BATCH: 149

25/10/2023

DT/NT : DT

LESSON : DevOps

SUBJECT: Kubernetes-1

İnstalling

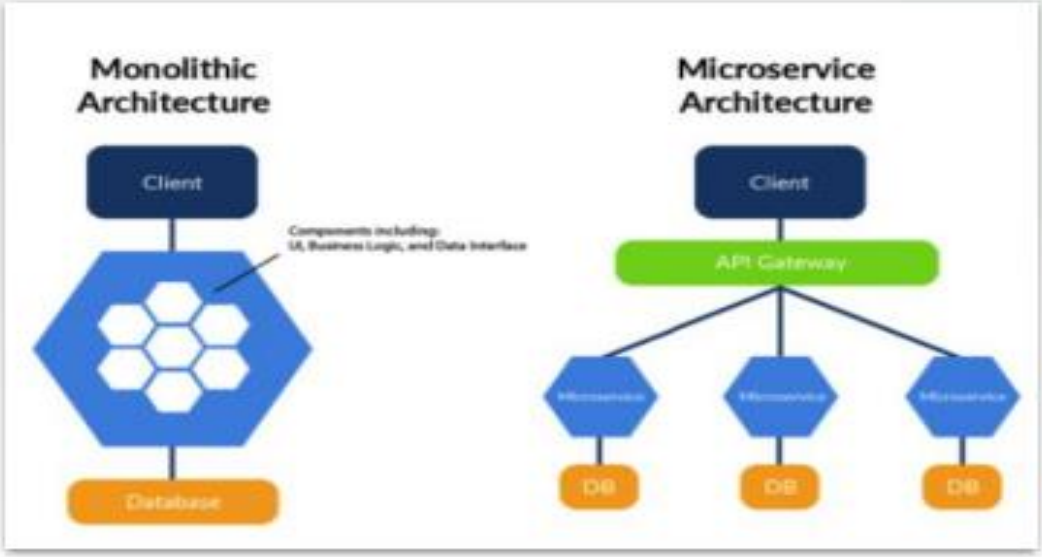
TECHPRO EDUCATION

Table of Contents

Monolith vs Microservices

Monolith vs Microservices

- Microservice applications are scalable, flexible and more reliable.
- Features are independent.
- Each service is deployed individually. Continuous deployment is easier.
- Complex



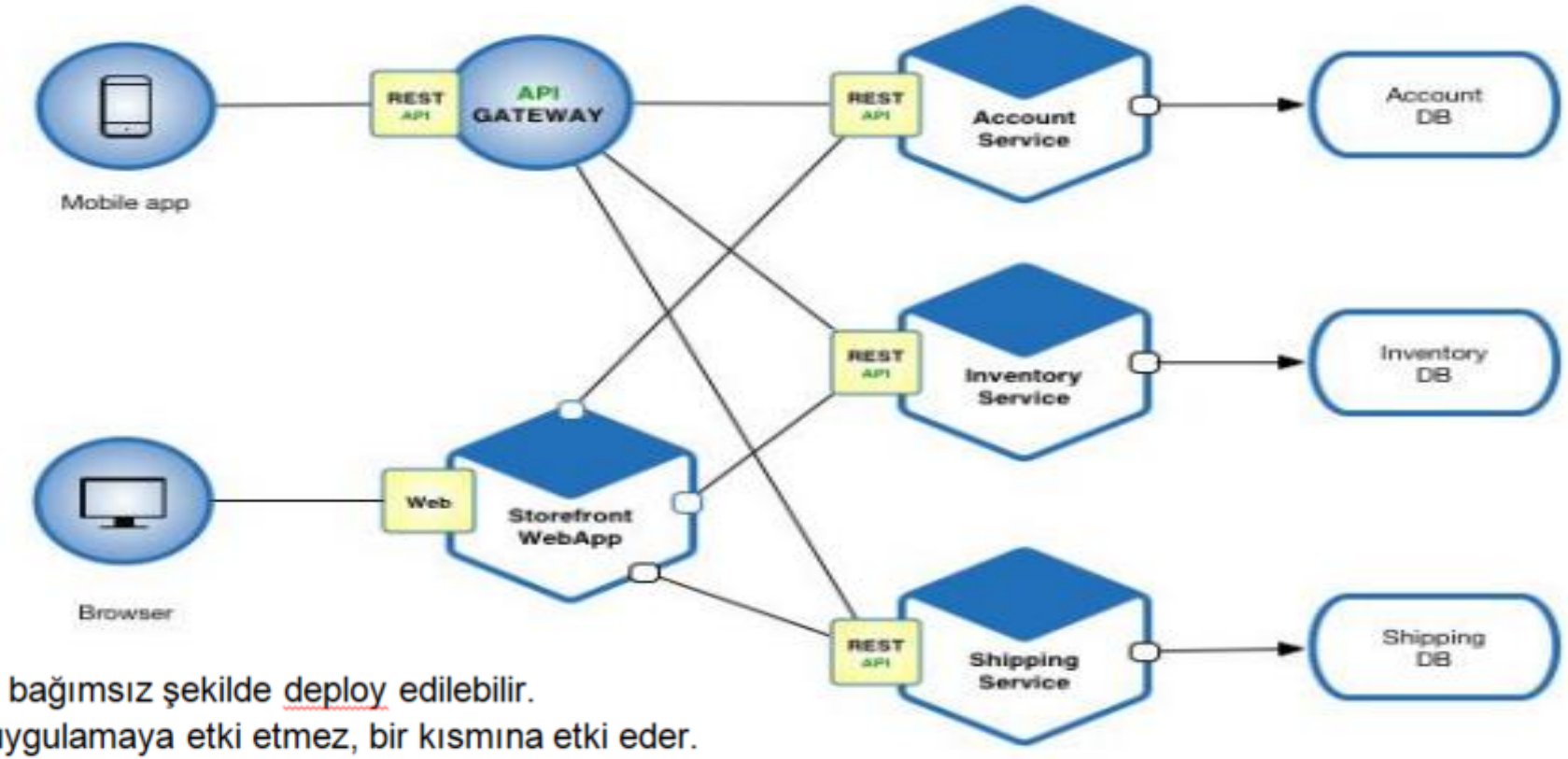
- Monolith applications are simple to develop, test and deploy.
- Difficult to scale.
- Features depend on each other.
- Not cost-efficient in the long run.
- Entire codebase is deployed. Continuous deployment is difficult.
- Difficult to maintain, update.
- Change in a framework will affect the entire application.

Monolith vs Microservices

The word 'monolith' means 'one massive stone'. So we can describe monolithic as a large unified block.

Kodu geliştirmesi kolay çünkü herkes java kullanıo biliyor.
Deploy etmesi kolaydır,
Test ve debug işlemi de kolaydır.

Monolith vs Microservices



Bütün servisler bağımsız şekilde deploy edilebilir.
Bug lar bütün uygulamaya etki etmez, bir kısmına etki eder.
Yeni özellikler eklemek çok daha kolaydır.

Declarative vs Imperative

imperative focuses on **how** and declarative focuses on **what**.

1. Temeli oluşturun
2. Çerçeveye yerleştirin
3. Duvarları ekleyin
4. Kapı ve pencereleri ekleyin

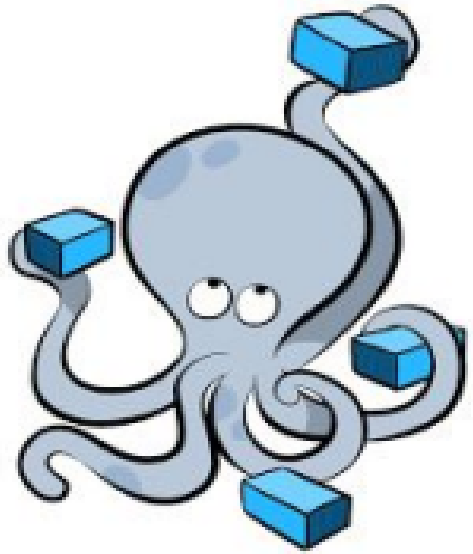
Imperative approach:

1. Build the foundation
2. Put in the framework
3. Add the walls
4. Add the doors and windows

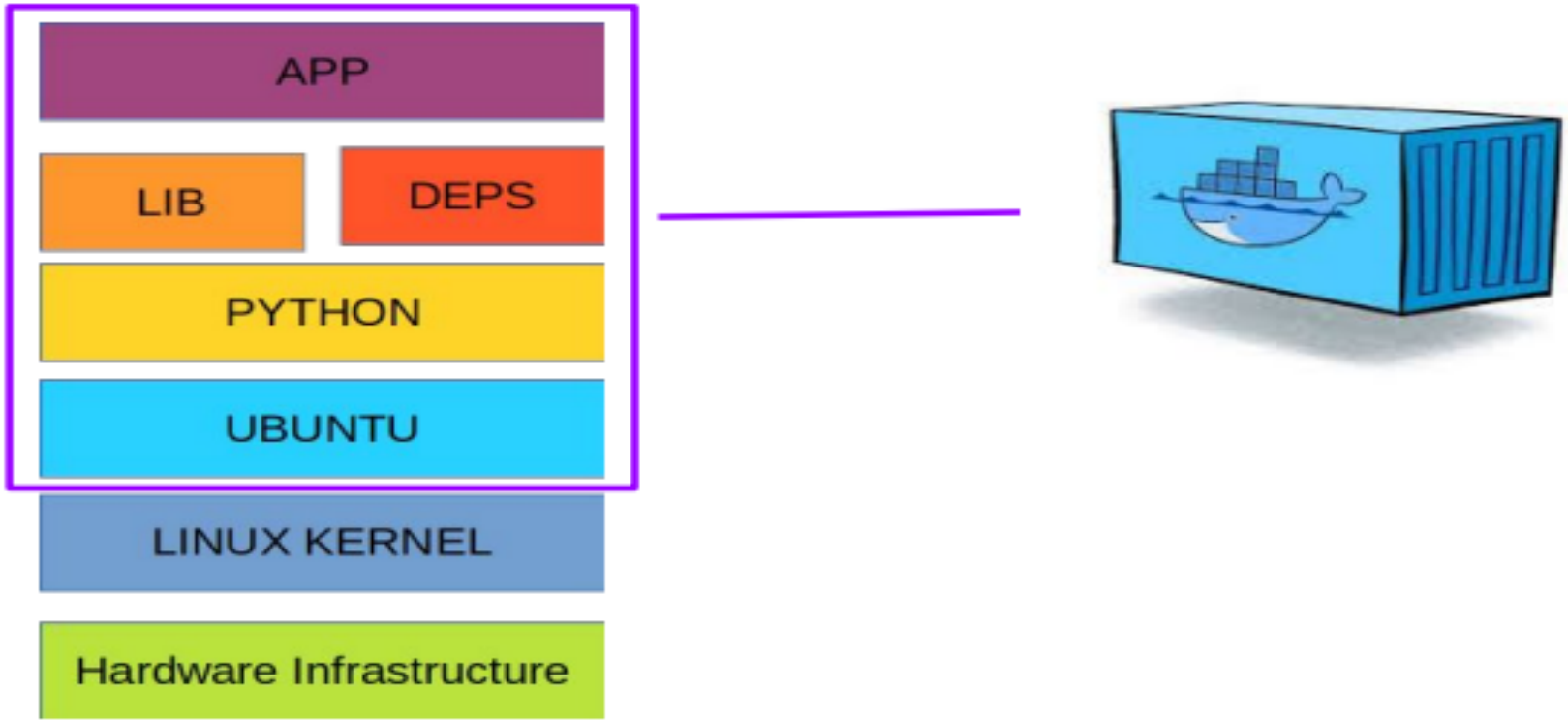
Declarative approach:

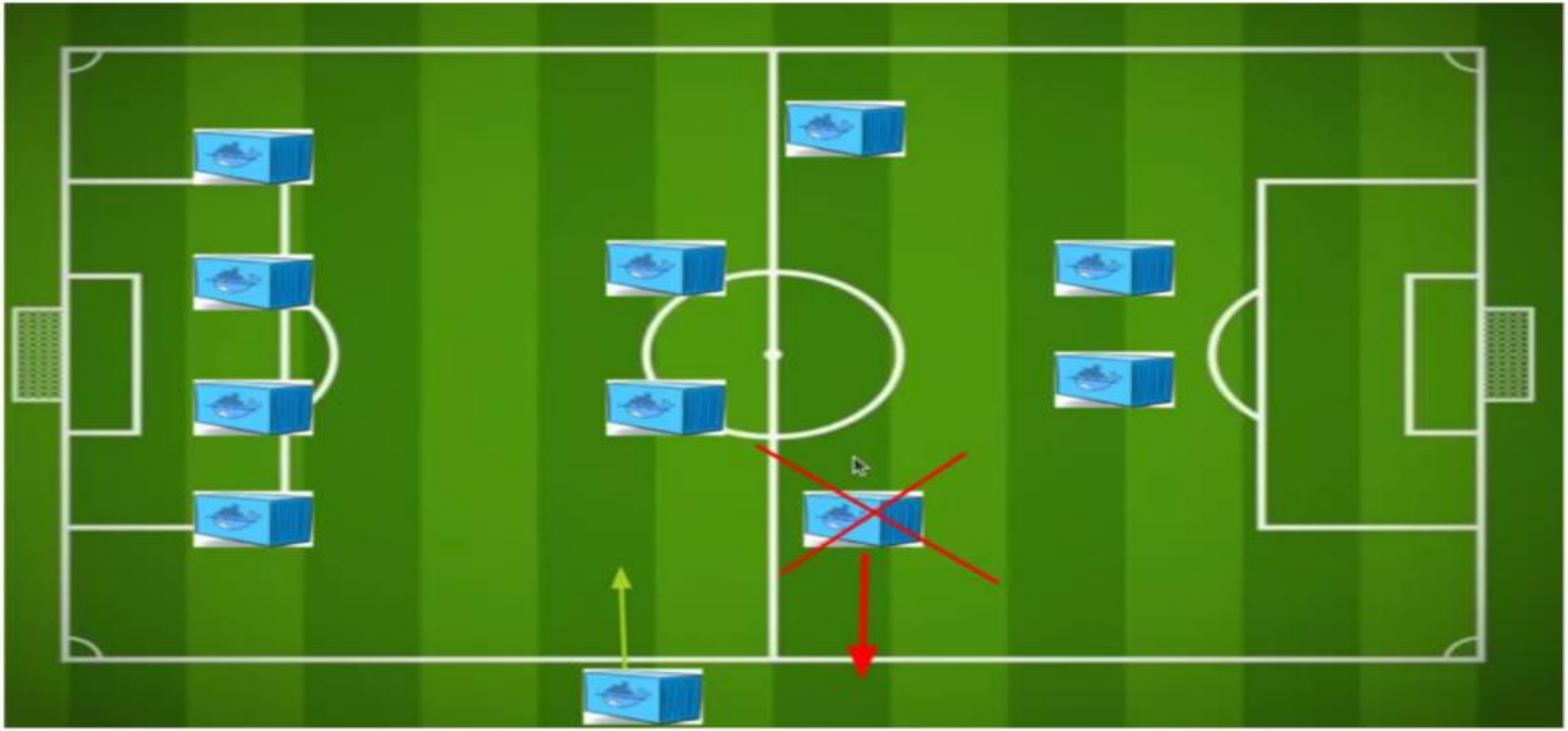
I want a tiny and cute house.

Orchestration



Orchestration

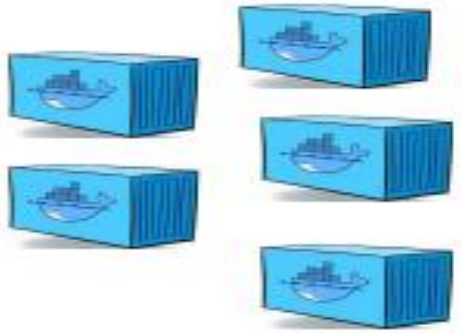




Orchestration

- Containers are great, but when you get lots of them running, at some point, you need them all working together in harmony to solve business problems.
- Tools to manage, scale, and maintain containerized applications are called orchestrators, and the most common example of this is **Kubernetes**.

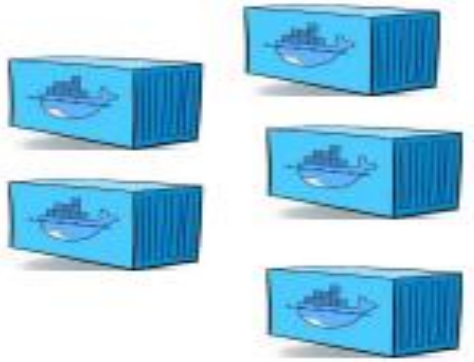
Orchestration



Container orchestration is used to automate the following tasks at scale:

- Provisioning and deployments of containers
- Availability of containers
- Load balancing, traffic routing and service discovery of containers
- Health monitoring of containers

Orchestration



- Securing the interactions between containers.
- Configuring and scheduling of containers
- Allocation of resources between containers

What is Kubernetes?

What is Kubernetes?

- Born in Google
- Donated to CNCF in 2014
- Open source (Apache 2.0)
- v1.0 July 2015
- Written in Go/Golang
- Often shortened to k8s

Kubernetes

k8 s

CNCF: Cloud Native Computing Foundation

What is Kubernetes?

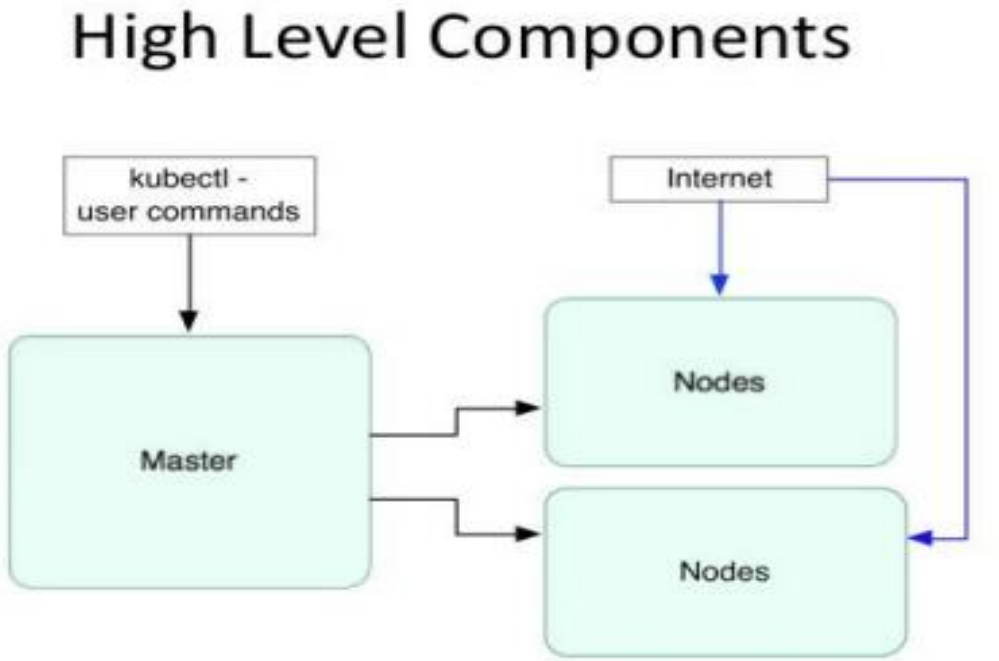
- Kubernetes is **Open Source Orchestration** system for Containerized Applications.
- Kubernetes is a platform that **eliminates the manual processes** involved in **deploying** containerized applications.
- Kubernetes used to manage the **State of Containers**.
    - Start Containers on Specific Nodes.
    - Restart Containers when gets Killed.
    - Move containers from one Node to Another.

Why you need Kubernetes?

Kubernetes supplies you with:

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

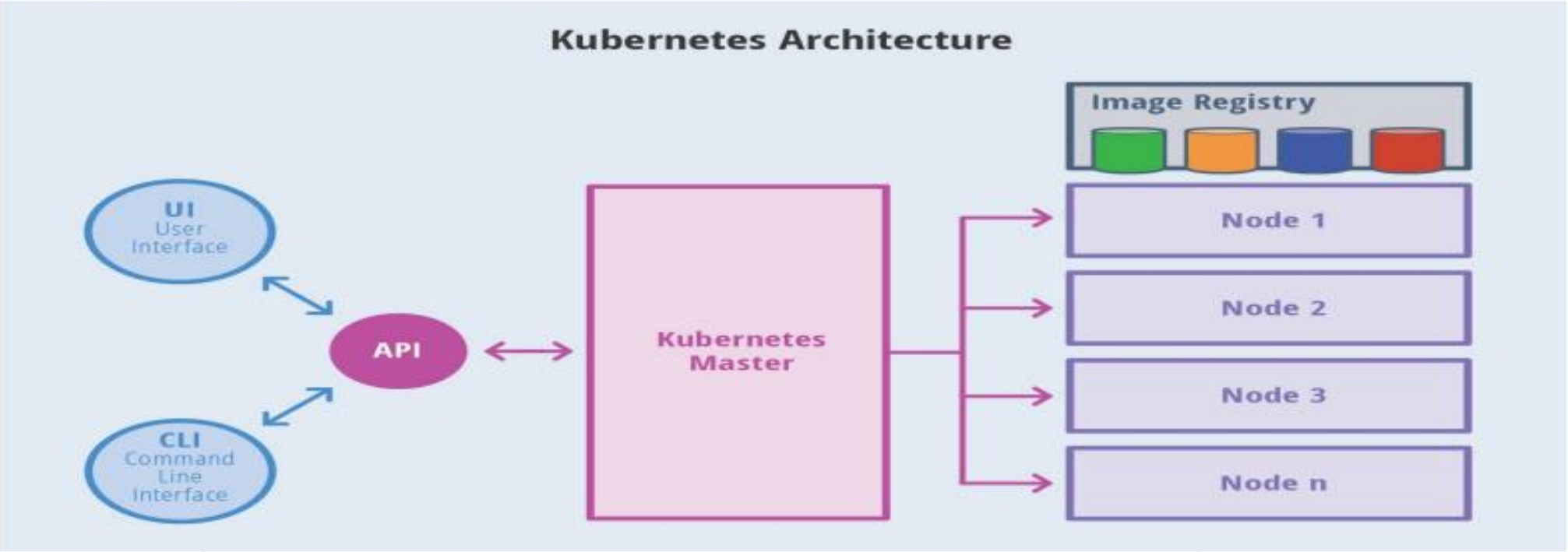
Kubernetes Components



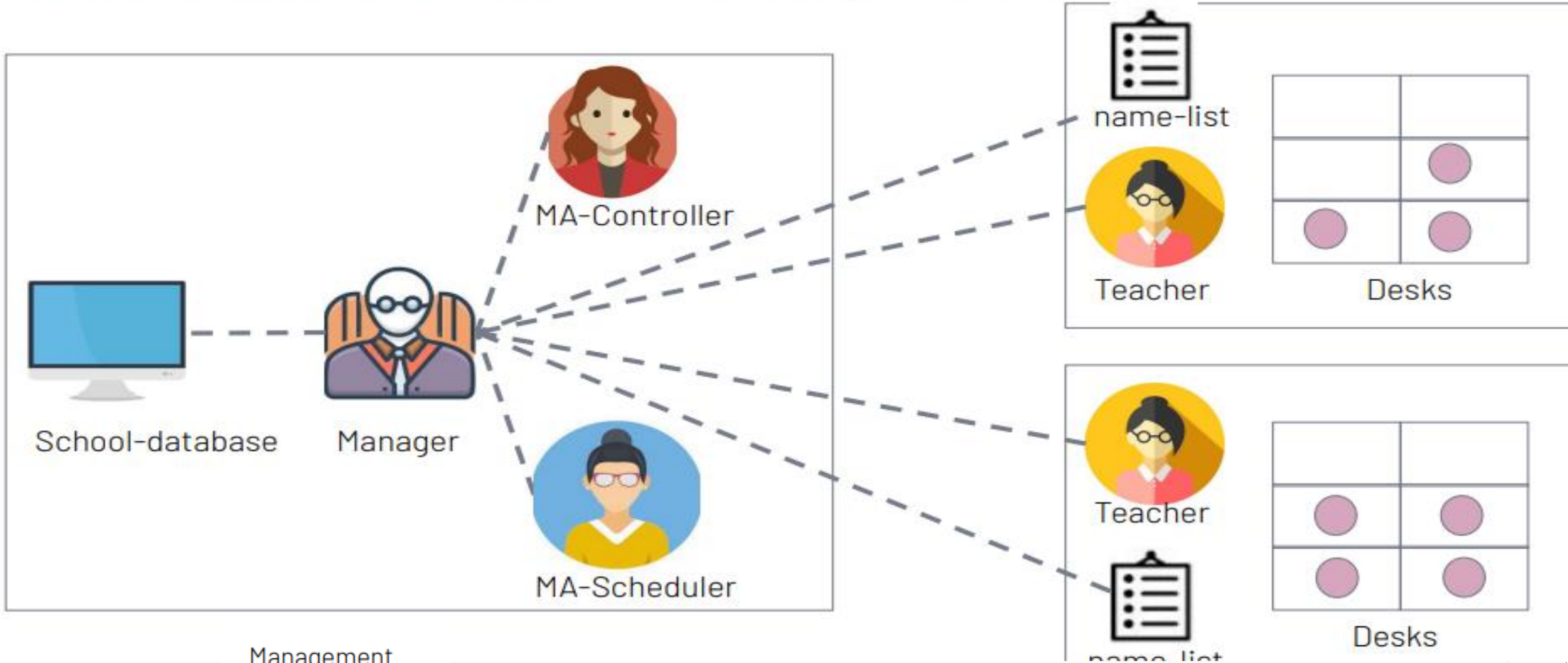
Kubernetes Components

Kubernetes has the following main components:

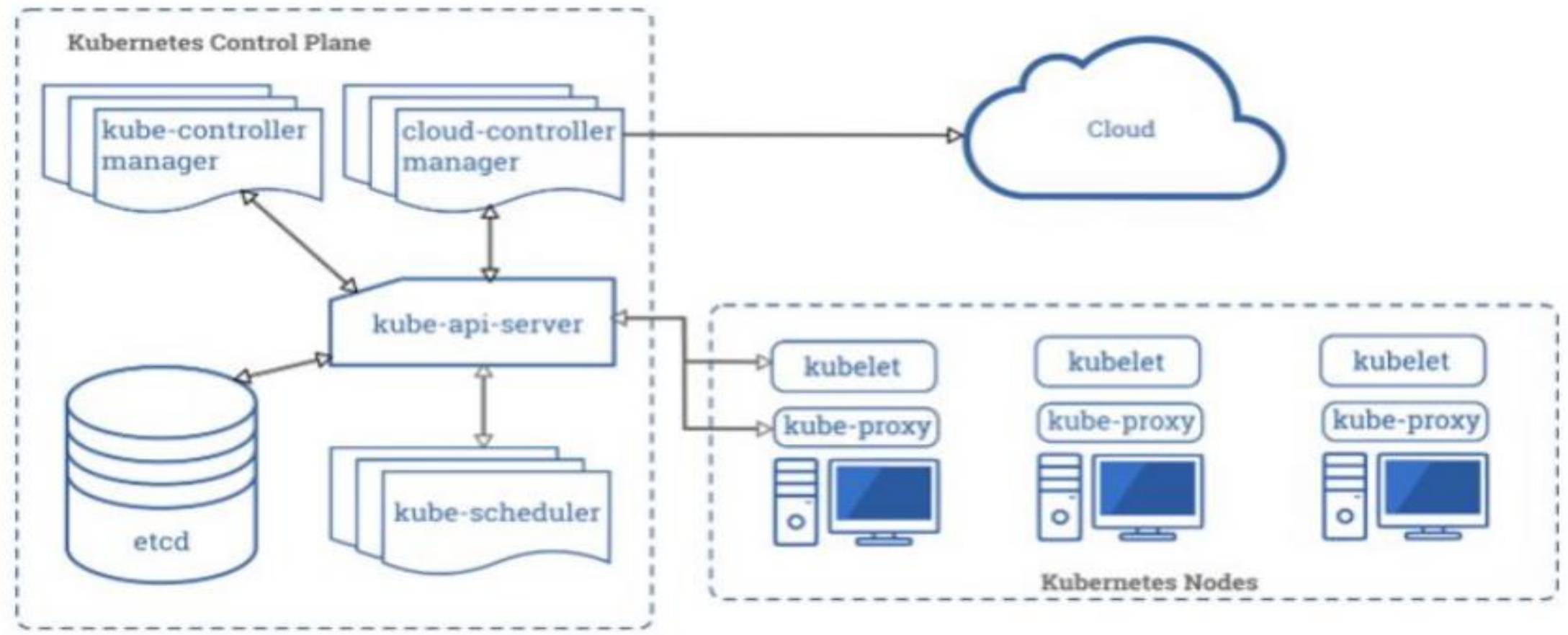
- One or more master nodes
- One or more worker nodes.



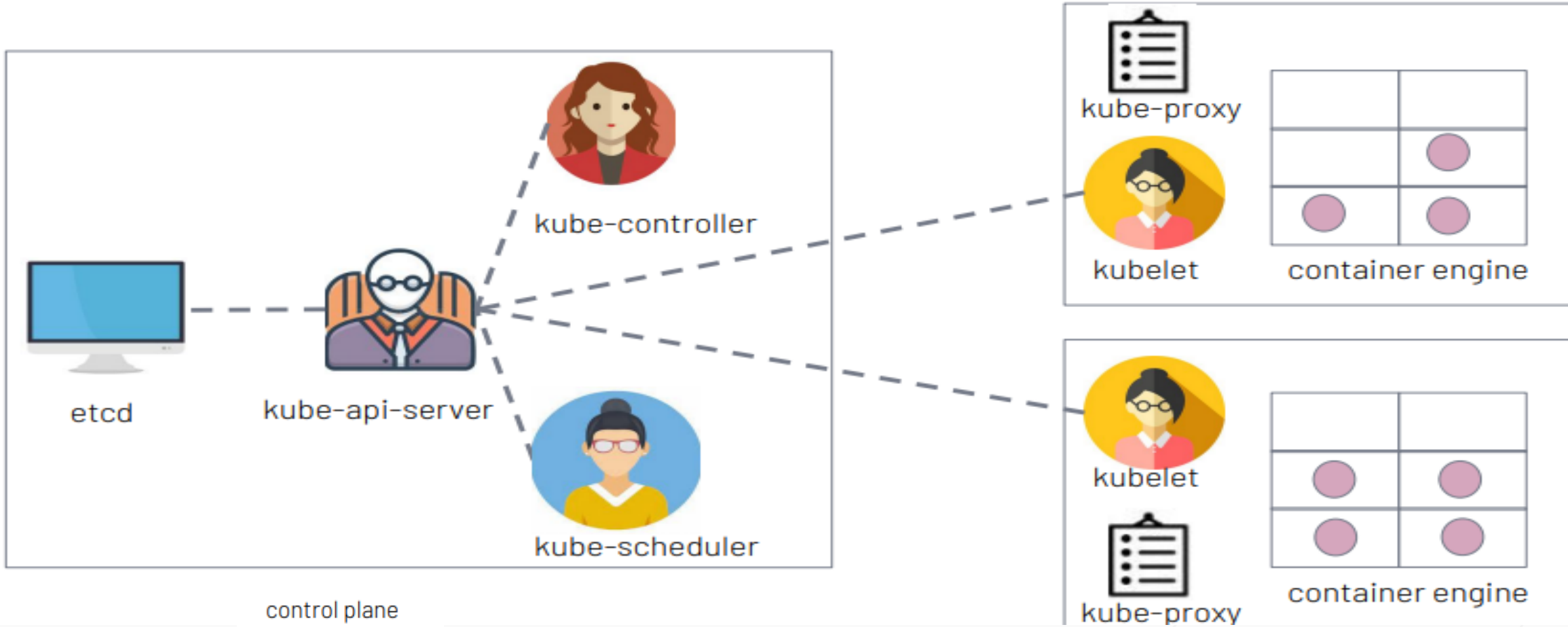
Control Plane Components



Control Plane Components



Control Plane Components

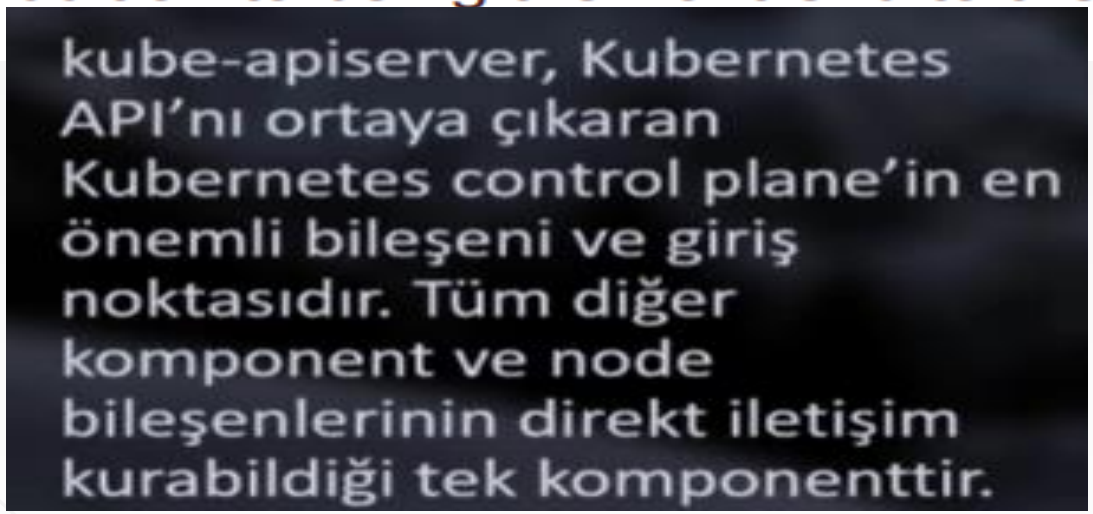



CONTAINER
POD
KUBERNETES

Control Plane Components

kube-apiserver:

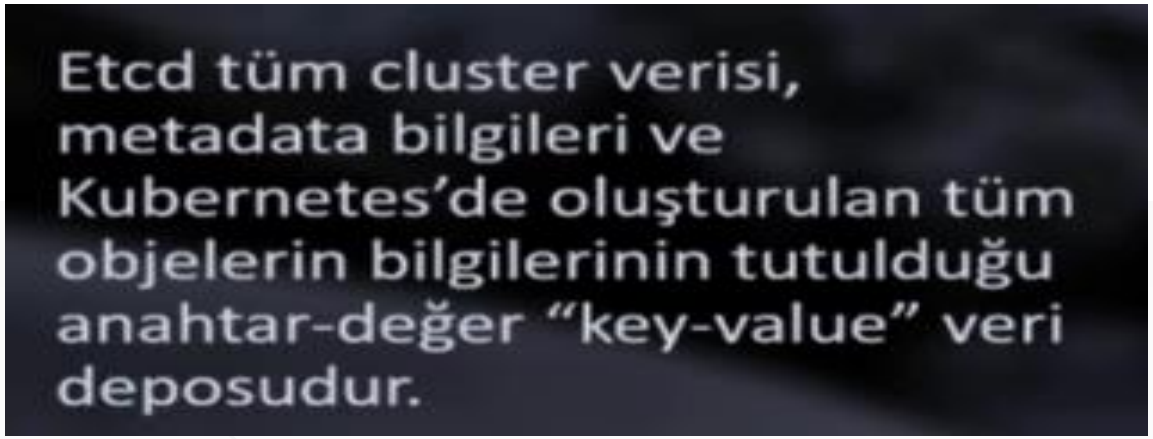
- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes strictly through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.



Control Plane Components

etcd:

- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.



etcd

Control Plane Components

kube-controller-manager:

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and steers the cluster towards the desired state

Control Plane Components

kube-scheduler:

- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

Node Components

kubelet:

- Acts as the node agent responsible for managing the lifecycle of every pod on its host.
- Kubelet understands YAML container manifests that it can read from several sources:
  - file path
  - HTTP Endpoint
  - etcd watch acting on any changes
  - HTTP Server mode accepting container manifests over a simple API.

Node Components

kube-proxy:

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- Available Proxy Modes:
  - Userspace
  - iptables
  - ipvs (default if supported)

Node Components

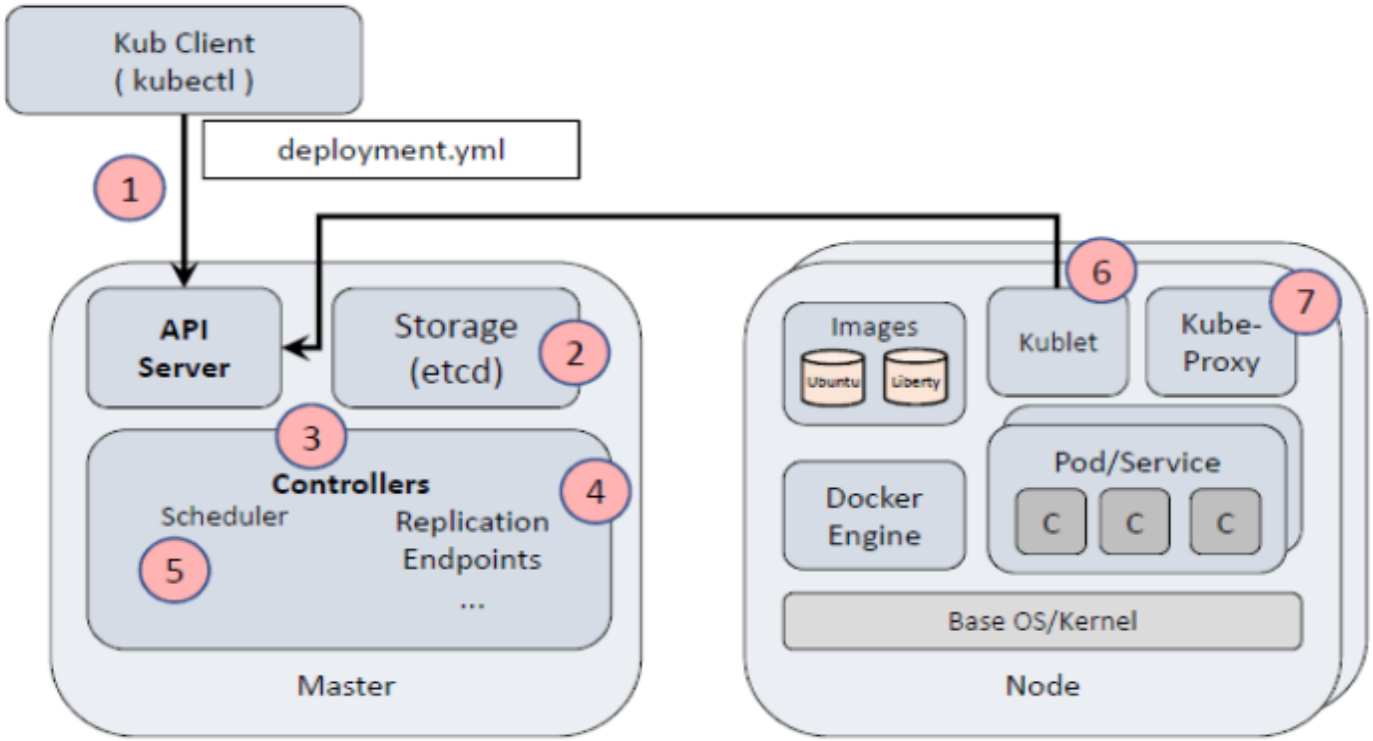
Container Runtime Engine:

- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
  - Containerd (docker)
  - Cri-o
  - Rkt
  - Kata (formerly clear and hyper)
  - Virtlet (VM CRI compatible runtime)

Kubernetes Nasıl Çalışır?

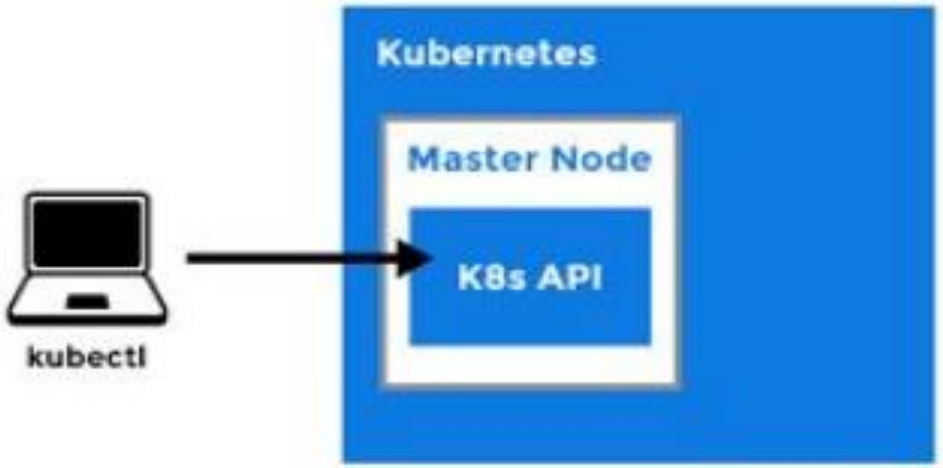
1. Kubectl(kubernetes client) isteği API server'a iletir.
2. API Server isteği kontrol eder etcd'ye yazar.
3. etcd yazdığına dair bilgilendirmeyi API Server'a iletir.
4. API Server, yeni pod yaratılacağına dair isteği Scheduler'a iletir.
5. Scheduler, pod'un hangi server'da çalışacağına karar verir ve bunun bilgisini API Server'a iletir.
6. API Server bunu etcd'ye yazar.
7. etcd yazdığına dair bilgiyi API Server'a iletir.
8. API Server ilgili node'daki kubelet'i bilgilendirir.
9. Kubelet, Docker servisi ile docker soketi üzerindeki API'yi kullanarak konuşur ve konteyner'ı yaratır.
10. Kubelet, pod'un yaratıldığını ve pod durumunu API Server'a iletir.
11. API Server pod'un yeni durumunu etcd'ye yazar.

1. Kullanıcı "**kubectl**" aracılığıyla yeni bir uygulama kuruyor
2. API server isteği alır ve DB'de saklar (**etcd**)
3. İzleyiciler/kontrolörler kaynak değişikliklerini algılar ve buna göre hareket eder
4. ReplicaSet izleyici/denetleyici yeni uygulamayı algılar ve istenen sayıda örnekle eşleşen yeni Pod('lar) oluşturur

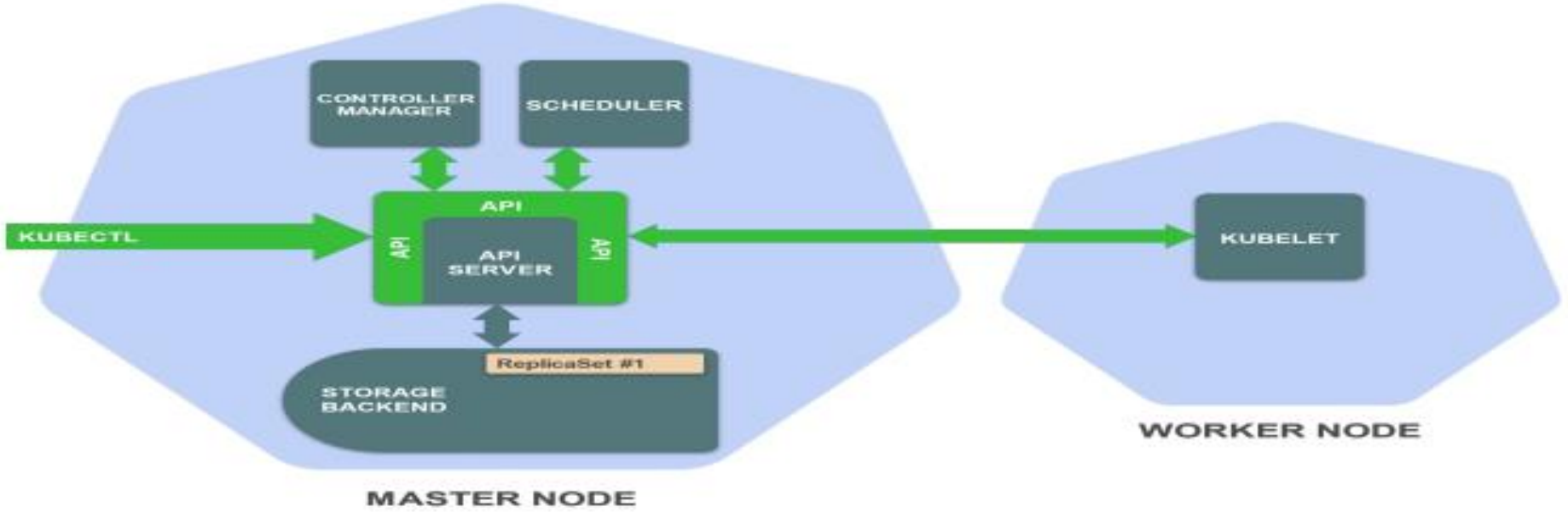


5. Scheduler (Zamanlayıcı), bir kubelet'e yeni Pod('lar) atar
6. Kubelet, Pod'ları algılar ve çalışan container aracılığıyla dağıtır (ör. Docker)
7. Kubeproxy, hizmet keşfi ve yük dengeleme dahil olmak üzere Pod'lar için ağ trafiğini yönetir

kubectl



kubectl



- **kubectl** is (almost) the only tool we'll need to talk to Kubernetes
- It is a rich CLI tool around the Kubernetes API
- Everything you can do with kubectl, you can do directly with the API
- kubectl can be pronounced "Cube C T L", "Cube cuttle", "Cube cuddle".

How to install and use K8s

- kind: https://kind.sigs.k8s.io/
- minikube: https://minikube.sigs.k8s.io/
- katacoda: https://killercoda.com/playgrounds
- play with kubernetes: https://training.play-with-kubernetes.com/
- kubernetes on Docker Desktop
- kubeadm
- kops

Advantages

- Less management overhead
- Automated upgrade of cluster
- Built-in monitoring and logging
- Scalable compute

Alternate k8s distributions

flannel

cilium

vmware NSX