

Сервис-ориентированные архитектуры

Введение

Глеб Игоревич Радченко

23.01.2021

Цели курса

- Изучить современные подходы к проектированию API сервис-ориентированных распределенных систем
- На практике освоить технологии разработки систем на базе технологий и подходов ZeroMQ, RPC/RMI, gRPC, REST, GraphQL
- Изучить технологии, обеспечивающие авторизацию и безопасность при организации сервис-ориентированных систем

Структура курса

- Основы распределенных вычислительных систем, протоколы передачи данных, форматы сериализации данных;
- Технологии организации связи в распределенных системах: сокеты, RPC/RMI, ZeroMQ.
- Понятие сервис-ориентированной архитектуры (СОА). Типы СОА API. Общие принципы организации СОА.
- RPC API на примере JSON RPC, gRPC, SOAP XML Веб-сервисов
- API Ресурсов на примере REST
- API Сообщений и очереди сообщений.
- Графовый API на примере GraphQL
- Обеспечение безопасности СОА: технологии OAuth, JSON Web Token.
- Обзор технологий облачных вычислительных систем. Концепция микросервисов как развитие СОА. Технология Docker.

Расписание занятий по курсу

- Курс включает в себя 8 занятий (Лекция + Практика)
- Занятия запланированы по субботам, с 9:30 до 12:30
- Начало занятий: 23 января

Система оценивания

- Экзамена не предусмотрено
- Итоговая оценка вычисляется по формуле:

$$Total = 0.7 * Practice + 0.3 * Test$$

где

- *Practice* – усредненная оценка за выполненные и сданные практические задания
- *Test* – усредненная оценка за тесты по лекционным материалам, выполняемые на занятиях



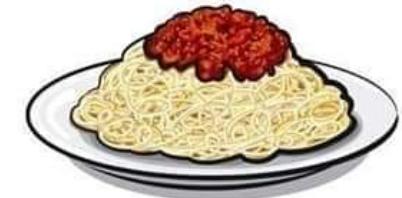
Платформы и языки для разработки

- Java – EJB, Spring, ...
- Python – Django, ...
- C# - WCF, ...
- Ruby – Rails, ...
- Go
- JavaScript - Node JS, ...
- C++

Материалы

- Maarten van Steen, Andrew S. Tanenbaum. Distributed Systems. Third edition Version 3.02 (2018) <https://www.distributed-systems.net/index.php/books/ds3/>
- Хорсдал К. Микросервисы на платформе .NET. Питер, 2018. 352 стр. <https://www.piter.com/collection/all/product/mikroservisy-na-platforme-net>
- Сэм Ньюмен. Создание микросервисов. Питер, 2016. 304 стр.
- Robert Daigneau. Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Addison-Wesley Professional, 2011. 352 р.
<http://books.google.ru/books?id=wIjJZbE08ZQC>

Эволюция архитектуры ПО



1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)



2000's

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)



2010's

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

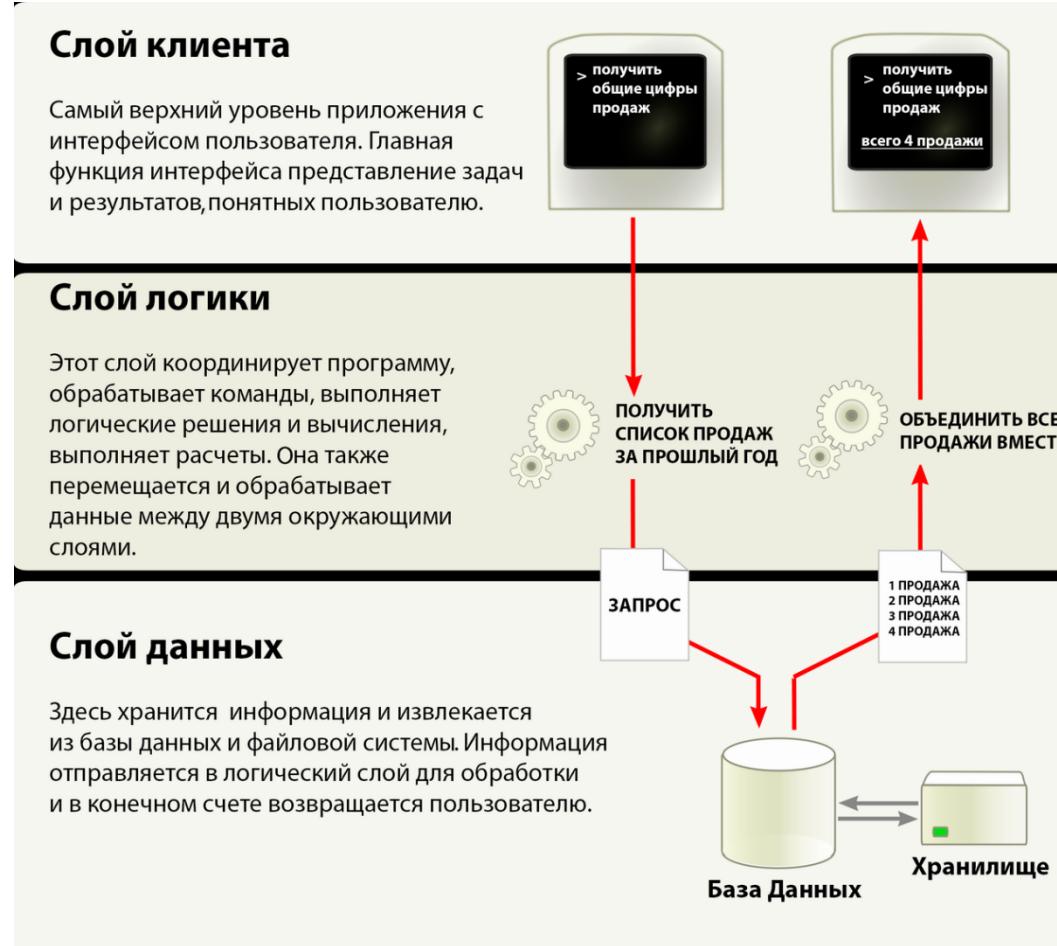
THE EVOLUTION OF
SOFTWARE ARCHITECTURE

By @benorama

WHAT'S NEXT?

PROBABLY PIZZA-ORIENTED ARCHITECTURE

Классика жанра: трехзвенная клиент-серверная архитектура



SQL базы данных как универсальный механизм интеграции компонентов

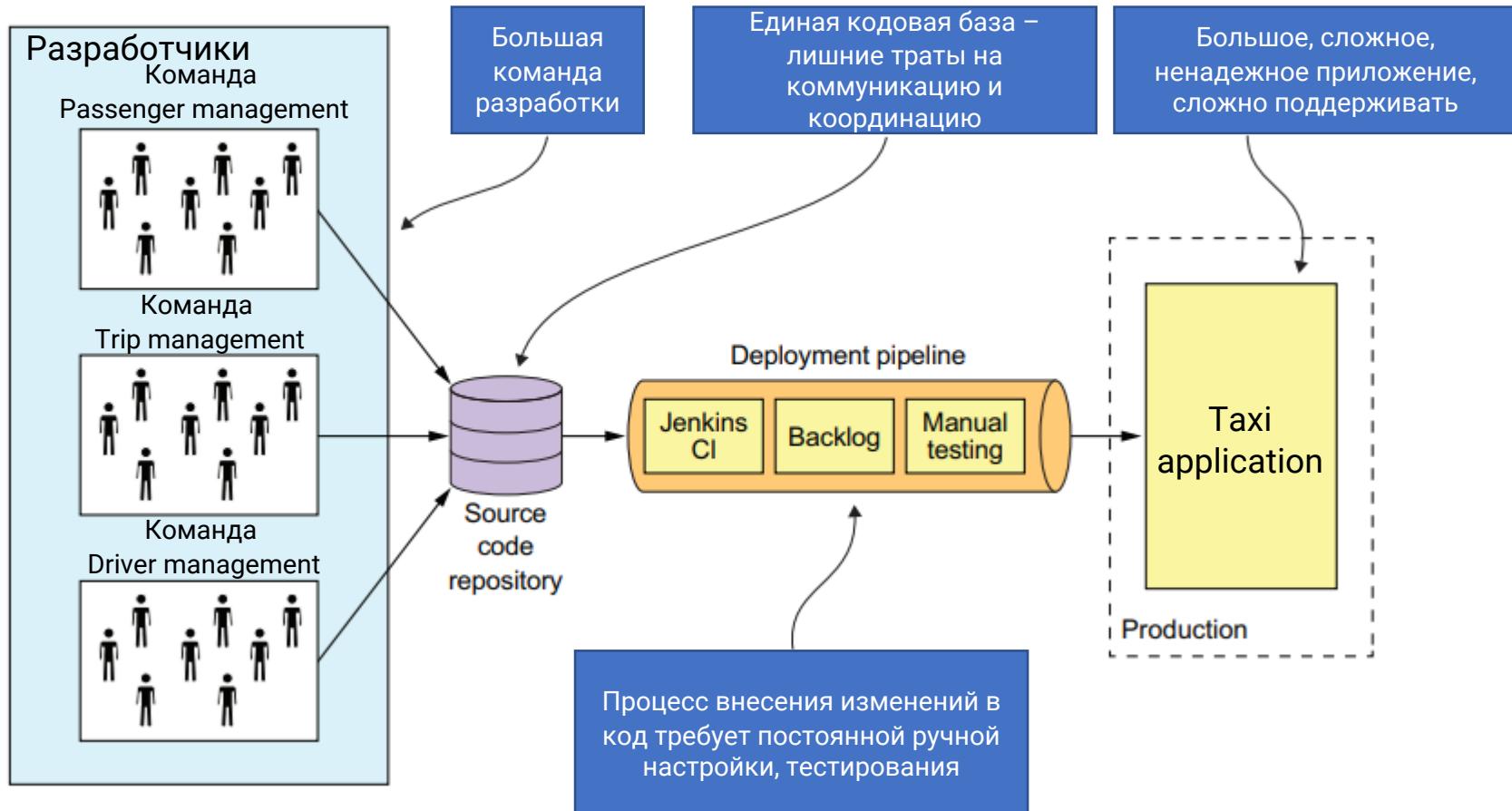
- С 1990 по 2010 годы SQL базы данных работали как универсальный механизм интеграции компонентов



Проблема масштабирования

- До 2000-х работал подход вертикального масштабирования:
 - Майнфрейм от IBM -> добавляем больше памяти -> большой майнфрейм от IBM
- Появление глобальных сервисов (читай Google) привело к тому, что вертикальное масштабирование перестало работать:
 - ↑↑ Стоимость оборудования
 - ↑↑ Стоимость поддержки
 - Потолок масштабирования одного сервера
- Горизонтальное масштабирование SQL баз данных – “противоестественный акт” (“unnatural act” © Мартин Фаулер)

Проблемы масштабирования монолитных приложений



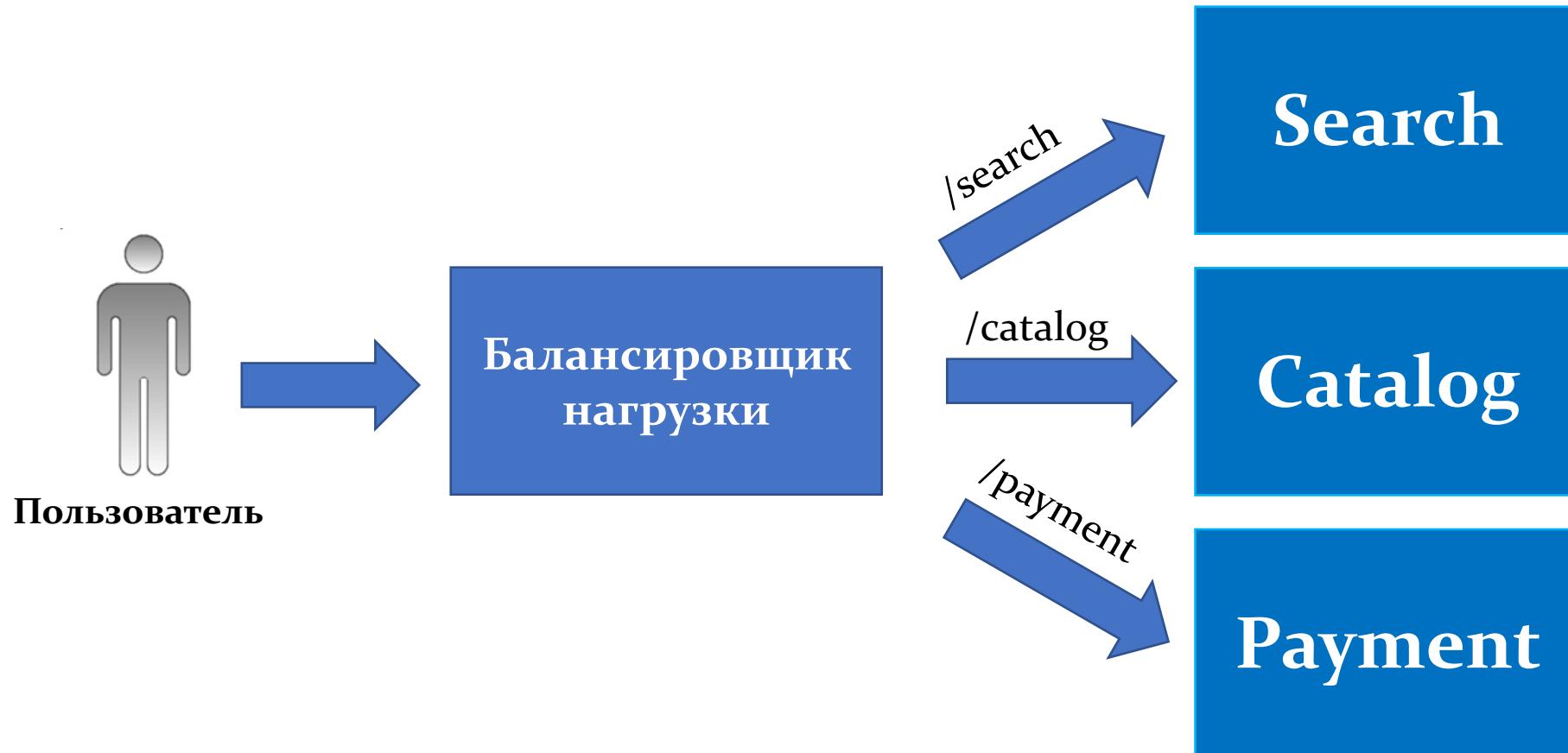
Горизонтальное масштабирование



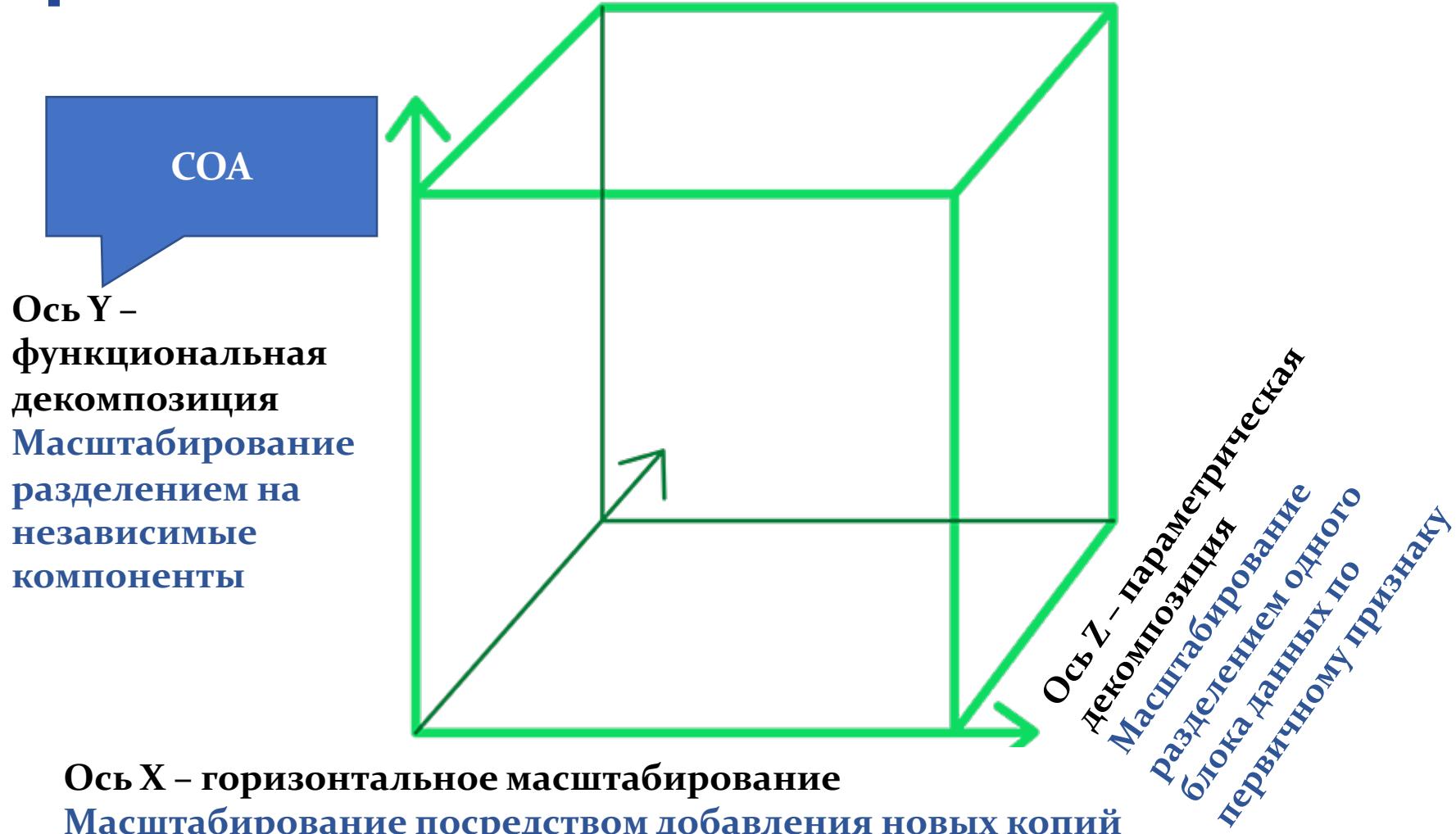
Параметрическая декомпозиция



Функциональная декомпозиция



Масштабирование веб-приложений – все три оси



Как всё начиналось



«Все команды с настоящего момента обязаны предоставлять данные и функциональность через **сервисные интерфейсы**.

Команды должны взаимодействовать друг с другом посредством этих интерфейсов

Другие методы меж-процессной коммуникации **запрещены**: никакого прямого доступа к чужой оперативной памяти, никакого прямого доступа к чужим хранилищам данных и др. **Всё взаимодействие – только через сервисные интерфейсы по сети.**

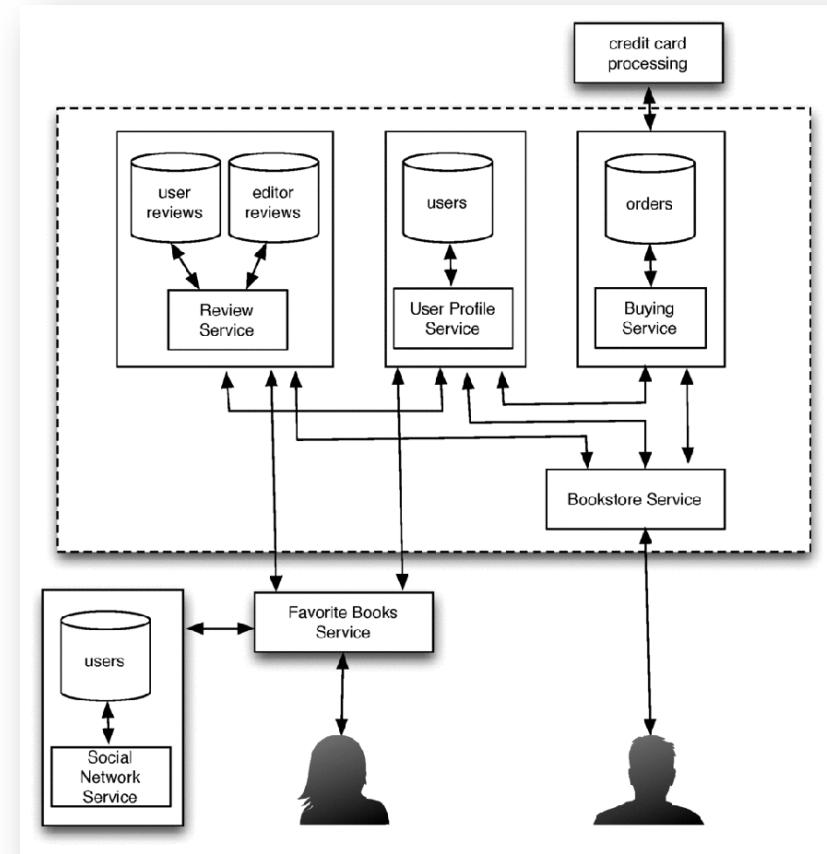
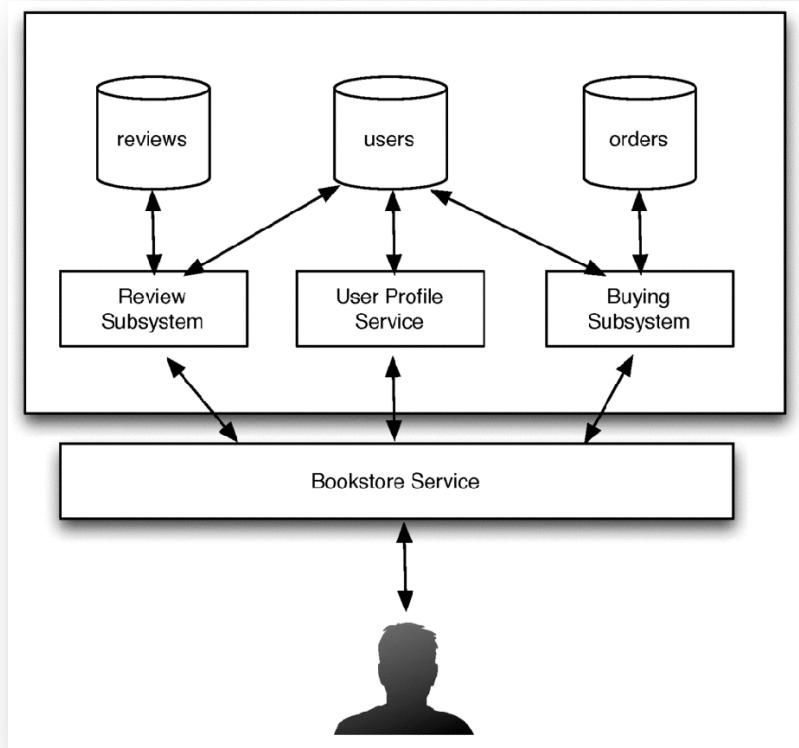
Не важно, какую технологию вы используете (HTTP, Corba...)

Все интерфейсы сервисов должны быть спроектированы таким образом, чтобы **быть общедоступными**. Команды должны разрабатывать интерфейсы так, чтобы сервисы можно было предоставить **потребителям из внешнего мира**. Без исключений.

Любой, кто не следует этим правилам будет уволен.
Спасибо, приятного дня!».

Джеф Безос
Глава Amazon в 2002 г.

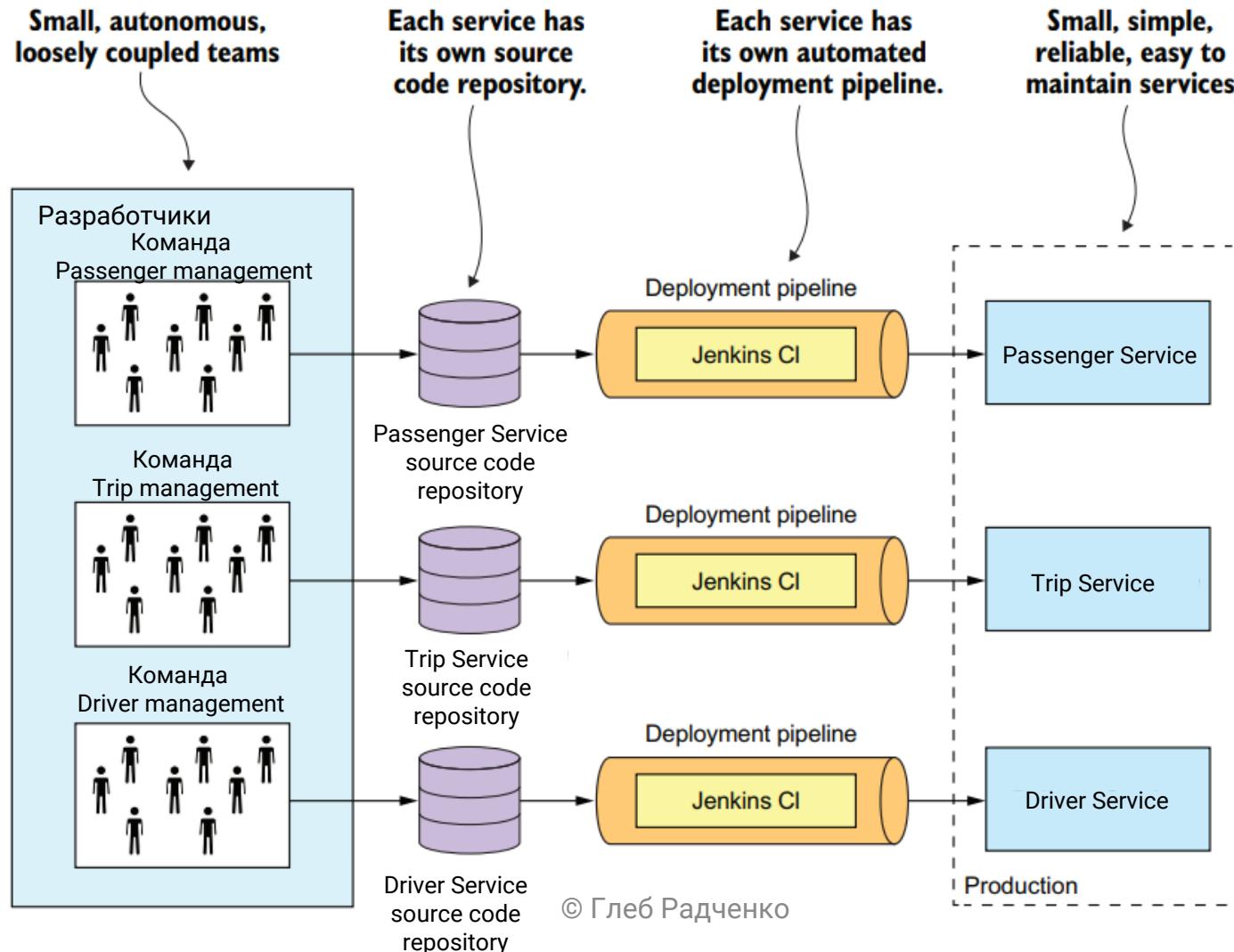
Монолитная VS Сервис-ориентированная архитектура



Микросервисная архитектура

- Микросервисная архитектура – это паттерн проектирования подразумевающий, что сложное приложение разделяется на ряд небольших независимых сервисов, взаимодействующих друг с другом посредством кроссплатформенного API.
- Каждый микросервис включает в себя бизнес-логику и представляет собой совершенно независимый компонент. Сервисы одной системы могут быть написаны на различных языках программирования и общаться друг с другом, используя различные протоколы.
- Отличительные особенности микросервисов:
 - Микросервисы независимы;
 - Микросервисы общаются друг с другой только при помощи сообщений;
 - Каждый микросервис может быть развернут, приостановлен, дублирован или перемещен независимо от других.

Микросервисы: организация команд и разделение компонентов

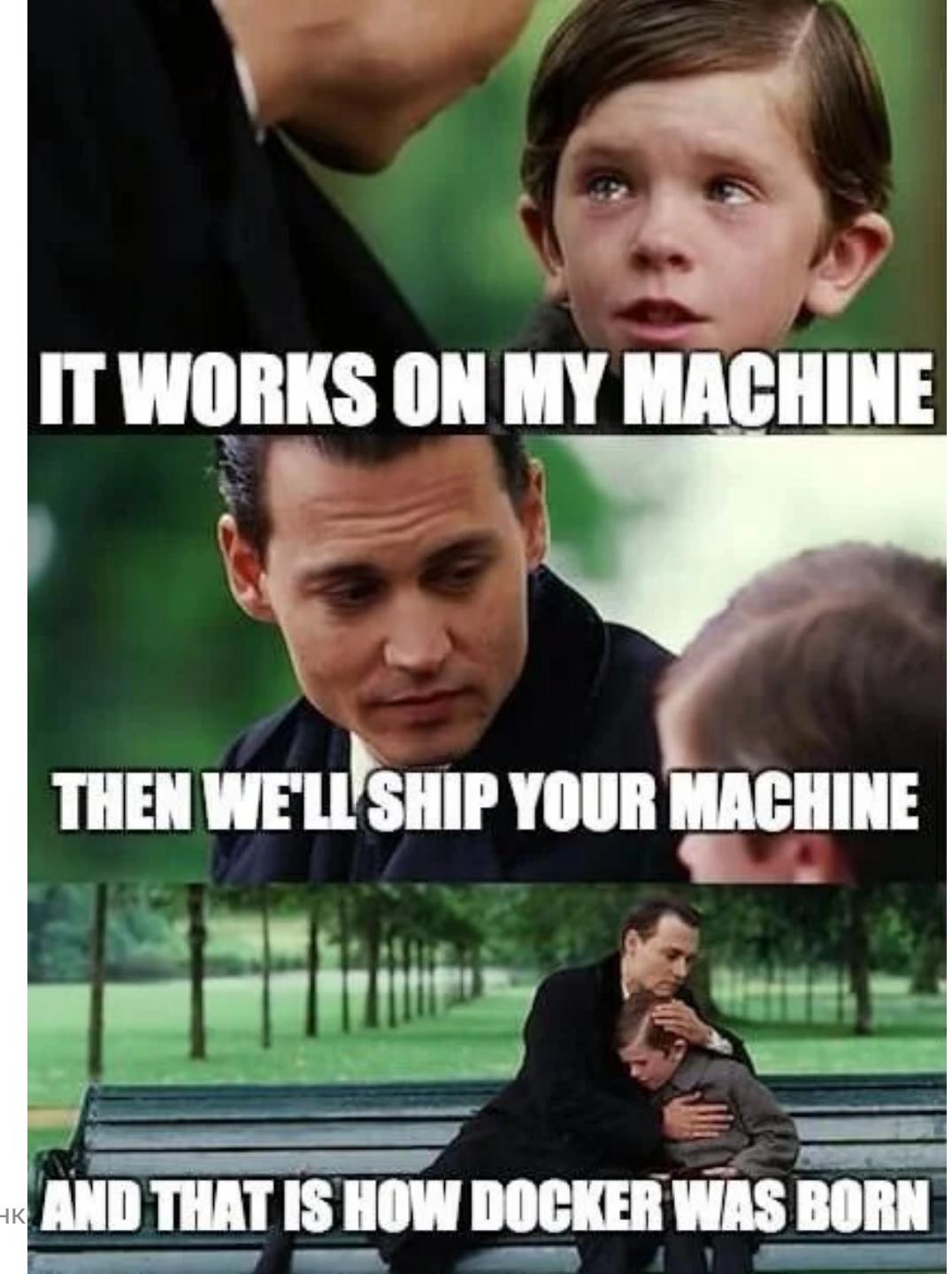


Microservices Patterns
by Chris Richardson

20/48

Exal Docker через Docker...

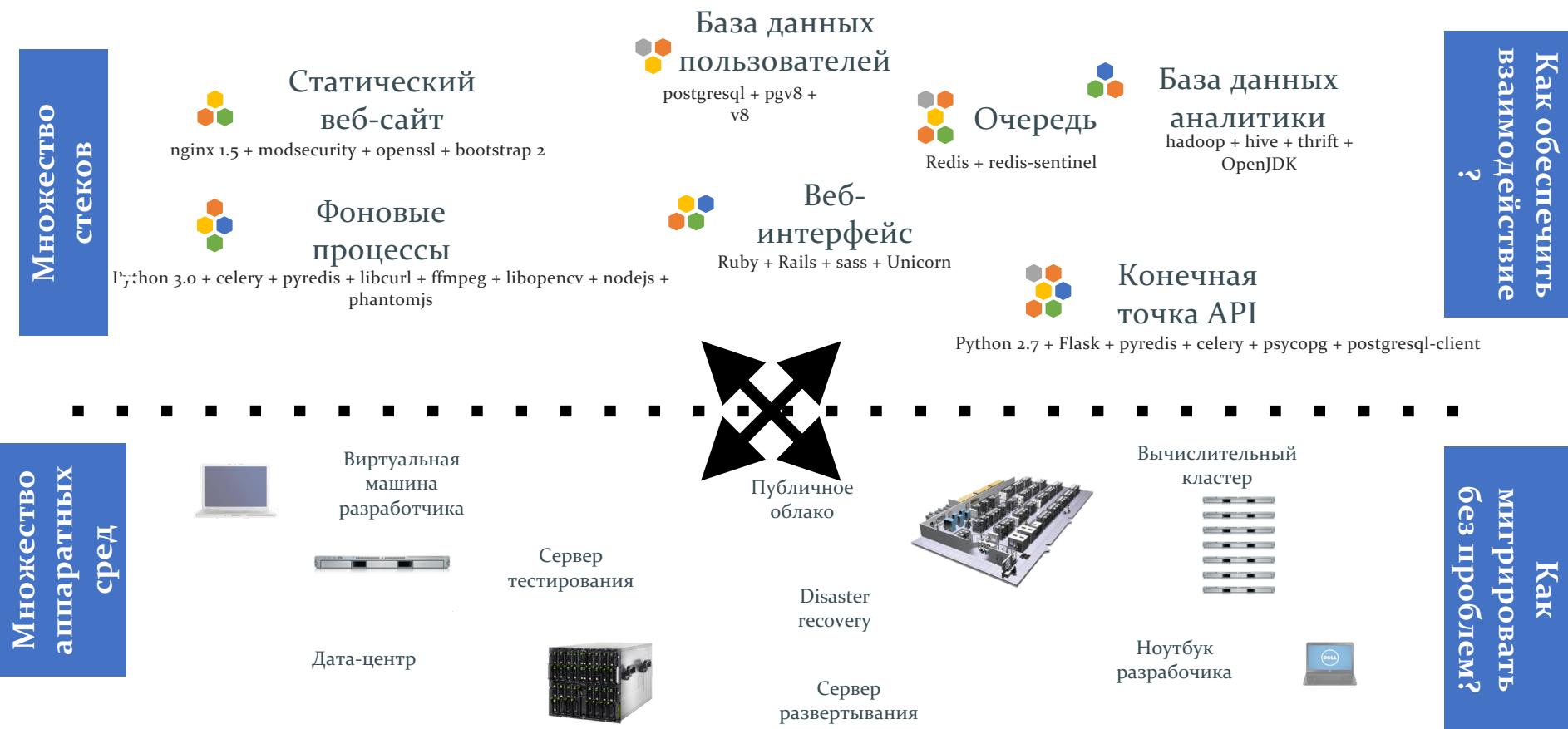
© Глеб Радченко



Процесс создания новых приложений стал совсем другим

1990 -> 2010	2010 -> н.в.
Долгоживущие	Разработка и развитие приложение постоянно
Монолитные приложения, созданные на базе одного стека	Слабосвязанные микросервисы
Развернуто на одном сервере	Развернуты на множестве независимых систем

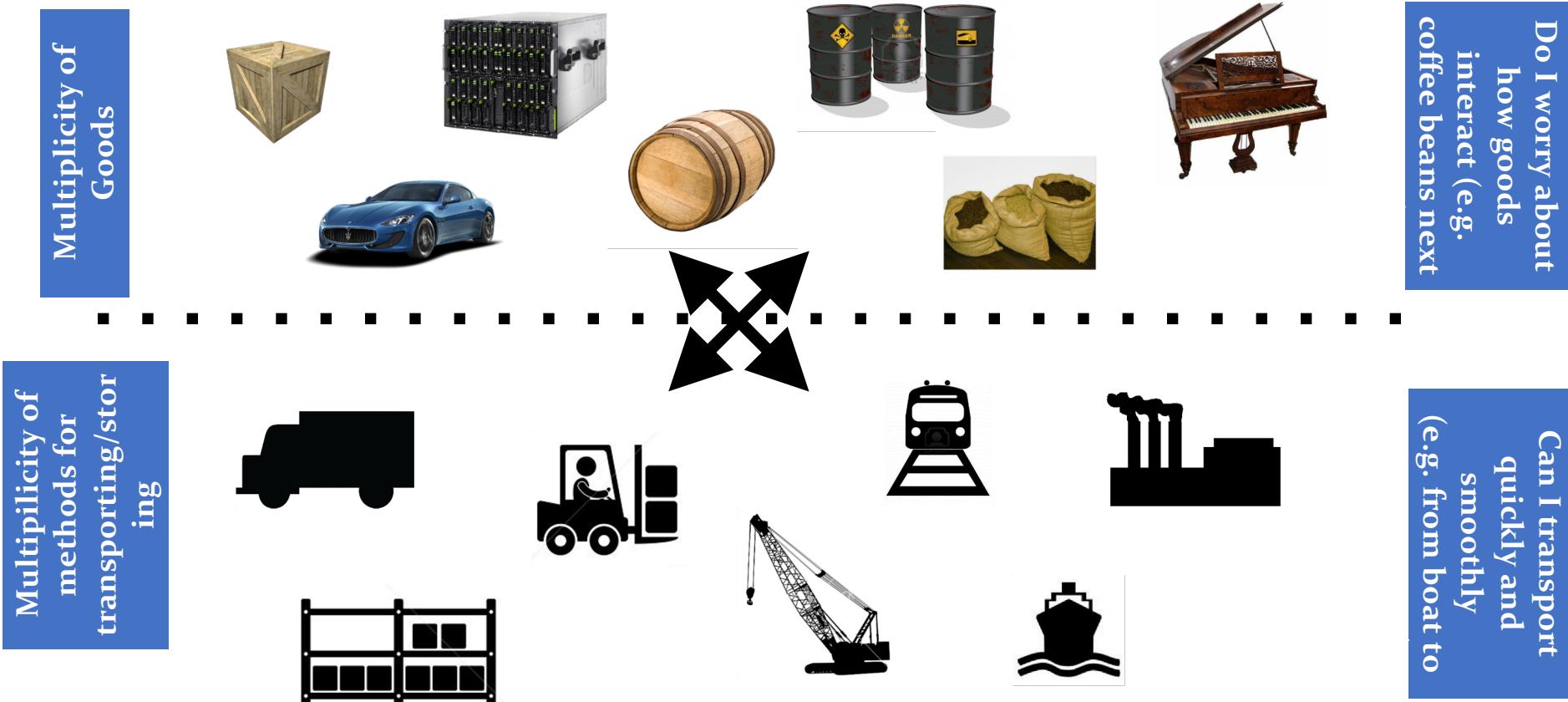
Проблема сегодня: Распределенные приложения



Матрица взаимодействий «Порочная Матрица»

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers	

Вдохновение: транспортировка до 1960



Решение: единая система контейнеров

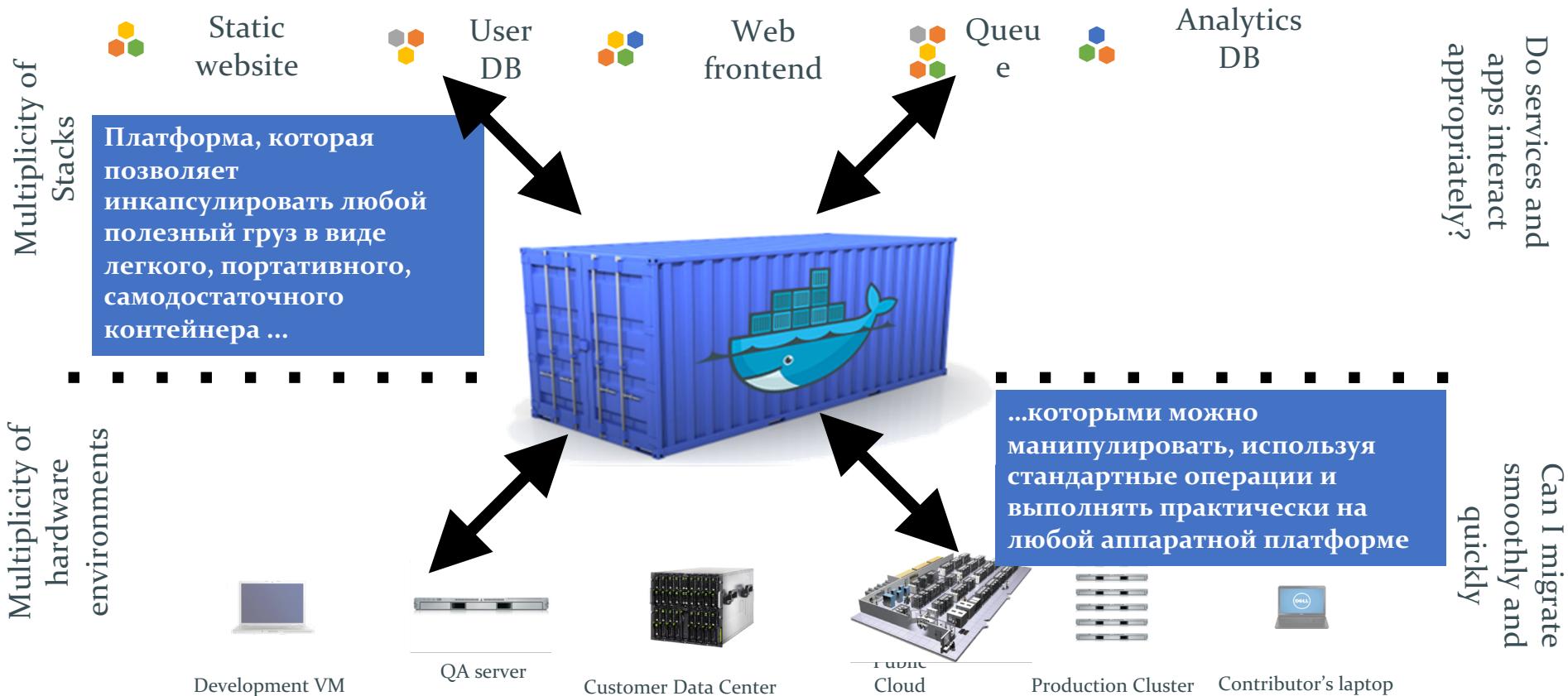


Решение: единая система контейнеров

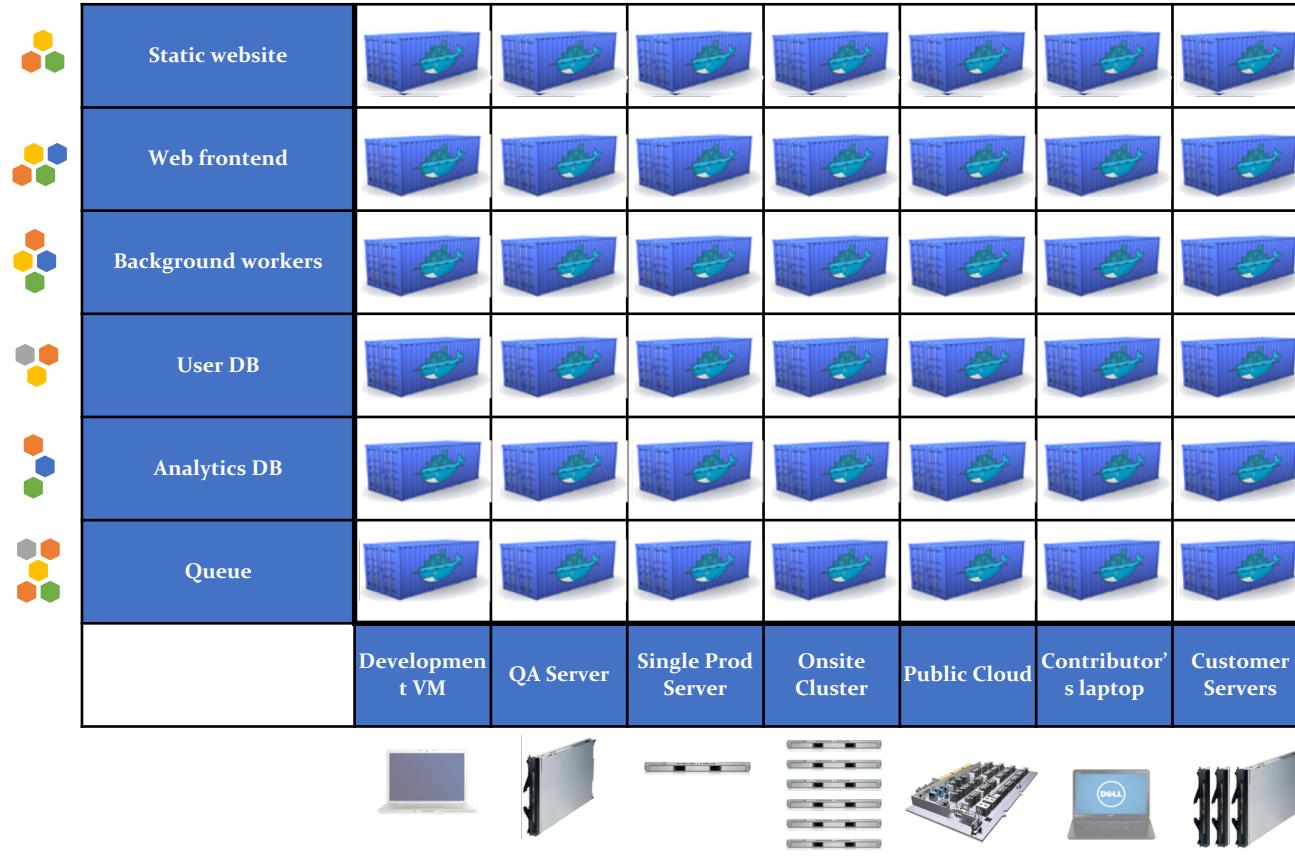


- 90% всех товаров перевозятся стандартными контейнерами
- уменьшение стоимости и времени для погрузки и разгрузки судов на порядок
- Массовое снижение потерь из-за кражи или повреждения
- Огромное снижение транспортных расходов в процентах от конечных товаров (от > 25% до < 3%)
- массивная глобализация
- 5000 судов доставляет 200M контейнеров в год

Экосистема для распределенных приложений



Уход от «Порочной Матрицы»



Откуда взялся Docker?

- 2008-2013 – стартап dotCloud PaaS
- LXC + AUFS + скрипты
- Service → Tool
- dotCloud R.I.P в феврале 2016 г.

Docker: второй заход <https://www.youtube.com/watch?v=8-UEfa1K9kA&t=1358s> <http://blog.amartynov.ru>

Что такое Docker

- Docker – это технология, которая обеспечивает возможность развертывания приложений на базе контейнерной виртуализованной среды
- Docker автоматизирует развертывание и управление приложениями в среде виртуализации на уровне операционной системы; позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, а также предоставляет среду по управлению контейнерами.

Docker: уточнения

- Docker написан на языке Go и базируется на методах контейнеризации, обеспеченных ядром Linux
- Docker не является технологией контейнеризации
- Docker поддерживает разные **среды выполнения**
- Docker – это система **упаковки и доставки** приложений для разных контейнерных технологий

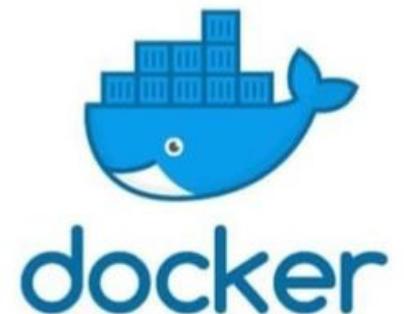
Компоненты платформы Docker

This is docker!

Docker emphasises on **isolation** of applications inside containers, so that different applications have no effect on each other.

Docker is smart.

Be like docker.



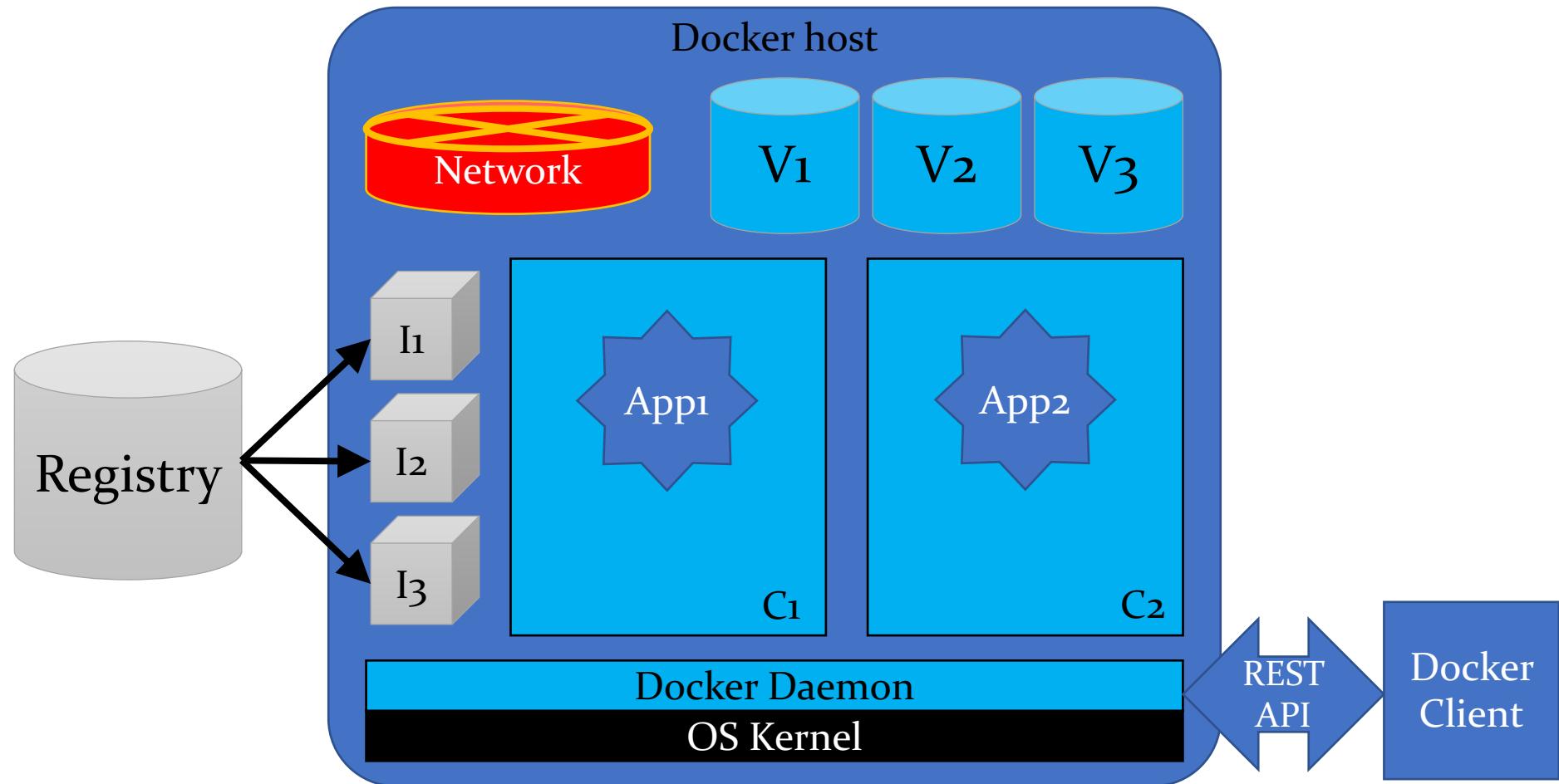
#staysafe #fightcorona

Составляющие Docker

Engine	Orchestration	Installation	Services
<ul style="list-style-type: none">• Daemon• REST API• Client	<ul style="list-style-type: none">• Machine mode• Swarm mode• Compose	<ul style="list-style-type: none">• For Linux• For Mac• For Win• For Azure• For AWS	<ul style="list-style-type: none">• Docker HUB• CS Engine• Trusted Registry• Docker Cloud• Docker Store

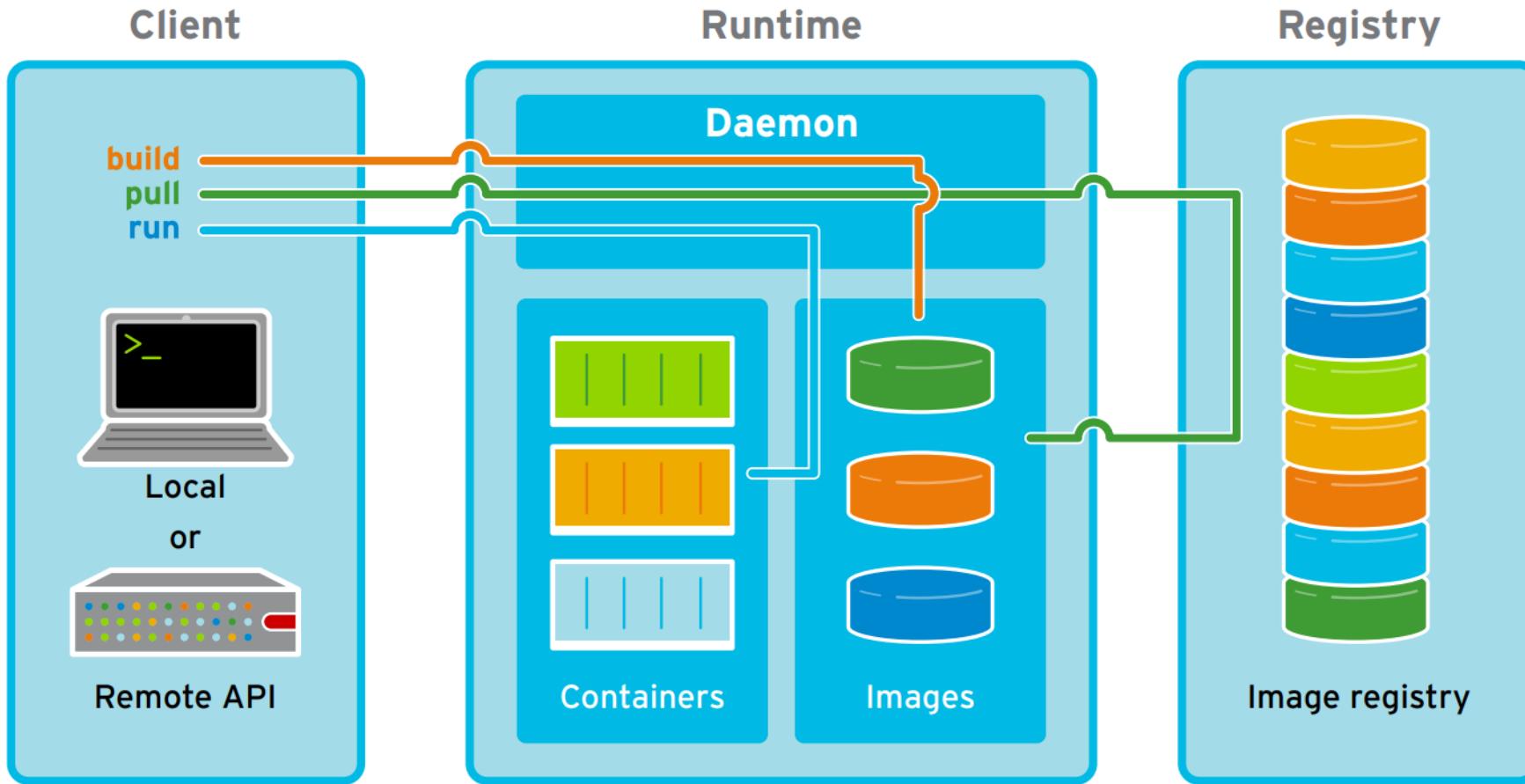
Docker: второй заход <https://www.youtube.com/watch?v=8-UEfa1K9kA&t=1358s>
<http://blog.amartynov.ru>

Docker on a Host



Docker: второй заход <https://www.youtube.com/watch?v=8-UEfa1K9kA&t=1358s>
<http://blog.amartynov.ru>

Docker Runtime



Docker Daemon управляет

- Контейнерами (containers)
- Образами (images)
- Томами (storage volumes)
- Виртуальными сетями (networks)

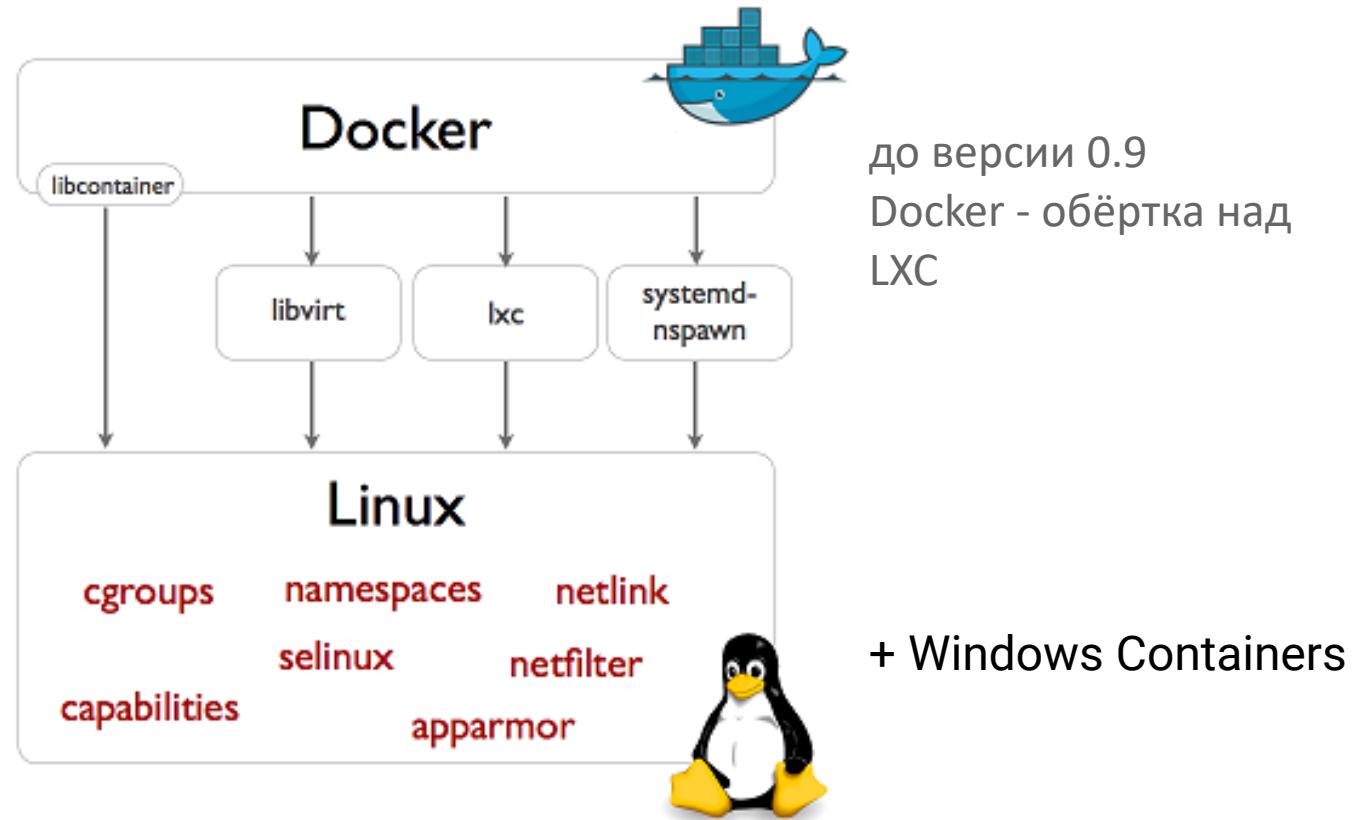
Контейнер

- Это один или несколько процессов запущенных в изолированном окружении:
 - Не видно внешних процессов
 - Своя корневая файловая система
 - Своя конфигурация сети (hostname, IP..)
 - И другое (UID и др)
 - Ограничение по: CPU, MEM, Net I/O, Block I/O

Контейнер Docker (Container)

- Docker не изобрел контейнеры.
- Docker более не завязан на конкретный тип контейнеризации
- Но Docker сделал использование контейнеров удобным, доступным всем.

Технологии виртуализации которые использует Docker



Образ Docker (Image)

- Образ (image) – это **read-only / immutable** заготовка файловой системы.
- Он содержит файлы ОС, приложения, все необходимые библиотеки и зависимости.
- Все, кроме самого ядра.
- Это – способ распространения приложений.

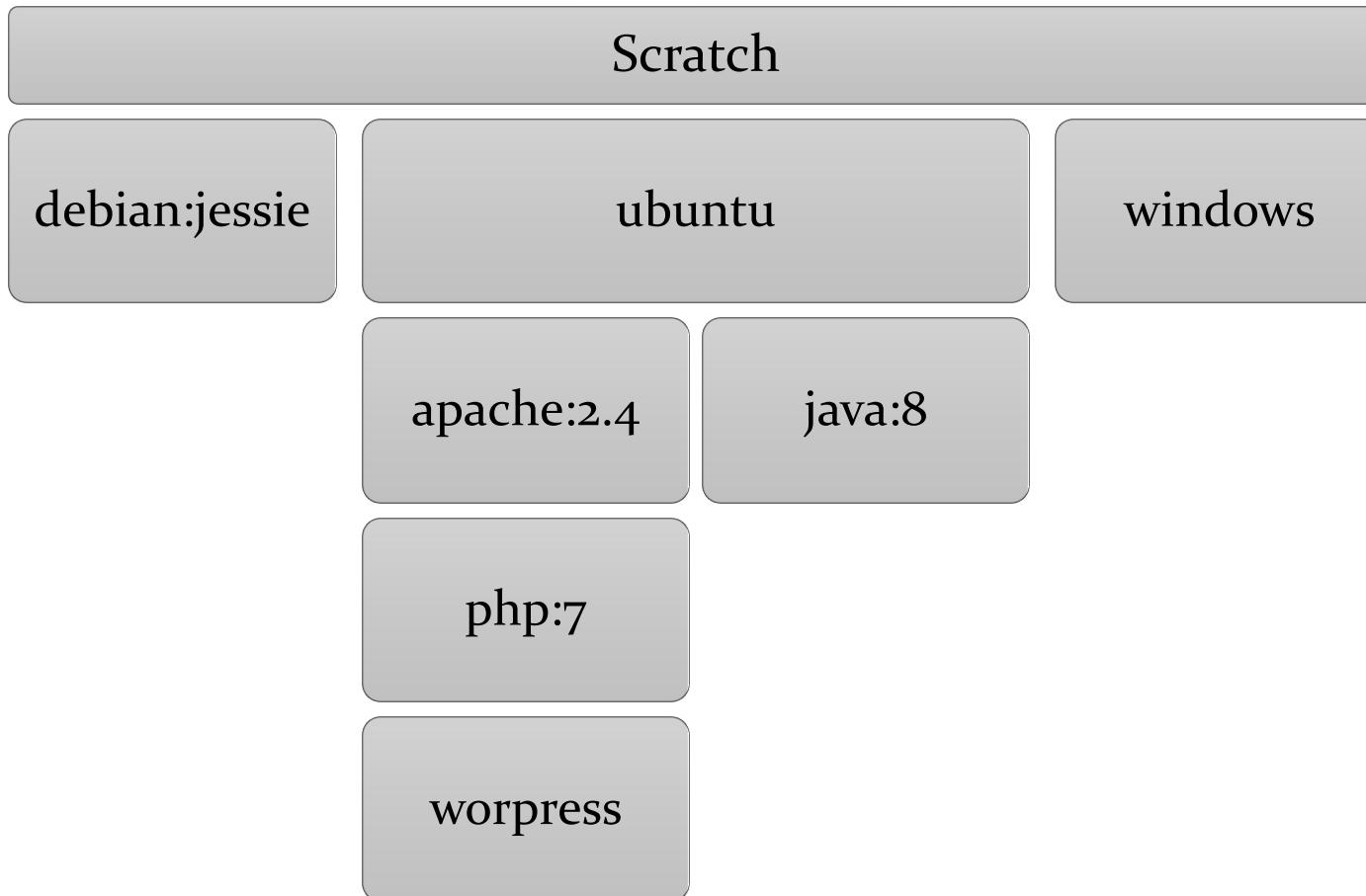
Образ Docker (Image)



По внутреннему устройству образ напоминает репозиторий Git:

- Состоит из слоев (слой ~ коммит)
- Каждый слой имеет уникальный ID
- Слои наследуются, образуя граф зависимостей
- Образы можно **commit**, **push**, **pull** и **tag**

Образ Docker (Image)



Docker Hub

<https://hub.docker.com>

- Официальные образы
- Пользовательские образы
- Automated builds

Docker Hub

 nginx official	4.5K STARS	10M+ PULLS	DETAILS
 busybox official	851 STARS	10M+ PULLS	DETAILS
 redis official	2.9K STARS	10M+ PULLS	DETAILS
 ubuntu official	5.0K STARS	10M+ PULLS	DETAILS
 docker official	1.2K STARS	10M+ PULLS	DETAILS
 alpine official	1.6K STARS	10M+ PULLS	DETAILS
 mongo official	2.5K STARS	10M+ PULLS	DETAILS

 mysql official	3.4K STARS	10M+ PULLS	DETAILS
 swarm official	530 STARS	10M+ PULLS	DETAILS
 hello-world official	196 STARS	10M+ PULLS	DETAILS
 postgres official	2.8K STARS	10M+ PULLS	DETAILS
 elasticsearch official	1.7K STARS	10M+ PULLS	DETAILS
 node official	3.1K STARS	10M+ PULLS	DETAILS
 httpd official	772 STARS	10M+ PULLS	DETAILS
 wordpress official	1.4K STARS	10M+ PULLS	DETAILS

Dockerfile (Wordpress)

```
1 FROM php:5.6-apache
2
3 # install the PHP extensions we need
4 RUN apt-get update && apt-get install -y libpng12-dev libjpeg-dev && rm -rf /var/lib/apt/lists/* \
5     && docker-php-ext-configure gd --with-png-dir=/usr --with-jpeg-dir=/usr \
6     && docker-php-ext-install gd mysqli opcache
7
8 # set recommended PHP.ini settings
9 # see https://secure.php.net/manual/en/opcache.installation.php
10 RUN { \
11         echo 'opcache.memory_consumption=128'; \
12         echo 'opcache.interned_strings_buffer=8'; \
13         echo 'opcache.max_accelerated_files=4000'; \
14         echo 'opcache.revalidate_freq=2'; \
15         echo 'opcache.fast_shutdown=1'; \
16         echo 'opcache.enable_cli=1'; \
17     } > /usr/local/etc/php/conf.d/opcache-recommended.ini
18
19 RUN a2enmod rewrite expires
20
21 VOLUME /var/www/html
22
23 ENV WORDPRESS_VERSION 4.6.1
24 ENV WORDPRESS_SHA1 027e065d30a64720624a7404a1820e6c6fff1202
25
26 RUN set -x \
27     && curl -o wordpress.tar.gz -fSL "https://wordpress.org/wordpress-${WORDPRESS_VERSION}.tar.gz" \
28     && echo "$WORDPRESS_SHA1 *wordpress.tar.gz" | shasum -c - \
29 # upstream tarballs include ./wordpress/ so this gives us /usr/src/wordpress
30     && tar -xzf wordpress.tar.gz -C /usr/src/ \
31     && rm wordpress.tar.gz \
32     && chown -R www-data:www-data /usr/src/wordpress
33
34 COPY docker-entrypoint.sh /usr/local/bin/
35 RUN ln -s /usr/local/bin/docker-entrypoint.sh /entrypoint.sh # backwards compat
36
37 # ENTRYPOINT resets CMD
38 ENTRYPOINT ["docker-entrypoint.sh"]
39 CMD ["apache2-foreground"]
```

Инструкции Dockerfile

Dockerfile – это простой текстовый файл, в котором содержится список команд Докер-клиента. Это простой способ автоматизировать процесс создания образа.

- **FROM** python:3-onbuild – базовый образ
- **RUN** apt-get update – команды, выполняемые при сборке образа
- **EXPOSE** 5000 – открыть порт для доступа извне
- **CMD** ["python", "./app.py"] - какие команды нужно выполнить при старте

Спасибо за внимание!

Готов ответить на
ваши вопросы!

© Глеб Радченко



BUT THEY COULD NOT UNDERSTAND ITS ALIEN LANGUAGE

