

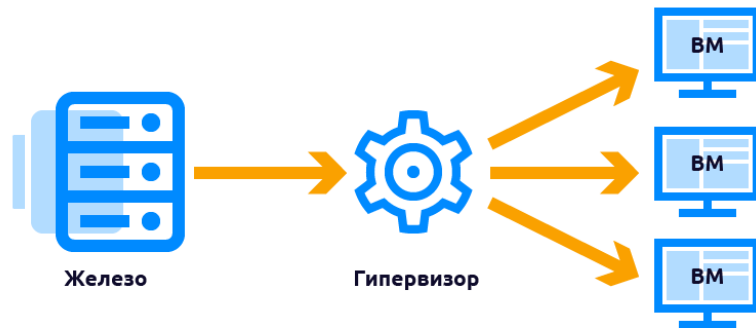
**NFE** Networking  
For  
Everyone

Docker

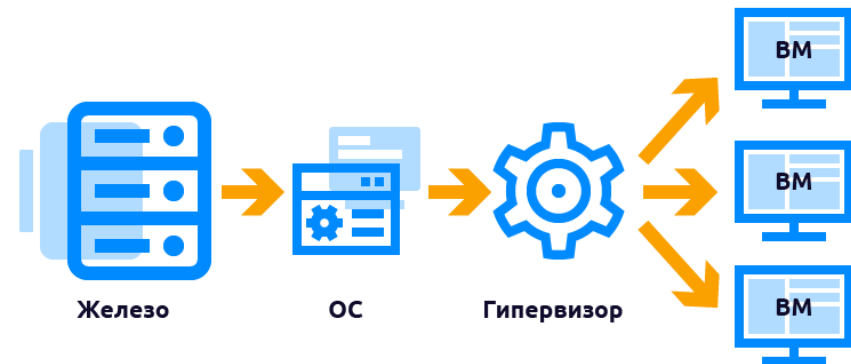


# Виртуализация

- Средство уплотнения окружений на одном и том же аппаратном сервера.
- VM (Виртуальная Машина) - среда эмуляции аппаратного обеспечения сервера.
- Гипервизор - ПО, обеспечивающее запуск и управление VM
  - Тип 1 - Самостоятельная ОС, работает на аппаратном уровне сервера, VMware vSphere ESXi, Microsoft Hyper-V
  - Тип 2 - ПО, работающее поверх ОС сервера VMware Workstation, Parallels Desktop, VirtualBox, UMT,



Тип 1



Тип 2

# Виртуализация

- Изолированность - VM не знают о присутствии друг друга и не имеют общих программных компонентов
  - Независимость - Сбой внутри VM не влияет на остальные VM
  - Использование ресурсов - VM используют физические ресурсы сервера
  - Гибкость - каждой VM требуется своя ОС, но она может быть разной
  - Доступность - изменения внутри VM сохраняются, VM можно переносить
- 
- Размер - экземпляры VM могут быть большими
  - Время - очень большое время создания и развертывания

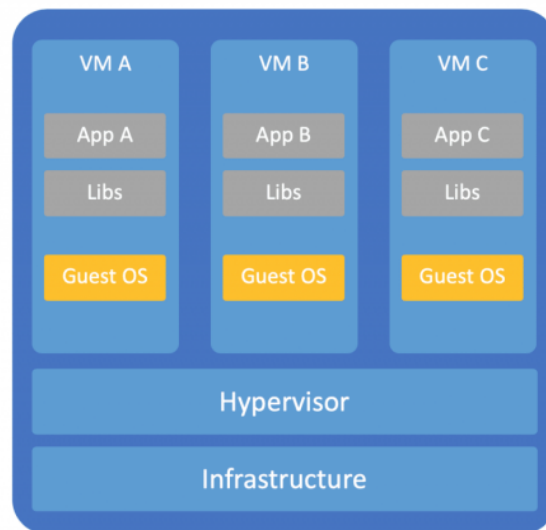
System Virtual Machine - эмуляция оборудования хоста для эмуляции всей операционной системы

Process Virtual Machine - aka Application Virtual Machine, эмуляция среды программирования для выполнения отдельного процесса (Java Virtual Machine)

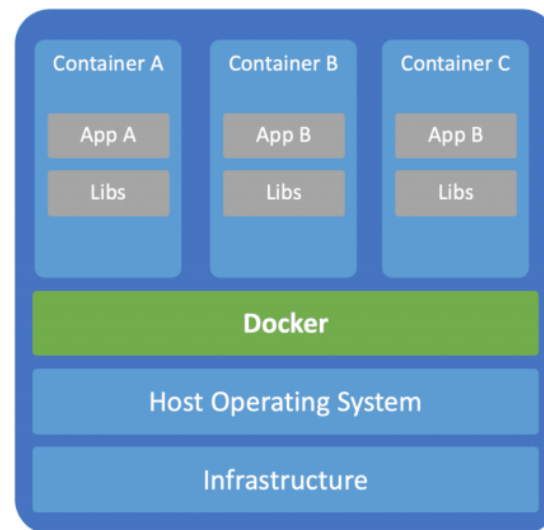
# Виртуализация и контейнеризация

Виртуализация vs. Контейнеризация - не конкуренты. Занимают разные ниши использования.

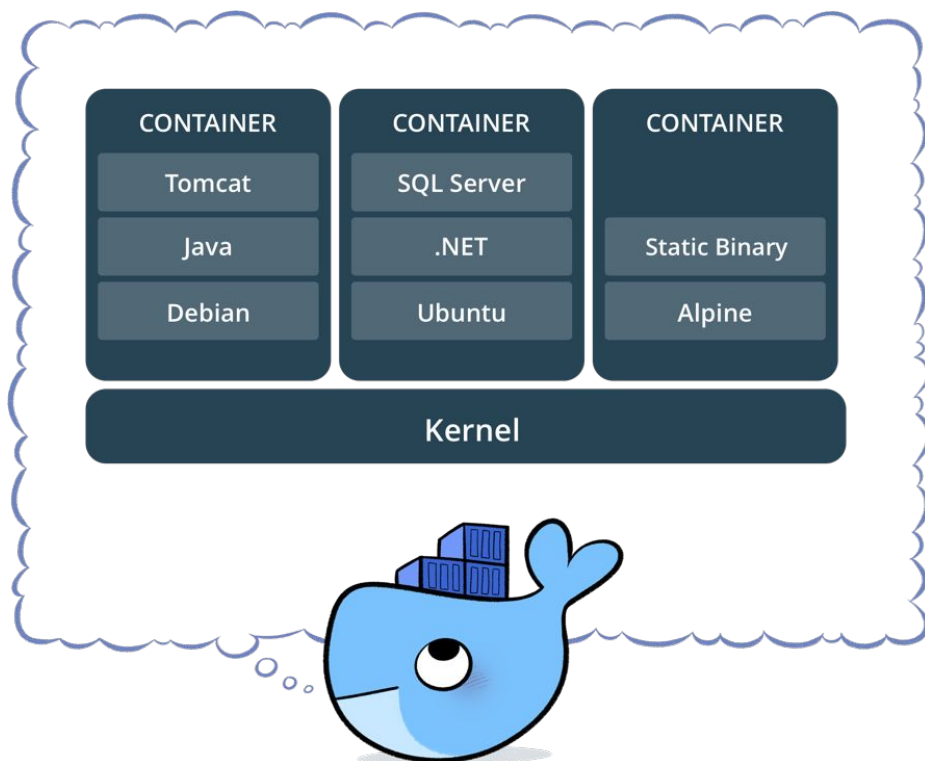
## Virtual Machines



## Container



# Контейнеризация



- Стандартизированный механизм упаковки приложений и зависимостей.
- Изоляция приложений друг от друга.
- Использование общего ядра OS.
- Возможность запуска практически на всех linux дистрибутивах.
- Быстрота развертывания.
- Версионность.
- Возможность повторного использования.
- Низкий time-to-market.

# Docker quick start

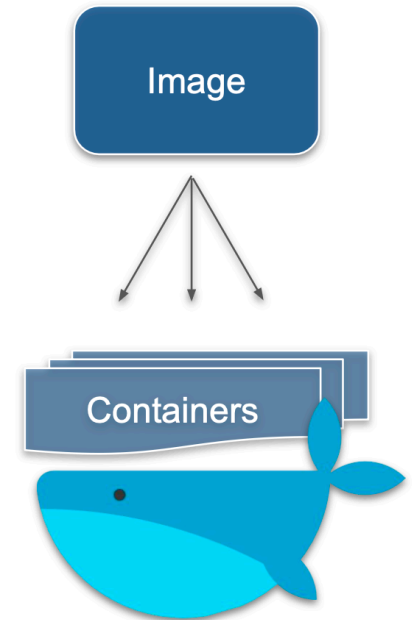
Docker позволяет создавать, доставлять и запускать приложения.

Самый базовый функционал это запустить (run) уже кем-то созданный образ.

Ключевые компоненты: образ и контейнер.

Образ это неизменяемая заготовка, шаблон для контейнера. Аналог шаблону VM или определению класса в ООП. Статичный объект, не процесс.

Контейнер - процесс (или множество) работающие в памяти. Созданный на базе образа. Динамический объект, может быть запущен, остановлен, поставлен на паузу.



# Docker quick start

```
alexigna@mbp-alexigna ~ % docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
70f5ac315c5a: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Пример запуска  
контейнера из образа

dockerfile для создания образа

Создание образа из dockerfile

Создание (запуск) контейнера

## Dockerfile

```
FROM ubuntu:latest

RUN apt update \
    && apt install -y nginx

CMD [ "nginx", "-g", "daemon off;" ]
```

```
alexigna@mbp-alexigna ~ % docker build -t mynginx .
[+] Building 0.0s (7/7) FINISHED
```

```
alexigna@mbp-alexigna ~ % docker run mynginx
```

# Docker



Docker - это платформа контейнеризации для автоматизации развертывания приложений в виде контейнеров, выполняемых в облаке или локальной среде.

Docker - это компания, которая разрабатывает эту технологию.

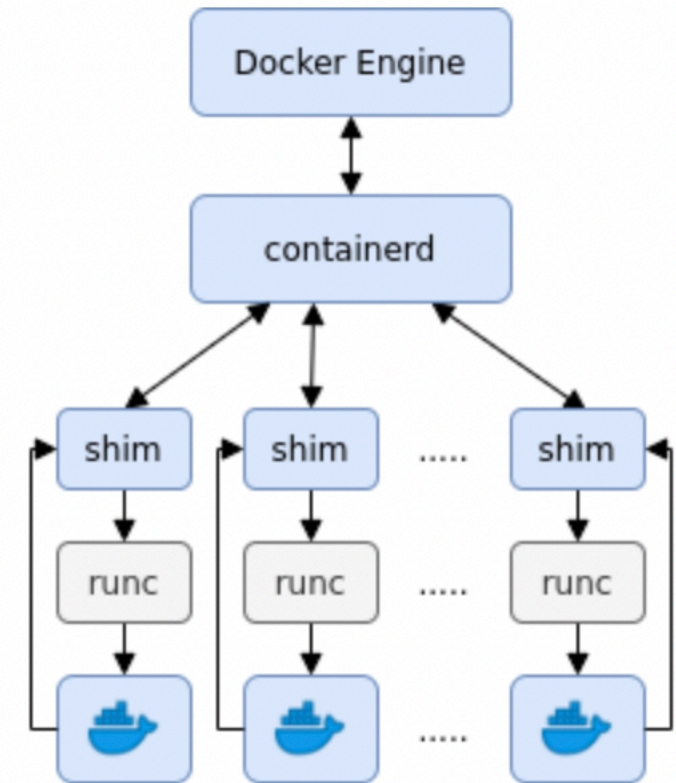


Компоненты Docker



# Архитектура Docker

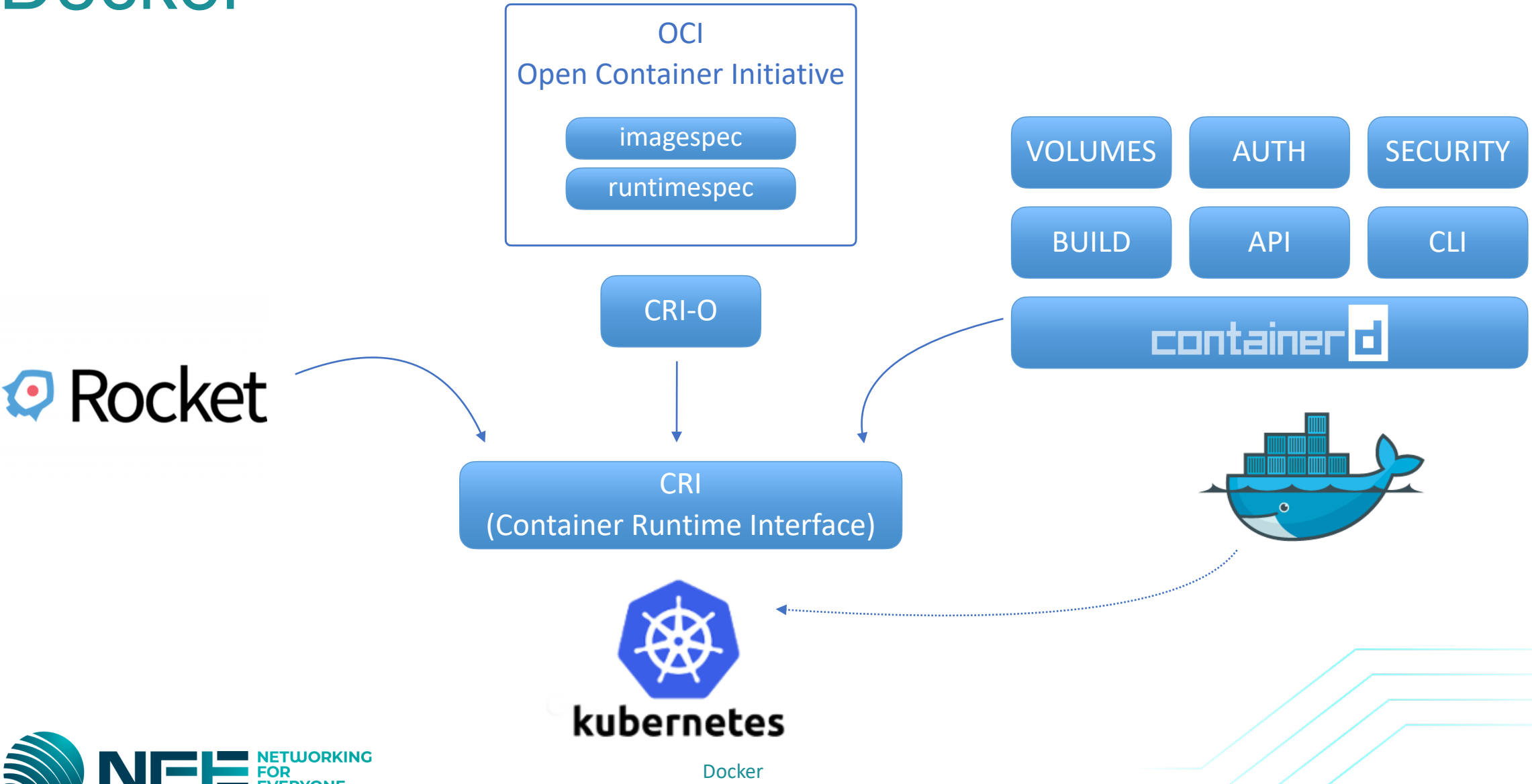
- *Docker* – взаимодействие с пользователем (dockerd, cli, api).
- *containerd* – high-level runtime для управления образами (скачивание и размещение в registry), управление сетью, volumes, метрики Prometheus, и прочее
- *runc* – low-level runtime для запуска контейнеров, умеет только создавать и запускать контейнеры, но не управлять образами
- *shim* – прослойка, позволяющая отключить containerD демон от контейнера после его запуска



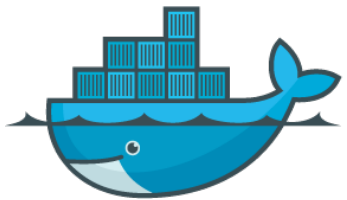
# Архитектура Docker

- Запуск cli команды, например `docker run`
- Клиент `docker-cli` конвертирует запрос и отправляет `dockerd` демону через `unix socket`
- `DockerD` демон слушает сокет `/var/run/docker.sock`, запросы можно отправлять и другими средствами (например `Portainer` для управления контейнерами/образами через `web`, или `curl`)
- При получении команды, демон вызывает `containerd` посредством `gRPC`
- `containerd` использует `runc` для взаимодействия с ядром ОС создавая контейнер
- Процесс контейнера стартует как дочерний для `runc`

# Docker

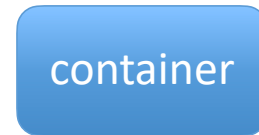


# Docker



OCI - спецификация

Описывает  
спецификацию на  
образы контейнеров и  
их запуск



**kubernetes**



cli/API/user tools

CRI - k8s API

Container Runtime

runC - low-level  
container runtime

Используем для  
создания, запуска,  
управления  
контейнерами

Описывает как k8s  
взаимодействует с  
различными  
container runtime

containerD - имплементация от Docker, через  
cri-plugins становится CRI-совместимой

CRI-O - изначально разработана как CRI-  
совместимый runtime (RedHat/IBM/Intel/etc)

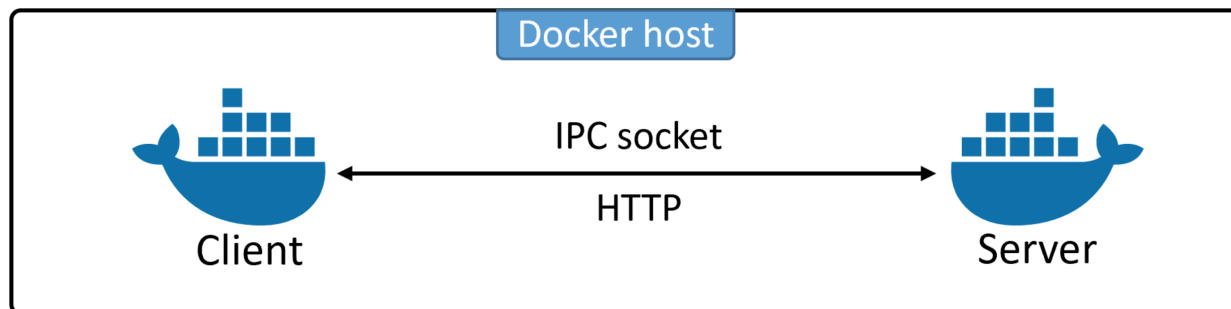
Создание и запуск  
контейнера



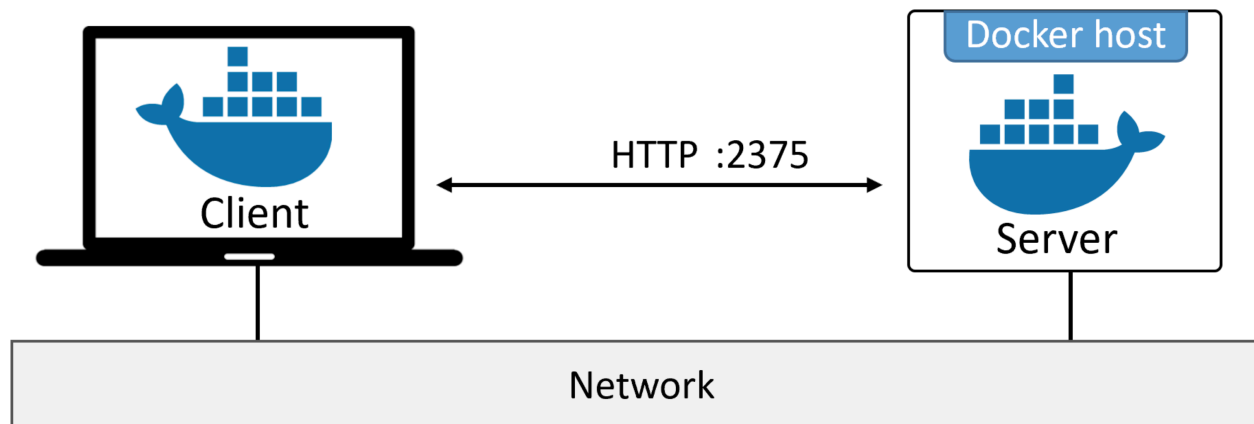
Docker

# Взаимодействие клиента и демона

По-умолчанию через IPC посредством HTTP

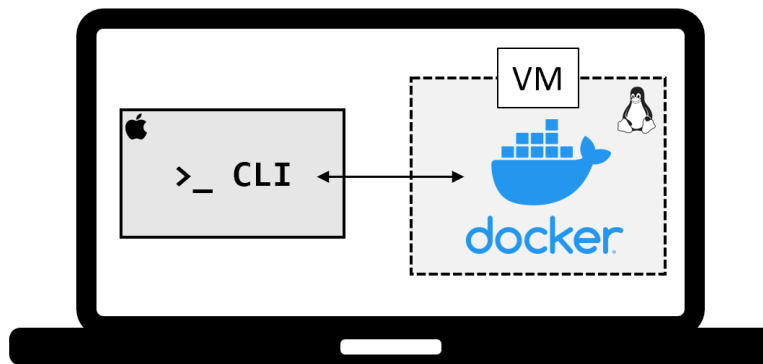


Есть возможность коммуникации через сеть



# Docker на MAC

- Docker Desktop
- Engine запускается на VM (скрыта от пользователя)
- В терминале можно вводить команды docker

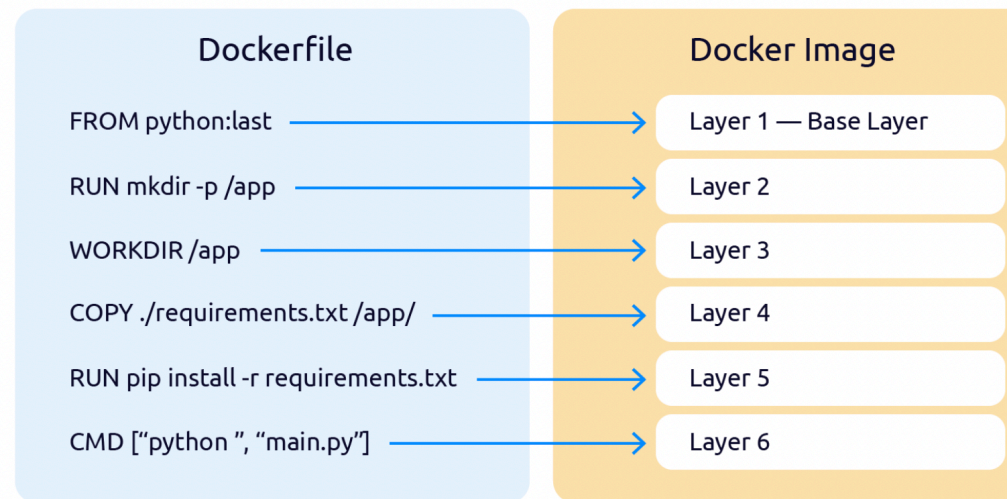


# Docker образы

- Образ – шаблон, из которого создаются контейнеры
- Ближайший аналог – шаблон VM
- Контейнер это не образ
- Образ содержит все необходимое для запуска приложения и само приложение
- Включает в себя
  - код приложения
  - зависимости
  - необходимые библиотеки ОС
- Если есть образ, то для запуска приложения нужен только компьютер (с установленным docker)

# Строение образов

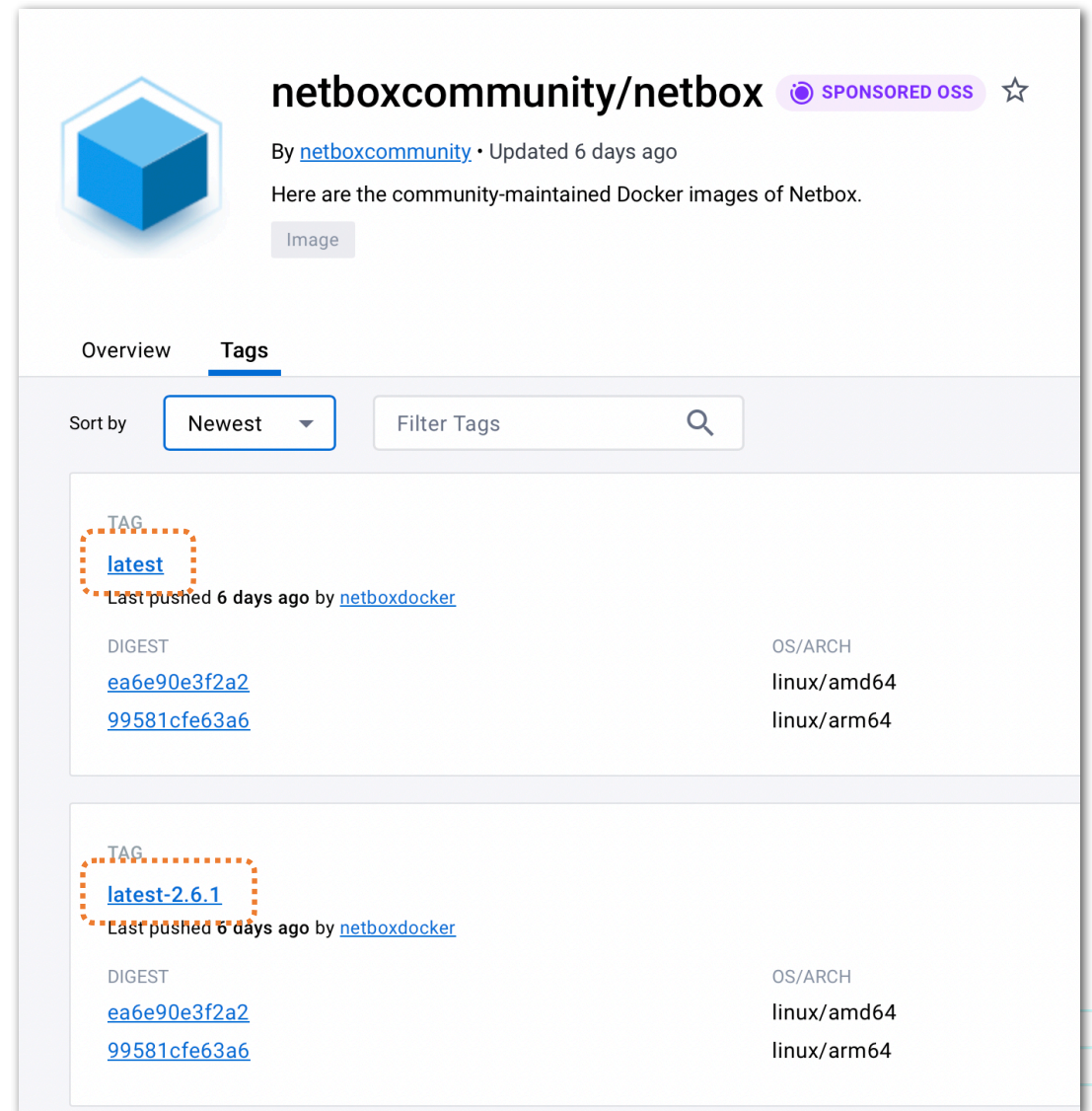
- Образ – набор слоев (уровней), которые доступны в режиме «только для чтения»
- Докер собирает эти слои и представляет единым объектом
- Каждая команда, используемая при сборке образа создает новый слой





# Тэги

- У каждого образа может быть тэг (и необязательно один)
- Если при загрузке из репозитория не указать тэг явно, Docker автоматически считает его равным *"latest"*



**netboxcommunity/netbox** SPONSORED OSS ☆

By [netboxcommunity](#) • Updated 6 days ago

Here are the community-maintained Docker images of Netbox.

Image

Overview **Tags**

Sort by **Newest** Filter Tags

TAG	DIGEST	OS/ARCH
<b>latest</b> Last pushed 6 days ago by <a href="#">netboxdocker</a>	<a href="#">ea6e90e3f2a2</a> <a href="#">99581cfe63a6</a>	linux/amd64 linux/arm64
<b>latest-2.6.1</b> Last pushed 6 days ago by <a href="#">netboxdocker</a>	<a href="#">ea6e90e3f2a2</a> <a href="#">99581cfe63a6</a>	linux/amd64 linux/arm64

# Мультиархитектура

- Docker поддерживает мультиархитектурные образы
- Образы для разных архитектур разные, теги одинаковые
- В зависимости от архитектуры сервера, где выполняется скачивание образа, загружается нужная архитектура
- Запуск контейнера из образа для “чужой” архитектуре возможен, но не гарантирован

```
alexigna@mbp-alexigna ~ % uname -m
arm64

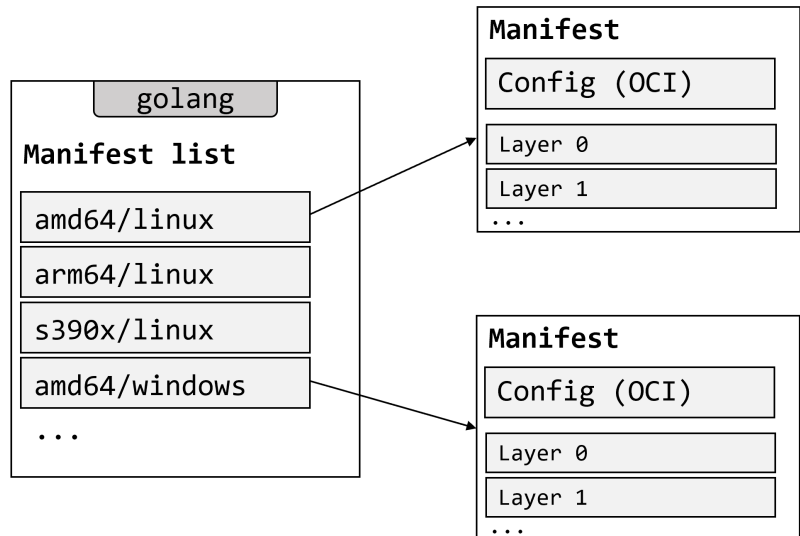
alexigna@mbp-alexigna ~ % docker pull busybox:1.27.1
1.27.1: Pulling from library/busybox
9e87eff13613: Pull complete
Digest: sha256:2605a2c4875ce5eb27a9f7403263190cd1af31e48a2044d400320548356251c4
Status: Downloaded newer image for busybox:1.27.1
docker.io/library/busybox:1.27.1

What's Next?
View summary of image vulnerabilities and recommendations → docker scout
quickview busybox:1.27.1

alexigna@mbp-alexigna ~ % docker run --rm -it busybox:1.27.1 /bin/sh
WARNING: The requested image's platform (linux/amd64) does not match the
detected host platform (linux/arm64/v8) and no specific platform was requested
/ # uname -m
x86_64
```

# Манифест

Манифест – json файл с информацией об образе Docker



```
alexigna@mbp-alexigna ~ % docker manifest inspect golang
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
  "manifests": [
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 1585,
      "digest": "sha256:832f2f74baa3e2b00ace688cb2fa934dffeade39f5b4c0cc8b1cff8d3fb084a0",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 1584,
      "digest": "sha256:9c18424fc460e122d4be7c997ea418437279265f31b8ed6a9f66f386dffdb84a",
      "platform": {
        "architecture": "arm",
        "os": "linux",
        "variant": "v5"
      }
    },
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 1584,
      "digest": "sha256:6cc298414ff826c8808bf1f6e932a27faaf048ee24890aa157b7ef7c2468d3e9",
      "platform": {
        "architecture": "arm",
```

# Образ для другой архитектуры

- buildx – плагин для docker-cli Расширяющий возможности создания (build) образов

<https://github.com/docker/buildx>

- Позволяет в том числе указать целевую архитектуру(ы), для которой нужно собрать образ

```
(venv) alexigna@mbp 09.solver % docker buildx build --platform linux/amd64 -t cr.yandex/crp01pvhgjo2ah62me/solver:latest .
[+] Building 3.4s (10/10) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 213B
=> [internal] load metadata for docker.io/library/python:3.10-slim
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:9d721d5a95cc5e18c663260ba1b63b339df3cb15958a6717b9d35baff227ba30
=> [internal] load build context
=> => transferring context: 64B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY ./main.py .
=> CACHED [4/5] COPY ./requirements.txt .
=> CACHED [5/5] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:4699ab36086a67fbf105b772d429643a2a51d4996669b0df8807facf2fb63795
=> => naming to cr.yandex/crp01pvhgjo2ah62me/solver:latest
(venv) alexigna@mbp 09.solver %
```

docker:desktop-linux	0.0s
	0.0s
	0.0s
	0.0s
	3.3s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s
	0.0s

# Работа с образами

`docker image ls` - просмотр образов на локальной машине

```
alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
```

`docker image pull <name>[:<tag>]` - скачать образ из репозитория

```
alexigna@mbp-alexigna ~ % docker image pull python:3.10-slim
3.10-slim: Pulling from library/python
3ae0c06b4d3a: Pull complete
d0a1460068e4: Pull complete
c9907b7bc879: Pull complete
f71221c8b230: Pull complete
e19dc5ab54ca: Pull complete
Digest: sha256:5ddc6ea17ec33701f8d5a6777a7b9953b7786eddeafb28b0d4e84011ebe6976b
Status: Downloaded newer image for python:3.10-slim
docker.io/library/python:3.10-slim
alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
python              3.10-slim    e0cbf388c936     4 weeks ago     170MB
```

# Работа с образами

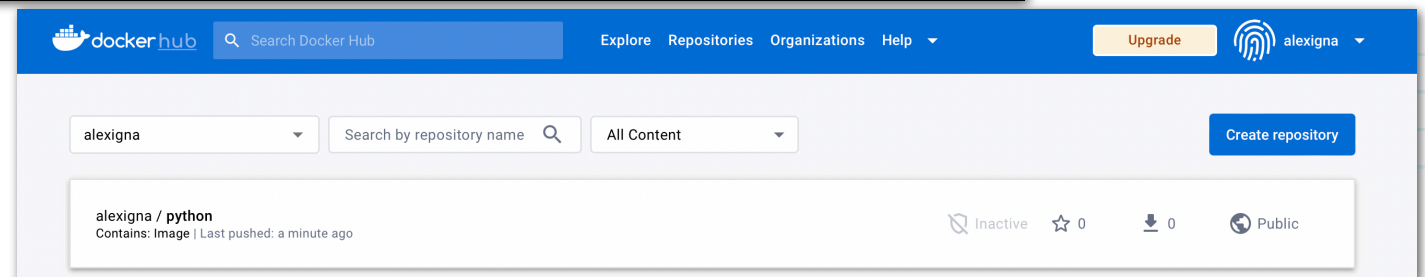
`docker image tag SRC_name[:tag] TGT_name[:tag]` - проставить теги образу

```
alexigna@mbp-alexigna ~ % docker tag python:3.10-slim python:prod
alexigna@mbp-alexigna ~ % docker tag python:3.10-slim alexigna/python:3.10-slim

alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY          TAG          IMAGE ID      CREATED       SIZE
alexigna/python     3.10-slim    e0cbf388c936  4 weeks ago  170MB
python              3.10-slim    e0cbf388c936  4 weeks ago  170MB
python              prod         e0cbf388c936  4 weeks ago  170MB
```

`docker image push <name>[:<tag>]` - загрузить образ в репозиторий

```
alexigna@mbp-alexigna ~ % docker push alexigna/python:3.10-slim
The push refers to repository [docker.io/alexigna/python]
e7902a4f70d5: Mounted from library/python
b9ba805c6ff3: Mounted from library/python
01dee7757be9: Mounted from library/python
1895af3a999f: Mounted from library/python
efd1965f1684: Mounted from library/python
3.10-slim: digest:
sha256:08444ac465c5dfedafb81e17ee8bf2cec192428592dbc2c435927d822687ae8f size: 1370
```



# Работа с образами

`docker image rm <name>:<tag>` - удалить образ

```
alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY TAG IMAGE ID CREATED SIZE
alexigna/python 3.10-slim e0cbf388c936 4 weeks ago 170MB
python 3.10-slim e0cbf388c936 4 weeks ago 170MB
python prod e0cbf388c936 4 weeks ago 170MB

alexigna@mbp-alexigna ~ % docker image rm python:prod
Untagged: python:prod

alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY TAG IMAGE ID CREATED SIZE
alexigna/python 3.10-slim e0cbf388c936 4 weeks ago 170MB
python 3.10-slim e0cbf388c936 4 weeks ago 170MB

alexigna@mbp-alexigna ~ % docker image rm python:3.10-slim
Untagged: python:3.10-slim
Untagged: python@sha256:5ddc6ea17ec33701f8d5a6777a7b9953b7786eddeafb28b0d4e84011ebe6976b

alexigna@mbp-alexigna ~ % docker image rm alexigna/python:3.10-slim
Untagged: alexigna/python:3.10-slim
Untagged: alexigna/python@sha256:08444ac465c5dfedafb81e17ee8bf2cec192428592dbc2c435927d822687ae8f
Deleted: sha256:e0cbf388c9365e0164711f3cf396943351048488a289861a54579e2b0084f2e3
Deleted: sha256:f855cea8e1e76b89dd94d4684a784746f38e89ec675c53281d1c9eb3c80bdafeb
Deleted: sha256:d61ab0865d9c6e21670d90a878dc16fd50768b5f95c2d62a2eb434af92086cc6
Deleted: sha256:827e93050eb79160850cf549607350c82e13fb17b15a7b941db7c5878f951bd8
Deleted: sha256:6f02f932314bb4cbdbe804d7277c5386fd8da564dc334b09a6122a37145fd6c5
Deleted: sha256:efd1965f1684506744544d66c57387a60bd89607480e2dbc89bf3e8a30081bc1
alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY TAG IMAGE ID CREATED SIZE
```

# Работа с образами

`docker image save/load` - сохраняет и загружает образ в/из локального файла

```
alexigna@mbp-alexigna ~ % docker image save python:3.10-slim | gzip > python-slim.tar.gz
alexigna@mbp-alexigna ~ % ls -lh python-slim.tar.gz
-rw-r--r--  1 alexigna  staff   48M Jul 18 12:08 python-slim.tar.gz

alexigna@mbp-alexigna ~ % docker image rm python:3.10-slim
alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY    TAG                IMAGE ID           CREATED          SIZE
python        3.10-slim         e0cbf388c936      4 weeks ago     170MB

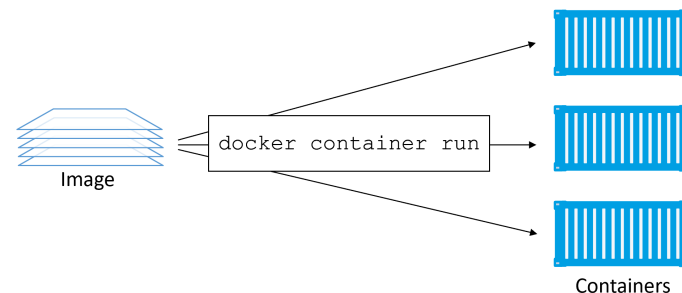
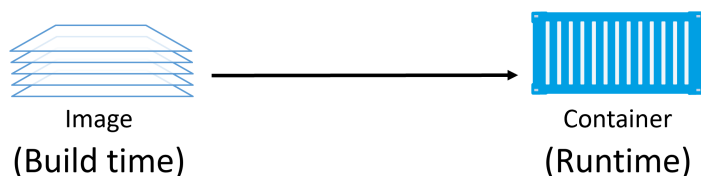
alexigna@mbp-alexigna ~ % docker load < python-slim.tar.gz
efd1965f1684: Loading layer [=====>] 100.1MB/100.1MB
1895af3a999f: Loading layer [=====>] 9.491MB/9.491MB
01dee7757be9: Loading layer [=====>] 53.71MB/53.71MB
b9ba805c6ff3: Loading layer [=====>] 5.12kB/5.12kB
e7902a4f70d5: Loading layer [=====>] 12.89MB/12.89MB
Loaded image: python:3.10-slim

alexigna@mbp-alexigna ~ % docker image ls | egrep "TAG|python"
REPOSITORY    TAG                IMAGE ID           CREATED          SIZE
python        3.10-slim         e0cbf388c936      4 weeks ago     170MB
```



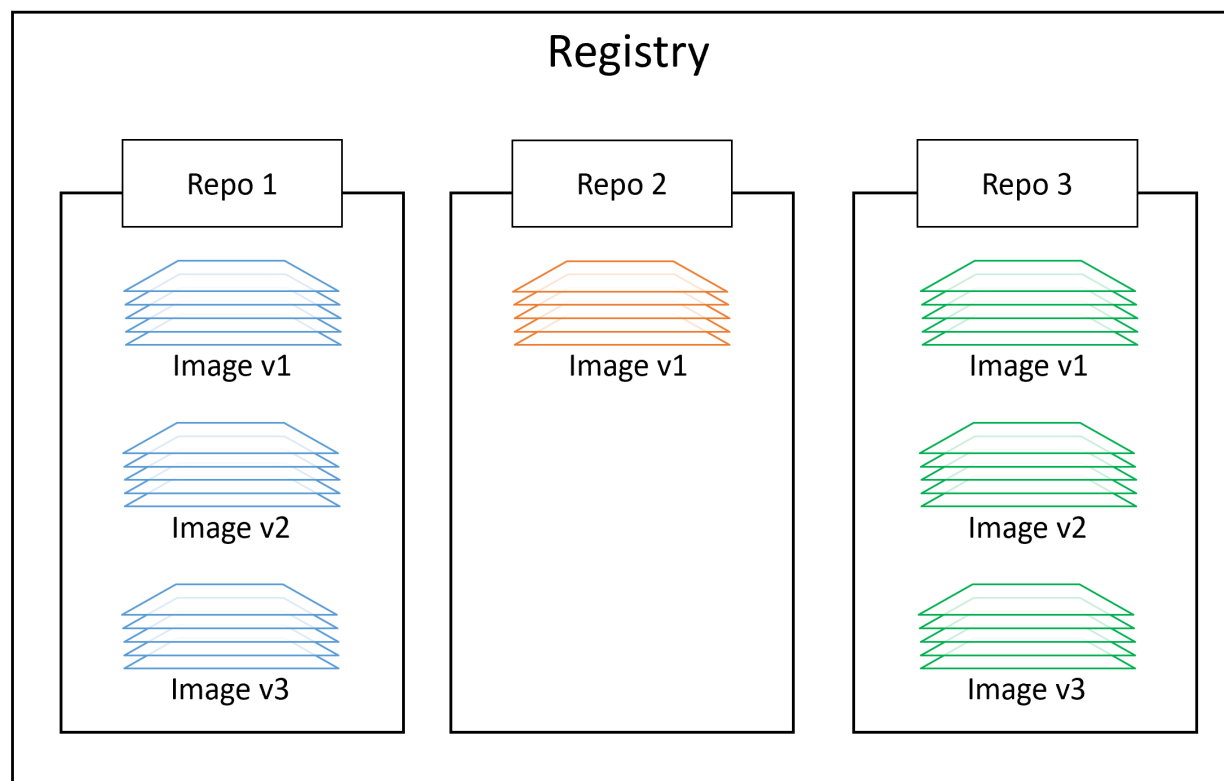
# Репозиторий

- Репозиторий предназначен для распространения образов
- Но можно не использовать репозиторий и передавать образы в виде файлов
- Наиболее известный реестр (объединение репозитория) – <https://hub.docker.com/>
- Репозиторий может предоставлять дополнительный функционал, например анализ на уязвимости.



# Реестр образов

Реестр состоит из одного или более репозиториев



# Загрузка образа в репозиторий

- Авторизоваться в docker registry ([hub.docker.com](https://hub.docker.com))

```
(venv) alexigna@mbp % docker login
Username: alexigna
Password:
Login Succeeded
```

- Или другом registry (например cr.yandex <https://cloud.yandex.ru/docs/container-registry/operations/authentication>)

```
(venv) alexigna@mbp % cat key.json | docker login --username json_key --password-stdin cr.yandex
Login Succeeded
```

- Назначить теги на загружаемые образы в соответствии с требованиям registry. Например:

- для [hub.docker.com](https://hub.docker.com) по формату `<username>/<имя Docker-образа>:<тег>`. Адрес репозитория не указывается, docker по-умолчанию считает, что это его собственный репозиторий
- для cr.yandex: `cr.yandex/<ID реестра>/<имя Docker-образа>:<тег>`

```
(venv) alexigna@mbp % docker tag solver:latest alexigna/solver:latest
(venv) alexigna@mbp % docker tag solver:latest cr.yandex/crp01pvhgjo2ah62me/solver:latest
```

- Загрузить образ в репозиторий (push)

```
(venv) alexigna@mbp % docker push alexigna/solver:latest
The push refers to repository [docker.io/alexigna/solver]
...
(venv) alexigna@mbp % docker push cr.yandex/crp01pvhgjo2ah62me/solver:latest
The push refers to repository [cr.yandex/crp01pvhgjo2ah62me/solver]
...
```

**NFE** Networking  
For  
Everyone

