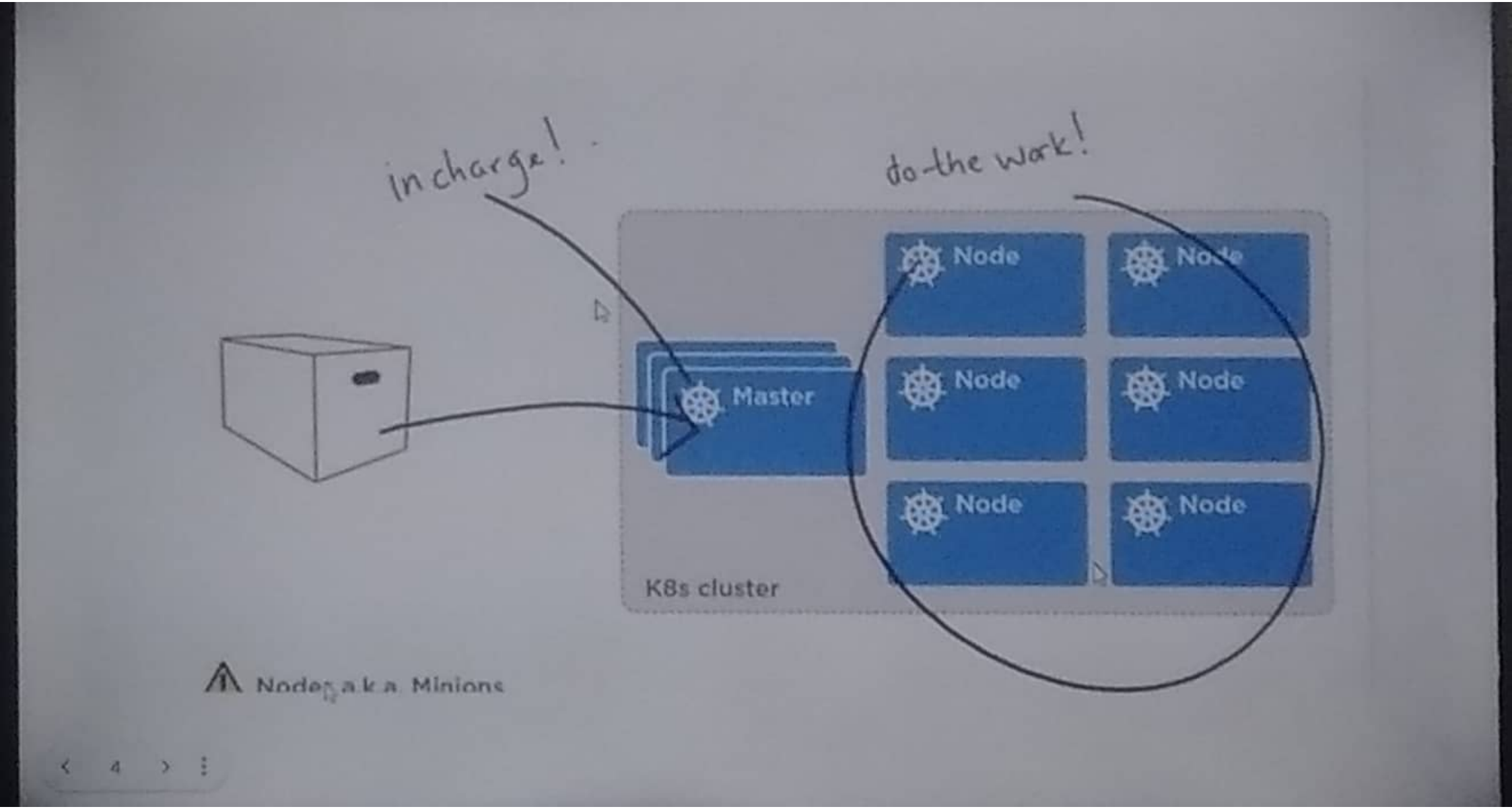
kubernetes

- Kubernetes is a orchestration tool
- By using this we can achieve high
    1. availability
    2. scalability
    3. Desired state

Scanned with CamScanner



Node a.k.a Minions

Scanned with CamScanner

Four Components

API Server:Authentication person make a decision

Cluster Storage: memory

Controller: Monitoring

Scheduler: Work Assign to Nodes

Scanned with CamScanner

Cluster store

Persistent storage

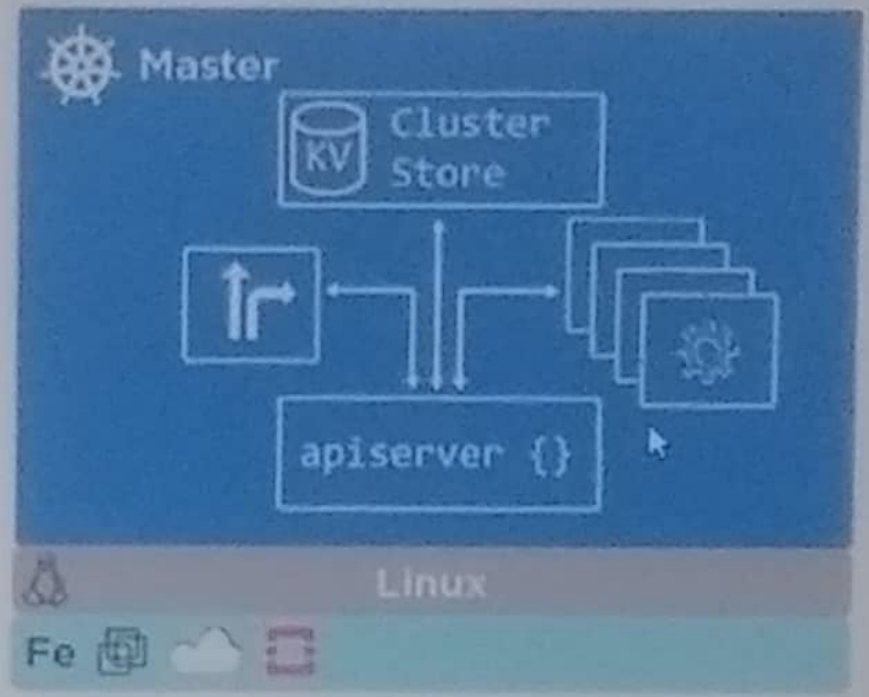
Cluster state and config

Uses etcd

Distributed, consistent, watchable...

The *"source of truth"* for the cluster

Have a backup plan for it!



Cluster Storage keep every information in cluster as a Key:Value

E.g apiserver is a brain which takes decision based on memory

Scanned with CamScanner

kube-controller-manager

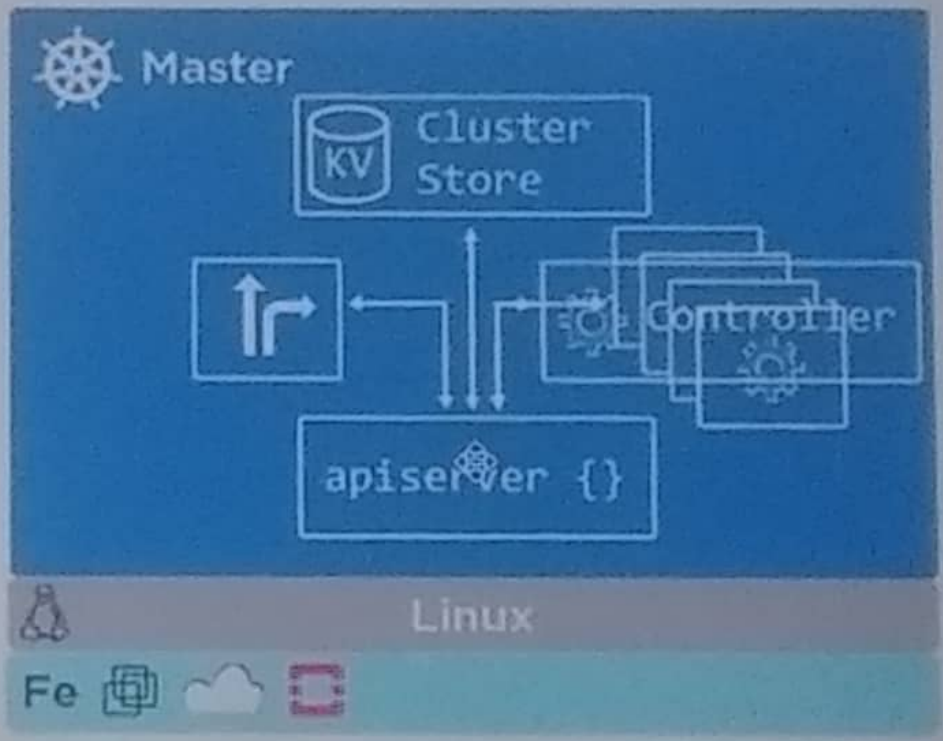
Monitoring

Controller of controllers

- Node controller
- Endpoints controller
- Namespace controller
- ...

Watches for changes

Helps maintain *desired state*



Kubernetes Controller If any changes takes update cluster Storage as a Key:value

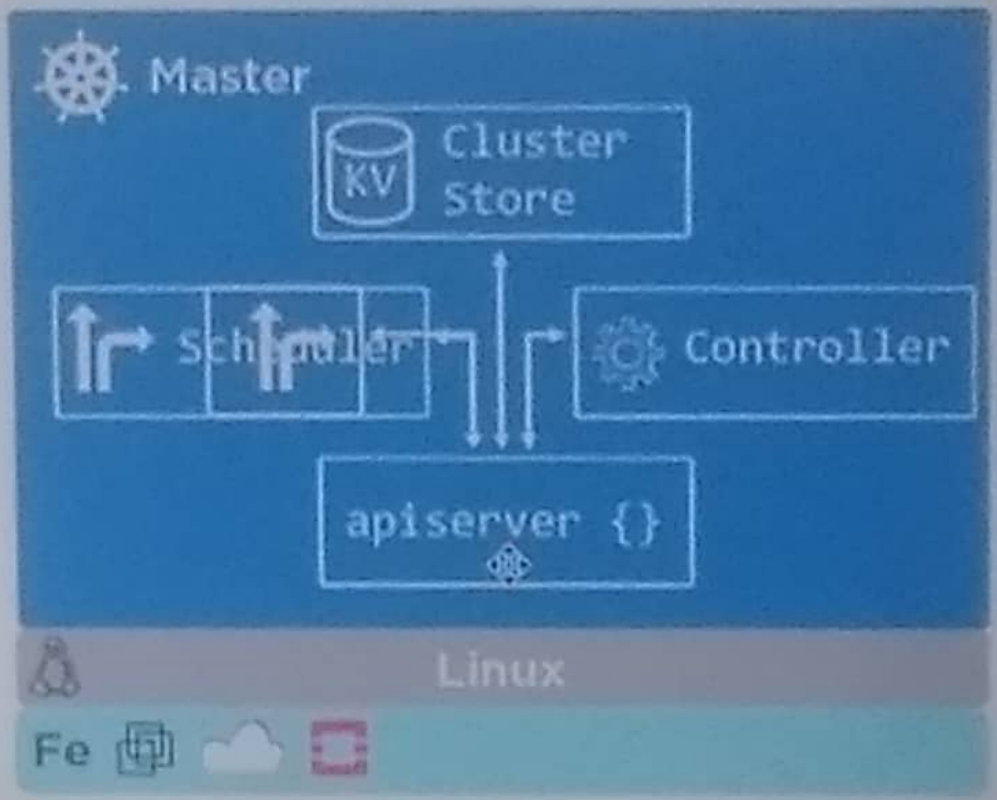
Scanned with CamScanner

kube-scheduler

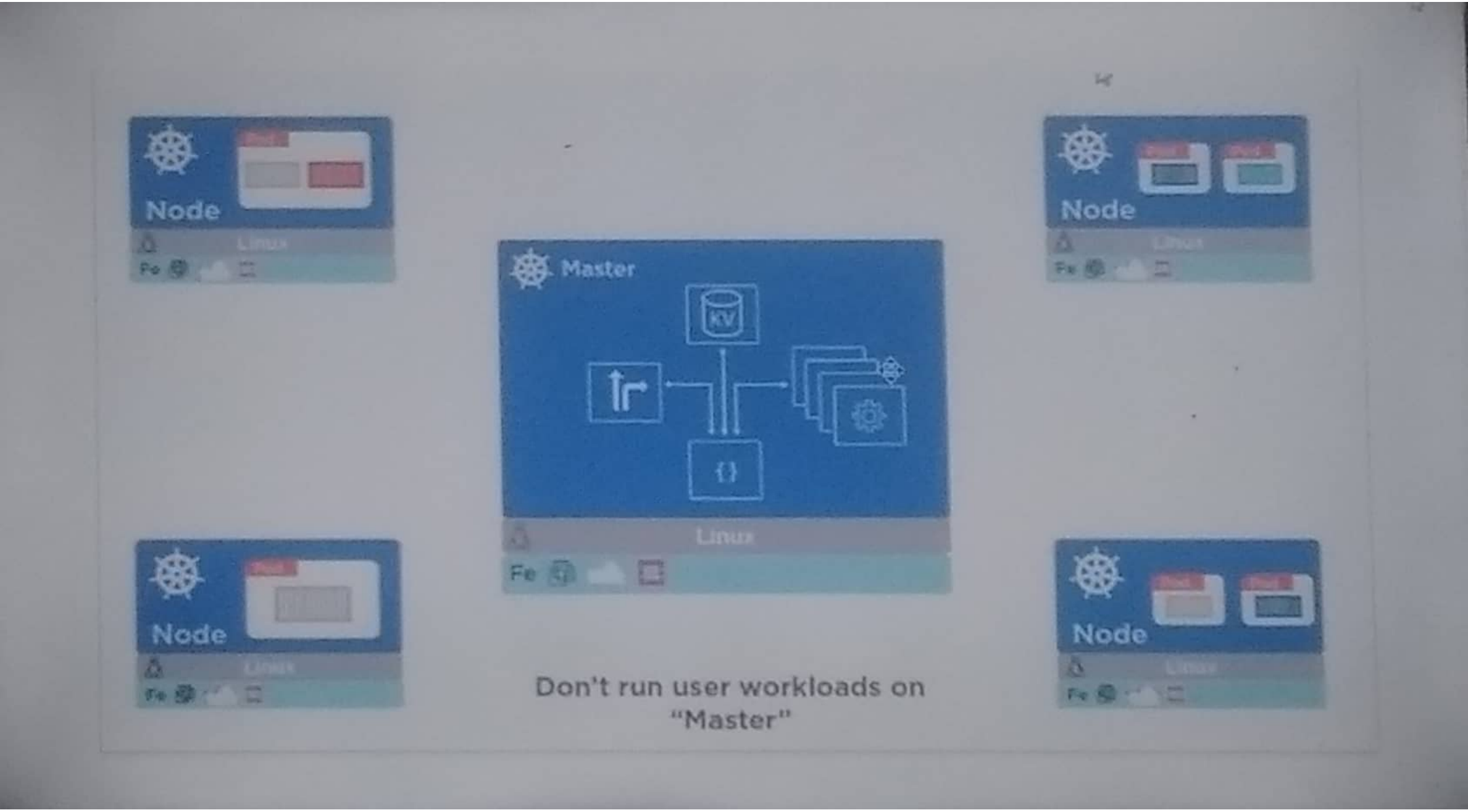
Watches apiserver for new pods

Assigns work to nodes

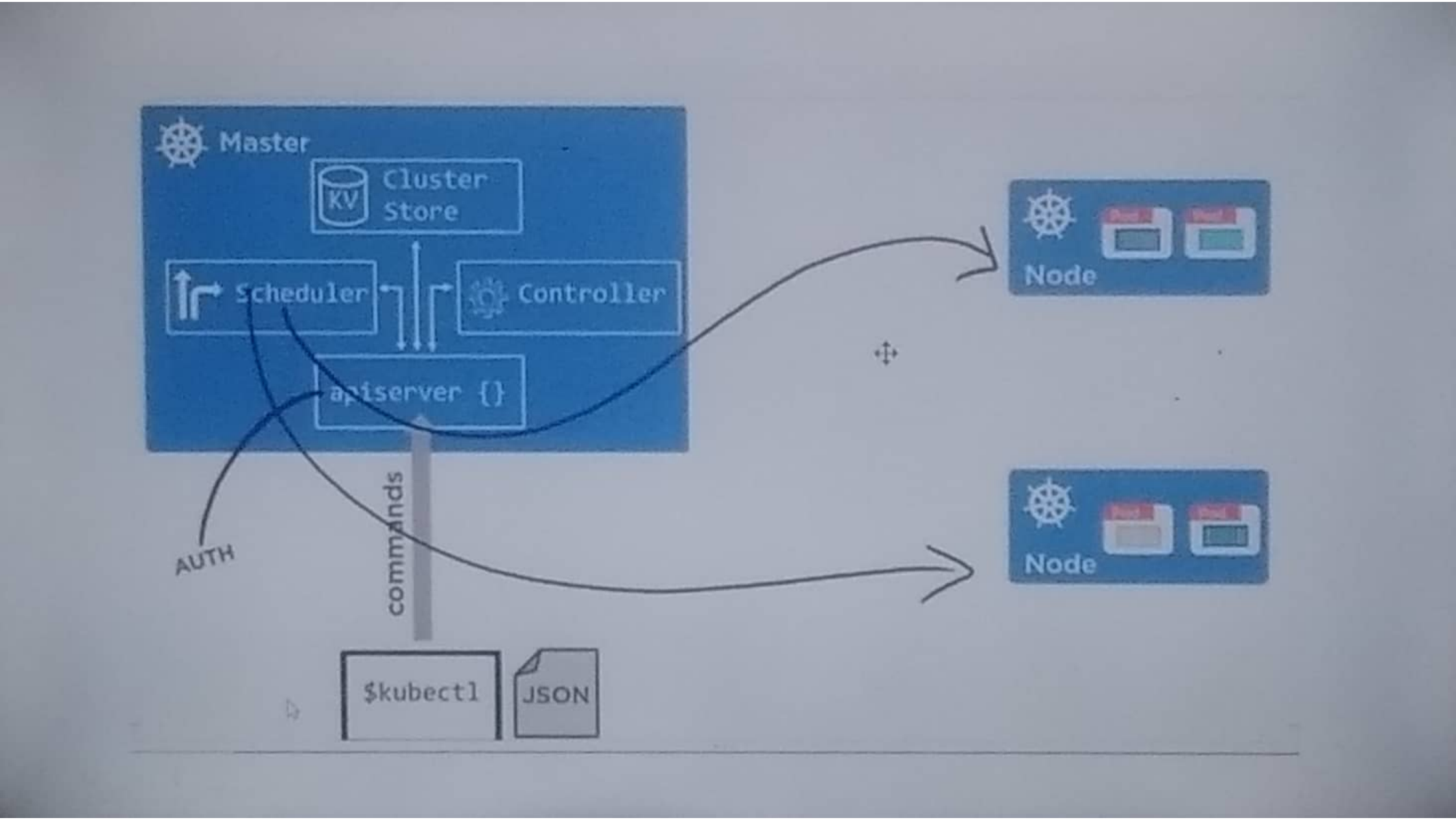
- affinity/anti-affinity
- constraints
- resources
- ...



Scanned with CamScanner



Scanned with CamScanner



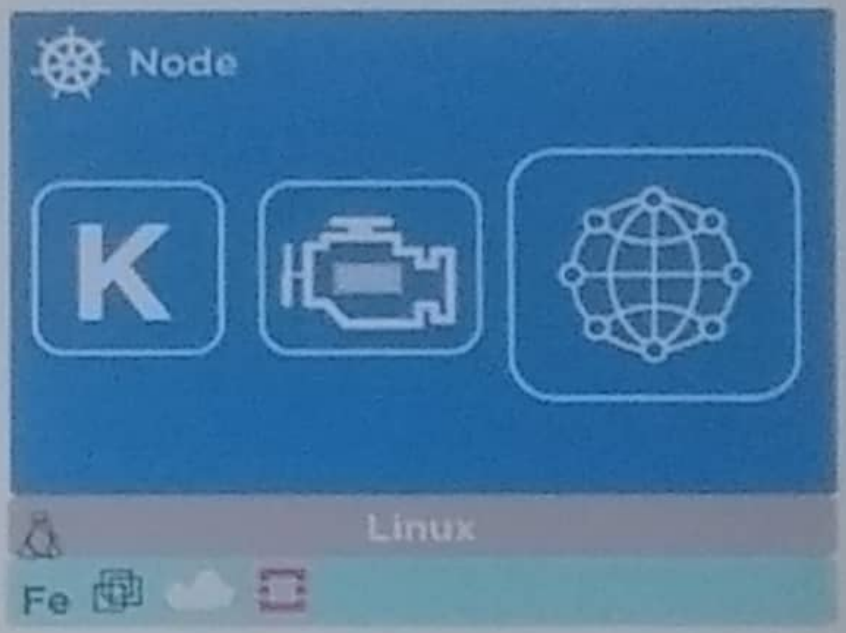
Scanned with CamScanner

Node

Kubelet
Main Kubernetes agent

Container engine
Docker or rkt

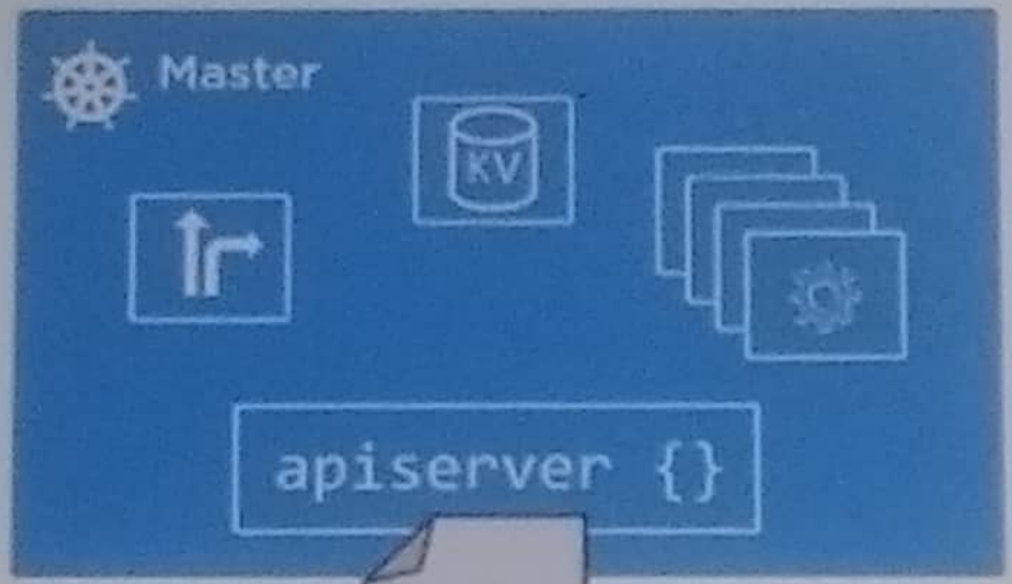
kube-proxy
Kubernetes networking



Scanned with CamScanner

Declarative Model
&
Desired State

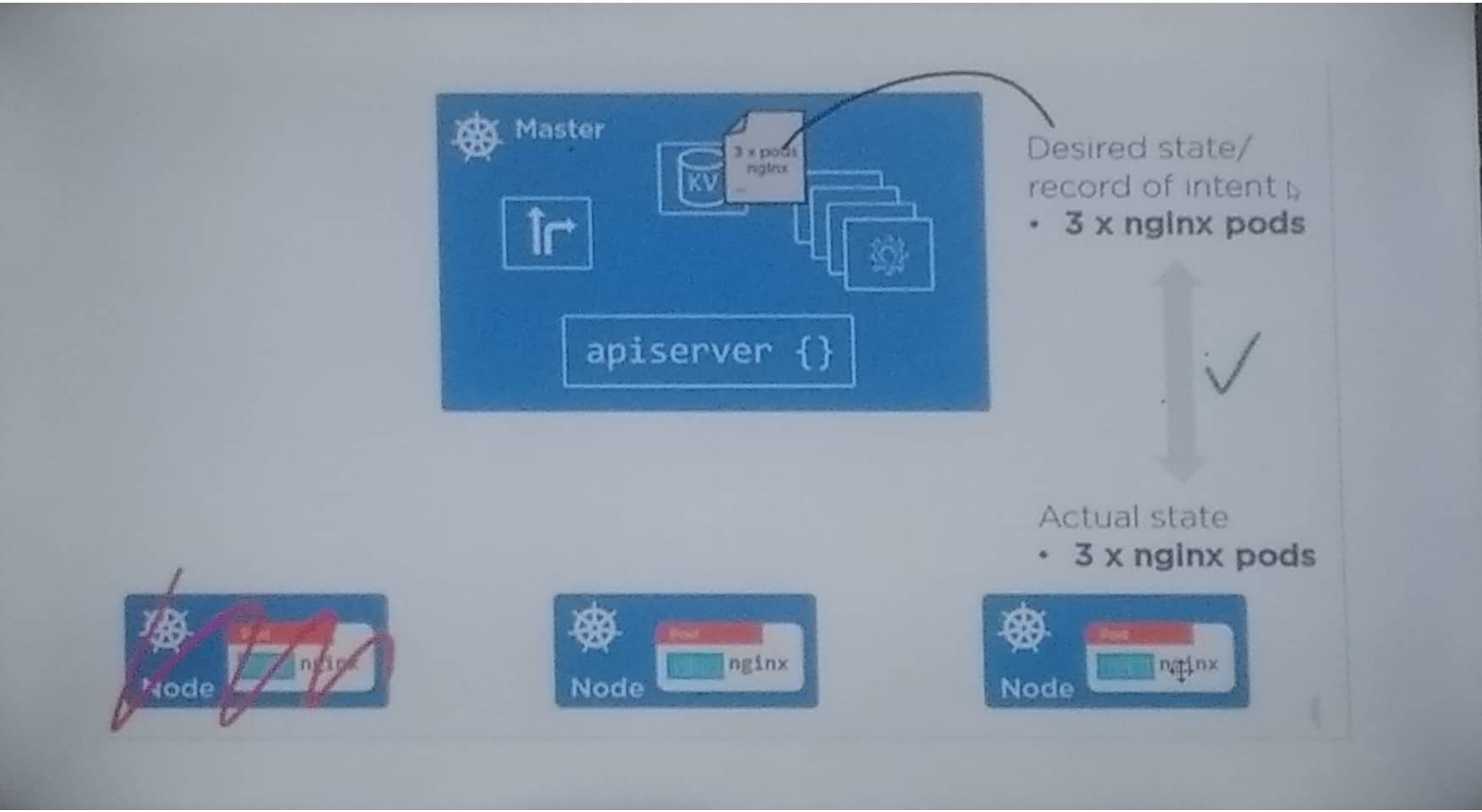
Scanned with CamScanner



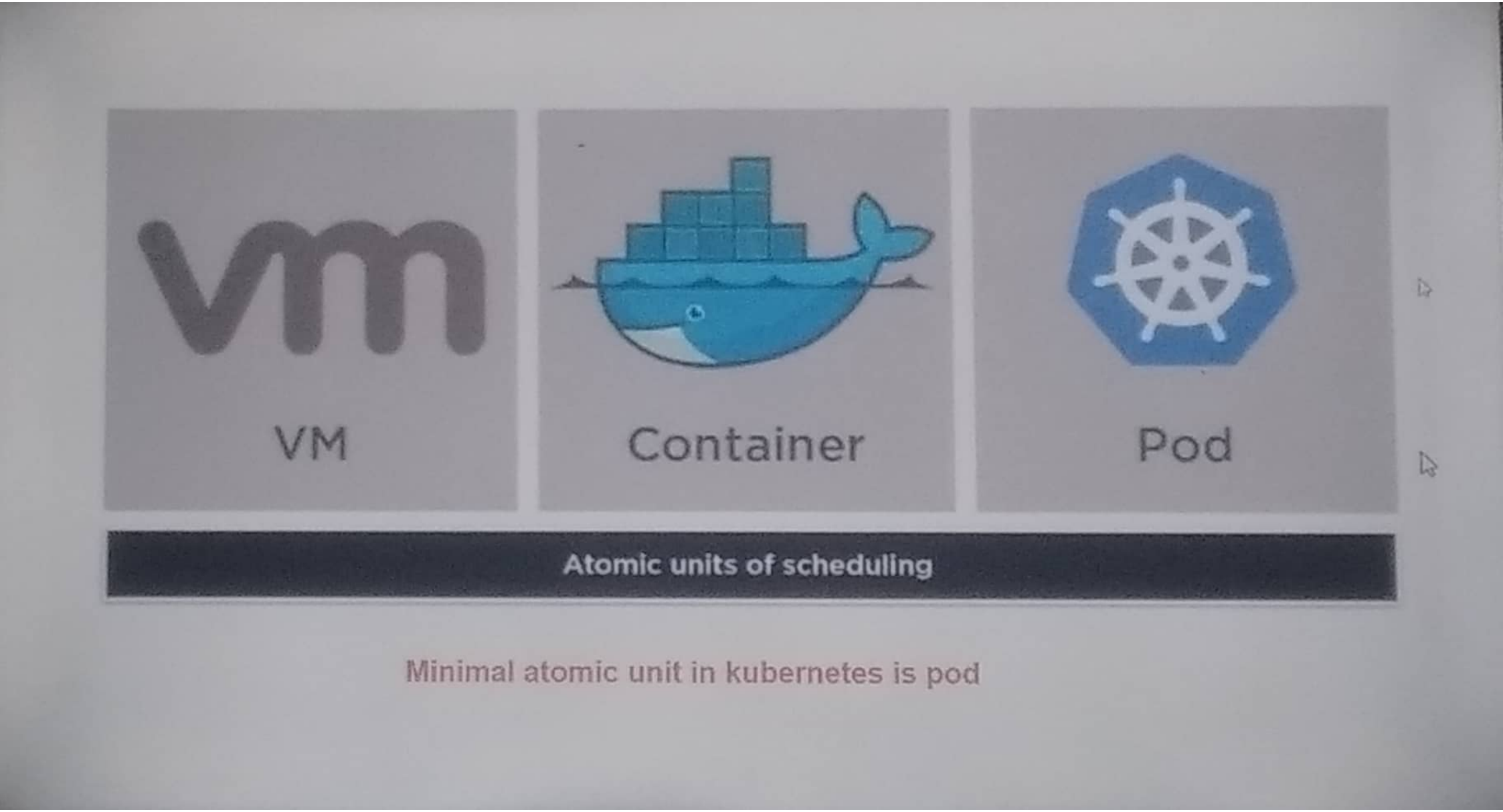
YAML or JSON

Describe desired

Scanned with CamScanner



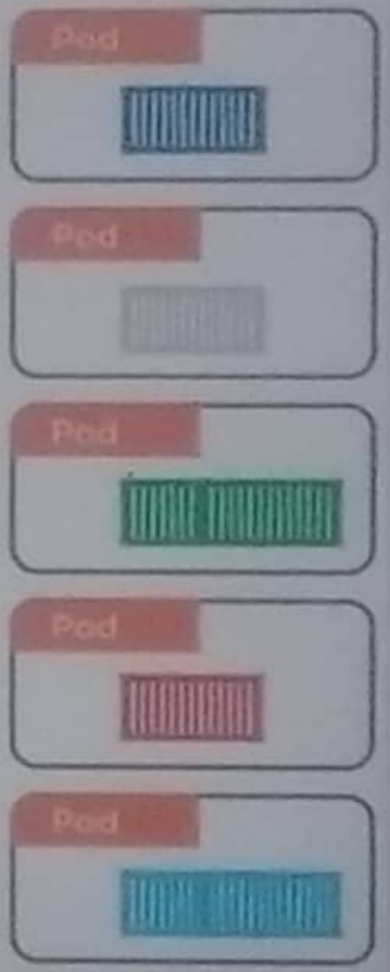
Scanned with CamScanner



Scanned with CamScanner

Containers always run inside of pods

Pods can have multiple containers
(advanced use-case)



We Recommend only one container in one pod

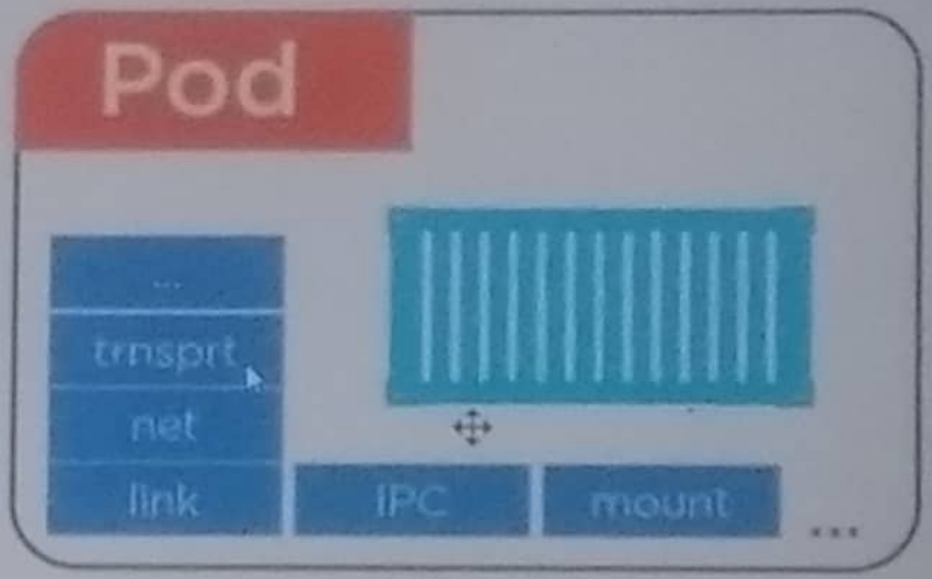
Scanned with CamScanner

Ring-fenced environment

- Network stack
- Kernel namespaces
- ...

n containers

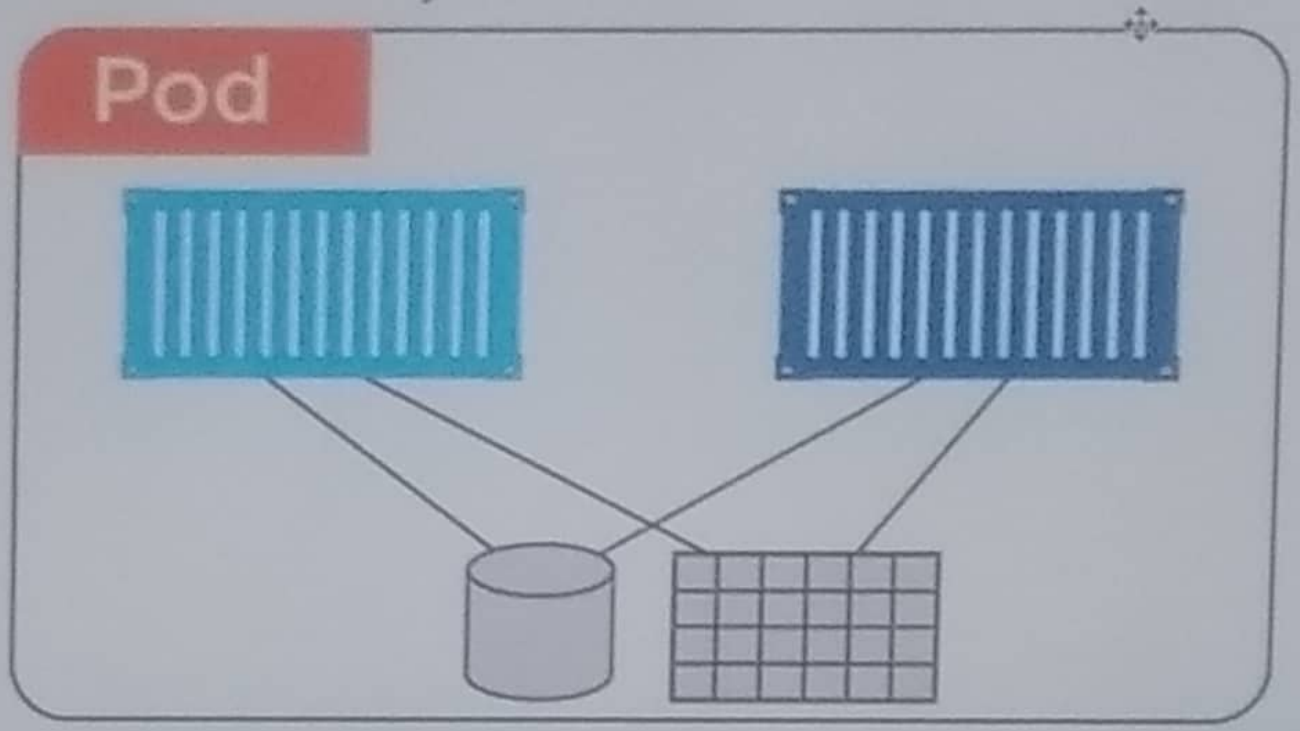
All containers in pod share the pod environment



We Recommend only one container in one pod

Scanned with CamScanner

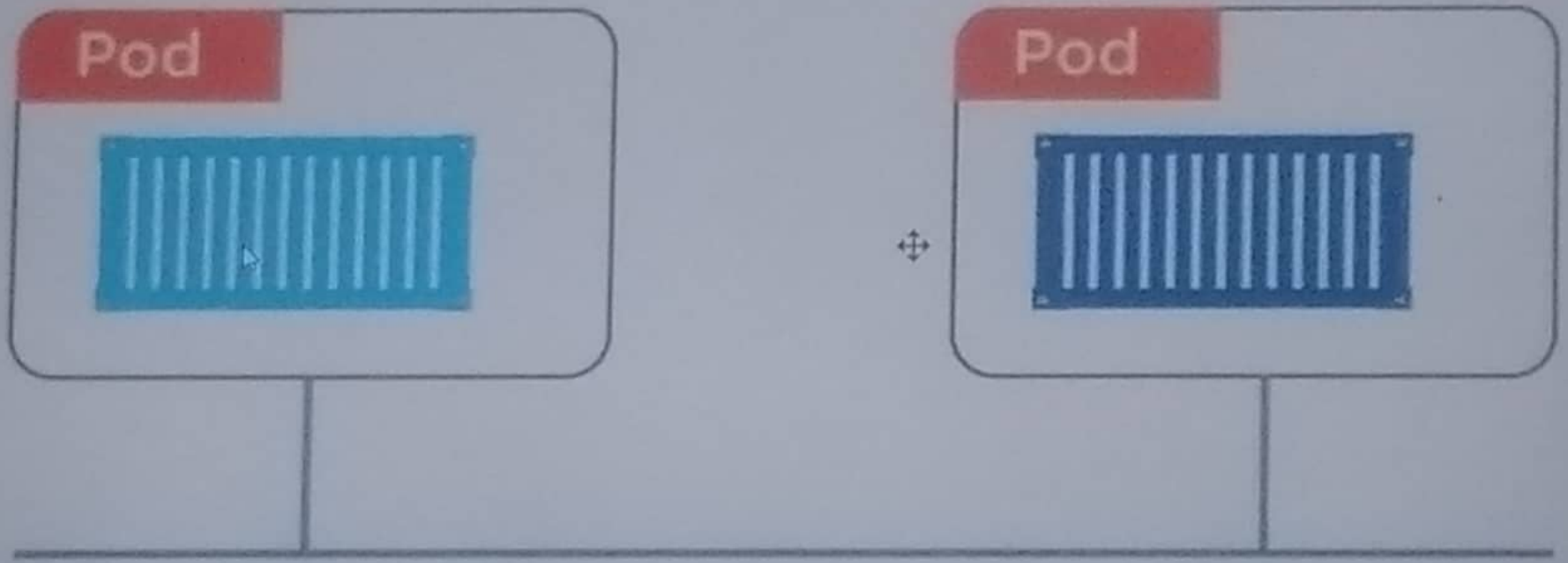
Tight Coupling



In Tight couple if any wrong in one container it affects second container

Scanned with CamScanner

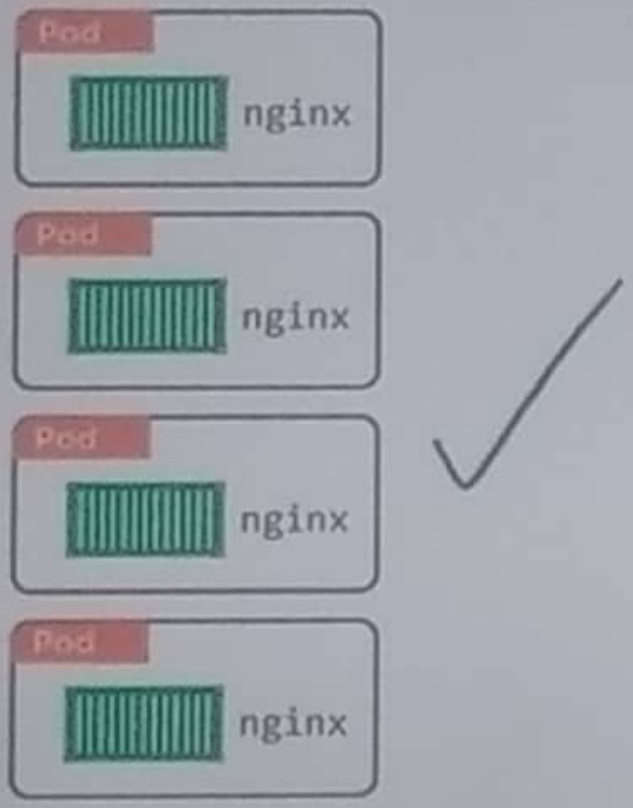
Loose Coupling



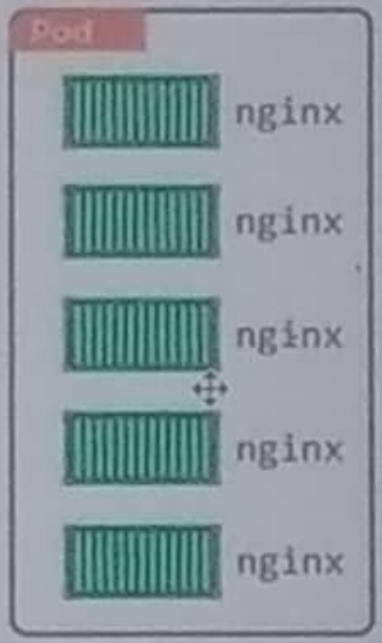
Most Recommend Method

Scanned with CamScanner

Pods and Scaling



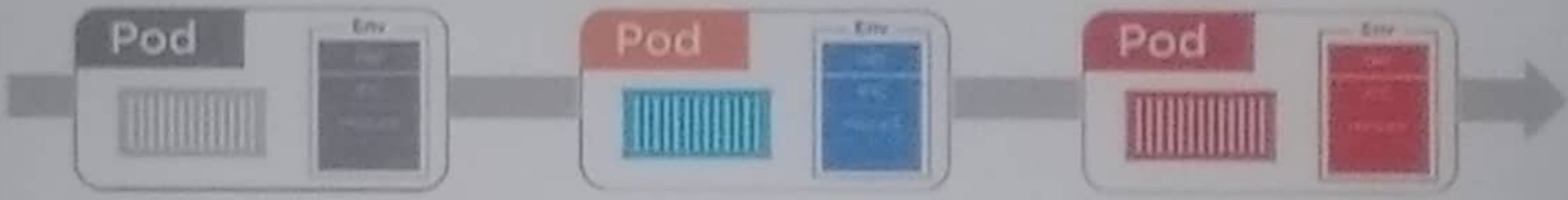
Most Recommend



Not Recommend

Scanned with CamScanner

Pod Lifecycle



Phase: pending

Phase: running

Phase: succeeded/failed

Scanned with CamScanner

To setup cluster we need

1. To implement this we need three service : One master and two nods
2. On master and each node we have to install Docker
3. Install Kubernetes on master and each node
    1. Kubeadm
    2. Kubelet
    3. kubectl
4. After Installing Kubernetes one should be master and 2 Nodes by running init kubeadm unit on master we get a document by using this document we can configure this cluster

Scanned with CamScanner