BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SEMİNER DERSİ PROJESİ

Dockerfile ile Node.js Uygulaması Çalıştırma

Sevgi Nur ÖKSÜZ

2023-2024 BAHAR DÖNEMİ

ÖNSÖZ

Bu kılavuzda Bursa Teknik Üniversitesi Bilgisayar Mühendisliği 3. Sınıf Bahar Dönemi Seminer dersi kapsamında, Docker ile uygulama geliştirmenin tanıtılması, Node.js ve Docker hakkında bilgi verilmesi amacıyla hazırlanmıştır. Docker temelleri, Node.js ile yeni proje oluşturulması, projenin hazırlıkları ve çalıştırılması dahil olmak üzere tüm süreç adım adım açıklanmaktadır.

Adımların gerçekleştirilebilmesi için internet bağlantısı olan bir cihaza sahip olunması gerekmektedir.

Projeyi hazırlarken destek olan herkese teşekkür ederim.

Sevgi Nur ÖKSÜZ
Bursa 2024

İÇİNDEKİLER

ÖZET

Bu proje, Docker kullanarak Node.js tabanlı bir web uygulamasının nasıl konteynerize edileceğini ve MongoDB ile Redis gibi hizmetlerle nasıl entegre edileceğini göstermektedir. Adım adım kurulum ve yapılandırma süreçleriyle, modern yazılım geliştirme tekniklerini öğrenmek isteyenler için kapsamlı bir rehber sunulmuştur.

Proje kapsamında:

Node.js kullanarak temel bir web sunucusu oluşturulmuş,

MongoDB ile veri tabanı entegrasyonu sağlanmış,

Redis ile önbellekleme mekanizması eklenmiş,

Tüm bileşenler Docker ve Docker Compose kullanılarak konteynerize edilmiştir.

Bu kılavuz, Dockerfile ve docker-compose.yml dosyalarının nasıl oluşturulacağını ve hizmetlerin nasıl başlatılacağını detaylandırmaktadır.

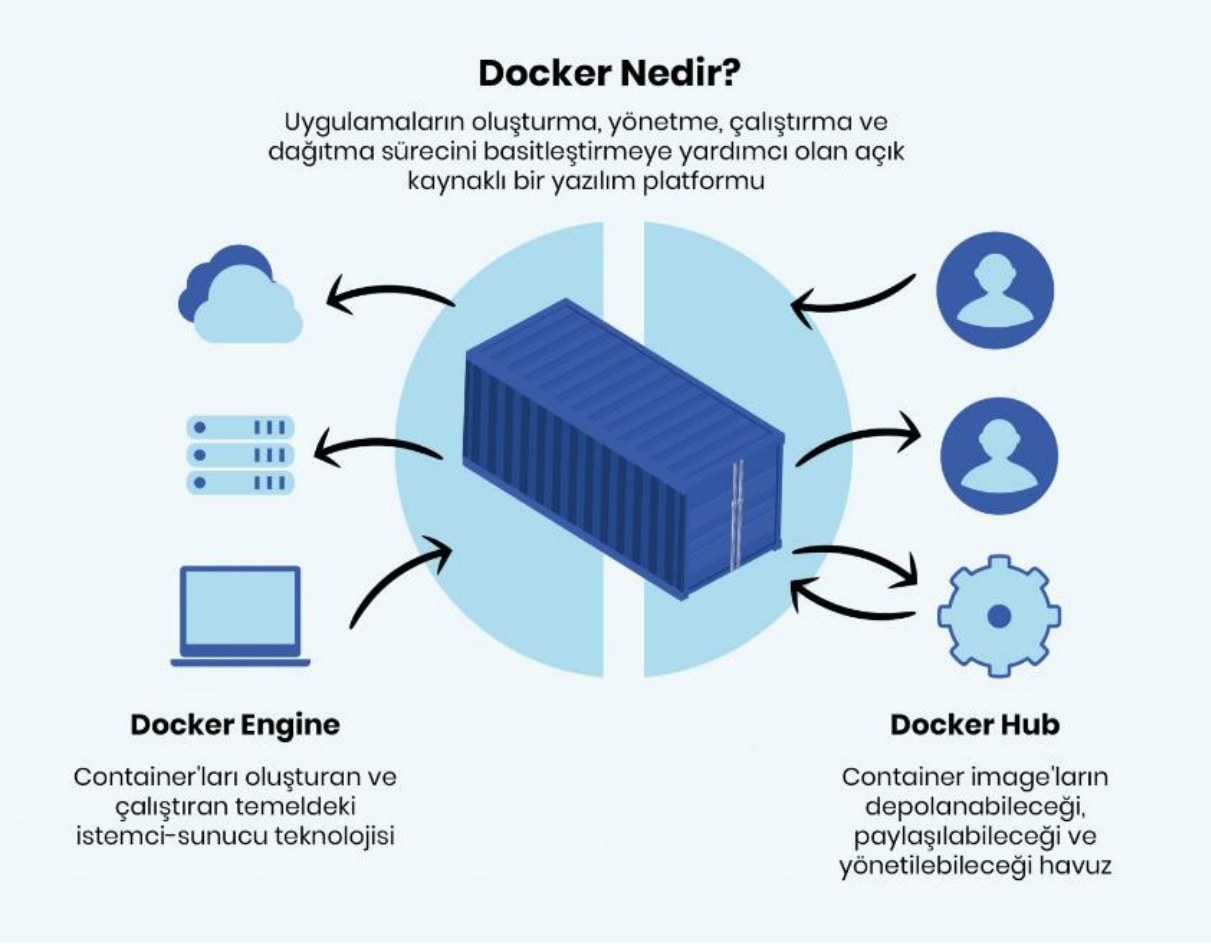
Sonuç olarak, mikro servis mimarisi ve konteyner teknolojileri hakkında temel bilgiler sunarak, geliştiricilere uygulamalarını daha taşınabilir ve yönetilebilir hale getirme konusunda yardımcı olmayı amaçlamaktadır.

1. GENEL BİLGİLER

1.1. DOCKER NEDİR?

Docker open source bir ‘konteyner’ teknolojisidir. Docker, aynı işletim sistemi üzerinde, yüzlerce hatta binlerce birbirinden izole ve bağımsız konteynerlar sayesinde sanallaştırma sağlayan bir teknolojidir. [1] Yüksek seviyeli API'ler için konteynerleri çalışan işlemlerden izole eder. Docker, LXC (Linux Containers) ve AUFS (Advanced Multi-Layered Unification Filesystem) teknolojilerini kullanır. LXC, Unix işlemlerini izole ederken, AUFS copy-on-write dosya sistemleri oluşturur. Docker, Linux Kernel, cgroups ve namespace bileşenlerinden faydalanır.[2]

Her uygulama aynı işletim sistemi üzerinde birbirinden bağımsız olarak kendi bağımlılıklarıyla çalışabilir ve uygulamalar kolaylıkla olduğu haliyle başka bir ortama taşınabilir. Özellikle geliştiricilerin yakındığı ve kendi makinelerinde çalışan kodun ya da uygulamanın sunucuda çalışmaması şeklindeki sorunların oluşmasının en başından önüne geçer. Aslında Docker geliştiricinin geliştirme şeklini ve uygulamalarımızı dağıtma yöntemlerimizi temelinden değiştirerek bizi mikroservis (microservice) mimarisine de hazır hale getirir [3].



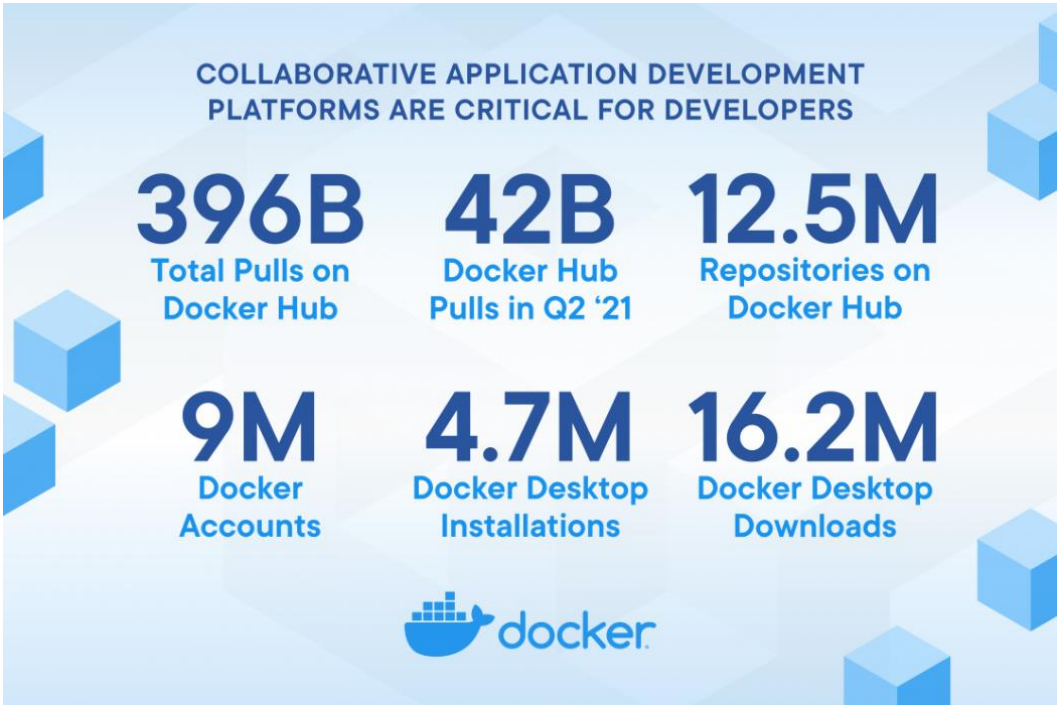
Şekil 1 Docker Nedir?

1.2. DOCKER TARİHÇESİ

Linux Kernel'e 2008 yılında eklenen Linux Containers (LXC) üzerine kurulu bir teknolojidir. Docker, 2013 yılında open-source bir proje olarak dotCloud isimli PaaS cloud hizmetleri sunan bir San Francisco firması tarafından tanıtıldı. Kurucusu Solomon Hykes' dir.[4]

Başlangıçta Docker, şirketin bulut işini binlerce sunucu üzerinde yürütmek için geliştirdiği teknolojinin doğal bir uzantısıydı. Google tarafından geliştirilen ve statik olarak yazılan bir programlama dili olan Go ile yazılan Docker, odak noktasını Docker ve Docker ekosisteminin geliştirilmesine kaydırdı. [5]

2021 Stack Overflow Developer Survey verilerine göre, Docker yazılımcılar arasında çok popüler. Git ve Kubernetes ile birlikte en sevilen ve en çok istenen araçlar arasında yer alıyor. JetBrains'in DevOps anketine göre, Docker kullanıcıları DevOps mühendisleri veya altyapı geliştiricileri olarak üç kat, mimar olarak iki kat ve takım lideri olarak %30 daha fazla bulunuyor. Ayrıca, Docker kullanıcıları daha kıdemli pozisyonlarda çalışmaya eğilimli. [6]



Şekil 2 Docker Kullanımı (2021)

1.3. DOCKER NE İÇİN KULLANILIR?

Docker, bir dizi faydalı özellik sunarak çeşitli amaçlar için kullanılmaktadır:

- **Kaynak Verimliliği:** Docker, daha az kaynak kullanarak birden çok iş yükünü çalıştırmanıza olanak tanır. Bu, sistem kaynaklarını verimli bir şekilde kullanarak daha fazla uygulama örneğini aynı anda çalıştırmanızı sağlar.

- **İzolasyon ve Ayırma**: Docker, uygulamaları birbirinden izole ederek ve ayırarak çalıştırır. Bu, her uygulamanın kendi bağımsız ortamını oluşturmasını sağlar, böylece bir uygulamanın diğerlerine olan etkisi en aza indirgenir.

- **Ortam Standartları:** Docker, geliştirme ve yayın döngüleri arasında tutarlılık sağlamak için ortamları standart hale getirir. Bu, geliştirme ekibinin her zaman aynı ortamı kullanmasını ve uygulamanın farklı aşamalarında tutarlı sonuçlar elde etmesini sağlar.

- **Geliştirme Kolaylığı:** Docker, geliştirme yaşam döngüsünü kolaylaştırır ve CI/CD (Continuous Integration/Continuous Deployment) iş akışlarını destekler. Geliştiriciler, Docker konteynerlerinde uygulamaları çalıştırarak yerel ortamda hızlı bir şekilde test edebilir ve dağıtım süreçlerini otomatikleştirebilirler.

- **Taşınabilirlik:** Docker konteynerleri, çoklu bulut platformlarında çalışabilen son derece taşınabilir iş yükleri oluşturmanıza olanak tanır. Bu, uygulamalarınızı istediğiniz herhangi bir ortamda kolayca dağıtmanızı ve çalıştırmanızı sağlar.

Docker ayrıca şu şekillerde de kullanılabilir:

- **Maliyet Verimliliği:** Sanal makinelerle karşılaştırıldığında daha uygun maliyetli bir alternatif sunar.

- **Sürüm Kontrolü:** Docker, bir uygulamanın farklı sürümlerini ve değişikliklerini yönetmek için bir sürüm kontrol sistemi olarak kullanılabilir. [8], [4]

1.4. DOCKER'IN ÖZELLİKLERİ VE AVANTAJLARI

Docker, uygulamaları hızlı ve kolay bir şekilde çalıştırabilme olanağı sunar. Bunun için, hazır imajları hub.docker.com'dan veya kendi imajlarımızı oluşturarak kullanabiliriz. Bu sayede, uygulamamız için gerekli olan framework bağımlılıklarını kurmak zorunda kalmayız.

Docker imajları, Docker Registry aracılığıyla kaydedilebilir. Büyük projelerde, temel imajları saklamak için Docker Local Registry kullanılabilir. Bu, uygulamanın geliştirilme ve dağıtım süreçlerini hızlandırır ve düzenler.

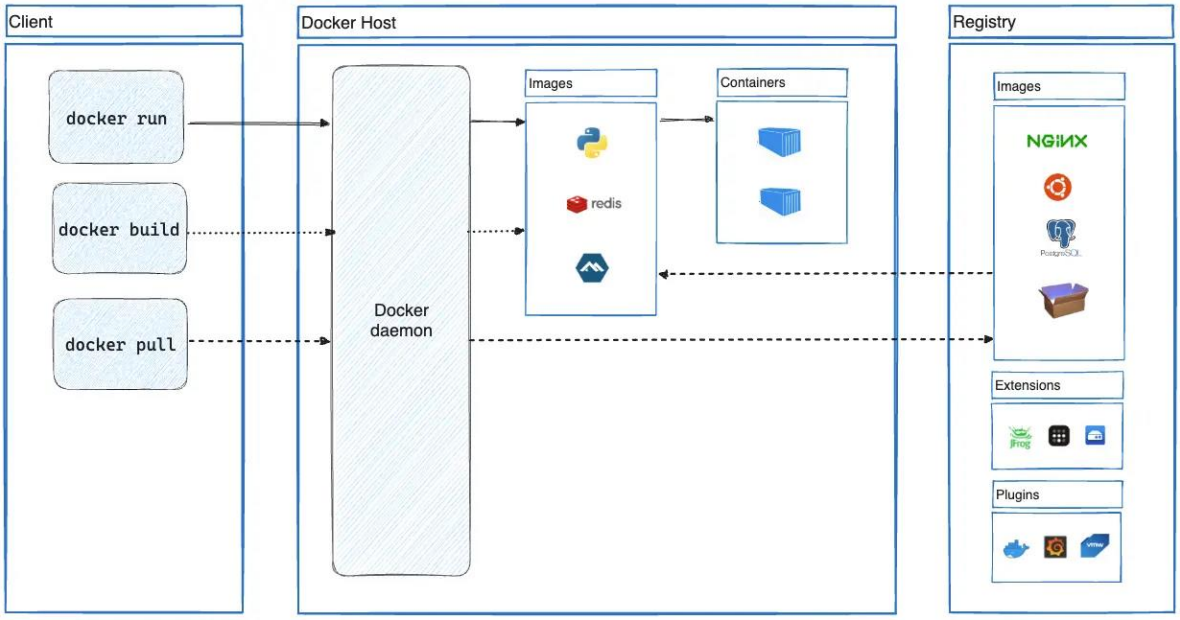
Docker ayrıca uygulamaların test edilmesini kolaylaştırır. Docker konteynerleri, çalışabilirliğin test edilmesi için ideal bir ortam sunar. Ayrıca, Docker Swarm veya Kubernetes gibi orkestrasyon araçlarıyla uygulamaların dağıtımı, ölçeklendirilmesi ve yönetimi sağlanabilir. Bu sayede, uygulama geliştirme süreci daha verimli ve düzenli hale gelir.

Docker'ın tek bir işletim sistemi üzerinde çalışması, kaynak kullanımını optimize ederken, işletim sisteminden bağımsız olması da uygulamaların farklı ortamlarda kolayca çalıştırılmasını sağlar. Ayrıca, Docker bulut servisleriyle de entegre bir şekilde çalışabilir, böylece uygulamaların bulut altyapılarıyla sorunsuz bir şekilde entegrasyonu mümkün olur. [4]

1.5. DOCKER MİMARİSİ

Docker, bir istemci-sunucu mimarisi kullanır. Docker istemcisi, Docker konteynerlerinizi oluştururken, çalıştırırken ve dağıtırken ağır işlemleri gerçekleştiren Docker sunucusuyla iletişim kurar.

Docker istemcisi ve sunucusu aynı sistemde çalışabilir veya bir Docker istemcisini uzak bir Docker sunucusuna bağlayabilirsiniz. Docker istemcisi ve sunucusu, UNIX soketleri veya bir ağ arabirimi üzerinden bir REST API kullanarak iletişim kurar. Docker Compose ise bir dizi konteynerden oluşan uygulamalarla çalışmanıza olanak tanır. [10]



Şekil 3 Docker Mimarisi

1.5.1. DOCKER DAEMON

Docker daemon (dockerd), Docker API isteklerini dinler ve görüntüler, konteynerler, ağlar ve birimler gibi Docker nesnelerini yönetir. Bir daemon, Docker hizmetlerini yönetmek için diğer daemonlarla da iletişim kurabilir. [10]

"dockerd", konteynerleri yöneten kalıcı süreçtir. Docker, daemon ve istemci için farklı ikili dosyalar kullanır. Daemon'u çalıştırmak için dockerd yazarsınız.[11]

1.5.2. DOCKER İSTEMCİSİ

Docker istemcisi (docker) çoğu Docker kullanıcısı için ana arayüzdür. "docker run" gibi komutlar aracılığıyla Docker ile etkileşimi kolaylaştırır ve bunlar daha sonra yürütülmek üzere "dockerd"a iletilir. Docker API'sini kullanan docker komutu birden fazla daemon ile etkileşime girebilir. [10]

Docker daemon'una komut satırı arayüzü (CLI) sağlar ve görüntülerin bir kayıt defterinden çekilmesini, bir Docker ana bilgisayarında çalıştırılmasını ve durdurulmasını yönlendirir.

Öne çıkan komutlar arasında "docker build", "docker pull" ve "docker run" bulunur.[12]

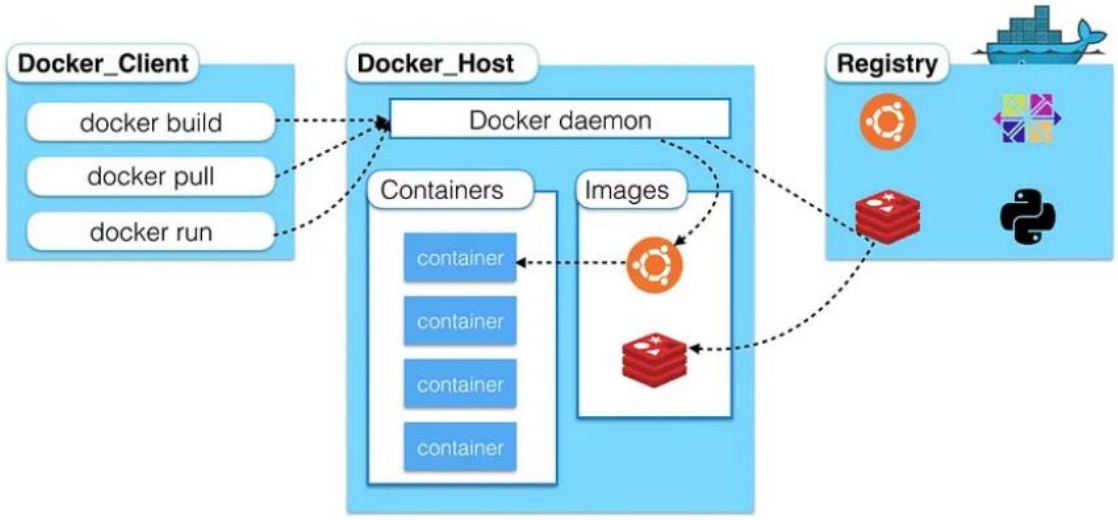
1.5.3. DOCKER DESKTOP

Docker Desktop, kolayca konteyner uygulamaları ve mikro servisler oluşturmanıza ve paylaşmanıza olanak sağlayan bir GUI (Grafik Kullanıcı Arayüzü) uygulamasıdır. Mac, Windows veya Linux ortamlarında çalışabilir. Docker Desktop, Docker arka plan programı (dockerd), Docker istemcisi (docker), Docker Compose, Docker Content Trust, Kubernetes ve Credential Helper gibi bileşenleri içerir. [13]

1.5.4. DOCKER REGISTRY

Docker registry'leri, Docker imajlarını depolar. Docker Hub, herkesin kullanabileceği genel bir registry'dir. Büyük şirketler ve bireyler, buradan imajlarını public olarak paylaşabilirler. Private olarak imaj saklamak için ücret talep edilir. Docker'ı kurduğunuzda, yükleme sırasında imajları bu siteden arar ve indirir. [13]

"docker pull" veya "docker run" komutlarını kullandığınızda, gerekli imajlar yapılandırılmış registry'lerden alınır. "docker push" komutunu kullandığınızda ise imaj, yapılandırılmış registry'e gönderilir. [10]



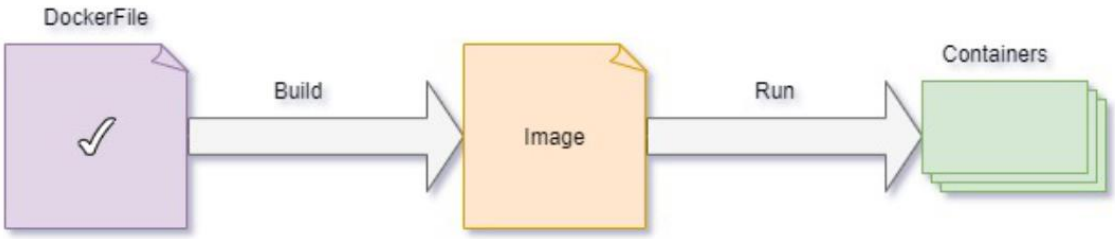
Şekil 4 Docker Registry

1.5.5. DOCKER NESNELERİ

Docker kullanırken, görüntüler (images), konteynerler (containers), ağlar (networks), birimler (volumes), eklentiler (plugins) ve diğer nesneler oluşturur ve kullanır. İşte bu nesnelerden bazılarına dair kısa bir genel bakış:

1.5.5.1. DOCKER İMAJI

Docker imajı, bir uygulamayı çalıştırmak için gerekli tüm bileşenleri içeren bir çalıştırılabilir yazılım paketidir. Bu imaj, bir konteynerin nasıl örneklenmesi gerektiğini belirleyerek hangi yazılım bileşenlerinin nasıl çalışacağını tanımlar. [14]

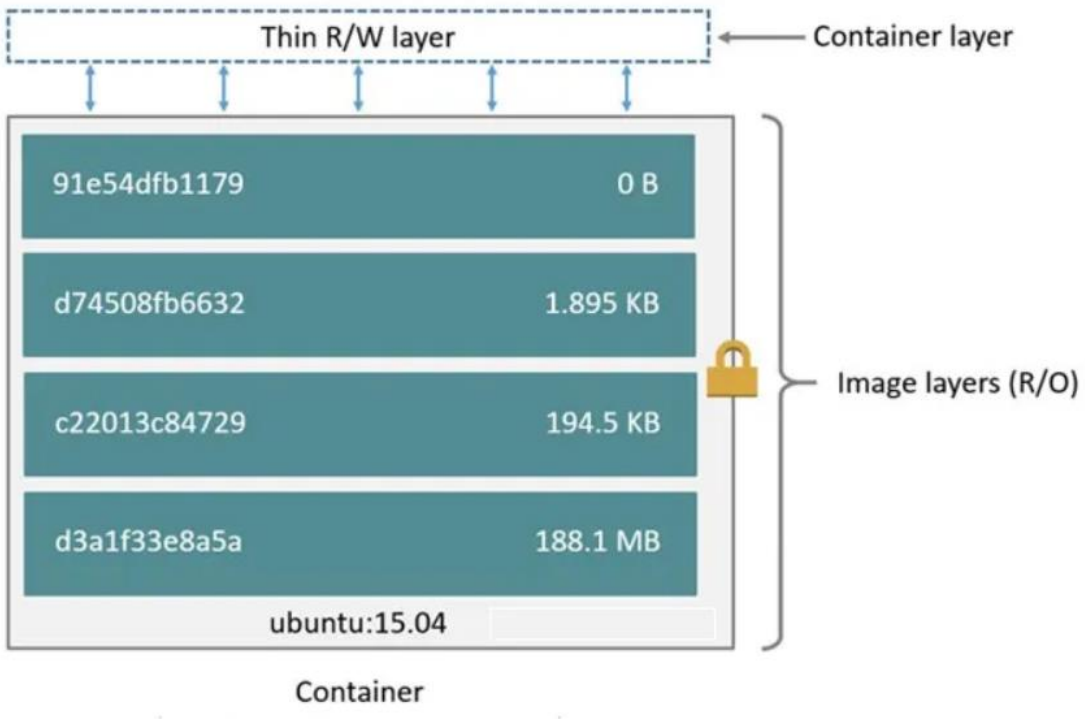


Şekil 5 Docker İmaj ve Konteyner Yapısı

1.5.5.2. DOCKER KONTEYNER

Docker Konteyner, imajların çalıştırılmış veya durdurulmuş halidir. Yazılımlarımızı paketleyip çalışma zamanı bağımlılıklarını yönetebildiğimiz, işletim sistemi sanallaştırmasını kullanmadan daha az kaynak tüketimi ile işlem izolasyonu sağlayan bir yapı parçacığıdır.

İndirdiğiniz imajı kullanarak yeni bir konteyner oluşturduğunuzda Docker, bu konteyner içinde okunabilir ve yazılabilir bir katman daha ekler. Bu katmana "intermediate image" de denir ve sadece okunabilir imaj katmanlarının üzerinde konteynerın değişiklik yapmasına olanak tanır. Konteyner silindiğinde, bu katman da konteynerle birlikte silinir. [15]

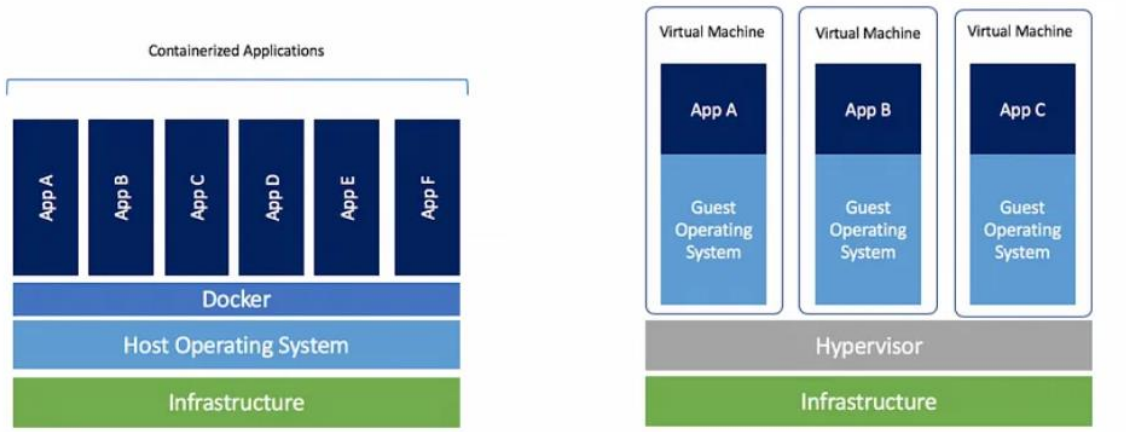


Şekil 6 Konteyner Yapısı

1.5.5.2.1. DOCKER KONTEYNER VE SANAL MAKİNE

Konteynerlar, aynı işletim sistemi veya sanal makine üzerinde çalışan diğer ortamlardan izole edilmiş yapılar olarak tanımlanır. Her bir konteyner, kendi içinde birçok süreci, servisi ve ağı barındırabilir. Bir konteyner, bir sanal makineye benzer, ancak bir konteyner işletim sistemi kernelini paylaşırken, sanal makine kendi işletim sistemine sahiptir.

Docker'ın kullandığı konteyner türü LXC'dir ve bunun yanında LXD ve LXCFS gibi konteyner türleri de bulunmaktadır. [16]



Şekil 7 Konteyner ve Sanal Makine Farkı

1.5.5.3. DOCKERFILE

Bir dizi komut içeren basit bir metin dosyasıdır. Bu komutlar / talimatlar, yeni bir docker imajı oluşturmak için temel imaj üzerinde eylemler gerçekleştirmek üzere art arda yürütülür.

Docker imajları oluşturmanıza yardımcı olacaktır.

Docker dosyasında bulunan her bir satır, docker imajının bir katmanını temsil eder. [18]

Dosya ismi kesinlikle **Dockerfile** olmalıdır ve herhangi bir **uzantısı yoktur**.

Dockerfile içerisinde hangi imajın kullanılacağı, hangi dosyaları içereceği ve hangi uygulamanın hangi parametrelerle çalışacağı yazılır. [4]

1.5.5.4. DOCKER COMPOSE

Docker Compose, birden fazla konteyner içeren Docker uygulamalarını tanımlamak ve çalıştırmak için kullanılır. Uygulamanızın servislerini yapılandırmak için bir YAML dosyası kullanır ve tek bir komutla bu servisleri oluşturup başlatabilirsiniz.[1]

Compose, production, staging, development, testing ve diğer CI iş akışları gibi tüm ortamlarda çalışır**.** [1]

Konteynerleri birlikte çalıştırmak ve konuşlandırmak için daha az kod yazmanıza olanak tanır ve birden çok konteyner arasındaki bağımlılıkları yönetir. Örneğin, bir web uygulaması için web sunucusu, veritabanı sunucusu ve önbellek sunucusu gibi konteynerleri koordine eder.[20]

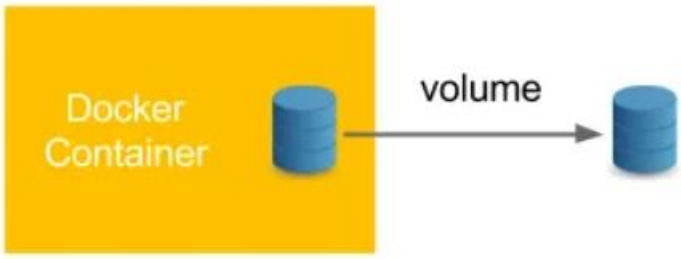
Şekil 8 Docker Compose Sembolik Gösterim

1.5.5.5. DOCKER VOLUME

Docker volume, konteyner verilerinin kalıcılığını ve paylaşımını sağlar.

Konteynerin konteyner verilerini ana bilgisayar birimlerine yazmasını sağlayan konteynerin içine monte edilmiş Docker ana bilgisayarının dizinidir. [18]

Mekanizma aşağıdaki şemada gösterilmiştir:



Şekil 9 Docker Volume Mekanizması

1.5.5.6. DOCKER NETWORK

Docker ağı, bir kullanıcının bir Docker konteynerini istediğimiz kadar ağa bağlamasına olanak tanır.

Docker Ağları, Docker konteynerleri için tam izolasyon sağlamak için kullanılır.

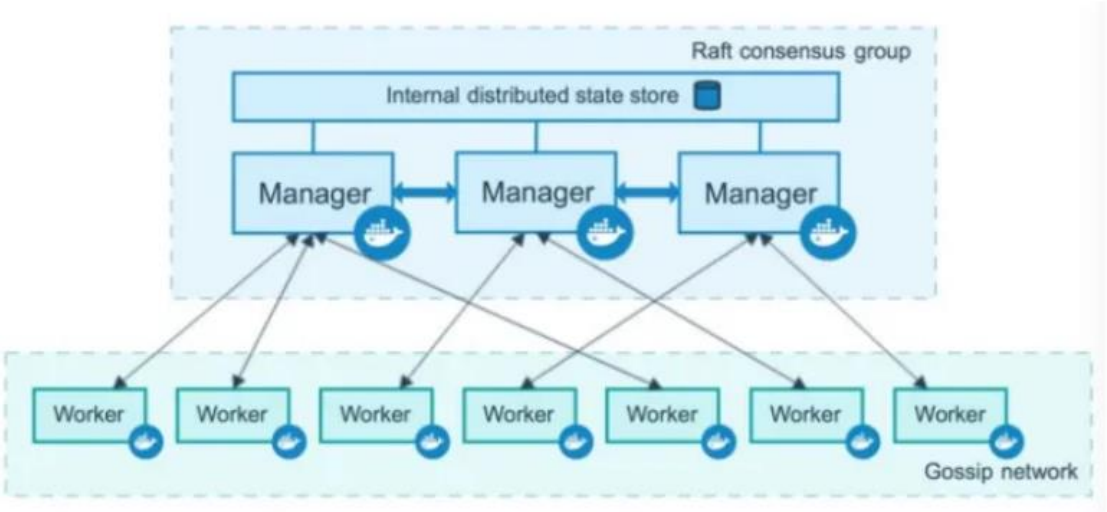
Docker aşağıdaki ağ sürücülerini içerir:

- **Bridge:** Bridge, konteyner için varsayılan bir ağ sürücüsüdür. Birden fazla docker aynı docker host ile iletişim kurduğunda kullanılır.
- **Host:** Konteyner ve host arasında ağ izolasyonuna ihtiyaç duymadığımızda kullanılır.
- **None:** Tüm ağ bağlantısını devre dışı bırakır.
- **Overlay:** Overlay, Swarm hizmetlerinin birbirleriyle iletişim kurmasını sağlar. Konteynerlerin farklı Docker ana bilgisayarlarında çalışmasını sağlar.
- **Macvlan:** Macvlan, konteynerlere MAC adresleri atamak istediğimizde kullanılır.[18]

1.5.5.7. DOCKER SWARM

Swarm, yönetici (master) ve işçi (worker) makinelerden oluşan bir kümedir (cluster). Docker ile birlikte sunulan bu servis, birden fazla ana bilgisayarda birden fazla instance çalıştırarak tek nokta hatası problemlerini çözmeyi amaçlar.

Docker Swarm, entegre cluster yönetimi, ölçekleme, periyodik güncellemeler, güvenlik, yük dengeleme, dahili DNS, konteynerlar arası durum yönetimi ve bileşen gruplandırma gibi çözümler sunar. [19]



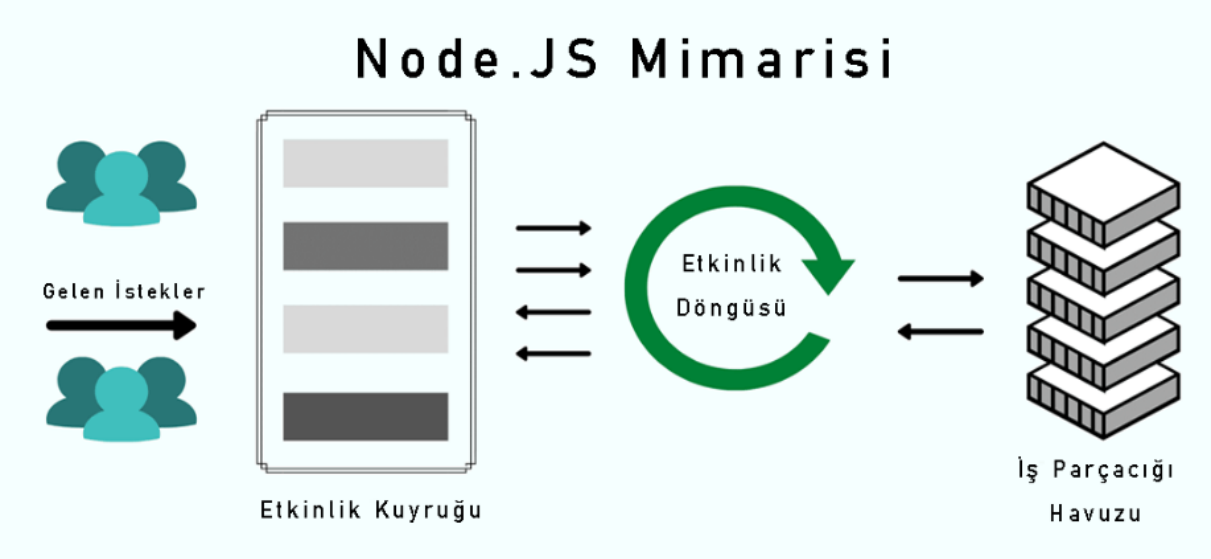
Şekil 10 Docker Swarm Yapısı

1.6. NODE.JS NEDİR?

Node.js, JavaScript çalıştırmak için kullanılan, 2009 yılında Ryan Dahl tarafından geliştirilen açık kaynaklı ve platformlar arası bir ortamdır. Node.js, bir programlama dili değildir; JavaScript'in asenkron yapısını kullanarak hızlı ve verimli çalışır. V8 JavaScript motoru üzerinde çalıştığı için, JavaScript kodunu doğrudan makine koduna derler ve yüksek performans sağlar.[21]

1.6.1. NODE.JS MİMARİSİ VE ÇALIŞMA MANTIĞI

Node.js, aynı anda birden fazla istemciyi işlemek için "Tek İş Parçacıklı Olay Döngüsü" mimarisini kullanır. Çok iş parçacıklı istek-yanıt modelinde, birden fazla istemci istek gönderir ve sunucu, her birini işlerken yanıt verir. Eşzamanlı çağrıları işlemek için bir iş parçacığı havuzu kullanılır ve her istek geldiğinde, ayrı bir iş parçacığı atanır. [22]



Şekil 11 Node.js Mimarisi

1.6.2. NODE.JS AVANTAJLARI

- **Tek İş Parçacıklı Olay Döngüsü Mimarisi**

  Node.js, aynı anda birden çok isteği engellemeden verimli bir şekilde işleyen tek iş parçacıklı bir olay döngüsü mimarisi kullanır. Bu, yüksek verim ve düşük gecikme süresi sağlar ve gerçek zamanlı uygulamalar için idealdir.

- **Hızlı İşlem Hızı**

  Google'ın V8 JavaScript motoru üzerine kurulu olan Node.js, JavaScript kodunu makine koduna derleyerek hızlı performans sağlar. Bu hız, Node.js'i yüksek performanslı uygulamalar için mükemmel kılar.

- **Kolay Ölçeklenebilirlik**

  Node.js, çok sayıda eşzamanlı bağlantıyı işleyebilir ve kümelemeyi destekleyerek yükü birden çok CPU çekirdeğine dağıtabilir. Bu, ölçeklenebilir uygulamalar oluşturmayı kolaylaştırır.

- **Büyük ve Aktif Geliştirici Topluluğu**

Node.js, geniş ve aktif bir geliştirici topluluğuna sahiptir. Bu topluluk, yeni modüller, araçlar ve çerçeveler oluşturarak platforma katkıda bulunur ve geliştiricilerin sorunlarına çözüm bulmalarına yardımcı olur.

- **Çok Yönlülük ve Esneklik**

Node.js, web geliştirme, sunucu tarafı komut dosyası oluşturma ve komut satırı araçları gibi çeşitli uygulamalarda kullanılabilir. Esnek yapısı, geliştiricilerin özel ihtiyaçlarına yönelik uygulamalar oluşturmasına olanak tanır.[23]

1.6.3. NODEJS UYGULAMALARI

- Web sunucuları ve API'ler
- Komut satırı araçları
- Masaüstü uygulamaları
- Gerçek zamanlı sohbet programları ve çevrimiçi oyunlar
- Gerçek zamanlı işbirliğine dayalı düzenleme araçları
- Gerçek zamanlı analitik ve veri görselleştirme araçları
- Veri tabanları ile entegrasyon
- Çalışan makine öğrenimi algoritmaları
- Nesnelerin İnterneti (IoT) uygulamaları [24]

1.7. MONGOOSE NEDİR?

Mongoose, MongoDB işlemlerini kolaylaştıran bir ODM modülüdür ve geliştirilen uygulamaya göre modeller oluşturarak çalışır.

MongoDB veritabanında veriler karmaşık yapılarla saklanabilir. Bu durum, verilerle işlem yaparken zorluklara neden olabilir.

Bu sorunları aşmak ve veri alanlarını belirlemek için modelleme yapılır.[25]

1.8. REDIS NEDİR?

C ile yazılmış, key-value şeklinde tasarlanmış bir NoSQL veritabanıdır. Veriyi bellekte tuttuğu için çok hızlı okuma ve yazma yapılır. [26]

2. GEREKLİ KURULUMLAR

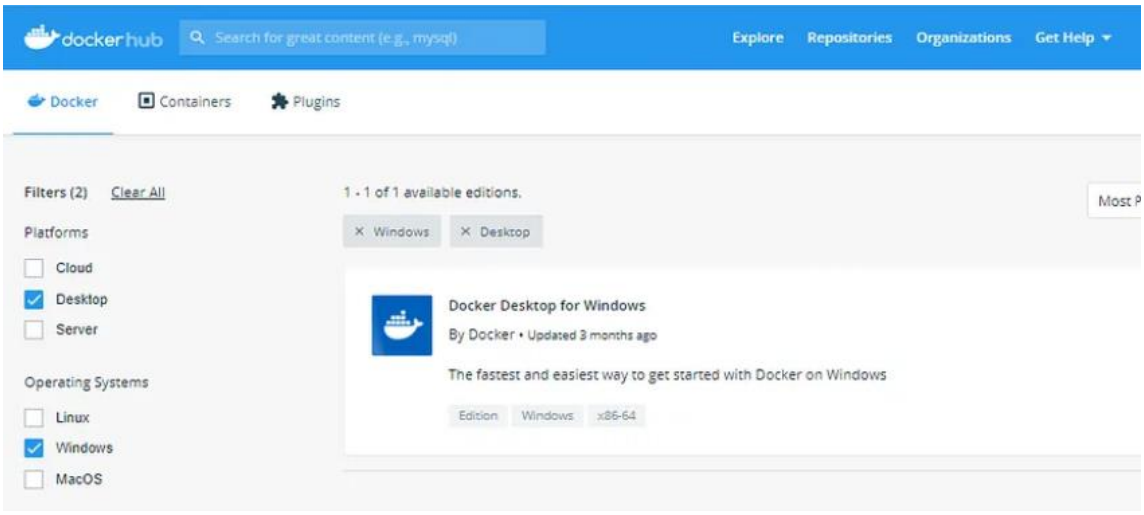
2.1. DOCKER KURULUM

- **WINDOWS KURULUM**

Docker'ı İndirme

İlk adım olarak, Docker'ı resmi web sitesinden indirmeniz gerekiyor. İlgili işletim sistemi sürümüne uygun olan Docker Desktop sürümünü

https://www.docker.com/products/docker-desktop/ üzerinden bulun ve indirin.



Şekil 12 Docker Hub Resmi Sitesi

Docker'ı Kurma

İndirilen Docker yükleyici dosyasını çalıştırarak kurulum sürecini başlatın. Kurulum sihirbazı size rehberlik edecektir. İlgili adımları takip ederek Docker'ı bilgisayarınıza kurun.

Docker Desktop'ın Başlatılması

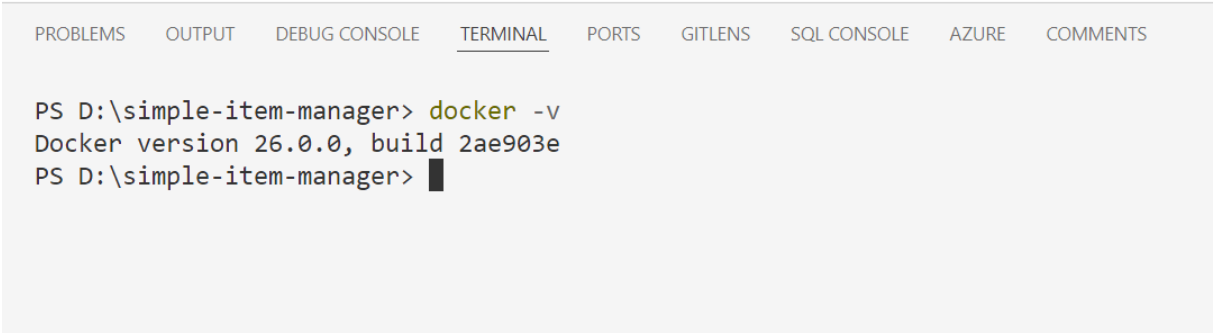
Kurulum tamamlandıktan sonra, Docker Desktop uygulamasını başlatın. Sistemde (Windows) Docker ikonunu bulabilirsiniz. İkon üzerine tıklayarak Docker Desktop'ı başlatın.

Docker'ın Çalışıp Çalışmadığını Kontrol Etme

Docker Desktop başladıktan sonra terminali açın. Aşağıdaki komutu kullanarak Docker'ın doğru bir şekilde yüklendiğini ve çalıştığını kontrol edebilirsiniz:

- **docker**
- **docker --version**
- **docker run hello-world**

Yukarıdaki komutlar herhangi bir hata vermeden çalışıyorsa, Docker başarıyla kurulmuş ve çalışıyor demektir.



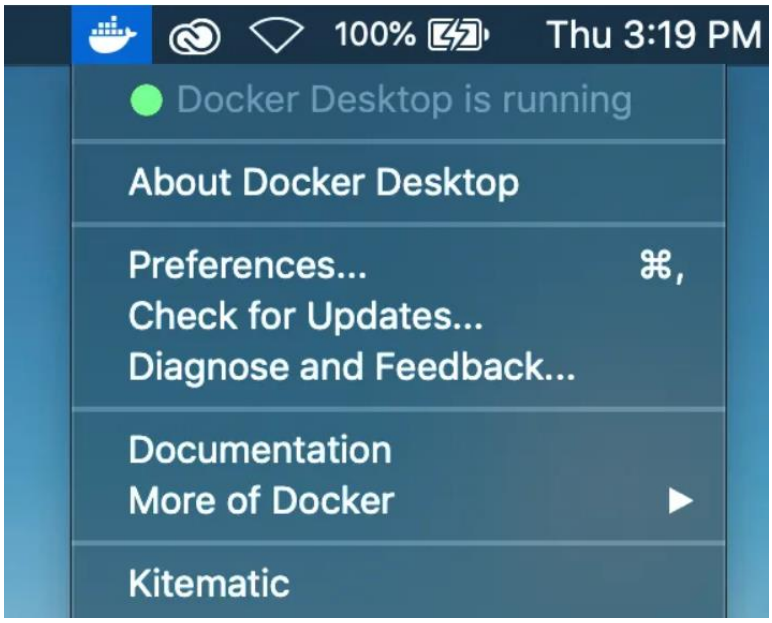
Şekil 13 docker -v Komutunun Çalıştırılması

- **LINUX KURULUM**

hub.docker.com sitesinde **explore** -> **Docker CE** -> **operating system** -> Linux seçip gerekli talimatları takip ederek dockerı sisteminize kurabilirsiniz.[4]

- **MACOS KURULUM**

İlk olarak aşağıdaki link aracılığıyla *"docker.dmg"* dosyasını indirmemiz gerekiyor. İndirip Application klasörünün içine attıktan sonra Docker App' i açıyoruz.



Şekil 14 Docker MacOS Kurulum

Yukarıda görüldüğü gibi Docker çalışır vaziyette karşımıza geliyor.

Sıradaki adım ise Terminal' i açıp yüklenip yüklenmediğini anlamamız için ufak bir test etmemiz gerekiyor. Karşınıza çıkan console' a "docker" yazıp basarsanız docker' ın kurulduğunu anlayabilirsiniz. [27]

2.2. NODE.JS KURULUM

- **WINDOWS**

Windows işletim sisteminde Node.js kurulumu aşağıdaki gibi klasik kurulum adımlarını takip ederek tamamlanır.

- node -v
- node --version

Kurulum tamamlandıktan sonra kurulum sonucu Windows komut istemini (CMD veya PowerShell) çalıştırdıktan sonra yukarıdaki komutlarından birini yazarak Node.js kurulum sonucunu ve Node.js sürümünü görebilirsiniz.

Şekil 15 Node.js Kurulumunu Kontrol

Node.js kurulumunu başarıyla yaptıysanız Windows komut istemi yukarıdaki gibi bir sonuç verecektir.[28]

- **LINUX**

Debian ve Ubuntu tabanlı linux dağıtımlarında Node.js kurulumu için aşağıdaki komut ile işletim sistemindeki paket bilgilerini güncellemeniz yararlı olacaktır.

- sudo apt-get update

İşletim sisteminizde curl paketi yüklü değilse curl paketini yüklemeniz gerekiyor.

- sudo apt-get install curl

Curl paketini yükledikten sonra artık kuruluma geçebiliriz.

Kurulum için öncelikle kurulum paketini indiriyoruz. (kurulum paketi sürüme göre farklılık gösterir)

- curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash –

Kurulum paketini indirdikten sonra kuruluma başlayabiliriz.

- sudo apt-get install -y nodejs

Kurulum işlemi tamamlandıktan sonra kurulumun başarılı olup olmadığını aşağıdaki komutlar ile kontrol edebilirsiniz.[28]

- node -v
- node –version

- **MACOS**

Mac OS işletim sistemi Node.js kurulumu Windows işletim sistemine benzer şekilde gerekli olan kurulum dosyası indirilerek yapılır.

- node -v
- node –version

Kurulum işlemi tamamlandıktan sonra kurulumun başarılı olup olmadığını yukarıdaki komutlar ile kontrol edebilirsiniz. [28]

2.3. REDIS KURULUM

Kurulum MacOS ve Linux tabanlı işletim sistemlerinde aşağıdaki adreste yer alan adımlar takip edilerek yapılır.

- https://redis.io/download

Aşağıdaki adımlarda indirilen Redis arşivi, çıkartılarak derlenir.

- wget https://download.redis.io/releases/redis-6.2.4.tar.gz
- tar xzf redis-6.2.4.tar.gz
- cd redis-6.2.4
- make

Kurulum sonrası Redis server çalıştırılır.

- src/redis-server

Arka planda çalıştırmak için ampersand(&) işareti sonuna eklenebilir.

- src/redis-server &

Redis içerisinde yer alan komut yorumlayıcısı kullanılarak Redis işlemleri yapılır.

- src/redis-cli

Komut yorumlayıcısına aşağıdaki komut yazılarak test edilir.

- Ping

Windows tabanlı işletim sistemlerinde WSL veya Windows Subsystem for Linux olarak adlandırılan özellik kullanılarak yukarıdaki adımlar takip edilerek Redis çalıştırılabilir. [29]

2.4. MONGODB KURULUM

- **WINDOWS**

MongoDB kurulumu için gerekli dosyalar indirildikten sonra klasik Windows kurulum adımlarını takip ederek kurulum yapılır.

Kurulum sırasında gelişmiş kurulum seçilerek veritabanı dosyalarının yer alacağı klasör ve MongoDB servisi ayarlanabilir.

MongoDB veritabanını komut yorumlayıcısında kullanabilmek için ortam değişkenlerindeki path alanına mongod.exe dosyasının bulunduğu klasörün eklenmesi gerekir.

Gerekli olan ayarlar yapıldıktan sonra komut yorumlayıcısına aşağıdaki komut yazılarak kurulum kontrol edilir.[30]

- mongod --version

- **LINUX**

Linux tabanlı işletim sistemlerinde kurulum işletim sistemindeki paket yöneticisine göre farklılık gösteri.

İşletim sisteminde grafik arayüzü var ise işletim sistemine ait kurulum dosyaları indirilerek kurulum yapılır.

Kurulum komut yorumlayıcısı ile yapılacaksa ilk olarak paket yöneticisine göre gerekli olan repo eklenir.

Repo eklendikten sonra işletim sisteminde bulunan paket yöneticisine göre aşağıdaki komutlarla kurulum tamamlanır.[30]

- sudo apt-get install -y mongodb-org
- sudo yum install -y mongodb-org

Kurulum sonrası komut yorumlayıcısına aşağıdaki komut yazılarak kurulum kontrol edilir.

- mongod –version

- **MACOS**

MongoDB kurulumu için gerekli dosyalar indirildikten sonra arşivden çıkarılır.

Arşivden çıkarılan dosyalar usr/local/bin adresine kopyalanarak kurulum yapılır.

Kurulum tamamlandıktan sonra PATH değişkenine MongoDB dizini gösterilerek kurulum tamamlanır.

MacOS tabanlı işletim sisteminde kurulum Homebrew paket yöneticisi ile aşağıdaki komut ile de yapılabilir.

- brew install mongodb

Kurulum sonrası komut yorumlayıcısına aşağıdaki komut yazılarak kurulum kontrol edilir.[30]

- mongod –version

3. UYGULAMA

3.1. DOCKERFILE OLUŞTURMA

dockerfile > ...

You, 4 days ago | 2 authors (sevginuroksuz and others)

```
1  # Temel imaj olarak resmi Node.js imajını kullan
2  FROM node:14
3  # Uygulama dosyalarını konteyner içindeki çalışma dizinine kopyala
4  WORKDIR /usr/src/app
5  # package.json dosyasını kopyala
6  COPY package* json./
7  # Bağımlılıkları yükle
8  RUN npm install
9  # Uygulama dosyalarını kopyala
10 COPY . .
11 # Uygulamanın hangi portta çalışacağını belirle
12 EXPOSE 3000
13 # Uygulamayı başlat
14 CMD ["node", "server.js"
15
```

Şekil 16 Dockerfile Oluşturma

3.2. DOCKER-COMPOSE.YML DOSYASINI OLUŞTURMA

docker-compose.yml > volumes

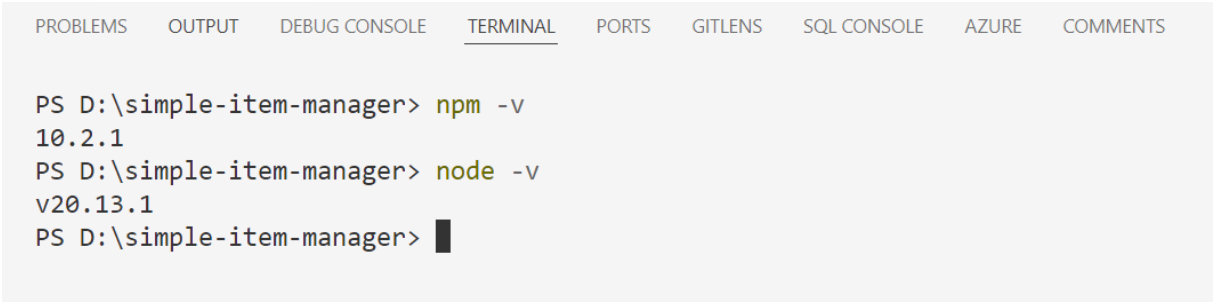
You, 22 seconds ago | 2 authors (sevginuroksuz and others) | docker-compose.yml - The Compose specification establishes a standard fo

```yaml
version: '3.8'

services:
  # MongoDB servisi
  mongo:
    image: mongo
    container_name: mongodb
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db    # MongoDB verileri için kalıcı depolama

  # Redis servisi
  redis:
    image: redis
    container_name: redis
    ports:
      - "6379:6379"

  # Uygulama servisi
  app:
    build: .    # Dockerfile'den imaj oluştur
    ports:
      - "3000:3000"    # Uygulamanın dış dünyaya hangi porttan açılacağı
    depends_on:
      - mongo    # Uygulamanın MongoDB'ye ve Redis'e bağımlılığı
      - redis
    environment:
      MONGO_URL: mongodb://mongo:27017/myapp    # MongoDB bağlantı URL'si
      REDIS_URL: redis://redis:6379    # Redis bağlantı URL'si

volumes:
  mongo-data:    # MongoDB veri birimi
```

Şekil 17 Docker-Compose.yml Dosyası Oluşturma

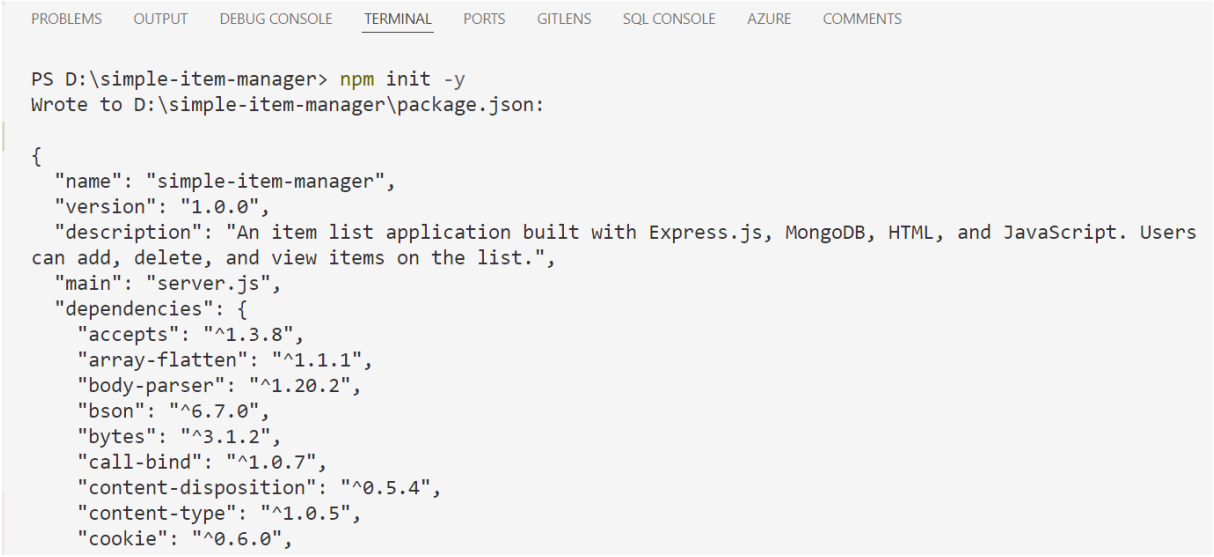
3.3. UYGULAMAYI ÇALIŞTIRMA

- Docker uygulamasını çalıştırıyoruz.
- Terminali açıyoruz.
- "npm -v" ve "node -v" komutlarıyla node.js ve npm kurulumunu kontrol ediyoruz:



Şekil 18 Node.js Kurulum Kontrolü

- "npm init -y" komutunu terminale yazarak package.json dosyasını oluşturuyoruz.



Şekil 19 Package.json Dosyasını Oluşturma

- “npm install express mongoose” komutunu çalıştırarak express ve mongoose adlı paketlerin yüklenmesini sağlıyoruz.

```
PS D:\simple-item-manager> npm install express mongoose

up to date, audited 85 packages in 1s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\simple-item-manager>
```

Şekil 20 MongoDB Kurulum

- Docker-Compose.yml dosyamızı çalıştırmak için “docker-compose up -d” komutunu terminale yazıyoruz ve enter tuşuna basıyoruz.

```
PS D:\simple-item-manager> docker-compose up -d
time="2024-05-28T21:57:33+03:00" level=warning msg="D:\\simple-item-manager\\docker-compose.yml: `versi
on` is obsolete"
[+] Running 4/4
 ✓ Network simple-item-manager_default  Created                                    0.0s
 ✓ Container mongodb                    Started                                   0.1s
 ✓ Container redis                      Started                                   0.1s
 ✓ Container simple-item-manager-app-1  Started                                   0.1s
PS D:\simple-item-manager>
```

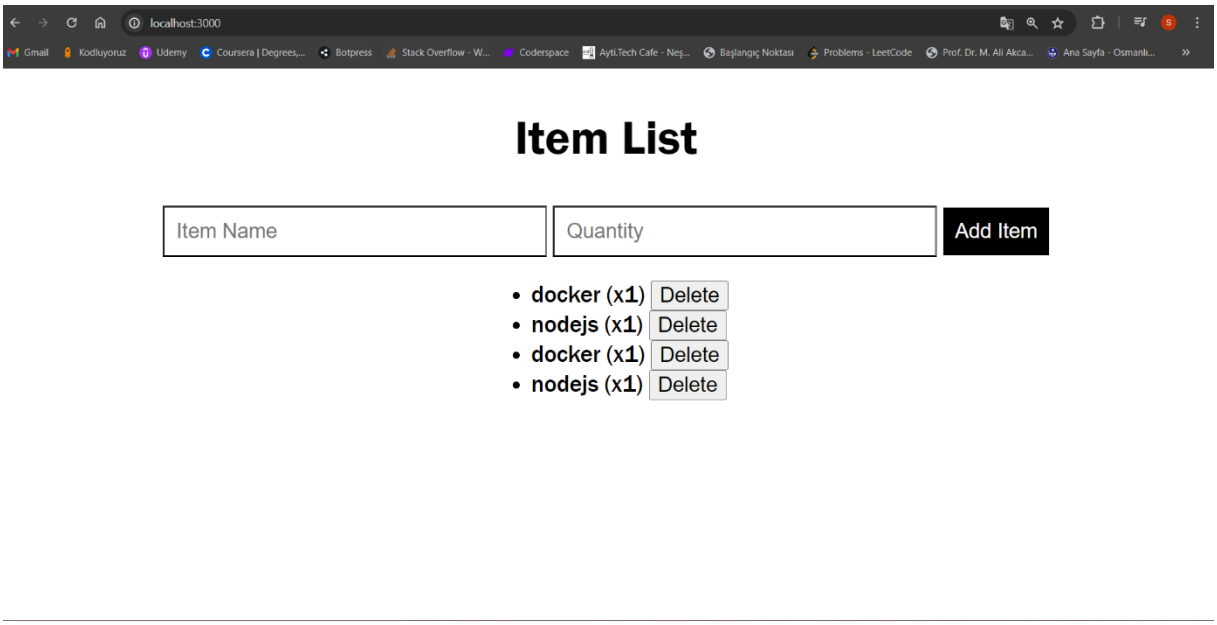
Şekil 21 Docker-Compose.yml Dosyasını Çalıştırma

- “node server.js” komutuyla node.js uygulamamızı başlatıyoruz.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SQL CONSOLE  AZURE  COMMENTS

PS D:\simple-item-manager> node server.js
(node:7892) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no ef
fect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:7892) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has
 no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server running at http://localhost:3000
```

Şekil 22 Node.js Uygulamasını Başlatma

- Uygulamamıza localhost:3000 yazarak tarayıcıdan açıyoruz.



Şekil 23 Uygulamanın Çalışan Görüntüsü

4. SONUÇ

Bu proje, Docker Compose kullanarak MongoDB ve Redis ile entegre bir mikroservis uygulamasının nasıl oluşturulacağını başarıyla göstermiştir. Proje, konteynerleştirme teknolojilerinin uygulama geliştirme sürecini nasıl basitleştirdiğini ve uygulamaların taşınabilirliğini nasıl artırdığını ortaya koymuştur. Ayrıca, MongoDB ve Redis gibi farklı veri yönetim sistemlerinin bir arada kullanılarak nasıl güçlü ve esnek bir veri altyapısı oluşturulabileceği de gösterilmiştir.

Docker Compose dosyası, uygulamanın tüm bileşenlerinin kolayca yönetilebilmesini ve çalıştırılabilmesini sağlamış, geliştirme ve dağıtım süreçlerini önemli ölçüde kolaylaştırmıştır. Bu proje, mikroservis mimarisi ve konteyner teknolojilerini kullanarak modern uygulama geliştirme pratiklerini başarıyla uygulamıştır.

5. KAYNAKLAR

1. https://medium.com/batech/docker-nedir-docker-kavramlar%C4%B1-avantajlar%C4%B1-901b37742ee0
2. “Uygulama Sanallaştırmada Yeni Bir Yaklaşım”- Docker Özge AYAZ1 , Galip AYDIN
3. “Mevcut Bir Bulut Uygulamanın Docker ve Docker Teknolojilerini Kullanarak Bağımsız Çalışabilir Hale Getirilmesi Deneyimi” - Serdar Mumcu
4. https://deniz-turkmen.medium.com/docker-nedir-3986ab6fc47
5. “Docker: Lightweight Linux Containers for Consistent Development and Deployment”- Dirk Merkel
6. https://www.docker.com/blog/docker-index-shows-surging-momentum-in-developer-community-activity-again/
7. https://aws.amazon.com/tr/docker/#:~:text=Docker%20neden%20kullan%C4%B1lmal%C4%B1%3F,%C3%A7al%C4%B1%C5%9Fabilen%20tek%20bir%20nesneniz%20olur.
8. https://bulutistan.com/blog/docker-nedir/#Docker_Ne_Icin_Kullanilir
9. https://docs.docker.com/get-started/overview/#:~:text=with%20fewer%20resources.-,Docker%20architecture,to%20a%20remote%20Docker%20daemon.
10. https://docs.docker.com/reference/cli/dockerd/
11. https://www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/#:~:text=The%20Docker%20client%20provides%20a,run%20on%20a%20Docker%20host.
12. https://erdincuzun.com/docker/01-03-docker-mimarisi/
13. https://www.geeksforgeeks.org/what-is-docker-image/
14. https://medium.com/devopsturkiye/temel-d%C3%BCzeyde-docker-mant%C4%B1%C4%9F%C4%B1-ve-kavramlar%C4%B1-bde5418858d4
15. https://medium.com/@AleynaaCelik/docker-nedi%CC%87r-daba94b88ac5
16. https://www.turhost.com/blog/docker-nedir-ne-ise-yarar/
17. https://medium.com/@mrdevsecops/docker-objects-e561f0ce3365
18. https://medium.com/aws-certified-user-group-turkey/docker-%C3%BCzerine-genel-bak%C4%B1%C5%9F-docker-swarm-kullan%C4%B1m%C4%B1-46d9c17df6f6
19. https://eneshazr.medium.com/docker-ve-docker-compose-nedir-4ea7781a0fa4
20. https://coderspace.io/sozluk/nodejs
21. https://www.ihs.com.tr/blog/node-js-nedir/
22. https://www.niobehosting.com/blog/node-js-nedir-ve-avantajlari-nelerdir/
23. https://www.cozumpark.com/node-js-nedir-neden-node-js-kullanmalisiniz/
24. https://www.yusufsezer.com.tr/node-js-mongoose/
25. https://medium.com/@yrn_brn/docker-kurulumu-a7d1967626e1
26. https://medium.com/onedio-mobile-development-team/macos-i%C3%A7in-docker-kurulumu-ve-docker-%C3%BCst%C3%BCne-activemq-kurulumu-cc0c06c4411e
27. https://www.yusufsezer.com.tr/node-js-kurulumu/
28. https://www.yusufsezer.com.tr/redis/
29. https://www.yusufsezer.com.tr/mongodb-kurulumu/