Laboratuvarın GitHub Deposunu Klonlayın

Laboratuvarı GitHub’dan klonlamak için aşağıdaki komutu kullanabilirsiniz (komutu tıklayabilir veya manuel olarak yazabilirsiniz). Bu, laboratuvar dizininin bir kopyasını “linux\_tweet\_app” adlı yeni bir alt dizinde oluşturacaktır.

git clone <https://github.com/dockersamples/linux_tweet_app>

### Bir DockerID’niz olmalı

Bir DockerID’niz yoksa (Docker Hub’a erişmek için kullanılan ücretsiz bir giriş), lütfen [Docker Hub] (https://hub.docker.com) adresini ziyaret edin ve bir tane için kaydolun. Daha sonraki adımlar için buna ihtiyacınız olacak.

## **2. Basit Docker konteynerleri çalıştırmak**

Konteynerleri farklı çözümler için kullanabilirsiniz:

1. **Tek bir görevi çalıştırmak için:** Bu bir shell komut dosyası veya bir uygulama olabilir.
2. **Etkileşimli olarak:** Bu, sizi uzak bir konteynere bağlar, sunucuya SSH ile bağlanmanız gibi.
3. **Arka planda:** Web siteleri ve veritabanları gibi uzun süreli hizmetler için.

Bu bölümde bu seçeneklerin her birini deneyecek ve Docker ile iş yükünü nasıl yönetebileceğinizi göreceksiniz.

### Bir Alpine Linux konteynerinde tek bir görev çalıştırmak

Bu adımda yeni bir konteyner başlatacağız ve üzerinde hostname komutunu çalıştıracağız. Konteyner başlayacak, hostname komutunu çalıştıracak ve sonra çıkacaktır.

Linux konsolu üzerinde aşağıdaki komutu çalıştırın.

**docker container run alpine hostname**

Aşağıdaki çıktı ile alpine: latest imajının yerel olarak bulunamadığını göstermektedir. Bu olduğunda Docker, Docker Hub’dan otomatik olarak pull (indirir).

İmaj çekildikten sonra, konteynerin sunucu üzerindeki adı görüntülenir (aşağıdaki örnekte “888e89a3b36b”).

**Unable to find image 'alpine:latest' locally**

**latest: Pulling from library/alpine**

**88286f41530e: Pull complete**

**Digest: sha256:f006ecbb824d87947d0b51ab8488634bf69fe4094959d935c0c103f4820a417d**

**Status: Downloaded newer image for alpine:latest**

**888e89a3b36b**

Docker, konteynerin içinde başlayan işlem devam ettiği sürece konteynerin çalışmasını sağlar. Bu durumda, hostname çıktısı yazılır yazılmaz işlemi biter. Bu, konteynerin durduğu anlamına gelir. Ancak Docker, kaynakları varsayılan olarak silmez, bu nedenle konteyner hala Çıkış durumunda mevcuttur.

Tüm kapsayıcıları listeleyin.

**docker container ls –all**

Alpine konteynerinin Exited durumunda olduğunu göreceksiniz.

1. **Not:** Konteyner ID en soldaki sütunda bulunan ID bilgisidir. Yukarıdaki örnek için bu 888e89a3b36b dur. Bir görevi yerine getiren ve sonra çıkan konteynerler çok faydalı olabilirler. Bir Docker imajını(Kalıbını) yapılandırmak için bir komut dosyası çalıştırarak bir Docker imajı oluşturabilirsiniz. Bunu herkes sadece imajı çalıştırarak yapabilir - komut dosyasına veya ayarların bilgisine ihtiyaç yoktur.

### İnteraktif bir Ubuntu konteyneri çalıştırmak

Docker çalışan bilgisayarınızdan farklı bir Linux sürümünü konteyner olacak çalıştırabilirsiniz.

Şimdi vereceğimiz örnekte, bir Ubuntu çalışan bilgisayar üzerinde Alpine Linux konteyneri çalıştıracağız.

1. Bir Docker konteyneri çalıştırın ve komut satırına(shell) erişin.

**docker container run --interactive --tty --rm alpine /bin/ash**

Bu örnekte Docker için üç parametre veriyoruz, bunlar:

* --interactive interaktif bir oturum açmak istediğinizi belirtiyor.
* --tty pseudo-tty oluşturacaktır.
* --rm Docker için, işlem bittikten sonra konteynerin silinmesi gerektiğini belirtir.

İlk iki parametre sayesinde konteyner ile etkileşim içine girebilirsiniz, böylece konteyner üzerine doğrudan bağlanmış olursunuz.

Konteynere ayrıca ash işlemini (PID 1) çalıştırmasını söylüyoruz.

Konteyner çalışmaya başladığında doğrudan komut satırına root@<container id>:/# bağlanırsınız. Docker konteyner içerindeki komut satırına (shell) doğrudan bağlanmanızı sağlamıştır, bu sayede komut satırından konteyner içersinde direkt komut gönderip alabilirsiniz. Bu işlemi yapabilmek için herhangi bir network bağlantısı yapmadığınızı ve direkt olarak konsol erişimi sağladığımızı unutmayalım.

2- Buradaki komutları konteyner içersinde çalıştırın. ls / konteynerde bulunan kök klasörünün içeriğini listeleyecektir, ps aux konteyner içinde çalışan işlemleri gösterecektir, cat /etc/issue konteyner içinde hangi Linux dağıtımının çalıştığını gösterecektir, bu durumda Alpine linux çıktısı alırsınız.

**ls /**

**ps aux**

**cat /etc/issue**

3- exit yazarak komut satırı arayüzünden(shell) çıkabiliriz. Bunu yaparak ash işlemini sonlandırırsınız, bu ise çalıştığınız konteynerin durmansına sebep olur.

**Exit**

1. **Not:** Başlangıçta --rm parametresini kullandığımız için, Docker konteynerin işlemi bittikten sonra onu silecektir. Bunun anlamı docker container ls --all ile baktığınızda Alpine isminde konteyner yoktur.

4- Çalıştırdığımız işletim sisteminin ubuntu olduğunu aşağıdaki komutu yazdığımızda görürüz.

**cat /etc/issue**

Ubuntu bir makinede Alpine bir konteyner çalıştırabildiğimizi gördük. Daha önce belirtildiği gibi, konteyner içindeki Linux dağıtımının Docker servisini çalıştıran Linux dağıtımıyla eşleşmesine gerek yoktur.

Ancak, Linux konteynerleri, Docker servisi çalışan bir Linux çekirdeği çalıştırmasını gerektirir. Örneğin, Linux konteynerleri doğrudan Windows Docker konteyneri çalışan bilgisayarlarda çalışamaz. Aynısı Windows konteynerleri için de geçerlidir - Windows 10 üzerinde aktive edebileceğiniz WSL2 ile bu sorunu aşabilirsiniz.

Etkileşimli konteynerler, kendi imajınızı oluştururken çok kullanışlıdır. Bir konteyner çalıştırabilirsiniz, uygulamanızı yüklemek için ihtiyacınız olan tüm adımları doğrulayabilirsiniz ve yapılacak işlemleri Dockerfile ‘a kopyalayabilirsiniz.

Bir konteyneri [commit](https://docs.docker.com/engine/reference/commandline/commit/) (yükleyerek) ederek bundan bir imaj oluşturabilirsiniz - fakat bundan her zaman kaçınmalısınız. Bir [Dockerfile](https://docs.docker.com/engine/reference/builder/) dosyası kullanmanız her zaman için Docker imajları oluşturmak için en iyi yöntemdir.

### Arkaplanda çalışan bir MySQL konteyner yaratalım

Çoğu konteyner için en doğru çalıştırılma yöntemi arka planda çalıştırmaktır. Burada basit bir MySQL konteyneri bu şekilde çalıştıralım.

1- Aşağıdaki komutla yeni bir MySQL konteyneri çalıştırın.

**docker container run \**

**--detach \**

**--name mydb \**

**-e MYSQL\_ROOT\_PASSWORD=my-secret-pw \**

**mysql:latest**

* --detach veya -d konteyneri arkaplanda çalıştırmamızı sağlar.
* --name çalışan konteyner için **mydb** ismini tanımlarız.
* -e bir ortam değişkenini tanımlamamızı ve mysql root parolasını verebilmemizi sağlar (NOT: Gerçek sistemler üzerinde bu şekilde bir parola tanımlaması yapmamalısınız).

Yerelde MySQL isimli bir Docker imajı bulunmadığı için docker otomatik olarak Docker hub üzerinden bu imajı indirecektir.

**Unable to find image 'mysql:latest' locallylatest: Pulling from library/mysql**

**aa18ad1a0d33: Pull complete**

**fdb8d83dece3: Pull complete**

**75b6ce7b50d3: Pull complete**

**ed1d0a3a64e4: Pull complete**

**8eb36a82c85b: Pull complete**

**41be6f1a1c40: Pull complete**

**0e1b414eac71: Pull complete**

**914c28654a91: Pull complete**

**587693eb988c: Pull complete**

**b183c3585729: Pull complete**

**315e21657aa4: Pull complete**

**Digest: sha256:0dc3dacb751ef46a6647234abdec2d47400f0dfbe77ab490b02bffdae57846ed**

**Status: Downloaded newer image for mysql:latest**

**41d6157c9f7d1529a6c922acb8167ca66f167119df0fe3d86964db6c0d7ba4e0**

MySQL işlemi çalıştığı sürece Docker bu konteynerin çalıştıracaktır.

2-Çalışan konteynerleri listelemek.

**docker container ls**

Konteynerinizin çalıştığını gözlemleyebilirsiniz.

3-Birkaç tane Docker komutunu çalıştırarak konteyner içersinde ne olduğunu gözlemleyebilirsiniz: docker container logs ve docker container top.

**docker container logs mydb**

MySQL Docker konteyneri içersindeki bilgileri bu şekilde gözlemleyebilirsiniz.

**<output truncated>**

**2017-09-29T16:02:58.605004Z 0 [Note] Executing 'SELECT \* FROM INFORMATION\_SCHEMA.TABLES;' to get a list of tables using the deprecated partition engine. You may use the startup option '--disable-partition-engine-check' to skip this check.**

**2017-09-29T16:02:58.605026Z 0 [Note] Beginning of list of non-natively partitioned tables**

**2017-09-29T16:02:58.616575Z 0 [Note] End of list of non-natively partitioned tables**

Konteyner içersinde çalışan işlemleri kontrol edelim.

**docker container top mydb**

Çalışan (mysqld) MySQL servisini aşağıdaki şekilde görebilirsiniz.

MySQL çalışıyor olmasına rağmen, Docker servisi çalışan bilgisayara hiçbir ağ bağlantı noktası yayınlanmadığı için konteyner içinde izole edilmiştir. Bağlantı noktaları açıkça yayınlanmadıkça ağ trafiği ana bilgisayardan konteynerlara ulaşamaz.

4-Çalışan MySQL sürümünü görmek için için docker container exec kullanabiliriz.

docker container exec konteyner içersinde bir komutu çalıştırmanızı sağlar. Bu örnekte, bizler docker container exec komutunu kullanarak mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version konteyner içerisinde çalıştıracağız.

**docker exec -it mydb \**

**mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD –version**

MySQL sürümü bilgisini bu şekilde alabilirsiniz, ayrıca birde uyarı çıktısını alırsınız.

**mysql: [Warning] Using a password on the command line interface can be insecure.**

**mysql Ver 14.14 Distrib 5.7.19, for Linux (x86\_64) using EditLine wrapper**

5-docker container exec komutunu kullanarak çalışan bir konteyner içerinde yeni bir shell(kabul) arayüzü açabilirsiniz. Aşağıdaki komutu yazdığınızda MySQL içerinde çalışan konteyner (sh) içinde bağlantı sağlayabilirsiniz.

**docker exec -it mydb sh**

Shell arayüzünün farklı olduğunu görebilirsiniz. Şuanda çalıştırdığınız shell arayüzü sh olduğu için bu şekilde bir fark oluşur.

6-Aynı komutu yazarak sürüm kontrolünü tekrar yapalım, fakat şimdi çalıştırdığımız arayüz bağlandığımız sh komut satırından olsun.

**mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version**

exit yazarak buradaki shell oturumunu sonlandırabilirsiniz.

## **3. Bir uygulamayı Docker ile paketlemek ve çalıştırmak**

Bu adımda, kendi uygulamalarınızı Docker imajları olarak nasıl paketleyebileceğimizi öğreneceğiz, bunu yaparken [Dockerfile](https://docs.docker.com/engine/reference/builder/) (<https://docs.docker.com/engine/reference/builder/>) dosyaları kullanacağız.

Dockerfile dosya yapısı oldukça basittir. Buradaki işlem sırasında, bir Dockerfile’dan basit bir NGINX web sitesi oluşturacağız.

### Basit bir websitesi imajı oluşturmak

Tweet gönderebileceğimiz basit bir web sitesi yapalım, bu işlem için kullanacağımız bir Dockerfile gerekecektir.

1-linux\_tweet\_app klasörü içersine gidelim.

**cd ~/linux\_tweet\_app**

**2-**Dockerfile dosyası içeriğine bakalım.

**cat Dockerfile**

Dockerfile’daki bu satırların her birinin ne yaptığına bir bakalım:

* [FROM](https://docs.docker.com/engine/reference/builder/#from) Oluşturmakta olduğumuz yeni imaj için başlangıç noktası olarak kullanılacak temel görüntüyü belirtir. Bu örnek için nginx:latest kullanıyoruz.
* [COPY](https://docs.docker.com/engine/reference/builder/#copy) docker servisi çalışan bilgisayar üzerinden kopyalanacak dosyaları tanımlar. Buradaki örnekte, COPY kullanılarak iki dosya imaj içerisine kopyalanır: index.html, ve linux.png isimli bir grafik dosyasıdır.
* [EXPOSE](https://docs.docker.com/engine/reference/builder/#expose) hangi ağ portlarının uygulama tarafından erişileceğini belirler.
* [CMD](https://docs.docker.com/engine/reference/builder/#cmd) imajdan bir konteyner başlatıldığında hangi komutun çalıştırılacağını belirtir. Burada çalışacak komutu ve çalışma sırasında kullanılacak argümanları belirtebiliriz.

3-Kopyala/Yapıştır işlemlerini kolaylaştırmak için, DockerID bilginizin olduğu bir ortam değişkenini çıktı olarak alın(Henüz bir docckerID ‘ye sahip değilseniz lütfen [Docker Hub](https://hub.docker.com/) (<https://hub.docker.com/>)üzerinden alın).

DockerID bilginizin yazılabilmesi için aşağıdaki bilginin manüel olarak girilmelidir.

export DOCKERID=<your docker id>

4=Doğru şekilde saklandığından emin olmak için değişkenin değerini terminale çıktı olarak gönderin. !! Önemli, ortam değişkeni olarak mutlaka bir DOCKERID tanımınızın olması gerekir, eğer Docker Hub üzerinden bir DockerID almadıysanız, kendi isiminizi tanımlayabilirsiniz.!!

**echo $DOCKERID**

Boş geliyorsa, test işleminizi kolaylaştırmak için (kendi adınızı kullanın)

**export DOCKERID=senol** gibi bir tanım yapabilirsiniz.

5-docker image build komutu ile Dockerfile dosyasını kullanarak bir Docker imajı oluşturabiliriz.

* --tag imaja farklı bir isim vermemizi sağlar. Bu durumda DockerID, uygulama adı ve bir sürümden oluşan bir isim tanımlarız. Docker ID’nin isim içersine eklenmesi, daha sonra imajı **Docker Hub**‘da saklamamıza izin verir. a
* . Docker’a şuan bulunduğumuz dizini kullanmamızı söyler.

komutun sonunda (.) (nokta) olduğundan emin olun.

**docker image build --tag $DOCKERID/linux\_tweet\_app:1.0 .**

6- docker container run komutunu kullanarak yeni oluşturduğunuz imaj ile yeni bir konteyner yaratabilirsiniz.

Bu konteyner bir NGINX web sunucusu çalıştıracağı için --publish parametresini kullanarak 80 portunu sunucu üzerindeki 80 portuna çıktı olarak gönlendirebiliriz. Bu sayede Docker servisini çalıştıran bilgisayara gelen 80 portu (Web sunucusu) isteklerini konteyner içersindeki 80 portuna yönlendirmiş olursunuz. --publish parametresinin kullanımı şu şekildedir Sunucu\_Port:Konteyner\_Port.

**docker container run \**

**--detach \**

**--publish 80:80 \**

**--name linux\_tweet\_app \**

**$DOCKERID/linux\_tweet\_app:1.0**

Sunucu üzerine 80 numaralı bağlantı noktasına(port) gelen herhangi bir dış trafik artık 80 numaralı bağlantı noktasındaki konteyner’a yönlendirilecektir.

Daha sonraki bir adımda, iki farklı bağlantı noktasından gelen trafiği nasıl eşleştireceğinizi göreceksiniz - bu çok gereklidir, çünkü sunucu üzerindeki port numaraları sadece bir kere kullanılabilir.

[çalıştırdığınız web sitesine buradan erişin](https://egitim.acikkaynakfikirler.com/) (<http://ip10-3-163-2-cbei7fo6rqngcdno9ktg-80.direct.lab2.acikkaynakfikirler.com/>) **güvenlik dolaysı ile erişim kapalıdır!!!**.

Web sitesine bir kez erişebildikten sonra durdurup silebiliriz.

**docker container rm --force linux\_tweet\_app**

**Not:** Burada --force parametresini çalışan konteyneri durdurup silmek için kullandık. Bu komut konteyneri doğrudan durdurup, durmuş olan konteyner kalıcı olarak silinecektir.

1. Canlı ortam olarak kullandığınız bir platformda docker container stop komutunu kullanıp servisleri normal şekilde durdurmak gerekir. Daha sonrasında docker container rm ile kalıcı şekilde konteyner silinebilir.

## **4. Çalışan bir Web sitesini değiştirmek**

Bir uygulama üzerinde çalışırken, kaynak kodunuzda her değişiklik yaptığınızda, konteyneri durdurmanız, imajı yeniden oluşturmanız ve yeni bir sürüm çalıştırmanız çok kullanışlı bir yöntem değildir.

Bu işlemi kolaylaştırmanın bir yolu, yerel makinedeki kaynak kodu dizinini çalışan konteyner içersine bağlamaktır. Bu, Docker servisi çalıştıran sunucudaki dosyalarda yapılan herhangi bir değişikliğin konteyner içersinden anında algılanmasına olanak verir.

Bu işlemi [bind mount](https://docs.docker.com/engine/admin/volumes/bind-mounts/) (<https://docs.docker.com/storage/bind-mounts/>) dediğimiz bir yöntem ile yapabiliriz.

bind mount yöntemini kullandığınızda, sunucu üzerindeki (Docker servisinin çalıştığı) bir dosya veya dizini çalışan konteyner üzerine bağlamış olursunuz.

### Web uygulamamızı bir “bind mound” ile başlatmak

1-Web uygulamamızı başlatıp bulunduğumuz klasörü çalışacak konteyner üzerine bağlayalım.

Bu örnek içersinde bulunduğumuz dizini(PWD çıktısı) --mount parametresini kullanarak konteyner içersine /usr/share/nginx/html dizinine bağlayalım.

Bu komutu linux\_tweet\_app dizini içersinden çalıştırdığınıza emin olun.

**docker container run \**

**--detach \**

**--publish 80:80 \**

**--name linux\_tweet\_app \**

**--mount type=bind,source="$(pwd)",target=/usr/share/nginx/html \**

**$DOCKERID/linux\_tweet\_app:1.0**

Dockerfile dosyası içinden anladığımız üzere usr/share/nginx/html konumu web sunucusunun dosyalara eriştiği kök dizini.

1. [website](https://egitim.acikkaynakfikirler.com/) site erişilebilir olmalıdır, **güvenlik sebebiyle bloklanmıştır!!!**.

komut satırından “curl localhost” diyerek URL çıktısını görebilirsiniz.

### Çalışan web uygulamasını değiştirelim

Bind mount çözümleri, yerel dosya sisteminde yapılan herhangi bir değişikliğin aynı anda çalışan konteynera etki etmesi anlamına gelir.

1-Yeni bir index.html konteyner içersine kopyalayın.

Daha önce indirdiğiniz Git raposu içinde bulunan index.html dosyasının birkaç farklı sürümünü mevcuttur. Manuel olarak ls komutunu ~/linux\_tweet\_app klasörü içinde çalıştırarak dosyaları görebilirsiniz. Bu adımda index.html dosyasını index-new.html dosyası ile değiştireceğiz.

**cp index-new.html index.html**

2-Çalışan [website](https://egitim.acikkaynakfikirler.com/) sitesine gidip **ekranı yenilersek** veya komut satırı üzerinden curl localhost dediğimizde. Sitenin değiştiğini göreceğiz.

vi kullanabiliyorsanız klasör içinde bulunan index.html dosyasına farklı değişiklikler yapabilirsiniz. Sayfayı güncellediğinizde yine değişikliklerin etkin olduğunu görürsünüz. curl localhost. Daha fantastik şeyler de yapabiliriz, bunun için exec komutunu kullanabiliriz. exec echo "ABC">>index.html dediğinizde dosyanın içini doğrudan değişitrebilirsiniz.

index.html değiştirmemize ve yerelde bulunan dosyamızı değiştirmemizin nasıl bir değişiklik yarattığını gördük, docker imajını değiştirmemiş olduk, konteyner’in yeniden başlatılması gerekmez.

Bunu göstermek için, çalışan konteynerinizi durdurun ve 1.0 isimli imajınızı bind mount olmadan çalıştırın.

1-Konteyneri durdurun ve silin.

**docker rm --force linux\_tweet\_app**

2-Varolan sürümü bind mount olmadan çalıştırın.

**docker container run \**

**--detach \**

**--publish 80:80 \**

**--name linux\_tweet\_app \**

**$DOCKERID/linux\_tweet\_app:1.0**

3-Farkedeceğiniz üzere [website](https://egitim.acikkaynakfikirler.com/) sorguladığınızda curl localhost ilk çıktıyı görürüz.

4-Konteyneri durdurun ve silin

**docker rm --force linux\_tweet\_app**

### İmajı güncelleyelim

index.html dosyası içinde yaptığımız değişikliklerin kalıcı olabilmesi için yeni bir imaj sürümü yapmalıyız.

1-Yeni bir imaj oluşturun ve 2.0 ile tag tanımı yapın.

index.html dosyasını Docker servisi çalışan sunucu üzerinde değiştirdiğimizi anlayabiliriz. Bunun anlamı yeni bir docker image build komutu çalıştırdığınızda güncellenmiş index.html yeni imaj içersinde bulunacaktır.

(.) komut sonunda eklemeyi unutmayınız lütfen.

**docker image build --tag $DOCKERID/linux\_tweet\_app:2.0 .**

İmajın ne kadar hızlı oluştuğu farketmişsinizdir! Bunun sebebi Docker imajın sadece gerekli yerlerini güncelledi. Bu sayede imajın tüm katmanlarının tekrar oluşturulmasıne gerek kalmaz.

2-şimdi sistem üzerindeki imajı tekrar kontrol edelim.

**docker image ls**

Web uygulamasının her iki sürümünün de imajların içersinde olduğunu görebilirsiniz.

### Yeni sürümü test edelim

1-İmajın yeni sürümü ile yeni bir konteyner çalıştıralım.

**docker container run \**

**--detach \**

**--publish 80:80 \**

**--name linux\_tweet\_app \**

**$DOCKERID/linux\_tweet\_app:2.0**

2-Yeni sürümün çalıştığını [website](https://egitim.acikkaynakfikirler.com/) sitesi üzerinden kontrol edebiliriz veya curl localhost ile bu çıktıyı alabilirsiniz.

Her iki sürümü aynı anda çalıştırabiliriz. Dikkat etmemiz gereken tek konu 80 portunu her iki konteyner için kullanamayacak olmamız.

“80” portunu 2.0 sürümü imajımızdan oluşan konteynerimizi çalıştırmak için kullanıyoruz. Yeni çalıştıracağımız konteyneri “8080” portu üzerinden çalıştıracağız. Ek olarak çalıştıracağımız konteyner için farklı bir isim vermemiz gerekiyor (old\_linux\_tweet\_app)

3-Şimdi çalıştıracağımız konteyneri imajın eski sürümünden başlatıyoruz.

Buradaki konteynerin 8080 portunu Docker sunucusu üzerine eşleştirdiğine dikkat edin. Bunun temel sebebi Docker sunucusu üzerindeki portların sadece bir kere kullanılabilmesi, aynı port üzerinde birden fazla network eşleştirmesi yapamayız.

**docker container run \**

**--detach \**

**--publish 8080:80 \**

**--name old\_linux\_tweet\_app \**

**$DOCKERID/linux\_tweet\_app:1.0**

4-Çalışan diğer konteyner üzerine buradan [website](https://egitim.acikkaynakfikirler.com/) veya curl http://localhost:8080 üzerinden erişebilirsiniz.

### Oluşturduğunuz imajı Docker Hub üzerine yüklemek

1-Docker sunucunuz üzerindeki imajları listeleyin.

**docker image ls -f reference="$DOCKERID/\*"**

iki tane linux\_tweet\_app isimli imajınız olduğunu göreceksiniz - bir tanesi 1.0 ile TAG’li diğeri 2.0 ile TAG’lenmiş.

Bu imajlar sadece Docker çalıştırdığınız sunucu üzerindeki yerel sunucudadır. Buradaki işleminiz bittiğinde bu Docker imajı silinecektir. Buradaki adım ile oluşturduğumuz docker imajını internete açık olan bir repository üzerine yükleyeceğiz.

Dağıtım Docker platformu üzerinde oluşturulur. Yerel olarak imajları oluşturabilir, bunları da özel veya genel [registry](https://docs.docker.com/registry/) ‘lere(<https://docs.docker.com/registry/> ) yükleyebilirsiniz, bu sayede diğer kullanıcılar bu imajları kullanabilir. Erişimi olan herhangi birisi, sizin oluşturduğunuz imajı indirip, imajınız ile konteyner yaratabilir. Bu imaj ile yaratılacak konteynerlerin davranışları her zaman için aynı olacaktır, çünkü imajın içinde tamamen konfigüre edilmiş bir uygulama mevcuttur - bu imajı çalıştırmak için tek ihtiyacınız olan şey Docker çalıştıran bir linux sunucu olacaktır.

Size önerim burada oluşturduğunuz imajı kendi bilgisayarınıza indirip çalıştırmanızdır, arada hiç bir farkın olmadığını görebilirsiniz.

[Docker Hub](https://hub.docker.com/) docker imajları için varsayılan docker registry(kayıt defteri) platformudur.

2-İmajlarınızı yüklemeden önce, Docker Hub üzerine bağlanmanız gerekli.

**docker login**

Sorulduğunda docker ile ilgili kullanıcı/parola bilgilerini girmelisiniz.

3-Uygulamanızın 1.0 sürümünü docker image push komutu ile yükleyebilirsiniz.

**docker image push $DOCKERID/linux\_tweet\_app:1.0**

Docker hub üzerine yükleme sırasındaki ilerleme durumunu gözlemleyebilirsiniz.

4-Şimdi sürüm 2.0 yükleyebiliriz.

**docker image push $DOCKERID/linux\_tweet\_app:2.0**

Birkaç satır içersindeki Layer already exists yazısını görebilirsiniz. Bunun sebebi docker hub içersine yüklediğiniz konteyner katmanlarından var olanları kopyalanmaz, sadece değişiklik olanlar yüklenecektir.

İnternet üzerinden https://hub.docker.com/r/<Docker ID'niz>/ adresine giderseniz, yüklediğiniz imajı görebilirsiniz. Docker hub üzerinde yüklenmiş olan imajlar genel kullanıma açıktır, ve herhangi birisi sizin yüklediğiniz imajı indirip kullanabilir. Bu işlemleri yapmak için dockerID’nizin olmasına dahi gerek yoktur. O yüzden imajlarınızın içinde şifre ve özel bilgilerin olmamasına mutlaka dikkat etmelisiniz.