DOCKER KOMUTLARI

`docker container ls`	Docker konteynerları listeler.
`docker ps -a`	
`docker image ls`	Docker imajları listeler.
`docker volume ls`	Docker volumleri listeler.
`docker volume rm volume ad`	Docker volumü siler.
`docker volume prune`	Tüm docker volumleri siler.
`docker image rm nginx`	Docker imajı siler.
`docker image prune -af`	Tüm docker imajları siler.
`docker image pull centos`	Centos docker imajını çeker.
`docker container run ozgurozturknet/app1`	ozgurozturknet/app1 isimli imajdan bir container yaratır.
`docker container run -d httpd:alpine`	httpd:alpine isimli imajdan detached bir container yaratır.
`docker logs 280`	Konteyner loglarına bakmayı sağlar.
`docker container stop/start 280`	Konteyneri durdurur/başlatır.
`docker container run -d -p 80:80 ozgurozturknet/adanzyedocker`	ozgurozturknet/adanzyedocker imajından
`docker container exec -it 9ee sh`	Container'a bağlanır.
`docker container run alpine ls`	alpine imajından bir container yaratıp,
`docker volume create alistirma1`	"alistirma1" isimli bir volüme yaratır.
`docker volume inspect alistirma1`	Bir veya daha fazla volum ile ilgili ayrıntılı bilgileri görüntüler.
`docker container run --name birinci -it -v alistirma1:/test alpine`	alpine imajından "birinci" adlı container'ı interactive modda yaratıp bağlanır,
`docker container run -d --name websunucu1 -p 80:80 -v C:\deneme:/usr/local/apache2/htdocs ozgurozturknet/adanzyedocker`	
`docker network create kopru1`	Varsayılan driver olan bridge ile yeni bir network yaratır.
`docker network inspect kopru1`	Belirtilen network ayarlarının ayrıntılı gösterimi sağlanır.
`docker network create --driver=bridge --subnet=10.10.0.0/16 --ip-range=10.10.10.0/24 --gateway=10.10.10.10 kopru2`	
`docker container run -dit --name websunucu --net kopru1 ozgurozturknet/adanzyedocker sh`	
`docker attach websunucu`	Daha önce -dit ile oluşturulan container terminaline girişi sağlar.
`docker network connect kopru2 database`	Bir cihazı (database makinası) yeni bir ağa (kopru2 bridge network'üne) bağlar.
`docker network disconnect kopru2 database`	Bir cihazın (database makinası) bir ağ (kopru2 bridge network'ü) bağlantısını keser.
`docker logs con1`	Bir container'ın loglarını getirir.
`docker logs --details con1`	Bir container'ın loglarına sağlanan ek ayrıntıları gösterir.
`docker logs -t con1`	Container'ın zaman damgalarını gösterir.
`docker logs -f con1`	Container'ın logunu canlı olarak gösterir.
`docker logs -n 5 con1`	Container logunun sonundan itibaren 5 satırı gösterir.
`docker logs --until 2021-09-29T18:34:09.221297705Z con1`	Container'ın loglarını bir zaman damgasından önce gösterir.
`docker logs --since 2021-09-29T18:34:05.406332227Z con1`	Container'ın loglarını bir zaman damgasından sonra gösterir.
`docker top con1`	Bir container'da çalışan işlemleri görüntüler.
`docker stats` veya `docker stats con1`	Konteyner(ler) kaynak kullanım istatistiklerinin canlı akışını görüntüler.
`docker container run -d --memory=100m --memory-swap=200m ozgurozturknet/adanzyedocker`	
`docker container run -d --cpus="1.5" ozgurozturknet/adanzyedocker`	
`docker container run -d --cpus="2" --cpuset-cpus="0,3" ozgurozturknet/adanzyedocker`	
`docker container run -it --env VAR1=deneme1 -e VAR2=deneme2 ubuntu bash`	
`docker network create --driver=bridge --subnet=10.10.0.0/16 --ip-range=10.10.10.0/24 --gateway=10.10.10.10 alistirma-agi`	
`docker container run -d -p 8080:80 --name web1 --net alistirma-agi nginx:1.16`	
`docker container run -d --name websrv --net alistirma-agi --cpus="2" -p 80:80 --env-file env.list ozgurozturknet/webkayit`	
`docker container run -d --name mysqldb --net alistirma-agi --memory=1g --env-file envmysql.list ozgurozturknet/webdb`	
`docker image pull ubuntu:20.04`	ubuntu imajının 20.04 tagli olanını sisteme çeker.
`docker image tag ubuntu:18.04 kazimesen/alistirma:ubuntu`	ubuntu:18.04 imajına kazimesen/alistirma:ubuntu olarak tag ekler.
`docker image push kazimesen/alistirma:ubuntu`	alistirma:ubuntu olarak tag eklenen bu yeni imajı docker hub'a gönderir.

Dockerfile Talimatları ve Açıklamaları

FROM Ör: FROM ubuntu:18.04	İmajın hangi imajdan oluşturulacağını belirten, Dockerfile içerisinde olması zorunlu tek talimattır.
LABEL Ör: LABEL version:1.0.8	İmaj metadata'sına key=value şeklinde değer çiftleri eklemek için kullanılır. Örneğin team=development gibi.
RUN Ör: RUN apt-get update; apt-get install nginx	İmaj oluşturulurken shell'de bir komut çalıştırmak için kullanılır.
WORKDIR Ör: WORKDIR /usr/src/app	cd xxx komutuyla ile istediğimiz klasöre geçmek yerine bu talimat kullanılarak istediğimiz klasöre geçer ve oradan çalışmaya devam ederiz.
USER Ör: USER poweruser	Bu talimat; komutları çalıştıracak kullanıcı belirtmeyi sağlar.
COPY Ör: COPY /source /user/src/app	İmaj içine dosya veya klasör kopyalamada kullanılır.
ADD Ör: ADD https://wordpress.org/latest.tar.gz /temp	COPY gibi, dosya-klasör kopyalamanın yanında ADD kaynağın bir url olmasına da izin verir. ADD kaynakta belirtilen .tar dosyasını, imaja .tar olarak değil de açarak kopyalar.
ENV Ör: ENV TEMP_FOLDER="/temp"	Imaj içinde environment variable tanımlamak için kullanılır
ARG Ör: ARG VERSION:1.0	ARG ile de variable tanımlanır. Fakat bu variable sadece imaj oluşturulurken yani build aşamasında kullanılır.
VOLUME Ör: VOLUME /myvol	Imaj içerisinde volume tanımlar. Eğer bu volume host sistemde varsa container bunu kullanır, yoksa oluşturur.
EXPOSE Ör: EXPOSE 80/tcp	Bu imajdan oluşturulacak containerlara erişilecek portu yani yayın yapacağı portu belirtir.
ENTRYPOINT Ör: ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]	Bu talimat ile bir containerın çalıştırılabilir bir uygulama gibi ayarlanabilmesini sağlanır.
CMD Ör: CMD java merhaba	Bu imajdan container yaratıldığında varsayılan olarak çalıştırılacak komut bu talimat ile belirtilir.
SHELL Ör: SHELL ["powershell", "-command"]	Dockerfile'ın komutları işleyeceği shell'in belirtildiği talimattır. Linux için varsayılan shell ["/bin/sh", "-c"],Windows için ["cmd", "/S", "/C"].
HEALTHCHECK HEALTHCHECK --interval=5m --timeout=3s CMD curl -f http://localhost/ || exit 1	Konteynerin çalışıp çalışmadığını kontrol eder. Docker varsayılan olarak container içerisinde çalışan ilk prosesi izler ve o çalıştığı sürece container çalışmaya devam eder. Ama process çalışsa bile düzgün işlem yapmayabilir.

1-Her docker imajında bir CMD veya ENTRYPOINT talimatı bulunmalıdır.
2-CMD veya ENTRYPOINT talimatlarının her ikisi de bu imajdan container yaratıldığında çalıştırılacak uygulamanın belirtilmesini sağlar.
3-ENTRYPOINT ile girilen komut runtime'da yani container çalıştırılırken değiştirilemezken CMD ile yazılan ise değiştirilebilir.
4-ENTRYPOINT ve CMD aynı anda kullanılırsa CMD'de ENTRYPOINT talimatında yazılana parametre olarak eklenir.

```
docker image build -t kazimesen/alistirma:sari --build-arg RENK=sari .
docker image build -t kazimesen/alistirma:kirmizi --build-arg RENK=kirmizi .
docker container run -d -p 80:80 --name kirmizi kazimesen/alistirma:kirmizi
docker container run -d -p 8080:80 --name sari -e KULLANICI="deneme" kazimesen/alistirma:sari
docker run -d -p 5000:5000 --restart always --name registry registry
docker image tag kazimesen/alistirma:sari 127.0.0.1:5000/sari:latest
docker image tag kazimesen/alistirma:kirmizi 127.0.0.1:5000/kirmizi:latest
docker image tag kazimesen/alistirma:java 127.0.0.1:5000/java:latest
docker push 127.0.0.1:5000/java:latest
docker push 127.0.0.1:5000/sari:latest
docker push 127.0.0.1:5000/kirmizi:latest
```

Exec Form	Shell Form
CMD ["java", "uygulama"]	**CMD java uygulama**

1: Eğer komut Shell formunda girilirse Docker bu imajdan container yaratıldığında bu komutu varsayılan shell'i çalıştırarak onun içinde girer.
Bu nedenle containerda çalışan 1. Process yani pid1 bu shell process'i olur.

2: Eğer komut Exec formunda girildiyse Docker herhangi bir shell çalıştırmaz ve komut direk process olarak çalışır ve container'in pid1'i o process olur.

3: Exec formunda çalıştırılan komutlar herhangi bir shell processi çalışmadığı için Environment Variable gibi bazı değerlere erişemezler. Bunu göz önünde bulundurmak gerekir.

4: Eğer Entrypoint ve CMD birlikte kullanılacaksa Exec form kullanılmalıdır. Shell formu kullanıldığında CMD'deki kumutlar ENTRYPOINT'e parametre olarak aktarılmaz.

Multi-stage Build : İmaj yaratma aşamasını kademelere bölmeye ve ilk kademede yaratılan imaj içerisindeki dosyaları bir sonraki kademede oluşturulacak imaja kopyalama imkanı sağlar.

```
FROM mcr.microsoft.com/java/jdk:8-zulu-alpine AS derleyici
COPY /source /usr/src/uygulama
WORKDIR /usr/src/uygulama
RUN javac uygulama.java

FROM mcr.microsoft.com/java/jre:8-zulu-alpine
WORKDIR /uygulama
COPY --from=derleyici /usr/src/uygulama .
CMD ["java" , "uygulama"]
```

```
FROM ubuntu:latest
WORKDIR /gecici
ADD https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tgz .
CMD ls -al"
```

```
FROM ubuntu:latest
WORKDIR /gecici
ARG VERSION
ADD https://www.python.org/ftp/python/${VERSION}/Python-${VERSION}.tgz .
CMD ls -al
```

```
docker image build -t uygulama:3.7.1 .
```

```
docker image build -t x1 --build-arg VERSION=3.7.1 .
```

Docker Commit

```
docker container run -it --name con1 ubuntu:latest bash
```

```
root@a0c865a6ee83:/# apt-get update
root@a0c865a6ee83:/# mkdir /temp
root@a0c865a6ee83:/# cd /temp
root@a0c865a6ee83:/temp# apt-get install wget
root@a0c865a6ee83:/temp# wget https://wordpress.org/latest.tar.gz
root@a0c865a6ee83:/temp# exit
```

```
docker commit con1 kazimesen/con1:latest
```

```
docker commit -c 'CMD ["java", "uygulama"]' con1 kazimesen/con1:ikinci
```

```
kazm@ubuntu:~/Desktop/AdanZyeDocker/kisim5/bolum59$ mkdir image
kazm@ubuntu:~/Desktop/AdanZyeDocker/kisim5/bolum59$ cd image/
kazm@ubuntu:~/Desktop/AdanZyeDocker/kisim5/bolum59/image$ docker save kazimesen/con1:latest -o con1imaj.tar
kazm@ubuntu:~/Desktop/AdanZyeDocker/kisim5/bolum59/image$ ls -al
total 125972
drwxrwxr-x 2 kazm kazm      4096 Oct  5 21:46 .
drwxrwxr-x 3 kazm kazm      4096 Oct  5 21:44 ..
-rw------- 1 kazm kazm 128984576 Oct  5 21:46 con1imaj.tar
kazm@ubuntu:~/Desktop/AdanZyeDocker/kisim5/bolum59/image$ docker load -i con1imaj.tar
Loaded image: kazimesen/con1:latest
```

Docker Compose

Docker Compose, çoklu Docker uygulamalarını tanımlama ve çalıştırma aracıdır. Compose ile uygulamanın hizmetlerini yapılandırmak için bir YAML dosyası kullanılır.
Sonra, tek bir komut ile tüm hizmetler oluşturulur ve başlatılır.

Docker Swarm

Docker Swarm, Docker Engine'e entegre bir container orchestration çözümüdür. Bir Docker ana bilgisayar havuzunu tek bir sanal ana bilgisayara dönüştürür.

Raft Consensus Algoritması ve Manager Node

Swarm birden fazla manager node destekler ve bu sayede yüksek erişilebilirlik sağlar. Bir manager'da sorun olursa diğer manager devreye girerek işi yürütür.
Bu manager nodelardan yanlızca 1 tanesi lider olarak seçilidir ve tüm yönetim lider tarafından yapılır.Diğer manager nodelar pasif durumdadır.
Siz pasif manager node'lardan birine bir komut verip iş yapmasını isterseniz bu sadece proxy görevi görür ve komutu lider node'a iletir.
Ortamda birden fazla manager olduğu durumlarda bir adet lider seçilmelidir. Swarm bunu otomatik halleder ve bunun için Raft Consensus algortimasını kullanır.
Raft algoritması lider seçimi için kuralları belirler.
Örneğin ortamda 5 manager olan bir durumda lider node bir şekilde erişlemez hale gelirse belirli bir zaman sonra kalan 4 node kendi aralarında oylama yaparak bir lider belirler.
Artık swarm cluster'ın yönetimi bu yeni lider tarafından sağlanır.
Raft consensus algoritması makisimum (N-1)/2 sayıda replikanın devre dası kalmasını tolere eder.
Örneğin 5 swarm manager olarak kurgulanan bir yapıda 2 manager devre dışı kalırsa kalan 3 node kendi aralarında anlaşarak çalışmaya devam edecektir.
Fakat 3 node devre dışı kalırsa kalan 2 node çoğunluğu sağlayıp anlaşamayacakları için sorun çıkacak ve management altyapısı çalışmayacaktır.
Raft algoritmasının düzgün çalışabilmesi ve lider seçiminin sorunsuz olabilmesi için ortamın her zaman tek sayida manager node'la kurulmuş olması gerekir.
Bu nedenle Docker Swarm 1 veya 3 veya 5 veya 7 manager ile kurulmalıdır. 9 - 11 diye devam edebilir ama 7'den fazla manager daha fazla sorun çıkacaktır.

```
docker swarm init --advertise-addr 192.168.0.28
docker swarm join-token manager
docker swarm join --token SWMTKN-1-10y95ap3upwuv1qsg2d0cbu9w8sqbqwv9x80mebw2jy2i2g8s4-dqldhi45quun7ij4czg6m63ib 192.168.0.28:2377
docker swarm join-token worker
docker swarm join --token SWMTKN-1-10y95ap3upwuv1qsg2d0cbu9w8sqbqwv9x80mebw2jy2i2g8s4-9ofikt9i86uwosz4rod77feet 192.168.0.28:2377
docker service create --name test --replicas=5 -p 8080:80 nginx

[node1] (local) root@192.168.0.28 ~
$ docker service ps test
ID             NAME      IMAGE          NODE      DESIRED STATE   CURRENT STATE           ERROR     PORTS
iohqd9x8jn7b   test.1    nginx:latest   node4     Running         Running 2 minutes ago
jqlzglznjfwg   test.2    nginx:latest   node5     Running         Running 2 minutes ago
iaf1yqv4vnrq   test.3    nginx:latest   node1     Running         Running 2 minutes ago
p4rwwvlxzpza   test.4    nginx:latest   node2     Running         Running 2 minutes ago
u7052j1zevxd   test.5    nginx:latest   node3     Running         Running 2 minutes ago
```

Docker Service Mode:

Replicated: Oluşturmak istenen servisin kaç replica olacağı belirtilir. Swarm uygun olan node'larda o kadar replica oluşturur.
Global: Oluşturmak istenen servisin kaç replica olacağı belirtilmez. Swarm altındaki her node üstünde 1 replica oluşturur.

Overlay Network

Bir Swarm cluster oluşturulduğunda ingress adında bir overlay network otomatik olarak oluşturulur. Aksi belirtilmedikçe oluşturulan servisler bu overlay network'e bağlanır.
İstendiğinde aynen user defined bridge network'teki gibi user defined overlay network de yaratılabilir.
Overlay network'lerin temel yönetim katmanının haberleşme altyapısı encrypte'dır. Fakat buraya bağlanan container'ların birbirleriyle iletişimi varsayılan olarak encrypted değildir.
Overlay network yaratırken --opt encrypted ile trafiğin encrypted olması sağlanabilir. Fakat bu network trafiğini biraz yavaşlatır.
Aynı overlay network'teki servislerin container'ları birbirleriyle herhangi bir port kısıtlaması olmaksızın haberleşir ve sanki aynı network'te imiş gibi çalışırlar.
Swarm altında yaratılan servisler aynı overlay network üzerinde birbirlerine servis isimleriyle ulaşılabilir. Docker burada hem dns, hem de load balancing hizmeti sunmaktadır.
Overlay network üzerinde aynı zamanda port publish de edilebilir. Docker Swarm overlay network'lerde ingress routing mesh destekler.
Port publish edip Docker host üstünden o porta erişilirse, Docker o host üstünde o portun publish olduğu bir container bulunmasa bile bulunan bir host'a trafiği yönlendirir ve cevap verir.

```
docker network create -d overlay over-net
docker service create --name web --network over-net -p 8080:80 --replicas=3 ozgurozturknet/web
docker service create --name db --network over-net ozgurozturknet/fakedb
```

Update ve Rollback

```
docker node ls
docker network create -d overlay over-net
docker service create --name websrv --network over-net -p 8080:80 --replicas=10 ozgurozturknet/web:v1
docker service ls
docker service ps websrv
docker service update --detach --update-delay 5s --update-parallelism 2 --image ozgurozturknet/web:v2 websrv
watch docker service ps websrv
docker service rollback --detach websrv
watch docker service ps websrv
```

Docker Secret : Container'larda plain text olarak bulundurulması güvenlik zaafiyeti yaratabilecek kullanıcı adı ve şifre gibi bilgilerin encrypted olarak transferini sağlayan objedir.

```
docker swarm init
nano kullaniciadi.txt
nano sifre.txt
docker secret create kullanici_adi kullaniciadi.txt
docker secret create sifre sifre.txt
echo "Bu bir denemedir" | docker secret create deneme -
docker service create -d --name secretdeneme --secret kullanici_adi --secret sifre --secret deneme ozgurozturknet/web
docker exec -it 481 sh
echo "sadksahdkdk2" | docker secret create sifre2 -
docker service update --secret-rm sifre --secret-add sifre2 secretdeneme
```

Docker Stack

```
docker stack deploy -c docker-compose.yml ilkstack
docker stack ls
docker stack services ilkstack
docker service ps ilkstack_websrv

docker stack deploy -c docker-compose.yml wpstack
```