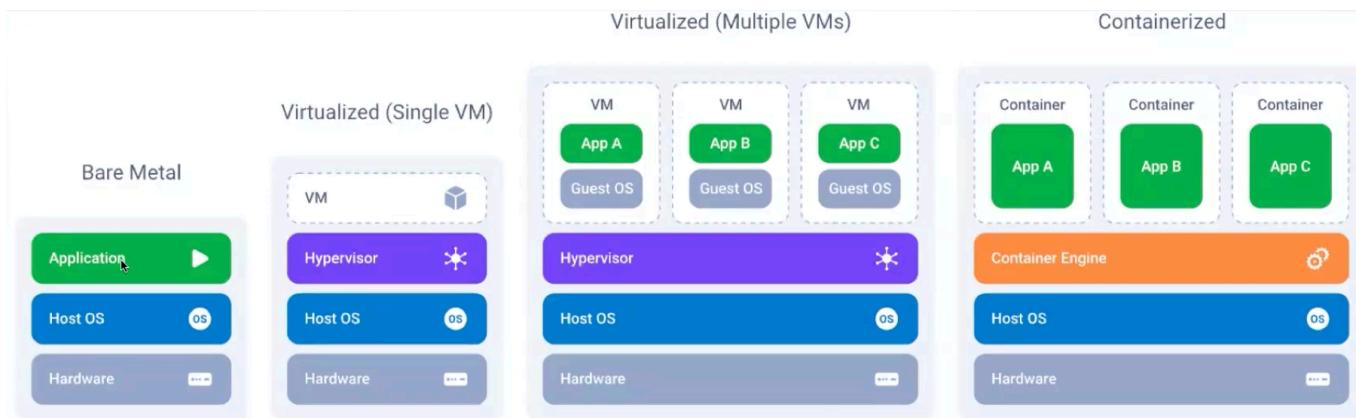


# Introduction to Docker

## Deploy Ama Nasıl?



## Docker'a Giriş

### Tanım :

Uygulamalar ve bağımlılıklarını bir araya getirerek hafif ve taşınabilir konteynerler içinde paketleyen açık kaynaklı bir platformdur.

### Amaç :

Geliştirme, test ve üretim ortamlarında uygulamaların tutarlı bir şekilde çalışmasını sağlamak.

### Neden Docker?

#### Kaynak Verimliliği:

- Sanal makinelerden daha hafif; daha az kaynak kullanır.

#### Hızlı Dağıtım:

- Uygulamaların hızlı bir şekilde dağıtılmasını ve ölçeklendirilmesini sağlar.

#### Tutarlılık:

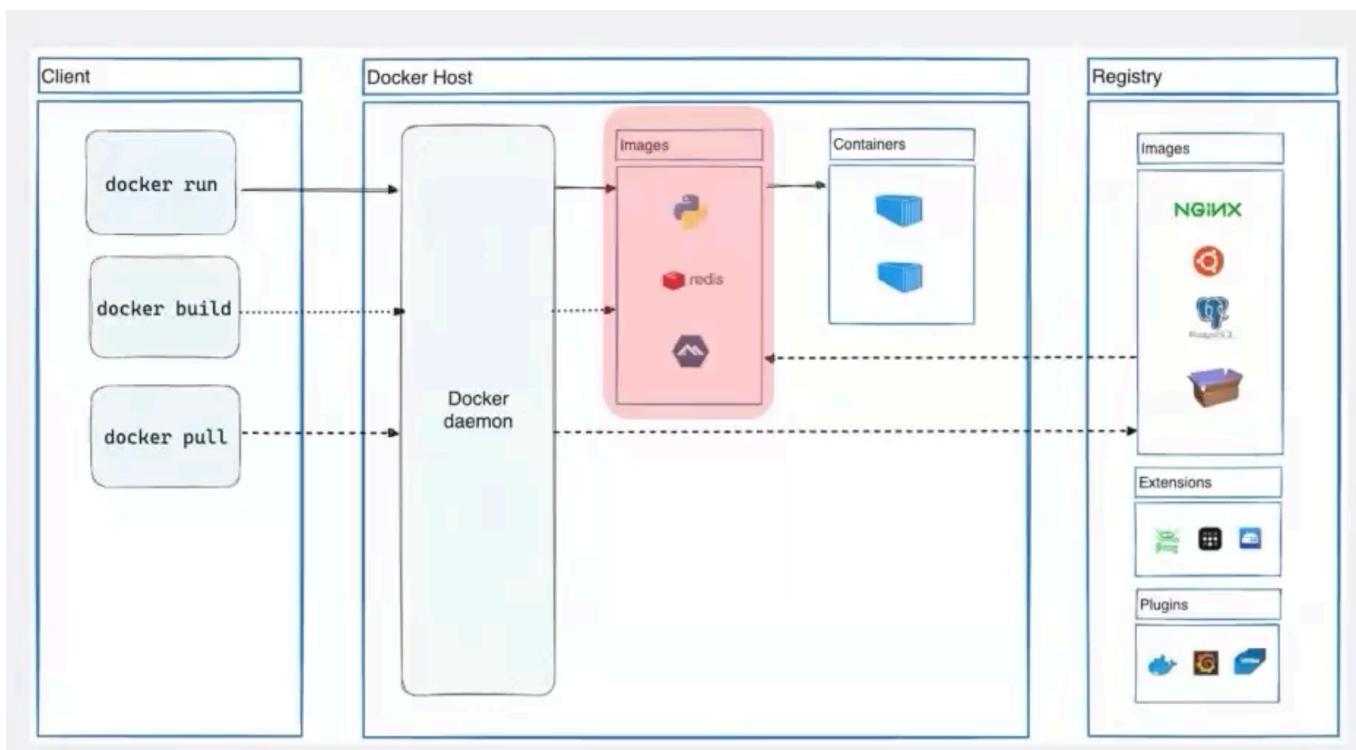
- "Benim makinemde çalışıyor" sorununu ortadan kaldırır.

#### Ekosistem ve Araçlar:

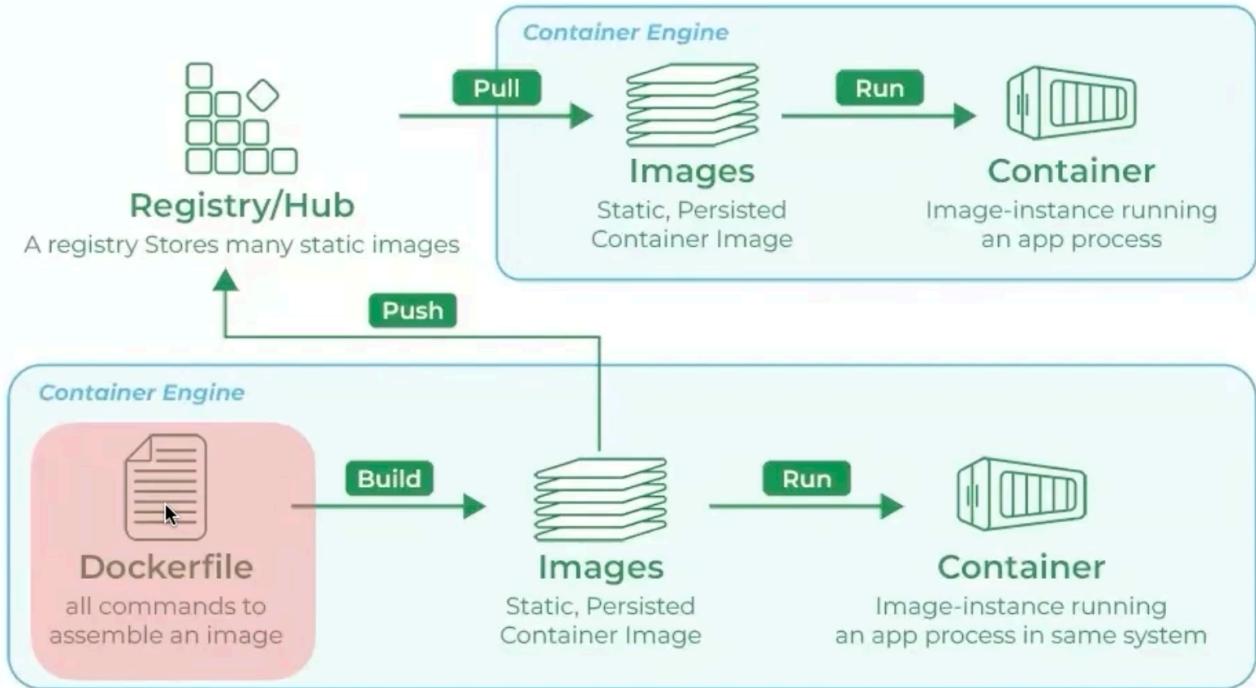
- Docker Hub gibi geniş bir görüntü deposuna sahiptir.
- Docker Compose, Swarm gibi ek araçlar sunar.

## Temel Bileşenler

- **Görüntüler (Images)**: Uygulamaların ve bağımlılıklarının salt okunur şablonlarıdır. Bir konteyner oluşturmak için alınır.
- **Konteynerler (Containers)**: Görüntülerin çalıştırılabilir örnekleridir. Uygulamaların izole edilmiş ortamda çalışmasını sağlar.
- **Registry**: Görüntülerin depolandığı ve paylaşıldığı merkezi depolarıdır. Docker Hub en yaygın kullanılan kayıt deposudur.

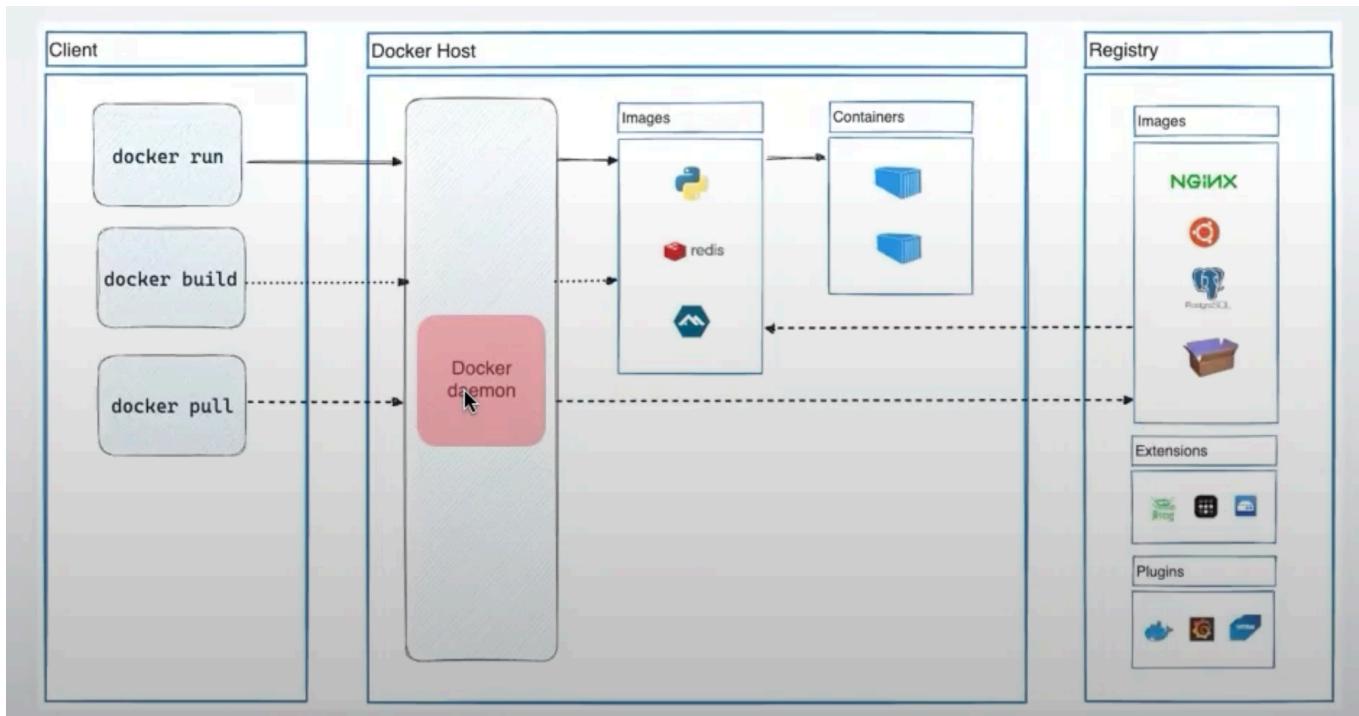


- **Dockerfile**: Görüntü oluşturmak için talimatları içeren metin dosyasıdır. Katmanlı yapı sayesinde verimli görüntü oluşturmayı sağlar.

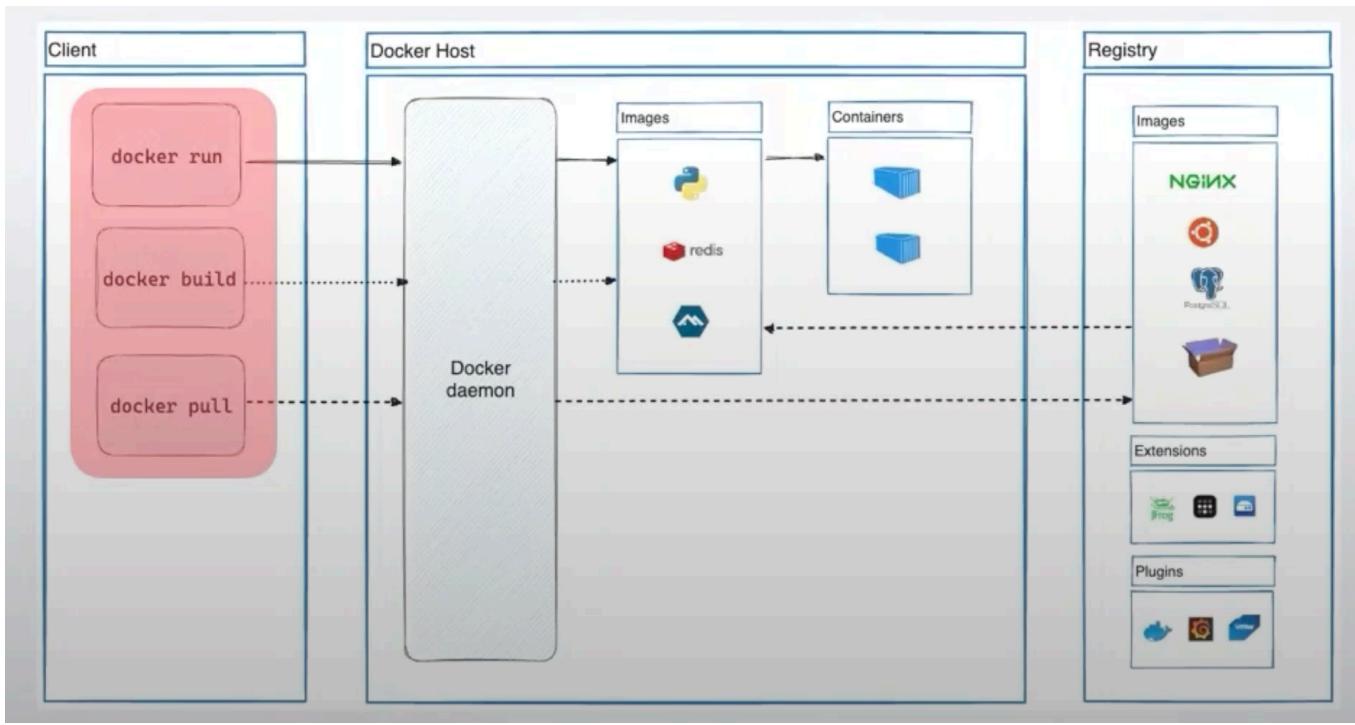


## Docker Mimarisi

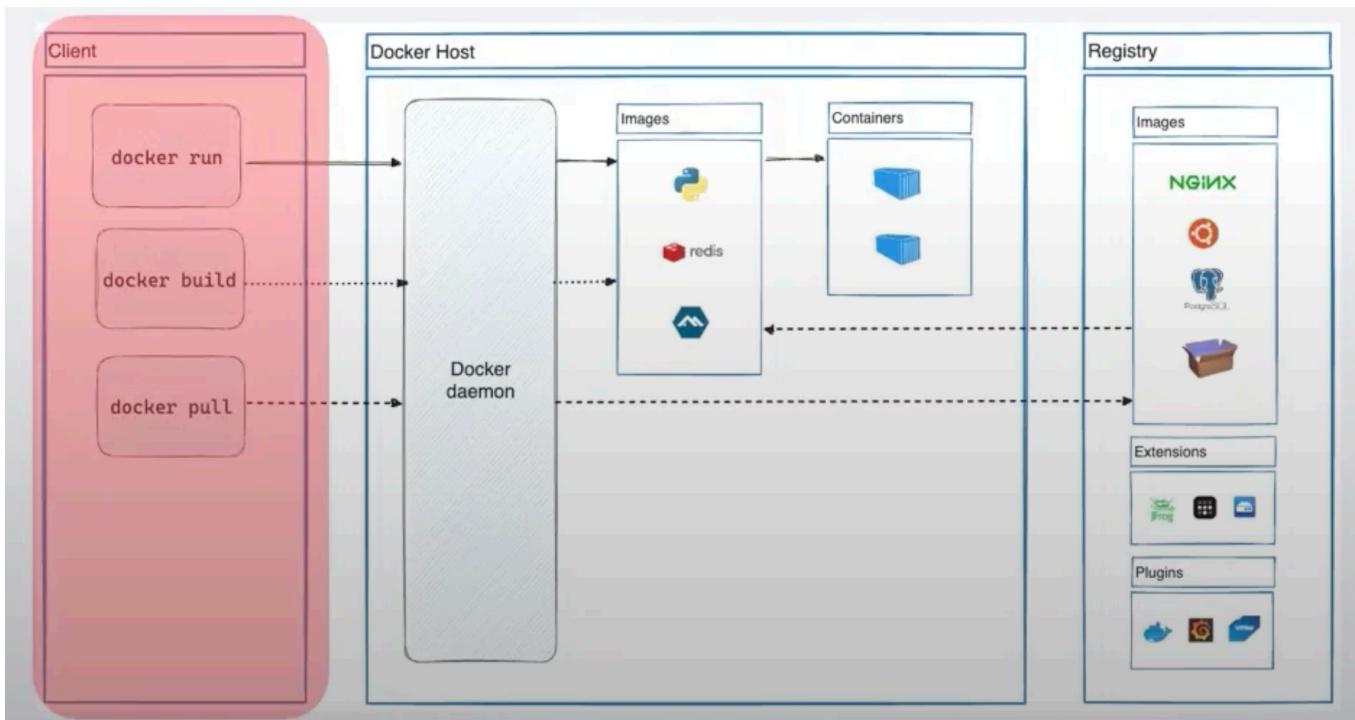
- **Docker Daemon (dockerd):** Arka planda çalışan ve Docker nesnelerini (görüntüler, konteynerler, ağlar hacimler) yöneten süreç



- **Docker CLI (Komut Satırı Arayüzü):** Kullanıcıların Docker Daemon ile etkileşim kurmasını sağlar. docker komutu ile erişilir.



- **Docker Client:** Docker Daemon ile iletişim kuran ve kullanıcı isteklerini işleyen bileşen.



## Komutlar

- \*\*docker run\*\*
  - Bir görüntüden yeni bir konteyner oluşturur ve çalıştırır.
  - Örnek: docker run hello-world
  - Örnek: docker run -e POSTGRES\_PASSWORD=admin postgres:latest

- **docker ps**
  - Çalışan konteynerlerin listesini gösterir.
  - "-a" seçeneği ile tüm konteynerleri (çalışan ve durdurulmuş) listeler.
- **docker stop**
  - Belirtilen konteyneri durdurur.
  - Örnek: `docker stop container_id`
- **docker pull**
  - Bir görüntüyü Docker Hub'dan veya başka bir kayıt deposundan çeker.
  - Örnek: `docker pull ubuntu`
- **docker build**
  - Bir Dockerfile'dan görüntü oluşturur.
  - Örnek: `docker build -t my-image .`
- **docker rm**
  - Bir veya daha fazla konteyneri siler.
  - Örnek: `docker rm container_id`
- **docker rmi**
  - Bir veya daha fazla görüntüyü siler.
  - Örnek: `docker rmi image_id`
- **docker images**
  - Görüntüleri listeler.
  - Örnek: `docker images`
- **docker inspect**
  - Docker nesneleri hakkında düşük seviyeli bilgiler döndürür.
  - Örnek: `docker inspect hello-world`

## Dockerfile

- Temel Komutlar
  - **FROM:** Temel görüntüyü belirtir.
  - **RUN:** Komutları çalıştırır ve görüntüye dahil eder.
  - **COPY** veya **ADD:** Dosyaları görüntüye kopyalar.
  - **CMD** veya **ENTRYPOINT:** Konteyner başlatıldığında çalışacak komutu belirtir.
  - **EXPOSE:** Konteynerin dinleyeceği portu belirtir.
  - **ENV:** Ortam değişkenlerini ayarlar.

## Örnek Dockerfile

```
# Temel görüntü olarak OpenJDK 17 kullan
FROM openjdk:17-jdk-slim

# Uygulama kaynak kodunu görüntüye kopyala
COPY . /app

# Çalışma dizinini ayarla
WORKDIR /app

# Maven'i kur ve bağımlılıkları yükle
RUN apt-get update && \
    apt-get install -y maven && \
    mvn clean package

# Uygulamanın çalışacağı portu belirt
EXPOSE 8080

# Uygulamayı başlat
ENTRYPOINT ["java", "-jar", "target/app.jar"]
```