Docker Compose

docker-compose.yml, service'ler, depends_on, Compose'ın otomatik servis-adı-ile-networking'i, ve volumes -- "Docker Networking" ile "Docker Volumes"un elle yaptığı her şeyi tek bir dosyaya ve docker compose up'a indirerek.

Orta 35 dk

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

---

Son iki dersin elle yaptığı her şey -- `docker network create`, `docker volume create`, ve her birinde eşleşen `--network` ve `-v` flag'leriyle yazılan iki ayrı `docker run` komutu ("Docker Networking" ve "Docker Volumes") -- gerçek, çalışan Docker, ama aynı zamanda her seferinde elle hatırlanması, yeniden yazılması, ve tutarlı tutulması gereken bir komut dizisi. Docker Compose, o diziyi tek, bildirimsel bir dosyaya çeviren araçtır. Bu ders, tam olarak aynı iki-container kurulumunu -- bu projenin kendi uygulaması ve veritabanını -- tek bir `docker-compose.yml` olarak yeniden inşa ediyor.

Docker Compose Nedir

Docker Compose, ilişkili bir container kümesini (Compose'un kendi kelime dağarcığında **service**'ler) tarif eden bir YAML dosyası okur ve hepsini birer komutla ayağa kaldırır -- ya da hepsini indirir. Alttaki mekanizmayla ilgili hiçbir şey değişmiyor: Compose hâlâ bir ağ oluşturuyor, hâlâ volume'lar oluşturuyor, hâlâ container'ları `docker run` flag'lerinin eşdeğeriyle çalıştırıyor -- yalnızca her `docker network create` / `docker volume create` / `docker run`'ın her seferinde ayrı ayrı, doğru sırayla yazılmasını gerektirmek yerine, hepsini tek bir dosyadan türetiyor.

Compose Dosyası: `docker-compose.yml`

Bir Compose dosyasının üst seviyesinde bu proje için önemli iki bölüm var: `services`, çalıştırılacak her container için bir giriş, ve `volumes`, o servislerin ihtiyaç duyduğu named volume'lar (bir named volume'un gerçekte ne olduğu için "Docker Volumes"a bkz.).

```
services:
  db:
    # ...
  app:
    # ...

volumes:
  db-data:
```

Burada, üst seviyede bir kez bildirilen `db-data`, "Named Volume'lar"ın zaten işlediği tam olarak aynı türde bir named volume -- Compose, `docker compose up` ilk çalıştığında onu otomatik olarak oluşturur, `docker volume create`'in elle yaptığı gibi.

Bir Servis Tanımlamak: `db`

```
services:
  db:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: secret
    volumes:
      - db-data:/var/lib/postgresql/data
```

Bu, "Docker Volumes"un elle kullandığı `docker run` komutunun doğrudan bir çevirisi: `image`, "Bir Image Çekmek: `docker pull`"dan aynı `postgres:16`; `environment`, aynı `-e POSTGRES_PASSWORD=secret`; ve `volumes`, "PostgreSQL Verisini Nerede Saklar"ın zaten açıkladığı aynı `/var/lib/postgresql/data` yoluna aynı named volume'u mount ediyor -- yalnızca komut satırı flag'leri yerine YAML olarak ifade edilmiş.

Bir Servis Tanımlamak: `app`

```
services:
  app:
    build: .
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/postgres
      SPRING_DATASOURCE_PASSWORD: secret
    ports:
      - "8080:8080"
```

`build: .`, Compose'a önceden inşa edilmiş bir image çekmek yerine bu projenin kendi multi-stage Dockerfile'ını ("Bir Spring Boot Uygulamasını Docker'a Taşımak"tan) inşa etmesini söyler -- elle `docker build` çalıştırmakla aynı etki, yalnızca `docker compose up`'ın bir parçası olarak otomatik olarak tetiklenir. `ports`, "Bir Container Çalıştırmak: `docker run`"un zaten işlediği aynı `-p 8080:8080`.

depends_on — Başlangıç Sırası

`depends_on: [db]`, Compose'a `app`'i başlatmadan önce `db`'yi başlatmasını söyler -- olmadan, ikisi de pratik olarak aynı anda başlardı, ve `app`, henüz initialize'ı bitirmemiş bir `db`'ye bağlanmayı deneyebilirdi.

> `depends_on` kendi başına yalnızca `db` **container'ının başlamasını** bekler, içindeki PostgreSQL'in gerçekten bağlantı kabul etmeye *hazır* olmasını değil -- bunlar iki farklı andır, ve yavaş başlayan bir veritabanı, `depends_on` yerinde olsa bile `app`'in ilk bağlantı denemesini başarısız kılabilir. Bir `healthcheck`, bu boşluğu doğru şekilde kapatır -- bu kategorideki bir sonraki ders "Production İçin Docker (Java Uygulamaları)", bunu tam olarak işliyor.

Compose'ta Volume'lar

`db` servisinin `volumes:` listesinde referans verilen `db-data` volume'u, "Compose Dosyası: `docker-compose.yml`"da gösterildiği gibi, dosyanın üst seviyesinde bir kez bildirilmelidir -- bir servis, Compose'a dosyanın başka hiçbir yerinde söylenmemiş bir volume adı icat edemez. Compose aslında volume'un gerçek adının önüne, oluşturma zamanında projenin kendi adını ekler (`docker volume ls`'te yalnızca `db-data` değil, `learning-platform_db-data` olarak görünür) -- YAML dosyasındaki kısa isim, servislerin referans verdiği şeydir; öne eklenmiş isim ise onu aynı makinedeki tamamen farklı bir Compose projesinden aynı isimli bir volume'la çakışmaktan koruyan şeydir.

Compose'ta Networking (Otomatik)

Compose'un gerçek, anlamlı iş tasarrufu yaptığı yer burası: "Docker Networking", iki container birbirine isimle ulaşabilmeden önce açık bir `docker network create` ve her `docker run`'da `--network` gerektirdi. Compose bu adımı tamamen atlar -- tek bir `docker-compose.yml`'de tanımlanan her servis otomatik olarak aynı, Compose'un oluşturduğu ağa yerleştirilir, ve **her servisin adı, o aynı dosyadaki her diğer servis için hostname'i olur**, hiçbir konfigürasyon gerekmeden. Yukarıdaki `app`'in `SPRING_DATASOURCE_URL`'sinin PostgreSQL'e `db:5432`'de ulaşmasının nedeni tam olarak bu -- `db`, bu aynı dosyadaki diğer servisin adıdır, "İsimle Container'dan Container'a İletişim"deki elle oluşturulmuş ağda `learning-platform-db`'nin yaptığı gibi otomatik olarak çözümlenir.

Çalıştırmak: `docker compose up` / `docker compose down`

```
docker compose up -d
```

`docker compose up`, şu anki dizindeki `docker-compose.yml`'i okur, eksik olan ağı ve volume'ları oluşturur, ihtiyaç duyulan image'ları inşa eder ya da çeker, ve her servisi başlatır -- `-d`, onu detached çalıştırır, `docker run -d` ile aynı anlam. `docker compose down` bunu tersine çevirir -- Compose'un oluşturduğu her container'ı ve ağı durdurup kaldırır -- ama bilinçli olarak varsayılan olarak named volume'lara **dokunmaz**, "Docker Volumes"un işlediği tam olarak aynı nedenle: bir volume'un, sıradan container yaşam döngüsü olaylarından daha uzun yaşaması amaçlanır, ve `docker compose down` kendisini tam olarak bu olarak ele alır, sıradan bir söküm, veri yok eden bir şey değil. `docker compose down -v`, gerçekten amaçlandığında volume'ları da kaldırmak için açık, bilinçli bir tercihtir.

Hepsini Bir Araya Getirmek

Bu proje için tam `docker-compose.yml` -- "Docker Networking" ve "Docker Volumes"un elle inşa ettiği tam olarak aynı uygulama-artı-veritabanı kurulumu, tek bir dosya olarak ifade edilmiş:

```
services:
  db:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: secret
    volumes:
      - db-data:/var/lib/postgresql/data

  app:
    build: .
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/postgres
      SPRING_DATASOURCE_PASSWORD: secret
    ports:
      - "8080:8080"

volumes:
  db-data:
```

Ve baştan sona çalıştırmak:

```
#!/bin/sh
# Everything "Docker Networking" and "Docker Volumes" typed by hand --
# docker network create, docker volume create, and two docker run commands
# -- replaced by a single docker-compose.yml and one command.

docker compose up -d
# Output (trimmed):
# [+] Running 3/3
# => Network learning-platform_default   Created
# => Volume "learning-platform_db-data"  Created
# => Container learning-platform-db-1    Started
# => Container learning-platform-app-1   Started

docker compose ps
# Output:
# NAME                        IMAGE          STATUS         PORTS
# learning-platform-app-1     learning-platform-app   Up 5 seconds   0.0.0.0:8080->8080/tcp
# learning-platform-db-1      postgres:16             Up 5 seconds   5432/tcp

docker compose logs app
# Output (trimmed):
# app-1  | Started LearningPlatformApplication in 3.2 seconds

curl http://localhost:8080/en
# Output (trimmed):
# <!doctype html> ... the homepage renders, meaning app reached db by service name ...

# Stop and remove everything Compose created -- containers and the default
# network. The named volume, "learning-platform_db-data", is deliberately
# left behind (see "Docker Volumes") -- it needs an explicit -v to remove.
docker compose down
# Output (trimmed):
# => Container learning-platform-app-1   Removed
# => Container learning-platform-db-1    Removed
# => Network learning-platform_default   Removed

# The data survives -- confirm the volume is still there.
docker volume ls
# Output:
# DRIVER    VOLUME NAME
# local     learning-platform_db-data
```

Buradaki her parça, bu kursun zaten tek tek işlediği bir şey -- bir servisin `image` / `build`'i, `environment`'ı, `volumes`'u, `ports`'u, başlangıç sırası için `depends_on`, ve `app`'e `db`'ye isimle ulaşmanın bir yolunu veren otomatik dosya-başına ağ. Compose yeni bir mekanizma tanıtmıyor; yalnızca her seferinde doğru yeniden yazılması gereken bir komut dizisini, her seferinde aynı şekilde okunan tek bir dosyayla değiştiriyor.

Yaygın Hatalar

- Bir servisin `volumes:` listesinde, dosyanın üst-seviye `volumes:` bölümünde hiç bildirilmemiş bir volume adına referans vermek -- Compose'un o adın var olduğunu bilmesinin başka bir yolu yok.

- `depends_on`'un bağımlılığın yalnızca *başladığını* değil, tamamen *hazır* olduğunu garanti ettiğini varsaymak -- "`depends_on` — Başlangıç Sırası" altındaki uyarıya bkz.
- Alışkanlıkla `docker compose down -v` çalıştırıp bir veritabanının verisinin gittiğine şaşırmak -- düz `docker compose down` (`-v` olmadan) sıradan sökümü zaten güvenle ele alır; `-v` bilinçli, ayrı bir tercihtir.
- Bir container'ın gerçek adını (önceki derslerdeki `learning-platform-db` gibi) bir Compose servisinin konfigürasyonuna sabit kodlamak, aynı `docker-compose.yml`'deki *diğer servisin adını* kullanmak yerine -- Compose'un kendi otomatik networking'i servis adlarını kullanır, bir container'a başka bir yerde elle verilmiş herhangi bir ismi değil.

Best Practices

- Gerçek herhangi bir çok-container kurulumu için -- bu proje gibi bir projenin yerel geliştirmesi dahil -- manuel bir `docker network create` / `docker volume create` / `docker run` dizisi yerine Compose'a başvur.
- Bu dersinki gibi (zaten hassas olmayan, illüstratif) değerler için bir servisin ortam değişkenlerini Compose dosyasının kendisinde tut -- gerçek bir deployment tipik olarak `POSTGRES_PASSWORD` gibi sırları ayrı, commit edilmemiş bir `.env` dosyasına ya da uygun bir secret mekanizmasına taşır.
- Başlangıç sırası için `depends_on`'u kullan, ama bir bağımlılığın gerçek hazır olma durumu -- yalnızca sürecinin başlaması değil -- gerçekten önemli olduğunda onu gerçek bir `healthcheck`'le eşleştir (sonraki, "Production İçin Docker (Java Uygulamaları)"da işleniyor).
- Sıradan söküm için varsayılan olarak düz `docker compose down`'a başvur, ve `-v`'yi her yazıldığında bilinçli, ayrı bir karar olarak ele al.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Docker Compose, bir servis kümesini tarif eden bir `docker-compose.yml` okur ve hepsini birer komutla ayağa kaldırır ya da indirir -- manuel bir `docker network create` / `docker volume create` / `docker run` dizisini tek, bildirimsel bir dosyayla değiştirir.
- Bir servisin `image` / `build`'i, `environment`'ı, `volumes`'u, ve `ports`'u, bu kurs boyunca zaten işlenen `docker run` flag'lerinin doğrudan YAML eşdeğerleridir.
- Named volume'lar dosyanın üst seviyesinde bir kez bildirilir ve onlara ihtiyaç duyan herhangi bir servisten isimle referans verilir.
- Tek bir Compose dosyasındaki her servis otomatik olarak aynı ağa yerleştirilir, ve her servisin adı diğer her servis için hostname'i olur -- elle `docker network create` ya da `--network` gerekmez.
- `docker compose down`, container'ları ve ağı kaldırır ama bilinçli olarak varsayılan olarak named volume'ları dokunulmadan bırakır; `docker compose down -v`, onları da kaldırmak için açık tercihtir.

Cheat Sheet

```
services:
  db:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: secret
    volumes:
      - db-data:/var/lib/postgresql/data
  app:
    build: .
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/postgres
    ports:
      - "8080:8080"

volumes:
  db-data:
```

```
docker compose up -d      # her şeyi oluştur/başlat, detached
docker compose ps         # bu projenin çalışan servislerini listele
docker compose logs <svc> # bir servisin çıktısını oku
docker compose down       # container'ları + ağı durdur ve kaldır (volume'lar kalır)
docker compose down -v    # named volume'ları da kaldır
```

Terimler Sözlüğü

- **Service**: bir `docker-compose.yml` içindeki, YAML olarak ifade edilmiş bir `docker run` komutuna eşdeğer, tek bir container tanımı.
- **`depends_on`**: bir servisin container'ının başlamasını başka birininkinden sonraya sıralayan, bağımlılığın tamamen hazır olmasını beklemeyen bir Compose ayarı.
- **Compose network**: Compose'un her dosya için otomatik olarak oluşturduğu, her servisin diğer her servise kendi servis adıyla ulaşabildiği ağ.
- **`docker compose down -v`**: named volume'ları da kaldıran, açık, bilinçli tercih edilen söküm biçimi -- düz biçim bunu yapmaz.