Docker Nedir?

Docker en net tanımlamayla open source bir ‘container’ teknolojisidir. Docker, aynı işletim sistemi üzerinde, yüzlerce hatta binlerce birbirinden izole ve bağımsız containerlar sayesinde sanallaştırma sağlayan bir teknolojidir. Web uygulamalarımızın kolayca kurulumunu, testini, çalışmasını ve deploymentını sağlar. Bunun yanında sunucu maliyetlerini önemli ölçüde azaltır.

Docker Daemon:

Hypervisor’ün dockerdaki karşılığıdır. Bütün CPU ve RAM vb gibi işletim sistemine ait işlerin yapıldığı bölümdür.

Container Nedir:

Docker Daemon tarafından Linux çekirdeği içerisinde birbirinden izole olarak çalıştırılan process’lerin her birine verilen isimdir. Virtual Machine (Sanal Makina) analojisinde Docker’ı Hypervisor’e benzetirsek fiziksel sunucu üzerinde halihazırda koşturulmakta olan her bir işletim sisteminin (sanal sunucunun) Docker’daki karşılığı Container’dır.

Image Nedir:

Containerlar layer halindeki Image’lardan oluşur. Docker Image ise containerlara kurulacak ve run edilecek olan uygulamaların veya OS’lerin image dosyalarıdır. Örnek verecek olursak mysql, mongodb, redis, ubuntu, mariadb.. Yüzlercesi mevcut. Buyrun: [Docker Images List](https://hub.docker.com/search?q=&type=image).

Fakat burada şöyle bir ayrıntı var. Aslında container içerisindeki imagelerin içeriklerinin ve tanımlamalarının gerekli tutulduğu bir dosya var. Bu dosyamızın adı: Dockerfile. Tamamen bu isimde, ne büyük ne de küçük harflerden oluşan, bir bakıma containerlar içindeki imagelerın registration’ını yapan bir dosya. Burda önemli nokta şu: Her bir image Dockerfile dediğimiz bu dosyanın altında tanımlanması zorunlu.

Docker Registry Nedir:

Gelelim bence işin en zevkli kısmına. Tıpkı github gibi geliştiriciler açık kaynaklı olarak docker imagelerini yükleyerek ve [DockerHub](https://hub.docker.com/)’ta paylaşarak imagelerin bizim de indirip kullanmamıza olanak sağlıyorlar. Kısaca imagelar Docker Registrylerde tutuluyor. Örneğin siz postgres image’ını kullanmak istiyorsunuz; [postgres image](https://hub.docker.com/_/postgres) linkinden

docker pull postgres

komutu ile indirip artık bu image ile container oluşturabiliyorsunuz. Aslında github’a çok benziyor. Siz de kendiniz imageleri oluşturup yükleyip başkalarının da sizin yarattığınız imageları kullanmalarını sağlayabiliyorsunuz. İsterseniz Private olarak ta registerynizi tutabilirsiniz. Docker bu hizmeti de size sunuyor.

Docker CLI Nedir:

Kullanıcının Docker Daemon ile konuşmasını sağlayan, docker komutlarının çalıştırıldığı CLI ekranıdır.

Docker Compose Nedir:

Compose, birden fazla containere sahip docker uygulamalarını tanımlamak ve çalıştırmak için kullanılır. Compose ile uygulamanızın servislerini configure etmek için bir YAML dosyası kullanılır. Ardından, tek bir komutla configure ettiğiniz ayarlar ile tüm servislerinizi oluşturup ve başlatabilirsiniz.

Compose tüm ortamlarda çalışır: production, staging, development, testing ve diğer CI iş akışları da dahil olmak üzere.

Compose kullanmak temel olarak üç adımlı bir işlemdir:

1. Uygulamanızın ortamını Dockerfiledosyası ile herhangi bir yerde yeniden üretilebilecek şekilde tanımlayın .
2. docker-compose.yml uygulamalarınızı izole ortamda beraber çalışacak şekilde yaml dosyası içinde set edebilirsiniz.
3. docker-compose up komutuyla birlikte tüm uygulamanızı başlatır ve çalıştırabilirsiniz.

docker-compose.ymldosyası aşağıdaki gibidir :

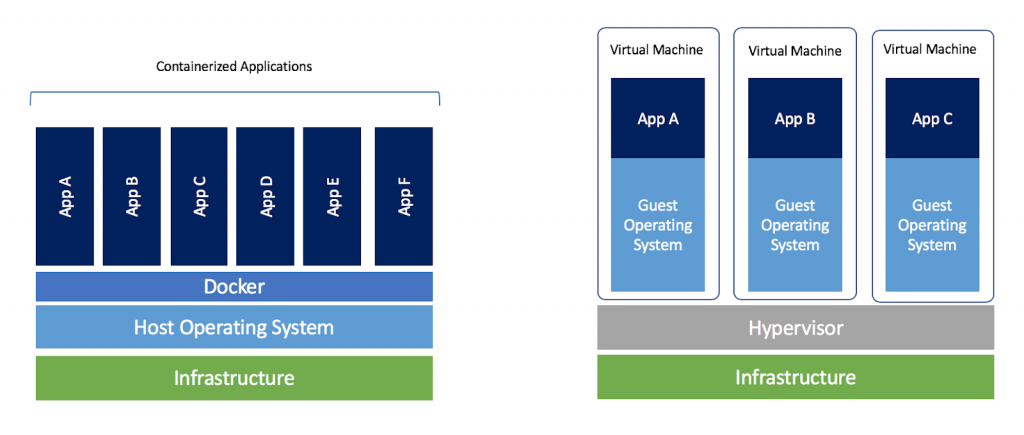
version: '2.0'  
services:  
 web:  
 build: .  
 ports:  
 - "5000:5000"  
 volumes:  
 - .:/code  
 - logvolume01:/var/log  
 links:  
 - redis  
 redis:  
 image: redis  
volumes:  
 logvolume01: {}

Docker’ın Avantajları Nelerdir?

* Docker saniyeler içerisinde başlar, çünkü içerisinde barındırdığı her bir container sadece birer processtir. Böylece karşımıza lightweight bir yapı karşımıza çıkar. Bu da bizi sanal makinelerin hantallığından kurtarmış oluyor aslında :)
* Daha hızlı deployment süreci: İşte bence en önemli avantajı diyebilirim docker için. Dockerı kullanmak için yeni bir environment kurmaya gerek yoktur. Farklı sunucularda çalışmak isteyen developerlar sadece docker imageleri indirip o sunucuda imageleri çalıştırmaları yeterlidir. Böylece ‘benim makinemde çalışıyordu, sunucuda neden çalışmıyor!’ gibi sorunlardan da kurtulmuş oluruz :)
* Daha Kolay Yönetim ve Ölçeklendirme: Bir sanal makineye göre docker üzerindeki containerleri çok daha kolay bir şekilde çalıştırabiliriz veya istediğimiz zaman yok edebiliriz. Containerleri manage etmek için farklı toollar mevcut. En çok ta Orchestrator diye nitelendirdiğimiz Kubernetes teknolojisi daha popüler olarak kullanılıyor. Kubernetes, kısaca container kullanan uygulamaların dağıtımını, ölçeklendirmesini ve yönetilmesini otomatik hale getiren açık kaynak kodlu bir sistem.
* Daha İyi Kaynak Kullanımı Sanal makinelere göre tek bir sunucu üzerindeki kaynak tüketimi dockerda çok daha verimlidir. Daha az kaynak tüketimi ile daha fazla containeri çalıştırabiliriz.
* Deployment Verimliliği: Dockerın en güzel yanlarında birisi de şu: Siz localinizde test ettiniz, uygulamanızı Test veya Live ortamınıza attınız. (veya daha çeşitli ortamlar (dev, staging, pre-prod vs..) Localde çalıştırdığınız her şey burada da aynı şekilde çalışacak. Container ve Imagelerin tutulduğu dockerfile’ları da Git üzerinde tutmanız işinizi daha da kolaylaştıracağından eminim.
* Farklı İşletim Sistemlerine Destek Vermesi Docker Windows, Linux, MacOs gibi farklı işletim sistemlerine destek verir.
* Popüler Cloud Servislerle Entegre Edilebilir. Docker , AWS, Microsoft Azure, Ansible, Kubernetes, Istio ve daha fazla tool ve cloud hizmetlerle entegre şekilde çalışabilir.

Docker ve Sanal Makine Arasındaki Farklar Nedir?

Image for post



Docker open source bir sanallaştırma teknolojisidir. Ama bildiğimiz klasik sanal makinelerden (Hypervisor, VMware) biraz farklı olarak sanallaştırma yapar. Bunun nedeni ise sanal makinelerde bulunan hypervisor katmanının bulunmaması ve container dediğimiz birbirinden tamamen bağımsız ve izole processlerden oluşmasıdır. Docker üzerinde host edildiği tek bir OS(İşletim sistemi) üzerinde yüzlerce ve binlerce docker conteiner çalışabilir ve bu conteinerlar sistem dosyalarını paylaşımlı olarak kullandıkları için kaynak tüketimleri oldukça düşük olduğu için maliyetleri düşüktür.

Sanal makine sistemlerinde, her bir sanal makine kendi işletim sistemini kullanılır ve kendi kütüphaneleri vardır. Aslında az çok hepimiz VMware kullanmışızdır. Örneğin Windows makinemizde MacOs işletim sistemini koşturmak istersek VMware ile bunu sağlayabiliyoruz. Ama farkettiyseniz bu sizin için maliyetli olur çünkü kaynak tüketimi fazladır, çünkü tamamen farklı bir işletim sistemini ayağa kaldırırsınız ve de açılış hızı yavaştır; bir süre beklemek zorunda kalırsınız. Fakat Docker teknolojisi, contenierları çalıştımak için üzerinde host edildiği tek bir işletim sistemine bağlıdır, kaynak tüketimi azdır. Ayrıca conteinerlar saniyeler içinde kullanıma hazır hale geliyor, istediğiniz zaten duraklatabiliyor, durdurabiliyor veya yeniden başlatabiliyorsunuz.

VM (Virtual Machine)

* OS : Tam işletim sistemi
* İzolasyon : Yüksek
* Çalışır hale gelmesi : Dakikalar
* Versiyonlama : Yok
* Kolay paylaşılabilirlik : Düşük

Docker

* OS : Küçültülmüş işletim sistemi imajı
* İzolasyon : Daha düşük
* Çalışır hale gelmesi : Saniyeler
* Versiyonlama : Yüksek
* Kolay paylaşılabilirlik : Yüksek

Microservice Mimarisinde Docker Neden Önemli?

Gelelim dockerın microservice mimarisinde kullanım avantajlarına. Dockerın microservicelerde kullanımının en büyük avantajı sunuculardaki maliyetlerin düşürülmesi ve sunucu kaynaklarnın tüketiminin en aza indirilmesi olarak açıklayabiliriz.

Mikroserviceleri küçük uygulamalara benzediğinden, farklı ortamlar sağlamak için mikroserviceleri kendi VM instancelarına dağıtmamız gerekir. Tahmin edebileceğiniz gibi, bir sanal makinenin tamamını uygulamanın yalnızca küçük bir bölümünü dağıtmaya ayırmak en etkili yöntem değildir. Bununla birlikte, Docker ile performans ek yükünü azaltmak ve aynı sunucuya binlerce mikroservice dağıtmak mümkündür. Çünkü Docker conteinerları sanal makinelere göre çok daha az sistem kaynağı gerektirir..

Docker, uygulamaları geliştirmek ve dağıtmak için diğer tüm yazılım araçlarından daha fazla özellik sunar. Yazılımın geliştirilmesinde ve deployment esnasında bize büyük kolaylık sağlar.