Init Docker diye bir kavram var unutma bak.

<https://container.training/>

[*Jérôme Petazzoni*](mailto:jerome.petazzoni@gmail.com)

**3P Security – 3 p of security**

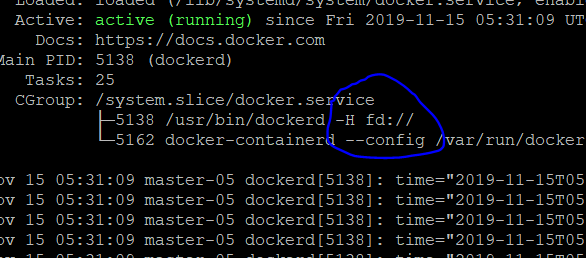
**Docker in norma bir programdan farkı metadatadır. Mesela volume u, netwotk, port vs vs vs kullanıcalak password….**

**Bir container I –rm ile çalıştıramıyorsak yani korkuyorsak bu iş olmamış demektir. Yanşi uygulama cloud native değil demektir.**

**Sensible default (çok öenmli bir kavram)**

**CLOUD NATIVE uygulama geliştime**

PYTHONUNBUFFERED buna bakmalısn. Hata çözerken işine yarayacak



-H fd:// sadece local linux socket dinlediğini göserir. Burad ip yazsaydı oradan gelen komutları da dinleyeceğini gösterir. Güvenlik açığıdır

**Docker Log**

Docker logs –f (logları devamlı ekrana yazar) (--follow)

Dockler tail 10 logs –f (son 10 logu al sonra diğerlerini calı olarak göster)

**-KILL TERM –vs KILL –KILL**

**Ubntu term de 10 sn bekler sonra kill eder**

Exited status da yazan rakamdan 128 çıkartıp KILL listesinde çıkış sebebini görebiliriz.

Kill listesi için alttaki komut

Kill –l

Örneğin SIGKILL gittiyse status çıkış sebebi 137 görünür. 197-128 = 9 bu durumda sistem memory eksikliğinden de çıkmış olabilir. Yada elle manule SIGKILL gönderimiş olabilir.

SIGHUB sinyali uygulamayı restart etmeden örneğin konfigürasyonu güncelledikten sonra nginx e cocfigürasyonu alması söylenebilir.

Attach vs exec (attach dan ziyade exec kullanılmalı)

IMAGE 0 , SECRET 0

**Imperative Image oluşturma yani herşey manual**

Image da komut yazıp yeni oluşan bu çalışan container ıima ge olarak saklayıp tag ve repository sini değiştirebiliriz.

Docker I commitlersek bize image create eder.

Layer laın en üstüne Write edilebilir Alana bizim komutlarımızın sonucumu ontainerda Yazar commit ettikten sonra o alan yeni bir layer olarak sistme yazar ve yeni bir iamge ceate etmiş olur.

Docker diff diyerek yaılabilir layer ile diğer leri arasındaki farkı görebiliriz.

Ayrıca docker history komutu ile hangi aşamalrdan geçtiini de görebiliriz.

**Declerative yol ise dockerfile yazmak**

FROM Ubuntu

RUN apt-get update

RUN apt-install figlet

Bu dosyayı Dockerfie olarak kaydedersek. Ve docker build (.) (nokta var) dersek bu lasördeki tüm doyaları docker aemon a gonderir ve image create olur.

Eğer bulunduğumuz klasördeği herdosyayı ve kalsörü daemona göndermek istemiyorsak –rf koutuyla belirtmeliyiz.

Container rı imagedan run iel her çalıştırdığımızda conraşner a –t ile tag verebiliriz. Versiyonlmaka isteyebiliriz.

Dockerfile de context de değişmesi muhtemel yerleri en alta yazki üstteki kodlar cache lensin. Build aşaması kısa sürsün.

**CMD and ENRYPOINT**

Docker container ları –rm ile çalıştırmak best practice. Kubernetes de –rm default çalışıyor.

Örneğin

Docker run –rm dockerimage [varsa parametreler komutlar]

Iş bitince kendisi durdudğunda containerda silinmiş oluyor.

Json format kulalanamk daha iyi.

Etrypoint I enson hangi uygulamayı çalıştırmak istiyorsak orada bırakmaıyızki bu container ilgili uygulama için kullanıcıldan alınacak parametreleri bu uygulamda çalıştırsın.

Entrypoint de –entrypoint ile docker run dan sonra ezerek istediğimiz uygulamayı entrypoint yapabiliriz. Ancak best practice değil.

**MULTI-STAGE**

Örneğin compiler yutkleyip bir çalışabilir app ceate edip bunu açıştıran bir image imiz olsun. Docker file işe bunu yaparsak compiler kodları da içinde olacağı için image çok şişmiş olacak. Içeri fazldan kurduklarımı dockerfile da remove komutuyla silmiş bile olsak sonuçta onceki layer kayıtlı olduğu olduğu için aslında silmiş olmayacağız. Bunun için ilk fromdan ikinci from a kadar olna bölümü from:0 olarak kabül dedip oaradaki çıktıyı ikinci from a girdi yaparsak daha küçük bir iage elde etmiş oluruz.

EXEC (shell içinde exec komutu)

Bu komut ile shell kendini kapatarak exec içinde pelirtilen programa devreder.

Bir shell scripti yazıp bunnu container da çalıştırdığımızda conainer içindeki PID si 1 olan bizim shell scriptimiz ancak bu shell bash içinde bizim scriptimizi açlaıştırmak için bash açıyor biz KILL TERM komutu göndrediğimizde contair ın term olması için bu komut PID 1 olan processe gidecek ancak asıl shell script inin çalıştırıldığı PID ye ulaşmamış olacak. O yüzden asıl çalıştırdığımız komutu entrypoint yapmamız gerekit.

Bunun için gokhan hocanın bir blog yazsısı da var.

Docker ipuçları ve cevaplar bolum 2 de <https://gokhansengun.com/docker-ipuclari-soru-ve-cevaplar-bolum-2/>

Bu bloga da ise ikinci yöntee var term komutunu yakalayıp child processe de gönderilebilir.

**Dockerfile HAzırlarken**

Production a göre hazırlayıp developer makinasına göre ezme yapmak dah iyidir. (doküman başlığı : Development Compose file) (PDF 257)

**Inspection**

Docker nginx mesela default da 80 portuna ayarlı gelir. Çünki developer un problem hangi portu dinlemesi değil. Yani bu iş sistemcinin işi o load balancer ile farklı portları dinleyip 80 e yönlendirebilir.

Yani hiç kominikasyon en iyi komnikasyondur : ) (eğer birğey belli ise farkllaştıamamak best practice dir. Ssl ise 443 dür. Yani en azında developer için böyle)

Ancak mesela postgre için farklıdır. On da aslında default olanını bırakmak lazım.

Burda dockerfile da expose komutu sadece iyi niyet mesajı aslında. Yoksa firewall luk yapmıyor. Sadece bilgi veriyor bu portdan çalışıyorum diye.

Developer aslında logları devamlı consol a yazdırır docker da bu consol ddaki logları host da biryre yazar bunular inpeck kutuyla adresini görebiliriz.

Yani developerla kominikasyonu bitirmeye çaçııyor. Developer en yapıyorsa doğal olarak sistem buna ayak uyduruarak kominikasyonu azaltıyor.

Docker inspect [containeradi]

Docker logging config den bu kısma bak docker decumnets dan.

Eğer container devamlı retart oluyorsa velog yoksa

Docker commit <container\_id> debugimage (dockerdan yeni bir mage oluşturup olduğu hali)

Daha sonra oluşan image I exec ile run yapıp problemi görebiliriz.

**DOCKER NETWOTKING**

**Bridge**

Docker container ların bağlandığı network driver default docker0 interface i (diyelimki gateway ip 172.17.0.1 bu interfae in) bu interface a bağlı bütük container birbirine full açıktır. Açılmamasını ya da birbirini görmemesini isteiğimiz container lar için ayrı bir device interface oluşturusa (yani gateway) bunun ip si 172.17.1.1 olabilir yani virtual subnet oluşturmuş olurum.

Bridge modda sadece içeriden erişilir.

Tabi eğer iki network interface varsa yani iki adet network varsa (bridge) ozaman yeni oluşturacağımız container I ilgilli bridge networkune bağlamak için –net:<bridgeadı (genelde docker0, docker:1 gibi isimi olur yada biz ne isim verirsek)>. Ancak bu durumda rule table a yazmamız lazım. Ilgili rule table a yazmamız lazım. (PDF 326)

**Host Mode**

Dışarıdan container çalıştırışırken –net=host dediğimizde artık dışarıdan erişilir. Çünki host demek hosta gelen buna geliyor demek.

**Docker Mode**

Iki container ın aynı ip yi alması demek. Bu durumda container ı aynı network namespace I ne almış oluyoruz. Yani aynı POD daki iki container I aynı ip yialır demek. Bu durumda aynı portu dinleyemezler.

Örneğin NGINX ile PHP ayrı container ama birbirlerine aynı local host diyerek erişebilliyorlar. Localhost:80 diyelim php localhost:8080 nginx…

Uygulamaya özel bir local area network kuruyoruz ancak ip ler aynı dinledikleri port farklı. Bir grup uygulama diyelim ama beraber biribirlerine bağımlı ancak tek bir uygulama gibi çalışyor …

Nicolaka/nwtshoot diye bir dockerimage var

Diyelimki bir container da bazı paketler yok ve biz buna paketde kurmak isemiyoruz ama bir problem çözmek istiyoruz. Ozaman network namespacei container modda problem çzömek istediğimiz container adı vererek o docker containerda diyelimki kurulu omyan paketlerle alakalı komutları sanki o containredaymışız gibi çalıştırabiliyoruz.

NAT mode en yavaşı sonra bridge sonra host (yani en hızlı host diyebiliriz.)

319 sayfadan itibaren network kısımlarını okumaya başla.

**SERVICE DISCOVERY (PDF 332)**

Service discovery with containers

**Overlay networks**

Normalde network host üzerinde olur ancak bazen farklı VM lerde haberleşebilen container lara ihtiyaç olur. Sebebi ise diyelimki gerçek LB dan networkumuzu soyutlamamız gerekebilir.

Burada bir magic var ipspec çalışıyor aslında network paketi içinde ip adresi gidiyor. Burayı iyi bir oku mantığını anlamak için. (IP SPEC) böylecce cluster management yani swarm ve kubernetes çalışabiliyor.

Normal iki vm arasındaki swith ve ip leri üzerinde hepsini kapsayan bir network olduğunu varasaylım buna overlay network denir.

Burada gerçel network e ise underlay network denie. Örneğin en büyük overlay network ise internetttir.

**VOLUMES**

Container ile volume yaşam döngüsü birbirinden farklıdır.

Docker volume create test-volume

Docker run –it –rm –v test-volume:/data/ ubuntu bash

Postgres

Docker run –rm –d –p 5432:5432 –name postgres Postgres:9.4 (docker file da zaten volum yazılı yani en iyi komunikasyon iletişimsizliktir. Yani bunu çalışan unutsa bile best practice docker file da vlume belitmek olduğu için Postgres dockergfile yazan bunu zaten yazmış.)

Docker volume ls dediğimizde potgres in volume unu görebiliriz.

Ancak docker da –rm default olmadığı için örneğin Postgres I –rm ile çalıştırırsak volume u da uçuruyor.

Uçurmaması için kendi volumumuzü create edip.

Bu yüzden Postgres in dockerfile ına bakacak olursa orada data folder I /var/lib/Postgres/data

Docker run –v test-volume: /var/lib/Postgres/data –rm –d –p 5432:5432 –name postgres Postgres:9.4

**Bir volume u başka bir volume a paylaştırmak mümkün bakılabilir buraya.**

**Bind Mound**

Docker run –rm –it –v $(pwd):data Ubuntu

Bu durumda çaloştığımız container a mount etmiş olduk

Mesela localde python var onu mount edebiliriz. Ya da java yı

Development amaçlkı güzel bir yaklaşım.

Storage gibi bir sistemimiz varsa onun için Volume plugin ler var onları kullanabiliriz.

Mount ise –v nin yeni versiyonu.

**DOCKER COMPOSE (PDF 456)**

Launching Our First Stack with Compose

Buradki örnege iyi bak. Redis adresi olarak counter.oy doyasında sadece redis yazıyor. Çünki DNS hallediyor.

Docker-compose scale web=10

Yani compose içindeki web servisini scale et.

Peki eğer redis I scale edersek counter.py dosyasında redis yazıyordu DNS olarak. Bu durumda dns 2 adres dönecek yada rastgele. Kubernetes araya bir servis kurup ona bir ip veriyor ve orada arkadaki redislerden birini döner.

Peki bu 10 adet web container ın ip leri ne . bunun inin network mode container olarak networkshoot image ini yukleyip web e nslookup yapabiliriz. Yada diğer dns sorgularını

Docker da ise rastgele round robin yapıyor.

Peki bu ornekte biz local klasörü bind ediyoruz web servisini 10 tane yaparsak bu nasıl olacak yani local klasör hepsine mi mount olmuş olacak (EVET).

Docker-compose down –v (colume le birlikte bütün containerları siler)

Diyelimki docker-compose scale web=2 dedik ve web e exec yapmak istiyoruz.

Docker-compose exec web dersek nereye gidecek. Rastgele birine giriyor.

Eğer Docker compose dosyasında container ların iplerini belirlemek istiyorsak yml doyasında services levelinde network belirleyip container ların ip lerini belirleyebiliriz.

Ayrıca volumes da var. istenirse ortak volume kullanılabilir.

Docker-compose ps (çalışsan containerler)

Docker-compose up hem build alıyor hemde ayağa kaldırıyor.

Docker-compose build dersek eğer deişiklik varsa build alır imagelerı ve cache ler.

!! Dikkat RUN komutuyla imaga a yazma

!! RUN build aşamasında

!! CMD çalışma aşamasında