




11 МАЙ 2024 Г.

# DOCKER, ИЗТЕГЛЯНЕ И СТАРТИРАНЕ НА DOCKER IMAGE

СИМЕОН МАРКОВ

Клас: 11А



## 1. Какво е Docker ?

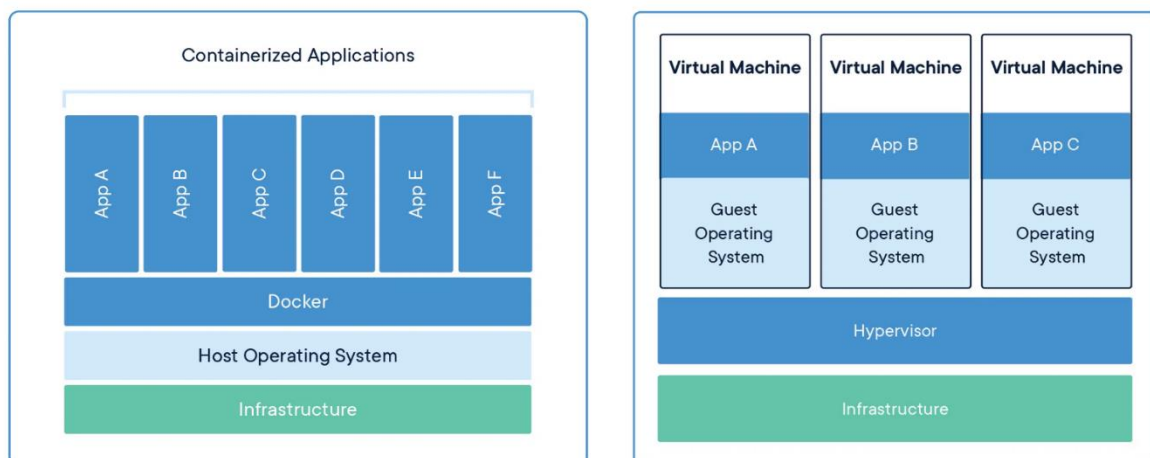
- **Определение:** Docker е платформа за контейнеризация с отворен код, чрез която може да се опакова приложение и всичките му зависимости в стандартизирана единица, наречена контейнер.
- Позволява на потребителите да „пакетират“ приложение с всичките му зависимости в стандартизирана единица (контейнер) за разработка на софтуер.
- За разлика от виртуалните машини, контейнерите нямат високи режимни разходи и по този начин позволяват по-ефективно използване на основната система и ресурси.
- Всеки **контейнер** съдържа всички елементи, необходими за изграждането на софтуерен компонент и гарантира, че той е изграден, тестван и внедрен гладко.
- Това позволява на разработчиците в екип едновременно да изграждат множество парчета софтуер.

## 2. Как работи Docker ?

- **Docker** използва изолиране на ресурсите в ядрото на операционната система, за да изпълнява множество контейнери на една и съща операционна система.
- **Docker** използва архитектура **клиент-сървър**.
- **Процес на работа:**
  - 1) За да има пуснат контейнер с приложението, първо трябва да се създаде т.нар. **Docker image**.
  - 2) За да създаде **image**, трябва да създаде файл с името **“Dockerfile”** (без разширение). То представлява прост текстов файл, в който се описват стъпките за създаването на image. След това се стартира **build** през Docker и image-а е създаден.
  - 3) Накрая се стартира Docker контейнер от вече създадения image. Контейнерът е **стартираната инстанция** на image-а.

### 3. Какво представляват контейнерите ?

- **Определение:** Контейнерът е стандартна единица софтуер, която пакетира кода и всичките му зависимости, така че приложението да работи бързо и надеждно от една изчислителна среда в друга.
- Позволява на разработчиците да пакетират приложения с всички необходими части, като библиотеки и други зависимости.
- Контейнерите съдържат целия комплект, необходим за приложение, така че приложението може да се изпълнява изолирано.
- **Предимства:**
  - Преносимост: Контейнерите са проектирани да бъдат платформено независими.
  - Ефективност: Споделят операционната система на хост системата, което намалява режимните разходи за работа на виртуална машина с множество операционни системи.
  - Консистенция: Пакетират всички необходими компоненти, включително кода на приложението, времето на изпълнение, библиотеките и зависимостите, в едно устройство.
  - Изолация: Осигуряват лека и изолирана среда за работещи приложения.
  - Бързо внедряване: Могат да бъдат създадени и стартирани бързо, често отнема само секунди.
- **Сравняване на контейнери и виртуални машини:** Контейнерите и виртуалните машини имат сходни ползи за изолиране и разпределение на ресурсите, но функционират по различен начин, защото **контейнерите виртуализират операционната система** вместо хардуера. Контейнерите са по-преносими и ефективни.



## 4. Какво представляват Docker images ?

- **Определение:** Файл, състоящ се от няколко слоя, използван за изпълнение на код в контейнер на Docker.
- Изображенията са шаблони само за четене, съдържащи инструкции за създаване на контейнер. Изображението на Docker създава контейнери, които да се изпълняват на платформата Docker.
- Може ръчно да се създават с помощта на **Dockerfile**.
- Информират как контейнерът трябва да инсталира, определяйки кои софтуерни компоненти ще работят и как.

## 5. Какво е DockerFile ?

- **Определение:** Dockerfiles са текстови файлове, които изброяват инструкциите, които **Docker daemon** трябва да следва, когато изгражда изображение на контейнер.
- Когато се изпълнява командата **docker build**, редовете в **Dockerfile** се обработват последователно, за да сглоби image.
- **Синтаксис на някои от инструкциите:**
  - Коментари: Започват с „#“:

```
# Comment  
INSTRUCTION arguments
```

- Ключова дума **FROM**: Създайте нов етап на изграждане от базово изображение (image):

```
FROM ubuntu:20.04
```

- Ключова дума **COPY**: Добавя файлове и папки към файловата система на image:

```
COPY main.js /app/main.js
```

- Ключова дума **ADD**: Добавяне на локални или отдалечени файлове и директории.

```
ADD http://example.com/archive.tar /archive-content
```

- Ключова дума **RUN**: Изпълнява команда вътре в изображението (image), което изграждате. Той създава нов слой image върху предишния:

```
RUN apt-get update && apt-get install -y nodejs
```

- Ключова дума **ENV**: Задаване на променливи на средата.

```
ENV PATH=$PATH:/app/bin
```

## 6. Какво е Docker Daemon ?

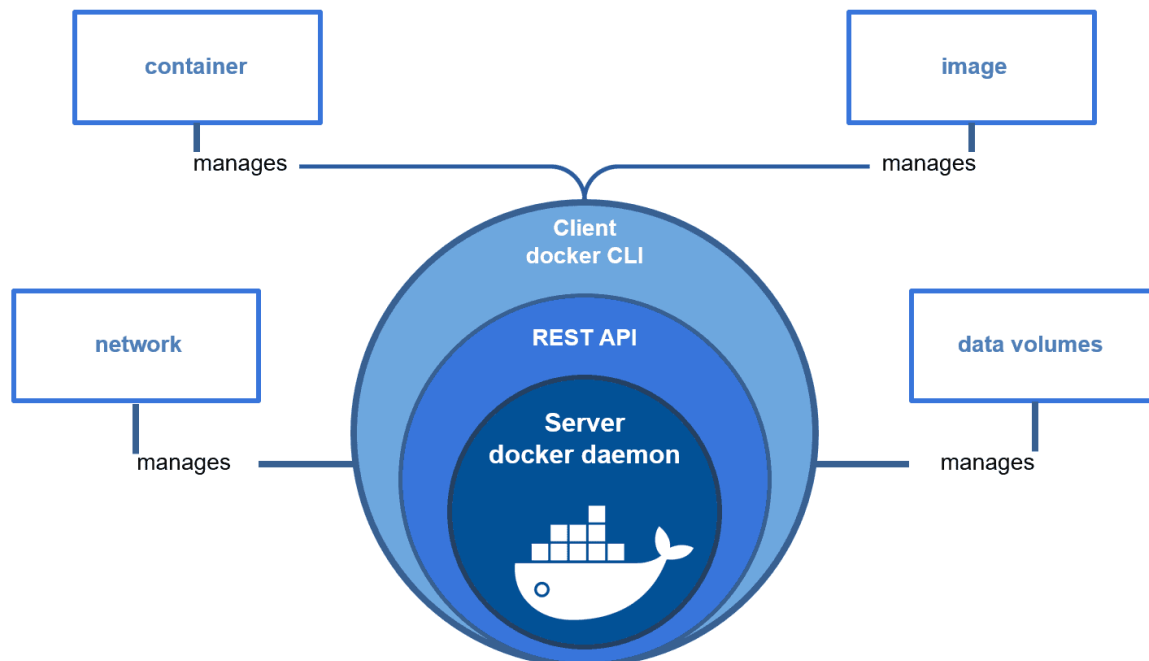
- **Определение:** Dockerd (Daemon) е постоянният процес, който управлява контейнерите.
- Важната функция на Docker Daemon е да наблюдава управлението на жизнения цикъл на контейнерите, обхващайки задачи за създаване, изпълнение и мониторинг.
- Той действа като централен процесор, работещ в тясно сътрудничество с основната операционна система за осигуряване и контрол на контейнерите.
- **Docker CLI** действа като удобен за потребителя интерфейс, който преобразува командите в API повиквания, които daemon-а може да разбере.

## 7. Какво е Docker Hub ?

- **Определение:** Docker Hub е услуга за хранилище и е услуга, базирана на облак, където хората натискат своите Docker Container Images и също така изтеглят Docker Container Images от Docker Hub по всяко време или навсякъде чрез интернет.
- Той предоставя функции, като например може да push-не изображения (images) като частен или публичен регистър, където може да се съхраняват и споделят изображения (images) на Docker.
- Като цяло това улеснява намирането и повторното използване на изображения (images).

## 8. Архитектура на Docker

- Docker има “клиент-сървър” архитектура. Клиентът е Docker CLI, а сървърът се нарича Docker Daemon.
- Този Даемон получава инструкции от CLI под формата на команди в конзолата или REST API заявки.
- **Docker Engine:** съдържа CLI и Daemon.
- Docker клиентът и сървърът могат да са на една машина, но могат да бъдат и на различни такива.



- **Други компоненти и инструменти в Docker архитектурата включват:**
  - Надежден регистър (Trusted Registry): Това е хранилище, което е подобно на Docker Hub, но с допълнителен слой контрол и собственост върху съхранението и разпространението на изображения на контейнери.
  - Docker Swarm: Това е част от Docker Engine, който поддържа балансиране на клъстерното натоварване за Docker.
  - Universal Control Plane: Това е уеб-базиран, унифициран интерфейс за управление на клъстери и приложения.
  - Compose: Този инструмент се използва за конфигуриране на услуги за приложения с множество контейнери, преглед на състоянието на контейнерите, изход на регистрационния файл на потока и изпълнение на едноинстанционни процеси.
  - Content Trust: Този инструмент за сигурност се използва за проверка на целостта на отдалечените регистри на Docker чрез потребителски подписи и маркери за изображения.

## 9. Запознаване с Docker Hub

- 1) Регистрирайте се за безплатен акаунт в Docker: <https://hub.docker.com/signup>
- 2) Създайте първото си хранилище:
  - Влезте в Docker Hub.
  - На страницата Хранилища изберете **Създаване на хранилище**.
  - Наименувайте хранилището.
  - Задайте ниво на видимост.
  - Изберете **Създаване**
- 3) Изтегляне и инсталиране на Docker Desktop:
  - Download and install Docker Desktop.
  - Влезте в Docker Desktop, като използвате Docker ID, който създадохте в стъпка едно.
- 4) Издърпайте и стартирайте изображение (Image) на контейнер от Docker Hub:
  - Във вашия терминал, стартирайте **docker pull hello-world**, за да дръпнете изображението от Docker Hub. Трябва да видите изход, подобен на:
  - Стартирайте **docker run hello-world**, за да стартирате изображението локално. Трябва да видите изход, подобен на:

```
docker pull hello-world

Using default tag: latest

latest: Pulling from library/hello-world

2db29710123e: Pull complete

Digest:
sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce6333
1ac66dca625

Status: Downloaded newer image for hello-world:latest

docker.io/library/hello-world:latest
```

- Стартирайте **docker run hello-world**, за да стартирате изображението локално. Трябва да видите изход, подобен на:

```
$ docker run hello-world

Hello from Docker!

This message shows that your installation appears to be
working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.

2. The Docker daemon pulled the "hello-world" image from
the Docker Hub.

   (amd64)

3. The Docker daemon created a new container from that
image which runs the

   executable that produces the output you are currently
   reading.

4. The Docker daemon streamed that output to the Docker
client, which sent
```



5) Изграждане и изпращане на изображение на контейнер към Docker Hub от вашия компютър:

- Започнете, като създадете Dockerfile, за да посочите приложението си:

```
# syntax=docker/dockerfile:1

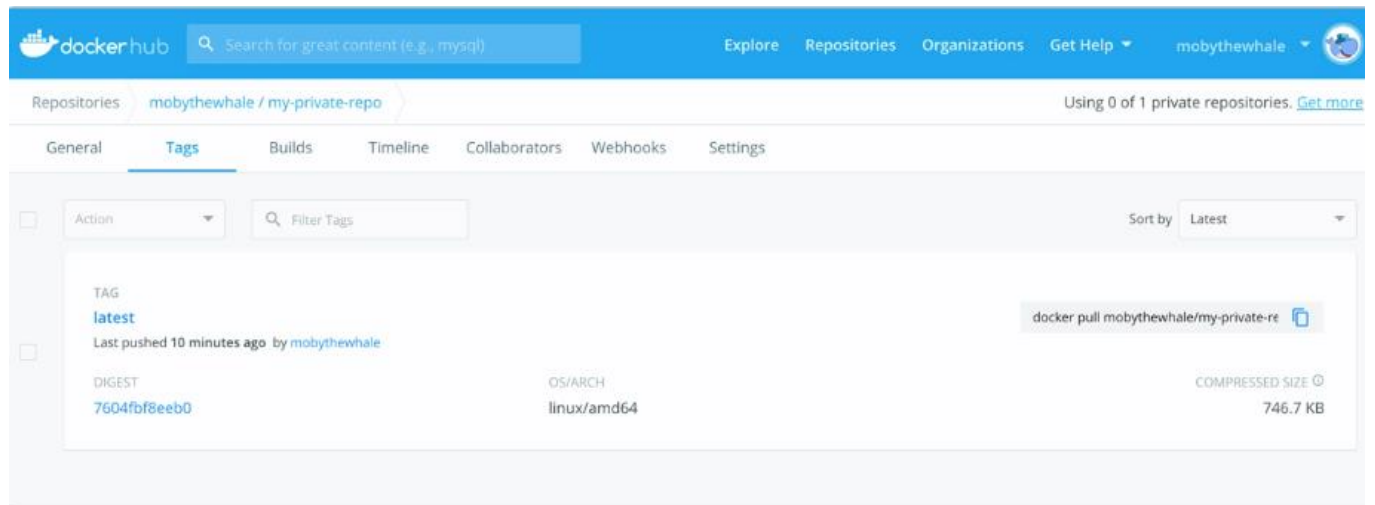
FROM busybox

CMD echo "Hello world! This is my first Docker image."
```

- Стартирайте `docker build -t <your_username>/my-private-repo .` за да изградите своя Docker образ (image).
- Стартирайте `docker run <your_username>/my-private-repo`, за да тествате изображението на Docker локално.
- Стартирайте `docker push <your_username>/my-private-repo`, за да прокарате изображението на Docker в Docker Hub:

```
$ cat > Dockerfile <<EOF
FROM busybox
CMD echo "Hello world! This is my first Docker image."
EOF
$ docker build -t mobythewhale/my-private-repo .
[+] Building 1.2s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 110B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/busybox:latest
=> CACHED [1/1] FROM docker.io/library/busybox@sha256:a9286defaba7n3a519
=> exporting to image
=> => exporting layers
=> => writing image sha256:dcdb1fd928bf257bfc0122ea47accd911a3a386ce618
=> => naming to docker.io/mobythewhale/my-private-repo
$ docker run mobythewhale/my-private-repo
Hello world! This is my first Docker image.
$ docker push mobythewhale/my-private-repo
The push refers to repository [docker.io/mobythewhale/my-private-repo]
d2421964bad1: Layer already exists
latest: digest: sha256:7604fbf8eeb03d866fd005fa95cddb802274bf9fa51f7dafba6658294efa9baa size: 526
```

- Вашето хранилище в Docker Hub сега трябва да показва нов последен маркер под Tags:



## 10. Предимства и недостатъци на Docker

- **Предимства:**

- Висока степен на преносимост, така че потребителите да могат да се регистрират и споделят контейнери през различни хостове.
- По-ниско използване на ресурсите.
- По-бързо внедряване в сравнение с виртуалните машини.
- Docker редовно добавя подобрения в сигурността на платформата Docker, като сканиране на изображения, сигурно въвеждане на възли, идентичност на криптографски възли, сегментиране на клъстери и сигурно тайно разпространение.

- **Недостатъци:**

- Броят на контейнерите, които са възможни в едно предприятие, може да бъде труден за ефективно управление.
- Използването на контейнери се развива от гранулиран виртуален хостинг до оркестрация на компонентите и ресурсите на приложението. В резултат на това разпространението и взаимното свързване на компонентизирани приложения - които могат да включват стотици ефимерни контейнери - се превръща в основна пречка.

## 11. Източници

- <https://www.geeksforgeeks.org/introduction-to-docker/>
- <https://www.techtarget.com/searchitoperations/definition/Docker>
- <https://www.docker.com/resources/what-container/>
- <https://docker-curriculum.com/>
- <https://docs.docker.com/get-started/overview/>
- <https://www.freecodecamp.org/news/how-docker-containers-work/>
- <https://circleci.com/blog/docker-image-vs-container/>
- <https://spacelift.io/blog/dockerfile>
- <https://docs.docker.com/reference/dockerfile/>
- <https://intellipaat.com/blog/docker-daemon/>
- <https://docs.docker.com/reference/cli/dockerd/>
- <https://docs.docker.com/docker-hub/quickstart/>