Мониторинг приложения и инфраструктуры

План

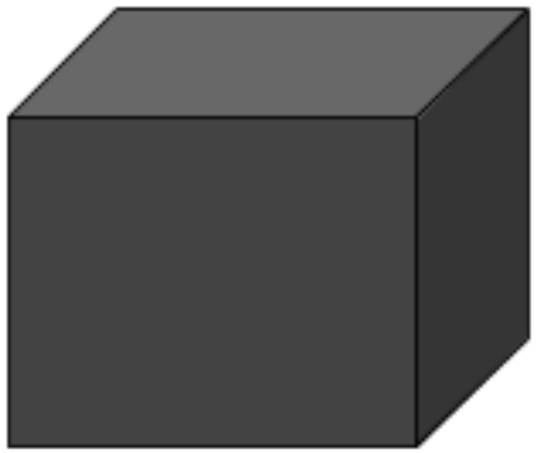
- Метрики, основные типы метрик
- Мониторинг инфраструктуры
- Мониторинг приложения
- Мониторинг бизнес логики

Метрики

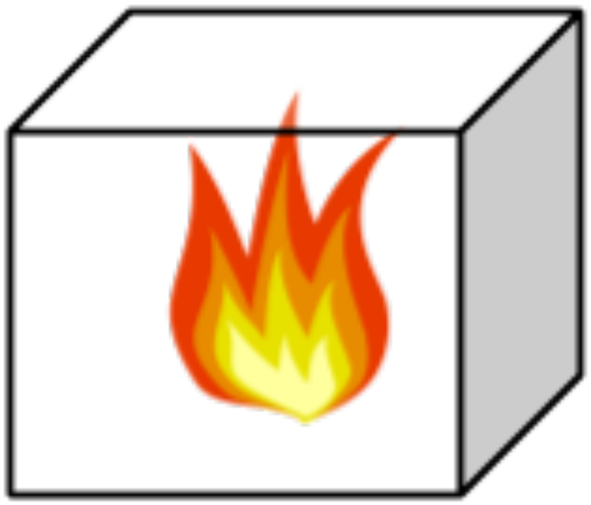
Blackbox + Whitebox

- Простого пинга недостаточно
- Обе модели должны дополнять друг друга

Вид снаружи



Вид изнутри



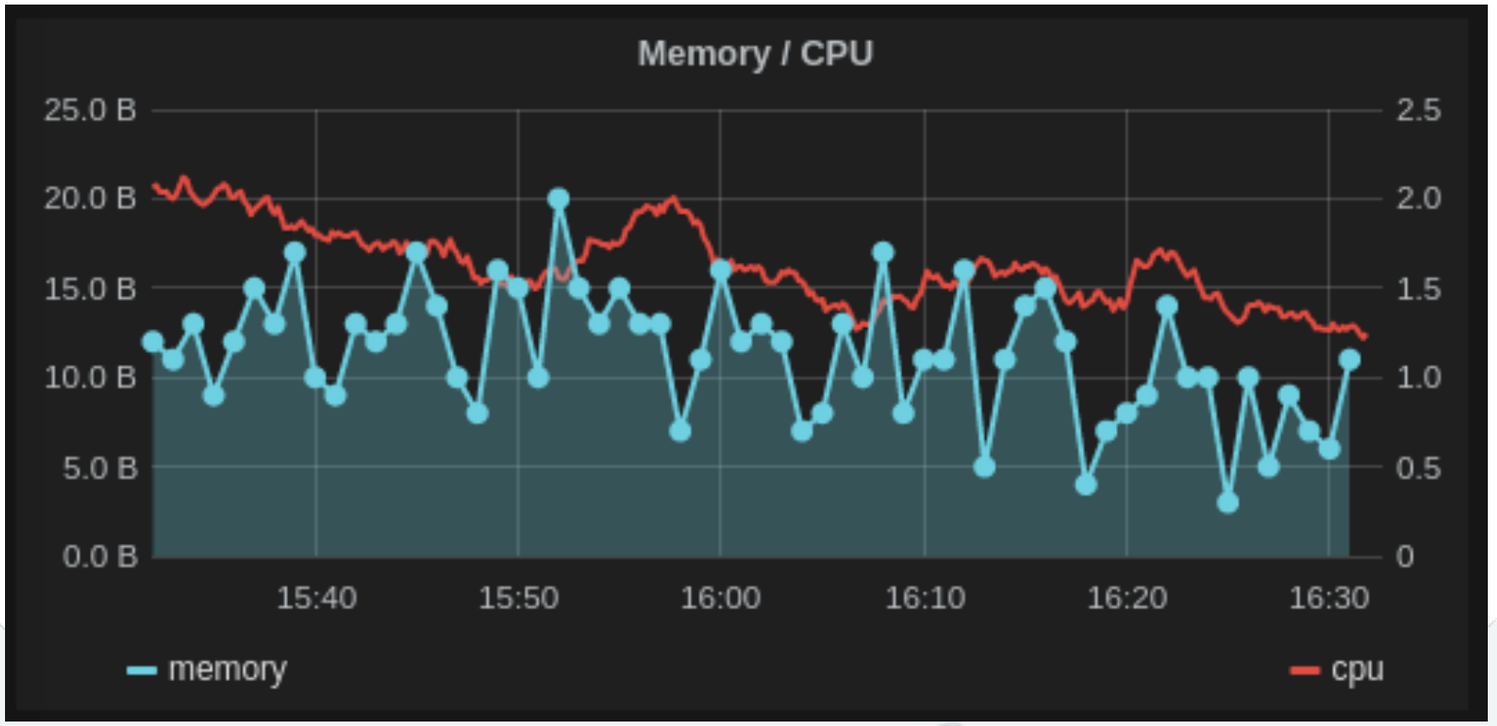
Метрика

- Описывает свойство наблюдаемой системы в определенный момент времени
- Представляет собой пару "ключ = значение"
- Пример: users_online = 15
- Собираются системой мониторинга с определенным временным интервалом

Временной ряд (time-series)

- Набор значений метрик(и) за определенный период времени

- Примеры TSDBs: InfluxDB, Graphite, Prometheus, Elasticsearch

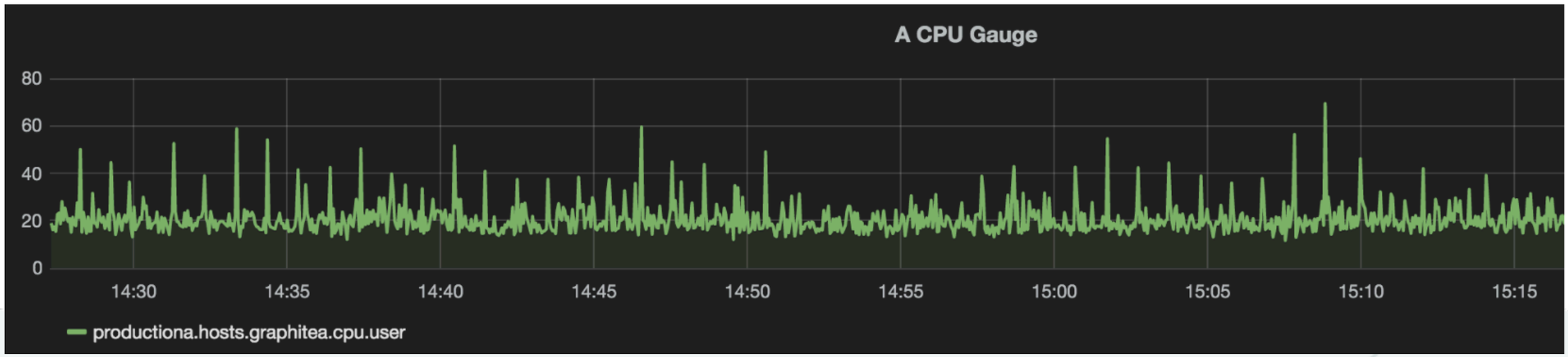


Основные типы метрик

- Gauges
- Counters
- Timers

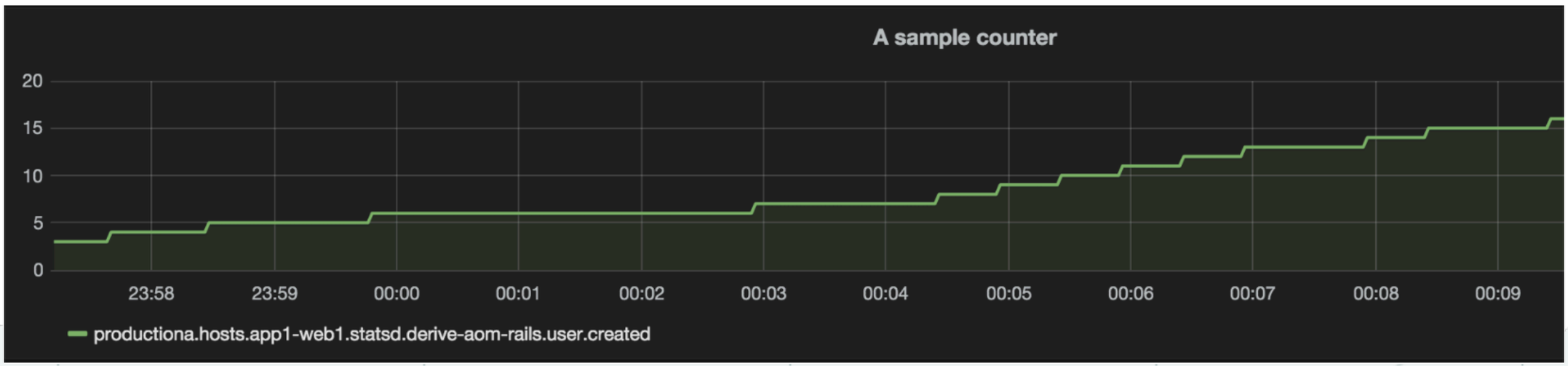
Gauge

- Численное значение, которое может непредсказуемо меняться со временем
- Может увеличиваться и уменьшаться
- Примеры: использование CPU, memory, disk; количество пользователей онлайн



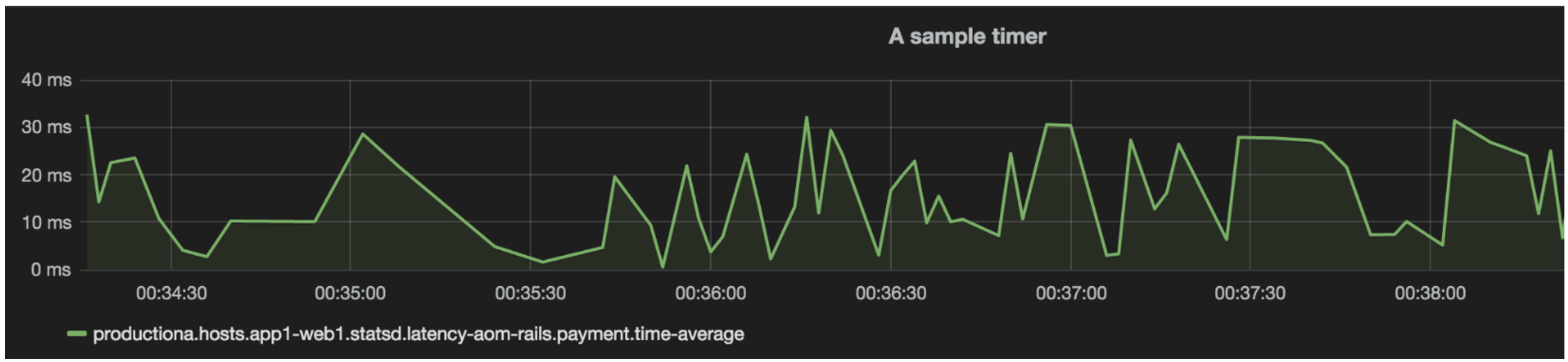
Counter

- Счетчик - численное значение, которое может расти со временем
- Может только увеличиваться, но возможен сброс на ноль
- Примеры: uptime, количество HTTP запросов, количество регистраций пользователей, число продаж

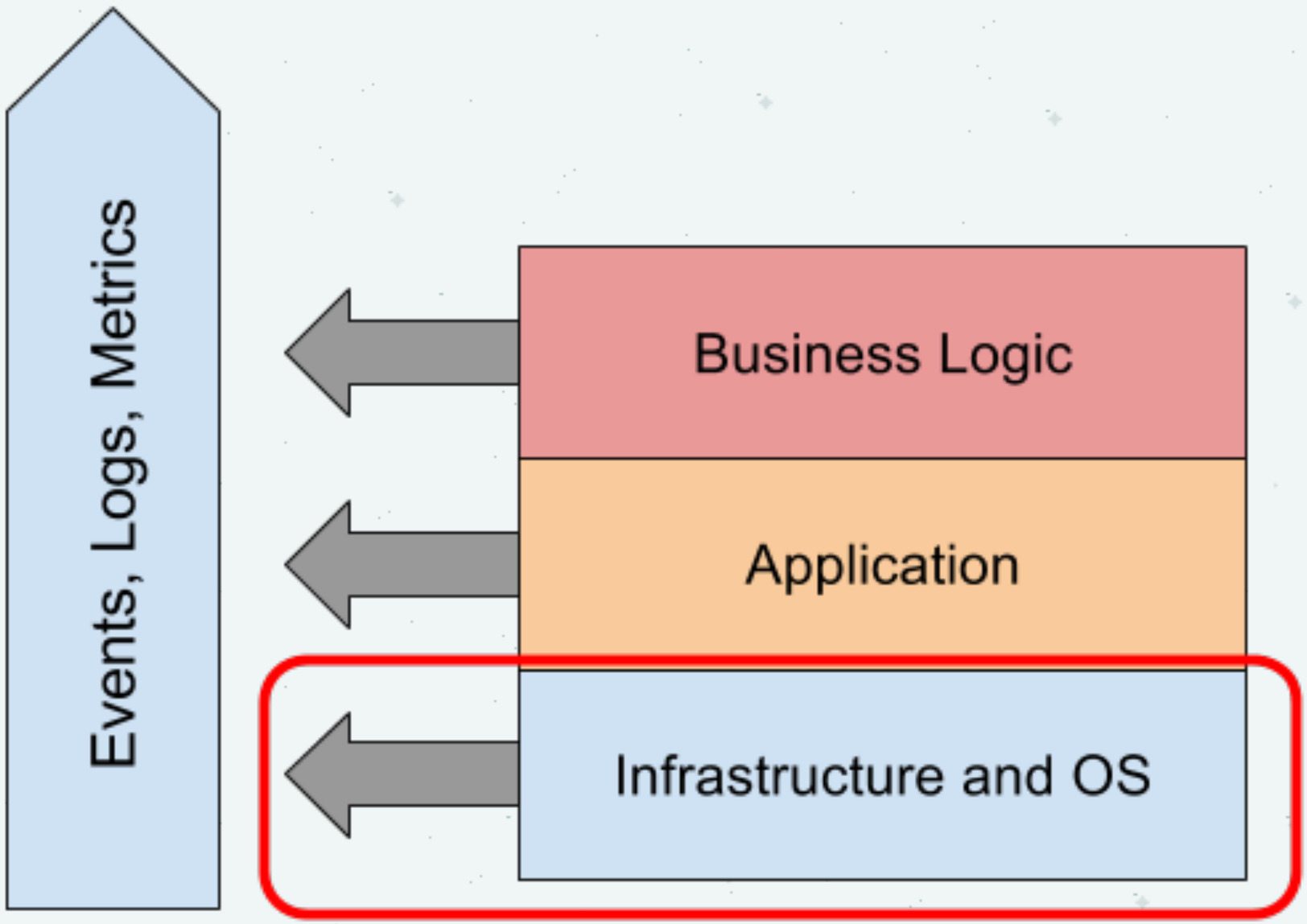


Timers

- Позволяют отследить длительность операции
- Примеры: время ответа веб сервера, длительность запроса к БД



Мониторинг инфраструктуры



Метрики хоста

- CPU
- Memory
- Processes
- Disk
- Network
- etc

Чем собирать?

- Агенты мониторинга специфичные системе мониторинга
- Более универсальные инструменты с плагинами: collectd
- Сервисы платформы: Stackdriver (GCP), CloudWatch (AWS)

Метрики Docker контейнера

- CPU
- Memory
- Network
- Block I/O
- \+ Docker daemon

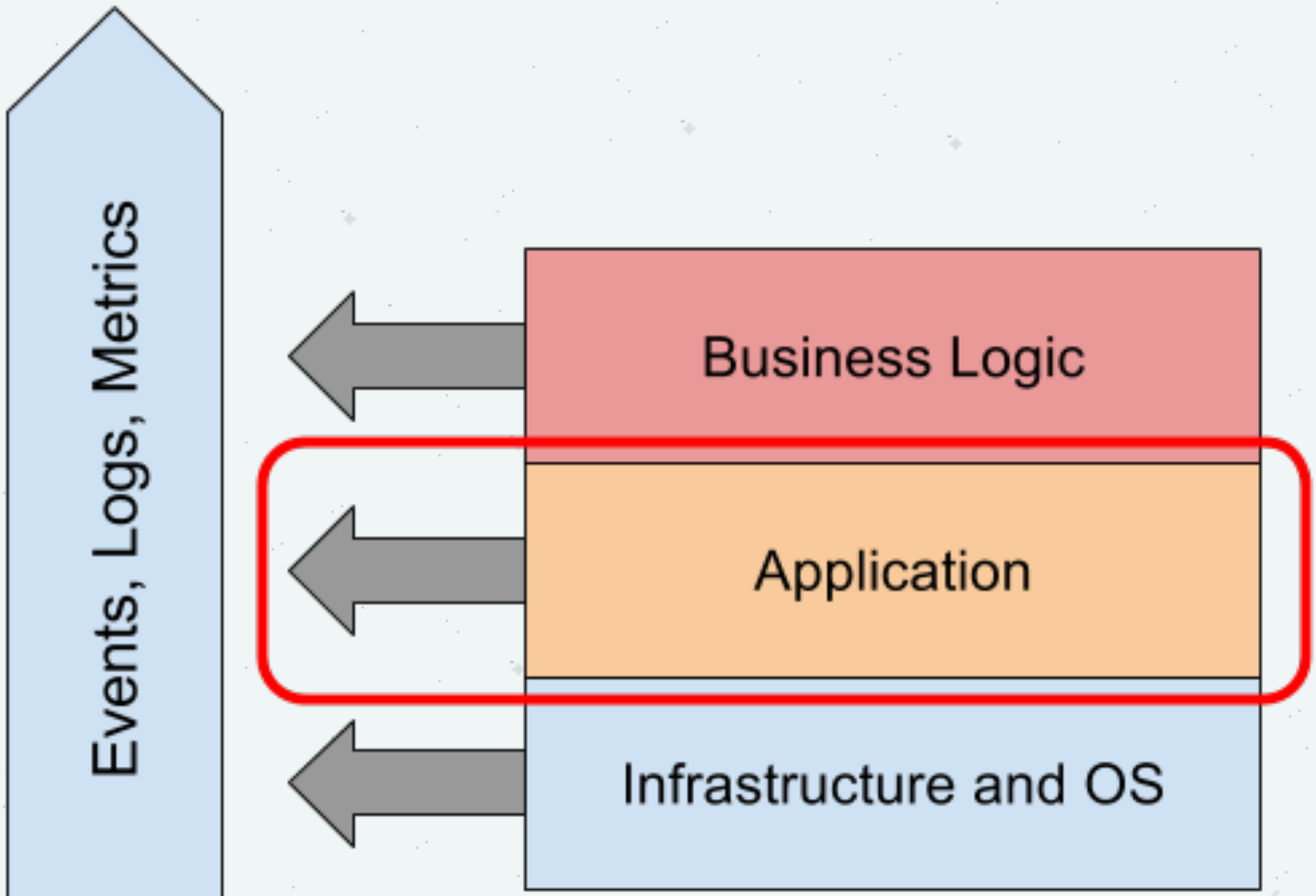
Чем собирать?

- команда docker stats
- cAdvisor
- Heapster
- collectd
- etc

Метрики сервисов

- БД, очереди
- Load balancer
- Сервер приложения
- Сторонние сервисы
- **Все, от чего зависит стабильность работы вашего продукта**

Мониторинг приложения



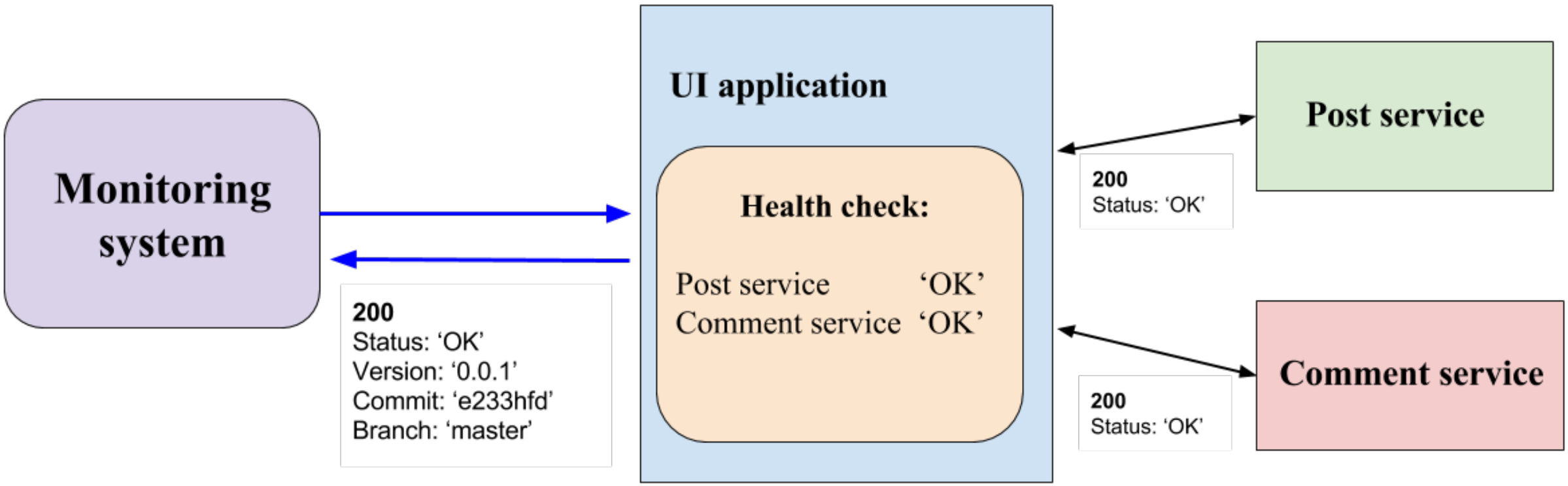
Метрики приложения

- Позволяют узнать о состоянии и производительности (performance) кода
- Описываются в самом коде приложения
- Отражают опыт использования приложения конечным пользователем
- Примеры метрик: время ответа на запросы, количество неудачных логинов пользователей

Health check

- Проверка с целью убедиться, что наше приложение доступно и полноценно работает
- Возврат кода 200 не означает, что приложение работает, как ожидается

Пример


Monitoring system
UI application
Health check:
Post service ‘OK’
Comment service ‘OK’
Post service
Comment service
200
Status: ‘OK’
200
Status: ‘OK’
Version: ‘0.0.1’
Commit: ‘e233hfd’
Branch: ‘master’
200
Status: ‘OK’

Как создавать метрики приложения?

- Нужен простой стандарт для создания метрик - общая библиотека
- Многие системы мониторинга имеют готовые клиентские библиотеки

Пример

Время поиска пользователя:

```
def show
  STATSD.time("user.find") do
    @user = User.find(params[:id])
  end
  unless current_user.admin?
    unless @user == current_user
      redirect_to :back, :alert => "Access denied."
    end
  end
end
```

Пример reddit app

На GitHub есть Ruby библиотека для создания метрик для сбора Прометеем.

Простой счетчик:

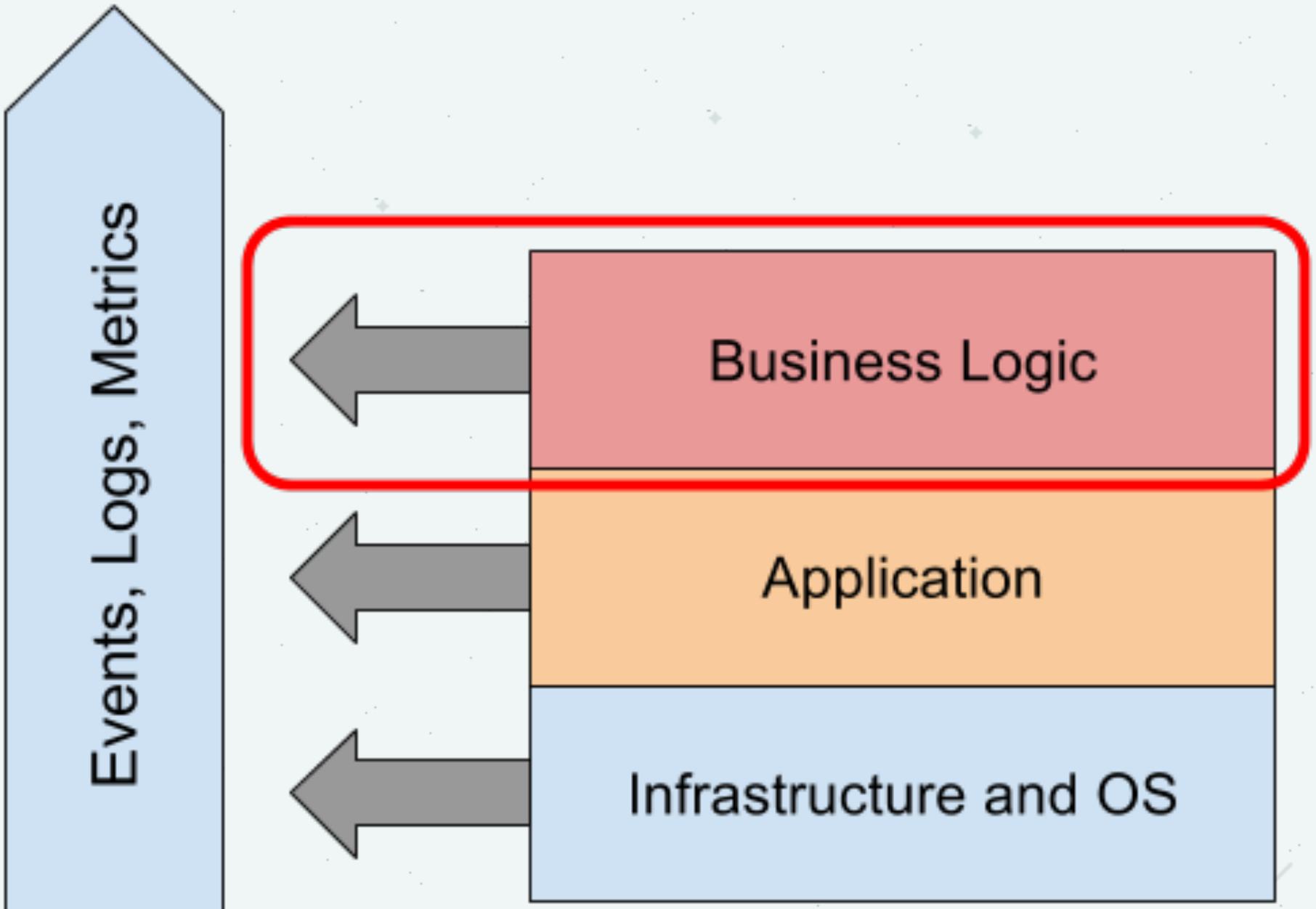
```
require 'prometheus/client'

prometheus = Prometheus::Client.registry

# create a new counter metric
http_requests = Prometheus::Client::Counter.new(:http_requests, 'A
counter of HTTP requests made')
prometheus.register(http_requests)

# start using the counter
http_requests.increment
```

Создание метрик

```
+prometheus = Prometheus::Client.registry
+
+comment_health_gauge = Prometheus::Client::Gauge.new(:comment_health, 'Health status of Comment
+comment_health_db_gauge = Prometheus::Client::Gauge.new(:comment_health_mongo_availability, 'Che
+prometheus.register(comment_health_gauge)
+prometheus.register(comment_health_db_gauge)
+
+## Schedule healthcheck function
+build_info=File.readlines('build_info.txt')
+
+scheduler = Rufus::Scheduler.new
+
+scheduler.every '3s' do
+  check = JSON.parse(healthcheck(mongo_host, mongo_port))
+  comment_health_gauge.set({ version: check['version'], commit_hash: build_info[0].strip, branch
+  comment_health_db_gauge.set({ version: check['version'], commit_hash: build_info[0].strip, bra
+end
```

Установка значений

Мониторинг бизнес логики



Бизнес метрики

- Выступают средством проверки бизнес идей
- Индикатор успеха приложения среди пользователей
- Примеры метрик: количество регистраций за последний месяц, количество продаж, значение средней покупки

Пример

```
include Metric

  def pay_user(user, amount)
    pay(user.account, amount)
    Metric.increment 'payment'
    Metric.increment "payment.amount, #{amount.to_i}"
    Metric.increment "payment.country, #{user.country}"
    send_payment_notification(user.email)
end
```

И в конце...

Collecting data is cheap, but not having it when you need it can be expensive, so you should instrument everything, and collect all the useful data you reasonably can.

"monitoring 101" Datadog блог

To be continued ...