Docker CMD ve ENTRYPOINT Komutları: Anlaşılması ve Farkları

1. Docker CMD (Command):

- `CMD`, bir konteyner çalıştırıldığında varsayılan olarak çalıştırılacak komutu belirlemek için kullanılır. Ancak, bu komut konteyner çalıştırılırken başka bir komut verilirse, o komut `CMD` komutunun yerini alır. Yani `CMD`, "varsayılan komut" işlevi görür.
- `CMD` komutunda genellikle komut satırı olarak çalışacak uygulama belirtilir ve argümanlar verilebilir.

Örnek:

Dockerfile

```
FROM nginx
CMD ["nginx", "-g", "daemon off;"]
```

Bu komut, `nginx` sunucusunu arka planda çalıştırır, fakat eğer kullanıcı konteyneri çalıştırırken farklı bir komut verirse bu komutun yerini alır.

2. Docker ENTRYPOINT:

- `ENTRYPOINT`, bir konteyner başlatıldığında her zaman çalıştırılacak komutu belirtir. `ENTRYPOINT` ile belirtilen komut, konteyner çalıştırılırken değiştirilemez (ancak `--entrypoint` bayrağı kullanılarak üzerine yazılabilir).
- `ENTRYPOINT` komutunun amacı, bir konteynerin çalıştırdığı ana işlemi tanımlamaktır. Bu nedenle genellikle komutlar tek bir görevi yerine getirmek için ayarlanır.

Örnek:

Dockerfile

```
FROM nginx
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

Bu durumda, konteyner her çalıştırıldığında `nginx` sunucusu başlatılır ve kullanıcı başka bir komut verse bile `nginx` çalışır.

3. CMD ve ENTRYPOINT Arasındaki Farklar:

- `CMD` genellikle varsayılan komut olarak kullanılırken, `ENTRYPOINT` her zaman çalıştırılan bir komuttur.
- `CMD`, konteyner çalıştırılırken farklı bir komut verilirse devre dışı kalabilir. `ENTRYPOINT` ise genellikle devre dışı bırakılamaz.

- `ENTRYPOINT`, komut çalıştırılırken sabit bir komut sağlamayı amaçlar. `CMD` ise opsiyonel bir komut sağlar.

Örnekler:

Örnek 1: Sadece CMD ile

Dockerfile:

```
FROM ubuntu:latest
CMD ["echo", "Hello World"]
```

Adımlar:

1. Dockerfile'ı oluşturun ve `docker build -t cmd-example .` komutuyla imajı oluşturun.
2. `docker run cmd-example` komutunu çalıştırın.
    - Çıktı: `Hello World`
3. Şimdi konteyneri başka bir komut ile çalıştırın: `docker run cmd-example echo "Hello from CMD"`
    - Çıktı: `Hello from CMD`

Bu örnekte, CMD komutu `docker run` sırasında verilen komutla değiştirilebiliyor.

Örnek 2: Sadece ENTRYPOINT ile

Dockerfile:

```
FROM ubuntu:latest
ENTRYPOINT ["echo", "Hello from ENTRYPOINT"]
```

Adımlar:

1. Dockerfile'ı oluşturun ve `docker build -t entrypoint-example .` komutuyla imajı oluşturun.
2. `docker run entrypoint-example` komutunu çalıştırın.
    - Çıktı: `Hello from ENTRYPOINT`
3. Şimdi konteyneri başka bir komut ile çalıştırın: `docker run entrypoint-example echo "This won't override"`
    - Çıktı: `Hello from ENTRYPOINT echo This won't override`

Bu örnekte, `ENTRYPOINT` komutu her zaman çalışır ve kullanıcı komutları ona eklenir.

Örnek 3: CMD ve ENTRYPOINT Birlikte Kullanımı

Dockerfile:

```
FROM ubuntu:latest
ENTRYPOINT ["echo"]
CMD ["Hello from CMD"]
```

Adımlar:

1. Dockerfile'ı oluşturun ve `docker build -t combined-example .` komutuyla imajı oluşturun.
2. `docker run combined-example` komutunu çalıştırın.
    - Çıktı: `Hello from CMD`
3. Şimdi konteyneri başka bir komut ile çalıştırın: `docker run combined-example World`
    - Çıktı: `World`

Bu örnekte, `ENTRYPOINT` sabit komut olarak çalışıyor, `CMD` ise varsayılan argüman sağlıyor. Eğer başka bir argüman verilirse, `CMD`'nin yerini alır.

Özet:

- `CMD`: Varsayılan komut, kullanıcı tarafından değiştirilebilir.
- `ENTRYPOINT`: Sabit komut, konteyner her çalıştığında devreye girer.
- İkisi birlikte kullanıldığında, `ENTRYPOINT` ana komut olarak çalışır, `CMD` ise varsayılan argüman sağlar.

Advanced örnekler

1-ENTRYPOINT ve CMD ile Varsayılan Argümanlar

Bu örnekte, ENTRYPOINT ile bir script çalıştıracağız ve CMD ile varsayılan argümanları sağlayacağız. Eğer kullanıcı farklı bir argüman verirse CMD'nin yerine geçecek.

Dockerfile:

```
FROM python:3.9-slim

# Bir Python scripti oluşturun
RUN echo 'import sys; print(f"Hello {sys.argv[1]}!")' > /hello.py

# ENTRYPOINT ile scripti çalıştır ve CMD ile varsayılan argümanı sağla
ENTRYPOINT ["python", "/hello.py"]
CMD ["World"]
```

Adımlar:

1. Dockerfile'ı oluşturun ve

   `docker build -t python-example .` komutuyla imajı oluşturun.

2. `docker run python-example` komutunu çalıştırın.
   - Çıktı: `Hello World!`
3. Şimdi konteyneri başka bir argüman ile çalıştırın:

```
docker run python-example Docker
```

   - Çıktı: `Hello Docker!`

Bu örnekte, ENTRYPOINT Python scriptini çalıştırırken CMD varsayılan argümanı sağlıyor. Fakat, kullanıcı farklı bir argüman verdiğinde CMD argümanı yerine geçiyor

2-ENTRYPOINT ile Script ve CMD ile Ek Argümanlar

Bu örnekte, ENTRYPOINT sabit bir script çalıştırırken CMD ek argümanlar sağlayacak. ENTRYPOINT ve CMD birlikte nasıl çalışır bunu göreceksiniz.

```
FROM alpine:latest
```

```
# Bash yükleyin
RUN apk add --no-cache bash

# Bir bash scripti oluşturun
RUN echo -e '#!/bin/bash\n echo "Running main script";\n echo
"Arguments: $@"' > /entrypoint.sh && chmod +x /entrypoint.sh

# ENTRYPOINT olarak bash'i ve scripti çalıştırın
ENTRYPOINT ["/bin/bash", "/entrypoint.sh"]

# CMD ile varsayılan argümanları belirtin
CMD ["DefaultArg1", "DefaultArg2"]
```

```
docker build -t argument .

docker run -it --entrypoint /bin/sh argument
docker run argument 1arg 2arg
```

Çıktı:

```
Running main script
Arguments: 1Arg1 2Arg2
```

3-Çok Aşamalı Dockerfile (Multi-stage) ve ENTRYPOINT

Bu örnekte, çok aşamalı bir Dockerfile kullanarak ENTRYPOINT ve CMD'yi nasıl entegre edeceğimizi göstereceğiz. İmajın boyutunu optimize ederken ENTRYPOINT ile komutu çalıştıracağız.

Dockerfile:

```
# İlk aşama: Derleme aşaması
FROM golang:1.20-alpine AS builder

# Uygulamayı derle
WORKDIR /app
COPY main.go .
RUN go build -o myapp main.go

# İkinci aşama: Minimal çalışma aşaması
FROM alpine:latest

# Uygulamayı kopyala
```

```
COPY --from=builder /app/myapp /usr/local/bin/myapp

# ENTRYPOINT ile uygulamayı çalıştır, CMD ile varsayılan argümanı sağla
ENTRYPOINT ["myapp"]
CMD ["DefaultArgument"]
```

`main.go` dosyası (aynı dizinde olmalı):

```
package main

import (
    "fmt"
    "os"
)

func main() {
    args := os.Args[1:]
    if len(args) > 0 {
        fmt.Println("Running with argument:", args[0])
    } else {
        fmt.Println("Running with no arguments")
    }
}
```

Adımlar:

1. Dockerfile ve `main.go` dosyasını aynı dizine kaydedin.
2. Dockerfile'ı oluşturun ve

`docker build -t go-multi-stage .` komutuyla imajı oluşturun.

`docker run go-multi-stage` komutunu çalıştırın.

Çıktı: `Running with argument: DefaultArgument`

3. Şimdi konteyneri farklı bir argümanla çalıştırın:

`docker run go-multi-stage CustomArgument`

Çıktı: `Running with argument: CustomArgument`

Bu örnekte, ENTRYPOINT sabit bir komut çalıştırıyor ve CMD varsayılan bir argüman sağlıyor. Çok aşamalı yapı sayesinde imaj boyutunu optimize ederken ENTRYPOINT ve CMD kullanarak esneklik sağlıyoruz.