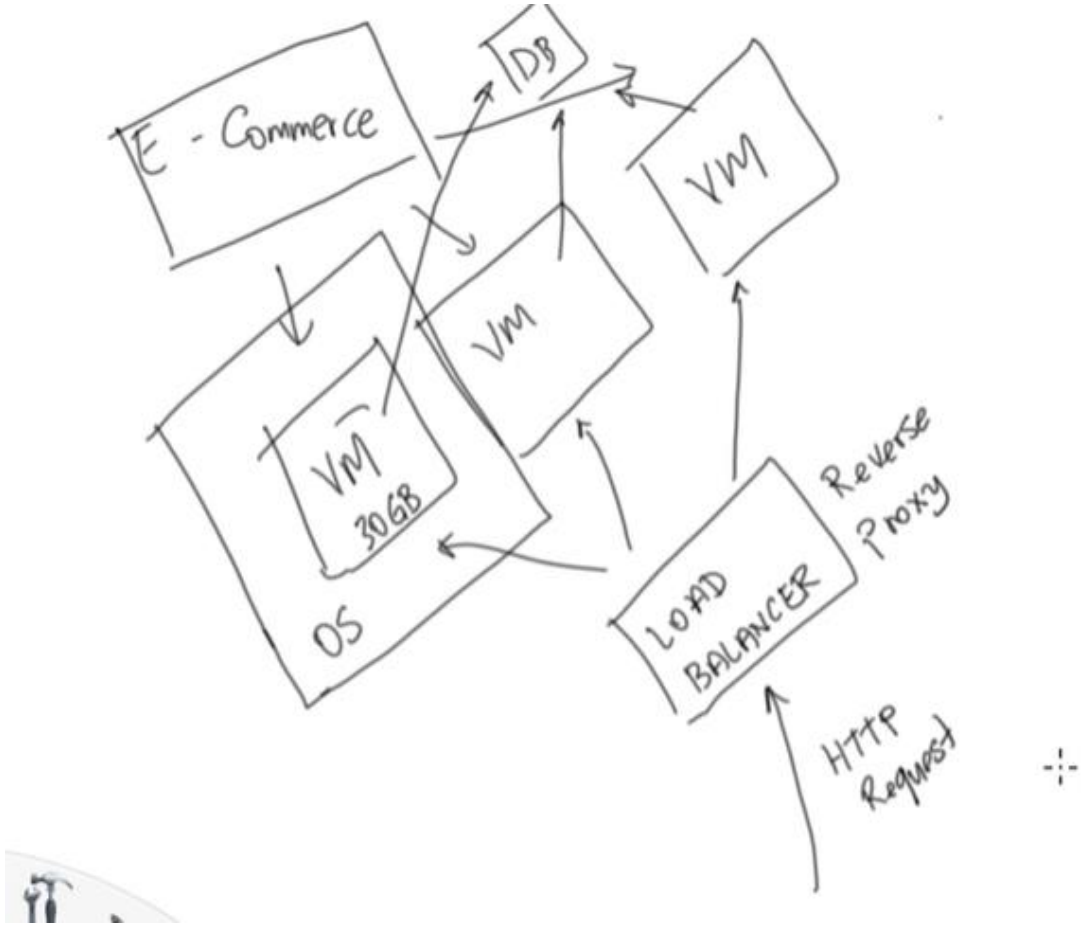
KUBERNETES

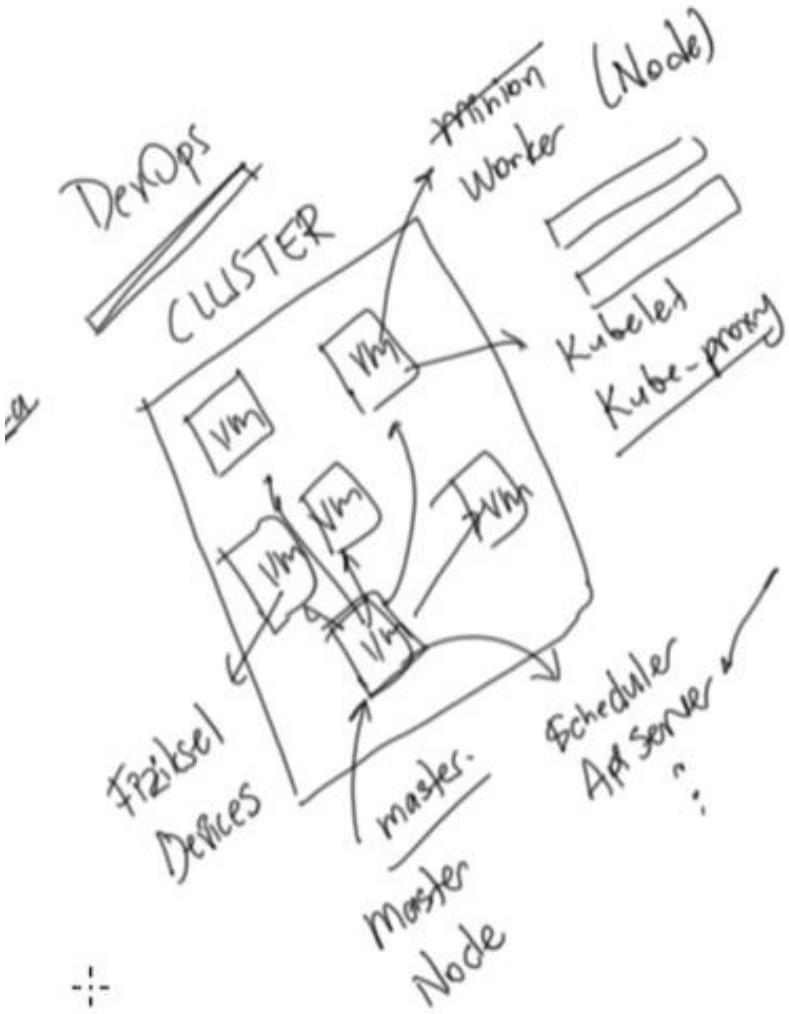
Kubernetes Nedir?

- Konteynerları çalıştırmayı, yönetmeyi, dağıtımlarını otomatikleştirmeyi, giriş oluşturmayı sağlayan açık kaynaklı bir sistemdir.
- Uygulamaları mikro hizmet mimarisinde dağıtmayı, çalıştırmayı kolaylaştırır. Bunu ana bilgisayar üzerinde soyut katman oluşturarak yapar. Böylece geliştirme ekipleri uygulamalarını dağıtabilir veya üzerinde test yapabilir.(Mikro servis mimarisi = Uygulama yönetimi, taşınabilir ortam)
- Konteyner kümeleme aracı.
- Kubernetesin çalışmak için konteynera ihtiyacı vardır.
- Tüm alt yapıyı tek bir parça olarak görmemizi ve yönetmemizi sağlar.

Kubernetesin ana bileşeni cluster (küme). Cluster; **master** ve **worker** adında özel hizmet eden sanal veya fiziksel makinadan oluşur.

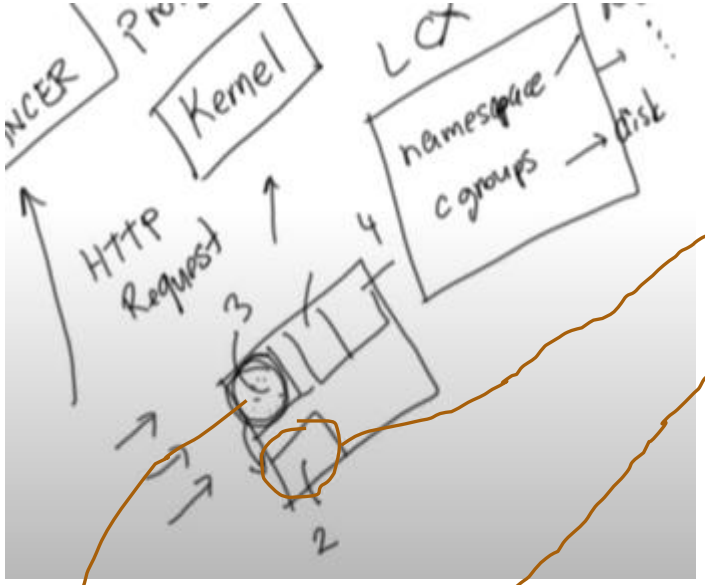


Bir internet sitem var ve bunu kullanıcıya sunmak için sanal makinalar kullanıyorum. Çok fazla alan kaplıyor. İşletim sistemi içerisinde işimize yaramayacak şeyler de var. bu yüzden çok kullanışlı bir yöntem değil.



Kubernetesin sunduğu imkan bu.

Cluster sanki tek bir bilgisayarmış gibi.

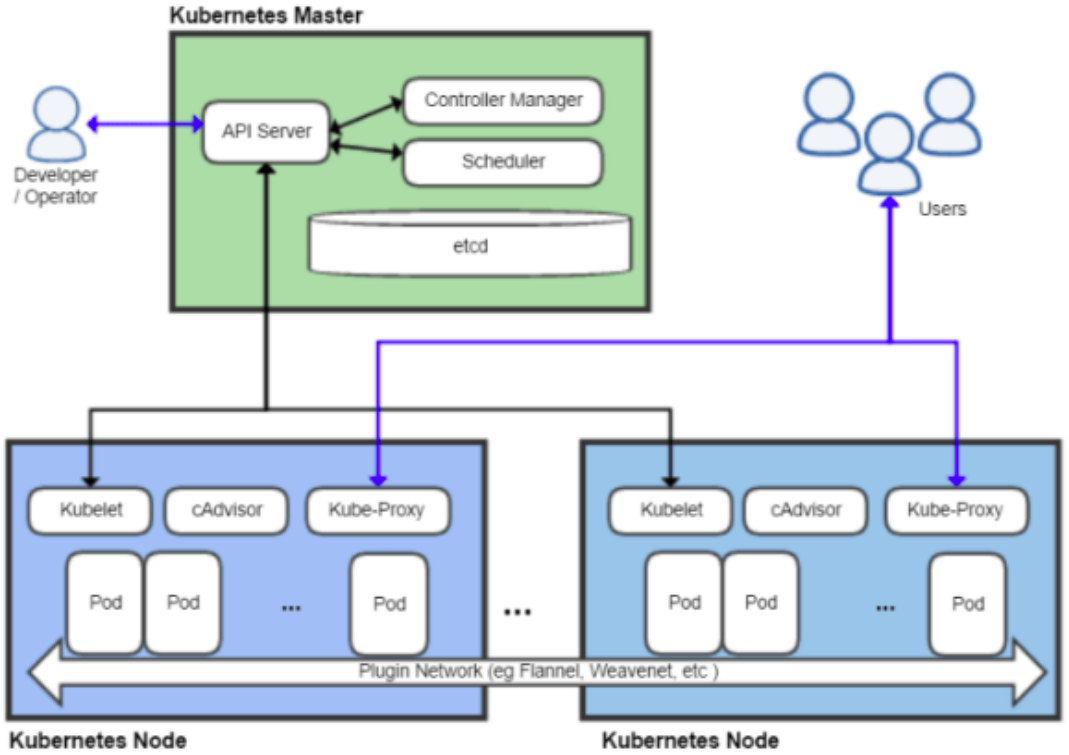


Konteyner

İzole edilmiş ortamlar oluşturuyor tek bir işletim sistemi içerisinde. Ortak çekirdek kullanıyorlar.

Bu konteynerların kolay kullanılması için docker (image) kullanılıyor. Soyutlama yapıyor.

1) Control plane: Master
2) Nodes: where pods get scheduled
3) Pods: hold containers



Kubernetes Mimarisi

- Master ve Worker nodeların kümelenmesinden oluşur.

Worker nodeları kendi arasında overlay networku ile iletişim sağlar.

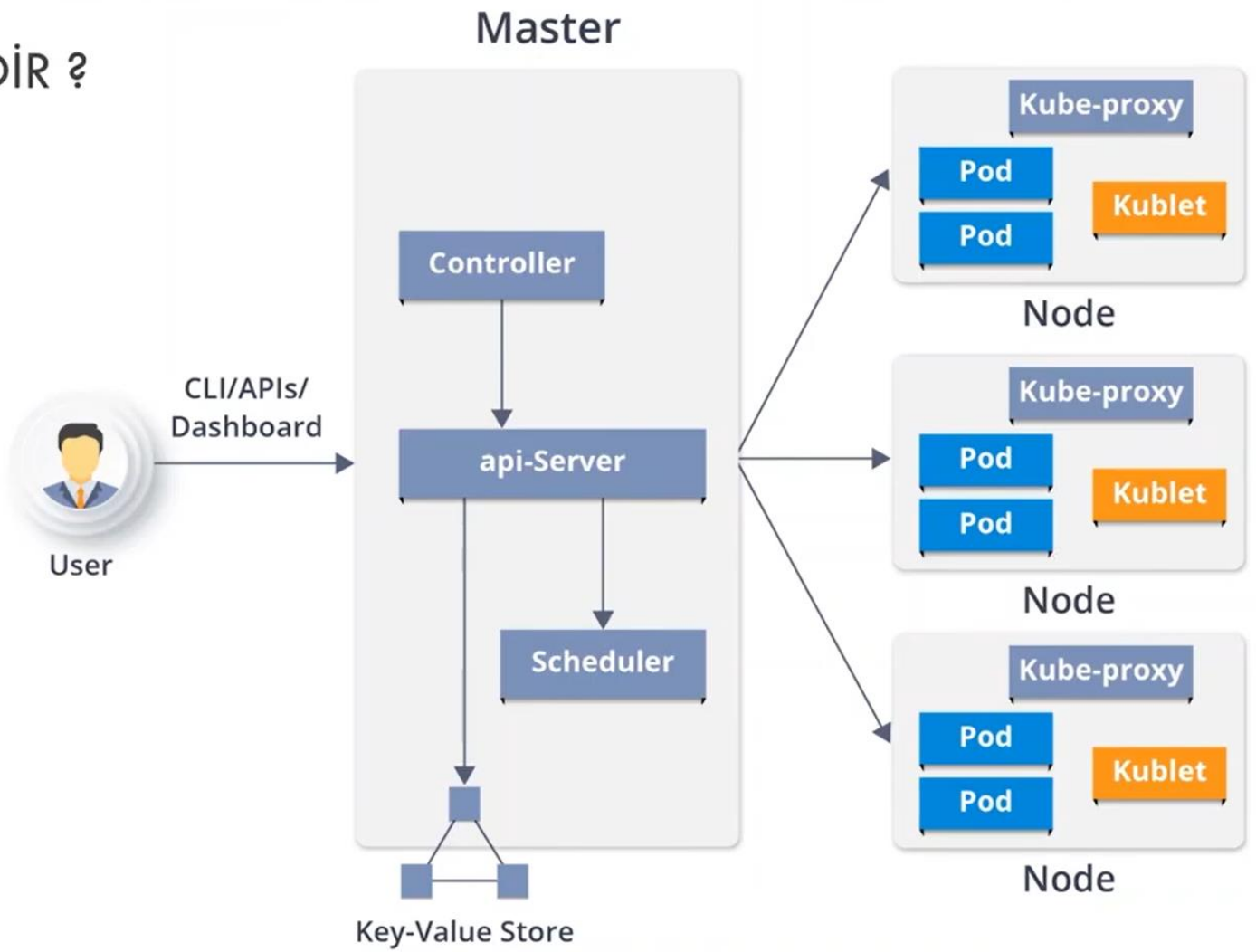
Kubelet-Processlerin çalışmasını sağlar

Workerlarda, içerisinde geliştirilmiş uygulamaların konteynerları vardır. Worker nodelarda uygulama çalışır.

Master nodeda kubernetesle ilgili olan önemli processler çalışır.

KUBERNETES NEDİR ?

- Kubernetes Mimarisi
  Kubernetes Bileşenleri



Master Node

- API Server: Aynı zamanda bir kontaynerdır. Nodelardan gelen tüm istekler burada yönetilir. Sürekli nodelarla iletişim halindedir.
- Controller Manager: API Server aracılığıyla clusterda neler olduğunu kontrol eder.
- Scheduler: Podların hangi nodea yerleştirileceğine karar verir.
- Etcd:Dağıtık mimari ile tasarlanmıştır. (Dağıtık mimari: birden fazla sunucunun birbirleri arasında iletişim kurmasını sağlar). Tüm cluster verisinin saklandığı bileşendir. Verileri key/value database yapısında saklar. Master node ile aynı olan server veya farklı clusterdaki node larda çalışabilir.

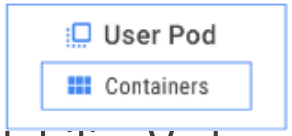
Worker Node

- Pod
- Kubelet: Konteynerların pod içerisinde çalışmasını sağlar. Kubernetes tarafından oluşturulmayan konteynerları yönetemez. API Serverdan gelecek isteklere karşı onu izler, ilgili servisle konuşup podu çalıştırır.
- Kube-Proxy: Podlara IP adresi atar. Podun içerisindeki konteynerlar paylaşımlı IP kullanırlar.
- Container-Engine: Konteyner yönetimi yapar. Start-stop

Pod Nedir?

Worker nodeun birimlerinden birisidir.

Kullanıcıyla interaktif etkileşimde olan en kücük birimdir. Konteynerın çalışma alanıdır.



Pod, konteyner için ambalaj gibidir.

Worker node'unde birden fazla pod olabilir. Ve bu podların içerisinde de birden fazla konteyner bulunabilir.

Genelde uygulama başına 1 poddur. Mesela database bir pod, server bir pod olabilir.

Her poda cluster içerisinde benzersiz IP verilir. Her pod service ve özel IP adrese sahiptir.

Pod içerisinde bir konteyner ölürse, otomatik olarak tekrar başlatılır

Podlar kendi arasında servicelerle iletişim sağlar. Eğer pod silinirse yerine yeni gelen pod da iletişime hemen dahil olur.

Podların istenen durumlarını da YAML veya JSON ile tanımlarsın.

Namespaces: tek clusterda birden fazla sanal cluster oluşturma imkanı sağlıyor. Çok fazla kullanıcısı, geliştiricisi olan projelerde namespace kullanılır.

YAML

```yaml
language: java
jdk:
- oraclejdk8
os:
- linux
script: mvn clean install
before_script:
- sudo apt-get update -qq
- sudo apt-get install -y rpm
deploy:
  provider: releases
  skip_cleanup: true
  on:
    tags: true
    all_branches: true
    repo: rahmanusta/AsciidocFX
```

YAML – JSON Nedir?

YAML: Bağımsız veri değişim formatıdır. Herhangi dilde oluşturulan nesneler YMAL formatında kolaylıkla temsil edilebilir.

JSON: Esnek veri değişim formatıdır. Temel amacı küçük boyutlarda veri alışverişi yapmaktır.

```json
{
    "language": "java",
    "jdk": [
        "oraclejdk8"
    ],
    "os": [
        "linux"
    ],
    "script": "mvn clean install",
    "before_script": [
        "sudo apt-get update -qq",
        "sudo apt-get install -y rpm"
    ],
    "deploy": {
        "provider": "releases",
        "skip_cleanup": true,
        "on": {
            "tags": true,
            "all_branches": true,
            "repo": "rahmanusta/AsciidocFX"
        }
    }
}
```

JSON

Konteyner Nedir?

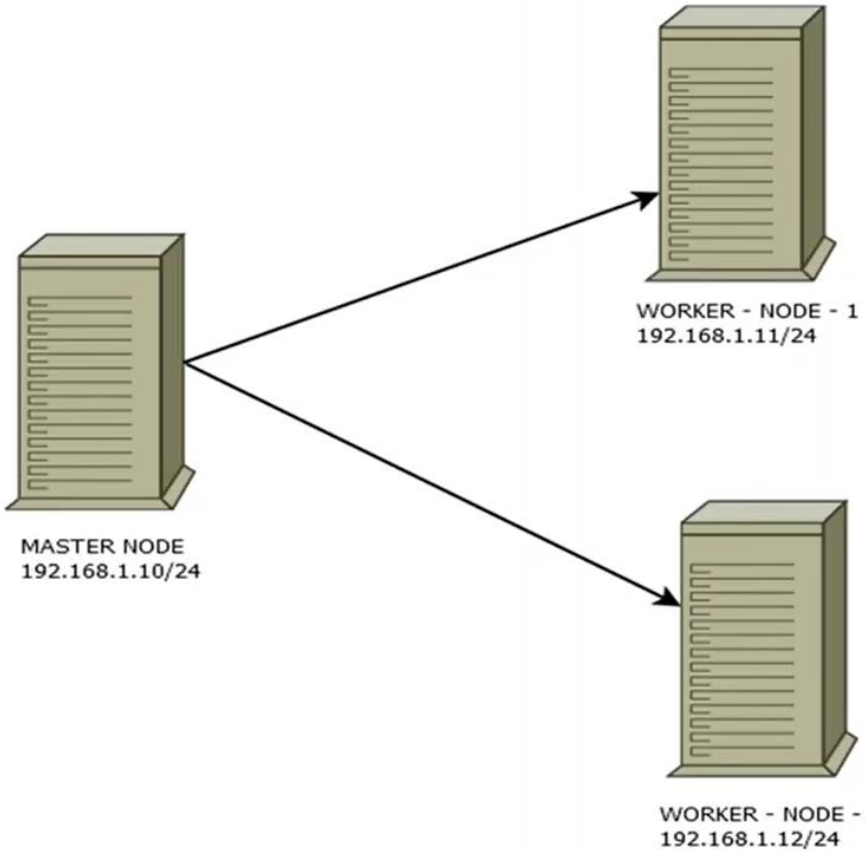
Geliştiricinin uygulamayı kütüphane ve diğer bağımlılıklar gibi ihtiyaç duyduğu tüm parçalarla paketlemesine ve hepsini tek bir paket olarak göndermesine olanak sağlar.

Sunucular arası aktarılan şey konteynerdir.

Podların içerisinde yer alırlar.

Konteynerlar birbirlerinden izole çalışırlar.

Fiziksel sunucu üzerinde halihazırda koşturulmakta olan her bir işletim sisteminin (sanal sunucunun) Docker'daki karşılığı konteynerdır.



Kubernetes Kurulumu

https://kubernetes.io/docs/tasks/tools/install-kubectl/

Kubernetes kurmak için kubectl ve minikube a ihtiyacın var

Kubectl= cluster ile iletişimi mümkün kılan araçlardır.

Minikube = geliştirme yapılan bilgisayar üzerine cluster dağıtımını sağlar.

Kubernetes Cluster Oluşturma(ubuntu)

- Önce server lar oluşturuyoruz.

Server	Hostname
1	kube-01
2	kube-02
3	kube-03

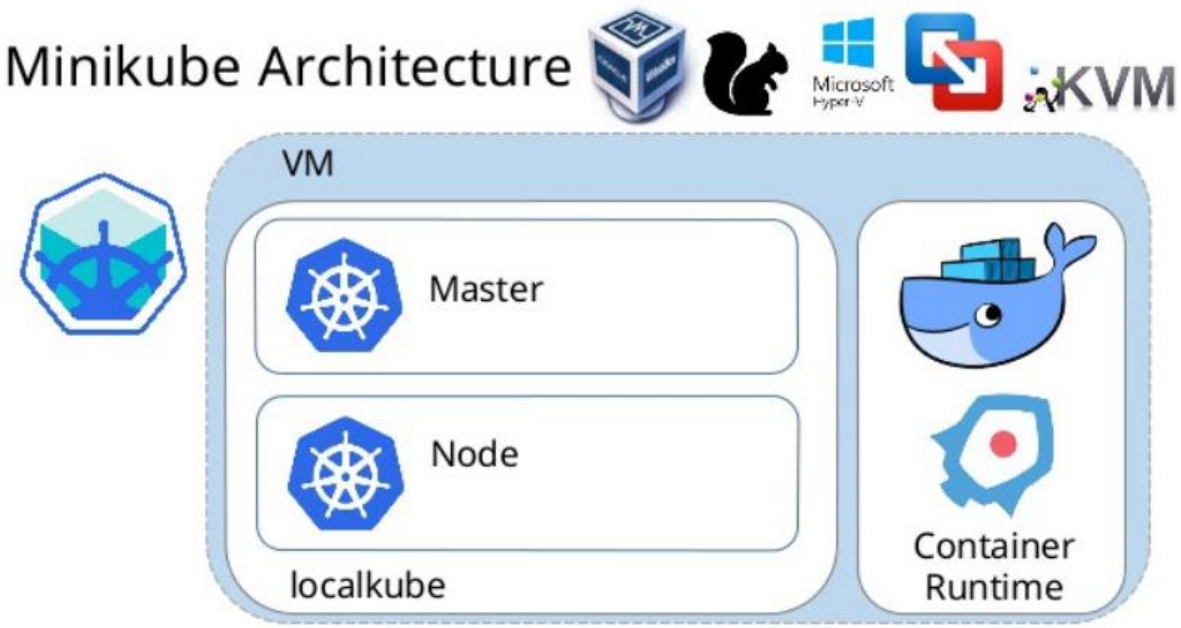
- Serverlardan birisini master olarak tanımlıyoruz

Hostname	Role
kube-01	Master
kube-02	Node
kube-03	Node

- Clusterdaki tüm serverları Kubernetesi çalıştıracak şekilde ayarlıyoruz.
- Master node u kuruyoruz
- Artık istenilen sayıda makinaya katılabilirsin. (kube-02 ve kube-03 mesela)

Kubernetes Çalışma Şekli(pod oluşumu)

1. Kubectl(kubernetes client) isteği API Server a iletir
2. API Server isteği kontrol eder, etcd ye yazar
3. Etcd yazdığına dair bilgili API Server a geri gönderir.
4. API Server yeni pod yaratılacağı isteğini scheduler a iletir
5. Scheduler podun hangi serverda çalışacağına karar verir. Bunun bilgisini API Server a yollar
6. API Server bunu etcd ye yazar
7. Etcd yazıldığına dair bilgiyi API Server a geri gönderir.
8. API Server ilgili node(worker node) daki kubeleti bilgilendirir.
9. Kubelet, Docker servisi ile docker socket üzerindeki API yi kullanarak konteyner yaratır.
10. Kubelet, podun yaratıldığını, pod durumunu API Server a gönderir
11. API Server podun yeni durumunu etcdye yazar.



Kubernetes ortamına bağlanmak ve yönetmek için **kubectl** ye ihtiyaç vardır. O yüzden minikube kurulumu yaparken de kubectl nin bilgisayara kurulması gerekir.

Minikube

Kubernetes'in local ortamda çalışabilmesi için minikube a ihtiyacı vardır.

Minikube, Kubernetes üzerinde testleri ve geliştirmeleri yapmak için local bilgisayardan kullanılan bir clusterdır.

Farklı işletim sistemlerinde çalışabilir.

Mimarisinde Master, node(worker), konteyner kullanır.

Docker

Docker bir konteyner teknolojisidir. Sanallaştırma platformudur.

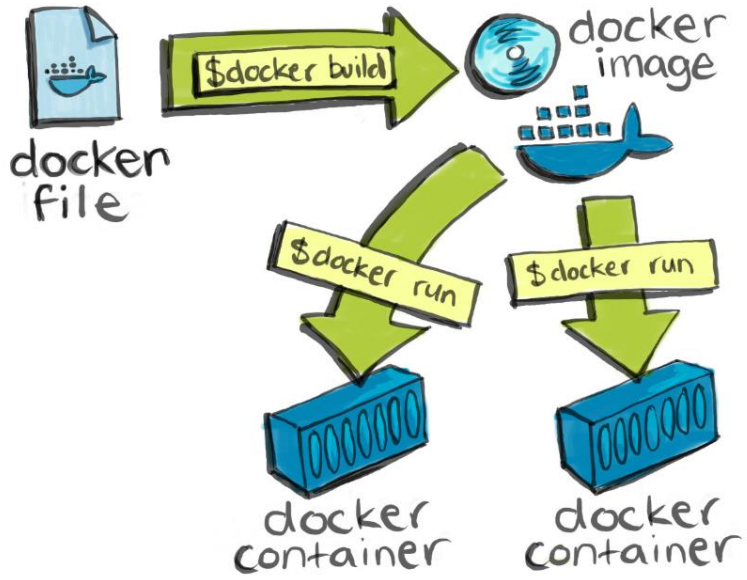
Yazılımların çalışması için her şeyi içeren konteyner şeklinde paketler.

Docker ile farklı sunuculara uygulama gönderip, test edilebilir.

LXC sanallaştırma mekanizması üzerine kurulu.

Konteynerlarda çalıştırılır.

Docker imaj: geliştirdiğiniz ya da test etmek istediğiniz yazılımın tüm verileriyle beraber yedeğini almak.



Docker

Docker file, İçerisinde yayınlanacak olan uygulamanın nasıl bir ortamda çalışacağına dair talimatları barındırmaktadır.

Docker image çalışacak uygulamanızın ve uygulamanızın altyapısında çalışan gerekli işletim sistemi kütüphanelerinin bulunduğu bir yapıdır. Imagei, konteyner yaratmak için gereken talimatların bulunduğu bir şablon olarak düşünebiliriz.

Docker container, imagelerın çalıştırılmış durdurulmuş halidir.

Docker registry, imajların tutulduğu ve dağıtıldığı bir ortamdır. Aynı Github'da olduğu gibi elimizdeki imajları docker registry'sine push edebilir veya daha önceden yüklenmiş olan bir docker imajı kendi localimize çekebiliriz.

Docker Nasıl Çalışır- Kurulum

Konteynerdaki sunucunun işletim sistemini sanallaştırır. Docker her sunucuya yüklenir. Konteyner oluşturma başlatma gibi basit komutlar kullanılır.

-Repoyu güncelle

$ sudo apt-get update

-Docker'ı yüklemek için bu komutu kullan

$ sudo apt-get install -y docker.io

-Servisin çalıştığından emin olmak için bu komut

$ sudo systemctl status docker

Konteynerlar sanki kendi sanal makinalarında çalışıyorlarmış gibi davranırlar.

Dockerfile, image oluşturmak için kullanılan talimatların listesidir. image, işlemin içinde çalıştığı ortam ve konteyner çalışan bir imagedir.

Geliştirme ortamında dockerfile paylaşılır. Diğer kişi o dosyayı alıp image oluşturabilir ve seninle aynı geliştirme ortamına sahip olabilir.

Test senaryomuzu Node.Js ile hazırlanmış bir web uygulamasını , Dockerfile ile konfigüre ederek bir image haline getirip , docker hub üzerinde yayınlamak olarak tanımlayabiliriz.

Bizim yapmaya çalıştığımız şey bu?

Docker ve Kubernetes arasındaki fark

- Kubernetes bir çok cluster (küme) çalıştırırken Docker sadece bir node üzerinde çalışır. Clusterda en az bir master node ve worker nodelar oluyordu. Yani Kubernetes birden fazla server üzerinde çalışıyor demek.

Konteyner Düzenleme Araçları Faydaları

- Her zaman kullanıcının erişebilir durumda olması. High availability
- Uygulama hızlı yükleniyor, kullanıcı yüksek verim alıyor. High performance
- Data veya serverın zarar görmesi durumunda kurtarılabilir olması.

https://vimeo.com/245778144/4d1d597c5e