



T.C.
CUMHURİYET ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BULUT TABANLI SİSTEMLERİN İNCELENMESİ

Volkan ÇİZMECİOĞLU
LİSANS BİTİRME PROJESİ

Temmuz-2020
SİVAS

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.



Volkan ÇİZMECİOĞLU

Tarih: 29.07.2020

ÖZET

Bulut Tabanlı Sistemlerin İncelenmesi

Volkan ÇİZMECİOĞLU

Danışman: Dr. Öğr. Üyesi Halil ARSLAN

Bu lisans bitirme tezinde öncelikle bulut tabanlı sistemler genel olarak incelenmiş olup, avantajları ve dezavantajları karşılaştırılmıştır. Sanallaştırmanın ne olduğu anlatılmıştır. Ardından konteynerin ne olduğu, konteyner orkestrasyonunun ne olduğu ve nasıl çalıştığı incelenmiştir. Konteyner orkestrasyon araçlarından Docker ve Kubernetes ile ilgili bilgiler verilip, Docker ve Kubernetes kullanarak bulut tabanlı sistem örneği gösterilmiştir.

Anahtar Kelimeler: Kubernetes, Docker, Konteyner, Orkestrasyon

İÇİNDEKİLER

ÖZET	iv
İÇİNDEKİLER	v
KISALTMALAR	viii
1. BULUT TABANLI SİSTEMLER	1
1.1. Bulut Tabanlı Sistemlere Genel Bakış	1
1.2. Avantajlarının İncelenmesi	1
1.3. Dezavantajlarının İncelenmesi	2
1.4. Bulut Mimari Dağıtım Türleri	2
1.4.1. Genel bulut	2
1.4.2. Özel bulut	2
1.4.3. Hibrit bulut	2
1.5. Bulut Hizmet Türleri	3
1.5.1. Hizmet olarak altyapı(IaaS)	3
1.5.2. Hizmet olarak platform(PaaS)	3
1.5.3. Sunucusuz bilgi işlem	3
1.5.4. Hizmet olarak yazılım(SaaS)	3
2. SANALLAŞTIRMA	4
2.1. Sanallaştırma Nedir?	4
2.2. Avantajlarının İncelenmesi	5
3. KONTEYNER.....	6
3.1. Konteyner Nedir?	6
3.2. Konteynerlerin Avantajları?	6
3.3. Konteynerlerin Dezavantajları?	7
4. KONTEYNER ORKESTRASYONU	8
4.1. Orkestrasyon Nedir?	8
4.1. Nasıl Çalışır?	8

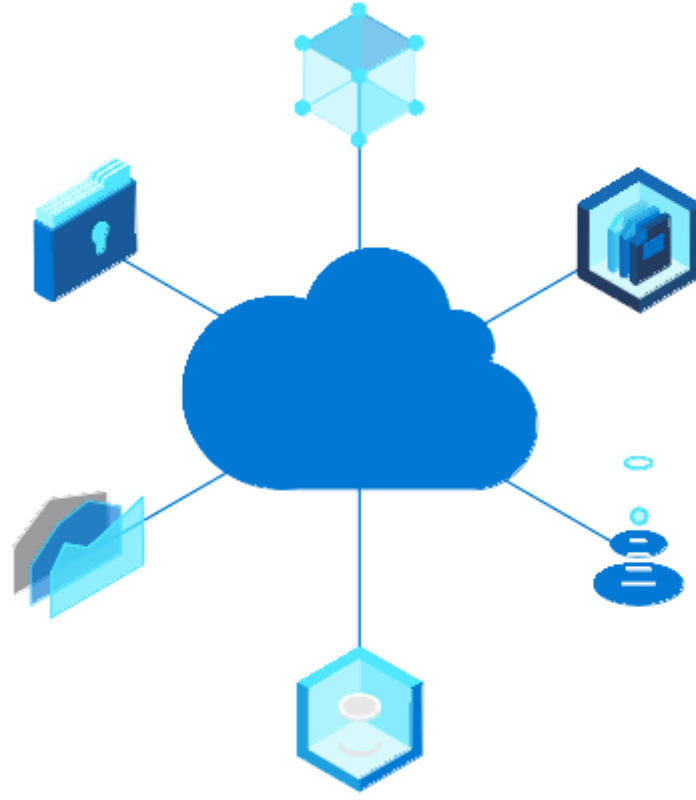
5. DOCKER.....	9
5.1. Docker Nedir?.....	9
5.2. Docker'ın Farkı?	9
5.3. Dockerfile Nedir?	10
5.4. Docker Container Nedir? Docker Image Nedir?	10
5.5. Docker Hub Nedir?.....	10
6. KUBERNETES	12
6.1. Kubernetes Nedir?	12
6.2. Kubernetes Yapısının İncelenmesi	14
6.3. Kubernetes Node'larının İncelenmesi	16
6.4. Bazı Terimler	18
6.5. Kubernetes Çalışma Şekli	19
7. KUBERNETES İLE BULUT TABANLI SİSTEM ÖRNEĞİ.....	21
7.1. Sistem Gereksinimleri.....	21
7.2. Örneğin Gerçekleştirilmesi	21
8. SONUÇLAR.....	66
8.1. Sonuçlar	66
9. KAYNAKLAR	67

KISALTMALAR

Kısaltmalar

- BT (Bilişim Teknolojileri)
- VM (Virtual Machine)

1. BULUT TABANLI SİSTEMLER



Şekil 1.1. Bulut tabanlı sistem görseli

1.1. Bulut Tabanlı Sistemlere Genel Bakış

Bulut tabanlı sistemlerde, bilgi işlem hizmetleri (suncu, depolama, veritabanı, ağ, yazılım, analiz, makine zekası) “bulut” (internet) üzerinden gerçekleştirilir.

1.2. Avantajlarının İncelenmesi

- Donanım ve yazılım satın alma, şirket içi veri işleme merkezi kurma ve çalıştırma maliyetlerini ortadan kaldırarak yatırım maliyetlerini azaltır. Böylelikle maliyet açısından avantaj sağlar.
- Bilgi işlem kaynaklarına ulaşımı kolaylaştırır ve hız açısından avantaj sağlar.
- BT kaynaklarını (işlem gücü, depolama, bant genişliği) istenilen zamanda istenilen coğrafi konumda sunulabilmesini sağlayarak küresel ölçekte avantaj sağlar.

- Veri işleme merkezinde yapılması gereken donanım ayarlama, yazılımlara düzeltmeler uygulama gibi işlemleri azaltarak verimliliği artırır.
- Verilerin yedeklenmesi ve bu yedeklenmiş verilere ulaşımın kolay olması sayesinde güvenilirlik açısından avantaj sağlar.
- Bulut sağlayıcıların sunduğu güvenlik hizmetleri sayesinde verilerin daha güvenli korunması sağlanır.

1.3. Dezavantajlarının İncelenmesi

- Sistemlerde yaşanabilecek sorunlar ve saldırılar yüzünden verilere erişimde sıkıntı yaşanabilir. Değerli veriler hırsızlığa uğrayabilir.
- Sistem bakım sürelerinde, sisteme erişememe durumları yaşanabilir.

1.4. Bulut Mimari Dağıtım Türleri

1.4.1. Genel bulut

Üçüncü taraf bulut hizmetleri sağlayıcıları tarafından sağlanır ve işletilir. Bu hizmet sağlayıcılar sunucu ve depolama gibi BT kaynaklarını internet üzerinden sunar. Genel bulut yapısında tüm donanım, yazılım ve diğer destekleyici altyapı bulut sağlayıcısına aittir.

1.4.2. Özel bulut

Özel bulut, bulut bilgi işlem kaynaklarının tek bir kurum veya kuruluş tarafından kullanılmasıdır. Şirketin kendi veri merkezinde fiziksel olarak bulunabilir. Özel bulutta hizmetler ve altyapı özel bir ağ üzerinden sağlanır.

1.4.3. Hibrit bulut

Genel ve özel bulutu, ikisi arasında veri ve uygulama paylaşımına olanak tanıyan bir teknoloji aracılığıyla birbirine bağlayarak birleştirir.

1.5. Bulut Hizmet Türleri

1.5.1. Hizmet olarak altyapı (IaaS)

Bulut bilgi işlem hizmetlerinin en temel kategorisidir. IaaS, bir bulut sağlayıcısından kullandıkça öde esasına dayalı olarak BT altyapısı (sunucular ve sanal makineler (VM), depolama, ağ, işletim sistemleri) kiralanmasına olanak tanır.

1.5.2. Hizmet olarak platform (PaaS)

Hizmet olarak platform, yazılım uygulamaları geliştirmek, test etmek, teslim etmek ve yönetmek üzere isteğe bağlı bir ortam sağlayan bulut bilgi işlem hizmetleri olarak tanımlanır. PaaS, geliştiricilerin web uygulamalarını veya mobil uygulamaları, geliştirme için gereken sunucular, depolama alanı, ağ ve veritabanlarından oluşan temel altyapıyı kurma ve yönetme endişesi taşımadan hızla oluşturmasını kolaylaştırmak üzere tasarlanmıştır.

1.5.3. Sunucusuz bilgi işlem

PaaS ile örtüşen sunucusuz bilgi işlem, sürekli olarak sunucuları yönetmeye zaman harcamadan, uygulama işlevleri geliştirmeye ve bunu yapmak için gerekli olan altyapıyı oluşturmaya odaklanır. Bulut sağlayıcısı sizin için kurulumun, kapasite planlamasının ve sunucu yönetiminin üstesinden gelir. Sunucusuz mimariler olay temellidir, yüksek düzeyde ölçeklenebilir ve yalnızca belirli bir işlev veya tetikleyici ortaya çıktığında kaynak kullanır.

1.5.4. Hizmet olarak yazılım (SaaS)

Hizmet olarak yazılım, yazılım uygulamalarını internet üzerinden isteğe bağlı olarak ve genellikle bir abonelik aracılığıyla dağıtma yöntemidir. SaaS sayesinde bulut sağlayıcıları, yazılım uygulamalarını ve temel altyapıyı barındırıp yönetmenin yanı sıra yazılım yükseltmeleri ve güvenlik düzeltme eki uygulama gibi bakım işlerini de üstlenir. Kullanıcılar uygulamalara genelde telefon, tablet veya bilgisayarlarında bulunan bir web tarayıcısı ile internet üzerinden bağlanır.

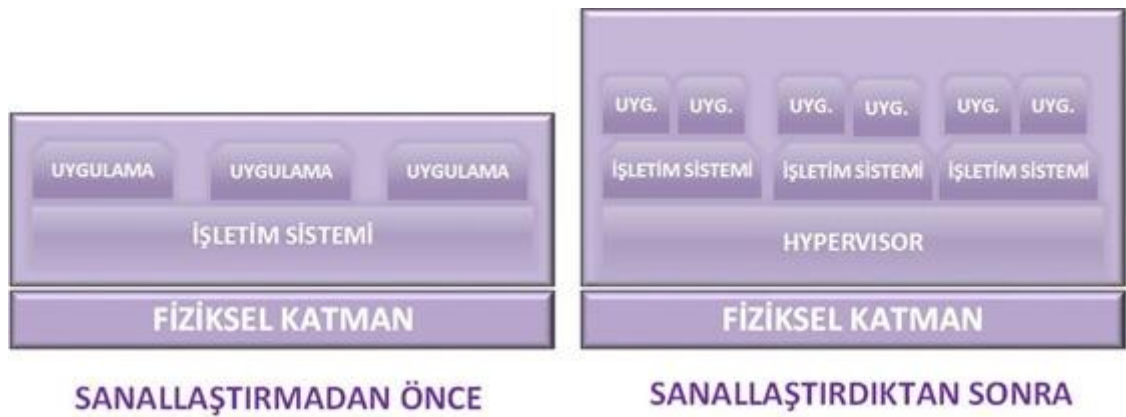
2. SANALLAŞTIRMA

2.1. Sanallaştırma Nedir?

Sanallaştırmada, fiziksel bilgi işlem ortamı yerine sanal bir ortam oluşturulur. Çoğu zaman donanımlar, işletim sistemleri, depolama cihazları ve diğer ortamların bilgisayar tarafından oluşturulan sürümlerini içerir. Böylelikle tek bir fiziksel makine birçok sanal makineye bölünebilir ve ortak kaynakları kullanmalarına rağmen birbirinden bağımsız olarak kullanılabilir hale gelir.

Sanallaştırma, bir bilgisayar ya da sunucudan birden çok kaynak oluşturarak iş gücünü dağıtır. Böylelikle daha az enerji harcanır, sunucu yükü azaltılır, bakım ve altyapı maliyetleri düşürülür.

Dört ana sanallaştırma kategorisi vardır. İlki, tek bir merkezi sunucunun bireyselleştirilmiş masaüstleri sunup yönetmesine olanak tanıyan masaüstü sanallaştırmasıdır. İkincisi, ağ bant genişliğinin bağımsız kanallara bölünüp bu kanalların belirli sunucu veya cihazlara atanmasını sağlamak üzere tasarlanan ağ sanallaştırmasıdır. Üçüncü kategori, uygulamaları donanımdan ve işletim sisteminden ayıran yazılım sanallaştırmasıdır. Dördüncüsü ise çeşitli ağ depolama kaynaklarını birden çok kullanıcının erişebilmesi amacıyla tek bir depolama cihazı şeklinde birleştiren depolama sanallaştırmasıdır.

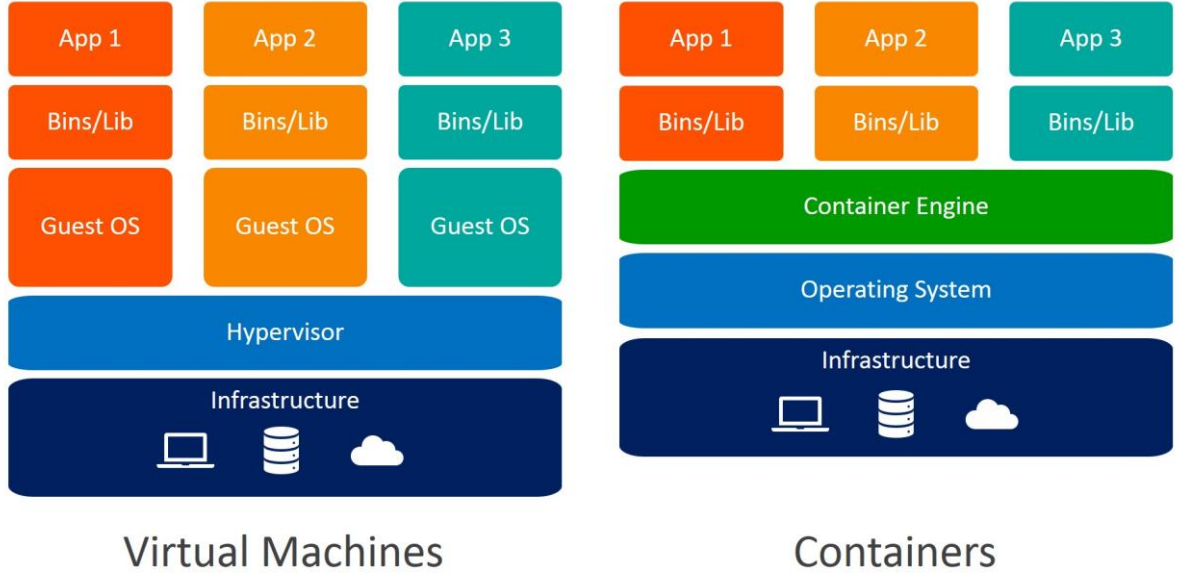


Şekil 2.1. Sanallaştırma görseli

2.2. Avantajlarının İncelenmesi

- Veri merkezi ölçeklendirme, raporlama, optimizasyon, yük dengeleme ve esneklik gibi konularda fayda sağlar.
- Veri merkezinde ihtiyaç duyulan fiziksel ekipmanların sayısını azaltır.
- Daha az fiziksel ekipman kullanılması ile enerji tasarrufu sağlanır.
- Sistem kaynaklarının yüksek verimlilikle kullanılmasını sağlar. Gereksiz kaynak kullanılmasını engeller.
- Gereksiz yatırımların yapılmasını engeller, yapılan yatırımların tam anlamıyla kullanılmasını sağlar.
- İhtiyaç halinde çok hızlı şekilde sunucu ve depolama kaynağı oluşturulmasına imkan sağlar.
- Donanım maliyetini düşürür.
- Yönetim kolaylığı sağlar. Birden çok sunucu ve bileşenin, takip edilmesi ve yönetilmesini sağlar.
- İş sürekliliği konusunda önemli rol oynar.

3. KONTEYNER



Şekil 3.1. Konteyner-Sanallaştırma karşılaştırması

3.1. Konteyner Nedir?

Konteynerler; bir geliştiricinin bir uygulamayı, kütüphaneler ve diğer bağımlılıklar gibi ihtiyaç duyduğu tüm parçalarla paketlemesini ve standart tek bir paket olarak göndermesini sağlar. Böylece uygulama bir bilgisayar ortamından diğerine gönderildiğinde hızlı ve güvenilir bir şekilde çalışır.

Konteynerlerin sanal makinalara göre en büyük avantajı tek bir işletim sistemi çalıştırarak onun üstünden gönderim yapmasıdır. Sanal makineler arasında gönderim yapıldığında ise tüm makinelerde gerekli işletim sistemi ve uygulamalar yüklü olması gerekmektedir.

3.2. Konteynerlerin Avantajları

- Sanal makineler gigabayt boyutunda yer kaplarken, konteynerler megabayt gibi çok daha düşük yer kaplarlar.
- Konteynerlerin çalışma öncesi hazırlık süresi çok azdır.

- Konteynerler uygulamalarımız farklı ortamlarda çalıştırmamızı kolaylaştırır.
- Konteynerler maliyet açısından verimlidir.

3.3. Konteynerlerin Dezavantajları

- Konteynerler sanal makinelere göre güvenlik açısından dezavantajlıdır. Çünkü konteynerlerin çekirdeğinin, sistemin diğer bileşenlerine ve root'a erişimi vardır. Bu yüzden çekirdekte bir güvenlik açığı olduğunda diğer kısımların da güvenliği tehlikeye düşebilir.
- Daha az esneklik sağlar. İşletim sistemi değiştirileceği zaman yeni bir sunucu başlatılması gerekir.

4. KONTEYNER ORKESTRASYONU

4.1. Orkestrasyon Nedir?

Konteyner orkestrasyonu, konteynerlerin çalışma döngüsünü yönetmekle ilgilidir. Konteyner orkestrasyon araçları bu yönetme işlemini kolaylaştırmak ve otomatikleştirmek için vardır.

4.2. Nasıl Çalışır?

Konteyner orkestrasyonu JSON veya YAML gibi dosyası üzerinden ayarlamaları yapılır. Bu yapılandırma dosyası kullandığımız konteyner orkestrasyon uygulmasına (Docker Swarm, Kubernetes) göre değişir. Orkestrasyon aracı bir konteyner sunucuya kurulacağı zaman en uygun sunucuyu seçer. Bunu bellek, işlemci vb. Sunucu kısıtlarını kontrol ederek yapar.

5. DOCKER

5.1. Docker Nedir?

Docker, yazılım geliştiriciler ve sistemciler için geliştirilen açık kaynaklı bir sanallaştırma platformudur. Docker ile Linux, Windows ve MacOSX üzerinde Linux ve Windows sanal containerler(makineler) çalıştırabilirsiniz. Bu platform sayesinde web sistemlerinin kurulumunu, testini ve dağıtımını kolaylıkla gerçekleştirebilirsiniz. En önemli özelliği belki de "Benim bilgisayarımda çalışıyordu, sunucuda neden çalışmadı acaba?" sorununu ortadan kaldırıyor olması.

Docker, yazılımların kurulmuş halinin imajını (.iso DVD imajı gibi) alıp tekrar kullanılabilir hale getiriyor. İsterseniz bu imajı bir kere oluşturup istediğiniz sunuculara gönderirsiniz, isterseniz her sunucuda sıfırdan imaj oluşturursunuz. Dockerfile adı verilen talimat dosyalarına bakarak her sunucu aynı imajı yeniden inşa edebilir. Bu sayede manuel müdahaleye gerek kalmaz.

Bir diğer güzel özelliği ise Dockerfile ve imajların geliştirilebilir olması. Talimatlara birkaç adım daha eklemek isterseniz önceki komutları tekrardan vermek yerine kaldığı en son yerden devam ediyor. Bu da zaman kazandırıyor.

5.2. Docker'ın Farkı

Bunu açıklamak için öncelikle var olan durumdan bir örnek üzerinde bahsedelim. Python/Flask ile geliştirdiğimiz bir web uygulamamız olsun.

Bunu Dockersız bir sunucuya kurmamız için gereken adımlar aşağı yukarı şu şekilde:

- Önce Python, virtualenv, pip gibi araçları kur.
- Git versiyon kontrol sistemini kur, gerekli depodan kodları çek.
- virtualenv ile sanal ortam oluştur, bağımlılıkları çek.
- Django sunucusunu çalıştır.
- Gerekirse nginx ile reverse proxy ayarı yap.

Veritabanı kurulum ve ayarları varsa işlem daha da uzuyor. Bunları manuel olarak yapmak gerekiyor, bu da her bir kurulum için zorluk demek. Özellikle de çok

sayıda bilgisayara aynı kurulum yapılacaksa ve belli aralıklarla güncelleme gerekebiliyorsa bu durum can sıkıcı olabiliyor. Docker da bu noktada imdadımıza yetişiyor. Tüm komutları tek bir Dockerfile dosyasına yazıp imaj oluşturup çok sayıda sunucuya bu imajı gönderebiliyoruz.

5.3. Dockerfile Nedir?

Dockerfile, bir yazılımın hangi baz işletim sistemi üzerine kurulacağı, hangi işletim sistemi ve programlama dili paketlerini gerektirdiğini, hangi komutla çalıştırılabileceğini anlatan bir script gibidir. Tek bir docker build komutuyla Dockerfile isimli dosya okunarak oradaki reçetemsî tanımlar baz alınarak sanal bir işletim sistemi oluşturulur, gereksinimler kurulur ve ortaya bir image (imaj) çıkar. Bu imajları bilgisayarımızda programı hızlıca çalıştırmak için docker run [image] komutunu kullanabiliriz. Bu durumda imajı baz alan bir container oluşturulacaktır.

5.4. Docker Container Nedir? Docker Image Nedir?

Docker, LXC sanallaştırma mekanizması üzerine kurulu. Bir Docker imajı, **container** denilen birimlerde çalıştırılıyor. Oyunculara rol vermek gibi düşünebiliriz. Her bir container bir süreç (process) kullanıyor. Bir makinada gücüne bağlı olarak binlerce docker containerı birden çalışabiliyor. Container imajları ortak olan sistem dosyalarını paylaşıyorlar. Dolayısıyla disk alanından tasarruf ediliyor. Şekilde görüldüğü gibi uygulama containerları ortak bin (exe) ve kütüphaneleri kullanıyorlar. Ancak klasik sanal makina sistemlerinde (VMWare, VirtualBox vs) her bir uygulama için ayrı işletim sistemi ve kütüphane dosyaları ayrılmak zorunda.

5.5. Docker Hub Nedir?

Docker Hub, oluşturulan docker imajlarının yüklenip ortak kullanıma açıldığı bir repodur (depo). Public ve Private olmak üzere iki tür repo vardır. Public olarak yayınlanan imajlar herkes tarafında görülebilmekte ve kullanılabilir. Mesela sürekli güncel sürümü yayınlanan Wordpress imajını indirip kolayca kullanabilir, kurulum derdinden kurtulabilirsiniz. Popüler birçok uygulamanın (mysql, apache, nginx, ghost, vs) hazırda ücretsiz imajları mevcut.

Bununla birlikte şirketiniz gerek baz imajlar, gerekse ürün imajlarını buradaki private reposuna koymak isteyebilir. Bu sayede frontend takımı backend kurulumu yapmak zorunda kalmadan backend kodunu docker container içerisinde çalıştırabilir.

Benzer şekilde deploymentları otomatikleştirebilirsiniz. Testler geçtiğinde Docker Hub reponuza yeni imaj yüklenir, ardından deployment tetiklenir ve bu imaj sunucularınıza yüklenir.

Docker Hub'daki imajları indirmek için terminalde `docker pull mysql` gibi bir komut uygulamanız yeterli. Ardından `docker run mysql` komutu ile yeni bir container oluşturabilirsiniz. Her komutta belirtilmesi gereken ek argümanlar mevcut. Bunları kullanım kılavuzunu inceleyerek görebilirsiniz.

6. KUBERNETES



Kubernetes

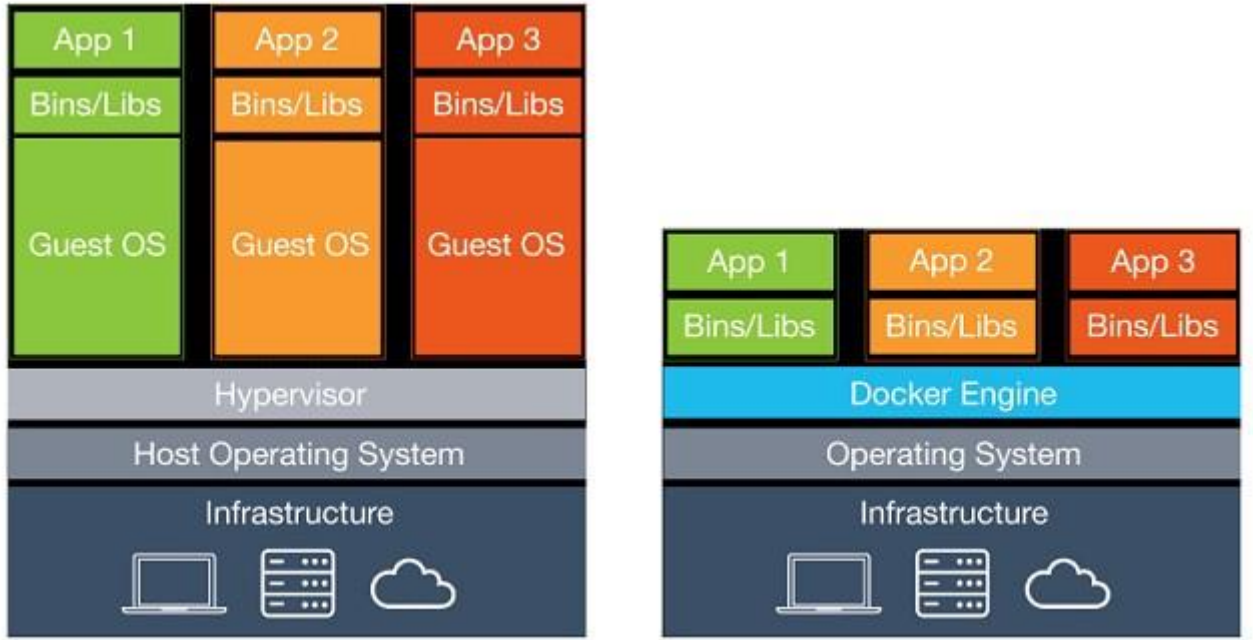
= Greek for "pilot" or
"helmsman of a ship"

Şekil 6.1. Kubernetes'in kelime anlamı

6.1 Kubernetes Nedir?

Kubernetes kelimesi Yunanca'dan geliyor ve anlamı dümenci (helmsman) veya pilot olarak karşımıza çıkıyor. Çoğu kaynaklarda kubernetesi *k8s* olarak yazıldığını görebilirsiniz. Bunun sebebi *k* ve *s* harflerinin arasında 8 tane harf olmasıdır.

Kubernetes Google tarafından GO dilinde geliştirilmiş Cloud Native Computing Foundation tarafından desteklenen mevcut konteyner haline getirilmiş uygulamalarınızı otomatik deploy etmek, sayılarını artırıp azaltmak gibi işlemler ile birlikte yönetmenizi sağlayan bir Konteyner kümeleme (container cluster) aracıdır.



Şekil 6.2. Sanallaştırma-Docker karşılaştırması

Bu teknolojiyi kullanarak uygulamalarımızı mikroservis mimarisi aracılığıyla daha kolay yönetebilir , daha kolay sorunlarımıza çözüm bulabilir ve daha da önemlisi taşınabilir bir ortam sağladığı için uygulamalarımızı çok kolay bir şekilde taşıyabiliriz. Sürüm geçişlerini de bu sayede kolay bir şekilde yapabiliriz.Bu özellikler aslında docker dediğimiz uygulama ile olmaktadır.Docker, dünyada en yaygın olarak kullanılan konteyner teknolojisidir.Aslında Docker, Linux Kernel'e 2008 yılında eklenen Linux Containers (LXC) üzerine kurulu bir teknolojidir. Docker, LXC'nin zengin mirasının üzerine oturmuştur fakat LXC'de manuel olarak yapılan işlemleri ustaca paketleyerek standart hale getirmiştir.Google bu teknolojiyi yıllardır kullandığını belirtiyor.Belki de bizim bilmediğimiz bir teknolojiyi şu an kullanmakta kimbilir.Bu teknolojinin hayatımıza girmesiyle birlikte bir çok ihtiyaç da ortaya çıkmıştır.

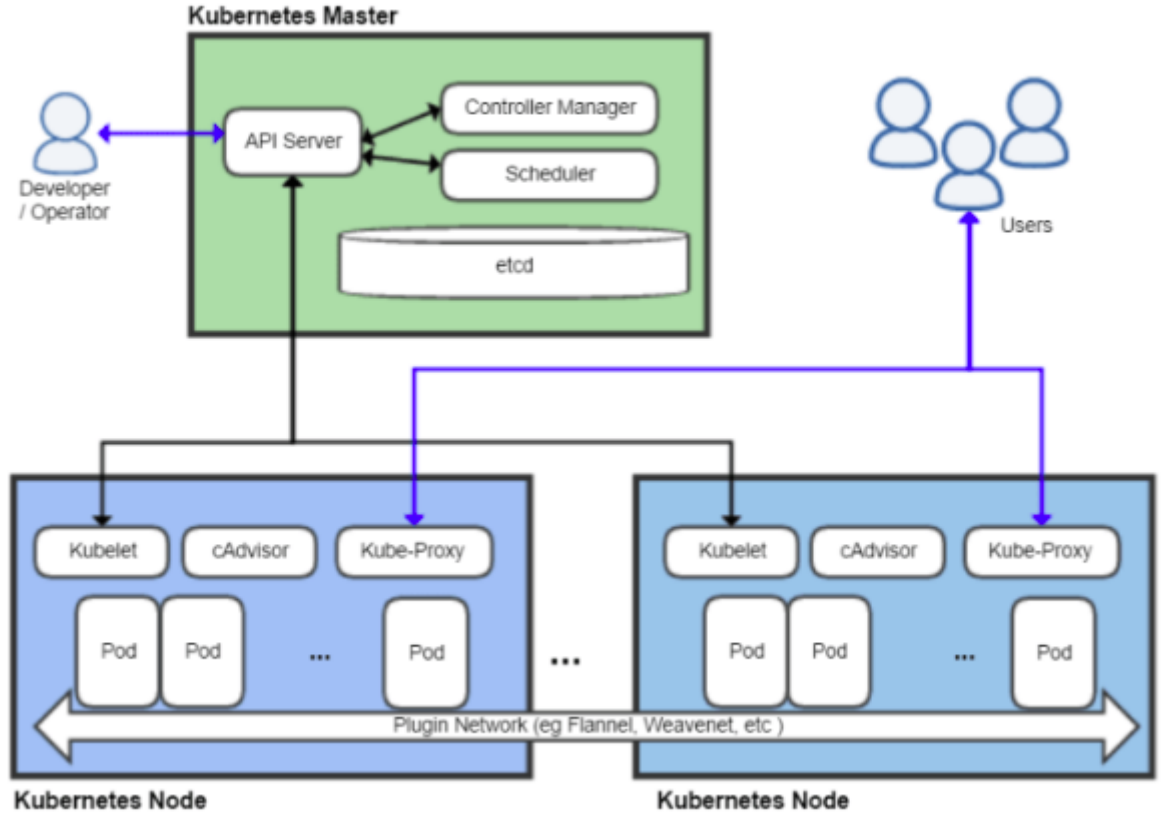
Örneğin;

- Konteyner çöktüğü zaman nasıl tekrar ayağa kaldırabilirim?
- CPU dar boğazına takıldığında nasıl daha çok konteyner ayağa kaldırabilir yada CPU belli bir seviye altında çalışmaya başladığında nasıl kullanılmayan konteyner'ları silebilirim?

Aslında genel olarak yönetim işlemlerinin daha kolay şekilde yapılmasına ihtiyaç duyulmuştur. Kubernetes, aslında bu ihtiyaçlar sonrasında ortaya çıkmıştır. Operasyonel işi zor ve maliyetli olan scaling, rollback, auto deployment, service discovery vb. gibi pek çok işlemi daha kolay bir şekilde yapmamızı sağlar. Bu sayede bir çok operasyonel işlemi tek bir elden yönetebilme imkanı sağlamaktadır. Birden fazla yönetim aracı vardır. Docker bu ihtiyaçlar için Docker Swarm 'ı ortaya çıkarmıştır. Onun dışında Apache Mesos vardır. Mesos, C++ dilinde ilk olarak Berkeley altında geliştirilen ve şu an Apache Foundation altında açık kaynak koda sahip olan projedir. Fakat Google, Kubernetes 'ı duyurunca piyasada önemli bir yer edinmiştir.

6.2. Kubernetesin Yapısının İncelenmesi

Kubernetes'in yapısına göz atacak olursak master ve worker dedikleri node'lardan oluşuyor. Bu worker dediğimiz node'lar içerisinde pod'lar ve pod'ların içerisinde konteyner'larımız bulunmaktadır. Bu yapı kendi içinde overlay dediğimiz bir network ile haberleşiyor.



Şekil 6.3. Kubernetes'in yapısı

Master node içerisinde 4 tane temel yapıtaşı vardır. Bunlar;

- API Server
- Controller Manager
- Scheduler
- Etcd

API Server: Master sunucumuza gelen tüm REST requestlerin yönetilmesinden sorumludur. Aslınsa Cluster'ın beyni diyebiliriz. Json file'ları ile yönetilir.

Controller Manager: Temel olarak, bir denetleyicidir, kümenin durumunu API Server izleme özelliğiyle izler ve bildirildiğinde, geçerli durumu istenen duruma doğru hareket ettirmek için gerekli değişiklikleri yapar.

Scheduler: Bir pod'un hangi node üzerinde çalışacağına karar verir , kubelet'i tetikler ve ilgili pod ve içindeki konteyner oluşturulur. Kısacası yeni bir pod oluşturulması isteğine karşı API server'ı izler.

Etd: Coreos tarafında yaratılmış open source distributed, tutarlı ve izlenebilir bir key value store (nosql database) diyebiliriz. Redhat Coreos 'si satın aldı aslında konteyner teknolojisinde çok güzel hamleler atıyor.

6.3. Kubernetes Node'larının İncelenmesi

Şimdi de kubernetes node'larının yapıtaşlarına bakalım.

- Kubelet
- Kube-Proxy
- Pod
- Container Engine

Kubelet: Node üzerindeki ana kubernetes ajanıdır. API server'dan gelecek isteklere karşı API Server'ı izler. İlgili docker servisi ile konuşarak Pod'u ayağa kaldırır ve bunun bilgisini API Server'a iletir.

Kube-Proxy: Kubernetes network'u diyebiliriz. Pod'lara IP adresi proxy sayesinde atanır. Bir pod'un içindeki tüm konteyner'lar bir adet paylaşımlı IP 'yi kullanır. Kube-proxy aynı zamanda bir servisin altındaki tüm pod'lara load-balance özelliği kazandırır.

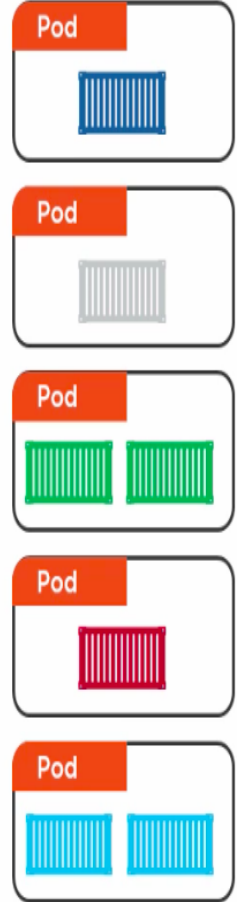
Container Engine:Konteyner yönetimini yapar.İmage'ları ilgili registry üzerinden çeker ve konteyner'ların start ve stop olmalarını sağlar.

Pod:Konteyner 'ların çalışma alanı diyebilir.Bir pod içerisinde birden fazla konteyner çalışabilir.Kubernetes tarafında çalışma şekli gereği yeni bir deploy isteği geldiğinde pod'un yeni versiyonu oluşturulur ve çalıştığı görüldüğünde diğer pod versiyonu kapatılır.Dolayısıyla pod içerisinde birden fazla konteyner olduğu durumda diğer uygulamala konteyner'ları etkileneceğinden bir pod içerisinde bir konteyner tavsiye edilir.Pod öldüğünde tekrar geri kalkmaz aynı imajdan onun yerine yeni bir pod ortaya çıkar.Aşağıdaki resime baktığınızda kafanızda daha iyi şekillenebilir.



Containers always run
inside of pods

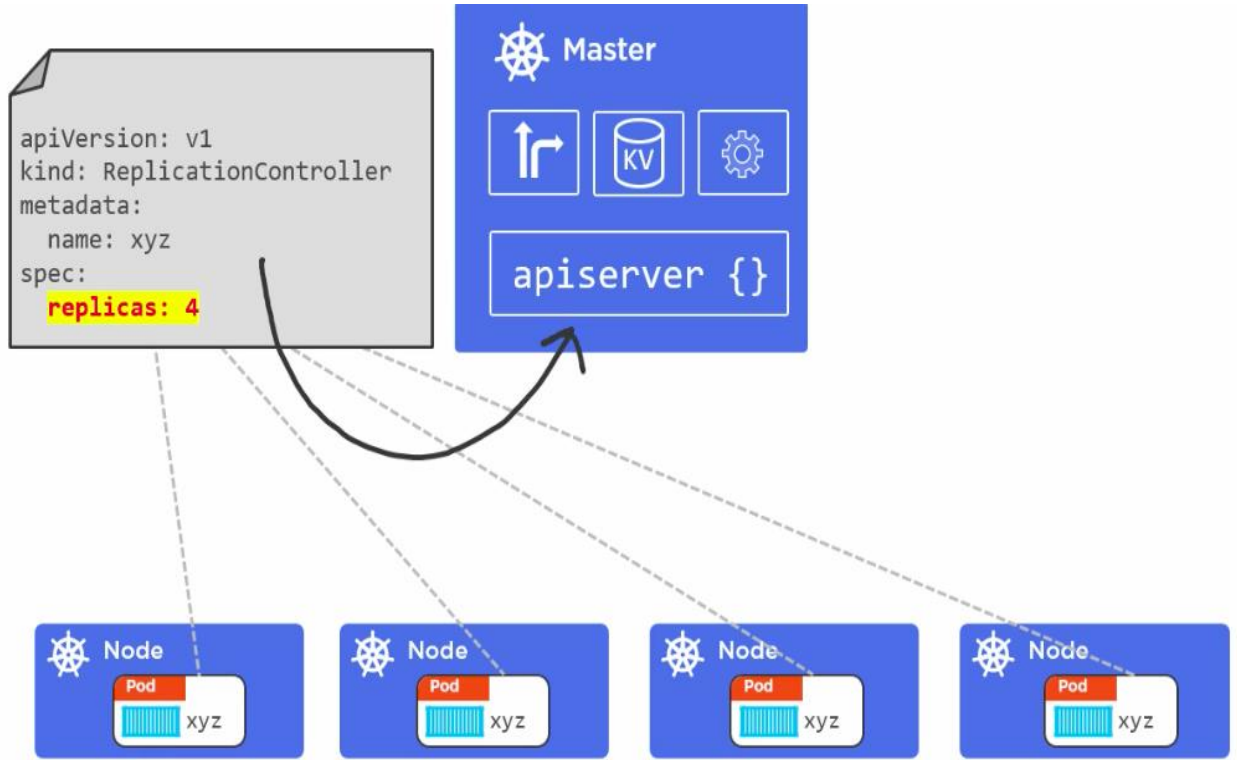
Pods can have multiple
containers
(advanced use-case)



Şekil 6.4. Pod görseli

6.4. Bazı Terimler

ReplicaSet: Bir poddan kaç tane olacağını replicaset ile belirtiyoruz. İstedığınız anda bir pod'u kesintisiz bir şekilde scale edebilir yada azaltabiliriz.



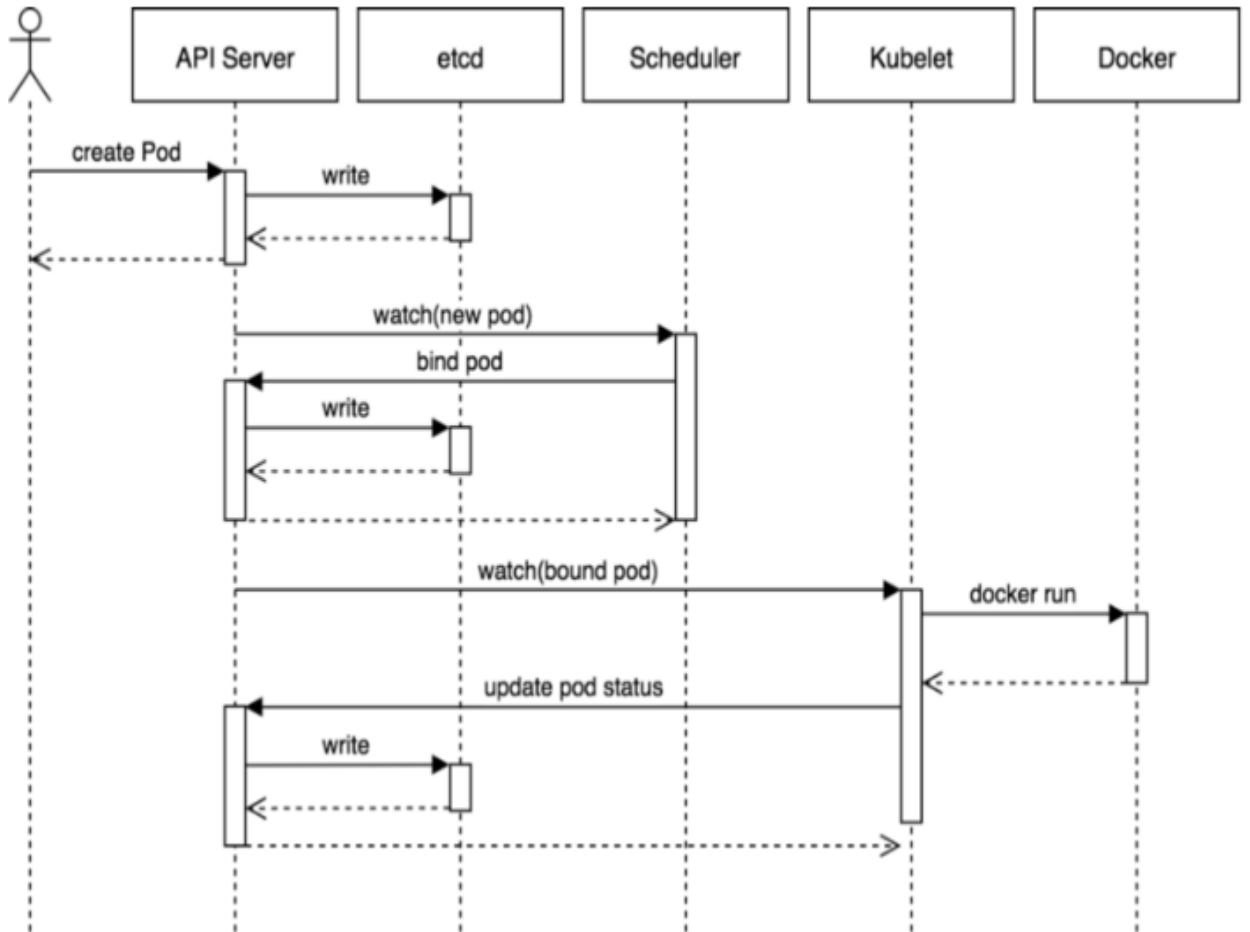
Şekil 6.5. Replica Set

Namespace: Namespace girerek aslında ortamlarımızı birbirinden izole edebiliyoruz.

Service: Service, pod'ların ön tarafında konumlanan ve gelen istekleri karşılayıp arka tarafa pod'lara gönderen katman. Blue-green deployment, pod scaling gibi işlemleri service sayesinde kesintisiz olarak yapabiliyoruz.

Secret: Parola, kullanıcı, token gibi bilgileri güvenli bir şekilde depolayacağımız alan secret'tır. Burada depoladığımız bilgilere verdiğimiz isimle uygulamamız içinde kullanabiliriz. Şifre yazarak değil de aslında öncesinde yarattığımız secret ile uygulamayı yönetiyoruz.

6.5. Kubernetes Çalışma Şekli



Şekil 6.6. Kubernetes'in çalışma yapısı

Kubecttl (kubernetes client) isteği API server 'a iletir.

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

Konteyner teknolojisini, bulut teknolojilerinde lider servis sağlayıcıları servis olarak sunmaktadır. Bunlar:

Amazon Elastic Container Service (Amazon ECS) ==> AWS

Azure Kubernetes Service(AKS)==> AZURE

Google Kubernetes Service ==> Google Cloud Platform

7. KUBERNETES ÖRNEK

7.1. Sistem Gereksinimleri

Ana sanal makine için sistem gereksinimleri:

- 4 GB RAM
- 2 Cores of CPU

Node'lar için sistem gereksinimleri:

- 2 GB RAM
- 1 Core of CPU

7.2. Örneğin Gerçekleştirilmesi

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
volkanub1@volkanub1-VirtualBox:~$ sudo su
[sudo] password for volkanub1:
root@volkanub1-VirtualBox:/home/volkanub1# apt-get update
```

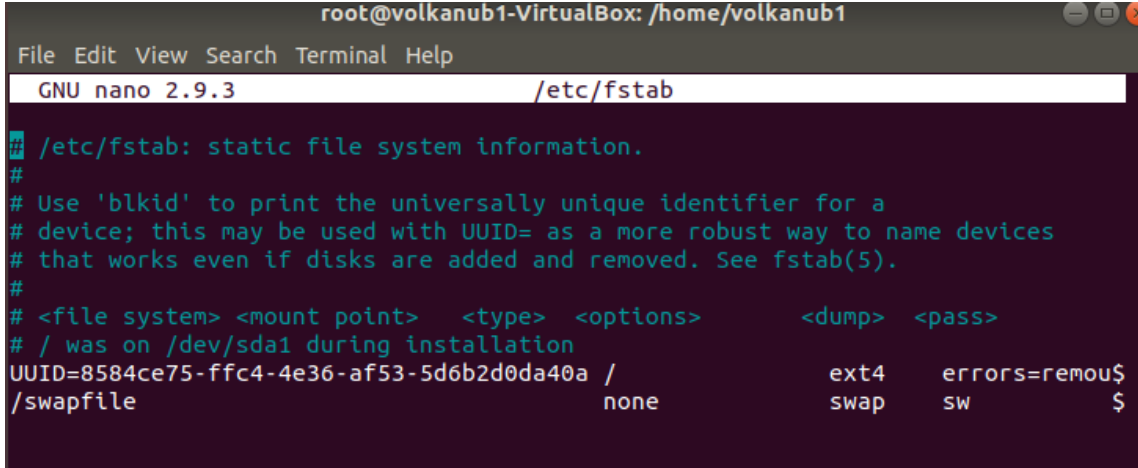
Şekil 7.2.1.

Şekil 7.2.1.'de linux depolarına (repository) güncelleme komutunu verdik.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
root@volkanub1-VirtualBox:/home/volkanub1# swapoff -a
root@volkanub1-VirtualBox:/home/volkanub1# nano /etc/fstab
```

Şekil 7.2.2.

Şekil 7.2.2.'de swap alanını kapattık ve fstab dosyasına eriştik.



```

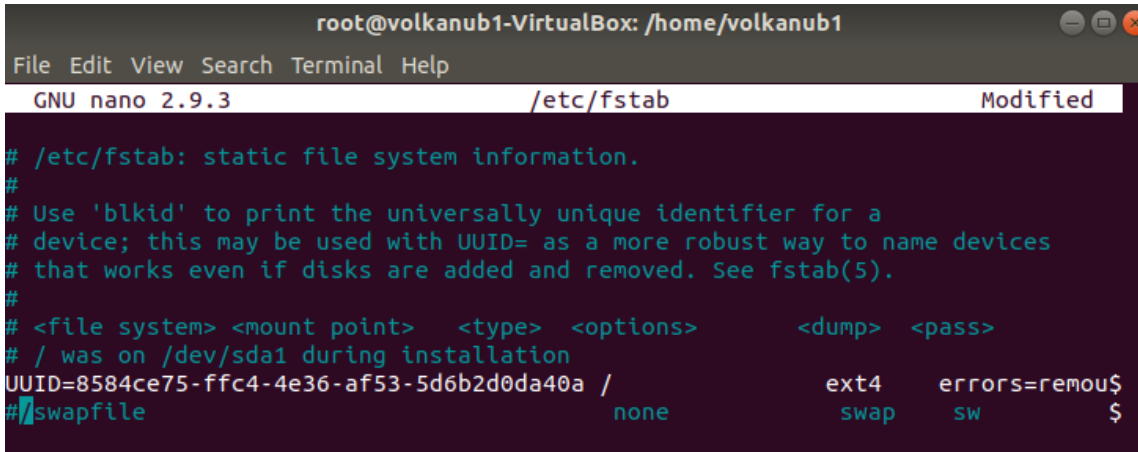
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/fstab

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=8584ce75-ffc4-4e36-af53-5d6b2d0da40a / ext4 errors=remou$
/swapfile none swap sw $

```

Şekil 7.2.3.

Şekil 7.2.3.'de fstab dosyasının içeriğini görüyoruz.



```

root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/fstab Modified

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=8584ce75-ffc4-4e36-af53-5d6b2d0da40a / ext4 errors=remou$
# /swapfile none swap sw $

```

Şekil 7.2.4.

Şekil 7.2.4.'de swap dosyasının satırına # işareti ekleyerek yorum satırı haline getirdik ve etkisiz hale geldi.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
root@volkanub1-VirtualBox:/home/volkanub1# nano /etc/hostname
```

Şekil 7.2.5.

Şekil 7.2.5.'de hostname'i açtık.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hostname
volkanub1-VirtualBox
```

Şekil 7.2.6.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hostname
kmaster
```

Şekil 7.2.7.

Şekil 7.2.6. ve 7.2.7.'de hostname'de yaptığımız değişikliği görüyoruz.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
root@volkanub1-VirtualBox:/home/volkanub1# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.10.2 netmask 255.255.255.240 broadcast 172.20.10.15
    inet6 fe80::dce:ef1b:bba4:841f prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:26:2f:e0 txqueuelen 1000 (Ethernet)
    RX packets 1575 bytes 1671906 (1.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1192 bytes 93451 (93.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 239 bytes 18609 (18.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 239 bytes 18609 (18.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@volkanub1-VirtualBox:/home/volkanub1#
```

Şekil 7.2.8.

Şekil 7.2.8.’de ifconfig komutu ile ip adresimizi öğrendik.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
root@volkanub1-VirtualBox:/home/volkanub1# nano /etc/hosts
```

Şekil 7.2.9.

```

root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts

127.0.0.1    localhost
127.0.1.1    volkanub1-VirtualBox

# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters

```

Şekil 7.2.10.

```

root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts

127.0.0.1    localhost
127.0.1.1    volkanub1-VirtualBox
172.20.10.2   kmaster
# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters

```

Şekil 7.2.11.

Şekil 7.2.9. , 7.2.10. ve 7.2.11’de hosts dosyasına öğrendiğimiz ip adresimizi ve host’umuza koyduğumuz ismimizi girdik.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
root@volkanub1-VirtualBox:/home/volkanub1# nano /etc/network/interfaces
```

Şekil 7.2.12.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
```

Şekil 7.2.13.

```
root@volkanub1-VirtualBox: /home/volkanub1
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s8
iface enp0s8 inet static
address 172.20.10.2
```

Şekil 7.2.14.

Şekil 7.2.12. , Şekil 7.2.13. ve Şekil 7.2.14.’de ip adresimizi statik hale getirdik. Bu adımdan sonra sanal makinemizi yeniden başlatarak yaptığımız değişikliklerin geçerli olmasını sağladık.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.10.2 netmask 255.255.255.240 broadcast 172.20.10.15
    inet6 fe80::dce:ef1b:bba4:841f prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:26:2f:e0 txqueuelen 1000 (Ethernet)
    RX packets 43 bytes 10287 (10.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 64 bytes 6756 (6.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

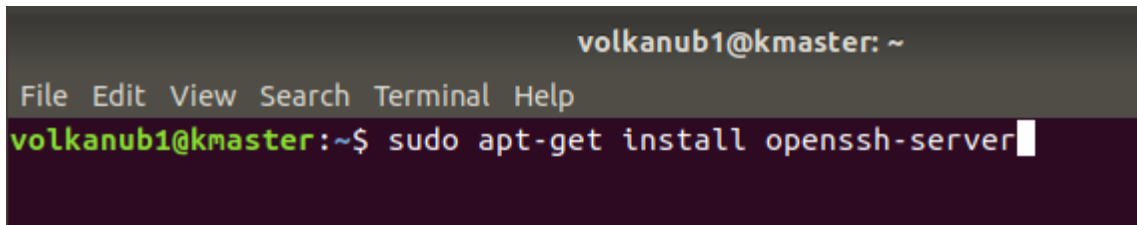
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 130 bytes 10078 (10.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 130 bytes 10078 (10.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

volkanub1@kmaster:~$

```

Şekil 7.2.15.

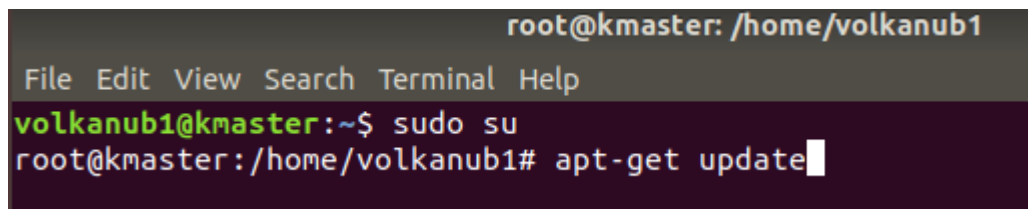
Şekil 7.2.15.’de sanal makinemizi yeniden başlattıktan sonra yaptığımız değişikliklerin geçerli olup olmadığını kontrol ettik. Gördüğümüz gibi makinemizin ismi kmaster’e dönüşmüş.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ sudo apt-get install openssh-server
```

Şekil 7.2.16.

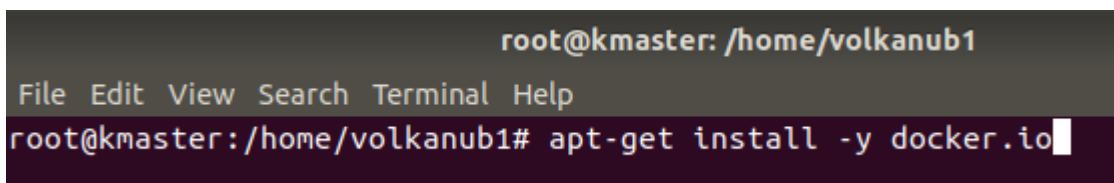
Şekil 7.2.16’da openssh server’i yükleme komutu verdik.



```
root@kmaster: /home/volkanub1  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ sudo su  
root@kmaster:/home/volkanub1# apt-get update
```

Şekil 7.2.17.

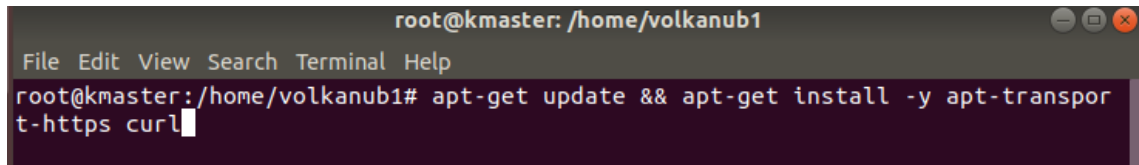
Şekil 7.2.17’de openssh yüklemesi sonrası linux depolarımıza güncelleme komutu uyguladık.



```
root@kmaster: /home/volkanub1  
File Edit View Search Terminal Help  
root@kmaster:/home/volkanub1# apt-get install -y docker.io
```

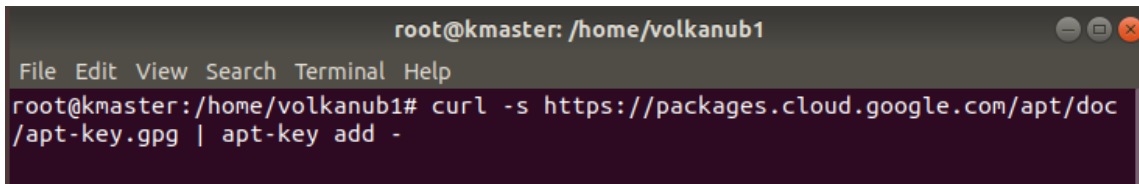
Şekil 7.2.18.

Şekil 7.2.18.’de sanal makinemize docker’ı yükleme komutu verdik.



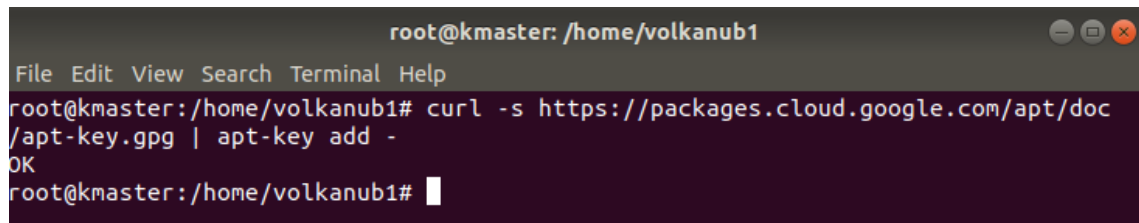
```
root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# apt-get update && apt-get install -y apt-transport-https curl
```

Şekil 7.2.19.



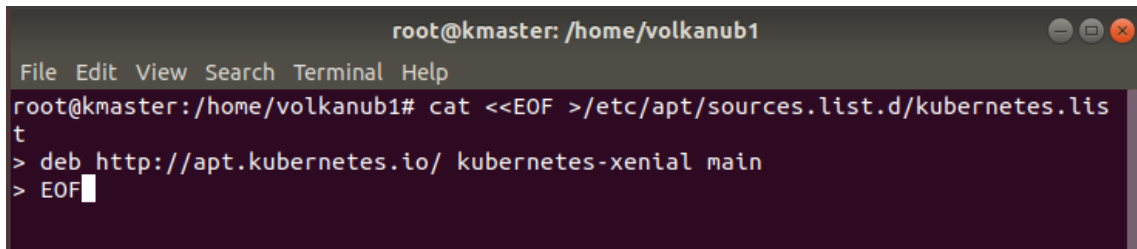
```
root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
```

Şekil 7.2.20.



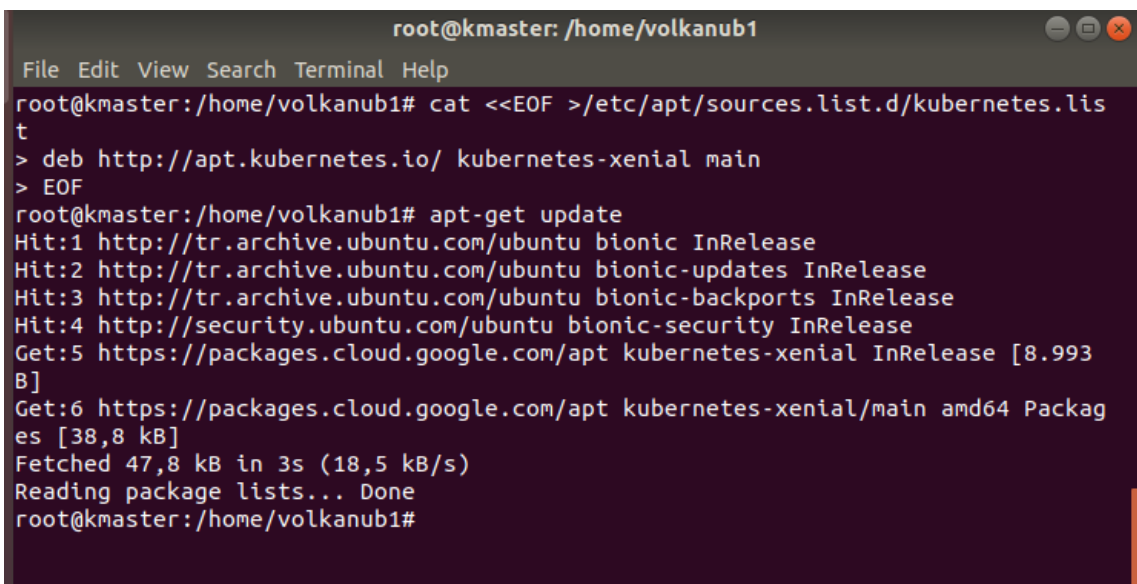
```
root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
OK
root@kmaster:/home/volkanub1#
```

Şekil 7.2.21.



```
root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# cat <<EOF >/etc/apt/sources.list.d/kubernetes.lis
t
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
```

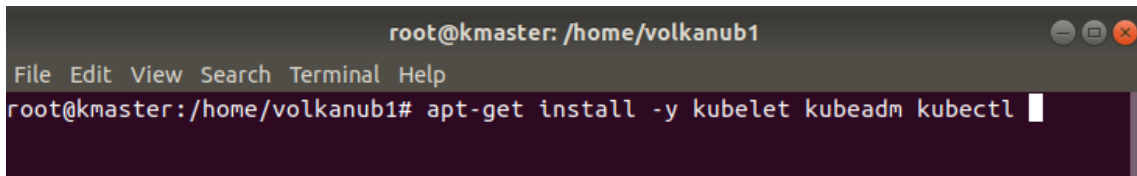
Şekil 7.2.22.



```
root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# cat <<EOF >/etc/apt/sources.list.d/kubernetes.lis
t
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
root@kmaster:/home/volkanub1# apt-get update
Hit:1 http://tr.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://tr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://tr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8.993
B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packag
es [38,8 kB]
Fetched 47,8 kB in 3s (18,5 kB/s)
Reading package lists... Done
root@kmaster:/home/volkanub1#
```

Şekil 7.2.23.

Şekil 7.2.19. , Şekil 7.2.20. , Şekil 7.2.21. , Şekil 7.2.22 ve Şekil 7.2.23’de kubernetes ortamının bileşenleri olan kubelet, kubeadm ve kubectl’in yüklenmesi öncesi gerekli komutları yazdık.



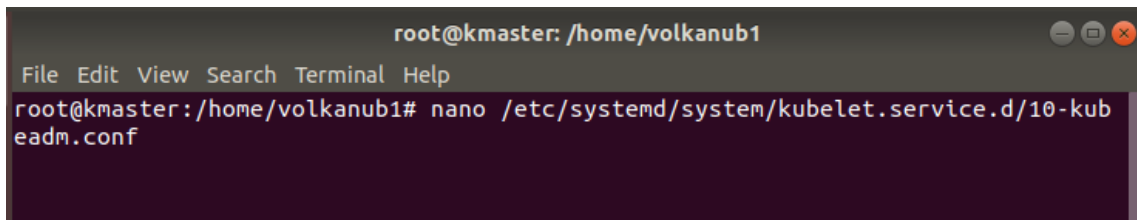
```

root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# apt-get install -y kubelet kubeadm kubectl

```

Şekil 7.2.24.

Şekil 7.2.24.'de kubeadm, kubectl ve kubelet yükleme komutlarını verdik. -y ile yükleme sırasında çıkacak sorulara yes seçeneğini belirttiğimizi ifade ettik.

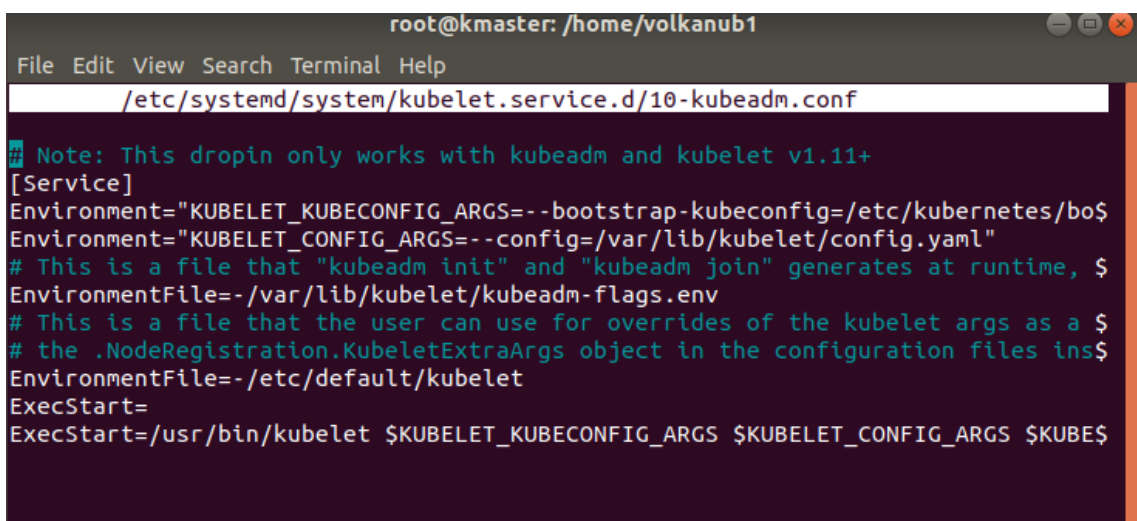


```

root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
root@kmaster:/home/volkanub1# nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

```

Şekil 7.2.25.



```

root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf

# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bo$
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, $
EnvironmentFile=/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a $
# the .NodeRegistration.KubeletExtraArgs object in the configuration files ins$
EnvironmentFile=/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBE$

```

Şekil 7.2.26

```

root@kmaster: /home/volkanub1
File Edit View Search Terminal Help
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf Modified

# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bo$
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, $
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a $
# the .NodeRegistration.KubeletExtraArgs object in the configuration files ins$
EnvironmentFile=-/etc/default/kubelet
Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBE$

```

Şekil 7.2.27.

Şekil 7.2.25. , Şekil 7.2.26. ve Şekil 7.2.27’de kubernetes konfigürasyonunda yapılması gereken değişikliği yaptık. Buraya kadar olan işlemlerin hepsi node olarak belirlediğimiz ikinci sanal makinemizde de yapılacak.

```

root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
volkanub2@volkanub2-VirtualBox:~$ sudo su
[sudo] password for volkanub2:
root@volkanub2-VirtualBox:/home/volkanub2# apt-get update

```

Şekil 7.2.28.

Şekil 7.2.28.’de linux depolarına (repository) güncelleme komutunu verdik.

```

root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
root@volkanub2-VirtualBox: /home/volkanub2# swapoff -a
root@volkanub2-VirtualBox: /home/volkanub2# nano /etc/fstab

```

Şekil 7.2.29.

Şekil 7.2.29.'de swap alanını kapattık ve fstab dosyasına eriştik.

```

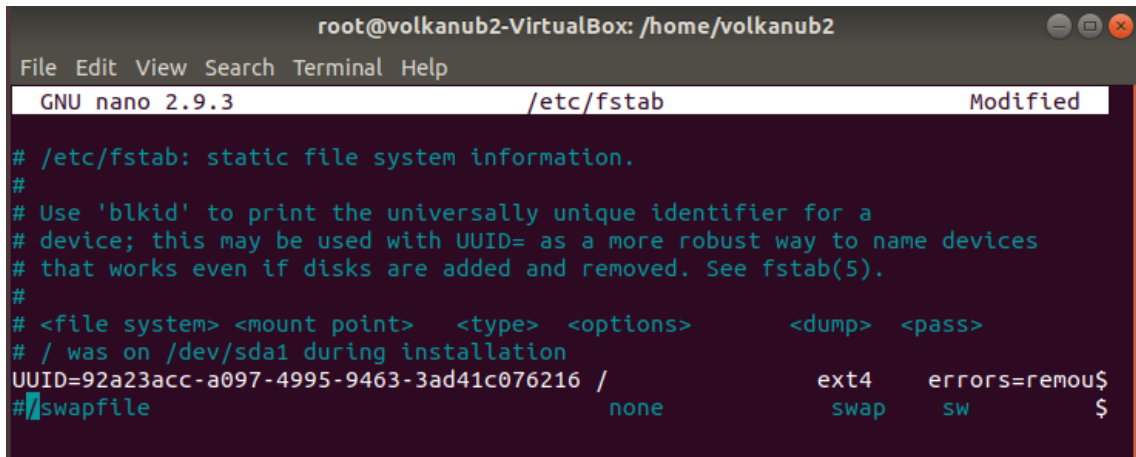
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/fstab

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=92a23acc-a097-4995-9463-3ad41c076216 / ext4 errors=remou$
/swapfile none swap sw $

```

Şekil 7.2.30.

Şekil 7.2.30.'de fstab dosyasının içeriğini görüyoruz.



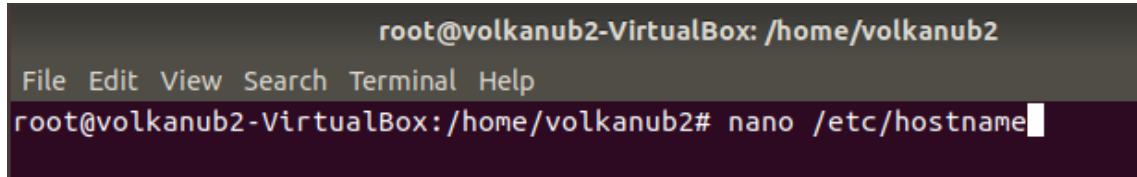
```

root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/fstab Modified
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=92a23acc-a097-4995-9463-3ad41c076216 / ext4 errors=remou$
# swapfile none swap sw $

```

Şekil 7.2.31.

Şekil 7.2.31.'de swap dosyasının satırına # işareti ekleyerek yorum satırı haline getirdik ve etkisiz hale geldi.



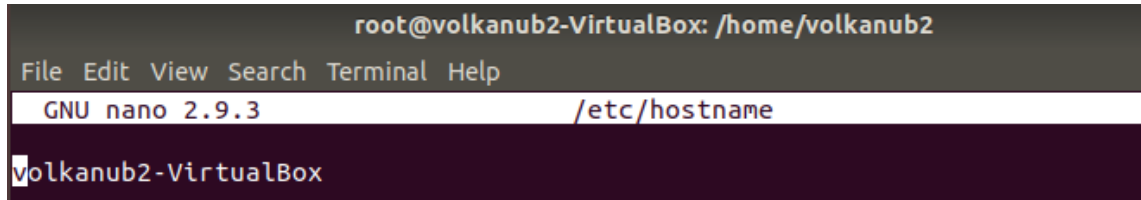
```

root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
root@volkanub2-VirtualBox:/home/volkanub2# nano /etc/hostname

```

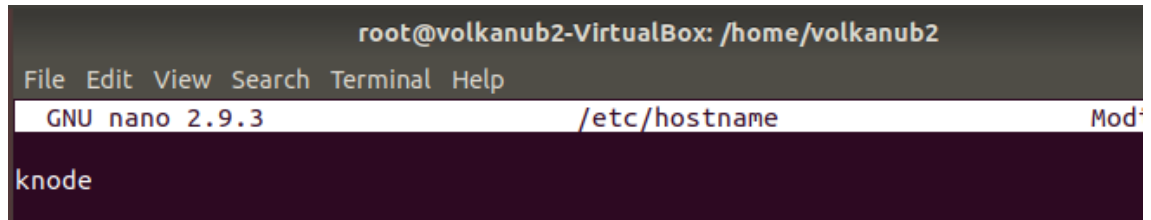
Şekil 7.2.32.

Şekil 7.2.32.'de hostname'i açtık.



```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hostname
volkanub2-VirtualBox
```

Şekil 7.2.33.



```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hostname Mod
knode
```

Şekil 7.2.34.

Şekil 7.2.33. ve 7.2.34’de hostname’de yaptığımız değişikliği görüyoruz.

```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
root@volkanub2-VirtualBox:/home/volkanub2# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.10.3 netmask 255.255.255.240 broadcast 172.20.10.15
    inet6 fe80::9353:f68a:5df6:671a prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:53:7d:79 txqueuelen 1000 (Ethernet)
    RX packets 13096 bytes 15732127 (15.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8737 bytes 690273 (690.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 453 bytes 47367 (47.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 453 bytes 47367 (47.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

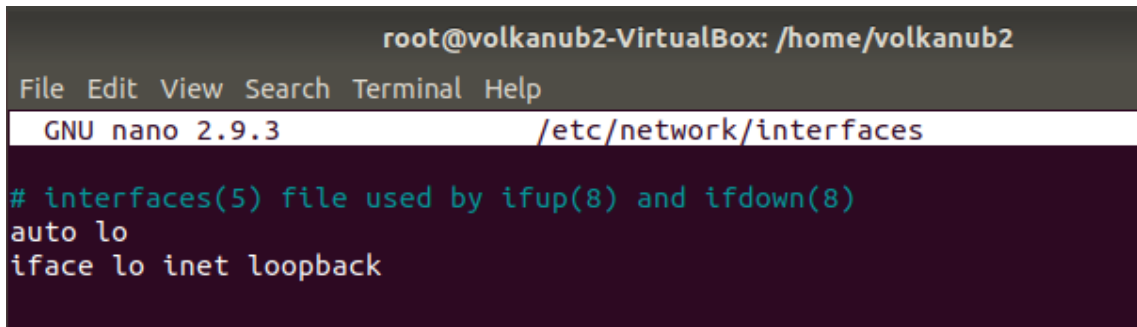
root@volkanub2-VirtualBox:/home/volkanub2#
```

Şekil 7.2.35.

Şekil 7.2.35.’de ifconfig komutu ile ip adresimizi öğrendik.

```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
root@volkanub2-VirtualBox:/home/volkanub2# nano /etc/network/interfaces
```

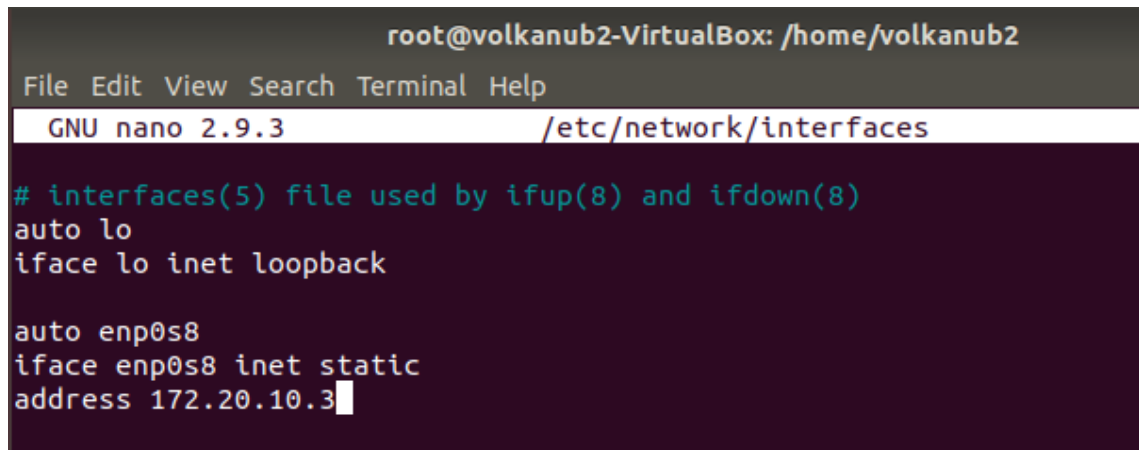
Şekil 7.2.36.



```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
```

Şekil 7.2.37.



```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s8
iface enp0s8 inet static
address 172.20.10.3
```

Şekil 7.2.38.

Şekil 7.2.36. , Şekil 7.2.37. ve Şekil 7.2.38.’de ip adresimizi statik hale getirdik.

```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
root@volkanub2-VirtualBox:/home/volkanub2# nano /etc/hosts
```

Şekil 7.2.39.

```
root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts
127.0.0.1    localhost
127.0.1.1    volkanub2-VirtualBox
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

Şekil 7.2.40.

```

root@volkanub2-VirtualBox: /home/volkanub2
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts

127.0.0.1    localhost
127.0.1.1    volkanub2-VirtualBox
172.20.10.3   knode
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

```

Şekil 7.2.41.

Şekil 7.2.39. , 7.2.40. ve 7.2.41’de hosts dosyasına öğrendiğimiz ip adresimizi ve host’umuza koyduğumuz ismimizi girdik. Bu adımdan sonra sanal makinemizi yeniden başlatarak yaptığımız değişikliklerin geçerli olmasını sağladık.

```

volkanub2@knode: ~
File Edit View Search Terminal Help
volkanub2@knode:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.20.10.3  netmask 255.255.255.240  broadcast 172.20.10.15
    inet6 fe80::9353:f68a:5df6:671a  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:53:7d:79  txqueuelen 1000  (Ethernet)
    RX packets 50  bytes 12661 (12.6 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 62  bytes 6592 (6.5 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

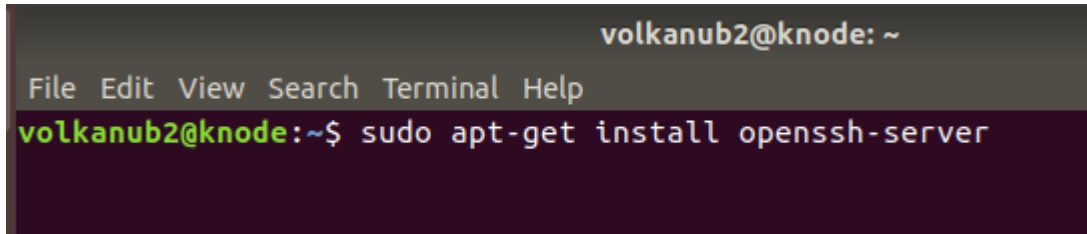
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 127  bytes 9838 (9.8 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 127  bytes 9838 (9.8 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

volkanub2@knode:~$

```

Şekil 7.2.42.

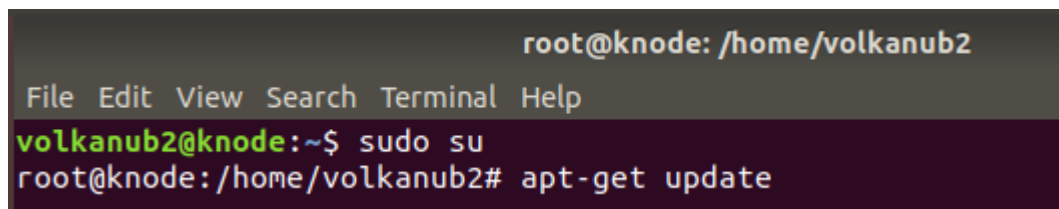
Şekil 7.2.42.'de sanal makinemizi yeniden başlattıktan sonra yaptığımız değişikliklerin geçerli olup olmadığını kontrol ettik. Gördüğümüz gibi makinemizin ismi knode'a dönüşmüş.



```
volkanub2@knode: ~  
File Edit View Search Terminal Help  
volkanub2@knode:~$ sudo apt-get install openssh-server
```

Şekil 7.2.43.

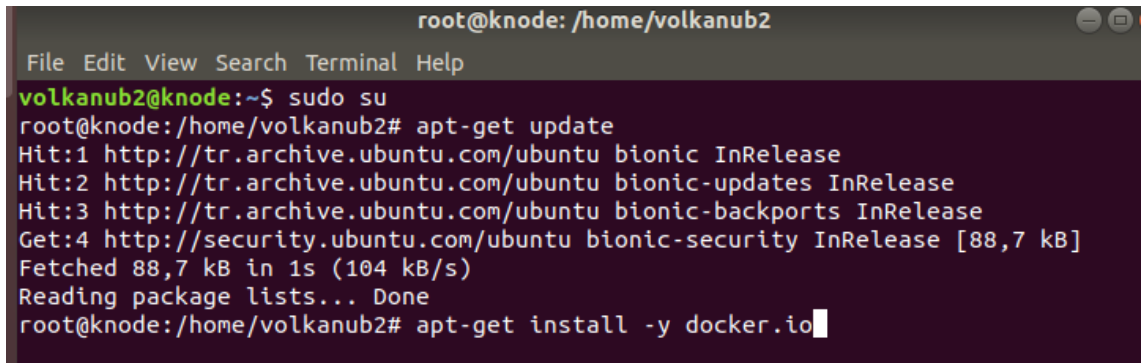
Şekil 7.2.43.'da openssh server'i yükleme komutu verdik.



```
root@knode: /home/volkanub2  
File Edit View Search Terminal Help  
volkanub2@knode:~$ sudo su  
root@knode:/home/volkanub2# apt-get update
```

Şekil 7.2.44.

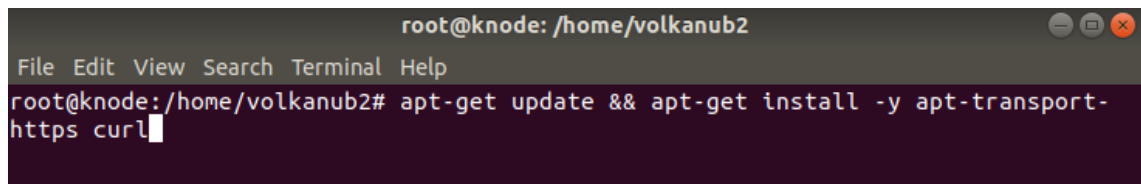
Şekil 7.2.44'de openssh yüklemesi sonrası linux depolarımıza güncelleme komutu uyguladık.

A terminal window titled 'root@knode: /home/volkanub2' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'volkanub2@knode:~\$'. The user enters 'sudo su', changing the prompt to 'root@knode:/home/volkanub2#'. Then, 'apt-get update' is entered, showing output for bionic, bionic-updates, bionic-backports, and bionic-security. Finally, 'apt-get install -y docker.io' is entered, with the cursor at the end of the command.

```
root@knode: /home/volkanub2
File Edit View Search Terminal Help
volkanub2@knode:~$ sudo su
root@knode:/home/volkanub2# apt-get update
Hit:1 http://tr.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://tr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://tr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Fetched 88,7 kB in 1s (104 kB/s)
Reading package lists... Done
root@knode:/home/volkanub2# apt-get install -y docker.io
```

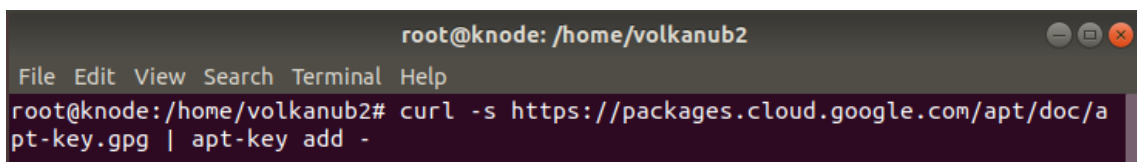
Şekil 7.2.45.

Şekil 7.2.45.'de sanal makinemize docker'ı yükleme komutu verdik.

A terminal window titled 'root@knode: /home/volkanub2' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'root@knode:/home/volkanub2#'. The user enters 'apt-get update && apt-get install -y apt-transport-https curl', with the cursor at the end of the command.

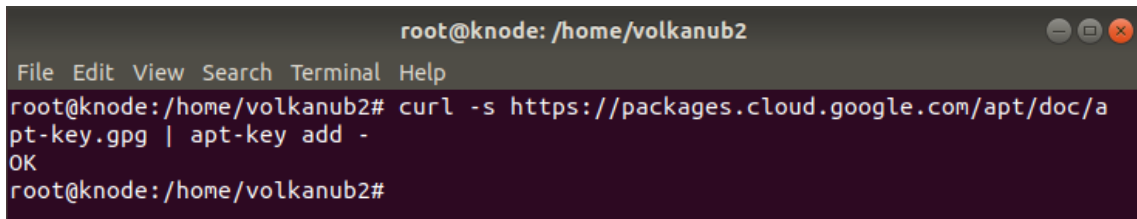
```
root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode:/home/volkanub2# apt-get update && apt-get install -y apt-transport-https curl
```

Şekil 7.2.46.

A terminal window titled 'root@knode: /home/volkanub2' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'root@knode:/home/volkanub2#'. The user enters 'curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -', with the cursor at the end of the command.

```
root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode:/home/volkanub2# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
```

Şekil 7.2.47.

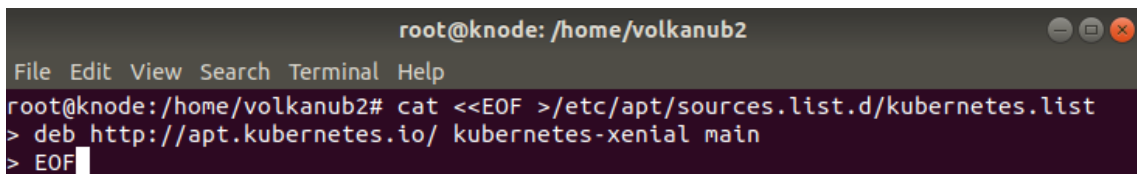


```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode: /home/volkanub2# curl -s https://packages.cloud.google.com/apt/doc/a
pt-key.gpg | apt-key add -
OK
root@knode: /home/volkanub2#

```

Şekil 7.2.48.

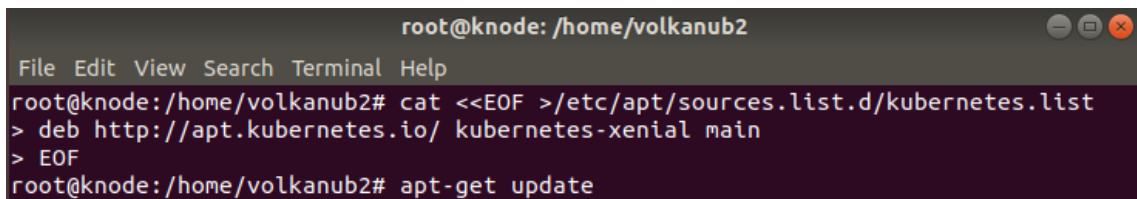


```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode: /home/volkanub2# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF

```

Şekil 7.2.49.



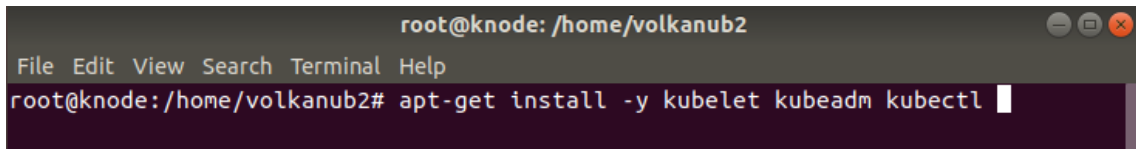
```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode: /home/volkanub2# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
root@knode: /home/volkanub2# apt-get update

```

Şekil 7.2.50.

Şekil 7.2.46. , Şekil 7.2.47. , Şekil 7.2.48. , Şekil 7.2.49 ve Şekil 7.2.50.'de kubernetes ortamının bileşenleri olan kubelet, kubeadm ve kubectl'in yüklenmesi öncesi gerekli komutları yazdık.



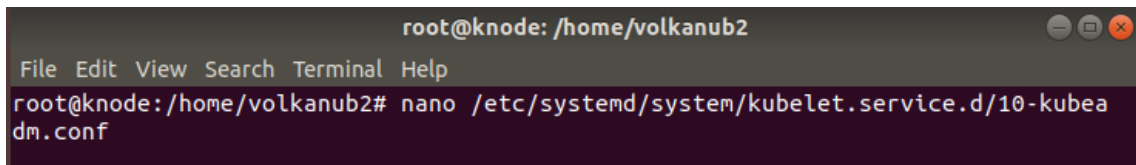
```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode:/home/volkanub2# apt-get install -y kubelet kubeadm kubectl

```

Şekil 7.2.51.

Şekil 7.2.51.'de kubeadm, kubectl ve kubelet yükleme komutlarını verdik. -y ile yükleme sırasında çıkacak sorulara yes seçeneğini belirttiğimizi ifade ettik.

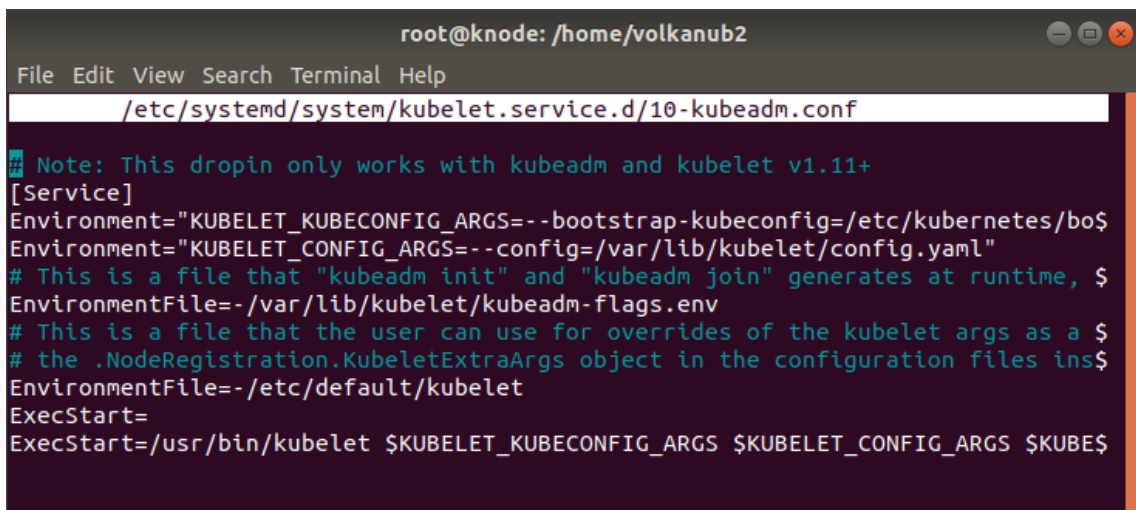


```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
root@knode:/home/volkanub2# nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

```

Şekil 7.2.52.



```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf

# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bo$
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, $
EnvironmentFile=/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a $
# the .NodeRegistration.KubeletExtraArgs object in the configuration files ins$
EnvironmentFile=/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBE$

```

Şekil 7.2.53.

```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf Modified

# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bo$
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, $
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a $
# the .NodeRegistration.KubeletExtraArgs object in the configuration files ins$
EnvironmentFile=-/etc/default/kubelet
Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBE$

```

Şekil 7.2.54.

Şekil 7.2.52. , Şekil 7.2.53. ve Şekil 7.2.54’de kubernetes konfigürasyonunda yapılması gereken değişikliği yaptık.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ sudo nano /etc/hosts

```

Şekil 7.2.55.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts
127.0.0.1    localhost
127.0.1.1    volkanub1-VirtualBox
172.20.10.2   kmaster
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

```

Şekil 7.2.56.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts
127.0.0.1    localhost
127.0.1.1    volkanub1-VirtualBox
172.20.10.2   kmaster
172.20.10.3   knode
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

```

Şekil 7.2.57.

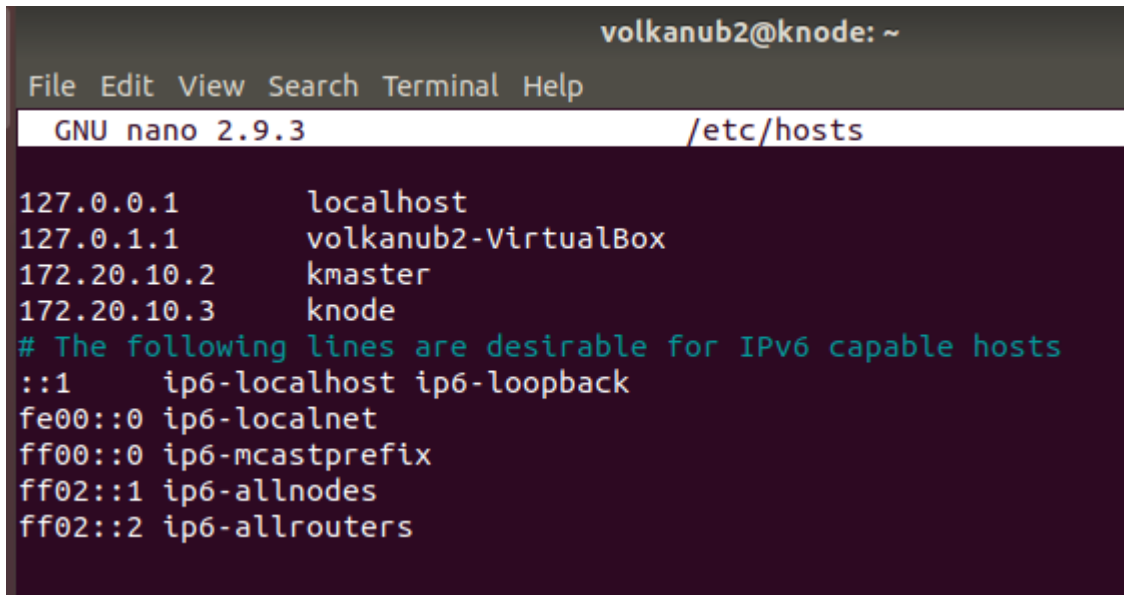
Şekil 7.2.55. , Şekil 7.2.56. ve Şekil 7.2.57.'de ana makine olarak belirlediğimiz sanal makineye node olarak belirlediğimiz sanal makinemizi tanıttık.

```
volkanub2@knode: ~  
File Edit View Search Terminal Help  
volkanub2@knode:~$ sudo nano /etc/hosts
```

Şekil 7.2.58.

```
volkanub2@knode: ~  
File Edit View Search Terminal Help  
GNU nano 2.9.3 /etc/hosts  
127.0.0.1    localhost  
127.0.1.1    volkanub2-VirtualBox  
172.20.10.3   knode  
# The following lines are desirable for IPv6 capable hosts  
::1         ip6-localhost ip6-loopback  
fe00::0     ip6-localnet  
ff00::0     ip6-mcastprefix  
ff02::1     ip6-allnodes  
ff02::2     ip6-allrouters
```

Şekil 7.2.59.



```

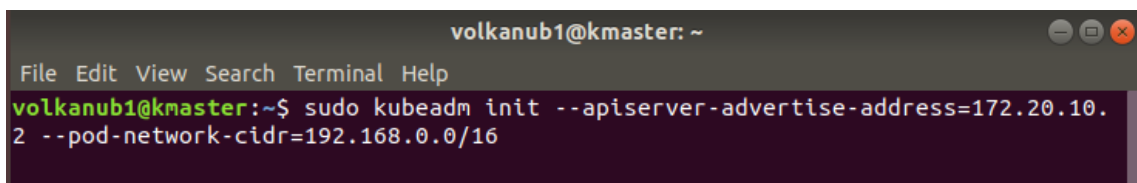
volkanub2@knode: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts

127.0.0.1    localhost
127.0.1.1    volkanub2-VirtualBox
172.20.10.2   kmaster
172.20.10.3   knode
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

```

Şekil 7.2.60.

Şekil 7.2.58. , Şekil 7.2.59. ve Şekil 7.2.60.'da node olarak belirlediğimiz sanal makineye ana makine olarak belirlediğimiz sanal makinemizi tanıttık. Bu aşamdan sonra aksi belirtilene kadar yapılacak tüm adımlar ana sanal makinemizde gerçekleştirilecek.



```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ sudo kubeadm init --apiserver-advertise-address=172.20.10.2 --pod-network-cidr=192.168.0.0/16

```

Şekil 7.2.61.

Şekil 7.2.61'de kubernetes kümemizi çalıştırmak için gerekli komutu girdik.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
[bootstrap-token] configured RBAC rules to allow certificate rotation for all n
ode client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" na
mespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotata
ble kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each a
s root:

kubeadm join 172.20.10.2:6443 --token ursr7j.vbdpn3bohpfj7ydo8 \
  --discovery-token-ca-cert-hash sha256:b1560ebefd5eb9a9b33eadf8a7d88c22ceb1f
1c06086f90a46a233c889fa0e9e
volkanub1@kmaster:~$

```

Şekil 7.2.62.

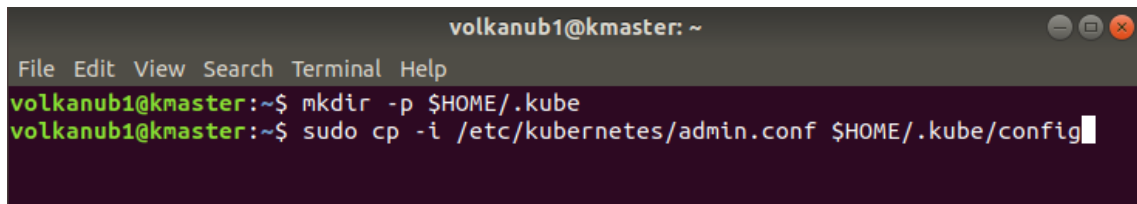
Şekil 7.2.62’de Şekil 7.2.61’deki komut sonrası elde ettiğimiz bilgileri görüyoruz. Kümemizi çalıştırmak için gerekli komutlar, pod network dağıtımı için gerekli olan komutu ve node’larımızın kubernetes kümemize katılmak için gerekli olan komutumuzu görüyoruz. “Kubeadm join” ile başlayan katılım komutumuzu not alıyoruz. Böylelikle node’umuza yazmak için gerektiğinde erişebilelim.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ mkdir -p $HOME/.kube

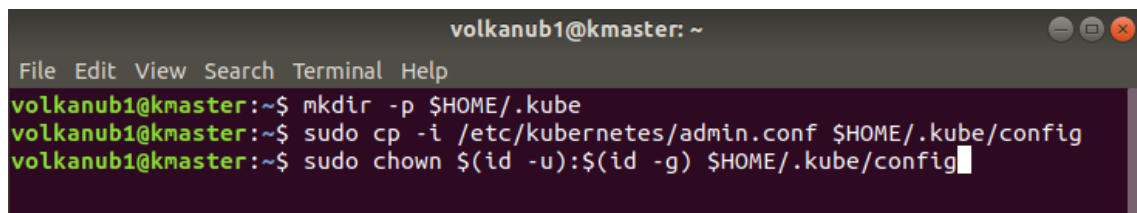
```

Şekil 7.2.63.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ mkdir -p $HOME/.kube  
volkanub1@kmaster:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

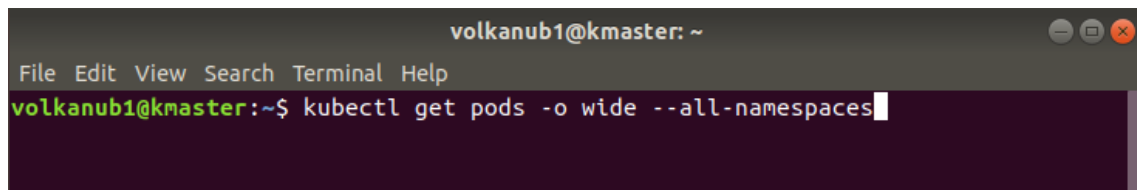
Şekil 7.2.64.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ mkdir -p $HOME/.kube  
volkanub1@kmaster:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
volkanub1@kmaster:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Şekil 7.2.65.

Şekil 7.2.63. , Şekil 7.2.64. ve Şekil 7.2.65.'de kubernetes kümemizi kullanmaya başlamak için gerekli olan komutları yazdığımızı görüyoruz.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ kubectl get pods -o wide --all-namespaces
```

Şekil 7.2.66.

```

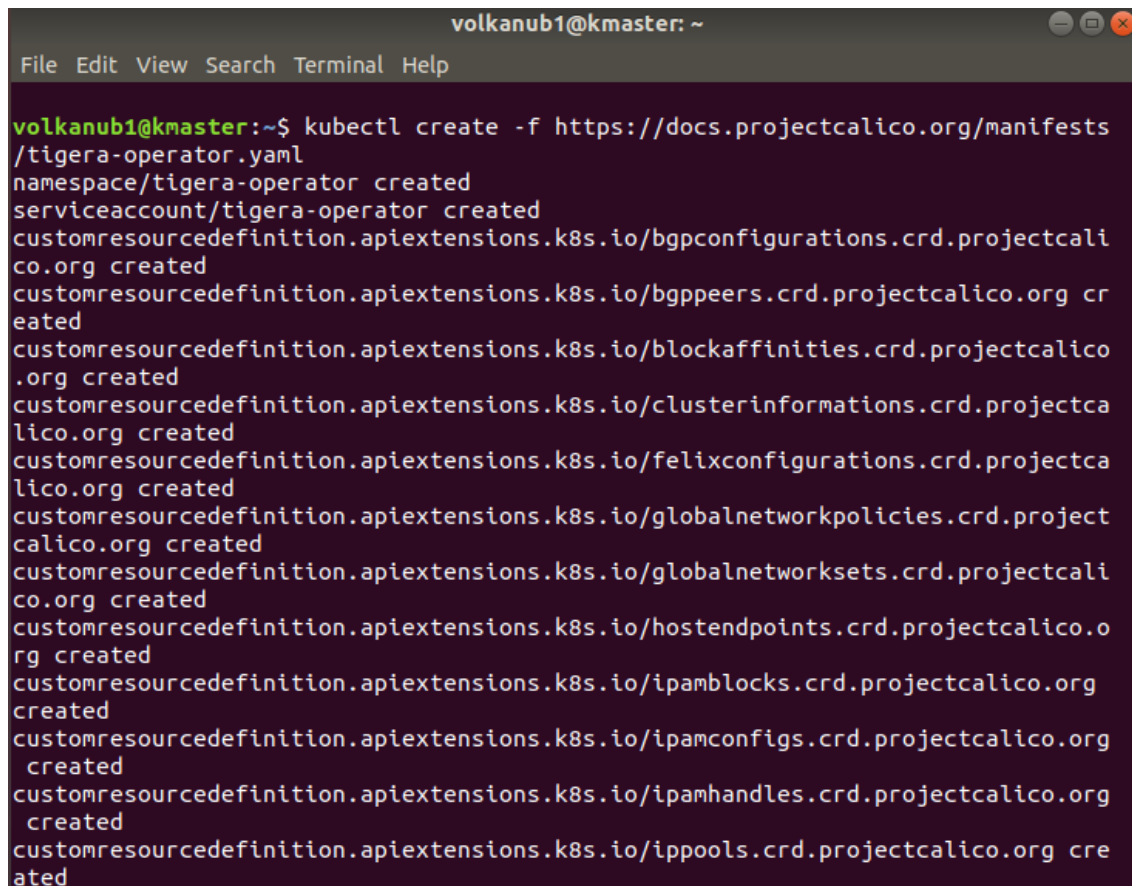
volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl get pods -o wide --all-namespaces

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-66bff467f8-f6blk	0/1	Pending	0	11
m	<none>	<none>			
kube-system	coredns-66bff467f8-kqscx	0/1	Pending	0	11
m	<none>	<none>			
kube-system	etcd-kmaster	1/1	Running	0	11
m	172.20.10.2 kmaster	<none>			
kube-system	kube-apiserver-kmaster	1/1	Running	0	11
m	172.20.10.2 kmaster	<none>			
kube-system	kube-controller-manager-kmaster	1/1	Running	0	11
m	172.20.10.2 kmaster	<none>			
kube-system	kube-proxy-6lbgm	1/1	Running	0	11
m	172.20.10.2 kmaster	<none>			
kube-system	kube-scheduler-kmaster	1/1	Running	0	11
m	172.20.10.2 kmaster	<none>			

Şekil 7.2.67.

Şekil 7.2.66. ve Şekil 7.2.67.'de kubectl üzerinden çalışan podları kontrol ediyoruz.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
  
volkanub1@kmaster:~$ kubectl create -f https://docs.projectcalico.org/manifests/  
/tigera-operator.yaml  
namespace/tigera-operator created  
serviceaccount/tigera-operator created  
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcali  
co.org created  
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org cr  
eated  
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico  
.org created  
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectca  
lico.org created  
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectca  
lico.org created  
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.project  
calico.org created  
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcali  
co.org created  
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.o  
rg created  
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org  
created  
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org  
created  
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org  
created  
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org cre  
ated
```

Şekil 7.2.68.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/installations.operator.tigera.io created
customresourcedefinition.apiextensions.k8s.io/tigerastatuses.operator.tigera.io created
clusterrole.rbac.authorization.k8s.io/tigera-operator created
clusterrolebinding.rbac.authorization.k8s.io/tigera-operator created
deployment.apps/tigera-operator created
volkanub1@kmaster:~$

```

Şekil 7.2.69.

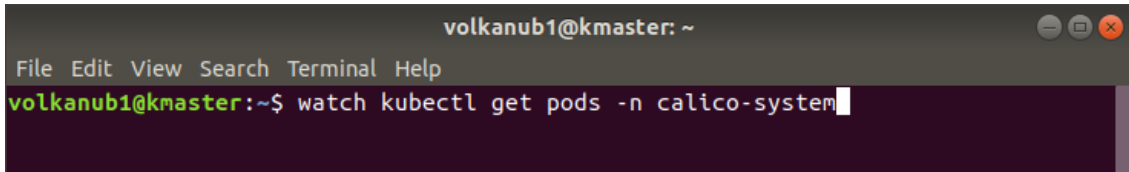
```

volkanub1@kmaster:~$ kubectl create -f https://docs.projectcalico.org/manifests/custom-resources.yaml
installation.operator.tigera.io/default created

```

Şekil 7.2.70.

Şekil 7.2.68. , Şekil 7.2.69. ve Şekil 7.2.70’de Calico Pod Network’ü yüklemek için gerekli komutları yazıyoruz.

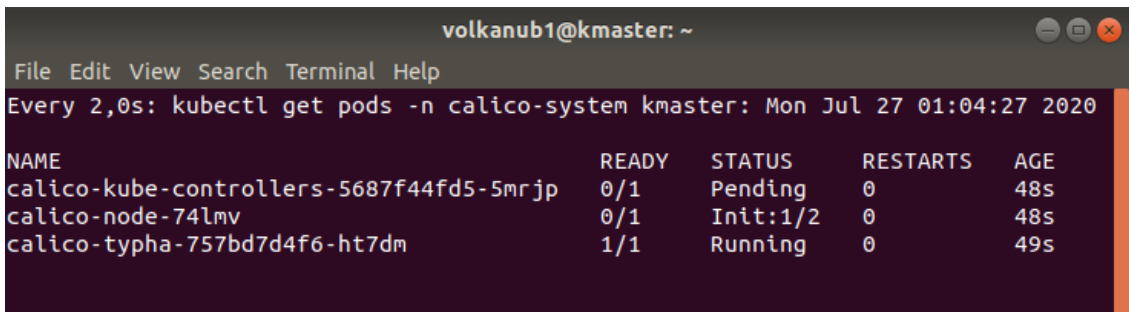


```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ watch kubectl get pods -n calico-system

```

Şekil 7.2.71.



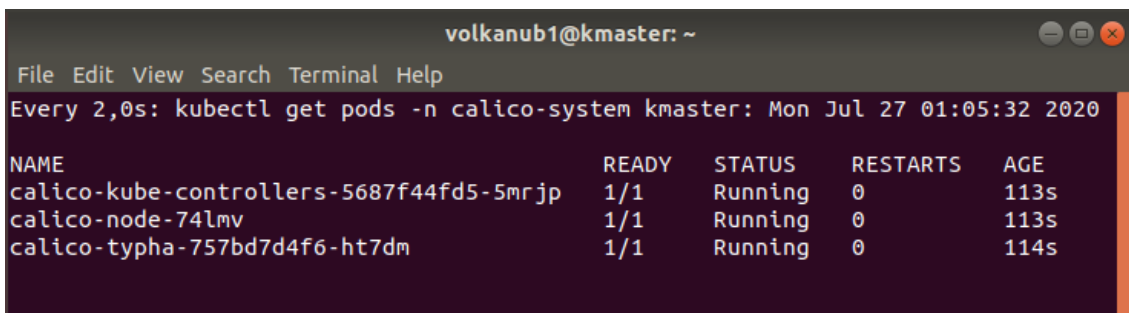
```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
Every 2,0s: kubectl get pods -n calico-system kmaster: Mon Jul 27 01:04:27 2020

```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-5687f44fd5-5mrjp	0/1	Pending	0	48s
calico-node-74lmv	0/1	Init:1/2	0	48s
calico-typha-757bd7d4f6-ht7dm	1/1	Running	0	49s

Şekil 7.2.72.



```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
Every 2,0s: kubectl get pods -n calico-system kmaster: Mon Jul 27 01:05:32 2020

```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-5687f44fd5-5mrjp	1/1	Running	0	113s
calico-node-74lmv	1/1	Running	0	113s
calico-typha-757bd7d4f6-ht7dm	1/1	Running	0	114s

Şekil 7.2.73.

Şekil 7.2.71. , ve Şekil 7.2.72. ve Şekil 7.2.73'de yükleme sonrası calico pod network'ü kontrol ediyoruz.

```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl get pods -o wide --all-namespaces

```

Şekil 7.2.74.

```

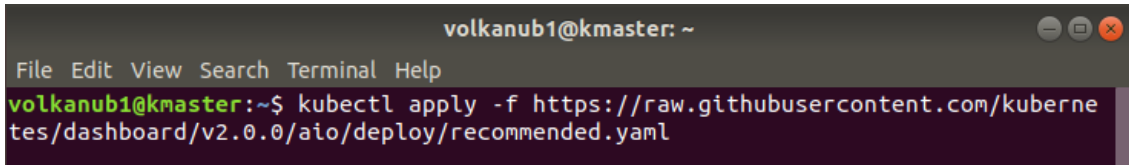
volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl get pods -o wide --all-namespaces

```

NAMESPACE	NAME	READY	STATUS
calico-system	calico-kube-controllers-5687f44fd5-5mrjp	1/1	Running
0	2m27s	192.168.189.1	kmaster
calico-system	calico-node-74lmv	1/1	Running
0	2m27s	172.20.10.2	kmaster
calico-system	calico-typha-757bd7d4f6-ht7dm	1/1	Running
0	2m28s	172.20.10.2	kmaster
kube-system	coredns-66bff467f8-f6blk	1/1	Running
0	19m	192.168.189.3	kmaster
kube-system	coredns-66bff467f8-kqscx	1/1	Running
0	19m	192.168.189.2	kmaster
kube-system	etcd-kmaster	1/1	Running
0	19m	172.20.10.2	kmaster
kube-system	kube-apiserver-kmaster	1/1	Running
0	19m	172.20.10.2	kmaster
kube-system	kube-controller-manager-kmaster	1/1	Running
0	19m	172.20.10.2	kmaster
kube-system	kube-proxy-6lbjm	1/1	Running
0	19m	172.20.10.2	kmaster
kube-system	kube-scheduler-kmaster	1/1	Running
0	19m	172.20.10.2	kmaster
tigera-operator	tigera-operator-6659cdcd96-lpg4z	1/1	Running
0	3m	172.20.10.2	kmaster

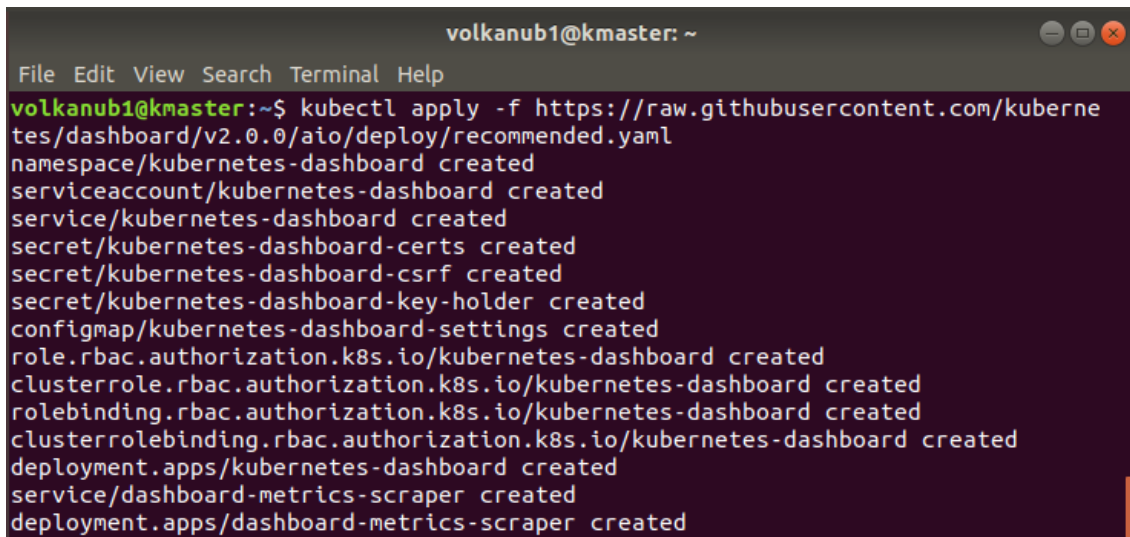
Şekil 7.2.75.

Şekil 7.2.74. ve Şekil 7.2.75’de çalışan tüm pod’larımızı kontrol ediyoruz.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

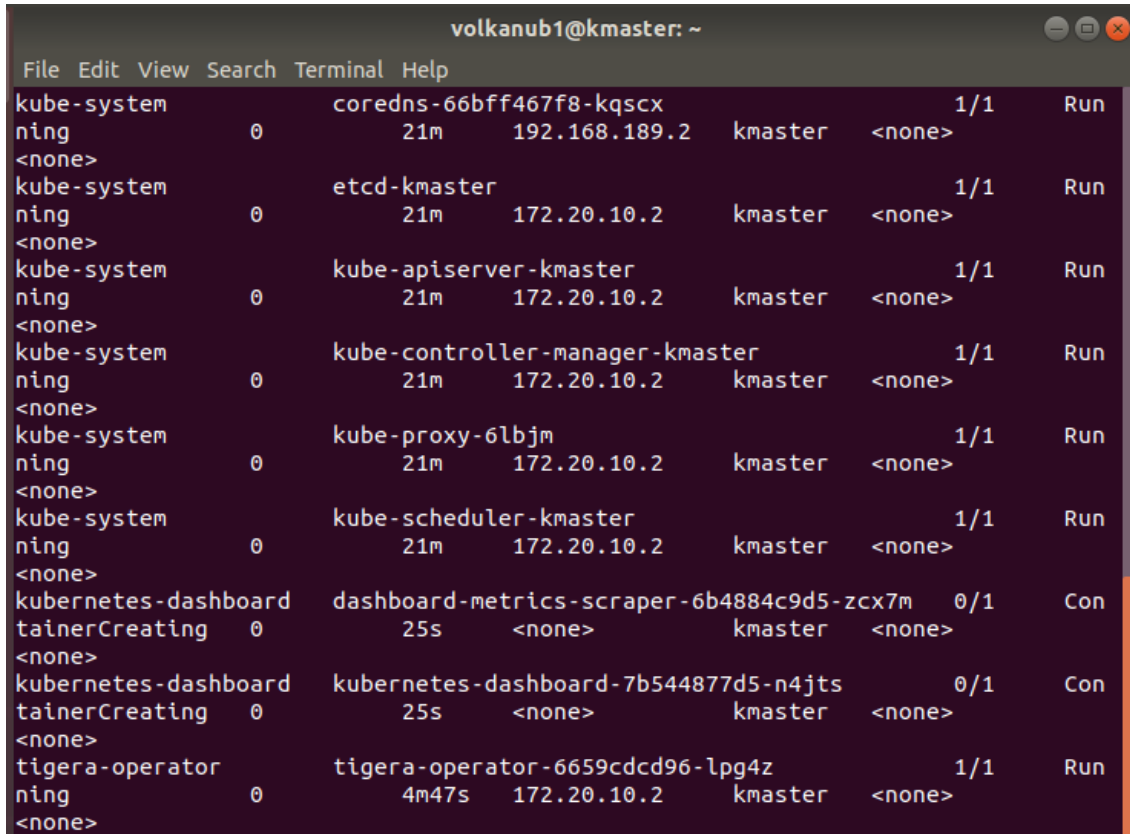
Şekil 7.2.76.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml  
namespace/kubernetes-dashboard created  
serviceaccount/kubernetes-dashboard created  
service/kubernetes-dashboard created  
secret/kubernetes-dashboard-certs created  
secret/kubernetes-dashboard-csrf created  
secret/kubernetes-dashboard-key-holder created  
configmap/kubernetes-dashboard-settings created  
role.rbac.authorization.k8s.io/kubernetes-dashboard created  
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created  
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  
deployment.apps/kubernetes-dashboard created  
service/dashboard-metrics-scraper created  
deployment.apps/dashboard-metrics-scraper created
```

Şekil 7.2.77.

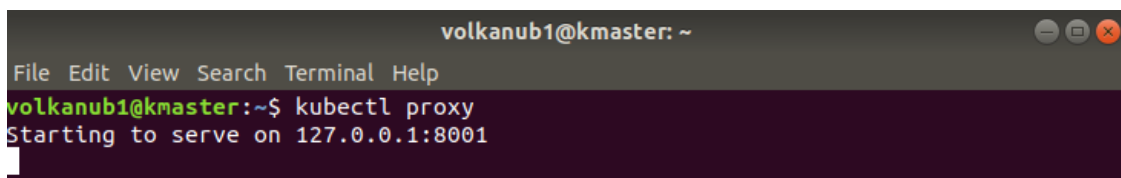
Şekil 7.2.76. ve Şekil 7.2.77’de kubernetes için gerekli dashboard’ı (kontrol paneli) yüklüyoruz.



Pod Name	Age	Image	IP Address	Node	Container Image	Ready	Status
kube-system	0	coredns-66bff467f8-kqscx	192.168.189.2	kmaster	<none>	1/1	Run
kube-system	0	etcd-kmaster	172.20.10.2	kmaster	<none>	1/1	Run
kube-system	0	kube-apiserver-kmaster	172.20.10.2	kmaster	<none>	1/1	Run
kube-system	0	kube-controller-manager-kmaster	172.20.10.2	kmaster	<none>	1/1	Run
kube-system	0	kube-proxy-6lbjm	172.20.10.2	kmaster	<none>	1/1	Run
kube-system	0	kube-scheduler-kmaster	172.20.10.2	kmaster	<none>	1/1	Run
kubernetes-dashboard	0	dashboard-metrics-scraper-6b4884c9d5-zcx7m	<none>	kmaster	<none>	0/1	Con
kubernetes-dashboard	0	kubernetes-dashboard-7b544877d5-n4jts	<none>	kmaster	<none>	0/1	Con
tigera-operator	0	tigera-operator-6659cdcd96-lpg4z	172.20.10.2	kmaster	<none>	1/1	Run

Şekil 7.2.78.

Şekil 7.2.78’de kubernetes dashboard’ımızın eklendiğini görebiliriz.



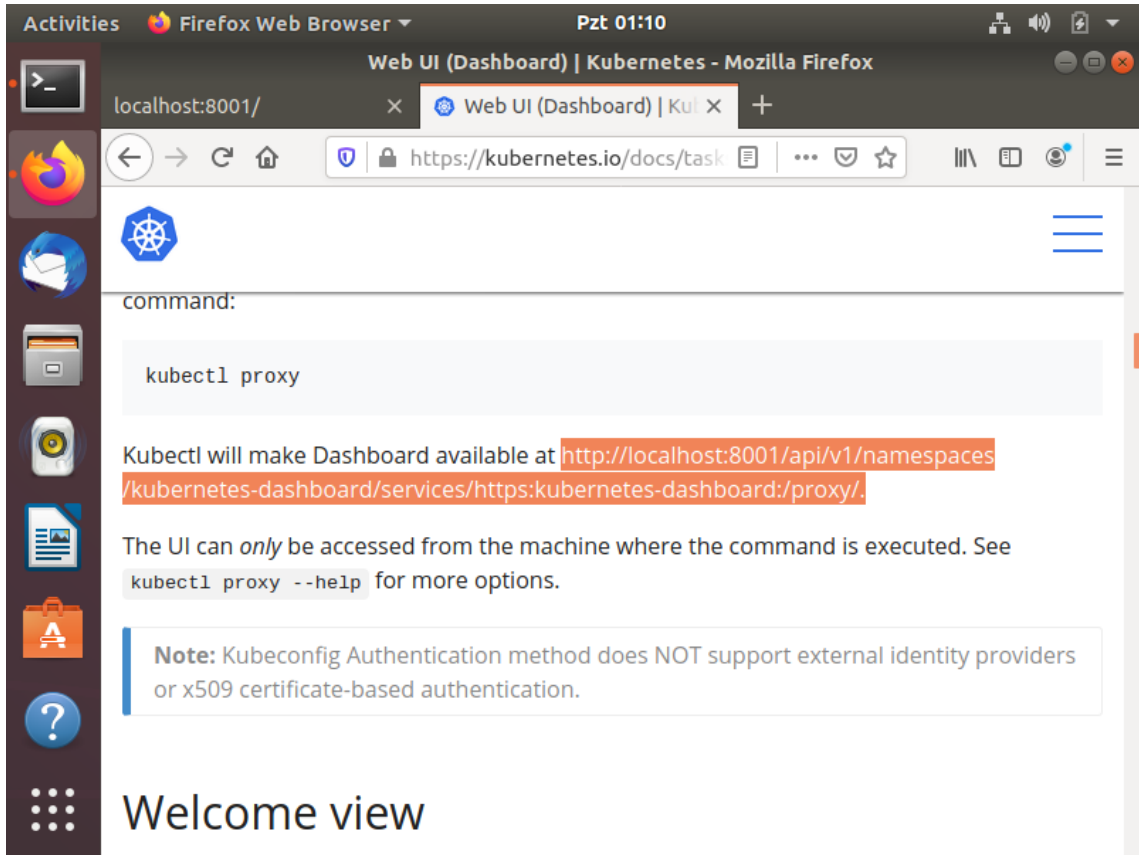
```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl proxy
Starting to serve on 127.0.0.1:8001

```

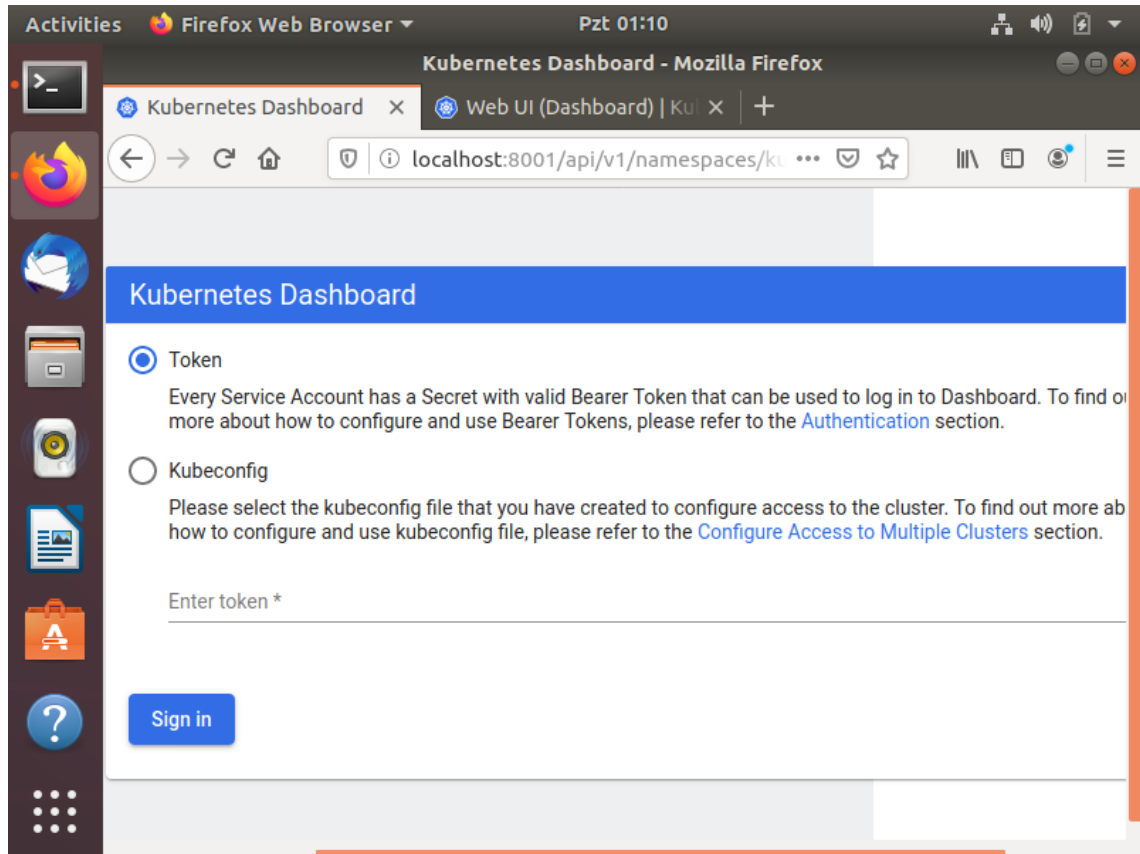
Şekil 7.2.79.

Şekil 7.2.79.’da kubectl proxy komutu ile kubernetes dashboard’ımıza erişebilir hale geldik. (Eğer terminalde bu komut çalışır olmaz ise dashboard’ımıza erişemeyiz.)



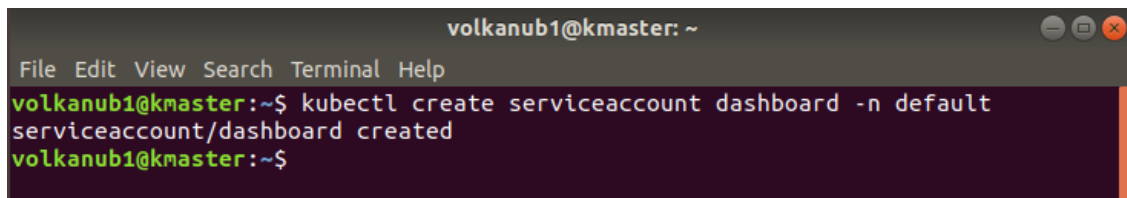
Şekil 7.2.80.

Şekil 7.2.80.'de <https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/> adresine girerek dashboard'a erişim linkini öğreniyoruz.



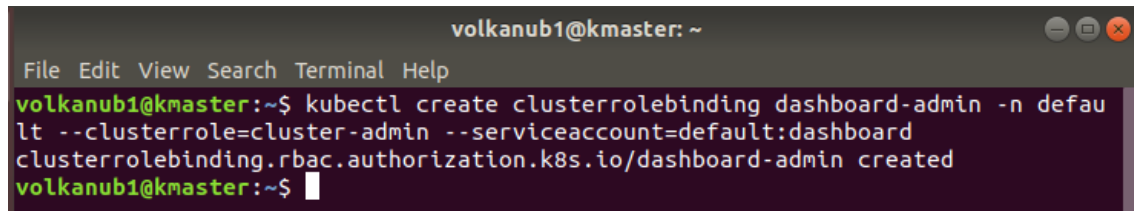
Şekil 7.2.81.

Şekil 7.2.81.'de `http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/`. Linkini kullanarak dashboard'ımıza erişiyoruz. Görüldüğü gibi dashboard'a giriş yapmamız için token veya kubeconfig kullanmalıyız.



Şekil 7.2.82.

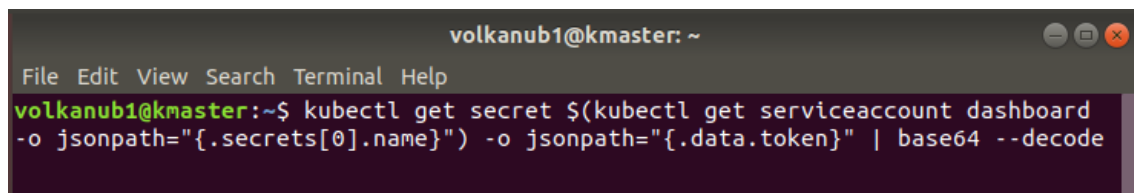
Şekil 7.2.82’de dashboard’ımızın default namespace’inde bir servis hesabı oluşturuyoruz.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ kubectl create clusterrolebinding dashboard-admin -n default --clusterrole=cluster-admin --serviceaccount=default:dashboard  
clusterrolebinding.rbac.authorization.k8s.io/dashboard-admin created  
volkanub1@kmaster:~$
```

Şekil 7.2.83.

Şekil 7.2.83.’de oluşturduğumuz servis hesabımıza cluster role olarak admin’i atıyoruz.



```
volkanub1@kmaster: ~  
File Edit View Search Terminal Help  
volkanub1@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
```

Şekil 7.2.84.

Şekil 7.2.84.’de dashboard’a giriş yapmak için gerekli token’ı elde eden komutu giriyoruz.

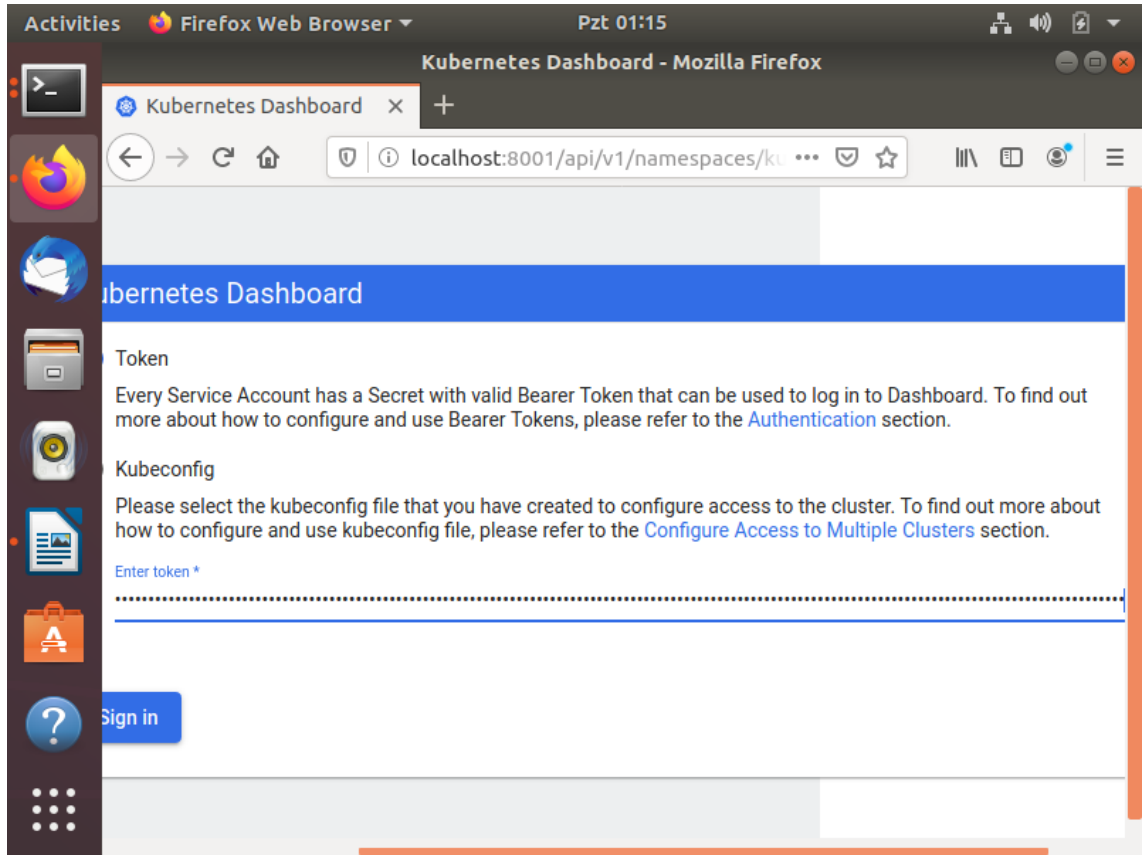
```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard
-o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6InFrakJUzkrVVEVCbVffWHIxb2p6b3JLRkJSTTByNG9RZW9YeUd
kTE1tTkUifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9
zZXJ2aWNlYWVjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYW
Njb3VudC9zZW5yZXQubmFtZSI6ImRhc2hib2FyZC10b2tlbi1mYnd4eiIsImt1YmVybmV0ZXMuaW8vc
2VydmljZWJjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkYXNoYm9hcmQiLCJrdWJlcm5ldGVz
LmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJjMDRjMzExMTAyLTRjNjg
tYTAwZi0wZTkzMTRjZGVlMmQiLCJzdWIiOiJzeXN0ZW06c2VydmljZWJjY291bnQ6ZGVmYXVsdDpkYX
NoYm9hcmQifQ.gq15jxSvUF7_YNz7pFeRl8lwgcEBz_hSzHZtuIbmcz8TUWAfuj5fEE0Q_ulvLK73ML
P2g9AYxy_jyWgBZoIY9guc5zMdmlv6ZwaVwGu8fuAIZRdCnXKBSGVUi9kwJX9BxJXVlhIkX328mJkow
Z05W8TPVZ1b8D-JEwcvafvvjPqCNUlKtpPOx-vKPU1Exm_t3GGQ05NW9RgWikzF0oJbVP2MHw0eF3-g
aqo4_uYwXm2as48CpHPPQ8hL9Yn0kYjBbeJ8SbI0C2DLKLIQYN3Yw7Bwru3KVV9QfNDxcIdXetL5IpV
R8SgCywgtELsgRtleX-oc0WkCwXmoggnzmpazSAvolkanub1@kmaster:~$

```

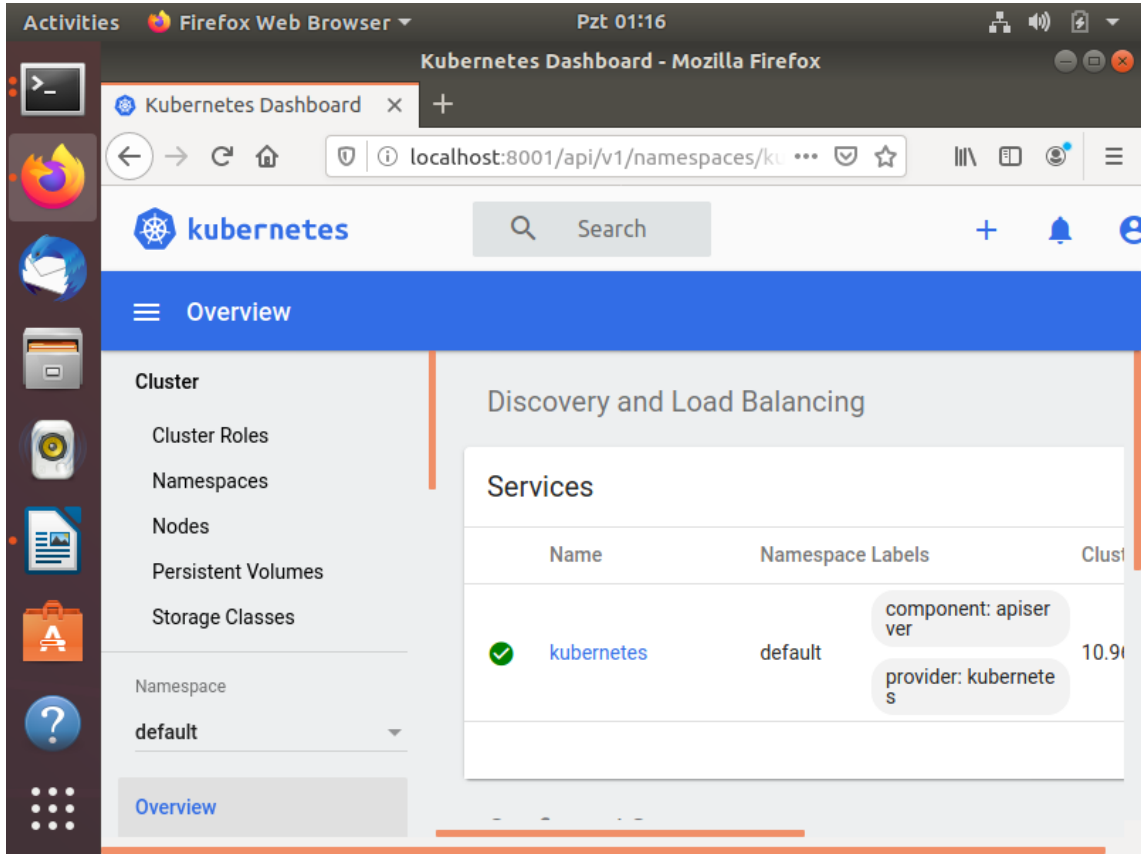
Şekil 7.2.85.

Şekil 7.2.85.'de token'ımızı görüyoruz.



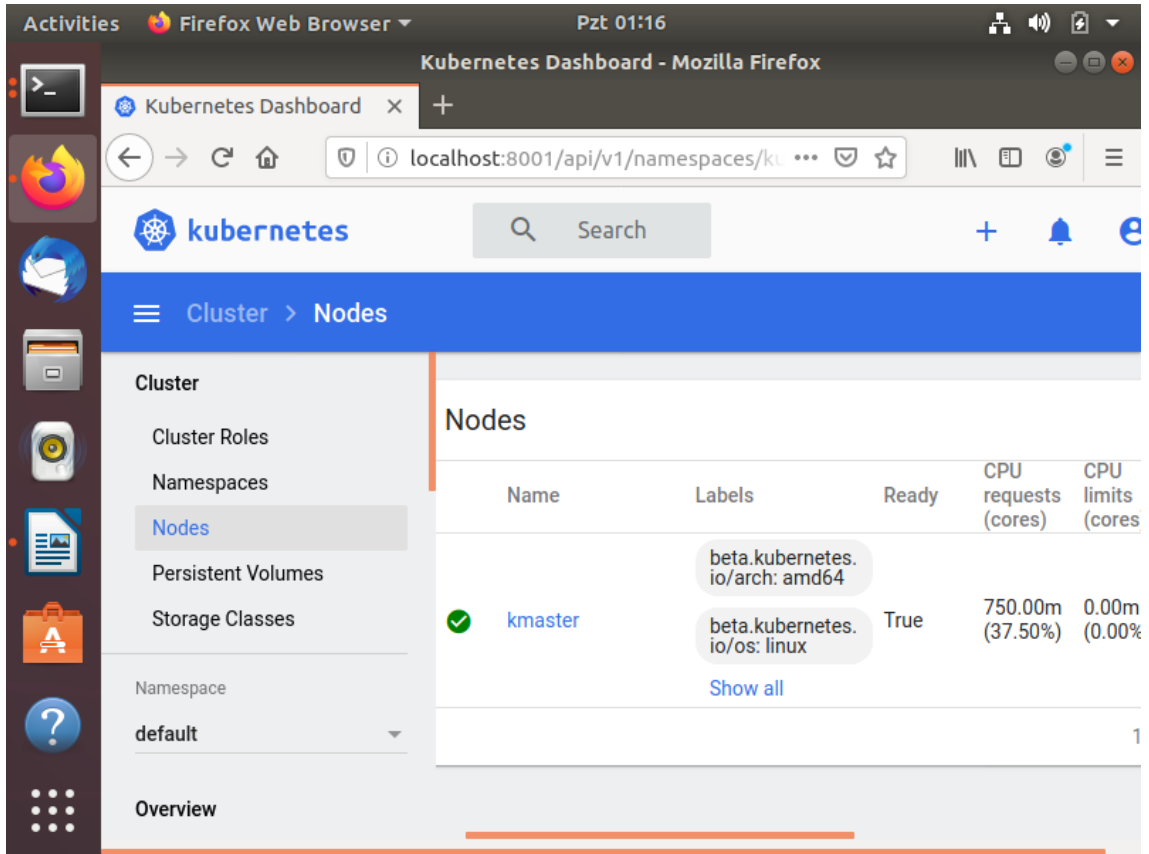
Şekil 7.2.86.

Şekil 7.2.86.'da token'ımızı girerek dashboard'a erişiyoruz.



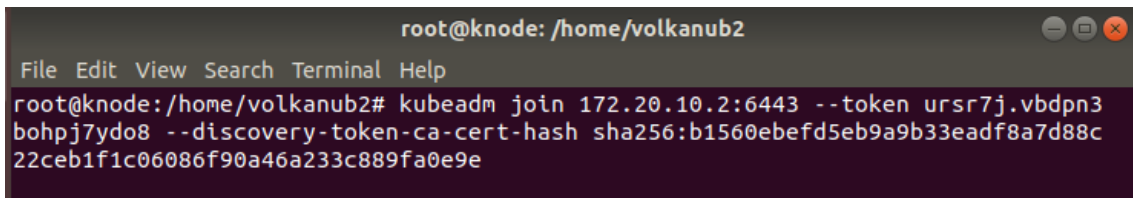
Şekil 7.2.87.

Şekil 7.2.87.'de kubernetes servisimizin aktif olduğunu görüyoruz.



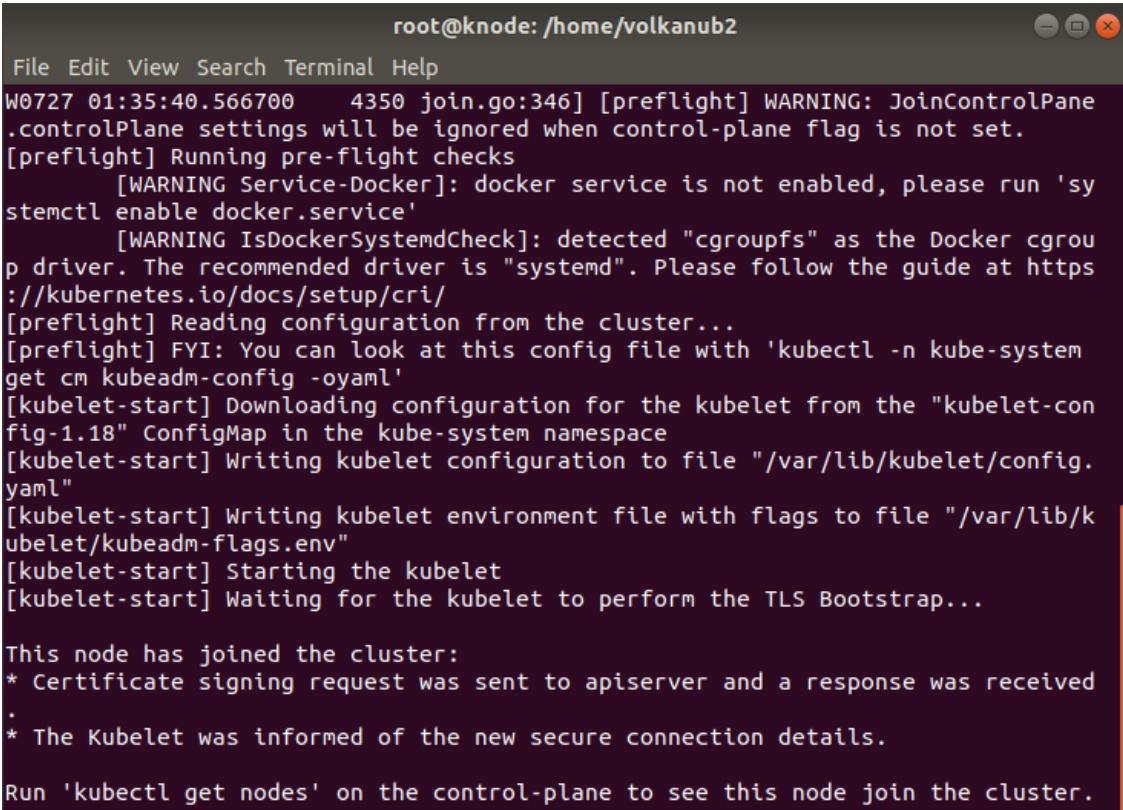
Şekil 7.2.88.

Şekil 7.2.88.'de sadece kmaster olarak belirlediğimiz ana makinemizin cluster'ımızda olduğunu görüyoruz.



Şekil 7.2.89.

Şekil 7.2.89.'daki işlemimizi node olarak belirlediğimiz sanal makinede gerçekleştiriyoruz. Böylece cluster'ımıza katılacak.



```

root@knode: /home/volkanub2
File Edit View Search Terminal Help
W0727 01:35:40.566700 4350 join.go:346] [preflight] WARNING: JoinControlPlane
.controlPlane settings will be ignored when control-plane flag is not set.
[preflight] Running pre-flight checks
[WARNING Service-Docker]: docker service is not enabled, please run 'sy
stemctl enable docker.service'
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgrou
p driver. The recommended driver is "systemd". Please follow the guide at https
://kubernetes.io/docs/setup/cri/
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-con
fig-1.18" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.
yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/k
ubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiservert and a response was received
.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

```

Şekil 7.2.90.

Şekil 7.2.90.'da Şekil 7.2.89'da gerçekleştirdiğimiz işlemin başarılı olduğunu yani node olarak belirlediğimiz makinenin kubernetes clusterımıza katılım sağlayabildiğini görüyoruz.

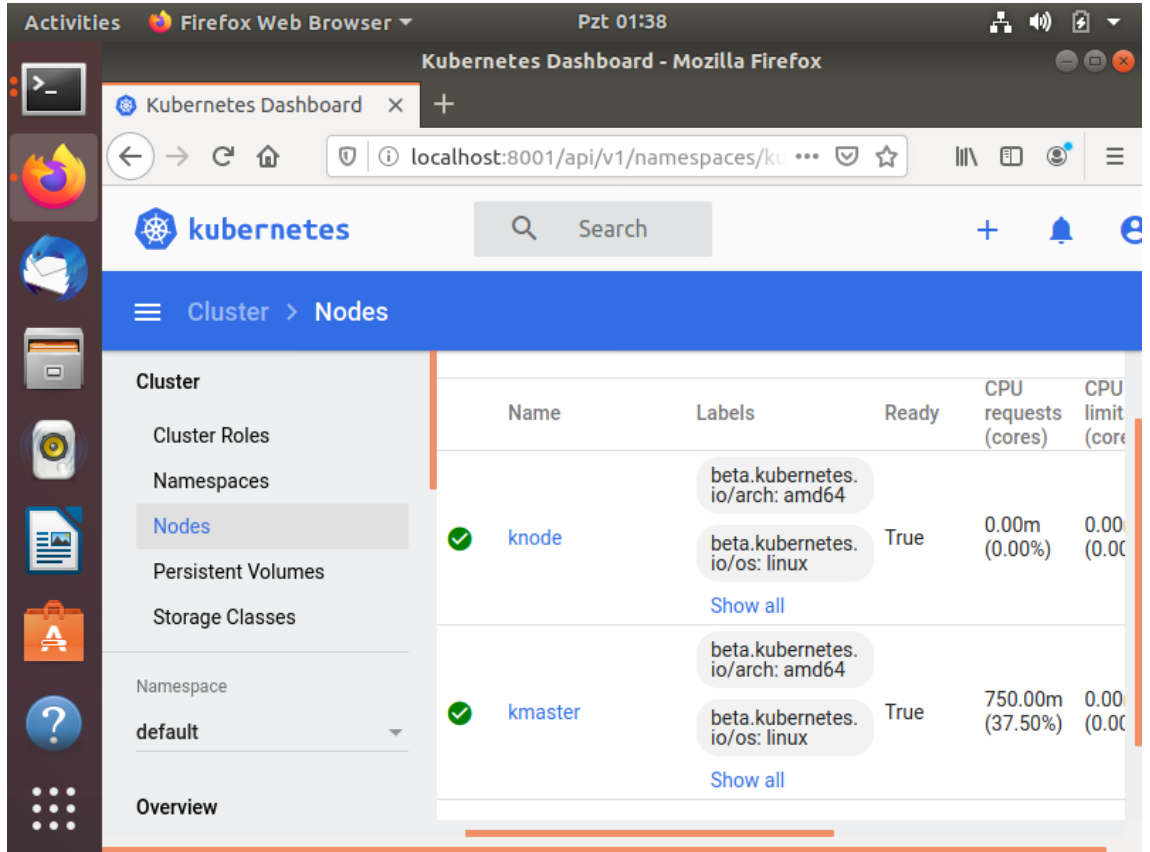
```

volkanub1@kmaster: ~
File Edit View Search Terminal Help
volkanub1@kmaster:~$ kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
kmaster     Ready     master   50m   v1.18.6
knode       NotReady  <none>    52s   v1.18.6
volkanub1@kmaster:~$

```

Şekil 7.2.91.

Şekil 7.2.91.'de ana makinemizden node'ları kontrol komutunu yazdığımızda node olarak belirlediğimiz makinenin görüldüğü teyit etmiş oluyoruz.



Şekil 7.2.92.

Şekil 7.2.92.'de dashboard'ımızdan knode'un da katıldığını görüyoruz. Böylelikle bulut tabanlı bir sistem örneği oluşturmuş oluyoruz.

8.SONUÇLAR

8.1. Sonuçlar

Bulut tabanlı sistemlerin gerek maliyetler, gerekse esneklik açısından faydaları göz önünde bulundurulduğunda fiziksel çözümler yerine seçmenin birçok zaman daha mantıklı olduğu göz önüne alınmalıdır. Son yıllarda konteyner orkestrasyon araçlarının kullanımının yaygınlaştığı ve bu orkestrasyon araçlarının kullanmayı bilmemin birçok avantaj sağlayacağı aşıkardır.Bu yüzden bu alanlarda hem çalışmalar yapmak, hem de yatırım yapmak avantajlı gözükmemektedir.

KAYNAKLAR

- [1] “Bulut bilişim nedir? Başlangıç klavuzu” 2020. [Online]. Bu adreste erişilebilir: <https://azure.microsoft.com/tr-tr/overview/what-is-cloud-computing/> [Erişim zamanı: 29.07.2020]
- [2] <https://kubernetes.io/docs/home/> 2020. [Online]. [Erişim zamanı: 29.07.2020]
- [3] “Sanallaştırma Nedir” 2020. [Online] Bu adreste erişilebilir: <https://azure.microsoft.com/tr-tr/overview/what-is-virtualization/> [Erişim zamanı: 29.07.2020]
- [4] “Sanallaştırma (Virtualization) Nedir? Türleri Nelerdir?” 2017. [Online] Bu adreste erişilebilir: <https://radore.com/blog/sanallastirma-virtualization-nedir-turleri-nelerdir.html> [Erişim zamanı: 29.07.2020]
- [5] Emre ÖZKAN, “Kubernetes nedir?” 2018. [Online] Bu adreste erişilebilir: <https://medium.com/devopsturkiye/kubernetes-nedir-eb5c5d149e69> [Erişim zamanı: 29.07.2020]
- [6] Ahmet Emre ALADAĞ, “Docker Nedir?” 2015. [Online]. Bu adreste erişilebilir: <https://www.emrealadag.com/docker-nedir.html> [Erişim zamanı: 29.07.2020]
- [7] “Konteyner Orkestrasyonu” 2020. [Online]. Bu adreste erişilebilir: <https://www.consulta.com.tr/tr/yonetim-danismanligi/konteyner-orkestrasyonu/42/> [Erişim zamanı: 29.07.2020]
- [8] “Docker (Konteyner) Nedir? Docker Güvenliği Nasıl Sağlanır? 2018. [Online]. Bu adreste erişilebilir: <https://www.bgasecurity.com/2018/04/docker-konteyner-nedir-docker-guvenligi-nasil-saglanir/> [Erişim zamanı: 29.07.2020]
- [9] Hemant SHARMA, “How To Install Kubernetes Cluster On Ubuntu 16.04” 2019. [Online]. Bu adreste erişilebilir: <https://www.edureka.co/blog/install-kubernetes-on-ubuntu> [Erişim zamanı: 29.07.2020]