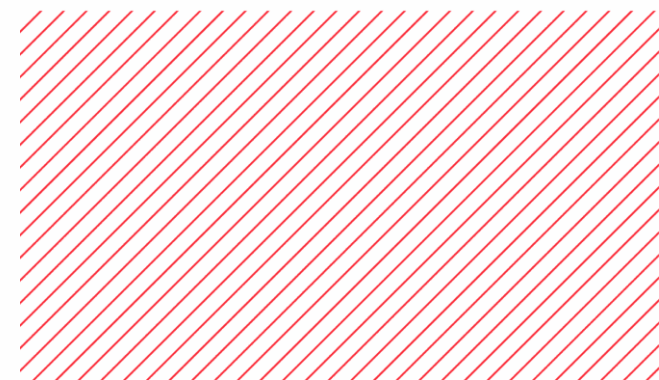




Docker и REST-services

Михаил Марюфич, MLE



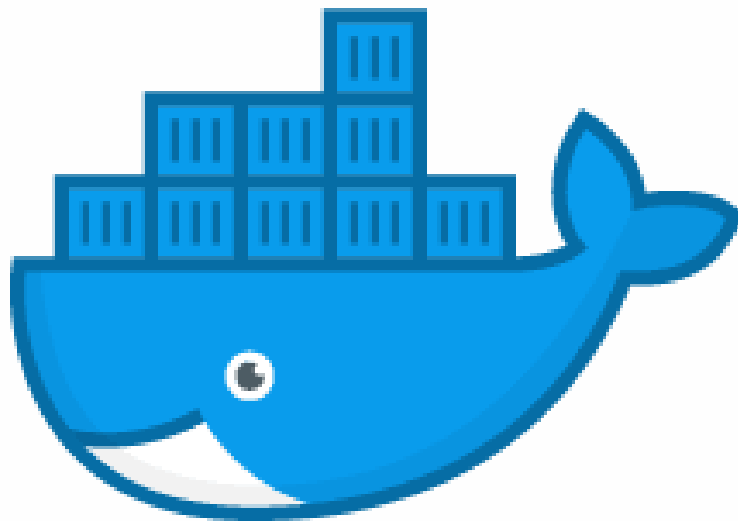


Содержание занятия

- Что такое docker и как он работает?
- Использование docker в МЛ проектах
 - для обучения и инференса (батч и онлайн)
- Rest services

DOCKER

Что такое docker?



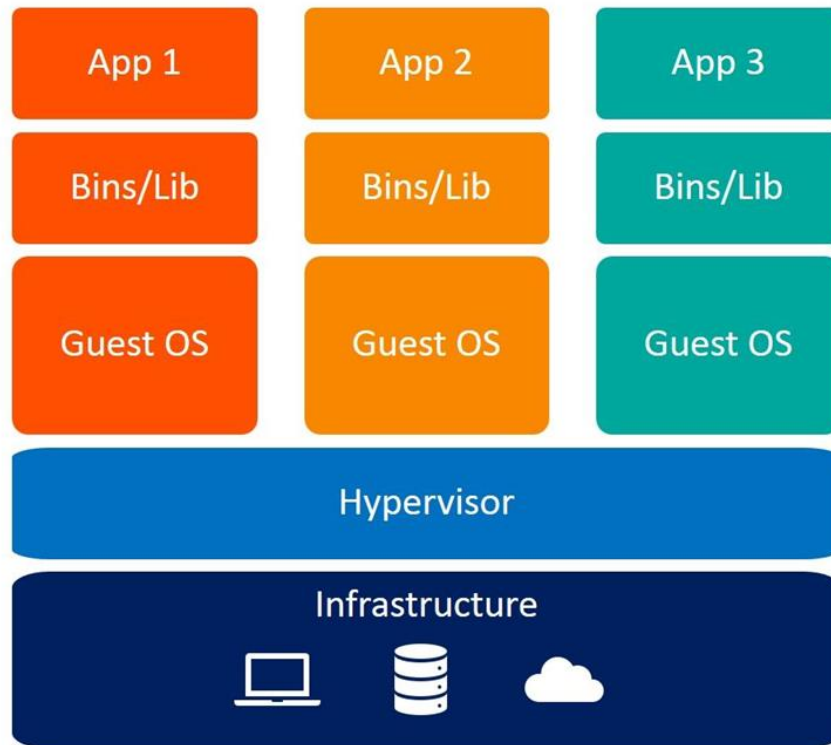
docker



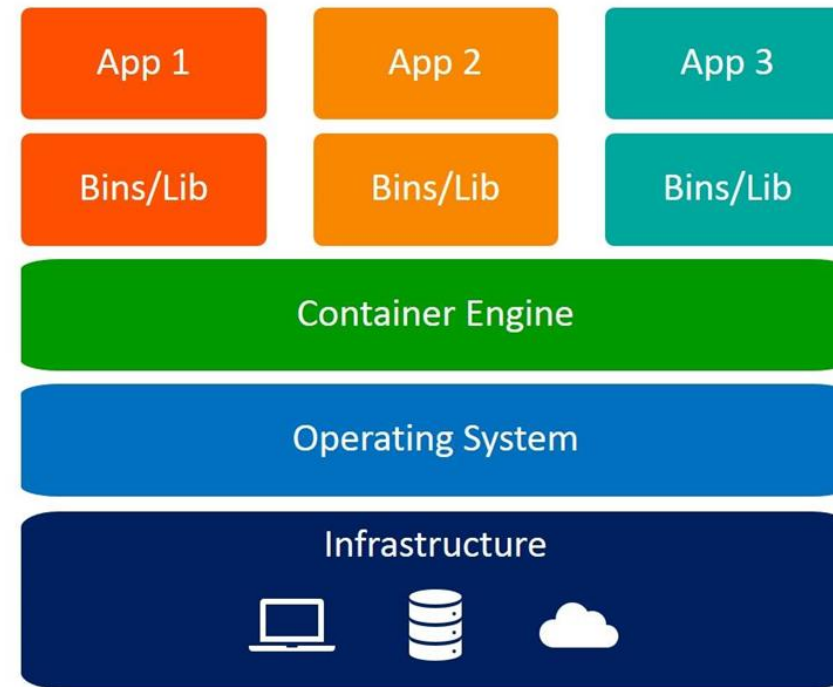
Чем docker не является

Контейнер — это НЕ виртуальная машина, а приложение и его зависимости упакованные в стандартизированное, изолированное, легковесное окружение.

Отличие от VM



Virtual Machines



Containers



Что мы можем запаковать в docker?

- App (your Java/Ruby/Go/... app)
- Libraries (libxml, wkhtmltopdf, ..)
- Services (postgresql, redis, ...)
- Tooling (sbt, ant, gems, eggs, ...)
- Frameworks&runtime (jre, ruby, ...)
- OS packages (libc6, tar, ps, bash, ...)

Что docker нам даёт?

- Абстракция от host-системы
- Легковесное изолированное окружение
- Простое управление зависимостями
- Стандартизация описания окружения, сборки, деплоймента





Docker

docker год появления



News

Images

Videos

Shopping

More

Settings

Tools

About 63,500 results (0.48 seconds)

Docker Платформа распределённых приложений

Разработчики:

Docker, Inc

Дата премьеры системы:

2013/03/13

Дата последнего релиза:

2019/11/18

Технологии:

PaaS - Platform As A Service - Бизнес-платформа как сервис, Виртуализация, Средства разработки приложений

Mar 13, 2013



Компоненты docker

- Daemon
- Client
- Registry



Daemon

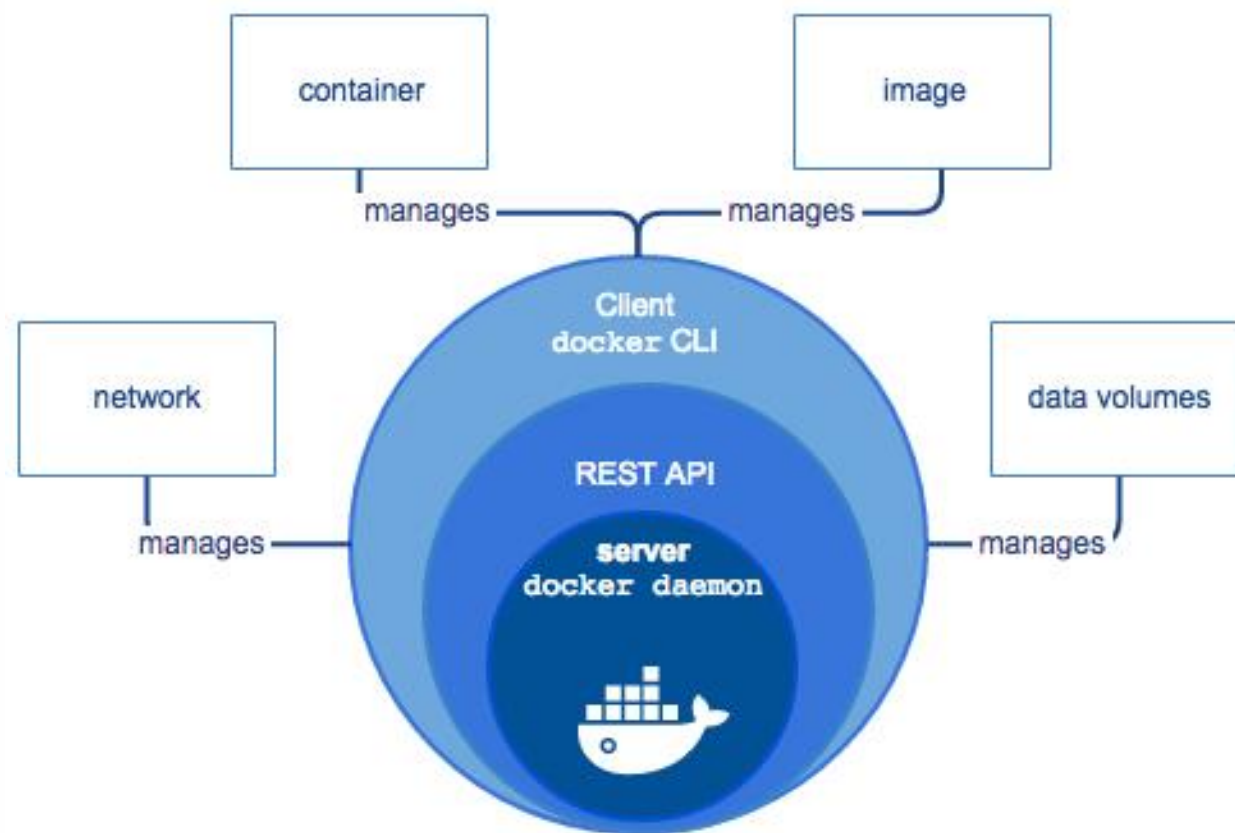
- Предоставляет API
- Управляет Docker-объектами
- Общается с другими docker daemon'ами
- Запускается на хост машине, где планируется запускать контейнеры



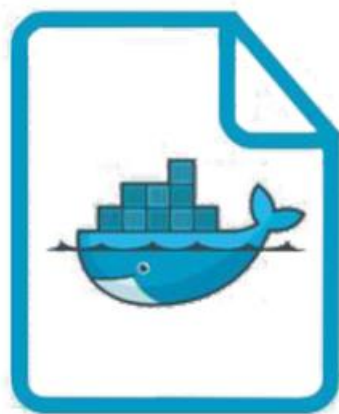
Client

- Принимает команды пользователя
- Общается по API с docker daemon'ом

Взаимосвязь

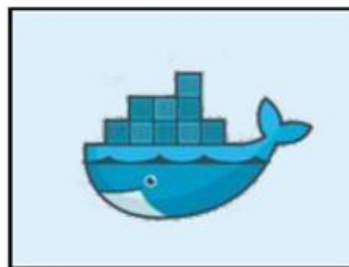


Сущности в docker



Dockerfile

Build



Docker
Image

Run



Docker
Container



Dockerfile

Текстовый файл с build инструкциями, которые декларативно описывают image

```
# our base image
FROM alpine:3.5

# Install python and pip
RUN apk add --update py2-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

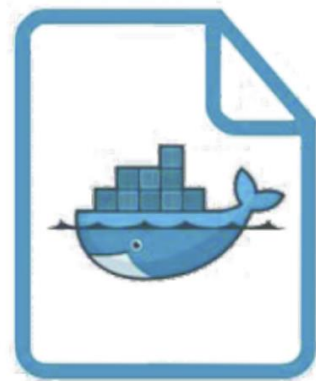
# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

Docker image

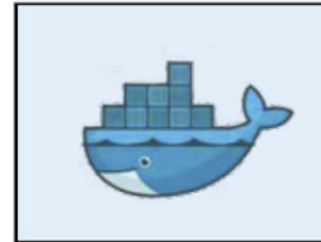
- неизменяемая сущность, **snapshot** контейнера
- состоит из **слоев(layers)**
- получается из **dockerfile** путем операции **build**

docker build -t imagename .



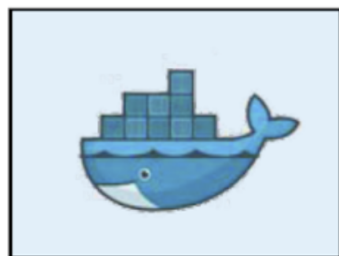
Dockerfile

Build



Docker
Image

Docker container



Docker
Image

Run →

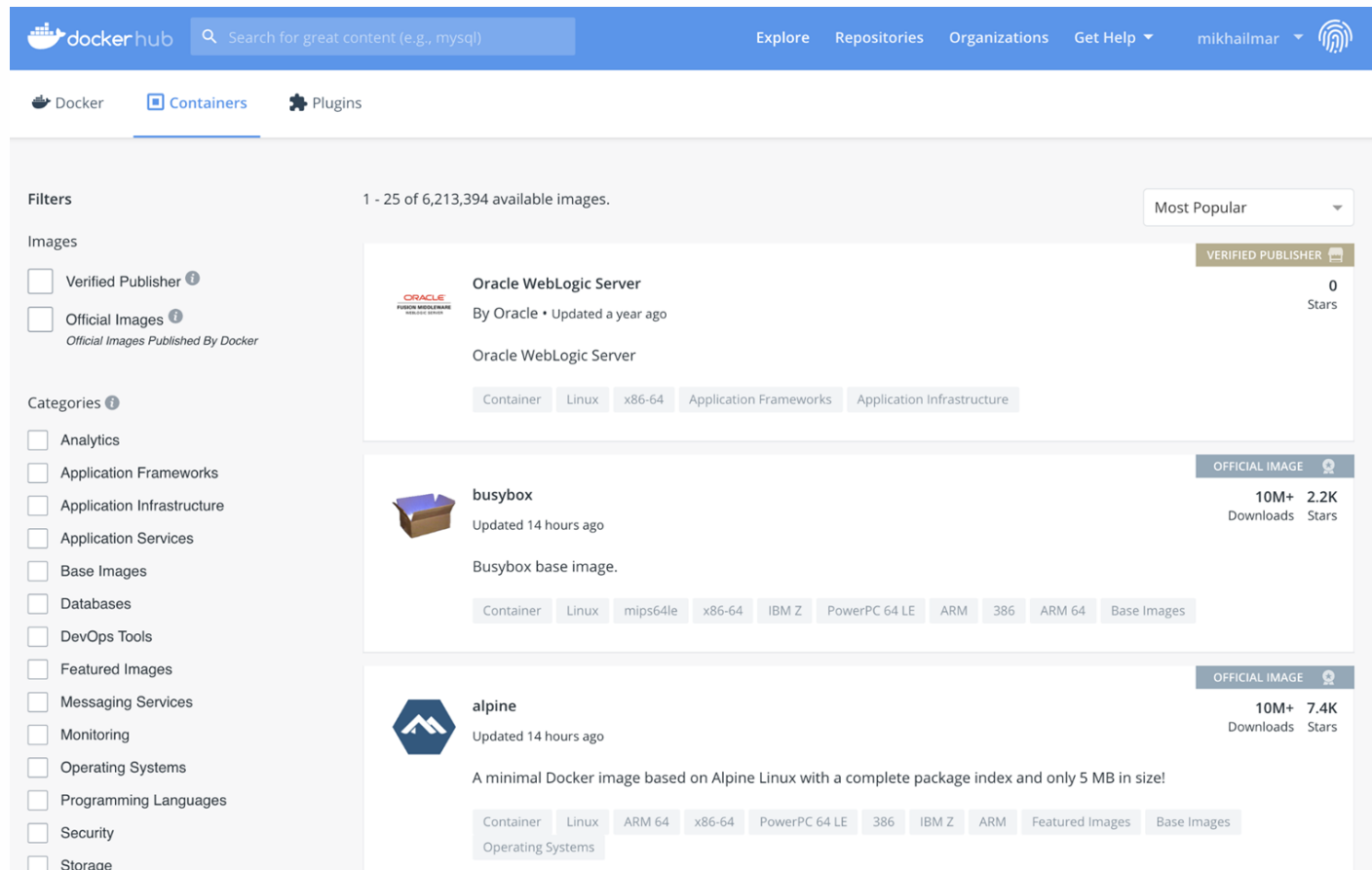


Docker
Container

DEMO — hello world

Docker registry

Место удаленного хранения образов



The screenshot displays the Docker Hub interface. At the top, there's a navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, Get Help, and a user profile. Below the navigation bar, there's a section for Filters and a list of images. The filters section includes checkboxes for Verified Publisher, Official Images, and various categories like Analytics, Application Frameworks, Application Infrastructure, Application Services, Base Images, Databases, DevOps Tools, Featured Images, Messaging Services, Monitoring, Operating Systems, Programming Languages, Security, and Storage. The list of images shows three entries: Oracle WebLogic Server, busybox, and alpine. Each entry includes the image name, a brief description, the number of stars, and a list of tags.

Filters 1 - 25 of 6,213,394 available images. Most Popular

Images

- ☐ Verified Publisher
- ☐ Official Images
Official Images Published By Docker

Categories

- ☐ Analytics
- ☐ Application Frameworks
- ☐ Application Infrastructure
- ☐ Application Services
- ☐ Base Images
- ☐ Databases
- ☐ DevOps Tools
- ☐ Featured Images
- ☐ Messaging Services
- ☐ Monitoring
- ☐ Operating Systems
- ☐ Programming Languages
- ☐ Security
- ☐ Storage

Oracle WebLogic Server VERIFIED PUBLISHER
By Oracle • Updated a year ago
Oracle WebLogic Server
Container Linux x86-64 Application Frameworks Application Infrastructure 0 Stars

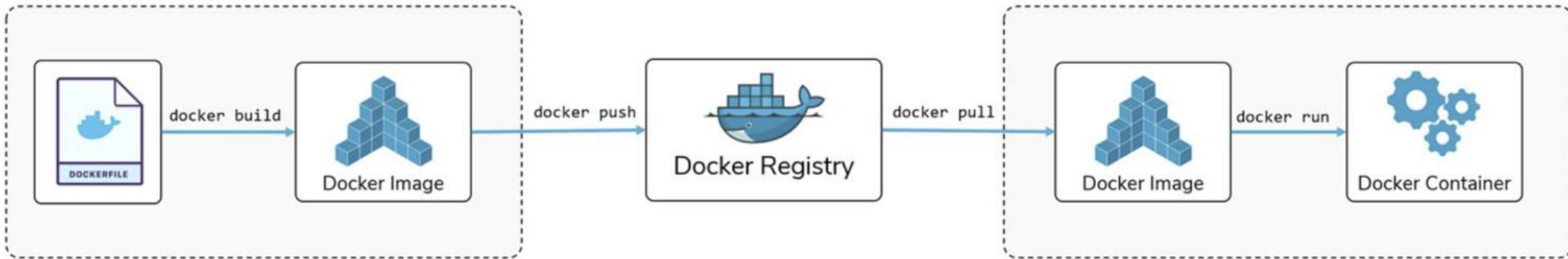
busybox OFFICIAL IMAGE
Updated 14 hours ago
Busybox base image.
Container Linux mips64le x86-64 IBM Z PowerPC 64 LE ARM 386 ARM 64 Base Images 10M+ Downloads 2.2K Stars

alpine OFFICIAL IMAGE
Updated 14 hours ago
A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!
Container Linux ARM 64 x86-64 PowerPC 64 LE 386 IBM Z ARM Featured Images Base Images Operating Systems 10M+ Downloads 7.4K Stars

<https://hub.docker.com/>

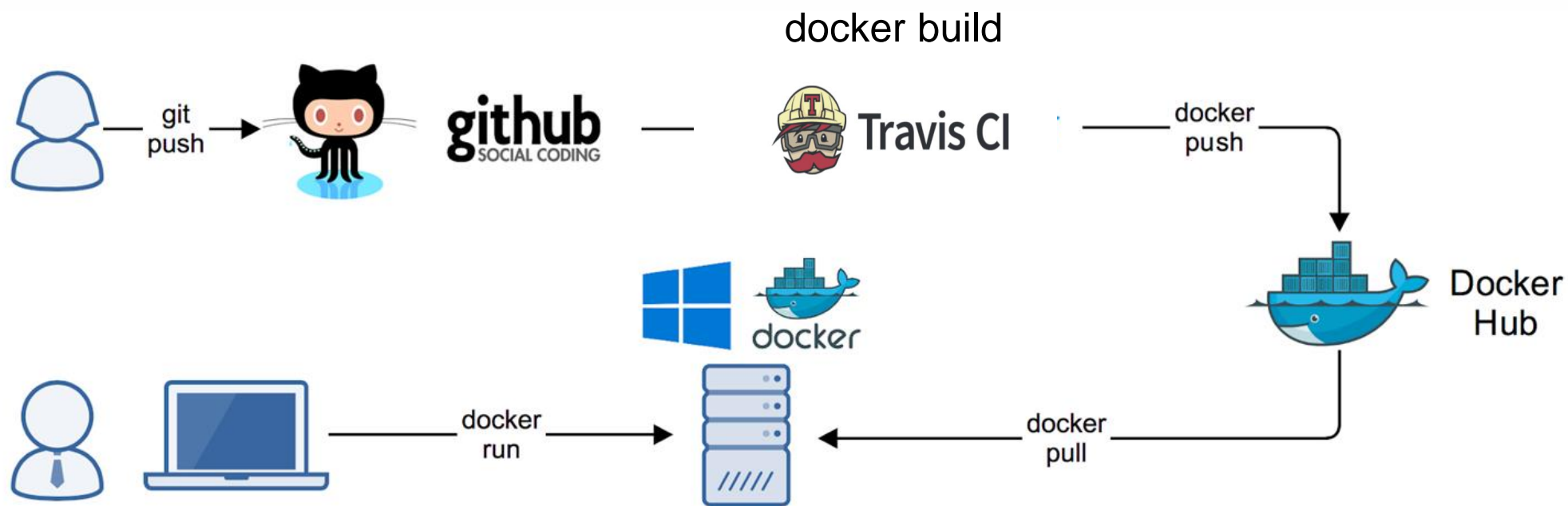
Docker pull/push

`docker push mikhailmar/helloworld:1.0`
`docker pull mikhailmar/helloworld:1.0`



DEMO — pull/push

Жизненный цикл контейнера





Часто используемые команды docker

docker build

docker run/ docker run -it --rm

docker pull/push

docker exec

docker attach

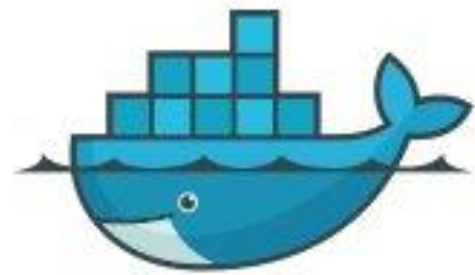
docker kill, stop, start

docker container ls

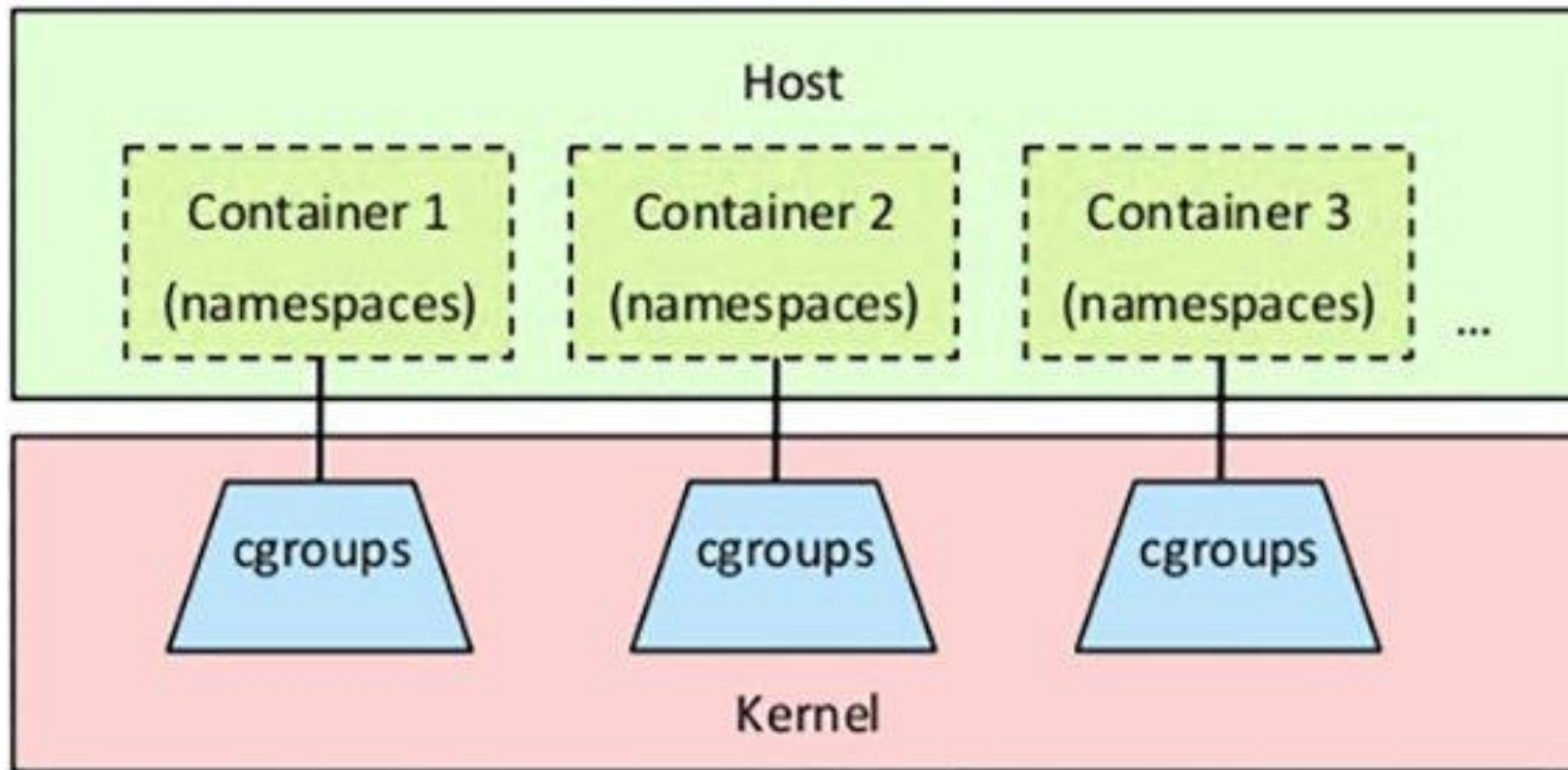
docker image ls

docker stats

Внутренности Docker



docker





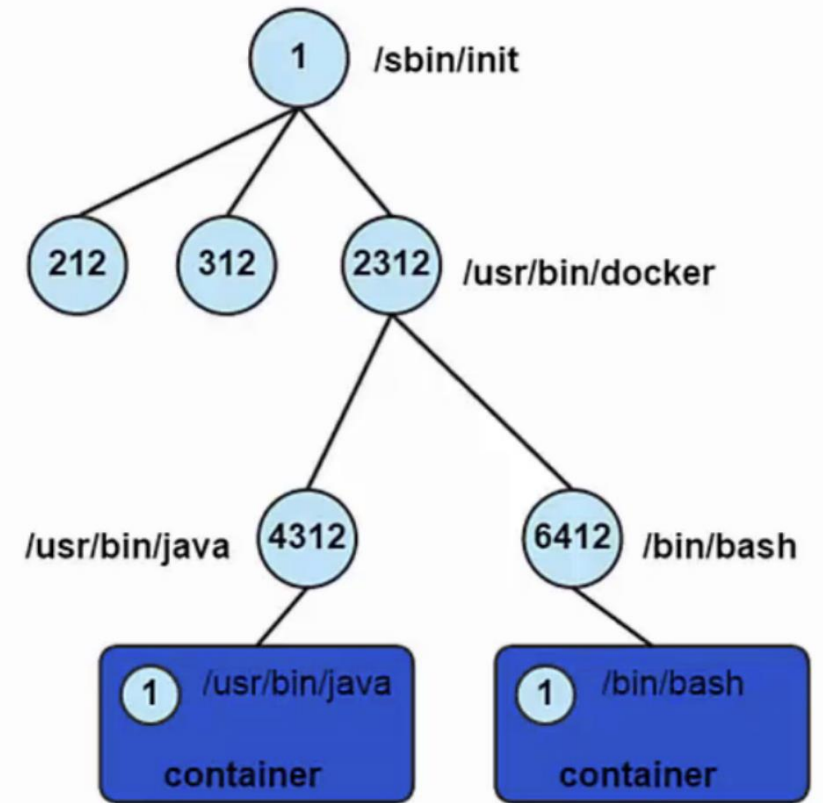
Namespaces

Механизм в unix, который отвечает за изоляцию

Namespace	Flag	Page	Isolates
Cgroup	CLONE_NEWCGROUP	cgroup_namespaces(7)	Cgroup root directory
IPC	CLONE_NEWIPC	ipc_namespaces(7)	System V IPC, POSIX message queues
Network	CLONE_NEWNET	network_namespaces(7)	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	mount_namespaces(7)	Mount points
PID	CLONE_NEWPID	pid_namespaces(7)	Process IDs
Time	CLONE_NEWTIME	time_namespaces(7)	Boot and monotonic clocks
User	CLONE_NEWUSER	user_namespaces(7)	User and group IDs
UTS	CLONE_NEWUTS	uts_namespaces(7)	Hostname and NIS domain name

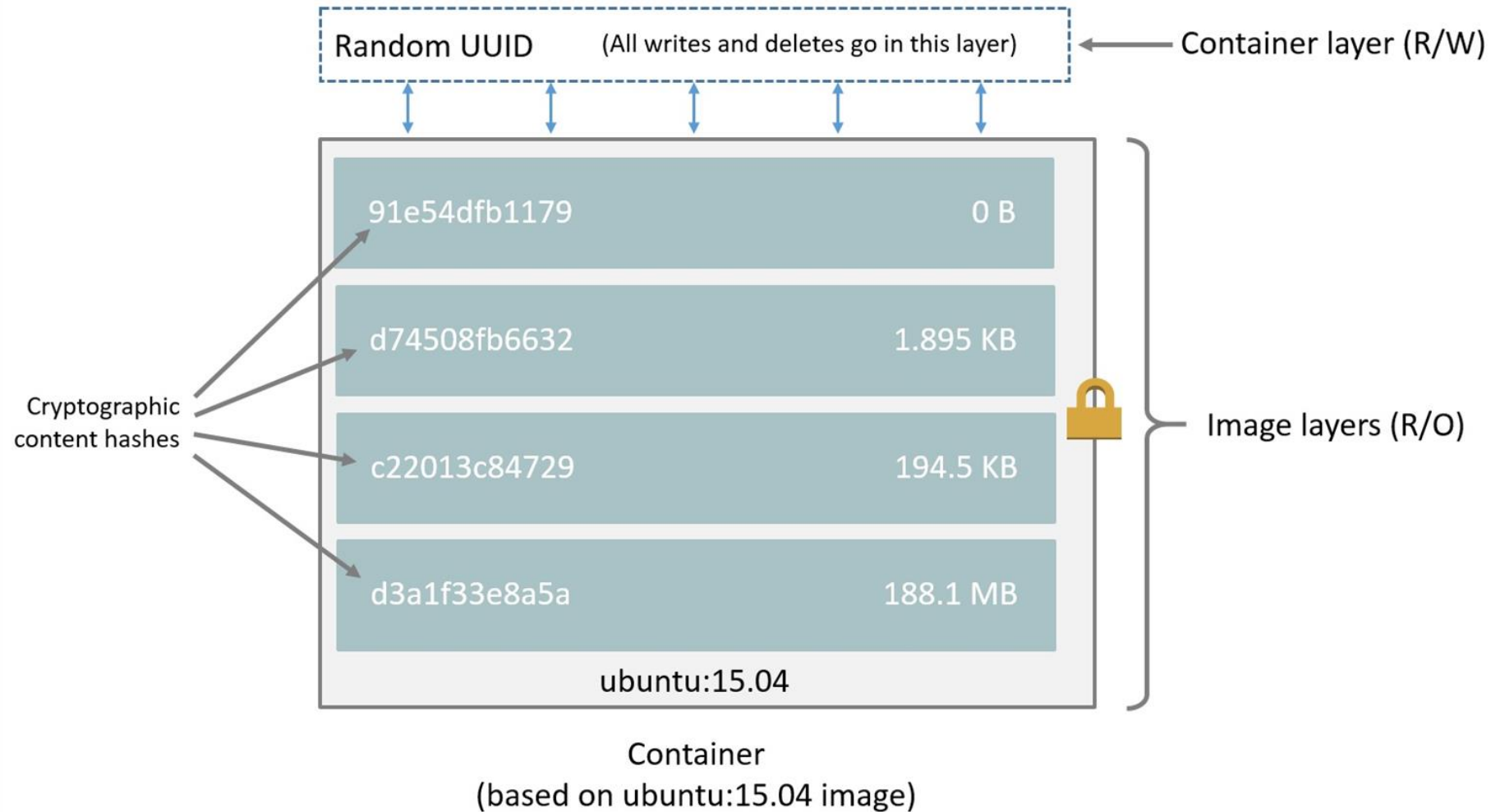
PID namespace

- Процессы внутри pid namespace'a видят только процессы из этого же namespace
- Каждый pid namespace имеет свою нумерацию процессов (начиная с 1)
- Когда процесс с pid 1 умирает, то умирает весь namespace PID namespace'ы могут быть вложенными



DEMO — PID

UnionFS



UnionFS

91e54dfb1179	0 B
d74508fb6632	1.895 KB
c22013c84729	194.5 KB
d3a1f33e8a5a	188.1 MB
ubuntu:15.04	

Image

RUN ...

RUN ...

ADD/COPY



Cgroups

cgroups - используется для управления тем, сколько и каких ресурсов может использовать контейнер

- `blkio` — устанавливает лимиты на чтение и запись с блочных устройств;
- `cpuacct` — генерирует отчёты об использовании ресурсов процессора;
- `cpu` — обеспечивает доступ процессов в рамках контрольной группы к CPU;
- `cpuset` — распределяет задачи в рамках контрольной группы между процессорными ядрами;
- `devices` — разрешает или блокирует доступ к устройствам;
- `freezer` — приостанавливает и возобновляет выполнение задач в рамках контрольной группы
- `hugetlb` — активирует поддержку больших страниц памяти для контрольных групп;
- `memory` — управляет выделением памяти для групп процессов;
- `net_cls` — помечает сетевые пакеты специальным тэгом, что позволяет идентифицировать пакеты, порождаемые определённой задачей в рамках контрольной группы;
- `netprio` — используется для динамической установки приоритетов по трафику;
- `pids` — используется для ограничения количества процессов в рамках контрольной группы.

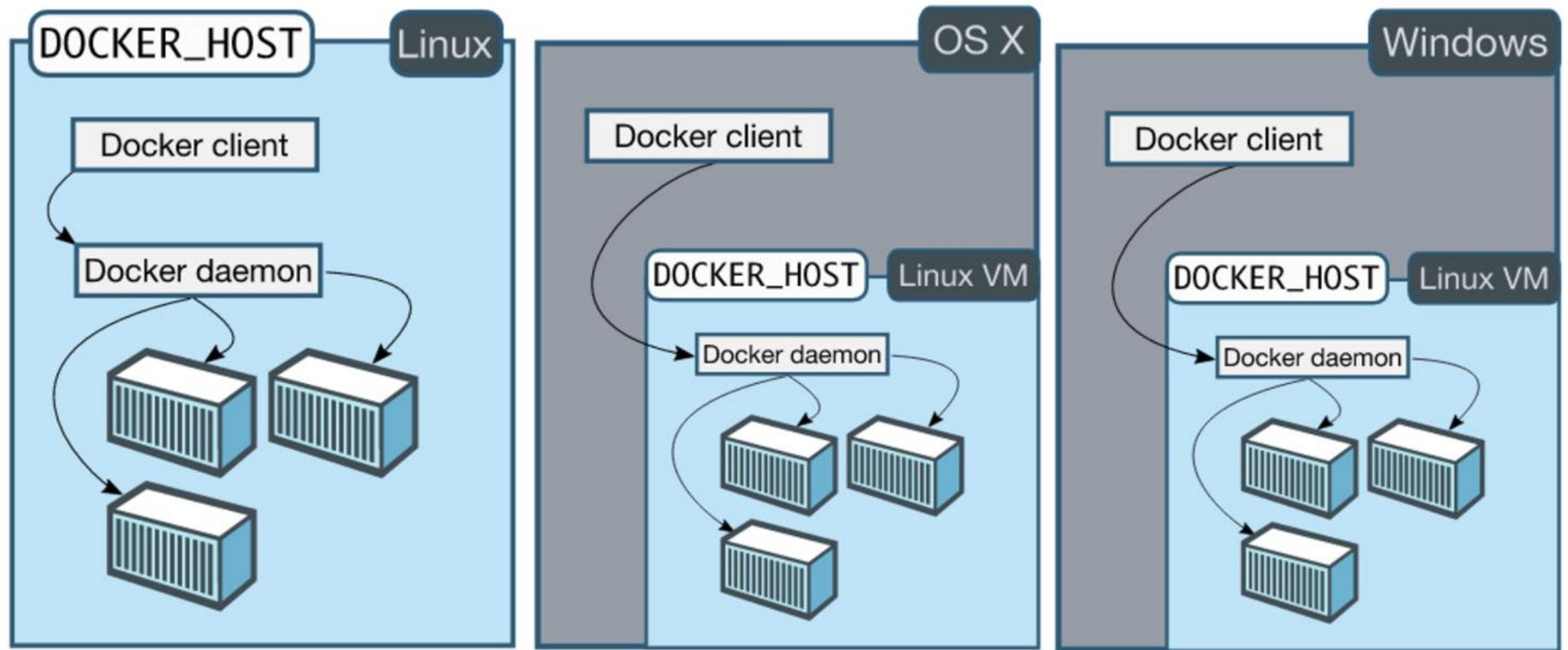
DEMO — ограничим число памяти

Best practice

Слоев поменьше
Базовые образа полегче
Часто меняющееся вниз, редко наверх

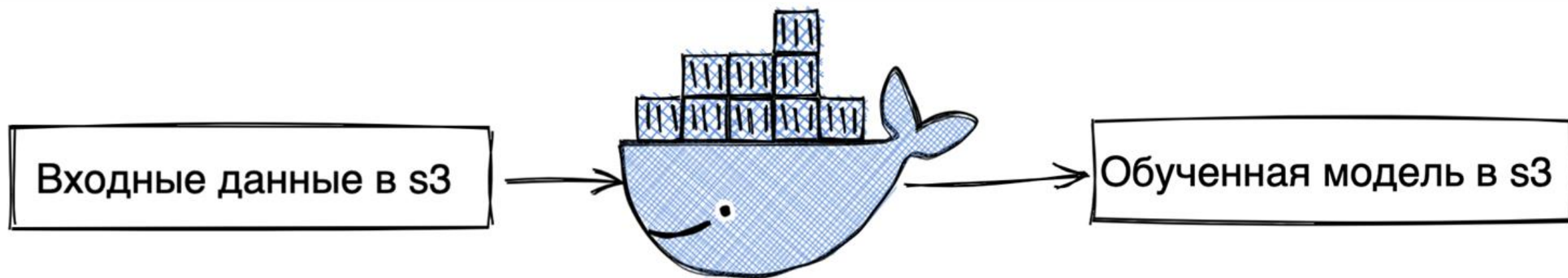


Docker в Windows/Mac



DEMO: обрачиваем тренировку ML модели в Docker

Обучение модели в docker



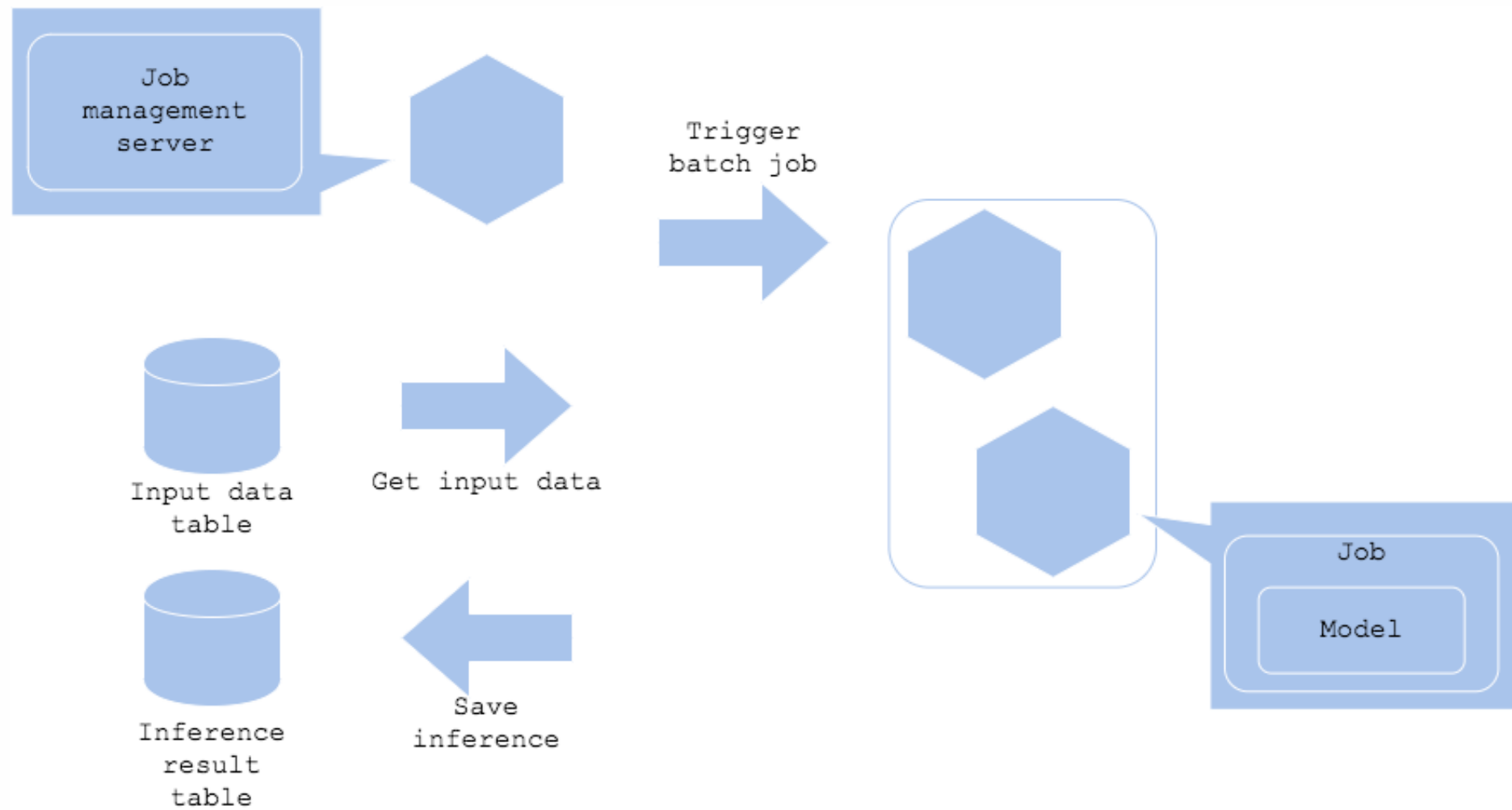
Перерыв



Использование ML модели

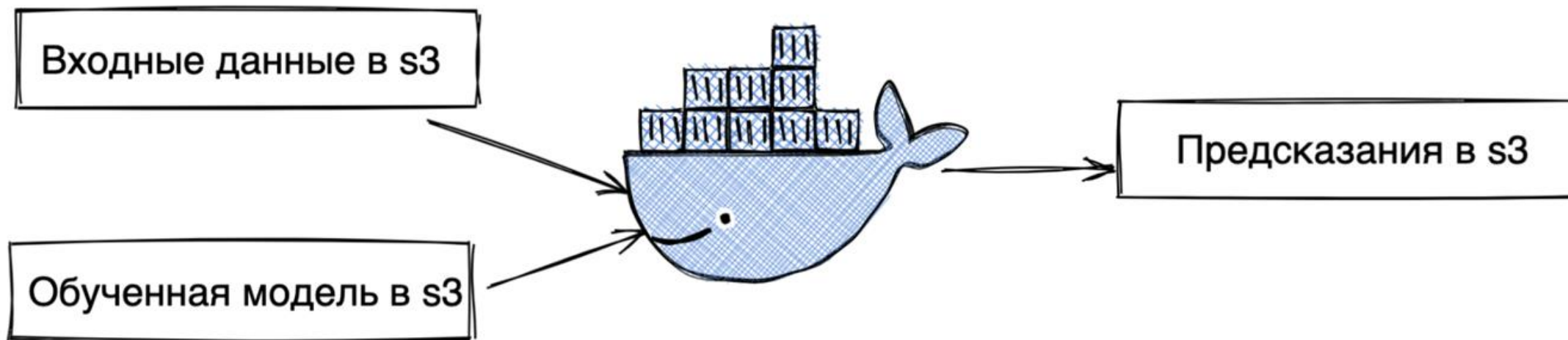
- **Пакетное использование**
- **Онлайн использование**

Пакетный паттерн

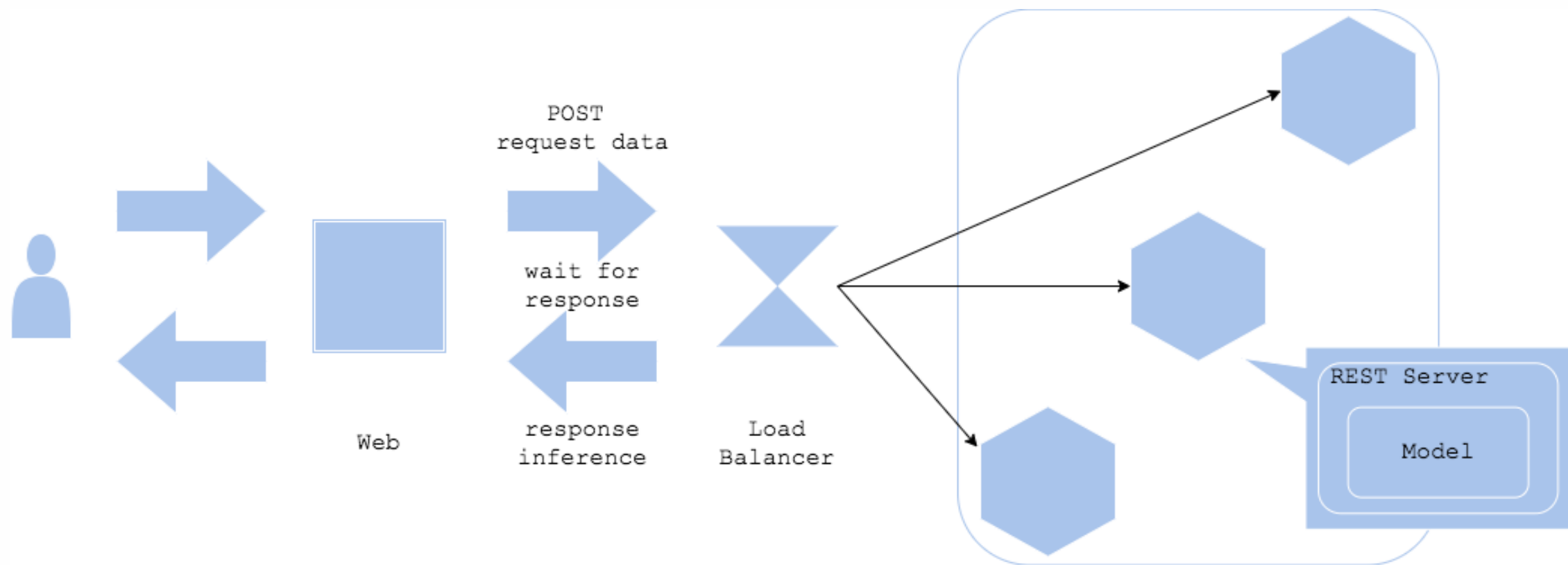


DEMO: обрачиваем батч инференс ML модели в Docker

Batch инференс модели в docker



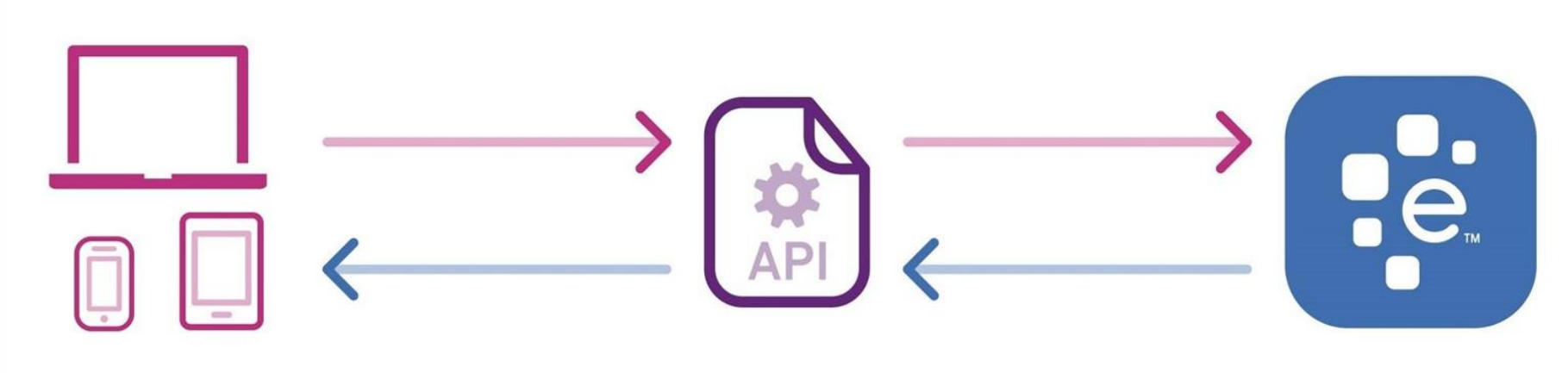
Синхронный паттерн



https://github.com/mercari/ml-system-design-pattern/blob/master/Serving-patterns/Synchronous-pattern/design_en.md

API

Способ взаимодействия программных компонентов.

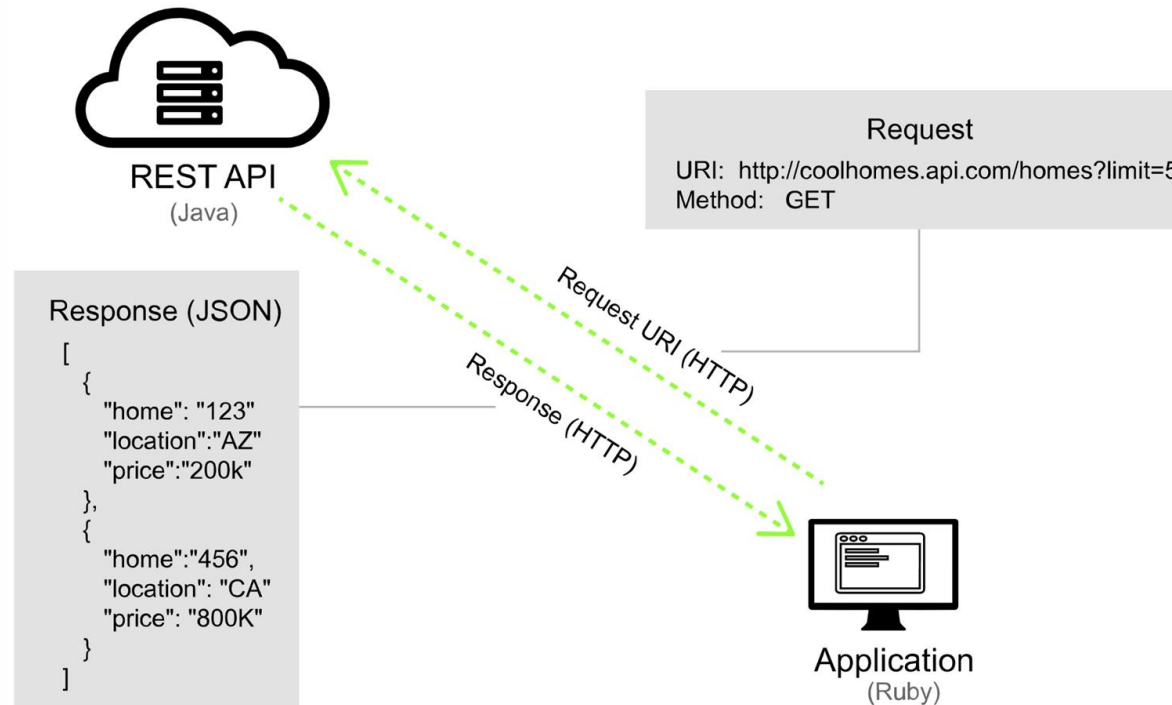


2 основных реализации: **REST**, **rpc**

REST API

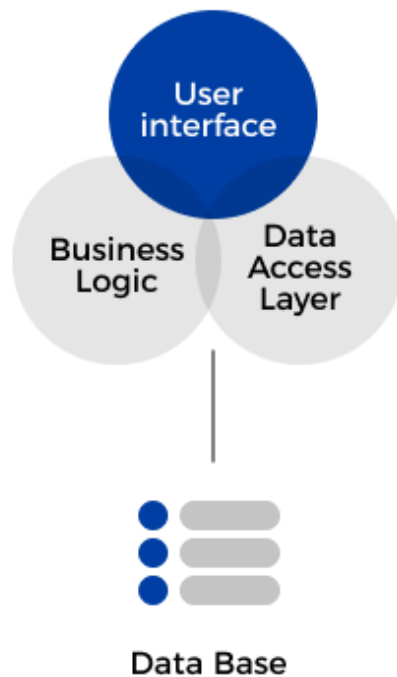
REST (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Работает поверх HTTP

REST API model

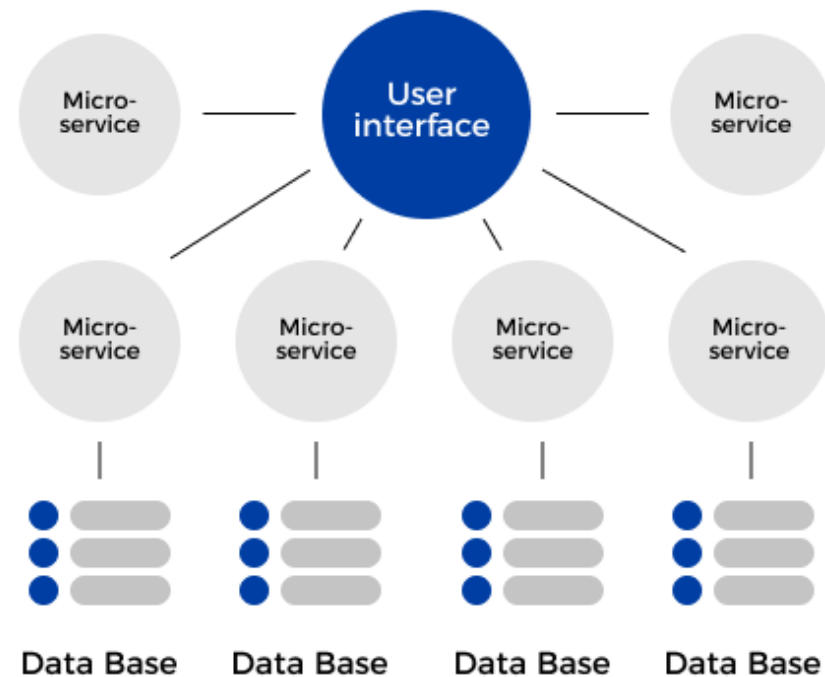


Микросервисное приложение

MONOLITHIC ARCHITECTURE



MICROSERVICE ARCHITECTURE



Monolith

Video Sharing Platform

Microservices

Upload

Streaming

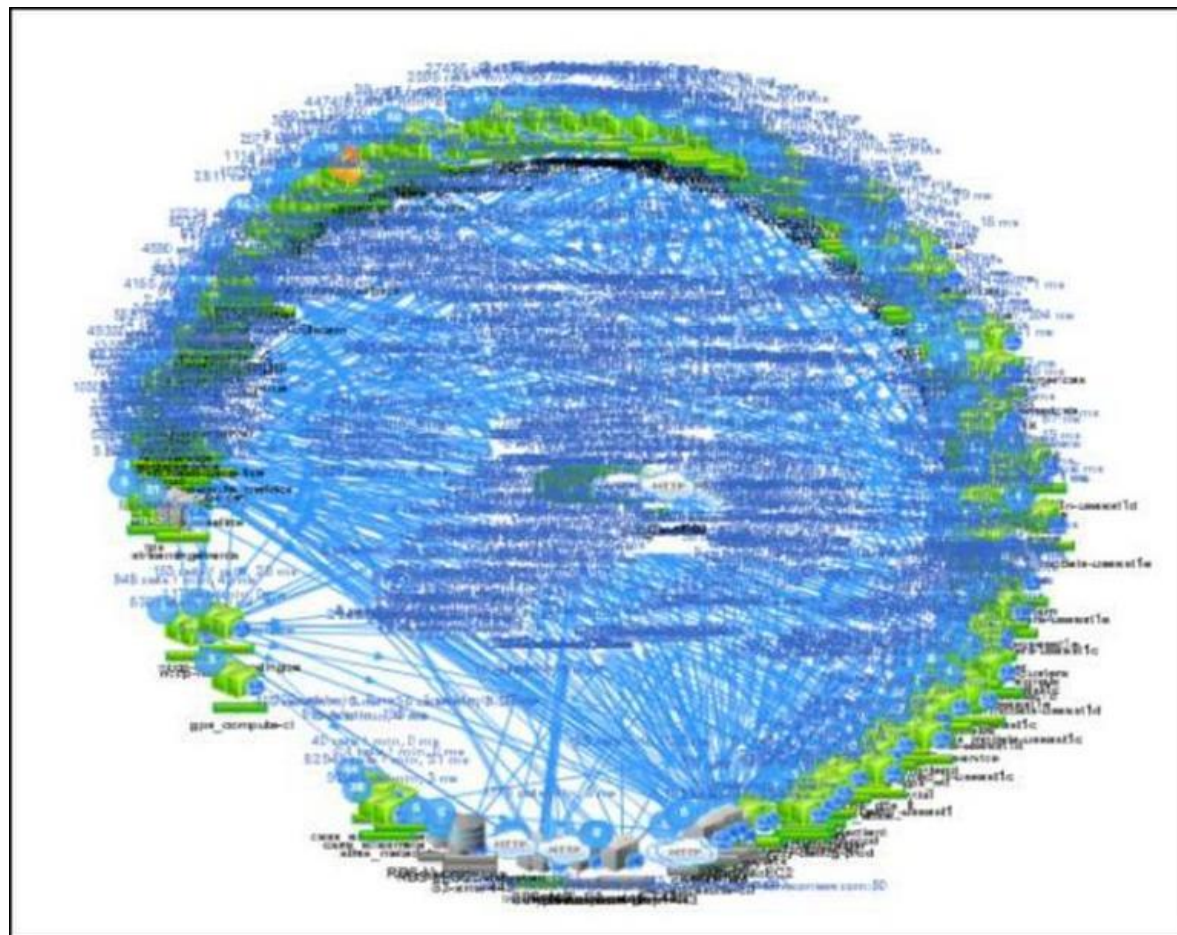
Transcode

Download

Recommendations

Subscriptions

Netflix infrastructure

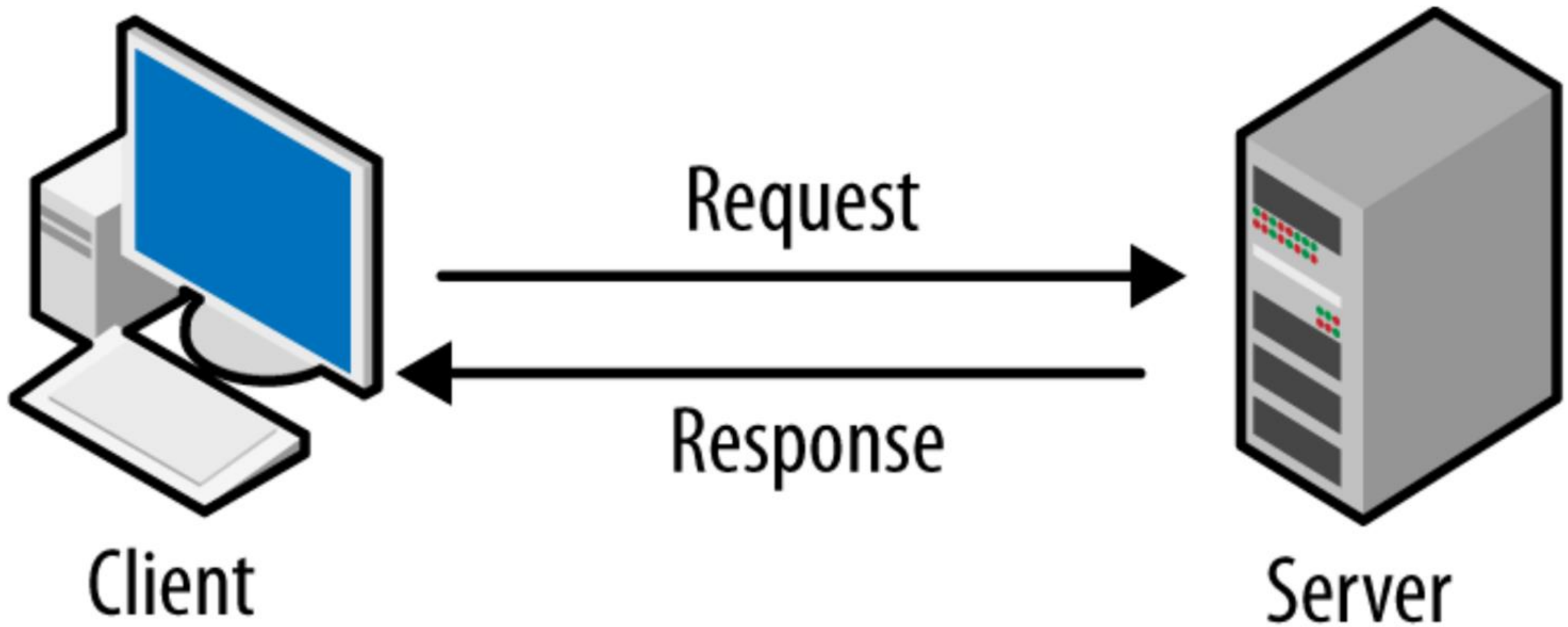




Restful

- Client-server
- Stateless
- Cache
- Uniform Interface
- Layered System
- Code on demand (Optional)

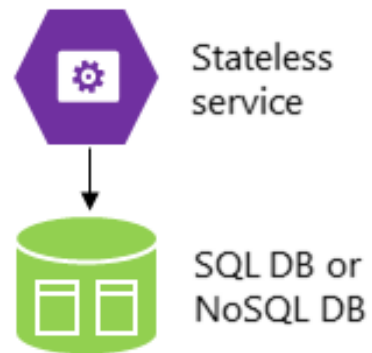
Client-server



Stateless

Stateless Services

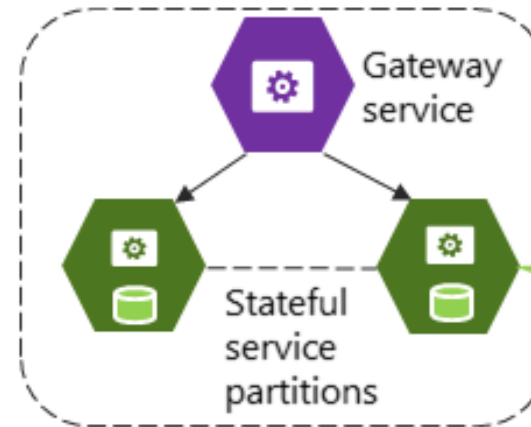
Business microservice A



Stateless microservice with separate store

Stateful Services

Business microservice B



Stateful microservice with in-memory data. Low latency between business logic and data

Stateless



Stateless

- Server requests based on information with each request
- Does not rely on the earlier requests information
- Server does not hold requests information
- Different servers can different information at once



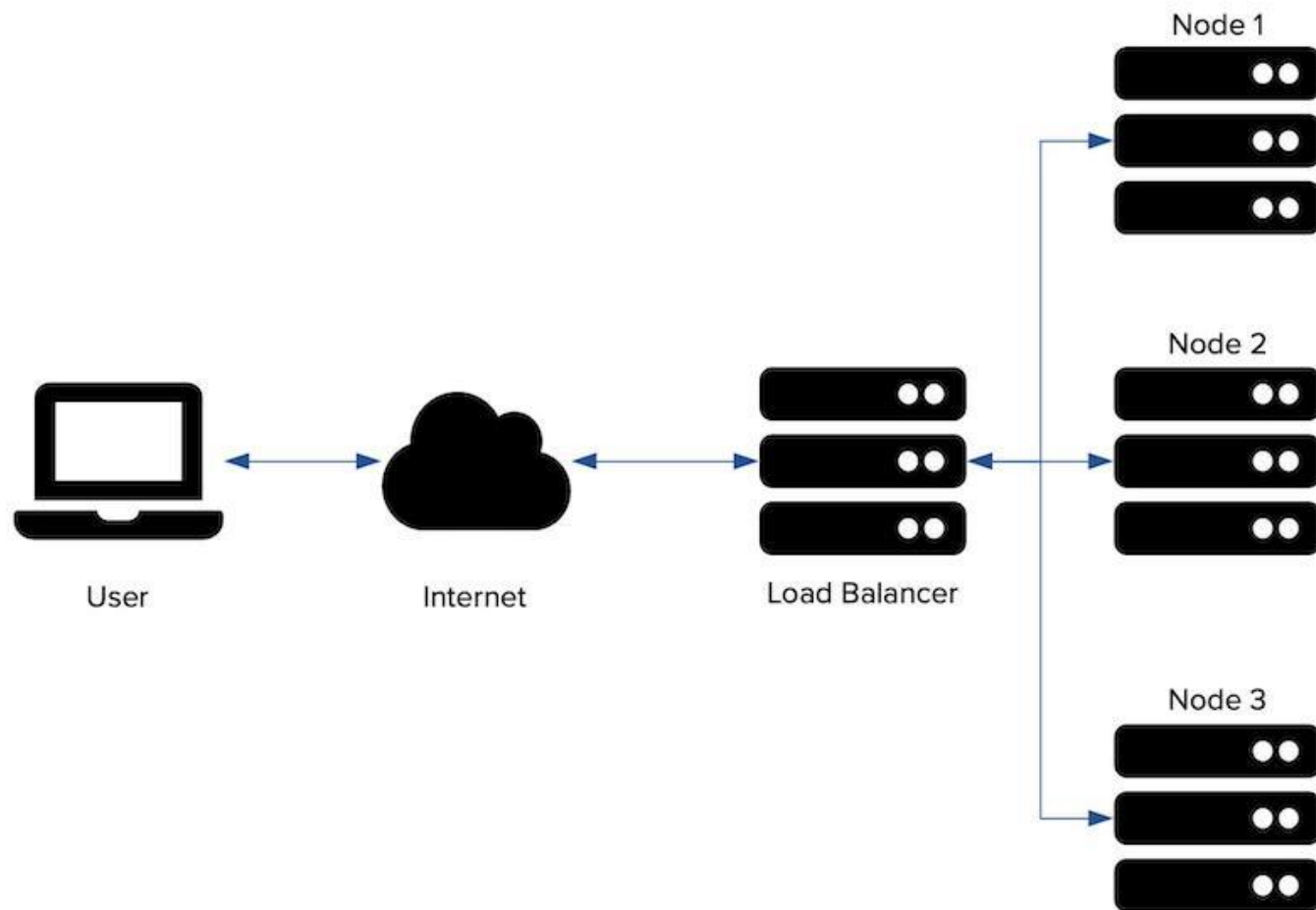
Stateful

- Based on internet protocol which requires a tight state
- Requests based on the information relayed with each request
- Information stored from earlier requests
- Same server must be used to process all requests

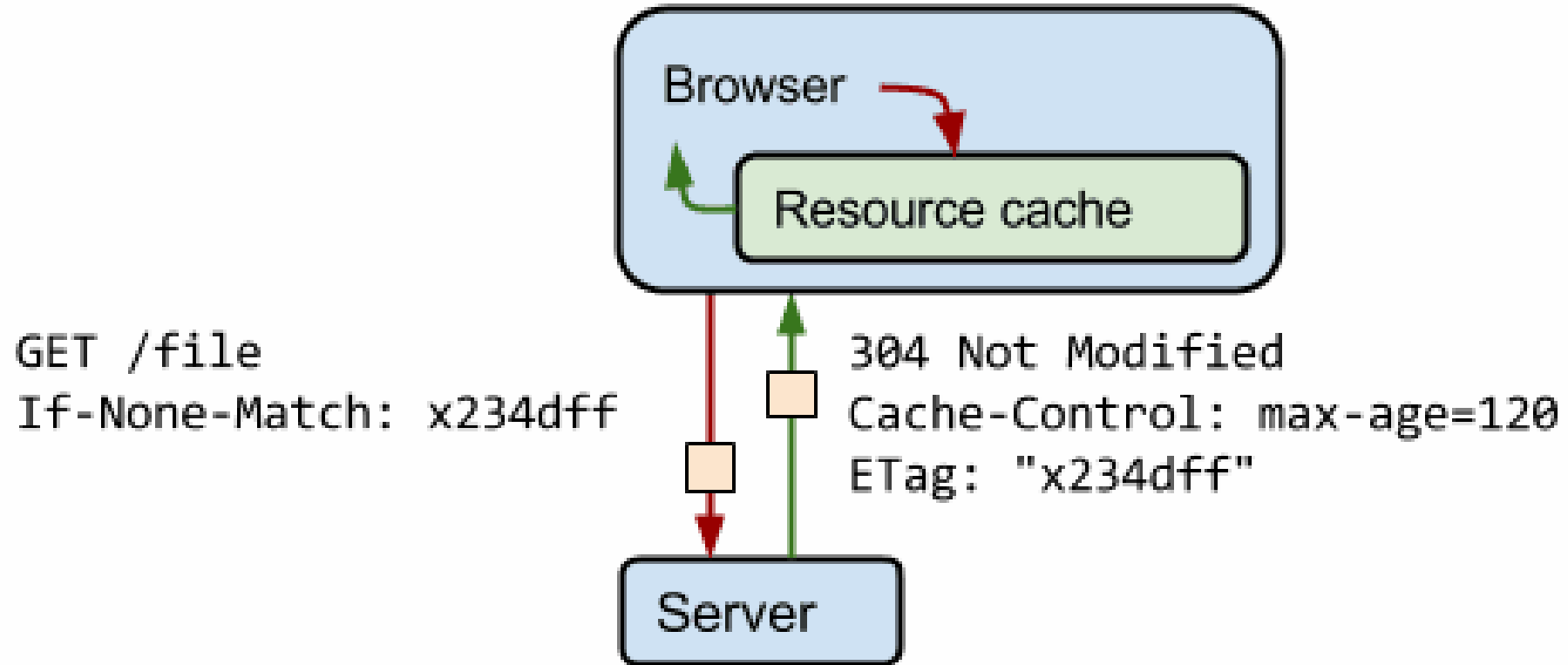


XENONSTACK

Балансировка нагрузки



Cache





Uniform representation

GET

- Retrieves a resource
- Guaranteed not to cause side-effect (SAFE)
- Cacheable

POST

- Creates a new resource
- Unsafe, effect of this verb isn't defined by HTTP

PUT

- Updates an existing resource
- Used for resource creation when client knows URI
- Can call N times, same thing will always happen (idempotent)

DELETE

- Removes a resource
- Can call N times, same thing will always happen (idempotent)

Task	Method	Path
Create a new customer	POST	/customers
Delete an existing customer	DELETE	/customers/{id}
Get a specific customer	GET	/customers/{id}
Search for customers	GET	/customers
Update an existing customer	PUT	/customers/{id}



Uniform representation — используется endpoints

```
https://api.example.com/customers
```

```
https://api.example.com/customers?lastname=Skywalker
```

```
https://api.example.com/customers/932612
```



Uniform representation — Manipulation of Resources through Representations

```
{  
  "id": 12,  
  "firstname": "Han",  
  "lastname": "Solo"  
}
```



HTTP response status codes

HTTP Status Codes

Level 200 (Success)

200 : OK

201 : Created

203 : Non-Authoritative
Information

204 : No Content

Level 400

400 : Bad Request

401 : Unauthorized

403 : Forbidden

404 : Not Found

409 : Conflict

Level 500

500 : Internal Server Error

503 : Service Unavailable

501 : Not Implemented

504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway

HATEOAS - Hypermedia as the Engine of Application State

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
Accept: application/xml
...
```

Ответ будет таким::

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
  <account_number>12345</account_number>
  <balance currency="usd">100.00</balance>
  <link rel="deposit" href="https://bank.example.com/accounts/12345/deposit" />
  <link rel="withdraw" href="https://bank.example.com/accounts/12345/withdraw" />
  <link rel="transfer" href="https://bank.example.com/accounts/12345/transfer" />
  <link rel="close" href="https://bank.example.com/accounts/12345/close" />
</account>
```

Ответ содержит ссылки на депозит, снятие, перевод и закрытие аккаунта

<https://ru.wikipedia.org/wiki/HATEOAS>

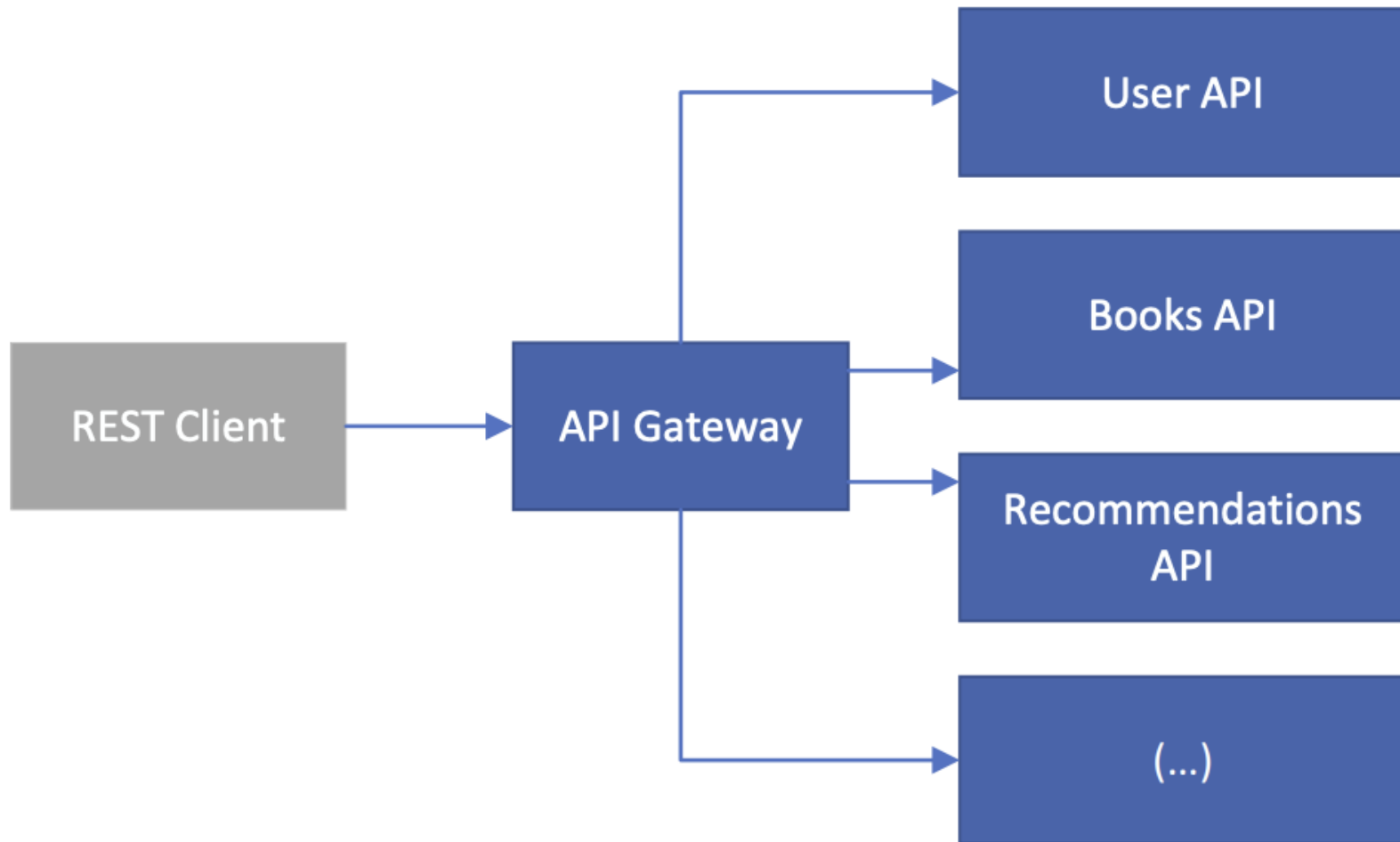
HATEOAS - Hypermedia as the Engine of Application State

В случае отрицательного баланса, доступен только депозит:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
  <account_number>12345</account_number>
  <balance currency="usd">-25.00</balance>
  <link rel="deposit" href="https://bank.example.com/account/12345/deposit" />
</account>
```

Layered System



THE LITTLE BOOK ON REST SERVICES

by Kenneth Lange



The Little Book on REST Services



FastAPI

Фреймворк для создания веб сервисов на Python

- валидация (pydantic)
- быстродействие
- встроенная документация API



<https://fastapi.tiangolo.com/>

<https://habr.com/ru/post/478620/>



Простейший веб-сервис на FastAPI

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"message": "Hello World"}
```

DEMO: обрачиваем ML модель в REST сервис

DEMO: запускаем ML сервис в докере



Docker и REST-services

Михаил Марюфич, MLE

