|  |  |
| --- | --- |
|  | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**  **ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И УПРАВЛЕНИЕ** |

Проект

по Операционни системи

„**Docker**“

**Изготвили: Проверил**:

Теодор Миленов, 121212186 Ас. Боян Петров

Найден Герасимов,121212232

ФКСУ, 4-ти курс, гр. 54

Гр. София , 16.09.2016

Съдържание

[Увод 3](#_Toc461753444)

[Какво представлява Docker 3](#_Toc461753445)

[Изграждане(build) 4](#_Toc461753446)

[Доставяне(ship) 4](#_Toc461753447)

[Стартиране(run) 4](#_Toc461753448)

[Docker архитектура 4](#_Toc461753449)

[Компоненти на Docker платформата 5](#_Toc461753450)

[Изображения(images) 5](#_Toc461753451)

[Регистри 5](#_Toc461753452)

[Контейнери 6](#_Toc461753453)

[Инсталация на Docker 6](#_Toc461753454)

[Windows 6](#_Toc461753455)

[Linux 6](#_Toc461753456)

[Docker-node-app 7](#_Toc461753457)

[Структура на проекта 7](#_Toc461753458)

[docker-compose.yml 7](#_Toc461753459)

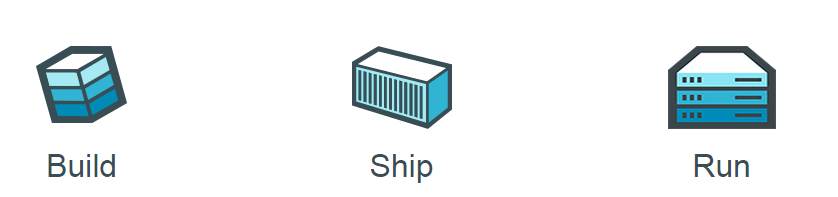
[Команди за управление на приложението 7](#_Toc461753460)

# Увод

Винаги при разработката на мащабни уеб приложения, проблемът с инфраструктурата нужна за изграждането на жизнения им цикъл и поддържането на гъвкав и надежден такъв е бил налице. Леснотата при разработката и доставянето на нови версии на една комплексна система е била под въпрос именно заради голямата сложност, свързана с изграждане, доставяне и стартиране на уеб приложения. Затова и голяма част от вниманието на разработчици и системни администратори е приковано върху тези три компонента от жизнения цикъл на едно софтуерно решение. Това е довело и до появата на така нареченият Docker, като оптимално обединяване на изграждане, доставяне и стартиране на уеб приложения, под контрола на една лесна за използване функционалност.

# Какво представлява Docker

Docker е платформа с отворен код, изградена на базата на контейнер технологията, която позволява изграждане, доставяне и стартиране на разнообразни приложения, отделяйки ги от инфраструктурата. Тази изолирана среда дава възможност на потребителите да работят едновременно с много контейнери на даден сървър. С Docker може да се използват максимално ресурсите на сървърна машина, заради олекотеният му принцип на работа и липсата на сложна конфигурация и настройка на средата за разработка. Важно е да се отбележи, че въпреки че Docker технологията много прилича на вид виртуализация, коренно се различава от такава. Именно тази разлика дава предимство на Docker пред виртуалните машини – той е много по-бърз и изисква по малко машинни ресурси.



## Изграждане(build)

С помощта на Docker контейнерите, изграждането на приложения става изключително лесно. Имайки предвид, че всеки Docker контейнер съществува в своя собствена изолирана среда, нуждата от обща среда за тестване и разработка отпада. Използвайки Docker разработчиците и QA-ите работещи по дадено приложение могат напълно свободно да го изграждат без да се намесват в задълженията на своите колеги.

## Доставяне(ship)

Управлението на жизнения цикъл на едно приложение е изключително лесно с помощта на Doker. Тази иновативна технология позволява проекиране на целия цикъл на разработване, тестване и дистрибуция на приложението. Пакетирайки всяко приложение на едно място заедно с неговите зависими инфраструктурни конфигурации, се осигурява бързо и независимо от платформа дистрибутиране.

## Стартиране(run)

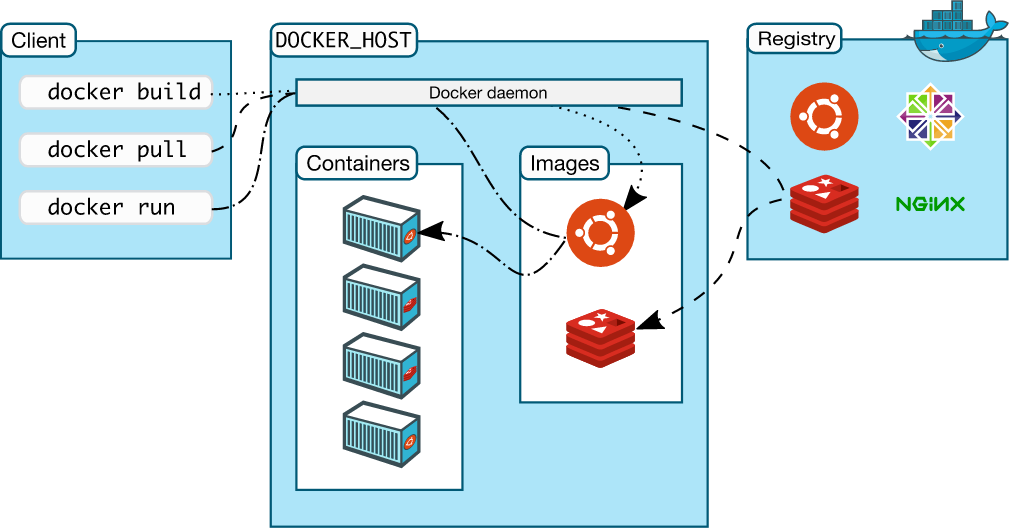
Тъй като всеки Docker контейнер може да бъде стартиран в каквато и да е Unix среда, стартирането на приложение, което е вградено в контейнер става много бързо и независимо от средата на разработка. Времето нужно за смяна на версия на дадено приложение с нова такава се свежда до минимум. Това подпомага както по-бързата разработка на приложения, така и по надеждното им управление в production среди.

# Docker архитектура

Докер използва клиент-сървър архитектура. Клиентът комуникира с daemon, който поема тежките работни процеси при изграждането, стартирането и дистрибуцията на Docker контейнерите. И клиентът и daemon могат да работят на една и съща система, а също могат да се свържат отдалечено един към друг. Комуникацията се осъществява посредством socket или RESTful API.

Docker клиентът представлява първичен потребителски интерфейс. Приема зададените от потребителите команди и постоянно комуникира с Docker daemon.

Docker daemon е процесът, който се изпълнява на сървърната машина. Директна комуникация с него няма.



## Компоненти на Docker платформата

### Изображения(images)

Изображенията представляват read-only темплейти, които описват поведението на един контейнер. Изображенията се използват също така и за създаването на самите контейнери. Docker предоставя опростен начин за създаване на нови изображения или за обновяване на съществуващи такива. Грижи се също и за това да има възможност за сваляне на вече готови изображения(създадени от някой друг). Изображенията са изграждащият компонент на Docker.

### Регистри

Регистрите се грижат за съхранението на изображенията. Това са публични или лични регистри, от които могат да се теглят или качват изображения. Официалният публичен регистър се нарича Docker Hub. Той предоставя огромни колекции от вече съществуващи изображения, които могат да се използват. Регистрите са дистрибуционният компонент на Docker.

### Контейнери

Docker контейнерите предоставят на разработчиците всички необходими инструменти за изграждането, тестването и стартирането на приложения. Контейнерите много приличат на директориите. Те съхраняват всичко, което е необходимо за изграждането и стартирането на едно качествено приложение. По-горе е споменато, че контейнерите се изграждат с помощта на изображенията. Docker Контейнерите с лекота могат да бъдат управлявани, стартирани, спирани, преместени и дори изтрити. Също така, всеки един контейнер е изолирана и безопасна платформа на приложение. Docker контейнерите са стартиращия компонент на Docker.

Чрез комбиниране на олекотена контейнер виртуализираща платформа снабдена с работни инструменти и процеси, дадени приложения могат да се управляват и разгръщат безпроблемно.

В контейнерите се „пакетират“ парченца софтуер в изолирана файлова система, която съдържа всичко необходимo, за да се стартира: код, runtime, системни настройки, системни библиотеки – всичко, което може да се инсталира на даден сървър. Това гарантира сигурно стартиране на приложението, без изменения и без значение от средата.

# Инсталация на Docker

## Windows

Същността на функционалността, която Docker предлага е в ядрото на Unix операционните системи. Затова и инсталацията на Docker под Windows е по-различна и не толкова проста като тази под Unix система.

За самата инсталация е нужен инсталационен пакет, който може да се свали безплатно от официалния сайт на Docker(<https://www.docker.com/>). След приключване на инсталацията се подготвя виртуална машина, която се стартира с помощта на VirtualBox(който също трябва да бъде инсталиран на Windows машината). Тази виртуална машина е напълно конфигурируема и служи за хост, на който да бъде инсталиран Docker daemon. Заедно с инсталацията на виртуалната машина, се инсталират и приложенията Docker client и docker-compose.

## Linux

Начините за инсталация на Docker на Linux хост машина са различни за различните дистрибуции на операционната система. За всяка една може да се намери подробна информация отново в сайта на Docker. Принципът обаче е един и същ за всички дистрибуции – сваля се инсталационен пакет, който се грижи за всичката конфигурация свързана с Docker engine. След успешна инсталация трябва да се стартира Docker daemon процеса.

# Docker-node-app

С цел демонстрация на работата на Docker и docker-compose е разработено малко Node.js приложение, което използва MySQL база данни. Приложението отваря http сървър, чрез който може да се записват изображения в различни формати на файловата система на хост машината. С помощта на Docker е приложено контейнеризиране на отделните компоненти на приложението, а именно сървърният код и базата данни. За по-лесно управление на жизнения цикъл на docker-node-app е използван docker-compose. Това, което е постигнато с docker-compose е че цялото приложение може да бъде изграждано, стартирано и спирано само чрез една команда.

## Структура на проекта

*db/Dockerfile –* съдържа темплейта, по-който се изгражда изображението за контейнера за базата данни.

*image-service/\* -* всички файлове необходими за изграждането на Node.js приложението и неговото изображение.

*docker-compose.yml –* конфигурационен файл съдържащ настройки за docker-compose

## docker-compose.yml

В този файл се съдържат всички настройки на docker-compose конфигурацията. Ето е описание на всяка една от тях:

* services: [<service\_name1>,<service\_name2>] - задава масив от сървиси, които представляват набор от контейнери вдигнати в една и съща виртуална мрежа.
* build – оказва пътя до Dockerfile-а нужен за изграждането на изображение за сървиса
* restart - задаване на желано поведение относно рестартиране на контейнер при спиране на сървис
* container\_name – име на сървиса. Това име ще бъде използвано и за hostname, през който останалите сървиси ще достъпват ресурси от контейнера посредством виртуалната мрежа.
* depends\_on – чрез това свойство се оказва синхронизация между стартиране на отделните сървиси.
* ports – портове, които да бъдат отворени от хост машината и пренасочени към портове на сървиса.

## Команди за управление на приложението

1. Изграждане – *docker-compose build*
2. Стартиране – *docker-compose up*
3. Стартиране без пренасочване на Output потоците на отделните сървиси към текущия – *docker-compose up –d*
4. Спиране – *docker-compose stop*
5. Проверка на текущо пуснатите контейнери – *docker ps*
6. Проверка на всички изградени контейнери – *docker ps –a*
7. Изобразяване на всички изображения – *docker images*
8. Изтриване на контейнери – *docker rm <container>*
9. Изтриване на изображения – *docker rmi <image>*