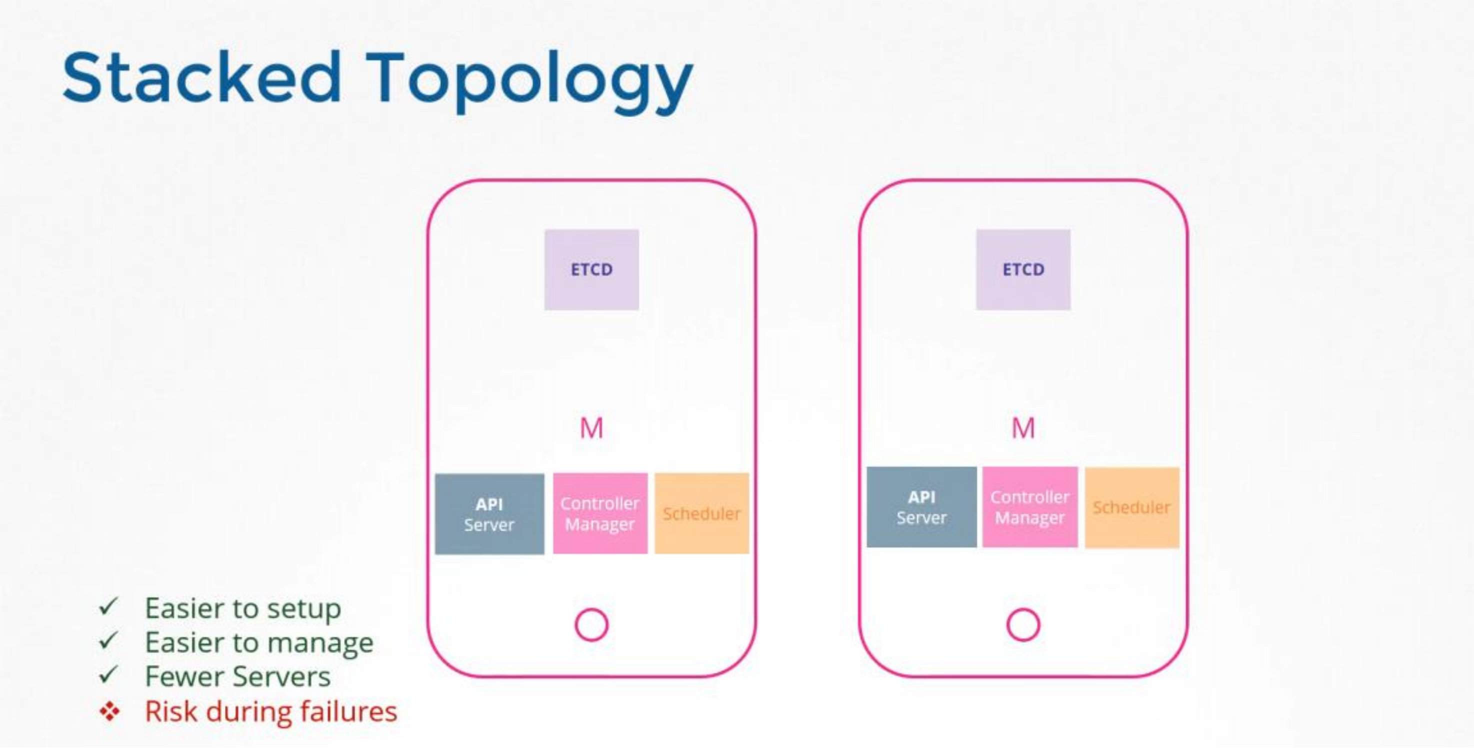


```
kube-controller-manager --leader-elect true  [other options]
```





```
kube-controller-manager --leader-elect true  [other options]
                        --leader-elect-lease-duration 15s
                        --leader-elect-renew-deadline 10s
                        --leader-elect-retry-period 2s
```



Stacked Topology

ETCD

M

API Server

Controller Manager

Scheduler

ETCD

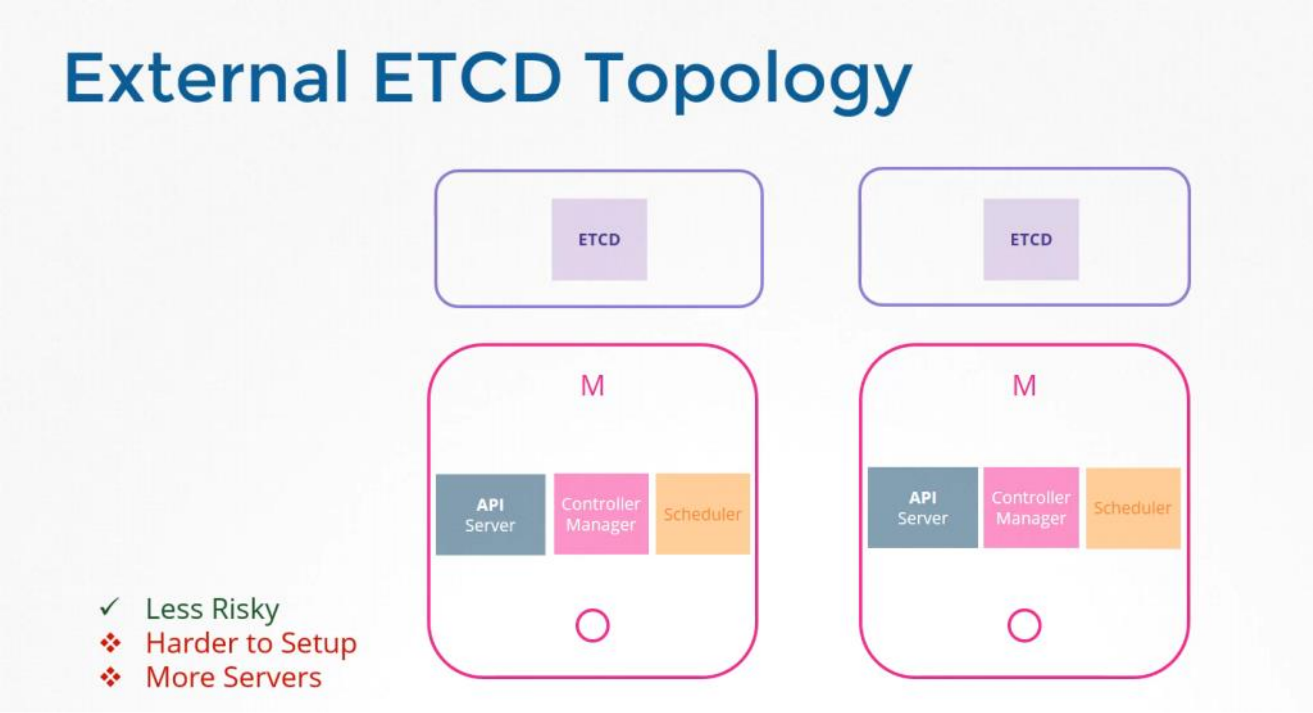
M

API Server

Controller Manager

Scheduler

- ✓ Easier to setup
- ✓ Easier to manage
- ✓ Fewer Servers
- ❖ Risk during failures



External ETCD Topology

ETCD

ETCD

M

API Server

Controller Manager

Scheduler

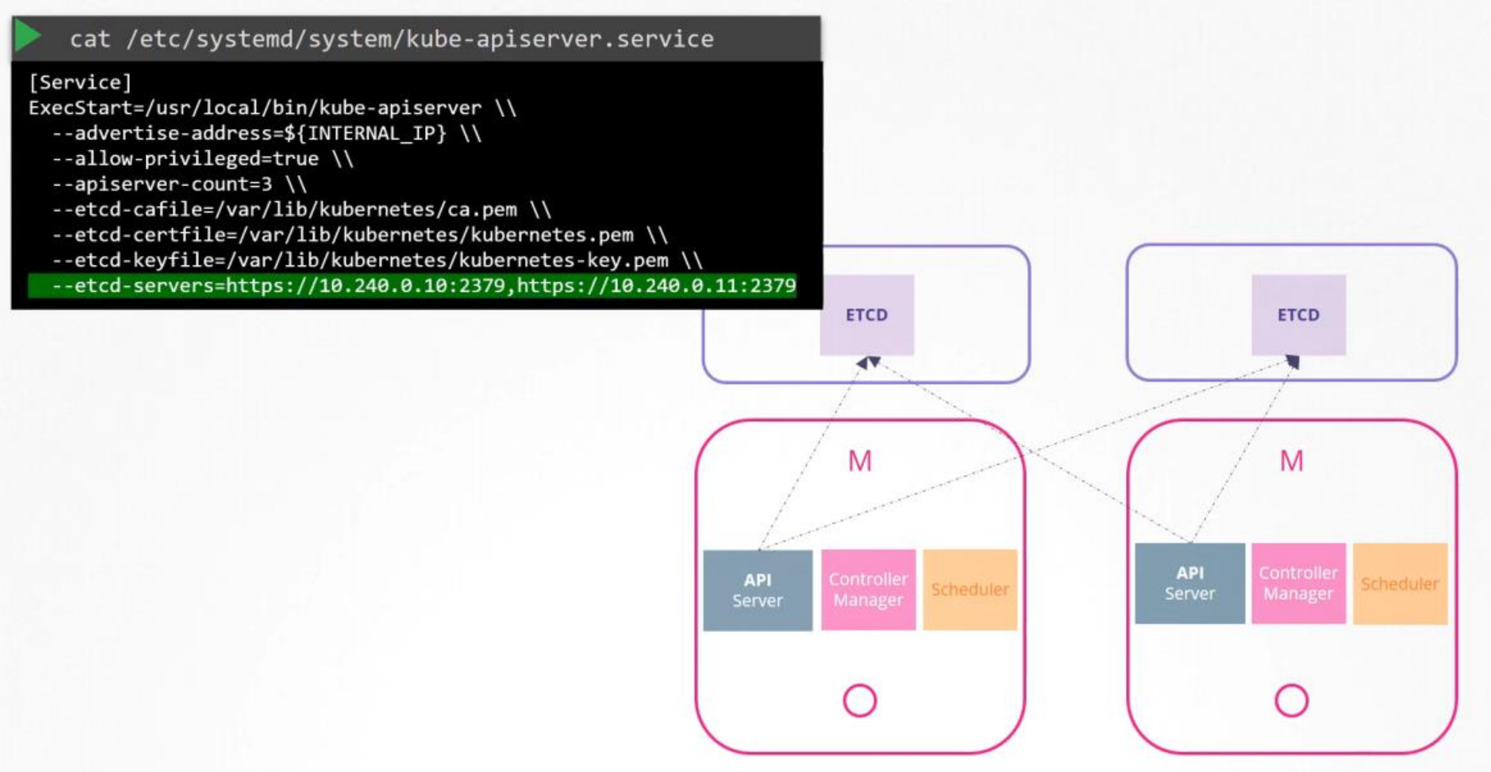
M

API Server

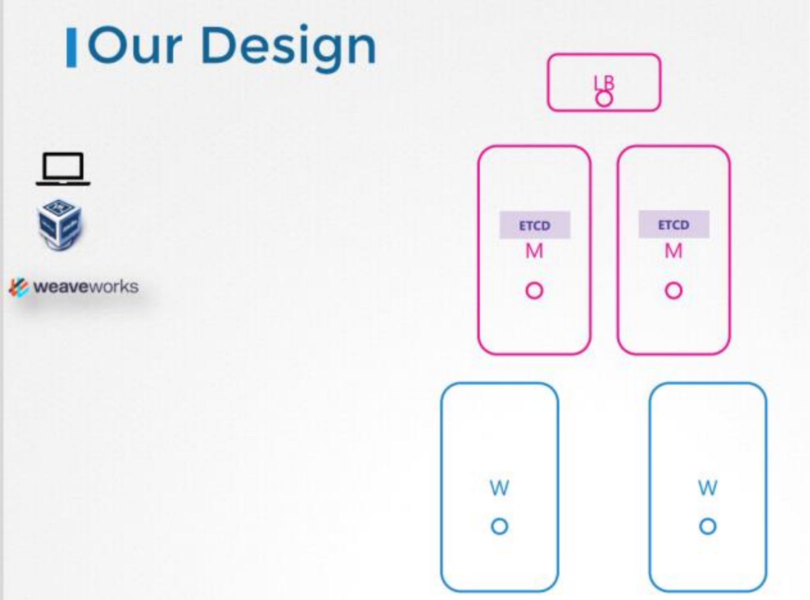
Controller Manager

Scheduler

- ✓ Less Risky
- ❖ Harder to Setup
- ❖ More Servers



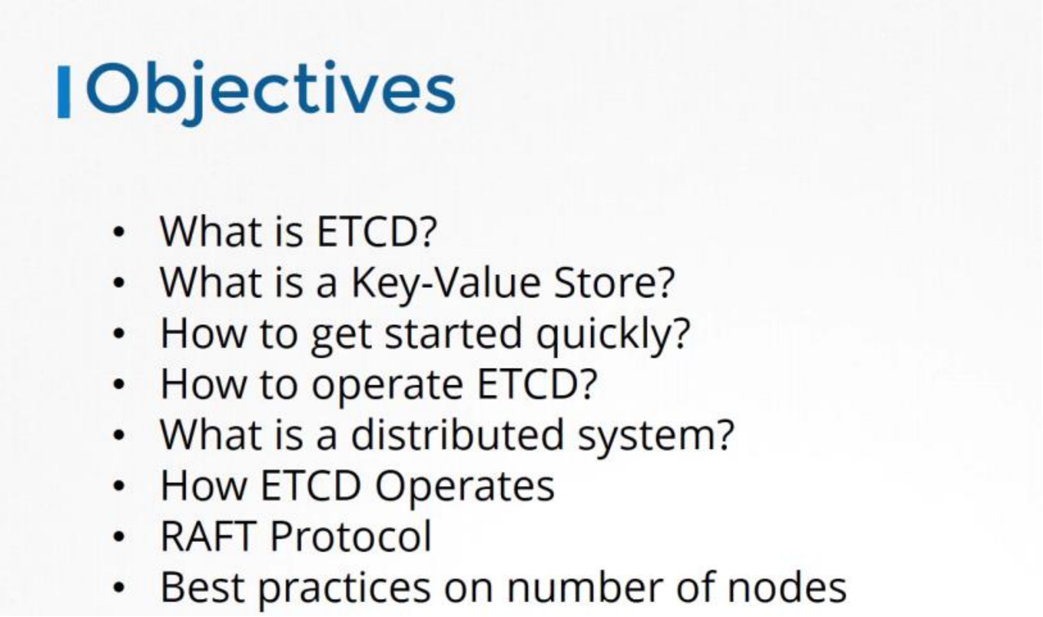
```
cat /etc/systemd/system/kube-apiserver.service
[Service]
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\
  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\
  --etcd-servers=https://10.240.0.10:2379,https://10.240.0.11:2379
```



Our Design

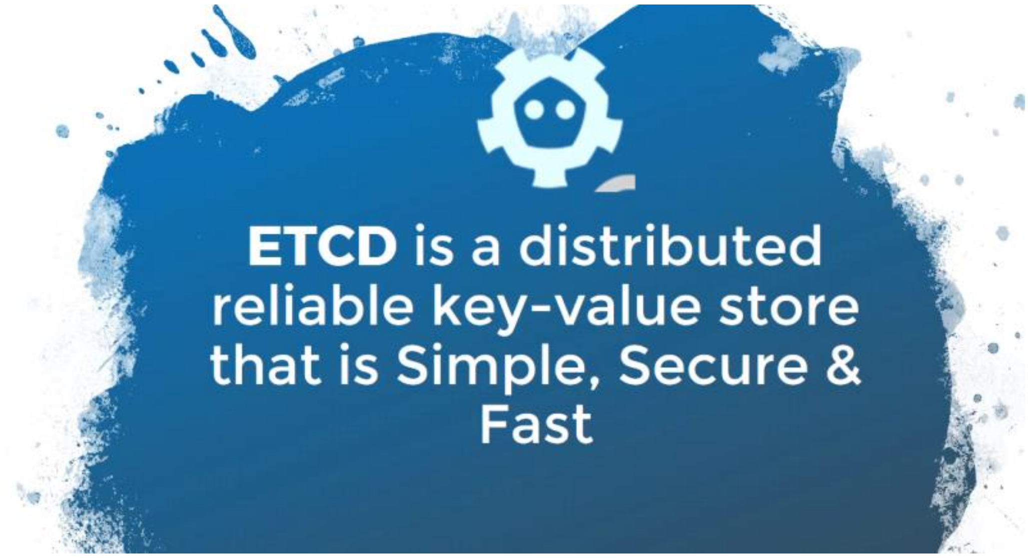
ETCD

In HA

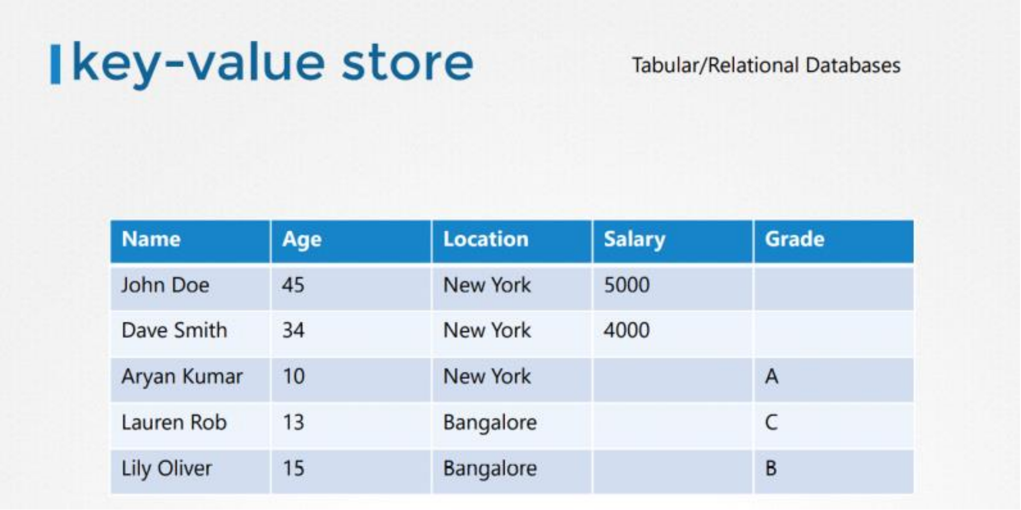


Objectives

- What is ETCD?
- What is a Key-Value Store?
- How to get started quickly?
- How to operate ETCD?
- What is a distributed system?
- How ETCD Operates
- RAFT Protocol
- Best practices on number of nodes



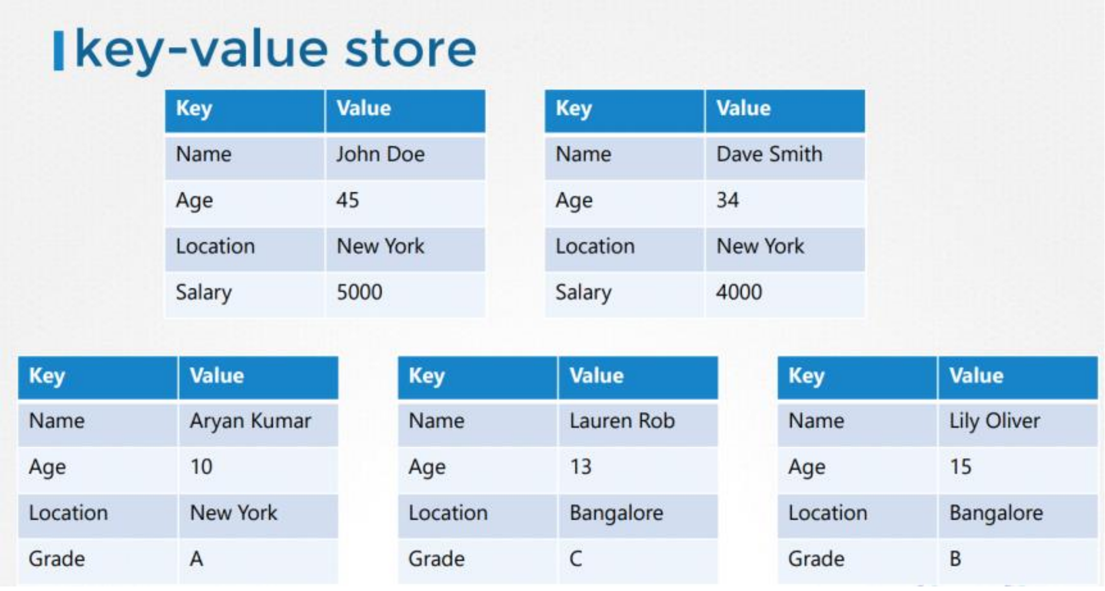
ETCD is a distributed reliable key-value store that is Simple, Secure & Fast



key-value store

Tabular/Relational Databases

Name	Age	Location	Salary	Grade
John Doe	45	New York	5000	
Dave Smith	34	New York	4000	
Aryan Kumar	10	New York		A
Lauren Rob	13	Bangalore		C
Lily Oliver	15	Bangalore		B



key-value store

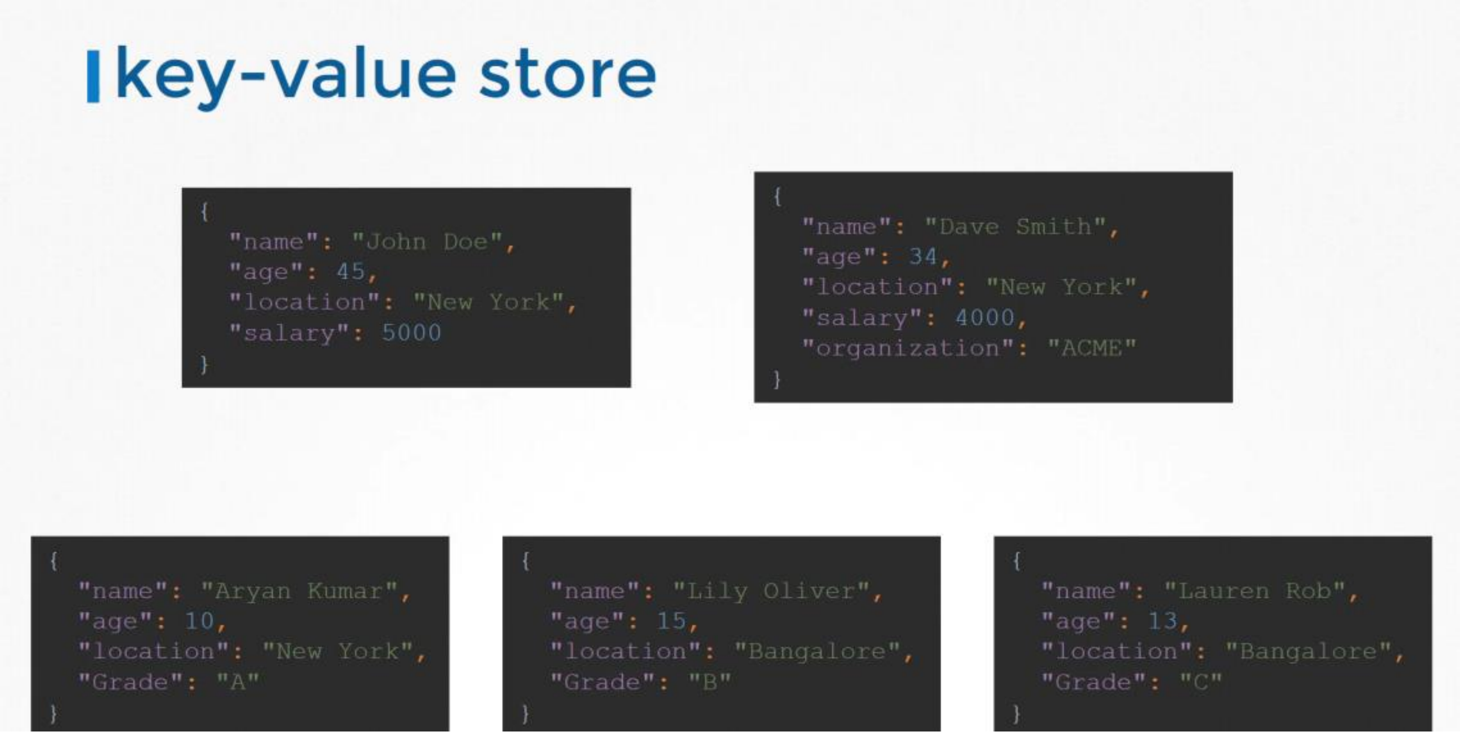
Key	Value
Name	John Doe
Age	45
Location	New York
Salary	5000

Key	Value
Name	Dave Smith
Age	34
Location	New York
Salary	4000

Key	Value
Name	Aryan Kumar
Age	10
Location	New York
Grade	A

Key	Value
Name	Lauren Rob
Age	13
Location	Bangalore
Grade	C

Key	Value
Name	Lily Oliver
Age	15
Location	Bangalore
Grade	B



key-value store

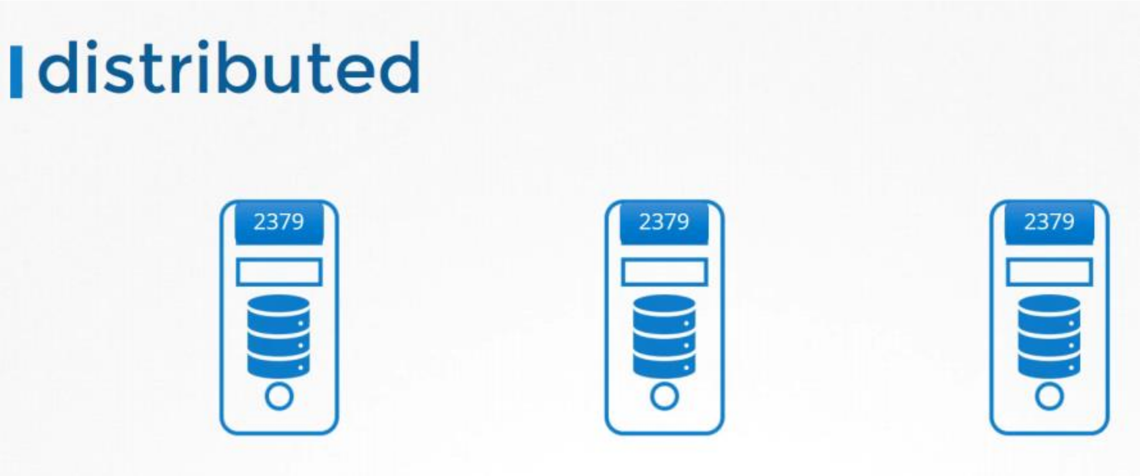
```
{
  "name": "John Doe",
  "age": 45,
  "location": "New York",
  "salary": 5000
}
```

```
{
  "name": "Dave Smith",
  "age": 34,
  "location": "New York",
  "salary": 4000,
  "organization": "ACME"
}
```

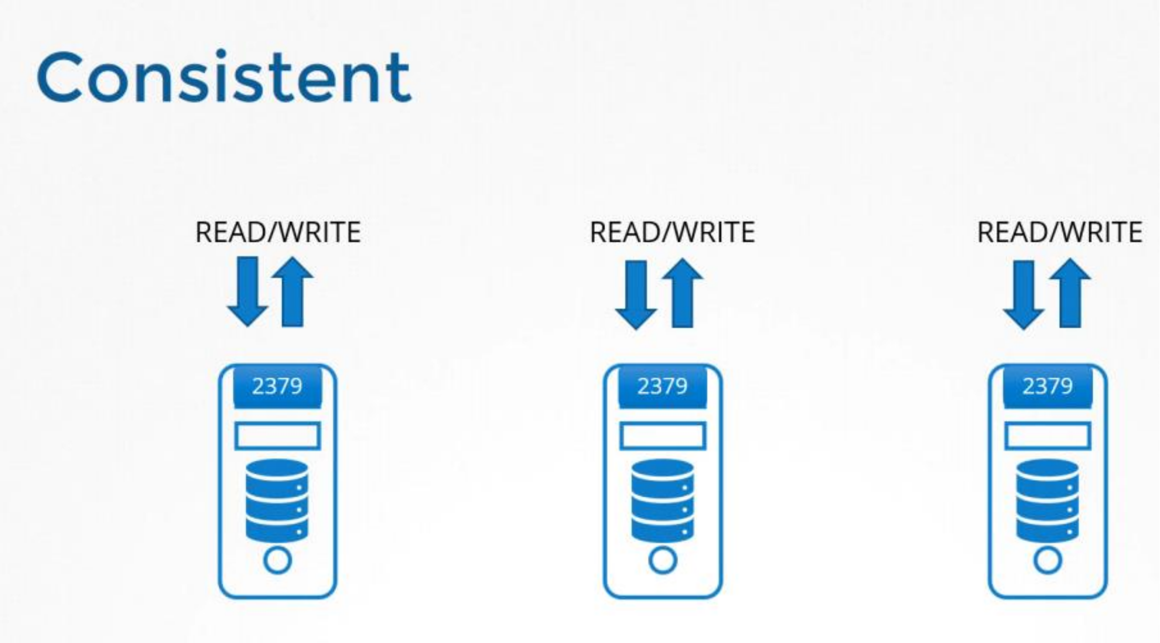
```
{
  "name": "Aryan Kumar",
  "age": 10,
  "location": "New York",
  "Grade": "A"
}
```

```
{
  "name": "Lily Oliver",
  "age": 15,
  "location": "Bangalore",
  "Grade": "B"
}
```

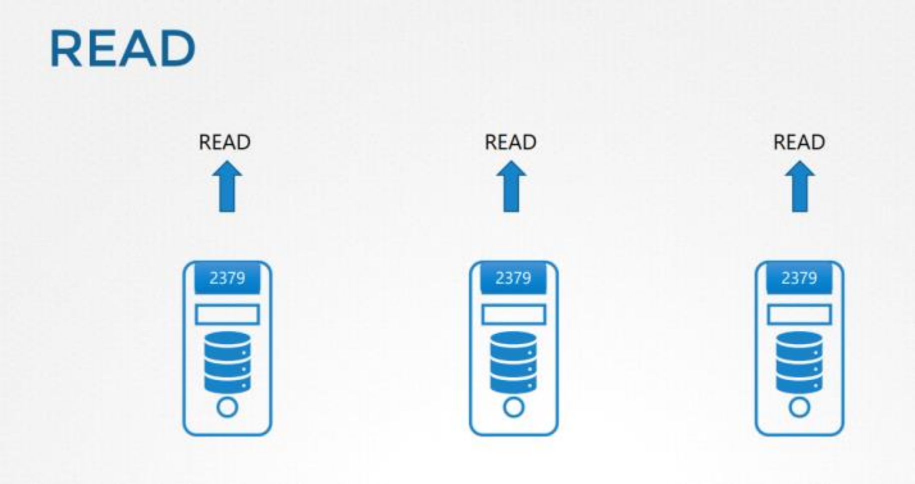
```
{
  "name": "Lauren Rob",
  "age": 13,
  "location": "Bangalore",
  "Grade": "C"
}
```



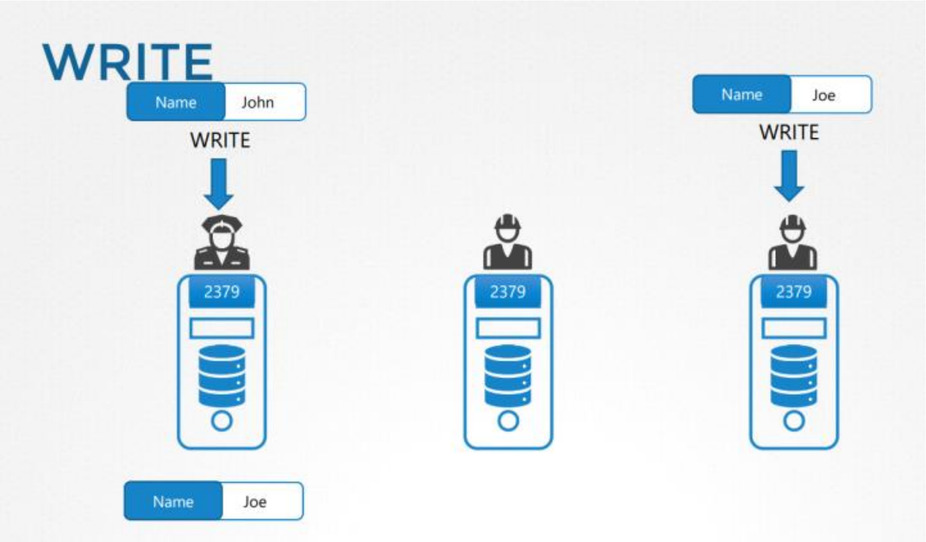
distributed



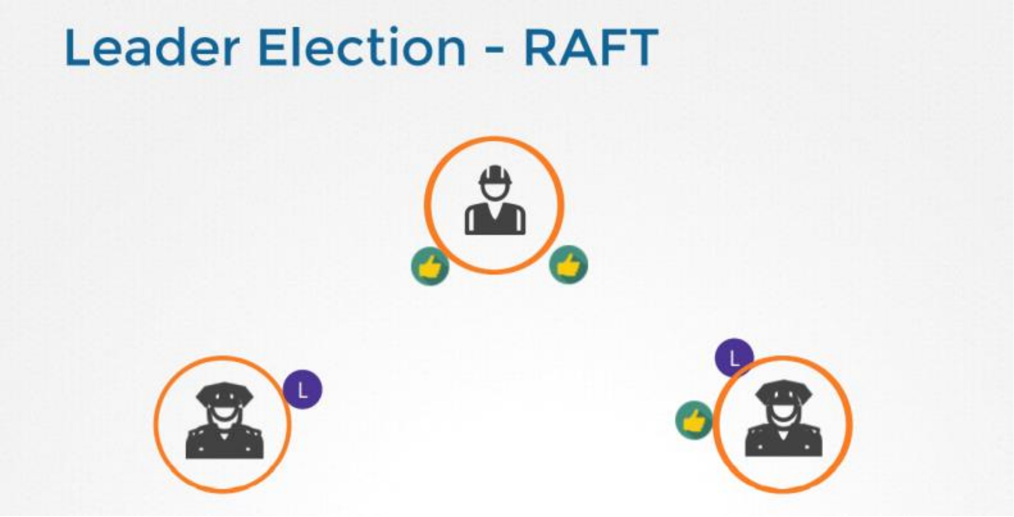
Consistent



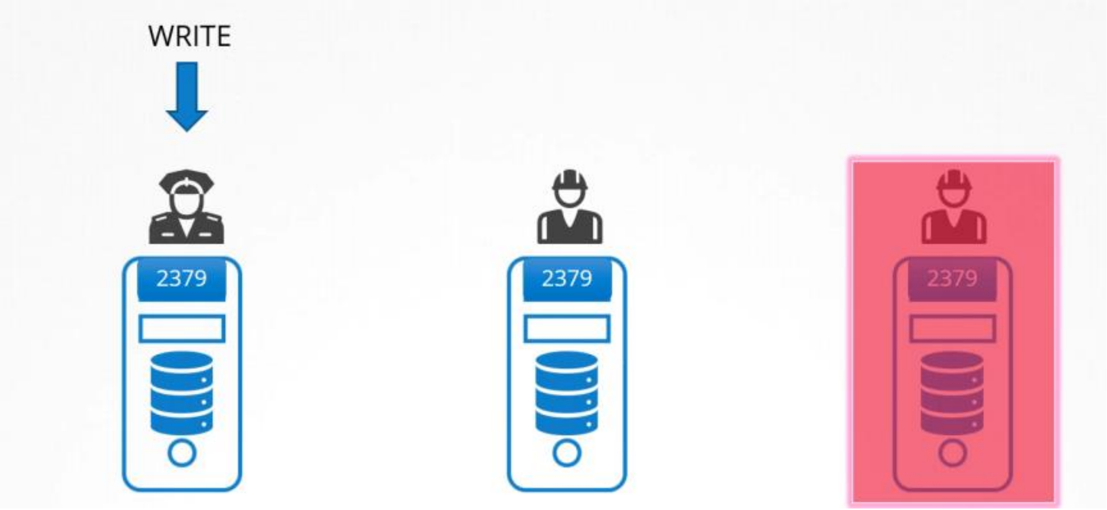
READ



WRITE

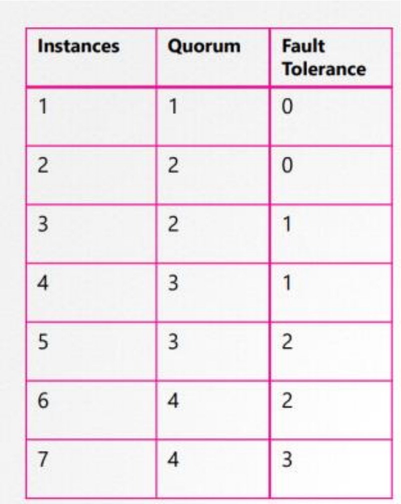


Leader Election - RAFT

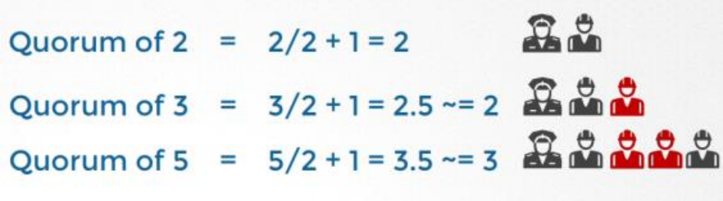


$$\text{Majority} = N/2 + 1$$

$$\text{Quorum} = N/2 + 1$$



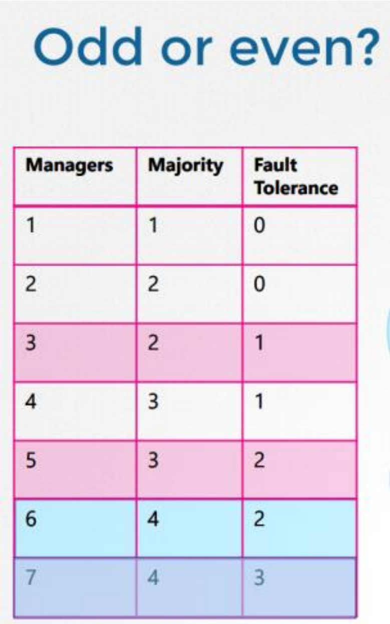
Instances	Quorum	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3



Quorum of 2 = 2/2 + 1 = 2

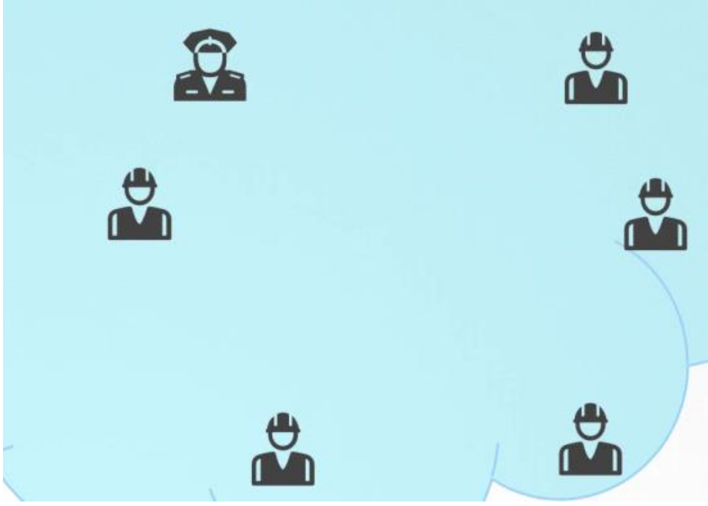
Quorum of 3 = 3/2 + 1 = 2.5 ~= 2

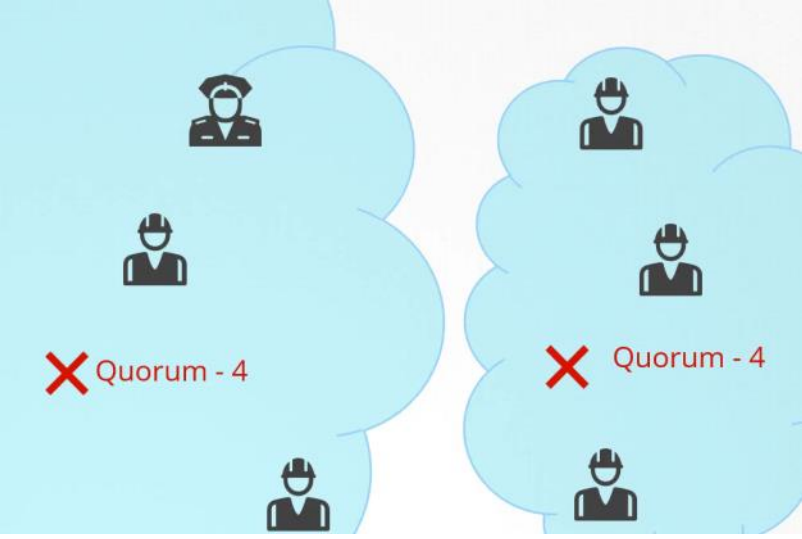
Quorum of 5 = 5/2 + 1 = 3.5 ~= 3

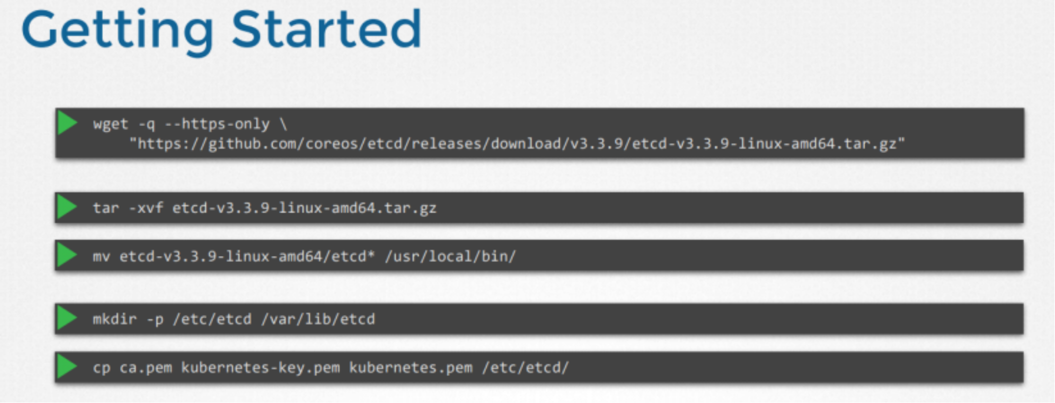


Odd or even?

Managers	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3







Getting Started

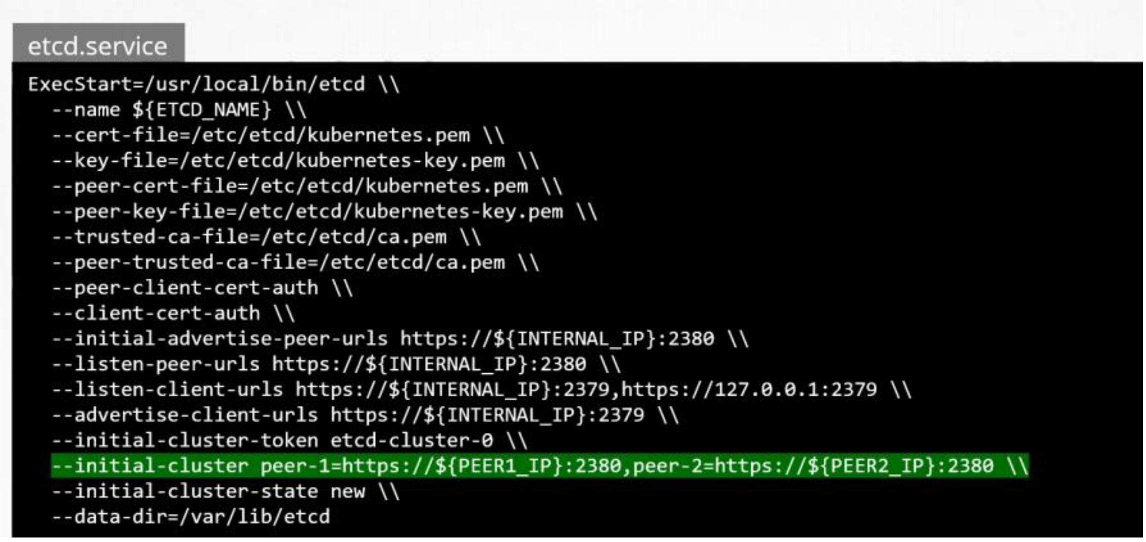
```
wget -q --https-only \
    "https://github.com/coreos/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz"
```

```
tar -xvf etcd-v3.3.9-linux-amd64.tar.gz
```

```
mv etcd-v3.3.9-linux-amd64/etcd* /usr/local/bin/
```

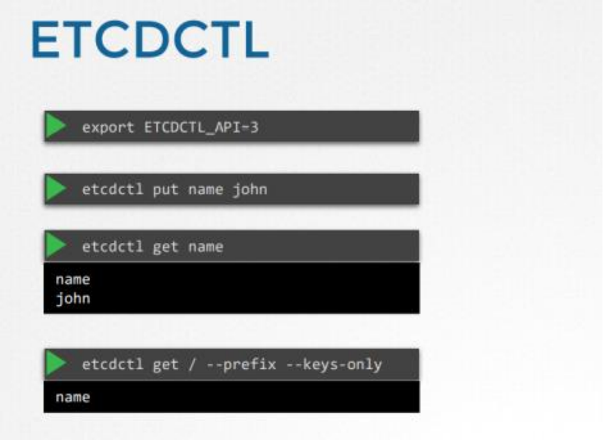
```
mkdir -p /etc/etcd /var/lib/etcd
```

```
cp ca.pem kubernetes-key.pem kubernetes.pem /etc/etcd/
```



etcd.service

```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,https://127.0.0.1:2379 \\
  --advertise-client-urls https://${INTERNAL_IP}:2379 \\
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster peer-1=https://${PEER1_IP}:2380,peer-2=https://${PEER2_IP}:2380 \\
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
```



ETCDCTL

```
export ETCDCTL_API=3
```

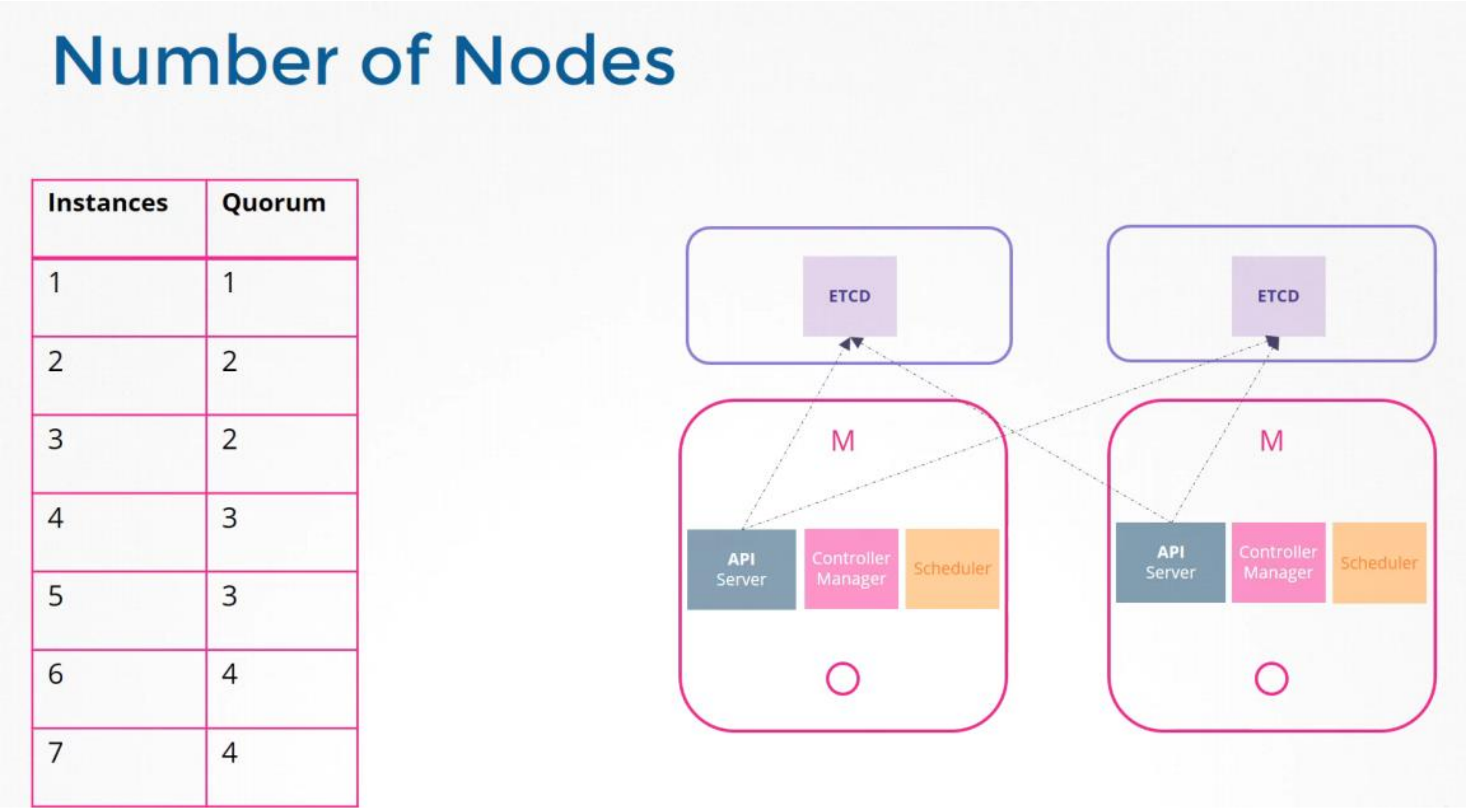
```
etcdctl put name john
```

```
etcdctl get name
```

```
name
john
```

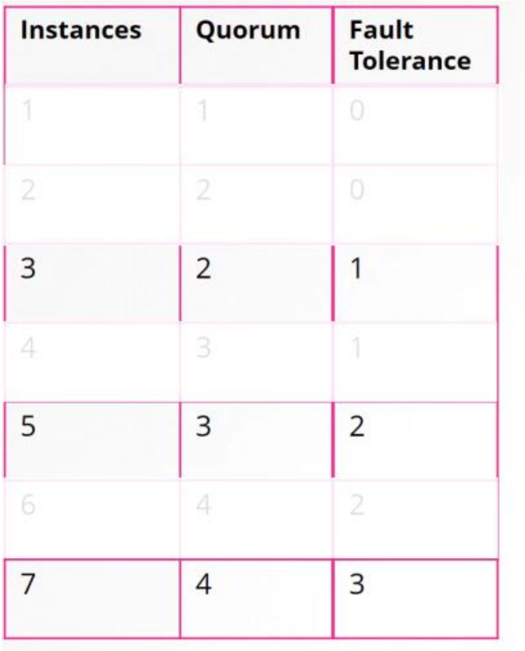
```
etcdctl get / --prefix --keys-only
```

```
name
```



Number of Nodes

Instances	Quorum
1	1
2	2
3	2
4	3
5	3
6	4
7	4



Instances	Quorum	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

Important Update: Kubernetes the Hard Way

Installing Kubernetes the hard way can help you gain a better understanding of putting together the different components manually.
An optional series on this is available at our youtube channel here:

https://www.youtube.com/watch?v=uUupRagM7m0&list=PL2We04F3Y_41jYdadX55fdJplDvgNGENo

The GIT Repo for this tutorial can be found here:
https://github.com/mmumshad/kubernetes-the-hard-way