

Мониторинг компонентов Kubernetes и приложений, работающих в нем

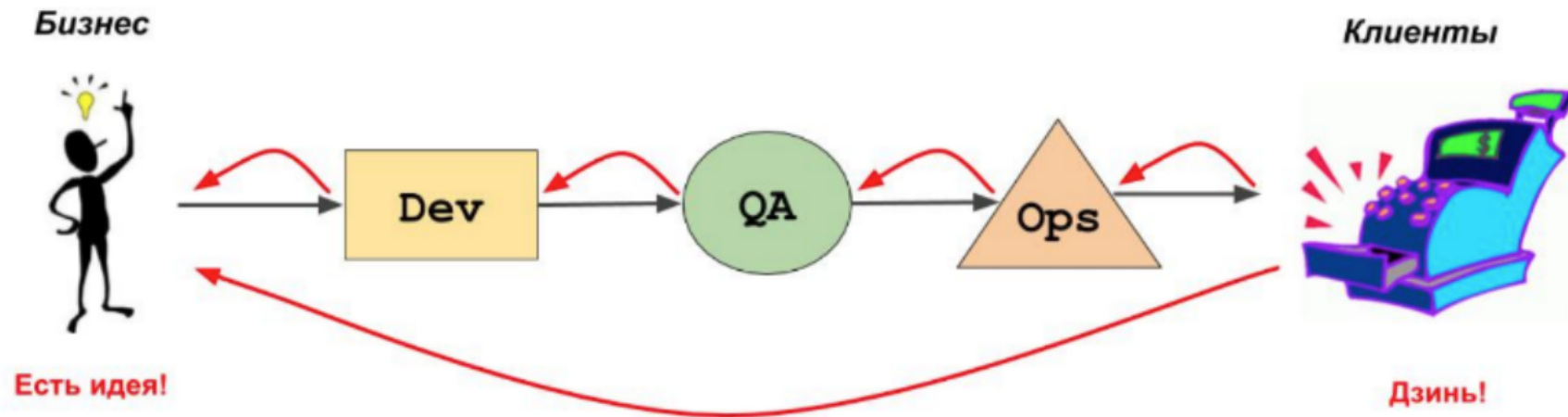
Не забудь включить запись!



План

- Принципы мониторинга
- Kubernetes Probes
- Мониторинг Kubernetes
- Prometheus Operator
- Визуализация

Второй путь DevOps



Что отслеживать

- Наше приложение (продукт)
- Ресурсы, на которых работает наше приложение (платформа)

Паттерны мониторинга

RED Pattern

Применяется для мониторинга сервисов (продуктов)

- **Rate** - Количество запросов в секунду (RPS)
- **Errors** - Процент запросов, завершившихся ошибкой
- **Duration** - Длительность запроса (latency)

USE Pattern

Применяется для мониторинга платформы

- **Utilisation** - среднее время, когда ресурс был занят обслуживанием запроса
- **Saturation** - длина очереди запросов, ожидающих обработки
- **Errors** - процент операций над ресурсом, завершившихся ошибкой

Kubernetes Probes

Probes

Периодические проверки pod'а на жизнеспособность.

Как узнать, что сервис “жив” и готов к работе?

- **ExecAction** - выполнить команду и ждать exit code 0
- **TCPSocketAction** - проверить, что TCP-порт открыт
- **HTTPGetAction** - отправить HTTP GET-запрос

Liveness probes

- Проверяет что приложение запущено и "живо"
- Если probe неуспешна - перезапуск контейнера*

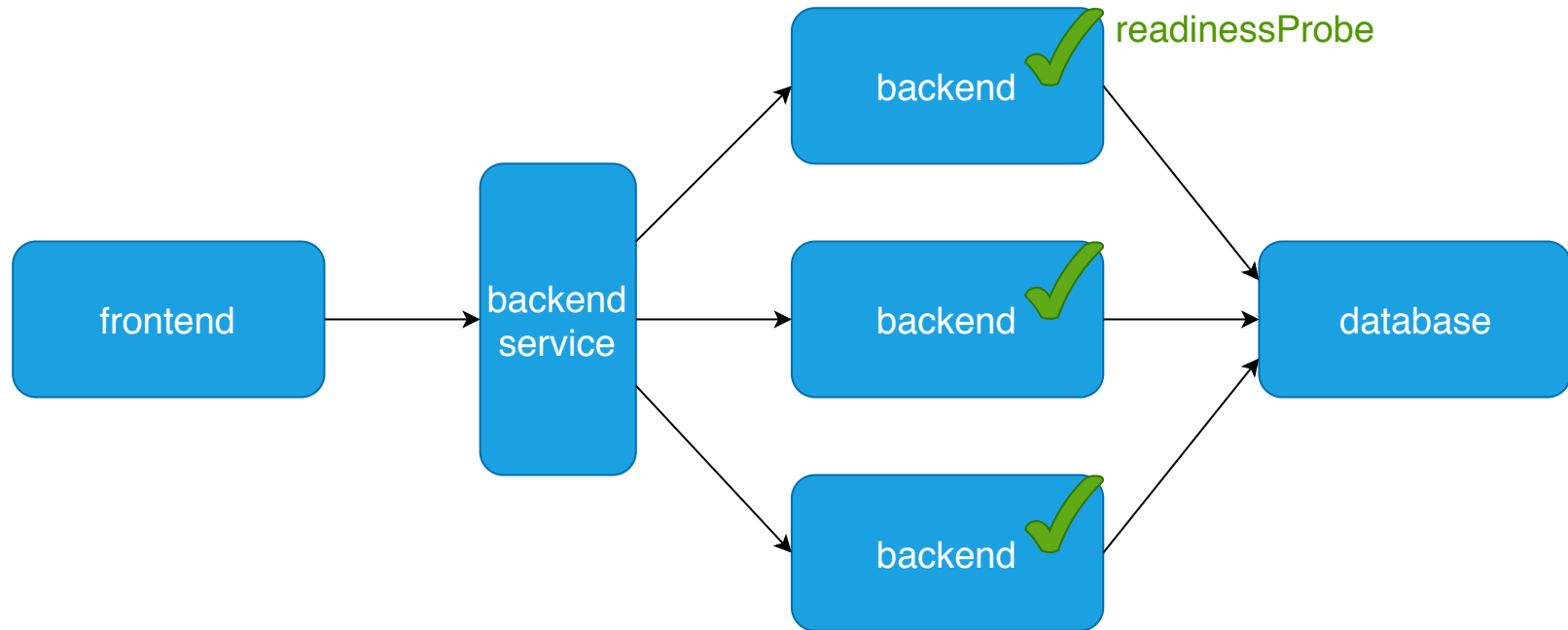
```
1 apiVersion: v1
2 kind: Pod
3 ...
4 spec:
5   containers:
6     - name: liveness
7       image: liveness
8       livenessProbe:
9         tcpSocket:
10          port: 8080
11         initialDelaySeconds: 5
12         periodSeconds: 10
```

Readiness probes

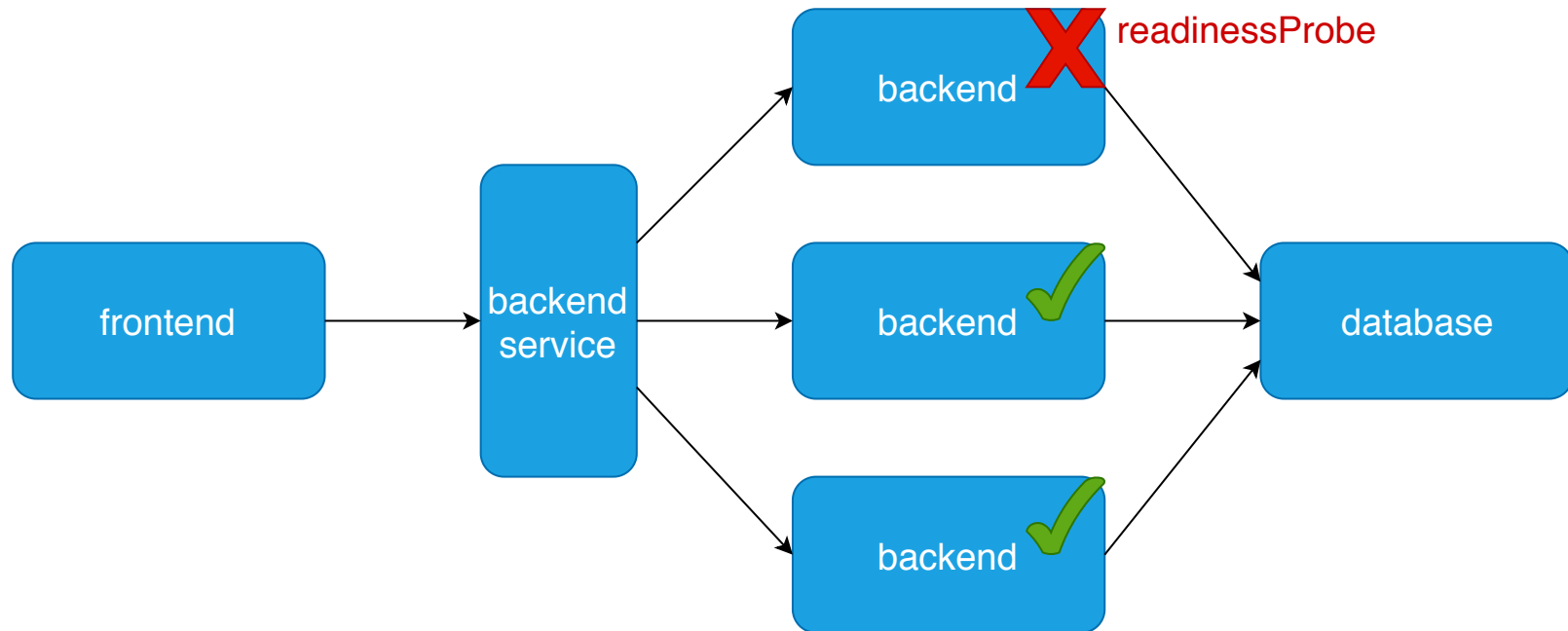
- Проверяет что приложение готово обслуживать запросы
- Если probe неуспешна - удаление pod из балансировки

```
1 apiVersion: v1
2 kind: Pod
3 ...
4 spec:
5   containers:
6     - name: readiness
7       image: readiness
8       readinessProbe:
9         httpGet:
10           path: /healthz
11           port: 8080
12         initialDelaySeconds: 3
13         periodSeconds: 3
```

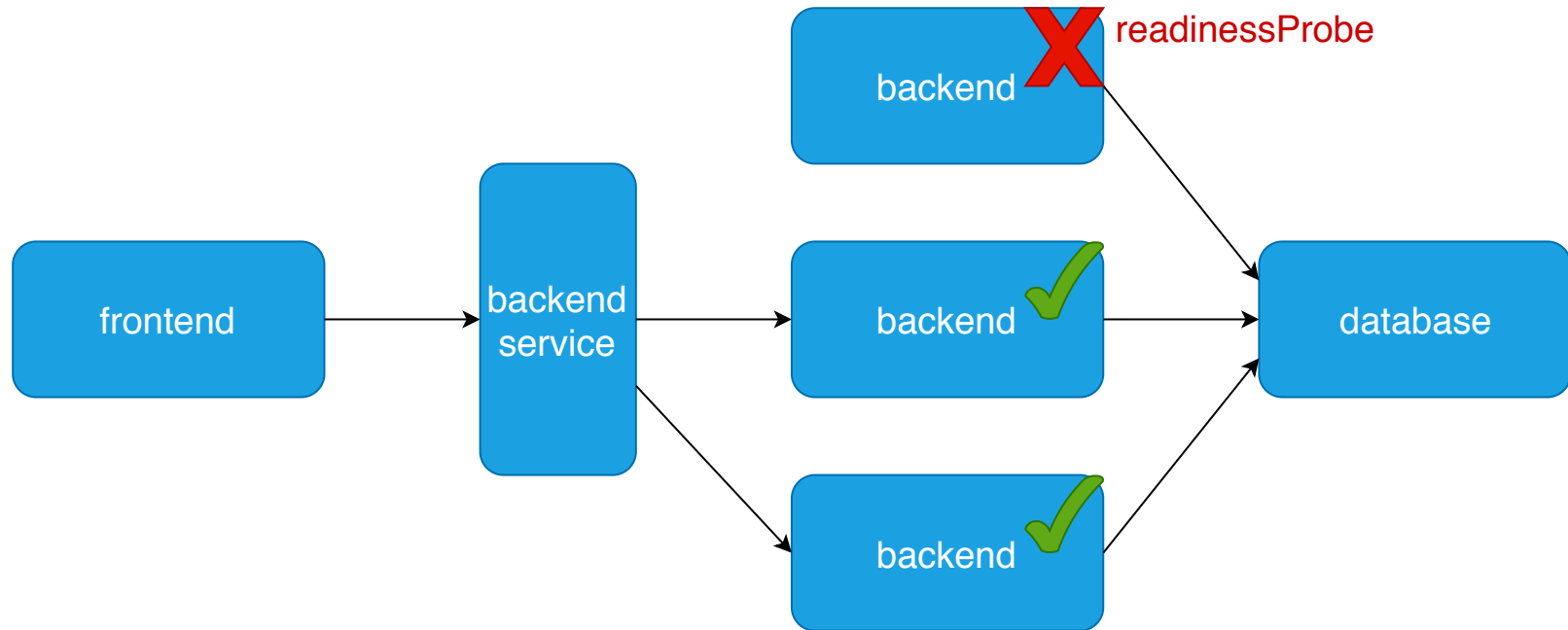
Readiness probes: Good



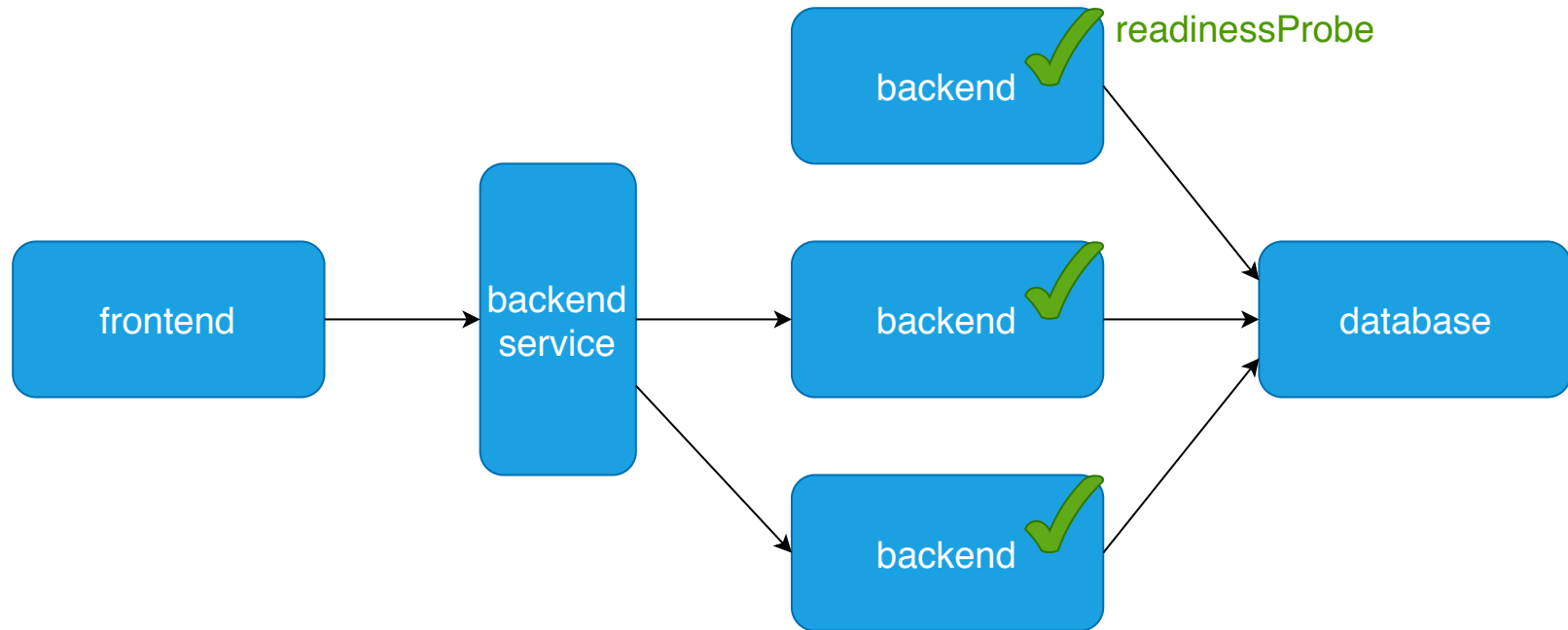
Readiness probes: Failed



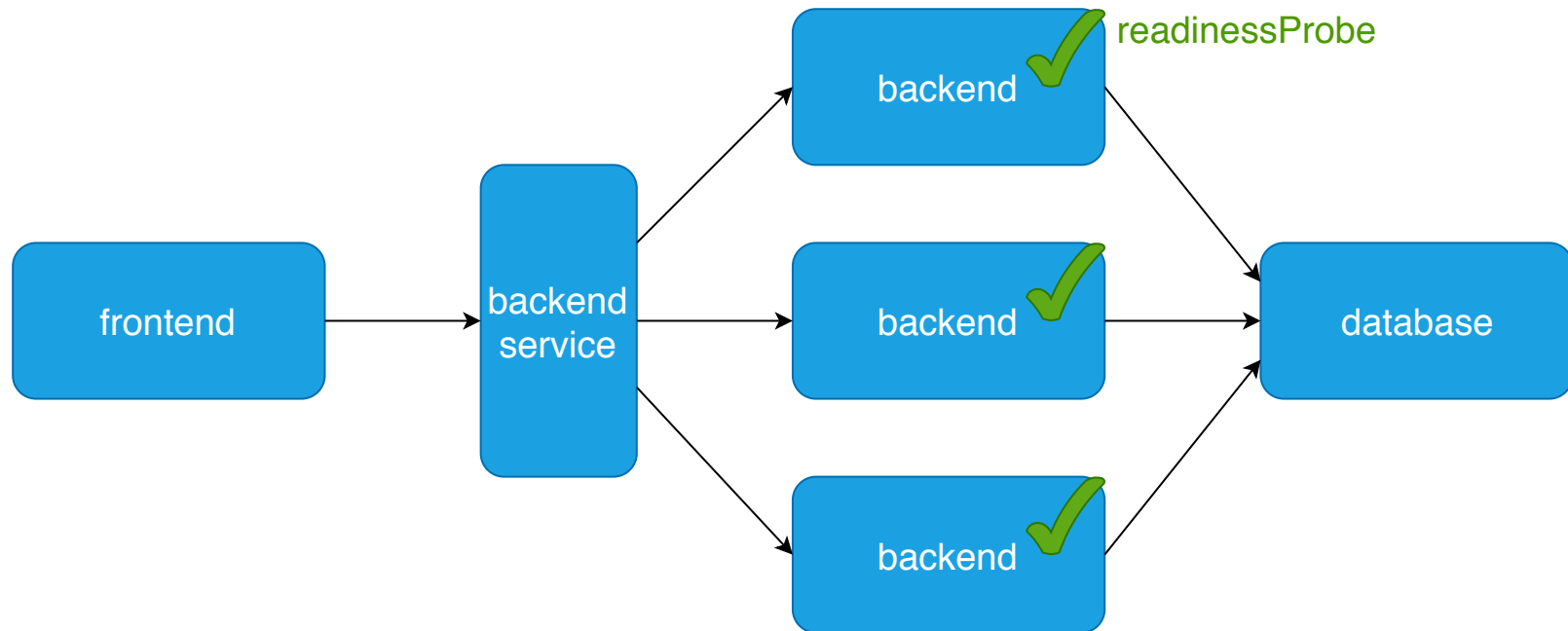
Readiness probes: Disabled



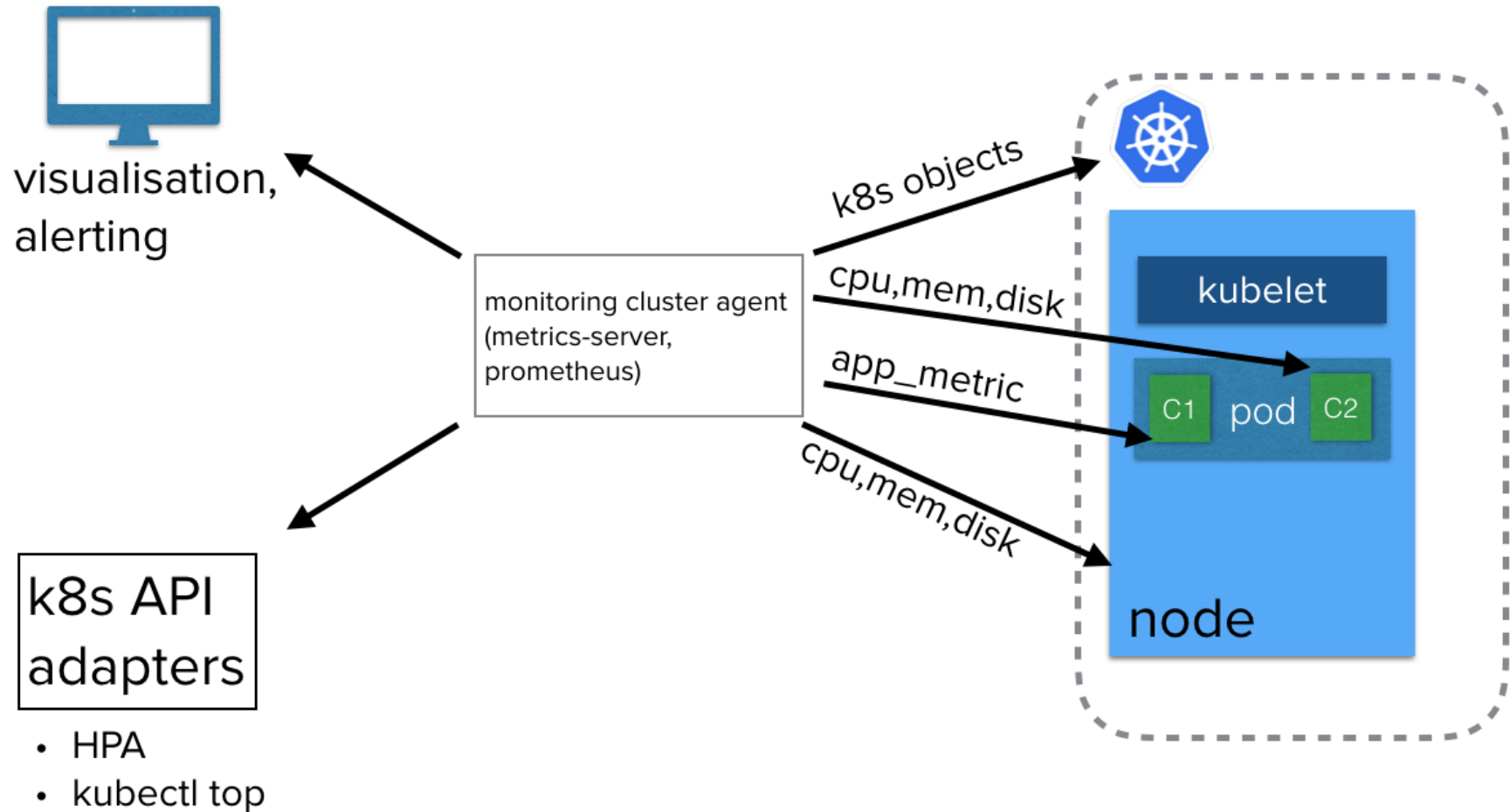
Readiness probes: Recover



Readiness probes: Good



Monitoring pipeline



Архитектура мониторинга в Kubernetes

Мониторинг в Kubernetes делится на 2 большие группы:

- core metrics pipeline - предназначен для использования самим Kubernetes
- monitoring pipeline - отдается на откуп инфраструктурной команде и предназначен для "полноценного" мониторинга

metrics-server

- Инструмент из коробки
- Собирает основные метрики кластера с kubelet'ов
- Используется внутренними компонентами
- Отдает метрики в Metrics API format

kubectl top

```
$ kubectl top nodes
```

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
gke-cluster-f9c66281-rgld	116m	12%	1030Mi	88%
gke-cluster-f9c66281-dbb2	72m	7%	845Mi	72%

```
$ kubectl top pods
```

NAME	CPU(cores)	MEMORY(bytes)
post-test-post-54b7cbcc69-pnb7f7	4m	42Mi
post-test-post-54b7cbcc69-9md82	4m	41Mi
post-test-post-54b7cbcc69-mhj5r	4m	42Mi

kube-state-metrics

- Аддон Kubernetes
- Собирает метрики ресурсов Kubernetes из API (deployment, services, etc.)
- Отдает метрики в формате Prometheus

[kube-state-metrics](#)

Node exporter

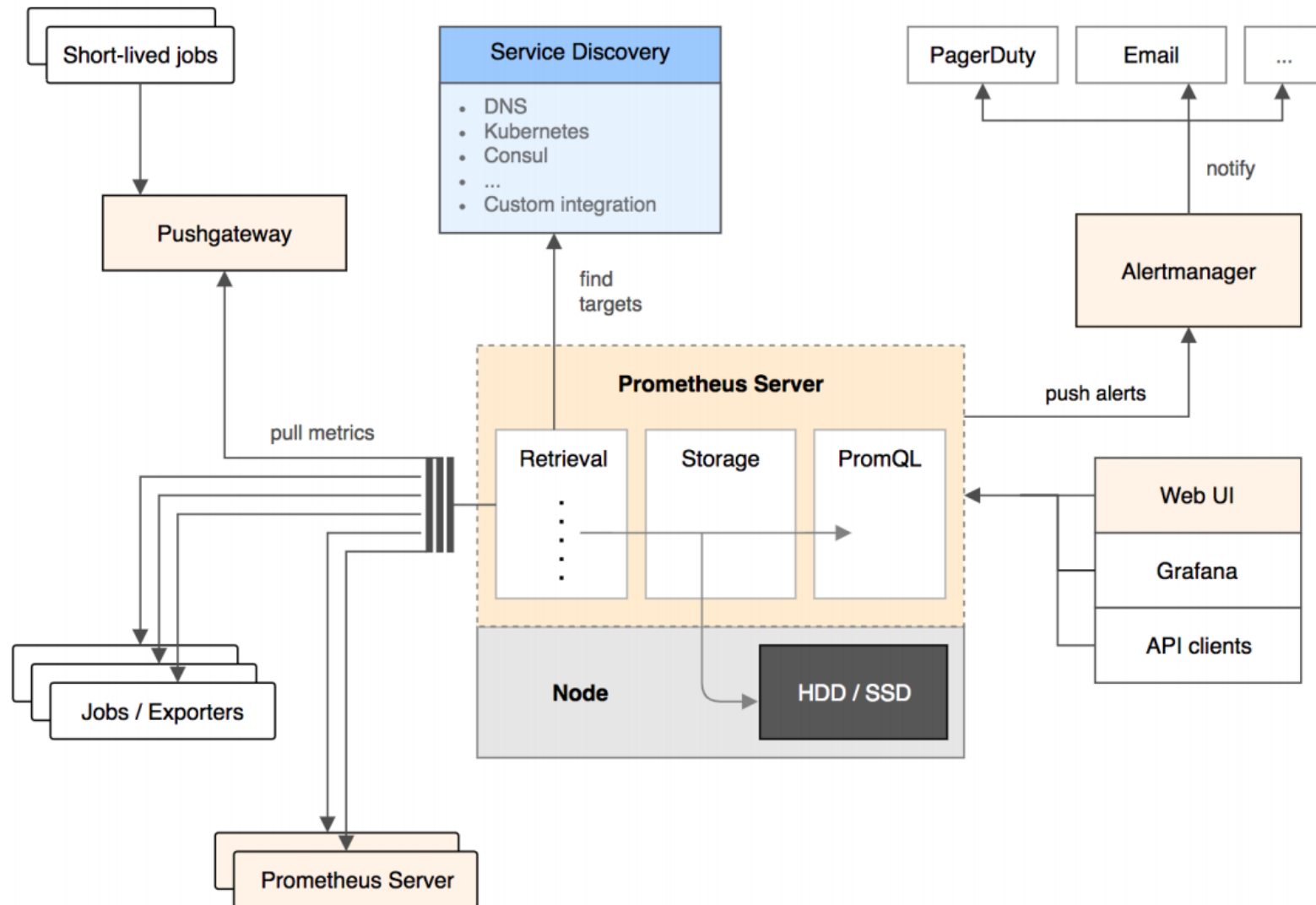
- Не аддон Kubernetes (никак с ним не связан)
- Собирает метрики нод (CPU, память, диск)
- Отдает метрики в формате Prometheus

Prometheus

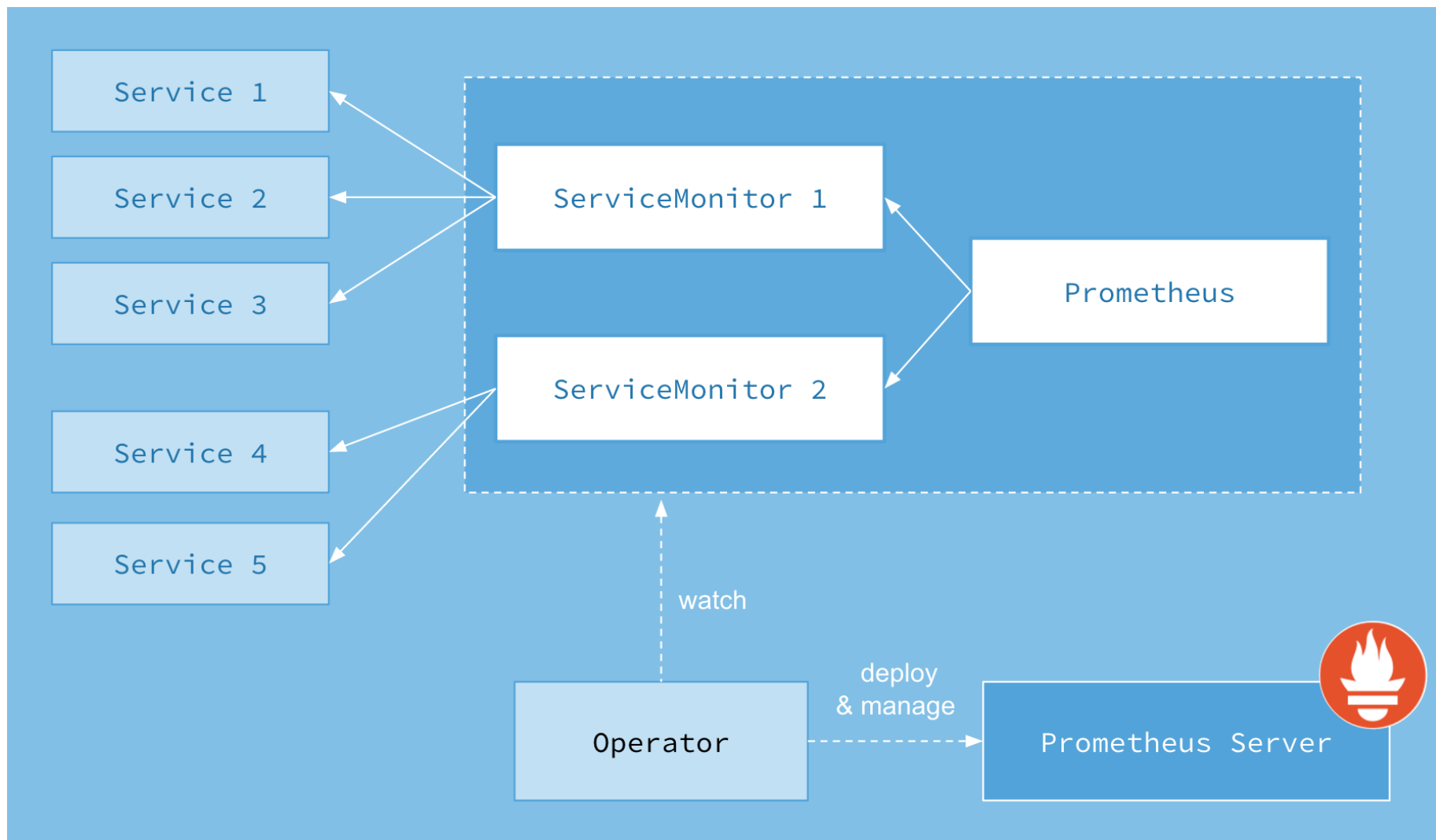
Prometheus

- TSDB для хранения метрик
- язык запросов PromQL для взаимодействия с БД
- Whitebox, Pull система, опрашивающая по HTTP
- Alertmanager для оповещений
- **Service Discovery в Kubernetes**

Архитектура Prometheus



Prometheus operator



CRD Prometheus Operator

Prometheus Оператор использует следующие CRD:

- **Prometheus** - разворачивает инстансы Prometheus
- **ServiceMonitor** - описывает сервисы для мониторинга, оператор генерирует конфигурацию и отдает Prometheus
- **PrometheusRule** - описывает правила Prometheus
- **Alertmanager** - разворачивает инстансы Alertmanager

Prometheus Operator

Преимущества:

- Описание желаемого состояния инстансов и их конфигурации с помощью манифестов
- Генерация конфигурации на основе Service Kubernetes

Конфигурация ServiceMonitor

С помощью CRD ServiceMonitor описываются сервисы, с которых необходимо собирать метрики

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: example-app
  labels:
    team: frontend
spec:
  selector:
    matchLabels:
      app: example-app # селектор сервисов для сбора метрик
  endpoints:
    - port: web
```

Определение Prometheus

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
spec:
  serviceAccountName: prometheus
  serviceMonitorSelector:
    matchLabels:
      team: frontend # селектор ресурсов ServiceMonitor для генерации конфигурации
resources:
  requests:
    memory: 400Mi
```

Интерфейс Prometheus

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy

default/ui/0 (2/2 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.36.1.4:9292/metrics	UP	endpoint="metrics" instance="10.36.1.4:9292" job="ui-service" namespace="default" pod="ui-deployment-59c5cb75cb-prjzw" service="ui-service"	768ms ago	17.71ms	
http://10.36.1.7:9292/metrics	UP	endpoint="metrics" instance="10.36.1.7:9292" job="ui-service" namespace="default" pod="ui-deployment-59c5cb75cb-jgfmn" service="ui-service"	5.161s ago	10.49ms	

Grafana

- Open source инструмент для построения дашбордов систем мониторинга
- Поддерживает получение данных из Graphite, Elasticsearch, OpenTSDB, Prometheus и InfluxDB и баз SQL
- Плагины для интеграции с другими системами мониторинга
- Hub с готовыми дошбордами

Дашборды

- Дашборды в Grafana хранятся в **.json**
- Есть возможность их импортировать/экспортировать
- Начиная с версии 4.0 поддерживается версионирование дашбордов при изменении с возможностью отката
- При желании, свои дашборды можно опубликовать в маркетплейсе <https://grafana.com/dashboards>

Hub для дашбордов

На сайте Grafana доступны сотни готовых дашбордов. Доступен поиск и фильтрация по различным критериям.

Dashboards

Official & community built dashboards

Filter by:

Data Source

Prometheus

Panel Type

All

Category


Docker


Collector


All


Search within this list

🔍

**Docker and system monitoring** by Thibaut Mottet
A simple overview of the most important Docker host and container metrics. (cAdvisor/Prometheus)
PROMETHEUS NODEEXPORTER
Downloads: 3664

**Docker Dashboard** by Brian Christner
Docker Monitoring Template
PROMETHEUS NODEEXPORTER
Downloads: 5933

**Docker Engine Metrics** by Basilio Vera
Draw some docker metrics
PROMETHEUS OTHER
Downloads: 594

**Docker Host & Container Overview** by uschtwill
A simple overview of the most important Docker host and container metrics. (cAdvisor/Prometheus)
PROMETHEUS NODEEXPORTER
Downloads: 6632

PromQL

Язык запросов PromQL

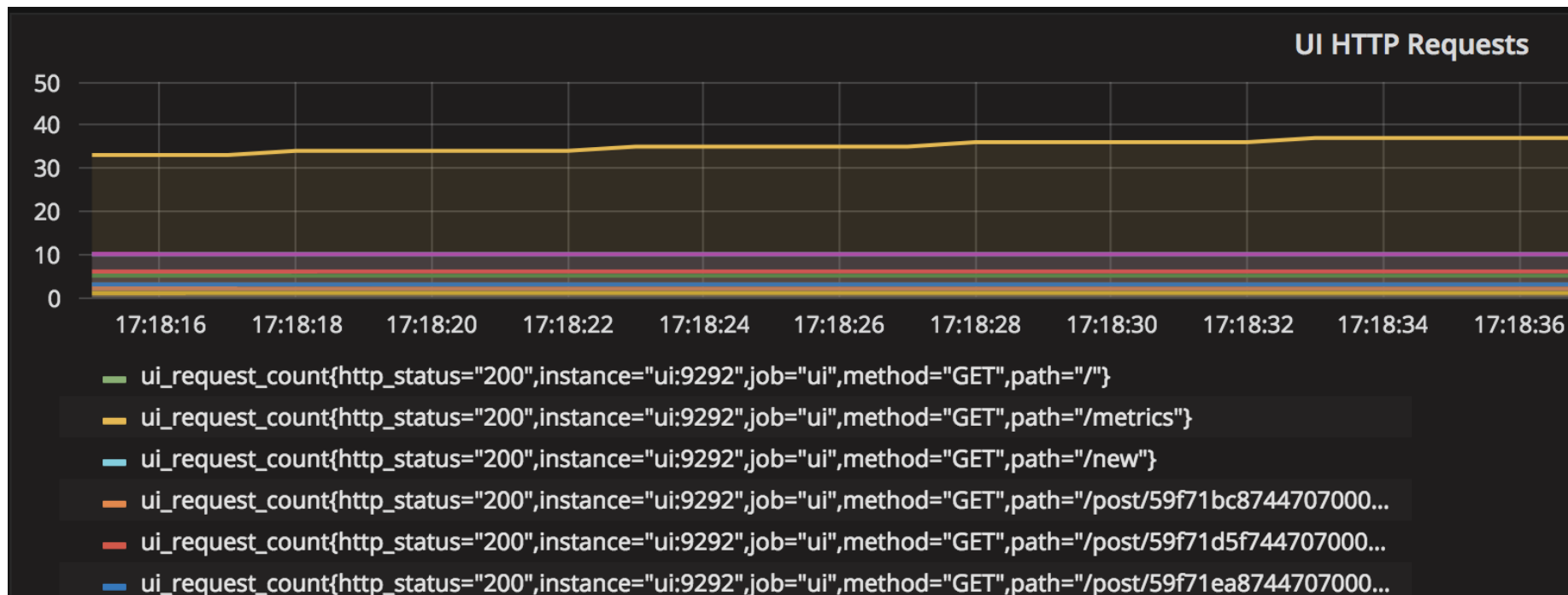
PromQL - язык запроса данных, реализованный в Prometheus. Запросы данных в PromQL состоят из:

- **Literals** (литералы) - числа, строки
- **Time series Selectors** - выборка вектора значений метрики из временного ряда за определенный момент или за промежуток времени. Например: `node_cpu`, `node_cpu{mode='idle'}`, `node_cpu{mode='idle'} offset 5m`, `node_cpu{mode='idle'} [1m]`
- **Operators** (операторы) - различные операторы: арифметические, сравнения, работы над векторами и агрегации
- **Functions** (функции) - большой список функций, которые можно применять к вектору временного ряда

Time series selector

Отообразим метрику:

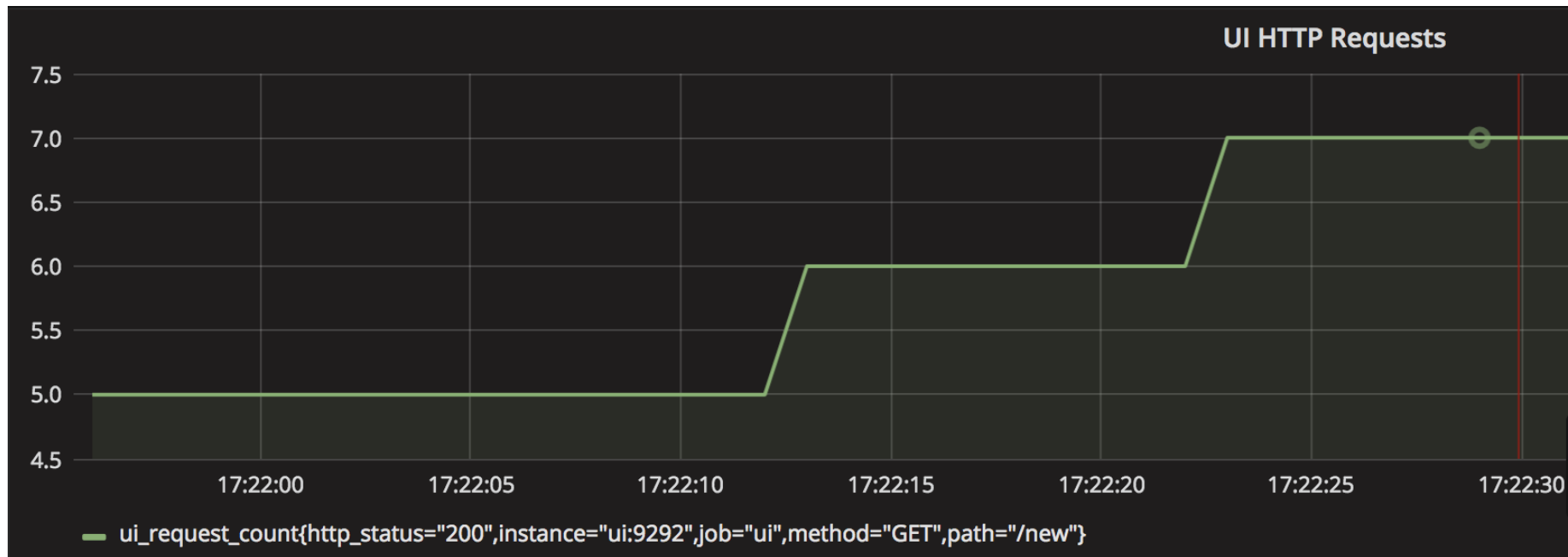
`ui_request_count`



Time series selector

Уточняем метрику, используя лейблы:

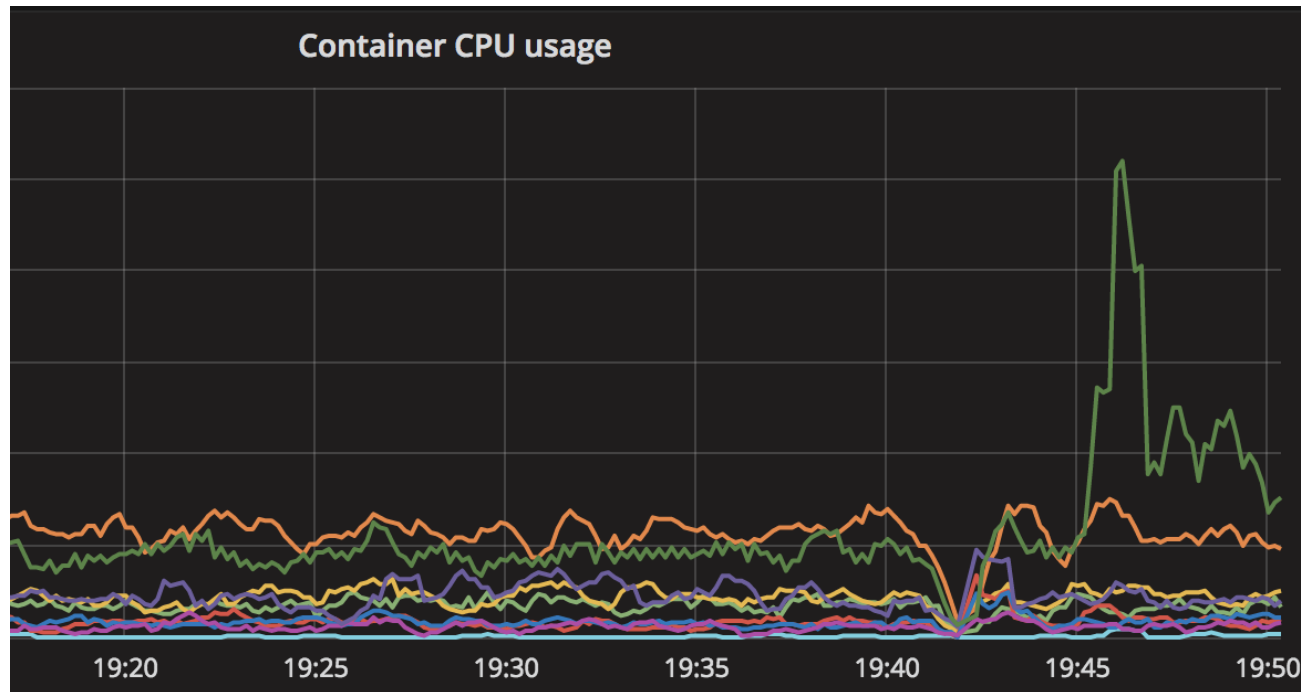
```
ui_request_count{method="GET",path="/new"}
```



Функции PromQL

Результат запроса:

```
rate(container_cpu_user_seconds_total{image!=""}[1m])
```



Полезные ссылки

1. книга Brendan Gregg. [Systems Performance: Enterprise and the Cloud](#)
2. заметка про [USE и RED](#)
3. статья про [SRE Golden Signals](#)
4. статья от DataDog [Monitoring 101: Собираем правильные данные](#)
5. Алексей Иванов, Dropbox. [Практический опыт мониторинга распределённых систем. Слайды.](#)
6. Владимир Рычев, Google. [Как я научился не волноваться и полюбил пейджер](#) (про концепцию SRE, SLA/SLO/SLI и др.). [Слайды.](#)

Полезные ссылки

1. Владимир Иванов, Booking. [Graphite в booking.com](#)
2. Конференции и др.: [Monitorama](#), [FOSDEM](#), [Velocity](#), [Uptime community](#)
3. Примеры [Runbook от GitLab](#)