Base Infrastructure

Матвей Попов

План

- Что такое инфраструктура?
- Как выглядит наше приложение?
- Как его можно запускать?
- Где его можно запускать?
- Как его доставить до контура?
- Как следить за приложением?
- Как устроено логирование?
- Как организовывать эксплуатацию приложения?

А что такое Инфраструктура(Who)?

Инфраструктура

Это совокупность сооружений, зданий, систем и служб, необходимых для нормального функционирования экономики и обеспечения повседневной жизни населения

А что это значит в контексте backend приложения?

Почти тоже самое

Инфраструктура

Это совокупность серверов, сервисов, систем хранилищ артефактов и секретов, необходимых для нормального функционирования приложения и обеспечения качественного процесса разработки

Будем двигаться от простого к сложному

Сначала посмотрим на наше приложение

Из чего оно состоит?

- Набор наших исходных файлов с кодом
- Описание необходимых зависимостей для нашего проекта
- Набор файлов с описанием конфигурационных переменных

Что с этим нужно делать?

1. Скомпилировать
2. Запустить тесты
3. Запустить приложение локально
4. Запустить его на сервере

Как его можно запускать?

- Через кнопку в идеи
- Через сборщик проектов
- Через Kotlin SDK

В итоге это всегда SDK

А где его можно запускать?

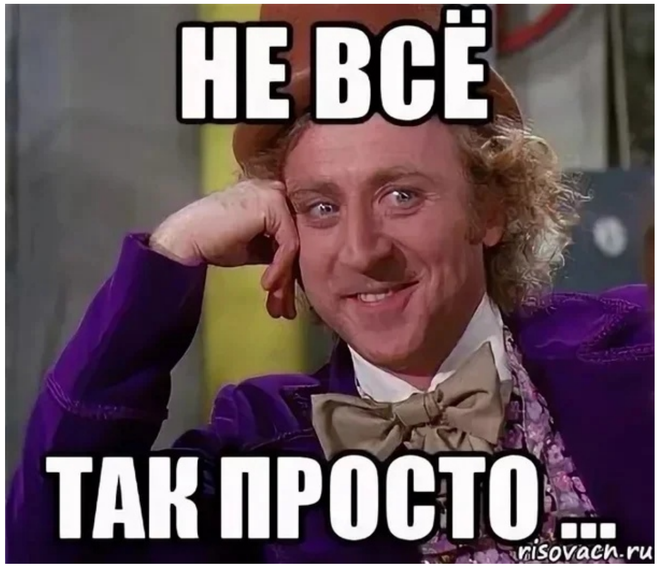
- На своей локальной машине
- На локальной машине другого разработчика
- На тестовом сервере
- На боевом сервере
- На виртуальной машине

Что в итоге мы должны сделать?

1. Скомпилировать и упаковать наши исходники вместе с зависимостями
2. Переместить этот архив на некоторый сервер
3. Запустить приложение со всеми необходимыми конфигурациями
4. Зарабатывать деньги


imagination

Все не так просто…



С чего же начать?

Начнем с кода

- Продукт разрабатывает ни один человек
- Нужно версионировать код
- Нужно код хранить в защищенном месте
- Хотелось бы иметь автоматические проверки и lintor-ы
- А еще CI/CD хотелось бы….

Нам нужен сервис хранения исходного кода

Их есть не мало

Тогда действуем так:

1. Берем любой сервис
2. Размещаем его на наших серверах
3. Размещаем там свой код
4. Настраиваем доступы
5. Настраиваем CI/CD

Ну теперь то можно зарабатывать деньги?

Теперь посмотрим на запуск нашего приложения

Рассмотрим ситуацию:

1. Вы пишите код на своем новеньком Apple MacBook Pro 🥰
2. Сделали feature-request
3. Другой разработчик запускает ваш код на своем Lenovo 2007 💩
4. У него нет каких-то системных пакетов и все падает
5. Говорите ему купить нормальный ноутбук
6. Он обижается и увольняется

Другими словами

1. Ваше приложение может разрабатываться на разных машинах
2. Ваше приложение может запускаться на разных серверах
3. Сервера могут быть очень разные

Что нам бы хотелось?

- Абстрагироваться от конкретной машины
- Декларативно настраивать окружение
- Изолировать приложение от остальных систем
- Чтобы все было максимально просто и понятно

Виртуализация

Сокрытие конкретной реализации за универсальным стандартизированным методом обращения к ресурсам. Иными словами, это создание абстракции над аппаратным обеспечением

Виды виртуализации

- Аппаратная виртуализация
- Виртуализация рабочих столов
- Виртуализация на уровне OS(контейниризация)

Гипервизор

Обеспечивает изолированную среду выполнения для каждой виртуальной машины, а также управляет доступом ВМ и гостевых ОС к аппаратным ресурсам физического сервера

Есть несколько типов

Native, bare-metal

Hosted


Виртуальная машина
Виртуальная машина
Виртуальная машина
Гипервизор
Хостовая ОС
Аппаратное обеспечение

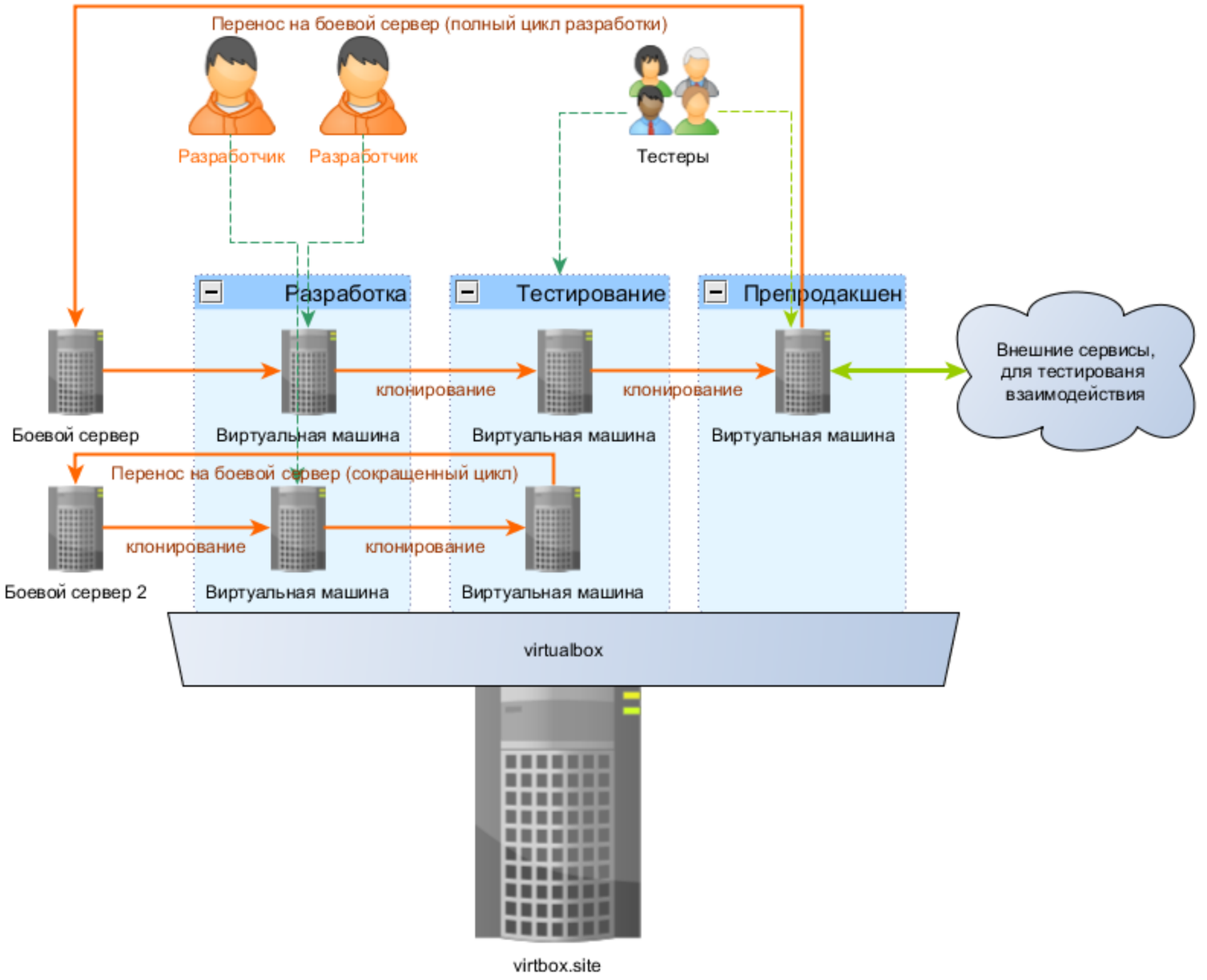
Hybrid


Базовая ОС
Виртуальная машина
Виртуальная машина
Гипервизор
Аппаратное обеспечение

Какие есть имплементации?

VirtualBox

Как тогда выглядит запуск?


Перенос на боевой сервер (полный цикл разработки)
Разработчик
Разработчик
Тестеры
Разработка
Тестирование
Препродакшен
Внешние сервисы, для тестированя взаимодействия
клонирование
клонирование
Боевой сервер
Виртуальная машина
Виртуальная машина
Виртуальная машина
Перенос на боевой сервер (сокращенный цикл)
клонирование
клонирование
Боевой сервер 2
Виртуальная машина
Виртуальная машина
virtualbox
virtbox.site

Что мы получили?

- ПО для абстракции от конкретного аппаратного обеспечения
- Конфигурирование виртуальной машины
- Одинаковое окружение для локальной разработки и для боевой
- Уплотнение окружения на одном аппаратном обеспечении
- Полная изоляция запускаемых приложений

- Очень мощный и большой инструмент

Виртуализация - полное* изолирование окржуния

Минусы

- Очень много весит
- Полностью изолирует ядра и адресное пространство
- Виртуальная машина - полностью готовая операционная система
- Занимает все выделенные ресурсы

Есть ли что-то более подходящее?

Контейниризация

Метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства вместо одного

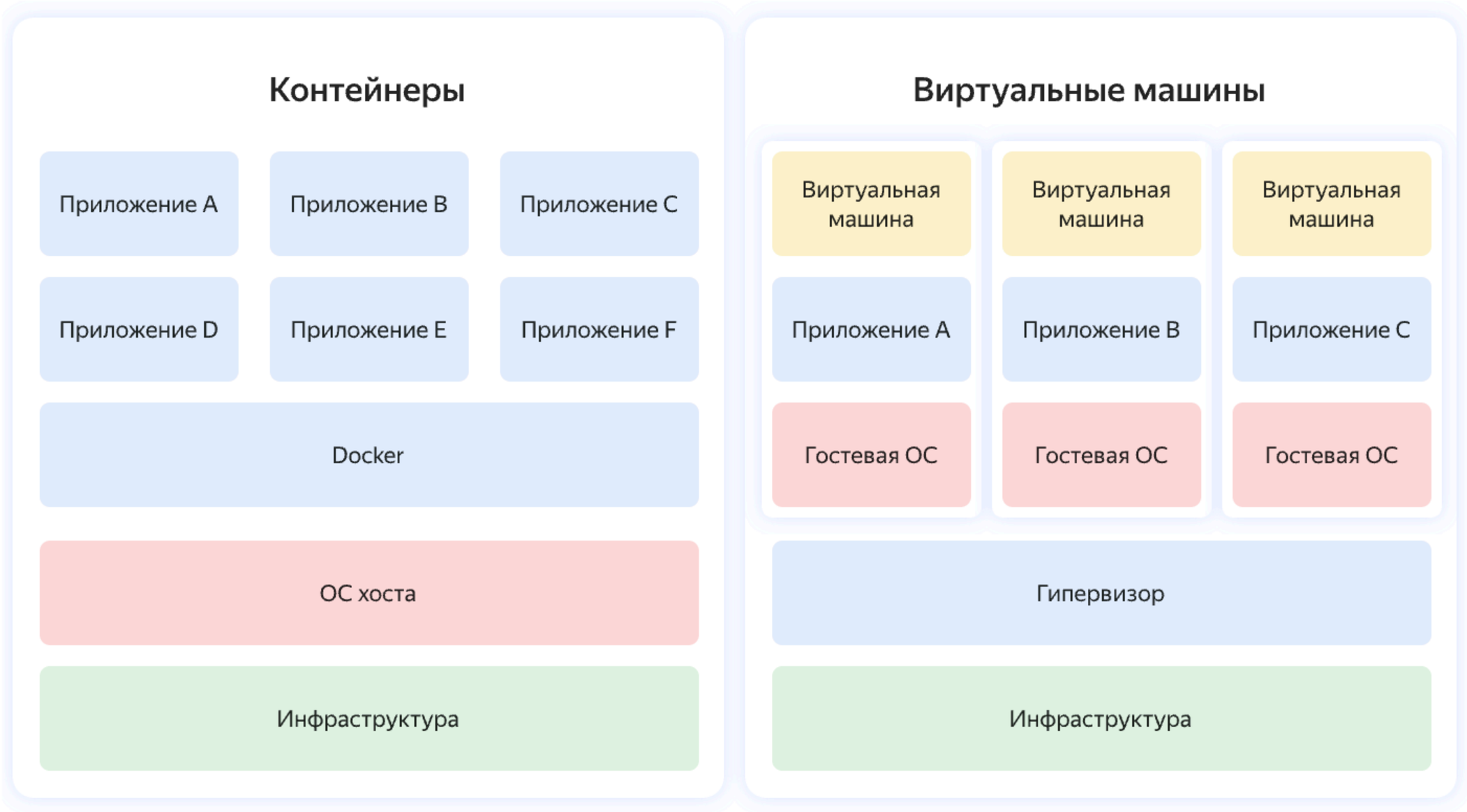
Проще говоря

- Изолируем окружение
- Не изолируем исполнительные ресурсы

Преимущества

- Автоматизированное развертывание
- Перенос программ в современные среды
- Экономия места и памяти
- Повышение безопасности систем
- Увеличение скорости и эффективности
- Перераспределение ресурсов

В чем же различие?


Контейнеры
Приложение A
Приложение B
Приложение C
Приложение D
Приложение E
Приложение F
Docker
ОС хоста
Инфраструктура
Виртуальные машины
Виртуальная машина
Виртуальная машина
Виртуальная машина
Приложение A
Приложение B
Приложение C
Гостевая ОС
Гостевая ОС
Гостевая ОС
Гипервизор
Инфраструктура


docker

Docker

Программное обеспечение для автоматизации развертывания и управления приложениями в среде виртуализации на уровне операционной системы. Позволяет упаковать приложение со всем его окружением и зависимостями в контейнер

Итого:

- Упаковываем наше приложение вместе со всеми зависимости
- Изолируем от остальных систем
- Абстрагируемся от host машины
- Можно удобно управлять и следить
- Декларативно описываем образ
- Есть некоторое хранилище образов

Кажется, это то, что нам нужно!

И как теперь с этим всем работать?

Нашими контейнерами нужно как-то управлять…

Оркестарция контейнеров

- Автоматизировать установку приложений и сервисов, упакованных в контейнер
- Автоматизировать развертывание новых версий, запись логов, мониторинг и отладку
- Управлять масштабированием
- Настраивать автономную работу контейнеров в зависимости от нагрузки на систему

Какое есть популярное решение?

kubernetes

Тема очень большая….

Оставим на другую лекцию

Вернемся к нашему приложению

Что мы сейчас имеем:

- Храним и управляем нашим кодом в системе контроля версий
- Настроили CI/CD процессы
- Упаковываем наше приложение в Docker образ
- Управляем нашим контейнерами через k8s

Ура! Можно зарабатывать деньги!

А потом мы начали получать жалобы

А как нам следить за эксплуатацией?

Нам нужно:

- Собирать логи с наших приложений
- Собирать метрики с наших приложений
- Следить за состоянием наших машин
- Получать оповещения если что-то пошло не так
- Визуализировать все метрики

Нам нужен сервис хранения временных рядов

Time Series (временные ряды)

Данные, которые изменяются во времени

Особенности

- Время фиксации
- Характеристика процесса, которые называют уровнями ряда
- Append-only режим
- Записи не рассматриваются отдельно друг от друга

Какие есть решения?


influxdb

Некоторые факты:

- Написана на языке Go
- Релиз в октябре 2013
- Возможность работы в кластерном режиме
- SQL-подобный язык запросов
- Есть удобный графический интерфейс

Получается нам нужно самим организовывать эксплуатацию?

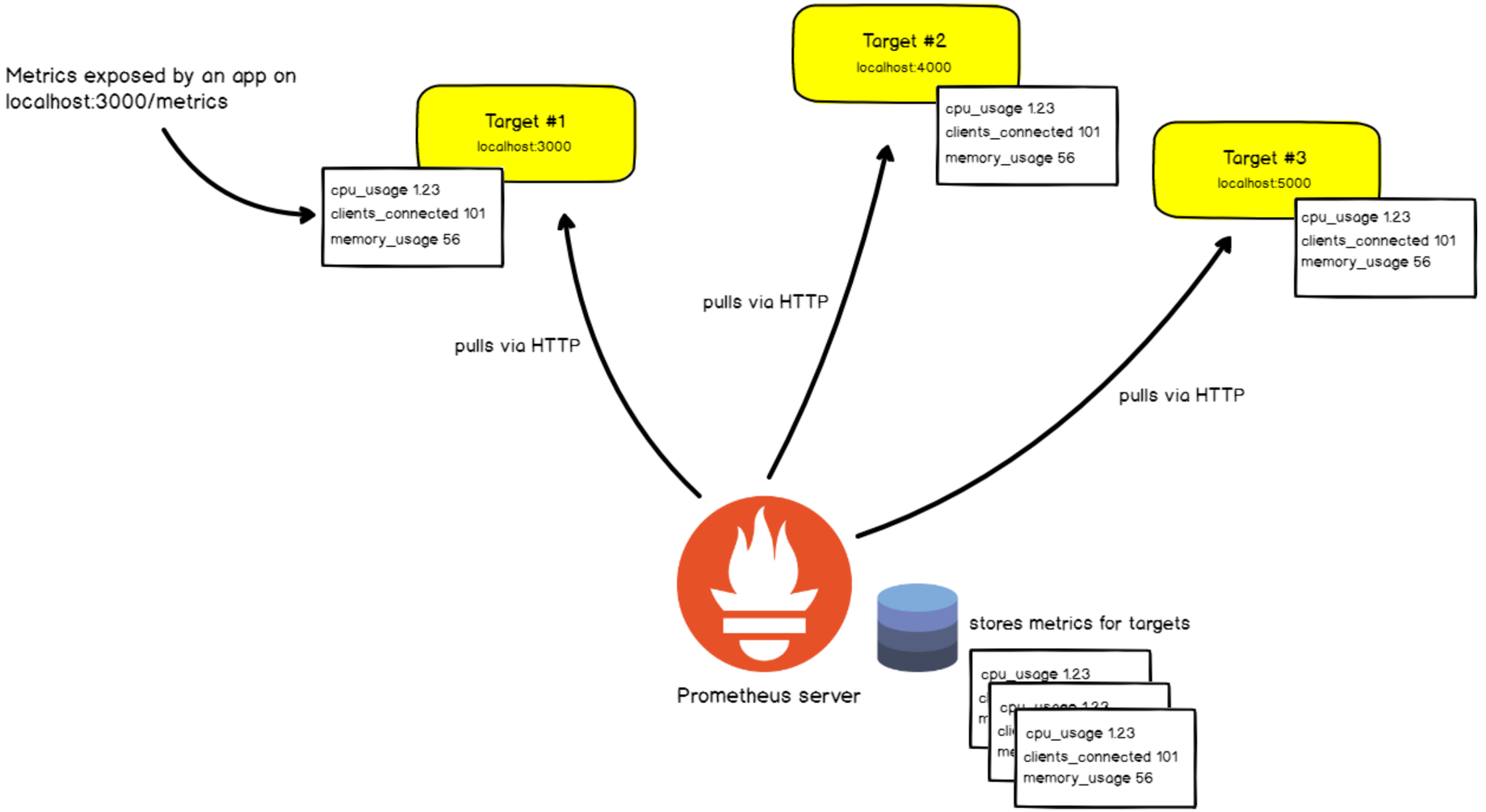

Prometheus

Prometheus

База данных временных рядов. Содержит большую экосистему инструментов для мониторинга самых разных систем

Как он работает?

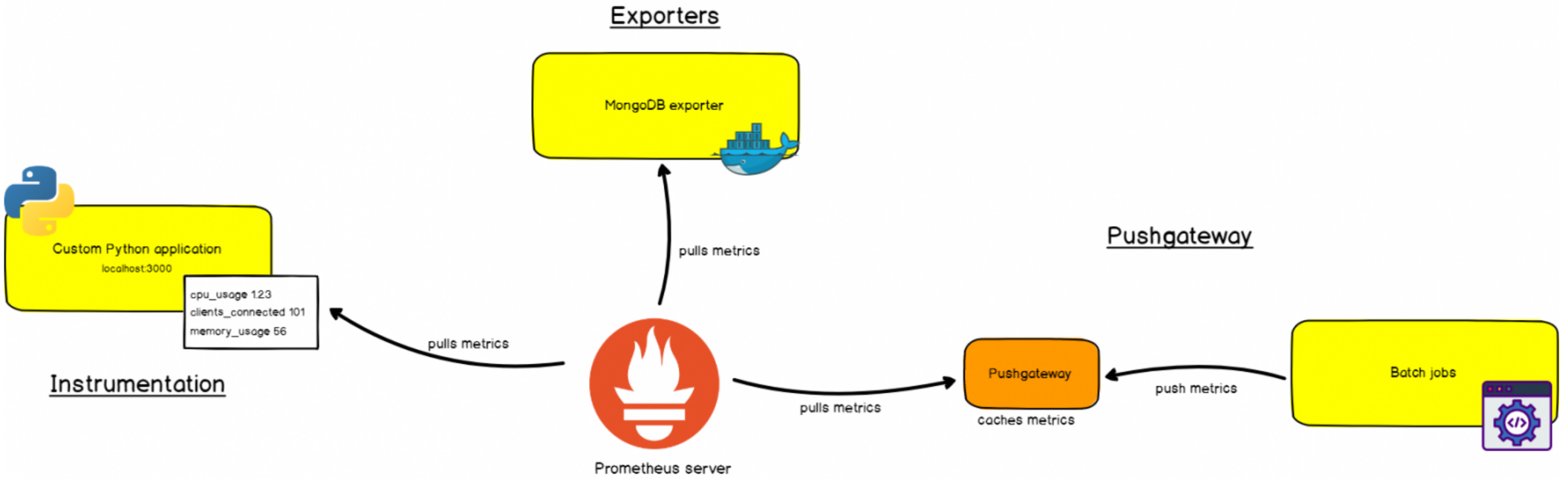
What does Prometheus do?



Как метрики могут извлекаться?

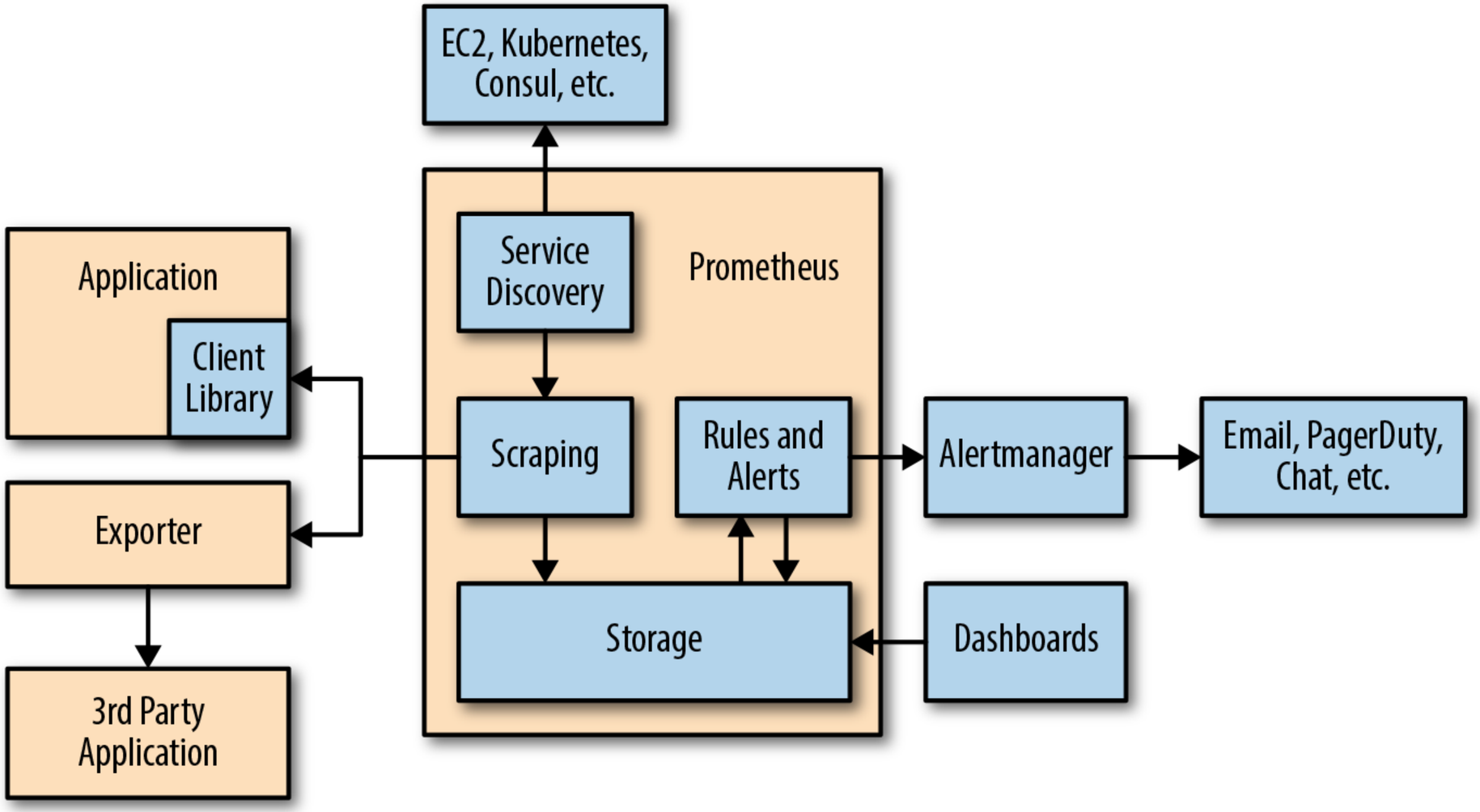
- Инструментирование приложения
- Использование готовых экспортеров
- Использование Pushgateway

Ways to gather metrics in Prometheus



Из чего состоит инфраструктура?

- AlertManager
- Визуализация данных
- Обнаружение сервисов


EC2, Kubernetes, Consul, etc.
Service Discovery
Prometheus
Application
Client Library
Scraping
Rules and Alerts
Alertmanager
Email, PagerDuty, Chat, etc.
Exporter
Storage
Dashboards
3rd Party Application

Отлично, теперь мы собираем метрики со всего чего только можно!

А как их визуализировать?


Grafana

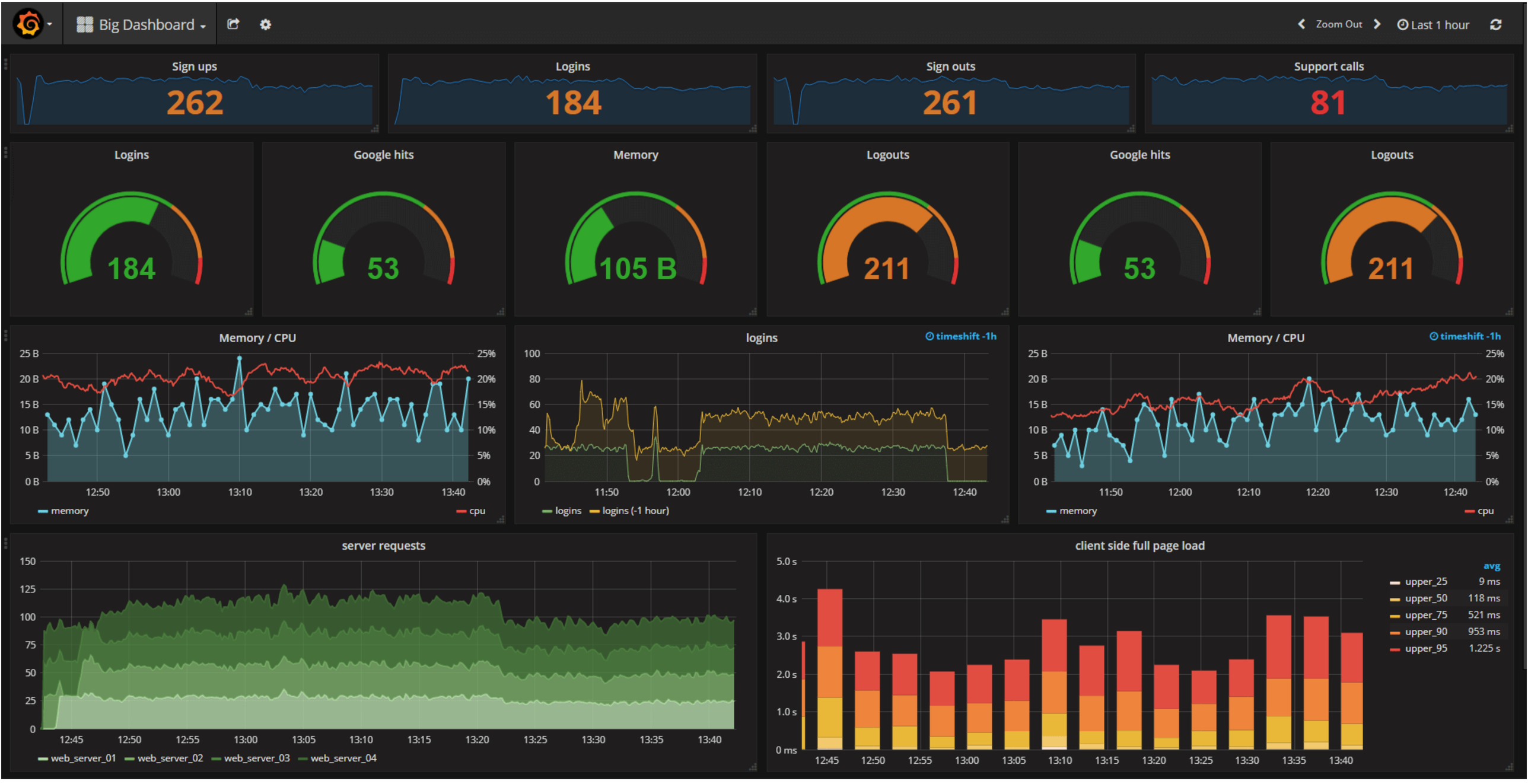
Grafana

Платформа с открытым исходным кодом для визуализации, мониторинга и анализа данных

Преимущества

- Поддерживает множество источников данных
- Очень большое Community
- Очень гибкий инструментарий для визуализации
- Динамические dashboard
- Mixed data source

Примеры


Big Dashboard
Zoom Out
Last 1 hour
Sign ups
262
Logins
184
Sign outs
261
Support calls
81
Logins
184
Google hits
53
Memory
105 B
Logouts
211
Google hits
53
Logouts
211
Memory / CPU
25 B
20 B
15 B
10 B
5 B
0 B
25%
20%
15%
10%
5%
0%
12:50
13:00
13:10
13:20
13:30
13:40
memory
cpu
logins
timeshift -1h
100
80
60
40
20
0
11:50
12:00
12:10
12:20
12:30
12:40
logins
logins (-1 hour)
Memory / CPU
timeshift -1h
memory
cpu
server requests
150
125
100
75
50
25
0
12:45
12:50
12:55
13:00
13:05
13:10
13:15
13:20
13:25
13:30
13:35
13:40
web_server_01
web_server_02
web_server_03
web_server_04
client side full page load
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
0 ms
avg
upper_25 9 ms
upper_50 118 ms
upper_75 521 ms
upper_90 953 ms
upper_95 1.225 s

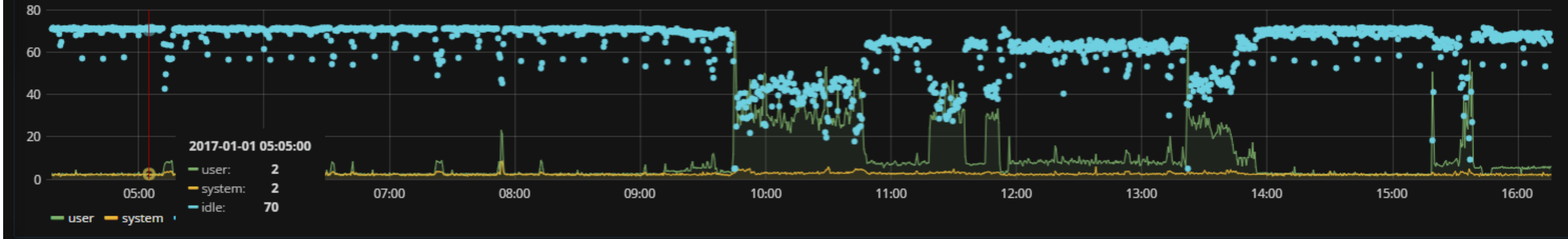
HOME Dashboard

Zoom Out | Last 12 hours | Refresh every 30s

Hostname: jupiter

Uptime	Free space	Total HDD space	Plex Streams
19 week	N/A	11.91 TB	0

Host / CPU



Host / HDD

Time	path	FREE	USED	TOTAL
16:18	/mnt/external/backups	1.38 TB	448.66 GB	1.92 TB
16:18	/mnt/internal/raid	111.00 GB	11.52 TB	11.63 TB
16:18	/mnt/internal/sata	4.59 GB	1.92 TB	1.92 TB

Host / Temp

Time	feature	TEMP
16:18	core_1	39.00 °C
16:18	core_0	37.00 °C

Host / HDD / temp

Metric	Min	Max	Avg	Current
sdc	26.00 °C	26.00 °C	26.00 °C	26.00 °C
sdd	33.00 °C	33.00 °C	33.00 °C	33.00 °C
sde	36.00 °C	36.00 °C	36.00 °C	36.00 °C
sdg	36.00 °C	36.00 °C	36.00 °C	36.00 °C
sdi	35.00 °C	35.00 °C	35.00 °C	35.00 °C
sdj	35.00 °C	35.00 °C	35.00 °C	35.00 °C
sdk	35.00 °C	35.00 °C	35.00 °C	35.00 °C

Réinitialiser 100 %

Вроде все сделали…

Теперь мы:

1. Храним наш код в VCS
2. Используем CI/CD для доставки и проверок
3. Разворачиваем наш код в Docker
4. Наши контейнеры оркестрирует Kubernetes
5. Собираем метрики с помощью Prometheus и Grafana

Теперь вы DevOps!

Что такое DevOps?

DevOps

Это методология взаимодействия всех участников цикла разработки и взаимная интеграция их рабочих процессов, которая помогает обеспечить качество продукта.

Более простыми словами

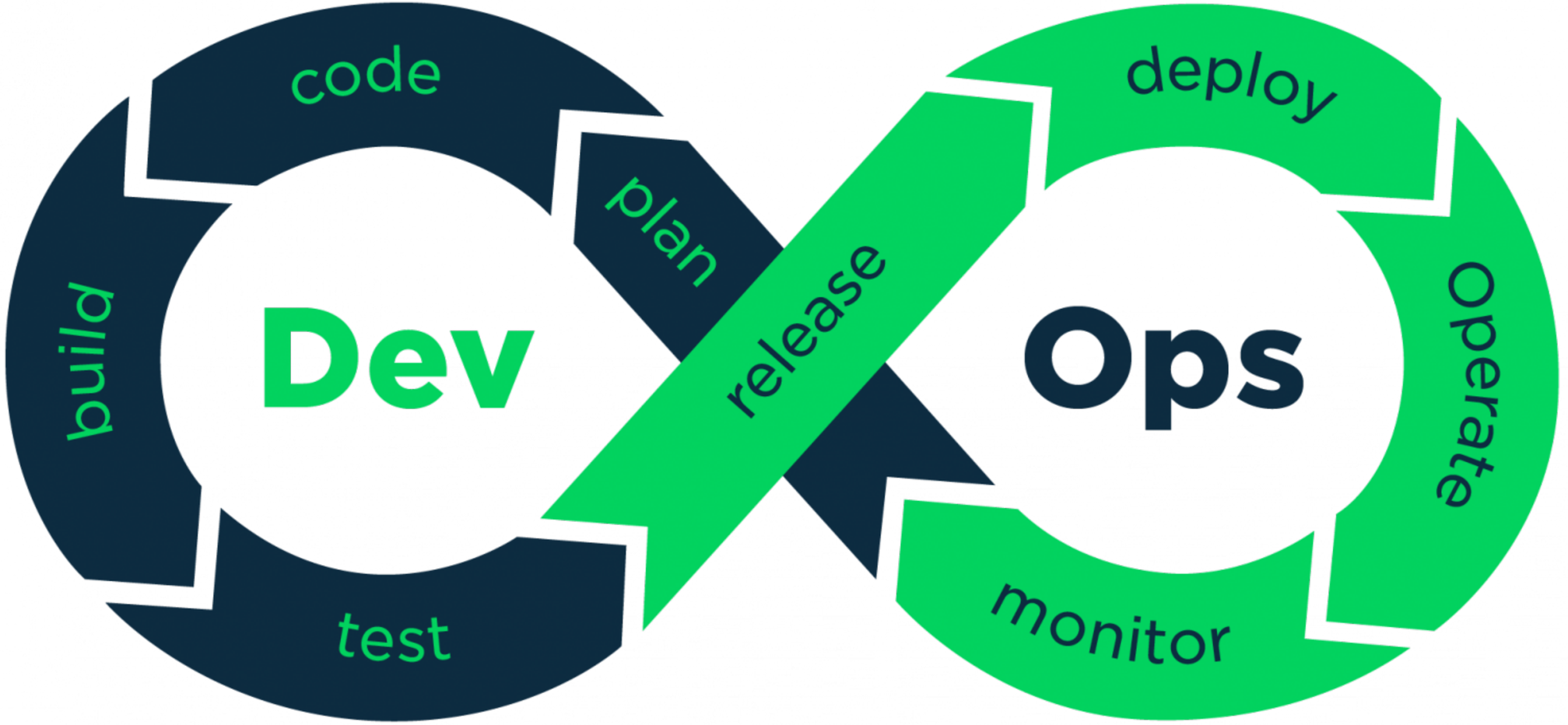
Методология DevOps предназначена для эффективной организации, создания и обновления программных продуктов и услуг

В чем это выражается?

- Увеличение скорости разработки
- Улучшение сотрудничества и коммуникации
- Улучшение качества продукта
- Сокращения затрат и оптимизация ресурсов

DevOps - это сочетание разработки(Dev) и поддержки(Ops) продуктов.


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net


code
plan
build
Dev
release
deploy
Ops
operate
test
monitor

Итог:

Все рассмотренные решения и сервисы улучшают процесс разработки и эксплуатации

Вопросы?

Материалы

Их очень много и они будут в README.md

Спасибо за внимание!