MikroServisler ve Docker

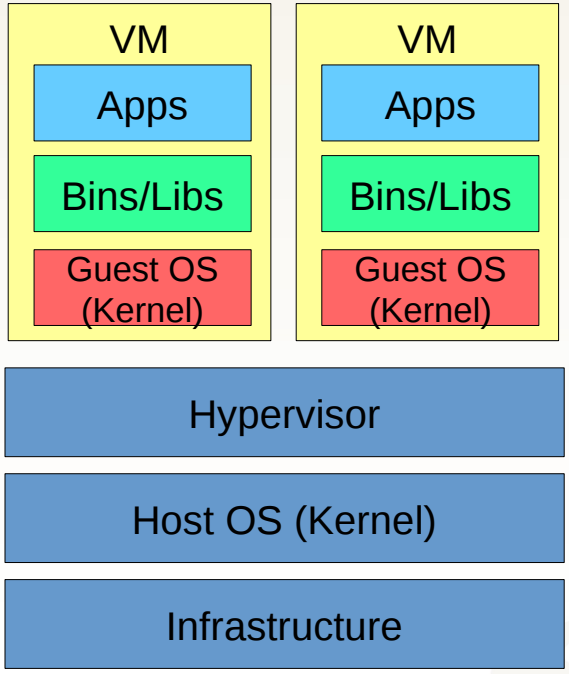
harezmi bilişim çözümleri

Docker (Engine) Nedir?

- Docker bir **sanallaştırma** (virtualization) uygulamasıdır
- Uygulamaların **sanal bir platform** üzerinde çalıştırılmasını sağlar
- Bu sanal platform herhangi **bir fiziksel makine** veya başka **bir sanal makine** üzerinde çalışabilir

Klasik Sanallaştırma (Virtualization)

- Bir **Host** işletim sistemi üzerinde **Hypervisor** yardımı ile bir **Guest** işletim sistemini çalıştırmayı sağlar



Sanal makineler birbirlerinden tamamen izoledirler

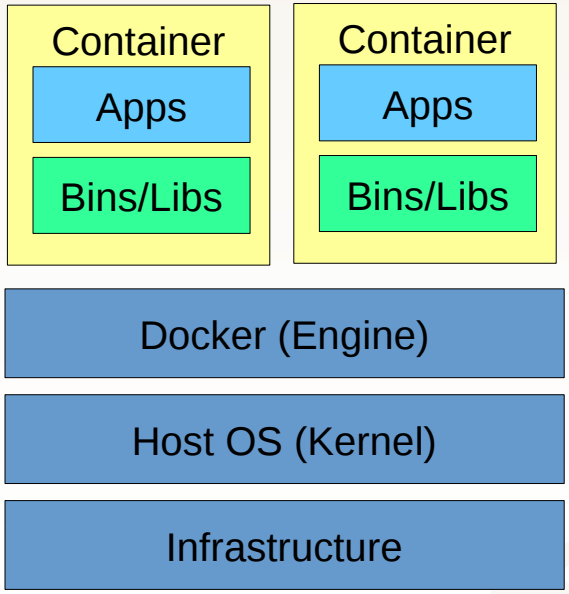
Her bir sanal makineye kendine ait bir disk ve hafıza alanı ayrılır

Her sanal makine'de ayrı guest İşletim sistemi çalışır

Uygulamalar da bu guest işletim sistemlerinde ayrı ayrı çalışırlar

Docker ile Sanallaştırma

- Uygulamaları çalıştırmak için **Linux Container** kullanılarak yapılan sanallaştırma işlemine **containerization** adı verilir



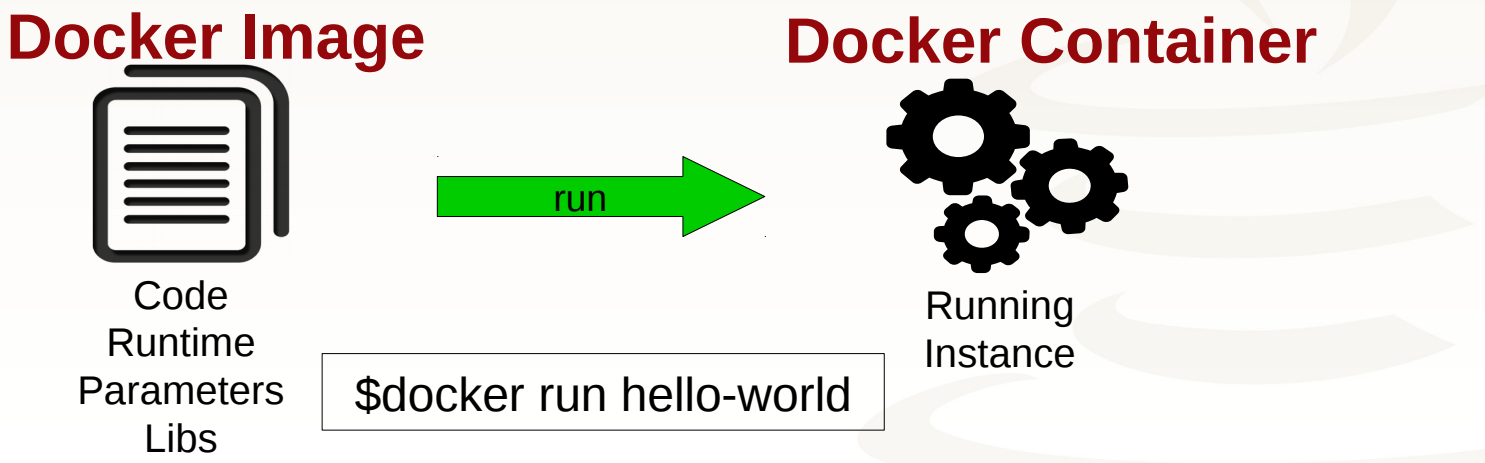
Container'lar aynı işletim sistemi üzerinde çalışırlar

Her bir container, host işletim sistemindeki bir process'e karşılık gelir

Container içerisinde guest OS olmaz

Temel Docker Kavramları

- **Image**: Uygulamayı sanal ortamda çalıştırmak için gerekli bütün herşeyi içeren **paket**'tir
- **Container**: Image'ın çalışma zamanındaki **instance**'ıdır



Temel Docker Kavramları

- **Repository**: Bir image'ın varyasyonlarını tutan yapıdır
- **Tag**: Bir image'ın her bir varyasyonuna verilen **isim** veya **etiket**tir

REPOSITORY	TAG	IMAGE ID
app1	latest	fb434121fc77
app2	latest	91c95931e552
app2	v1.1	91c95931e552
app2	v1.0	1234abcd5678

```
$docker run app2
$docker run app2:v1.0
```

Temel Docker Kavramları

- **Registry**: Üzerinde docker repository'leri barındıran **depo**'dur
  - Public veya private olabilir
  - **Docker Hub** herkese açık public repository'dir
  - http://hub.docker.com
- **Namespace**: Public registry içerisinde repository isimlerindeki çakışmayı önlemek için eklenen ön ektir
  - Username/repo:tag
  - Namespace/repo:tag

Temel Docker Kavramları

- **Volume**: Container'a ait **veriyi saklama**yı ve container'lar arasında **veri paylaşımı**nı sağlar
  - Host OS üzerinden **erişilebilir bir dizin yapısı**dır
  - Container'lardan bağımsız yaratılabilir, container'larla ilişkilendirilebilir

Docker Image Nasıl Oluşturulur?

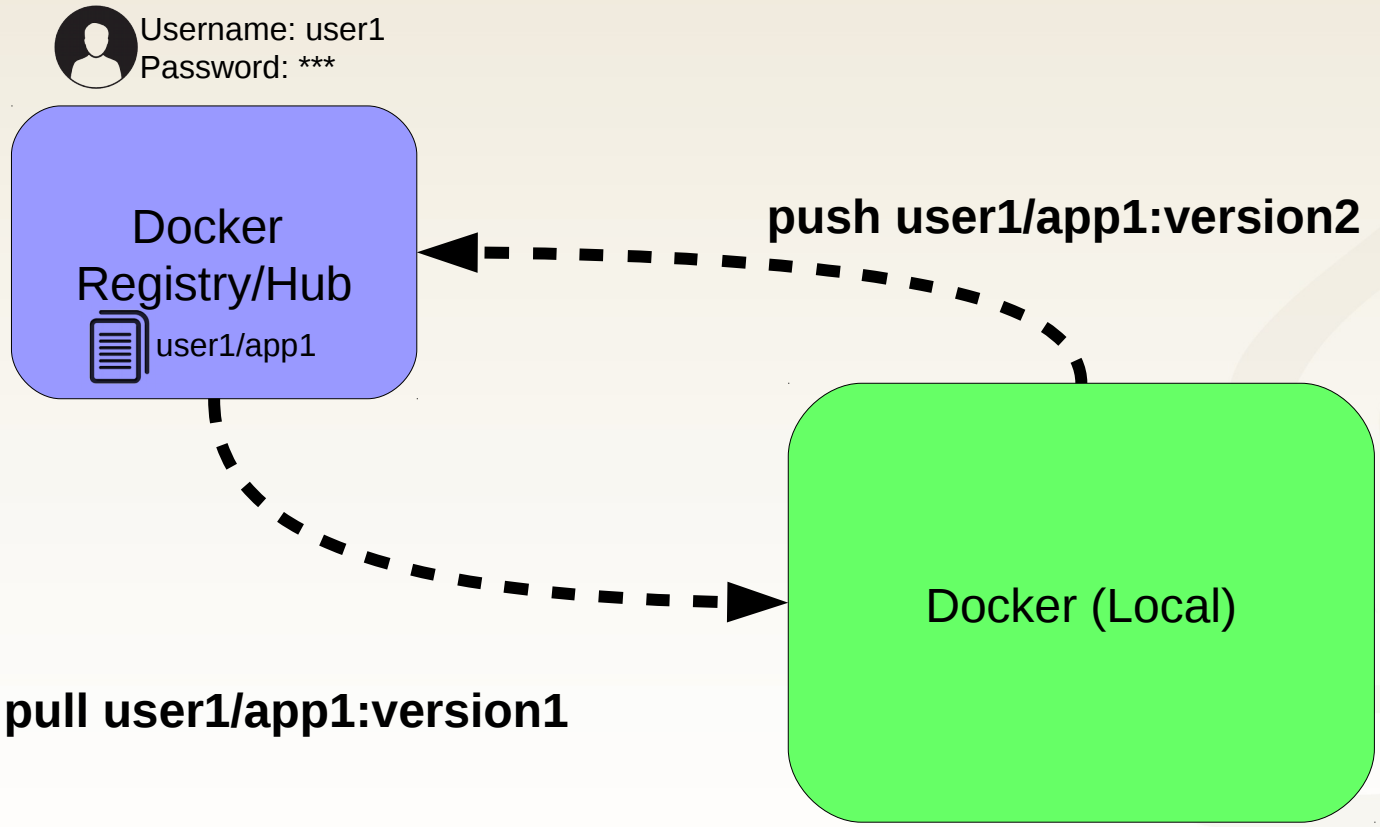
- Docker image'larını oluşturmak için **docker build <DIR>** komutu kullanılır
- Komut <DIR> dizini altında **Dockerfile** adında bir dosya arar
- Dockerfile dosyası **build esnasında işletilecek komutları** içerir

Dockerfile

```
FROM openjdk:8-jdk-alpine
COPY foo-service.jar app.jar
ARG SERVER_PORT=8080
ENV server.port=${SERVER_PORT}
EXPOSE ${SERVER_PORT}
ENTRYPOINT ["java","-jar","app.jar"]
```

```
$docker build -t myapp:v1 ./
$docker run -p 80:8081 myapp:v1
```

Pull/Push Image



Docker Compose

- Birden fazla docker container'a sahip uygulamaları **tanımlamak ve çalıştırmak** için geliştirilmiş bir araçtır
- Docker dışında **ayrıca kurulmalı**dır
- Bütün container'lar **tek bir host** üzerinde çalıştırılırlar
- Konfigürasyon tanımları için **YAML dosya formatı** kullanılır

Docker Compose

- Her bir servis **Dockerfile** ile tanımlanır
- Ardından bunlar **docker-compose.yml** ile bir araya getirilir

docker-compose.yml

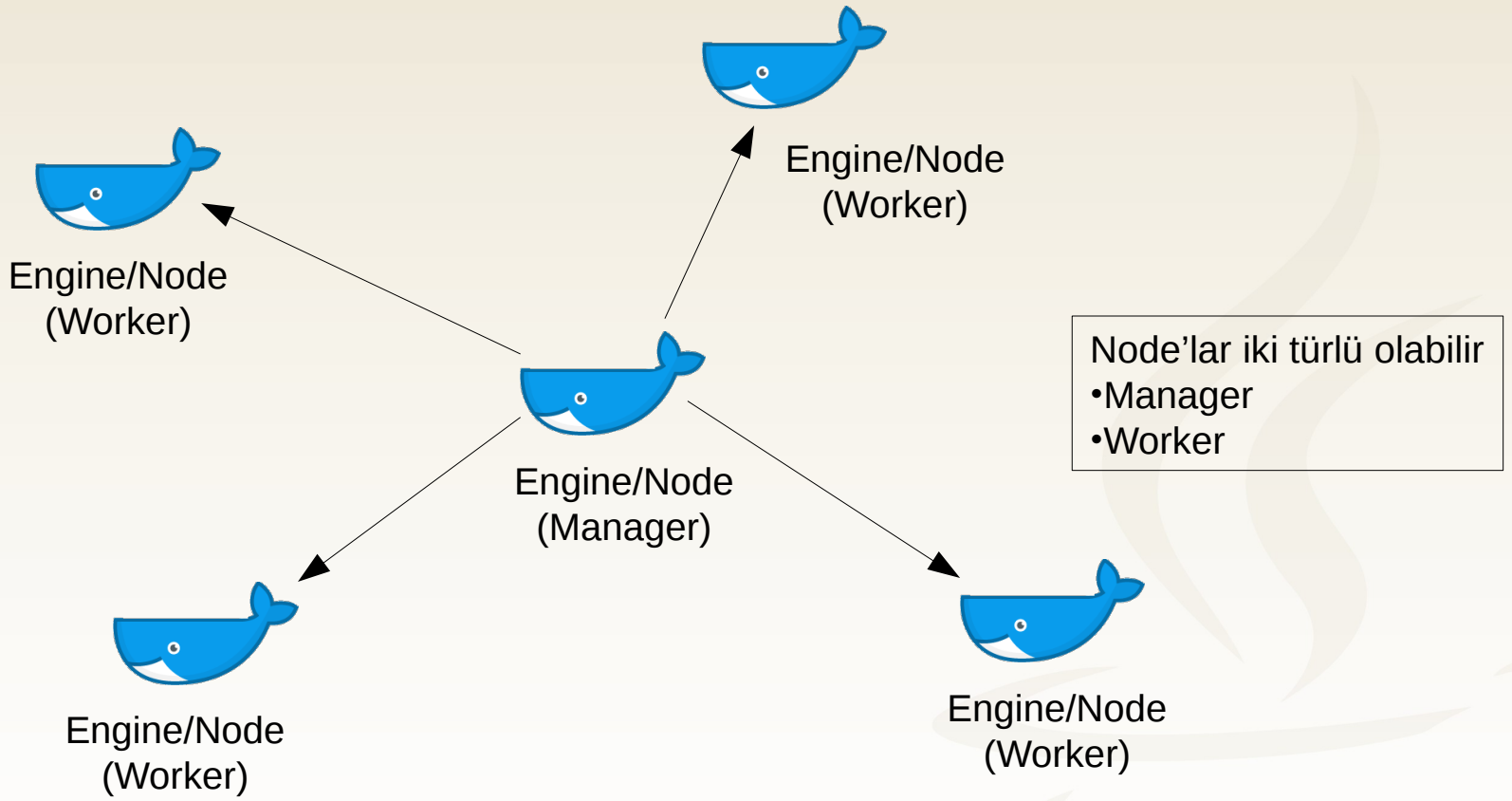
```
fooservice:
  image: fooservice:v1
  ports:
    - 8081:8081
barservice:
  image: barservice:v2
  ports:
    - 8082:8082
  environment:
    - VAR1=value
```

```
$docker-compose up -d
```

Docker Swarm

- **Swarm mode** dağıtık ortamda **bir grup docker engine**'i birbiri ile ilişkilendirip, yönetmeyi sağlar
- Gruptaki her bir **docker engine** ayrı bir **node** olarak kabul edilir
- Docker swarm **docker node'ları**n bir araya gelmesinden oluşan bir tür **cluster**'dır

Docker Swarm



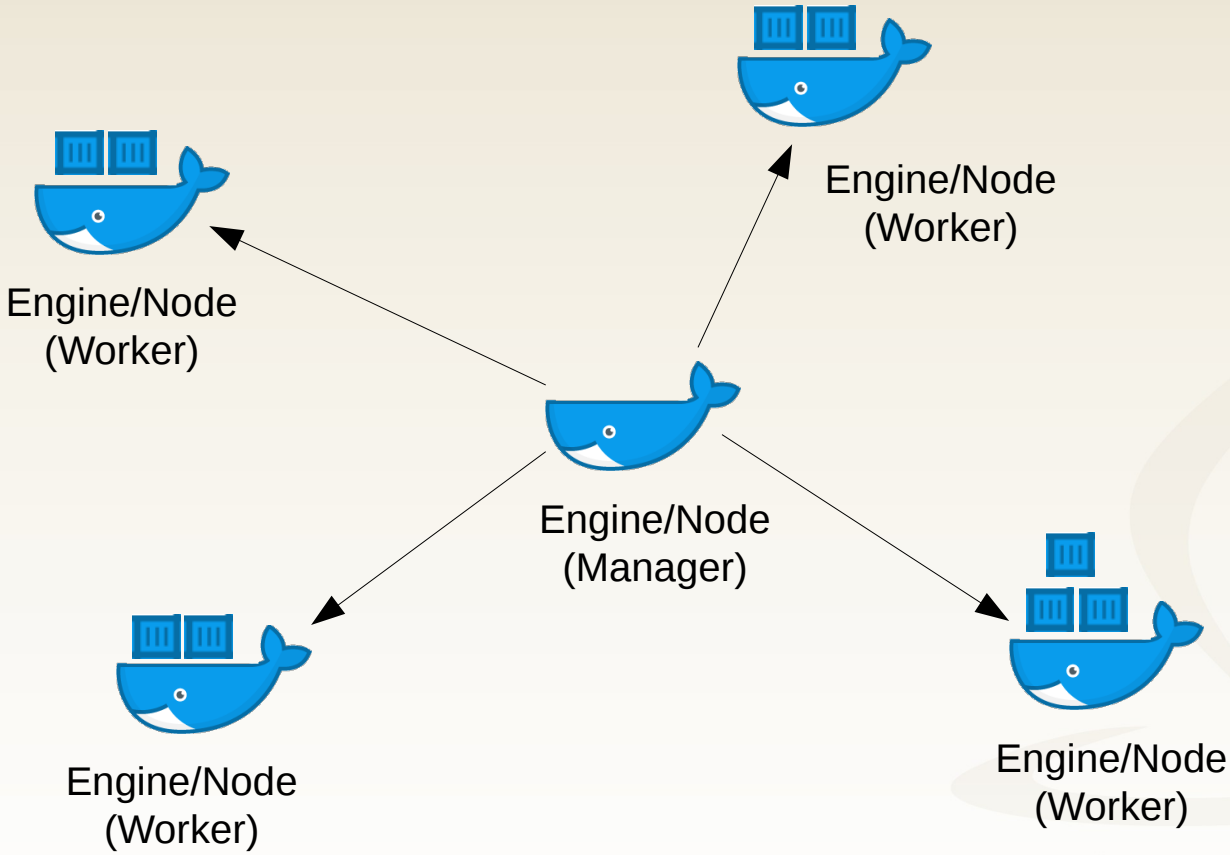
Node'lar iki türlü olabilir
- Manager
- Worker

```
$docker swarm init
$docker swarm join <manager node ip>:2377
```

Docker Service

- Dağıtık ortamda **bir grup docker container**'ın swarm node'lar üzerinde **çalıştırılması**nı ve **yönetilmesi**ni sağlar
- Docker swarm'daki container'lar **belirli bir image'dan** oluşurlar
- **Scale up/down** yapma imkanı sunar
- Docker service, swarm cluster'daki node'ları **instance düzeyinde yönetme**yi sağlar

Docker Service



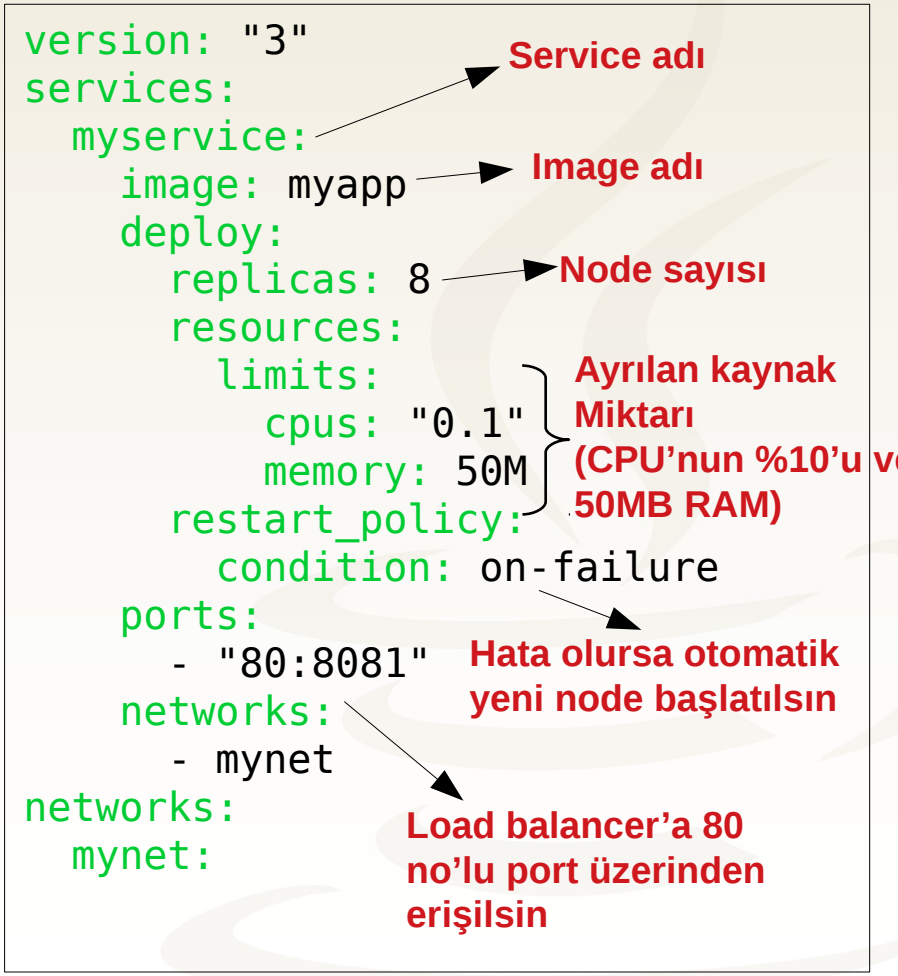

```
$docker service create –replicas 8 –name myservice –network mynet -p 80:8081 myapp:latest
$docker service update –replicas 9 myservice
```

Docker Stack

- Docker stack ise birden fazla docker service'i **topluca yönetmeyi** sağlar
- Konfigürasyon için **YAML** dosya formatı kullanılır

```
$docker stack deploy -c docker-stack.yml mystack
```

docker-stack.yml



İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com