

# Зачем и что нужно мониторить



Сергей  
Андрюнин



**Сергей Андрюнин**

DevOps инженер

RTLabs

---

# План занятия

1. [Что такое мониторинг](#)
2. [Зачем нужен мониторинг](#)
3. [Типы мониторинга](#)
4. [Деление мониторинга по доменам ответственности](#)
5. [Подходы при настройке мониторинга](#)
6. [Подходы при настройке мониторинга: SRE](#)
7. [Подходы при настройке мониторинга: логи не нужны](#)
8. [Итоги](#)
9. [Домашнее задание](#)

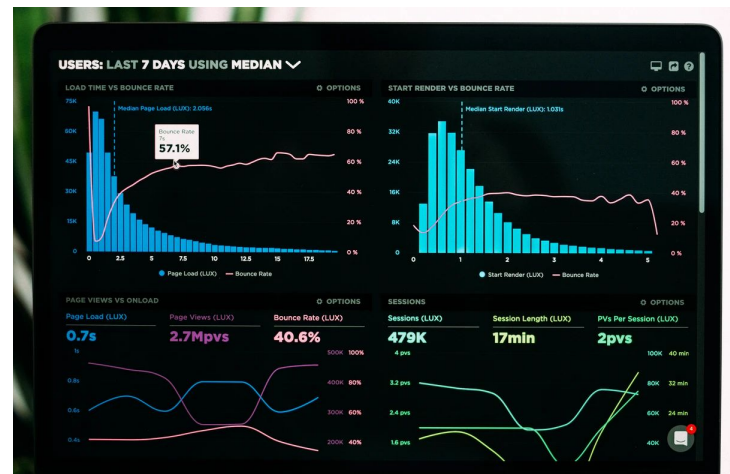


# Что такое мониторинг?

# Что такое мониторинг?

**Мониторинг** - это сбор, обработка, агрегирование и отображение в реальном времени количественных и качественных показателей системы.

Мониторинг позволяет улучшать, либо оставлять на приемлемом уровне качество обслуживания пользователей.



Взято с сайта: [unsplash.com](https://unsplash.com)

---

# Что такое мониторинг?

На практике мониторинг чаще всего включает в себя:

- **Оценку работоспособности ПО.** Например, количество успешных ответов http-сервера.
- **Оценку работоспособности оборудования.** Например, средняя загрузка ЦПУ за 5 минут.
- **Бизнес мониторинг.** Например, мониторинг движения финансовых средств.
- **Мониторинг безопасности ИС.** Например, мониторинг актуальности каких-либо сертификатов.



# Зачем нужен мониторинг

---

# Зачем нужен мониторинг

- **Анализ долгосрочных тенденций.** Получение качественных характеристик в обеспечение дальнейшей работоспособности системы. Например размер БД и его близость к критическим значениям.
- **Сравнение версий ПО.** Насколько изменения ПО повлияли на качество обслуживания.
- **Оповещение.** Превентивное выявление возможных отклонений в качестве обслуживания и увеличение скорости реакции на сбои.
- **Телеметрия текущей работоспособности приложения.** Получение текущей оценки характеристик информационной системы в режиме real-time.
- **Ретроспективный анализ,** Выявление узких мест информационной системы на основе полученных ранее данных мониторинга.





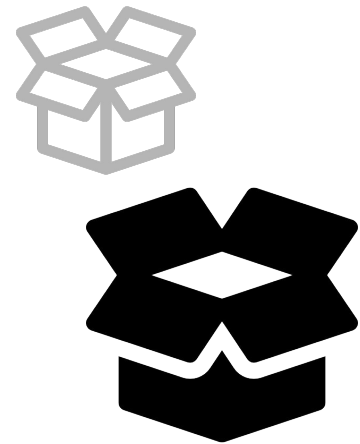
# Типы мониторинга

---

# Типы мониторинга

Чаще всего мониторинг разделяют на 2 типа:

- **White-box monitoring.** Наблюдение за системой “изнутри”. Сбор данных профилирования, логов, системные журналы.
- **Black-box monitoring.** Наблюдение, видимое извне. Например сбор возникших ошибок системы технической поддержкой или отделом тестирования.




---

## Типы мониторинга

Для DevOps-подхода **важнее white-box мониторинг**, так как только он позволяет спрогнозировать поведение системы и исправить проблемы до их возникновения.

Но про black-box мониторинг тоже не стоит забывать. Это “симптоматическая” диагностика, которая позволяет выявить редкие кейсы поломок системы.





# **Деление мониторинга по доменам ответственности**

---

## Деление мониторинга по доменам ответственности

Домены ответственности мониторинга можно разделить по системам сбора метрик:

- Система сбора временных рядов. Prometheus, Influxdb.
- Система сбора логов. ELK.
- Система перехватчик-ошибок. Sentry.

## Деление мониторинга по доменам ответственности

Системы сбора временных рядов позволяют получать телеметрию в режиме реального времени и хранить данные в таблицах в виде “метка-времени - значение”.

Такие системы нужны для непрерывного наблюдения за изменением в работоспособности системы.

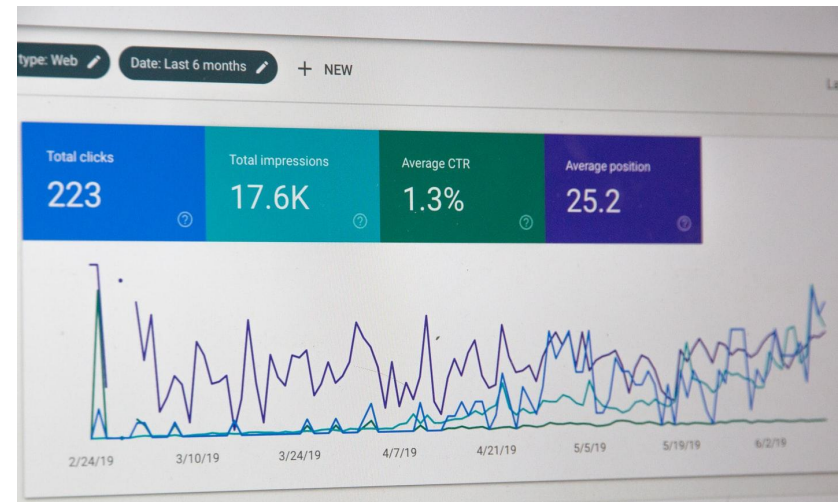
Пример метрики - доступное место на файловой системе

timestamp	CPU LA 15
2020-08-06 15:32:00	1,23
2020-08-06 15:32:30	1,30
2020-08-06 15:33:00	1,20
2020-08-06 15:32:30	2,00

# Деление мониторинга по доменам ответственности

Основные метрики для мониторинга в системах временных-рядов:

- CPU LA.
- RAM/swap
- IOPS
- inodes
- FS

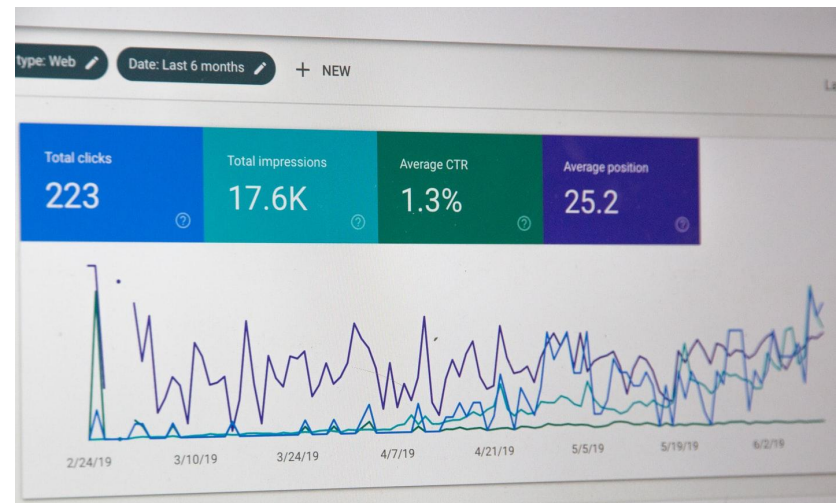


Взято с сайта: [unsplash.com](https://unsplash.com)

# Деление мониторинга по доменам ответственности


Дополнительные метрики для  
мониторинга в системах  
временных-рядов:

- **Process liveness**
- **IOwait**
- **Net traffic**
- **Custom?**



Взято с сайта: [unsplash.com](https://unsplash.com)





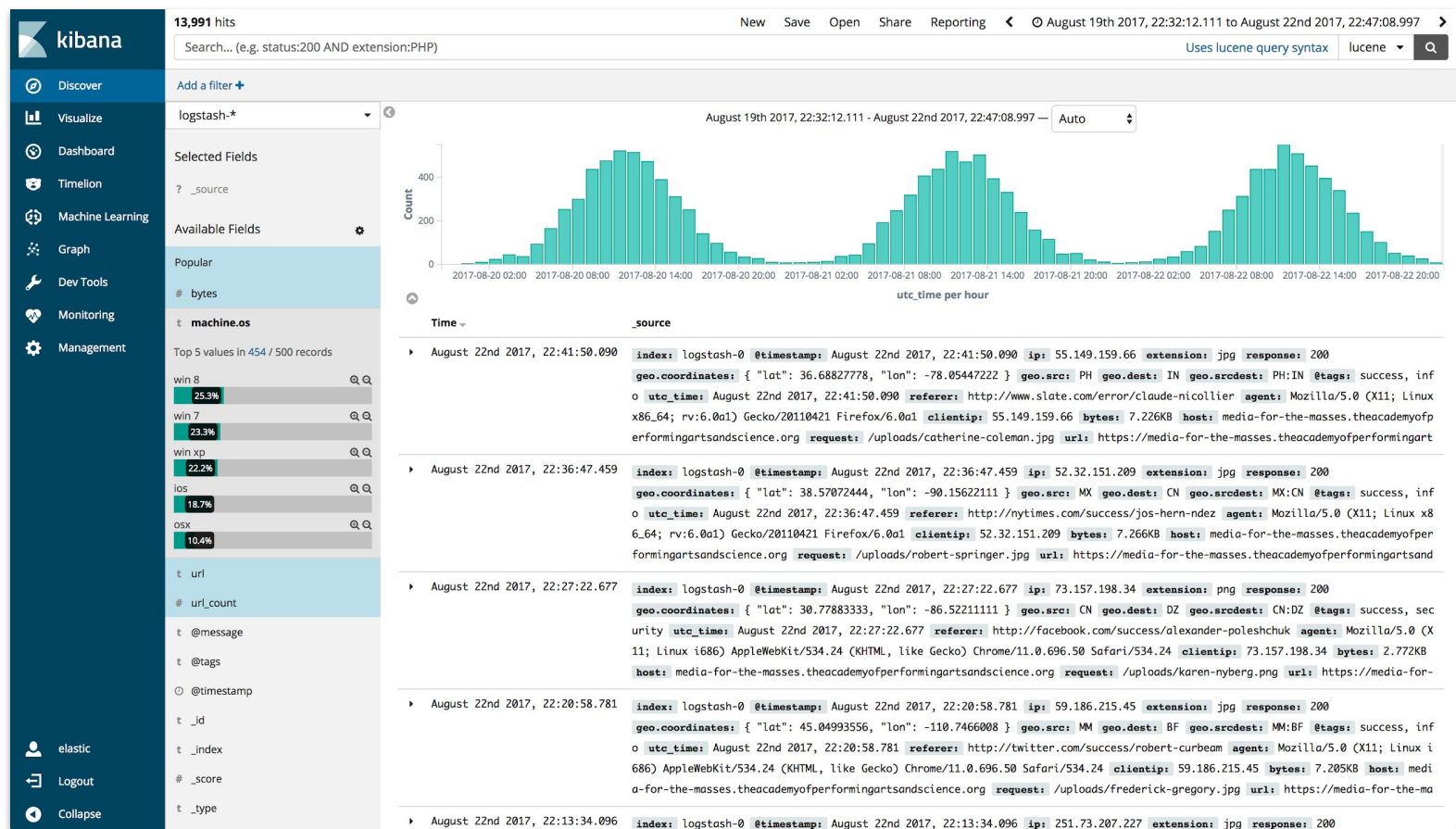
## Деление мониторинга по доменам ответственности

Системы сбора логов позволяют агрегировать логи приложений, системные журналы и т.п.

Такие системы чаще всего нужны **для детального разбора появившихся проблем**, нахождения их частоты появления, разбора поведения системы в каких-то условиях.

Пример - логи балансировщика запросов.

# Деление мониторинга по доменам ответственности



---

## Деление мониторинга по доменам ответственности

Системы “Перехватчик ошибок” **позволяют информировать о возникающих ошибках** ПО в режиме real-time.

При этом для этой системы можно указать дополнительные характеристики, которые будут в информировании.

Например:

- **Слой (stage/production).**
- **Hostname.**
- **Номер реплики приложения.**
- **Внутреннее состояние приложения.**

# Деление мониторинга по доменам ответственности

The screenshot shows the Sentry Frontend interface for a specific release. The top navigation bar includes links for Issues, Events, Overview, User Feedback, and Releases. The release is identified as e7919e9dfd00, released 2 days ago, with 11 commits by 4 authors. The main content area is divided into several sections: 'Issues Resolved in this Release' showing a 'TypeError' in 'app/components/versionHoverCard.jsx'; 'New Issues in this Release' showing another 'TypeError' in 'HTMLDocument.handleDocumentMo...'; '13 files changed in github:getsentry/sentry' listing files like 'package.json' and 'commitAuthorStats.jsx'; 'COMMITTS BY AUTHOR' showing a bar chart with 6 commits by one author and 1 by another; 'OTHER PROJECTS AFFECTED' showing 'CSP' with 3 new issues and 'Sentry Backend' with 2 new issues; and 'DEPLOYS' showing a 'prod' deployment 2 days ago. The interface is clean and modern, with a dark sidebar on the left.

**Sentry Frontend** (Unstar Project) (Project Settings)

Issues Events Overview User Feedback Releases

**Release e7919e9dfd00** 11 COMMITTS BY 4 AUTHORS NEW ISSUES 1 FIRST EVENT 2 days ago LAST EVENT a day ago

2 days ago

Overview New Issues All Issues Artifacts Commits

**Issues Resolved in this Release**

- TypeError** app/components/versionHoverCard in lastCommit  
Sentry Frontend Cannot read property 'lastCommit' of undefined

**New Issues in this Release**

- TypeError** eval at E\_c (:3:114), <anonymous> in HTMLDocument.handleDocumentMo...  
Sentry Frontend Cannot read property 'getAttribute' of undefined

**13 files changed in github:getsentry/sentry**

- package.json
- src/sentry/static/sentry/app/components/commitAuthorStats.jsx
- src/sentry/static/sentry/app/components/fileChange.jsx
- src/sentry/static/sentry/app/components/group/suggestedOwners.jsx
- src/sentry/static/sentry/app/components/releaseProjectStatSparkline.jsx

Show 8 collapsed files

**COMMITTS BY AUTHOR**

Author	Commits
[Avatar]	6
[Avatar]	2
SB	2
MB	1

**OTHER PROJECTS AFFECTED**

- CSP**  
3 new issues
- Sentry Backend**  
2 new issues

**DEPLOYS**


- prod 2 days ago

---

## Деление мониторинга по доменам ответственности

Основные метрики для мониторинга в системах сбора логов и ловли ошибок:

- **Good events/All events**
- **Traffic**



# Подходы при настройке мониторинга




## Подходы при настройке мониторинга

Приведенные подходы не являются формальными и не будут тем самым “золотым молотком”, который решит все проблемы.

Но желательно эти подходы знать и использовать в своей деятельности по необходимости.

Они позволяют экономить на ресурсах, выделенных для мониторинга, связать мониторинг с бизнес частью и улучшить качественную интерпретацию полученных метрик.



# Подходы при настройке мониторинга: SRE

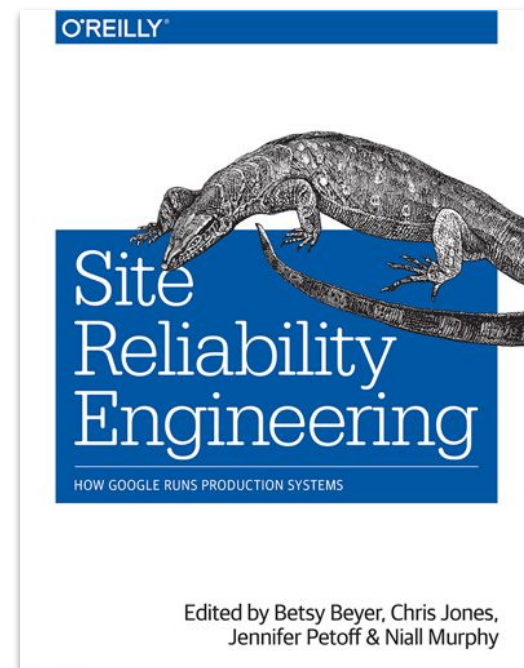


# Подходы при настройке мониторинга: SRE

Данный подход разработан компанией google и повсеместно внедряется в компаниях по всему миру.

Подход **SRE** покрывает большинство задач **DevOps**, в т.ч. задачи мониторинга.

Один из его главных плюсов - он **сделан практикующими инженерами для практикующих инженеров**, на основе многолетнего опыта.



Взято с сайта: [landing.google.com](https://landing.google.com/sre/)

---

# Подходы при настройке мониторинга: SRE

Основные тезисы подхода при настройке мониторинга:

- Ставим для мониторинга реальные задачи.
- Система мониторинга должна отвечать на два вопроса:  
что сломалось? почему сломалось?
- Используйте четыре золотых сигнала.
- Не исследуйте только средние значения.
- Выберите подходящий уровень детализации.

---

# Подходы при настройке мониторинга: SRE

Четырьмя золотыми сигналами в подходе SRE называются:

- **Время отклика.** Время, которое требуется для выполнения запроса.
- **Величина трафика.** Величина нагрузки, которая приходится на вашу систему. Например для веб - это количество http запросов, а для потокового аудио - это скорость передачи данных.
- **Уровень ошибок.** Количество или частота неуспешно выполненных запросов. Например ответ 500 от http сервера.
- **Степень загруженности.** Показатель того, насколько полно загружен ваш сервис. Это мониторинг компонентов, которые покажут загруженность вашего сервиса. Например для вычислений - это ЦПУ. Для In-memory БД - это RAM.

---

# Подходы при настройке мониторинга: SRE

Кроме golden-signals также существуют такие важные метрики, как:

- **SLO.** Целевой уровень качества обслуживания. Целевое значение или диапазон значений.
- **SLA.** Соглашение об уровне обслуживания. Явный или неявный контракт с внешними пользователями, включающий в себя последствия невыполнения SLO.
- **SLI.** Индикатор качества обслуживания. Конкретная величина предоставляемого обслуживания.

# Подходы при настройке мониторинга: SRE

Метрики SLO/SLA/SLI позволяют связать измеряемые “технические” значения с бизнес составляющей.

## Пример простого SLA:


Если наш сайт отдает http коды 4xx/5xx, то мы должны произвести денежную компенсацию пользователю, если это не запланированное техническое обслуживание.

## Пример SLO:


В 99% случаев мы должны отдавать http коды отличный от 4xx/5xx. 1% выделяется на техническое обслуживание.

## Расчет SLI может выглядеть таким образом:

$$SLI = (\text{summ\_2xx\_requests} + \text{summ\_3xx\_requests}) / (\text{summ\_all\_requests})$$



**Подходы при настройке  
мониторинга:  
логи не нужны**



## Подходы при настройке мониторинга: Логи не нужны

Данный подход предложен компанией “Яндекс” на DevOpsConf 2019.

Формулируется данный подход следующим образом:

*“Достаточно создать инфраструктуру доставки полезных метрик и данных вокруг сервисов. В результате появятся полезные метрики, которые говорят о сервисах, и прозрачно показывают все, что с ними происходит.”*

---

# Подходы при настройке мониторинга: Логи не нужны

## Основные тезисы данного подхода:

- Логи в сыром виде бесполезны, но их можно превратить в полезные метрики.
- Ошибки должны содержать контекст самой ошибки.





# Итоги

---

## Итоги

Мониторинг - неотъемлемая часть обеспечения качества информационных систем.

Мониторинг должен:

- Отвечать на вопросы “Что случилось?” и “Почему?”.
- Быть достаточно простым.
- Иметь подходящий уровень детализации
- Содержать в себе метрики, связанные с “бизнес” частью

---

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Сергей Андрюнин**