NECMETTİN ERBAKAN ÜNİVERSİTESİ

Mühendislik Ve Mimarlık Fakültesi

Bilgisayar Mühendisliği Bölümü

Bilgisayar Mühendisliği
Uygulama Geliştirme Projesi

Uygulama Konusu
Docker Uygulama Geliştirme

Öğrenci Bilgileri	
Ögr. No	
Ad Soyad	Büşra Yenidoğan

Mehmet HACIBEYOĞLU
Öğretim Görevlisi

Haziran
3 2017
Konya

İçindekiler

ONSOZ

2017 yılının 2. döneminde Uygulama Projesi olarak alınan bu proje temel anlatımlarla kullanıcıya Docker teknolojisi hakkında bilgi vermeyi amaçlamaktadır. Raporda daha çok uygulama ağrlıklı 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 koşuyormuşizlenimi 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 yafonksiyonlar 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 faklı olarak Docker iki farklı bileşene sahiptir: Docker Engine, containerların oluşturulması ve çalıştırılması; Docker Hub, dağıtık containerlar için bulut hizmeti. Docker Engine, koşan containerlar için hızlı ve kullanışlı arayüz ortamı sağlar.Ancak önceden,Container'ların koşması için önemli teknolojileri kullanmamız gerekiyordu.Docker Hub, imageleri indirmemiz için özel numaralandırma ortamı sağlıyor. Böylece bütün kullanıcılar imagelara 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]

1

Container ve VMS

Hypervisor(Sanallaştırma Platformları) bulundukları fziksel makina içerisinde farklı i şletim sistemleri kurulmasına olanak sağlar. Avantajı ; Kendilerine ayrılan disk bölümlerine kurulması ve hepsinin aynı işletim sistemi içerinde bulunmasıdır. Hyperversior da her hostun kendine ait işletim sistemi vardır. LXC'de ise Container'lar hostun işletim sistemini kullanmak zorundaydılar. Hyperversior'da her hostun bakımı ayrı ayrı yapılmalıdır. LXC'de ise bir işletim sisteminin yapılması yeterlidir.

Bazı önemli farklılıkları diyagramdan anlatmaya çalışalım :

Figure 1-1 : VMS deki üç ayrı uygulama host üzerinde koşmaktadır. Hyperversior için VMs oluşturmak ve çalıştırmak kontrol erişimi için gereklidir. OS ve hardware gerektiği zaman system call'ları yorumlamalıdır. Her VM'nin kopyası uygulamaların çalıştırılması ve kütüphane desteği için gereklidir.

Figure 1-2 : VM'den farklı olarak, Host kernelini koşan container'lar ile paylaşmaktadır. Bu şu demektir: her container, hostun kernelinda koşmaktadır. Container'ları durdurmak yada başlatmak VM gibi DockerEngine'nin görevidir. Eğer birden fazla aynı container var ise aynı kütüphaneyi paylaşabilirler. Ancak container içerisindeki processler istenmediği sürece bağlantılı değildir.

Sonuç olarak VMs ve containerlar aynı host üzerinde farklı uygulamaları kullanabilir.[2]

2

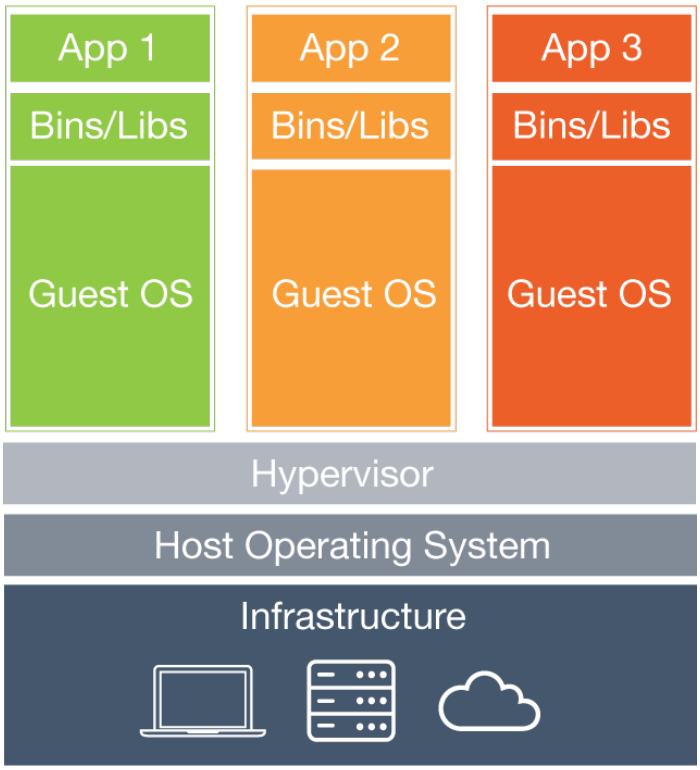


Figure 1-1 Üç VMs bir Host üzerinde koşuyor

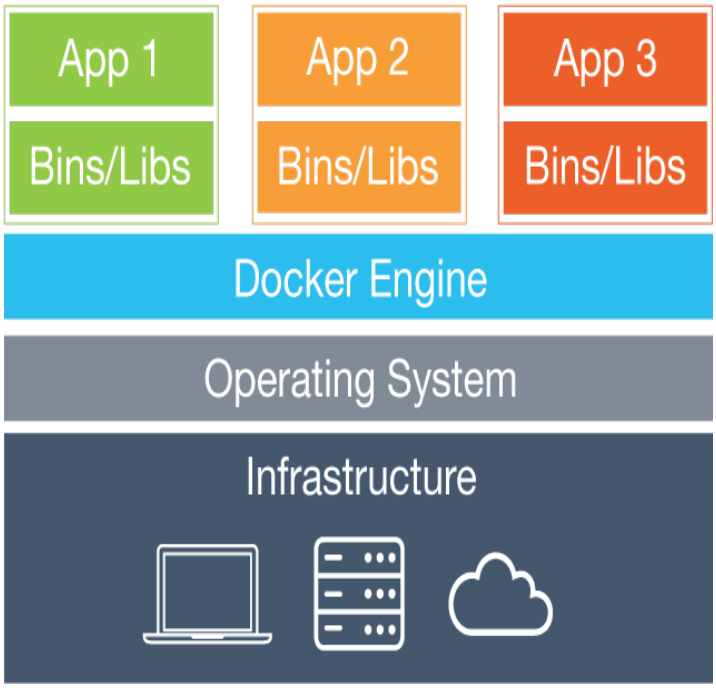


Figure 1-2 Üç Container bir Host üzerinde koşuyor[3]

3

Container’lar Ne ve Niçin

Container, yazılım geliştirmemiz, dağıtmamız ve çalıştırmamızda temel olarak değiştirmektedir. Geliştiriciler uygulamalarını kendi hostlarında çalıştırdıkları zaman herhangi bir dış etkenle değişilkiğe 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.
- Kullacılar 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 :

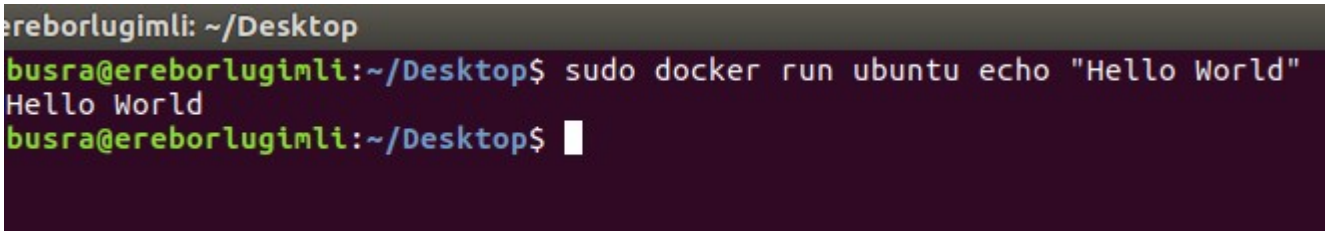
$ sudo docker version

```
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$
```

5

İlk image Çalışması

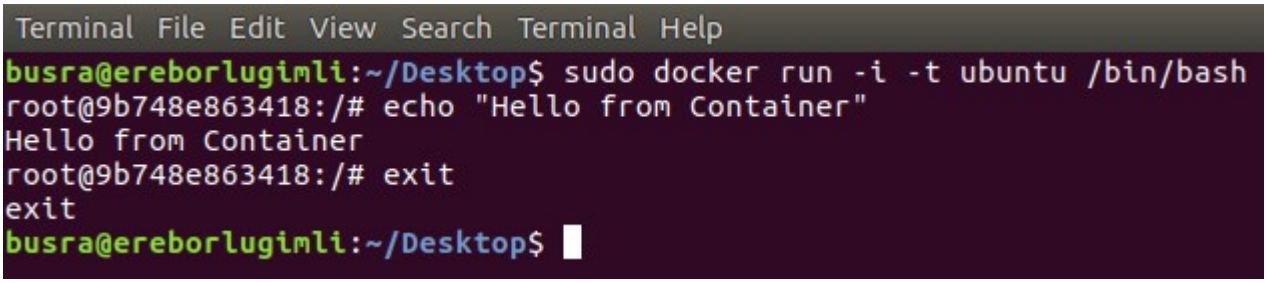
Docker'ı doğru kurduktan sonra, deniyoruz :



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

"Hello World" yazdırmış olduk.

Container'ın shell'ine geçmek istersek :



Basit Komutlar

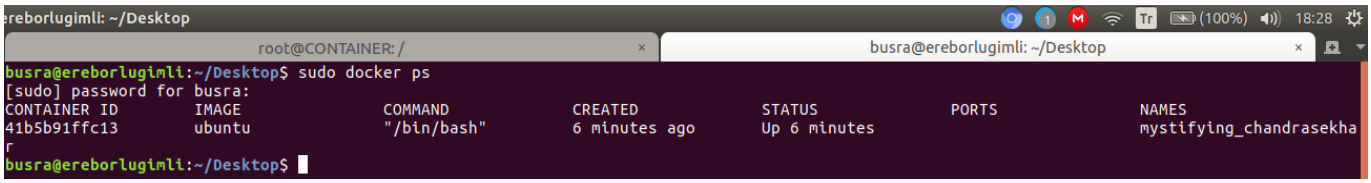
Biraz bilgi edinmek ve anlayabilmek adına çeşitli denemeler yapalım. Öncelikle yeni bir container başalatalım ancak -h flagi ile yeni bir hostname vererek :

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

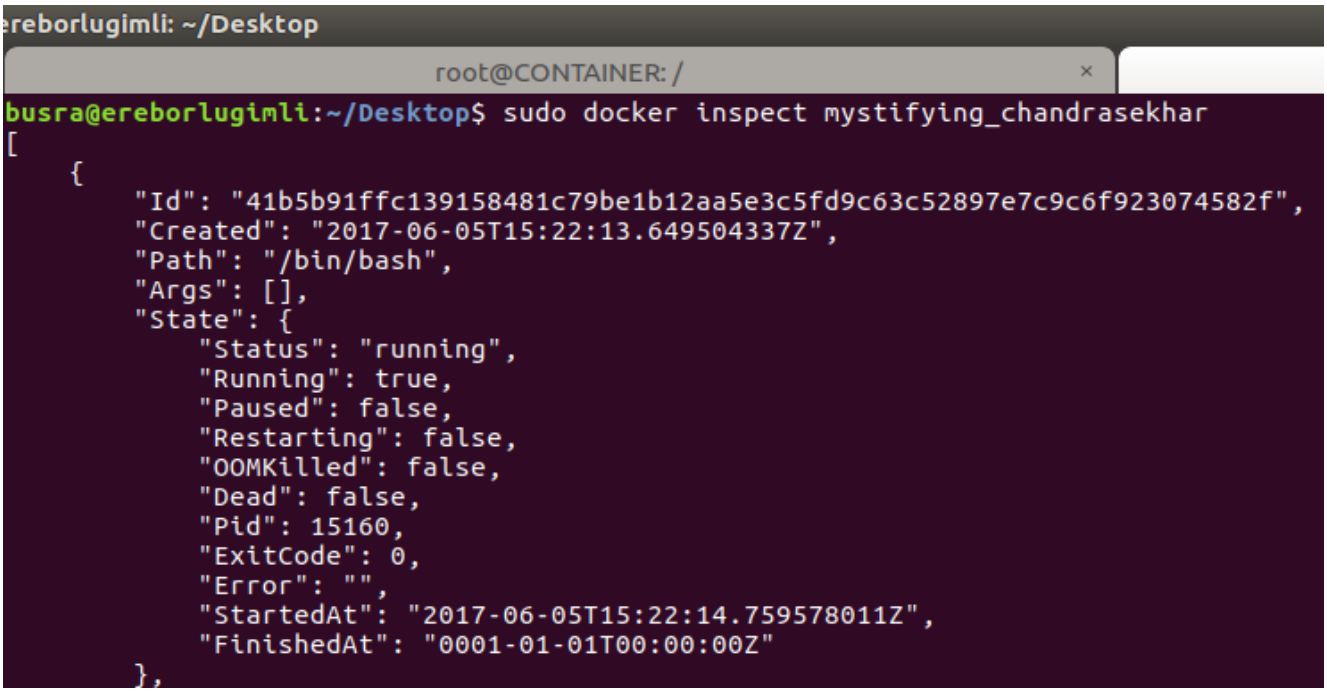
Daha sonra /bin dizinini /basket dizinine taşmaya çalı şalım.

```
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 sağayan "docker ps" komutunu çalıştırıyoruz :

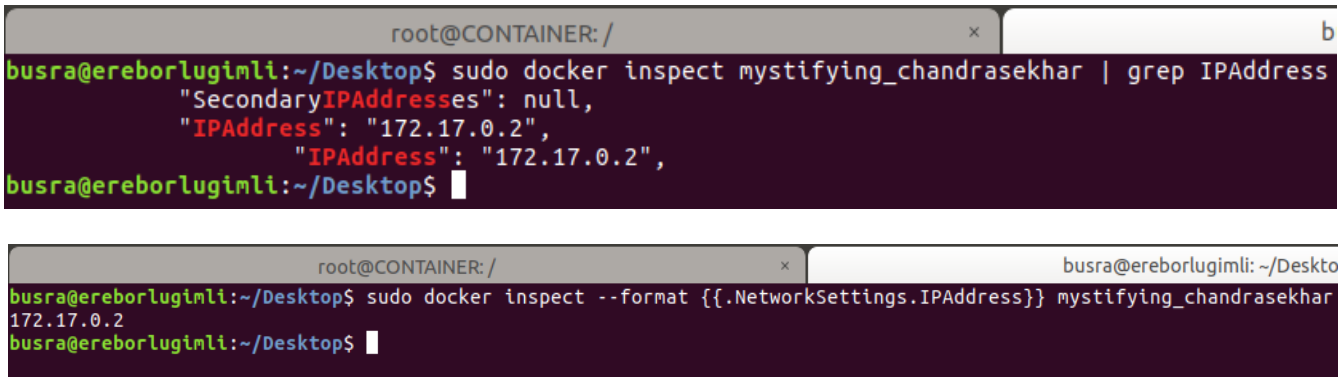


Bu komut bize container hakkında belli başı bazı bilgileri veriyor. Ancak Container hakkında bu bilgiler bizim için yeterli değil. Bu yüzden daha detaylı bir bilgi için container'ın adı ile "docker inspect" komutunu kullanarak daha fazla bilgiye ulaşyoruz.



Çıktının bir kısmı karşımızda olsada, bilgileri ayrıştırmak işimizi daha da kolaylaştırıcaktır. Örneğin ;

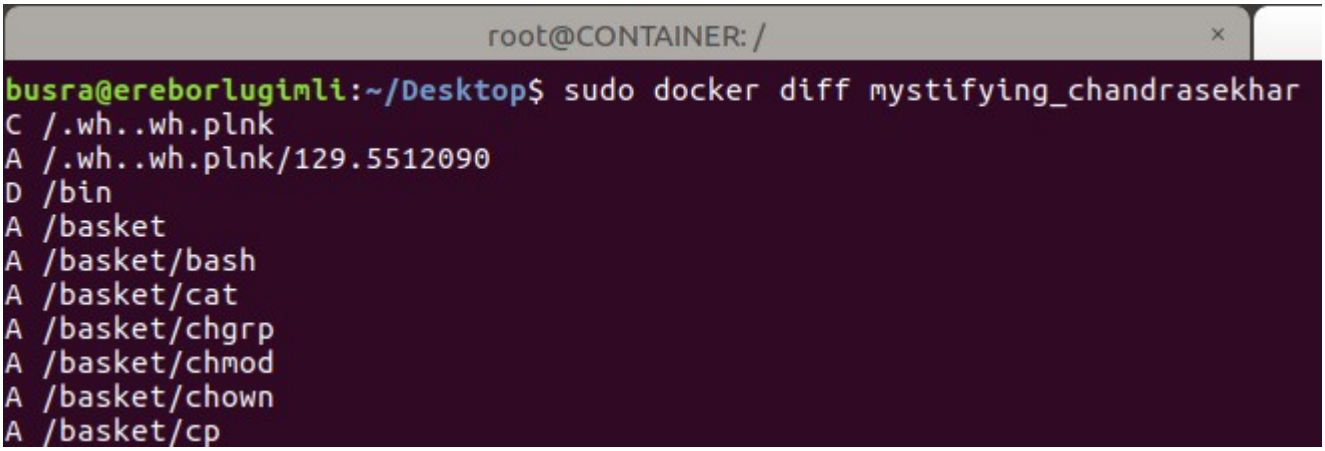
grep komutunu kullanarak IP Adresini öğrenebiliriz.



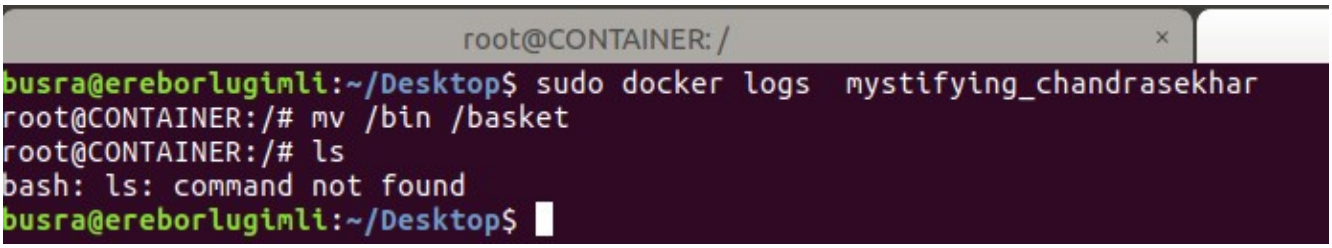
İki çıktıda bize IP adresi hakkında bilgi vericektir. Ancak bizim Container'daki yaptığımız değişikliği

gösteren komutumuz : [8]

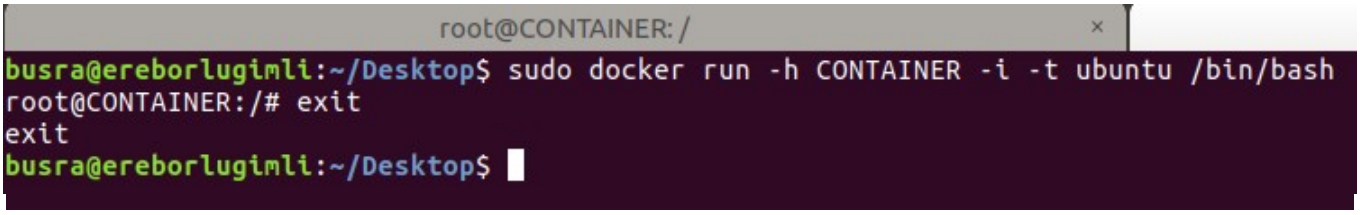
8



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



Container’dan çıkıyoruz :



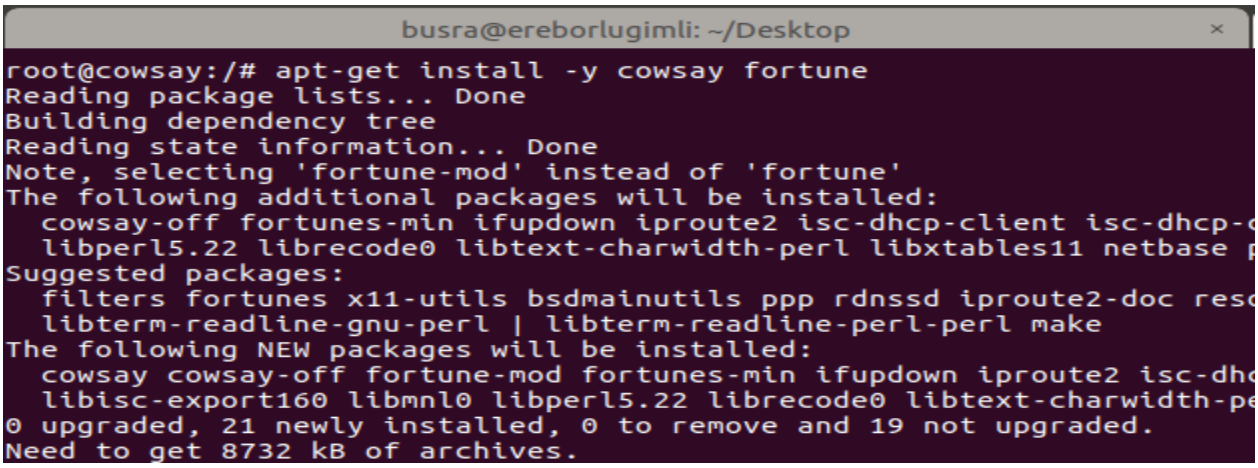
En sonunda da Container’ı siliyoruz :

Gördüğünüz gibi önce koşan durumdaki Container’ı durdumamızı istediği için önce durduruyoruz. Daha sonra silme işlemini gerçekleştiriyoruz.[9]

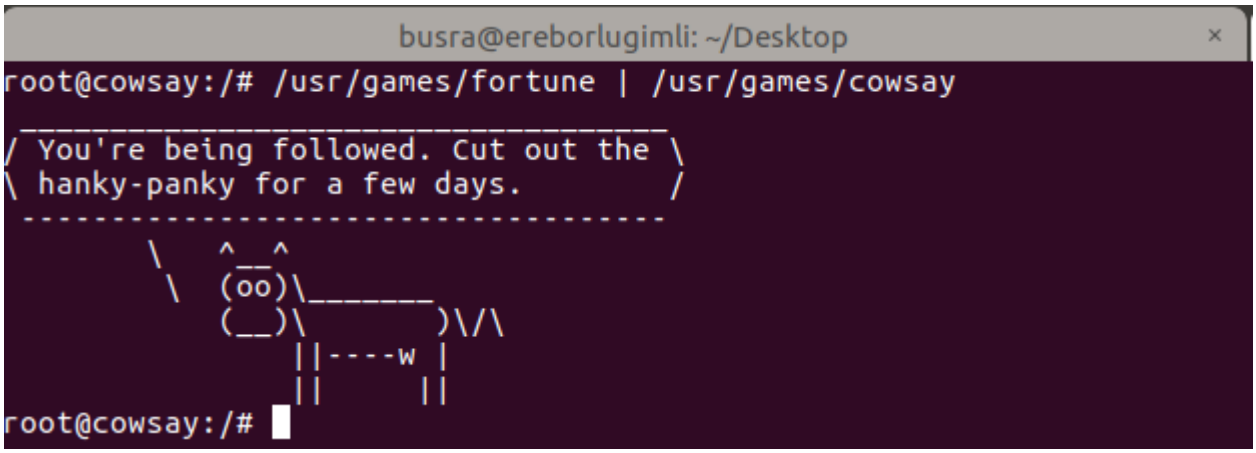
9

Basit Oyun Uygulaması

Hostname'i cowsay olan, Cowsay adında ki bir uygulamayı ubuntu Container üzerine kurup çalıştıralım.Container'ın shell'indeyken oyunu yüklemek için bu komutu giriyoruz :



Oyunu çalıştırmak için :



10

Dockerfile ile Image Yapımı

Dockerfle normal text dosyasıdır. Dockerfle kullanarak Docker image'ını oluşturuyoruz. 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$
```


11

Dockerfle içine yazdığımız komutları tek tek açıklayacak olursak:

FROM ile işletim sisteminin(ubuntunun) 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 Deamonla çalışan, Container'ların baz alacağı iştletim sistem veya uygulama olan metin bazlı(Dockerfile) binarydir.

Container : Docker Deamon tarafından Linux çekirdeği içerisinde birbirinden izole olarak çalıştırılan proceslerin herbirine verilen isimdir.

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

Repository : İlgili imageların oluşturduğu yapıdır.

Tag : Repositoryde ki image'ın alfanümerik tanımlamasıdır.

12

Redis'i Resmi Image Kullanarak Koşturmak

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 argumantınıda kullanacağız. Böylelikle Docker arka planda çalışıyor olucak. 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:~$
```


13

Docker Mimarisi

Docker Daemon : Linux Kernel ile direkt olarak iletişim halindedir.

Docker CLI : Docker Daemon ile iletişim kurmamızı sağlar.

Her ikiside 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 koşturulmasından ve yapılandırılmasından sorumludur.

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

14

Docker Komutları

ADD : Oluşturacağımız Image'a dosya sisteminden ve ya intertten dosya/klasör eklemek için kullanılır.

CMD : Sürekli çalışıcak 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 iletişimide 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 çalışıcak olan komutları koştururuz.

VOLUME : Volumler dosyalar yada dizileri direk olarak hosta bağlar yanı 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]

15

Basit Web Uygulaması – Nginx

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

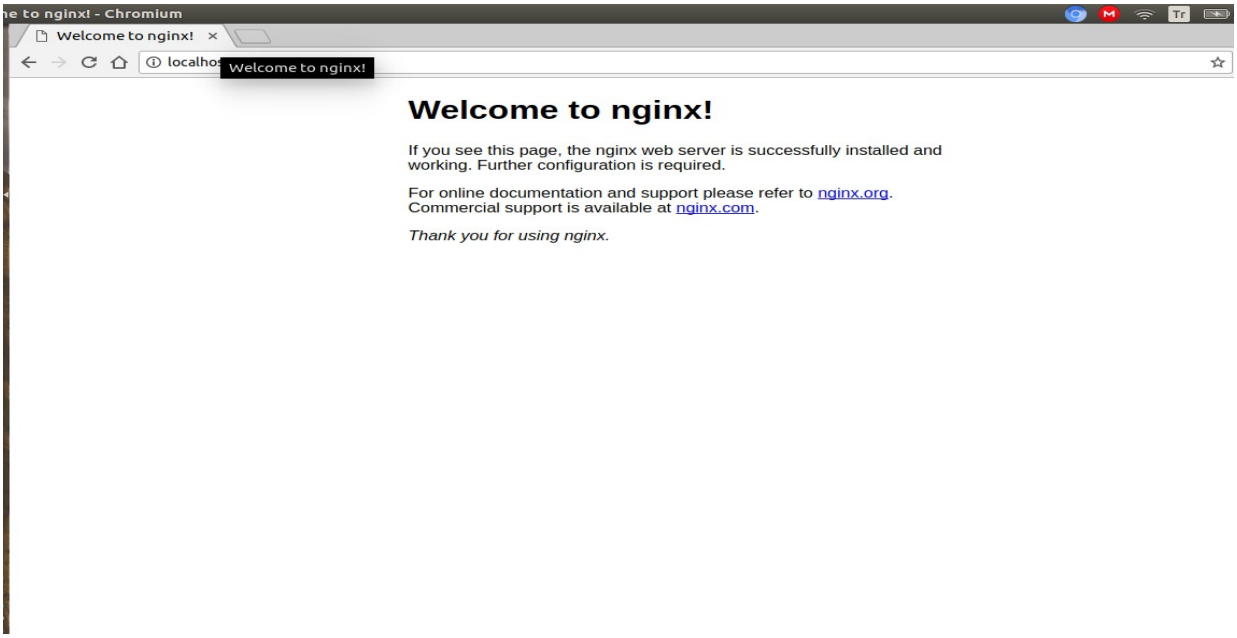
```
ereborlugimli: ~/Desktop
busra@ereborlugimli:~/Desktop$ sudo docker run -d -p 8000:80 nginx
[sudo] password for busra:
82db95e2cc54ee5c7e86fbf595221028f092d21d356dd66ec7f2080c7a4b703d
docker: Error response from daemon: driver failed programming externa
9c8465b9415d598c1b21d504f299f7dcfe2574): Bind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

Daha önceden elimizde olan nginx image'ını localhostumuzdaki 8000 numaralı port ile container daki 80 numararılı 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$
```


16

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

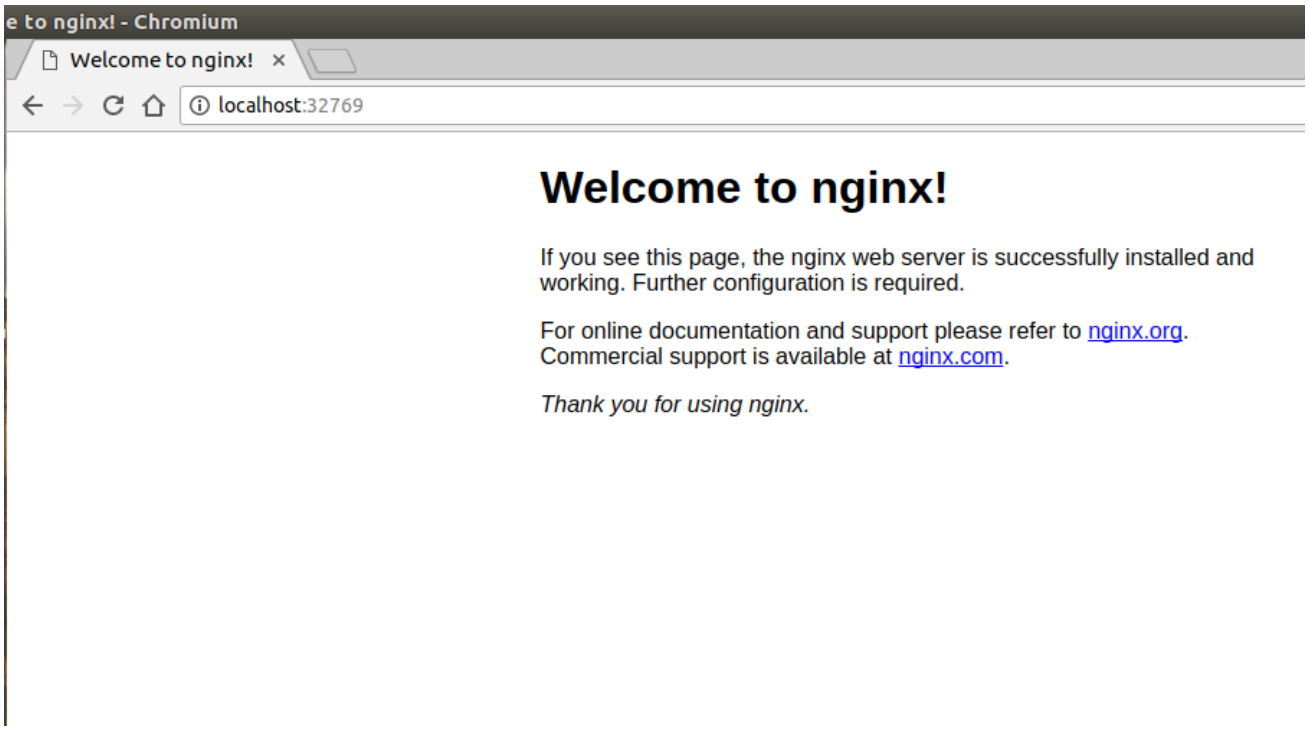


Bu şekilde bir çalışma yaptığımızda tamamen el ile ayarlanmış bir sonuca vardık. Ama alternatif bir yol olan otomatik port ataması ilede yapabilirdik.

17

```
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.

[18]

Koşturma 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 değişkeni 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.

19

Python ile Web Server Uygulaması

Geri dönüş değeri sadece "Hello World" olan bir uygulama yazıcaz. Öncelikle identidock adında bir klasör oluşturuyoruz. Daha sonra içine app adında bir klasör oluşturup içine identidock.py

```
busra@ereborlugimli:~/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@ereborlugimli:~/Desktop/Docker/identidock$ sudo docker run -d -p 5000:5000 identidock
41fef64953715b4aa3928319f1f0585111dc58fac92ffa1ed877208a2576c07e
busra@ereborlugimli:~/Desktop/Docker/identidock$
```

20

dosyasını oluşturuyoruz. Dosya içeriğini oluşturduktan sonra /identidock dizinine Dockerfle olşturuyoruz ve container'ı inşa etmiş oluyoruz. Ve "curl localhost:5000" komutu ile "Hello World geri dönüş değerini alıyoruz."

Eğer birden çok makinada aynı i şlemi yapmak isteseydik o zaman Docker'ın Compose toolunu kullanıcaktık. 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$
```

21

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 be 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 sağlanır.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 yapılandırdığımız zaman sadece değişen kapta yani containerda değişiklikler olur ve sadece değişen container yeniden yapılandırılır.

d. Her Kompozisyonu farklı ortamlar ve farklı kullanıcılar için değiştirmek isteyebiliriz. O zaman bu durumda extends alanı kullanrak oluşturma dosyasında değişiklikler yapabilriz.[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 olduğunuda unutmamakda fayda var. [23]

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: ~/D
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$
```

24

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 olu şuruyoruz.

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

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

Step2 :

Uygulamamız için minik bir python kodu yazıyoruz.

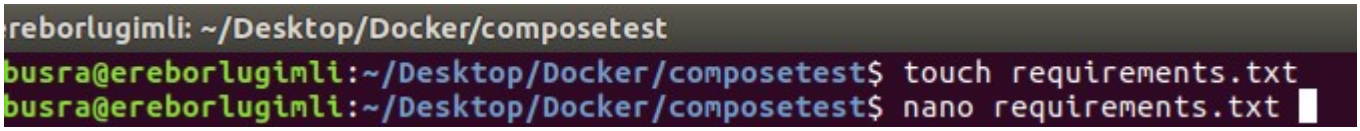
```
reborlugimli: ~/Desktop/Docker/composetest
busra@ereborlugimli:~/Desktop/Docker/composetest$ touch app.py
busra@ereborlugimli:~/Desktop/Docker/composetest$ nano app.py
busra@ereborlugimli:~/Desktop/Docker/composetest$
```

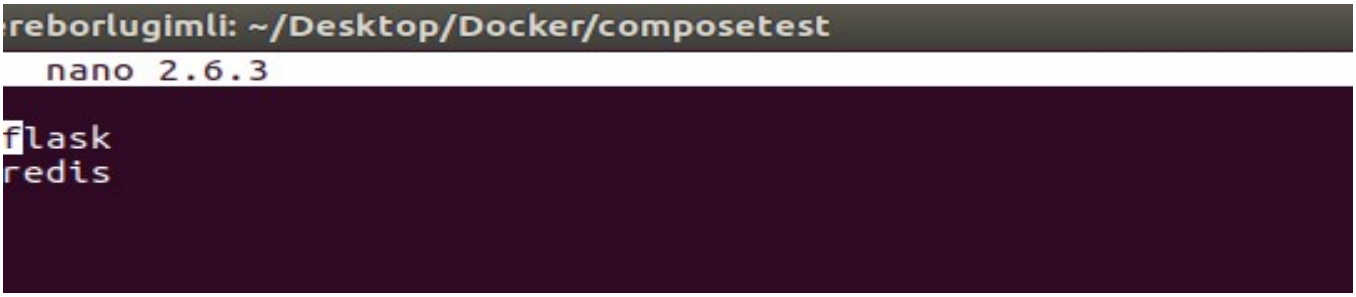
```
reborlugimli: ~/Desktop/Docker/composetest
  nano 2.6.3                                              File: app.py

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)
```

Chromium Web Browser

Step 3 :

Uygulamanın çalışması için gerekli olan uygulama isimlerini text dosyasına yazıyoruz.

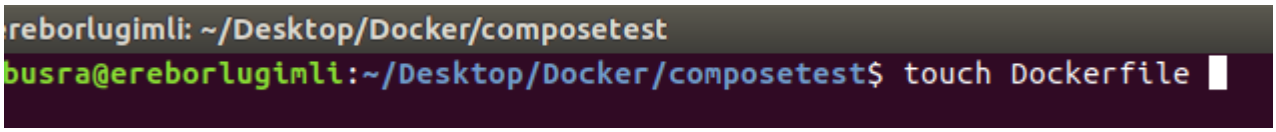


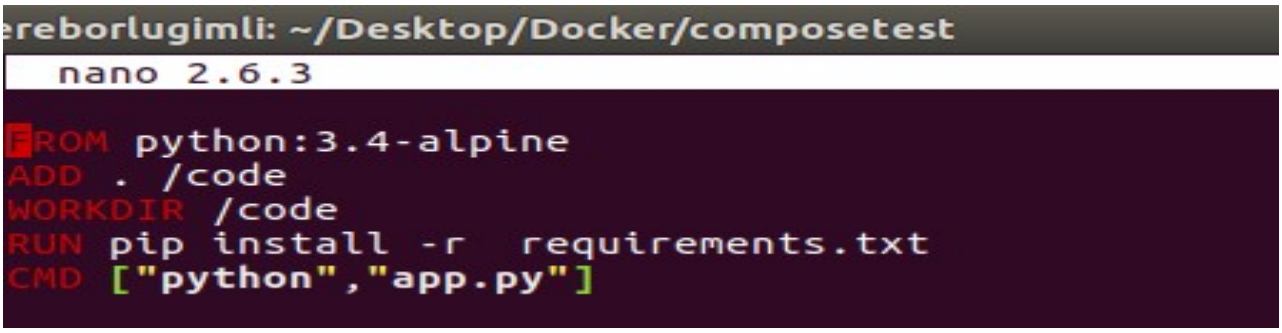


Step 4 :

Uygulamamızın yapılandırma dosyası yani konfgürasyon ayarlarının bulundu ğu docker fle dosya-

mızı yazıyoruz.





25
26

Step 5 :

Compose özelliğini kullanamımızı sağlayan docker-compose.yml dosyasını olu şturduk.

```
reborlugimli: ~/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"
```

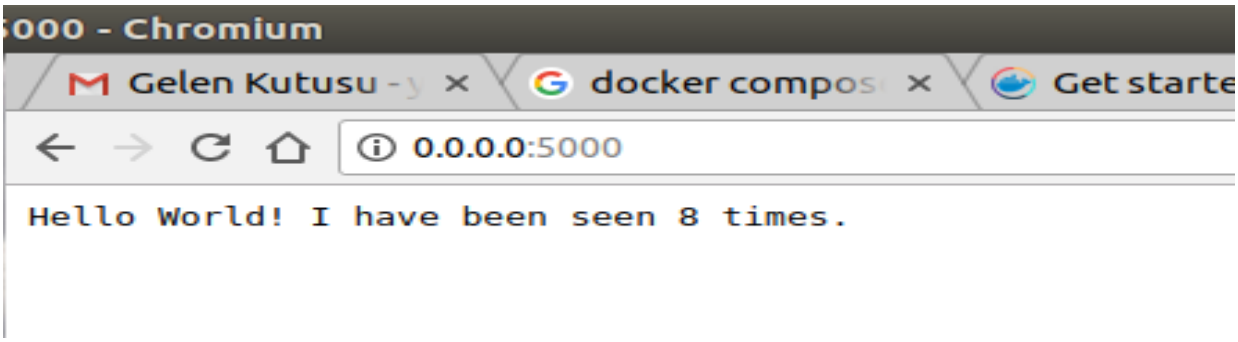
Step 6:

Son adımda "docker-compose up" ile yapılandırıyoruz.

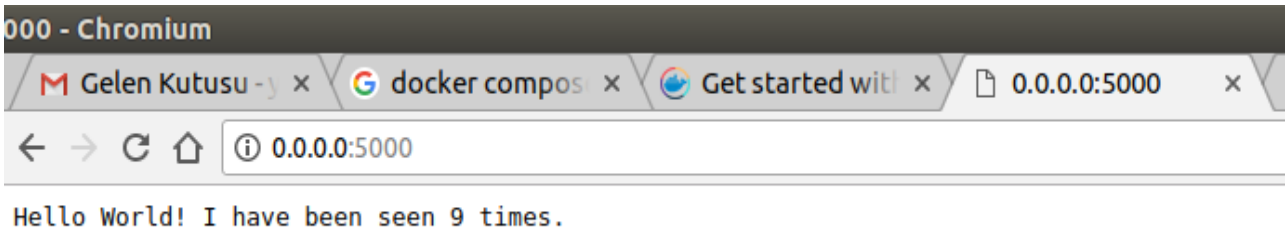
```
ereborlugimli: ~/Desktop/Docker/composetest
  nano 2.6.3

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.



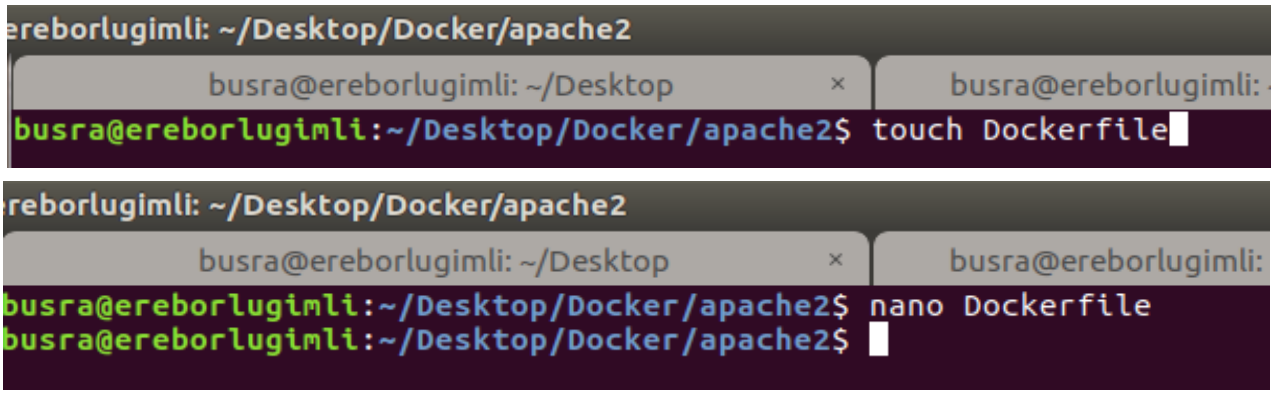
28

Basit Docker Uygulamaları

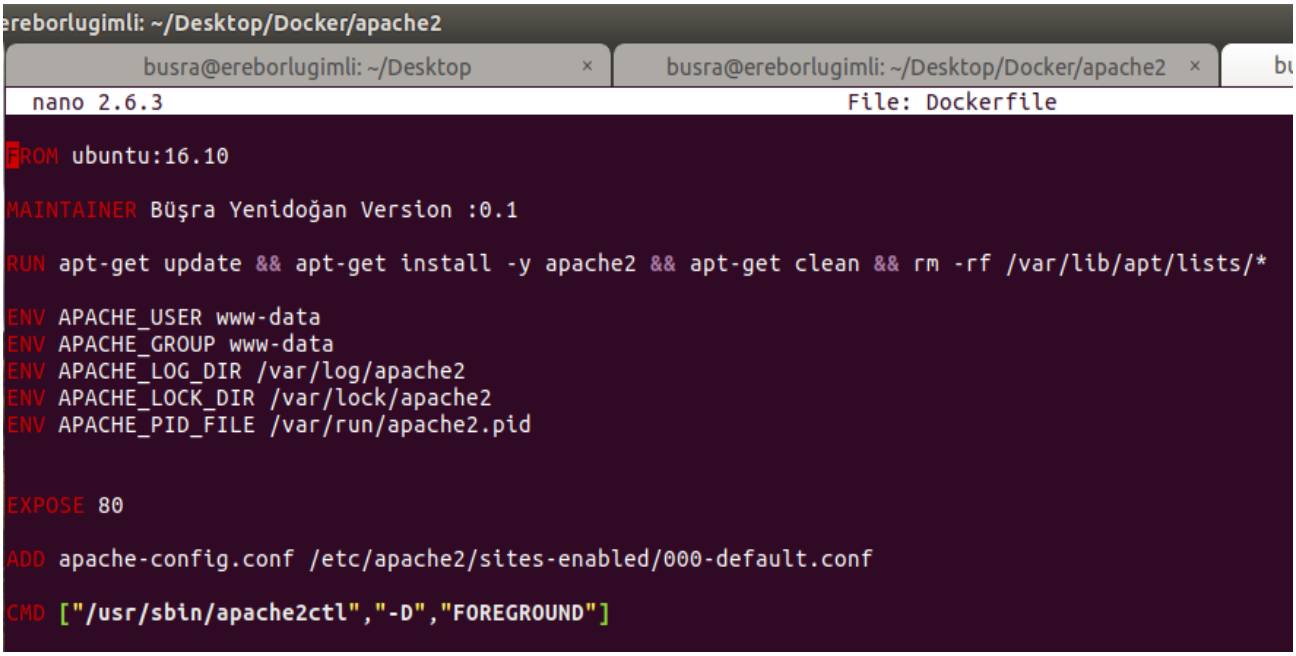
a.1 apache2

Step 1 :

Dockerfle dosyasını olu şturuyoruz.



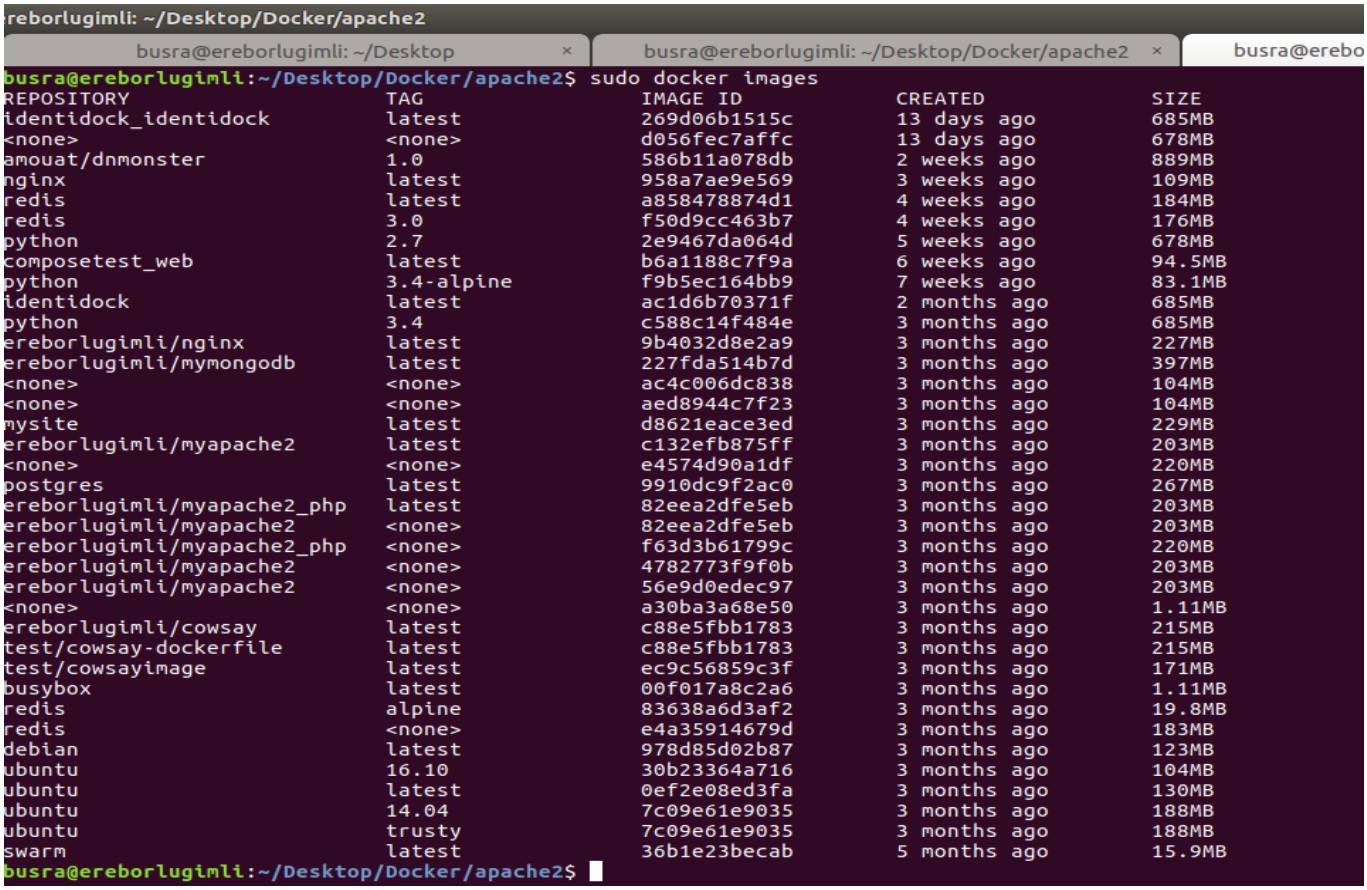
Gerekli yapılandırma ayarlarını yapıyoruz.



29

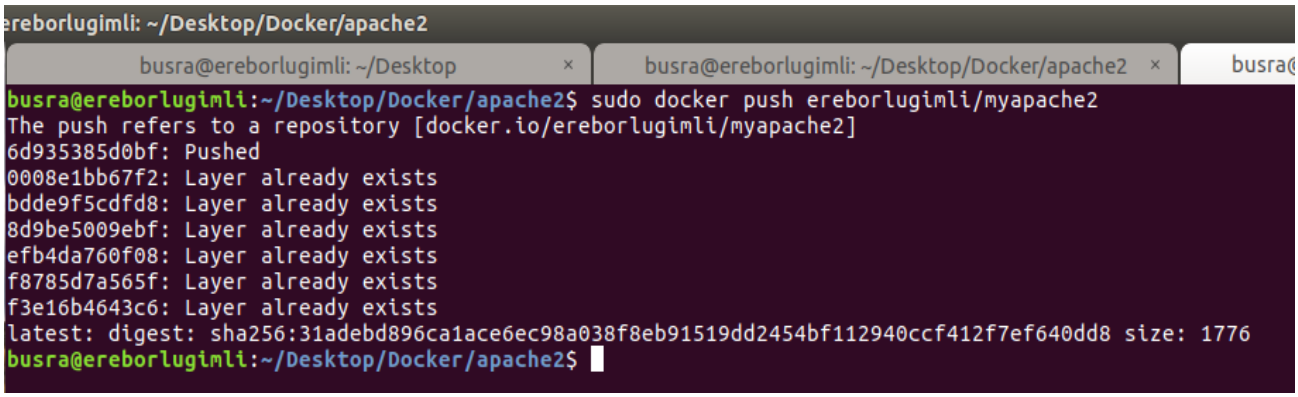
Step 2 :

Oluşturduğumuz docker imagelarını listeliyoruz.



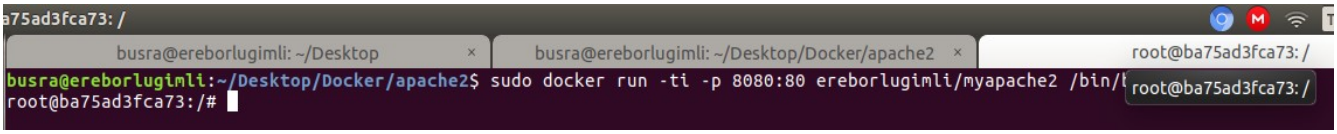
Step 3 :

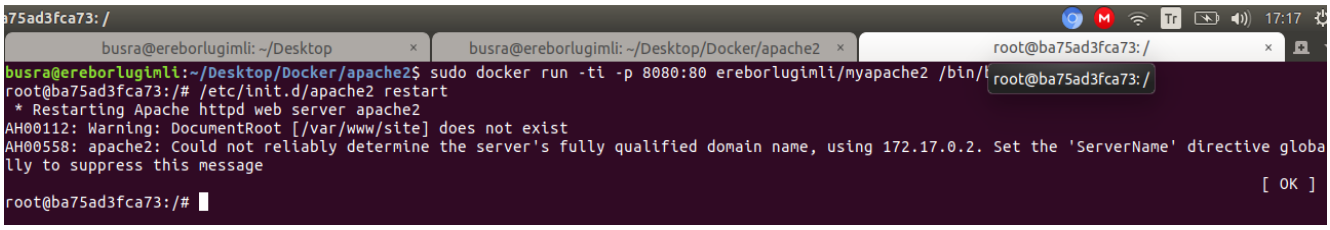
Docker Hub'a oluşturduğumuz image'ı yüklüyoruz.



Step 4 :

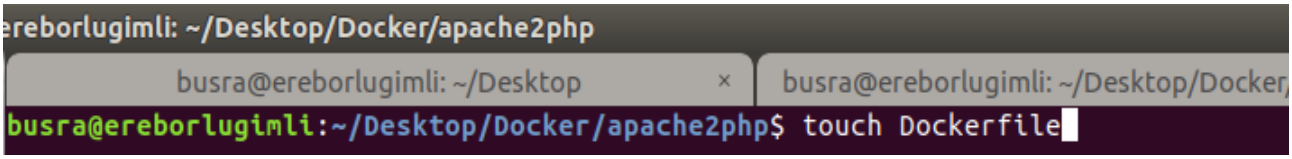
Oluşturduğumuz Container'ın shelline giriyoruz. Ardından apache2 servisini restartediyoruz.

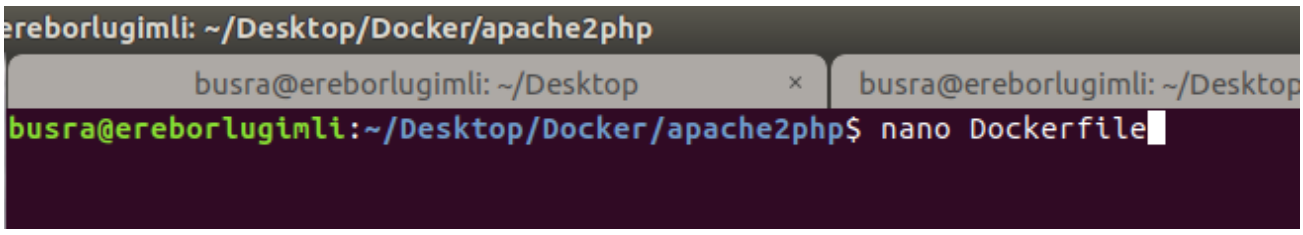




a.2 apache2&&php

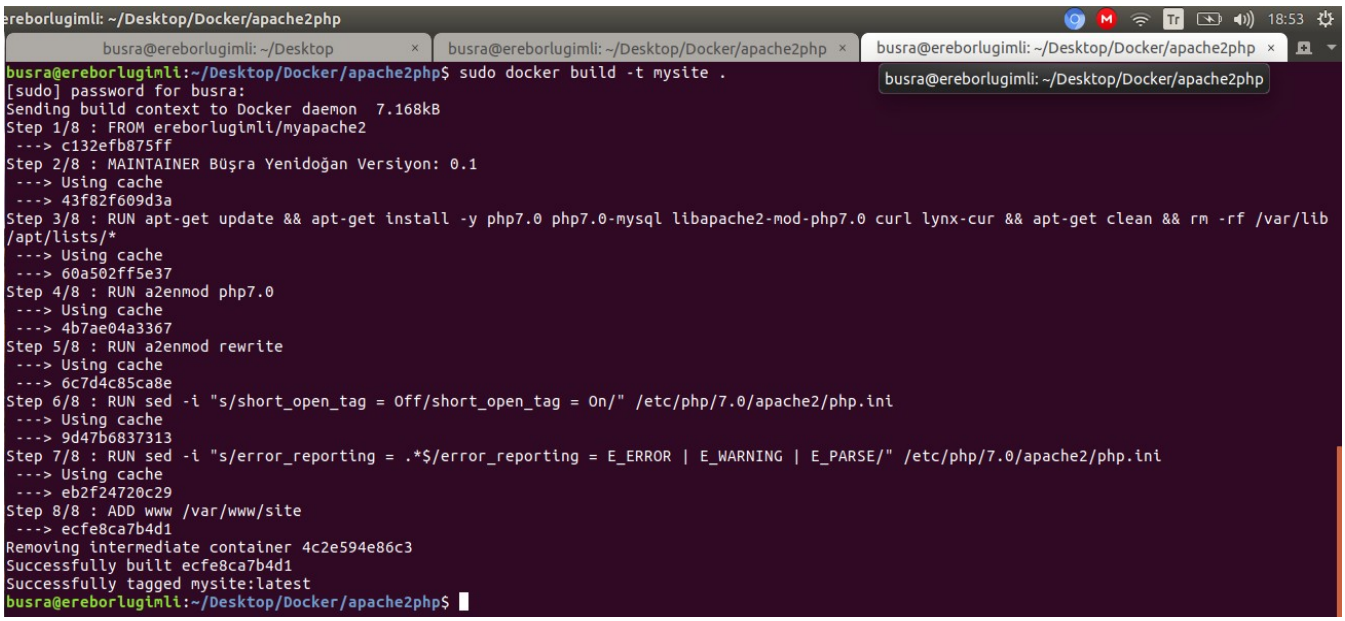
Step 1 :

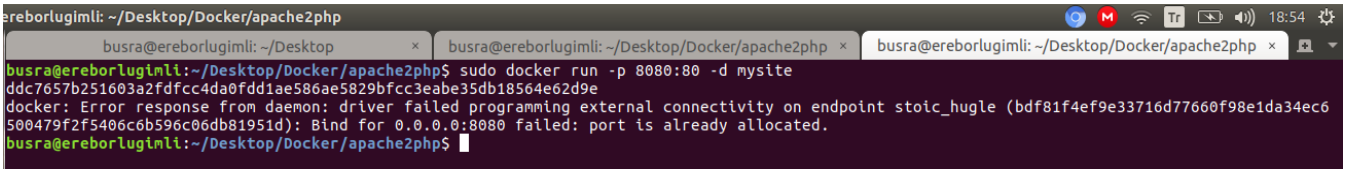




31

Step 2 :





Burda çıktı bir hata aldık dikkat edersiniz bu hatanın sebibi 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 görüntüleyebiliceksiniz.[32]

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"]
```

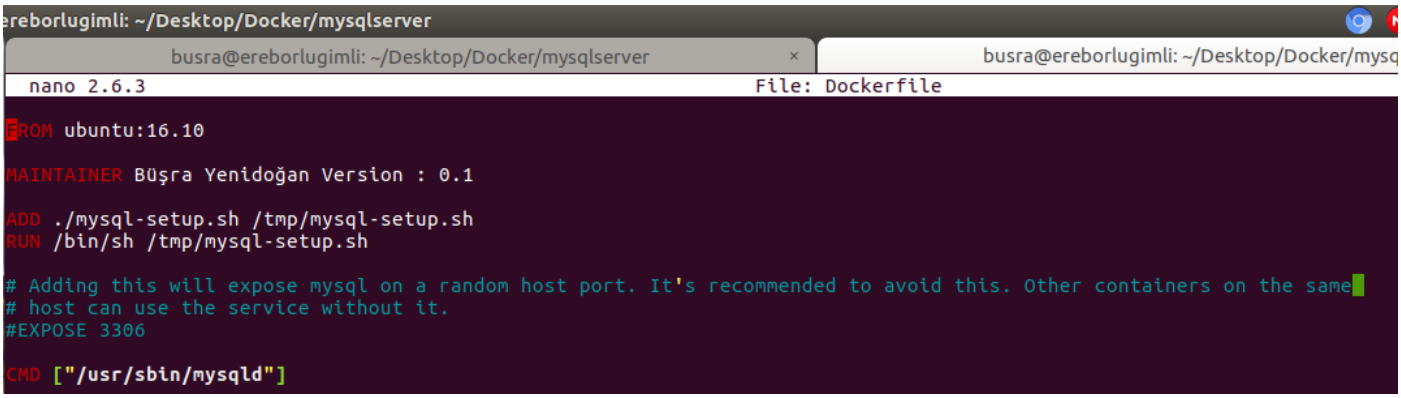
ereborlugimli: ~/Desktop/Docker/mongoDB

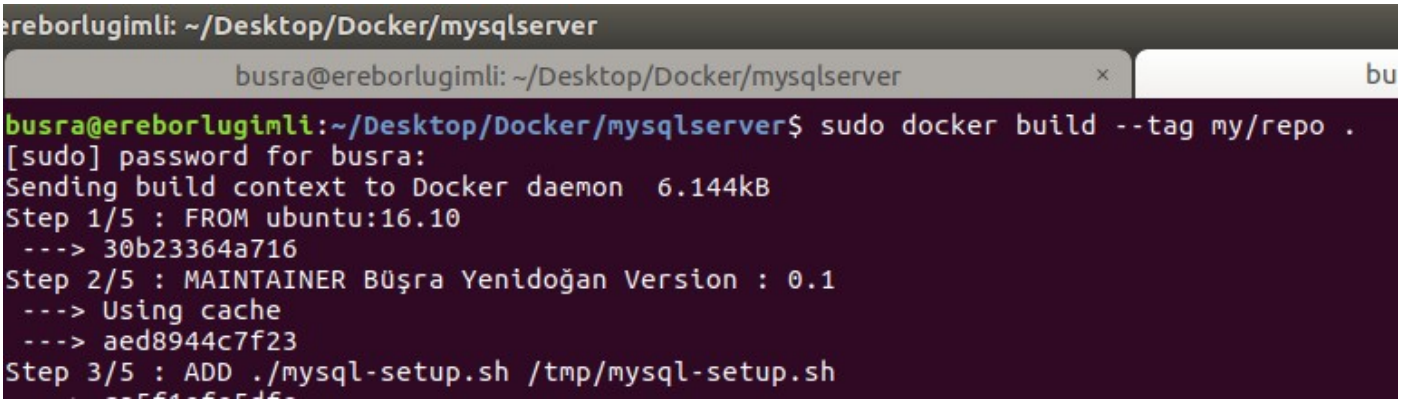
busra@ereborlugimli: ~/Desktop | busra@ereborlugimli: ~/Desktop/Docker/mongoDB | busra@ereborlugimli: ~/Desktop/Docker/mongoDB

```
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
 ---> c312699cff04
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





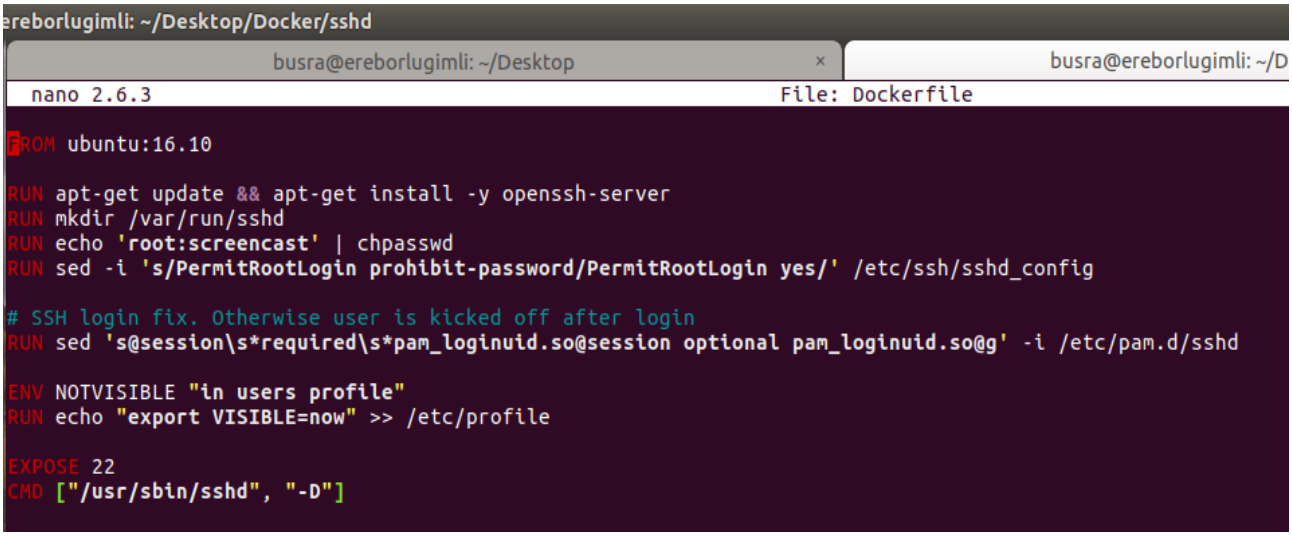
34

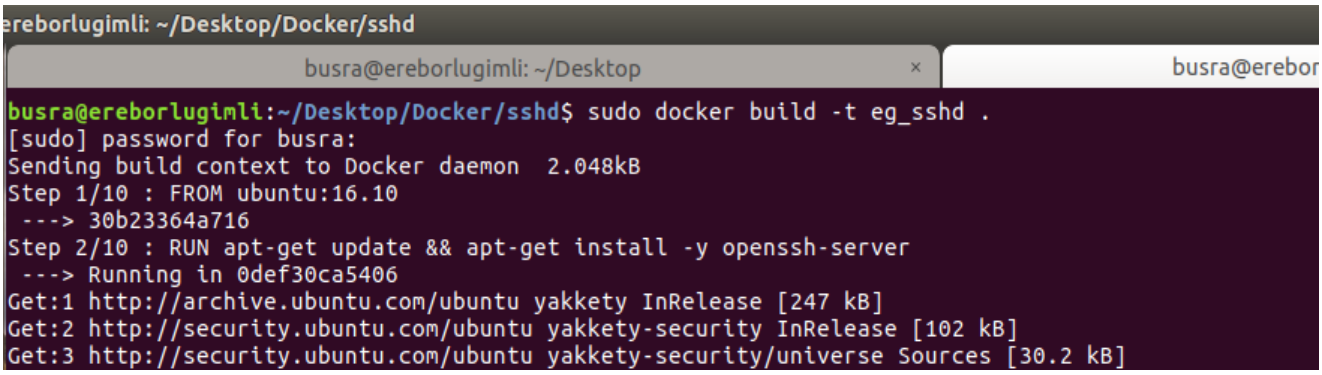
a.5 sshd

Step1 :

Konfg yani ayarlarımızın olduğu Dockerfle dosyamızı yazdıtan sonra containerımızı

yapılandırıyoruz.

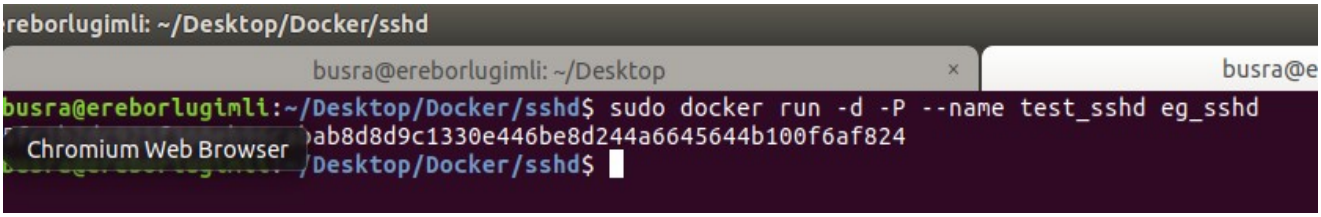


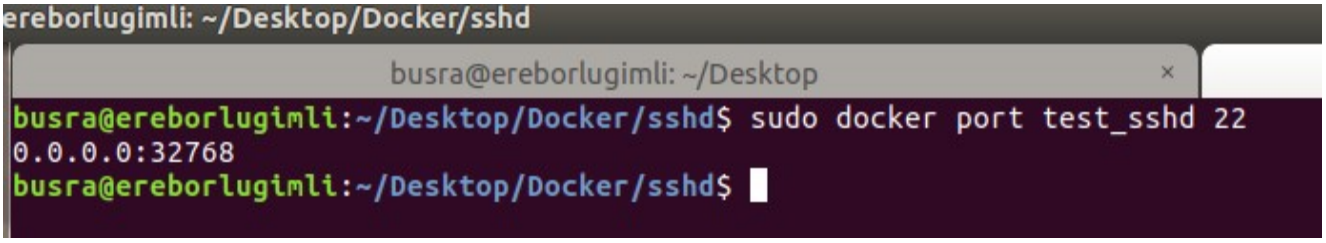


35

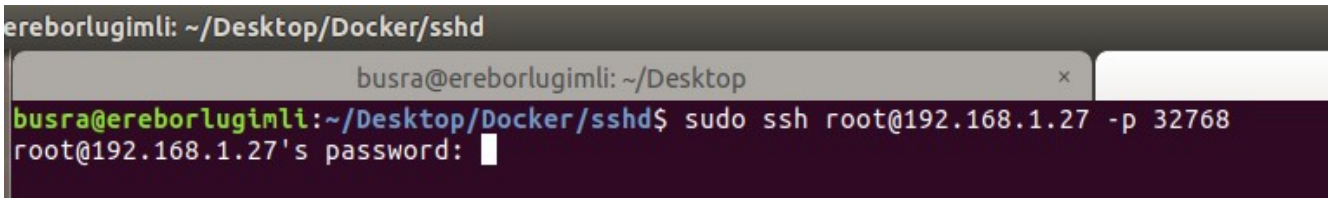
Step 2 :

Containerımızı çalıştırıp kendi IP adresimizden makinaya bağlanıyoruz.





36



KAYNAKLAR

- Using Docker Developing and Deploying Software with Containers

Adrian Mouat

- https://docs.docker.com/

- http://www.gokhansengun.com/[37]

37