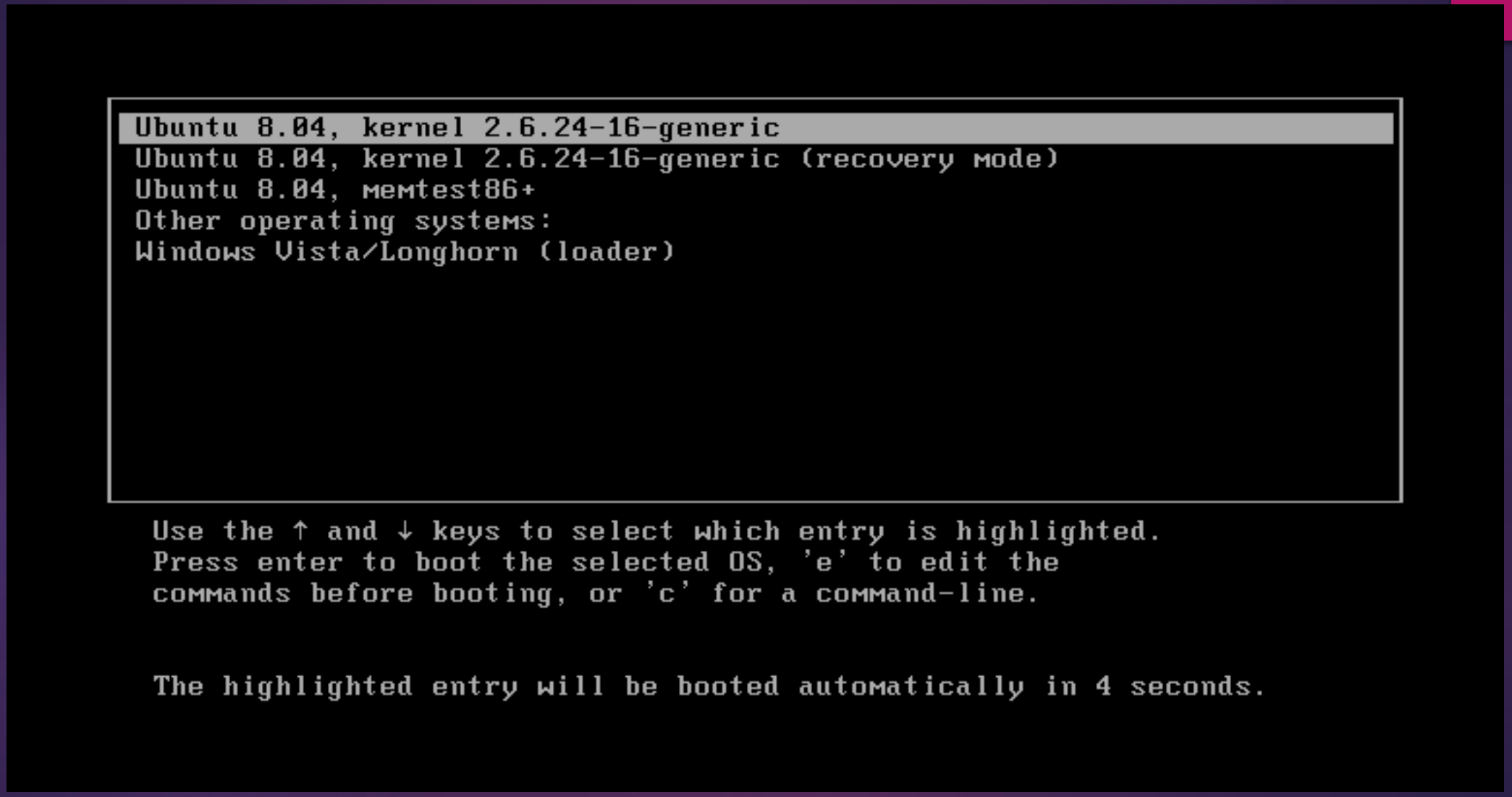
Контейнери, Docker технология

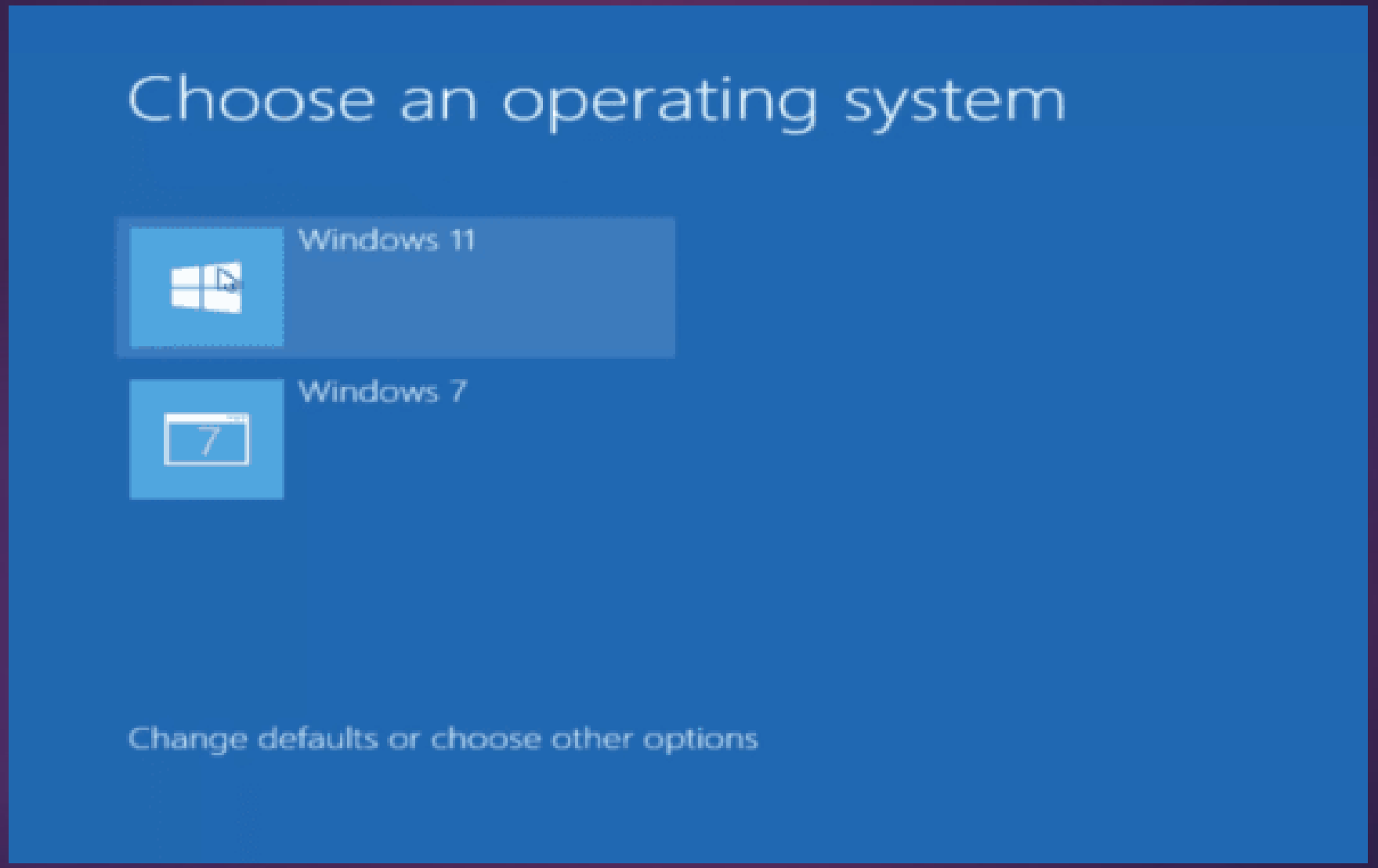
Необходимост от виртуализация ?

Как да стартираме повече от една ОС ?

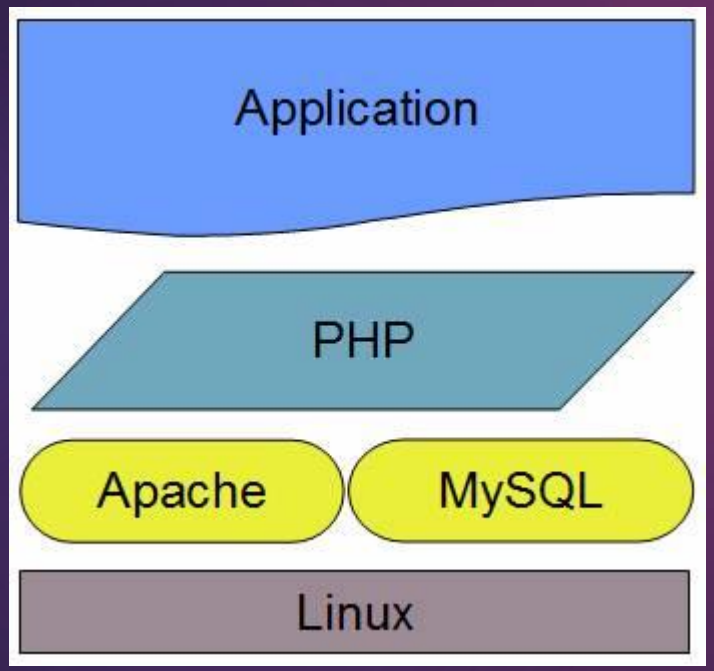
- **Стандартна инсталация (различни ОС, различни версии на ОС)**
- **Виртуализация. Виртуални машини**
- **Контейнери**

- **Въведение в Докер**


Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+
Other operating systems:
Windows Vista/Longhorn (loader)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
The highlighted entry will be booted automatically in 4 seconds.


Choose an operating system
Windows 11
Windows 7
Change defaults or choose other options

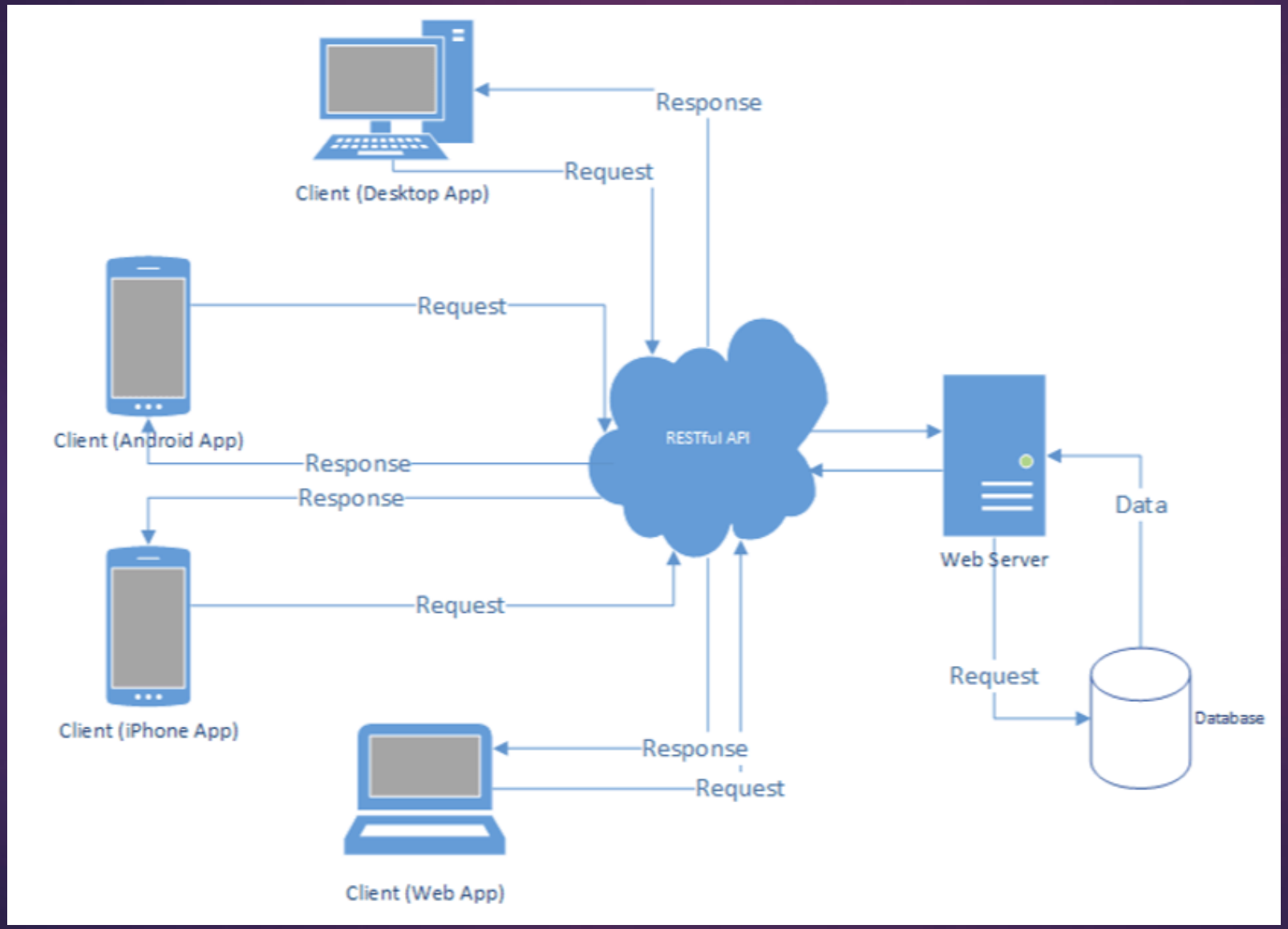
Класическа архитектура на използвани web приложения



1. Application
2. ASP.NET
3. IIS
4. SQL Server
5. Windows Server

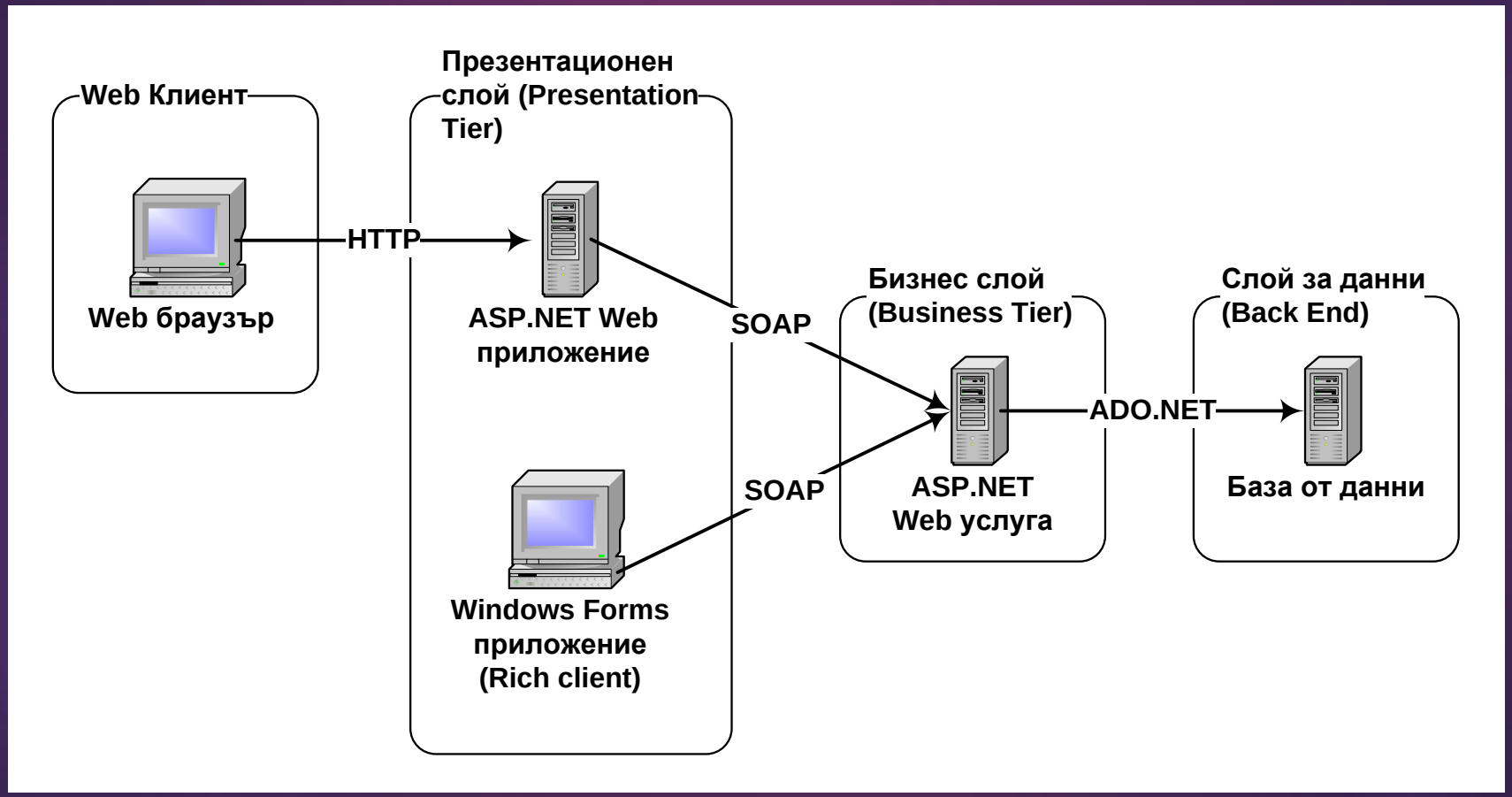
1. Java
2. Oracle
3. …….

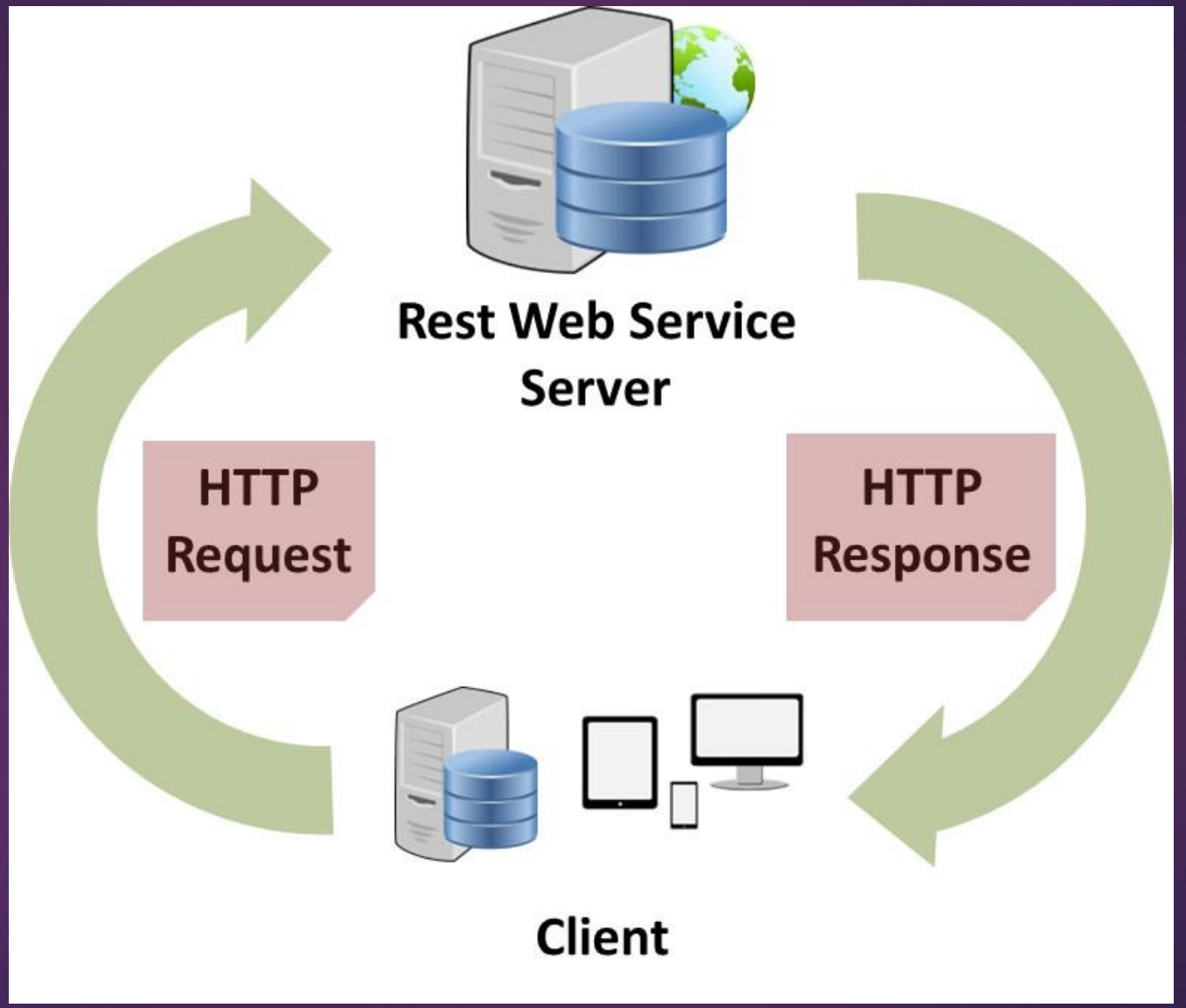
Web приложения – основни принципи



.NET Enterprise приложения

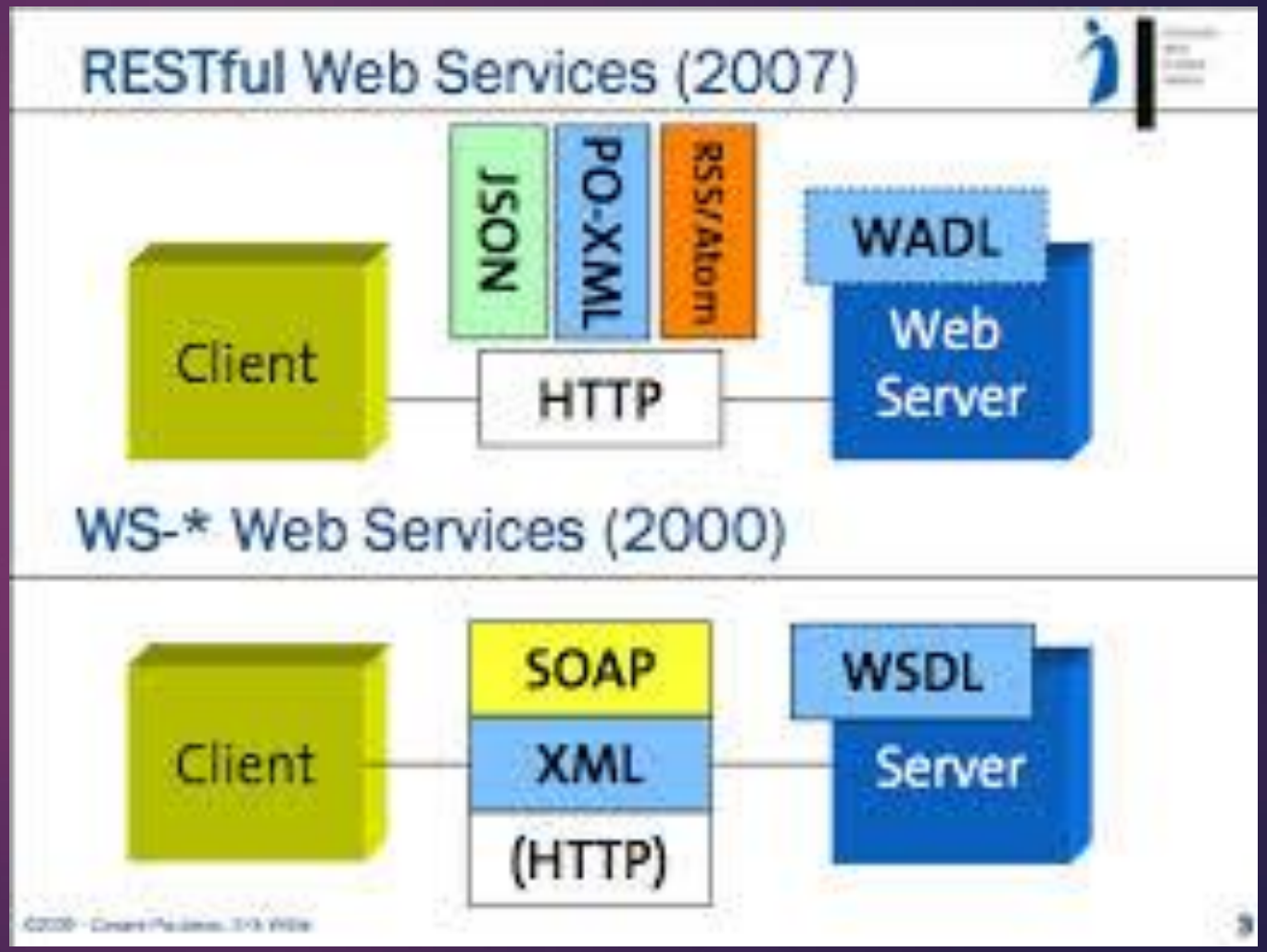
- Класическата Enterprise архитектура е трислойна:




Rest Web Service
Server
HTTP
Request
HTTP
Response
Client

REST услуги – **Re**presentational **S**tate **T**ransfer

- всяка заявка от клиента към сървъра трябва да съдържа всичко, което и е необходимо
- не трябва да се разчита на състояние съхранявано на сървъра
- против използването на cookie-та и сървърни сесии
- подобрява стабилността и scalability-то на приложението (по-малко натоварване на сървъра)
- не се разчита на определен протокол (няма WSDL и не би трябвало да зависи от HTTP)
- ясно дефиниран интерфейс, например HTTP методите GET, PUT, POST, DELETE
- само-описателни съобщения (чрез mime type, `application/json`)



Процеси в използваната класическа платформа – използване на една ОС

Процес 1 – Web Server

Процес 2 – DB Server

Процес 3 – Web Приложение, Услуги и т.н.

Всяко приложение – собствен процес

Проблеми при класическа архитектура

1. **Ограничения в ОС – само една;**
2. **Ограничения в средата – само съвместими с дадената ОС;**
3. **Разхищение на хардуер ;**
4. **Несъвместимост на версии;**
5. **„Война„ м/у разработчици и администратори ☺ ;**

РЕШЕНИЕ

1. Виртуализация ;

2.Технология използваща контейнери !

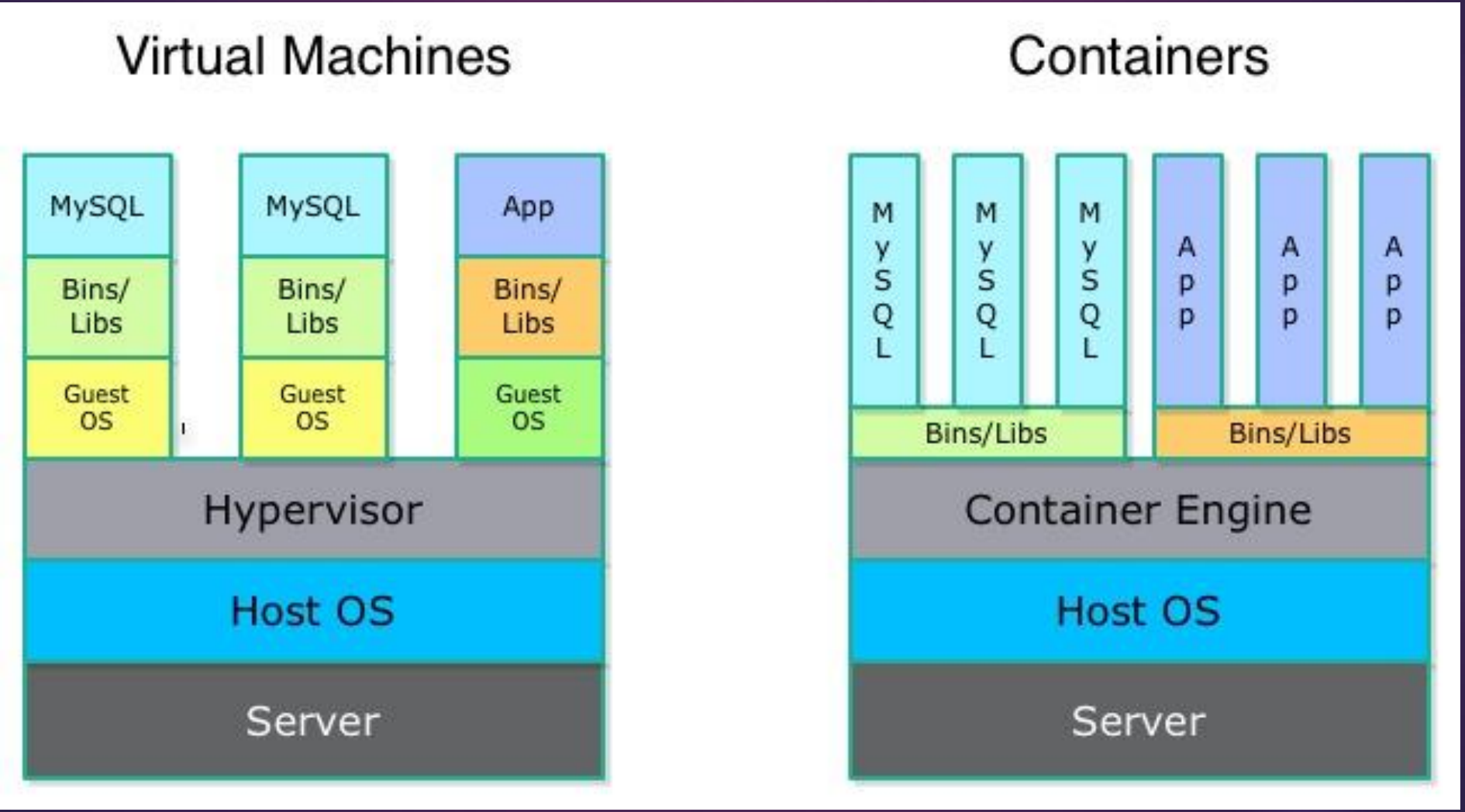
Какво са **контейнерите** и за какво са ни необходими ?

- Един контейнер съдържа цялата необходима среда за изпълнение: приложението, всички необходими библиотеки и други файлове за изпълнение, както и конфигуриращите файлове за стартирането, обединени в един пакет.

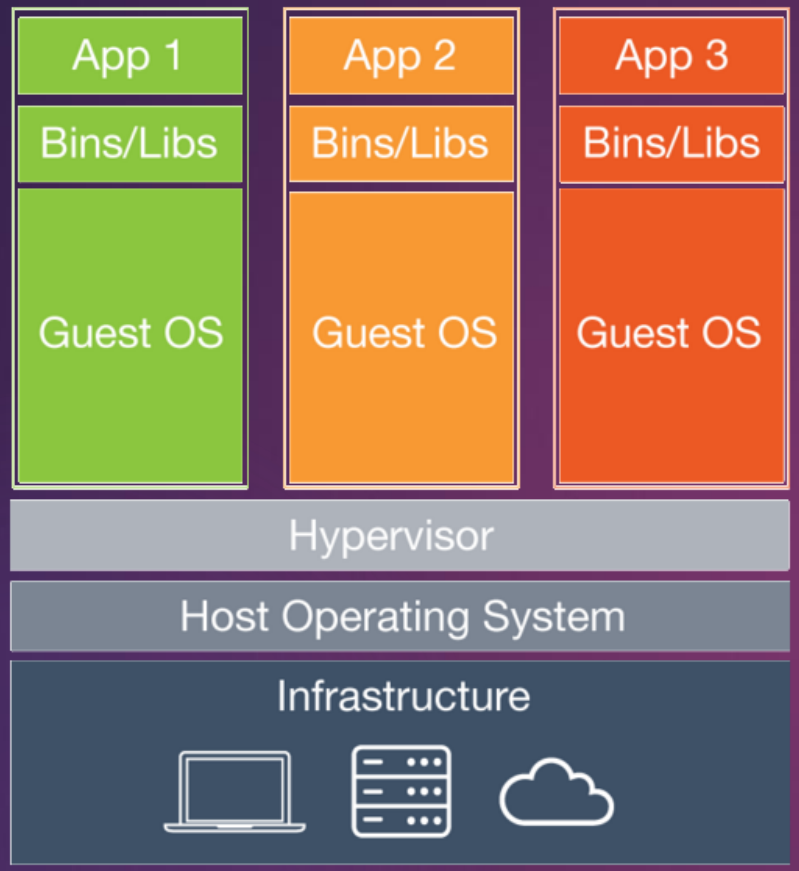
- Чрез контейнеризиране на платформата за приложения и всичките и необходимости, разликите в OS (операционна система) дистрибуциите и базисната инфраструктура се оеднаквяват.

- **docker ps** – Тази команда се използва за показване на работещите контейнери.
- **docker ps -a** – Тази команда се използва за показване на всички контейнери (независимо от състоянието)
- **docker kill** – Тази команда спира изпълнението му незабавно. Разликата между „docker kill“ и „docker stop“ е, че „docker stop“ дава време на контейнера да се изключи грациозно, в ситуации, когато отнема твърде много време, за да накара контейнера да спре, човек може да избере да docker kill.
- **docker stop** – Тази команда спира работещ контейнер
- **docker images** – Тази команда изброява всички локално съхранени докер изображения
- **docker port** – Тази команда се използва за показване на порта
- **docker stats** – Тази команда се използва за показване на подробности за контейнера заедно с използването на системни ресурси.
- **docker system df** – Тази команда се използва за получаване на цялата информация за използването на диска, групирана по компоненти на Docker.
- **docker rm** – Тази команда се използва за изтриване на спрян контейнер
- **docker rmi** – Тази команда се използва за изтриване на изображение от локално хранилище
- **docker rm -f** – Тази команда се използва за директно премахване на контейнера, без да го спирате
- **docker container prune** – Тази команда се използва за премахване на всички спрени контейнери
- **docker compose up -d** – Тази команда се използва за стартиране на всички услуги, дефинирани в docker-compose.yml

Виртуализация и контейнери

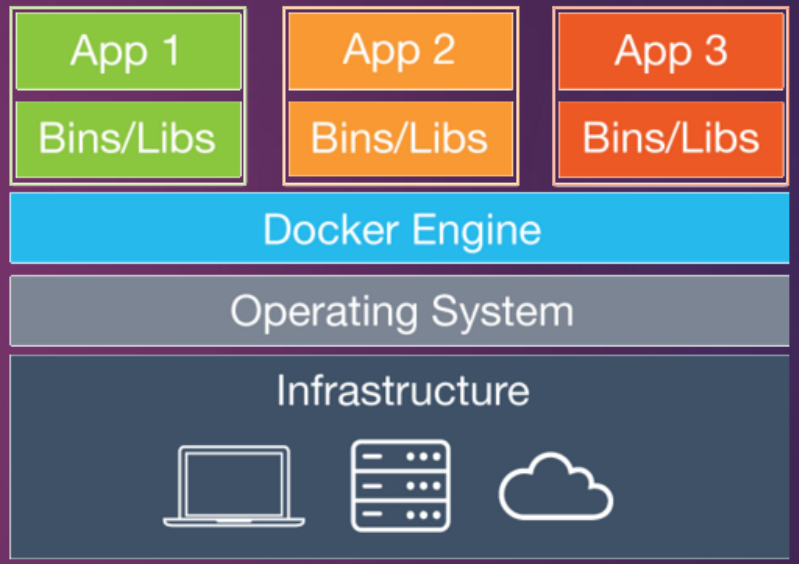


Virtual machines



Всяка виртуална машина включва приложение, необходимите му библиотеки, както и цялата операционна система – всяко, от които може да бъде с размер десетки гигабайти!

Containers



Контейнерите включват приложенията и всичко, което им е необходимо, но делят ядрото с останали контейнери. Те работят като изолирани процеси в потребителско пространство на сървъра. Също така не са прикачени към определена инфраструктура – Docker контейнерите работят на всеки компютър, на която и да е инфраструктура и във всеки облак.

ПРОЦЕСИ - ?

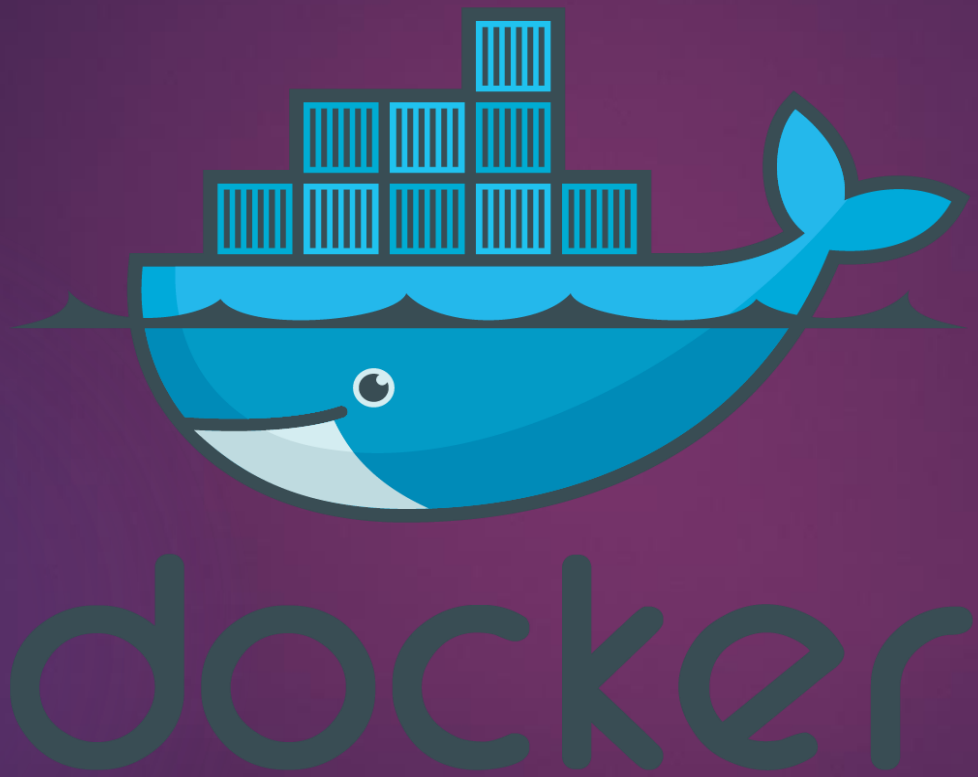
ВИРТУАЛИЗАЦИЯ - ВСЯКА ОС УПРАВЛЯВА СОБСТВЕНИТЕ СИ ПРОЦЕСИ;

КОНТЕЙНЕРИ – ВСЕКИ КОНТЕЙНЕР ЗАЕМА СОБСТВЕН ПРОЦЕС, СОБСТВЕНО ПРОСТРАНСТВО ОТ ФС.

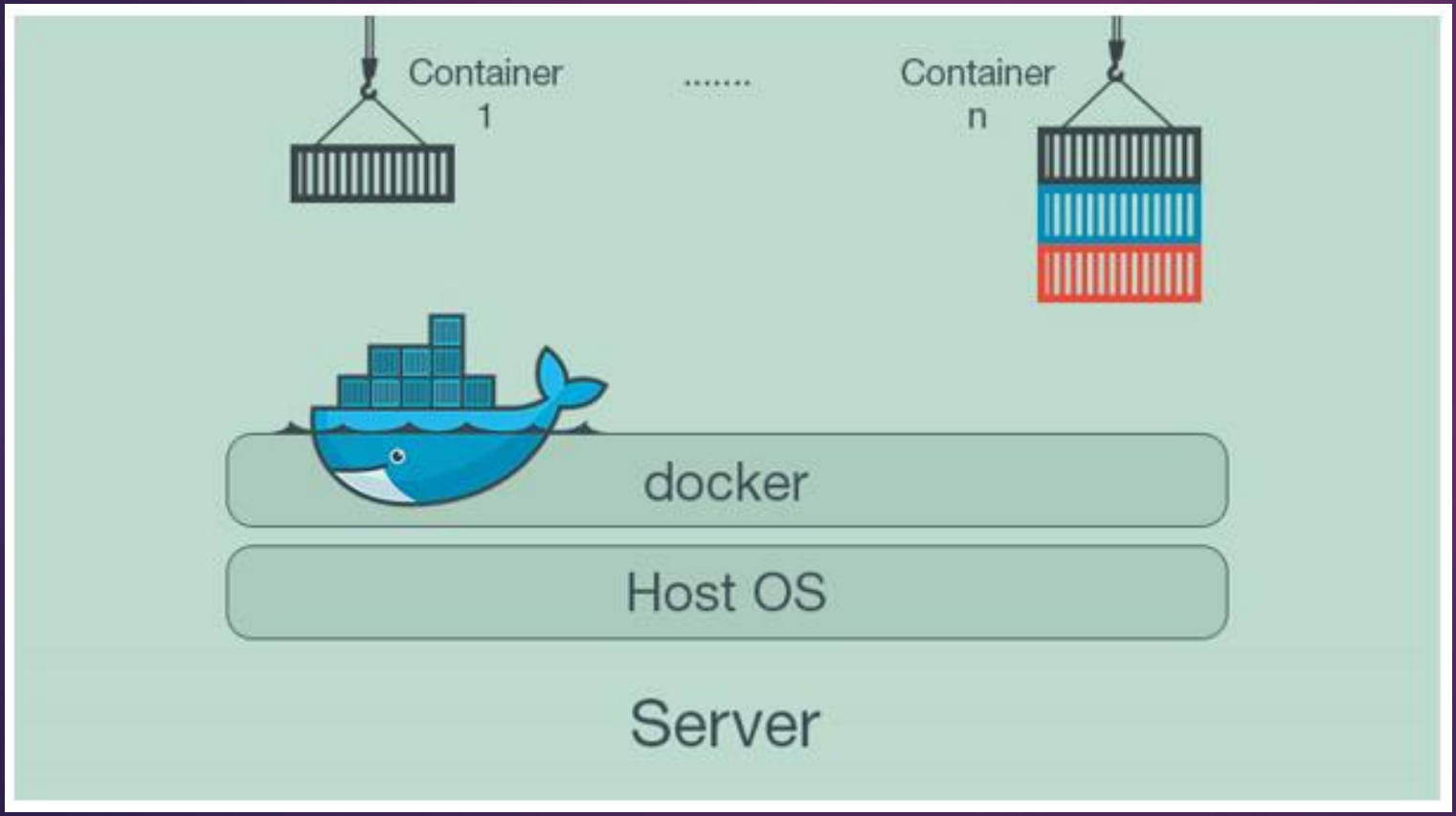
Moroever, LXC, LXD, CGManager и Docker

VMware vSphere, Virtual Box и Hyper-V

Docker



Каква е разликата между Docker и контейнерите?



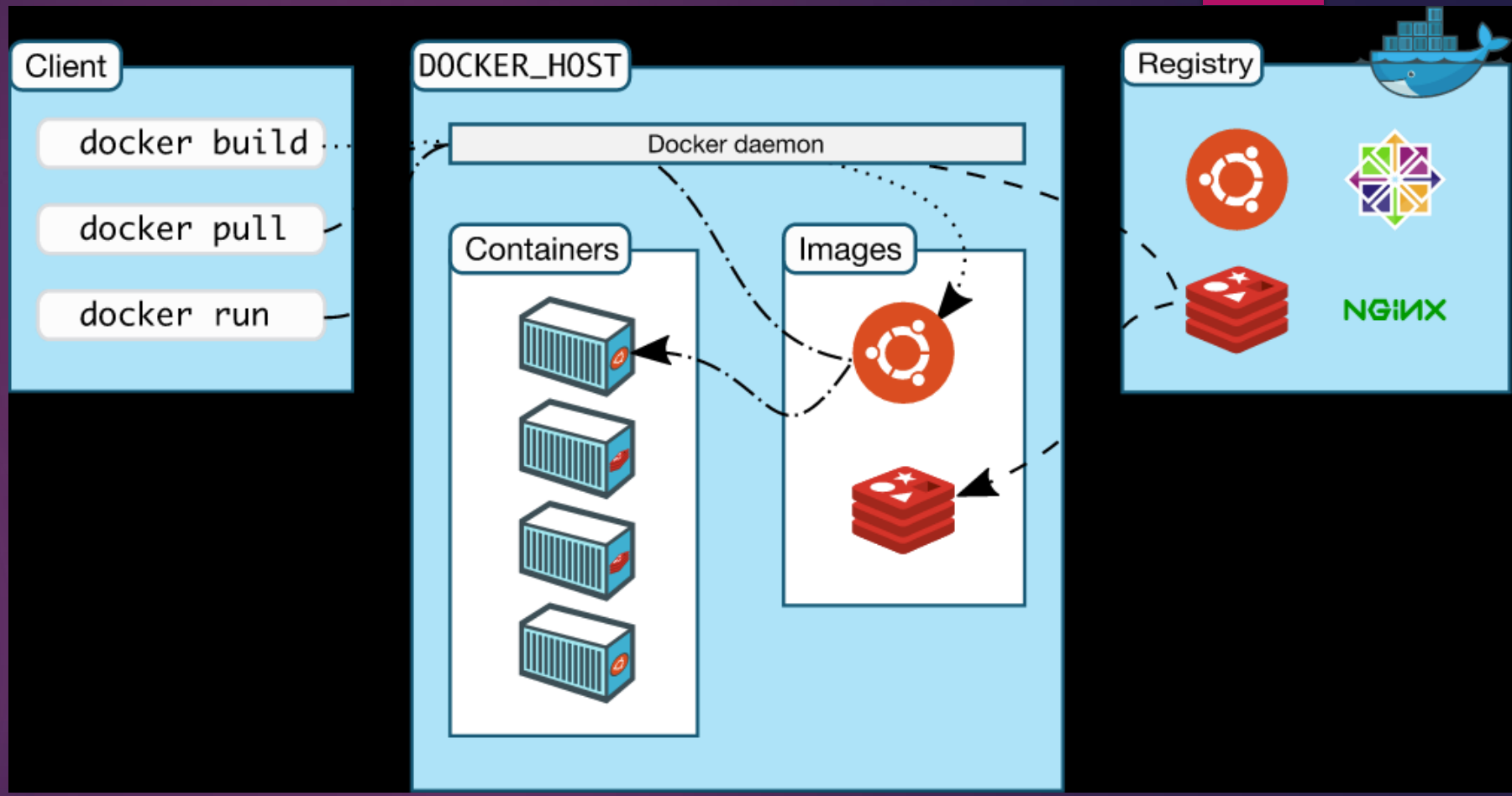
Изградена е преди почти 15 години в Linux, под формата на LXC (Linux Containers), като подобна виртуализация на ниво OS е предлагана и от FreeBSD jails, AIX Workload Partitions and Solaris Containers.

Контейнери

- **Изграждане (Build)**
- С помощта на контейнерите може да създадете приложения в изолирана среда без да се притеснявате за несъвместимост между QA машините и тези на разработчиците, както и без «счупвания» между различните платформи и езици.
- **Доставяне (Ship)**
- Иновативната технология ви позволява да проектирате целия цикъл на разработване, тестване и дистрибуция на приложението, както и да го управлявате чрез потребителски интерфейс.
- **Стартиране (Run)**
- Тази платформа ви дава възможност да разгърнете скалируеми услуги, сигурни и надеждни, на разнообразни платформи.

Какво е Docker?

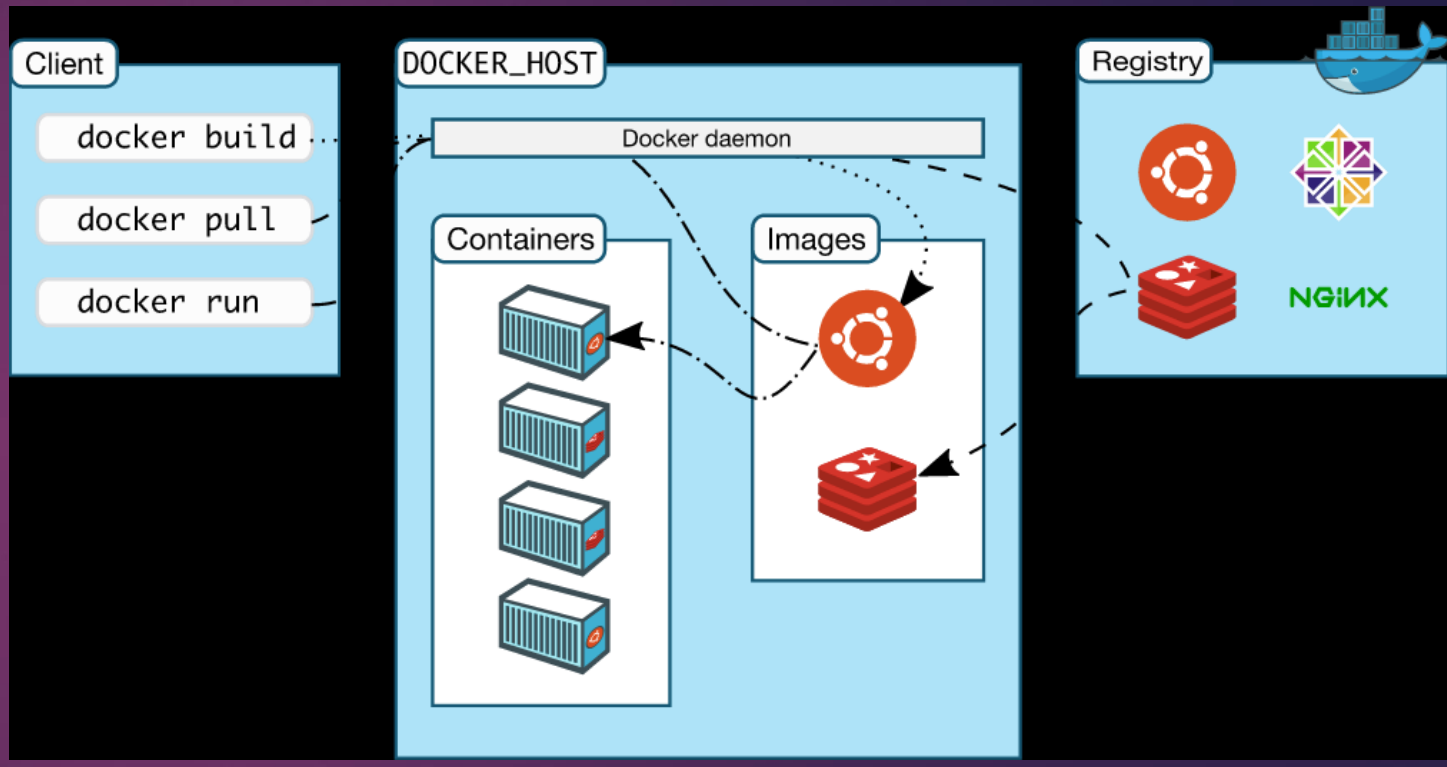
Docker е платформа с отворен код изградена на базата на контейнер технологията, която ви позволява да изграждате, доставяте и стартирате разнообразни приложения отделяйки ги от инфраструктурата. В същността си, платформата ви дава възможност да стартирате всякакви приложения, защитено изолирани в контейнери. Изолираната и безопасна среда ви позволява да работите едновременно с много контейнери на вашия сървър. Олекотеният принцип на работа, който не изисква допълнително натоварване на хипервайзор или сложна конфигурация и настройка на среда за разработка, ви позволява да използвате максимума на вашия хардуер.



Docker - структура

Docker използва клиент-сървър архитектура. Клиентът комуникира с **daemon**, който поема тежките работни процеси при изграждането, стартирането и дистрибуцията на Docker контейнерите.

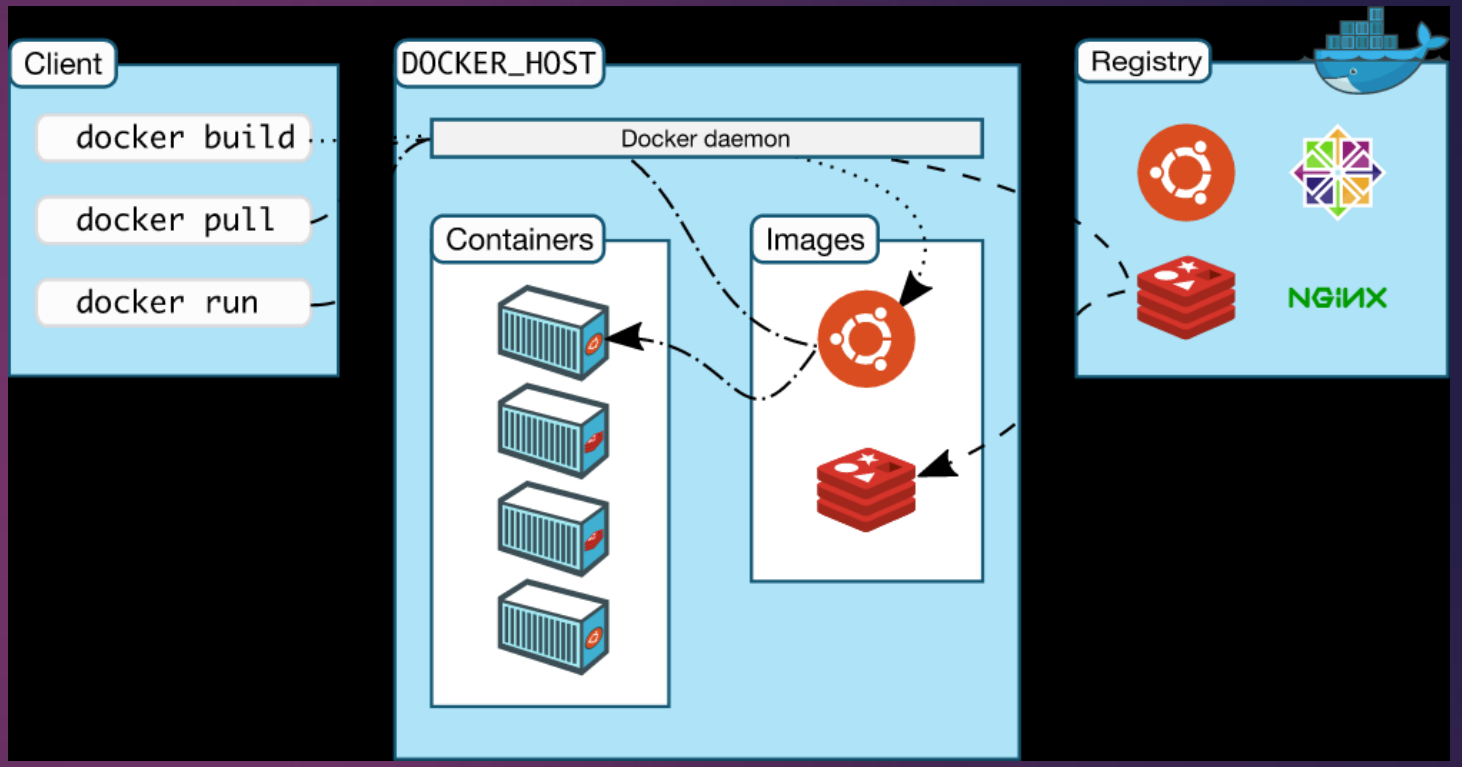
И **клиентът и daemon** могат да работят на една и съща система,
а също може да свържете **Docker** клиента към отдалечен Docker daemon. Двете комуникират чрез **sockets** или чрез **RESTful API**.



Docker - структура

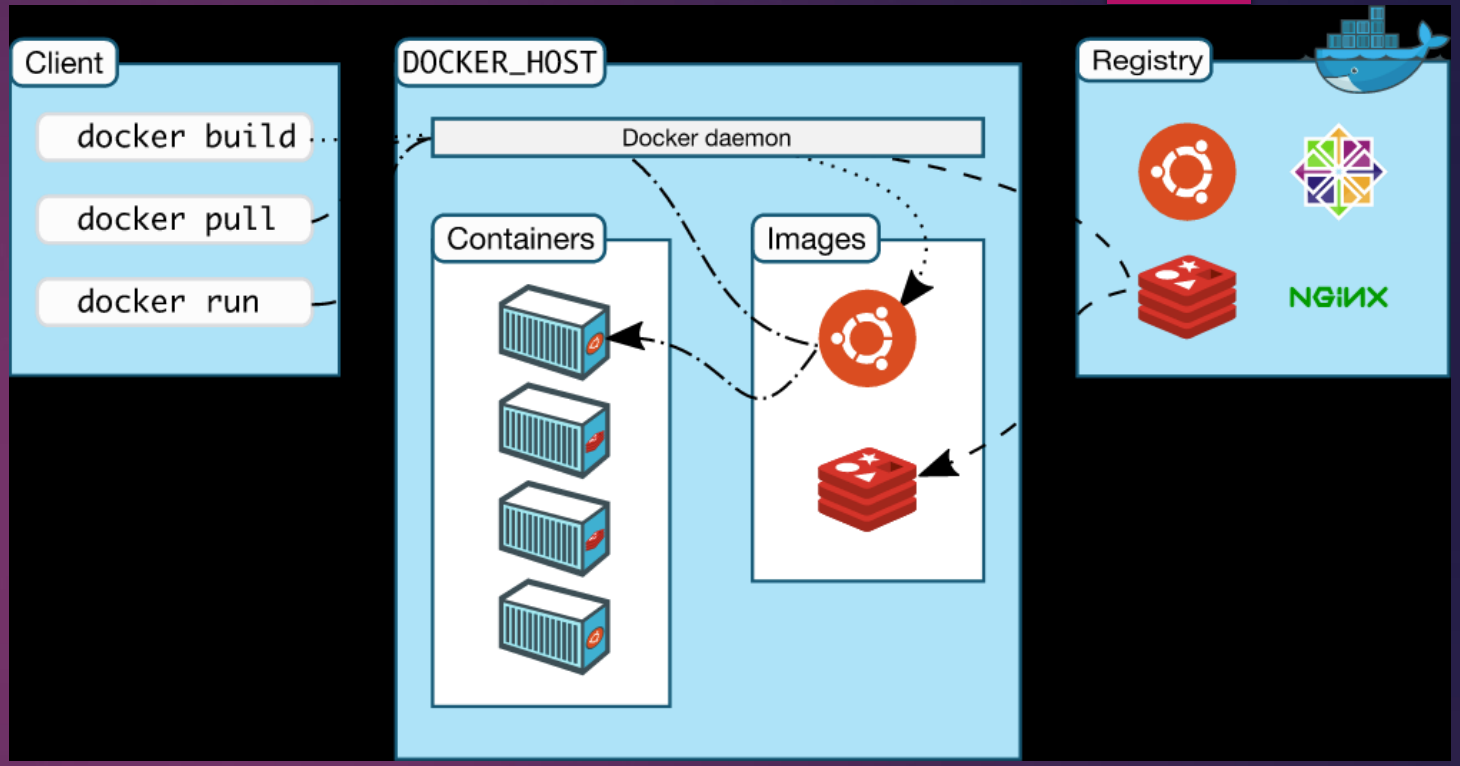
Docker клиент е първичният потребителски интерфейс на Docker.

Приема зададените команди от потребителите и постоянно комуникира с **Docker daemon – а.**



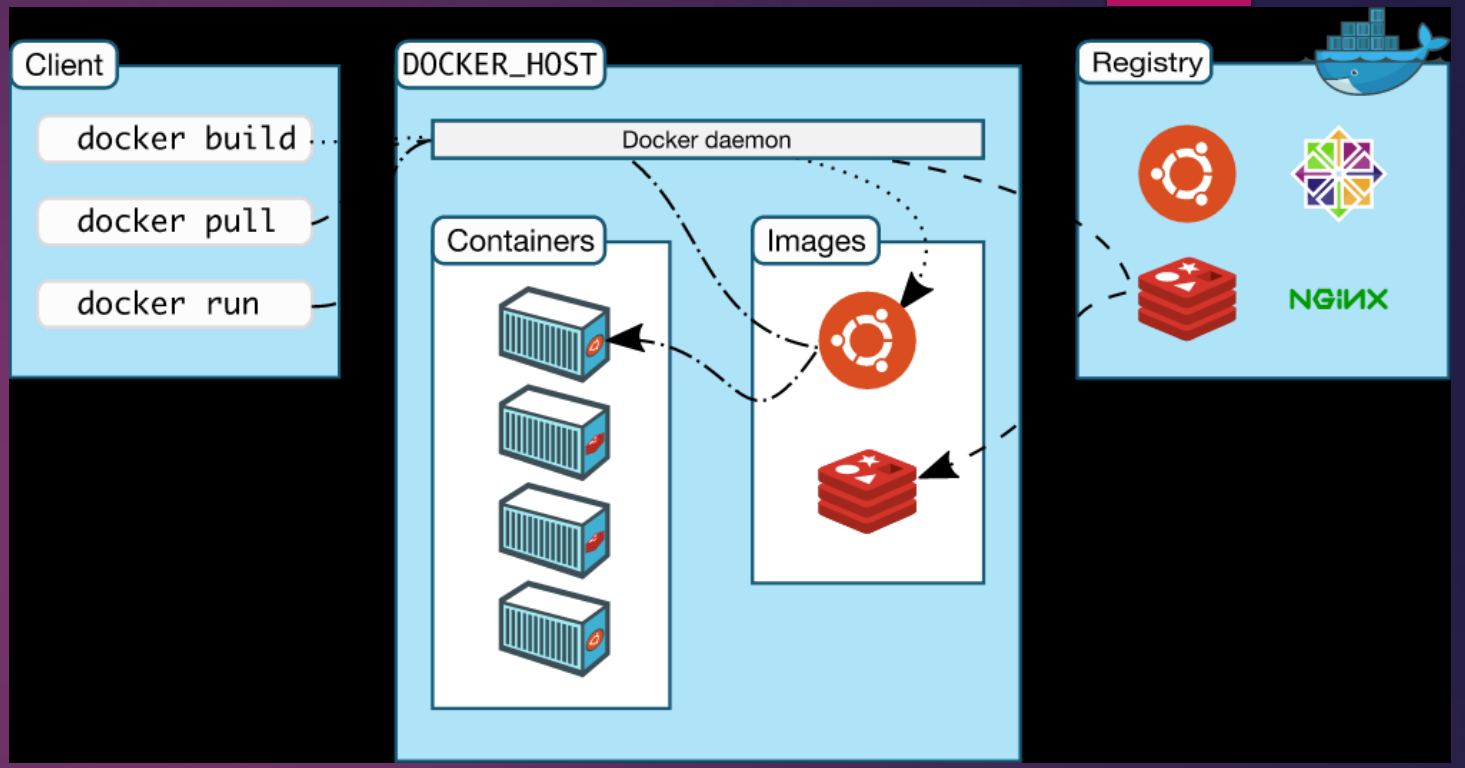
Docker - Структура

Docker daemon – както можете да видите на графиката по-горе, Docker daemon работи на сървър. Потребителите не работят директно с него, а през Docker клиента.



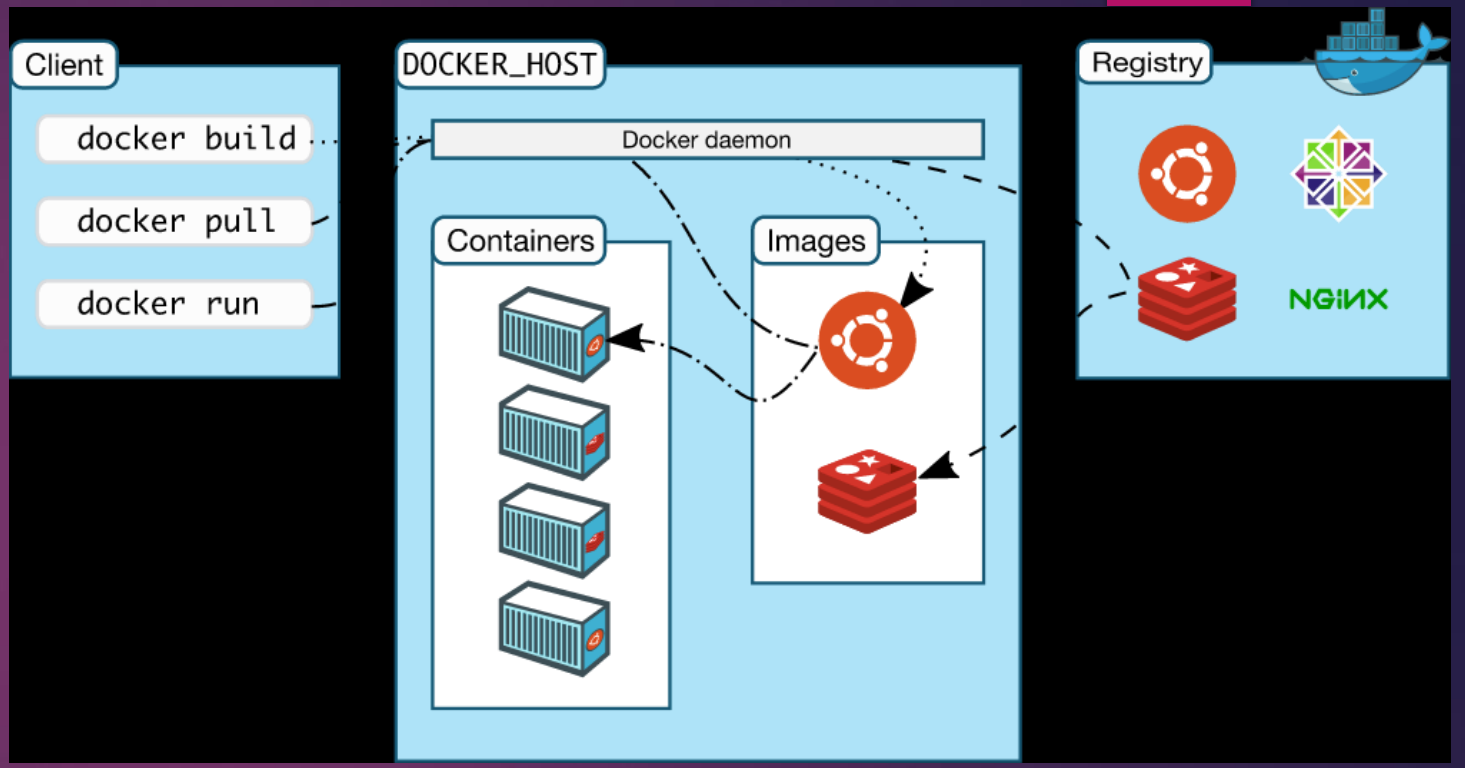
Docker - Структура

Docker изображения – read-only темплейти. Например, едно изображение може да има операционна система Ubuntu с Apache и инсталирано ваше уеб приложение. Изображенията се използват за създаването на Docker контейнери. Docker предоставя опростен начин за създаване на нови изображения или за ъпдейта на съществуващи такива, а също така може да изтегли и Docker изображения, вече създадени от други потребители. Изображенията са **изграждащият**компонент на Docker.



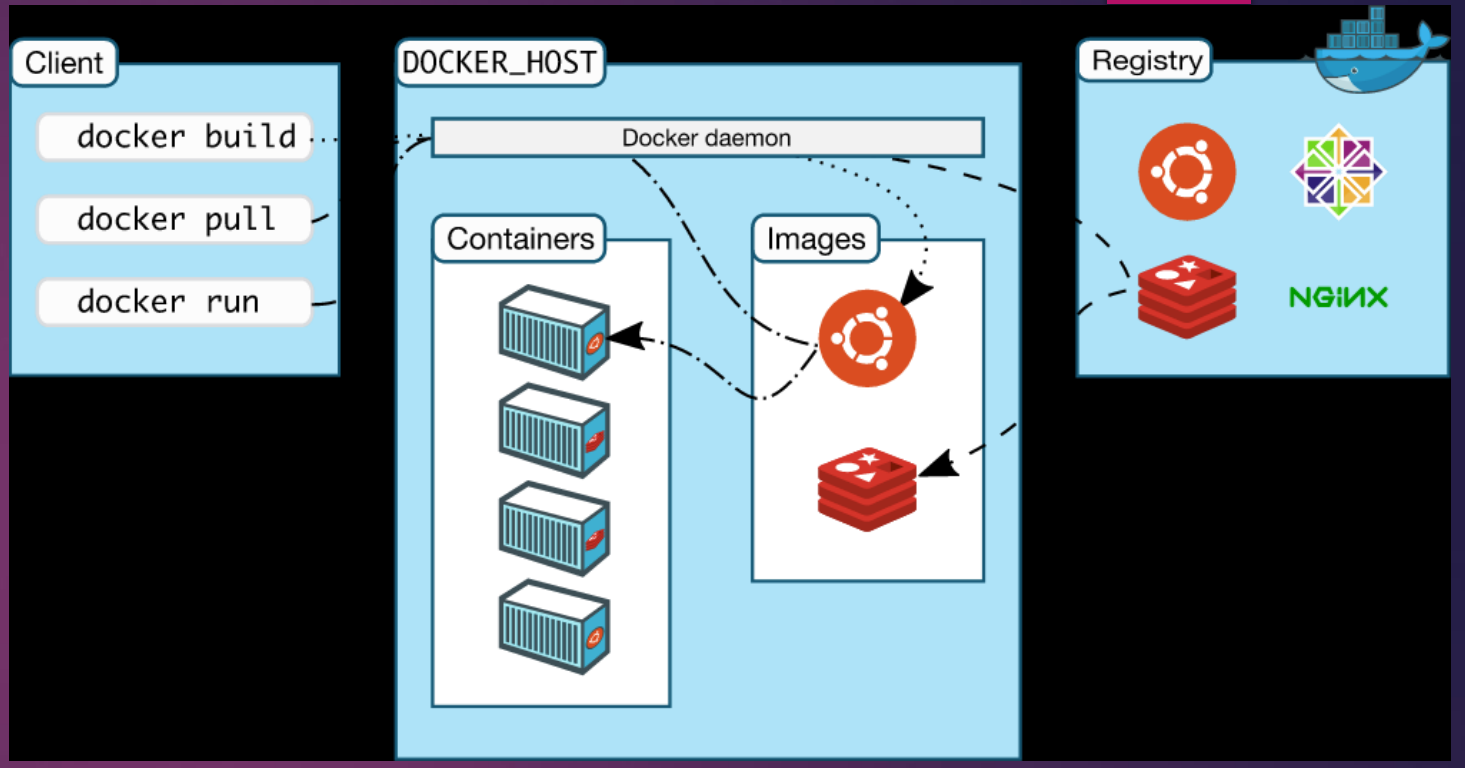
Docker - Структура

•**Docker регистри** – те съдържат изображенията. Това са публични или лични регистри, от които можете да качвате или теглите изображения. Публичният регистър се нарича **Docker Hub**. Той предоставя огромни колекции от съществуващи изображения, които можете да използвате и такива, които вие сте изработили или, които вече са създадени по-рано от друг. Регистрите са **дистрибуционният** компонент на Docker.



Docker - Структура

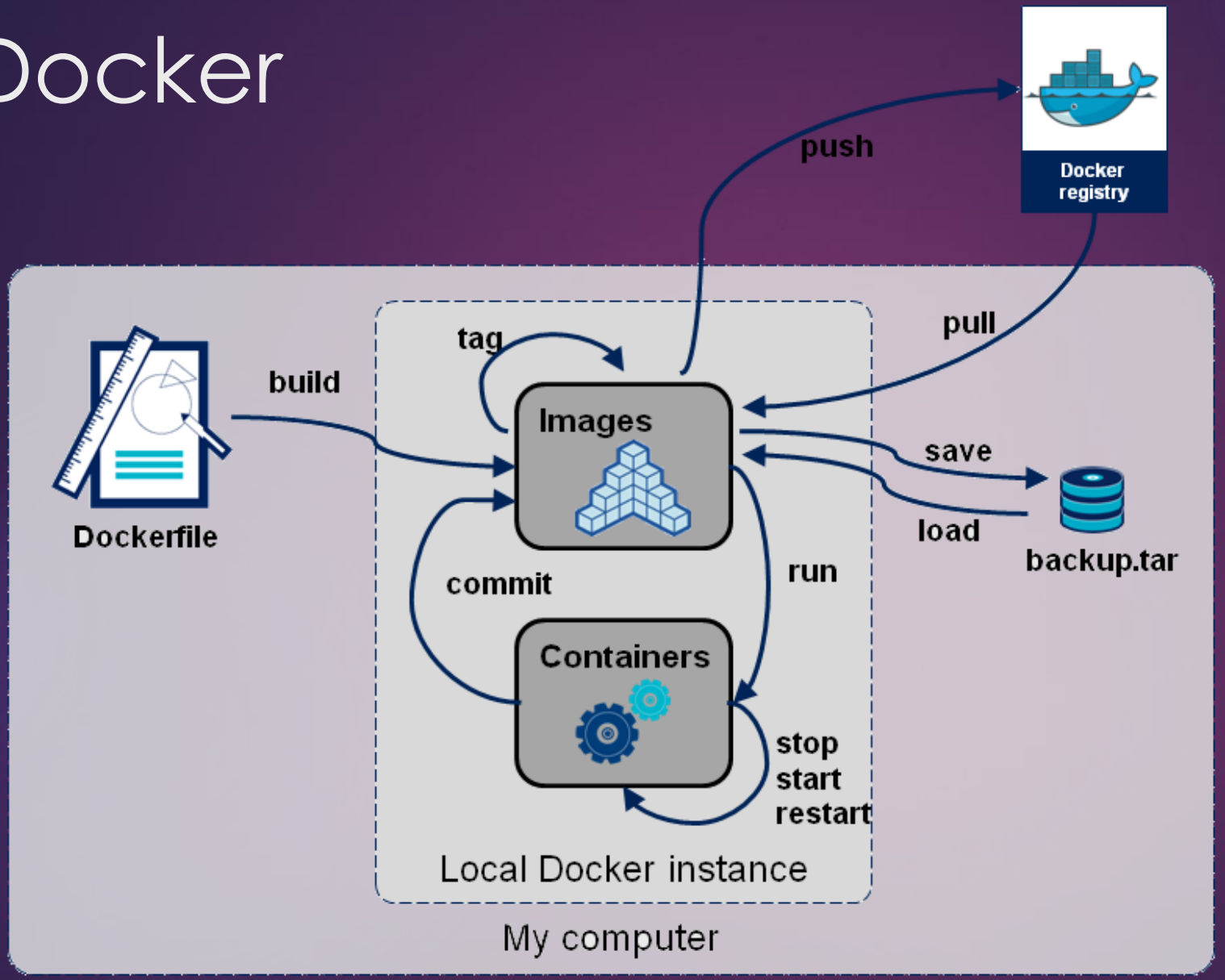
Docker контейнери – Docker контейнерите предоставят на програмистите, разработчиците и софтуерните инженери всички необходими инструменти за изграждането, тестването и пускането на модерни и атрактивни приложения.



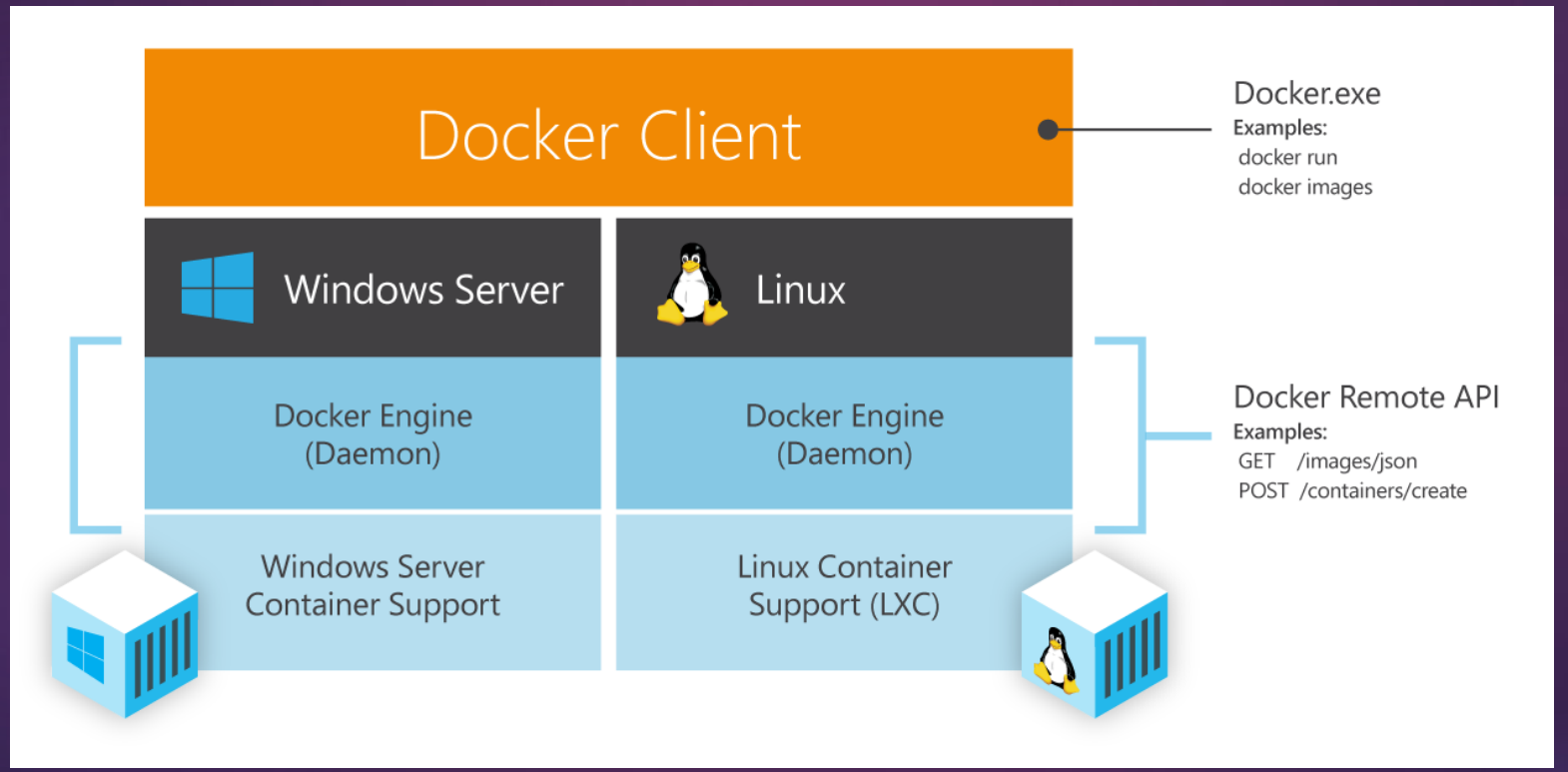
Механизъм на действие

- **Docker контейнерите** много приличат на директориите. Те съхраняват всичко, което ви е необходимо за изграждането и стартирането на едно качествено приложение. По-горе вече споменахме, че контейнерите се изграждат с помощта на изображенията. Docker контейнерите с лекота могат да бъдат управлявани, стартирани, спрени, 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, вие управлявате и разгръщате вашите приложения безпроблемно.
- В контейнерите се „пакетират“ парченца софтуер в изолирана файлова система, която съдържа всичко необходими, за да стартирате: код, runtime, системни настройки, системни библиотеки – всичко, което можете да инсталирате на вашия сървър. Това гарантира сигурно стартиране на приложението, без изменения и без значение от средата.

Docker


push
Docker registry
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

На каква операционна система трябва да бъдат стартирани Docker



Предимствата на Docker контейнерите:

- **Скалируемост**– Бързината на работа контейнерите, позволяват скалируемостта да се случва в реално време – без значение дали ще се увеличава или намалява;
- **Преносимост portable** – Този тип платформа позволява висока преносимост на информация при работни натоварвания. Контейнерите могат да работят на локалните машини на разработчиците, на физически или виртуални машини в дата центрове или на клауд сървъри;
- **Компакност** – Олекотена технология и бързина. Контейнер технологията предоставя надеждна и ефективна алтернатива на хипервайзор базирани виртуални машини. Това е от изключителна полза при силно компактните среди: например, изграждането на собствен клауд или работна платформа;
- **Олекотена работа** – Контейнерите работят на индивидуални машини, всяка от които споделя едно и също ядро на операционната система, така че да могат да стартират мигновено и да използват много по-ефективно RAM-та. Изображенията са изградени от слоеве, така че да могат да споделят общи файлове, което прави използването на диска и сваляне на файлове много по-ефективно;
- **Отворен код** – Docker контейнерите са базирани на отворени стандарти, позволявайки им да работят на всички водещи Linux дистрибуции и Microsoft операционни системи, както и да имат поддръжка към всяка инфраструктура.
- **Сигурност** – Контейнерите изолират приложенията едно от друго и прилежащата инфраструктура, като същевременно осигуряват и допълнителен слой на защита.