## Kubernetes Nedir?

Container tabanlı uygulamaları ölçeklendirebildiğimiz, yönetebildiğimiz ve dağıttığımız bir teknolojidir. Açık kaynaklıdır.

Kubernetes, dağıtık sistemleri esnek ve verimli bir şekilde çalıştırmamıza olanak sağlayan bir altyapıdır.

Geliştiricilere ve işletmecilere uygulama dağıtım süreçlerini kolaylaştırma ve hızlandırma imkanı sağlar.

Google tarafından oluşturuldu. Şuan açık kaynak bir proje olarak CNCF altında faaliyet gösteriyor.

### Container

Uygulamaların ve bu uygulamanın çalışması için gerekli bağımlılıkların tek bir birimde paketlendiği teknolojidir.

Container kullanımının avantajları:

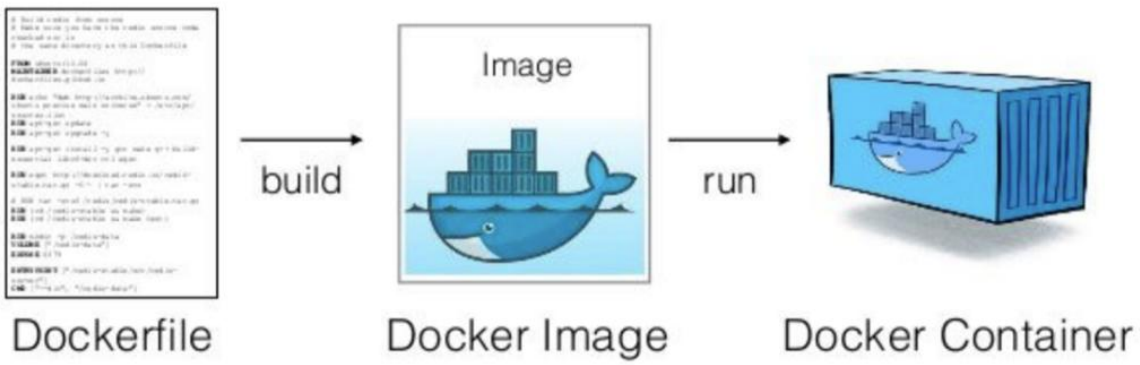
- İzolasyon
- Taşınabilirlik
- Hızlı Dağıtım
- Ölçeklenebilirlik

Container kullanımının dezavantajları:

- Güvenlik sorunları
- Karmaşık ortamların yönetim zorluğu

### Docker

Docker, Uygulamaların ve hizmetlerin izole, taşınabilir ve tutarlı bir şekilde çalıştırılmasını sağlayan açık kaynaklı bir konteynerleştirme platformudur.




### Container Orchestration

Çok sayıda container olduğunda bu kaynakları yönetmek için kullanılan bir süreçtir.

Container'ların otomatik olarak yönetilmesini, ölçeklenmesini, yönetilmesini ve kordinasyonunu içerir.

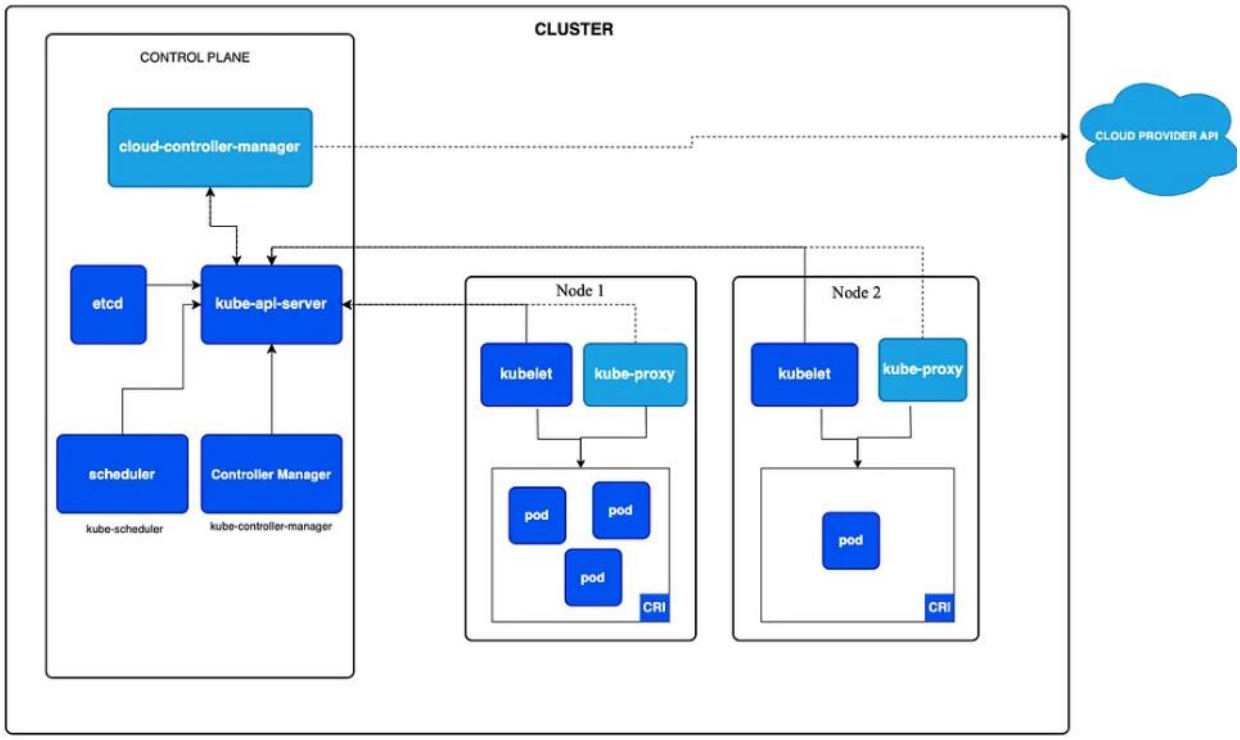
Conteiner Orchestration Temel Özellikleri

1- Planlama ve Yerleştirme
2- Dağıtım ve Ölçeklenebilirlik
3- Yük Dengeleme
4- Sağlık Kontrolleri
5- Otomatik Yeniden Başlatma
6- Güncellemeler ve Rollback

Container Orchestration Avantajları:

- Otomasyon
  Tekrarlayan görevleri otomatik hale getirir ve operasyonel yükü azaltır.
- Ölçeklenebilirlik
  Trafik değişikliklerine hızlı yanıt vererek container sayısını ayarlar. CPU ve RAM artışınıda yönetir.
- Dayanıklılık
  Uygulamaların kesintisiz çalışmasını sağlar.
- Verimlilik
  Kaynakların daha etkin kullanımını sağlar. Bu sayede maliyetleri düşürür.

Mimari Yapısı




**Node :** Kubernetes'te çalışan fiziksel ya da sanal bir makinadır. Kubernetes'in temel yapı taşıdır. Uygulamalarımız node üzerinde çalışır.

**Cluster:** Bir grup node'den oluşur. Uygulamaların çalıştığı, ölçeklendirildiği ve yönetildiği yerdir.

**Master Node:** Cluster'daki diğer node'ların yönetimi ve kontrolünden sorumludur.

**Worker Node:** Master node dışındaki node'ler worker node olarak adlandırılır. Uygulamaların çalışmasından sorumludur.

**Pod:** Container'ı içerisine alan kubernetesin en küçük birimidir.

## Master Node

### Etcd

Açık kaynaklıdır.  Özellikle yüksek erişilebilirlik ve veri **tutarlılığı sağlamak** için geliştiirilen bir anahtar-değer veri deposudur.

Kubernetes'in temel bileşenlerinden birisidir.

Küme durumu, yapılandırma verileri ve diğer meta verileri depolar.

Kubernetes'in çalışmasında merkezi bir rol oynar. Tüm küme durumu, yapılandırma verileri, kaynak bilgileri etcd'de saklanır.

### Api Server

Temel görevleri:

- API isteklerini işleme
- Yetkilendirme ve kimlik doğrulama
- Küme bileşenleri ve kullanıcılar arasındaki iletişimi yönetme

### Scheduler

Kubernetes'daki podların nodelere yerleştirilmesinden sorumludur.

Ana işlevi yeni oluşturulan podların gerekli kaynaklara sahip, kullanıcı tarafından tanımlanan yerleştirme kriterlerini karşılayan nodelere yerleştirmektir.

### Controller Manager

Küme durumunu istenen duruma getirmek için sürekli çalışan controller'lardan oluşur.

Bir controller sistem içerisindeki çeşitli bileşenlerin durumunu sürekli olarak izleyen ve tüm sistemi istenen işleyiş durumuna getirmek için çalışan bir süreçtir.

## Node Bileşenleri

### Kubelet

Her node'de çalışan bir bileşendir. Pod yönetimi yapar.

Kubelet, master node'den aldığı komutlar ile pod'ların yaşam döngüsünü yönetir.

- Container orkestrasyonu
- Pod yönetimi
- Node durumu izleme

Güvenilirlik sağlar. Sorunları otomatik giderir. Kaynaklar verimli kullanır. Yük dengelemesi yapar.

**Kube Proxy**

Ağ yönetim araçlarından biridir.

Podlar arasındaki ağ yönetimini sağlar.  Load balancing (yük dengeleme) yapar.

Ağ kurallarını uygular. Ağ kurallarını oluşturur ve yönetir. Bu kurallar servislarin hangi podlara yönlendirileceğini belirier.

- **Yerel Kubernetes** çözümlerine örnek: Minikube, Kind, k3s
- **Bulut tabanlı Kubernetes** servislerine örnek: Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS)

| Kriter | Yerel Kubernetes (Local) | Bulut Tabanlı Kubernetes (Cloud-based) |
|---|---|---|
| **Kurulum ve Yapılandırma** | Kullanıcı tarafından manuel yapılır | Sağlayıcı tarafından otomatik yapılır (örn. GKE, EKS) |
| **Maliyet** | Donanım maliyeti dışında genelde ücretsizdir | Kullanıma ve kaynaklara bağlı olarak ücretlidir |
| **Erişilebilirlik** | Sadece kurulu olduğu cihazdan erişilebilir | İnternet bağlantısıyla her yerden erişilebilir |
| **Ölçeklenebilirlik** | Sınırlı (cihaz kaynaklarına bağlı) | Yüksek; otomatik ölçeklendirme mümkündür |
| **Kullanım Amacı** | Geliştirme, test ve eğitim için idealdir | Üretim ortamları ve büyük ölçekli dağıtımlar için uygundur |
| **Performans** | Cihazın donanım kapasitesine bağlı | Yüksek performanslı altyapılar üzerinde çalışır |
| **Bakım ve Güncellemeler** | Kullanıcı sorumluluğundadır | Sağlayıcı tarafından yönetilir |
| **Entegrasyonlar** | Kısıtlı olabilir, manuel entegrasyon gerekir | Diğer bulut servisleriyle kolay entegrasyon |
| **Yedeklilik ve Hata Toleransı** | Genellikle yoktur veya sınırlıdır | Yüksek erişilebilirlik ve yedekleme özellikleri sunar |
| **Güvenlik** | Kullanıcının yapılandırmasına bağlı | Sağlayıcının güvenlik altyapısıyla desteklenir |

Pods

Kuberneteste dağıtım yapılan en küçük birimdir. Bir pod genellikle tek container bulundurur. Birden fazla varsa ortak bir ip adresi, ağ ve depolama kaynaklarını paylaşırlar. Fakat her containerın kendi dosya sistemi vardır. Aynı yaşam döngüsünü paylaşırlar.

Podların belirli durumları vardır.

- Pending
  Pod oluşturumuş henüz çalıştırılmamış
- Running
  Pod başarılı çalışmış
- Succeeded
  Pod tamamlandı ve içerisindeki tüm containerlar sonlandı
- Failed
  Pod tamamlanamadan hata ile sonlandı

Kubernetes'te pod yönetimi çok önemlidir.

### ReplicaSets

Kopya oluşturaan bir araçtır. Yani belirli bir sayıda, aynı tür podun sürekli çalışır durumda olmasını sağladığımız bir şeydir.

ReplicaSet, bize belirlediğimiz sayıyı garanti etmeye çalışır. Bu şekilde erişilebilirlik sağlanmış olur.

### Deployment

Uygulamaların yönetilmesi, ölçeklendirilmesi ve güncellenmesi için kullanılan bir kubernetes nesnesidir.

Belirli bir istek ve yapılandırmaya göre podların oluşturulmasını ve yönetilmesini sağlar.

- Rolling Update
  Eski podlar kademeli olarak yenisi ile değiştiirilir. Güncelleme sırasında kesinti olmaz. Kontrollü bir geçiş olur. Yeni podda sıkıntı olursa eskisine dönülebilir.
- Recreate
  Mevcutta bulunan podların tamamını kapatır. Kısa bir kesinti oluşur.
- Canary
  Yeni sürüm önce belirli bir kullanıcıya verilir. Sorun olmas ise yeni sürüm tüm kullanıcılara verilir.
- Blue/Green
  Mevcut sürüm ve yeni sürüm aynı anda çalıştırılır. Geçiş anında trafik eski sürümden yeni sürüme aktarılır. Yeni sürümde sorun olursa tarafik eski sürüme aktarılarak eski sürüme geri dönülür.

Deployment, ReplicaSet'ten daha fazla kontrol sunar.

### Deamonset

Eğer her node üzerinde bir pod çalıştırmak istiyorsak deamonset kullanırız.

### StatefulSet

Durumu olan yani state tutan uygulamaları yönetmekk için kullanılan bir yönetim nesnesidir.

Amacı her bir podun kimliğini koruyarak, sabit ve kararlı bir şekilde dağıtım ve ölçeklendirme sunmasıdır.

**Kalıcı depolama** özelliği sayesinde pod yeniden başlasa bile verilerin korunmasını sağlar. Kaybın önüne geçilmiş oluyor.

### Namespaces

Kubernetes ortamında birden fazla sanal küme oluşturmamızı sağlayan bir özelliktir.

Bir cluster içeriisindeki, aynı node içerisinde farklı kullanıcıların, projelerin izole edilmiş kaynaklara sahip olmasını sağlar. Bu bize izolasyon sağlar.

### Kubernetes Ağ Yapısı

Kubernetes ağ yapısı basitlik ve esneklik üzerine kurulur.

Kubernetes ile her pod'a bir ip adresi atanır. Bu ip adresleri üzerinden pod'lar birbiri ile haberleşebilir. Hehangi bir pod aynı ya da farklı none üzerinde olursa olsun birbiri ile haberleşebilir. Bu iletişim içinde flat networking denilen bir yöntem kullanılır.

Kubernetes podlar arasında ip bazlı iletişimi garanti eder.

Kubernetes'te network işlemleri, Service denilen bir nesne ile yönetilir. Kubernetes Service, bir poda bir ağ hizmeti sunan soyut bir yapıdır.

Service Türleri

- Cluster-IP
  Cluster dışı erişim olmaz. Cluster içerisinde podlara erişim sağlamak için kullanılan varsayılan bir service türüdür.
- Node-Port
  Cluster dışından bir poda erişim gerekiyorsa kullanılır. Bu node port üzerinden clusterı dışarı açarız. Her node'da bir port numarası açılır ve bu port üzerinden gelen trafik ilgili service yönlendirilir.
- Load-Balncer
  Bulut ortamında kubernetes kullanıyorsak bu service'yi sık kullanırız. Dışardan gelen istekleri kolayca karşılayabiliriz.
- Ingress
  Cluster dışından gelen http ya da https isteklerini kubernetes servicelere yönlendirir ve bu istekleri yönetir. Bir ya da birden fala url'i bir yada birden fazla service'ye aynı ip üzerinden erişim sağlatabilir. Bu sayade daha az public ip kullanımı sağlanır.

Service, load balancing işlemi yapar. Service'ye gelen yük podlar arasında eşit olarak dağıtılır.

Service bize DNS ve Service Dicovery hizmeti sunar. Kubernetes her service için bir dns ismi oluşturur. Service discovery ile uygulamanın hangi ip adresi ya da port üzerinden diğer servise erişebileceğini bulur. DNS tabanlı bir yaklaşım kullanır.

Varsayılan olarak cluster içerisindeki tüm podlar birbiri ile iletişim kurabilir. Herhangi bir ağ izolasyonu yoktur. Bu durum güvenlik sorunu ortaya çıkarabilir. Bu durumlarda Network Policy ortaya çıkmıştır. Network Policy, Kuernetes üzerindeki podlar arasındaki ağ tarafiğini yönetir.

### Kubernetes Storage

Depolama, containerların yaşam döngüsünden bağımsız olarak gerçekleşir. Verilerin kalıcılığı sağlanır.

**Volume,** pod içerisinde çalışan container tarafından paylaşılan depolama birimidir. Kubernetes'te birçok volume çeşidi vardır.

Access modes:

1- ReadWriteOnce (RWO): Sadece bir node üzerinde yazılıp okunabilir.
2- ReadOnlyMany (ROX): Birkaç node üzerinde sadece okunabilir.
3- ReadWriteMany RWX): Birkaç node üzerinde yazılıp okunabilir.

### ConfigMaps

Yapılandırma verilerimizi, kubernetes objelerine, podlara dahil etmek için kullanılır.

ConfigMaps verilerini environment olarak kullanabiliriz. Podlara environment olarak verilebilir. Volume olarak maunt edilebilir ya da komut satırında argüman olarak başlatılabilir.

### Secret

Hassas verileri, parolaları, ssh keyleri güvenli bir şekilde kullanabilmek, saklayabilmek için kullandığımız nesnedir.

Secretlar şifreli olarak saklanıyor ve yetkili podlar tarafından erişilebiliyor.

Podlara environment olarak verilebilir. Volume olarak maunt edilebilir

## Kubernetes Scheduling

### Labels & Selectors

Label'lar sayesinde nesneler gruplandırılabilir.

### Taints & Tolerations

Taints'ler bir nodu bazı podlar için uygunsuz, bozuk hale getirir.

### Node selector & affinity

Node selector, pod'ların sadece belirli node'lerde çalışmasını sağlar. Node affinity de node selectorün daha gelişmiş halidir ve belirli kurallara göre podların nodelere yerleştirilmesini sağlar.

### Resource Requirements

Podların ne kadar CPU, ne kadar RAM talep edeceğini ve maksimum ne kadar kullanabileceğini belirttiğimiz bir özelliktir.

### InitContainers

Podun ana containerı başlamadan önce çalışan yardımcı bir containerdır. Bu container asıl containerın çalışması için gerekli hazırlıkları yapar.

## RBAC

Kubernetes kaynaklarına,kullanıcılar ve diğer kubernetes nesnelerini kontrol eden bir sistemdir.

Yetkilendirme poliitikalarını yönetmek için roller ve kurallar kullanır.

Kubernetes kumesinin güvenliğini artırmak, kaynaklara erişimi sınırlandırma için kullanıcıların, grupların, uygulamaların yetkilendirildikleri kaynaklara erişim sağlaması için kullanılması gereken bir özelliktir.