



NECMETTİN ERBAKAN
ÜNİVERSİTESİ

Mühendislik Ve Mimarlık Fakültesi

Bilgisayar Mühendisliği Bölümü

**Bilgisayar Mühendisliği
Uygulama Geliştirme Projesi**

Uygulama Konusu
Docker Uygulama Geliştirme

Öğrenci Bilgileri	
Öğr. No	
Ad Soyad	Büşra Yenidoğan

Mehmet HACİBEYOĞLU
Öğretim Görevlisi

Haziran
3 2017
Konya

İçindekiler

Önsöz	
Docker'ın Doğuşu	1
Container ve VMS	2
Container'ların Ne ve Ne için	4
Linux'a Docker Kurulumu	5
İlk Image Çalışması	6
Basit Komutlar	7
Basit Oyun Uygulaması	10
Dockerfile ile Image Yapımı	11
Terminoloji	12
Redis'i Resmi Image ile Koşturmak	13
Docker Mimarisi	14
Docker Komutları	15
Basit Web Uygulaması-Nginx	16
Koşturma Komutları	19
Python ile Web Server Uygulaması	20
Docker-Compose	22
Docker-Compose Kurulumu	23
Docker-Compose Uygulama	25

Basit Docker Ugulamaları	29
a.1 apache2	29
a.2 apache2&&php	31
a.3 mongoDB	33
a.4 mysqlserver	34
a.5 sshd	35
Kaynaklar	37

ONSOZ

2017 yılının 2. döneminde Uygulama Projesi olarak alınan bu proje temel anlatımlarla kullanıcıya Docker teknolojisi hakkında bilgi vermeyi amaçlamaktadır. Raporda daha çok uygulama ağırlıklı ilerlemesi ile de "tutorial" olarak kullanılması hedeflenmektedir.

Yardımlarından dolayı Sayın Hocam YRD. DOÇ. DR. Mehmet Hacıbeyoğuna teşekkürlerimi borç bilirim.

Docker'ın Doğuşu

Docker dünyada en çok kullanılan yazılım konteynerlaştırma platformudur.Konteynerlaştırma kavramı ise matruşka bebekleri gibi iç içe koyulan konteynerlardan oluşmaktadır. Bu kavram ilk olarak 2008 yılında Linux tabanlı sistemlere eklenen Linux Containers(LXC) teknolojisi ile yaşam buldu.LXC'nin görevi Linux'ta aynı işletim sistemi içerisinde birbirinden izole olmuş şekilde çalışan Containerlar sağlamaktır. LXC, Container içerisinde bulunan processlere işletim sistemi üzerinde sadece kendisi koşturmuş izlenimi vererek aynı host üzerinde sanallaştırma ortamı sağlamış olur. LXC, kurulduğu hostdaki özellikleri aynı şekilde Container'larada aktarmış olur. Yani o işletim sistemine ait olan dosyalama sistemi, ortam değişkenleri ve ya fonksiyonlar gibi özellikler Container'da sahip olmuş olur. Bu durumda her bir Containerı birbirinden ayrı bir host yani işletim sistemi gibi düşünebiliriz o halde Container'lar birbirleri ile istemedikleri müddetçe iletişim halinde de olmazlar. Ancak bu teknolojinin(LXC) bazı yetersizlikleri 2013 yılında Docker'ın doğuşunun sebebi oldu herşeyden önce farklı olarak Docker iki farklı bileşene sahiptir: Docker Engine, containerların oluşturulması ve çalıştırılması; Docker Hub, dağıtık containerlar için bulut hizmeti. Docker Engine, çalışan containerlar için hızlı ve kullanışlı arayüz ortamı sağlar.Ancak önceden,Container'ların koşturması için önemli teknolojileri kullanmamız gerekiyordu.Docker Hub, imajeleri indirmemiz için özel numaralandırma ortamı sağlıyor. Böylece bütün kullanıcılar imajelara hızlı ve public bir şekilde erişebiliyor. Dockerın dahada ileri bir özelliği ise, geliştirici araçları olan:Swarm, Kitematic, Machine gibi araçları bünyesinde bulundurması.İlerleyen sayfalarda daha detaylı açıklıyor olacağız. ¹

Container ve VMS

Hypervisor(Sanallařtırma Platformları) bulundukları fiziksel makina ierisinde farklı iřletim sistemle-ri kurulmasına olanak saėar. Avantajı ; Kendilerine ayrılan disk blmlerine kurulması ve hepsinin aynı iřletim sistemi ierinde bulunmasıdır. Hypervsior da her hostun kendine ait iřletim sistemi vardır.

LXC'de ise Container'lar hostun iřletim sistemini kullanmak zorundaydılar. Hypervsior'da her hostun bakımı ayrı ayrı yapılmalıdır. LXC'de ise bir iřletim sisteminin yapılması yeterlidir.

Bazı nemli farklılıkları diyagramdan anlatmaya alıřalım :

Figure 1-1 : VMS deki  ayrı uygulama host zerinde kořmaktadır. Hypervsior iin VMs oluřurmak ve alıřtırmak kontrol eriřimi iin gereklidir. OS ve hardware gerektiėi zaman system call'ları yorumlamalıdır. Her VM'nin kopyası uygulamaların alıřtırılması ve ktphane desteėi iin gereklidir.

Figure 1-2 : VM'den farklı olarak, Host kernelini kořan container'lar ile paylařmaktadır. Bu řu demektir: her container, hostun kernelinde kořmaktadır. Container'ları durdurmak yada bařatmak VM gibi DockerEngine'nin grevidir. Eėer birden fazla aynı container var ise aynı ktphaneyi paylařabilirler. Ancak container ierisindeki processler istenmediėi srece baėantılı deėildir. Sonu olarak VMs ve containerlar aynı host zerinde farklı uygulamaları kullanabilir.²

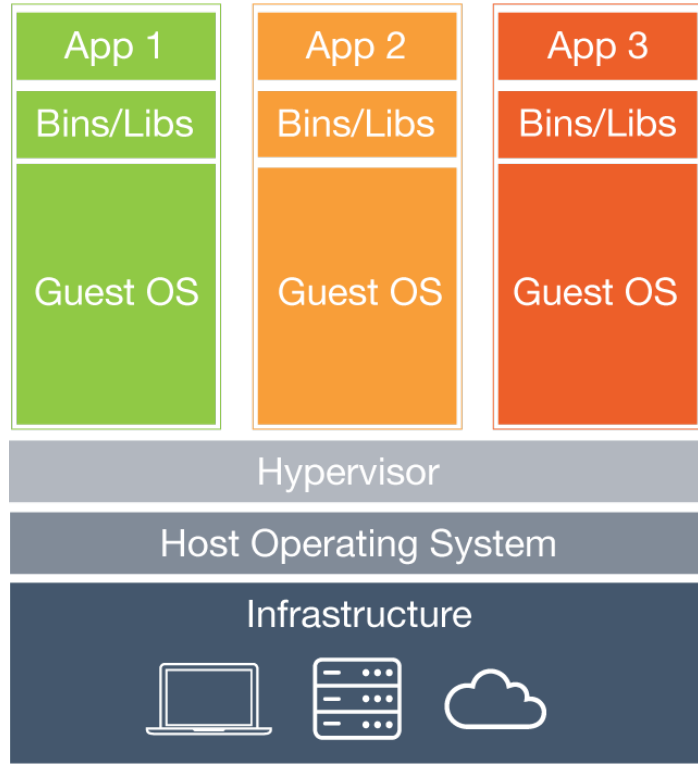


Figure 1-1 Üç VMs bir Host üzerinde koşuyor

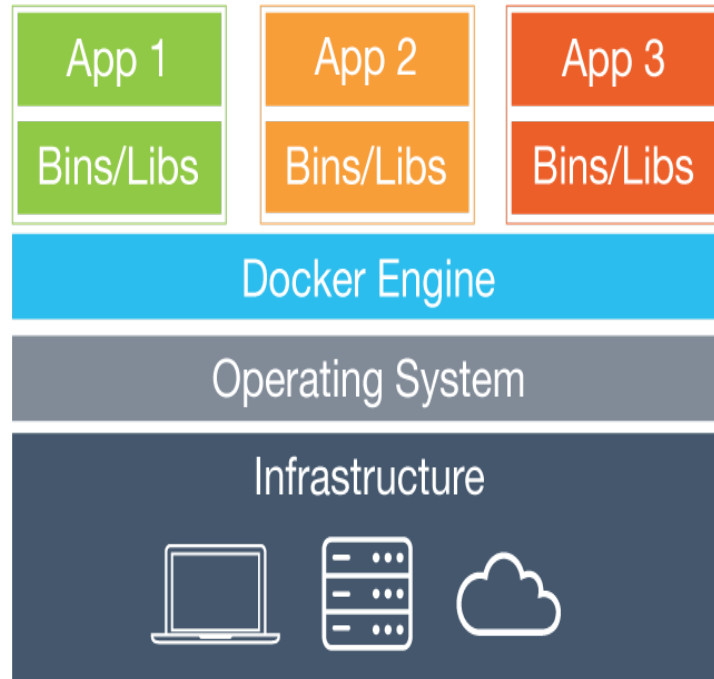


Figure 1-2 Üç Container bir Host üzerinde koşuyor³

Container'lar Ne ve Niçin

Container, yazılım geliştirmemiz, dağıtmamız ve çalıştırmamızda temel olarak değiştirmektedir.

Geliştiriciler uygulamalarını kendi hostlarında çalıştırdıkları zaman herhangi bir dış etkenle değişikliğe uğramadan çalışabilir. Container'lar uygulamaların bağımlılıklarını kapsülleyebilir ve uygulamayı kullandığımızda OS'dan izole tutabilir. Ancak Containerlar VMs aksine bazı bir kaç imkansız avantajlara sahiptir bunlar :

- Container'lar kaynaklarını OS ile paylaşırlar.
- Container'lar taşınabilirler.
- Kullanıcılar karmaşık uygulamaları çok zaman harcamadan indirip kurabilirler.
- Aynı zamanda bir çok container configure edilebilir.

Linux'a Docker Kurulumu

Docker kurmak için gerekli olan çok fazla bir şey yoktur ancak kernel versiyonu 3.10 ve ya üzeri olmalıdır. Terminale `uname -r` yazarak kontrol edebilirsiniz.

Kurulum :

```
$ curl https://get.docker.com > /tmp/install.sh
```

```
$ cat /tmp/install.sh
```

```
$ chmod +x /tmp/install.sh
```

```
$ /tmp/install.sh
```

Kurulum kontrolü :

```
$ sudo docker version
```

```
busra@ereborlugimli:~/Desktop$ sudo docker version
[sudo] password for busra:
Client:
 Version:      17.05.0-ce
 API version:  1.29
 Go version:   go1.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:15:36 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.05.0-ce
 API version:  1.29 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:15:36 2017
 OS/Arch:      linux/amd64
 Experimental: false
busra@ereborlugimli:~/Desktop$
```

İlk image Çalışması

Docker'ı doğru kurduktan sonra, deniyoruz :

```
ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ sudo docker run ubuntu echo "Hello World"
Hello World
busra@ereborlugimli:~/Desktop$
```

Daha önceden ubuntu image'ni pull ettiğimden dolayı yükleme aşama kısmını es geçip direk

"Hello World" yazdırmış olduk.

Container'ın shell'ine geçmek istersek :

```
Terminal File Edit View Search Terminal Help
busra@ereborlugimli:~/Desktop$ sudo docker run -i -t ubuntu /bin/bash
root@9b748e863418:/# echo "Hello from Container"
Hello from Container
root@9b748e863418:/# exit
exit
busra@ereborlugimli:~/Desktop$
```

Basit Komutlar

Biraz bilgi edinmek ve anlayabilmek adına çeşitli denemeler yapalım. Öncelikle yeni bir container başlatalım ancak -h flagi ile yeni bir hostname vererek :

```
ONTAINER: /  
busra@ereborlugimli:~/Desktop$ sudo docker run -h CONTAINER -i -t ubuntu /bin/bash  
root@CONTAINER:/#
```

Daha sonra /bin dizinini /basket dizinine taşımaya çalışalım.

```
busra@ereborlugimli:~/Desktop$ sudo docker run -h CONTAINER -i -t ubuntu /bin/bash  
root@CONTAINER:/# mv /bin /basket  
root@CONTAINER:/# ls  
bash: ls: command not found  
root@CONTAINER:/#
```

Dizin hakkında "ls" komutu ile bilgi edinmek istediğimiz zaman böyle bir durumla karşılaştık. O zaman container hakkında daha fazla bilgi edinmek için açmış olduğumuz oturumdaki terminalden ayrılarak yeni terminale geçip image hakkında bilgi almamıza sağlayan "docker ps" komutunu çalıştırıyoruz :

```
ereborlugimli: ~/Desktop  
root@CONTAINER: /  
busra@ereborlugimli:~/Desktop$ sudo docker ps  
[sudo] password for busra:  
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES  
41b5b91ffc13       ubuntu             "/bin/bash"         6 minutes ago       Up 6 minutes                   mystifying_chandrasekhar
```

Bu komut bize container hakkında belli başlı bazı bilgileri veriyor. Ancak Container hakkında bu bilgiler bizim için yeterli değil. Bu yüzden daha detaylı bir bilgi için container'ın adı ile "docker inspect" komutunu kullanarak daha fazla bilgiye ulaşıyoruz.

```
ereborlugimli: ~/Desktop
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker inspect mystifying_chandrasekhar
[
  {
    "Id": "41b5b91ffc139158481c79be1b12aa5e3c5fd9c63c52897e7c9c6f923074582f",
    "Created": "2017-06-05T15:22:13.649504337Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 15160,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2017-06-05T15:22:14.759578011Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    }
  },
]
```

Çıktının bir kısmı karşımızda olsada, bilgileri ayırtırmak işimizi daha da kolaylaştıracaktır. Örneğin ;
grep komutunu kullanarak IP Adresini öğrenebiliriz.

```
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker inspect mystifying_chandrasekhar | grep IPAddress
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",
busra@ereborlugimli:~/Desktop$
```

```
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker inspect --format [{.NetworkSettings.IPAddress}] mystifying_chandrasekhar
172.17.0.2
busra@ereborlugimli:~/Desktop$
```

İki çıktıda bize IP adresi hakkında bilgi vericektir. Ancak bizim Container'daki yaptığımız değişikliği gösteren komutumuz :⁸

```
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker diff mystifying_chandrasekhar
C /.wh..wh.plnk
A /.wh..wh.plnk/129.5512090
D /bin
A /basket
A /basket/bash
A /basket/cat
A /basket/chgrp
A /basket/chmod
A /basket/chown
A /basket/cp
```

Containerda yapılan değişiklikleri görebiliyoruz. Şimdi de en son Containerda yapmış olduğumuz işlemlerin çıktısını yani loglarını görmek istiyoruz. Bunun için “docker logs” komutunu kullanıyoruz :

```
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker logs mystifying_chandrasekhar
root@CONTAINER:/# mv /bin /basket
root@CONTAINER:/# ls
bash: ls: command not found
busra@ereborlugimli:~/Desktop$
```

Container'dan çıkıyoruz :

```
root@CONTAINER: /
busra@ereborlugimli:~/Desktop$ sudo docker run -h CONTAINER -i -t ubuntu /bin/bash
root@CONTAINER:/# exit
exit
busra@ereborlugimli:~/Desktop$
```

En sonunda da Container'ı siliyoruz :

Gördüğümüz gibi önce koşan durumdaki Container'ı durdumamızı istediği için önce durduruyoruz. Daha sonra silme işlemini gerçekleştiriyoruz.⁹

Basit Oyun Uygulaması

Hostname'ı cowsay olan, Cowsay adında ki bir uygulamayı ubuntu Container üzerine kurup

çalıştıralım.Container'ın shell'indeyken oyunu yüklemek için bu komutu giriyoruz :

```
busra@ereborlugimli: ~/Desktop
root@cowsay:/# apt-get install -y cowsay fortune
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'fortune-mod' instead of 'fortune'
The following additional packages will be installed:
  cowsay-off fortunes-min ifupdown iproute2 isc-dhcp-client isc-dhcp-d
  libperl5.22 librecode0 libtext-charwidth-perl libxtables11 netbase p
Suggested packages:
  filters fortunes x11-utils bsdmainutils ppp rdnssd iproute2-doc resco
  libterm-readline-gnu-perl | libterm-readline-perl-perl make
The following NEW packages will be installed:
  cowsay cowsay-off fortune-mod fortunes-min ifupdown iproute2 isc-dhc
  libisc-export160 libmn10 libperl5.22 librecode0 libtext-charwidth-pe
0 upgraded, 21 newly installed, 0 to remove and 19 not upgraded.
Need to get 8732 kB of archives.
```

Oyunu çalıştırmak için :

```
busra@ereborlugimli: ~/Desktop
root@cowsay:/# /usr/games/fortune | /usr/games/cowsay

/ You're being followed. Cut out the \
\ hanky-panky for a few days. /
-----
      \      ^__^
       (oo)\_______
            (_____)
               ||----w |
               ||     ||

root@cowsay:/#
```

10

Dockerfile ile Image Yapımı

Dockerfile normal text dosyasıdır. Dockerfile kullanarak Docker image'ını oluştururuz. Öncelikle

yeni bir klasör ve içine dosya oluşturuyoruz :

```
$ mkdir cowsay
```

```
$ cd cowsay
```

```
$ touch Dockerfile
```

```
$ nano Dockerfile :
```

```
FROM ubuntu:latest
```

```
MAINTAINER Büşra Yenidoğan <yenidoganbusra@gmail.com>
```

```
RUN apt-get update && apt-get install -y cowsay fortune
```

```
COPY entrypoint.sh /
```

```
ENTRYPOINT ["/entrypoint.sh"]
```

Şimdi "docker build" komutu ile aynı klasörün içinde image'ı oluşturabiliriz.

```
busra@ereborlugimli:~/Desktop/Docker/cowsay$ sudo docker build -t test/cowsay-dockerfile .
Sending build context to Docker daemon 6.144kB
Step 1/5 : FROM ubuntu:latest
--> 0ef2e08ed3fa
Step 2/5 : MAINTAINER Büşra Yenidoğan <yenidoganbusra@gmail.com>
--> Using cache
--> b5967f84199e
Step 3/5 : RUN apt-get update && apt-get install -y cowsay fortune
--> Using cache
--> 65219e0bf2ce
Step 4/5 : COPY entrypoint.sh /
--> Using cache
--> 29aa25f8ea0c
Step 5/5 : ENTRYPOINT /entrypoint.sh
--> Using cache
--> c88e5fbb1783
Successfully built c88e5fbb1783
Successfully tagged test/cowsay-dockerfile:latest
busra@ereborlugimli:~/Desktop/Docker/cowsay$
```

Dockerfile içine yazdığımız komutları tek tek açıklayacak olursak:

FROM ile işletim sisteminin(ubuntu'nun) en son(latest) güncel hali olacağını

RUN güncelleme ve kurulum için yapılması gerekenleri

COPY oluşturduğumuz dosyayı referans aldı

ENTRYPOINT oluşturduğumuz dosyayı kurduktan sonra varsayılan olarak çalıştırdı.

Terminoloji

Image : Docker Daemonla çalışan, Container'ların baz alacağı işletim sistemi veya uygulama olan metin bazlı(Dockerfile) binarydir.

Container : Docker Daemon tarafından Linux çekirdeği içerisinde birbirinden izole olarak çalıştırılan süreçlerin her birine verilen isimdir.

Registry : Image dağıtmak ve ev sahipliği yapmak için verilen hizmettir. Varsayılan olarak Docker Hub kullanılır.

Repository : İlgili imajların oluşturduğu yapıdır.

Tag : Repositoryde ki image'ın alfanümerik tanımlamasıdır.

Redis'i Resmi Image Kullanarak Koşurmak

Docker Hub'ta bulunan en son güncel Redis uygulamasını indirip kullanacağız.

```
Terminal File Edit View Search Terminal Help
busra@ereborlugimli:~$ sudo docker pull redis
[sudo] password for busra:
Using default tag: latest
latest: Pulling from library/redis
10a267c67f42: Pull complete
5b690bc4eaa6: Pull complete
4cdd94354d2a: Pull complete
71c1f30d820f: Pull complete
c54584150374: Pull complete
d1f9221193a6: Pull complete
d45bc46b48e4: Pull complete
Digest: sha256:548a75066f3f280eb017a6ccda34c561ccf4f25459ef8e36d6ea582b6af1decf
Status: Downloaded newer image for redis:latest
busra@ereborlugimli:~$
```

Redis container'ını başlatıcaz fakat bu sefer -d argümanını da kullanacağız. Böylelikle Docker

arka planda çalışıyor olacak. Çalıştırdığımız zaman Redis'in ID çıktısını almış olacağız.

```
Terminal File Edit View Search Terminal Help
busra@ereborlugimli:~$ sudo docker run --name myredis -d redis
27844fd9faae266947240db0706bc562fc4c00e4cd45e44f517e8efbd0f4d058
busra@ereborlugimli:~$
```

Şimdi Container'ın shell'ine geçeceğiz. Ardından database ile bir şekilde bağlantı kuracağız.

bunun için sadece "redis-cli" komutunu kullanıyor olup uygulamaya geçmiş olacağız.

```
Terminal File Edit View Search Terminal Help
busra@ereborlugimli:~$ sudo docker run --rm -it --link myredis:redis redis /bin/bash
root@38721149b93d:/data# redis-cli -h redis -p 6379
redis:6379> ping
PONG
redis:6379> set "abc" 123
OK
redis:6379> get "abc"
"123"
redis:6379> exit
root@38721149b93d:/data# exit
exit
busra@ereborlugimli:~$
```

Docker Mimarisi

Docker Daemon : Linux Kernel ile direkt olarak iletişim halindedir.

Docker CLI : Docker Daemon ile iletişim kurmamızı sağlar.

Her ikisinde Linux üzerinde TCP ile haberleşmektedirler. Docker CLI'dan gelen komular

TCP ile Engine'e iletilir ardından işlenip cevap olarak gönderilir. Linux üzerinde sistem böyle yürümektedir. Windows ve Mac OSX de farklıdır.

Swarm : Docker'ın kümeleme çözümüdür. Swarm sayesinde bir kaç Docker Host'u grup yapabilir.

Kullanıcılara kaynaklara erişim izni sağlayabilirsiniz.

Compose : Geliştiriciler için geliştirilen bu araç uygulamaların oluşturulmasından ve yapılandırılmasından sorumludur.

Machine : Docker Host'ların localine ya da uzaktan kurulum ve konfigürasyonunu yapar.

Docker Hosting : Bir çok bilindik bulut ortamına hizmet sağlar. Amazon, Google ve Digital Ocean vb.

Image Layers : Process ve ya servisi başlattığınız andan itibaren koşmasını istiyorsanız

ENTRYPOINT ve ya CMD komutlarını kullanmalısınız.

Docker Komutları

ADD : Oluşturacağımız Image'a dosya sisteminden ve ya interetten dosya/klasör eklemek

için kullanılır.

CMD : Sürekli çalışacak olan varsayılan komuylar belirlenir.

COPY : Sadece Host üzerindeki dosya/klasör'leri kopyalayabilir.

ENTRYPOINT : Image'a çalıştırılabilir dosya özelliği verir.

ENV : Değişken oluşturmamızı sağlar.

EXPOSE : Docker bünyesinde barındırdığı Networking-Modülü ile Container'lar ve Host'lar arasında iletişim kurar. Bu iletişimde TCP/UDP paketleri ile sağlar. Başka Container'lar ile iletişim kurmak isteyen Container'lar EXPOSE komutunu kullanır.

FROM : Referans alırız.

MAINTAINER : Kullanıcı iletişim bilgilerini ayarlar.

RUN : Yapılandırma sırasında çalışacak olan komutları koştururuz.

VOLUME : Voller dosyalar yada dizileri direk olarak hosta bağlar yani union dosya sistemine. Bunun anlamı Hostumuzun dosyalama sistemi ile direkt olarak diğer containerlar ile yaptığımız tüm değişiklikleri paylaşabiliriz. Bunun için "volume" instructionunu kullanmamız lazım.

WORKDIR : Kendisinden sonra gelen instructionu etiketliyerek path oluşturur.¹⁵

Basit Web Uygulaması – Nginx

Container'ın içinde koştan bir web server uygulaması yapacağız. Bu uygulama ile Containerdan internet dünyasına çıkış sağlamış olacağız.

```
ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ sudo docker run -d -p 8000:80 nginx
[sudo] password for busra:
82db95e2cc54ee5c7e86fbf595221028f092d21d356dd66ec7f2080c7a4b703d
docker: Error response from daemon: driver failed programming external
9c8465b9415d598c1b21d504f299f7dcfe2574): Bind for 0.0.0.0:8000 failed
busra@ereborlugimli:~/Desktop$
```

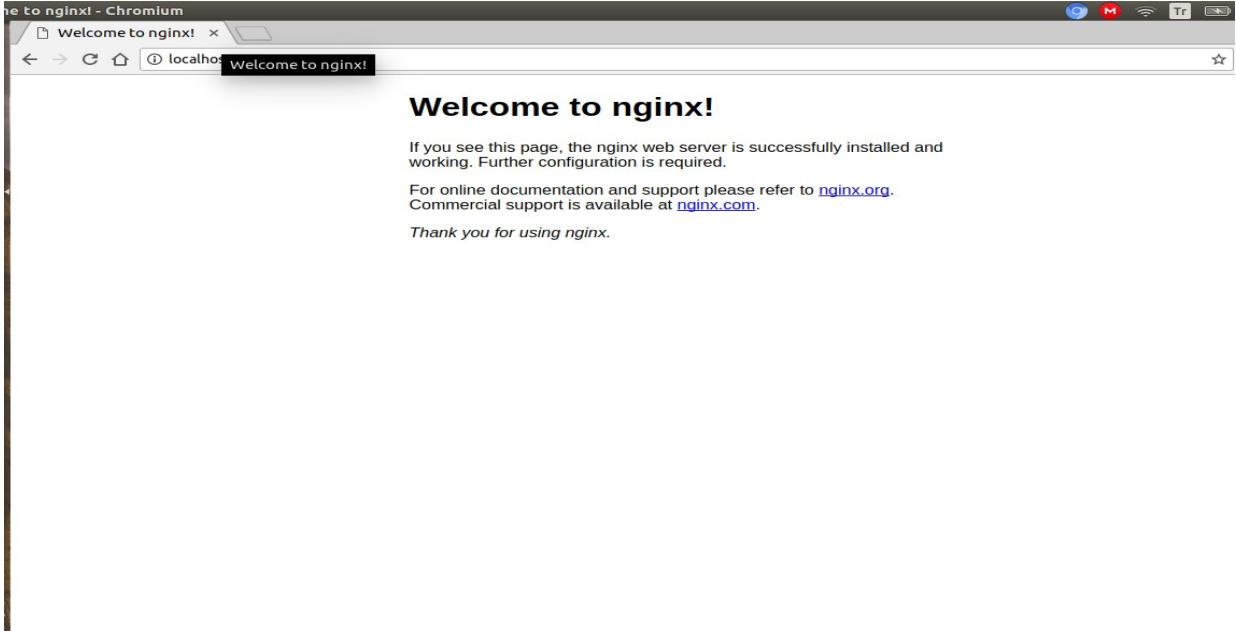
Daha önceden elimizde olan nginx image'nı localhostumuzdaki 8000 numaralı port ile containerdaki 80 numaralı port ile eşleyip çalıştırıyoruz. Ben daha önceden run ettiğim için ID ile birlikte hata türü döndü.

```
ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ curl localhost:8000
<!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>
busra@ereborlugimli:~/Desktop$
```

Localhostumuzun 8000 numaralı portunda bulunan sayfayı görüntülemek için curl komutundan yardım aldık. Eğer istersek browser'ımıza localhost:8000 yazarakta sayfa içeri ğini gösterebiliriz.¹⁷



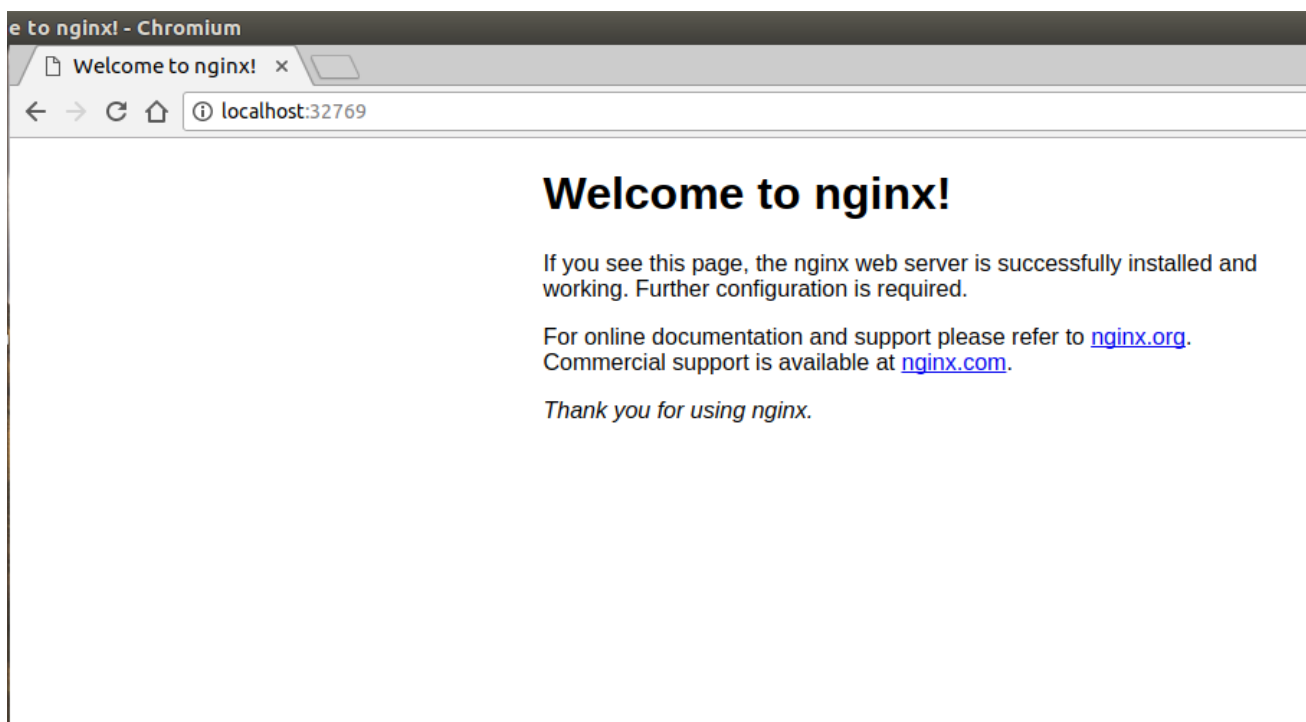
Bu şekilde bir çalışma yaptığımızda tamamen el ile ayarlanmış bir sonuca vardık. Ama alternatif bir yol olan otomatik port ataması ilede yapabiliirdik.

```
root@ereborlugimli: /home/busra/Desktop
root@ereborlugimli:/home/busra/Desktop# ID=$(docker run -d -P nginx)
root@ereborlugimli:/home/busra/Desktop# docker port $ID 80
0.0.0.0:32769
root@ereborlugimli:/home/busra/Desktop# curl localhost:32769
<!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@ereborlugimli:/home/busra/Desktop#
```

18



Gördüğünüz gibi localhostumuzda 32769 numaralı port ilede Nginx'in giri ş sayfasını görüntülemiş olduk.

Koşurma Komutları

- a --attach : Container başlatılırken terminalin (komut satırı input/output) Container'a tekrar aktif edilmesinin gerektiğini belirtir.
- d --detach : Container'ı arka planda çalıştırır ve container ID değerini döndürür.
- i --interactive : Container'a bash shell açmak istediğimiz zaman kullanırız.
- restart : Container'ı yinelemiş yani stop-start yapmış oluruz.
- rm : Otomatik olarak container'ı siler.
- t --tty : Container'a interaktif bash shell açacağımız zaman terminali attach etmemiz gerektiği zaman kullanırız.¹⁹
- e --env : Container'a çevre değ şkeni tanımlayacağımız zaman kullanırız.
- h --hostname :
- name : Container'a isim verdiğimiz zaman ismi container adresi olarakta düşünebiliriz.
- v --volume : Container'a dosya/klasör bağlantısı yapacağımız zaman kullanırız.
- expose : Container'a port ve port aralığı tanımlar.
- p --publish : Container portunu herkese erişilebilir kılar.
- P - -publish-all : Hostdaki tüm container portlarını erişilebilir kılar.

Python ile Web Server Uygulaması

Geri dönüş de ğeri sadece "Hello World" olan bir uygulama yazıcaz. Öncelikle identidock adında bir klasör oluřturuyoruz. Daha sonra içine app adında bir klasör oluřturup içine identidock.py

```
busra@ereborluginli:~/Desktop/Docker/identidock$ sudo docker build -t identidock .
[sudo] password for busra:
Sending build context to Docker daemon 6.144kB
Step 1/5 : FROM python:3.4
--> c588c14f484e
Step 2/5 : MAINTAINER pip install Flask==0.10.1
--> Using cache
--> 3848044d9709
Step 3/5 : WORKDIR app/
--> Using cache
--> 1611ae386ce5
Step 4/5 : COPY app /app
--> Using cache
--> 0ef894626d31
Step 5/5 : CMD python identidock.py
--> Using cache
--> ac1d6b70371f
Successfully built ac1d6b70371f
Successfully tagged identidock:latest
busra@ereborluginli:~/Desktop/Docker/identidock$ sudo docker run -d -p 5000:5000 identidock
41fef64953715b4aa3928319f1f0585111dc58fac92ffa1ed877208a2576c07e
busra@ereborluginli:~/Desktop/Docker/identidock$
```

20

dosyasını oluřturuyoruz. Dosya içeri ğini oluřturduktan sonra /identidock dizinine Dockerfile oluřturuyoruz ve container'ı inřa etmiř oluyoruz. Ve "curl localhost:5000" komutu ile "Hello World geri dönüş de ğerini alıyoruz."

Eğer birden çok makinada aynı işlemi yapmak isteseydik o zaman Docker'ın Compose toolunu

kullanıcağıktık. Göstericek olursak : ²¹

```
busra@ereborlugimli:~/Desktop/Docker/identidock$ sudo docker-compose build
Building identidock
Step 1/5 : FROM python:3.4
--> c588c14f484e
Step 2/5 : MAINTAINER pip install Flask==0.10.1
--> Using cache
--> 3848044d9709
Step 3/5 : WORKDIR app/
--> Using cache
--> 1611ae386ce5
Step 4/5 : COPY app /app
--> Using cache
--> c7124b648827
Step 5/5 : CMD python identidock.py
--> Using cache
--> 269d06b1515c
Successfully built 269d06b1515c
Successfully tagged identidock_identidock:latest
busra@ereborlugimli:~/Desktop/Docker/identidock$ sudo docker-compose up
Starting identidock_identidock_1 ...
Starting identidock_identidock_1 ... done
Attaching to identidock_identidock_1
identidock_1 | File "identidock.py", line 3
identidock_1 |     app=Flask(__name__) //object oluştur
identidock_1 |                                     ^
identidock_1 | SyntaxError: invalid syntax
identidock_identidock_1 exited with code 1
busra@ereborlugimli:~/Desktop/Docker/identidock$
```

Docker Compose

Orta ve büyük ölçekteki sistemleri Docker CLI kullanarak kullanıma sunmak ve bakımını yapmak pek makul değildir. Bu tip sistemlerde Containerların çalıştırılması durdurulması birbiri ile olan ilişkilerinin belirtilmesi yani basit bir şekilde yönetilmesi görevlerini yerine getirecek ayrı bir aracın varlığı gereklidir. İşte bu yüzden ;

Development + UAT(Kullanıcı Kabul) test ortamlarının kolay bir şekilde yönetilebilmesi için Docker Compose kullanılmaktadır.

docker-compose.yml dosyasında detayları verilen ve birbirleri ile ilişkileri tanımlanan servisler (Containerlar) tek bir komut ile ayağa kaldırılıp tek bir komut ile durdurulur ve yine tek bir komut ile silinir. Kısaca ;

- a. Tek bir ana bilgisayardan çoklu yalıtılmış ortam sağlar.Yani ana bilgisayarda, tek bir ortamın birden fazla kopyasını oluşturmak tek bir CLI kullanılır. Ve merkezden tüm ortamlar yönetilir.
- b. Containerlar oluşturulduğunda hacimleri korur. Daha önceki çalışmalardaki volume'ler yani değişkenler korunur.
- c. Herhangi bir containerda değişiklik yapıp yeniden yapılandırıdığımız zaman sadece değişkenlerde yani containerda değişiklikler olur ve sadece değişken container yeniden yapılandırılır.
- d. Her Kompozisyonu farklı ortamlar ve farklı kullanıcılar için çalıştırmak isteyebiliriz. O zaman bu durumda extends alanı kullanarak oluşturma dosyasında değişiklikler yapabiliriz.²²

docker-compose.yml genel görünümü :

```
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Docker Compose Kurulumu

64 bitlik versiyonlarda kurmak istediğimiz ubuntu versiyonumuza göre kurulum yapmalıyız. Benim-
kisi Yaketty 16.10 versiyonu olduğunu için, ubuntu versiyon makinema göre kurulum yapacağım.

Elbette ki, bir çok farklı kurulum yolu olduğunda unutmamakda fayda var. ²³

Step1:

```
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ sudo apt-get install \
> apt-transport-https \
> ca-certificates \
> curl \
> software-properties-common
[sudo] password for busra: 
```

Step2 :

```
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Step3:

```
busra@ereborlugimli:~/Desktop$ apt-key fingerprint 0EBFCD88
busra@ereborlugimli:~/Desktop$ 
```

Step4:

```
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ sudo add-apt-repository \
> "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
> $(lsb_release -cs) \
> stable"
```

Step5:

```
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ docker-compose --version
docker-compose version 1.13.0, build 1719ceb
busra@ereborlugimli:~/Desktop$ 
```

Docker-Compose Uygulama

Amcımız her f5 bastığımızda yani sayfayı yenilediğimizde sayacı arttıran localde yayınlanan docker compose uygulaması yapmak. Öncelikle :

Step1 :

Uygulamamızın dosyalarının bir arada olduğu bir klasör oluşturuyoruz.

```
ereborlugimli: ~/Desktop/Docker
busra@ereborlugimli:~/Desktop/Docker$ mkdir composetest/
```

```
ereborlugimli: ~/Desktop/Docker
busra@ereborlugimli:~/Desktop/Docker$ cd composetest/
```

Step2 :

Uygulamamız için minik bir python kodu yazıyoruz.

```
ereborlugimli: ~/Desktop/Docker/composetest
busra@ereborlugimli:~/Desktop/Docker/composetest$ touch app.py
busra@ereborlugimli:~/Desktop/Docker/composetest$ nano app.py
busra@ereborlugimli:~/Desktop/Docker/composetest$
```

```
ereborlugimli: ~/Desktop/Docker/composetest
nano 2.6.3 File: app.py

from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    count = redis.incr('hits')
    return 'Hello World! I have been seen {} times.\n'.format(count)

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

Step 3 :

Uygulamanın çalışması için gerekli olan uygulama isimlerini text dosyasına yazıyoruz.

```
ereborlugimli: ~/Desktop/Docker/composetest
busra@ereborlugimli:~/Desktop/Docker/composetest$ touch requirements.txt
busra@ereborlugimli:~/Desktop/Docker/composetest$ nano requirements.txt

ereborlugimli: ~/Desktop/Docker/composetest
nano 2.6.3

flask
redis
```

Step 4 :

Uygulamamızın yapılandırma dosyası yani konfgürasyon ayarlarının bulunduğu docker file dosyamızı yazıyoruz.

```
ereborlugimli: ~/Desktop/Docker/composetest
busra@ereborlugimli:~/Desktop/Docker/composetest$ touch Dockerfile

ereborlugimli: ~/Desktop/Docker/composetest
nano 2.6.3

FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

Step 5 :

Compose özelliğini kullanamamızı sağlayan docker-compose.yml dosyasını oluşturduk.

```
ereborlugimli: ~/Desktop/Docker/composetest  
busra@ereborlugimli:~/Desktop/Docker/composetest$ touch docker-compose.yml  
busra@ereborlugimli:~/Desktop/Docker/composetest$ nano docker-compose.yml
```

```
ereborlugimli: ~/Desktop/Docker/composetest  
nano 2.6.3
```

```
Version: '2'  
services:  
  web:  
    build: .  
    ports:  
      - "5000:5000"  
    volumes:  
      - .:/code  
  redis:  
    image: "redis:alpine"
```

Step 6:

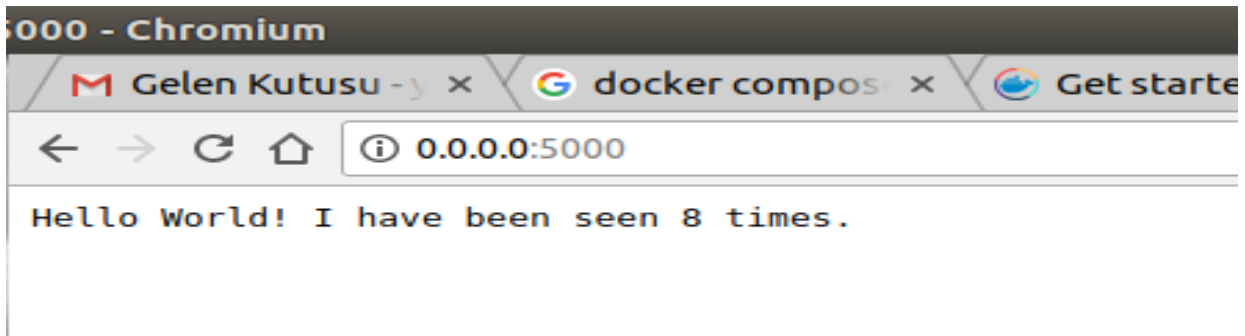
Son adımda "docker-compose up" ile yapılandırıyoruz.

```
ereborlugimli: ~/Desktop/Docker/composetest  
nano 2.6.3
```

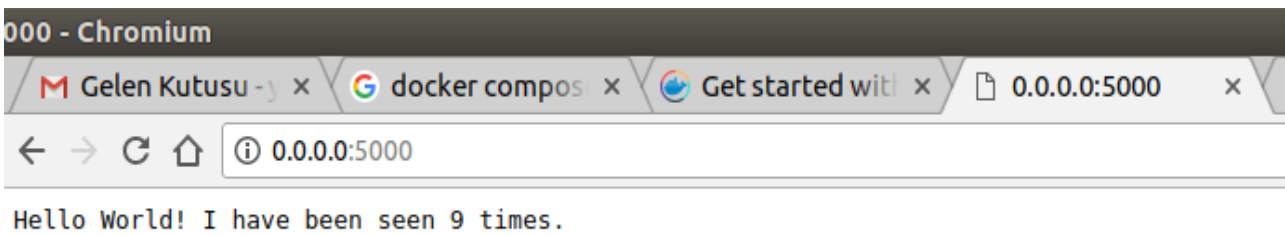
```
Version: '2'  
services:  
  web:  
    build: .  
    ports:  
      - "5000:5000"  
    volumes:  
      - .:/code  
  redis:  
    image: "redis:alpine"
```

Step 7 :

VE uygulamamızı test etme zamanı. "<http://localhost:5000>" giderek yazdığımız uygulamaya yönlendiriliriyoruz.



Her F5 yaptığımızda yani sayfayı yenilediğimizde times değeri +1 artıyor.



Basit Docker Uygulamaları

a.1 apache2

Step 1 :

Dockerfile dosyasını oluşturuyoruz.

```
ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/apache2$ touch Dockerfile
ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/apache2$ nano Dockerfile
busra@ereborlugimli:~/Desktop/Docker/apache2$
```

Gerekli yapılandırma ayarlarını yapıyoruz.

```
ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2
nano 2.6.3 File: Dockerfile
FROM ubuntu:16.10
MAINTAINER Büşra Yenidoğan Version :0.1
RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
ENV APACHE_USER www-data
ENV APACHE_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
ENV APACHE_LOCK_DIR /var/lock/apache2
ENV APACHE_PID_FILE /var/run/apache2.pid
EXPOSE 80
ADD apache-config.conf /etc/apache2/sites-enabled/000-default.conf
CMD ["/usr/sbin/apache2ctl","-D","FOREGROUND"]
```

Step 2 :

Oluşturduğumuz docker imajelerini listeliyoruz.

```
ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli:~/Desktop/Docker/apache2$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
identidock_nginx     latest             269d06b1515c       13 days ago        685MB
identidock            <none>             d056fec7affc       13 days ago        678MB
amouat/dnmonster     1.0                586b11a078db       2 weeks ago        889MB
nginx                latest             958a7ae9e569       3 weeks ago        109MB
redis                latest             a858478874d1       4 weeks ago        184MB
redis                3.0                f50d9cc463b7       4 weeks ago        176MB
python               2.7                2e9467da064d       5 weeks ago        678MB
composetest_web      latest             b6a1188c7f9a       6 weeks ago        94.5MB
python               3.4-alpine         f9b5ec164bb9       7 weeks ago        83.1MB
identidock            latest             ac1d6b70371f       2 months ago       685MB
python               3.4                c588c14f484e       3 months ago       685MB
ereborlugimli/nginx  latest             9b4032d8e2a9       3 months ago       227MB
ereborlugimli/mymongodb latest             227fd4514b7d       3 months ago       397MB
<none>                <none>             ac4c006dc838       3 months ago       104MB
<none>                <none>             aed8944c7f23       3 months ago       104MB
mysite               latest             d8621eace3ed       3 months ago       229MB
ereborlugimli/myapache2 latest             c132efb875ff       3 months ago       203MB
<none>                <none>             e4574d90a1df       3 months ago       220MB
postgres             latest             9910dc9f2ac0       3 months ago       267MB
ereborlugimli/myapache2_php latest             82eea2dfe5eb       3 months ago       203MB
ereborlugimli/myapache2 <none>             82eea2dfe5eb       3 months ago       203MB
ereborlugimli/myapache2_php <none>             f63d3b61799c       3 months ago       220MB
ereborlugimli/myapache2 <none>             4782773f9f0b       3 months ago       203MB
ereborlugimli/myapache2 <none>             56e9d0edec97       3 months ago       203MB
<none>                <none>             a30ba3a68e50       3 months ago       1.11MB
ereborlugimli/cowsay latest             c88e5fbb1783       3 months ago       215MB
test/cowsay-dockerfile latest             c88e5fbb1783       3 months ago       215MB
test/cowsayimage      latest             ec9c56859c3f       3 months ago       171MB
busybox               latest             00f017a8c2a6       3 months ago       1.11MB
redis                 alpine             83638a6d3af2       3 months ago       19.8MB
redis                 <none>             e4a35914679d       3 months ago       183MB
debian                latest             978d85d02b87       3 months ago       123MB
ubuntu                16.10              30b23364a716       3 months ago       104MB
ubuntu                latest             0ef2e08ed3fa       3 months ago       130MB
ubuntu                14.04              7c09e61e9035       3 months ago       188MB
ubuntu                trusty              7c09e61e9035       3 months ago       188MB
swarm                 latest             36b1e23becab       5 months ago       15.9MB
busra@ereborlugimli:~/Desktop/Docker/apache2$
```

Step 3 :

Docker Hub'a oluşturduğumuz image'ı yüklüyoruz.

```
ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2
busra@ereborlugimli:~/Desktop/Docker/apache2$ sudo docker push ereborlugimli/myapache2
The push refers to a repository [docker.io/ereborlugimli/myapache2]
6d935385d0bf: Pushed
0008e1bb67f2: Layer already exists
bdde9f5cdfd8: Layer already exists
8d9be5009ebf: Layer already exists
efb4da760f08: Layer already exists
f8785d7a565f: Layer already exists
f3e16b4643c6: Layer already exists
latest: digest: sha256:31adebd896ca1ace6ec98a038f8eb91519dd2454bf112940ccf412f7ef640dd8 size: 1776
busra@ereborlugimli:~/Desktop/Docker/apache2$
```

Step 4 :

Oluşturduğumuz Container'ın shelline giriyoruz. Ardından apache2 servisini restartediyoruz.

```
ba75ad3fca73: /
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2
root@ba75ad3fca73: /
busra@ereborlugimli: ~/Desktop/Docker/apache2$ sudo docker run -ti -p 8080:80 ereborlugimli/myapache2 /bin/t
root@ba75ad3fca73: /
```

```
ba75ad3fca73: /
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2
root@ba75ad3fca73: /
busra@ereborlugimli: ~/Desktop/Docker/apache2$ sudo docker run -ti -p 8080:80 ereborlugimli/myapache2 /bin/t
root@ba75ad3fca73: /
root@ba75ad3fca73: /# /etc/init.d/apache2 restart
* Restarting Apache httpd web server apache2
AH00112: Warning: DocumentRoot [/var/www/site] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globa
lly to suppress this message
[ OK ]
root@ba75ad3fca73: /#
```

a.2 apache2&&php

Step 1 :

```
ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker
busra@ereborlugimli: ~/Desktop/Docker/apache2php$ touch Dockerfile
```

```
ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2php$ nano Dockerfile
```

```
ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop/Docker/apache2php
nano 2.6.3 File: Dockerfile
FROM ereborlugimli/myapache2
MAINTAINER Büşra Yenidoğan Versiyon: 0.1
RUN apt-get update && apt-get install -y php7.0 php7.0-mysql libapache2-mod-php7.0 curl lynx-cur && apt-get clean && rm -rf /var/lib/apt/lists$
#Enable apache mods.
RUN a2enmod php7.0
RUN a2enmod rewrite
#Update the PHP .ini file, enable <? ?> tags and quieten logging.
RUN sed -i "s/short_open_tag = Off/short_open_tag = On/" /etc/php/7.0/apache2/php.ini
RUN sed -i "s/error_reporting = .*$/error_reporting = E_ERROR | E_WARNING | E_PARSE/" /etc/php/7.0/apache2/php.ini
#Copy this repo into place.
ADD www /var/www/site
```

Step 2 :

```
ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop/Docker/apache2php$ sudo docker build -t mysite .
[sudo] password for busra:
Sending build context to Docker daemon 7.168kB
Step 1/8 : FROM ereborlugimli/myapache2
--> c132efb875ff
Step 2/8 : MAINTAINER BÜşra Yenidoğan Versiyon: 0.1
--> Using cache
--> 43f82f609d3a
Step 3/8 : RUN apt-get update && apt-get install -y php7.0 php7.0-mysql libapache2-mod-php7.0 curl lynx-cur && apt-get clean && rm -rf /var/lib
/apt/lists/*
--> Using cache
--> 60a502ff5e37
Step 4/8 : RUN a2enmod php7.0
--> Using cache
--> 4b7ae04a3367
Step 5/8 : RUN a2enmod rewrite
--> Using cache
--> 6c7d4c85ca8e
Step 6/8 : RUN sed -i "s/short_open_tag = Off/short_open_tag = On/" /etc/php/7.0/apache2/php.ini
--> Using cache
--> 9d47b6837313
Step 7/8 : RUN sed -i "s/error_reporting = .*/error_reporting = E_ERROR | E_WARNING | E_PARSE/" /etc/php/7.0/apache2/php.ini
--> Using cache
--> eb2f24720c29
Step 8/8 : ADD www /var/www/site
--> ecfe8ca7b4d1
Removing intermediate container 4c2e594e86c3
Successfully built ecfe8ca7b4d1
Successfully tagged mysite:latest
busra@ereborlugimli:~/Desktop/Docker/apache2php$
```

```
ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli: ~/Desktop/Docker/apache2php
busra@ereborlugimli: ~/Desktop/Docker/apache2php$ sudo docker run -p 8080:80 -d mysite
ddc7657b251603a2fdfcc4da0fdd1ae586ae5829bfcc3eabe35db18564e62d9e
docker: Error response from daemon: driver failed programming external connectivity on endpoint stoic_hugle (bdf81f4ef9e33716d77660f98e1da34ec6
500479f2f5406c6b596c06db81951d): Bind for 0.0.0.0:8080 failed: port is already allocated.
busra@ereborlugimli:~/Desktop/Docker/apache2php$
```

Burda çıktı bir hata aldık dikkat edersiniz bu hatanın sebebi 8080 portunun daha öncede başka bir uygulama için tanımlanmış yani kullanılmış olmasından dolayıdır. O yüzden uygulamalara port tanımlarken kullandığınız portu containera vermemeyi yada tanımlı olan portdaki containerı stop etmeyi unutmayınız. Bu sorun ortadan kalktığı zaman browserınıza localhost:8080 yazdığınızda uygulamayı istediğiniz gibi görüntüleyebileceksiniz.³²

a.3 mongoDB

Step 1 :

```
busra@ereborlugimli: ~/Desktop × busra@ereborlugimli: ~/Desktop/Docker/mongoDB × busra@ereborlugimli: ~/Desktop/Docker,
nano 2.6.3 File: Dockerfile
FROM ubuntu
MAINTAINER BÜşra Yenidoğan Versiyon : 0.1
RUN apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
RUN echo "deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen" | tee -a /etc/apt/sources.list.d/10gen.list
RUN apt-get update
RUN apt-get -y install apt-utils
RUN apt-get -y install mongodb-10gen
#RUN echo "" >> /etc/mongodb.conf
CMD ["/usr/bin/mongod", "--config", "/etc/mongodb.conf"]
```

```
ereborlugimli: ~/Desktop/Docker/mongoDB
busra@ereborlugimli: ~/Desktop × busra@ereborlugimli: ~/Desktop/Docker/mongoDB × busra@ereborlugimli: ~/Desktop/Docker/mongoDB >
busra@ereborlugimli:~/Desktop/Docker/mongoDB$ sudo docker build --tag my/repo .
Sending build context to Docker daemon 4.608kB
Step 1/8 : FROM ubuntu
--> 0ef2e08ed3fa
Step 2/8 : MAINTAINER BÜşra Yenidoğan Versiyon : 0.1
--> Using cache
--> 2afacd08984b
Step 3/8 : RUN apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
--> Using cache
--> b9a1e0bb07a9
Step 4/8 : RUN echo "deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen" | tee -a /etc/apt/sources.list.d/10gen.list
--> Using cache
--> be526bd69695
Step 5/8 : RUN apt-get update
--> Using cache
--> c312699cfff04
Step 6/8 : RUN apt-get -y install apt-utils
--> Using cache
--> cb442fd017fe
Step 7/8 : RUN apt-get -y install mongodb-10gen
--> Using cache
--> bb11c5d3de80
Step 8/8 : CMD /usr/bin/mongod --config /etc/mongodb.conf
--> Using cache
--> 227fda514b7d
Successfully built 227fda514b7d
Successfully tagged my/repo:latest
busra@ereborlugimli:~/Desktop/Docker/mongoDB$
```

33

a.4 MysqlServer

```
FROM ubuntu:16.10
MAINTAINER B şra YenidoĒan Version : 0.1
ADD ./mysql-setup.sh /tmp/mysql-setup.sh
RUN /bin/sh /tmp/mysql-setup.sh

# Adding this will expose mysql on a random host port. It's recommended to avoid this. Other containers on the same
# host can use the service without it.
#EXPOSE 3306

CMD ["/usr/sbin/mysqld"]
```

```
ereborlugimli: ~/Desktop/Docker/mysqlserver
busra@ereborlugimli: ~/Desktop/Docker/mysqlserver
busra@ereborlugimli:~/Desktop/Docker/mysqlserver$ sudo docker build --tag my/repo .
[sudo] password for busra:
Sending build context to Docker daemon 6.144kB
Step 1/5 : FROM ubuntu:16.10
--> 30b23364a716
Step 2/5 : MAINTAINER Büşra Yenidoğan Version : 0.1
--> Using cache
--> aed8944c7f23
Step 3/5 : ADD ./mysql-setup.sh /tmp/mysql-setup.sh
--> 5f1a6f5d16
```


a.5 sshd

Step1 :

Konfg yani ayarlarımızın oldu ğ Dockerfile dosyamızı yazdıtan sonra containerımızı

yapılandırıyoruz.

```
ereborlugimli: ~/Desktop/Docker/sshd
busra@ereborlugimli: ~/Desktop
nano 2.6.3 File: Dockerfile
FROM ubuntu:16.10
RUN apt-get update && apt-get install -y openssh-server
RUN mkdir /var/run/sshd
RUN echo 'root:screencast' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
# SSH login fix. Otherwise user is kicked off after login
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd
ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile
EXPOSE 22
CMD ["/usr/sbin/sshd", "-D"]
```

```
ereborlugimli: ~/Desktop/Docker/sshd
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/sshd$ sudo docker build -t eg_sshd .
[sudo] password for busra:
Sending build context to Docker daemon 2.048kB
Step 1/10 : FROM ubuntu:16.10
--> 30b23364a716
Step 2/10 : RUN apt-get update && apt-get install -y openssh-server
--> Running in 0def30ca5406
Get:1 http://archive.ubuntu.com/ubuntu yakkety InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu yakkety-security InRelease [102 kB]
Get:3 http://security.ubuntu.com/ubuntu yakkety-security/universe Sources [30.2 kB]
```

Step 2 :

Containerımızı çalıştırıp kendi IP adresimizden makineye bağanıyoruz.

```
ereborlugimli: ~/Desktop/Docker/sshd
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/sshd$ sudo docker run -d -P --name test_sshd eg_sshd
ab8d8d9c1330e446be8d244a6645644b100f6af824
Chromium Web Browser
busra@ereborlugimli:~/Desktop/Docker/sshd$
```

```
ereborlugimli: ~/Desktop/Docker/sshd
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/sshd$ sudo docker port test_sshd 22
0.0.0.0:32768
busra@ereborlugimli:~/Desktop/Docker/sshd$
```

36

```
ereborlugimli: ~/Desktop/Docker/sshd
busra@ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop/Docker/sshd$ sudo ssh root@192.168.1.27 -p 32768
root@192.168.1.27's password:
```


KAYNAKLAR

- Using Docker Developing and Deploying Software with Containers

Adrian Mouat

- <https://docs.docker.com/>
- <http://www.gokhansengun.com/>³⁷