**Docker**

docker login -əmri Docker Hub və ya digər Docker registriesinə daxil olmaq üçün istifadə olunur. Bu əmr istifadəçinin Docker hesabına qoşulmasını təmin edir. Bu əmrdən sonra istifadəçi konteynerləri yükləyə, itələyə və ya digər əməliyyatları yerinə yetirə bilər.

docker logout əmri isə istifadəçini bu hesabdan çıxış etdirir, yəni artıq Docker Hub və ya digər registry ilə əlaqə kəsilir.

docker search

**Image**

docker search: Login oldugun hublarda image axtarmaq ucundur.

**docker image ls**: Sistemdəki mövcud Docker imiclərini siyahıya salır.

docker image ls

**docker images**: Sistemdəki mövcud Docker imiclərini siyahıya salır.

**docker image pull <image\_name>**: Docker Hub və ya digər bir registry-dən müəyyən bir imici yükləyir.

docker image pull ubuntu

**docker image rm <image\_name>**: Sistemdən müəyyən bir Docker imicini silir.

docker image rm ubuntu

**docker image inspect <image\_name>**: Bir Docker imici haqqında detallı məlumatları göstərir.

docker image inspect ubuntu

**docker image tag <old\_image\_name> <target\_image>**: Bir imici yeni bir adda etiketləmək üçün istifadə olunur.

docker image tag ubuntu myubuntu:latest

**docker image build**: Bir Dockerfile-dan yeni bir Docker imici yaradır.

1.docker image build -t myimage:latest .

2.docker build -t myimage:latest .

**Container**

**docker container ls**: Hazırda işləyən konteynerləri siyahıya salır.

docker container ls

**-a opsiyası**: Bütün konteynerləri (işləyən və dayandırılmış) siyahıya salır.

docker container ls -a

**ps**: Hazırda işləyən konteynerləri siyahıya salır.

docker ps

**-a opsiyası**: Bütün konteynerləri (işləyən və dayandırılmış) siyahıya salır.

docker ps -a

**docker container run**: Yeni bir konteyner işə salır.

1.docker container run ubuntu

2.docker run ubuntu

**-d opsiyası**: Konteyneri arxa planda işə salır.

docker container run -d ubuntu

**--name <name> opsiyası**: Konteynerə xüsusi ad təyin edir.

docker container run --name mycontainer ubuntu

**docker container stop <container\_id>**: İşləyən konteyneri dayandırır.

docker container stop mycontainer

**docker container start <container\_id>**: Dayandırılmış konteyneri yenidən işə salır.

docker container start mycontainer

**docker container rm <container\_id>**: Bir konteyneri silir.

docker container rm mycontainer

**-f**: İşləyən konteyneri mecburi silmek.

docker container rm -f mycontainer

**docker container exec <container\_id> <command>**: İşləyən konteynerin içində bir əmr icra edir.

docker container exec mycontainer bash

Yeni container qaldirib və terminala qosulmaq ucun

docker run --name mycontainer -p 8088:80 -it nginx bash

**docker container logs <container\_id>**: Bir konteynerin loglarını göstərir.

docker container logs mycontainer

Son 10 eded log goster.

docker container logs –tail 10 mycontainer

**Container CPU and Memeory**

docker run -d --name my\_container --cpus="1.5" --memory="512m" --memory-swap="1g" my\_image

**Container CPU and Memory result stats**

Dockerda umumi containirlarin nə qədər cpu ve ram

istifadəsini gosterir.

docker stats

Hər hansi bir containirin cpu və ram istifadəsini.

docker stats my\_container name

**Volume**

Volume yaratmaq (docker volume create):

docker volume create myvolume

Volume-ləri siyahıya salmaq (docker volume ls):

docker volume ls

Volume məlumatlarını göstərmək (docker volume inspect):

docker volume inspect myvolume

Volume silmək (docker volume rm):

docker volume rm myvolume

Konteyner yaradarkən volume-i qoşmaq üçün -v və ya --mount opsiyasından istifadə edə bilərsiniz.

docker run -d -v myvolume:/app/data myimage

Qeyd

docker cp əmri Docker konteynerindən faylları host sisteminə və ya host sistemindən konteynerə kopyalamaq üçün istifadə olunur. Bu əmr fayl və ya qovluqların konteynerlə host arasında keçidini asanlaşdırır.

docker cp <container\_name\_or\_id>:<source\_path> <destination\_path>

**Dockerfile**

**FROM**: İmicin yaradılacağı baza imicini təyin edir.

FROM ubuntu:latest

**LABEL**: İmici yaradan şəxsin məlumatlarını qeyd edir.

LABEL maintainer="[yourname@example.com](mailto:yourname@example.com)"

**RUN**: İmic yaradılarkən icra olunacaq əmrləri təyin edir. Bu əmrlər Docker imicinin qurulması zamanı icra olunur.

RUN apt-get update && apt-get install -y curl

**COPY**: Host sistemindən konteynerə fayl və ya qovluqları kopyalamaq üçün istifadə olunur.

* Məsələn, host sistemindəki app qovluğunu konteynerdə /usr/src/app qovluğuna kopyalamaq:

COPY ./app /usr/src/app

**ADD**: COPY əmrinə bənzəyir, lakin ADD ilə siz faylları arxivdən açaraq kopyalaya, həmçinin URL-lərdən faylları yükləyə bilərsiniz.

ADD my\_archive.tar.gz /usr/src/app

**CMD**: Konteyner başladıqda icra olunacaq əsas əmri təyin edir. Bu, konteynerin "default" əmri olur. Bir Dockerfile-da yalnız bir CMD əmri olmalıdır.

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

**ENTRYPOINT**: Konteynerin başlanğıcında çalışacaq proqramı təyin edir. CMD ilə birgə istifadə oluna bilər, belə ki, CMD ilə argümentlər verilə bilər.

ENTRYPOINT ["/usr/bin/myapp"]

**ENV**: Konteynerdə ətraf mühit dəyişənlərini təyin edir.

ENV ENVIRONMENT=production

**EXPOSE**: Konteynerin hansı portları dinləyəcəyini göstərir. Bu portları host maşına yönləndirmək lazımdır.

EXPOSE 80

**VOLUME**: Konteynerdə bir qovluğu host sisteminə mount etmək üçün istifadə olunur. Bu, konteynerdən asılı olmayan məlumatların saxlanması üçün idealdır.

VOLUME /data

**WORKDIR**: İş qovluğunu təyin edir. Bu, bütün əmrlərin və əmr fayllarının işlədiyi qovluq olacaq.

WORKDIR /usr/src/app

**USER**: Konteyner içindəki əmrləri icra edəcək istifadəçini təyin edir.

Misal 1.

USER nonrootuser

FROM ubuntu:latest # Baza imic seçilir

ARG Version=”v.1” #deyisken olaraq tanimlamaq ucundur

ENV ENVIRONMENT=production # Ətraf mühit dəyişəni təyin olunur

WORKDIR /usr/src/app # İş qovluğu təyin olunur

COPY . /usr/src/app # Fayllar konteynerə kopyalanır

RUN apt-get update && apt-get install -y python3 # Lazımi paketlər quraşdırılır

CMD ["python3", "app.py"] # Default əmri təyin edir

Misal 2.

FROM mcr.microsoft.com/dotnet/aspnet:8.0 #lazim olan linux image istifade edir (image varsa ondan yoxsa hub yuleryir) ve bu imagen icinde core projesini run etmek ucun mueyyen proqramlar var IIS ve ya kestrel

WORKDIR /app #image icinde app papka acir

COPY /bin/Release/net8.0/publish . #[hostumda olan .net core proje publish kopyalayir .(o demekdirki oldugum papkaya) linux icindeki app papkasina atir]

ENTRYPOINT ["dotnet", "test.dll"] #[containira request gelende bu dll islet]

docker build -t myproje:v1 . #[qeyd bu Dockerfile image halina getirmek ucun istifade olunur. Sonra bu imageden contaner hazirlanir]

**Docker-compese**

docker-compose.yml adlı bir konfiqurasiya faylı istifadə edərək bütün konteynerlərin necə qurulacağını, şəbəkələrin necə təyin ediləcəyini və volume-lərin necə istifadə olunacağını göstərir.

docker-compose up: Konfiqurasiya faylında göstərilən konteynerləri işə salır.

**-d opsiyası**: Konteynerləri arxa planda işə salır.

docker-compose up -d

docker-compose -f docker-copmose.yml up

**docker-compose down:**İşləyən konteynerləri dayandırır və silir, həmçinin yaradılan şəbəkə və volume-ləri də silir.

docker-compose down

**docker-compose start**

* docker-compose up əmri ilə dayandırılmış konteynerləri yenidən işə salır.

docker-compose start

**docker-compose stop**

* Konteynerləri dayandırır, lakin konteynerlərin məlumatlarını və qurulmuş şəbəkələri saxlayır.

docker-compose stop

**docker-compose build**

* Dockerfile-dan imicləri yenidən qurur.

docker-compose build

**docker-compose logs**

* Konteynerlərin loglarını göstərir.

docker-compose logs

**docker-compose exec**

* İşləyən konteynerdə bir əmr icra edir.

docker-compose exec <service\_name> <command>

Misal 1.

version: '3'

services:

web:

image: nginx:latest

ports:

- "8080:80"

volumes:

- ./html:/usr/share/nginx/html

db:

image: postgres:latest

environment:

POSTGRES\_DB: mydatabase

POSTGRES\_USER: user

POSTGRES\_PASSWORD: password

volumes:

- db\_data:/var/lib/postgresql/data

volumes:

db\_data:

**Compose da CPU ve Memory**

### **docker-compose.yml-də CPU və RAM Məhdudiyyətləri Təyin Etmək**

#### **1. deploy Bölməsində Resurs Məhdudiyyətləri:**

Bu konfiqurasiya Docker Swarm mode-də daha geniş istifadə olunur. deploy bölməsi konteynerlərin CPU və RAM istifadəsini limitləməyə imkan verir.

version: '3.8' # Versiya Swarm mode dəstəyini göstərir

services:

web:

image: nginx:latest

deploy:

resources:

limits:

cpus: '0.5' # CPU limit: 50% (0.5 CPU)

memory: 512M # RAM limit: 512 MB

reservations:

cpus: '0.25' # Minimum CPU tələbi: 25% (0.25 CPU)

memory: 256M # Minimum RAM tələbi: 256 MB

**Network**

**Şəbəkə Yaradılması**:bridge,host,none,overlay

docker network create --driver bridge my\_network

**Şəbəkə Siyahısına Baxış**:

docker network ls

**Şəbəkə Detalları**:

docker network inspect my\_network

**Konteynerə Şəbəkə Qoşulması**:

docker run --network my\_network my\_image

**Şəbəkəni silmək**

docker network rm my\_network