

DOCKER KURULUM & YÖNETİMİ LAB DÖKÜMANI

Part 1 Oracle Virtual Box ve Linux/Docker İmajı Kurulumu
Part 2 Docker & Container Yönetimi

Knowledge Club
DevOps Ekip Lideri
Erdeniz Ünvan

eunvan@knowledgeclub.net

knowledgeclub.netdevopsakademi.net



Part 1

Oracle Virtual Box ve Linux/Docker İmajı kurulumu:

Uygulamanız gereken adımlar aşağıdaki gibidir:

Docker on Linux 7 Setup - Virtualbox VM on Windows

Özet: Önce VirtualBox programı download edilip kurulacak.

Sonra Linux ova dosyası download edilip VirtualBox üzerinden import edilecek.

KURULUM

- 1) VirtualBox uygulamasını download edip kuralım.
Aşağıdaki linkten son versiyonu indirin veya 6.0 üstü bir versiyonu kullanabilirsiniz.
<https://www.virtualbox.org/wiki/Downloads>
- 2) Eğer açık değilse BIOS'tan Virtualization açılmalı.
Ayrıca Windows'ta eğer açıksa Hyper-V disable edilmeli.
VirtualBox ve Hyper-V birbiriyle uyumlu değildir.
- 3) Docker Administration Eğitimi için Linux VM'ini indirelim:
<https://drive.google.com/file/d/11IKjnCAbN4dtX3A0RDIKtJRksM6D3qJl/view>
- 4) Device Yönetimi için VirtualBox Oracle VM VirtualBox Extension Pack'i indirip kuralım:

NOT: Normalde VirtualBox ilk açılışında bunu size sorup otomatik kurabilir.
- 5) Virtualbox'i açıp File'dan Import Appliance seçeneği seçip
indirdiğimiz Docker_C75_base.ova dosyasını seçelim.
- 6) İşlem bitince Virtualbox'tan start ile VM'i açalım.

Test amaçlı giriş:

VM açıldığında Kullanıcı: admin Şifre: password ile login olalım.

Sag mouse tusu ile terminal uygulaması açalım.

\$ hostnamectl

Eğer yukarıdaki komut çalışıyorsa her şey tamam demektir. VM'i kapatabiliriz.

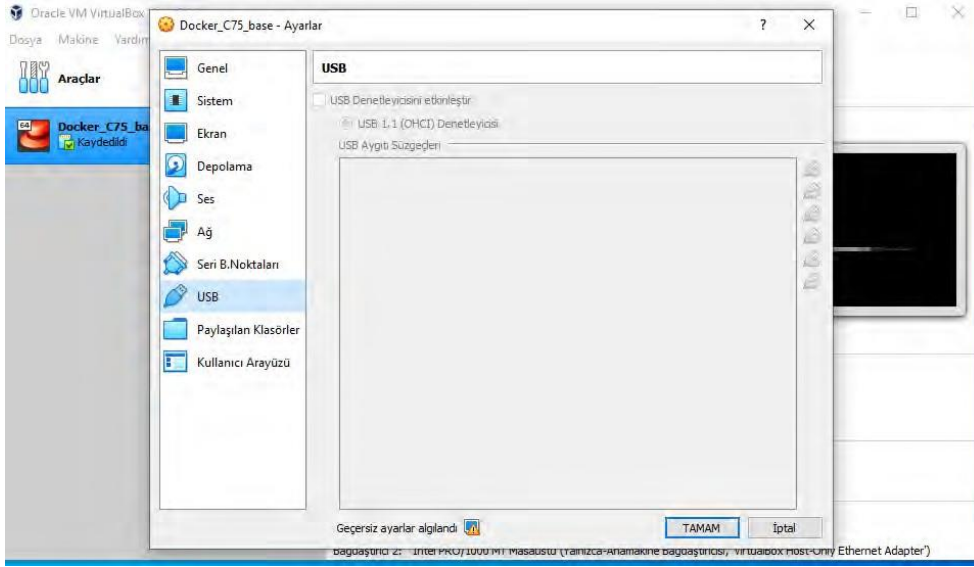
\$ poweroff

Önemli Not:

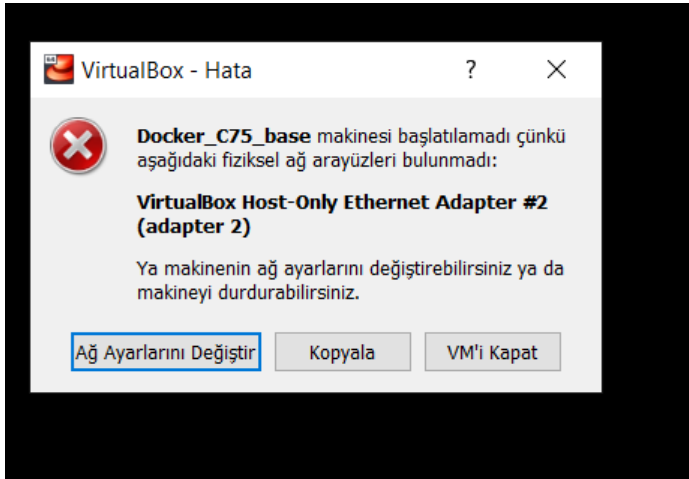
-Hyper-V kapalı olmalı ve Bios'da Virtualization ENABLE olmalı

-Oracle Virtual Box'u açtığınızda; import edeceğimiz Docker_C75_base imajına sağ tıklayalım.

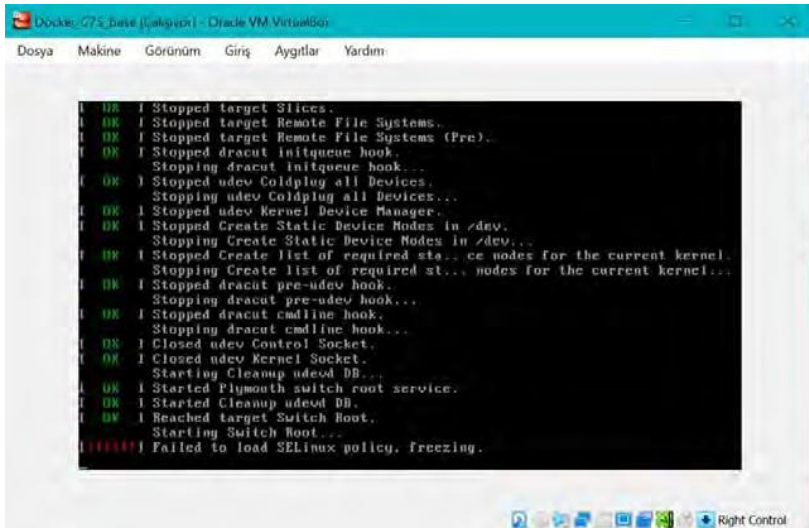
Aşağıda resimde görüldüğü gibi Ayarlar'da USB'de denetleyicisini etkinleştir'i DISABLE yapalım



Eğer yukardaki işlemi yapmazsanız; aşağıdaki hatayı alırsınız:



Eğer bilgisayarınızda aşağıdaki hatayı alıyorsanız ise; güvenlik protokollerinden ötürü Linux imajınız çalışmıyor demektir.



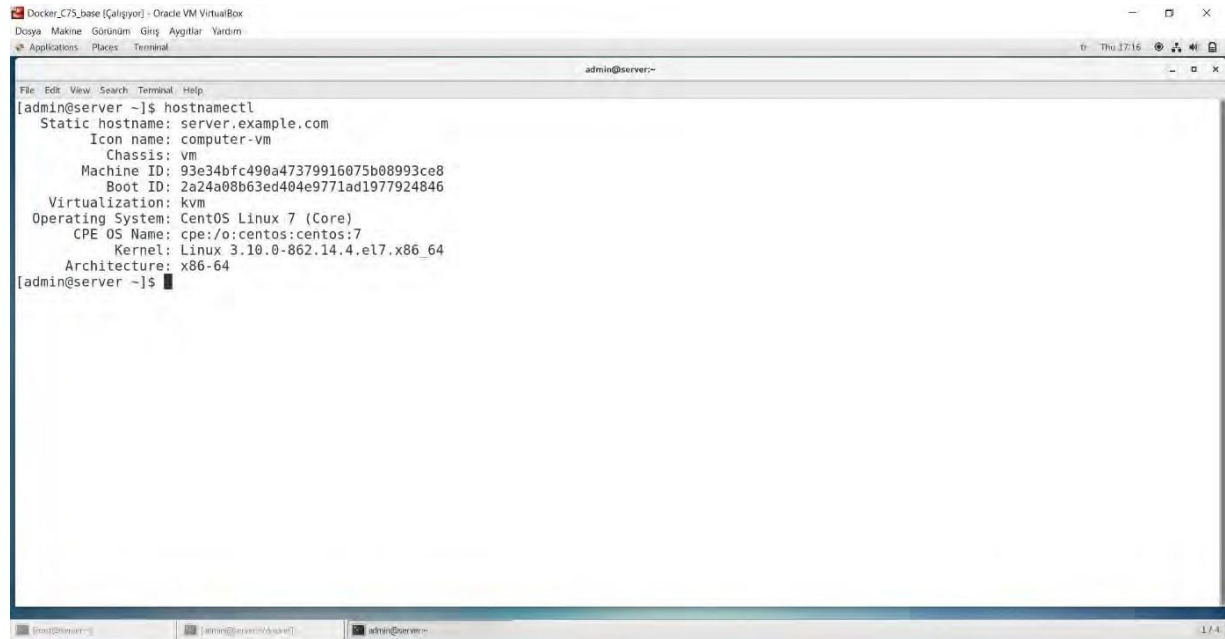
Part 2

Docker & Container Yönetimi

Bize bir docker host lazım. Bunun için en güzel çözüm bir Linux makine kurmaktır. Herhangi bir Linux makine bu iş için yeterlidir. Labımızda centos makine kullanacağız

Centos makinemizin özelliklerine bakalım:

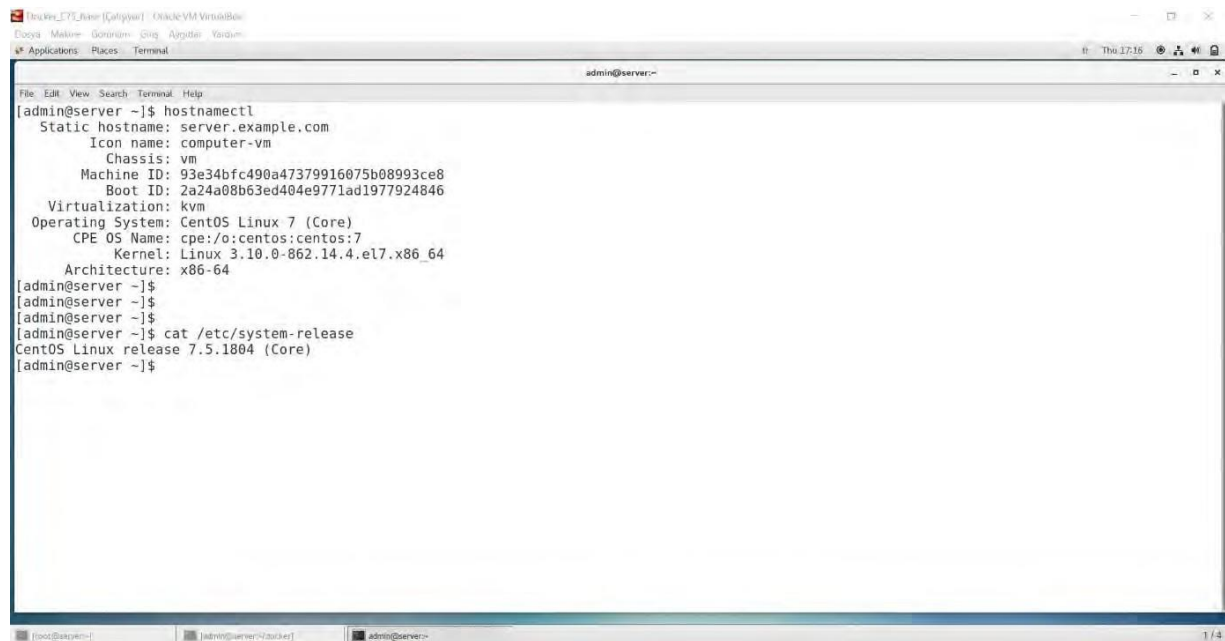
hostnamectl



```
admin@server ~]$ hostnamectl
Static hostname: server.example.com
Icon name: computer-vm
Chassis: vm
Machine ID: 93e34bfc490a47379916075b08993ce8
Boot ID: 2a24a08b63ed404e9771ad1977924846
Virtualization: kvm
Operating System: CentOS Linux 7 (Core)
CPE OS Name: cpe:/o:centos:centos:7
Kernel: Linux 3.10.0-862.14.4.el7.x86_64
Architecture: x86_64
admin@server ~]$
```

Daha fazla detay için ise:

cat /etc/system-release



```
admin@server ~]$ hostnamectl
Static hostname: server.example.com
Icon name: computer-vm
Chassis: vm
Machine ID: 93e34bfc490a47379916075b08993ce8
Boot ID: 2a24a08b63ed404e9771ad1977924846
Virtualization: kvm
Operating System: CentOS Linux 7 (Core)
CPE OS Name: cpe:/o:centos:centos:7
Kernel: Linux 3.10.0-862.14.4.el7.x86_64
Architecture: x86_64
admin@server ~]$
admin@server ~]$
admin@server ~]$
admin@server ~]$ cat /etc/system-release
CentOS Linux release 7.5.1804 (Core)
admin@server ~]$
```

Bu centos makinemizin ram'i, cpu'su hakkında bilgi sahibi olmak için:

Top



```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$ top -b 1 -n 1
top - 17:26:18 up 23:19, 4 users, load average: 0.35, 0.19, 0.11
Tasks: 232 total, 1 running, 231 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.9 us, 0.8 sy, 0.0 ni, 96.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 11851564 total, 7258598 free, 1342968 used, 3258628 buff/cache
KiB Swap: 2897148 total, 2897148 free, 0 used, 10079384 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR     S    %CPU  %MEM     time+ COMMAND
 3047 admin    20   0 1112340 83212 20504 S    0.0  0.5   0:00.21 evolution-calen
 3056 admin    20   0 1078276 96208 17268 S    0.0  0.8   0:13.50 gnome-software
 3066 admin    20   0 540208 16128 10500 S    0.0  0.1   0:00.39 abrt-applet
 3073 admin    20   0 591348 15816 5976 S    0.0  0.1   0:00.44 tracker-store
 3076 admin    39  19 736360 13624 8652 S    0.0  0.1   0:00.69 tracker-extract
 3083 admin    39  19 779636 10516 7356 S    0.0  0.1   0:00.66 tracker-miner-a
 3089 admin    39  19 561672 9728 6804 S    0.0  0.1   0:00.04 tracker-miner-u
 3094 admin    20   0 331664 9296 6828 S    0.0  0.1   0:00.06 snaplet
 3230 admin    20   0 1210044 63576 19104 S    0.0  0.5   0:00.29 evolution-calen
 3241 admin    20   0 302144 3392 2816 S    0.0  0.0   0:00.00 dbus-engine-sim
 3267 admin    20   0 1071136 28276 19928 S    0.0  0.2   0:00.69 evolution-adre
 3276 admin    20   0 1137556 54364 18080 S    0.0  0.5   0:00.28 evolution-calen
 3299 admin    20   0 1247548 26932 28360 S    0.0  0.2   0:00.13 evolution-adre
 3336 admin    20   0 315612 3252 2608 S    0.0  0.0   0:00.04 gvfsd-metadata
 3527 admin    20   0 8532 744 616 S    0.0  0.0   0:00.01 gnome-pty-helpe
 3528 admin    20   0 116732 3436 1692 S    0.0  0.0   0:00.32 bash
 3886 admin    20   0 181596 2936 2360 S    0.0  0.0   0:00.23 gconfd-2
 6913 root     20   0 232204 4376 3188 S    0.0  0.0   0:00.01 so
 6921 root     20   0 116756 3384 1684 S    0.0  0.0   0:00.17 bash
12207 admin    20   0 468936 4436 3524 S    0.0  0.0   0:00.01 gvfsd-network
12234 admin    20   0 493472 4624 3608 S    0.0  0.0   0:00.07 gvfsd-dnssd
13476 admin    20   0 116722 3440 1690 S    0.0  0.0   0:00.12 bash
22881 root     20   0 0 0 0 S    0.0  0.0   0:10.48 kworker/2:1
25386 root     20   0 0 0 0 S    0.0  0.0   0:00.00 kworker/1:0
25477 postfix    20   0 91820 4080 3060 S    0.0  0.0   0:00.65 pickup
25500 root     20   0 0 0 0 S    0.0  0.0   0:00.66 kworker/u8:1
26302 root     20   0 107396 5480 3476 S    0.0  0.0   0:00.03 dhclient
26404 root     20   0 0 0 0 S    0.0  0.0   0:00.65 kworker/0:0
26485 root     20   0 0 0 0 S    0.0  0.0   0:00.62 kworker/3:2
26488 root     20   0 0 0 0 S    0.0  0.0   0:00.14 kworker/u8:2
26623 root     20   0 0 0 0 S    0.0  0.0   0:00.01 kworker/2:0
26634 admin    20   0 116732 3420 1676 S    0.0  0.0   0:00.67 bash
26755 root     20   0 0 0 0 S    0.0  0.0   0:00.61 kworker/0:1
26777 root     20   0 0 0 0 S    0.0  0.0   0:00.60 kworker/3:0
26788 root     20   0 0 0 0 S    0.0  0.0   0:00.01 kworker/2:2
26824 root     20   0 0 0 0 S    0.0  0.0   0:00.61 kworker/0:2
26848 root     20   0 107948 352 276 S    0.0  0.0   0:00.60 sleep
```

Cırtl + C ile çıkın

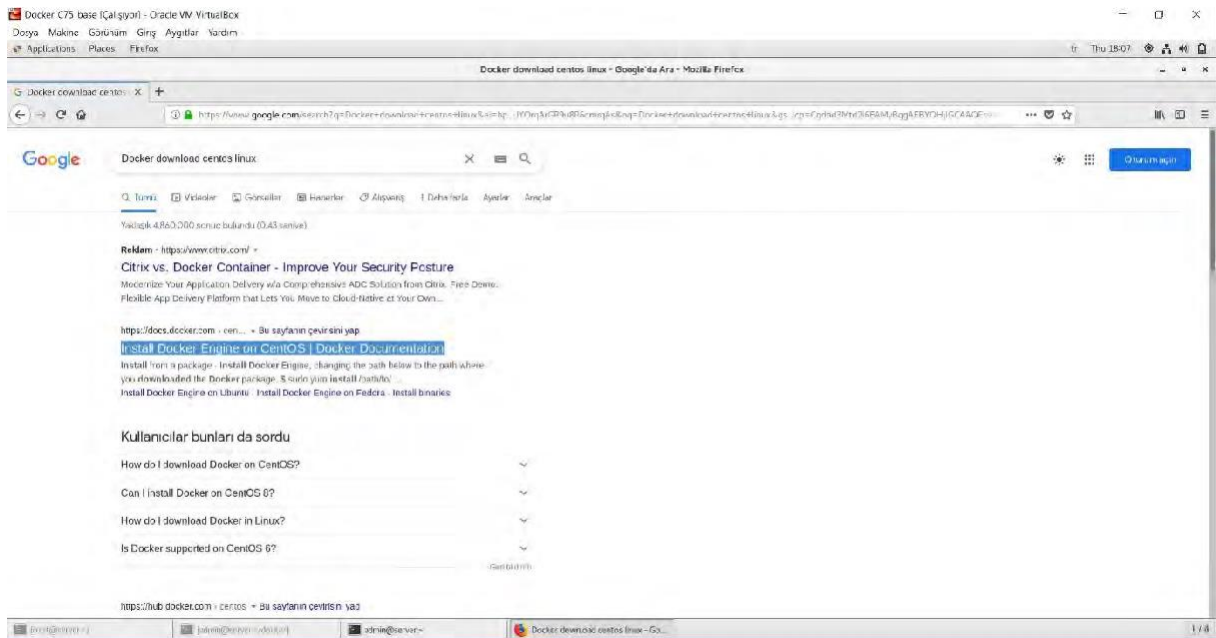
Bu Centos makinemiz fazlasıyla yeterli alt yapıya sahip

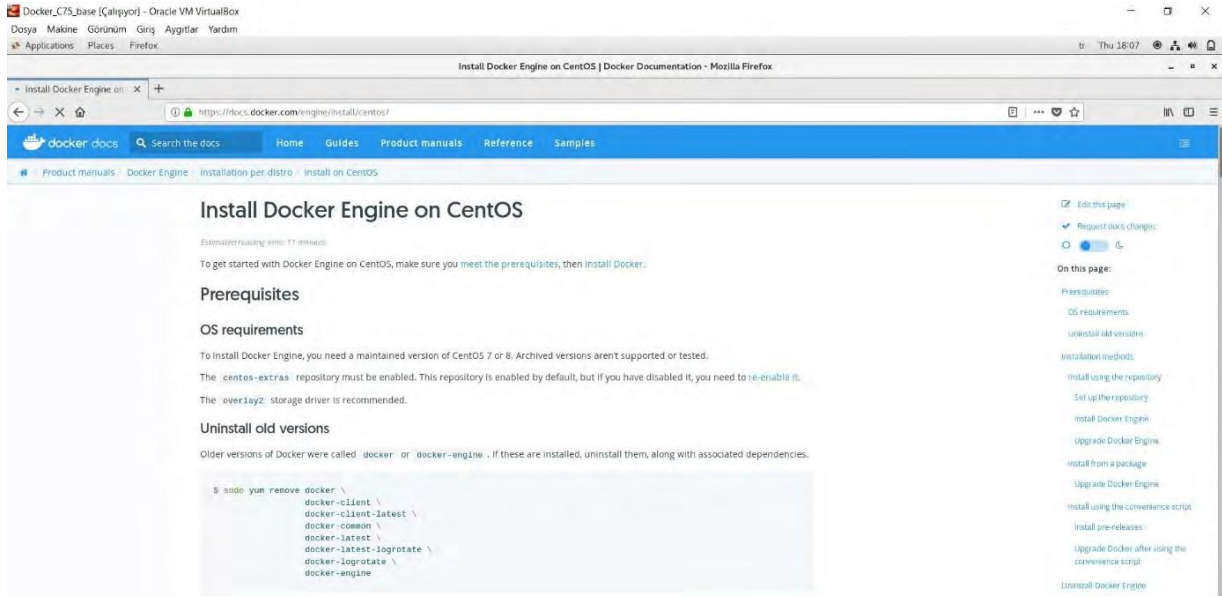
Bu Centos'ün üstüne Docker software'i kurmamız lazım

Bunun için web browser'ımızda Google da 'Docker download centos linux' yazın

Install Docker Engine on CentOS

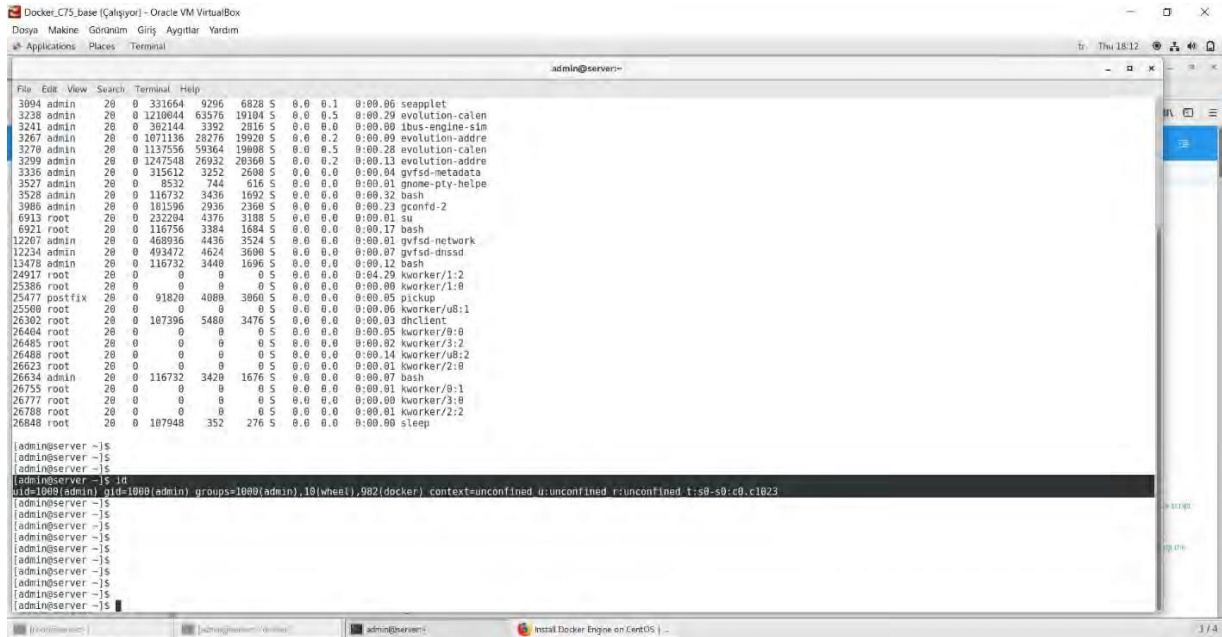
<https://docs.docker.com/engine/install/centos/>





Bu sayfadaki aşamaları takip etmemiz yeterli olacaktır.

Yum install diyerek docker'la ilgili softwareleri kurmamız ve bu işle ilgili bir kullanıcı yaratmamız gerekecektir. Bu iş için kullanacağımız kullanıcımız 'id' komutuyla görebiliriz.

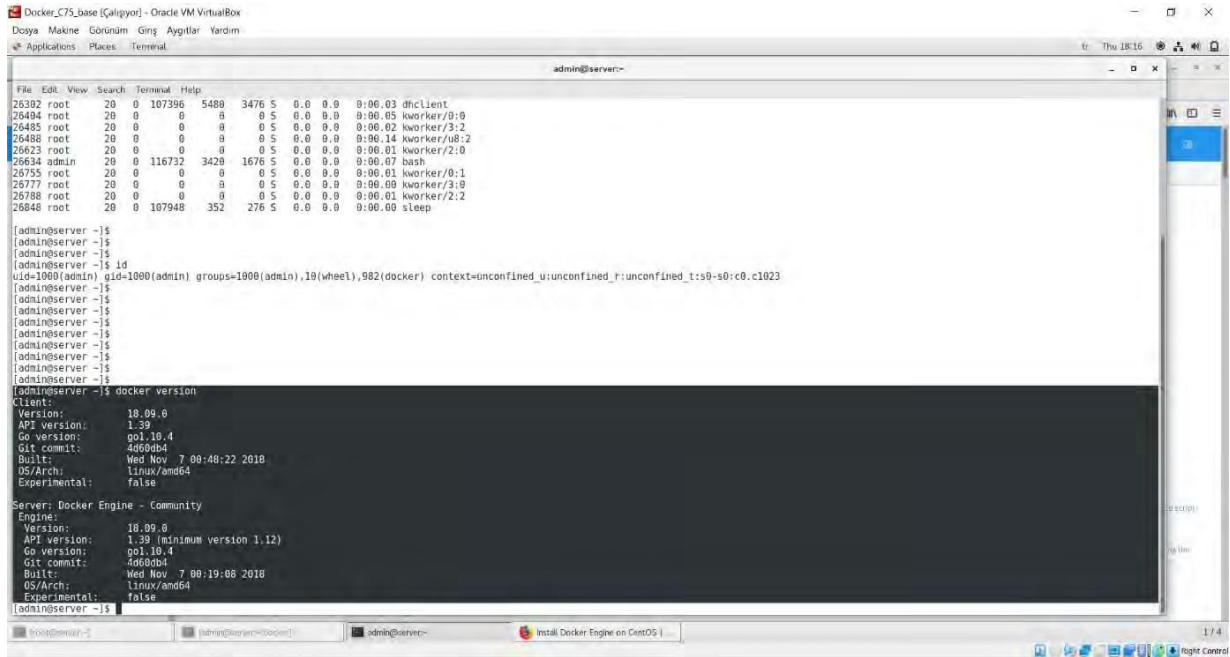


Bu sayede docker özelliği olan bir kullanıcı olduğunu görmekteyiz. Yani yukarıdaki linkteki adamları adım adım uygularsanız max. 15 dakika içinde bir docker host kurmuş olacaksınız.

Kendi labımıza gelecek olurdak; ilgili kullancıdan gördüğümüz üzere lokal makinemizin üzerinde docker software'i kurulmuş oldu.

Peki docker'in kurulu olup olmadığını nasıl anlayabiliriz ?

docker version komutunu yazalım.



```
admin@server:~$ docker version
Client:
Version:      18.09.0
API version:  1.39
Go version:   go1.10.4
Git commit:   4d60db4
Built:        Wed Nov  7 00:48:22 2018
OS/Arch:      linux/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      18.09.0
API version:  1.39 (minimum version 1.12)
Go version:   go1.10.4
Git commit:   4d60db4
Built:        Wed Nov  7 00:19:08 2018
OS/Arch:      linux/amd64
Experimental: false
```

Bir Linux makine aynı zamanda hem docker server'dır hem de docker client'tır. Docker version komutunu yazınca hem docker server'ın hem de docker client'ın hazır olduğunu görmekteyiz.

Daha detaylı bilgi almak için docker info komutunu yazmamız lazım

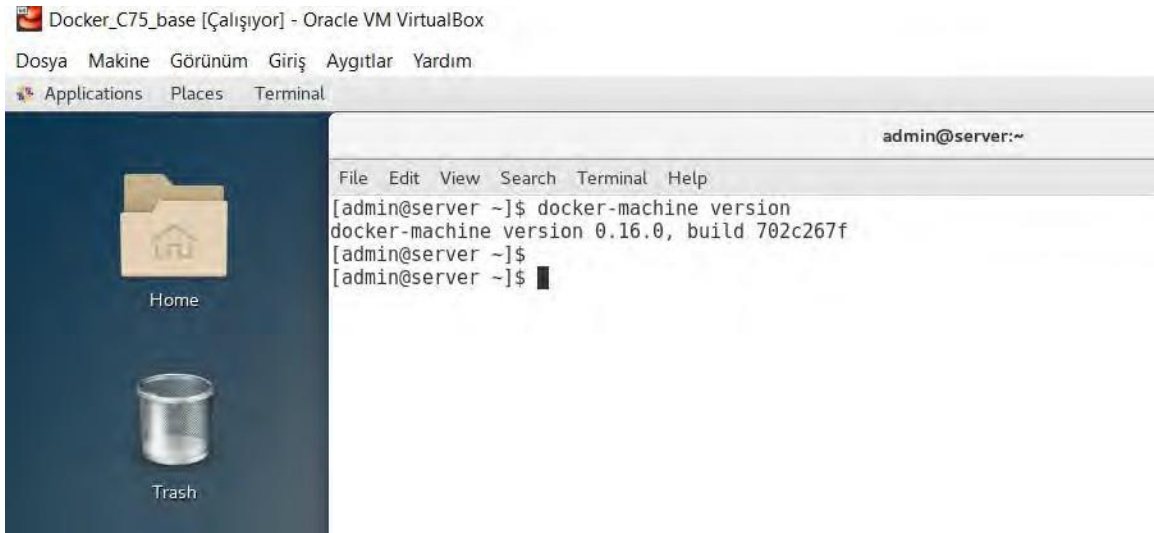


```
admin@server:~$ docker info
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 29
Server Version: 18.09.0
Storage Driver: overlay2
Backing Filesystem: xfs
Supports d type: true
Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: c446665c39c309567499bd938d47f22c7c39
runc version: 4fcd3a8177c594640722ac585fa9ca549971071
init version: fec3683
Security Options:
seccomp
  Profile: default
Kernel Version: 3.10.0-862.14.4.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 11.3GiB
Name: server.example.com
ID: 531T:1206:CB0N:BSX7:735P:ETNA:N7YE:Q0KB:1H7N:GZ4H:DLVJ:857K
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

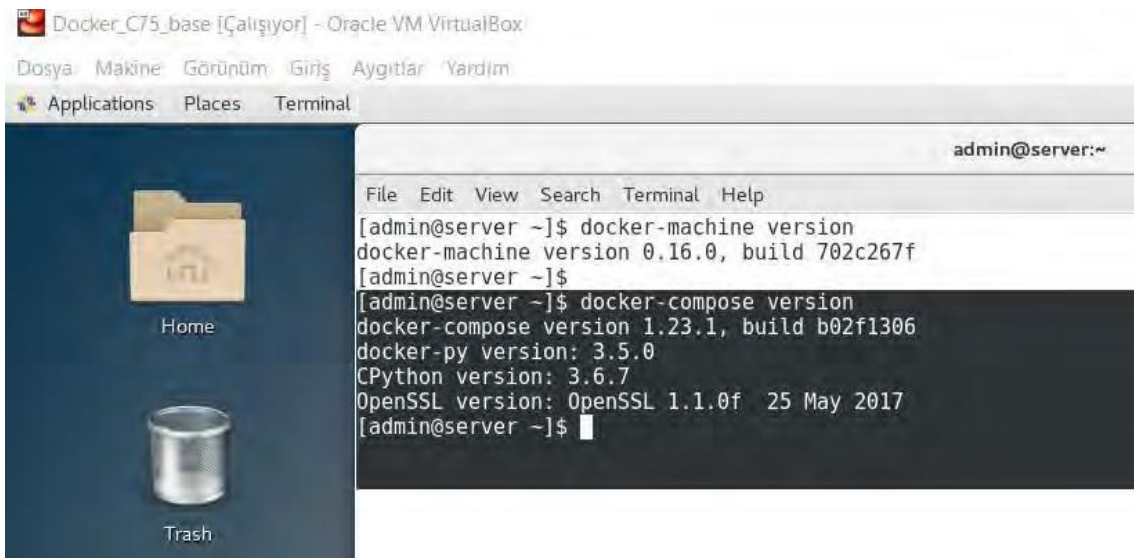
Bu sayede sistem ve versiyonla ilgili çok daha detaylı bir bilgi aldık.

Docker host ve docker client dışında fazladan docker machine hakkında bilgi almak istersek

docker-machine version

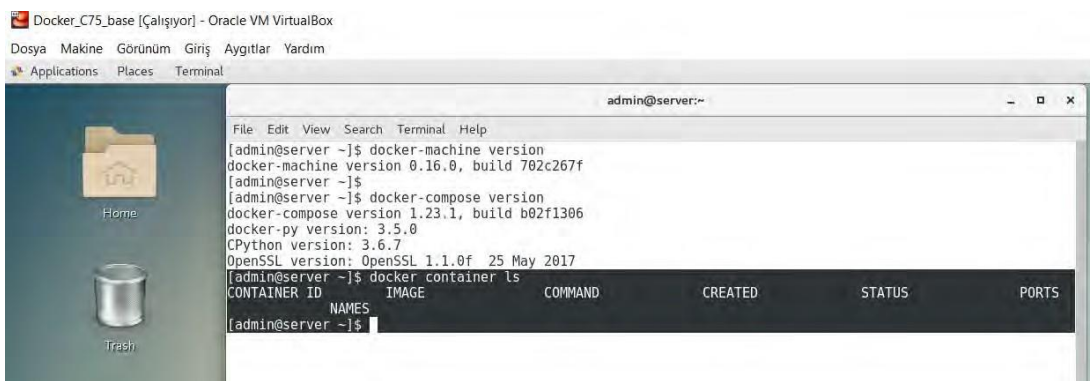


docker-compose version komutuyla da docker compose versiyonu hakkında bilgi alabiliriz.

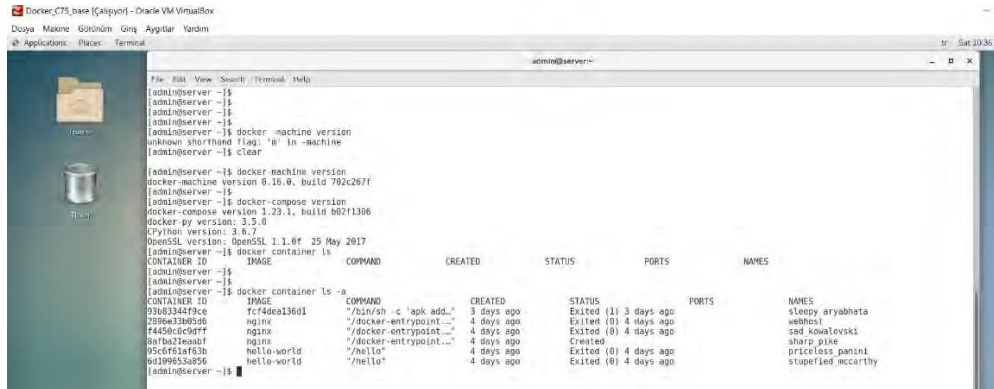


Docker compose ve docker machine'e şu an bu lab için ihtiyacımız yok. Çünkü docker composer çok sayıda container'ın yönetimi için kullanılır. Bizim bu lab için docker host ve docker client'a ihtiyacımız bulunmaktadır. Bu girişten sonra ilk container'ımız için çalışmamızın zamanı geldi. Python programlama da nasıl ilk komutumuz 'Hello World' ise burada da ilk container'ımız 'Hello World' olacaktır.

Docker'ın container ile ilgili olan tool'larına docker container ls komutu ile bakabiliriz.



Şu an makinemizde hiçbir container'ın aktif olarak çalışmadığını görmekteyiz. Peki daha önceden çalıştırdığımız ve tamamlanmış olan container'ları nasıl görebiliriz ? Linux'ta gizli dosyaları görmek için ls -a komutunu kullanırız. Daha önceden çalışmış olduğumuz ve tamamlanmış containerları da 'docker container ls -a' komutu ile görebiliriz.

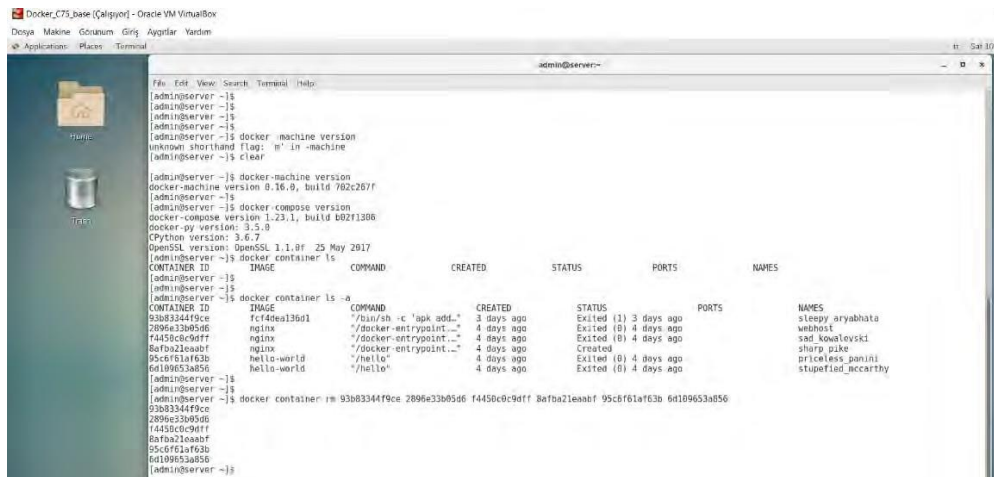


```

admin@server:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
93b03344f9ce        tc4dea136d1        "/bin/sh -c 'apk add..." 3 days ago          Exited (1) 3 days ago          sleepy_aryabhata
2896e33b05d6        nginx              "/docker-entrypoint..." 4 days ago          Exited (0) 4 days ago          webhost
f4458c9c9dff        nginx              "/docker-entrypoint..." 4 days ago          Exited (0) 4 days ago          sad_kowalewski
8afba21eaabf        nginx              "/docker-entrypoint..." 4 days ago          Created              sharp_pike
95c6f61af63b        hello-world        "/hello"            4 days ago          Exited (0) 4 days ago          priceless_panini
6d199653ab56        hello-world        "/hello"            4 days ago          Exited (0) 4 days ago          stupefied_mccarthy

```

Yukarıdaki screenshot'ta gördüğümüz üzere 3 ve 4 gün önce kullanmış olduğum Hello World ve nginx container'larını detaylarıyla görebilmekteyiz. Yukarıdaki resimde görünen bu geçmişte kullanılmış container'lar şu an çalışmıyorlar. Çalışmışlar ve bitmişler. Eğer bunları geçmişten silmek istiyorsam 'docker container rm (id number)' komutunu kullanabilirim.



```

admin@server:~$ docker container rm 93b03344f9ce 2896e33b05d6 f4458c9c9dff 8afba21eaabf 95c6f61af63b 6d199653ab56
93b03344f9ce
2896e33b05d6
f4458c9c9dff
8afba21eaabf
95c6f61af63b
6d199653ab56
admin@server:~$

```

'docker container ls -a' komutu ile tekrar geçmişe bakalım.



```

admin@server:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
93b03344f9ce        tc4dea136d1        "/bin/sh -c 'apk add..." 3 days ago          Exited (1) 3 days ago          sleepy_aryabhata
2896e33b05d6        nginx              "/docker-entrypoint..." 4 days ago          Exited (0) 4 days ago          webhost
f4458c9c9dff        nginx              "/docker-entrypoint..." 4 days ago          Exited (0) 4 days ago          sad_kowalewski
8afba21eaabf        nginx              "/docker-entrypoint..." 4 days ago          Created              sharp_pike
95c6f61af63b        hello-world        "/hello"            4 days ago          Exited (0) 4 days ago          priceless_panini
6d199653ab56        hello-world        "/hello"            4 days ago          Exited (0) 4 days ago          stupefied_mccarthy

```

Geçmişin silindiğini yukarıdaki resimde görmekteyiz.

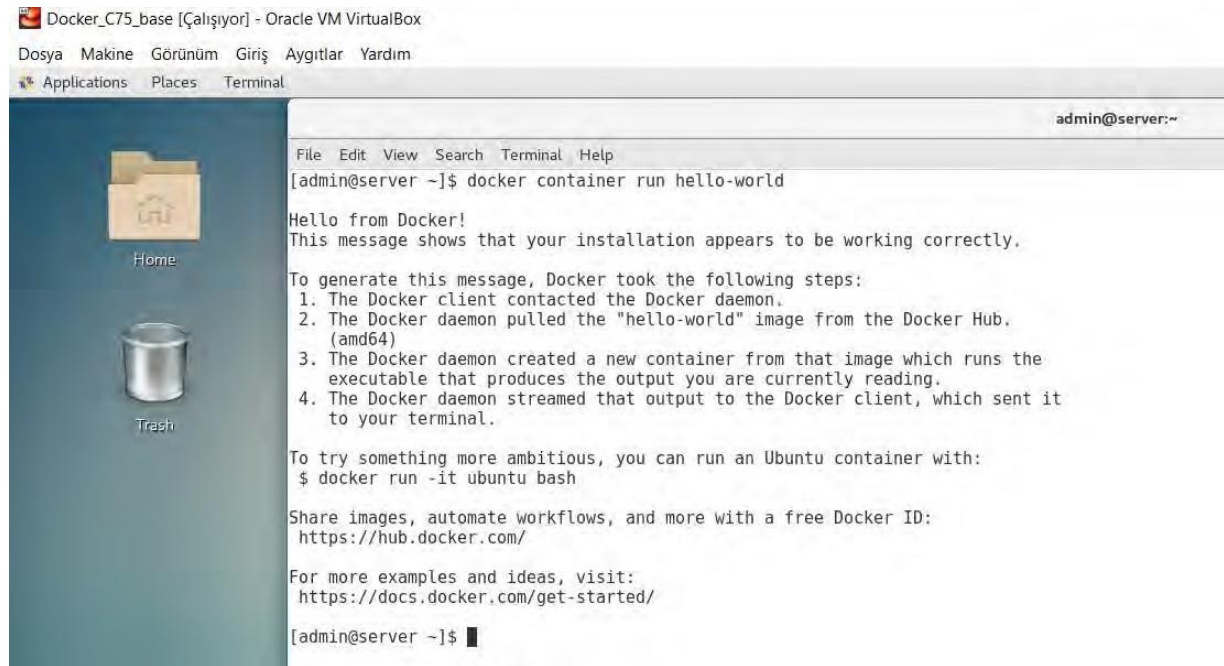
‘docker container ls -a’ komutu ile geçmişten bir şey görmediğiniz durumda tekrardan docker container ls’ demeye gerek yoktur çünkü zaten şu an için aktif olarak çalışan bir container’ımız bulunmamaktadır. Şu an makinemizde aktif olarak çalışan ve geçmişte çalışmış container yer almamaktadır.

Geçmiş temizlediğimize göre artık yeni bir başlangıç yapmak için ilk container’imizi çalıştırabiliriz.

Her container bir imaj üzerinden kurulur.

‘docker container run hello-world’ bu komutla container’imizi kurabiliriz.

Demek ki hello-world adında bir imajımız var. Bu imajda docker hub’ta yer alır. Peki bu yukardaki komutu yazdığımızda lokalimizde nasıl bir yapı olur ? Eğer imajı daha önceden lokalimize indirmişsek; bu imaj üzerinden çalışılır. Eğer imaj lokalimizde yok ise, docker hub üzerinden bu komut vasıtası ile download edilerek çalışma yapılır. Bütün container imajları docker hub’ta yer almaktadır.



```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal
admin@server:~

File Edit View Search Terminal Help
[admin@server ~]$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

[admin@server ~]$

```

İlk çalıştırdığımız docker container olan hello-world başarıyla kurulduğunu görmekteyiz.

Yukarıdaki resmimizde gördüğümüz output’un anlamı şudur:

Hello from Docker!

This message shows that your installation appears to be working correctly.

‘Kurulumumuz başarıyla tamamlandı’

To generate this message, Docker took the following steps:

‘Kurulumun tamamlanması için Docker aşağıdaki aşamaları tamamladı.’

1. The Docker client contacted the Docker daemon.
‘Burada Docker client bizim yani kodu yazan taraf. Docker daemon ise docker host’tur. Bu durum Docker Engine’nin başarıyla çalıştığını göstermektedir.’
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
‘Docker daemon komutumuz neticesinde Docker Hub’tan hello-world’ imajını indirdi.’

(amd64)

3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.

‘İmaj’ dan ilgili yeni container oluşturuldu.’

4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

‘Output ekrana yansıtıldı sonuç olarak.’

Peki docker hub’ın adresi nedir: <https://hub.docker.com/>

Peki bu container şu an çalışıyor mu ? Hemen docker container ls yazalım



```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

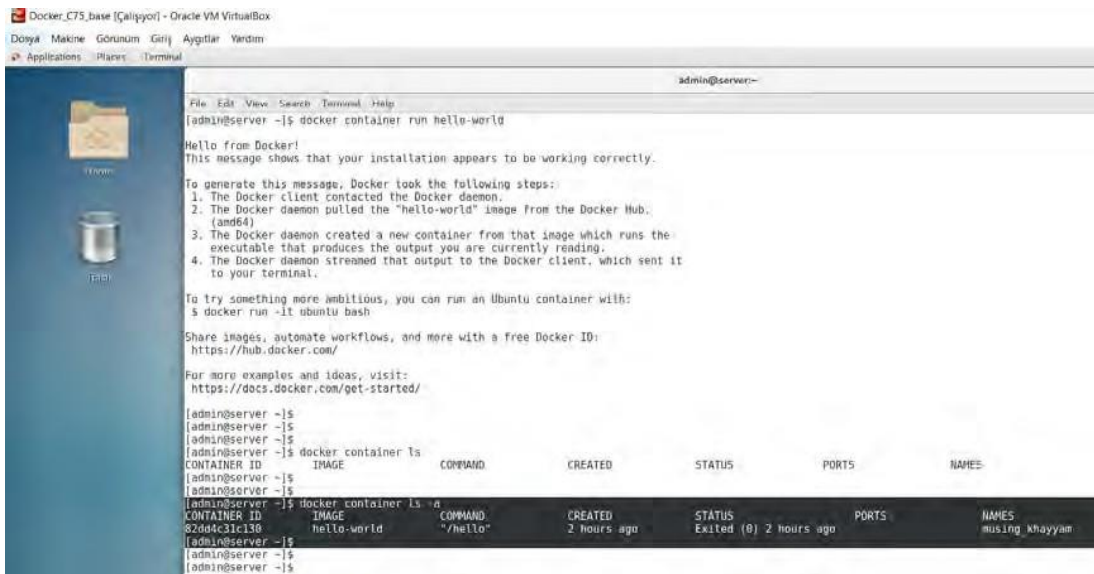
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

admin@server ~$
admin@server ~$
admin@server ~$
admin@server ~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
admin@server ~$
  
```

Hayır, çünkü hello-world çok kısa ömürlü bir container’dır. Bu uygulamanın amacı çalışıp, ekrana output’u yansıtip, sonlamaktır. Eğer bir web server ya da database container’i çalıştırsaydık; tabi ki bu container çalışmaya devam edecektir. Hello-world container’i gibi kısa ömürlü olmayacaktır.

Hello World container’i bittiği halde hidden olarak docker container ls -a komutuyla görebiliriz.



```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

admin@server ~$
admin@server ~$
admin@server ~$
admin@server ~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
admin@server ~$
admin@server ~$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS      PORTS   NAMES
82a0dc31c136   hello-world  "/hello"   2 hours ago    Exited (0) 2 hours ago    musing_khayyam
admin@server ~$
admin@server ~$
  
```

Şu an çalışmayan ve durdurulmuş container olan hello world'ün bilgisi ls -a komutuyla hidden envanterden görebildik.

Container'ı silmek isterseniz yukarıda kullandığımız gibi rm komutuyla id'sini kullanarak; silebilirsiniz. Container'ın id'si ve ismi otomatik olarak atanır. Biz kendimizde container'imiza isim verebiliriz. Ama eğer biz container'imize isim vermezsek; bu durumda otomatik olarak isim atanır. Aynı container'dan bir tane daha çalıştırırsanız; yeni container'ın farklı bir id ve otomatik isim aldığını göreceksiniz.

```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places

[admin@server ~]$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
82dd4c31c130        hello-world         "/hello"            2 hours ago         Exited (0) 2 hours ago              musing_khayyam

[admin@server ~]$ docker container run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

[admin@server ~]$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
875496f6c3bd        hello-world         "/hello"            3 seconds ago       Exited (0) 3 seconds ago              serene_minsky
82dd4c31c130        hello-world         "/hello"            2 hours ago         Exited (0) 2 hours ago              musing_khayyam

```

Yukarıdaki örneğimizde yeniden hello-world containerimizi çalıştırdık. Container çalıştı ve kapandı. Ve ls -a dediğimizde hidden envanterde farklı 2 isim ve id 'de 2 farklı hello-world container'larını görebiliyoruz. Container tarafından verilen otomatik isimler kendi database'inden gelmektedir. Bir sıfat ve bir isimden olmak üzere 2 kelimeden oluşmaktadır.

Container'ları isimleriyle de silebiliriz aşağıdaki resimde görüldüğü gibi

```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

[admin@server ~]$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
82dd4c31c130        hello-world         "/hello"            2 hours ago         Exited (0) 2 hours ago              musing_khayyam

[admin@server ~]$ docker container run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

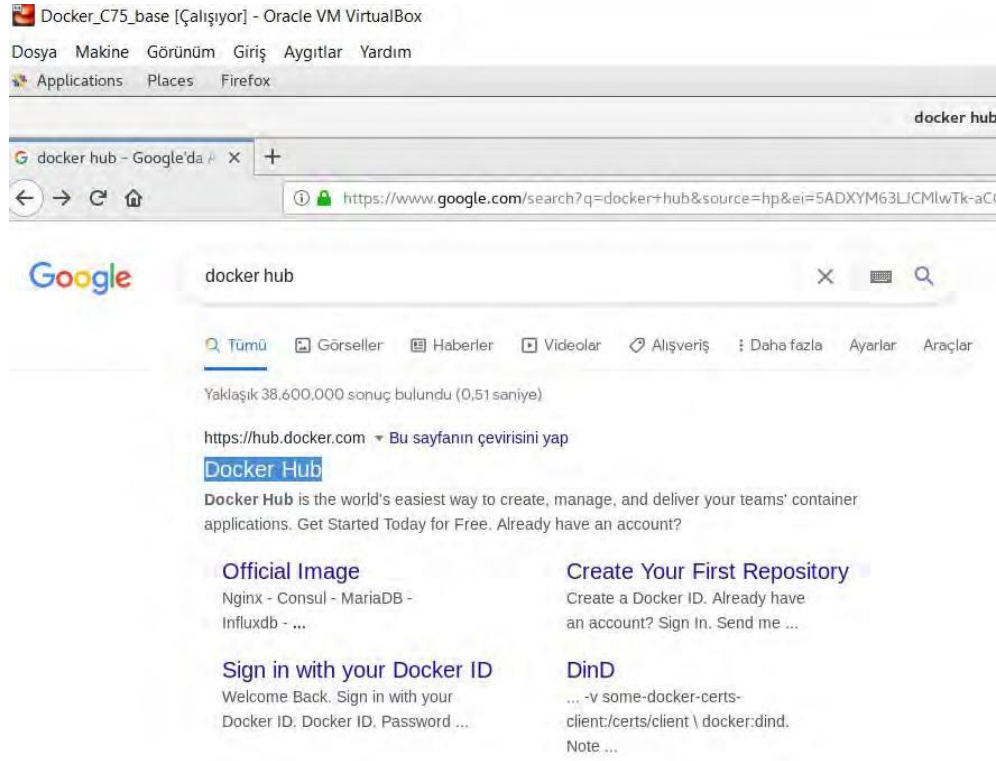
For more examples and ideas, visit:
https://docs.docker.com/get-started/

[admin@server ~]$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
875496f6c3bd        hello-world         "/hello"            3 seconds ago       Exited (0) 3 seconds ago              serene_minsky
82dd4c31c130        hello-world         "/hello"            2 hours ago         Exited (0) 2 hours ago              musing_khayyam

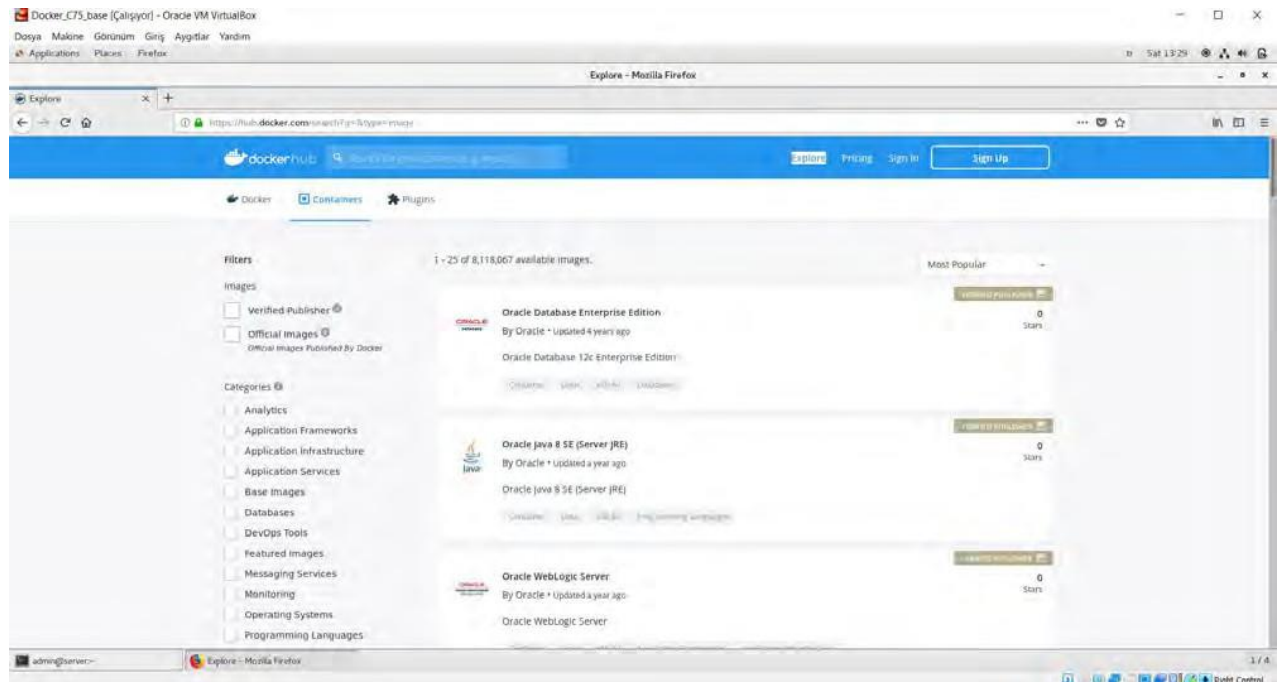
[admin@server ~]$ docker container rm serene_minsky
serene_minsky
[admin@server ~]$
[admin@server ~]$ docker container rm musing_khayyam
musing_khayyam
[admin@server ~]$
[admin@server ~]$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES

```

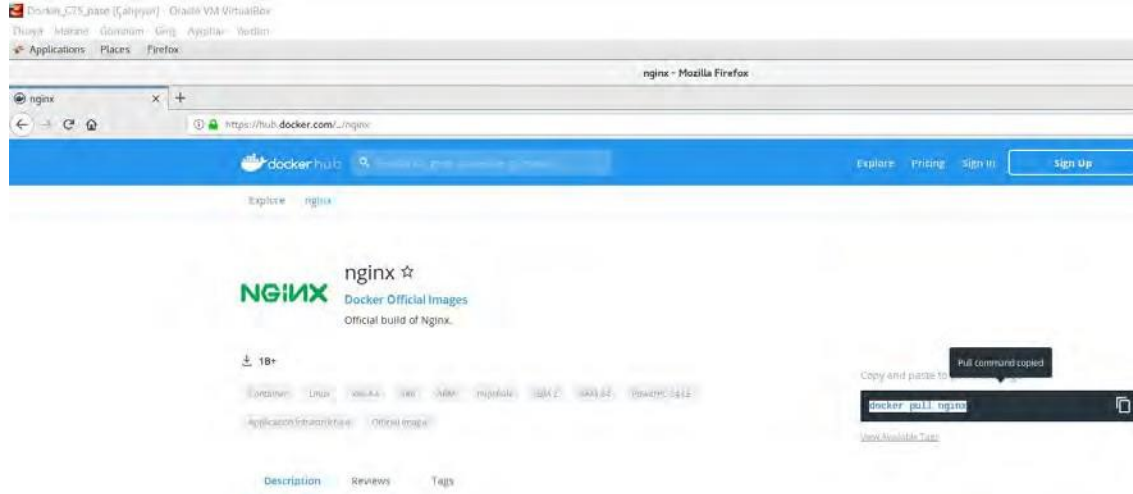

Daha önce değimdiğimiz gibi her bir container bir imaj üzerinden kurulmaktadır. Bunu Nesne Odaklı Programlama mantığındaki gibi düşünebilirsiniz. İmaj bizim için class dediğimiz sınıf'tır. Container ise o imajdan yani o sınıftan yaratılmış obje, nesnedir. İmaj bir kalıptır. Container ise o imajdan yaratılan bir uygulamadır. Bu imajlar üzerinde çalışmamız için de daha önceden başkaları tarafından bu imajların yaratılmış olması gerekmektedir. Bu labımızda NGINX dediğimiz web servis uygulamasını kuracağız. Bu uygulamaya geçmeden önce ise docker hub'ı ziyaret etmekte fayda vardır. Bunun için web browser'inızda google'a lütfen 'docker hub' yazın.



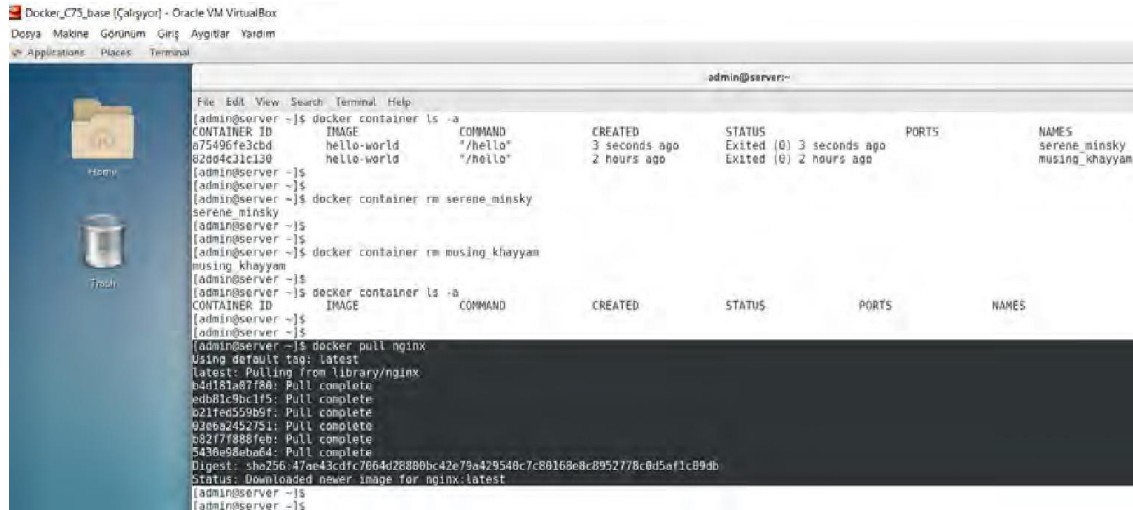
Explore sekmesinden public olarak kullanabileceğimiz imajlara bakabiliriz:



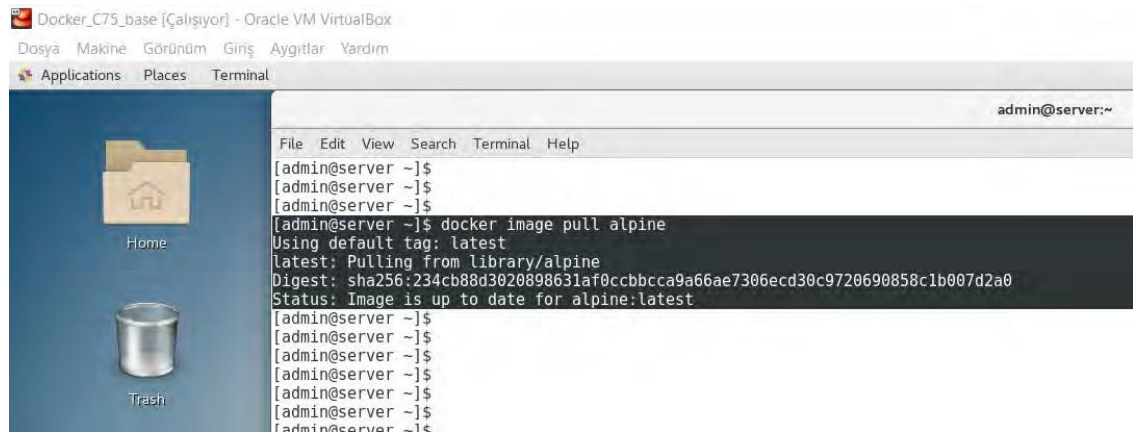
Biz bu labta nginx container'i ile çalışacağız. Komut satırında docker pull nginx yazdığımızda aşağıdaki sayfadan container imajı lokalimizdeki host makinemize download edilecektir.



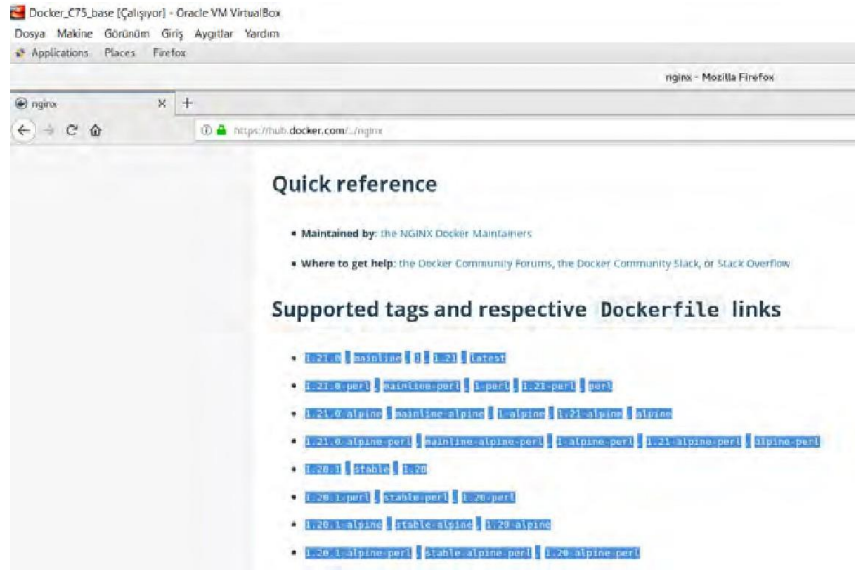
Terminal'de docker pull nginx komutunu yazarak nginx web servisini kurmuş olduk.



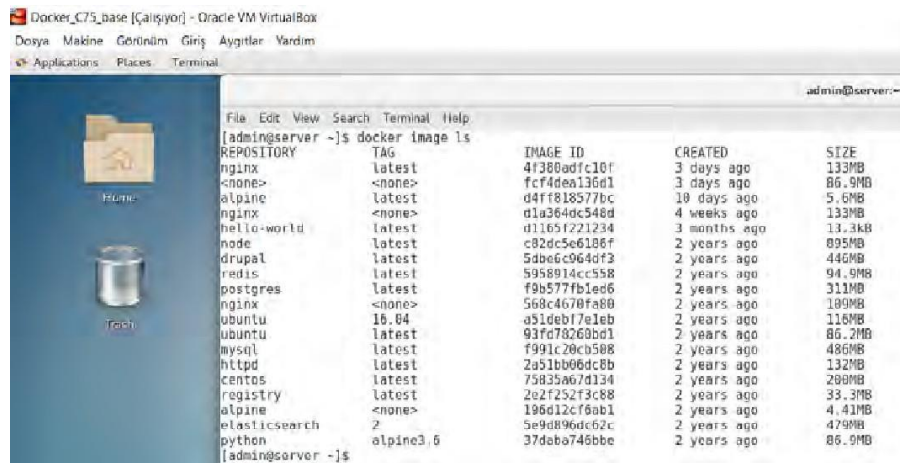
Docker Hub'tan hostumuza yaptığımız bu download, öncelikle imajın daha önceden yüklenip, yüklenmediğini kontrol eder. Eğer download yapılmışsa daha önceden, güncel imajın olup olmadığını kontrol edilir. Eğer imaj güncel değilse, en güncel olan sürüm docker hub'tan download edilir. Aşağıdaki alpine container imajını download ederken; karşılaştığımız durum, buna örnektir.



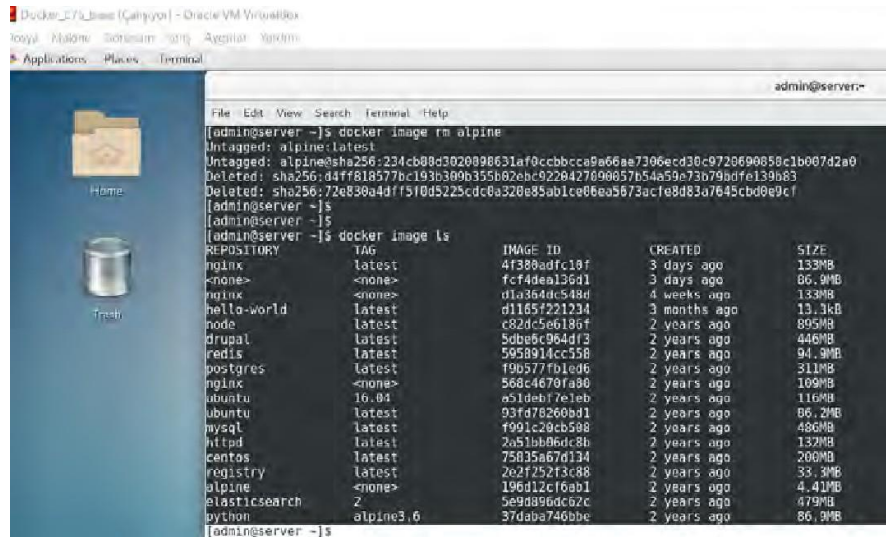
Her zaman her container imajının en güncel versiyonunu indirmek zorunda değilsiniz. İstedığınız versiyonu indirebilirsiniz. Nginx container'ın 1.21 ve 1.20 versiyonları aşağıda görüldüğü gibidir.



'docker image ls' komutumuyla lokalimdeki imajlarıma bakabilirim.



Bu imajlardan istediğinizi de 'docker image rm (imaj ismi)' komutuyla silebilirsiniz. Mesela alpine container'ini silelim ve tekrar ls diyelim. Sonuç aşağıdaki gibidir:



Alpine container'ini tekrar yüklemek istersem. 'docker image pull alpine' komutunu kullanmam lazım.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$
File Edit View Search Terminal Help
[admin@server ~]$
[admin@server ~]$
[admin@server ~]$ docker image pull alpine
Using default tag: latest
latest: Pulling from library/alpine
6843afab3874: Pull complete
Digest: sha256:234cb88d3020898631af0ccbcca9a66ae7306ecd30c9720690858c1b007d2a0
Status: Downloaded newer image for alpine:latest
[admin@server ~]$
[admin@server ~]$
```

'docker image ls' dediğimizde alpine'i tekrar görürüz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$
File Edit View Search Terminal Help
[admin@server ~]$
[admin@server ~]$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest             4f380adfc10f       3 days ago         133MB
<none>               <none>             fcf4deaf136d1      4 days ago         86.9MB
alpine               latest             d4ff818577bc       10 days ago        5.6MB
nginx                <none>             d1a364dc548d       4 weeks ago        133MB
hello-world          latest             d1165f221234       3 months ago       13.3kB
node                 latest             c82dc5e6186f       2 years ago        895MB
drupal               latest             5dbe6c964df3       2 years ago        446MB
redis                latest             5958914cc558       2 years ago        94.9MB
postgres             latest             f9b577fb1ed6       2 years ago        311MB
nginx                <none>             568c4670fa80       2 years ago        109MB
ubuntu               16.04             a51deb7e1eb        2 years ago        116MB
ubuntu               latest             93fd78260bd1       2 years ago        86.2MB
mysql                latest             f991c20cb508       2 years ago        486MB
httpd                latest             2a51bb06dc8b       2 years ago        132MB
centos               latest             75835a67d134       2 years ago        200MB
registry             latest             2e2f252f3c88       2 years ago        33.3MB
alpine               <none>             196d12cf6ab1       2 years ago        4.41MB
elasticsearch         2                 5e9d896dc62c       2 years ago        479MB
python               alpine3.6         37daba746bbe       2 years ago        86.9MB
[admin@server ~]$
```

Peki biz web servisimiz olan nginx'i nasıl çalıştırırız ?

Start A Simple Web Server

'docker container run --publish 80:80 nginx' komutunu çalıştıralım. Bu komutta --publish 80:80'deki ilk 80 docker host'umuzun transport layer'ın http protokolündeki default port'udur. İkinci 80 ise container'ın portudur. Yani docker host'un portunu container'ın portuna yönlendirmiş oluyoruz bu sayede.

'docker container run --publish 80:80 nginx' komutunu çalıştırdığımızda aşağıdaki hata mesajını alıyoruz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$
File Edit View Search Terminal Help
[admin@server ~]$ docker container run --publish 80:80 nginx
docker: Error response from daemon: driver failed programming external connectivity on endpoint goofy_goodall (392d8965af0a2884460b265cc1e07a384bc61092b669446c85610bd057867198): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use.
[admin@server ~]$
```


Çünkü şu an için http'de apache web servisi çalışmaktadır.

Lokal makinemizin IP'sine bakalım 'ip address' komutuyla.

```
Docker_C75_basë [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

admin@server:~$ docker container run --publish 80:80 nginx
docker: Error response from daemon: driver failed programming external connectivity on endpoint goofy_goodall (392d896stn tcp 0.0.0.0:80: bind: address already in use.
ERRO[0000] error waiting for container: context canceled
[admin@server ~]$
[admin@server ~]$
[admin@server ~]$ ip address
1: lo: <LOOPBACK,UP,LOWER UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d9:bf:0d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic enp0s3
        valid lft 65923sec preferred_lft 65923sec
    inet6 fe80::e739:ac27:25c8:c21e/64 scope link noprefixroute
        valid lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d4:cf:d5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.111/24 brd 192.168.56.255 scope global noprefixroute enp0s8
        valid lft forever preferred_lft forever
    inet6 fe80::b808:128e:32e2:fb8/64 scope link noprefixroute
        valid lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:16:bb:0b brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:16:bb:0b brd ff:ff:ff:ff:ff:ff
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:28:59:00:3d brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid lft forever preferred_lft forever
    inet6 fe80::42:28ff:fe59:3d/64 scope link
        valid lft forever preferred_lft forever
[admin@server ~]$
```

Şu an HTTP'de Apache web servisin aktif olarak çalıştığını görmek için 'systemctl status httpd.service' komutunu yazalım. Bunun için su – komutuyla ve password şifesiyle root hesabına geçmemiz lazım.

```
Docker_C75_basë [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

root@server:~# su -
Password:
Last login: Sat Jun 26 15:58:53 +03 2021 on pts/0
Last failed login: Sat Jun 26 16:15:08 +03 2021 on pts/0
There was 1 failed login attempt since the last successful login.
[root@server ~]# systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Sat 2021-06-26 15:34:38 +03; 4min ago
     Docs: man:apache2ctl(8)
           man:apache2ctl(8)
   Main PID: 13877 (httpd)
   Status: "Total requests: 10; Current requests/sec: 0; Current traffic: 0 B/sec"
     Tasks: 9
    Memory: 6.7M
   CGroup: /system.slice/httpd.service
           └─13877 /usr/sbin/httpd -DFOREGROUND
             └─13881 /usr/sbin/httpd -DFOREGROUND
               └─13882 /usr/sbin/httpd -DFOREGROUND
                 └─13883 /usr/sbin/httpd -DFOREGROUND
                   └─13884 /usr/sbin/httpd -DFOREGROUND
                     └─13885 /usr/sbin/httpd -DFOREGROUND
                       └─13886 /usr/sbin/httpd -DFOREGROUND
                         └─13887 /usr/sbin/httpd -DFOREGROUND
                           └─13888 /usr/sbin/httpd -DFOREGROUND

Jun 26 15:34:37 server: example.com systemd[1]: Starting The Apache HTTP Server...
Jun 26 15:34:38 server: example.com systemd[1]: AH00558: httpd: could not reliably determine the server's fully qualified domain name, using server.example.com. Set the 'ServerName' directive g...s this message
Jun 26 15:34:38 server: example.com systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
[root@server ~]#
```

Bizim NGINX container'ini çalıştırmamız için öncelikle bu Apache web servisini kapatmamız lazım. Kapatmadığımız sürece Apache servisi çalışmaya devam edecektir.

IP Adresimizin 192.168.56.111 olduğunu görmekteyiz. Yani biz linux veya windows host içinde web browser üzerinden bu IP adresini yazarsak apache web servisinin index'indeki metni görürüz.



Apache web servisi çalıştığı sürece ister Linux ister Windows Host'tan 192.168.56.11 nolu IP adresini herhangi bir web browser'da yazalım; karşımıza hep ' Knowledge is Power. Power is Knowledge Club' yazısı çıkacaktır. Bu yazı nasıl çıktı peki ? Çok basit index dosyasına ben yazdım. Birazdan size bunu nasıl değiştireceğimizi göstereceğim.

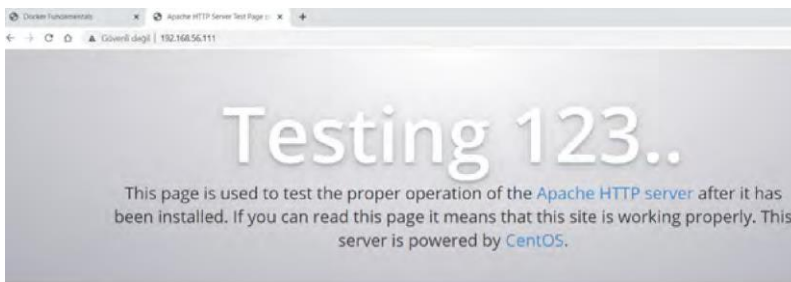
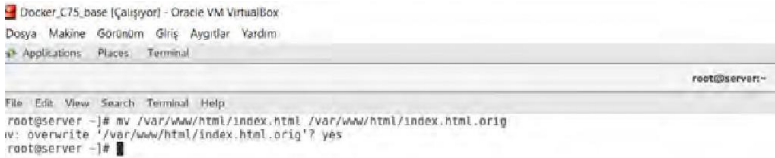
İlgili index metni 'cat /var/www/html/index.html' komutunu yazarak görüntüleyebiliriz root hesabından.



Bu metni değiştirmek istersem, 'cat > /var/www/html/index.html' komutunu yazıp, ilgili metni değiştirmem yeterlidir.

Bu dosya olmasaydı Apache Test sayfası gelmesi gerekirdi. Bu index'i orjinal dosya ile değiştirerek; bunu yapabiliriz.

Bunun için ilgili move komutumuz: 'mv /var/www/html/index.html /var/www/html/index.html.orig' komutuyla move ettim. Bu sayede ilgili index'i orjinal Apache test sayfası ile değiştirmiş oldum.



Bizim normalde default olarak görmek istediğimiz mesaj budur. Docker host üzerinde bir apache web servisini çalıştırıyorduk. Bunu nasıl kapatabiliriz ?

‘systemctl stop httpd.service’ komutu ile

Üstüne ek olarak komple disable etmek istersem ne yapmam lazım:

‘systemctl disable httpd.service’

Artık şimdi daha önceden çalıştırıp; hata aldığımız ‘docker container run --publish 80:80 nginx’ komutunu çalıştırabiliriz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

root@server:~
File Edit View Search Terminal Help
[root@server ~]# mv /var/www/html/index.html /var/www/html/index.html.orig
mv: overwrite '/var/www/html/index.html.orig'? yes
[root@server ~]# systemctl stop httpd.service
[root@server ~]# docker container run --publish 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2021/06/26 13:39:19 [notice] 1#1: using the "epoll" event method
2021/06/26 13:39:19 [notice] 1#1: nginx/1.21.0
2021/06/26 13:39:19 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2021/06/26 13:39:19 [notice] 1#1: OS: Linux 3.10.0-862.14.4.el7.x86_64
2021/06/26 13:39:19 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2021/06/26 13:39:19 [notice] 1#1: start worker processes
2021/06/26 13:39:19 [notice] 1#1: start worker process 32
2021/06/26 13:39:19 [notice] 1#1: start worker process 33
2021/06/26 13:39:19 [notice] 1#1: start worker process 34
2021/06/26 13:39:19 [notice] 1#1: start worker process 35
```

Artık NGINX container’imiz çalışmaktadır ve loglarımız da aktiftir. Ve logların sonuçlarını da görebilirim.

Logun nasıl değişeceğini görmek için öncelikle aynı adrese tekrar gidelim.



Şimdi log’un nasıl değiştiğine bakalım.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

root@server:~
File Edit View Search Terminal Help
[root@server ~]# mv /var/www/html/index.html /var/www/html/index.html.orig
mv: overwrite '/var/www/html/index.html.orig'? yes
[root@server ~]# systemctl stop httpd.service
[root@server ~]# docker container run --publish 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2021/06/26 13:39:19 [notice] 1#1: using the "epoll" event method
2021/06/26 13:39:19 [notice] 1#1: nginx/1.21.0
2021/06/26 13:39:19 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2021/06/26 13:39:19 [notice] 1#1: OS: Linux 3.10.0-862.14.4.el7.x86_64
2021/06/26 13:39:19 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2021/06/26 13:39:19 [notice] 1#1: start worker processes
2021/06/26 13:39:19 [notice] 1#1: start worker process 32
2021/06/26 13:39:19 [notice] 1#1: start worker process 33
2021/06/26 13:39:19 [notice] 1#1: start worker process 34
2021/06/26 13:39:19 [notice] 1#1: start worker process 35
192.168.56.1 - - [26/Jun/2021:13:42:02 +0000] "GET / HTTP/1.1" 200 612 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36"
```

Sayfayı refresh ettikçe logların değiştiğini aşağıdaki gibi görebilmekteyiz.

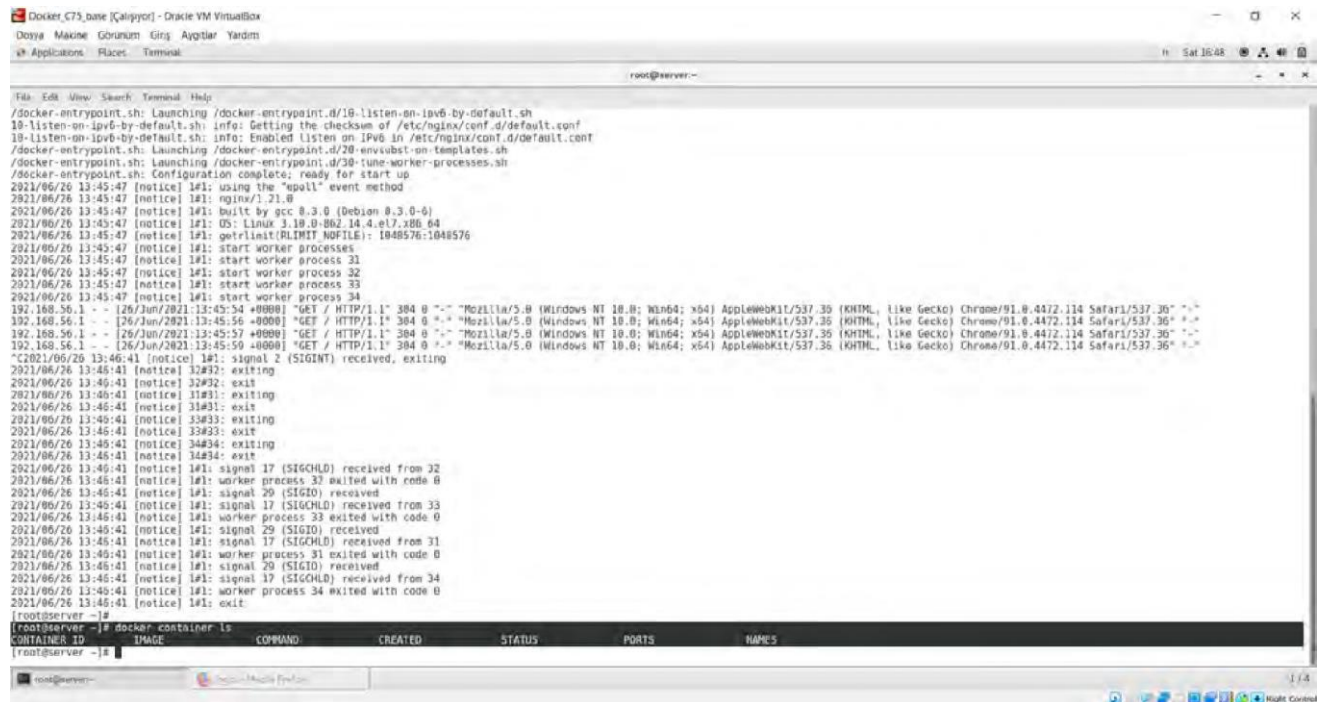
```
192.168.56.1 - - [26/Jun/2021:13:45:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
```

```
192.168.56.1 - - [26/Jun/2021:13:45:56 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
```

```
192.168.56.1 - - [26/Jun/2021:13:45:57 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
```

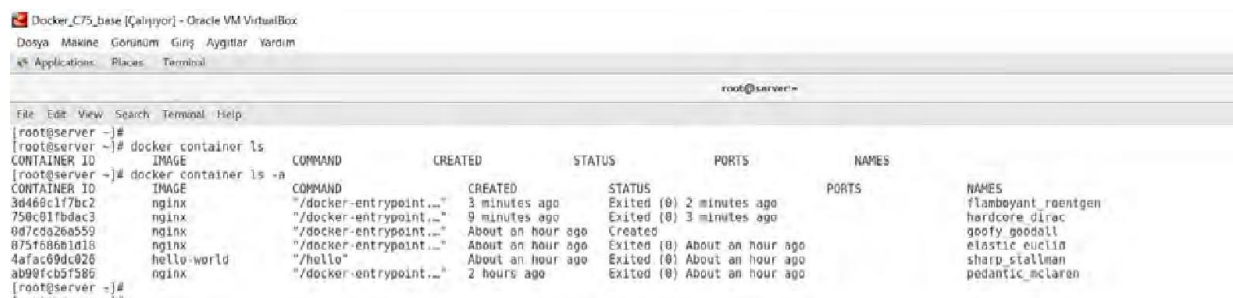
```
192.168.56.1 - - [26/Jun/2021:13:45:59 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
```

Cırtl+C ile çalışan container'ı kapatabilirsiniz.



```
File Edit View Search Terminal Help
[root@server ~]# docker-entrypoint.sh /etc/nginx/nginx.conf
192.168.56.1 - - [26/Jun/2021:13:45:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:56 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:57 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:59 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
2021/06/26 13:45:47 [notice] 1#1: using the "poll" event method
2021/06/26 13:45:47 [notice] 1#1: nginx/1.21.0
2021/06/26 13:45:47 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2021/06/26 13:45:47 [notice] 1#1: OS: Linux 3.10.0-882.14.4.el7.x86_64
2021/06/26 13:45:47 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2021/06/26 13:45:47 [notice] 1#1: start worker processes
2021/06/26 13:45:47 [notice] 1#1: start worker process 31
2021/06/26 13:45:47 [notice] 1#1: start worker process 32
2021/06/26 13:45:47 [notice] 1#1: start worker process 33
2021/06/26 13:45:47 [notice] 1#1: start worker process 34
192.168.56.1 - - [26/Jun/2021:13:45:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:56 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:57 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
192.168.56.1 - - [26/Jun/2021:13:45:59 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36" -
2021/06/26 13:46:41 [notice] 1#1: signal 2 (SIGINT) received, exiting
2021/06/26 13:46:41 [notice] 1#1: signal 17 (SIGCHLD) received from 32
2021/06/26 13:46:41 [notice] 1#1: worker process 32 exited with code 0
2021/06/26 13:46:41 [notice] 1#1: signal 29 (SIGIO) received
2021/06/26 13:46:41 [notice] 1#1: signal 17 (SIGCHLD) received from 33
2021/06/26 13:46:41 [notice] 1#1: worker process 33 exited with code 0
2021/06/26 13:46:41 [notice] 1#1: signal 29 (SIGIO) received
2021/06/26 13:46:41 [notice] 1#1: signal 17 (SIGCHLD) received from 31
2021/06/26 13:46:41 [notice] 1#1: worker process 31 exited with code 0
2021/06/26 13:46:41 [notice] 1#1: signal 29 (SIGIO) received
2021/06/26 13:46:41 [notice] 1#1: signal 17 (SIGCHLD) received from 34
2021/06/26 13:46:41 [notice] 1#1: worker process 34 exited with code 0
2021/06/26 13:46:41 [notice] 1#1: exit
[root@server ~]#
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@server ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3d46d1c17bc2        nginx              "/docker-entrypoint..." 3 minutes ago       Exited (0) 2 minutes ago      80/tcp             flamboyant-raentgen
758c317bdac3        nginx              "/docker-entrypoint..." 9 minutes ago       Exited (0) 3 minutes ago      80/tcp             hardcore-dirc
0d7c0a26a559        nginx              "/docker-entrypoint..." About an hour ago   Created                                80/tcp             goofy-goodall
075f686b1d18        nginx              "/docker-entrypoint..." About an hour ago   Exited (0) About an hour ago      80/tcp             elastic-euclid
4afac69dc026        hello-world        "/hello"            About an hour ago   Exited (0) About an hour ago      80/tcp             sharp-stallman
ab90fc057586        nginx              "/docker-entrypoint..." 2 hours ago         Exited (0) About an hour ago      80/tcp             pedantic-mclaren
[root@server ~]#
```

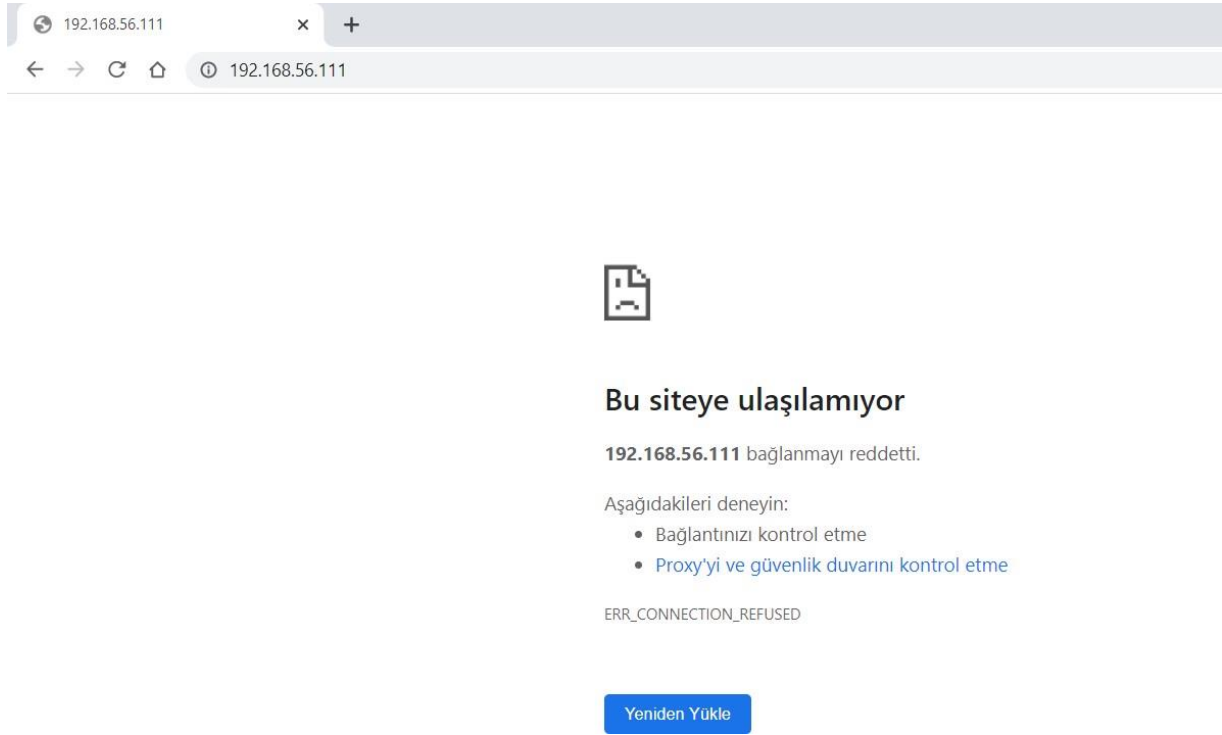
Artık ls içinde nginx container'ı gözükmemektedir. Ancak ls-a ile görebiliriz



```
File Edit View Search Terminal Help
[root@server ~]#
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@server ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3d46d1c17bc2        nginx              "/docker-entrypoint..." 3 minutes ago       Exited (0) 2 minutes ago      80/tcp             flamboyant-raentgen
758c317bdac3        nginx              "/docker-entrypoint..." 9 minutes ago       Exited (0) 3 minutes ago      80/tcp             hardcore-dirc
0d7c0a26a559        nginx              "/docker-entrypoint..." About an hour ago   Created                                80/tcp             goofy-goodall
075f686b1d18        nginx              "/docker-entrypoint..." About an hour ago   Exited (0) About an hour ago      80/tcp             elastic-euclid
4afac69dc026        hello-world        "/hello"            About an hour ago   Exited (0) About an hour ago      80/tcp             sharp-stallman
ab90fc057586        nginx              "/docker-entrypoint..." 2 hours ago         Exited (0) About an hour ago      80/tcp             pedantic-mclaren
[root@server ~]#
```

Çalıştırdığımız bütün nginx ve hello-world container'larını görebilmekteyiz.

Peki şu an aynı IP adresine gitmeyi denersem ne olur ? Şu an Nginx container'ı çalışmıyor.



Nginx'ı kapattığım için sayfaya ulaşamamaktadır.

Şimdi ise labımıza 'docker container run --publish 80:80 --detach --name webhost nginx' komutumuzla devam edelim.

Bu komut ne anlama gelmektedir ? publish 80:80'e kadar olan kısmını yukarıda açıklayıp; kullanmıştık. Detach demek, online kalmak için yani komut satırını almamız içindir. '--name webhost nginx' ise nginx container'ımıza webhost ismini veriyoruz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

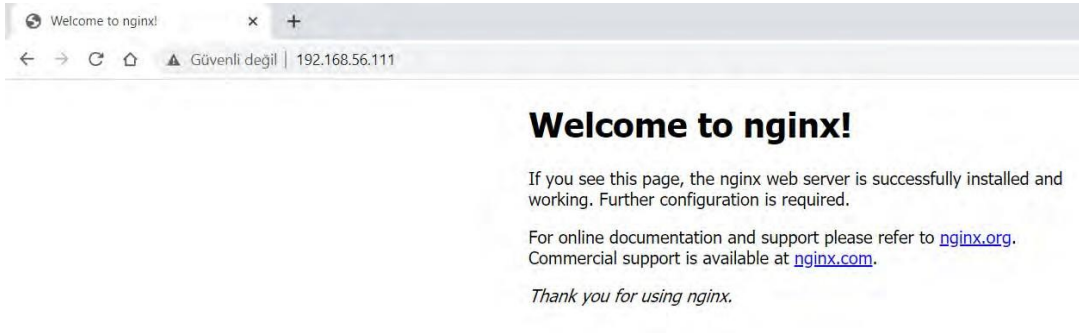
root@server:~#
root@server:~#
root@server:~#
root@server:~# clear
root@server:~# docker container run --publish 80:80 --detach --name webhost nginx
843c0d924aee0290dc47d31123f8aee7943cce2e3ca119382cf209a059056f9
root@server:~#
```

Şu an terminal'de komut girebiliyoruz. Peki bizim container'ımız çalışıyor mu ? Evet, Is diyip; görebiliriz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

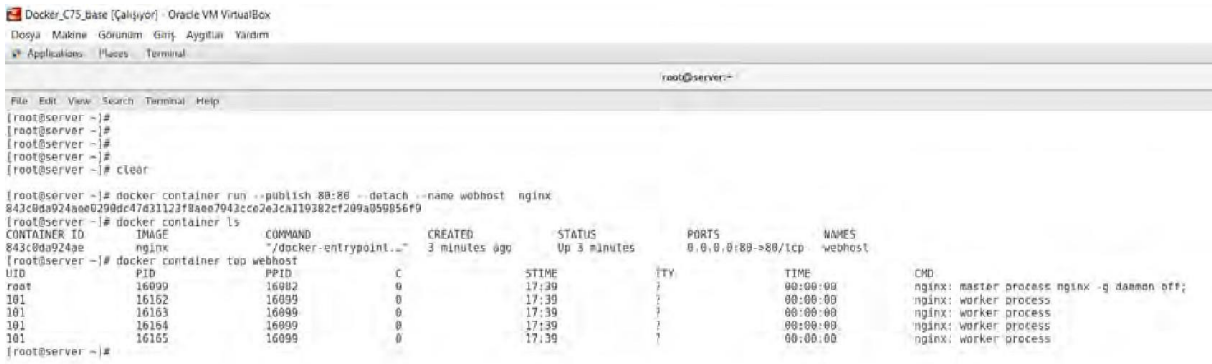
root@server:~#
root@server:~#
root@server:~#
root@server:~# clear
root@server:~# docker container run --publish 80:80 --detach --name webhost nginx
843c0d924aee0290dc47d31123f8aee7943cce2e3ca119382cf209a059056f9
root@server:~# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED        STATUS        PORTS                NAMES
843c0d924aee        nginx              "/docker-entrypoint..." 3 minutes ago  Up 3 minutes  0.0.0.0:80->80/tcp    webhost
root@server:~#
```

Hemen control edelim web sitesimize giderek:



Bildğiniz üzere Container'ın Hypervisor sanallaştırmasından pek çok farkı vardır. Herşeyin başında bir işletim sistemi yoktur. Container'ın detaylarına görmek için:

'docker container top webhost'



Container sanallaştırmasının, hypervisor sanallaştırmasından farkı sadece gerekli servislerin ve processlerin çalışmasıdır.

Peki biz şu an arka planda çalışmaya devam eden container'ımıza nasıl bağlanabiliriz ?

'docker container exec -it webhost bash' komutunu bu task için kullanabiliriz.

Bu sayede container'imizin üzerinde bash'ı kullanarak interaktif bir terminal açmış oluyoruz.

Container'imize bu şekilde bağlandıktan sonra container'imizden ilgili bilgileri alabiliriz. Mesela hostname bilgisini öğrenebiliriz.



Bu sayede container'imizin ID'sini görmüş olduk. Kendi makinemizin ID'sini ise host'umuzda yeni bir terminal sayfası açıp; admin hesabından root yazarak görebilirsiniz.



```

root@server:~# docker container exec -it webhost bash
root@843c0da924ae:/# hostname
843c0da924ae
root@843c0da924ae:/#

admin@server:~# hostname
server.example.com
admin@server:~#

```

Normalde admin hesabımızdan linux işletim sistemi üzerindeki processleri görebiliriz. Ama aynı şeyi container'da göremeyiz. Çünkü container tarafında root veya admin'de olduğu gibi bir işletim sistemimiz bulunmamaktadır.



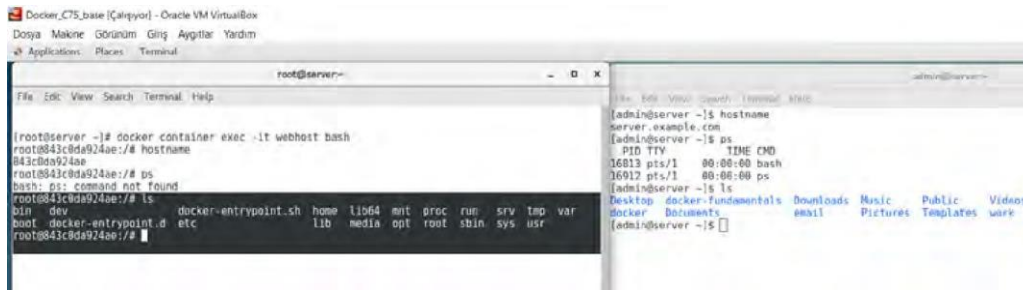
```

root@server:~# docker container exec -it webhost bash
root@843c0da924ae:/# hostname
843c0da924ae
root@843c0da924ae:/# ps
bash: ps: command not found
root@843c0da924ae:/#

admin@server:~# hostname
server.example.com
admin@server:~# ps
PID TTY TIME CMD
16613 pts/1 00:00:00 bash
16912 pts/1 00:00:00 ps
admin@server:~#

```

Container'da sadece nginx'i çalıştıracak kadar kaynağımız var. Bunu da kernel vasıtası ile docker engine bize bu ortamı sunuyor.



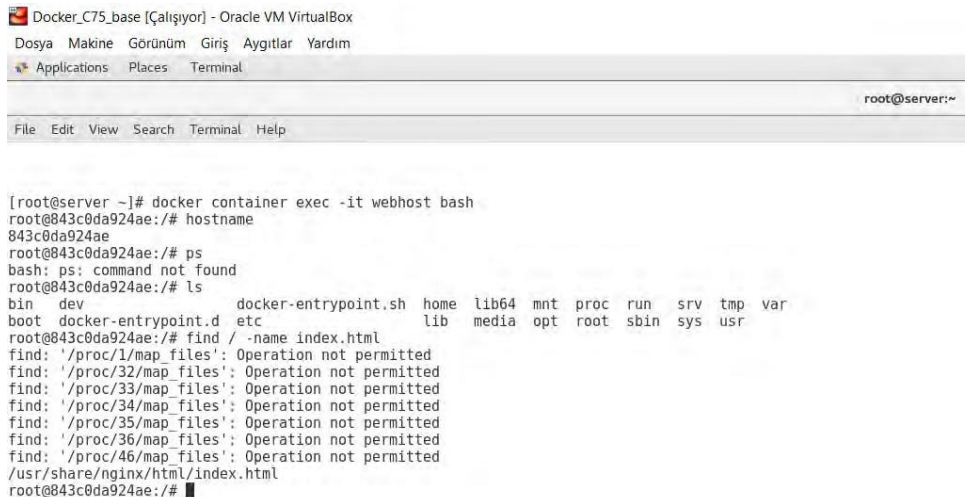
```

root@server:~# docker container exec -it webhost bash
root@843c0da924ae:/# hostname
843c0da924ae
root@843c0da924ae:/# ps
bash: ps: command not found
root@843c0da924ae:/# ls
bin  dev  docker-entrypoint.sh  home  lib64  mnt  proc  run  srv  tmp  var
boot  docker-entrypoint.d  etc  lib  media  opt  root  sbin  sys  usr

admin@server:~# hostname
server.example.com
admin@server:~# ps
PID TTY TIME CMD
16613 pts/1 00:00:00 bash
16912 pts/1 00:00:00 ps
admin@server:~# ls
Desktop  docker-fundamentals  Downloads  Music  Public  Videos
docker  Documents  email  Pictures  Templates  work
admin@server:~#

```

Sadece ls gibi basit komutları çalıştırabiliyoruz. Soldaki ekran bizim container'ımızken sağdaki taraf bizim için docker host'tur. Şu an biz sol ekranda container'ımızın içindeyiz. O zaman containerimizin içindeki index dosyasını bulabiliriz. Peki bunu nasıl yapabiliriz ? 'find / -name index.html' komutunu kullanarak; bu taskı gerçekleştirebiliriz.



```

root@server:~# docker container exec -it webhost bash
root@843c0da924ae:/# hostname
843c0da924ae
root@843c0da924ae:/# ps
bash: ps: command not found
root@843c0da924ae:/# ls
bin  dev  docker-entrypoint.sh  home  lib64  mnt  proc  run  srv  tmp  var
boot  docker-entrypoint.d  etc  lib  media  opt  root  sbin  sys  usr

root@843c0da924ae:/# find / -name index.html
find: '/proc/1/map_files': Operation not permitted
find: '/proc/32/map_files': Operation not permitted
find: '/proc/33/map_files': Operation not permitted
find: '/proc/34/map_files': Operation not permitted
find: '/proc/35/map_files': Operation not permitted
find: '/proc/36/map_files': Operation not permitted
find: '/proc/46/map_files': Operation not permitted
/usr/share/nginx/html/index.html
root@843c0da924ae:/#

```


Yani bizim index dosya uzantımızın olduğu yer: /usr/share/nginx/html/index.html. Bunu cat komutumuzla açarsak; aşağıdaki bilgilere ulaşırız.

```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

root@server:~

File Edit View Search Terminal Help

[root@server ~]# docker container exec -it webhost bash
root@843c0da924ae:/# hostname
343c0da924ae
root@843c0da924ae:/# ps
bash: ps: command not found
root@843c0da924ae:/# ls
bin dev docker-entrypoint.sh home lib64 mnt proc run srv tmp var
root docker-entrypoint.d etc lib media opt root sbin sys usr
root@843c0da924ae:/# find / -name index.html
find: '/proc/1/map_files': Operation not permitted
find: '/proc/32/map_files': Operation not permitted
find: '/proc/33/map_files': Operation not permitted
find: '/proc/34/map_files': Operation not permitted
find: '/proc/35/map_files': Operation not permitted
find: '/proc/36/map_files': Operation not permitted
find: '/proc/46/map_files': Operation not permitted
/usr/share/nginx/html/index.html
root@843c0da924ae:/# cat /usr/share/nginx/html/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@843c0da924ae:/#

```

Doğal olarak buradaki yazı daha önceden yukarıda paylaştığımız nginx web sayfasındaki yazı ile aynıdır. Çünkü o yazı buradan gelmektedir.

Peki ben bunu overwrite yapabilir miyim ? Evet tabi ki de cat > komutuyla yapabiliriz.

```

Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

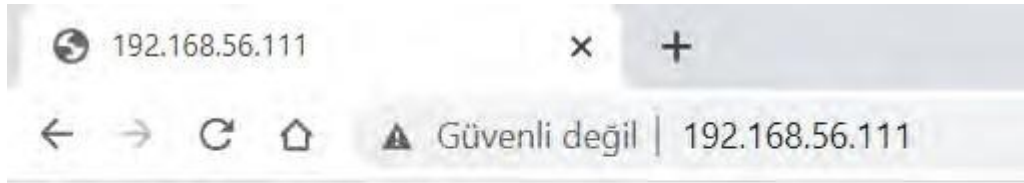
root@server:~

File Edit View Search Terminal Help

root@843c0da924ae:/# cat > /usr/share/nginx/html/index.html
May The Force Be With You
root@843c0da924ae:/#
root@843c0da924ae:/#

```

Şimdi tekrar nginx sayfasına bakarsak; sayfadaki mesajın bu şekilde değiştiğini göreceğiz.



May The Force Be With You

Bu özelliği sağlayan şey: Docker Internals'daki namespace fonksiyonu'dur.

Exit diyerek container'ımızdan çıkabiliriz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal
root@server:~
File Edit View Search Terminal Help
root@843c0da924ae:/# exit
exit
[root@server ~]#
```

Nginx container'ımız halen çalışıyor mu ? Is ile control edelim

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal
root@server:~
File Edit View Search Terminal Help
root@843c0da924ae:/# exit
exit
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
843c0da924ae        nginx              "/docker-entrypoint..." 2 hours ago        Up 2 hours          0.0.0.0:80->80/tcp   webhost
[root@server ~]#
```

Stop diyerek container'imizi durdurabiliriz. Rm diyerek ise silebiliriz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal
root@server:~
File Edit View Search Terminal Help
root@843c0da924ae:/# exit
exit
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
3c0da924ae         nginx              "/docker-entrypoint..." 2 hours ago        Up 2 hours          0.0.0.0:80->80/tcp   webhost
[root@server ~]#
[root@server ~]# docker container stop webhost
webhost
[root@server ~]#
```

Stop ettikten sonra ls dersek; container'ımızın çalışmadığını göreceğiz.

```
Docker_C75_base [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
Applications Places Terminal

root@server:~

File Edit View Search Terminal Help
root@843c0da924ae:/# exit
exit
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
843c0da924ae        nginx              "/docker-entrypoint..." 2 hours ago         Up 2 hours          0.0.0.0:80->80/tcp  webhost
[root@server ~]#
[root@server ~]#
[root@server ~]# docker container stop webhost
webhost
[root@server ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@server ~]#
```

Bu durumda artık web sayfamız çalışmayacaktır.



Bu siteye ulaşılamıyor

192.168.56.111 bağlanmayı reddetti.

Aşağıdakileri deneyin:

- Bağlantınızı kontrol etme
- Proxy'yi ve güvenlik duvarını kontrol etme

ERR_CONNECTION_REFUSED

Yeniden Yükle

Özetle gördüğümüz üzere container'lar processlerden ibarettir ve vm'ler gibi çalışmamaktadır. Bu yüzden container'ları birer application olarak düşünmek daha mantıklıdır.

Peki bu kullandığımız imajlar ilgili vendorlar tarafından nasıl yaratılır Docker Hub'ta ? İmajlar Dockerfile üzerinden yaratılır. Bu sayede dockerfile'dan nasıl imaj oluşturulduğuna detaylı bir şekilde bakabilir hatta imajı bu şekilde değiştirebilirsiniz bile.

Docker Hub'tan nginx container'ını seçip; en son versiyonun üstüne tıkladığınızda aşağıdaki detayları görebilirsiniz: