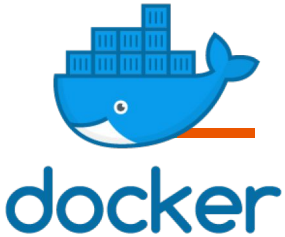

DOCKER



Nedir?

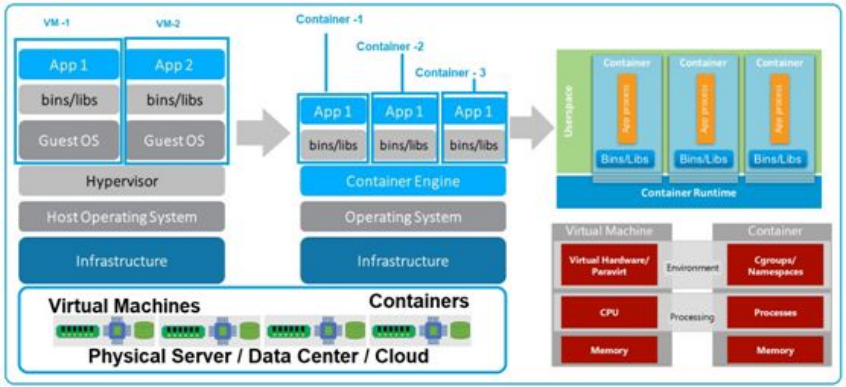
Docker, "containerization" olarak da bilinen işletim sistemi seviyesinde sanallaştırma sağlayan bir bilgisayar programıdır.

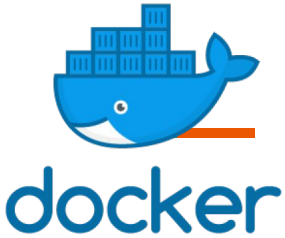
Herhangi bir donanımsal sanallaştırma teknolojisi kullanmadan container mantığıyla geliştirmelerimizi paketleyip, çalıştırdığımız ortamdır.

Container?: Docker ortamında kullanmak için hazırladığımız bir çıktıdır, Amacı geliştirmelerimizin belirli ihtiyaçlarını karşılayacak şekilde izole bir sanallaştırma ortamı sunmasıdır.

Neden?

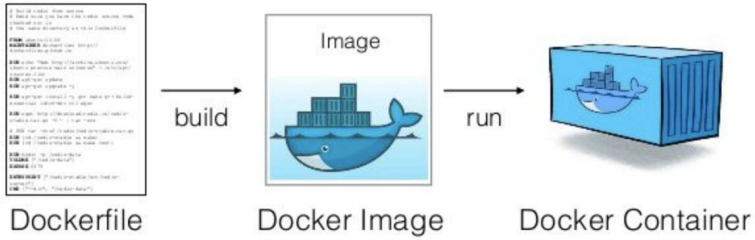
- Docker ortamı gerektiği kadar kaynak harcar, diğer donanımsal sanallaştırma teknolojilerinin tersine başlatmak için haricen bir donanımsal kaynağa ihtiyacı bulunmaz.
- Kurulu olduğu sistemin kaynaklarından “paylaşımlı” olarak kullanım yapılır.
- İçerisinde yeni bir işletim sistemi ayağa kaldırma ihtiyacı yoktur.
- Uygulamaların aynı standartta çalıştırılabilmesine olanak sağlar.
- Verimliliği arttırır ve orkestrasyon sağlar.



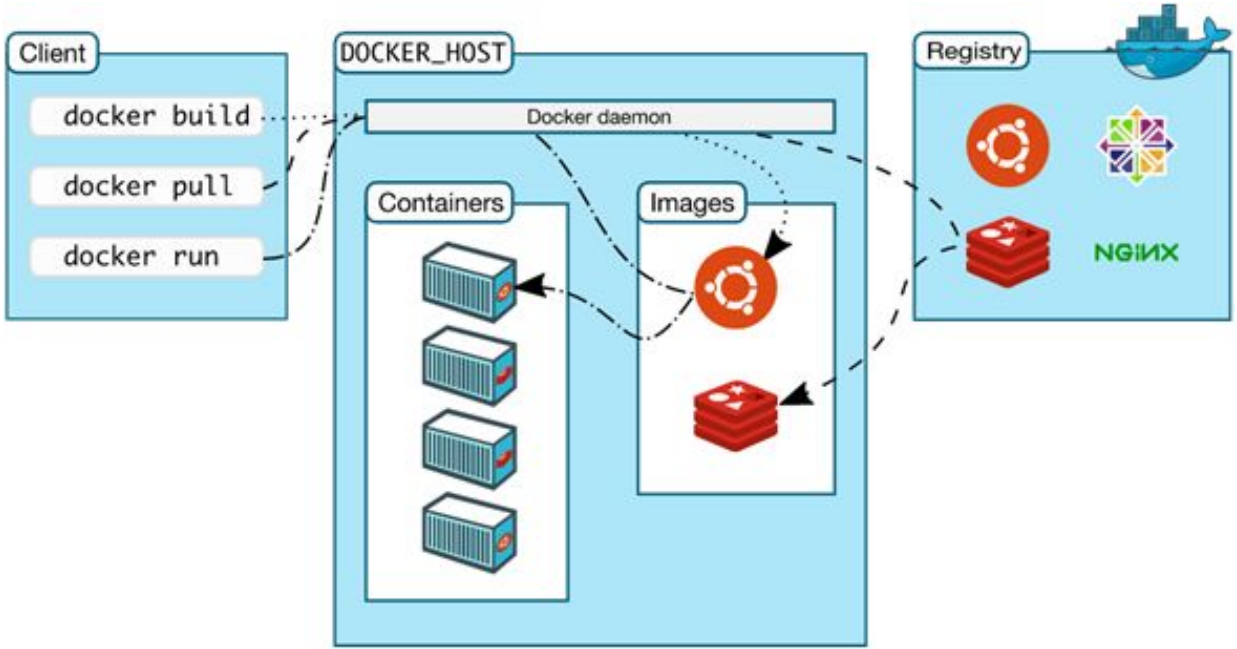


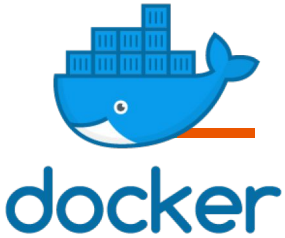
Nasıl?

- Docker'ın en temel ihtiyacı olan "Dockerfile"ı uygulamamızın ihtiyaçlarına ve akışına göre tasarlamamız gerekecektir.
- Dockerfile olduğu takdirde docker sizin için bu ortamı derleyecektir ve size bir "image" verecektir.
- Sizde docker ile paketlenmiş bir "image"ı kullanarak projenizi "container" olarak çalıştırabileceksiniz.



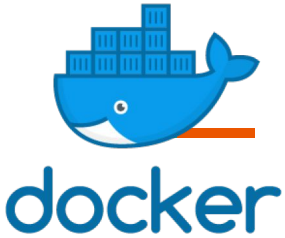
Mimari


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

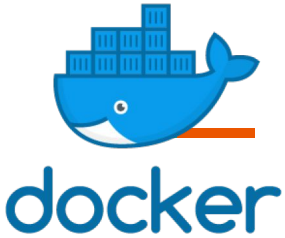


Örnek bir Dockerfile:

```
FROM maven:3.6-jdk-8 as jdk
WORKDIR /usr/src
COPY . .

RUN mvn -f common/pom.xml install \
    &&  mvn -f spring-app/pom.xml package

FROM tomcat:8.5.20-jre8 as tomcat

ENV MYSQL_HOST localhost
ENV MYSQL_USER root
ENV MYSQL_PASS pass
ENV MYSQL_DB springapp

COPY --from=jdk /usr/src/spring-app/target/backend.war /usr/local/tomcat/webapps

EXPOSE 80
ENTRYPOINT ["sh", "-c", "catalina.sh run"]
```



Nasıl Kurulur?



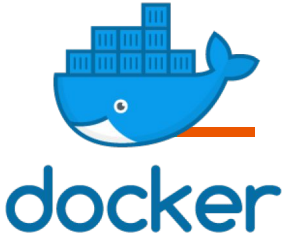
Temel Komutlar

docker build

Docker ortamında bir imaj oluşturulması için kullanılır.

--tag imaj için isim belirtir.
--no-cache build performansı için kullanılan caching mekanizmasını devre dışı bırakır.
--file Dockerfile dosyamızın konumu belirtilir.

```
 ---> 4e8e89fc647e
Step 17/17 : ENTRYPOINT ["java","-cp","app:app/lib/*","com.example.demo.DemoApplication"]
 ---> Running in 6ebbc282788b
Removing intermediate container 6ebbc282788b
 ---> aa9c2af4a54a
Successfully built aa9c2af4a54a
Successfully tagged springapp:1.0.0
root@s2:~/docker/springboot-app# docker build -f Dockerfile --no-cache -t springapp:1.0.0 .
```



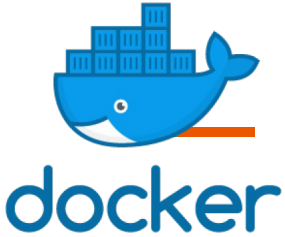
Temel Komutlar

docker run

İmajdan yeni bir container oluşturur.

-p 80:80 host:container formatında port yönlendirmesi sağlar.
--name container için bir isim belirtir.
--restart always olası problemlerde containerı yeniden başlatmayı sağlar.
-v /usr/app:/app host bilgisayardan bir volume kullandırmayı sağlar.
-d arkaplanda container çalışır.
--rm container durduğu takdirde otomatik silinir.
--it container attach edilerek çalıştırılır.

```
root@s2:~/docker/springboot-app# docker run --rm -p 80:8000 --name spring_container -d springapp:1.0.0
4513f00e04543063e244c005f426c22bb21ba95a1b8e84f5a1190e66e3dd2b13
root@s2:~/docker/springboot-app#
```



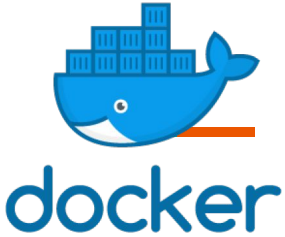
Temel Komutlar

docker exec

Container içerisinde bash erişimi veya herhangi bir komutun çalıştırılması için gerekli komuttur.

-i interaktif oturum, P + Q komutu ile detach edilebilir.
-t tty bağlantısı
-d komutu arkaplanda uygula

```
root@s2:~/docker/springboot-app# docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS                    PORTS                              NAMES
2634f174b25c        mysql:8.0                  "docker-entrypoint.s…"   9 days ago          Up 9 days                 0.0.0.0:3306->3306/tcp, 33060/tcp  mysql-server_db_1
bde99f7c7b72        jacobalberty/firebird:3.0  "/usr/local/firebird…"   9 days ago          Up 7 days (healthy)       0.0.0.0:3050->3050/tcp             firebird_firebird_1
root@s2:~/docker/springboot-app# docker exec -it 263 mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
root@s2:~/docker/springboot-app#
```



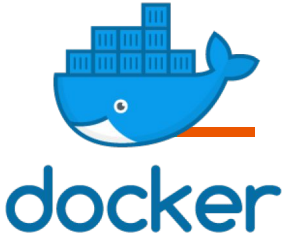
Temel Komutlar

docker ps

Docker üzerinde çalışan “container”ları göstermektedir.

-a parametresi ile kullanmanız kapatılmış ve silinmemiş “container”leri de gösterecektir.
-f status=exited parametresi sadece kapatılmış “container”leri gösterecektir.

```
root@s2:~/docker/springboot-app# docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS                  PORTS                                NAMES
4513f00e0454        springapp:1.0.0                "java -cp app:app/li…"   About a minute ago  Up 59 seconds           0.0.0.0:80->8000/tcp                 spring_container
2634f174b25c        mysql:8.0                      "docker-entrypoint.s…"   9 days ago          Up 9 days               0.0.0.0:3306->3306/tcp, 33060/tcp    mysql-server_db_1
bde99f7c7b72        jacobalberty/firebird:3.0      "/usr/local/firebird…"   9 days ago          Up 7 days (healthy)     0.0.0.0:3050->3050/tcp               firebird_firebird_1
root@s2:~/docker/springboot-app#
```



Temel Komutlar

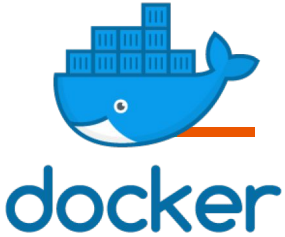
docker stop

Docker üzerinde çalışan “container”ı kapatmaktadır.

Sadece kapatılacak “container”ın id’sini alması yeterlidir.

--time 10 ile 10 saniye sonra kapatılacak şekilde parametre belirtilebilir.

```
root@s2:~/docker/springboot-app# docker stop 451
451
root@s2:~/docker/springboot-app# docker ps -a
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS                    PORTS                                NAMES
2634f174b25c        mysql:8.0                   "docker-entrypoint.s…"   9 days ago          Up 9 days                 0.0.0.0:3306->3306/tcp, 33060/tcp    mysql-server_db_1
bde99f7c7b72        jacobalberty/firebird:3.0   "/usr/local/firebird…"   9 days ago          Up 7 days (healthy)       0.0.0.0:3050->3050/tcp               firebird_firebird_1
root@s2:~/docker/springboot-app# docker run -p 80:8000 --name spring_container --restart always -d springapp:1.0.0
9df5c4a756c091b3df436af88f5c0df6b5268d060aa38b0c6318540fe3602ce3
root@s2:~/docker/springboot-app# docker stop 9df
9df
root@s2:~/docker/springboot-app# docker ps -a
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS                        PORTS                                NAMES
9df5c4a756c0        springapp:1.0.0             "java -cp app:app/li…"   15 seconds ago      Exited (143) 2 seconds ago                                         spring_container
2634f174b25c        mysql:8.0                   "docker-entrypoint.s…"   9 days ago          Up 9 days                     0.0.0.0:3306->3306/tcp, 33060/tcp    mysql-server_db_1
bde99f7c7b72        jacobalberty/firebird:3.0   "/usr/local/firebird…"   9 days ago          Up 7 days (healthy)           0.0.0.0:3050->3050/tcp               firebird_firebird_1
root@s2:~/docker/springboot-app#
```

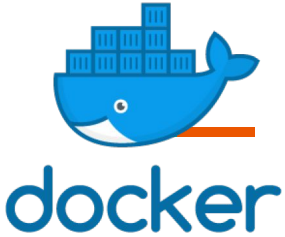


Temel Komutlar

docker images

Docker üzerinde indirilmiş olan veya derlenen “image”lar bulunur.

```
root@s2:~/docker/springboot-app# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
springapp                           1.0.0               aa9c2af4a54a        13 minutes ago      126MB
<none>                              <none>              dad3c8172f49        13 minutes ago      244MB
<none>                              1.0.0               e3eac2175d52        6 days ago          1.32GB
mysql                               8.0                 afaec1334369        12 days ago         471MB
python                              3.6                 971c66f6a27f        3 weeks ago         914MB
jacobalberty/firebird               3.0                 aa61209a7ef9        5 weeks ago         189MB
swarmpit/agent                      <none>              14028bf766eb        8 months ago        11.9MB
openjdk                             8-jdk-alpine        a3562aa0b991        9 months ago        105MB
marianaldenhoevel/firebirdwebadmin  latest              96c70bc1f9d0        2 years ago         446MB
```



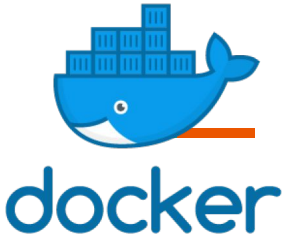
Temel Komutlar

docker rm

docker rmi

- rm komutu durdurulmuş bir containeri silerken kullanılır.
- rmi komutu ise imageları silmektedir.

```
root@s2:~/docker/springboot-app# docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS                        PORTS                                NAMES
9df5c4a756c0        springapp:1.0.0                "java -cp app:app/li…"   9 minutes ago       Exited (143) 9 minutes ago                                         spring_container
2634f174b25c        mysql:8.0                      "docker-entrypoint.s…"   9 days ago          Up 9 days                     0.0.0.0:3306->3306/tcp, 33060/tcp    mysql-server_db_1
bde99f7c7b72        jacobalberty/firebird:3.0      "/usr/local/firebird…"   9 days ago          Up 7 days (healthy)           0.0.0.0:3050->3050/tcp               firebird_firebird_1
root@s2:~/docker/springboot-app# docker rm 9df
9df
root@s2:~/docker/springboot-app# docker images
REPOSITORY                                TAG                 IMAGE ID            CREATED             SIZE
springapp                                 1.0.0               aa9c2af4a54a        18 minutes ago      126MB
marm                                      1.0.0               e3eac2175d52        6 days ago          1.32GB
mysql                                     8.0                 afaec1334369        12 days ago         471MB
python                                    3.6                 971c66f6a27f        3 weeks ago         914MB
jacobalberty/firebird                     3.0                 aa61209a7ef9        5 weeks ago         189MB
swarmpit/agent                            <none>              14028bf766eb        8 months ago        11.9MB
openjdk                                   8-jdk-alpine        a3562aa0b991        9 months ago        105MB
marianaldenhoevel/firebirdwebadmin        latest              96c70bc1f9d0        2 years ago         446MB
root@s2:~/docker/springboot-app# docker rmi aa9c2af4a54a
```



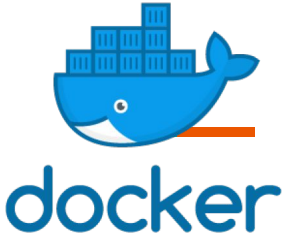
Temel Komutlar

docker cp

Container ile dosya alışverişi yapılmasına olanak sağlar.

Dosya alışverişi iki yönlü olabilir.

```
root@s2:~/docker/springboot-app# docker cp 263:/var/lib/mysql .
root@s2:~/docker/springboot-app# ls
Dockerfile  mvnw  mvnw.cmd  mysql  pom.xml  src
root@s2:~/docker/springboot-app#
```



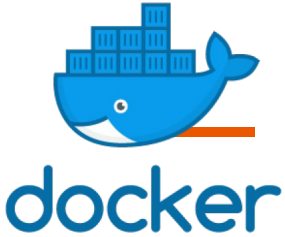
Temel Komutlar

docker logs

Container içerisinde gerçekleşen olayları takip edebilmenizi sağlar

-f Logları takip eder. Kullanmadığınız takdirde o ana kadarki loglar gelecektir.

```
root@s2:~/mysql-server# docker logs -f mysql-server_db_1
2020-03-05 04:30:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.19-1debian9 started.
2020-03-05 04:30:52+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2020-03-05 04:30:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.19-1debian9 started.
```

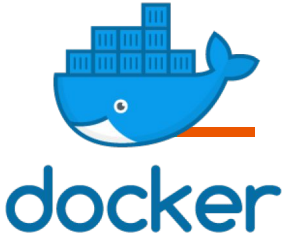


Temel Komutlar

docker stats

Docker hostunuzda bulunan aktif containerların kaynak tüketimlerini belirtir.

```
CONTAINER ID        NAME                    CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
15dd12bfcbac        mysql-server_db_1       0.94%               392.1MiB / 1.9GiB     20.15%              1.87kB / 0B         76.4MB / 20.6MB     38
bde99f7c7b72        firebird_firebird_1     0.00%               20.45MiB / 1.9GiB     1.05%               57.5MB / 185MB      329MB / 67MB        10
```

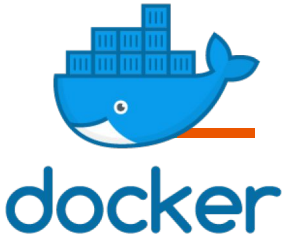


Temel Komutlar

docker system prune --volumes -f

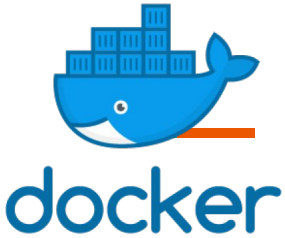
Docker ortamınızdaki *kullanılmayan* her volume, container, image, network katmanını silmektedir.

```
root@s2:~/docker/springboot-app# docker system prune --volumes -f
Deleted Volumes:
82840e30fc5100ed57597cea9dc6a7516e3da4608f190bcca8bcdda05d4e569c

Deleted Images:
untagged: swarmpit/agent@sha256:ebb5f35592d8c00196310af7084ec52352e5068b4f67de69b88a61c34e3104ec
deleted: sha256:14028bf766ebf77cf9b1c780d8390744d3d25a778b19407f79266a70eb5f872d
deleted: sha256:eb75335a76327a72d40d96ed69bea9a82b7c0658dd1d9b71f124e508f0d5c661

Total reclaimed space: 11.88MB
```



Nerede Kalmıştık?

```
FROM maven:3.6-jdk-8 as jdk
WORKDIR /usr/src
COPY . .

RUN mvn -f common/pom.xml install \
    &&  mvn -f spring-app/pom.xml package

FROM tomcat:8.5.20-jre8 as tomcat

ENV MYSQL_HOST localhost
ENV MYSQL_USER root
ENV MYSQL_PASS pass
ENV MYSQL_DB springapp

COPY --from=jdk /usr/src/spring-app/target/backend.war /usr/local/tomcat/webapps

EXPOSE 80
ENTRYPOINT ["sh", "-c", "catalina.sh run"]
```



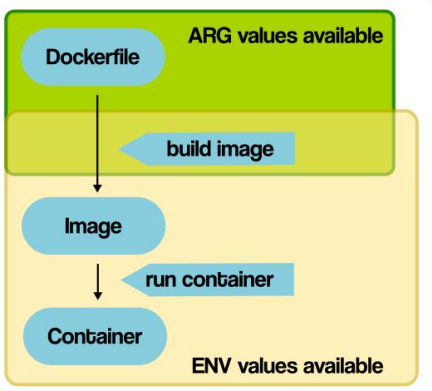
Daha fazla Dockerfile:

```
FROM ubuntu:latest
MAINTAINER Andrew Odewahn "odewahn@oreilly.com"

RUN apt-get update
RUN apt-get install -y python python-pip wget
RUN pip install Flask

ADD hello.py /home/hello.py

WORKDIR /home
```

```
FROM python:3.4-slim

RUN apt-get update && apt-get install -y \
                gcc \
                gettext \
                mysql-client libmysqlclient-dev \
                postgresql-client libpq-dev \
                sqlite3 \
        --no-install-recommends && rm -rf /var/lib/apt/lists/*

ENV DJANGO_VERSION 1.10.4
```

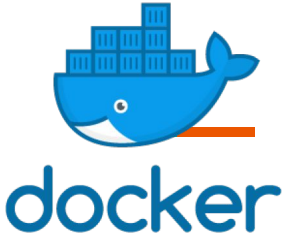
```
FROM frolvlad/alpine-oraclejdk8:slim
VOLUME /tmp
ADD gs-spring-boot-docker-0.1.0.jar app.jar
RUN sh -c 'touch /app.jar'
ENV JAVA_OPTS=""
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ]
```

ARG Ve ENV?



ENV is for future running containers.

ARG for building your Docker image.

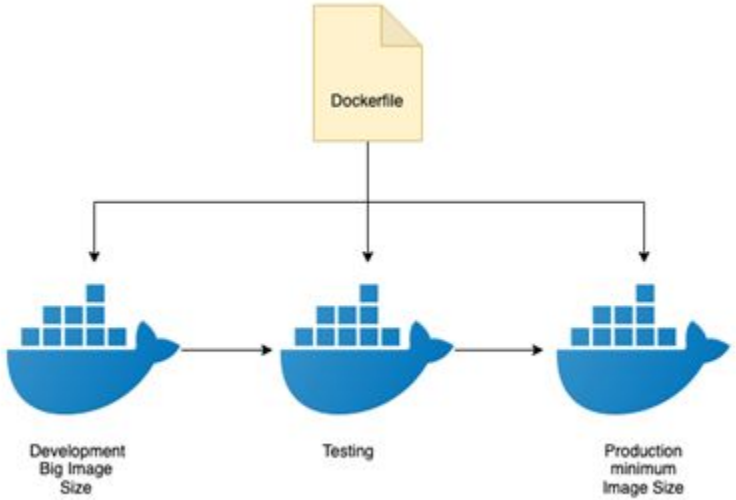


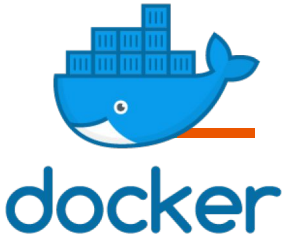
Multi Stage?

- Docker ile oluşturduğumuz imajlar daha önceki imajları extend ettiği için oluşan imaj birden fazla “layer” içermektedir.
- Bu layer’lar cache optimizasyonu sağlar. Container boyutları önemli ölçüde azaltır.
- Docker multi-stage build özelliği ile zamandan ve işlem gücünden büyük ölçüde tasarruf sağlar.

Multi Stage

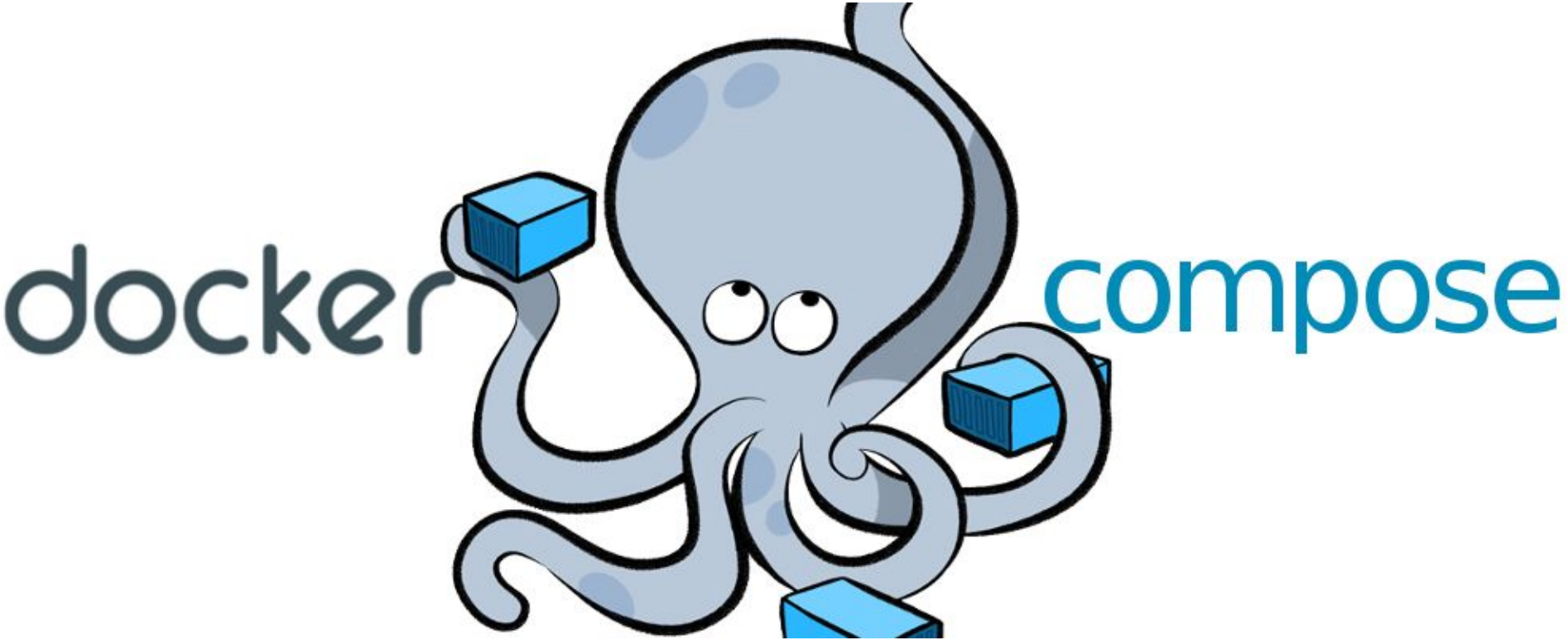
```
1  # Stage - Development
2  FROM Dev-Image as dev
3  . . .          stage 0
4  . . .
5
6  # Stage - Testing
7  FROM Test-Image as test
8  COPY --from=dev /src/app .
9  . . .          stage 1
10 . . .
11
12 # Stage - Production
13 FROM Minimum-Image as prod
14 COPY --from=dev /src/app .
15 . . .          stage 2
16 . . .
```

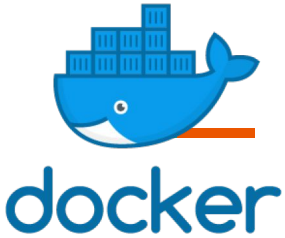




Sonuç Olarak:

- Versiyon yönetimi çok kolay, eski paketleri silmek zorunda değiliz, dilediğimiz zaman elimizin altında.
- Zamandan büyük oranda tasarruf sağlar.
- CI Ve CD pipelinelar için harikadır.
- Tamamen Tak Çalıştır, Kurulan her sistemde sorunsuz aynı standartlarda çalışacaktır.
- Dilediğiniz zaman kaynak kısıtlaması yapabilirsiniz.
- Orkestrasyon işlerinizin temelini hazırlar.


docker
compose

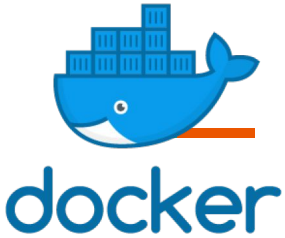


Compose Nedir?

Birden fazla “container”ın tek bir “stack” olarak çalıştırılması için gerekli olan taslaktır.

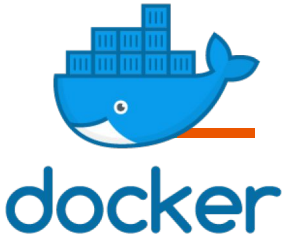
Temel olarak aynı göreve dahil uygulamalar bir compose’da toplanabilir.

Farklı network katmanları, volume’ler, environment tanımları tek bir “compose”da bulunabilir.



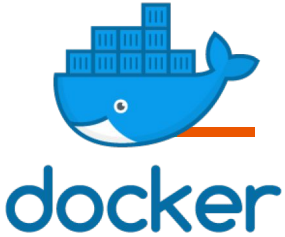
docker-compose.yml

```yaml
services:

  db:
    image: mysql:8.0
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=YR4kg4nY.
      - MYSQL_DATABASE=myapp
      - MYSQL_USER=myapp
      - MYSQL_PASSWORD=YR4kg4nY.
    ports:
      - 3306:3306
    volumes:
      - mysql_data:/var/lib/mysql
volumes:
  mysql_data:
```



Compose

```
web:
    image: tomcat:8
    container_name: tomcat-container
    ports:
        - "80:8080"
    hostname: foo.com
    volumes:
        - /app/webapps:/usr/local/tomcat/webapps
    depends_on:
        - db

db:
    image: mysql:8.0
    container_name: mysql-server
    environment:
        MYSQL_ROOT_PASSWORD: sqV2XZL.
        MYSQL_DATABASE: myapp
    volumes:
        db_volume:/var/lib/mysql

volumes:
    db_volume:
```



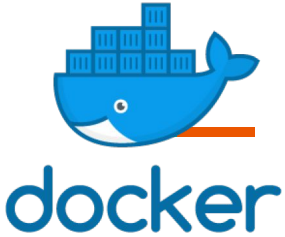
Temel Komutlar

docker-compose up

Compose dosyasını çalıştırır.

-d arkaplanda çalıştırır

```
root@s2:~/mysql-server# docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "mysql-server_default" with the default driver
Creating mysql-server_db_1 ... done
root@s2:~/mysql-server#
```

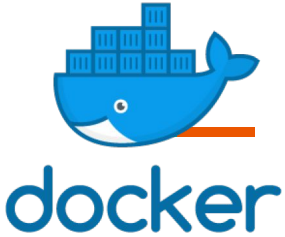


Temel Komutlar

docker-compose down

Compose dosyasını durdurur.

```
root@s2:~/mysql-server# docker-compose down
Stopping mysql-server_db_1 ... done
Removing mysql-server_db_1 ... done
Removing network mysql-server_default
```

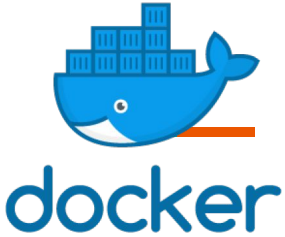


Temel Komutlar

docker-compose ps

Compose tarafından oluşturulan containerleri görüntüler.

```
root@s2:~/mysql-server# docker-compose ps
      Name                     Command               State                 Ports
-------------------------------------------------------------------------------------------
mysql-server_db_1   docker-entrypoint.sh --def ...   Up      0.0.0.0:3306->3306/tcp, 33060/tcp
root@s2:~/mysql-server#
```



Temel Komutlar

docker-compose build

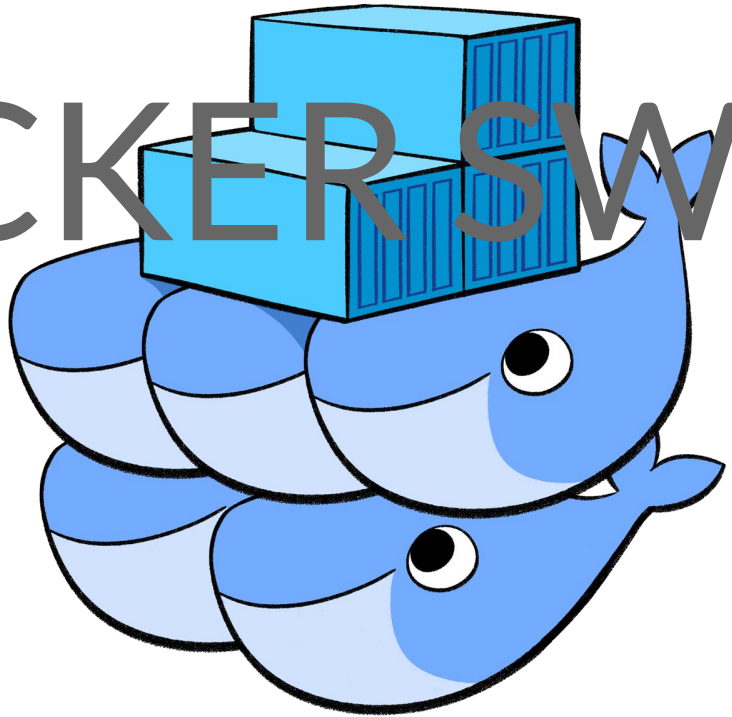
Derlenmesi gereken docker build'leriniz olduğu taktirde bu şekilde derleyebileceksiniz.

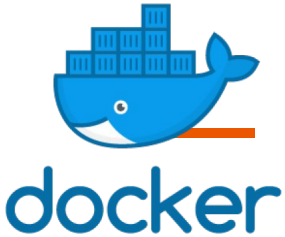
Compose içerisinde ayrıca bir build parametresi gerekecektir. Sadece imajlardan oluşan bir compose'unuz var ise aşağıdaki hatayı alırsınız.

```
root@s2:~/mysql-server# docker-compose build
db uses an image, skipping
```

```
root@s2:~/docker/springboot-app# docker-compose build
Building springboot-docker-compose-app-container
Step 1/17 : FROM openjdk:8-jdk-alpine as build
 ---> a3562aa0b991
Step 2/17 : WORKDIR /workspace/app
 ---> Running in 0f80b1bc875b
Removing intermediate container 0f80b1bc875b
 ---> 61a326585629
Step 3/17 : COPY mvnw .
 ---> b67db26f11da
Step 4/17 : COPY .mvn .mvn
 ---> a9e1bf06af4c
Step 5/17 : COPY pom.xml .
```

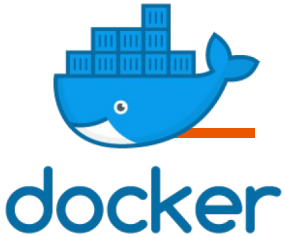
DOCKER SWARM





Swarm Nedir?

Birden fazla sunucu üzerinde container orchestration yapmayı sağlar.

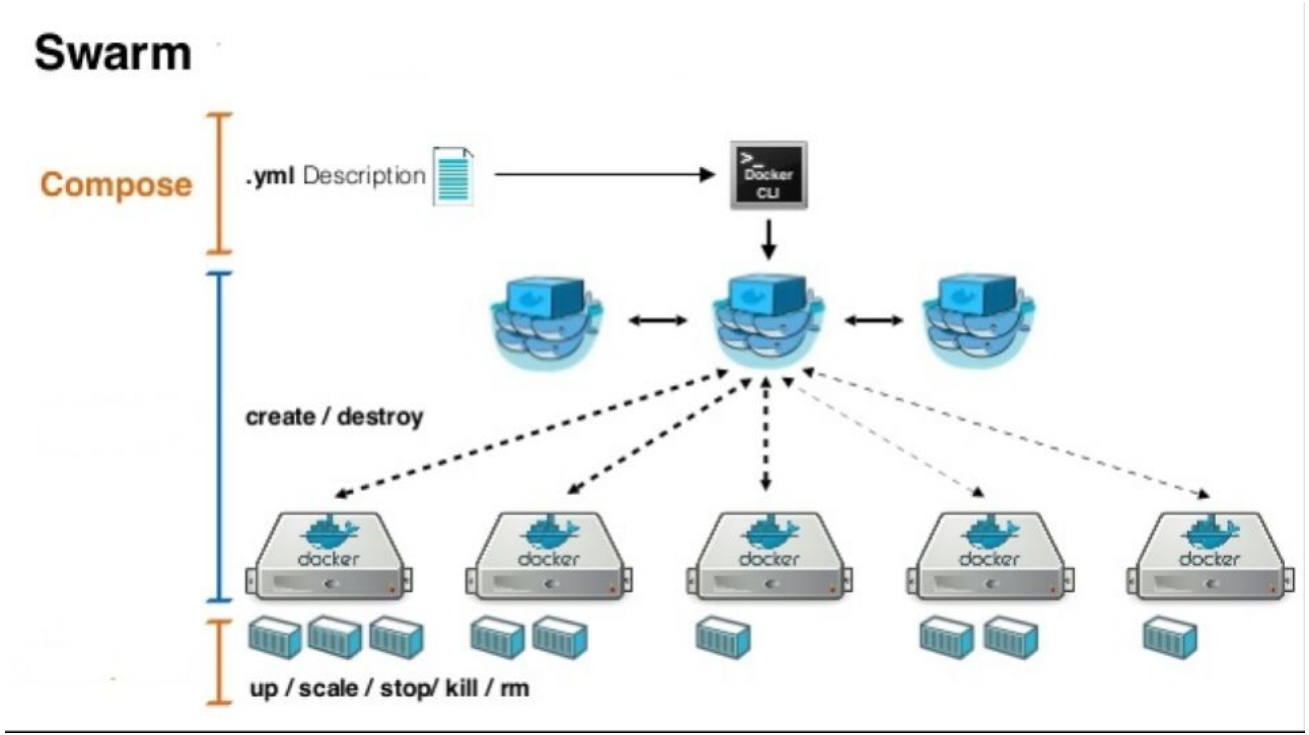


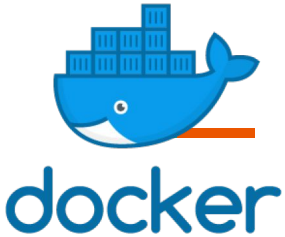
Neden?

Sadece Docker Swarm kullanarak

- Self Healing
- Load balancing
- Service discovery & Dns management
- Container scaling
- Rolling update ve Rollback
- Service monitoring

Nasıl?

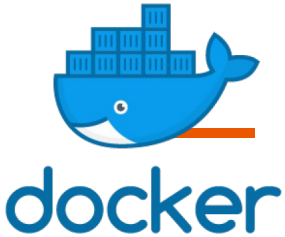

Swarm
Compose
.yml Description
Docker CLI
create / destroy
docker
docker
docker
docker
docker
up / scale / stop/ kill / rm



Nasıl?

docker-compose.yml dosyalarını kullanarak swarm üzerinde bir stack çalıştırabiliriz.

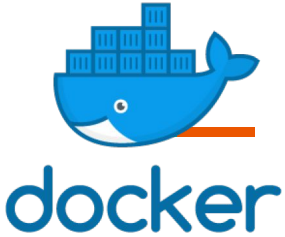
Swarm'ın tüm özelliklerini harici bir konfigurasyon yapmaya gerek kalmadan kullanmaya başlayabilirsiniz.



Self Healing

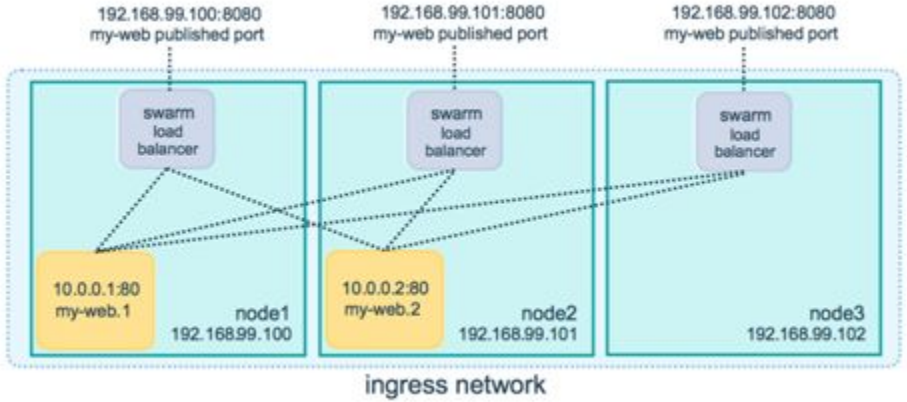
Container doğru çalışmadığı zamanda, haricen bir işlem yapmadan swarm sizin için yeni container'ı çalıştıracaktır.

Herhangi bir down-time olmayacaktır.



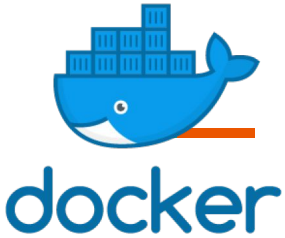
Load Balancing & Service Discovery

Servisin her zaman doğru adrese gitmesinden emin olmak, yük paylaşımı yapmak, stackler arasında geçersiz adreslemenin önüne geçilmesi için swarmın load balancing özelliğini kullanabilmekteyiz.



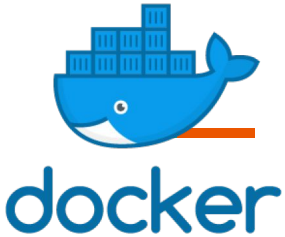
Container Scaling

Farklı yük kapasiteli sunucular için oluşturabileceğiniz etiketlere göre ağır yük ihtiyacınız olan uygulamaları doğru sunucularda barındırmak ve gerektiğinde aynı sunucunun replikalarını barındırarak anlık yük ihtiyaçlarını karşılayacaktır.



Rolling Update & Rollback

Herhangi bir down-time olmadan yeni güncellemelerin geçişinin yapılması, gerektiğinde bu güncellemelerin down-time olmadan geri alınması işlerini tek komutla yapılabilmesine olanak sağlar.



Service Monitoring

Cpu, Memory, Disk gibi kaynak tüketimlerinin loglanması,
Her bir node, container, service kırılımı ile bu tüketimlerin detaylandırılması,
Alertların oluşturulması için real time monitoring sağlayan bir yapıdır.

Hazır stackler olarak paylaşılmaktadır.

TEŞEKKÜRLER

https://github.com/Shavell
https://www.linkedin.com/in/Shavell

Ümit Mehmet Yarımbaş