T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI

#MİLLİ TEKNOLOJİ HAMLESİ

TÜBİTAK

# K8s & UKS

kubernetes

,TÜBİTAK ULAKBİM

TÜBİTAK

# İÇİNDEKİLER

TÜBİTAK

# KONTEYNERİZASYON

MODERN UYGULAMA STANDARTI

# Gündemimiz ve Hedefler

Neden Değişim ?

Kubernetes Nedir?

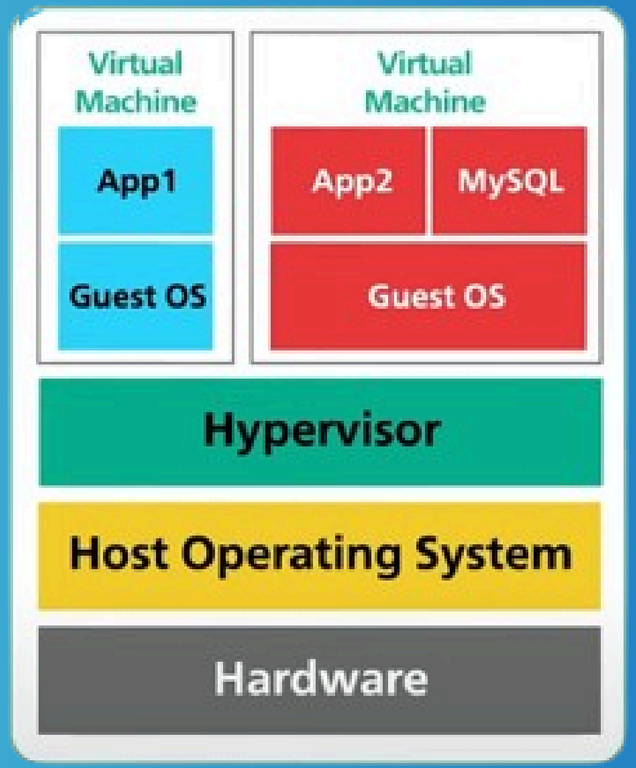
Docker’dan Farkı Nedir?

OpenStack Entegrasyonu ve Demo

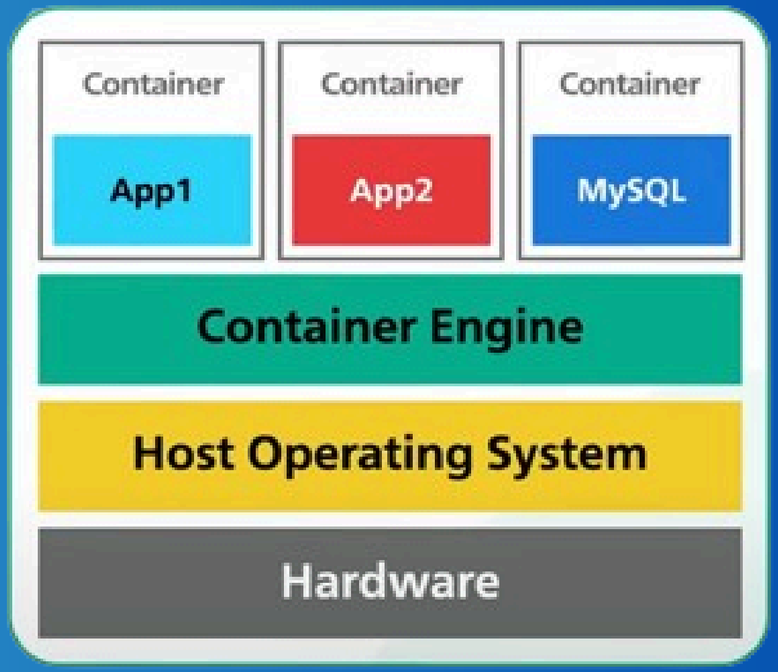
# Uygulama Dağıtımının Evrimi

## Bare Metal


## Virtualized




## Containerized



TÜBİTAK

# Konteyner Teknolojisi: Yazılımın Lojistik Devrimi

## STANDARTLAŞMA

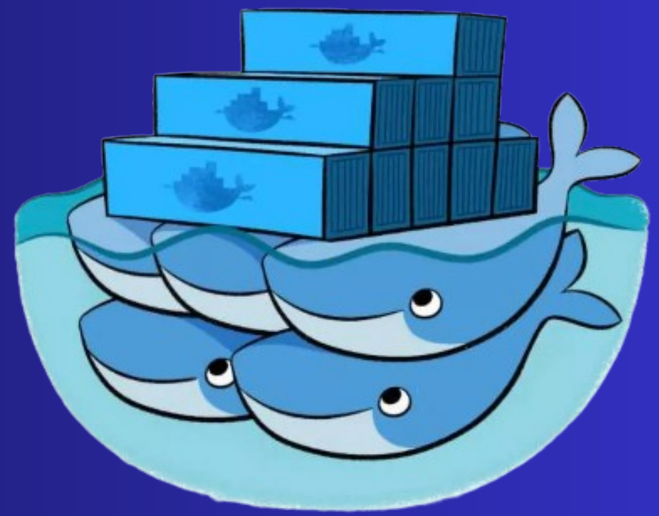
İçinde ne olduğundan bağımsız (Java, Python, .NET), dış dünyayla iletişimi standarttır.

## İZOLASYON

Her uygulama kendi kütüphane ve bağımlılıklarıyla, diğerlerinden etkilenmeden çalışır.

## HAFİF YAPI

İşletim sistemi çekirdeğini paylaştığı için çok düşük kaynak tüketir.



TÜBİTAK

# "Benim Makinemde Çalışıyordu" Problemine Son

¯_(ツ)_/¯
IT WORKS
ON MY MACHINE

## TAŞINABİLİRLİK

Geliştiricinin bilgisayarındaki ortamın %100 aynısı sunucuda da çalışır.


## DEĞİŞMEZLİK

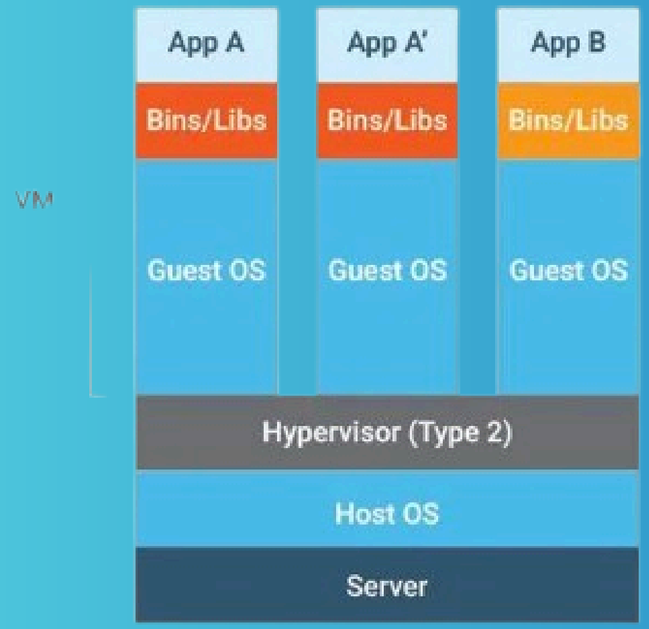
Çalışan bir konteyner üzerinde yama yapılmaz; yeni versiyon yeni bir paket olarak yayına alınır.

## HIZLI GERİ DÖNÜŞ

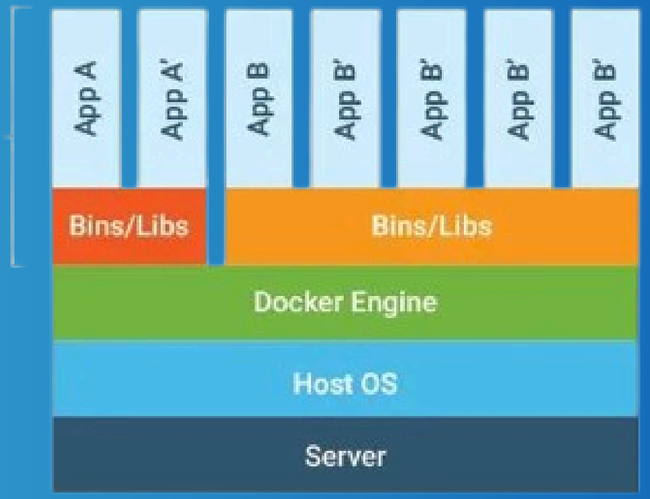
Bir hata mı çıktı? Saniyeler içinde eski çalışan pakete dönülebilir.

# Sanal Makine (VM) ve Konteyner Farkı

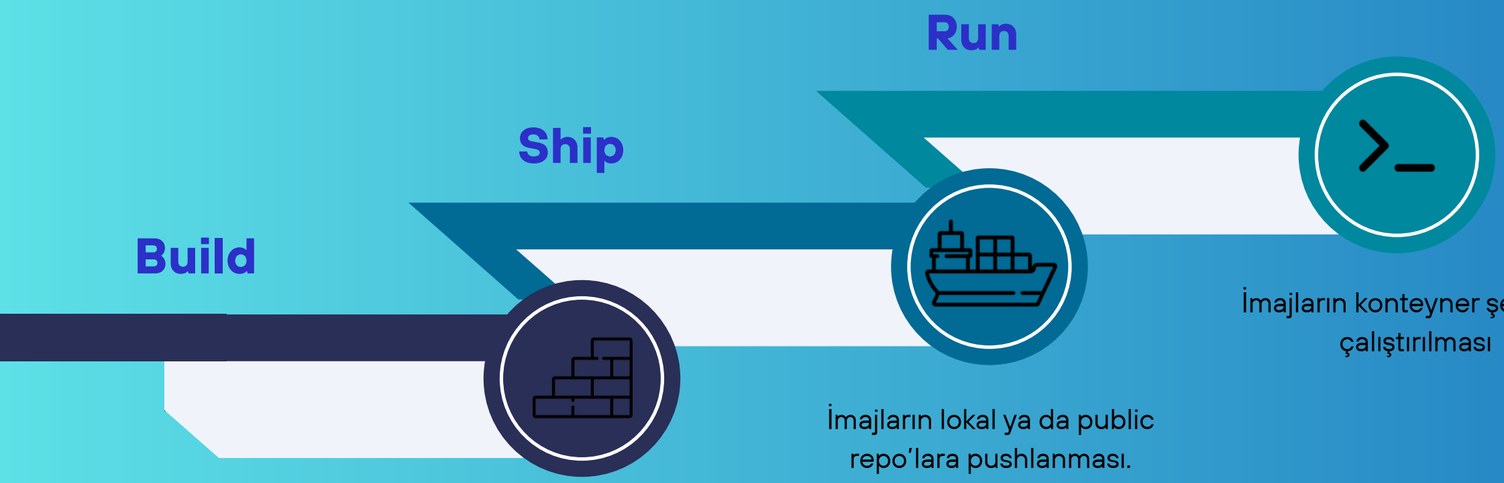
**Her uygulama için ayrı bir İşletim Sistemi (GB'larca veri ve RAM kaybı).**

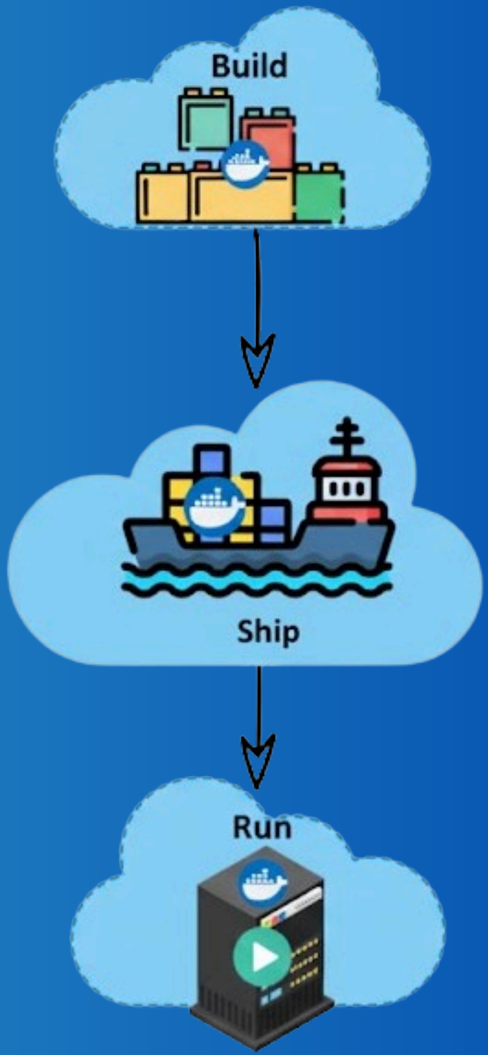



**Tek bir İşletim Sistemi üzerinde paylaşımlı mimari.**



# Docker Ekosistemi: Build, Ship, Run

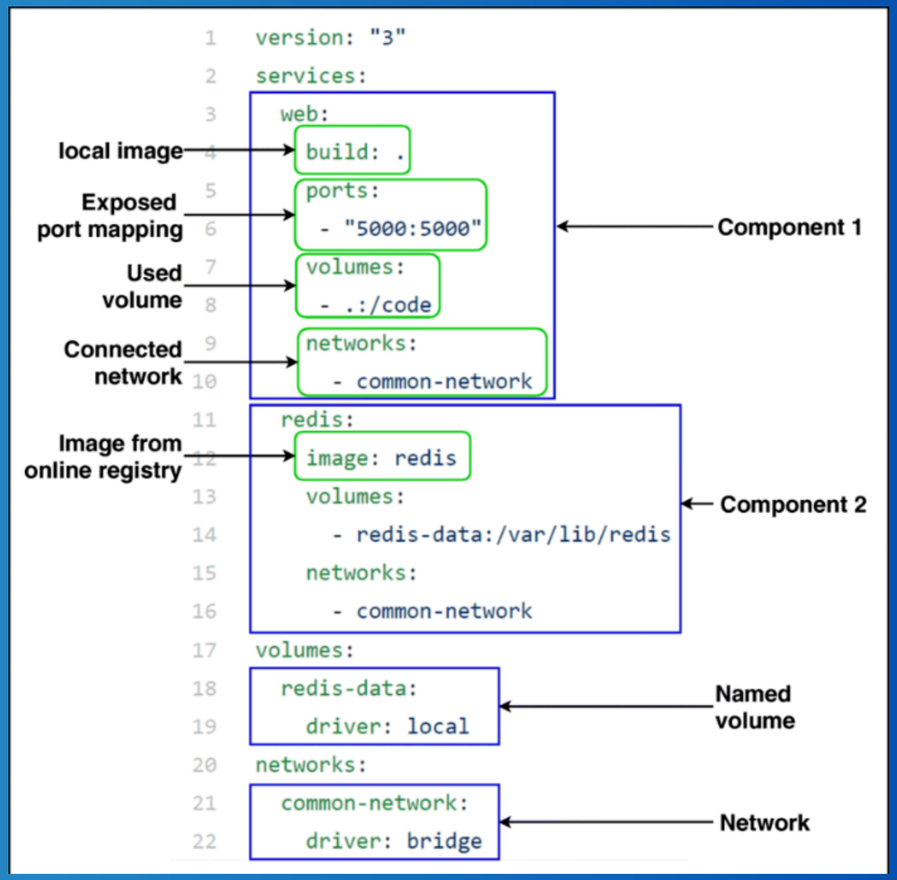


# Docker Compose: Yerel Geliştirmenin Sınırı

## Yetersiz Kalınan Noktalar

- Otomatik İyileştirme
- Farkı Sunuculara Yayılma
- Zero-Downtime



TÜBİTAK

# KUBERNETES

OPERASYONEL MÜKEMMELLİK

TÜBİTAK

# Kubernetes: Modern Altyapınızın Uyum İçinde Çalışn Orkestra Şefi

## TANIM

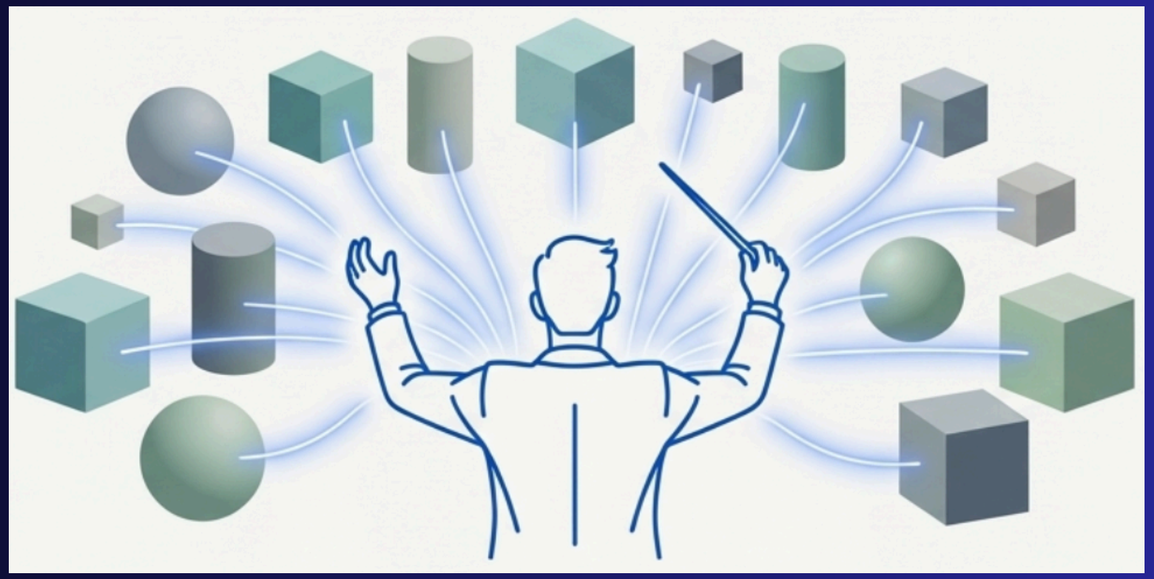
Konteynerize uygulamaların dağıtımını, ölçeklenmesini ve yönetimini otomatize eden platformudur.

## KÖKEN

Google'ın 15 yıllık üretim tecrübesiyle (Borg projesi) geliştirilmiştir

## PAZAR STANDARDI

Bulut bilişim dünyasının fiili "İşletim Sistemi" olarak kabul edilir



Docker bize birer enstrüman verdi, Kubernetes ise bu enstrümanların uyum içinde çalmasını sağlayan orkestra şefidir. Artık tek tek sunucularla değil, bir bütün olarak 'servis' ile ilgileniyoruz.

# Mimari Bakış: Kararları Alan Bir ‘Beyin’ ve İşi Yapan ‘Kaslar’

## CONTROL PLANE (BEYİN)

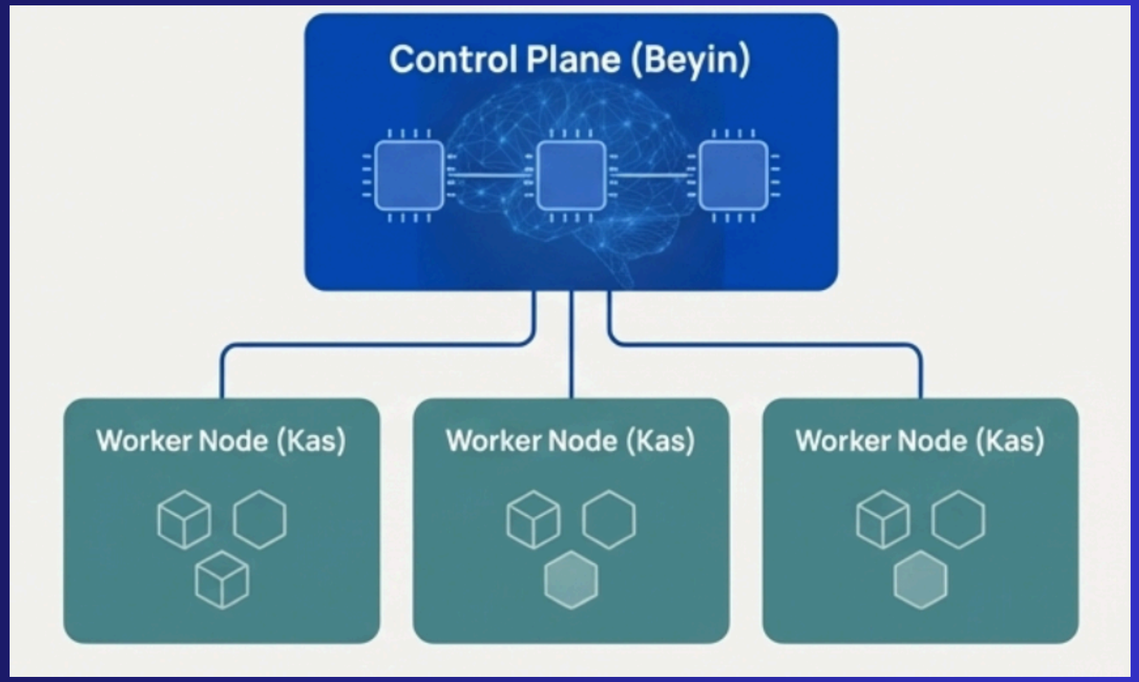
Sistemin beynidir. Kararları alır, sağlığı izler, talimatları dağıtır ve tüm orkestrayı yönetir

## WORKER NODES (KASLAR)

Sistemin beynidir. Kararları alır, sağlığı izler, talimatları dağıtır ve tüm orkestrayı yönetir

## HIGH AVAILABILITY

Yönetim katmanı (beyin) yedeklidir; sunuculardan (worker node) biri kapasa dahi sistem çalışmaya devam eder



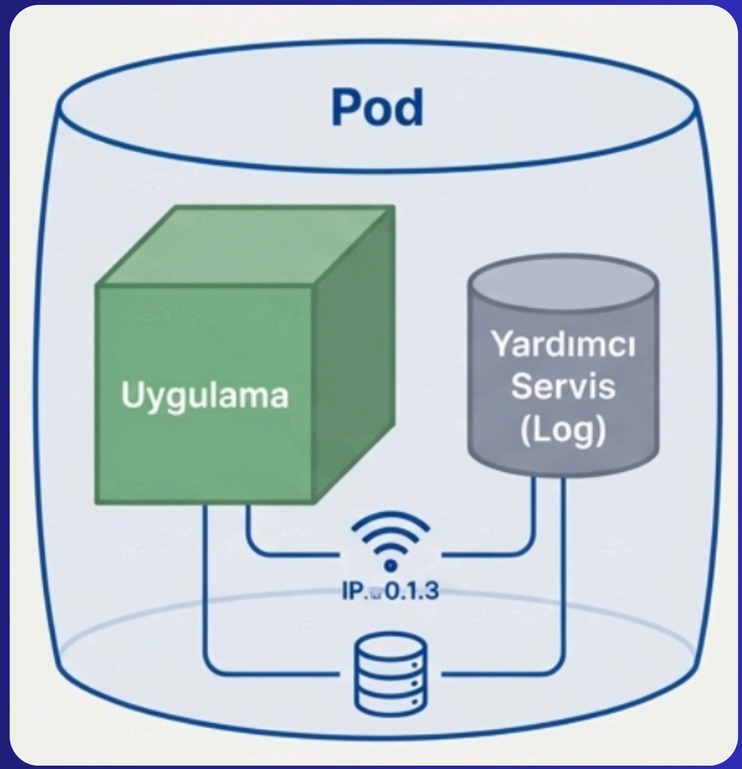

Resmi kurumlar için önemli konu sürekliliktir. K8s mimarisi, ‘tek bir hata noktasını’ ortadan kaldırır. Sunucu bozulabilir ancak master bunu anlar ve iş yükünü sağlam sunuculara anında kaydırır

# Pod: Uygulama Bileşenlerinizi Birleştiren Akıllı Kapsül

Kubernetes'in yönettiği en küçük dağıtım birimidir.

İçinde birbirleriyle sıkı sıkıya bağlı bir veya birden fazla konteyner barındırabilir.

Ortak Kaynaklar: Pod içindeki konteynerler aynı ap (IP adresi) ve depolama alanını paylaşırlar, bu da onların iletişimini kolaylaştırır.




K8s direkt konteynerlerle değil, 'Pod' adını verdiğimiz kapsülle çalışır. Bu bize, bir uygulamayı yardımcı servisleriyle birlikte (örneğin log toplama) tek bir paket olarak yönetme esnekliği sağlar.

# Bildirimsel Yapı: Sisteme Emir Vermez, Ulaşması Gereken Hedef Tanımlanır

## HEDEF BELİRLEMEK

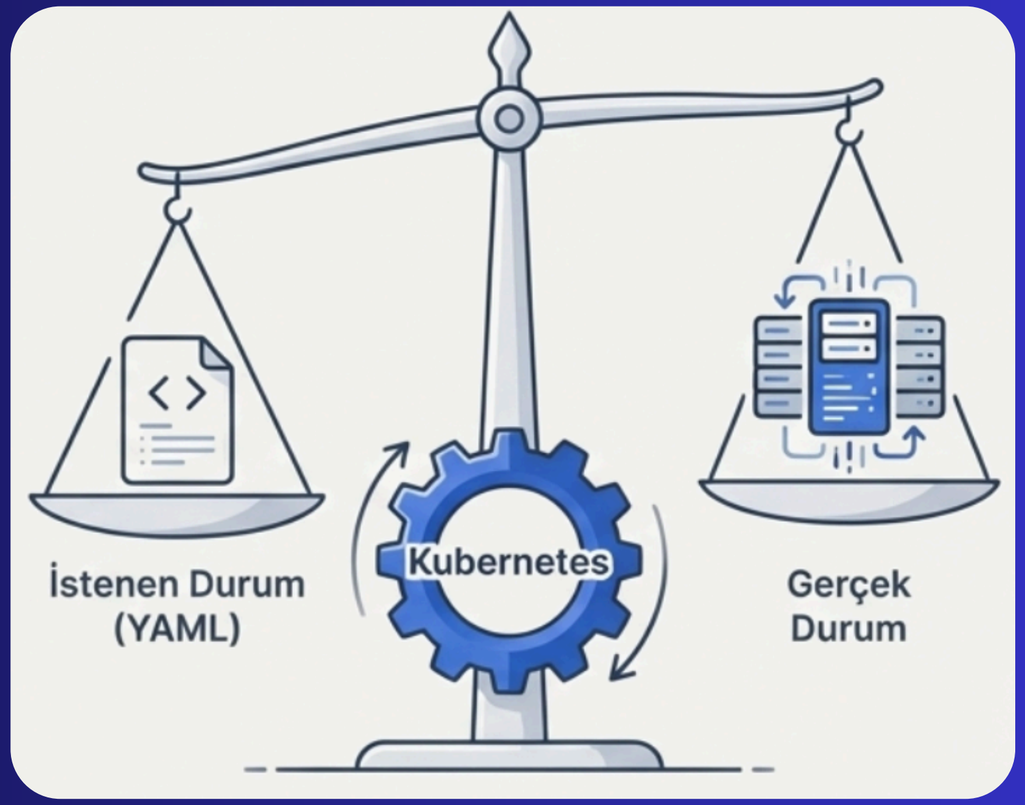
"3 adet web sunucusu çalıştır' demek yerine, "Sistemimde her zaman 3 adet web sunucusu çalışır halde olsun" dersiniz.

## YAML DOSYALARI

İstenen bu durum, insan tarafından okunabilir YAML dosyaları ile "Kod Olarak Altyapı" (Infrastructure as Code) şeklinde tanımlanır

## SÜREKLİ UZLAŞMA (RECONCILIATION LOOP)

Kubernetes, mevcut durumu sürekli olarak sizin tanımladığınız istenen durumla karşılaştırır ve aradaki farkı kapatmak için durmaksızın çalışır.




Bir sistem yöneticisine 'sunucuyu güncelle' demek yerine, sisteme YAML dosyasını verirsiniz. K8s o dosyayı okur ver sistemi o hale getirir. Bu, insan hatasını neredeyse sıfıra indirir.

TÜBİTAK

# Self Healing: Sistem Kendi Kendini İyileştirir, İnsan Müdahelesine Gerek Kalmaz

**Sağlıklı** → **Hata Algılandı** → **Otomatik Müdahele** → **Sistem İyileştirildi**

### Liveness Probes

Uygulama sağlıklı mı? Kubernetes, yanıt vermeyen uygulamaları otomatik olarak sonlandırır ve yerlerine yenilerini başlatır

### Readiness Probe

Uygulama trafik almaya hazır mı? Tam olarak hazır olmadan kullanıcı trafiğini ona yönlendirmez, böylece hatalı istekler engellenir

### Node Failure

Fiziksel bir sunucu çökerse, üzerindeki tüm uygulamalar (Pod'lar) saniyeler içinde başka bir sağlıklı sunucuda yeniden canlandırılır

Gece saat 3'te bir servis çöktüğünde kimsenin uyanmasına gerek yok. Kubernetes hatayı algılar, bozuk parçayı çöpe atar ve tertemiz bir kopyayı saniyeler içinde ayağa kaldırır.

# Roll(Back/Up): Güncellemeler Sıfır Kesintiyle Yapılır, Kullanıcılar Fark Etmez

## SIFIR KESİNTİ (ZERO DOWNTIME)

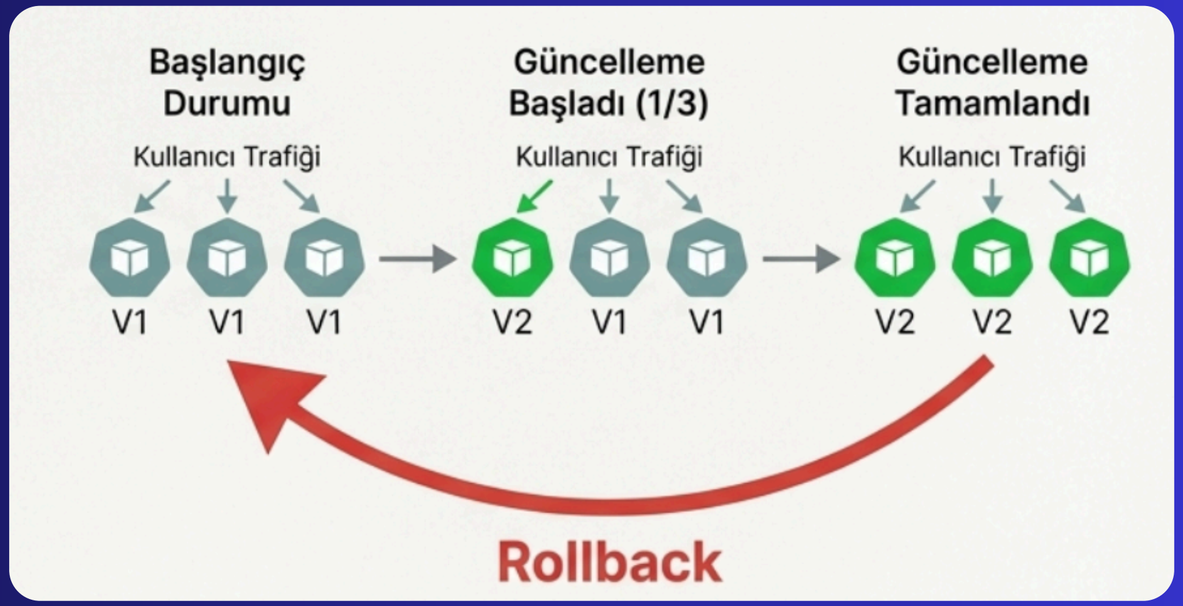
Yeni sürüm dağıtılırken, eski sürümün sağlıklı kopyaları kullanıcı trafiğini karşılamaya devam eder

## ADIM ADIM GEÇİŞ (ROLLING UPDATE)

Kubernetes önce tek bir kopyayı günceller, sağlığını kontrol eder. Başarılı olursa, diğer kopyaları da sırayla günceller.

## GERİ DÖNÜŞ (ROLLBACK)

Yeni sürümde bir sorun mu tespit edildi? Tek bir komutla saniyeler içinde güvenli olan eski sürüme geri dönülür.




Kullanıcılar siz güncelleme yaparjen ‘Bakım çalışması var’ ekranı görmez. Gün içinde, iş saatlerinde bile güvenle yeni özellikler yayınlanabilir.

TÜBİTAK

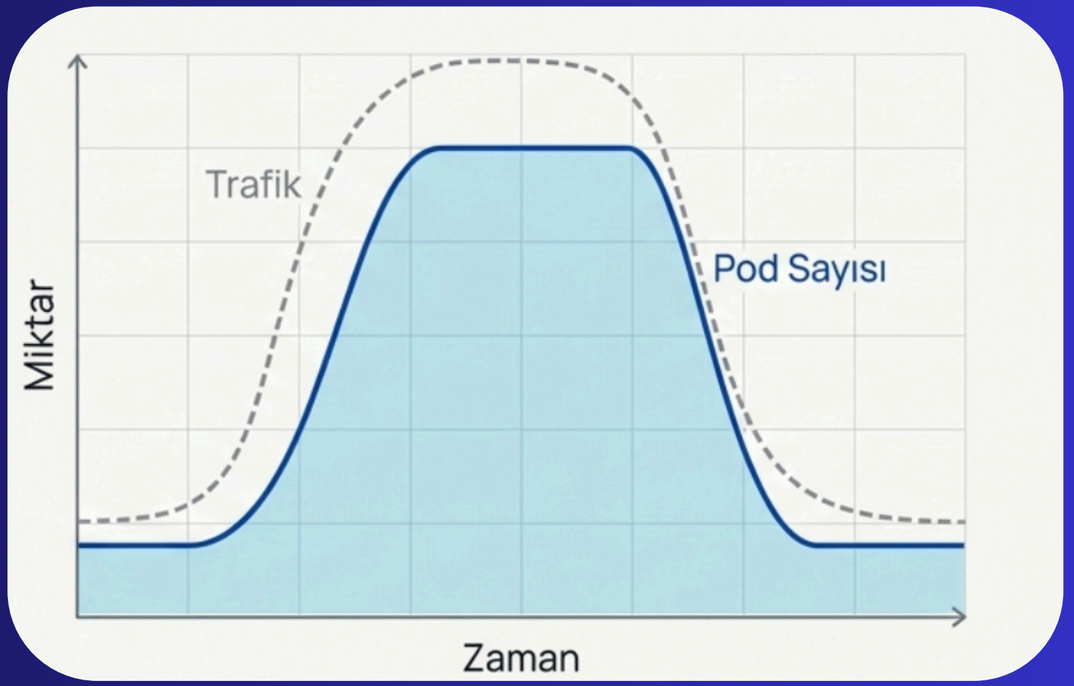
# Kaynak Ölçekleme: Sistem İhtiyaca Göre Otomatik Olarak Büyür ve Küçülür

## HORIZONTAL POD AUTOSCALER (HPA)

CPU veya bellek (RAM) kullanımı gibi metrikler belirli bir eşiği aştığında, Kubernetes uygulamanın kopya sayısını otomatik olarak artırır.

## VERİMLİLİK VE MALİYET OPTİMİZASYONU

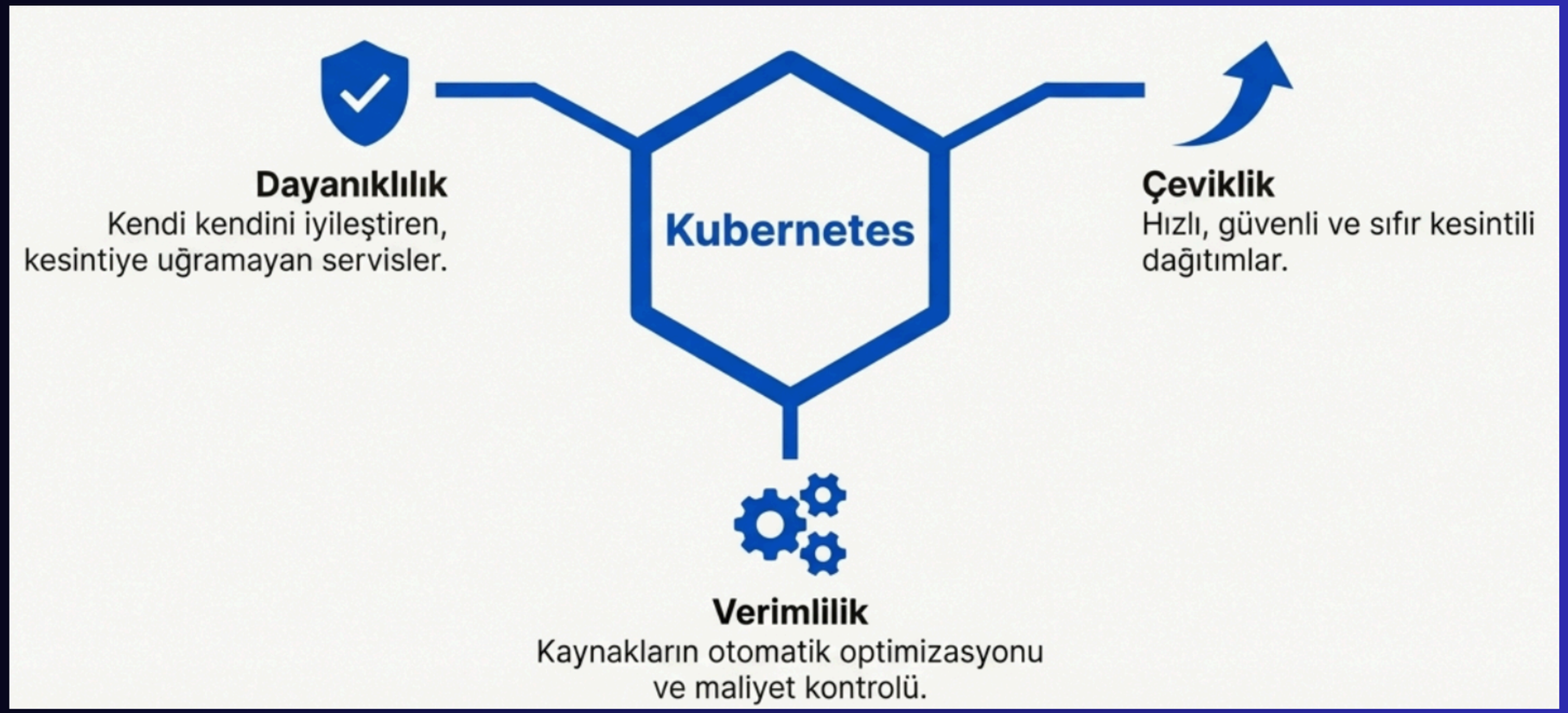
Yoğunluk azaldığında, fazla kopyalar otomatik olarak kapatılır. Bu sayede, sadece ihtiyaç duyulan kaynaklar kullanılır ve gereksiz maliyetler önlenir.




Trafiğin yoğun olduğu dönemlerde veya beklenmedik durumlarda sistemimiz dar boğaz yaşamaz. İhtiyaç kadar kaynak kullanır, böylece donanım maliyetlerimizi de optimize etmiş oluruz.

# Sonuç: Kubernetes Sadece Bir Araç Değil, Operasyonel Mükemmeliğe Giden Yoldur

Kubernetes, altyapı yönetimini paradigmasını değiştirir. Ekiplerin sürekli ‘’yangın söndürmek’’ yerine, değer yaratan işlere odaklanmasını sağlar: Bu şu anlama gelir:



TÜBİTAK


kubernetes

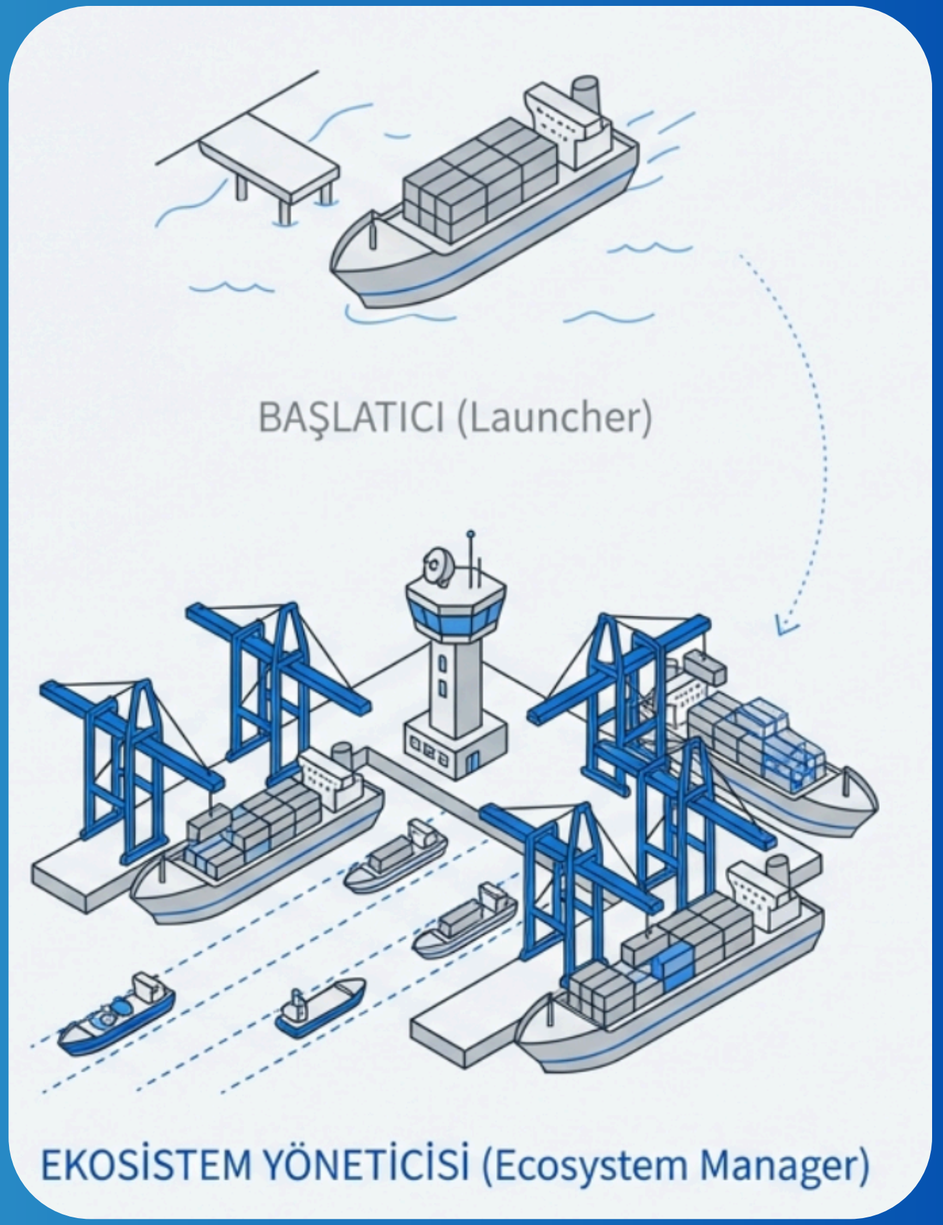
# KUBERNETES'E GÖÇ

COMPOSE TO K8S

# Yolculuğumuz: Konteynerleri Çalıştırmaktan Ekosistemleri Yönetmeye

**Docker Compose'dan Kubernetes'e Stratejik Geçişin Nedenleri ve Yolu**

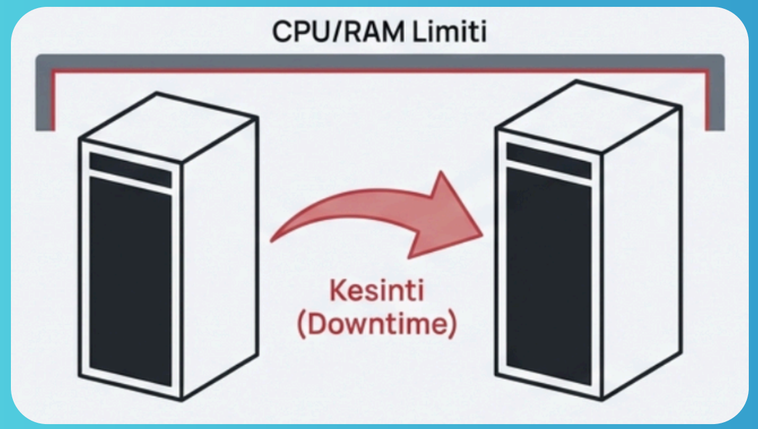
Docker Compose ile harika bir başlangıç yaptık ve uygulamalarımızı hızlı hayata geçirdik. Artık büyüdük ve ihtiyaçlarımız da bizimle birlikte evrildi. Neden ve nasıl Kubernetes'e geçmemiz gerekiyor ?



# Ölçeklenebilirlik Duvarı: Büyümenin Sonu mu, Yeni Bir Başlangıç mı ?

**Docker Compose**

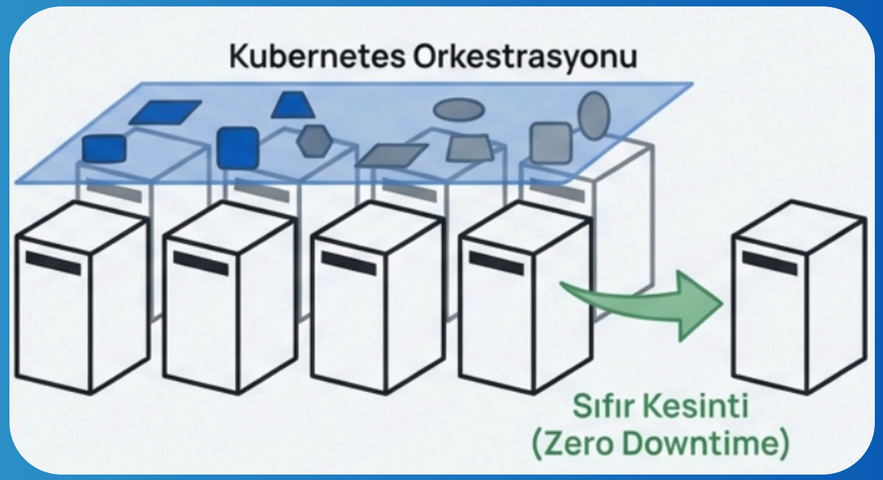
**Dikey Büyütme (Vertical Scaling)**




Tek sunucuya bağımlıdır. Yükseltme işlemi, uygulamanın durdurulmasını gerektirir. Bu bir ‘’başlatıcıdır’’.

**Kubernetes**

**Yatay Büyüme (Horizontal Scaling)**



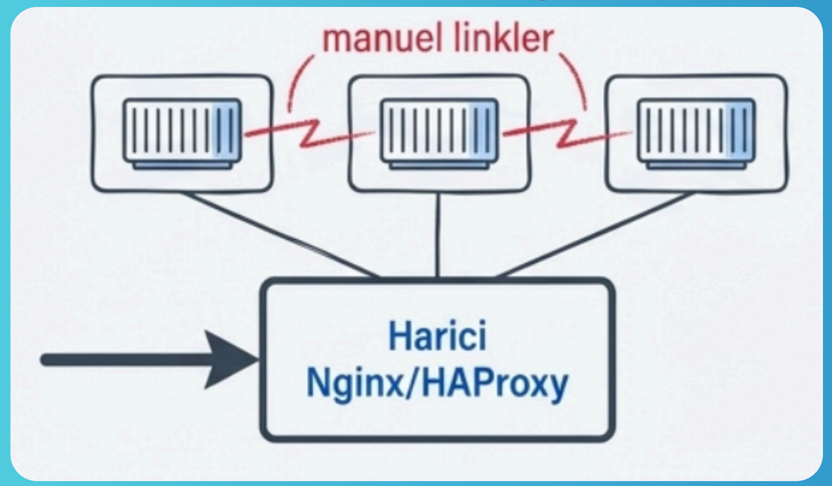

Sunucuları tek bir havuz olarak yönetir. Yeni kaynaklar kesintisiz eklenir ve iş yükü otomatik dağıtılır. Bu bir ‘’ekosistem yöneticisidir’’.

Dikey büyüme sonludur ve risklidir; yatay büyüme ise esnek ve güvenilirdir

# Servisler Birbirini Nasıl Bulur? Manuel Yapılandırmadan *Otomatik Keşfe*

**Docker Compose**

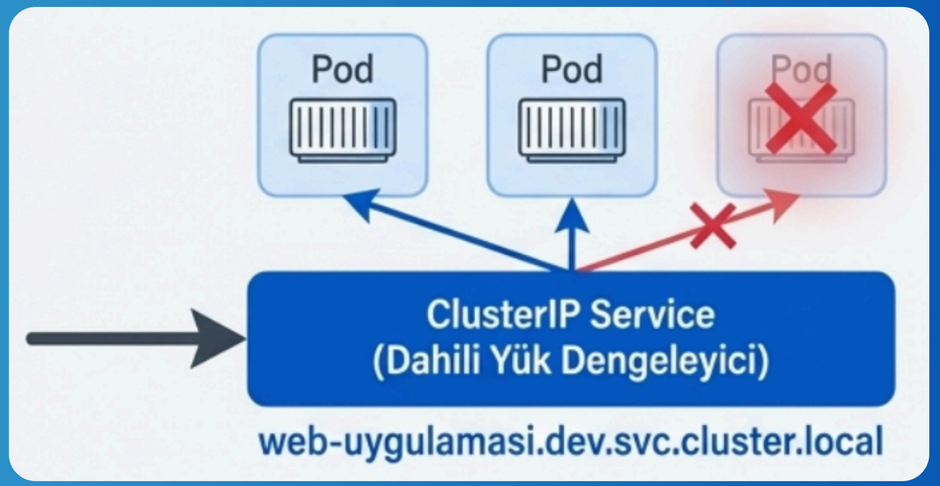
**Manuel Keşif ve Yük Dengeleme**




İletişim, sabit isimlere veya manuel linklere bağlıdır. Yük dengeleme için harici ve manuel yapılandırma gerektirir.

**Kubernetes**

**Otomatik Keşif ve Dahili Yük Dengeleme**



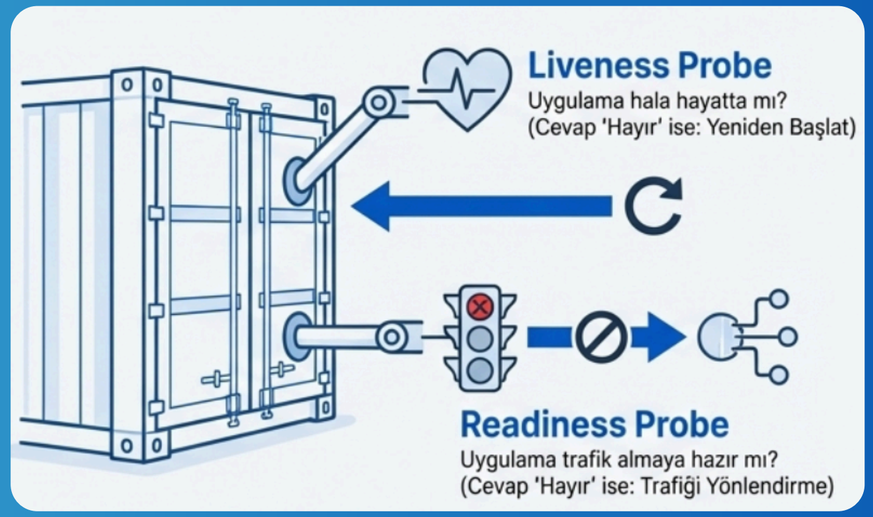

Her servisin kalıcı bir DNS adı ve IP adresi vardır. Pod'lar ölse veya IP'leri değişse bile servis adı asla değişmez. Gelen trafiği sağlıklı Pod'lar arasında otomatik dağıtır, bozuk olanı anında devreden çıkarır.

# Konteyneriniz Gerçekten ‘Sağlıklı’ mı? ‘Zombi’ Konteyner Tehlikesi

## Docker Compose


Docker, prosesin çalıştığını görür ama uygulamanın kilitlenip isteklere cevam veremez hale geldiğini fark etmez.

## Kubernetes

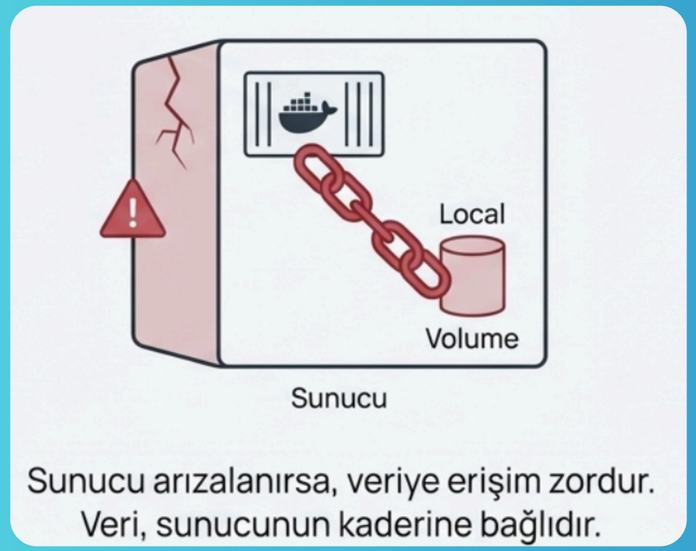



Kubernetes, uygulamanın hem ‘’hayatta’’ olduğunu hem de ‘’trafik almaya hazır’’ olduğunu sürekli olarak denetler.

Sonuç: Kullanıcıların 502/504 gibi hata sayfalarıyla karşılaşma ihtimalini minimize edilir. Gerçek çalışma zamanı (uptime) sağlanır

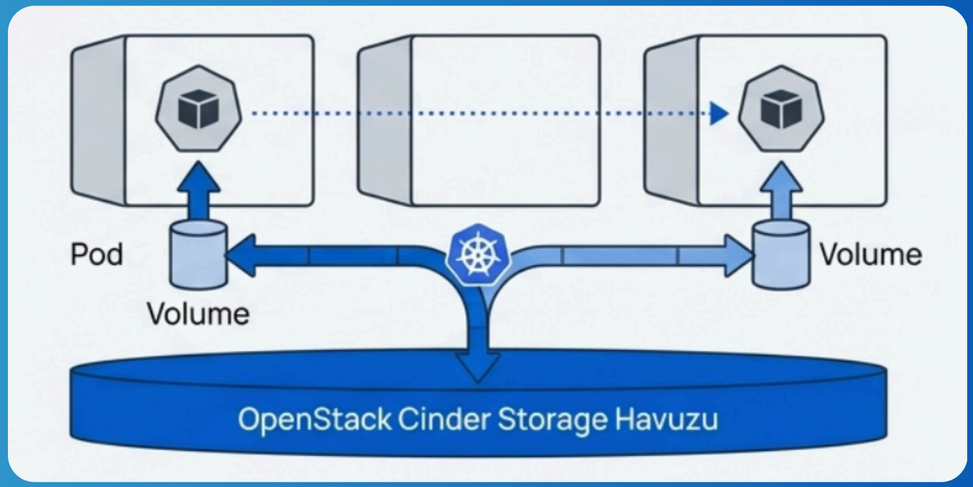
# Veri Nerede Yaşıyor? Sunucuya Bağımlı Depolamadan Özgür Veriye

## Docker Compose




Sunucu arızalanırsa, veriye erişim zordur. Veri sunucunun kaderine bağlıdır.

## Kubernetes




Kubernetes, depolamayı sunucudan ve Pod'dan ayırır. Pod nereye giderse, verisi de oraya gider

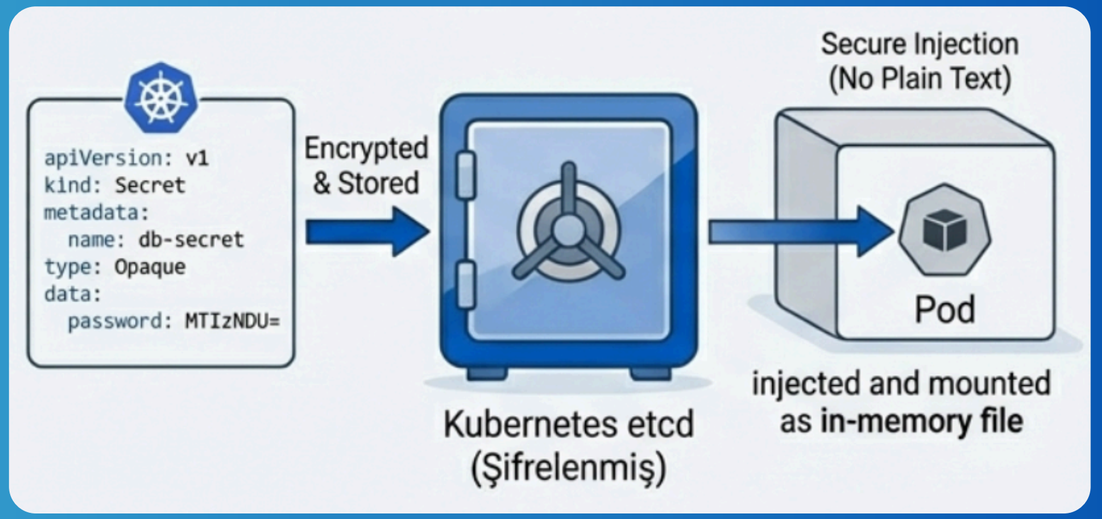
Veritabanu gibi 'Stateful' uygulamaları güvenle çalıştırmanın yolu budur.

# Şifreleriniz Nerede? .env Dosyalarından Şifrelenmiş Kasalara

## Docker Compose


Hassas veriler, kaynak kodunun yanında, genellikle şifrelenmemiş düz metin dosyalarında saklanır. Bu, riskli ve denetimi zordur

## Kubernetes




Hassas veriler merkezi olarak, şifrelenmiş bir şekilde yönetilir ve sadece ihtiyaç duyan uygulamalara çalışma anında, güvenli bir şekilde sunulur.

Bu yaklaşım, ISO 27001 ve KVKK gibi güvenlik standartlarına uyumu doğal olarak sağlar ve denetim süreçlerini kolaylaştırır.

# Pratik Dönüşüm: Tanıdık Fikirler, Daha Güçlü Yapılar

**Docker Compose**

```yaml
# Docker Compose (Mevcut Durum)
version: '3'
services:
  web-app:
    image: company/app:v1
    ports:
      - "80:8080"
    restart: always
```

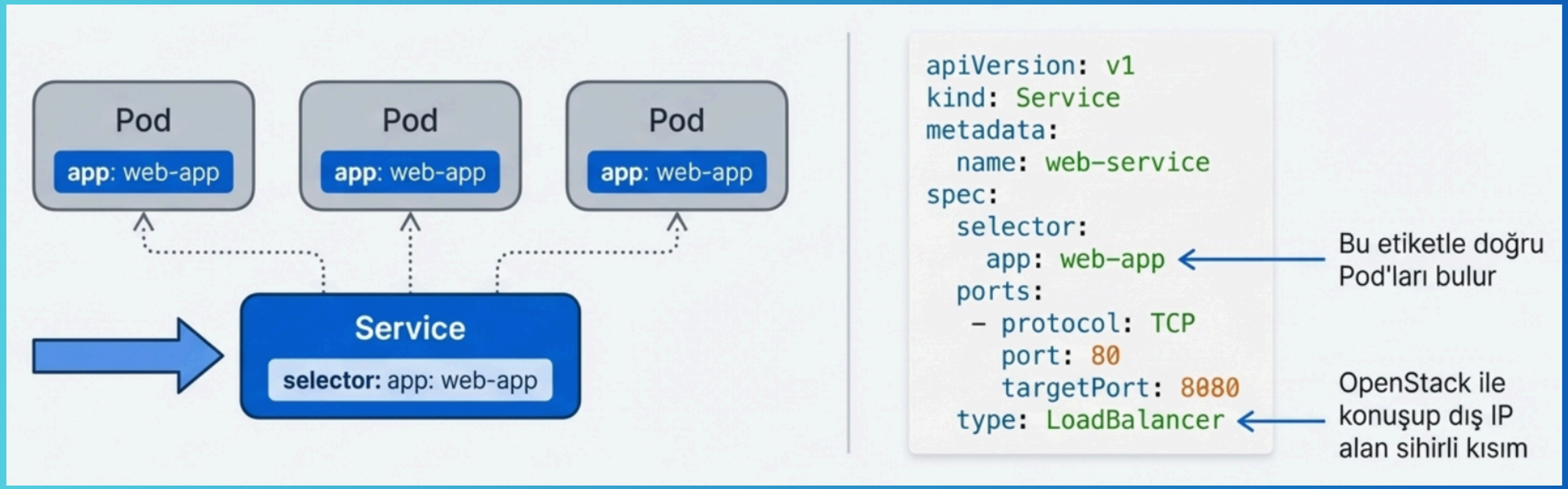
**Kubernetes**

```yaml
# Kubernetes (Gelecek Durum)
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web-app
  template:
    #... (Pod tanımı devam eder)
    spec:
      containers:
      - name: app
        image: company/app:v1
        ports:
        - containerPort: 8080
```

Compose'da sadece 'çalıştır' derken, K8s'te uygulamanın 'istenilen durumunu' tanımlarız. `replicas: 3` diyerek, yüksek erişilebilirliği saniyeler içinde garanti altına alırız.

# Pratik Dönüşüm: Pod’lar geçici, Servisler Kalıcıdır

Docker’da port eşlemesi doğrudan konteynere yapılır. Ancak Kubernetes’te Pod’lar (ve IP’leri) geçicidir. Bu yüzden Pod’ların önünde duran, sabit bir adres sağlayan ve yükü dengeleyen ayrı bir ‘Service’ objesine ihtiyaç duyarız.

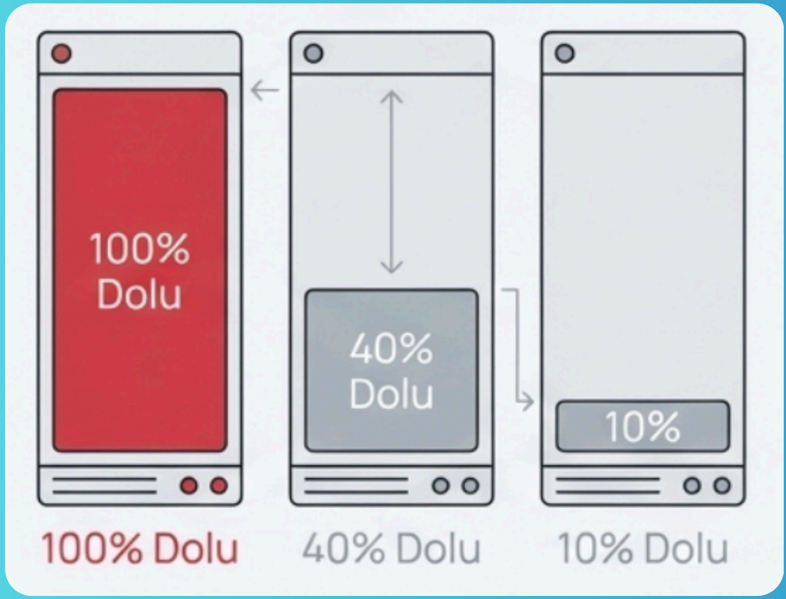



Service objesi sayesinde, arkadaki Pod’un IP’si ne olursa olsun, uygulama her zaman aynı isimle ve aynı IP ile erişilebilir kalır.

# Operasyonel Verimlilik: Sunucuları Tetris Gibi Doldurmak

**Docker Compose**

Manuel ve verimsiz kaynak kullanımı



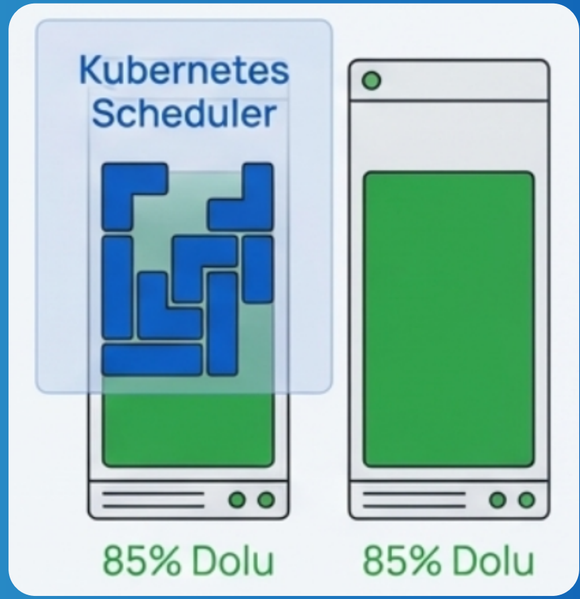

**%30-40**

**Daha Az Donanım İhtiyacı**

(Aynı iş yükü için)

**Kubernetes**

Akıllı ve otomatik Bin Packing



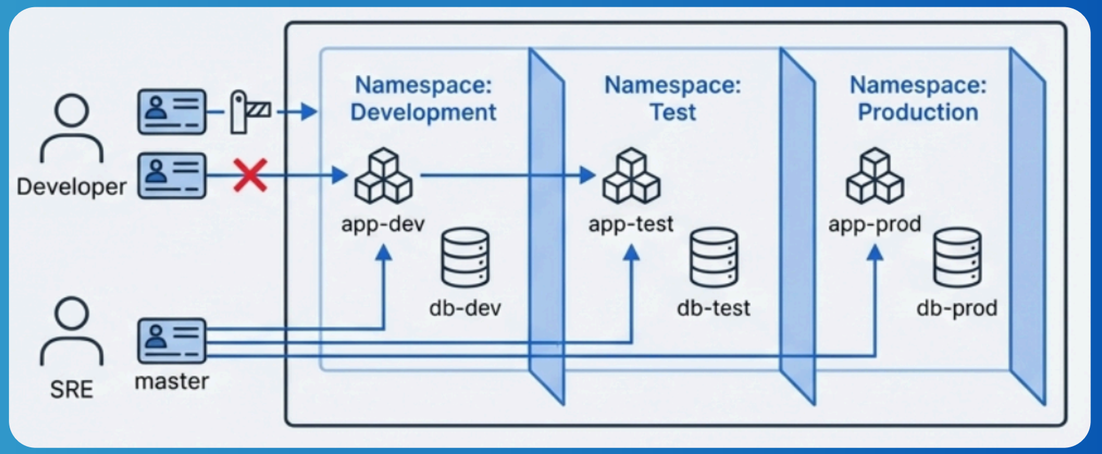

Kubernetes, hangi uygulamanın ne kadar CPU/RAM ihtiyacı olduğunu bilir ve onları mevcut sunuculara en verimli şekilde yerleştirir. Bu, atıl kapasiteyi ortadan kaldırır ve donanım maliyetlerini doğrudan düşürür.

# Herkesin Anahtarı Farklı: Rol Tabanlı Erişim ve Ortam İzolasyonu

**Docker**


Ya tam yetki, ya da hiç

**Kubernetes**




Namespaces ile Geliştirme, Test ve Prod ortamlarını aynı küme (cluster) üzerinde birbirine dokunmadan tamamen izole edebilirsiniz.

Şu gibi kurallar tanımlanabilir: ''A kullanıcısı sadece Test ortamındaki logları görebilsin ama Prod ortamındaki hiçbir şeye dokunamasın''.

# Altyapıya Değil, Standartlara Yatırım Yapın


**Bugün OpenStack üzerinde çalışan bir Kubernetes manifestosu, yarın hiçbir değişiklik olmadan başka bir bulutta veya veri merkezinde çalışır.**

Docker Compose yapılandırmaları bazen işletim sistemi veya altyapı detaylarına bağımlı olabilir. Kubernetes ise standart bir API sunarak bize altyapı özgürlüğü ve geleceğe dönük esneklik kazandırır.

TÜBİTAK

# Değişim Gereklidir: Sonraki Adımımız

| **Docker Compose** (Değerli Başlangıç Noktamız) | **Kubernetes (Profesyonel Geleceğimiz)** |
| --- | --- |
|  |  |
| Tek Sunucu Odaklı | Küme (Cluster) Odaklı |
| Manuel Yönetim | Otomatik Orkestrasyon |
| Hızlı Prototipleme | Üretim Ortamı (Production-Grade) Dayanıklılığı |
| BAŞLATICI | EKOSİSTEM YÖNETİCİSİ |

**Uygulamalarımız ve ihtiyaçlarımız evrildi. Şimdi sıra, operasyonel platformumuzun da onlarla birlikte evrilmesinde.**

T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI

#MİLLİ TEKNOLOJİ HAMLESİ

TÜBİTAK

# TEŞEKKÜRLER

TÜBİTAK | TÜRKİYE BİLİMSEL VE TEKNOLOJİK ARAŞTIRMA KURUMU

bulut@ulakbim.gov.tr