

“**docker search <anahtar kelime>**” komutu ile bir kelimeyi indirebileceğimiz imaj dosyaları içinden aradığımız kelimeleri içeren dosyaları gösterir.

```
PS C:\Users\Caner> docker search alpine
NAME                                STARS      OFFICIAL    DESCRIPTION                                     10941      [OK]
alpine                              11
alpinelinux/docker-cli             3
alpinelinux/alpine-gitlab-ci       7
alpinelinux/gitlab-runner-helper   0
alpinelinux/rsyncd                 2
alpinelinux/unbound                13
alpinelinux/alpine-drone-ci        0
alpinelinux/docker-alpine
```

“**docker pull <imaj adı>**” komutu ile istediğimiz imaj dosyasını bilgisayarımıza indirebiliriz.

```
PS C:\Users\Caner> docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
c6a83fedfae6: Pull complete
Digest: sha256:0a4eaa0eecf5f8c050e5bba433f58c052be7587ee8af3e8b3910ef9ab5fbe9f5
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

“**docker images**” komutu ile imaj dosyalarını listeleyebiliriz

```
PS C:\Users\Caner> docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
alpine              latest     324bc02ae123  2 days ago    7.8MB
nginx               latest     a72860cb95fd  4 weeks ago   188MB
shmilylty/oneforall latest     7c29bee9a9a5  3 months ago  455MB
```

“**docker image inspect 324bc02ae123**” komutu ile o imaj hakkında bilgileri alabiliriz. Inspect komutu volume container gibi docker içindeki benzer şeyler içinde bilgi alabiliriz.

```
PS C:\Users\Caner> docker inspect 324bc02ae123
[
  {
    "Id": "sha256:324bc02ae1231fd9255658c128086395d3fa0aedd5a41ab6b034fd649d1a9260",
    "RepoTags": [
      "alpine:latest"
    ],
    "RepoDigests": [
      "alpine@sha256:0a4eaa0eecf5f8c050e5bba433f58c052be7587ee8af3e8b3910ef9ab5f5f5"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2024-07-22T22:26:43.778747613Z",
    "ContainerConfig": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": null,
      "Cmd": null,
      "Image": "",
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": null
    }
  }
]
```

“**docker run -it alpine**” komutu ile run sayesinde alpine imajını çalıştırırken “-it” ile konteynerden interaktif bir shell almış oluruz.

```
PS C:\Users\Caner> docker run -it alpine
/ # |
```

“**docker run -it alpine ls**” komutu ile konsol açılır ve “ls” komutu çalışır. Bu komut yerine istediğimiz komutu koyabiliriz. Koyduğumuz bu komut veya komutlar çalıştıktan sonra shell kapanır.

```
PS C:\Users\Caner> docker run -it alpine ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run   /sbin  srv    sys    tmp
PS C:\Users\Caner> |
```

“**docker run -p 80:80 --rm --name yavuzlar nginx**” komutu sayesinde bilgisayarımızın 80 portunu(solda yazan), konteynerin portuna(sağda yazan) bağlar. “--name” parametresine “yavuzlar” değeri verildi ve nginx konteyneri çalıştırıldı. Bu durumda konteyner ismi yavuzlar olarak düzenlenmiş oldu. “--rm” parametresi ile konteyner durduğunda siliniyor.

```
PS C:\Users\Caner> docker run -p 80:80 --rm --hostname yavuzlar nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/07/24 23:40:49 [notice] 1#1: using the "epoll" event method
```

“**docker stop yavuzlar**” komutu ile yavuzlar adına oluşturduğumuz konteynerimizi durdurabiliriz. Konteyneri çalıştırırken kullandığımız “--rm” parametresi sebebiyle konteynerimiz durduğunda siliniyor. “**docker ps -a**” komutu ile kontrol ettiğimizde bu konteynerin silindiğini görüyoruz.

```
charming_elbakyan
PS C:\Users\Caner> docker stop yavuzlar
yavuzlar
PS C:\Users\Caner> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
a672b550553d   alpine   "/bin/sh"               17 minutes ago Exited (137) 10 minutes ago           hopeful_albattani
f5f600d29532   alpine   "/bin/sh"               19 minutes ago Exited (137) 10 minutes ago           reverent_swirles
PS C:\Users\Caner>
```

“**docker ps**” komutu ile aktif olarak çalışan docker konteynerleri ve bilgileri listelenir. Sonuna “-a” parametresini eklersek çalışmayan konteynerler de bu listeye dahil olur.

```
PS C:\Users\Caner> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
7d6a22bc071e   nginx     "/docker-entrypoint..." 4 seconds ago Up 4 seconds      0.0.0.0:80->80/tcp yavuzlar
```

“**docker run -e mysql_ROOT_PASSWORD=gizli -e mysql_ROOT_USER=caner [mysql server konteyner]**” şeklinde bir komut ile sistemin kullanması gereken enviromentleri verebiliriz.

“**docker ps a**” komutu ile aldığımız konteynerlerimiz arasından “charming_elbakyan” ismindeki konteyneri silmek için “**docker container rm charming_elbakyan**” komutunu giriyoruz.

```
PS C:\Users\Caner> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
7d6a22bc071e   nginx     "/docker-entrypoint..." 5 minutes ago Up 5 minutes      0.0.0.0:80->80/tcp yavuzlar
64e48c635d87   alpine   "ls"                    12 minutes ago Exited (0) 12 minutes ago           charming_elbakyan
a672b550553d   alpine   "/bin/sh"               13 minutes ago Exited (137) 6 minutes ago           hopeful_albattani
f5f600d29532   alpine   "/bin/sh"               15 minutes ago Exited (137) 6 minutes ago           reverent_swirles
```

```
PS C:\Users\Caner> docker container rm charming_elbakyan
charming_elbakyan
```

Dockerfile

```
FROM php:7.4-apache
```

Burada 7.4-apache image kullanıyoruz. php dosyalarını apache server üzerinde kullanabilir şekilde bir kurulum gerçekleşmiş oluyor.

```
WORKDIR /var/www/html
```

“WORKDIR” ile çalışma dizinimiz belirliyoruz.

```
COPY ./app .
```

“./app” dizini içerisinde bulunan dosyaları içinde bulunduğumuz dizine yani “WORKDIR” olarak tanımladığımız adrese kopyalıyoruz.

```
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
```

apache2.conf dosyamıza eklememiz gereken “ServerName” parametresini ve “localhost” değerini dosyamızın son satırına ekliyoruz.

RUN sayesinde komutları çalıştırabiliyoruz.

```
RUN apt-get update
```

docker konteynerimizin update yapmasını sağlar.

```
EXPOSE 80
```

“EXPOSE” komutu ile konteynerimiz 80 portunu dinliyor.

docker-compose.yml

Docker compose çok sayıda konteyneri bir dosya içerisinde konfigürasyonlar yaparak tek bir dosya ile çalıştırmaya yarar.

```
version: '3'
```

Docker versiyonumuzu belirliyoruz.

```
services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      - db
    ports:
      - "80:80"
    networks:
      - net

  db:
    image: mysql:latest
    environment:
      - MYSQL_DATABASE=yavuzlar
      - MYSQL_ROOT_PASSWORD=1
    volumes:
      - db_data:/var/lib/mysql
      - ./yavuzlar_messages.sql:/docker-entrypoint-initdb.d/yavuzlar_messages.sql
    ports:
      - "8080:3306"
    networks:
      - net
```

Burada bir ağaç yapısı var. Services altında servisler(konteynerler) bulunuyor. her servis kendine ait konfigürasyonlara sahip oluyor.

Bu örnekte 2 adet servisimiz var, "app" ve "db".

"app" servisi için konfigürasyonlar şu şekilde:

- dockerfile parametresi ile image dosyasının oluşturulmasını sağlayan "Dockerfile" ismindeki dosyayı işaret eder.
- "depends_on" ile "db" isimli servise bağımlılığı olduğunu tanımlamıştır.

- “ports” ile konteynerimizin ve ana bilgisayarımızın portlarını bağlarız. Bu şekilde dışarıdan gelen bağlantıları bilgisayarımızın portunu kullanarak dışarıdan bağlanabiliriz.
- “networks” ile konteynerin bağlı olacağı ağları belirleriz.

“db” servisi için konfigürasyonlar şı şekilde:

- “image” parametresi ile “mysql” docker imajının son versiyonunu kullanacağını belirtmiş.
- “enviroment” parametresi ile mysql konteynerimiz çalışırken ihtiyacı olacağı global değişkenleri tanımlamış.
- “volumes” parametresi ile db konteynerimizin kullanacağı ana bilgisayarımız ile bağlantılı olacak şekilde bir depolama konfigürasyonu yapılıyor. Yapılan bu konfigürasyon ile bu ortak konumda olan dosyalar hem konteyner içinde hemde konteynerin çalıştığı bilgisayar üzerinde ayrı ayrı depolanır. Bu sayede konteyner herhangi bir sebep ile kullanılamaz hale gelse bile bu veriler ana bilgisayarda yedeklenmiş olduğu konumdan alınarak kullanılabilir.
- “ports” parametresi ile bilgisayarımızın 8080 portunu, docker konteynerimizin 3306 portuna bağlıyoruz.
- “networks” parametresi ile “net” isimli olan ağı kullanacağımızı belirtiyoruz.

```
networks:  
  net:  
    driver: bridge
```

“networks” parametresi services adlı bölümün dışında kalıyor, yani ona bağımlı değil. “net” adındaki ağımızı bilgisayarın bu ağı dışarıdan erişilebilir bir şekilde tanımasını sağlayan “bridge” adlı değer verilmiş.

```
volumes:  
  db_data:
```

“volumes” altındaki “db_data:” ifadesi aşağıda görüldüğü şekilde karşısında bulunan değeri almak için kullanılıyor. Bu kısım tanıdık gelebilir çünkü “db” konteynerimiz için olan kısımda görmüştük.

```
    volumes:  
      - db_data:/var/lib/mysql
```