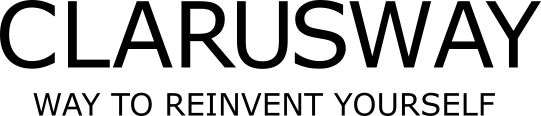




Giriş

 ©



İçindekiler

▶ Monolith ve Mikro Hizmetler Karşılaştırması

▶ Orkestrasyon

▶ Bildirimsel vs Zorunluluk

▶ Kubernetes nedir?

▶ Kubernetes'e neden ihtiyacınız var?

▶ Kubernetes bileşenleri

▶ kubectl

©

2



1

Monolith ve Mikro Hizmetler Karşılaştırması

©



Monolith ve Mikro Hizmetler Karşılaştırması

Taş demektir '.

kelime

'bir Öyle

'monolit' masif

biz

kutu

monolitik olanı büyük bir birleşik blok olarak tanımlar.



Monolith ve Mikro Hizmetler Karşılaştırması

İçinde

yazılım

gelişme

Kullanıcı Arayüzü

**monolitik** geleneksel uygulama

**mimarlık**

isa

olarak

için bir

tek inşa et

ve

bölünmez birim.

Monolitik Uygulama

Hizmetleri



Monolith ve Mikro Hizmetler Karşılaştırması

Diyelim ki bir e-ticaret uygulaması oluşturuyoruz ki

müşterilerden sipariş alır, envanteri ve mevcut krediyi doğrular ve gönderir .

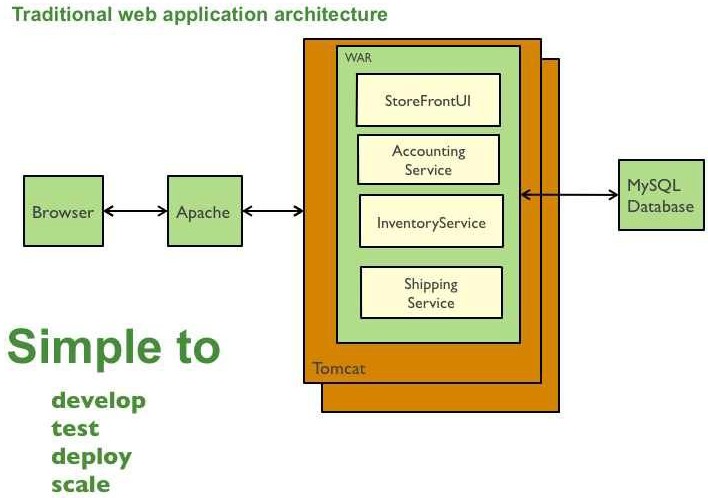
Uygulama, kullanıcı arabirimini uygulayan StoreFrontUI dahil olmak üzere çeşitli bileşenlerin yanı sıra kredi kontrolü, envanter bakımı ve siparişleri göndermek için bazı arka uç hizmetlerinden oluşur .



Monolith ve Mikro Hizmetler Karşılaştırması

Uygulama tek bir monolitik uygulama olarak dağıtılır. Örneğin, bir Java web uygulaması, Tomcat gibi bir web kapsayıcısında çalışan tek bir WAR dosyasından oluşur.

# Monolith ve Mikro Hizmetler Karşılaştırması



## Monolitik mimarinin artıları :

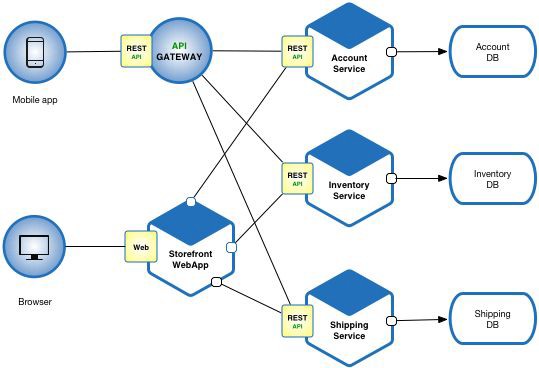
* **Geliştirilmesi daha kolay.** Monolitik yaklaşım, uygulama oluşturmanın standart bir yolu olduğu sürece, herhangi bir mühendislik ekibi monolitik bir uygulama geliştirmek için doğru bilgi ve yeteneklere sahiptir .
* **Dağıtımı daha kolay.** Farklı dosyaların birden çok dağıtımını gerçekleştirmek yerine uygulamanızı yalnızca bir kez dağıtmanız gerekir.
* **Test** etmek  **ve hata ayıklamak** daha kolay. Monolitik bir uygulama bölünmez tek bir birim olduğundan, uçtan uca testleri çok daha hızlı çalıştırabilirsiniz.

# Monolith ve Mikro Hizmetler Karşılaştırması



## Monolitik mimarinin eksileri :

* **Anlayış.**  Monolitik bir uygulama ölçeklendiğinde, anlaşılması çok karmaşık hale gelir.
* **Değişiklik yapma.** Herhangi bir kod değişikliği tüm sistemi etkiler, bu nedenle iyice koordine edilmesi gerekir.
* **Ölçeklenebilirlik.**  Bileşenleri bağımsız olarak ölçeklendiremezsiniz, yalnızca tüm uygulamayı ölçeklendirebilirsiniz.
* **Yeni teknoloji engelleri. Monolitik bir uygulamada yeni** bir teknoloji uygulamak son derece sorunludur, çünkü o zaman tüm uygulamanın yeniden yazılması gerekir.





Monolith ve Mikro Hizmetler Karşılaştırması



Monolith ve Mikro Hizmetler Karşılaştırması

**Mikro hizmetlerin artıları:**

* **Bağımsız bileşenler.**
  + Tüm hizmetler bağımsız olarak dağıtılabilir ve güncellenebilir, bu da daha fazla esneklik sağlar.
  + Bir mikro hizmetteki hatanın yalnızca belirli bir hizmeti etkisi vardır ve tüm uygulamayı etkilemez.
  + Bir mikro hizmet uygulamasına yeni özellikler eklemek, monolitik bir uygulamaya göre çok daha kolaydır.



Monolith ve Mikro Hizmetler Karşılaştırması

**Mikro hizmetlerin artıları:**

* **Daha kolay anlama.** Daha  küçük ve daha basit bileşenlere bölünür, bir mikro hizmet uygulamasının anlaşılması ve yönetilmesi daha kolaydır.
* **Daha iyi ölçeklenebilirlik.** Her öğe bağımsız olarak ölçeklendirilebilir. Bu nedenle, tüm süreç, içinde ihtiyaç olmasa bile tüm uygulamanın ölçeklendirilmesi gerektiğinde monolitlerden daha uygun maliyetli ve zaman etkilidir .

# Monolith ve Mikro Hizmetler Karşılaştırması



## Mikro hizmetlerin artıları:

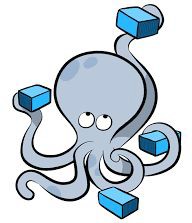
* **Teknoloji seçiminde esneklik.** Mühendislik ekipleri, başlangıçtan itibaren seçilen teknoloji ile sınırlı değildir. Her mikro hizmet için çeşitli teknolojiler ve çerçeveler uygulamakta özgürdürler.
* **Daha yüksek çeviklik seviyesi.** Bir mikro hizmet uygulamasındaki herhangi bir hata, tüm çözümü değil, yalnızca belirli bir hizmeti etkiler. Böylece tüm değişiklikler ve deneyler daha düşük riskler ve daha az hata ile uygulanır.

# Monolith ve Mikro Hizmetler Karşılaştırması



## Mikro hizmetlerin eksileri:

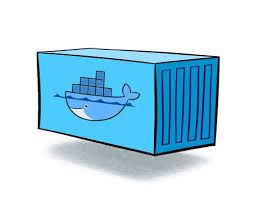
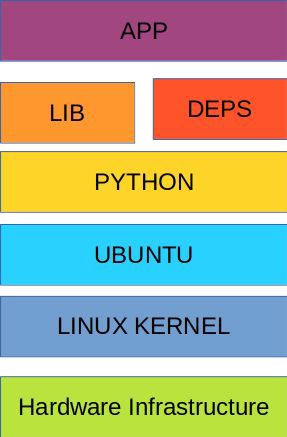
* **Ekstra karmaşıklık.** Mikro hizmet mimarisi dağıtılmış bir sistem olduğundan, tüm modüller ve veritabanları arasındaki bağlantıları seçmeniz ve ayarlamanız gerekir.
* **Sistem dağıtımı.** Mikro hizmetler mimarisi, birden çok modül ve veritabanından oluşan karmaşık bir sistemdir, bu nedenle tüm bağlantıların dikkatli bir şekilde ele alınması gerekir.
* **Test etme.**  Çok sayıda bağımsız olarak dağıtılabilir bileşen, mikro hizmet tabanlı bir çözümün test edilmesini çok daha zor hale getirir.



2

Orkestrasyon

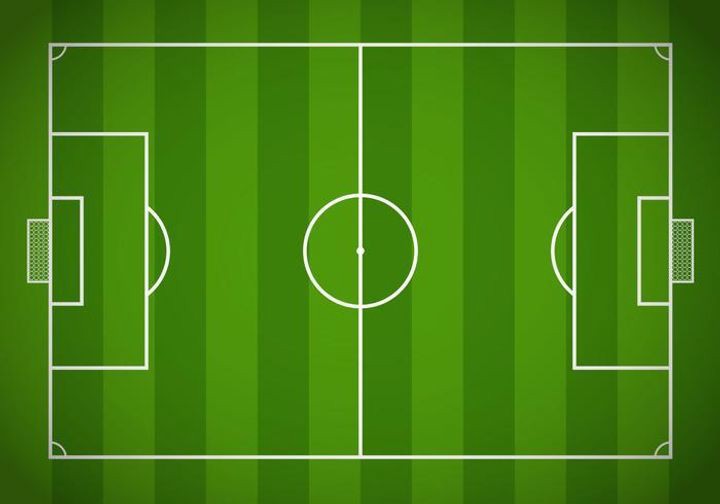
©



Orkestrasyon

©

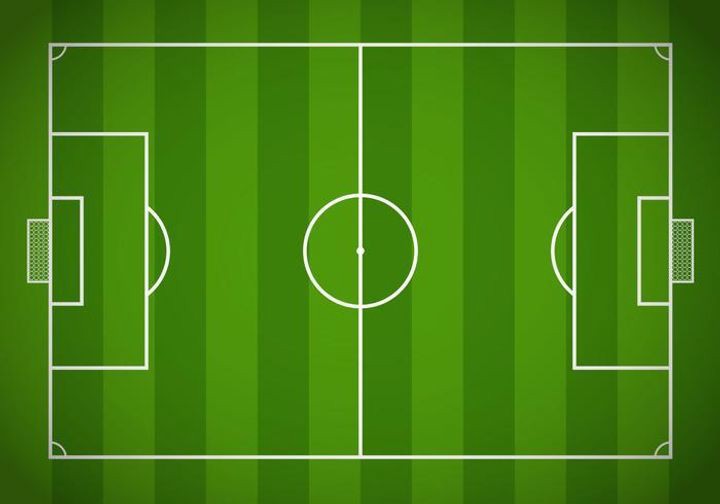
16



Orkestrasyon

©

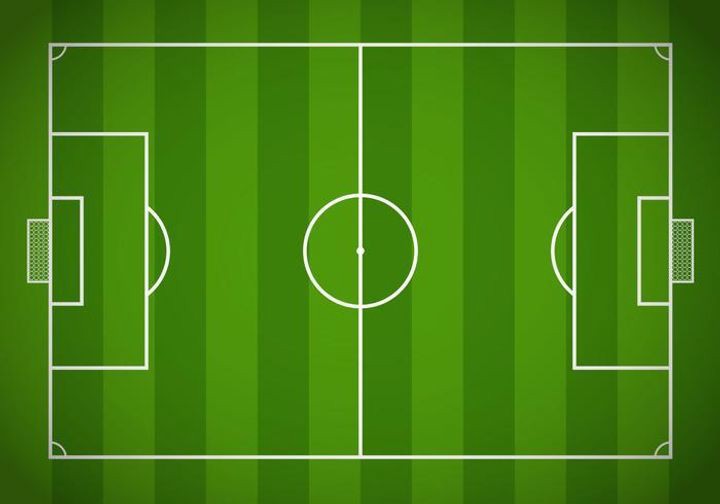
17



Orkestrasyon

©

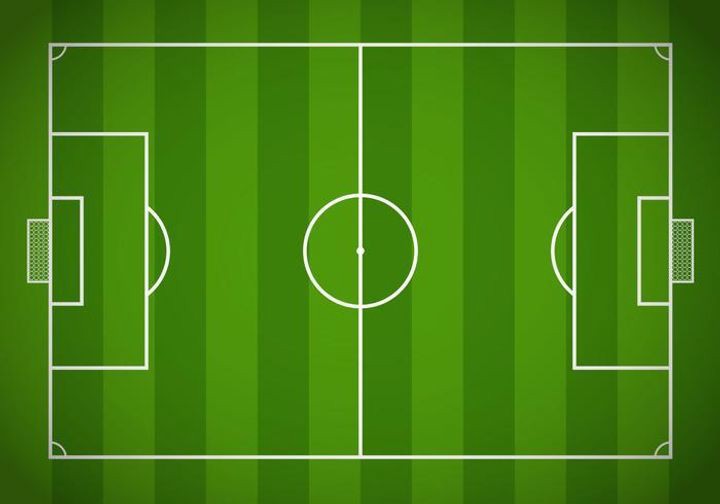
18



Orkestrasyon

©

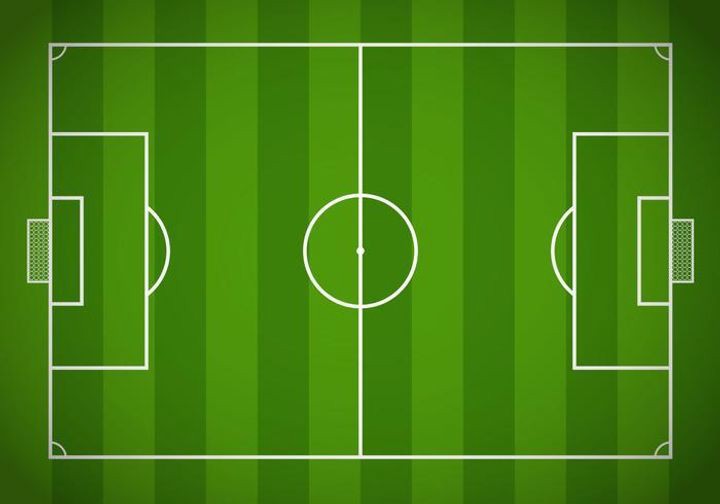
19



Orkestrasyon

©

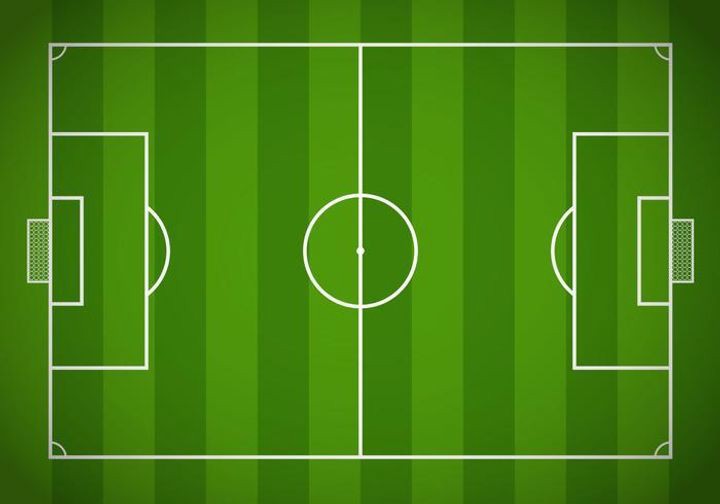
20



Orkestrasyon

©

21



Orkestrasyon

©

22



Orkestrasyon

* Konteynerler harikadır, ancak birçoğunu çalıştırdığınızda, bir noktada, iş sorunlarını çözmek için hepsinin uyum içinde birlikte çalışmasına ihtiyacınız vardır.
* Kapsayıcılı uygulamaları yönetmek, ölçeklendirmek ve sürdürmek için araçlar

orkestratörler olarak adlandırılır ve

commonexample

bunların en büyüğüdür

**Kubernetes'tir**.

©

23



Orkestrasyon

Kapsayıcı orkestrasyonu, aşağıdaki görevleri uygun ölçekte otomatikleştirmek için kullanılır:

* Kapsayıcıların sağlanması ve dağıtımı
* Kapsayıcıların kullanılabilirliği
* Yük dengeleme, trafik yönlendirme ve kapsayıcıların hizmet keşfi
* Kapsayıcıların sistem durumunu izleme

©

24



Orkestrasyon

* Kapsayıcılar arasındaki etkileşimlerin güvenliğini sağlama.
* Kapsayıcıların yapılandırılması ve zamanlanması
* Kapsayıcılar arasında kaynak tahsisi

©

25



3

Bildirimsel vs Zorunluluk

©





Bildirimsel Vs Zorunluluk

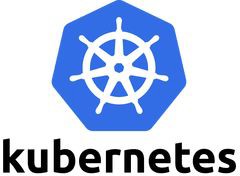
zorunluluk  **nasıl** odaklanır ve bildirimsel **neye** odaklanır.

**Zorunlu yaklaşım:**

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

**Bildirimsel yaklaşım:**

Küçücük ve şirin bir ev istiyorum.



4

Kubernetes nedir?

©



Kubernetes nedir?

CNCF: Bulut Yerel Bilgi İşlem Vakfı

©

29



Kubernetes nedir?

➤ Kubernetes, Kapsayıcılı Uygulamalar için **Açık Kaynak Orkestrasyon** sistemidir.

➤ Kubernetes, ilgili  **Manuel işlemleri ortadan kaldıran** bir platformdur

kapsayıcılı uygulamalar dağıtma.

➤ Kubernetes , **Kapsayıcıların Durumunu** yönetmek için kullanılır.

* Belirli düğümlerde kapsayıcıları başlatın .
* Öldürüldüğünde Konteynerleri yeniden başlatın.
* Kapsayıcıları bir Düğümden Diğerine taşıyın.

©

30



5

Kubernetes'e neden ihtiyacınız var?

©



Kubernetes'e neden ihtiyacınız nerede?

Kapsayıcılar, uygulamaların paketlenmesini ve çalıştırılmasını sağlamak için mükemmel bir yoldur . Bir üretim ortamında,

konteynerler olmalıdır

yönetmek

o

koşmak

uygulamalar ve kesinti süresi olmamasını sağlar.

©

32



Kubernetes'e neden ihtiyacınız nerede?

**Kubernetes size şunları sağlar:**

•

Hizmet keşfi dengelemesi

Depolama orkestrasyonu

ve

yük

•

•

•

•

•

Otomatik dağıtımlar ve geri almalar Otomatik kutu paketleme

Kendi kendini iyileştirme

Gizli

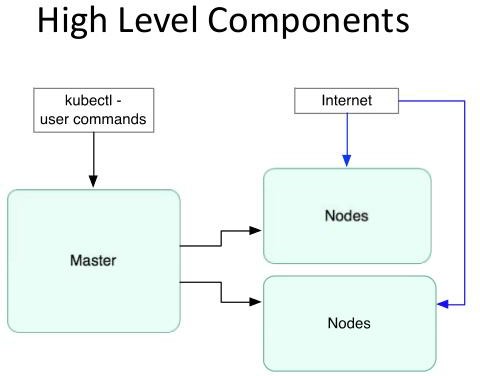
ve

konfigürasyon

yönetim

©

33



6

Kubernetes Bileşenleri

©



Kontrol Düzlemi Bileşenleri

isim-liste

MA-Denetleyici

Öğretmen

Saat

Okul-veritabanıYöneticisi

Öğretmen

MA-Zamanlayıcı

Masa

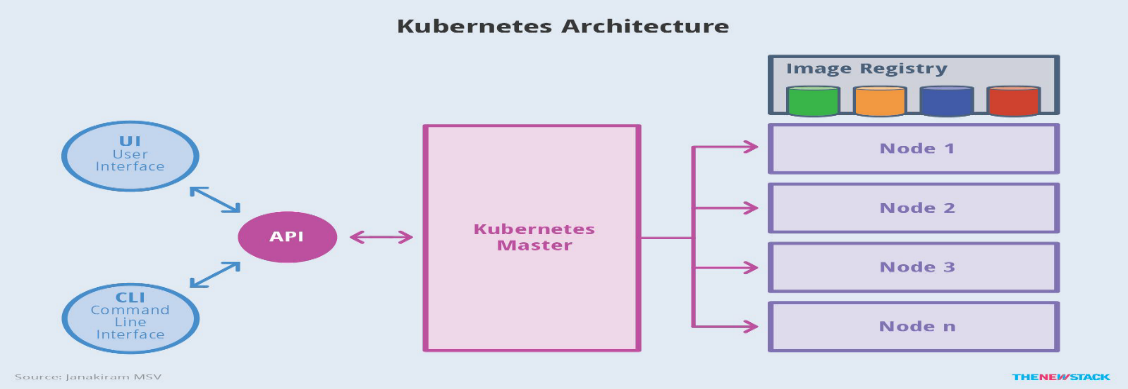
©

Yönetim

isim-liste

Sınıf

36



Kubernetes Bileşenleri

**Kubernetes aşağıdaki ana bileşenlere sahiptir**:Kubernetes  **has the following main components:**

* Bir veya daha fazla ana düğüm
* Bir veya daha fazla çalışan düğümü.

©

35

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |



Kontrol Düzlemi Bileşenleri

kube-proxy

kube-denetleyici

kubelet

konteyner Motor

vbd

kube-api-server

kubelet

kube-scheduler

konteyner motoru

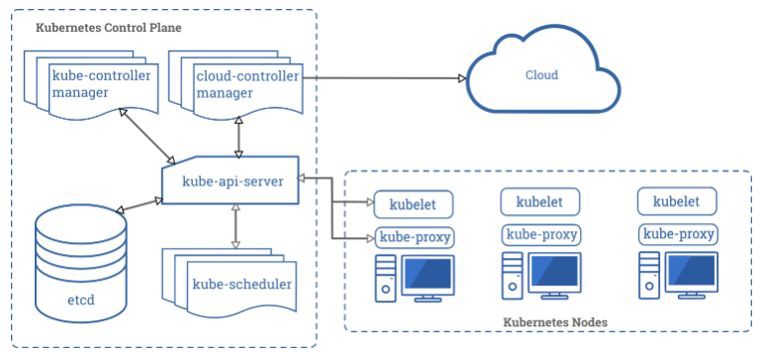
©

kontrol düzlemi

kube-proxy

Düğüm

38



Kontrol Düzlemi Bileşenleri

©

37

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

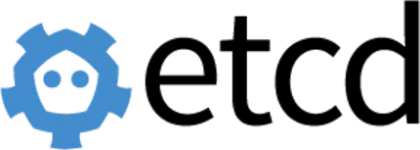
Kontrol Düzlemi Bileşenleri



**kube-apiserver:**

* + Kubernetes denetim düzlemine ve veri deposuna ileriye dönük bir REST arabirimi sağlar.
  + Tüm istemciler ve diğer uygulamalar kubernetes ile kesinlikle API Sunucusu üzerinden etkileşime girer .
  + Yedekleme veri deposunun ön ucu olmanın yanı sıra kimlik doğrulama ve kimlik doğrulama, istek doğrulama, mutasyon ve kabul denetimini işleyerek kümenin ağ geçidi denetleyicisi olarak hareket eder.

© 39 adet



Kontrol Düzlemi Bileşenleri

**etcd:**

* etcd, küme veri deposu görevi görür.
* Kubernetes ile ilgili amaç, kalıcı küme durumu için güçlü, tutarlı ve yüksek oranda kullanılabilir bir anahtar-değer deposu sağlamaktır.
* Nesneleri ve yapılandırma bilgilerini depolar.

©

40



Kontrol Düzlemi Bileşenleri

**kube-controller-manager:**

* Tüm çekirdek bileşenleri yöneten birincil daemon olarak hizmet eder

kontrol döngüleri.

* Apiserver aracılığıyla küme durumunu izler ve kümeyi istenen duruma yönlendirir

©

41



Kontrol Düzlemi Bileşenleri

**kube-scheduler:**

* İş yükü gereksinimlerini değerlendiren ve eşleşen bir kaynağa yerleştirmeye çalışan ayrıntılı ilke açısından zengin altyapı.
* Varsayılan zamanlayıcı çöp kutusu paketlemesini kullanır.
* İş Yükü Gereksinimleri şunları içerebilir: genel donanım gereksinimleri, benzeşim /benzeşim önleme, etiketler ve diğer çeşitli özel kaynak gereksinimleri.

©

42



Bağlantı Noktası Bileşenleri

**kubelet:**

* Ana bilgisayarındaki onu pod'un yaşam döngüsünü yönetmekten sorumlu düğüm aracısı olarak hareket eder.
* Kubelet, cesaret kapsayıcısının çeşitli kaynaklardan okuyabildiğini gösterir:
  + dosya yolu
  + HTTP (Genel) Uç Noktası
  + etcd saat herhangi bir değişiklik üzerinde hareket
  + Basit bir API üzerinden kapsayıcı bildirimlerini kabul eden HTTP Sunucusu modu.

©

43



Bağlantı Noktası Bileşenleri

**kube-proxy:**

* Onu düğümdeki ağ kurallarını yönetir .
* Kubernetes küme hizmetleri için bağlantı iletme veya yük dengeleme gerçekleştirir.
* Mevcut Proxy Modları:
  + Kullanıcı alanı
  + iptables
  + ipvs ( destekleniyorsa varsayılan)

©

44



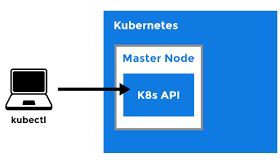
Bağlantı Noktası Bileşenleri

**Konteyner Çalışma Zamanı Altyapısı:**

* Kapsayıcı çalışma zamanı, kapsayıcıları yürüten ve yöneten AĞLAMAK (Kapsayıcı Çalışma Zamanı Arabirimi) uyumlu bir uygulamadır.
  + Kapsayıcılı (docker)
  + Oluşturun
  + Rkt
  + Kelime (eski adıyla berrak ve hiper)
  + Virtlet (VM CRI uyumlu çalışma zamanı)

©

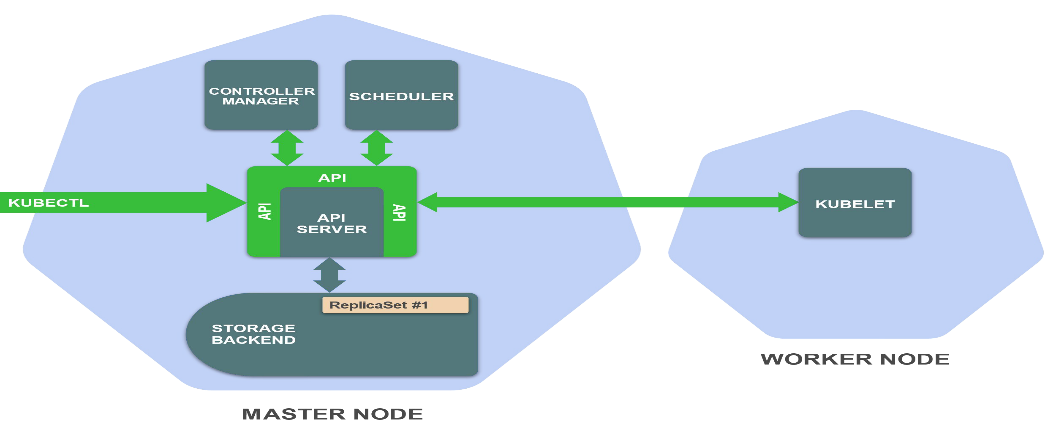
45



7

kubectl

©

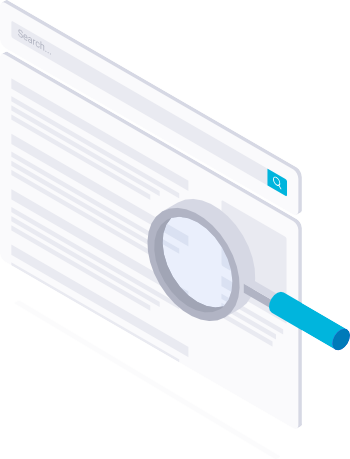


kubectl

* **kubectl**, Kubernetes ile konuşmamız gereken (neredeyse) tek araçtır
* Kubernetes UPA çevresinde zengin bir CLI aracıdır
* Kubectl ile yapabileceğiniz onu şeyi, doğrudan UPA ile yapabilirsiniz
* kubectl "Küp C T L", "Küp mürekkep balığı", "Küp sarılmak" olarak telaffuz edilebilir.

©

47



**TEŞEKKÜRLER!**

**Sorularınız mı var?**

©

4

8