

# Введение в Docker

# СОДЕРЖАНИЕ

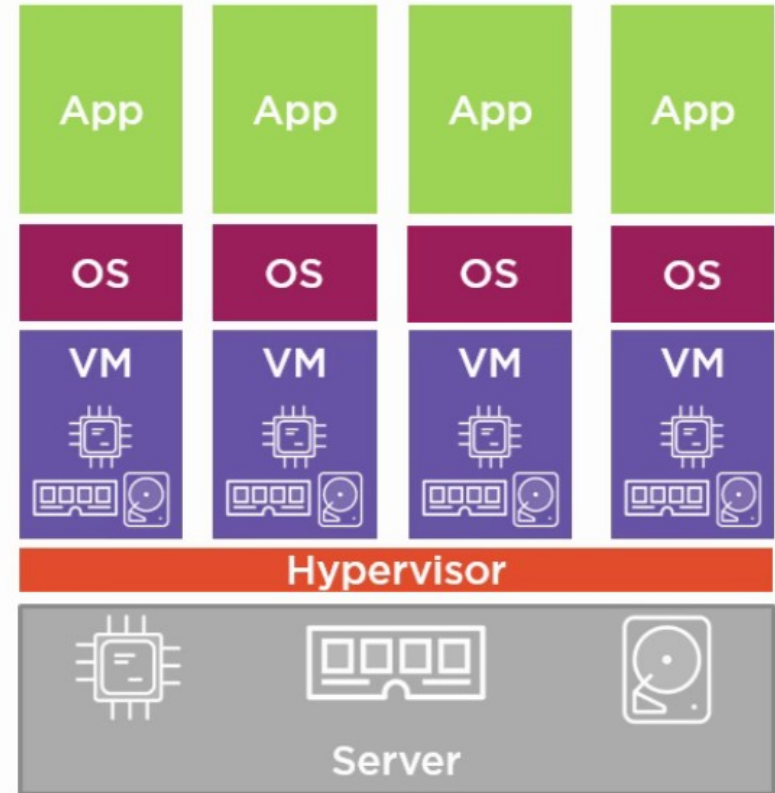
---

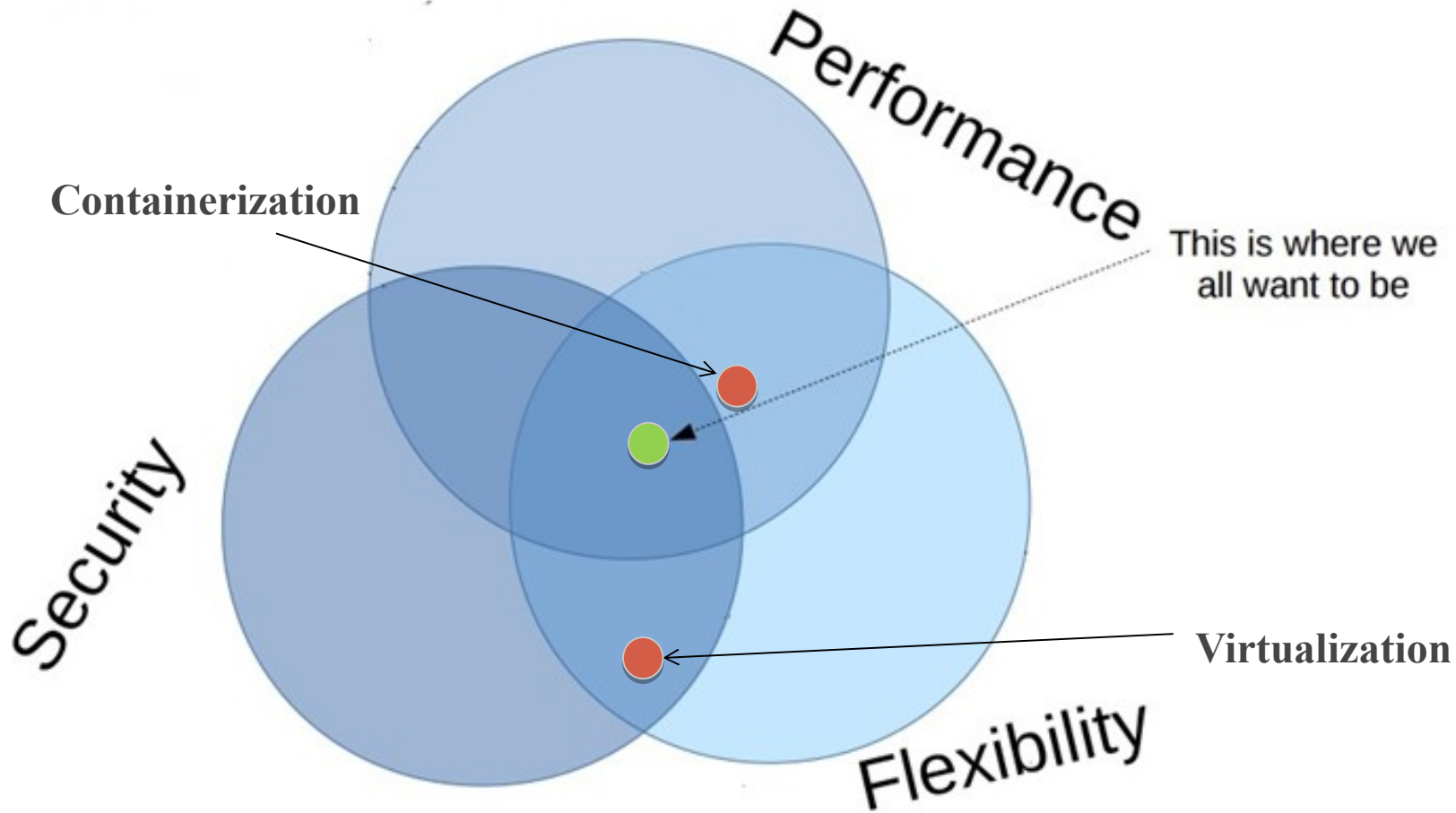
1. Что такое контейнеры
2. Как работает Docker
3. Применение Docker

# ВИРТУАЛИЗАЦИЯ

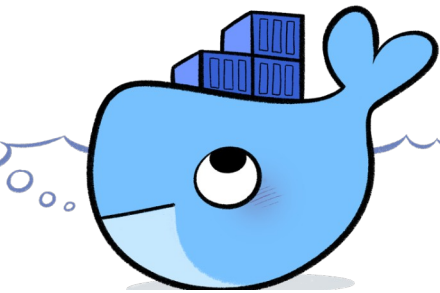
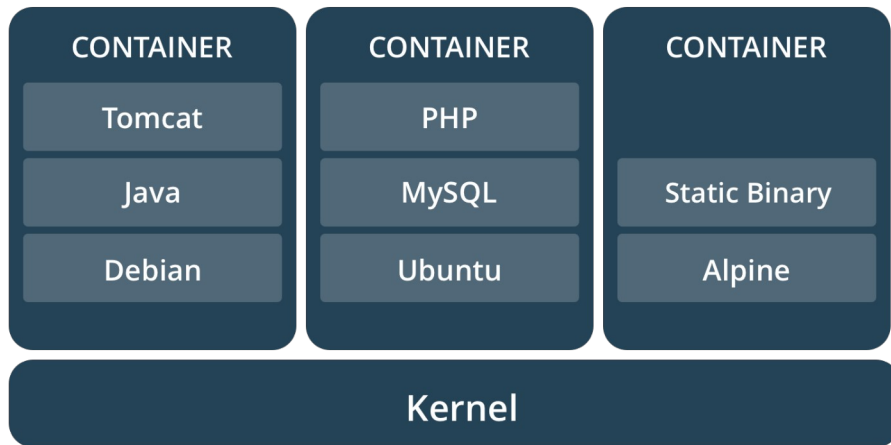
Каждая операционная система:

1. Использует процессор
2. Использует ОЗУ
3. Использует диск
4. Требует лицензии
5. Нуждается в обслуживании

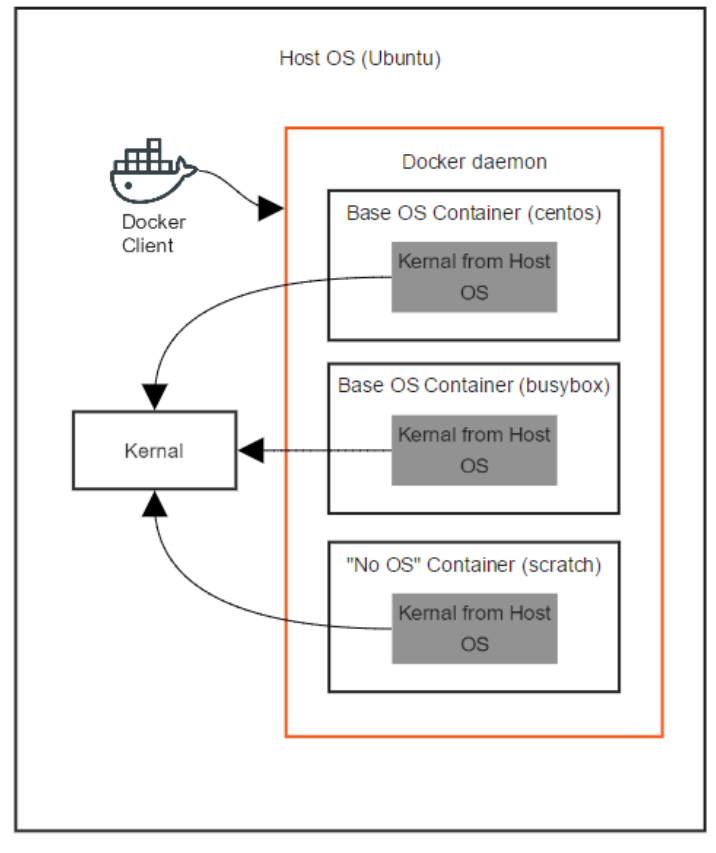




# ЧТО ТАКОЕ КОНТЕЙНЕР?

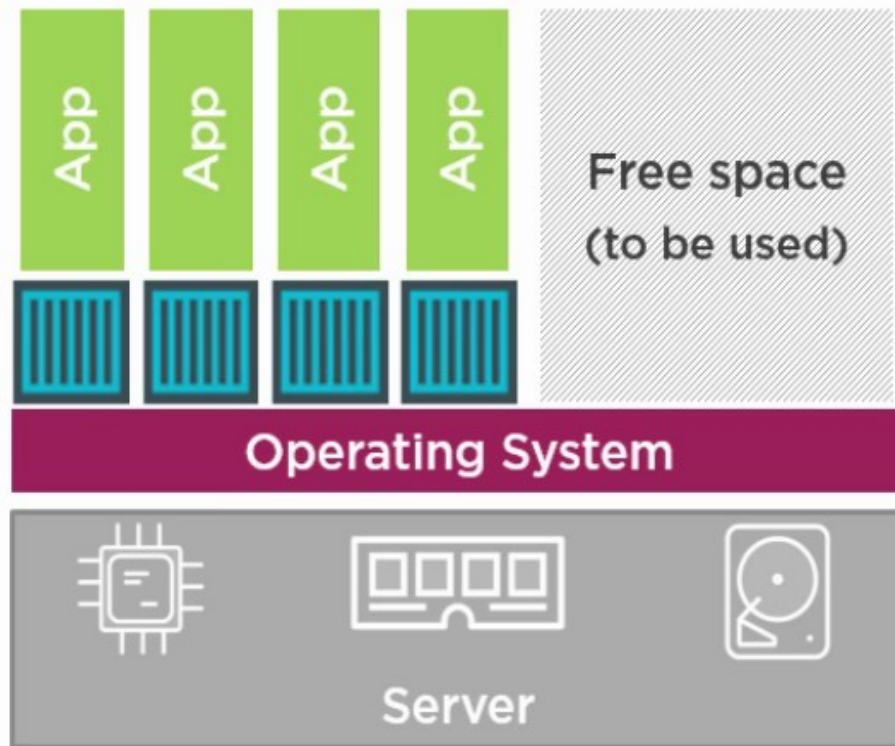


## Linux Containers

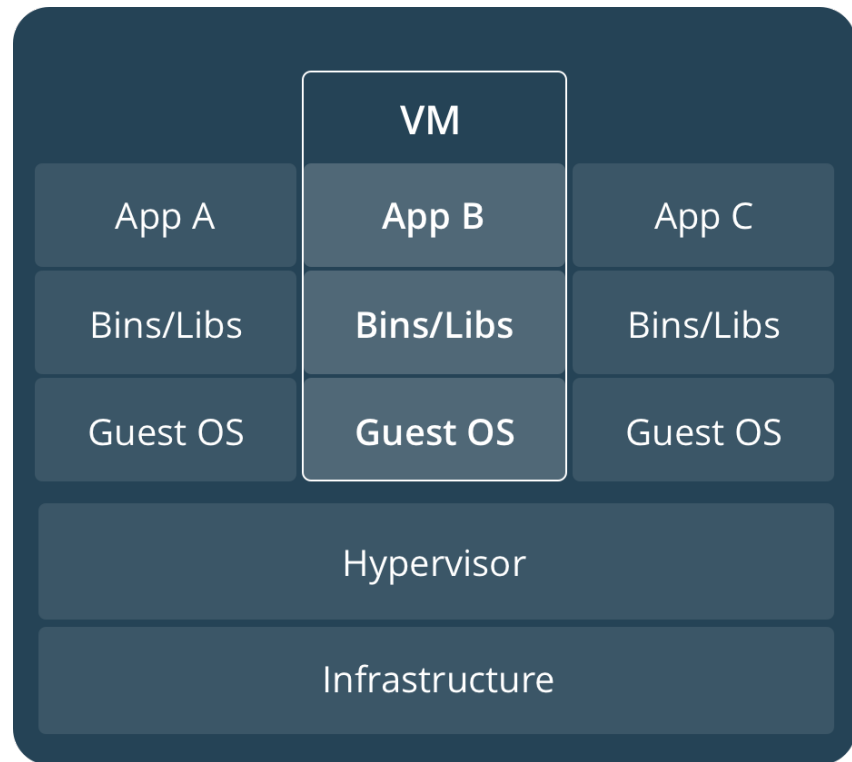
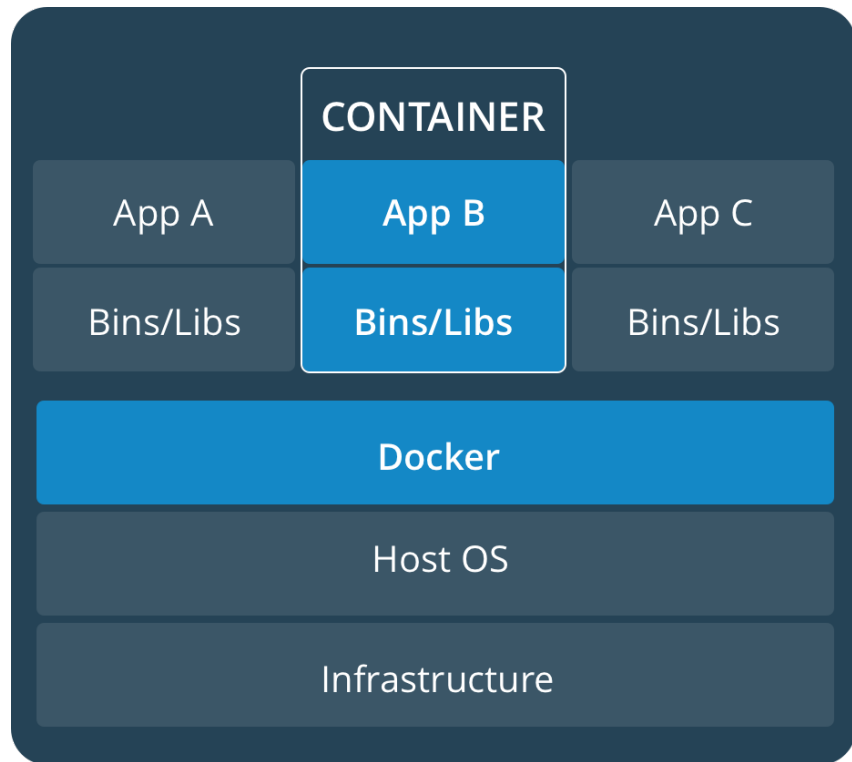


# КОНТЕЙНЕРЫ

1. Используют одно ядро
2. Не требуют лицензий
3. Просто масштабировать
4. Консолидируют ресурсы



# ВИРТУАЛЬНЫЕ МАШИНЫ И КОНТЕЙНЕРЫ

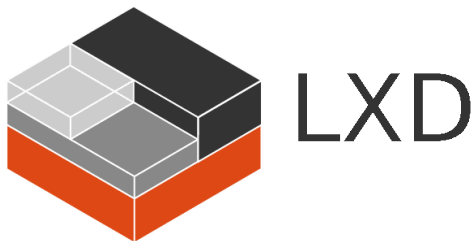


# ВИДЫ КОНТЕЙНЕРОВ

---



- Introduced in December 2014
- Production-ready release in February 2016
- Focuses on compatibility
- Supports multiple container formats, including Docker
- like Docker, it is optimized for application containers



- Introduced in November 2014
- Container hypervisor
- Operating System centric

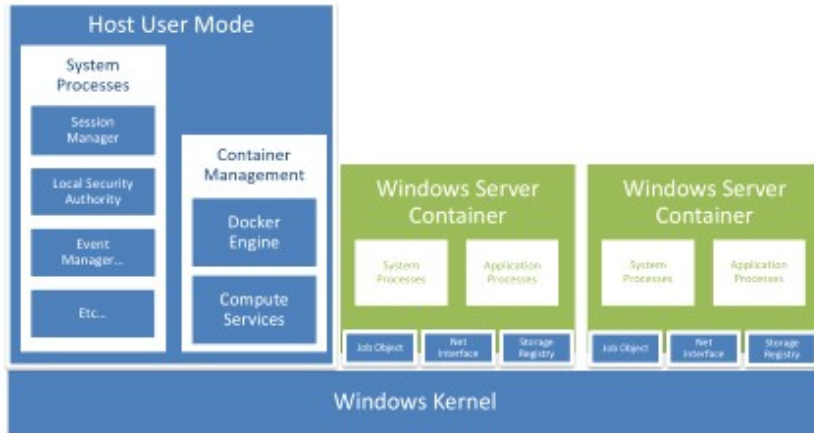


- An extension of the Linux kernel
- Uses containers for entire operating systems



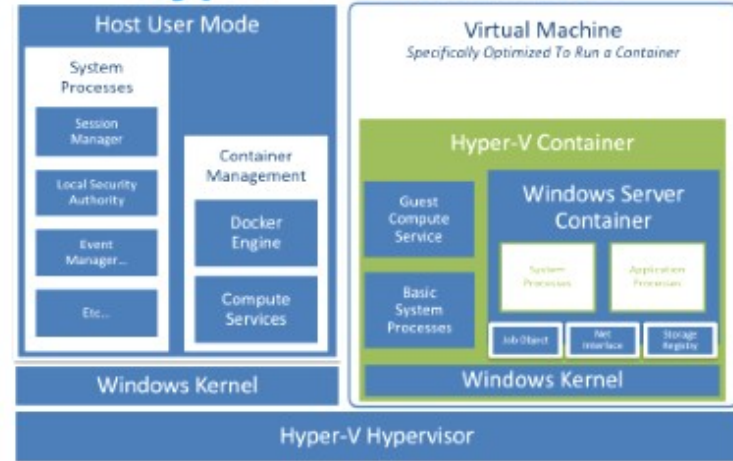
# WINDOWS КОНТЕЙНЕРЫ

## Windows Server Containers



**Windows Server Containers** – provide application isolation through process and namespace isolation technology.

## Hyper-V Containers



**Hyper-V Isolation** – expands on the isolation provided by Windows Server Containers by running each container in a highly optimized virtual machine.

# DOCKER КОНТЕЙНЕРЫ



## What

A process in the container shouldn't see other processes running in the system and should feel itself as the only process in the system

PROCESS  
ISOLATION

We should have facility to manage systems resources – (for example) to provide necessary amount of RAM, CPU shares and so on to the Container

COMPUTE  
RESOURCES

A process in the container should see/use the only that part of the filesystems which is required by it

FS ISOLATION

A process in the container should have enough permissions to manage/use kernel

KERNEL

A process in the container should have access to ethernet device

VENTH

A few containers can expose the same ports, but is shouldn't cause any problem

PORT MAPPING

A service should have facility to keep data, even it goes down accidently

VOLUMES

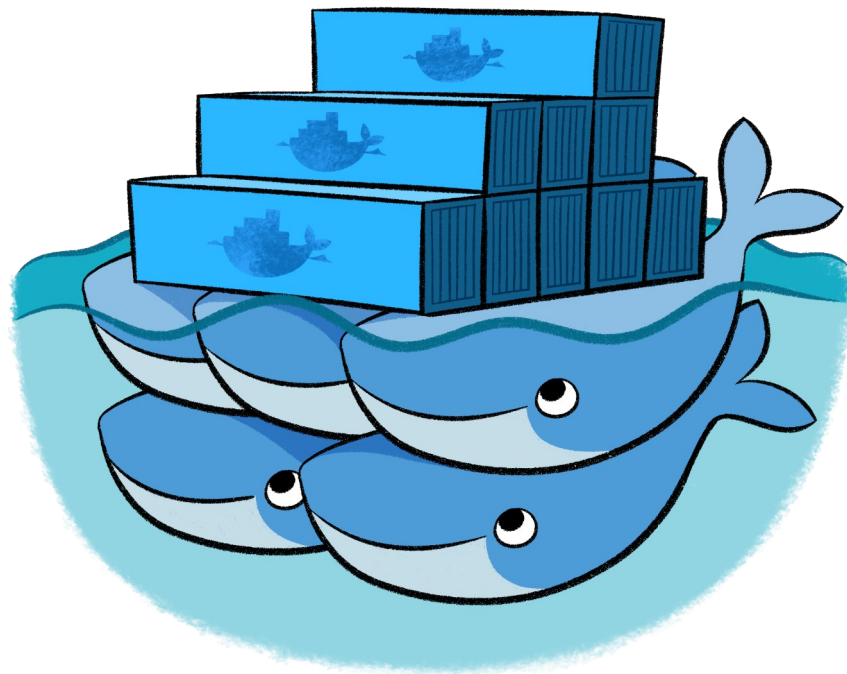
Service should be able to communicate with other services by IP/names

NETWORK

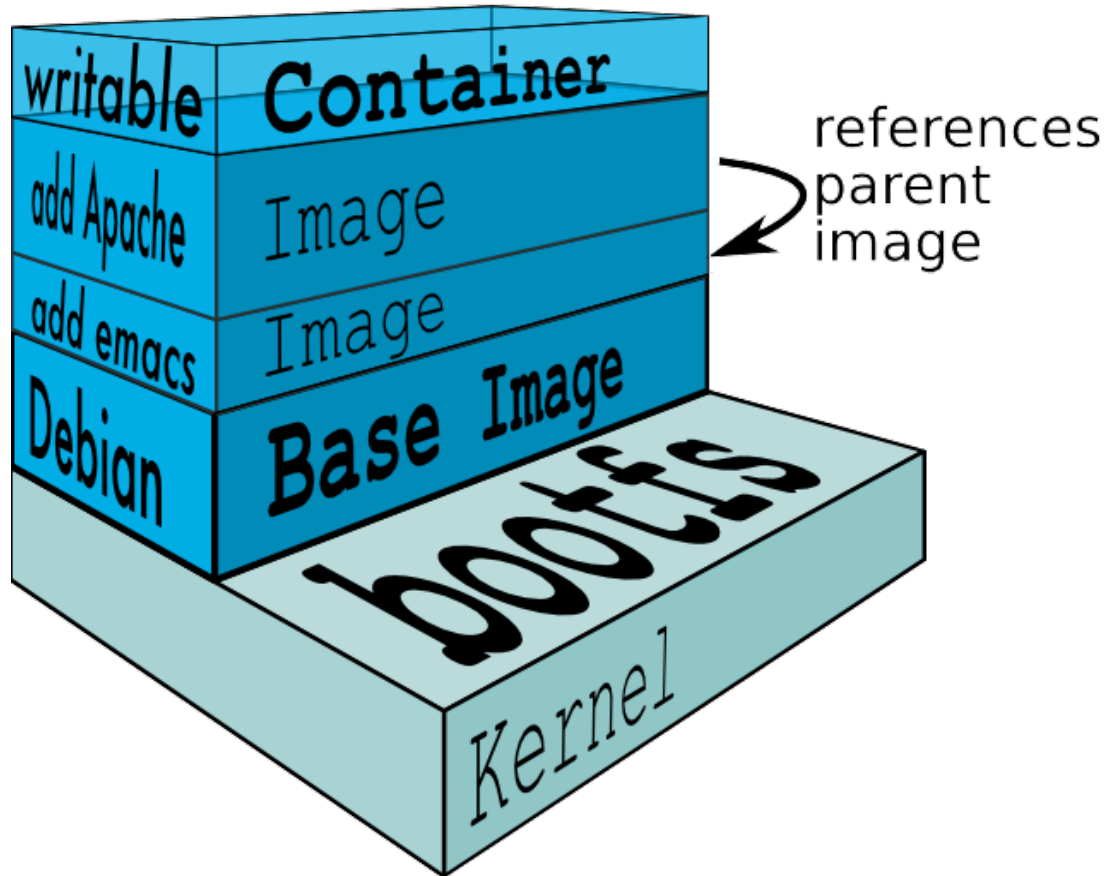
# КОМПОНЕНТЫ DOCKER

---

- 1 Docker images
- 2 Docker engine
- 3 Docker containers
- 4 Docker registry

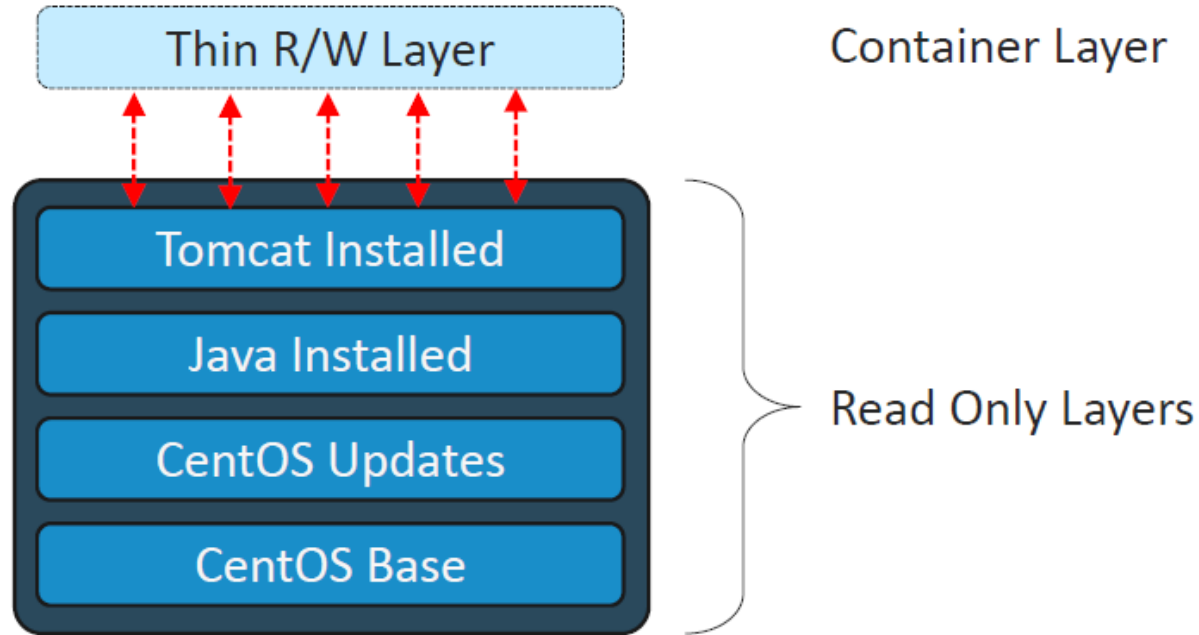


# ОБРАЗЫ И КОНТЕЙНЕРЫ



# ОБРАЗЫ И КОНТЕЙНЕРЫ

---



# СБОРКА DOCKER ОБРАЗОВ

---



## Dockerfile

```
FROM python:2.7

RUN pip install Flask
COPY . /

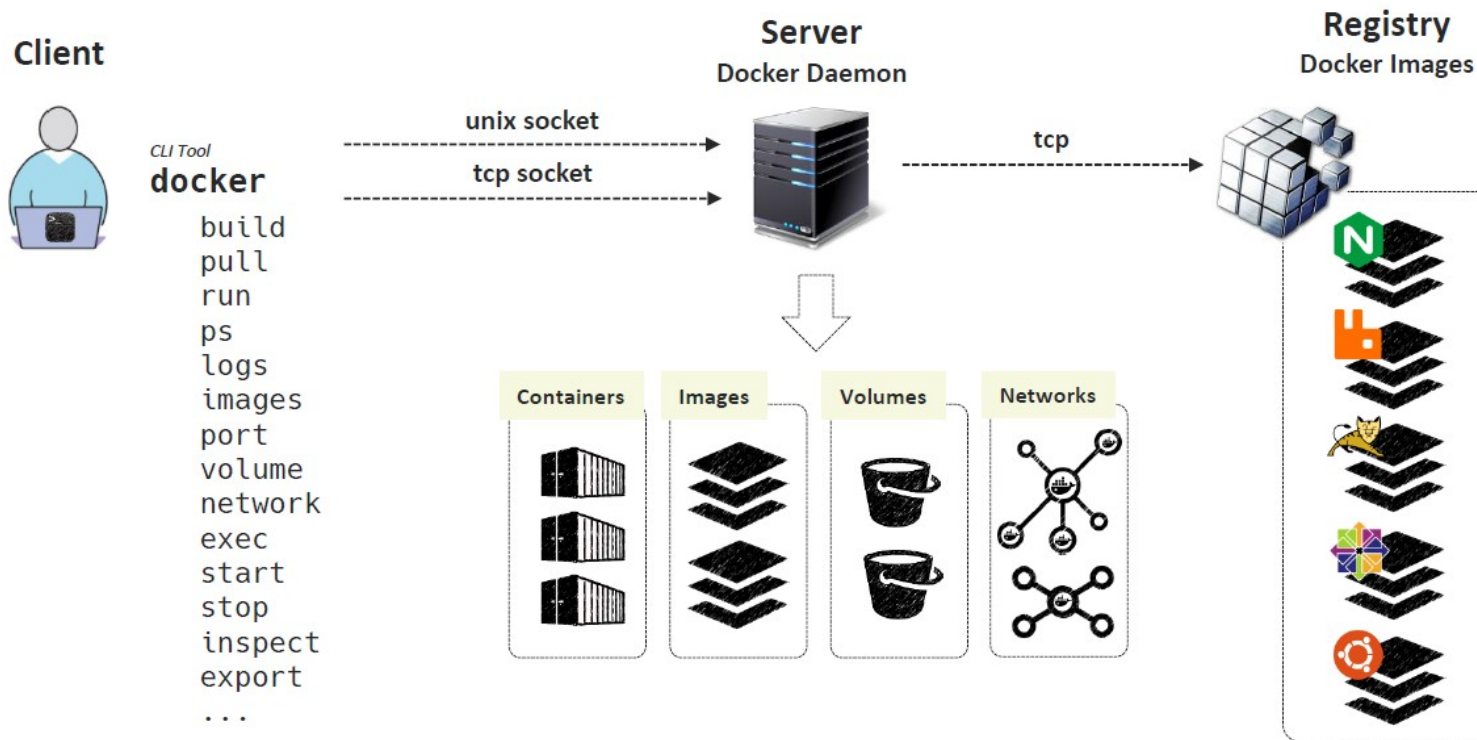
ENV FLASK_APP=hello.py
EXPOSE 5000
CMD flask run --host=0.0.0.0
```

Each instruction creates one layer:

- **FROM** creates a layer from the python:2.7 Docker image.
- **RUN** performs necessary commands
- **COPY** adds files from your Docker client's current directory
- **ENV** sets environment variable
- **EXPOSE** defines which ports will be exposed by container
- **CMD** specifies what command to run within the container

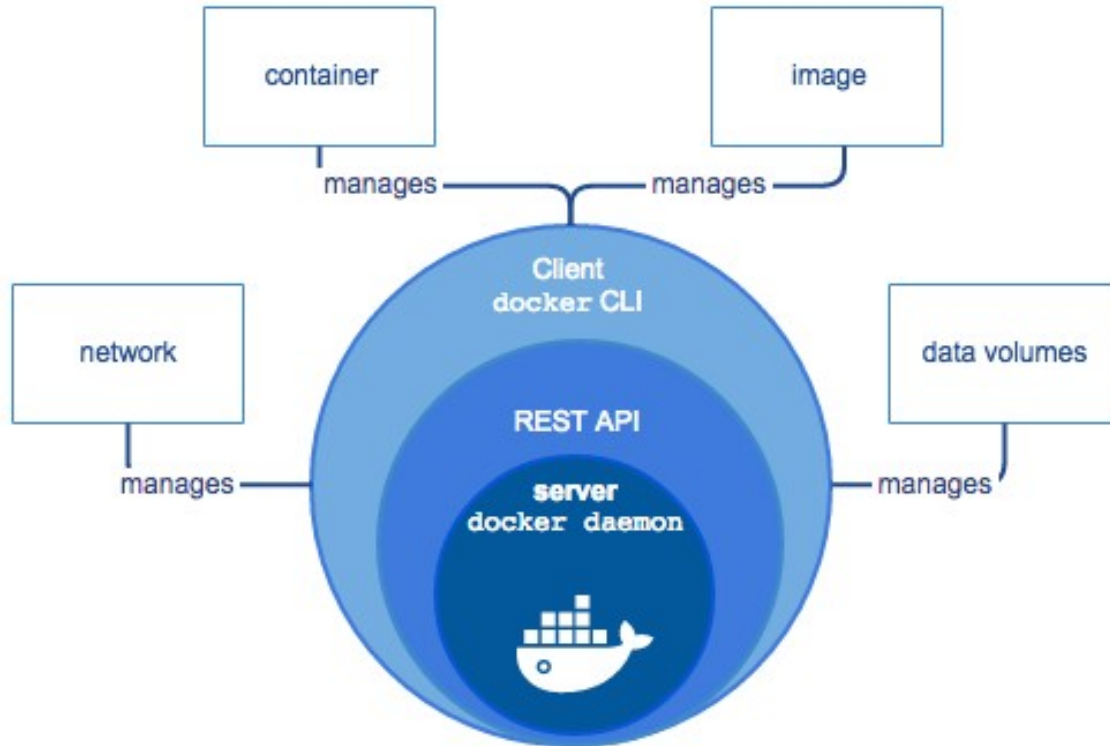
# АРХИТЕКТУРА DOCKER

## Docker Architecture



# КОМПОНЕНТЫ DOCKER ENGINE

---





# ЗАПУСК КОНТЕЙНЕРОВ

```
# docker run 50a986f614d5 # myhttpd:1.0  
^C
```

```
# docker run -d myhttpd:1.0  
9f761335efe268e9a82c4828d8f4be67b5824eb3266e8ba311343a7da45c67ff
```

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9f761335efe2	50a986f614d5	"/bin/sh -c 'httpd -...'"	5 seconds ago	Up 4 seconds	80/tcp	trusting_kilby

```
# docker run -P -d myhttpd:1.0  
74954ff14ec5e53ac9925bfd2873c654fe8978657764b4162ac494fc9afaab9f
```

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
74954ff14ec5	myhttpd:1.0	"/bin/sh -c 'httpd -...'"	9 seconds ago	Up 18 seconds	0.0.0.0:32768->80/tcp	fervent_noyce
9f761335efe2	myhttpd:1.0	"/bin/sh -c 'httpd -...'"	3 minutes ago	Up 3 minutes	80/tcp	trusting_kilby

```
# curl localhost:32768 # or curl <<VM_external_ip>>:32768  
my httpd container
```

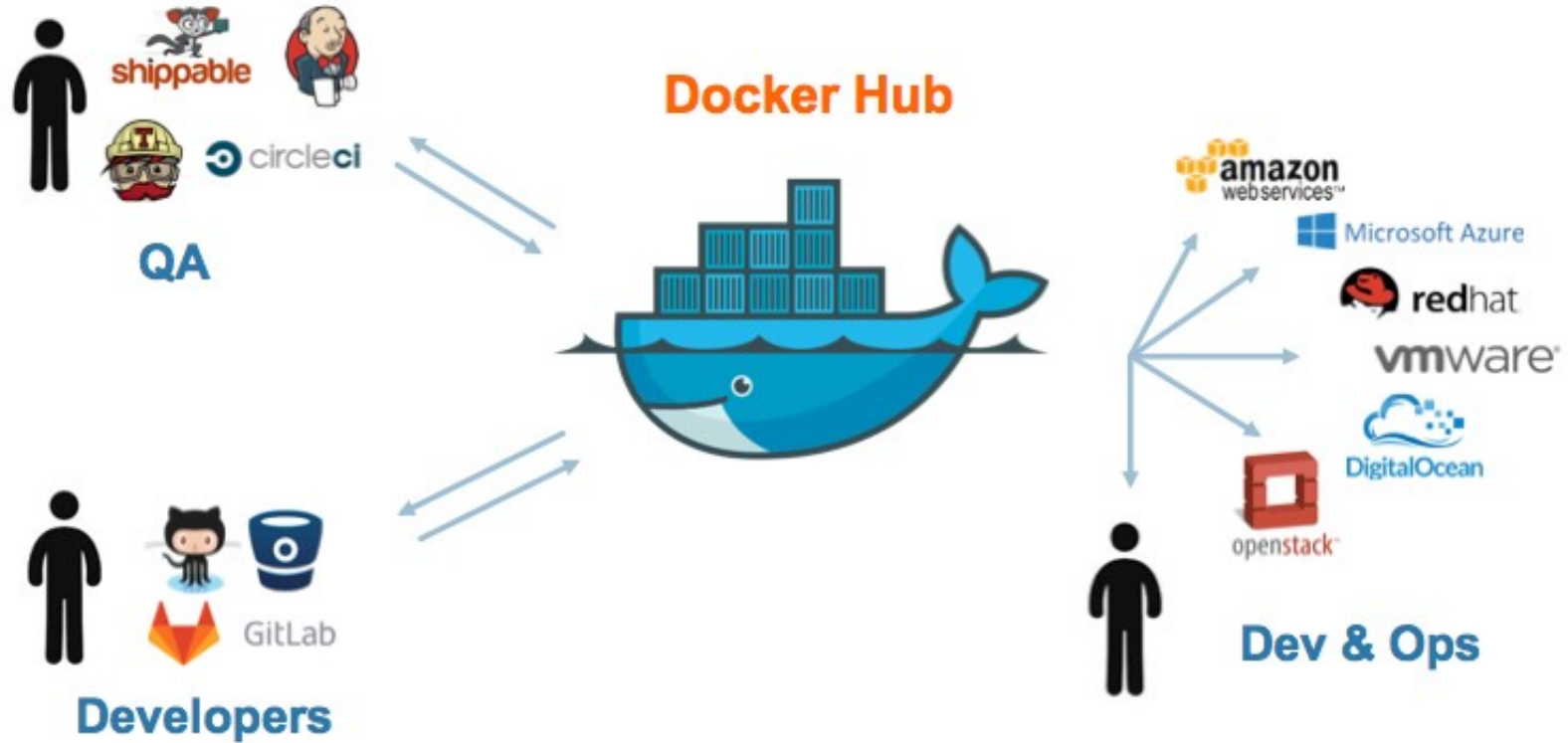
```
# docker run -d -p 8081:80 --name h8081 myhttpd:1.0  
fca7f4525bc618e7c503b73bfa680c055300e8b5c767d48e33669831e0bc5bec  
# docker run -d -p 127.0.0.1:8082:80 --name h8082 myhttpd:1.0  
014e5efa5ca90d9b7e50eebef3c7f020f08f0b5238f98420681ee348a4097829
```

```
# docker ps --format "table {{.Names}}\t{{.Image}}\t{{.ID}}\t{{.Ports}}" -n2
```

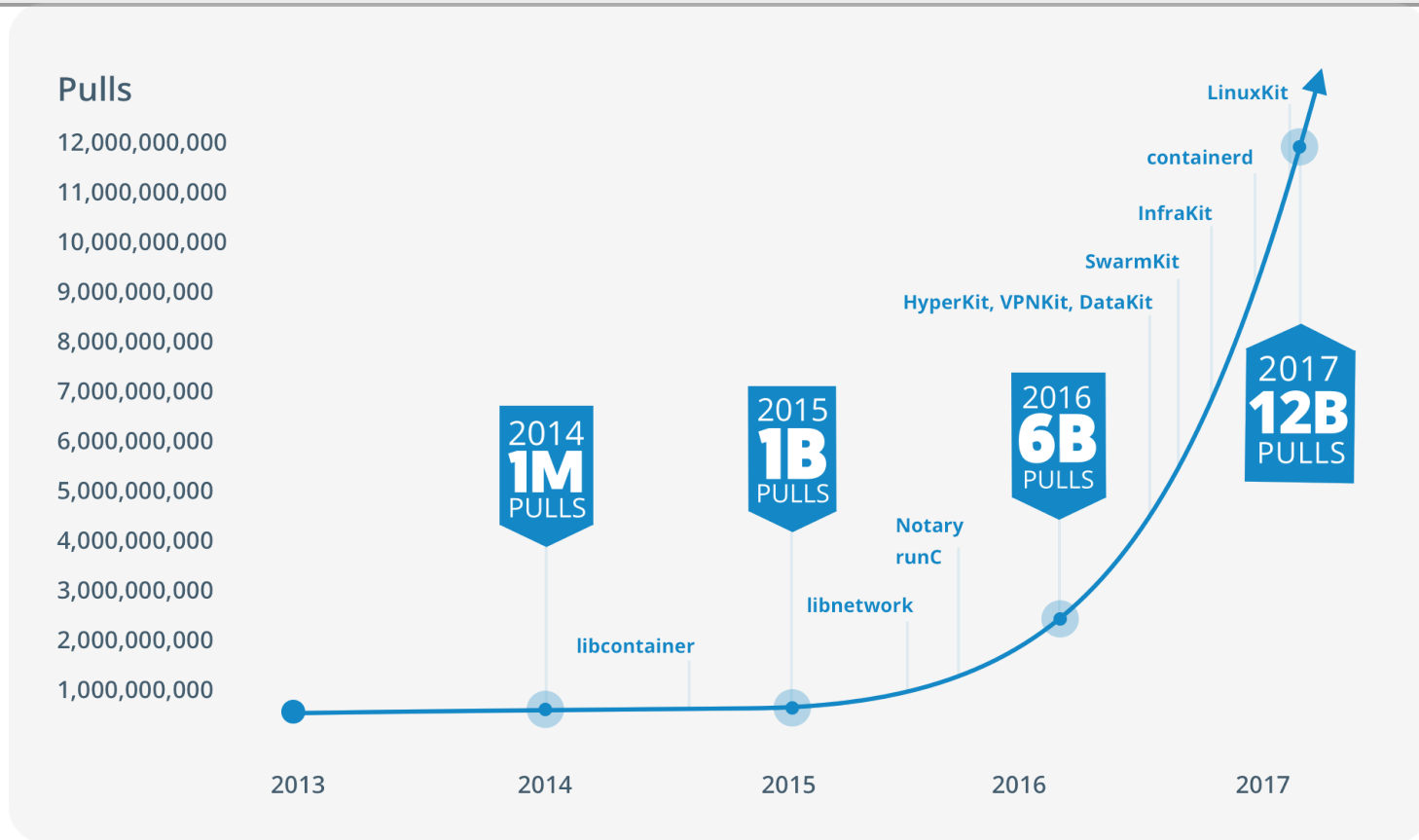
NAMES	IMAGE	CONTAINER ID	PORTS
h8081	myhttpd:1.0	fca7f4525bc6	0.0.0.0:8081->80/tcp
h8082	myhttpd:1.0	014e5efa5ca9	127.0.0.1:8082->80/tcp



# DOCKER REGISTRY – DOCKER HUB

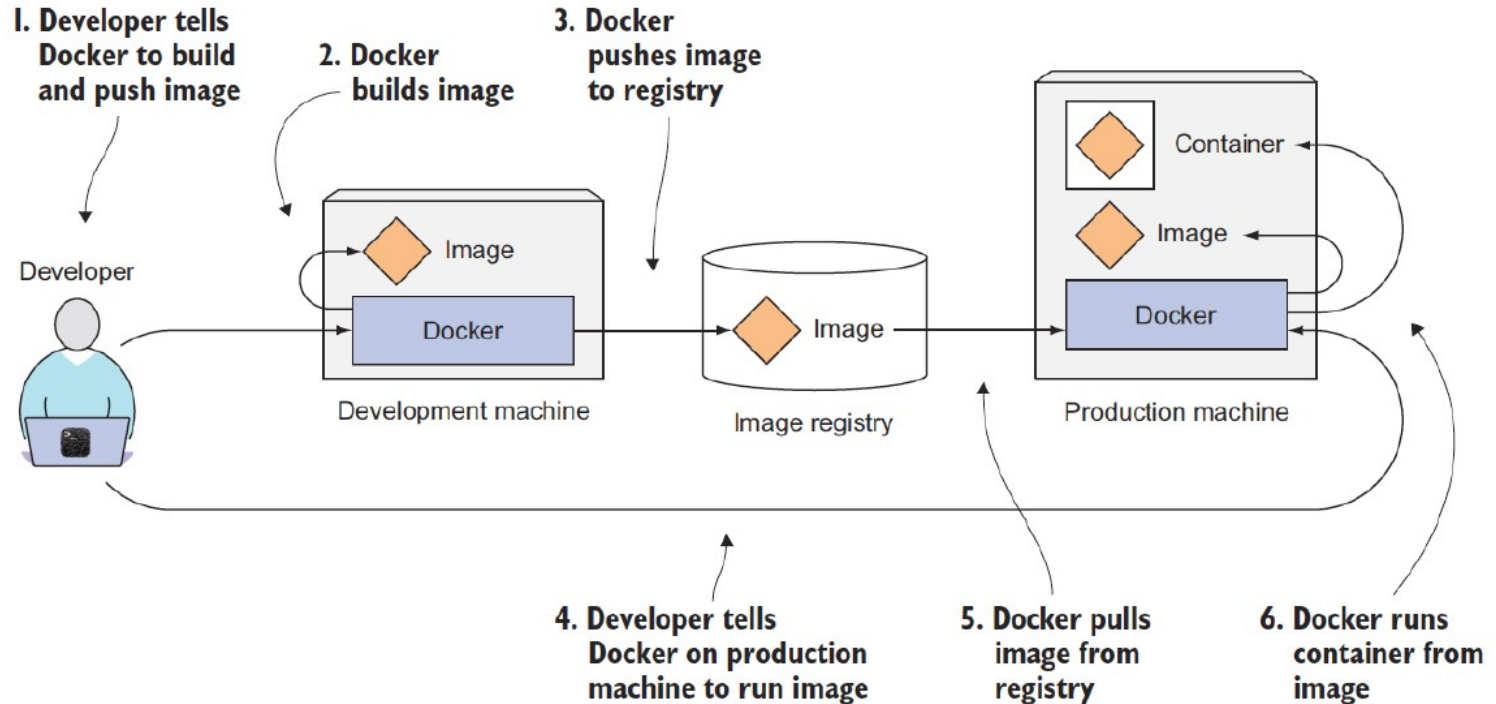


# ДИНАМИКА КОЛИЧЕСТВА СКАЧИВАНИЯ ОБРАЗОВ



# DOCKER WORKFLOW

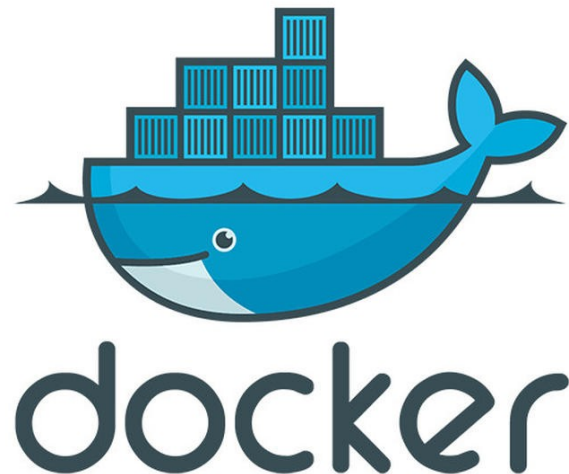
## Docker Workflow



# ДЛЯ ЧЕГО ИСПОЛЬЗОВАТЬ DOCKER?

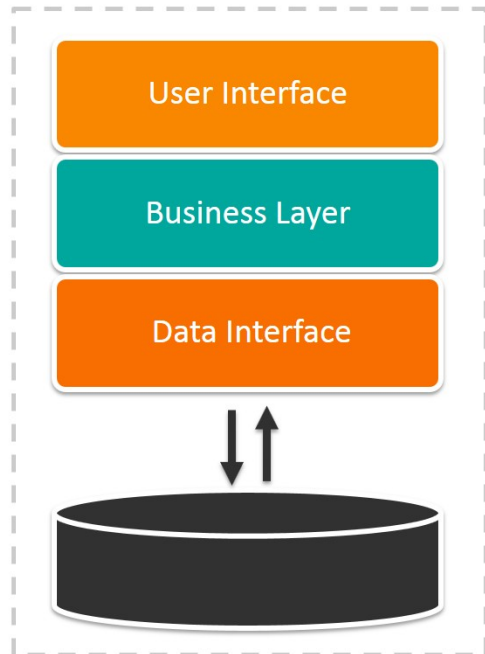
---

1. Быстрая доставка приложений
2. Простое развертывание и масштабирование приложений
3. Абстрагирование приложение от хоста
4. Унифицированный деплой

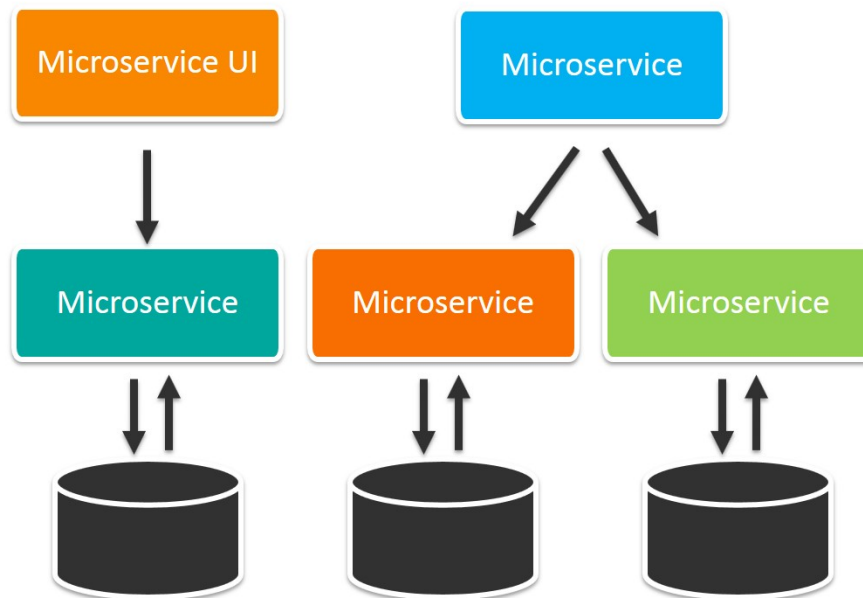


# ДЛЯ ЧЕГО ИСПОЛЬЗОВАТЬ DOCKER?

## Monolithic Architecture



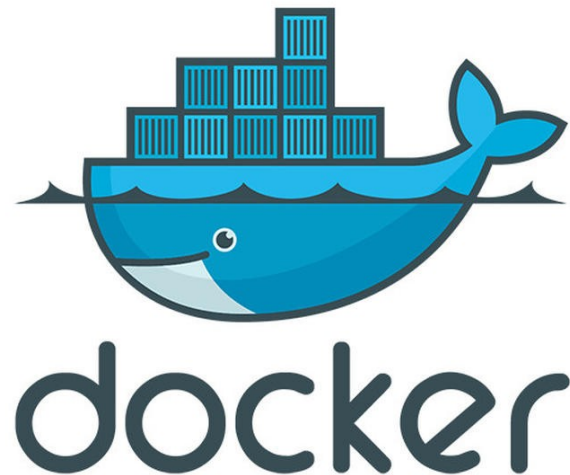
## Microservices Architecture



# НЕДОСТАТКИ DOCKER

---

1. Легкость использования
2. Сложность конфигурации
3. Сложность управления и поддержки
4. Незрелость



# РЕАЛЬНЫЙ ПРИМЕР ИСПОЛЬЗОВАНИЯ DOCKER



## CI/CD Platform

an orchestrated combination of enterprise delivery methodologies, architectural frameworks and development tools.



OPENSIFT

## Open Shift

Red Hat® OpenShift is a container application platform that brings Docker and Kubernetes to the enterprise.



## Kubernetes

Open-source system for automating deployment, scaling, and management of containerized applications.



docker

## Docker

Open-source containers technology which enables independence between applications and infrastructure in across the hybrid cloud.

AWS



Azure



Google



OpenStack

