Docker CLI Temelleri

docker pull, docker images, docker run, docker ps, docker logs, docker exec, docker stop /start, ve docker rm -- bir PostgreSQL container'ının tam yaşam döngüsü üzerinden, gerçek örneklerle anlatılıyor.

Başlangıç 25 dk

Son güncelleme: 31 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/docker-cli-fundamentals

---

"Docker Nedir?" kelime dağarcığını kurdu -- image, container, Docker Engine, Docker Hub. Bu ders, o kelime dağarcığını komut satırında işe koşuyor: bir image çekmek, onu bir container olarak çalıştırmak, ne çalıştığını incelemek, içine bakmak, ve sonra temizlemek. Buradaki her komut, "Docker Engine"de tarif edilen Docker Engine daemon'ıyla konuşuyor -- `docker` binary'si yalnızca istekleri yazan client.

Bir Image Çekmek: `docker pull`

`docker pull`, bir registry'den (varsayılan olarak Docker Hub, bkz. "Docker Hub ve Image Registry'leri") bir image getirir ve henüz hiçbir şey başlatmadan yerel olarak saklar.

```
docker pull postgres:16
```

```
16: Pulling from library/postgres
a2318d6c47ec: Pull complete
c9c5e91e622b: Pull complete
...
Status: Downloaded newer image for postgres:16
docker.io/library/postgres:16
```

`postgres` kısmı image'ın **repository adı**dır; iki noktadan sonraki `16` ise **tag**'idir -- bir repository'nin bir image'ın birden çok sürümüne iliştirebileceği bir etiket. Bir tag'i tamamen atlamak (`docker pull postgres`) örtük olarak `:latest` anlamına gelir, ki bu ona güvenmek yerine açıkça isimlendirilmeye değer -- "Best Practices" tam olarak nedenine geri dönüyor.

Yerel Image'ları Listelemek: `docker images`

Bir image çekildikten (ya da inşa edildikten -- "Docker İmajları ve Dockerfile"da işleniyor) sonra, kaldırılana kadar yerel depolamada durur. `docker images`, o anda orada olan her şeyi listeler:

```
docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
postgres     16        a1b2c3d4e5f6   2 weeks ago   434MB
```

Bu tamamen yerel, çevrimdışı bir listelemedir -- Docker Hub'a hiç ulaşmaz. Burada görünen bir image, "Image'lar vs. Container'lar"ın tarif ettiği tam olarak şeydir: henüz hiçbir şey çalıştırmayan, dondurulmuş bir şablon.

Bir Container Çalıştırmak: `docker run`

`docker run`, bir image'dan tek bir adımda bir container oluşturur ve başlatır -- image henüz yerel değilse önce otomatik olarak çeker.

```
docker run --name learning-platform-db \
  -e POSTGRES_PASSWORD=secret \
  -p 5432:5432 \
  -d postgres:16
```

```
7f8e9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f
```

Sürekli karşına çıkacağı için tek tek isimlendirilmeye değer dört flag:

```
--name <isim>       container için rastgele bir isim yerine, insan tarafından okunabilir
bir isim
-e ANAHTAR=DEĞER     container içinde bir ortam değişkeni ayarlar (buradaki
POSTGRES_PASSWORD,
                      resmi postgres image'ının admin şifresini ayarlamayı beklediği yol)
-p HOST:CONTAINER    bir port yayınlar -- bu kursta ileride "Docker Networking"de tam
olarak işleniyor
-d                   "detached" -- arka planda çalışır ve komut istemini hemen geri döndürür
```

Yazdırılan uzun karakter dizisi, yeni container'ın ID'sidir -- `docker run`'ı `-d` olmadan çalıştırmak, terminalinizi doğrudan container'ın çıktısına bağlardı, ki bu hızlı bir foreground kontrolü için kullanışlıdır ama container durana kadar terminali bloke eder.

Çalışan Container'ları Listelemek: `docker ps`

`docker ps`, container'ları listeler, varsayılan olarak yalnızca şu anda çalışanları:

```
docker ps
```

```
CONTAINER ID   IMAGE         COMMAND                  STATUS
PORTS                     NAMES
7f8e9a0b1c2d   postgres:16   "docker-entrypoint.s…"   Up 2 minutes   0.0.0.0:5432->5432
/tcp   learning-platform-db
```

> `docker ps` tek başına yalnızca çalışan container'ları gösterir -- bir an önce durdurduğun bir container ("Container'ı Durdurmak ve Başlatmak: `docker stop` / `docker start`"a bkz.) görünmez. Duruma bakılmaksızın her container'ı görmek için `-a` ekle (`docker ps -a`), ki bu genelde beklediğin bir container "kayıp" göründüğünde çalıştırılacak ilk şeydir.

Container Çıktısını Okumak: `docker logs`

Bir container'ın standart çıktısı ve standart hatası, detached çalışsa bile Docker tarafından yakalanır -- `docker logs`, bunları sonradan okumanın yoludur.

```
docker logs learning-platform-db
```

```
PostgreSQL init process complete; ready for start up.
...
database system is ready to accept connections
```

Log akışını sürekli takip etmek için `-f` ekle (`docker logs -f learning-platform-db`), tam olarak `tail -f`'in büyüyen bir dosyayı takip ettiği gibi -- bir container beklendiği gibi davranmıyorsa, daha karmaşık herhangi bir şeyden önce genelde başvurulacak ilk şey budur.

Bir Container'ın İçine Girmek: `docker exec`

`docker run` *yeni* bir container başlatır; `docker exec` ise **zaten çalışan** bir container'ın içinde *ek* bir komut çalıştırır -- en yaygın olarak, içinde interaktif bir shell almak için.

```
docker exec -it learning-platform-db psql -U postgres
```

```
psql (16.x)
Type "help" for help.

postgres=#
```

`-i` standart girişi açık tutar, `-t` bir terminal ayırır -- birlikte (`-it`) bunlar, oturumu tek bir komut çalıştırıp hemen çıkmak yerine interaktif yapan şeydir. Burada doğrudan `psql`'i çalıştırıyor, çünkü resmi `postgres` image'ının içinde zaten kurulu; bunun yerine `docker exec`

`-it learning-platform-db sh` çalıştırmak, dosyaları ya da ortam değişkenlerini doğrudan incelemek için kullanışlı, container içinde düz bir shell açardı.

Container'ı Durdurmak ve Başlatmak: `docker stop` / `docker start`

Çalışan bir container durdurulabilir, ve durmuş bir container -- henüz kaldırılmadığı sürece -- aynı isimle, aynı konfigürasyonla, ve ("Docker Volumes"a bkz., bu kursta ileride) aynı veriyle tekrar başlatılabilir.

```
docker stop learning-platform-db
```

```
learning-platform-db
```

```
docker start learning-platform-db
```

```
learning-platform-db
```

`docker stop` önce nazik bir kapatma sinyali gönderir ve yalnızca bir timeout'tan sonra zorla sonlandırır -- tam olarak bir Spring Boot uygulamasının doğrudan öldürülmek yerine kendi shutdown hook'larını çalıştırma şansı bulmasının aynı nedeni.

Temizlemek: `docker rm`

`docker rm`, **durmuş** bir container'ı kalıcı olarak kaldırır -- dosya sistemi, konfigürasyonu, o belirli container örneği hakkında her şey (oluşturulduğu image değil, o `docker images`'ta dokunulmadan kalır).

```
docker stop learning-platform-db
docker rm learning-platform-db
```

```
learning-platform-db
learning-platform-db
```

> `docker rm`, hâlâ çalışan bir container'ı kaldırmayı reddeder -- container'ı çalışıyor olarak adlandıran bir hata alırsın. `docker rm -f <isim>` durdurma adımını atlar ve doğrudan zorla kaldırır, ama bu nazik bir kapatma değil, sert bir kill'dir; container disk'e yazma flush etmek gibi temiz kapanmadan faydalanan bir şey yapıyorsa önce `docker stop`'u tercih et.

Hepsini Bir Araya Getirmek: PostgreSQL'i Bir Container'da Çalıştırmak

Tam yaşam döngüsü -- çekmek, çalıştırmak, doğrulamak, incelemek, ve temizlemek -- baştan sona, bu bölümün inşa ettiği aynı PostgreSQL container'ıyla, ve bu projenin kendi veritabanının eninde sonunda içinde çalışacağı tam container ("Docker Compose", bir sonraki kategoride, tam bir Spring Boot + PostgreSQL kurulumu için bu aynı sırayı otomatikleştiriyor).

```
#!/bin/sh
# A developer needs a local PostgreSQL instance to run this project against,
# without installing PostgreSQL itself on their own machine.

docker pull postgres:16
# Output (trimmed):
# Status: Downloaded newer image for postgres:16

docker run --name learning-platform-db \
  -e POSTGRES_PASSWORD=secret \
  -p 5432:5432 \
  -d postgres:16
# Output:
# 7f8e9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f

docker ps
# Output:
# CONTAINER ID   IMAGE         STATUS         PORTS                     NAMES
# 7f8e9a0b1c2d   postgres:16   Up 2 seconds   0.0.0.0:5432->5432/tcp   learning-platform-db

docker logs learning-platform-db
# Output (trimmed):
# database system is ready to accept connections

# The database is now reachable at localhost:5432, exactly as if PostgreSQL
# had been installed directly -- but running as an isolated container instead.

docker exec -it learning-platform-db psql -U postgres -c "SELECT version();"
# Output:
# PostgreSQL 16.x on x86_64-pc-linux-gnu, compiled by gcc ...

# Work is done for the day -- stop the container without deleting it.
docker stop learning-platform-db
# Output:
# learning-platform-db

# The next morning, resume with the exact same data still intact.
docker start learning-platform-db
# Output:
# learning-platform-db

# Done with this database entirely -- remove it for good.
docker stop learning-platform-db
docker rm learning-platform-db
# Output:
# learning-platform-db
# learning-platform-db

# The postgres:16 image itself is untouched -- "docker images" still lists it,
# ready to start a brand-new container from at any time.
```

Burada hiçbir şey yeni bir mekanizma değil -- her tek komut yukarıda zaten tanıtıldı; bu yalnızca onları gerçek bir çalışma oturumunda gerçekten yazılacakları şekilde zincirlenmiş olarak gösteriyor.

Yaygın Hatalar

- Hâlâ çalışan bir container'da `docker rm` çalıştırıp hatayla şaşırmak -- önce durdur, ya da `-f`'i kazara değil bilinçli olarak kullan.
- `-d`'yi unutup terminal "asılı" göründüğünde kafası karışmak -- olmadan, `docker run` container'ın çıktısına foreground'da bağlanır.
- `docker ps`'in var olan her container'ı gösterdiğini varsaymak -- varsayılan olarak yalnızca çalışanları gösterir; tam resim için `docker ps -a` kullan (bkz. "Çalışan Container'ları Listelemek: `docker ps`").
- Yepyeni bir container başlatmak için `docker exec`'e başvurmak -- `exec` yalnızca zaten çalışan bir container'a karşı çalışır; yeni bir container `docker run`'dan gelir.

Best Practices

- Her zaman belirli bir tag'i çek ve çalıştır (`postgres:16`), örtük `:latest`'i değil -- çalışan bir sistemin altında sessizce güncellenen bir image, bilinen bir sürüme sabitlenmiş olandan çok daha zor akıl yürütülen bir şeydir.
- Container'lara Docker'ın rastgele ürettikleri yerine anlamlı `--name` değerleri ver -- `learning-platform-db`, sonraki her komutta rastgele bir isimden çok daha kolay doğru şekilde referans verilebilir.
- Bir container yanlış davranıyor gibi göründüğünde, daha karmaşık herhangi bir şeyden önce `docker logs -f`'e başvur -- erken hata ayıklama sorularının çoğu, container'ın kendisinin zaten yazdırdığı şeylerle cevaplanır.
- Beklediğin bir container kayıp göründüğünde ilk sorun giderme adımı olarak `docker ps -a`'yı kullan (yukarıdaki "Tip"e bkz.).

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- `docker pull`, bir registry'den hiçbir şey başlatmadan bir image getirir; `docker images` yerel olarak saklanan şeyleri listeler.
- `docker run`, bir image'dan yeni bir container oluşturur ve başlatır; `docker ps` şu anda çalışan container'ları listeler (`docker ps -a` hepsi için).
- `docker logs`, bir container'ın yakalanmış çıktısını okur; `docker exec`, zaten çalışan bir container içinde -- tipik olarak interaktif bir shell -- ek bir komut çalıştırır.
- `docker stop` / `docker start`, var olan bir container'ı silmeden duraklatır ve devam ettirir; `docker rm`, durmuş bir container'ı kalıcı olarak siler (geldiği image'a dokunulmaz).
- Bunların hepsi, "Docker Nedir?"de tarif edilen Docker Engine daemon'ıyla konuşan client komutlarıdır -- CLI'ın kendisi gerçek işin hiçbirini yapmaz.

Cheat Sheet

```
docker pull <image>:<tag>          # bir registry'den image getir
docker images                      # yerel saklanan image'ları listele
docker run -d --name <isim> <img>  # container oluştur + başlat, detached
docker ps                          # çalışan container'ları listele
docker ps -a                       # her durumdaki tüm container'ları listele
docker logs <isim>                 # container çıktısını oku
docker logs -f <isim>              # container çıktısını canlı takip et
docker exec -it <isim> <komut>     # çalışan bir container içinde komut çalıştır
docker stop <isim>                 # çalışan bir container'ı nazikçe durdur
docker start <isim>                # durmuş bir container'ı devam ettir
docker rm <isim>                   # durmuş bir container'ı kalıcı olarak kaldır
```

Terimler Sözlüğü

- **Tag**: bir repository içindeki bir image'ın belirli bir sürümüne iliştirilen etiket (`postgres:16`); atlanması `:latest` anlamına gelir.
- **Detached mod (`-d`)**: bir container'ı arka planda çalıştırmak, çıktısına bağlanmak yerine terminal komut istemini hemen geri döndürmek.
- **`docker exec`**: zaten çalışan bir container içinde, en çok interaktif bir shell olmak üzere, ek bir komut çalıştırmak.
- **Nazik durdurma (graceful stop)**: `docker stop` 'un varsayılan davranışı -- bir kapatma sinyali göndermek ve zorla sonlandırmadan önce sürece temiz çıkma şansı vermek.