



Ф Л А Н Т



DevOps

TSDB – ВЗГЛЯД ИЗНУТРИ

Миграция на Prometheus и Cortex. Опыт Okmeter.

Владимир Гурьянов, Андрей Колаштов, Дмитрий Столяров



Ф Л А Н Т



DevOps

TSDb – Взгляд изнутри

Миграция на Prometheus и Cortex. Опыт Okmeter.

SIE1: Beginning

Владимир Гурьянов, Андрей Колаштов, Дмитрий Столяров



Владимир Гурьянов

Ведущий инженер

vladimir.guryanov@flant.com

t.me/Magvai69



Андрей Колаштов

Руководитель продукта okmeter

andrey.kolashtov@flant.com

t.me/kolashtov



Дмитрий Столяров

CTO & Co-founder

dmitry.stolyarov@flant.com

t.me/distol

Флант и okmeter

Как мы здесь оказались?

Как мы здесь оказались?



Ф Л А Н Т



200+

кластеров Kubernetes

Как мы здесь оказались?



ФЛАНТ



200+

кластеров Kubernetes

120 млн

метрик Prometheus

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера

Графики по утилизированным
и свободным серверным мощностям

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах

По **нодам** кластера

Drill-down по **примитивам** Kubernetes

overview

↳ *namespace*

↳ *controller*

↳ *pod*

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Как мы здесь оказались?



Ф Л А Н Т

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера
 - Control-plane
 - DNS
 - Сеть между нодами

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах

Drill-down по всем **vhost**

↳ *location*

Drill-down по всем **namespace**

↳ *ingress resource*

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

По времени
ответа сервера

По кодам
ответа сервера

И по другим
параметрам

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

В конце презентации расскажем,
где всё это можно **потрогать руками**.
Обещаем — там **кра-си-во!**

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали

Как мы здесь оказались?



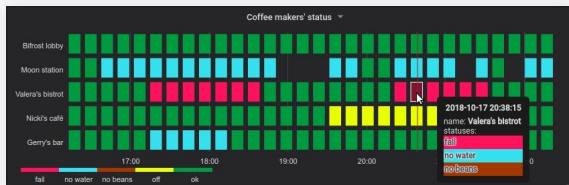
ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus



У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали



grafana-statusmap

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали



grafana-statusmap

9 000 000 установок

Нам пишут инженеры из Tesla

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали



grafana-statusmap



upmeter

Как мы здесь оказались?



Ф Л А Н Т

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали



grafana-statusmap



upmeter

Гоняет **смоук-тесты**
в каждом кластере

SLA каждого компонента
в каждом нашем кластере

Как мы здесь оказались?



ФЛАНТ

200+

кластеров Kubernetes

120 млн

метрик Prometheus

У нас крутой «базовый мониторинг»

- Общая утилизация мощностей кластера
- Детальная утилизация мощностей кластера в двух разрезах
- По состоянию компонентов кластера

Не менее крутой «мониторинг Ingress»

- Нагрузка балансировщика в двух разрезах
- Графики в процентном распределении

Мы разработали



grafana-statusmap



upmeter

Как мы здесь оказались?



Ф Л А Н Т



200+

кластеров Kubernetes

а ещё есть stateful

120 млн

метрик Prometheus

Как мы здесь оказались?



200+

кластеров Kubernetes

120 млн

метрик Prometheus

а ещё есть stateful

180+

MySQL

440+

PostgreSQL

45+

Cassandra

120+

MongoDB

140+

Elasticsearch

70+

Kafka

280+

RabbitMQ

1800+

Redis

Как мы здесь оказались?

а ещё есть stateful

180+

MySQL

440+

PostgreSQL

45+

Cassandra

120+

MongoDB

140+

Elasticsearch

70+

Kafka

280+

RabbitMQ

1800+

Redis

Как мы здесь оказались?

Исторически мы с 2017 года
мониторили всё это с помощью



а ещё есть stateful

180+

MySQL

440+

PostgreSQL

45+

Cassandra

120+

MongoDB

140+

Elasticsearch

70+

Kafka

280+

RabbitMQ

1800+

Redis

Как мы здесь оказались?



Как мы здесь оказались?



- Простая установка и ноль конфигурации

Как мы здесь оказались?



- Простая установка и ноль конфигурации
- Дашборды, графики и алерты из коробки

Как мы здесь оказались?



- Простая установка и ноль конфигурации
- Автодискавери множества процессов и их мониторинг из коробки
- Дашборды, графики и алерты из коробки



Как мы здесь оказались?



- Простая установка и ноль конфигурации
- Автодискавери множества процессов и их мониторинг из коробки



- Дашборды, графики и алерты из коробки
- Статистика баз данных



Как мы здесь оказались?



- Простая установка и ноль конфигурации
- Автообнаружение множества процессов и их мониторинг из коробки



- Дашборды, графики и алерты из коробки
- Статистика баз данных



Быстрый старт – всё уже продумали за вас

Как мы здесь оказались?



А ещё такие полезные и приятные фишки:

Как мы здесь оказались?



А ещё такие полезные и приятные фишки:

- Okmeter не теряет данные, если порвалась связь от агента до облака

Как мы здесь оказались?



А ещё такие полезные и приятные фишки:

- Okmeter не теряет данные, если порвалась связь от агента до облака
- Околонулевая нагрузка от агента, даже парсинге логов

Как мы здесь оказались?



А ещё такие полезные и приятные фишки:

- Okmeter не теряет данные, если порвалась связь от агента до облака
- Экономия трафика: высокоэффективный протокол для передачи метрик
- Околонулевая нагрузка от агента, даже парсинге логов

Как мы здесь оказались?



А ещё такие полезные и приятные фишки:

- Okmeter не теряет данные, если порвалась связь от агента до облака
- Экономия трафика: высокоэффективный протокол для передачи метрик
- Околонулевая нагрузка от агента, даже парсинге логов
- Кастомные метрики можно собирать через *statsd, script, logparser, http, sql-query, ...*

Как мы здесь оказались?



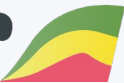
Ф Л А Н Т

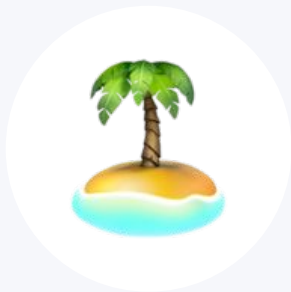
И в итоге → стратегическое партнёрство



Ф Л А Н Т

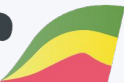
И в итоге → стратегическое партнёрство

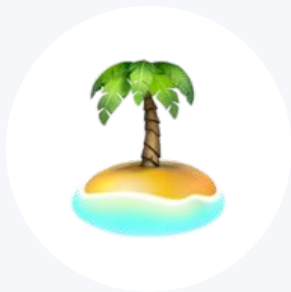
okmeter 



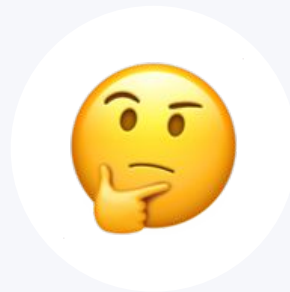
Ф Л А Н Т

И в итоге → стратегическое партнёрство

okmeter 



Ф Л А Н Т



Okmeter становится частью экосистемы Фланта



Ф Л А Н Т

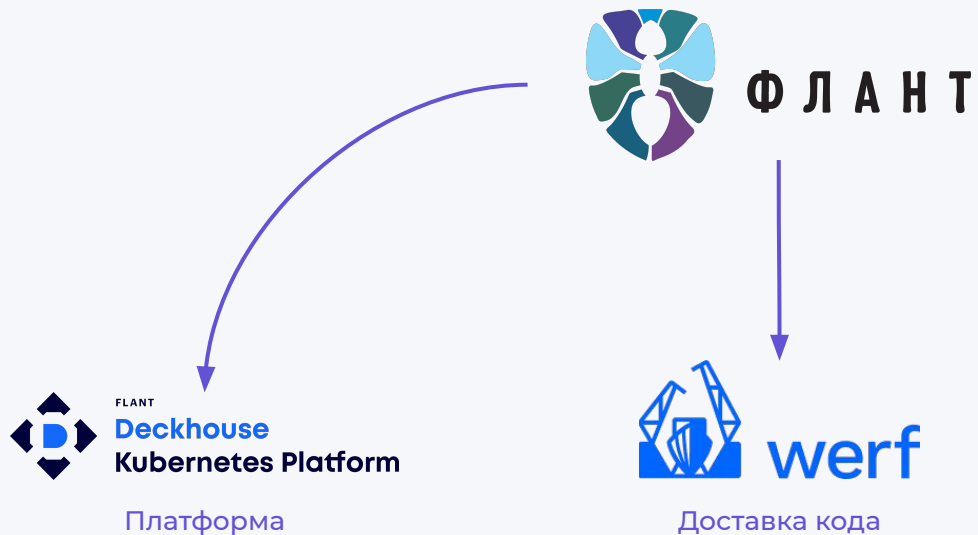
Okmeter становится частью экосистемы Фланта



Okmeter становится частью экосистемы Фланта



Okmeter становится частью экосистемы Фланта



Okmeter становится частью экосистемы Фланта



Okmeter становится частью экосистемы Фланта



Уже было

200+

кластеров Kubernetes

120 млн

метрик Prometheus

Okmeter становится частью экосистемы Фланта



Уже было

200+

кластеров Kubernetes

120 млн

метрик Prometheus



В результате к нам “приехало”

Okmeter становится частью экосистемы Фланта



Уже было

200+

кластеров Kubernetes

120 МЛН

метрик Prometheus



В результате к нам “приехало”

23 МЛН

метрик

от

4400+

серверов

от

290+

проектов

Что делать дальше?

Что делать дальше?

У нас есть план!

У нас есть план!

Для начала

- Объединить наработки

У нас есть план!

Для начала

- Объединить наработки



- Мощный дискавер
- Надёжный мониторинг stateful



Ф Л А Н Т

- Крутой мониторинг Kubernetes

У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in

У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in



**«Домашние
тапочки»**

У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in



«Домашние
тапочки»



Перспективный
стек

У нас есть план!

Для начала

- Объединить наработки

Мощный дискавер

Надёжный мониторинг stateful

Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in



«Домашние
тапочки»



Перспективный
стек



Флант против
vendor lock-in

У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in



**«Домашние
тапочки»**



**Перспективный
стек**



**Флант против
vendor lock-in**

У нас есть план!

Для начала

- Объединить наработки

 - Мощный дискавер

 - Надёжный мониторинг stateful

 - Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

 - okQL → PromQL

 - Grafana-совместимость

Конкретно:

- okQL → PromQL

- Grafana-совместимость

У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

А затем

У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

А затем

- Инцидент менеджмент

У нас есть план!

Для начала

- Объединить наработки

Мощный дискавер

Надёжный мониторинг stateful

Крутой мониторинг Kubernetes

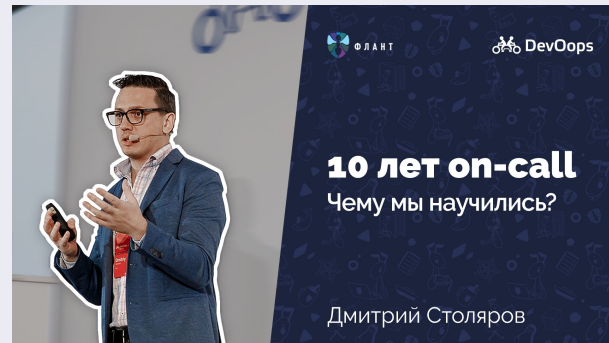
- Расширить аудиторию
и убрать vendor lock-in

okQL → PromQL

Grafana-совместимость

А затем

- Инцидент менеджмент



youtube.com/watch?v=7ZWGFngqSLU

У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

А затем

- Инцидент менеджмент

У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

А затем

- Инцидент менеджмент

- Логи



У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in
 - okQL → PromQL
 - Grafana-совместимость

А затем

- Инцидент менеджмент
- Логи
- Трейсы



У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

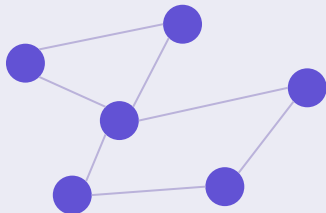
А затем

- Инцидент менеджмент

- Логи

- Трейсы

- Графы связей



У нас есть план!

Для начала

- Объединить наработки

Мощный дискавер

Надёжный мониторинг stateful

Крутой мониторинг Kubernetes

- Расширить аудиторию
и убрать vendor lock-in

okQL → PromQL

Grafana-совместимость

А затем

- Инцидент менеджмент

- Логи

- Трейсы

- Графы связей

- Мониторинг изменений



У нас есть план!

Для начала

- Объединить наработки

- Мощный дискавер

- Надёжный мониторинг stateful

- Крутой мониторинг Kubernetes

- Расширить аудиторию и убрать vendor lock-in

- okQL → PromQL

- Grafana-совместимость

А затем

- Инцидент менеджмент

- Логи

- Трейсы

- Графы связей

- Мониторинг изменений

- Распределенный blackbox-мониторинг и RUM

У нас есть план!

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in
 - okQL → PromQL
 - Grafana-совместимость

А затем

- Инцидент менеджмент
- Логи
- Трейсы
- Графы связей
- Мониторинг изменений
- Распределенный blackbox-мониторинг и RUM

И потом

- APM
- Anomaly detection и его друзья
- Smart root-cause analysis
- Co-pilot

У нас есть план!

Для начала

- Объединить наработки

Мощный дискавер

Надёжный мониторинг stateful

Крутой мониторинг Kubernetes

- Расширить аудиторию и убрать vendor lock-in

okQL → PromQL

Grafana-совместимость

А затем

- Инцидент менеджмент

- Логи

- Трейсы

- Графы связей

- Мониторинг изменений

- Распределенный blackbox-мониторинг и RUM

И потом

- APM

- Anomaly detection и его друзья

- Smart root-cause analysis

- Co-pilot

В итоге у нас есть...

В итоге у нас есть...

План

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in
 - okQL → PromQL
 - Grafana-совместимость

А затем

- Инцидент менеджмент
- Логи
- Трейсы
- Графы связей
- Мониторинг изменений
- Распределенный blackbox-мониторинг и RUM

И потом

- APM
- Anomaly detection и его друзья
- Smart root-cause analysis
- Co-pilot

В итоге у нас есть...

План

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in
 - okQL → PromQL
 - Grafana-совместимость

А затем

- Инцидент менеджмент
- Логи
- Трейсы
- Графы связей
- Мониторинг изменений
- Распределенный blackbox-мониторинг и RUM

И потом

- APM
- Anomaly detection и его друзья
- Smart root-cause analysis
- Co-pilot

Команда



В итоге у нас есть...

План

Для начала

- Объединить наработки
 - Мощный дискавер
 - Надёжный мониторинг stateful
 - Крутой мониторинг Kubernetes
- Расширить аудиторию и убрать vendor lock-in
 - okQL → PromQL
 - Grafana-совместимость

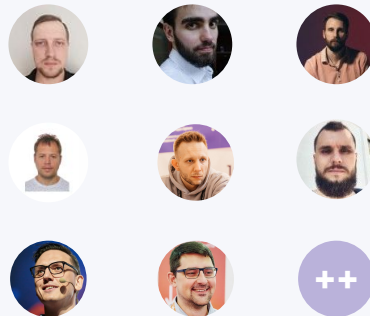
А затем

- Инцидент менеджмент
- Логи
- Трейсы
- Графы связей
- Мониторинг изменений
- Распределенный blackbox-мониторинг и RUM

И потом

- APM
- Anomaly detection и его друзья
- Smart root-cause analysis
- Co-pilot

Команда



База



200+

кластеров Kubernetes

120 млн

метрик Prometheus

okmeter

23 млн

метрик

от
4400+

серверов

от
290+

проектов

**А все ли у нас хорошо
с фундаментом?**

**А все ли у нас хорошо
с фундаментом?**

Storage

Как собираются данные с одного источника

Как собираются данные с одного источника

источник



Как собираются данные с одного источника

ИСТОЧНИК

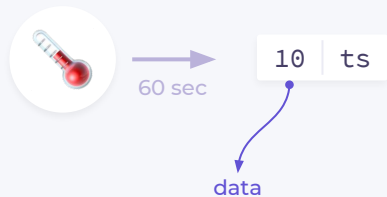


→
60 sec

10	ts
----	----

Как собираются данные с одного источника

ИСТОЧНИК

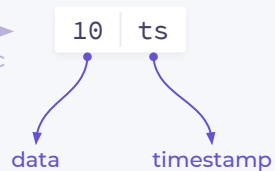


Как собираются данные с одного источника

ИСТОЧНИК



60 sec

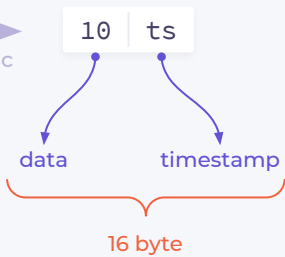


Как собираются данные с одного источника

ИСТОЧНИК



60 sec



Как собираются данные с одного источника

источник



→
60 sec

"файл"

10	ts
----	----

Как собираются данные с одного источника

источник



→
60 sec

"файл"

10 | ts

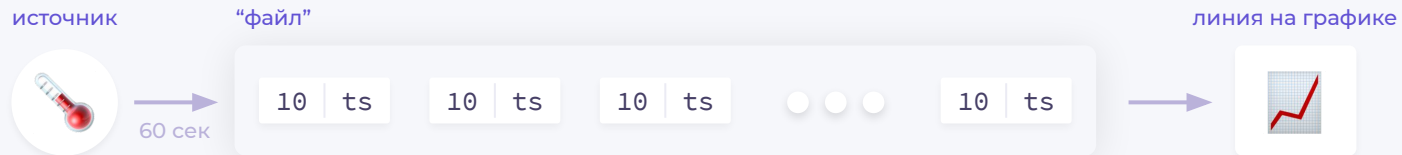
10 | ts

10 | ts



10 | ts

Как собираются данные с одного источника



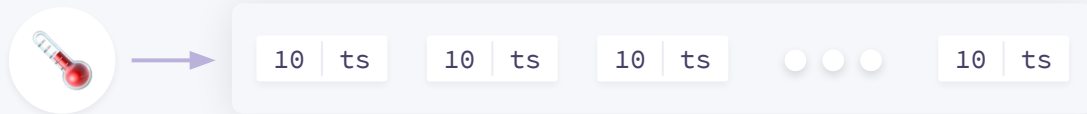
**А если у нас
1 000 000 источников?**

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”

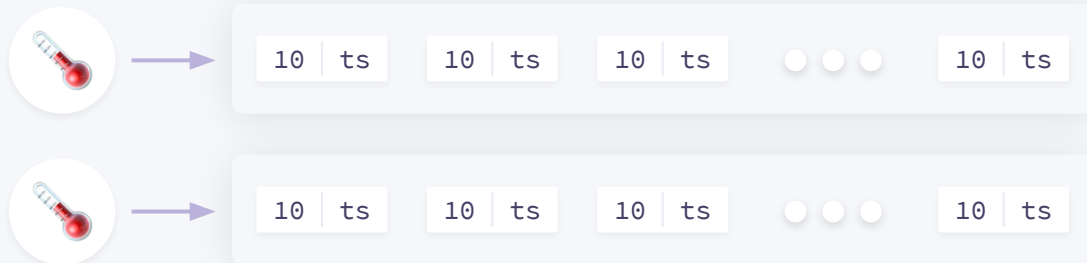
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



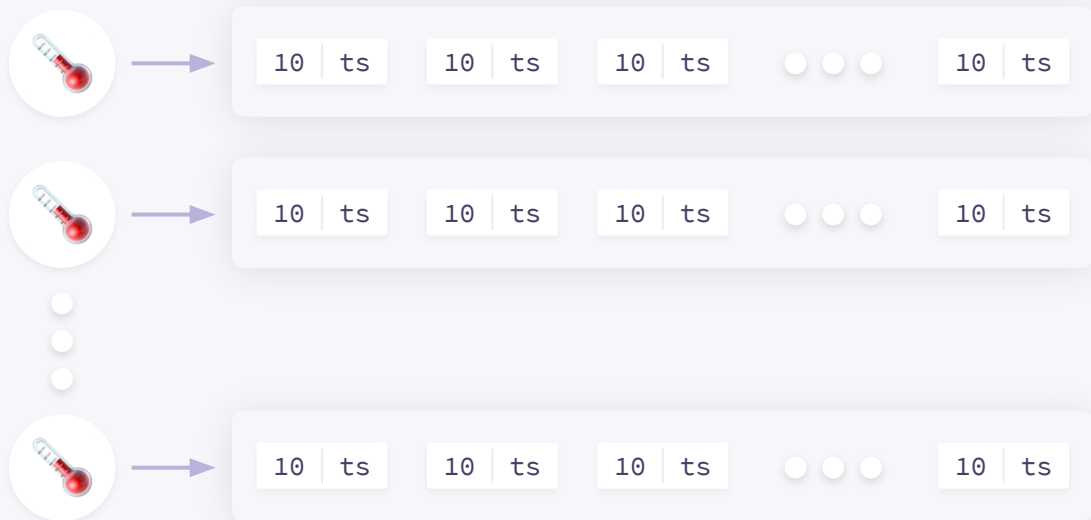
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



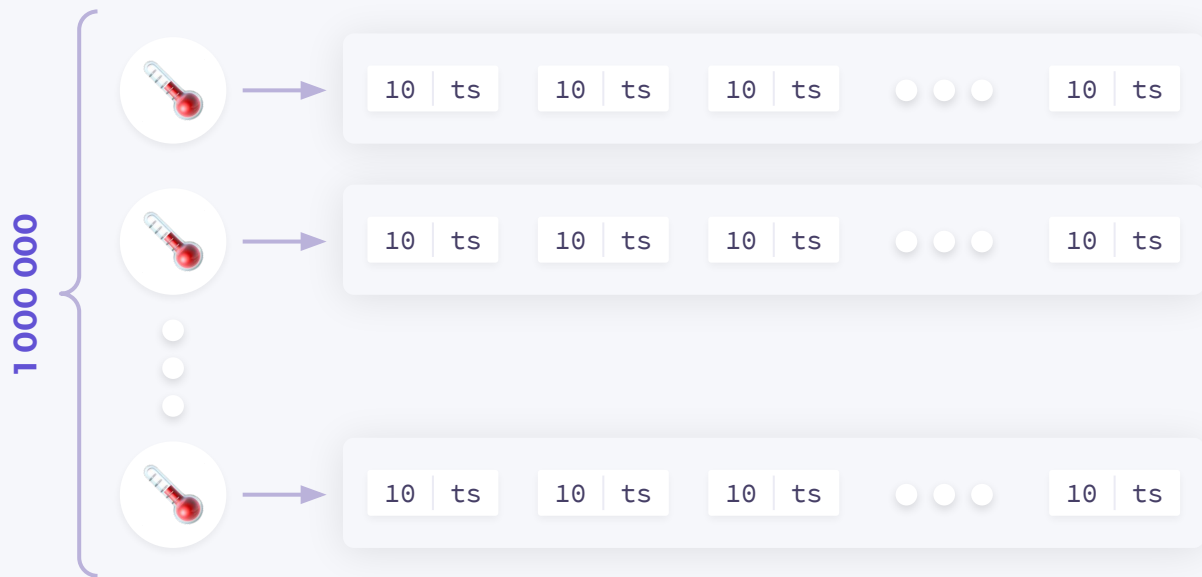
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



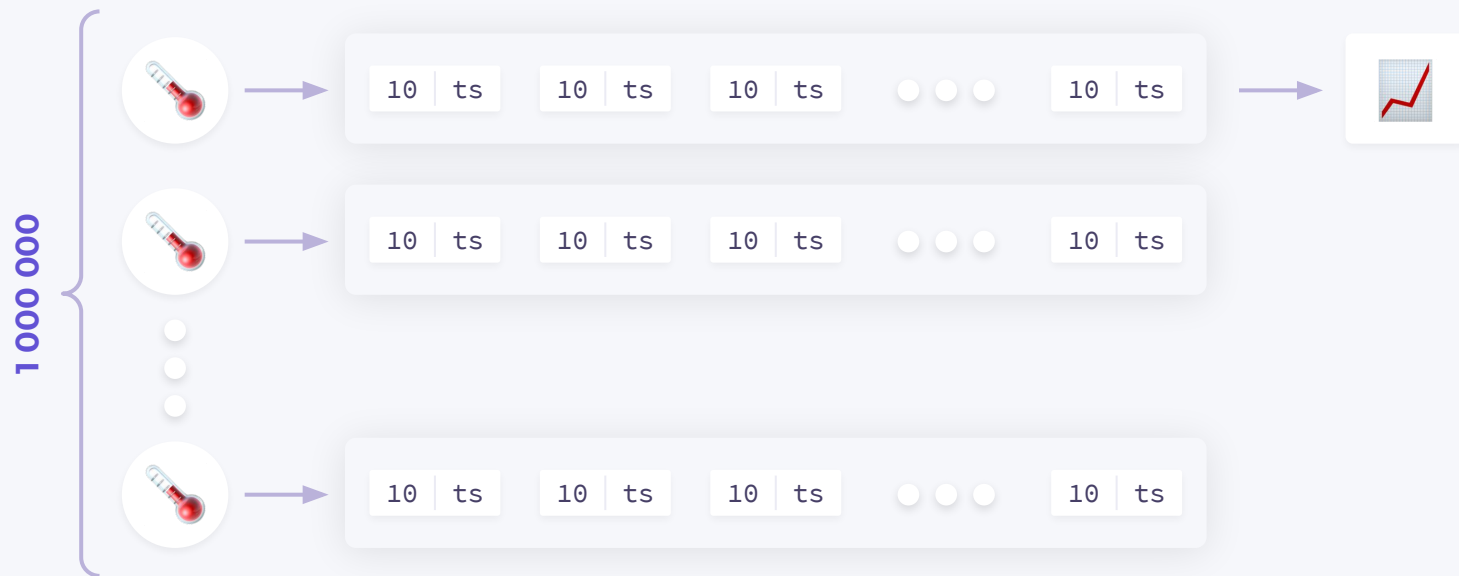
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



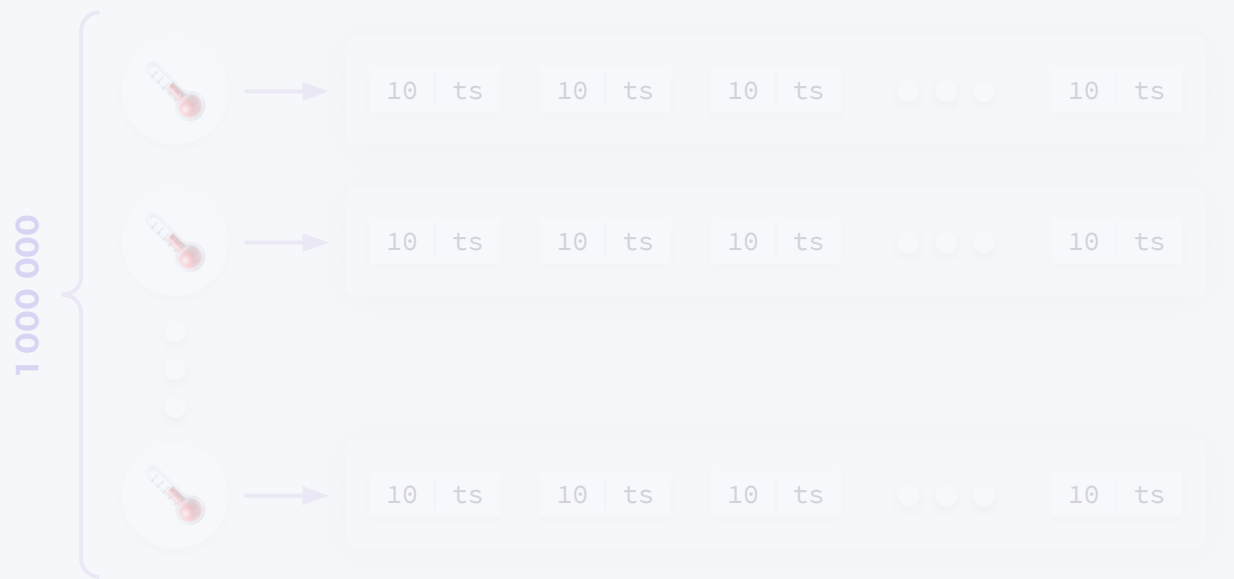
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



Как собираются данные со многих источников

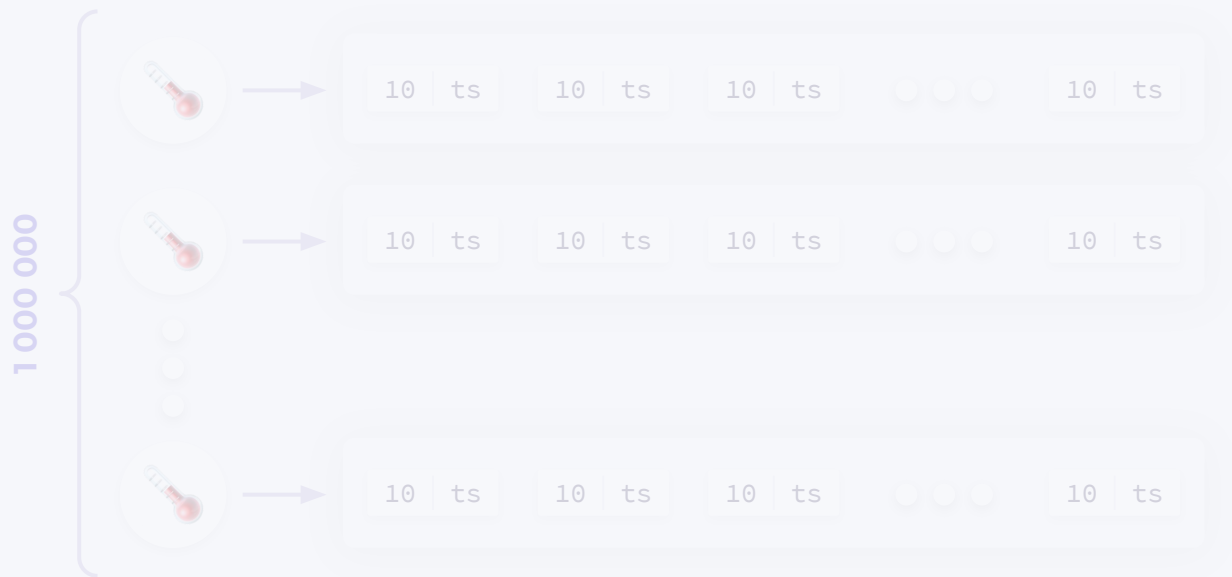
Вариант #1: с каждого источника в свой “файл”



1 1 000 000 IO / 60 sec

Как собираются данные со многих источников

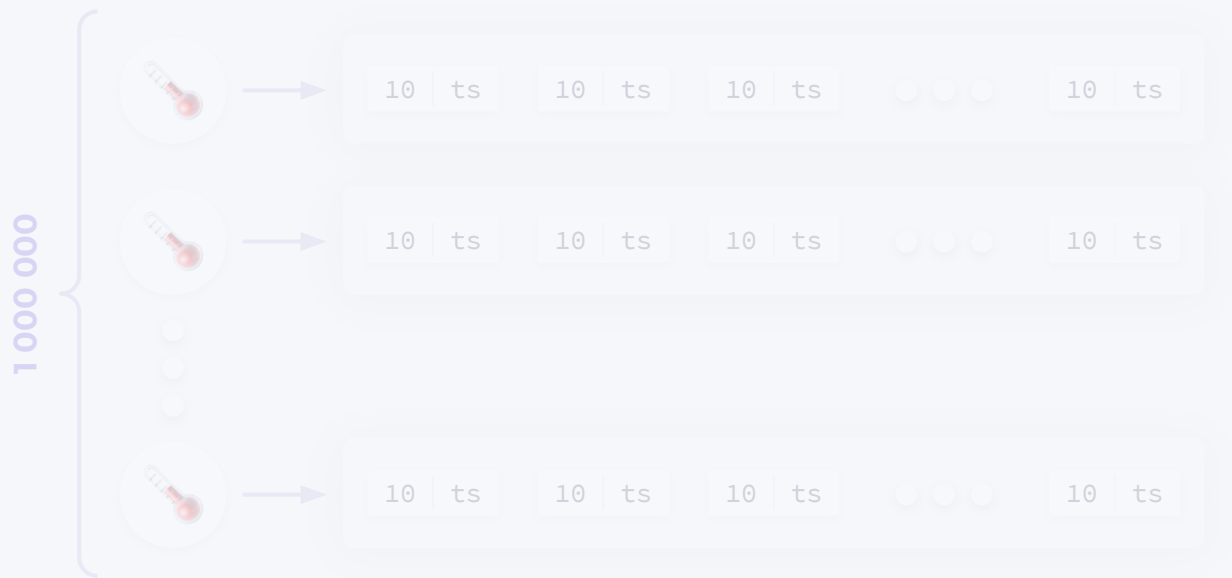
Вариант #1: с каждого источника в свой “файл”



1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”

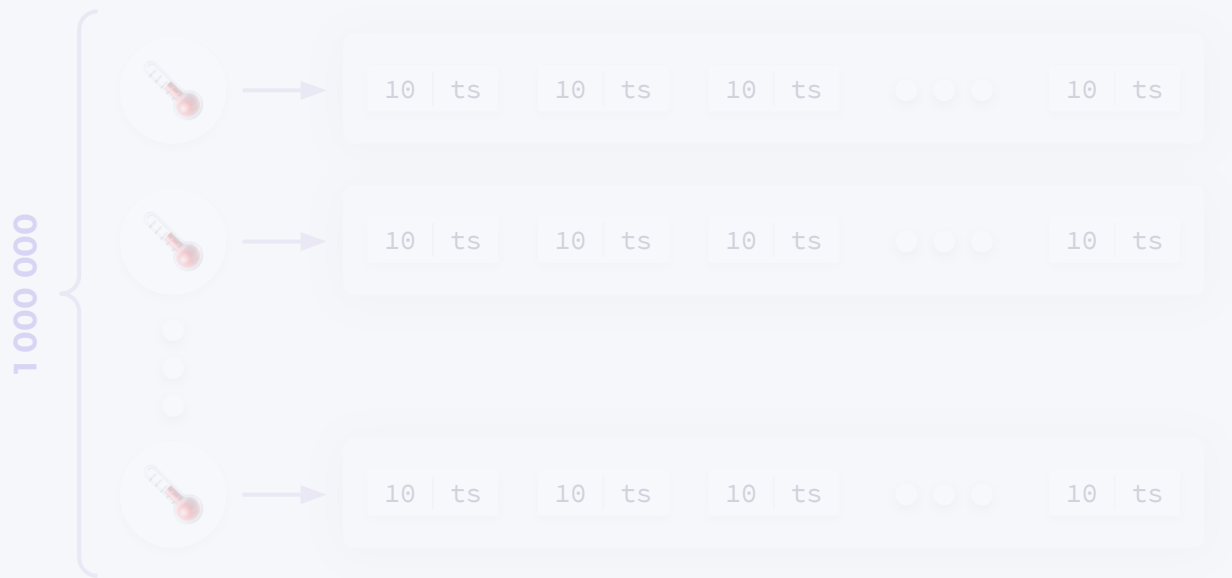


1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ byte}$

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”

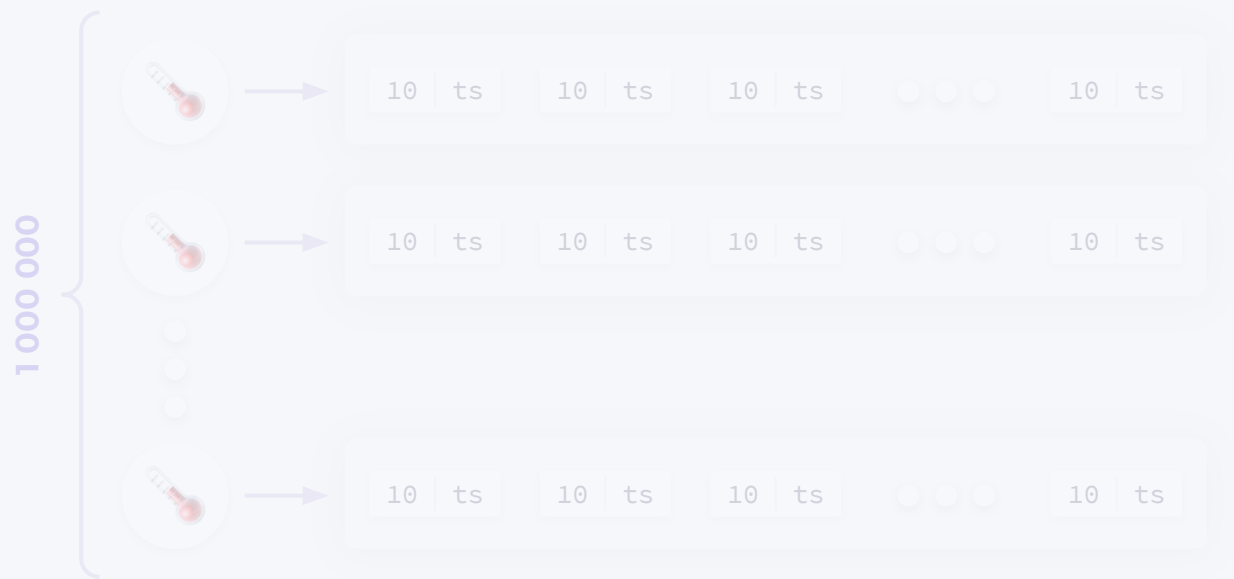


1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ byte} = 22 \text{ GB} / \text{day}$

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



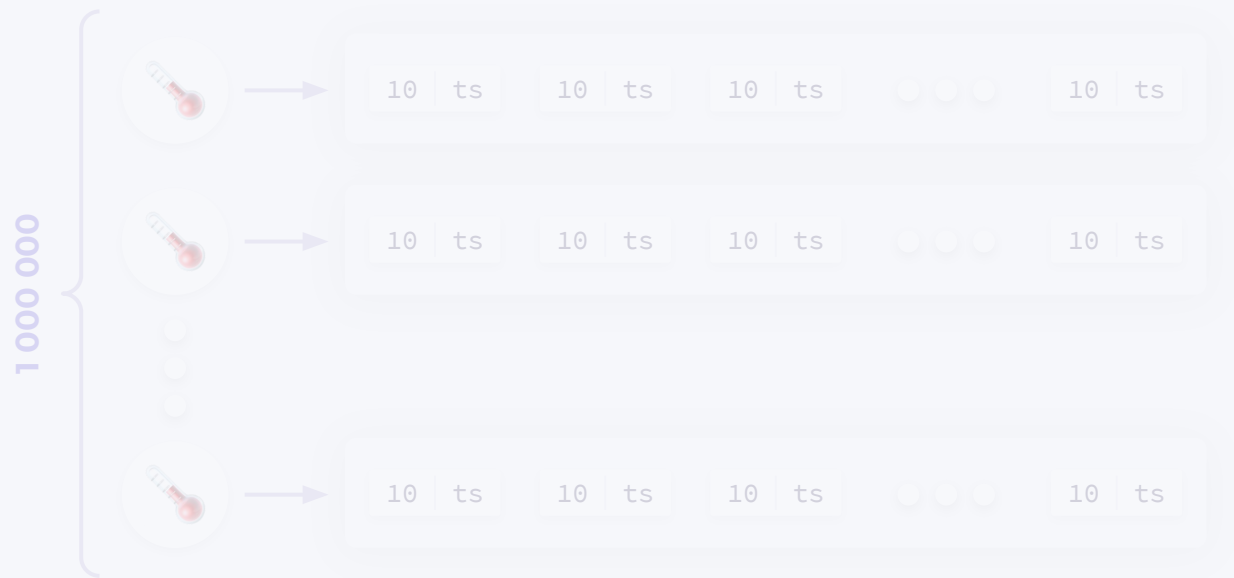
1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ byte} = 22 \text{ GB} / \text{day}$

16 byte

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



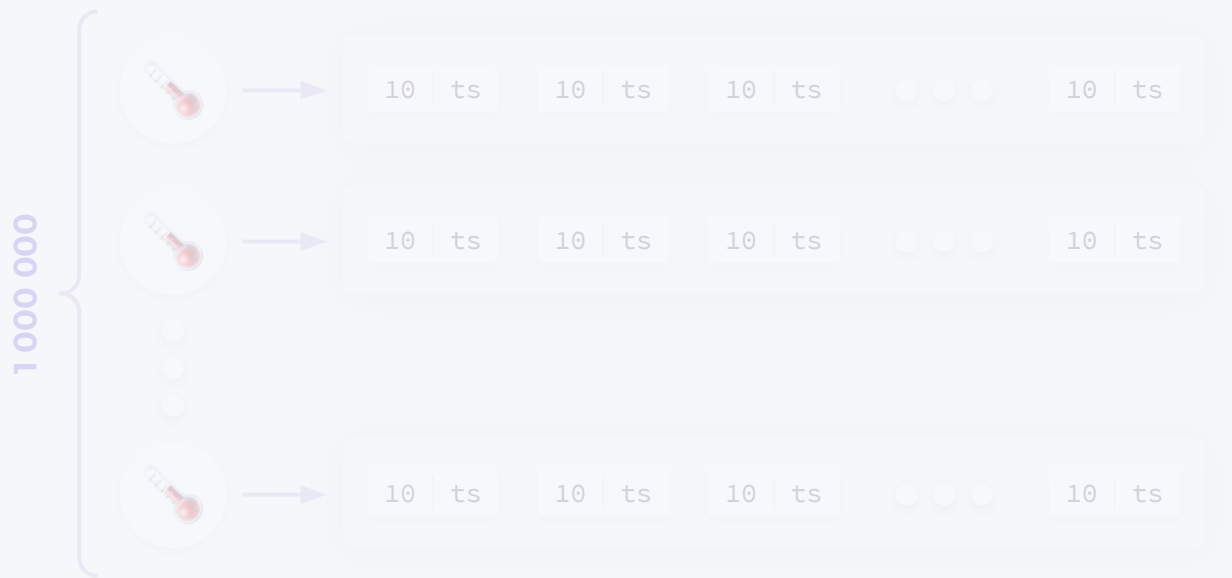
1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ byte} = 22 \text{ GB} / \text{day}$

16 byte → 

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



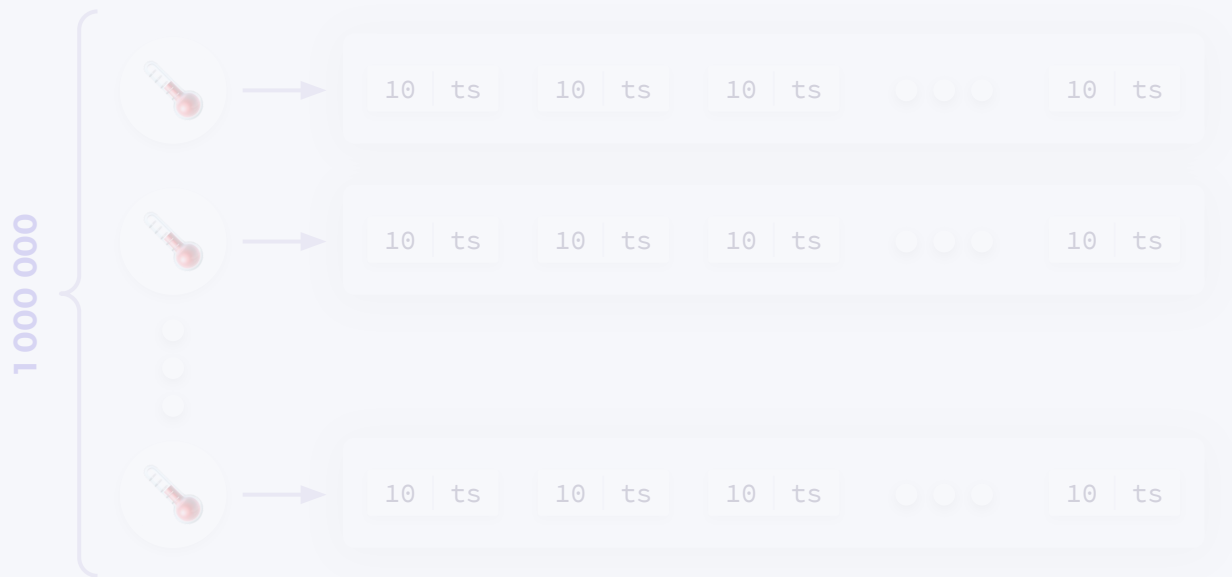
1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ byte} = 22 \text{ GB} / \text{day}$

$16 \text{ byte} \rightarrow \text{SSD} \rightarrow 16 \text{ KB}$

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”

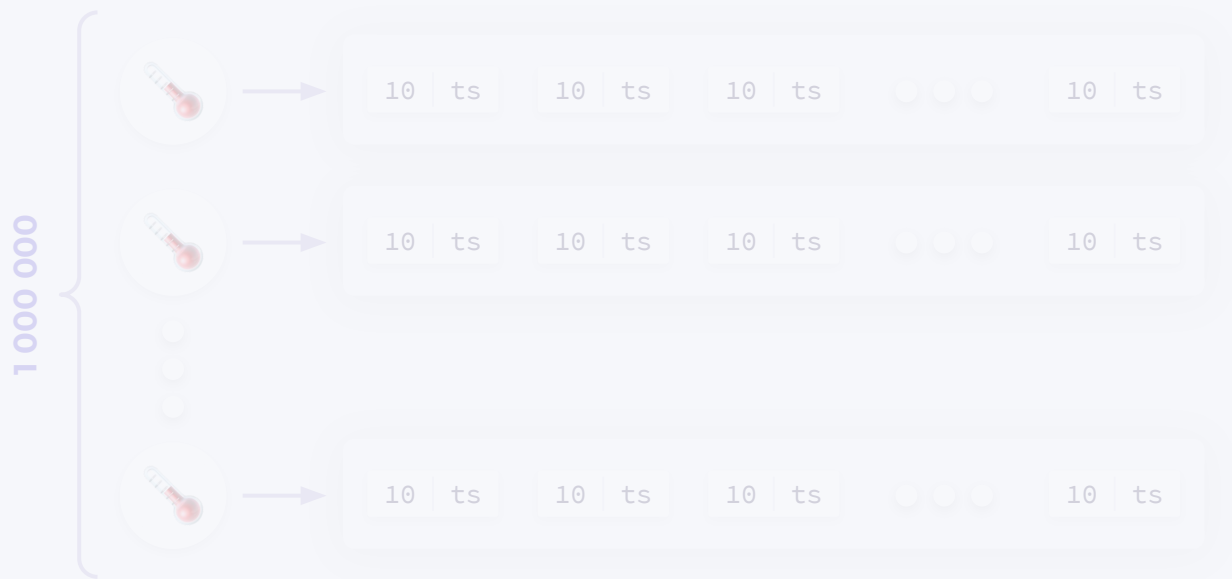


1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ KB}$

Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”

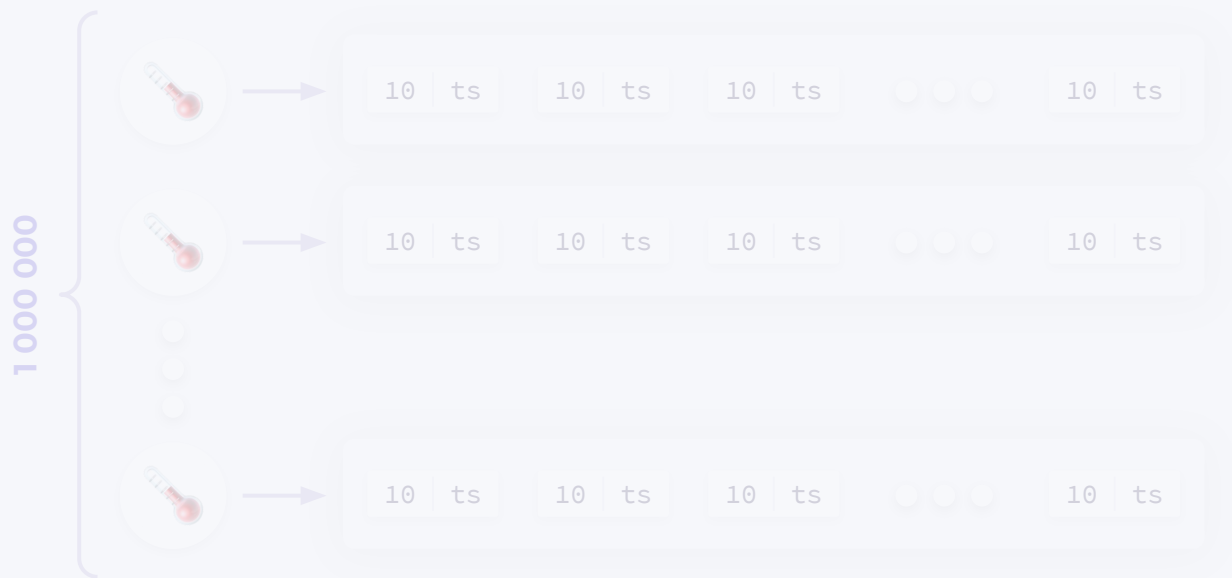


1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ KB} = 22 \text{ TB} / \text{day}$

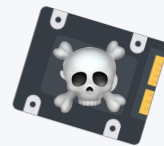
Как собираются данные со многих источников

Вариант #1: с каждого источника в свой “файл”



1 $1\,000\,000 \text{ IO} / 60 \text{ sec} = 16\,500 \text{ IOPS}$

2 $16\,500 \text{ IOPS} * 16 \text{ KB} = 22 \text{ TB} / \text{day}$

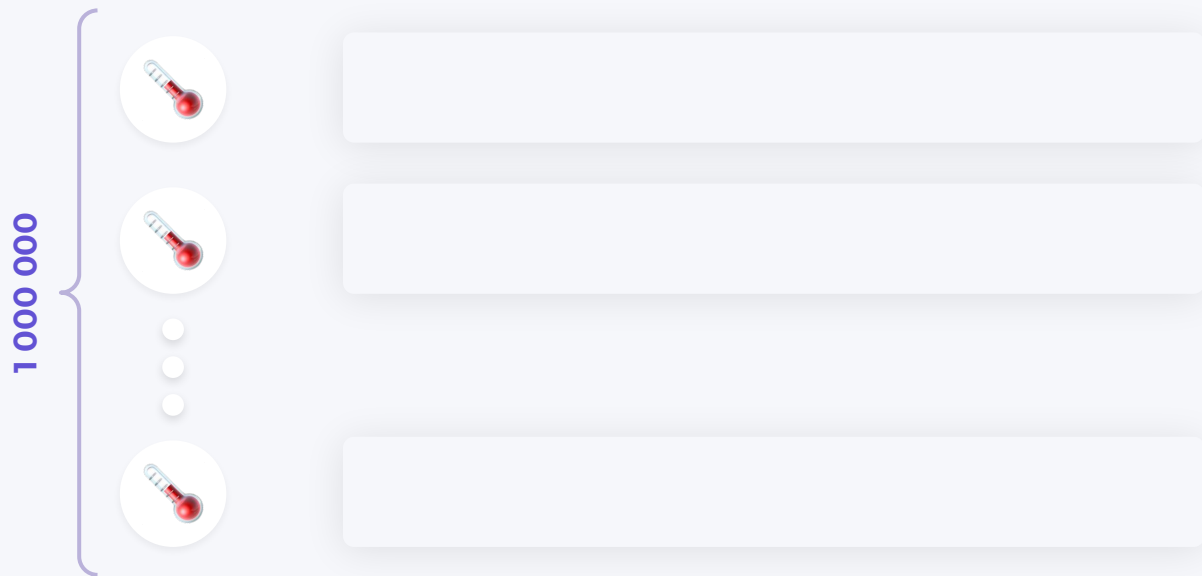


Как собираются данные со многих источников

Вариант #2: со всех источников в один файл

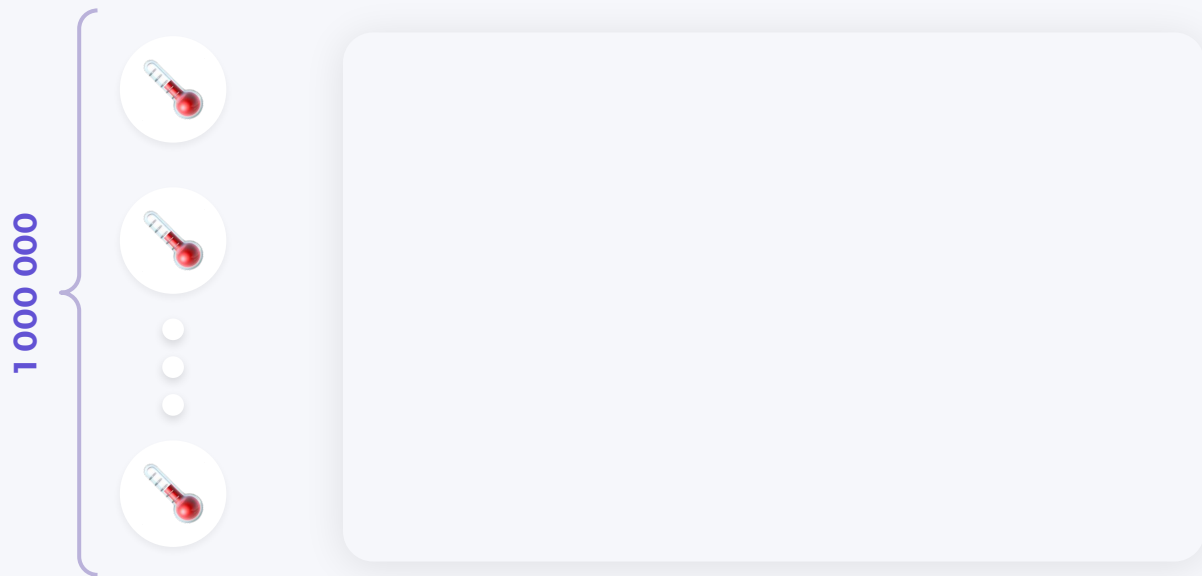
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



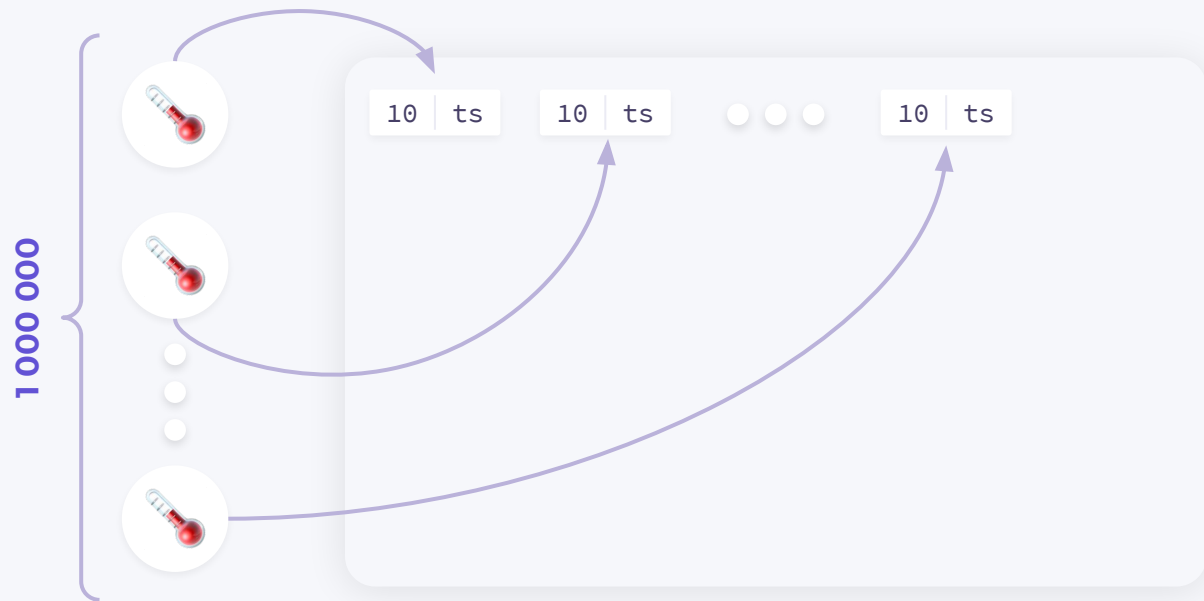
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



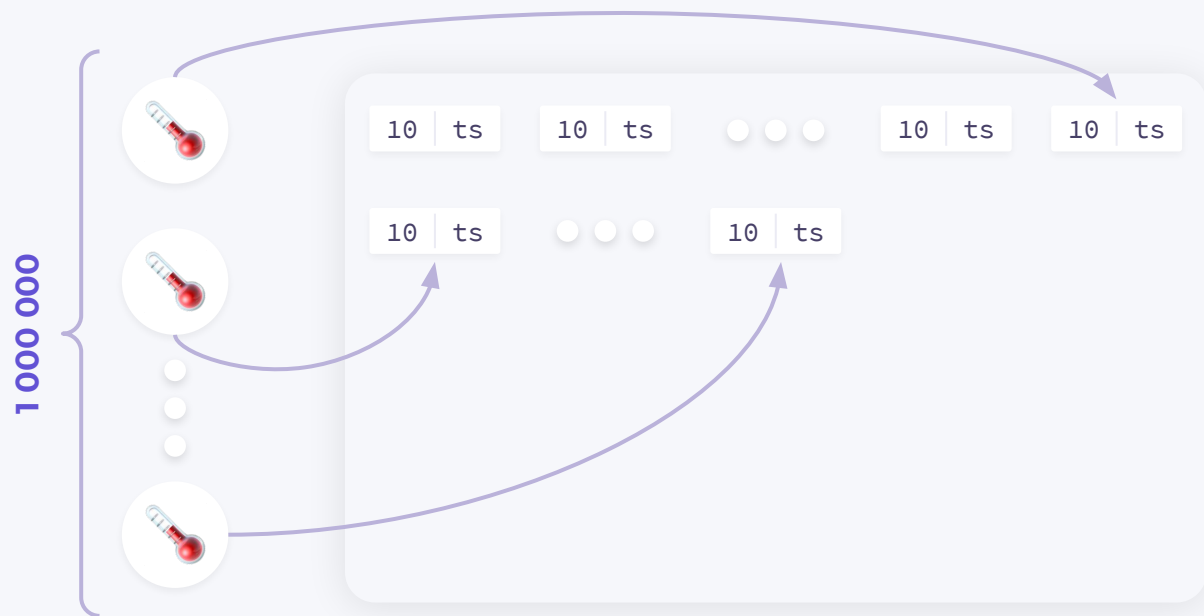
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



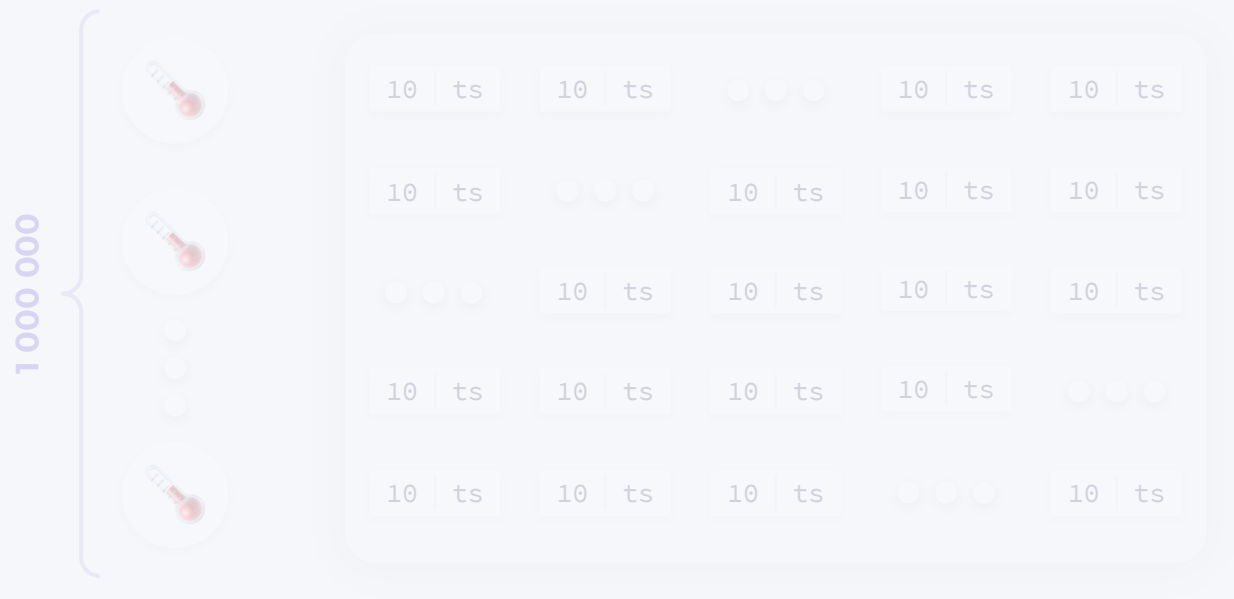
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



Как собираются данные со многих источников

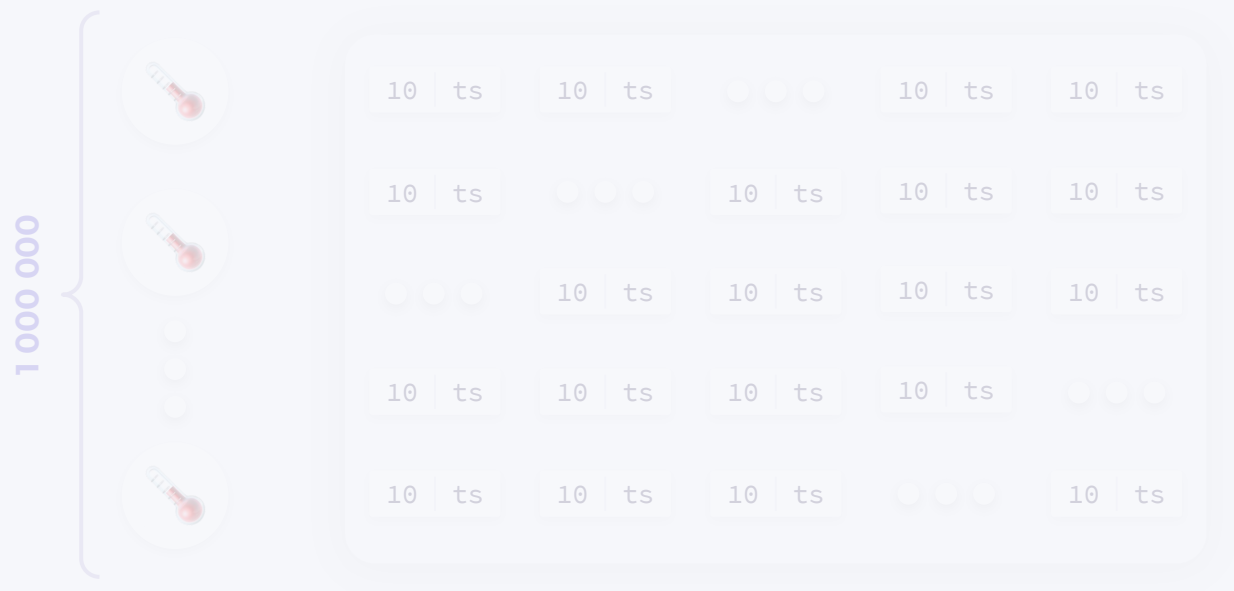
Вариант #2: со всех источников в один файл



16 byte * 1 000 000

Как собираются данные со многих источников

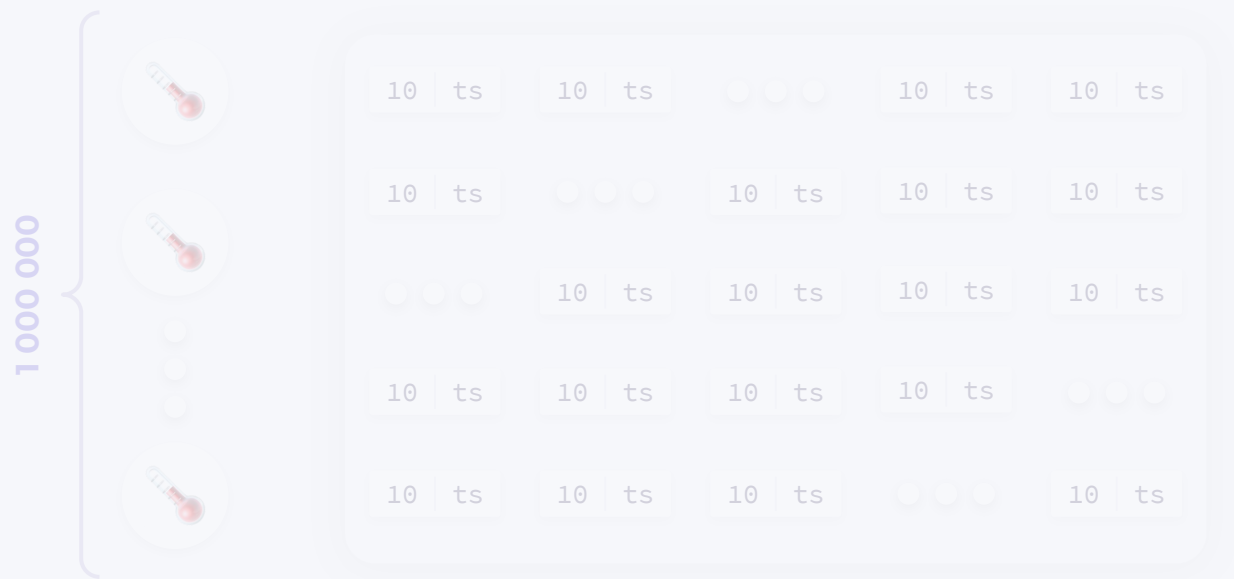
Вариант #2: со всех источников в один файл



16 byte * 1 000 000 = **15 MB / 60 sec**

Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



16 byte * 1 000 000 = 15 MB / 60 sec = 23 GB / day



Как собираются данные со многих источников

Вариант #2: со всех источников в один файл

1 000 000



10

ts

10

ts

● ● ●

10

ts

10

ts

10

ts

● ● ●

10

ts

10

ts

10

ts

● ● ●

10

ts

10

ts

10

ts

10

ts

10

ts

10

ts

10

ts

10

ts

● ● ●

10

ts

10

ts

10

ts

● ● ●

10

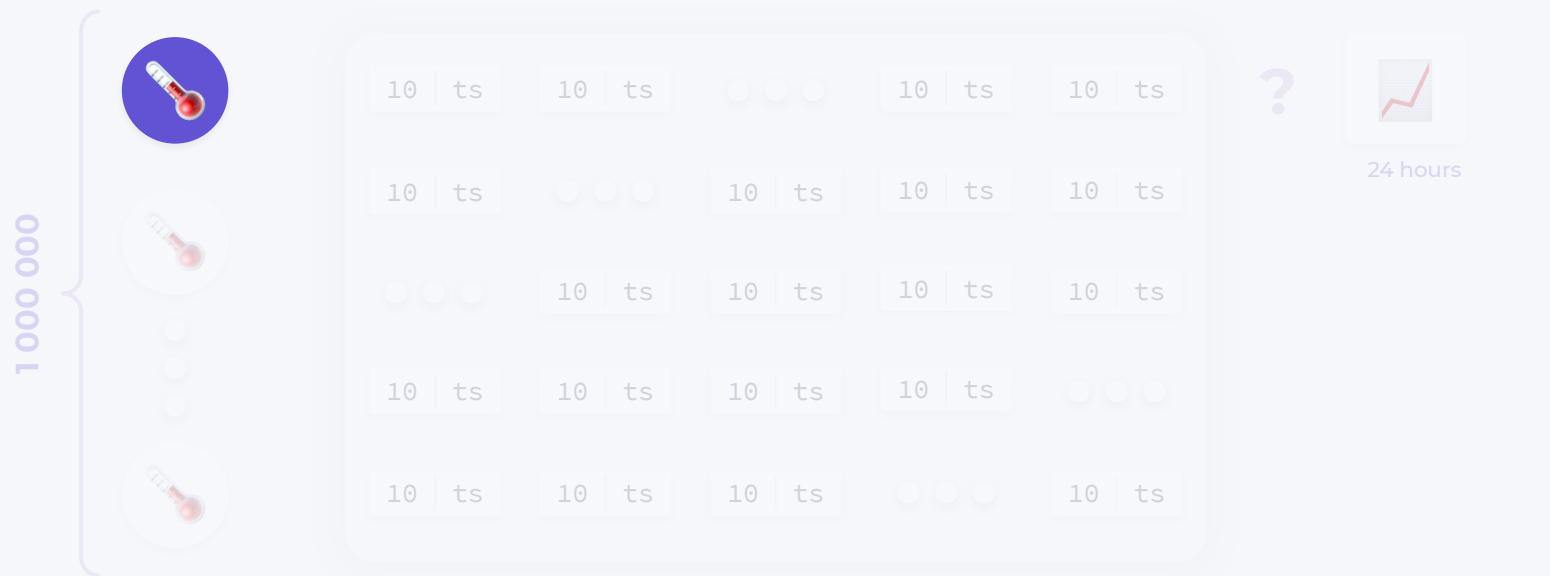
ts



24 hours

Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



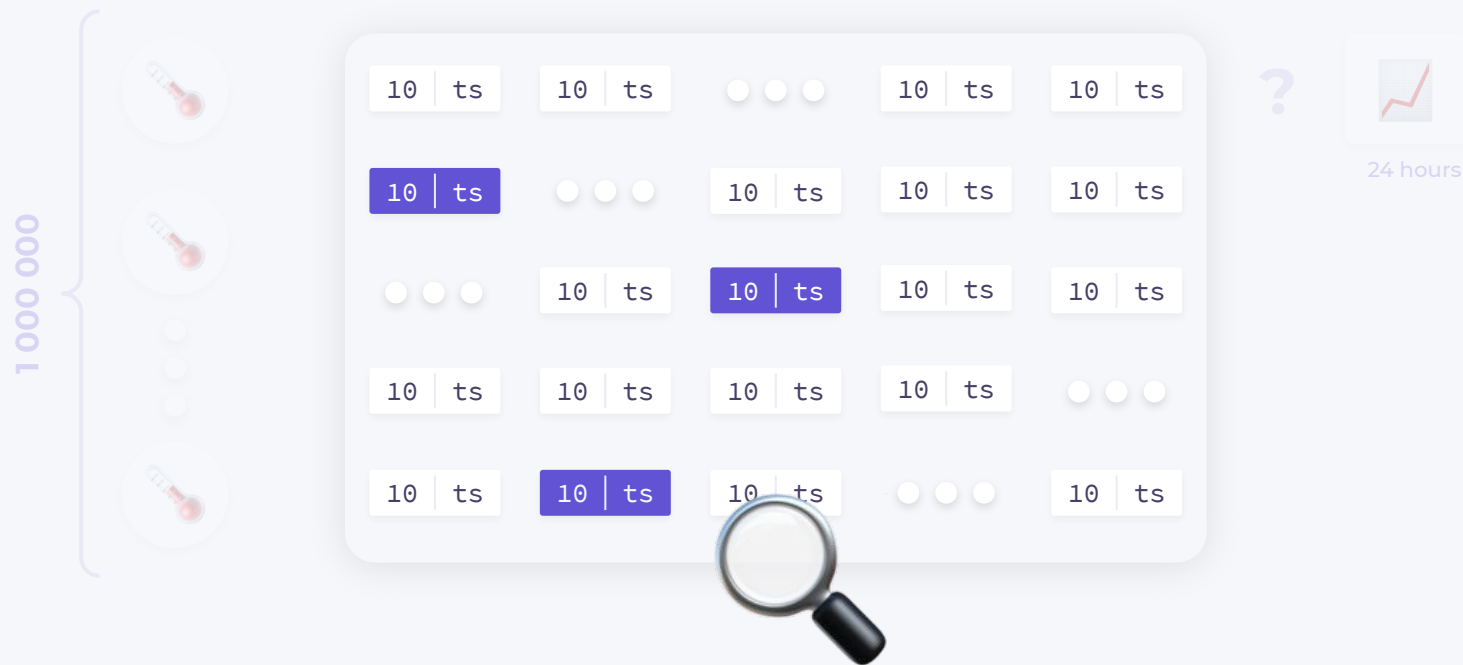
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



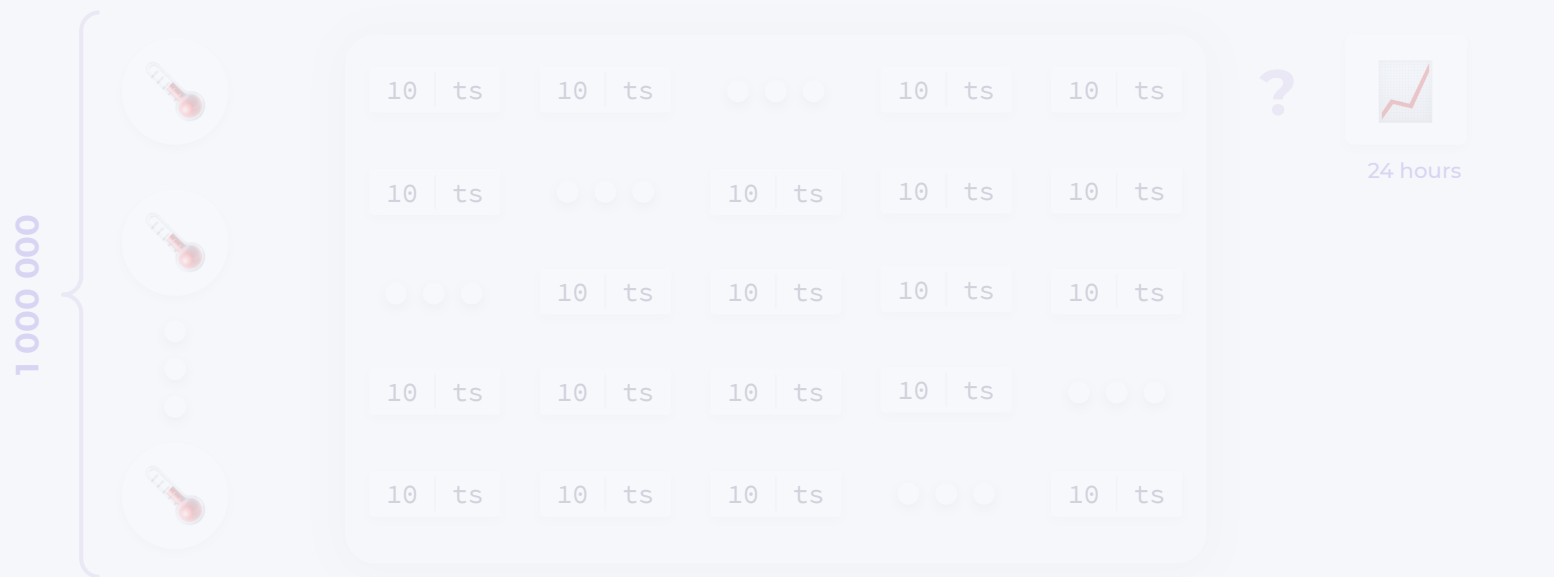
Как собираются данные со многих источников

Вариант #2: со всех источников в один файл



Как собираются данные со многих источников

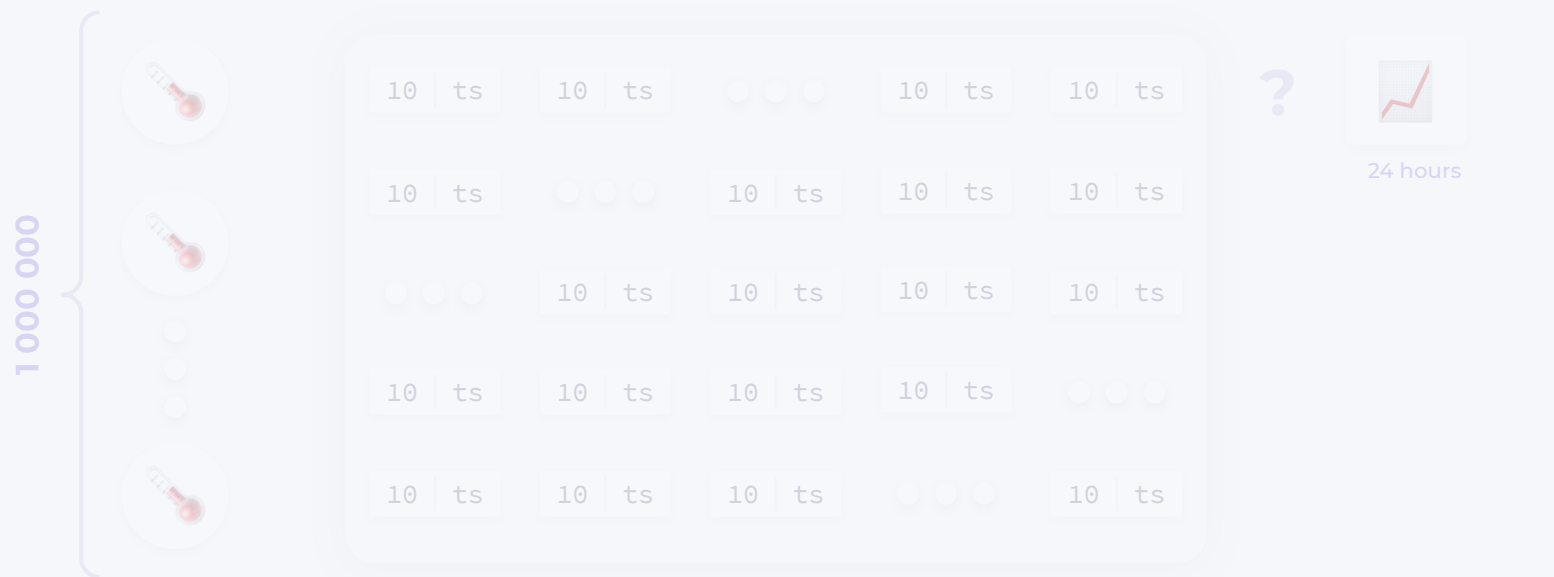
Вариант #2: со всех источников в один файл



$$16 \text{ byte} * 1\,000\,000 = 15 \text{ MB} / 60 \text{ sec} = 23 \text{ GB} / \text{day}$$

Как собираются данные со многих источников

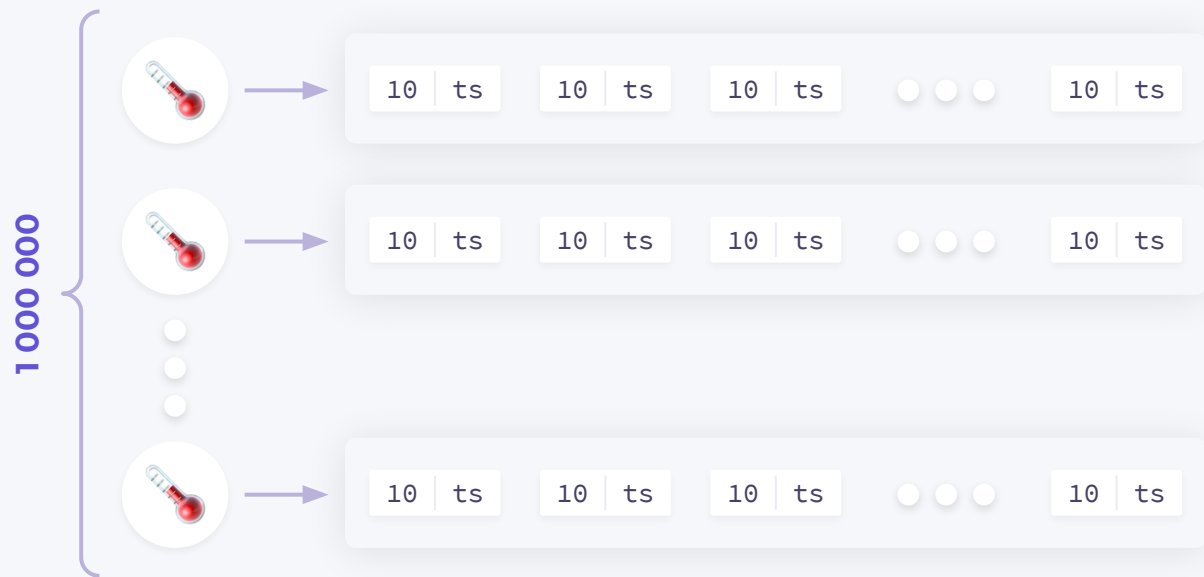
Вариант #2: со всех источников в один файл



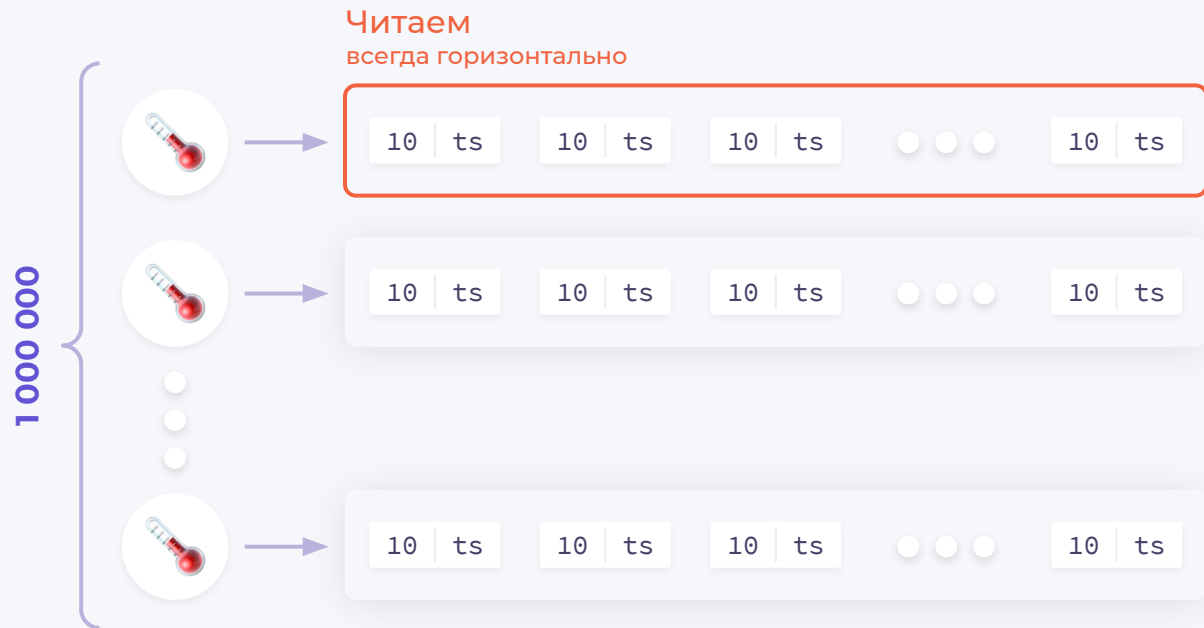
$$16 \text{ byte} * 1\,000\,000 = 15 \text{ MB} / 60 \text{ sec} = 23 \text{ GB} / \text{day}$$

Проблема

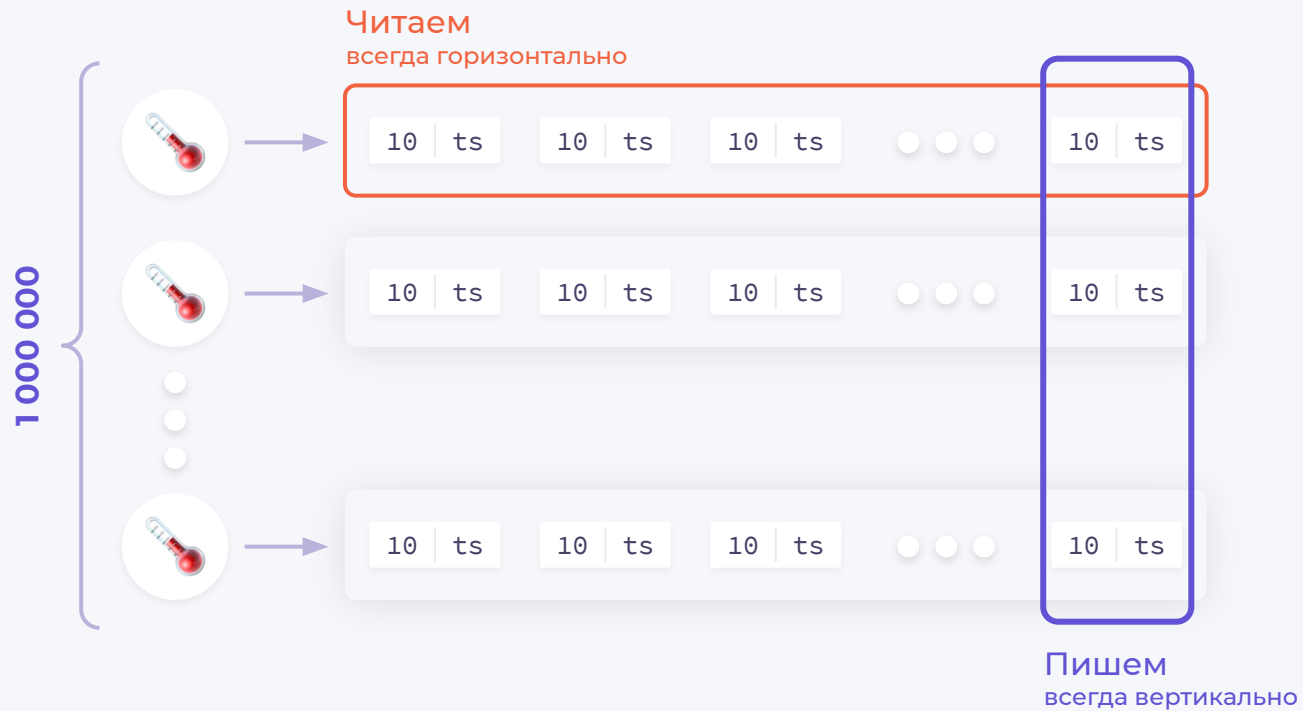
Проблема



Проблема

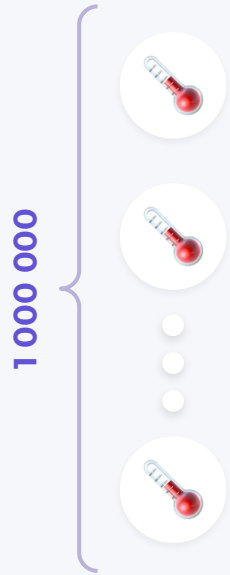


Проблема

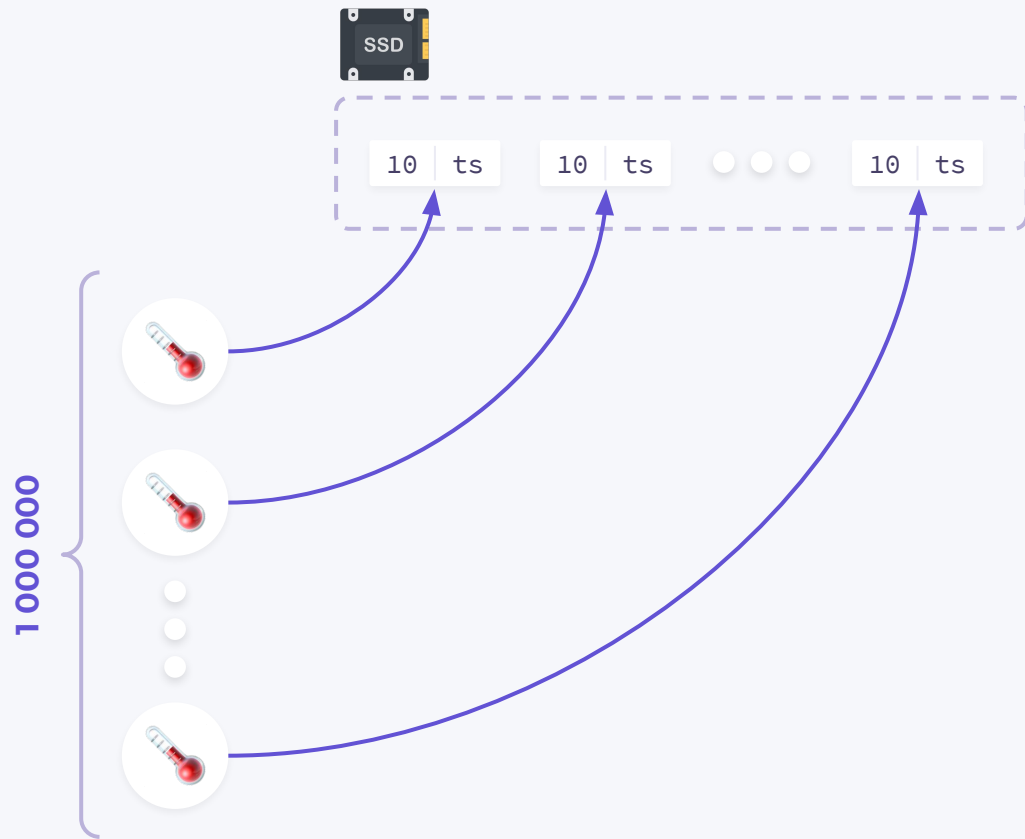


Как решать?

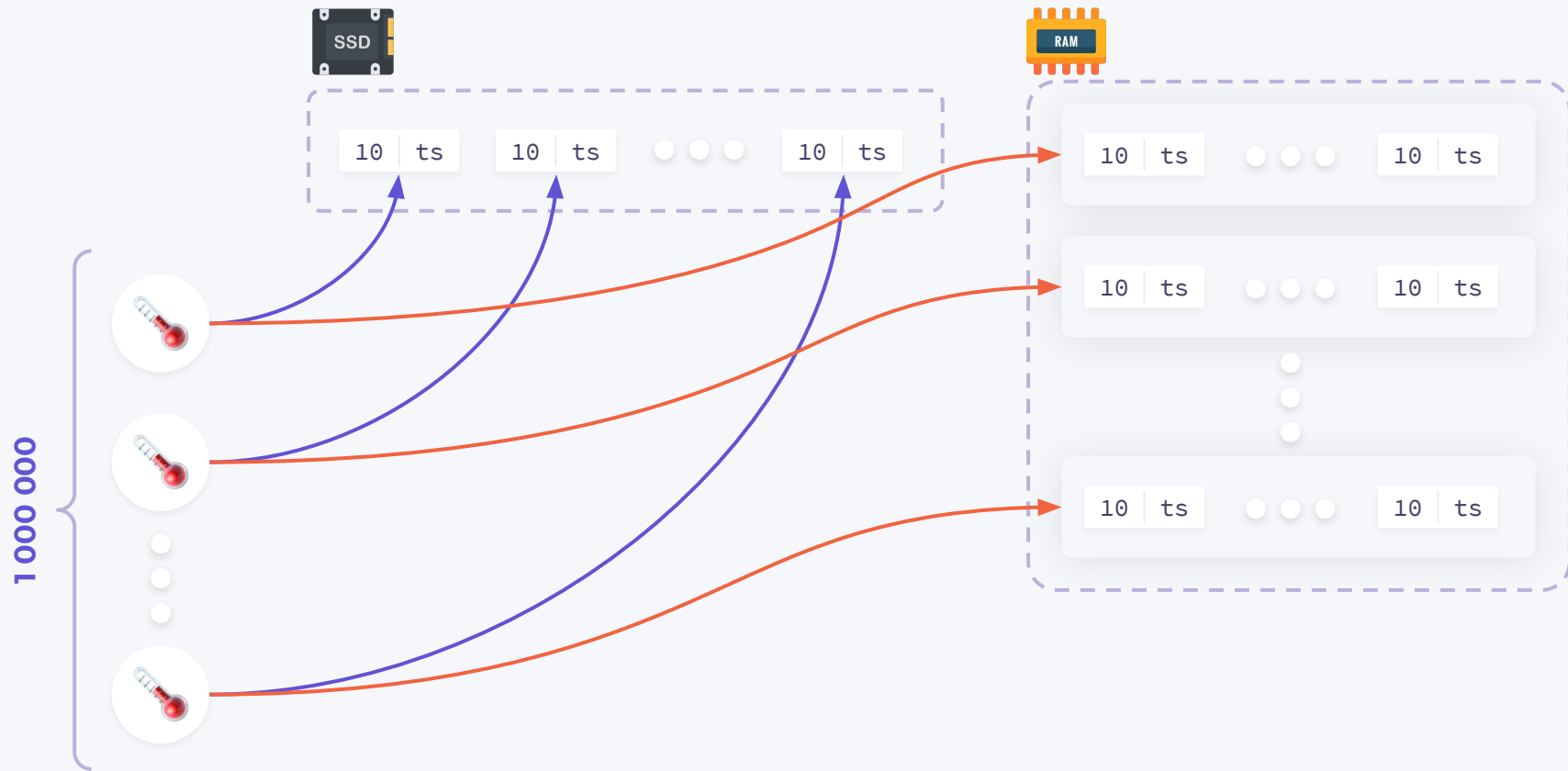
Как решать?



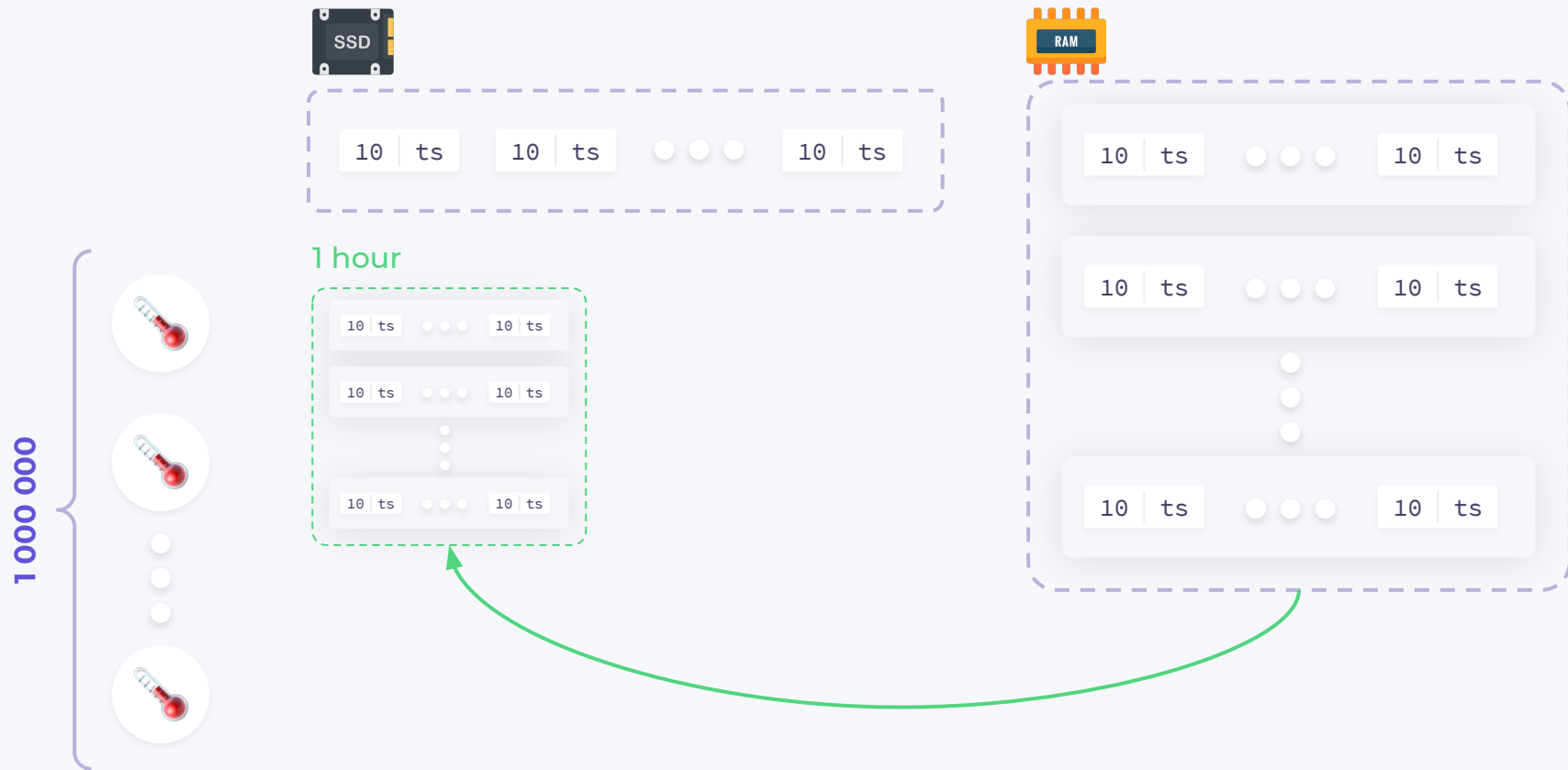
Как решать?



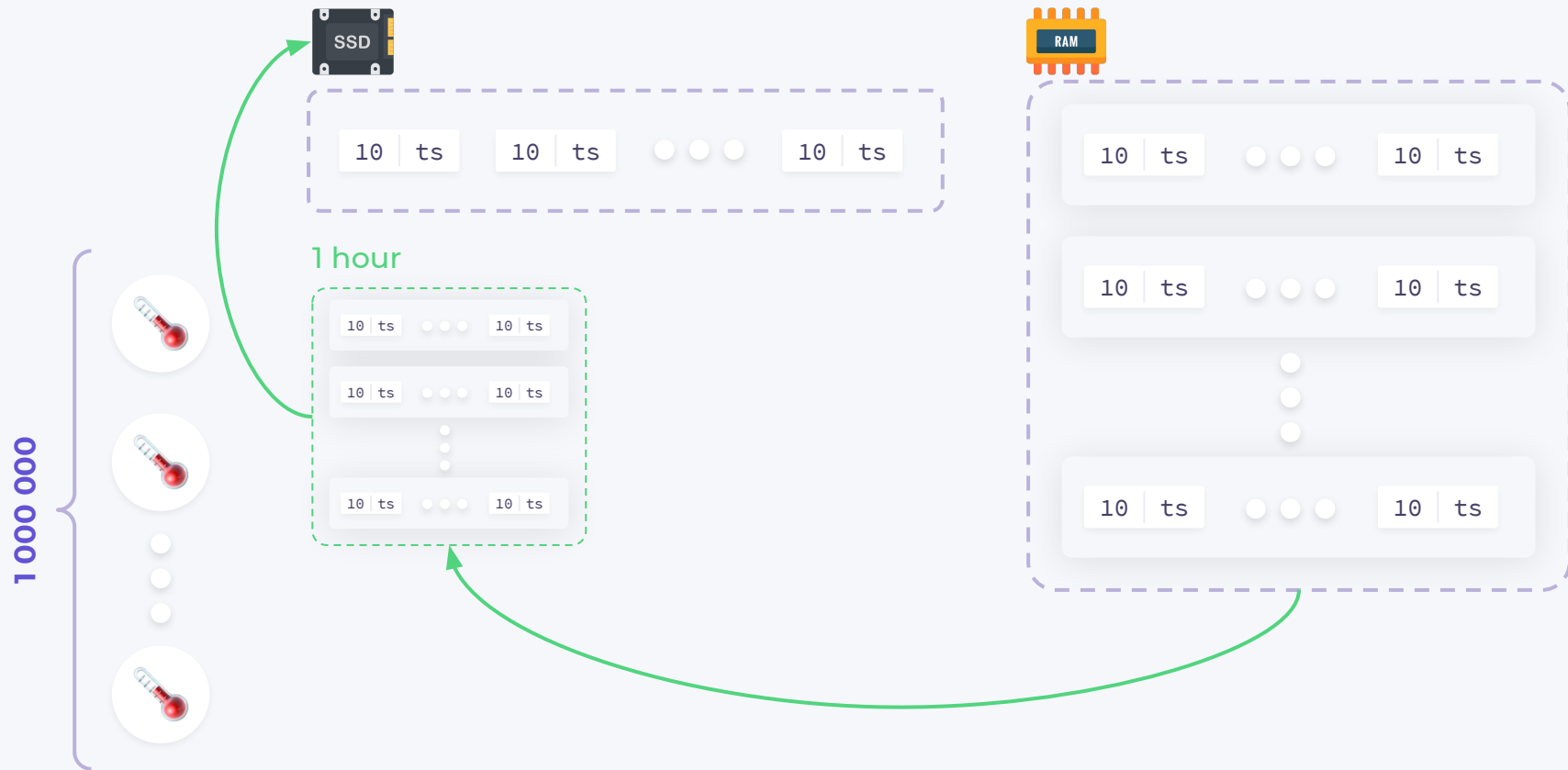
Как решать?



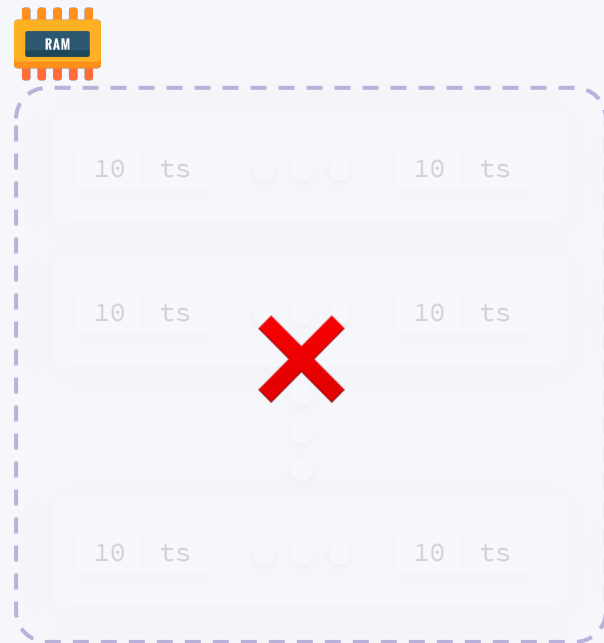
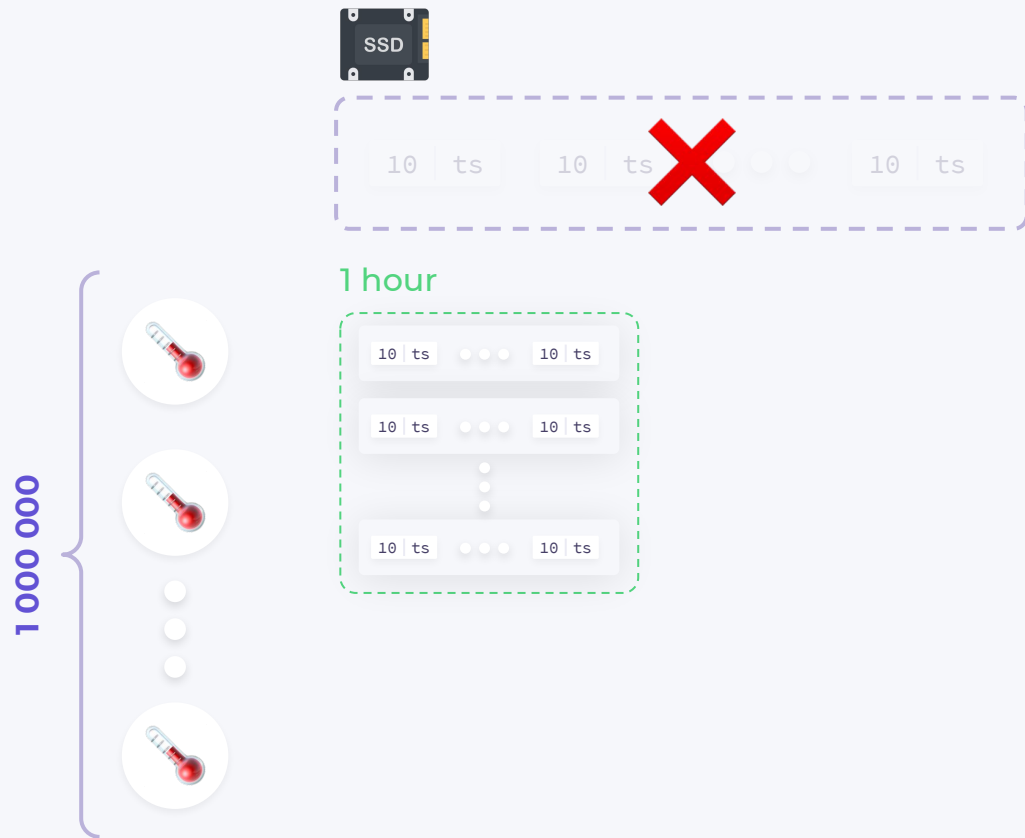
Как решать?



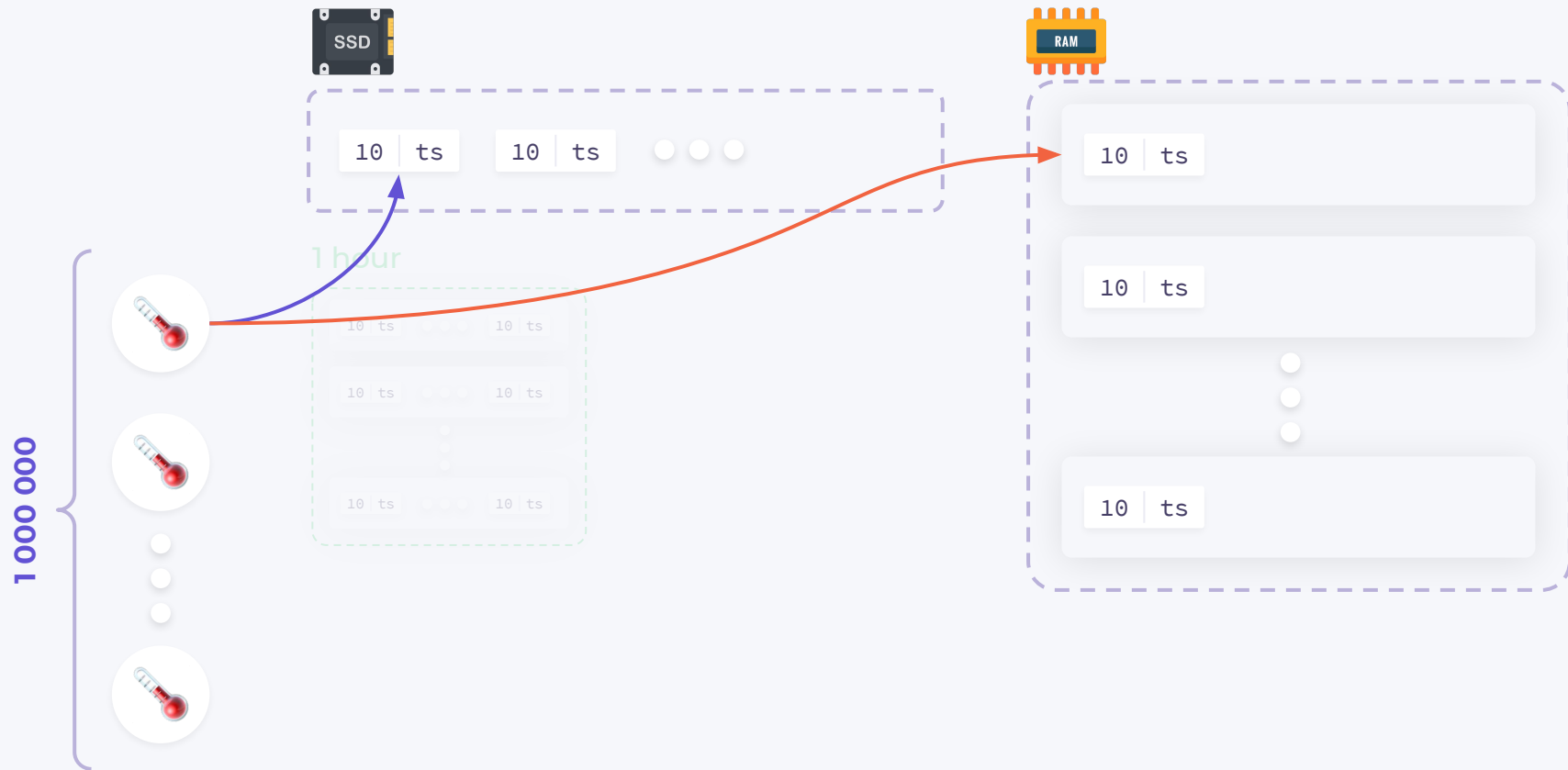
Как решать?



Как решать?



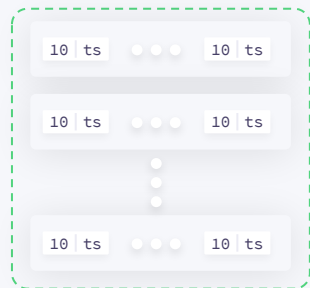
Как решать?



Как решать?

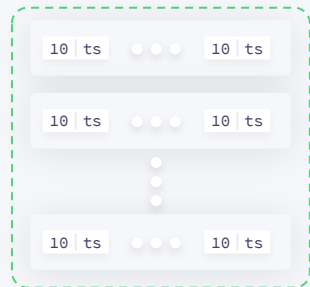
Как решать?

1 hour



Как решать?

1 hour

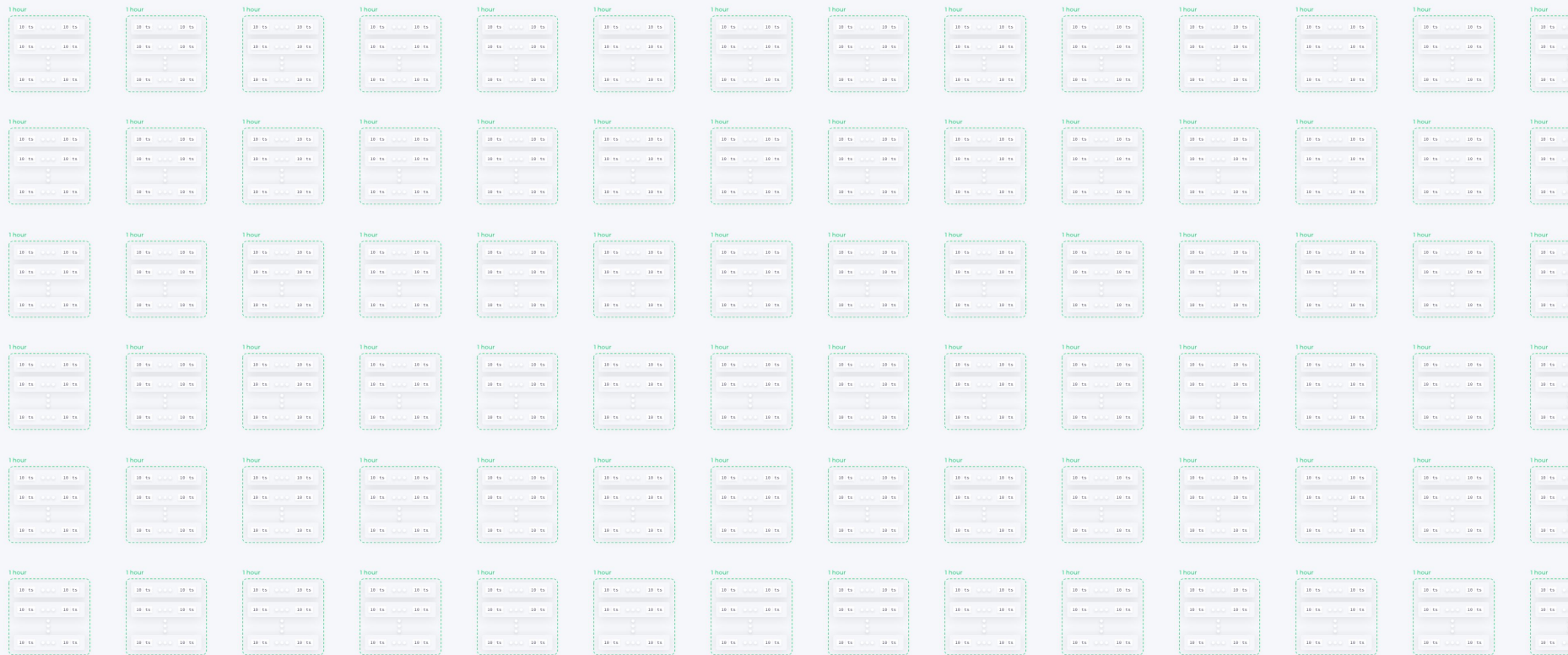


1 hour



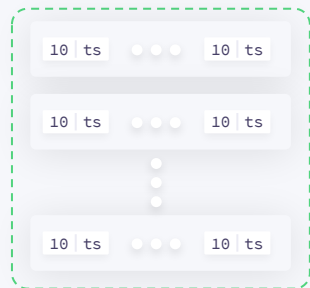
Как решать?

Спустя месяц...



Как решать?

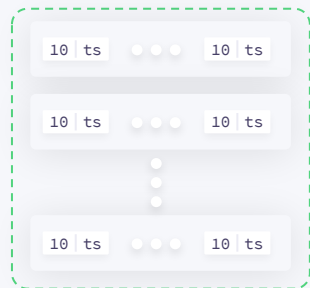
1 hour



1 hour



1 hour



1 hour



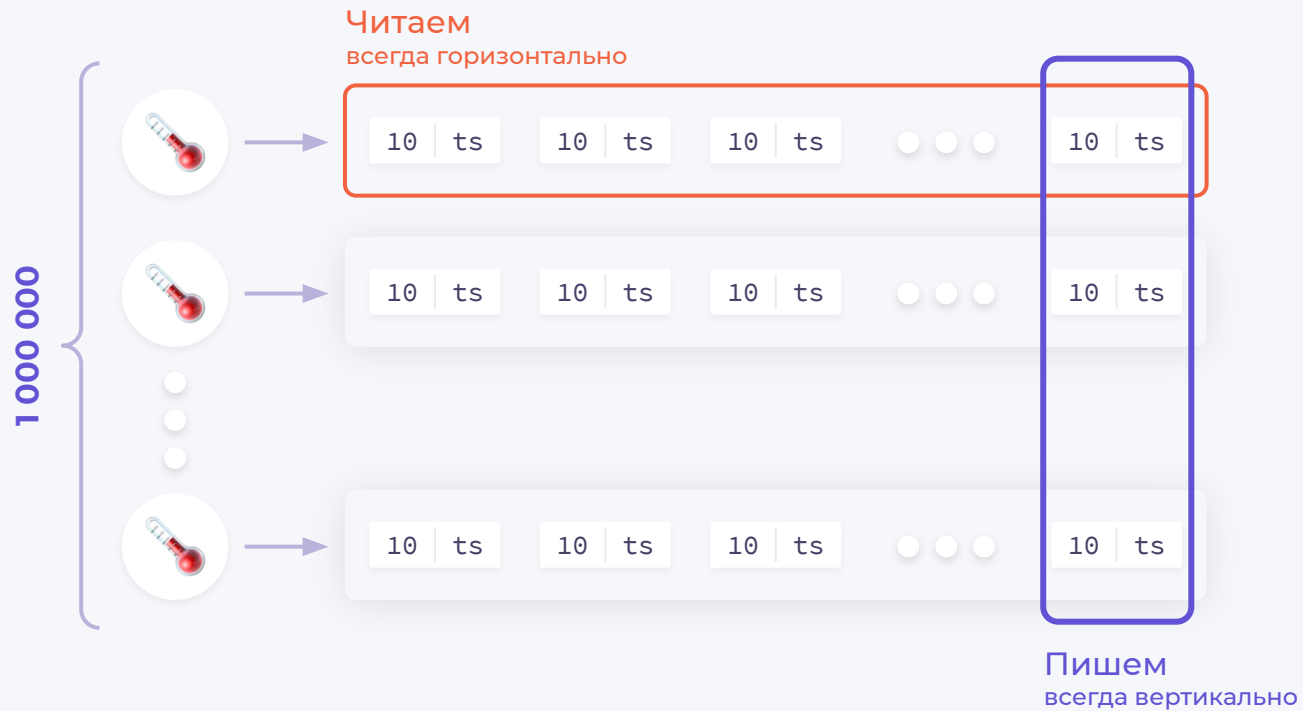
Операция “merge”



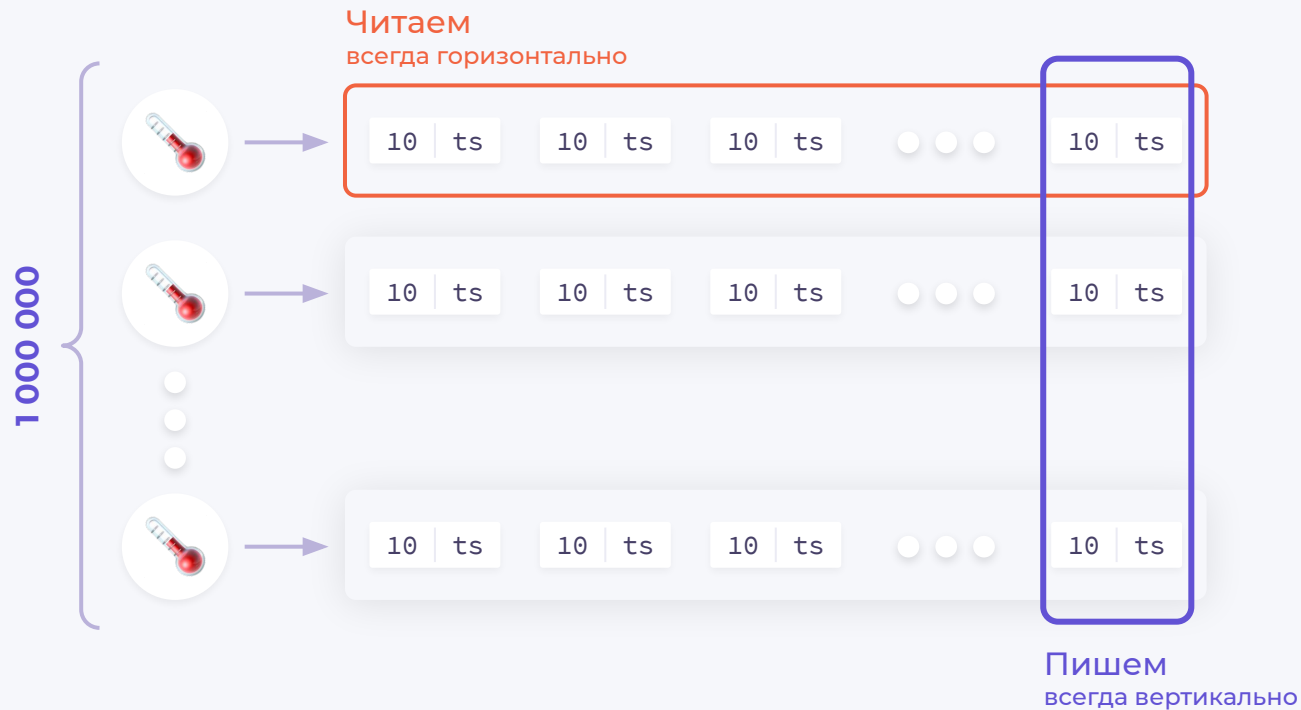
Операция “merge”



Проблема



Проблема больше не проблема





10	ts
----	----

10	ts
----	----

10	ts
----	----



10	ts
----	----



10	ts
----	----

10	ts
----	----

10	ts
----	----



10	ts
----	----



10	ts
----	----

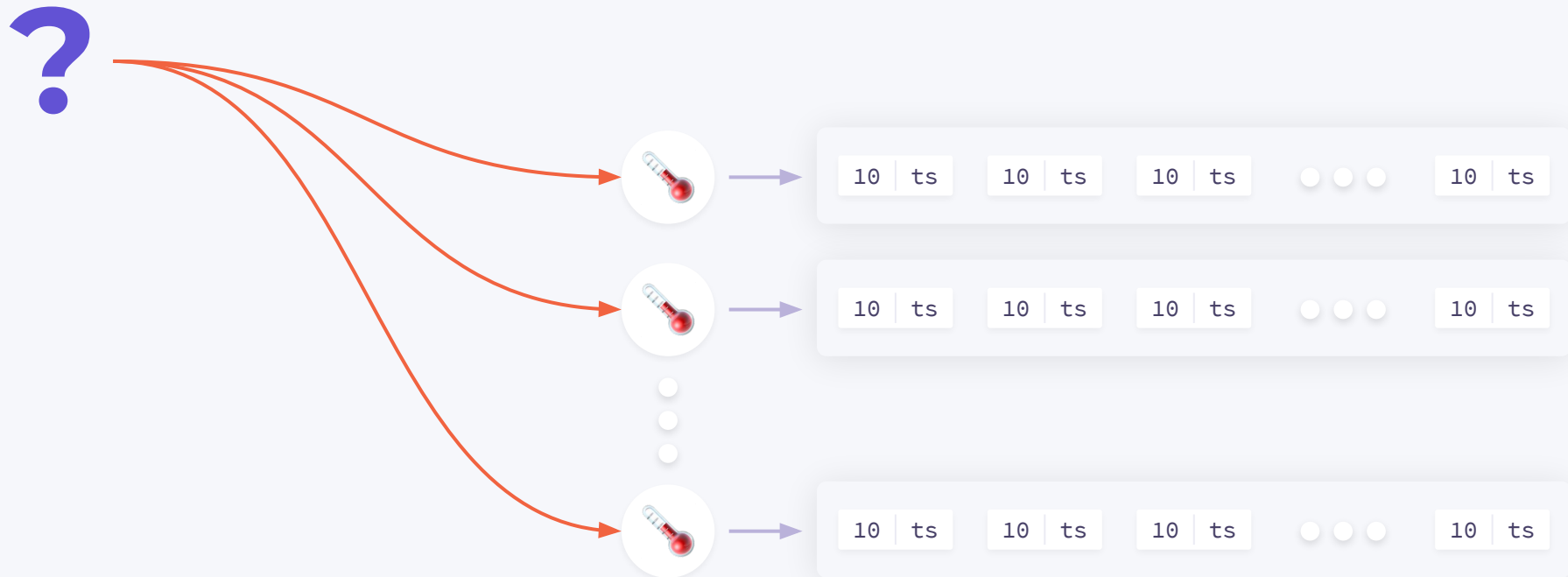
10	ts
----	----

10	ts
----	----



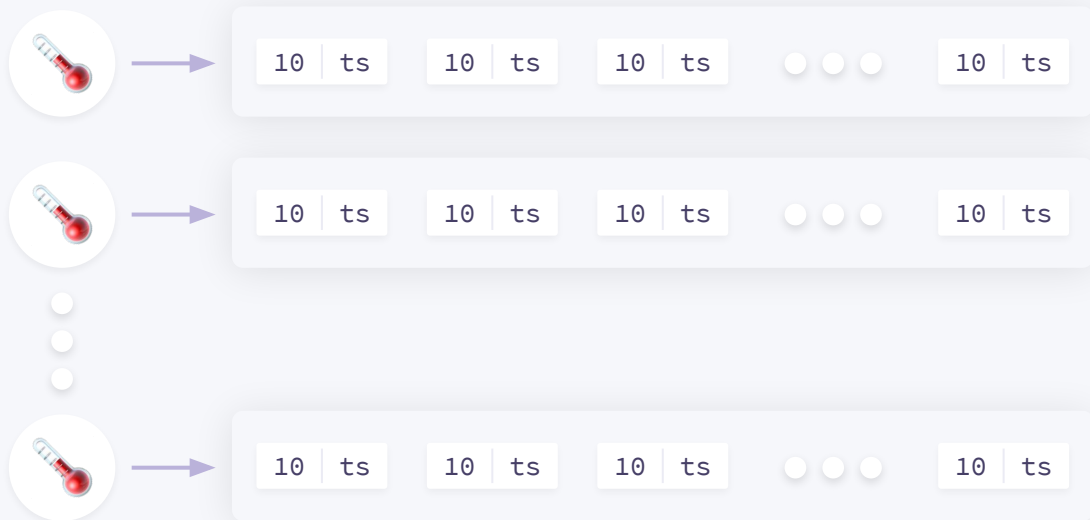
10	ts
----	----

Как идентифицировать?



Как идентифицировать?

LabelSet



Как идентифицировать?

LabelSet

```
{  
  ключ: значение,  
  ключ: значение, ...  
}
```


Как идентифицировать?

LabelSet

```
{  
  ключ:  значение,  
  ключ:  значение, ...  
}
```

```
{  
  name:  cpu_usage,  
  node:  curiosity,  
  core:  4  
}
```

Как идентифицировать?

LabelSet

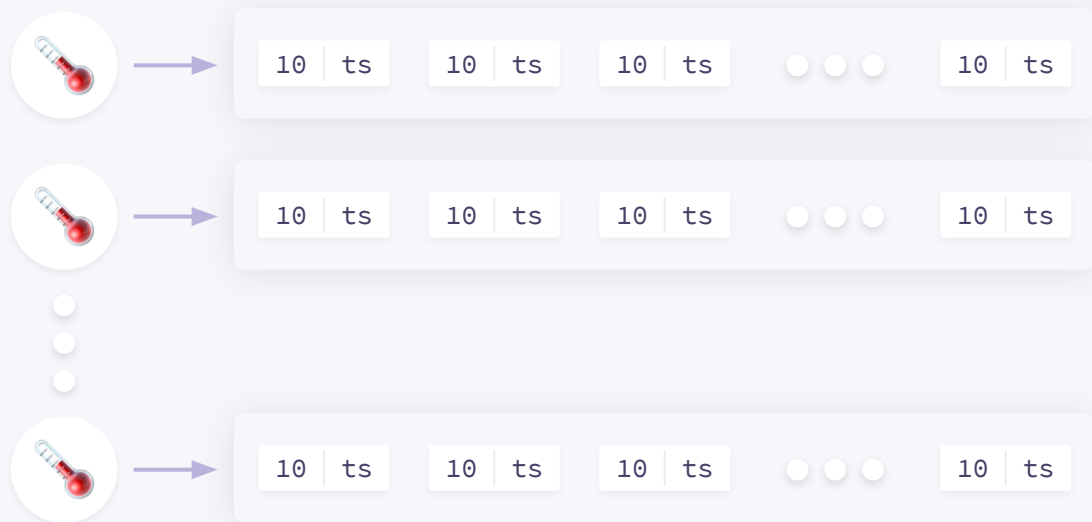
```
{  
  ключ:  значение,  
  ключ:  значение, ...  
}
```

```
{  
  name:  cpu_usage,  
  node:  curiosity,  
  core:  4  
}
```

```
{  
  name:      cpu_usage,  
  pod:       endeavour-123512315-ffskd,  
  namespace: mars,  
  node:      kube-a-1  
}
```

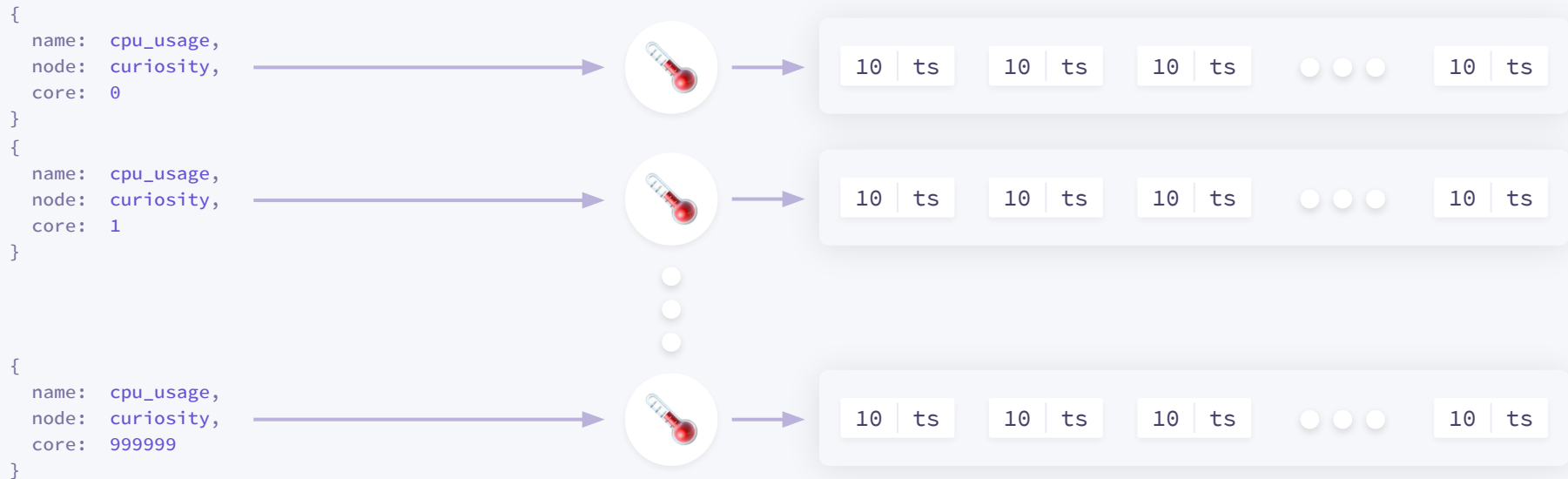
Как идентифицировать?

LabelSet



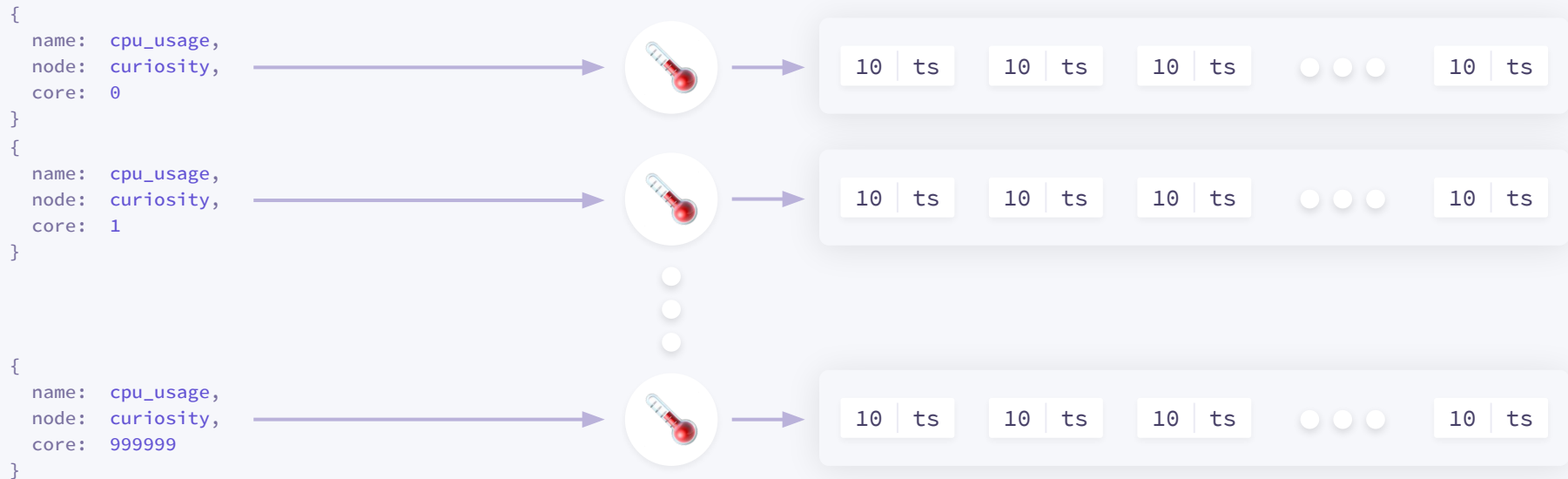
Как идентифицировать?

LabelSet



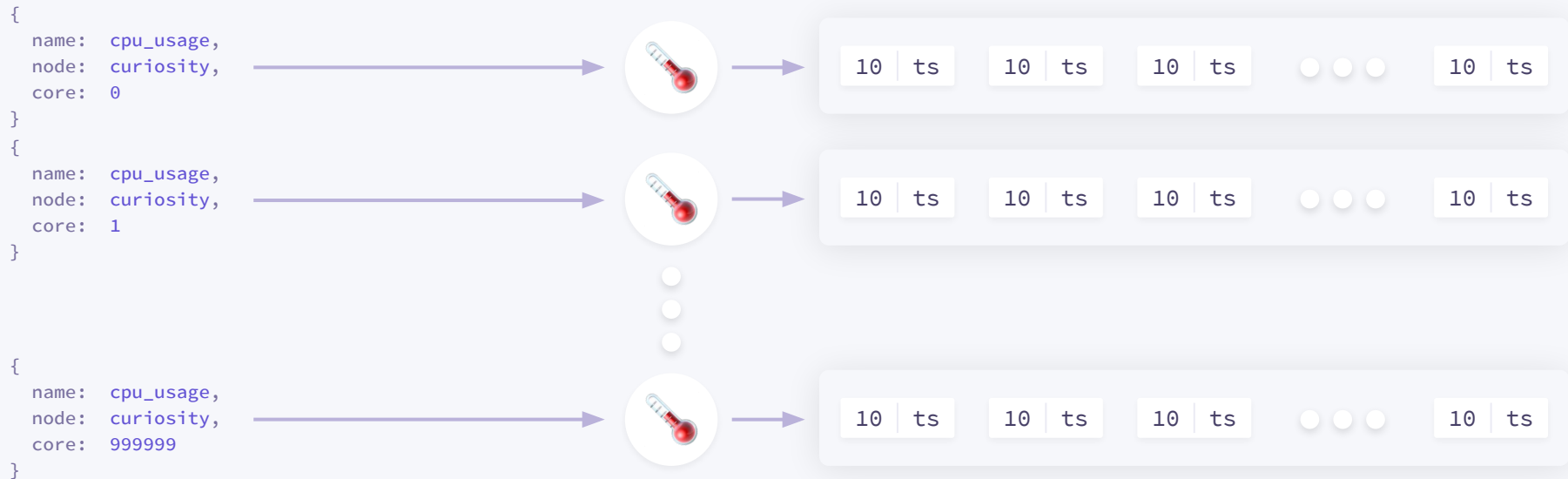
Как идентифицировать?

{ls}



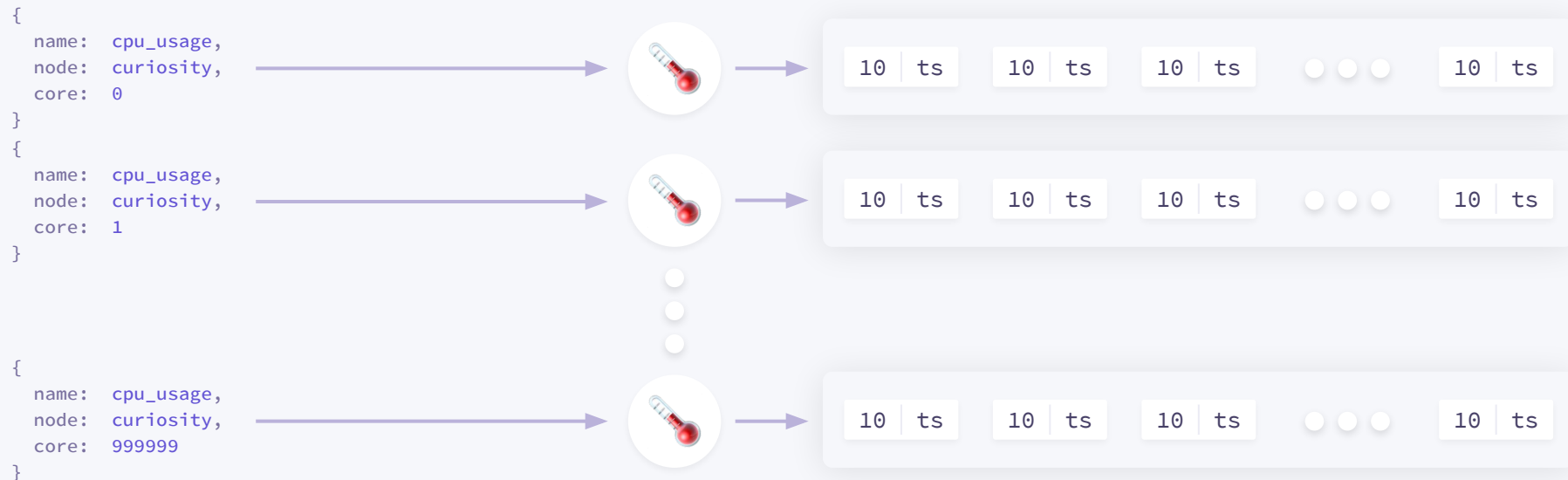
Как идентифицировать?

{ls} → **#t_id**

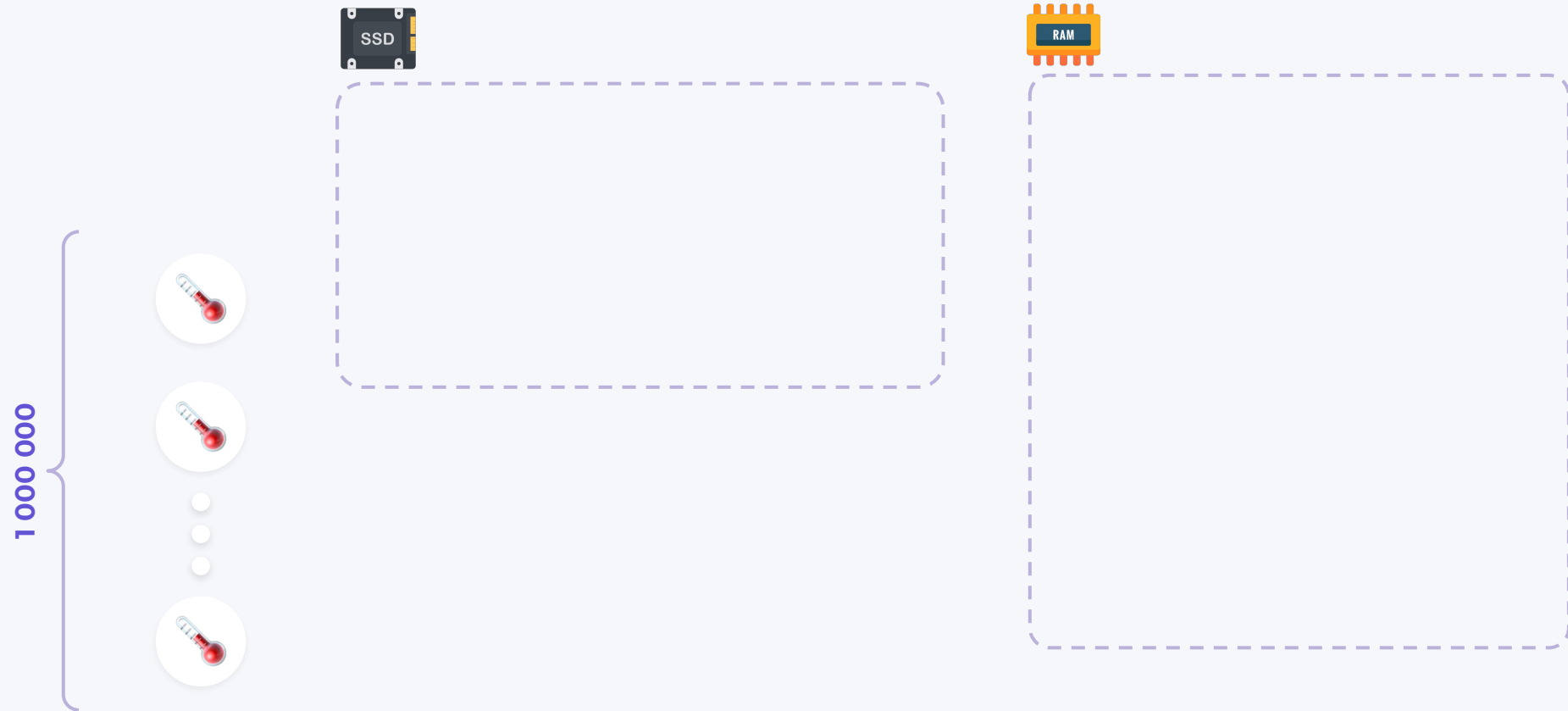


Как идентифицировать?

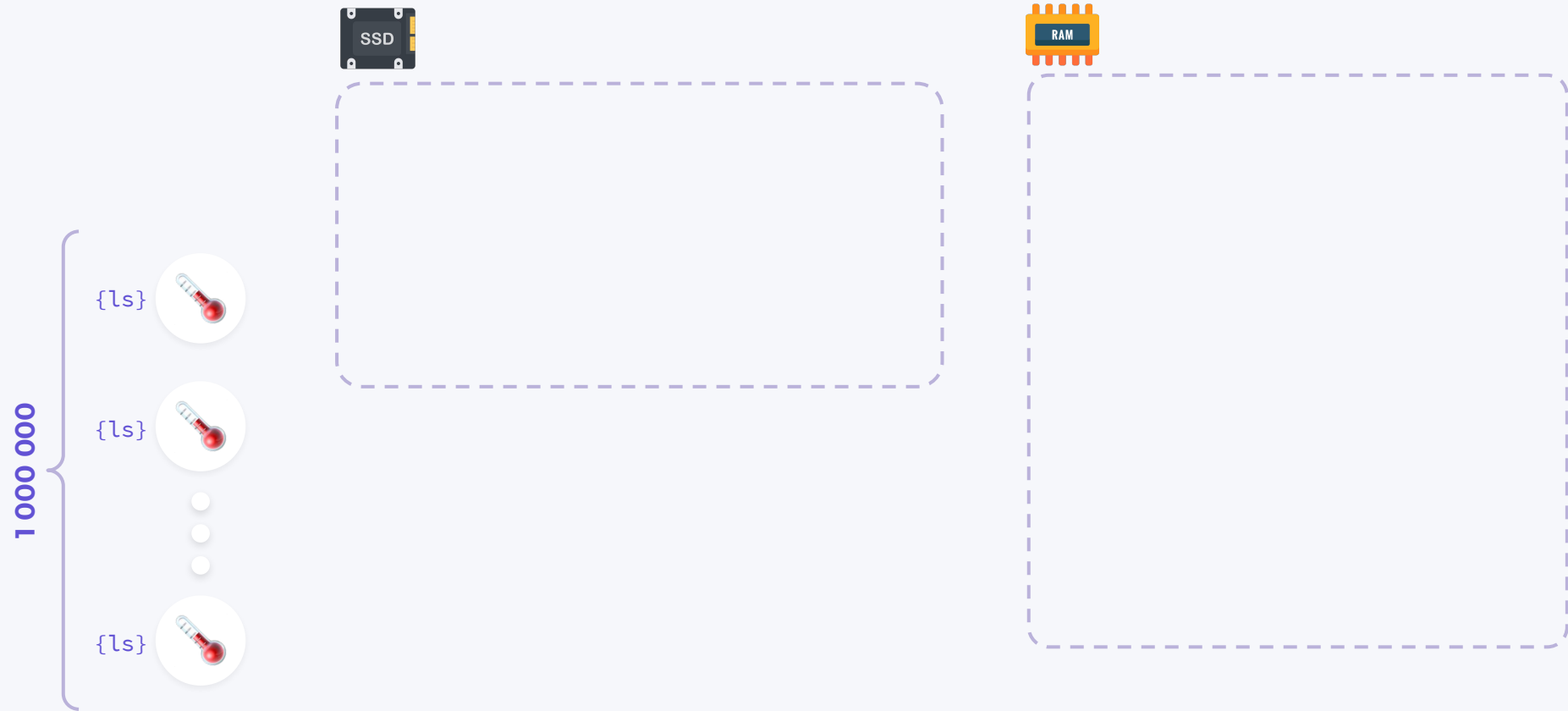
{ls} → **#t_id** → **Временной ряд**



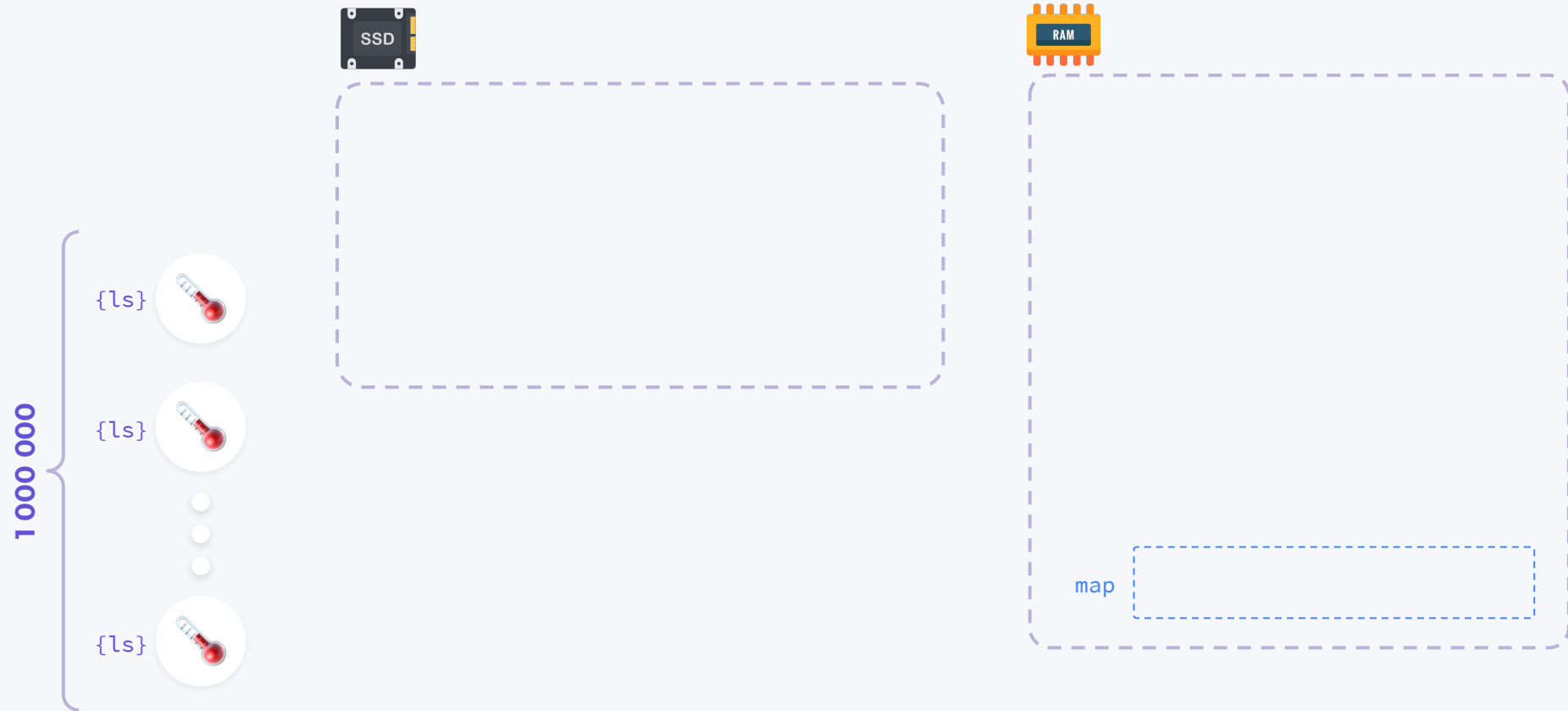
Как идентифицировать?



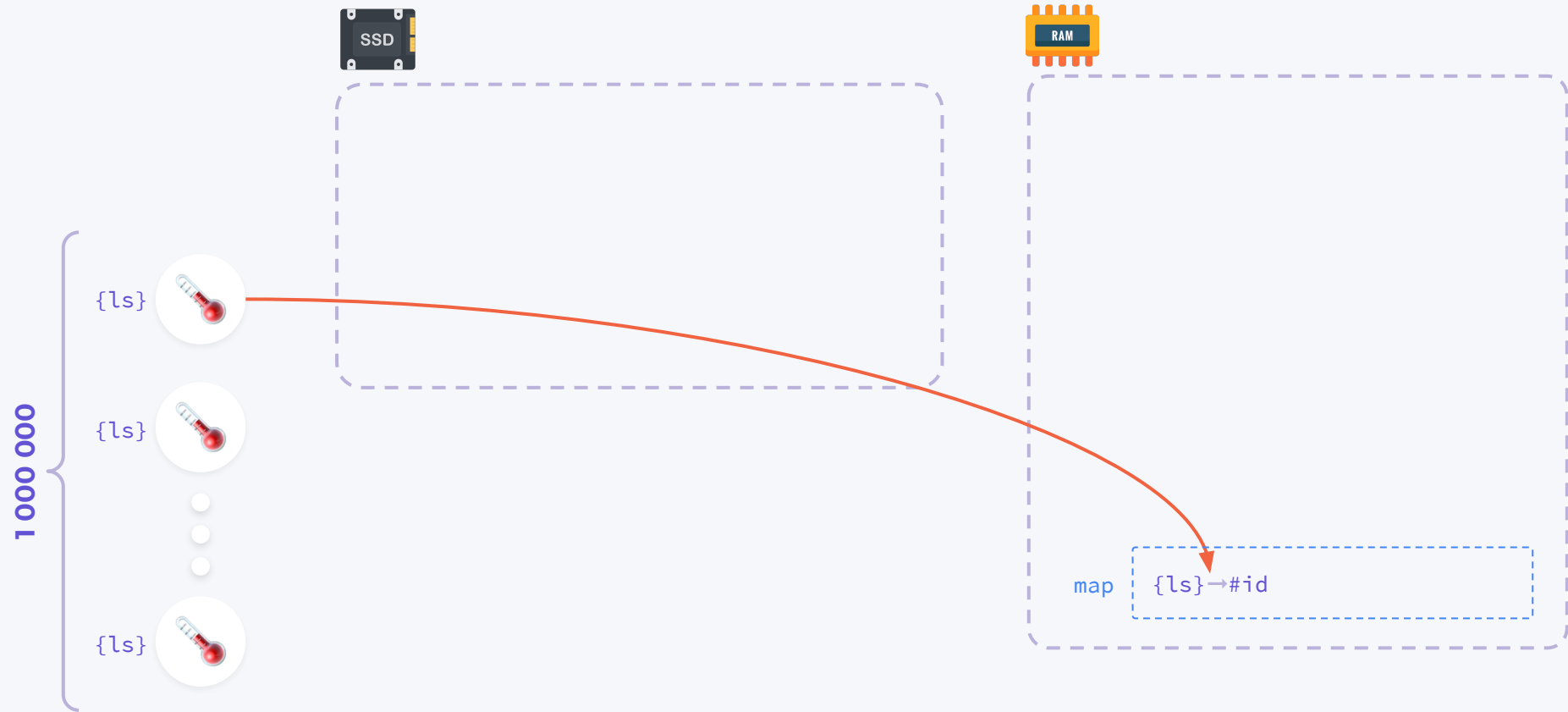
Как идентифицировать?



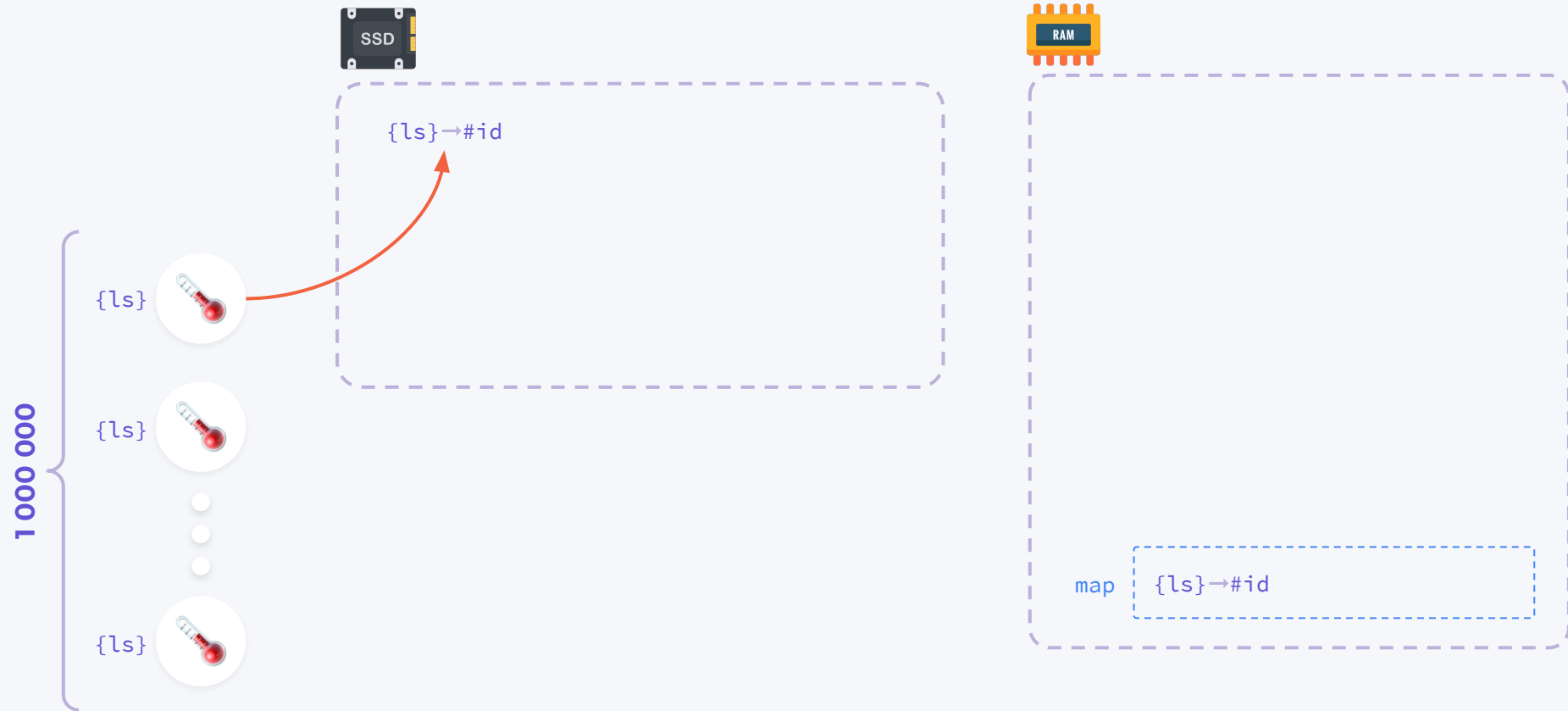
Как идентифицировать?



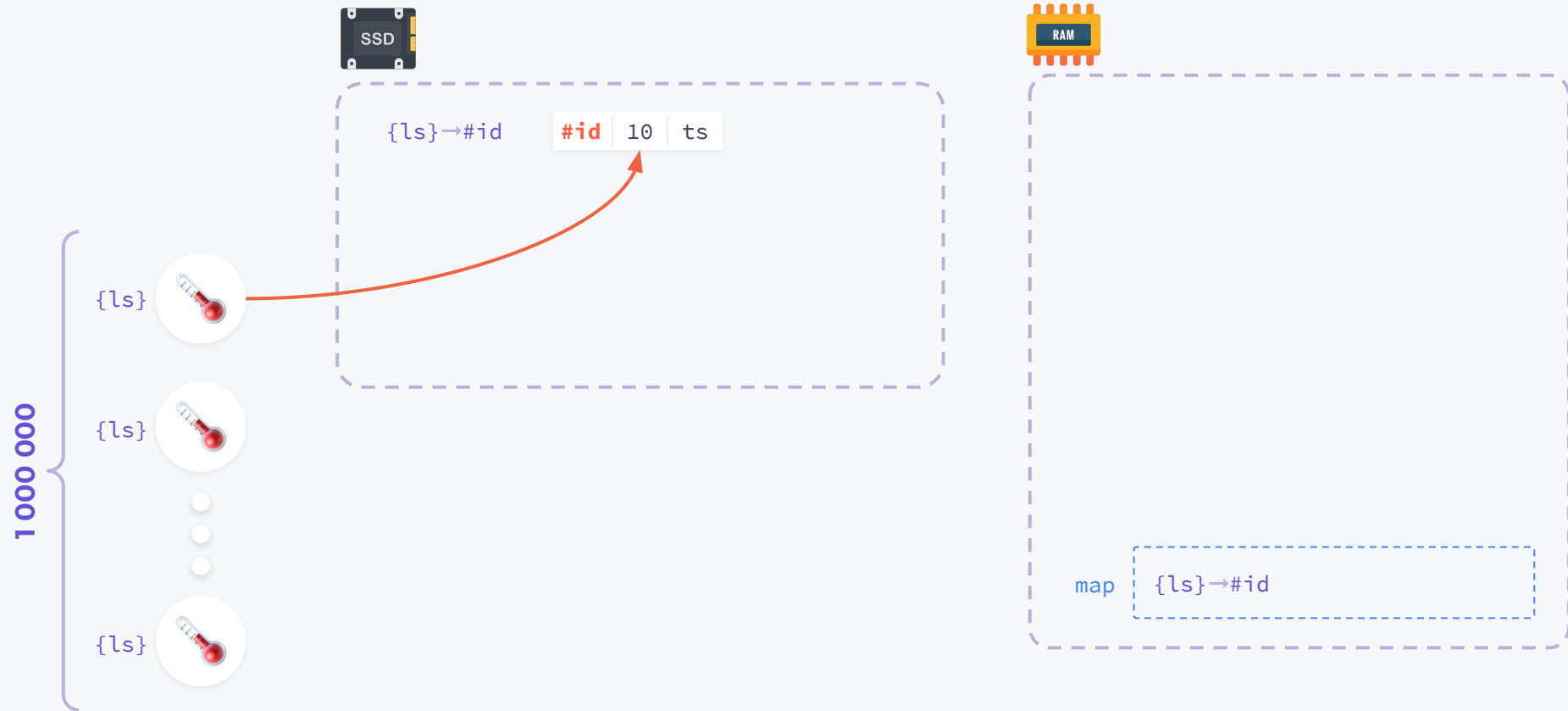
Как идентифицировать?



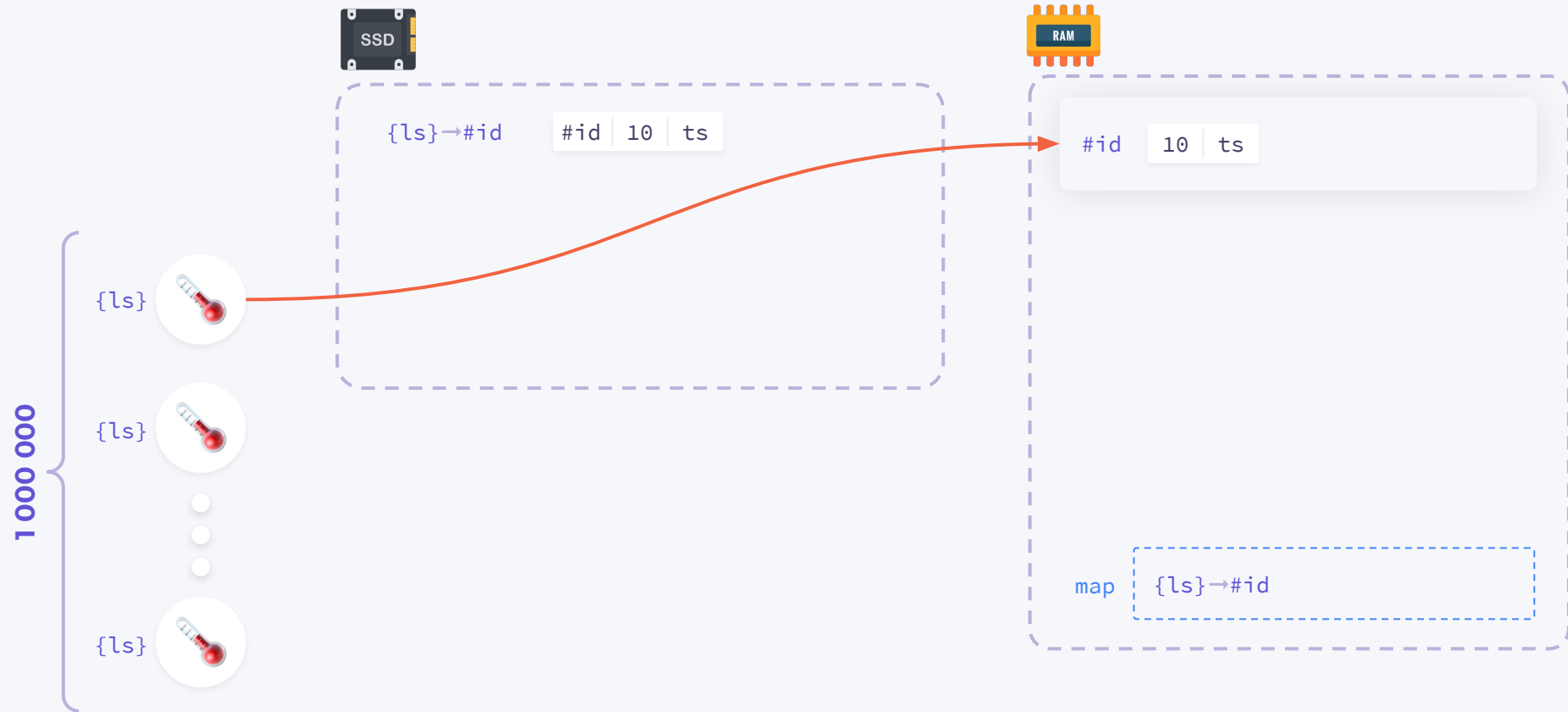
Как идентифицировать?



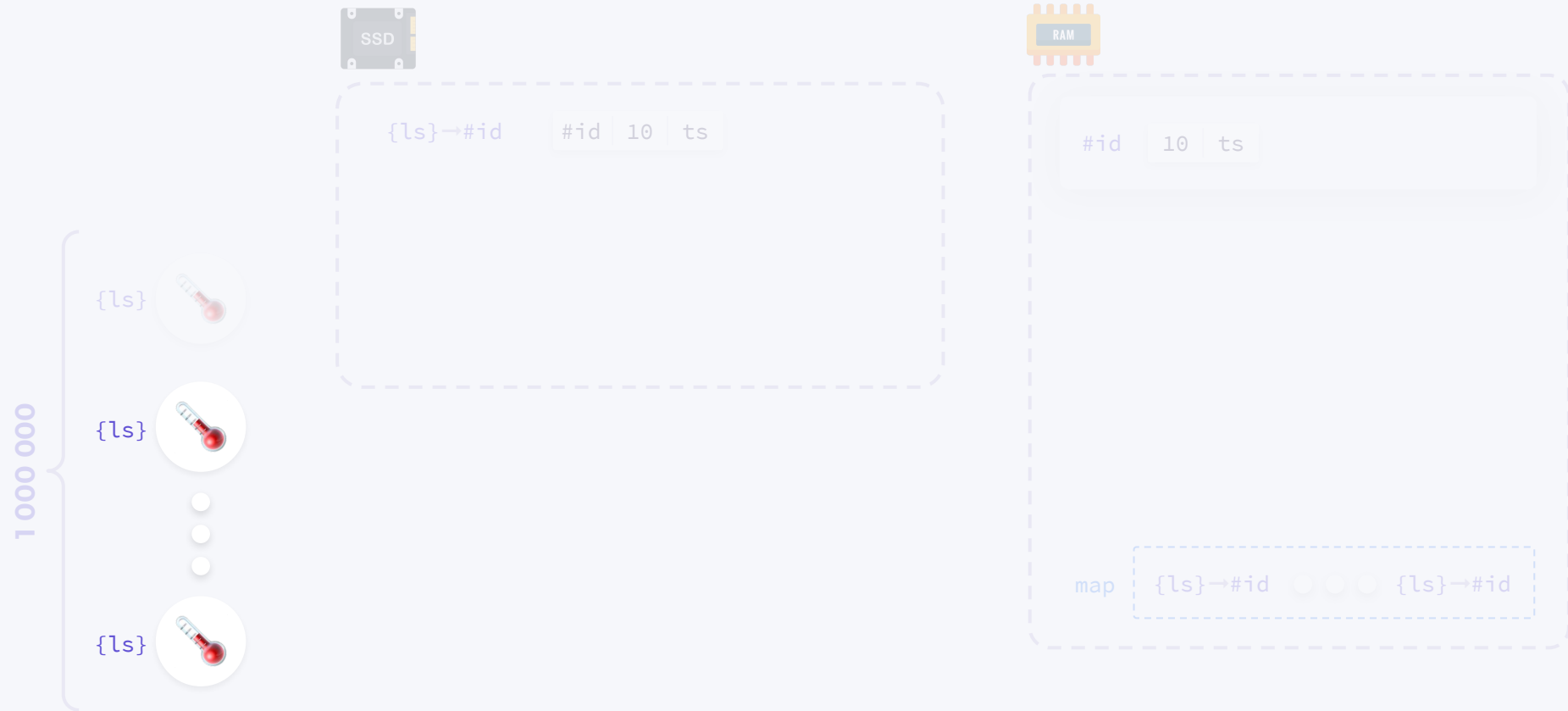
Как идентифицировать?



Как идентифицировать?



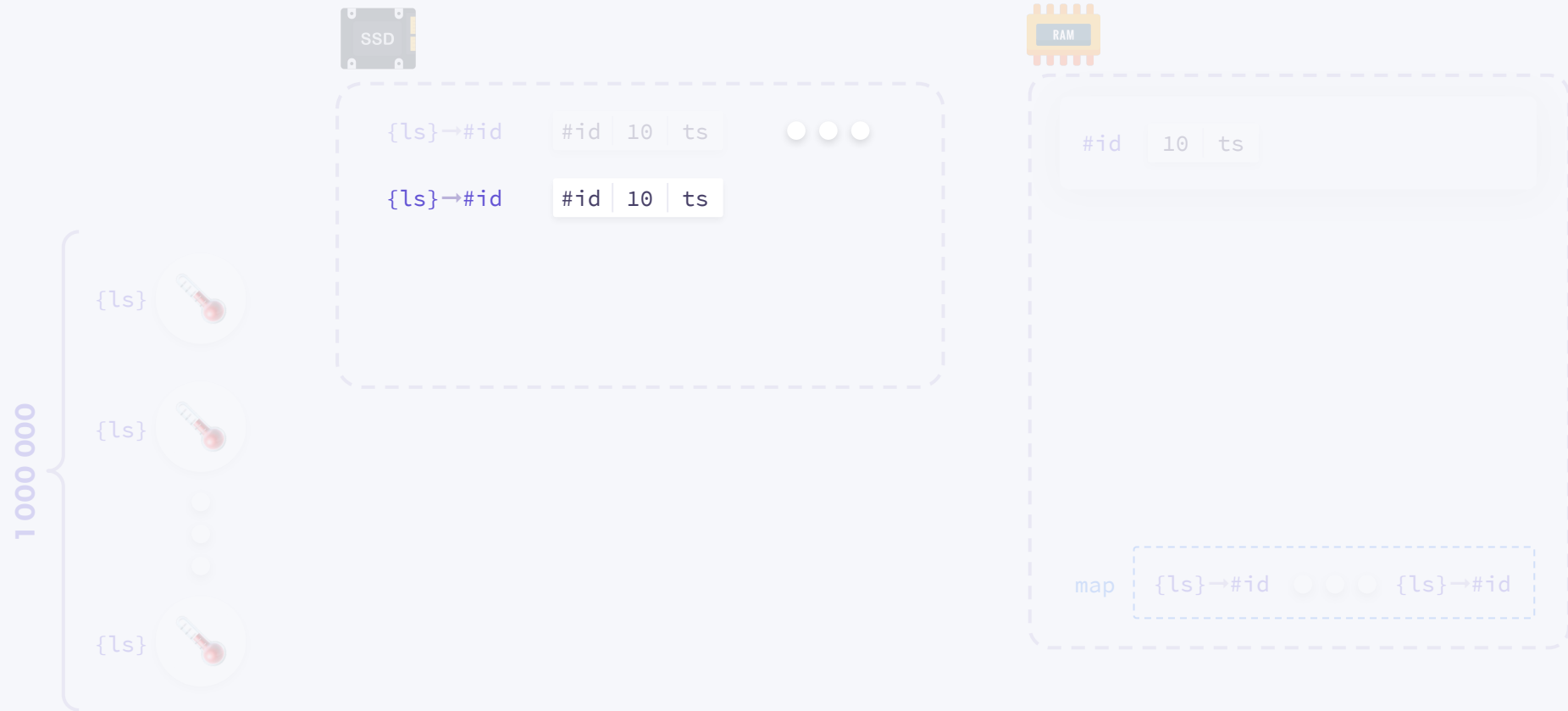
Как идентифицировать?



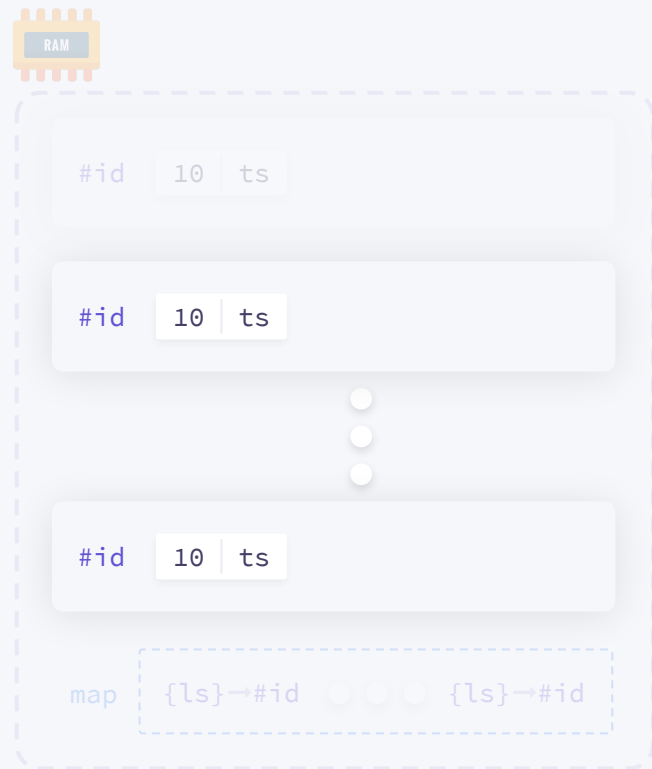
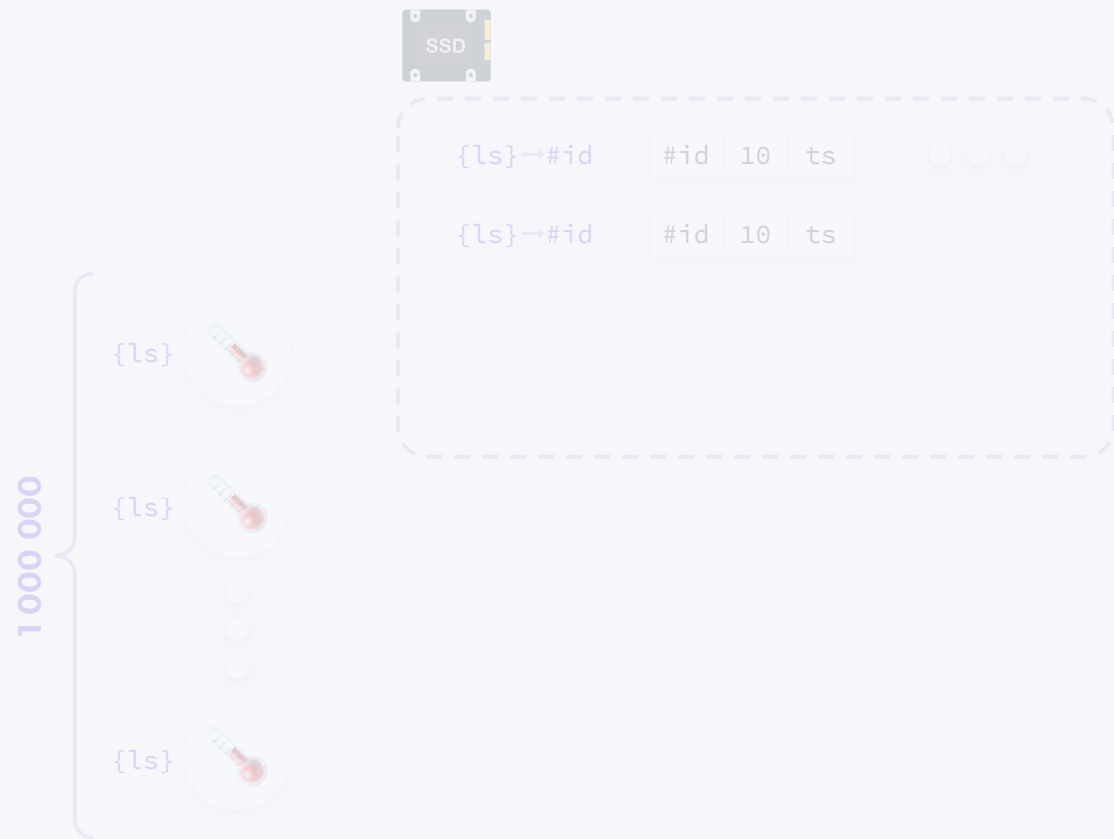
Как идентифицировать?



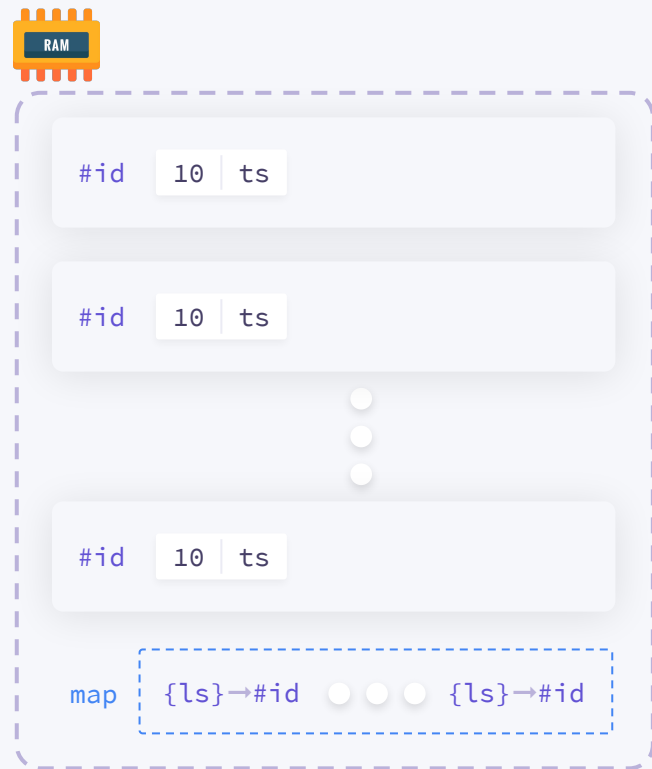
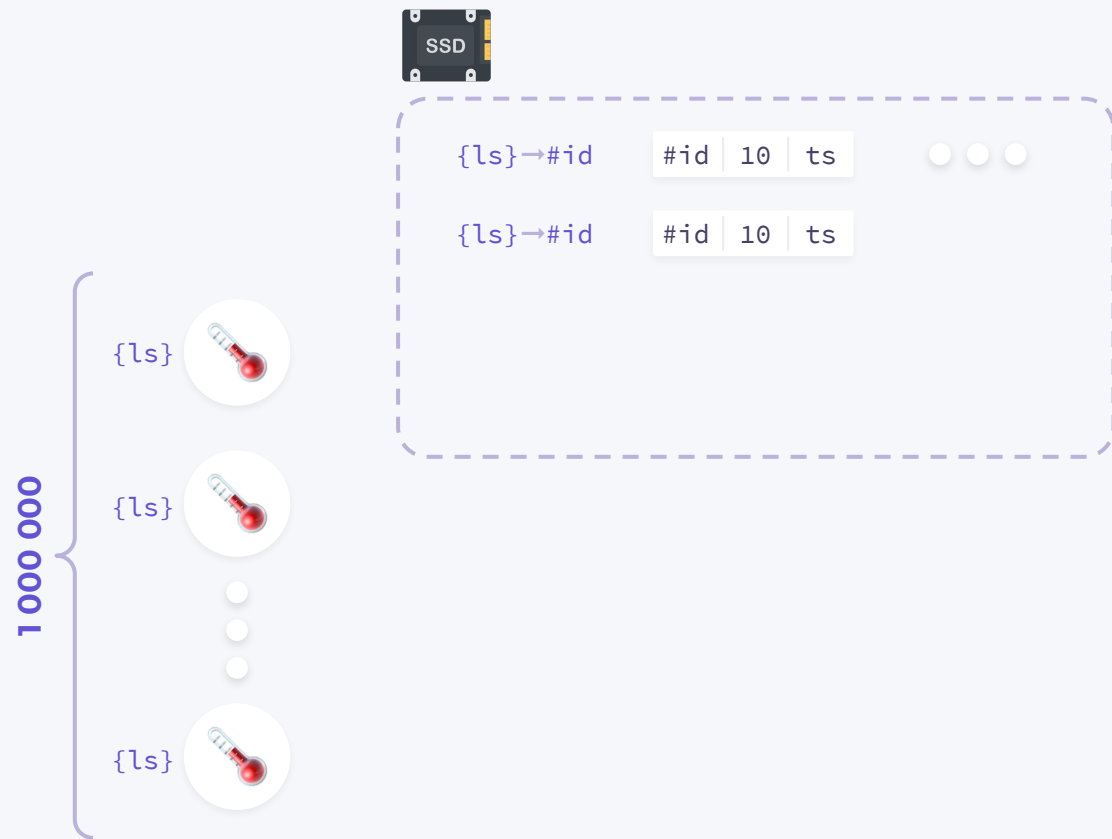
Как идентифицировать?



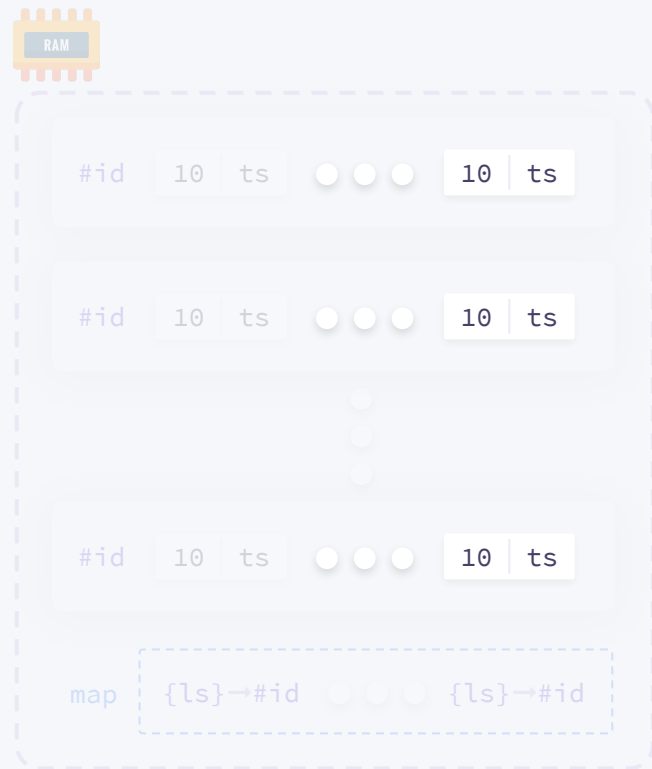
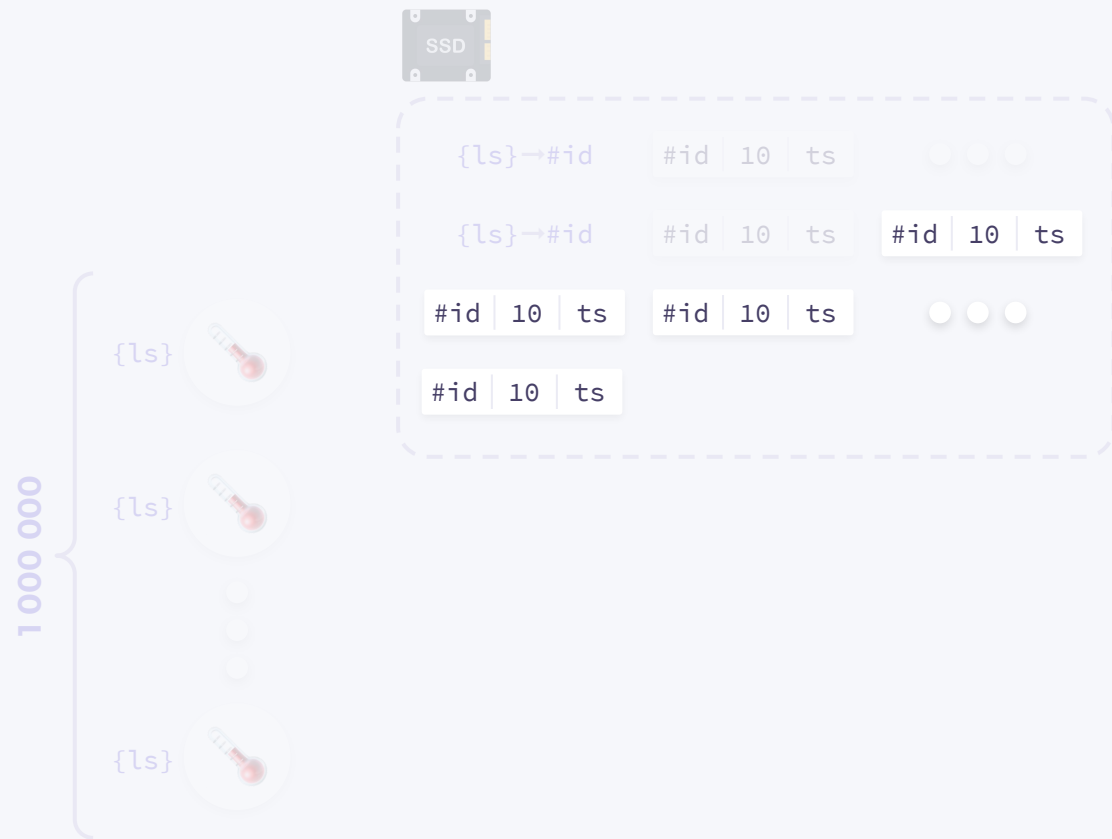
Как идентифицировать?



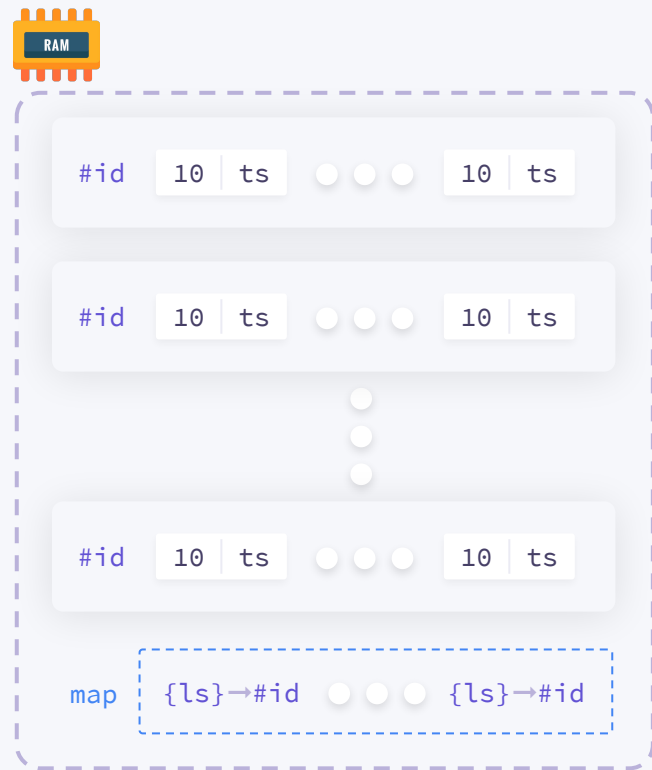
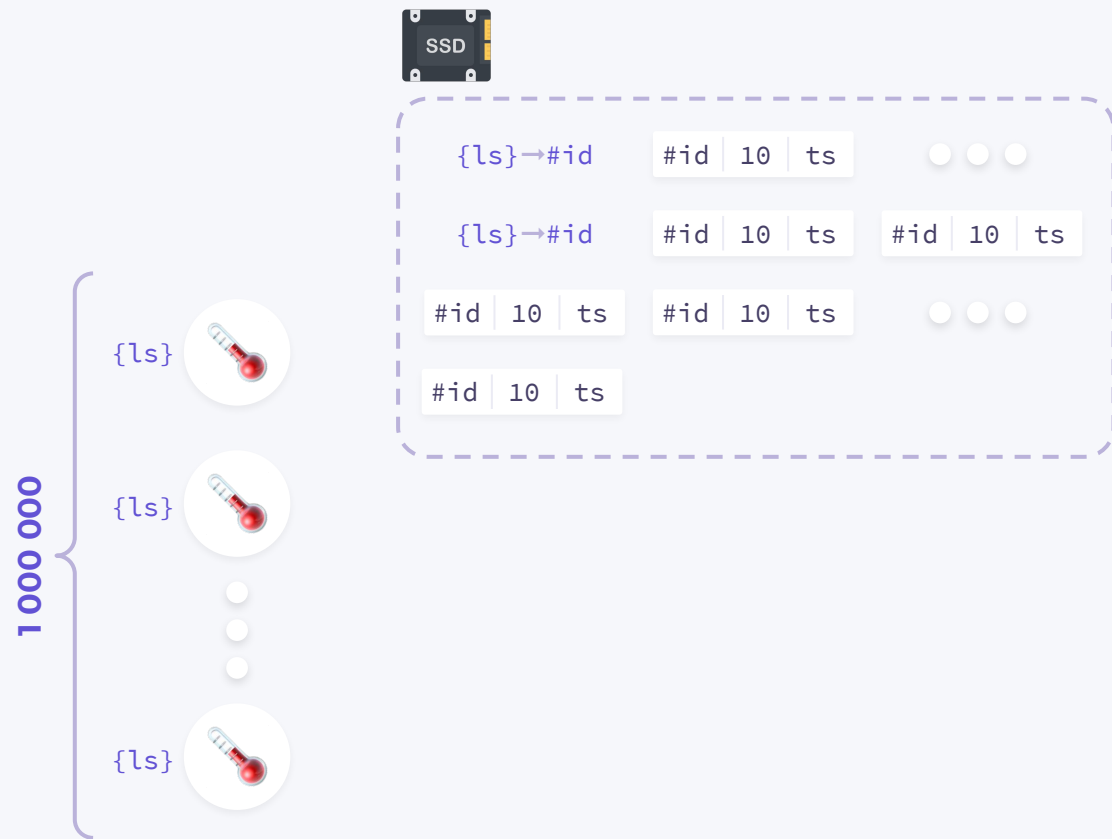
Как идентифицировать?



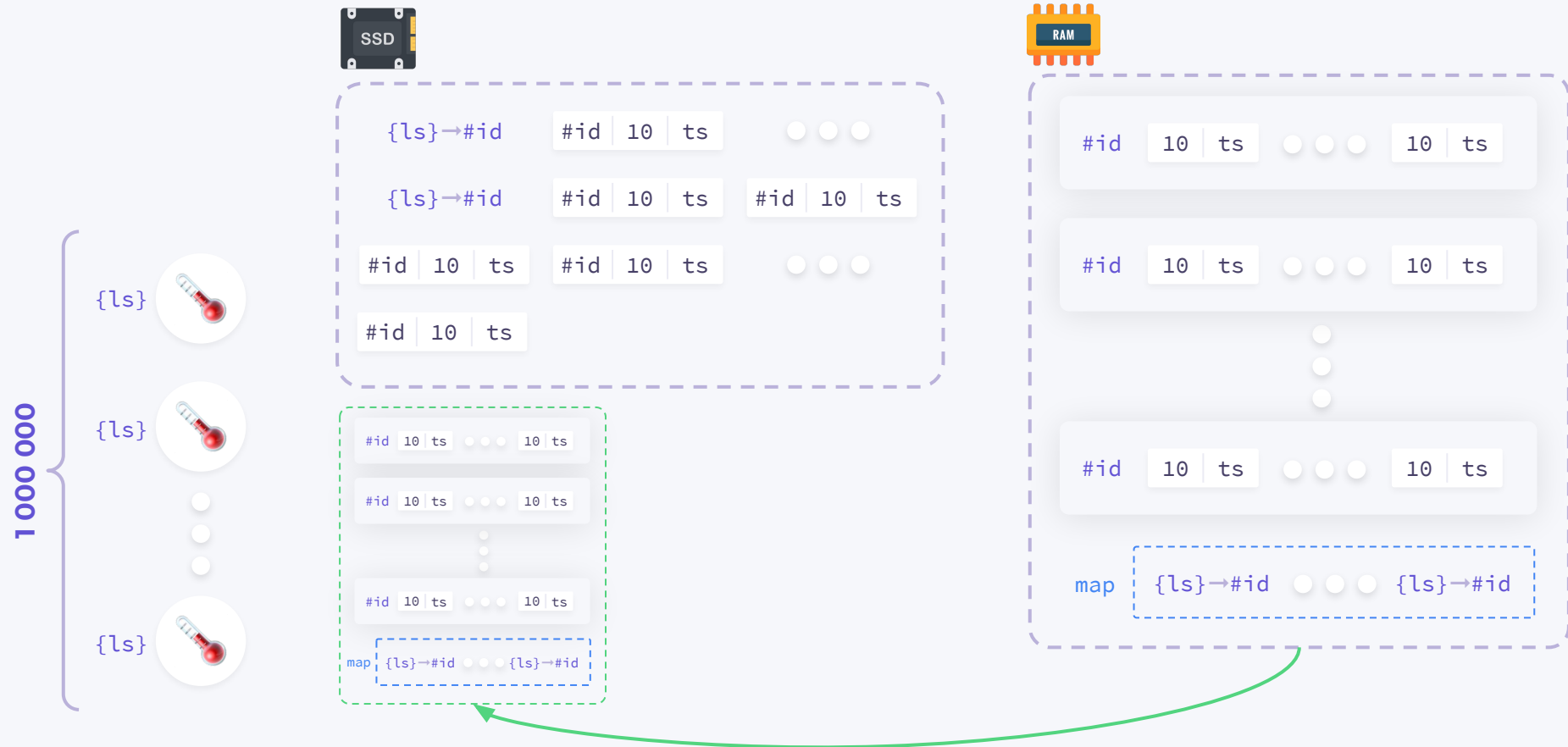
Как идентифицировать?



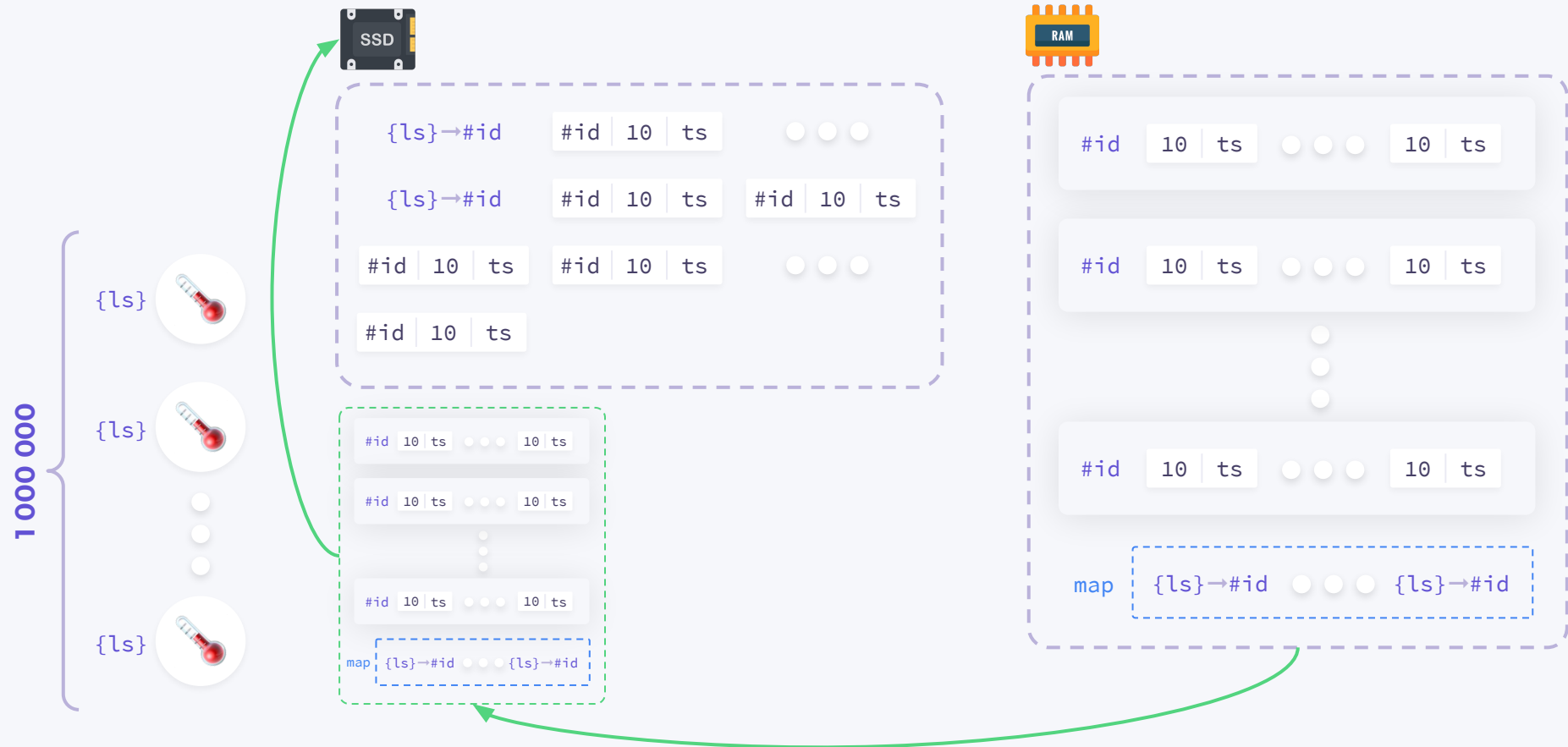
Как идентифицировать?



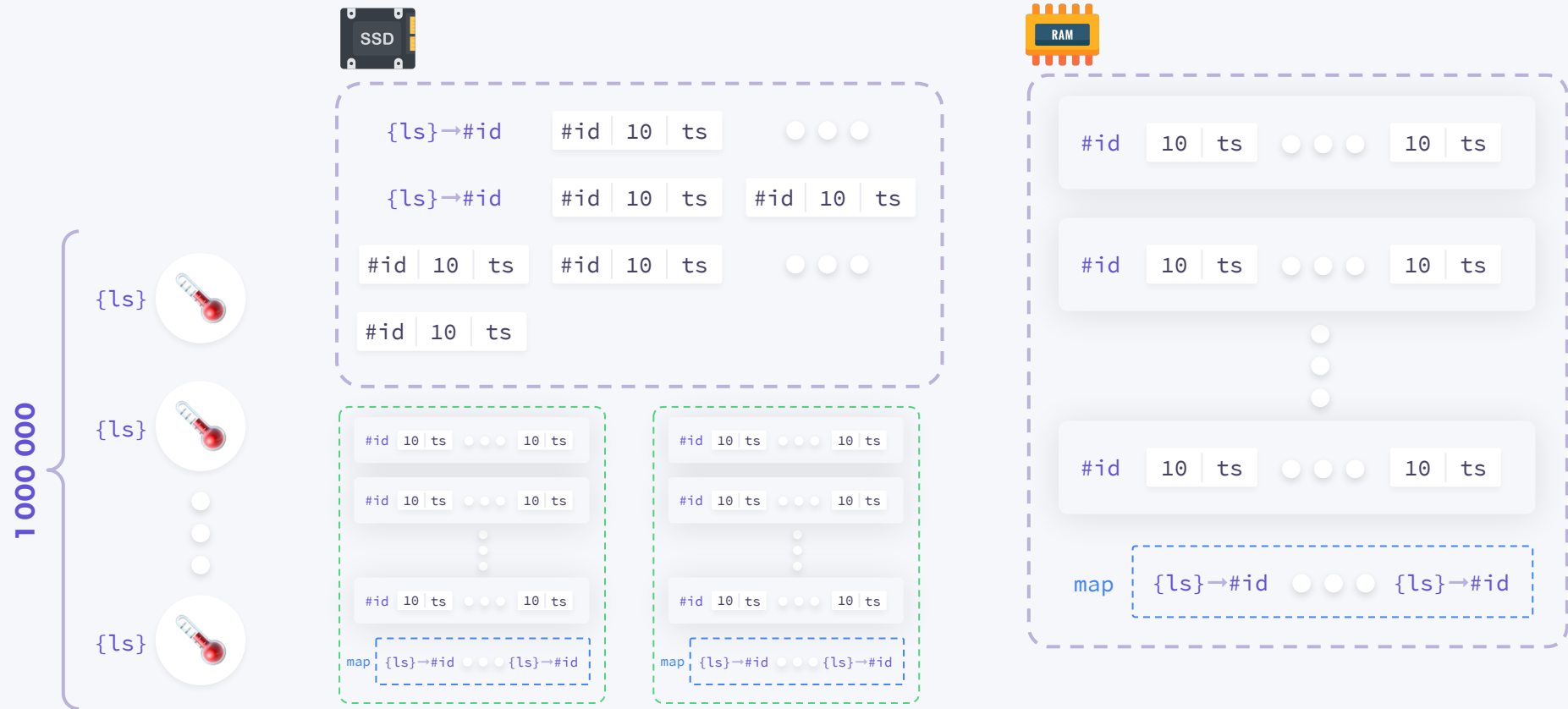
Как идентифицировать?



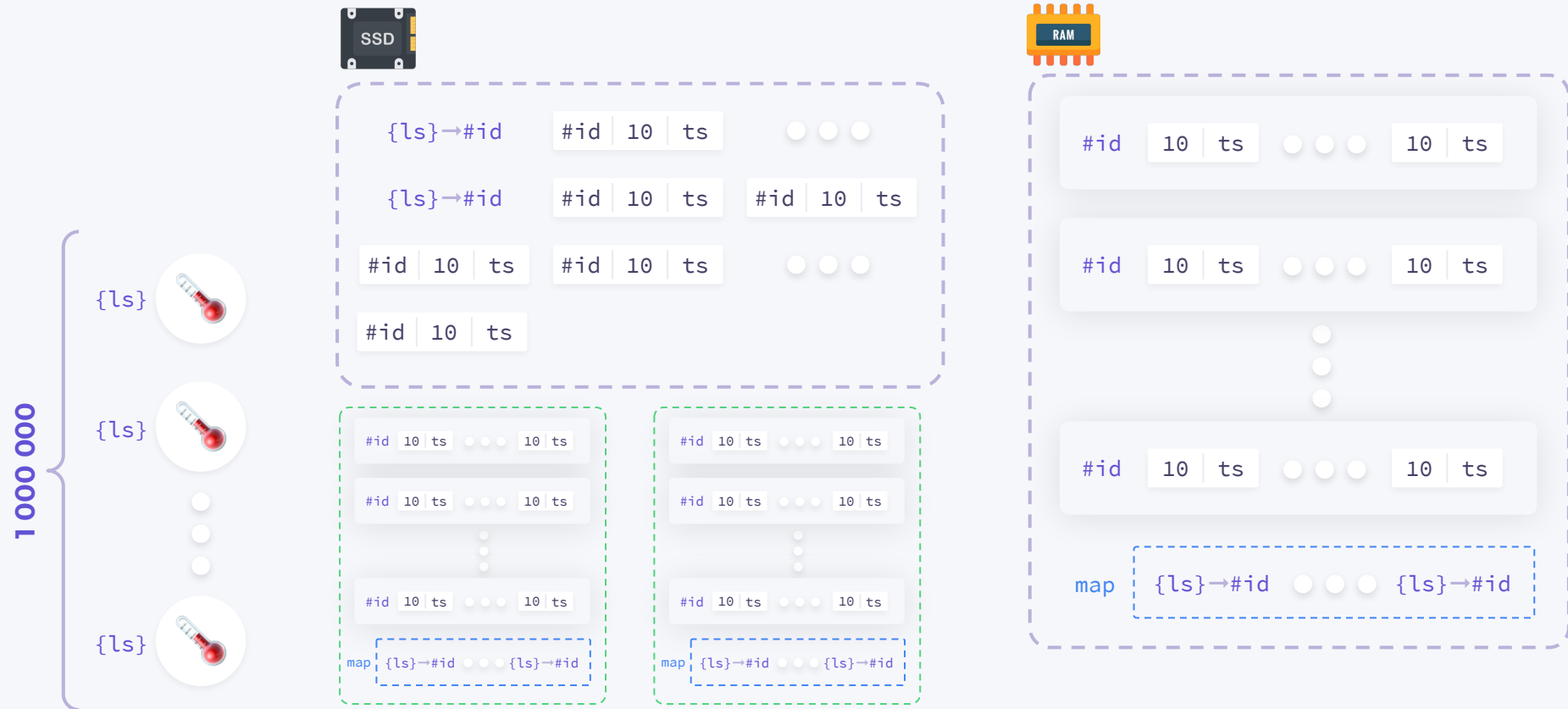
Как идентифицировать?



Как идентифицировать?



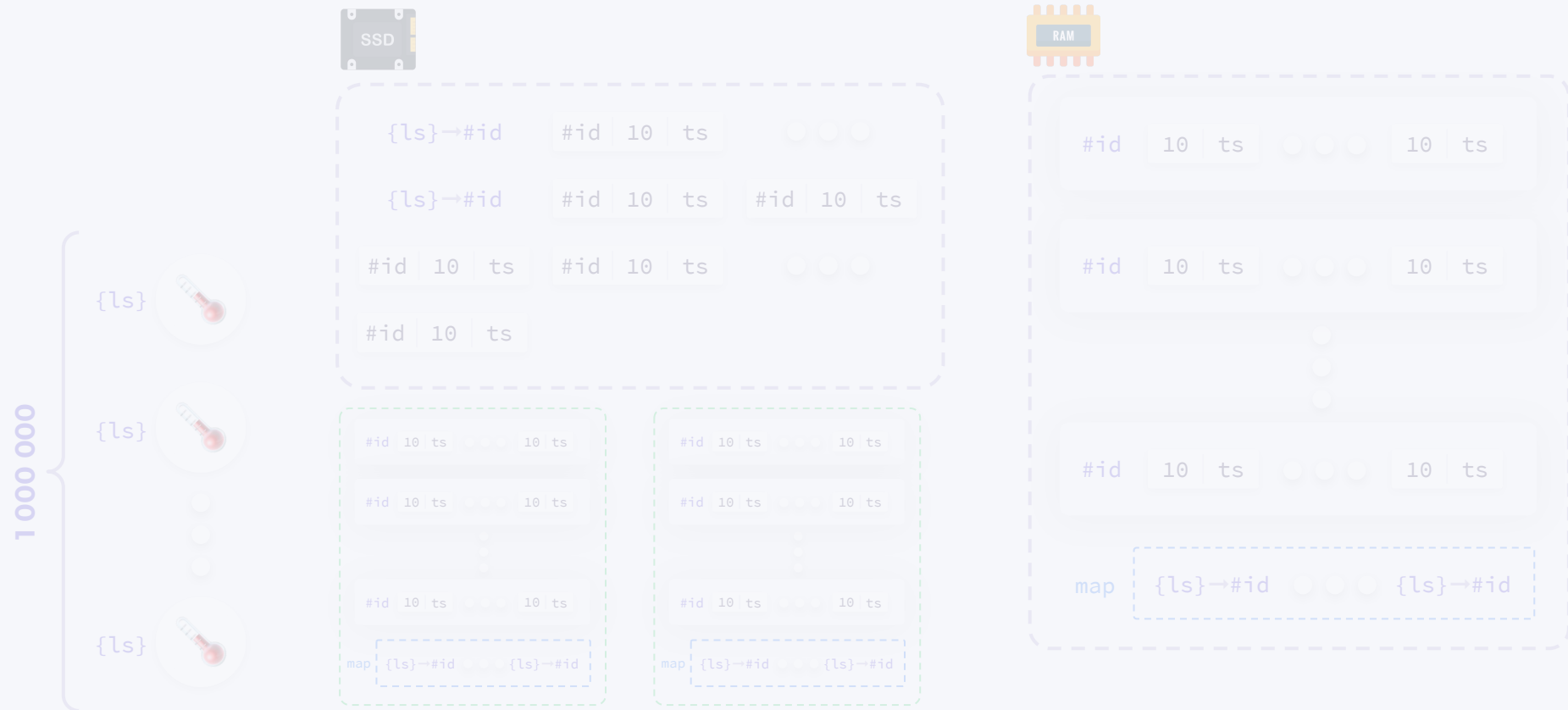
Рисуем график



Рисуем график

{ls}

{name: cpu_usage, node: curiosity, core: 4}

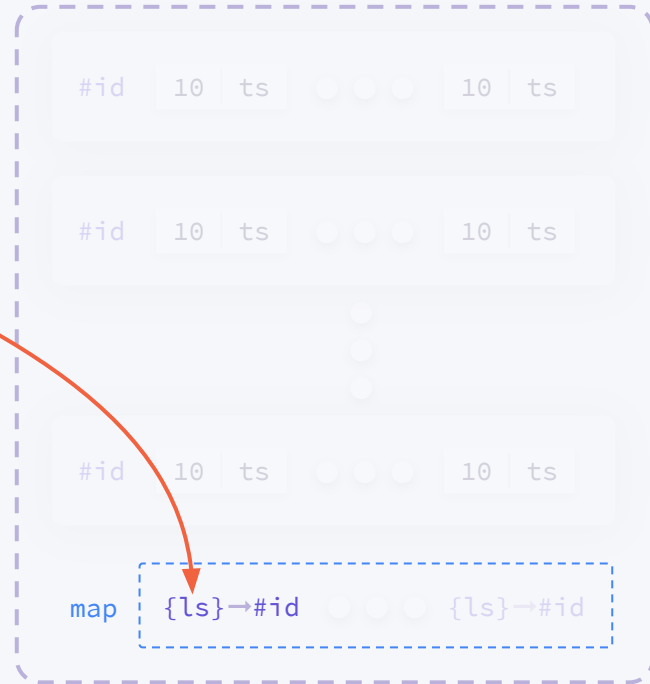
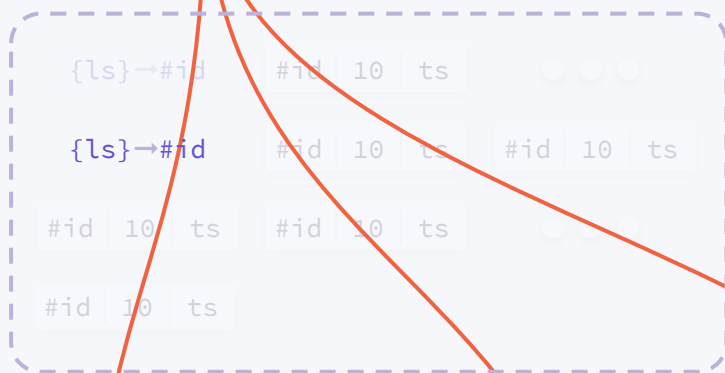


Рисуем график

{ls}

{name: cpu_usage, node: curiosity, core: 4}

1 000 000



{ls}

{ls}

{ls}

Рисуем график

{ls}

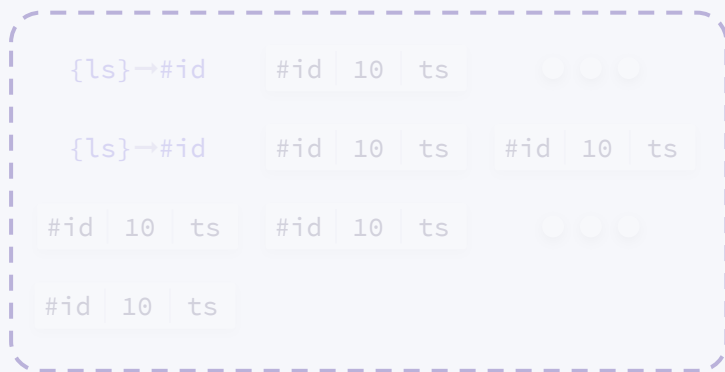
{name: cpu_usage, node: curiosity, core: 4}

1 000 000

{ls}

{ls}

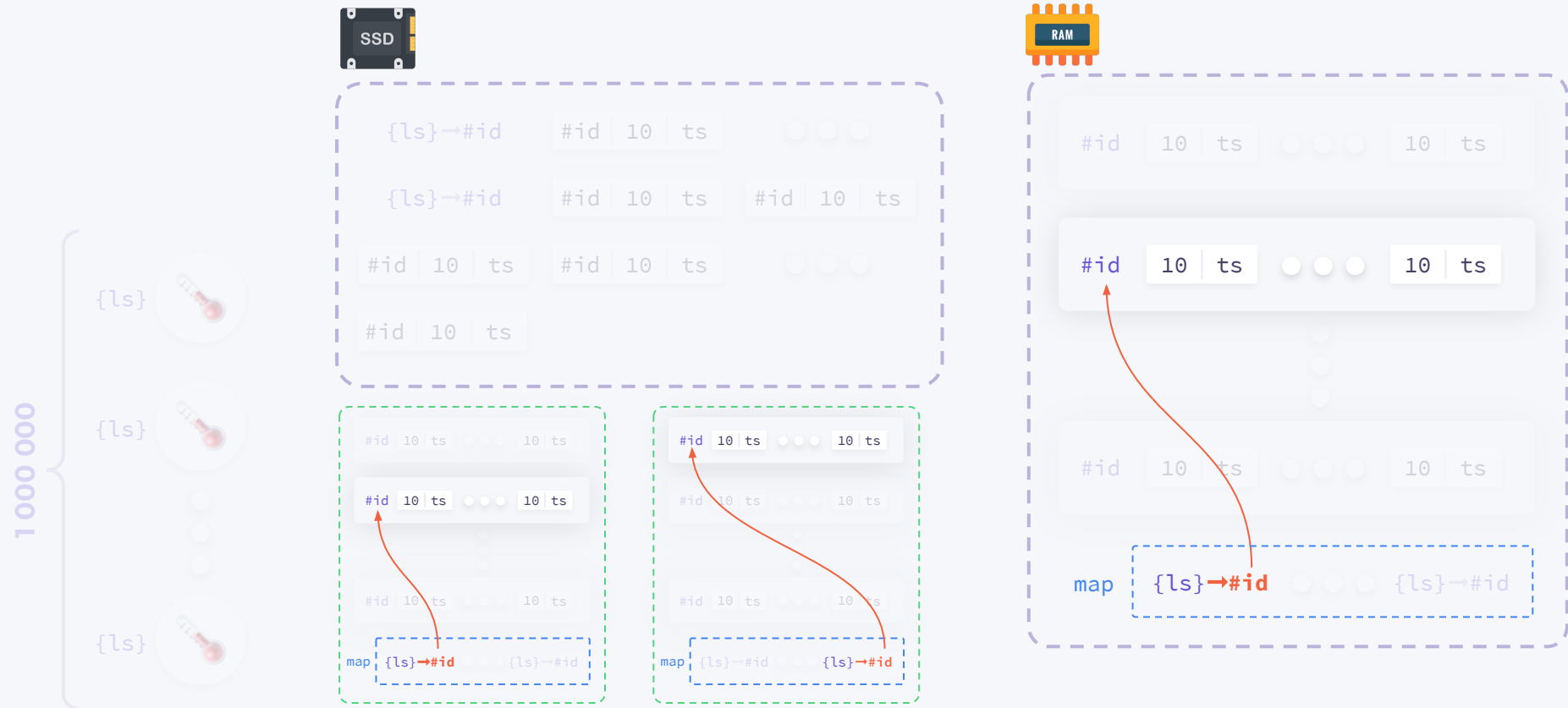
{ls}



Рисуем график

{ls}

{name: cpu_usage, node: curiosity, core: 4}



Рисуем график

{ls}

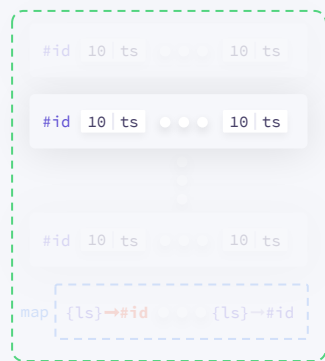
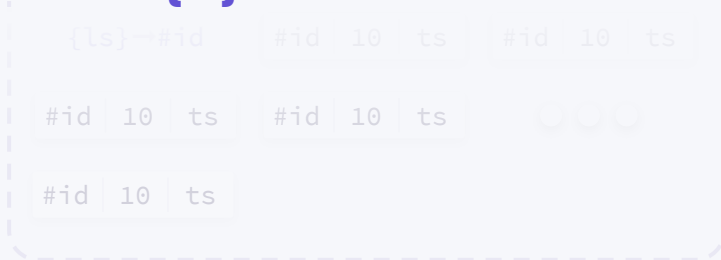
{name: cpu_usage, node: curiosity, core: 4}

1 000 000

{ls}

{ls}

{ls}

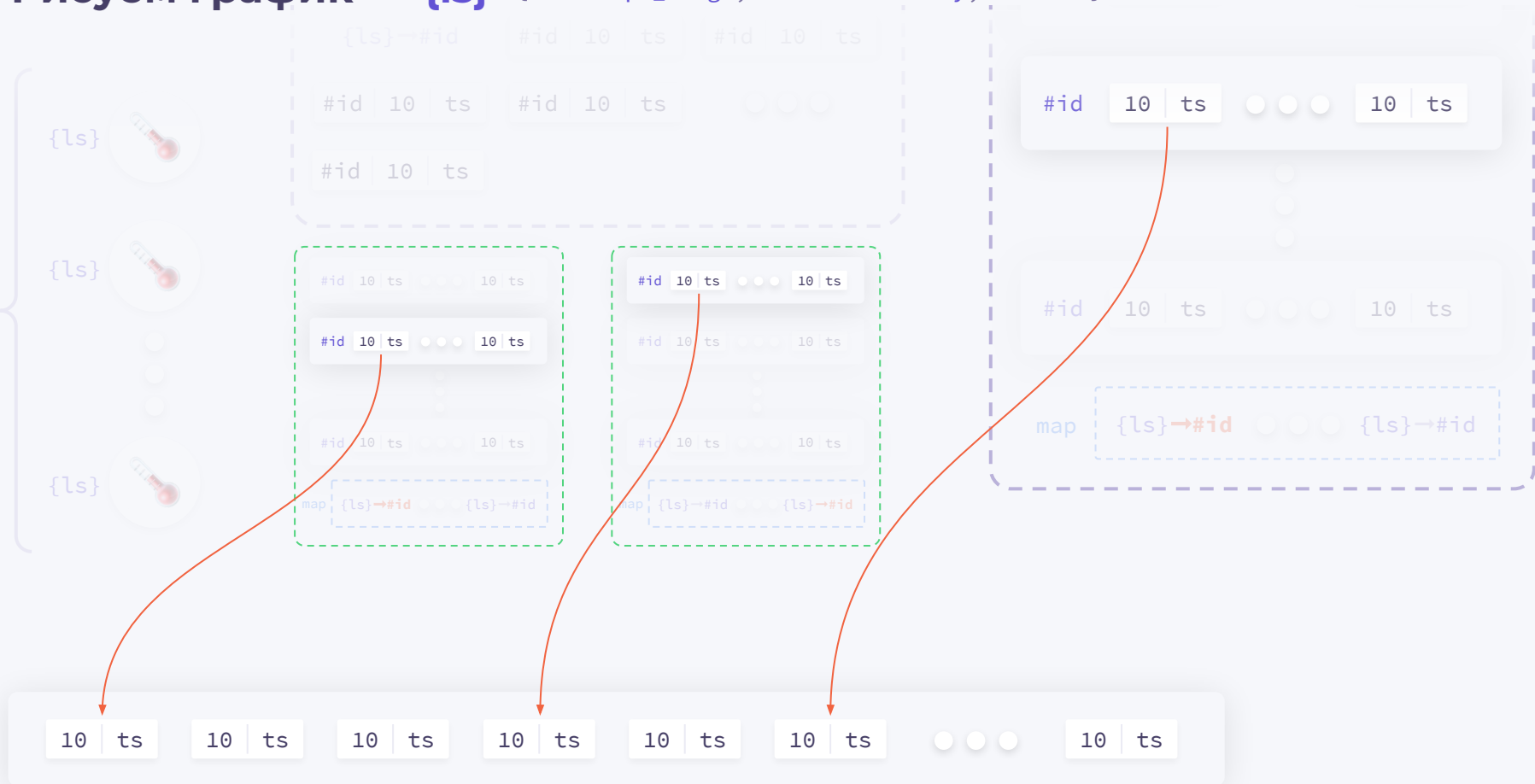


Рисуем график

{ls}

{name: cpu_usage, node: curiosity, core: 4}

1 000 000



Рисуем график

{ls}

{name: cpu_usage, node: curiosity, core: 4}

1 000 000

{ls}

{ls}

{ls}

{ls}→#id #id 10 ts #id 10 ts

#id 10 ts

#id 10 ts

#id 10 ts

#id 10 ts 10 ts

#id 10 ts 10 ts

#id 10 ts 10 ts

map {ls}→#id {ls}→#id

#id 10 ts 10 ts

#id 10 ts 10 ts

#id 10 ts 10 ts

map {ls}→#id {ls}→#id

#id 10 ts

10 ts

#id 10 ts

10 ts

map

{ls}→#id

{ls}→#id

10 ts

10 ts

10 ts

10 ts

10 ts

10 ts

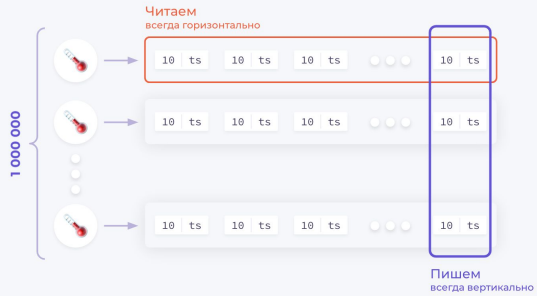
10 ts



В итоге

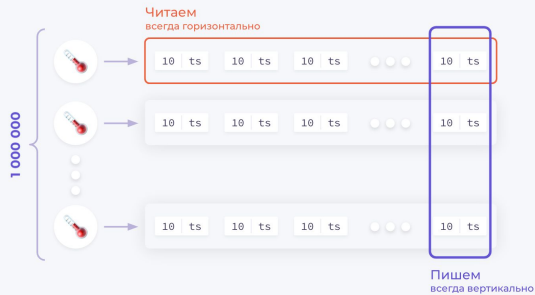
В итоге

“Переворот” на 90°



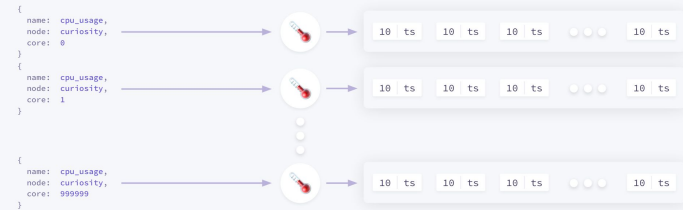
В итоге

“Переворот” на 90°



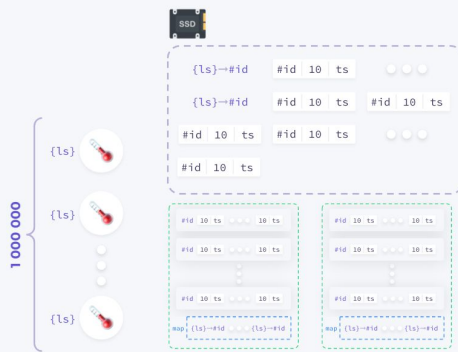
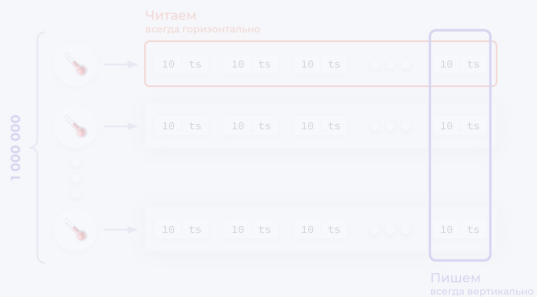
LabelSet'ы

{ls} → #t_id → Временной ряд

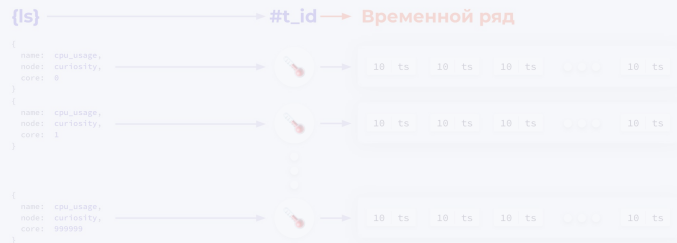


В итоге

“Переворот” на 90°

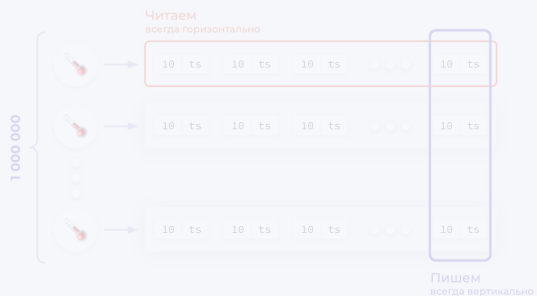


LabelSet'ы

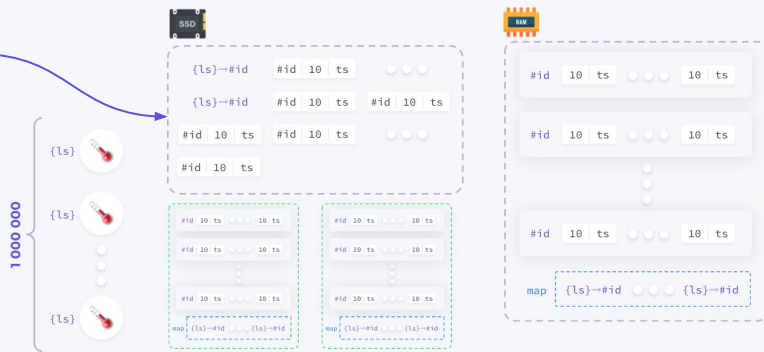


В итоге

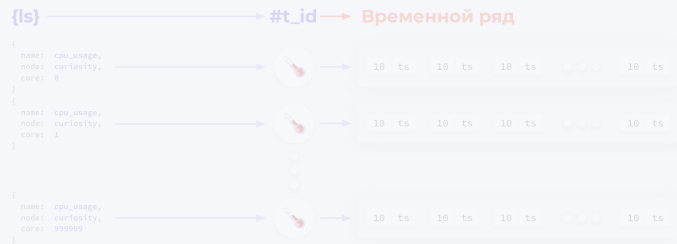
“Переворот” на 90°



Журнал

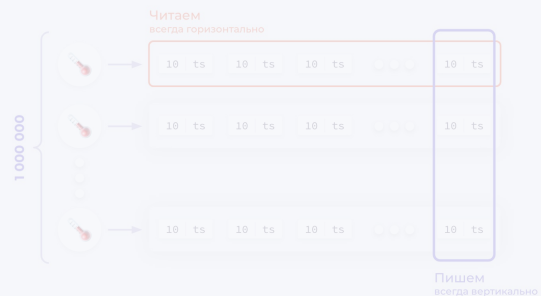


LabelSet'ы

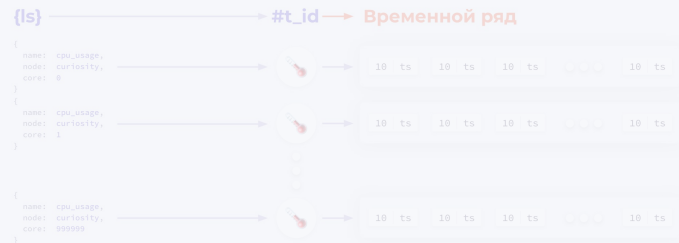


В итоге

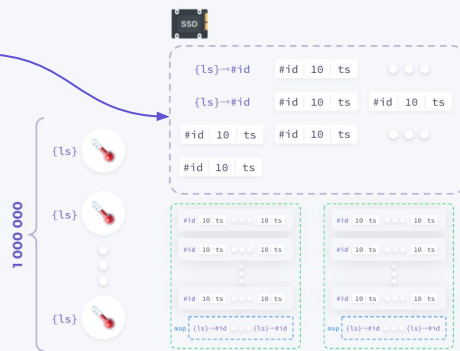
“Переворот” на 90°



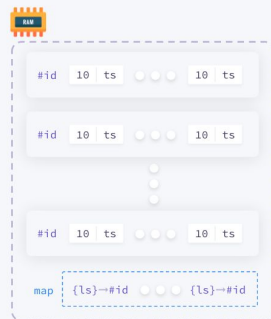
LabelSet'ы



Журнал



“Активный” блок

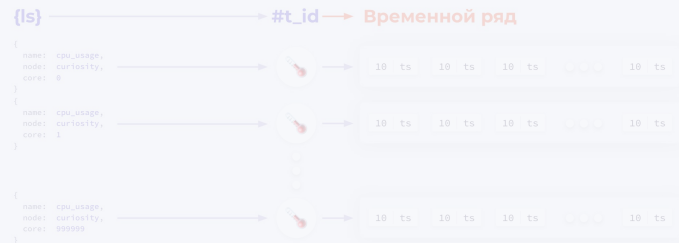


В итоге

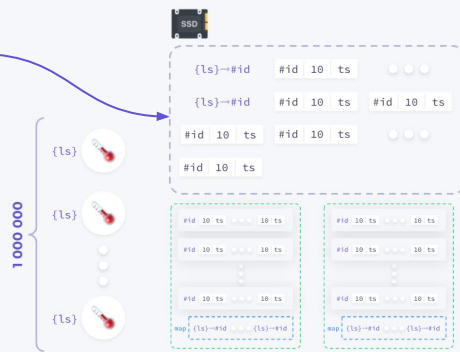
“Переворот” на 90°



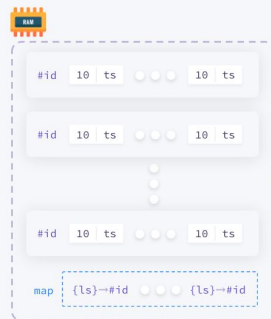
LabelSet'ы



Журнал



“Активный” блок

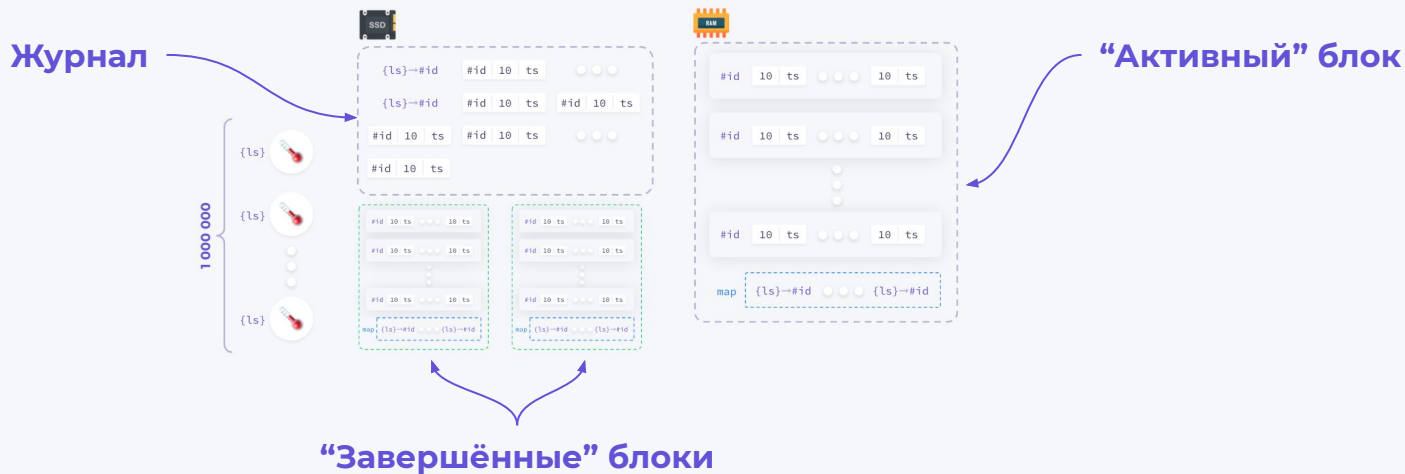


“Завершённые” блоки

В итоге

Log Structured Merge Tree

LSMT



Log Structured Merge Tree

TSDB на базе LSMT



Log Structured Merge Tree

TSDB на базе LSMT



Базы данных на базе LSMT



Google Cloud Bigtable



Clickhouse



Log Structured Merge Tree

TSDB на базе LSMT



Базы данных на базе LSMT



Google Cloud Bigtable



Clickhouse



Вариация на LSMT реализована в



Okmeter – собственный путь

Okmeter – собственный путь



Начало разработки – 2012 год



Okmeter – собственный путь



Начало разработки – 2012 год

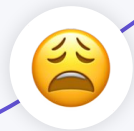


TSDB

Okmeter – собственный путь

okmeter 

Начало разработки – 2012 год



TSDB

Только первые шаги



Okmeter – собственный путь



Okmeter – собственный путь

Метрики

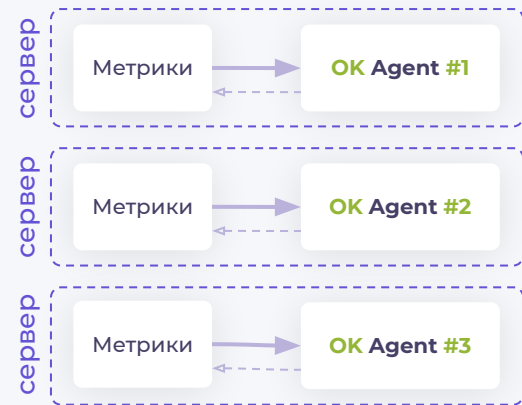
Метрики

Метрики

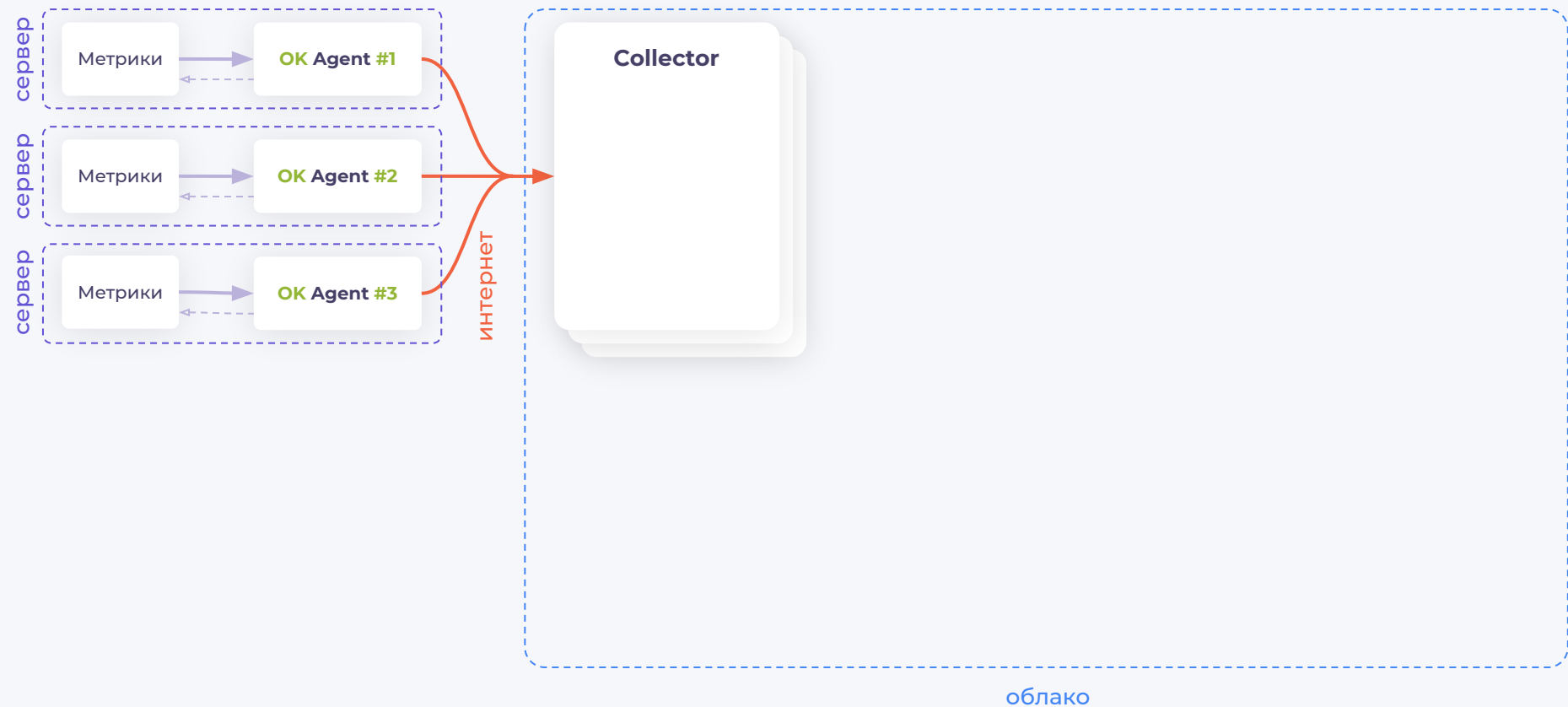
Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



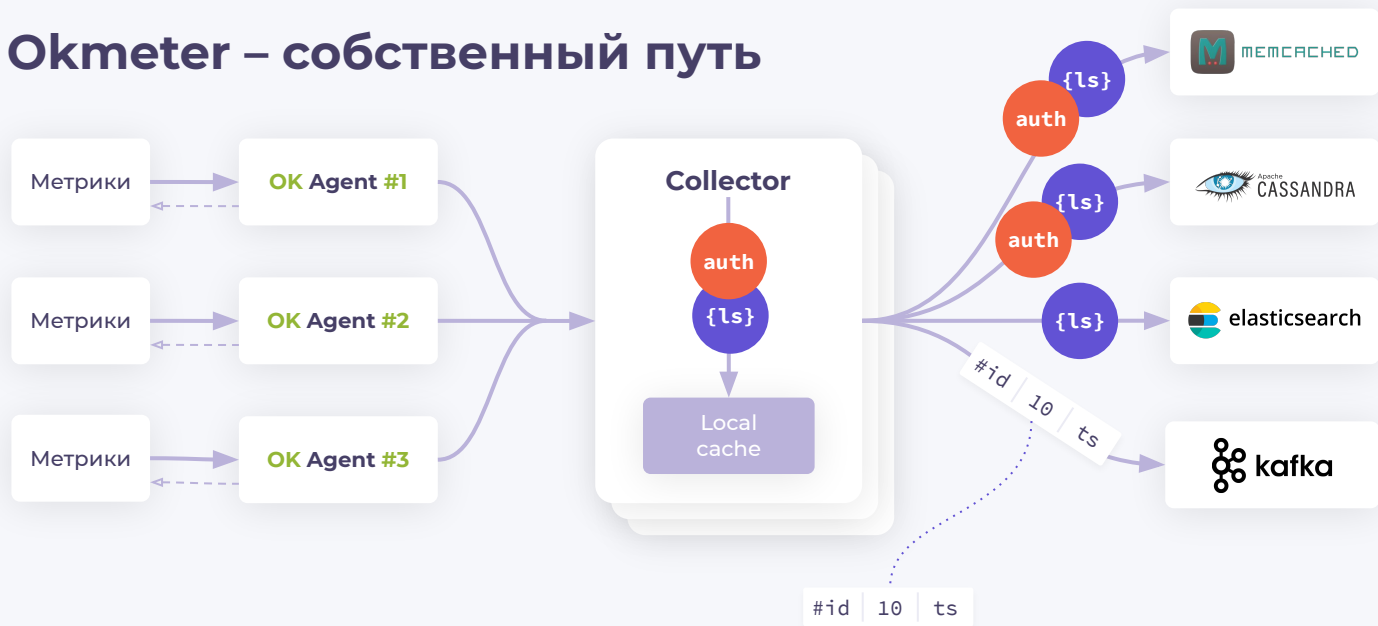
Okmeter – собственный путь



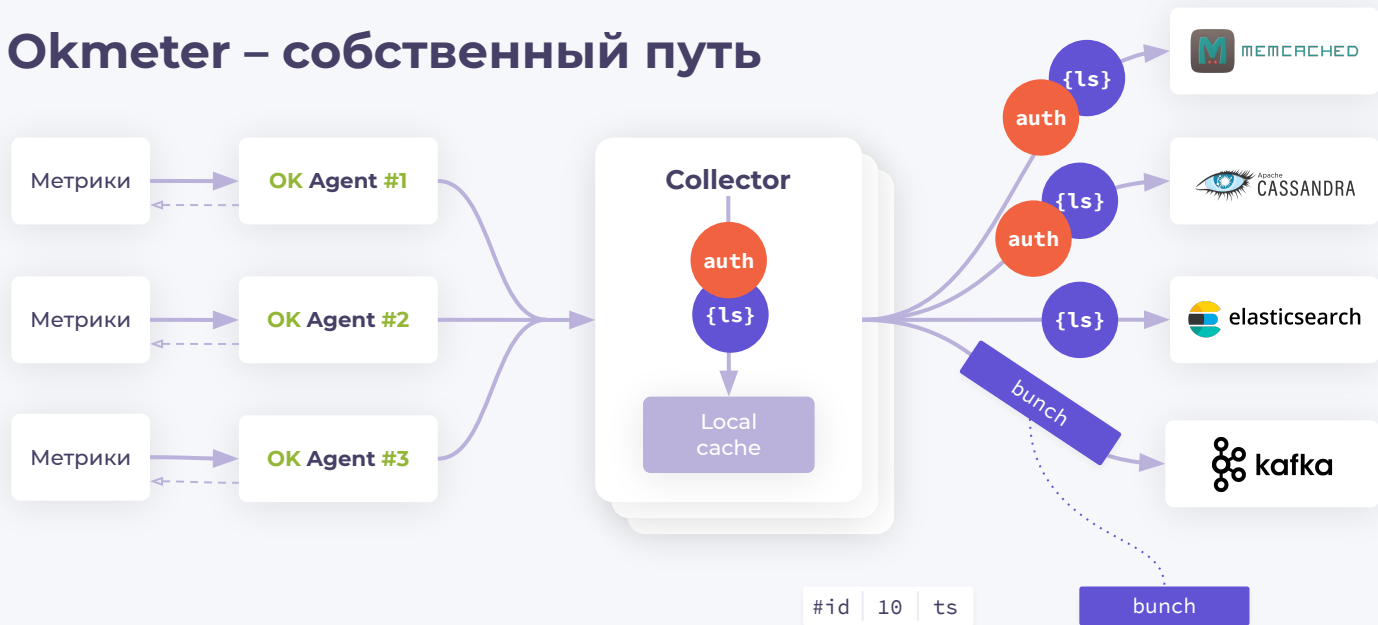
Okmeter – собственный путь



Okmeter – собственный путь



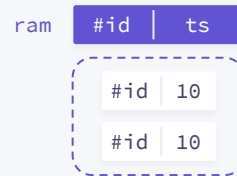
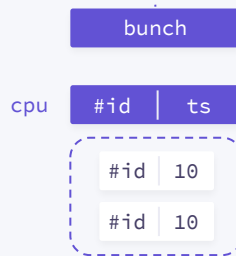
Okmeter – собственный путь



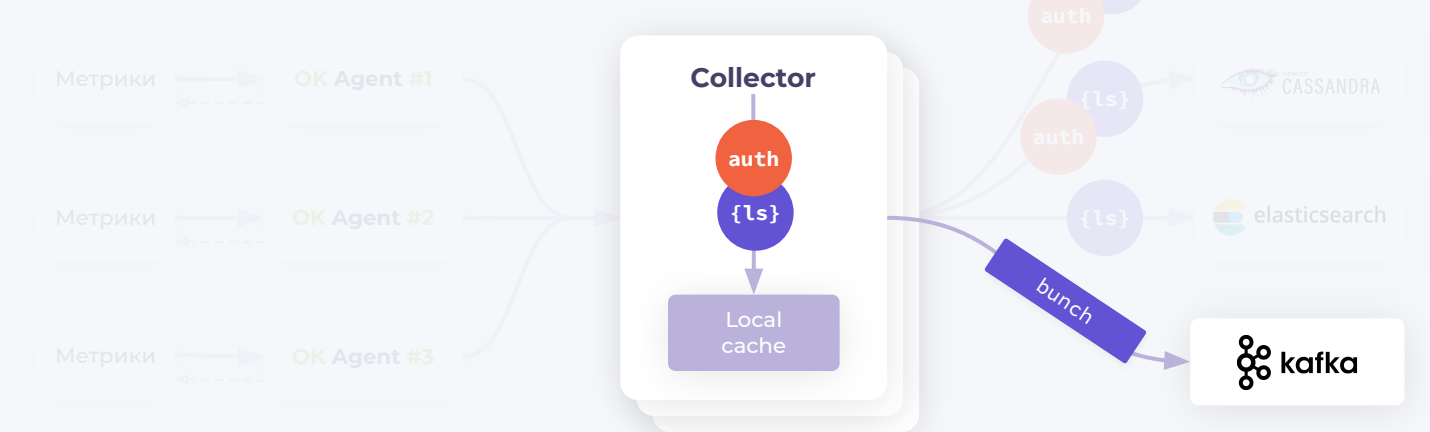
Okmeter – собственный путь



#id	10	ts

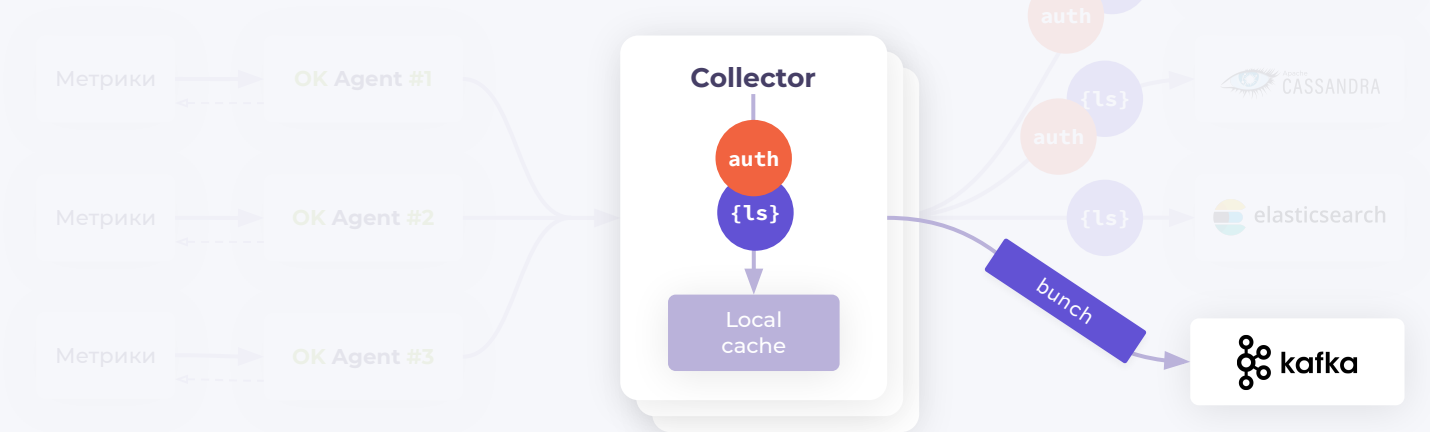


Okmeter – собственный путь



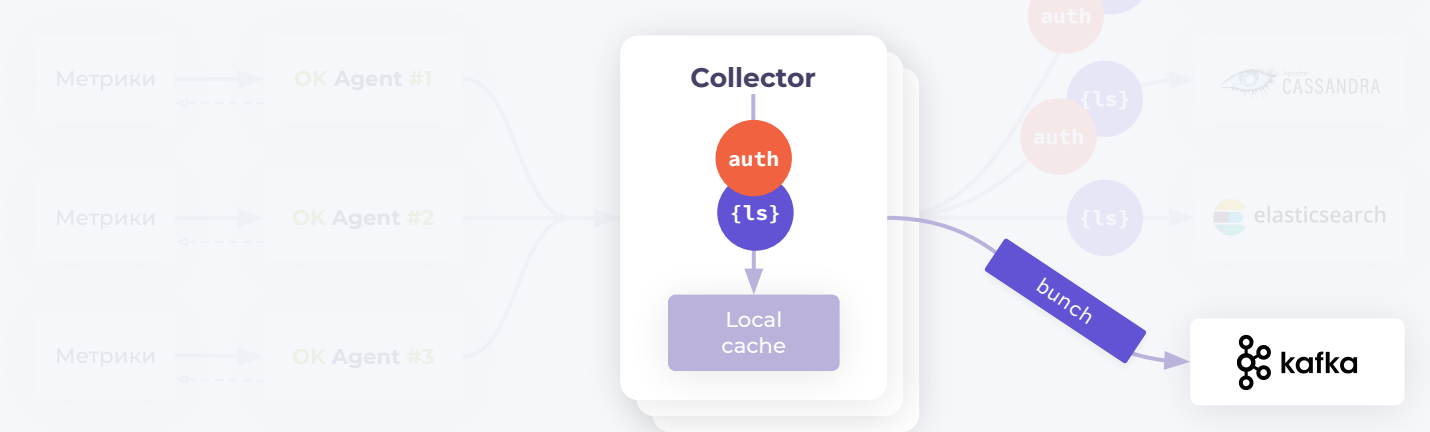
- Идея с bunch классная

Okmeter – собственный путь



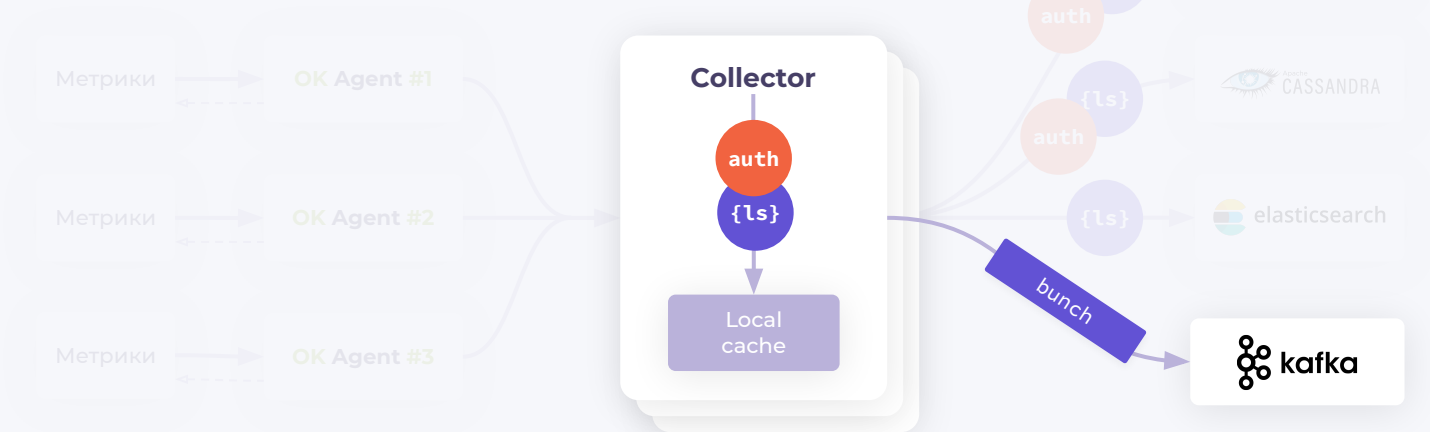
- Идея с bunch классная
- Позволяет сильно ускорить и запись и чтение

Okmeter – собственный путь



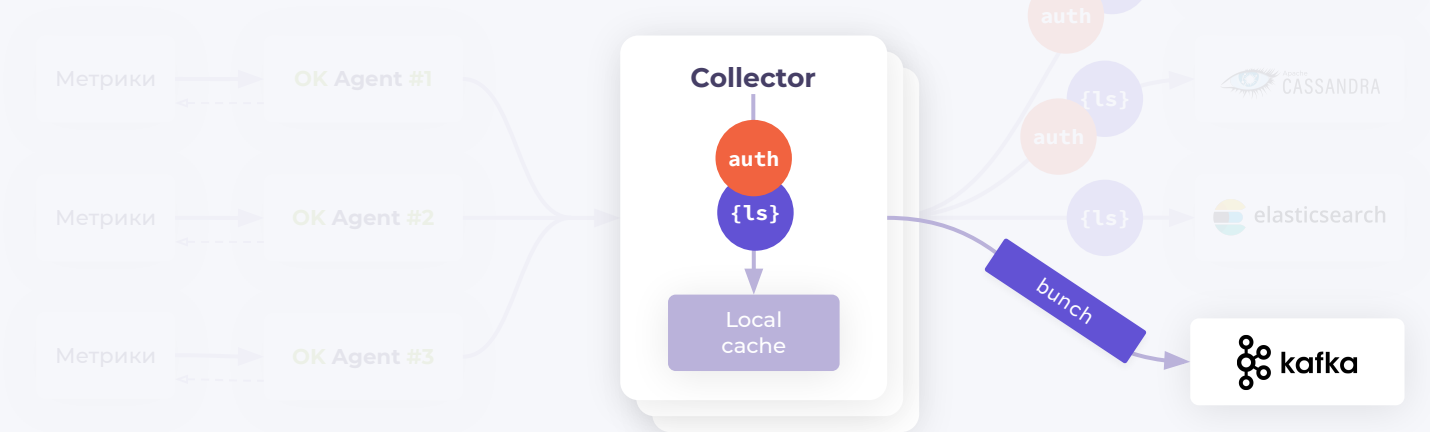
- Идея с bunch классная
- Позволяет сильно ускорить и запись и чтение. Но!

Okmeter – собственный путь



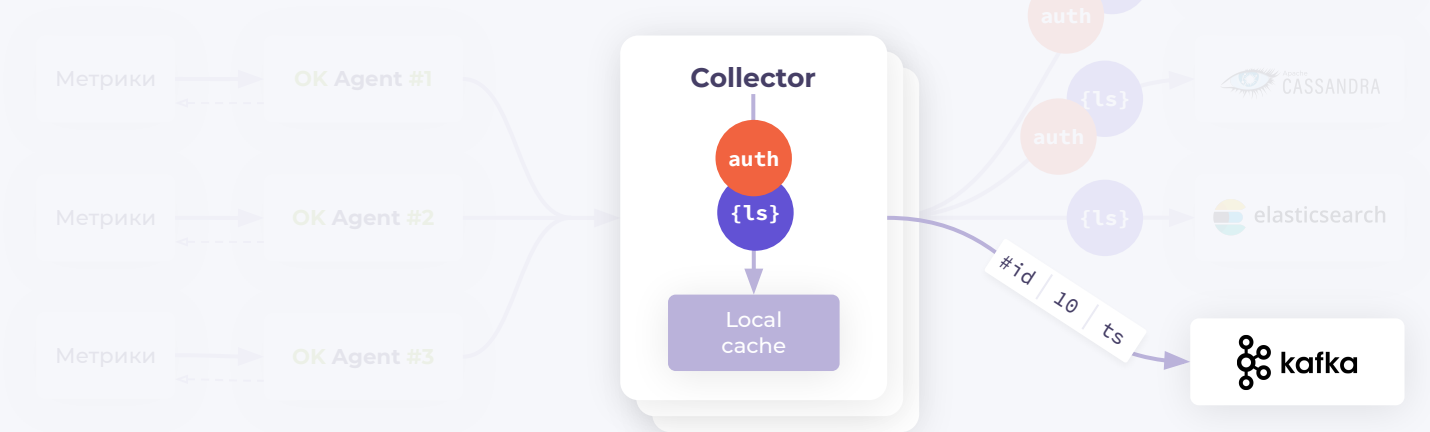
- Идея с bunch классная
- Позволяет сильно ускорить и запись и чтение. **Но!**
- Работает это только благодаря тому, что мы заранее знаем, какие будут метрики и как мы ими пользуемся

Okmeter – собственный путь

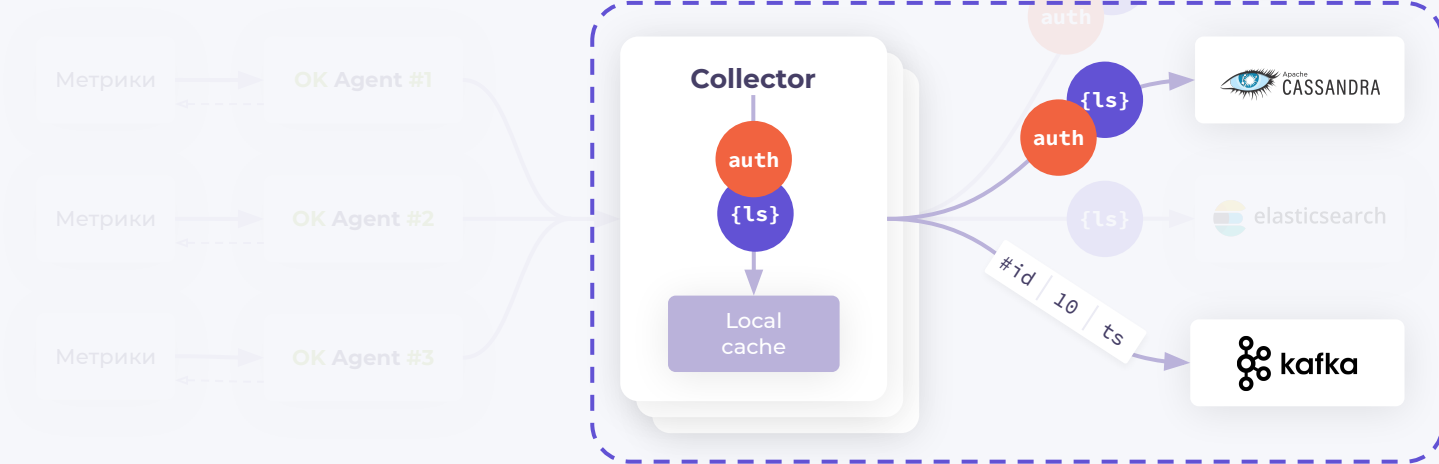


- Идея с bunch классная
- Позволяет сильно ускорить и запись и чтение. **Но!**
- Работает это только благодаря тому, что мы заранее знаем, какие будут метрики и как мы ими пользуемся
- В общем случае эта задача не решается

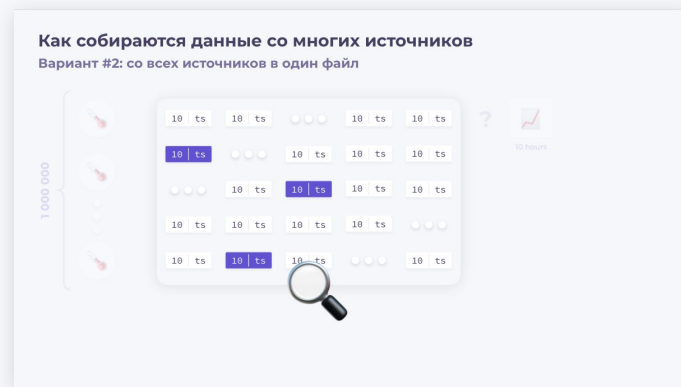
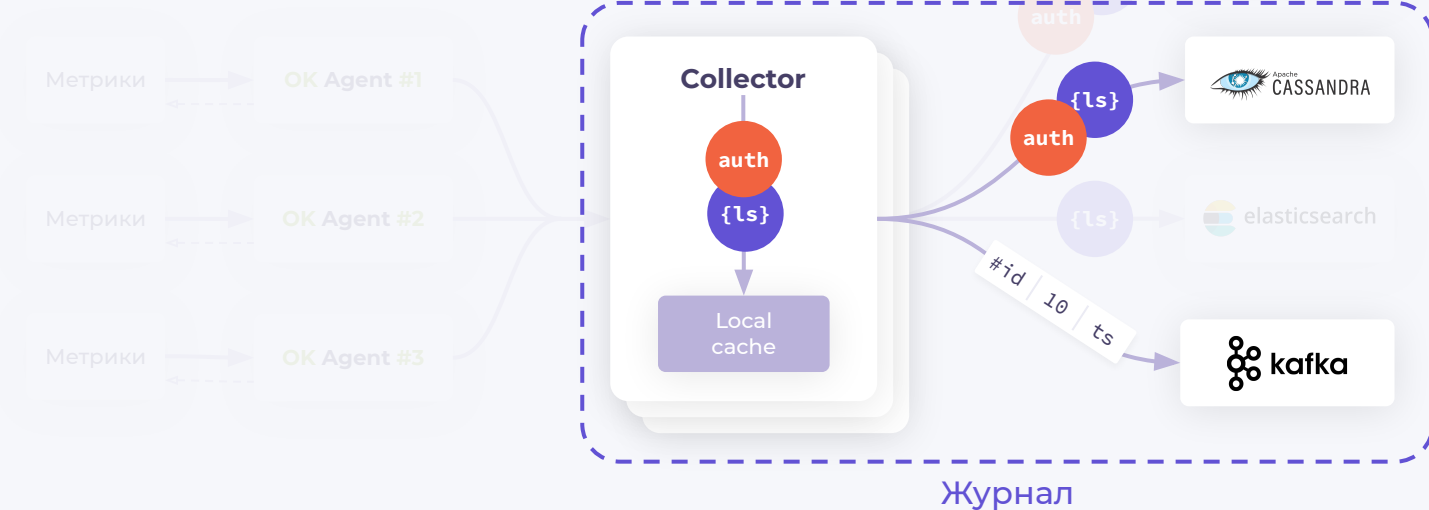
Okmeter – собственный путь



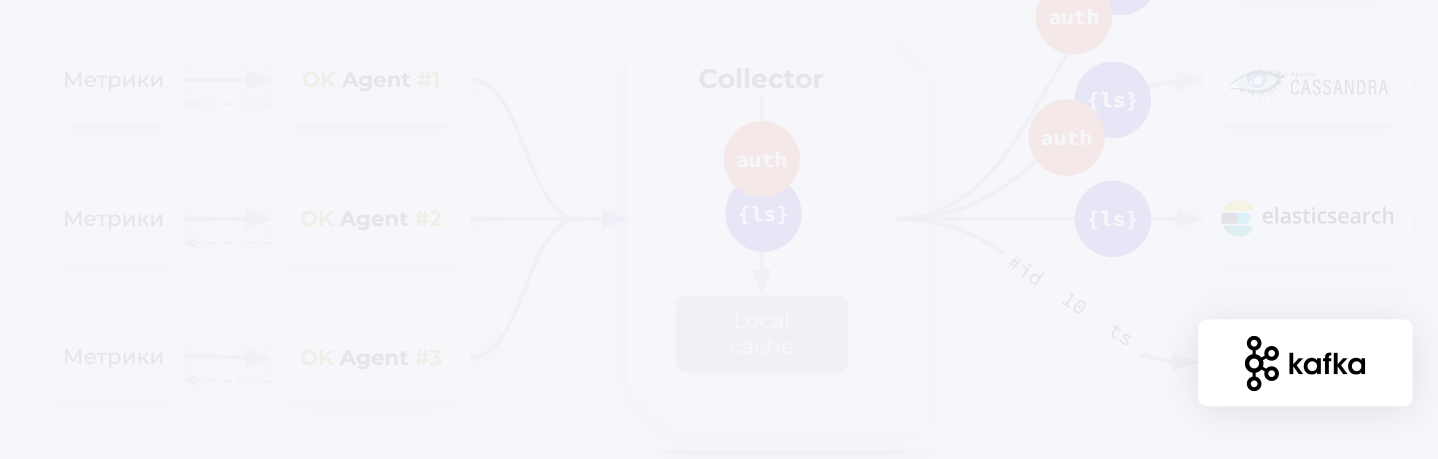
Okmeter – собственный путь



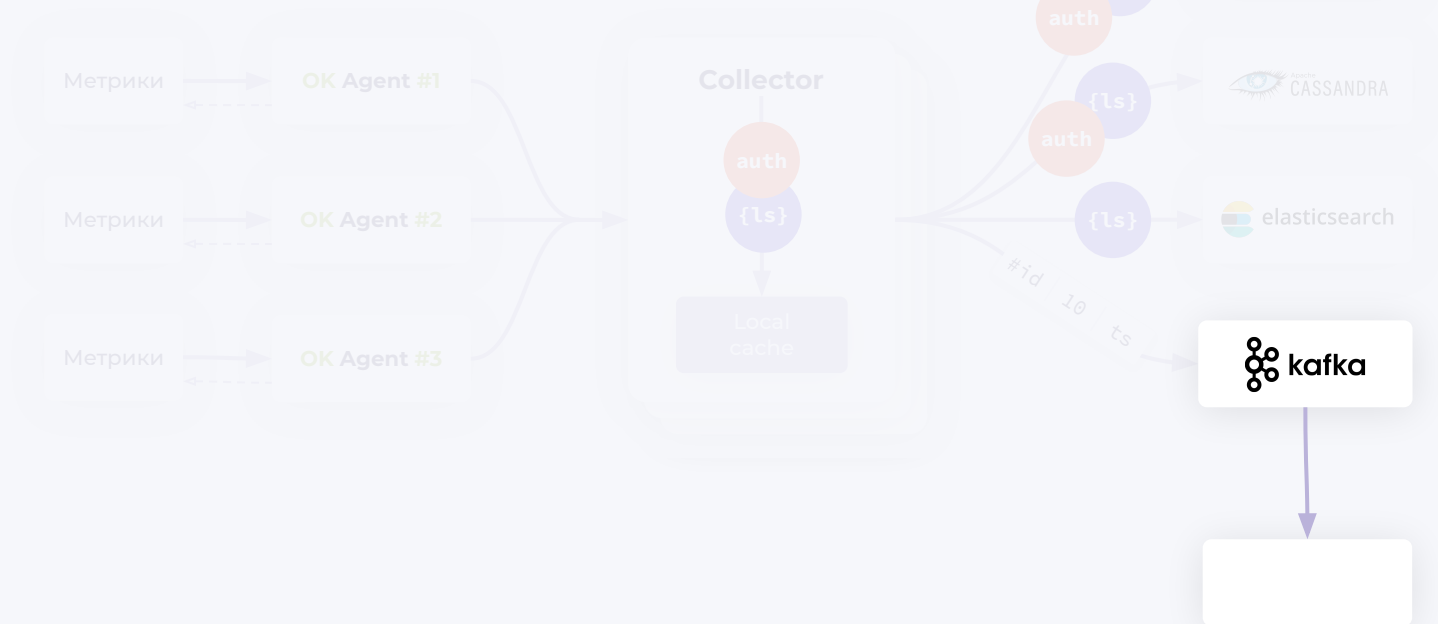
Okmeter – собственный путь



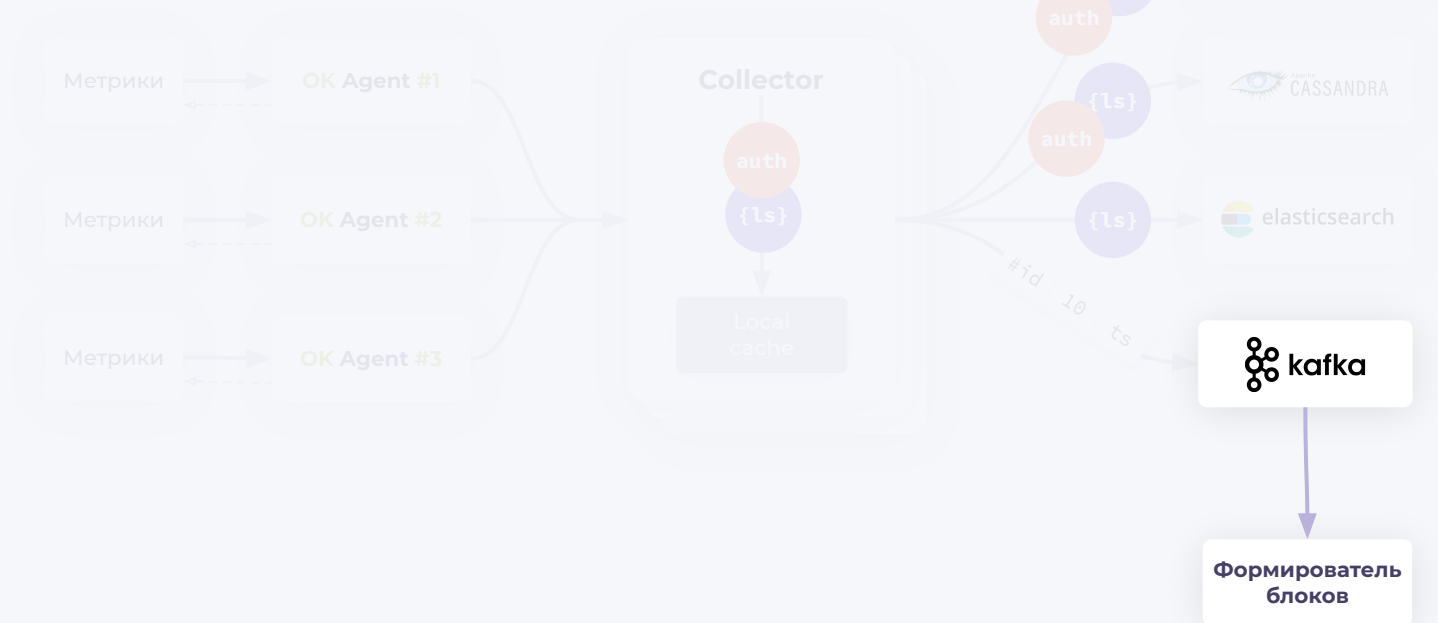
Okmeter – собственный путь



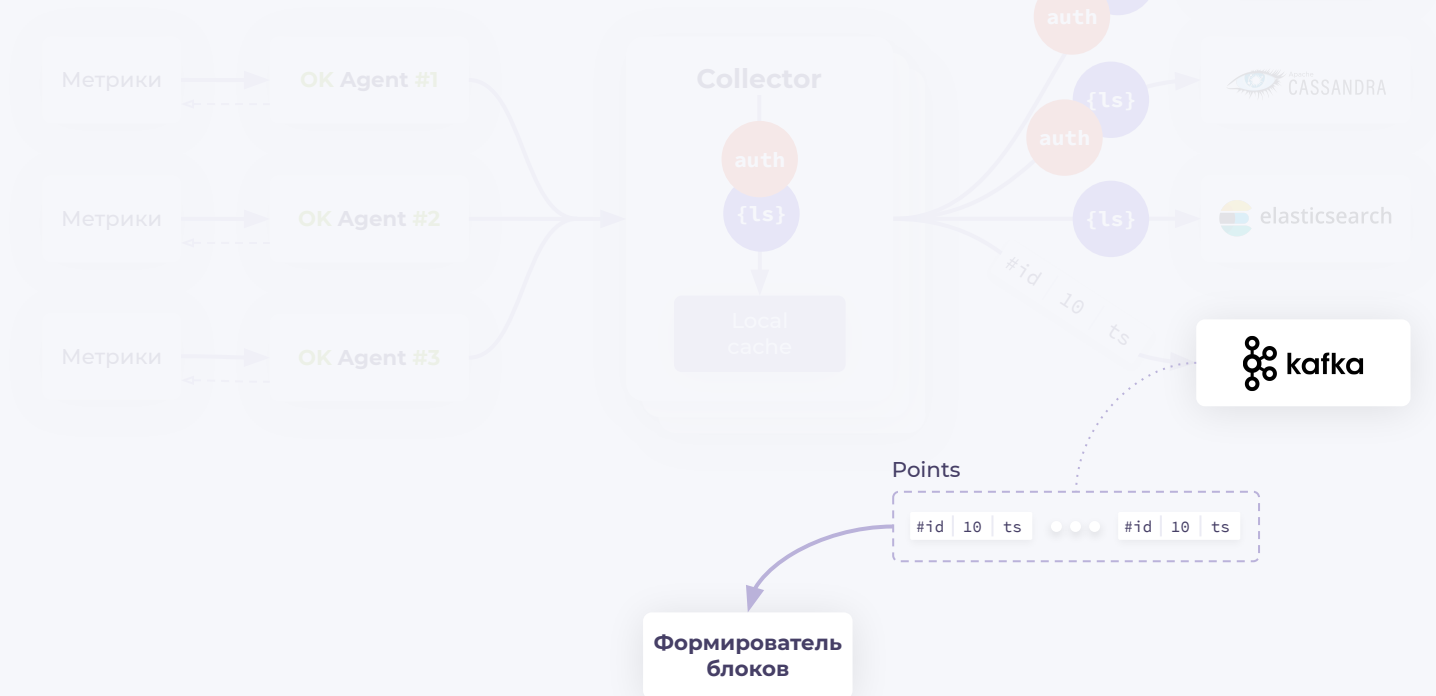
Okmeter – собственный путь



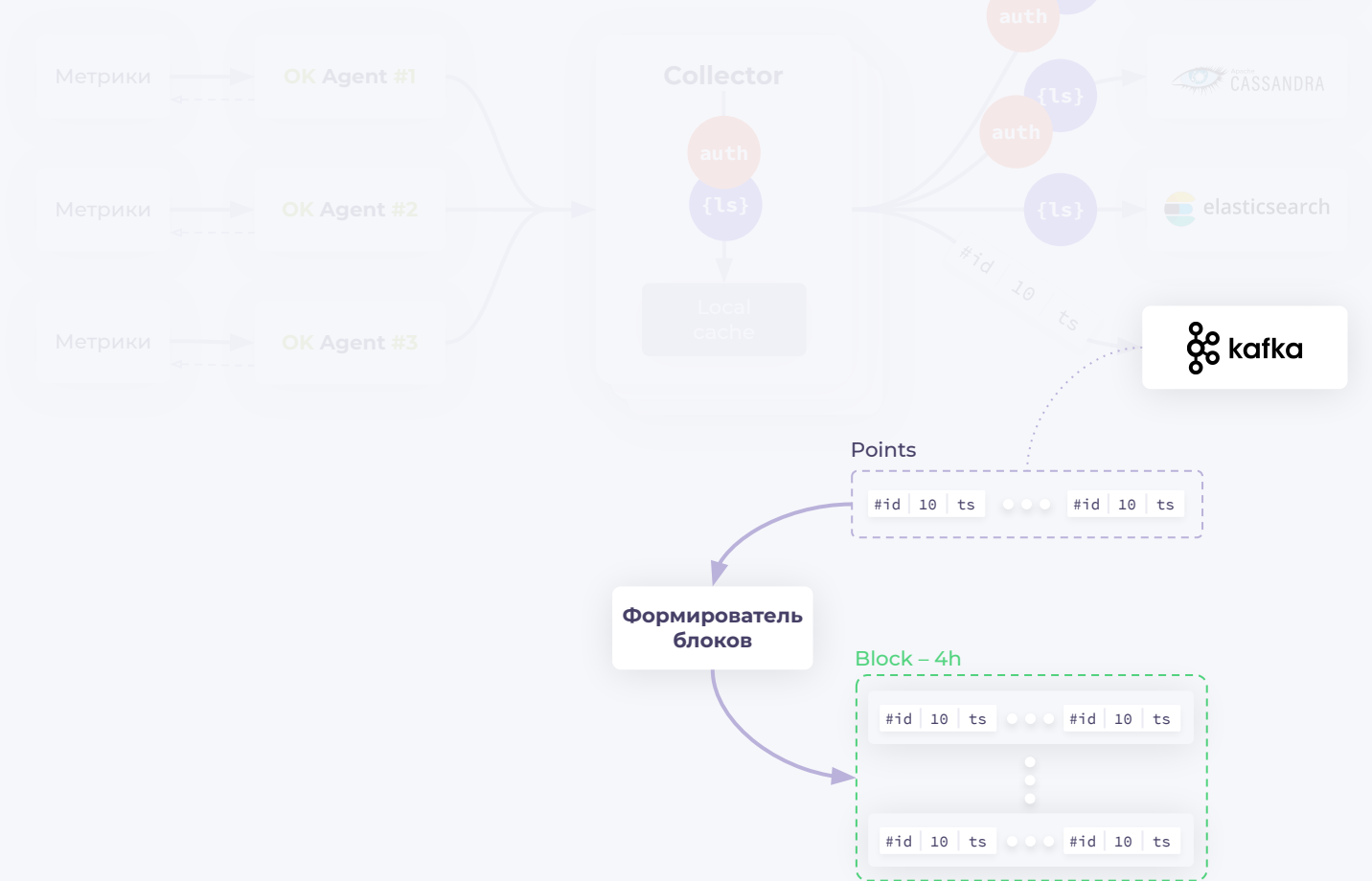
Okmeter – собственный путь



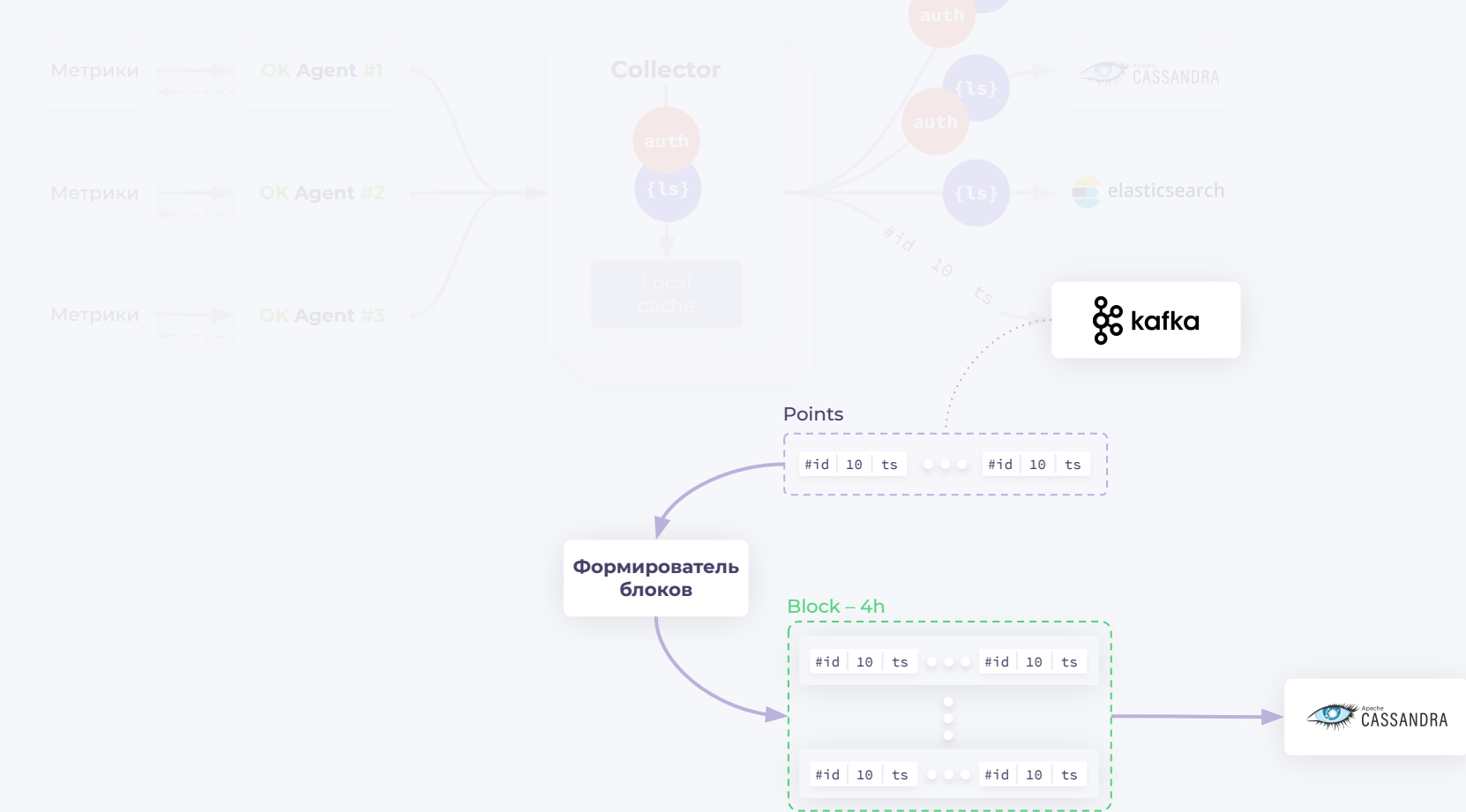
Okmeter – собственный путь



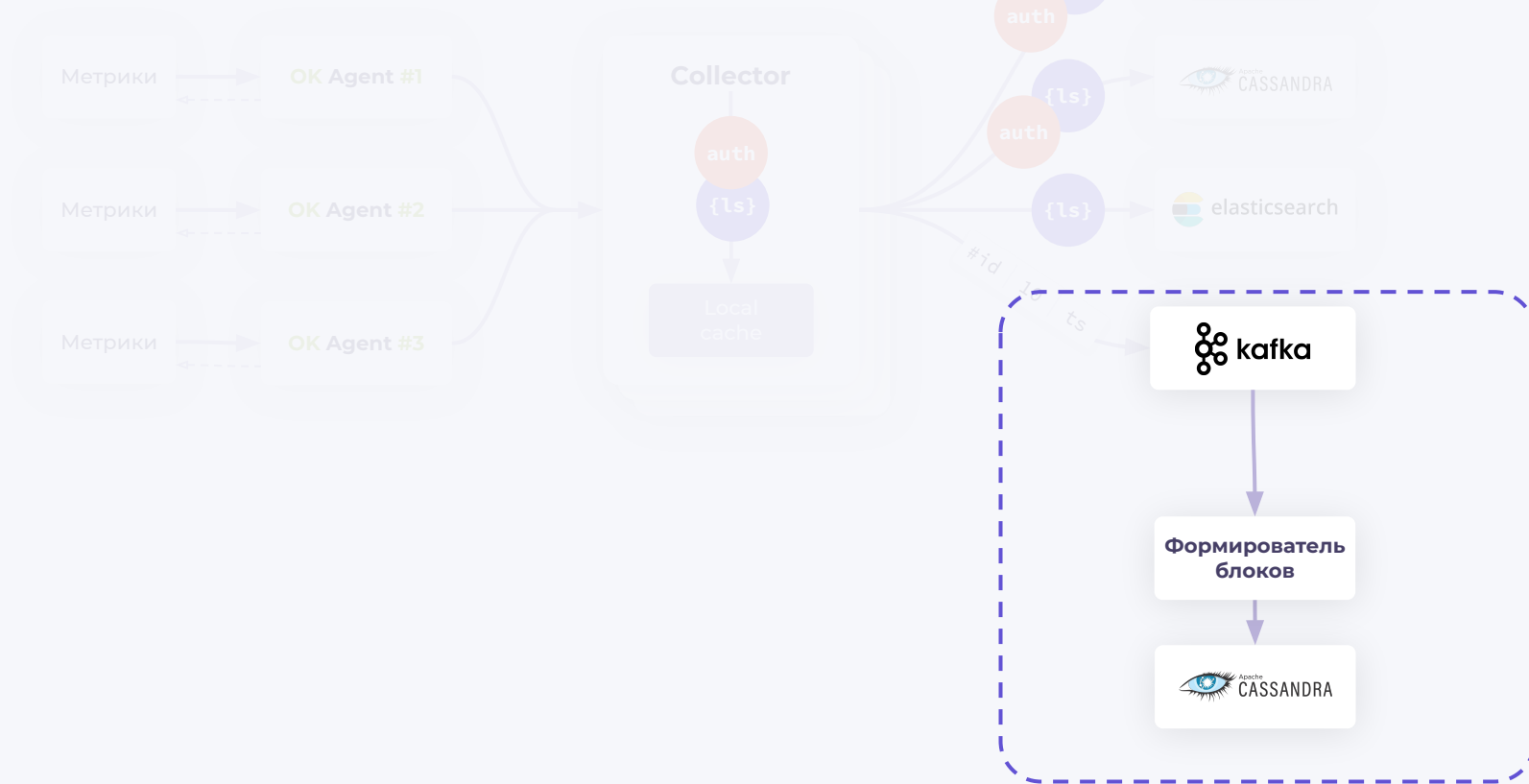
Okmeter – собственный путь



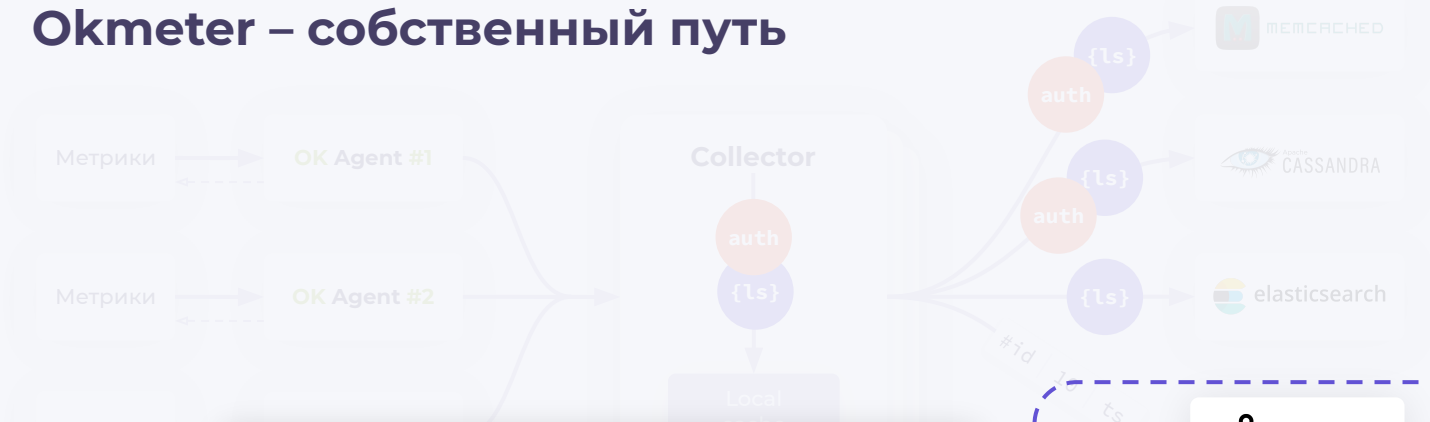
Okmeter – собственный путь



Okmeter – собственный путь

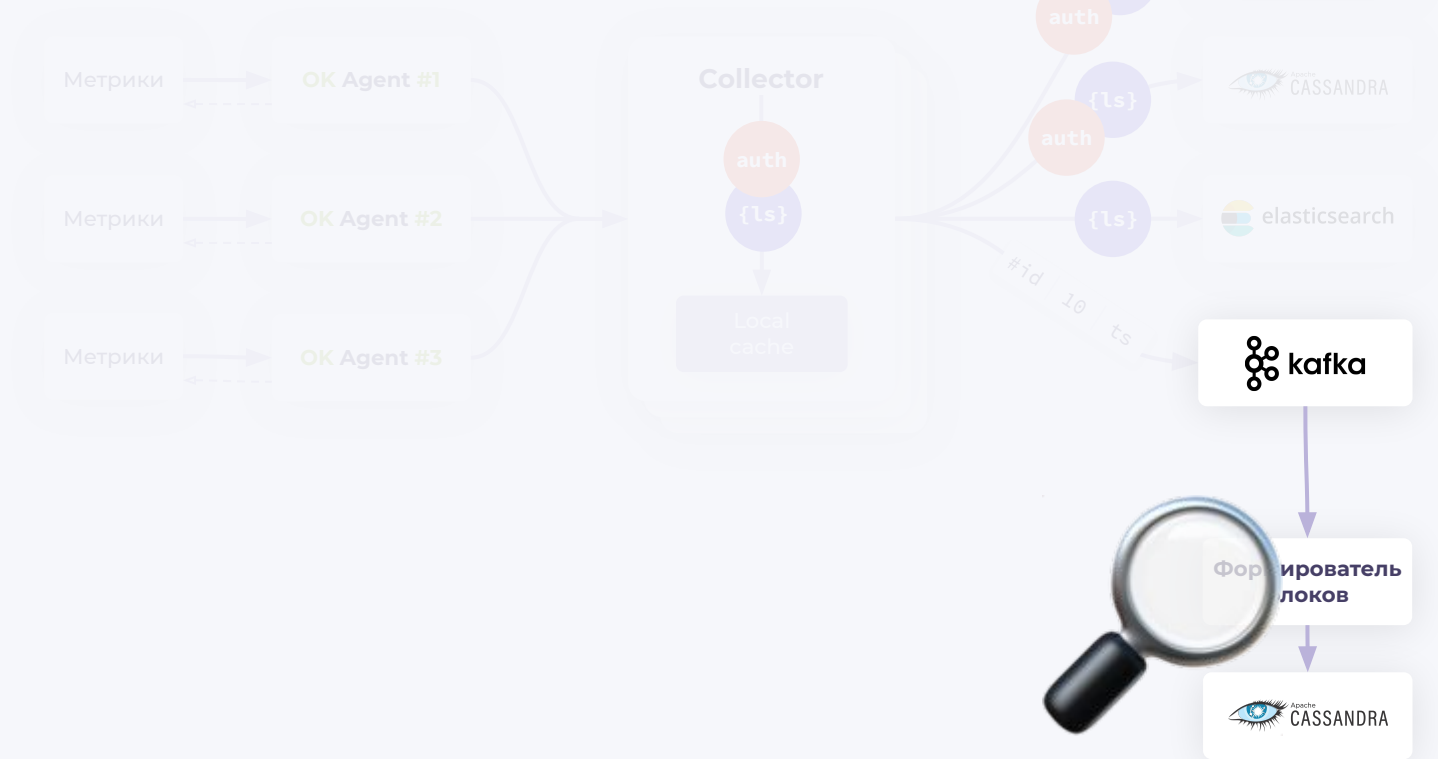


Okmeter – собственный путь



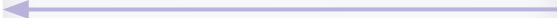
Блок

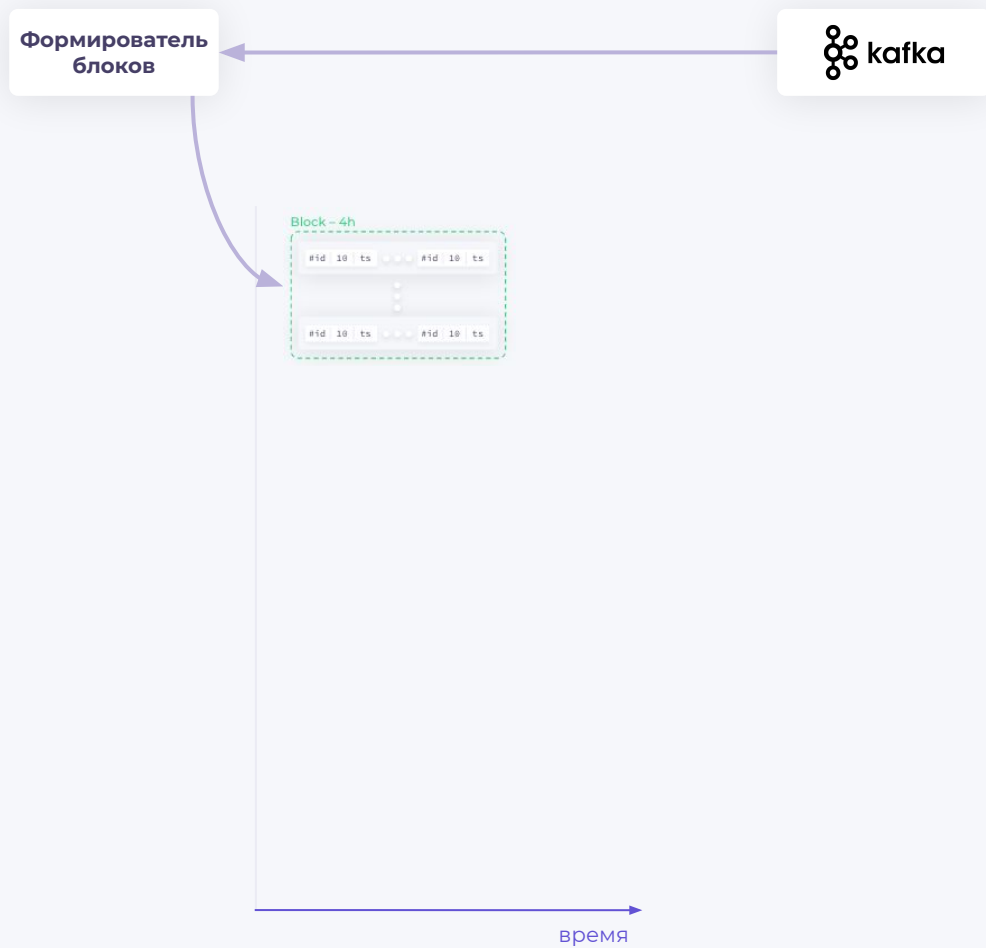
Okmeter – собственный путь

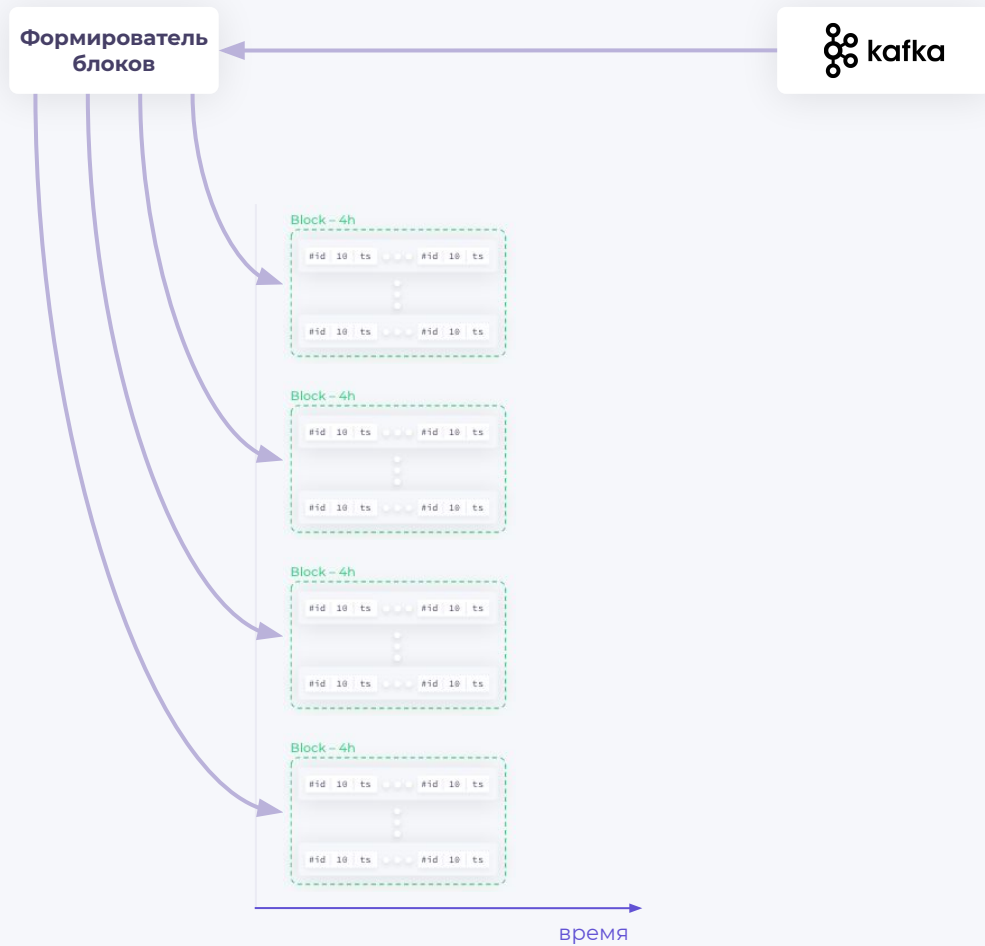


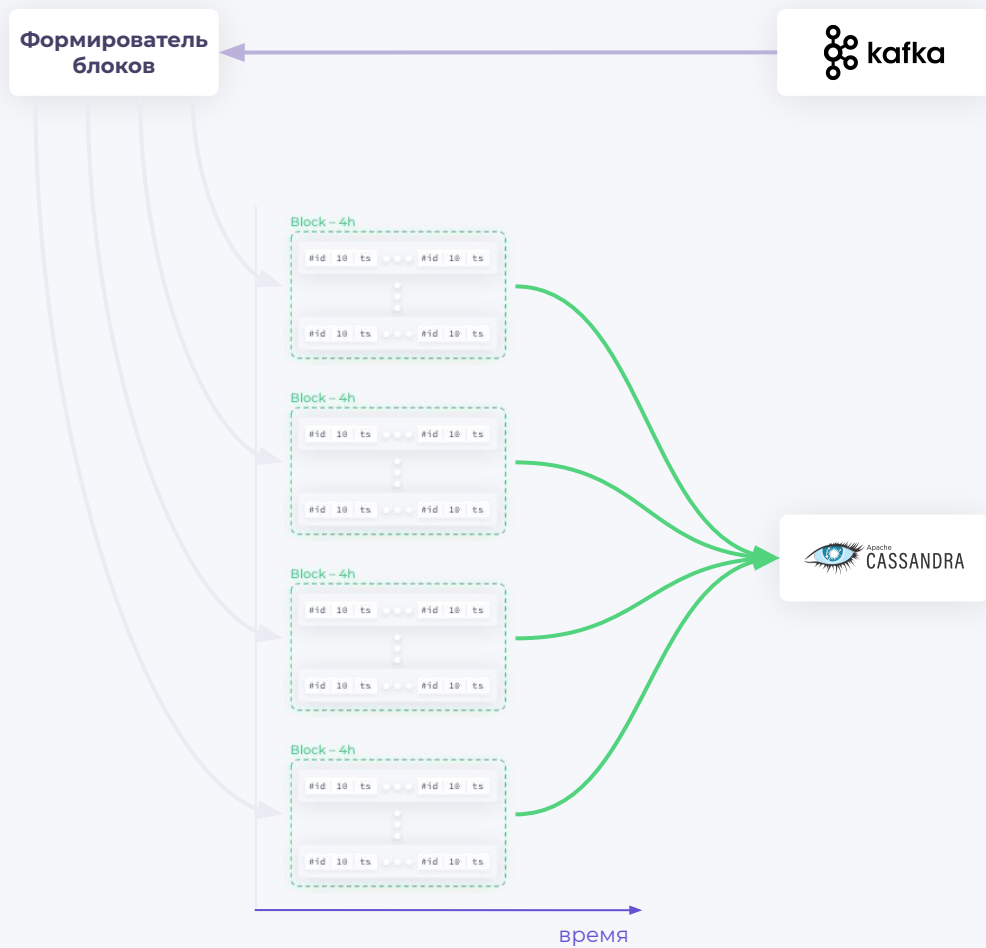
Формирователь
блоков

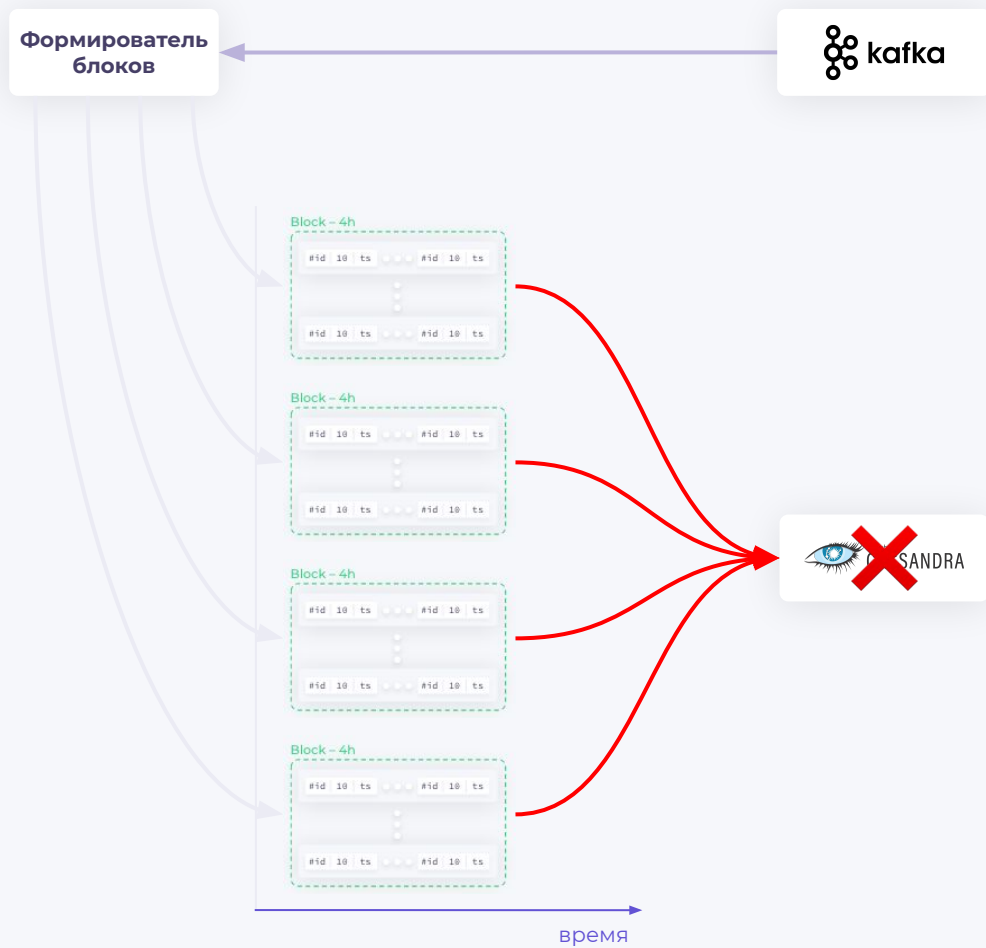
 kafka

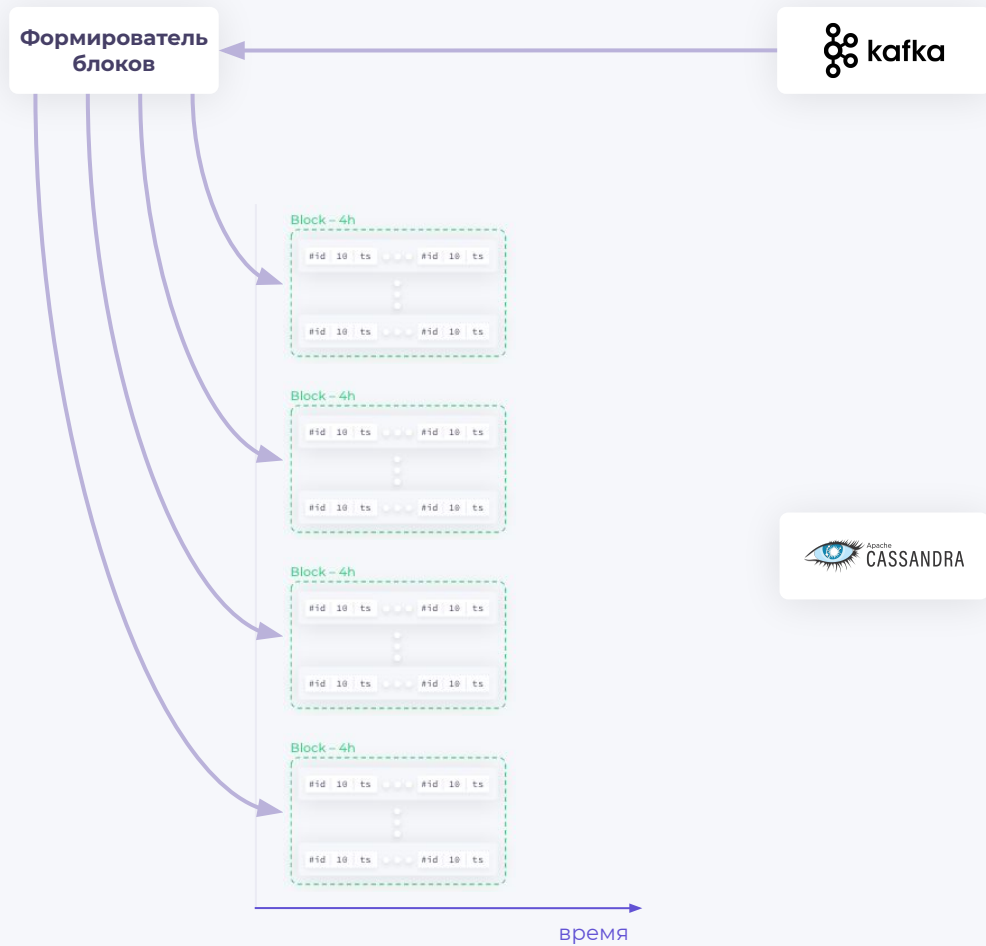








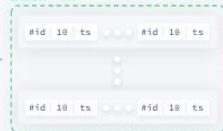




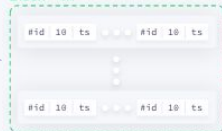
Формирователь
блоков



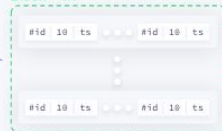
Block - 4h



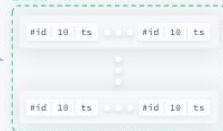
Block - 4h



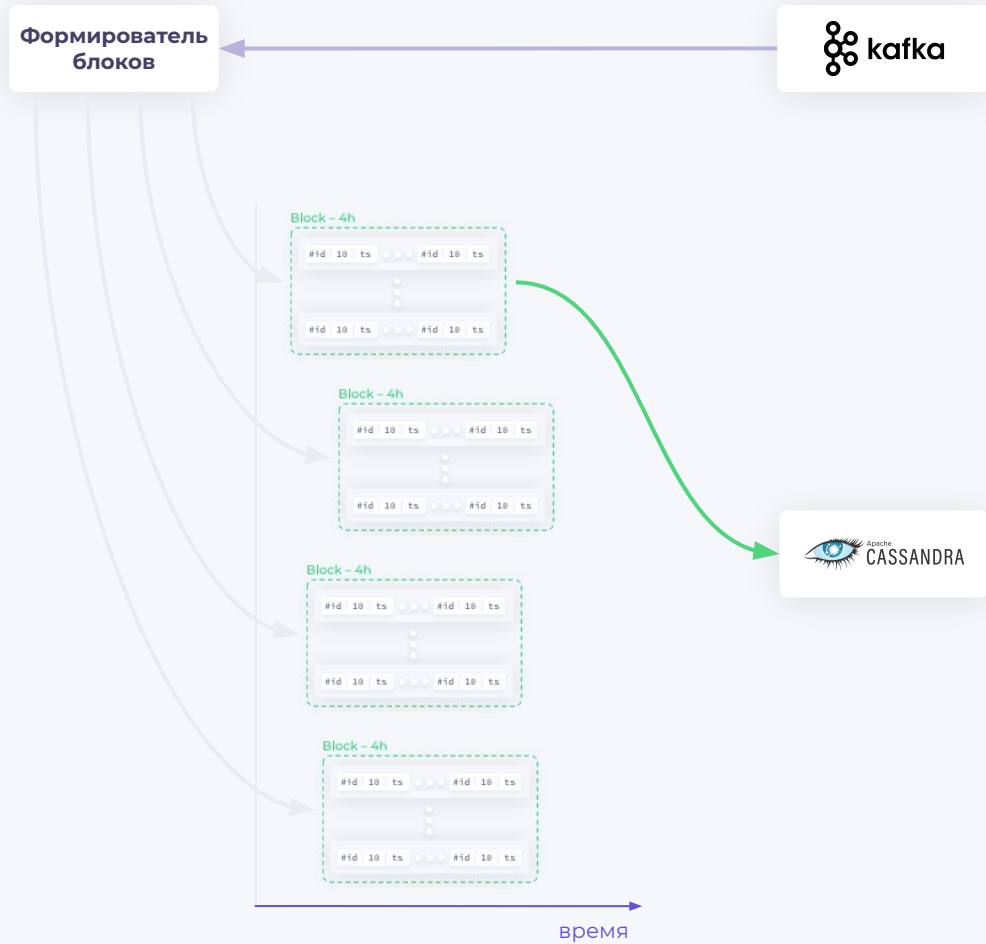
Block - 4h

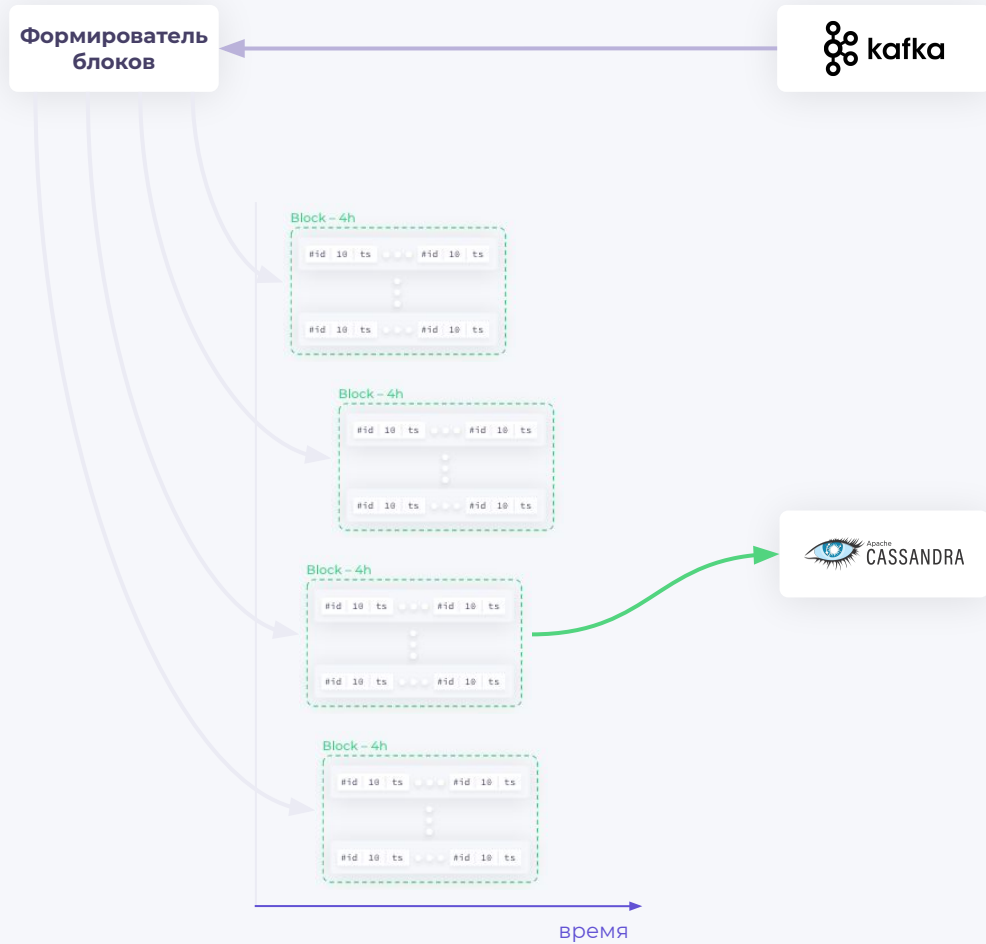


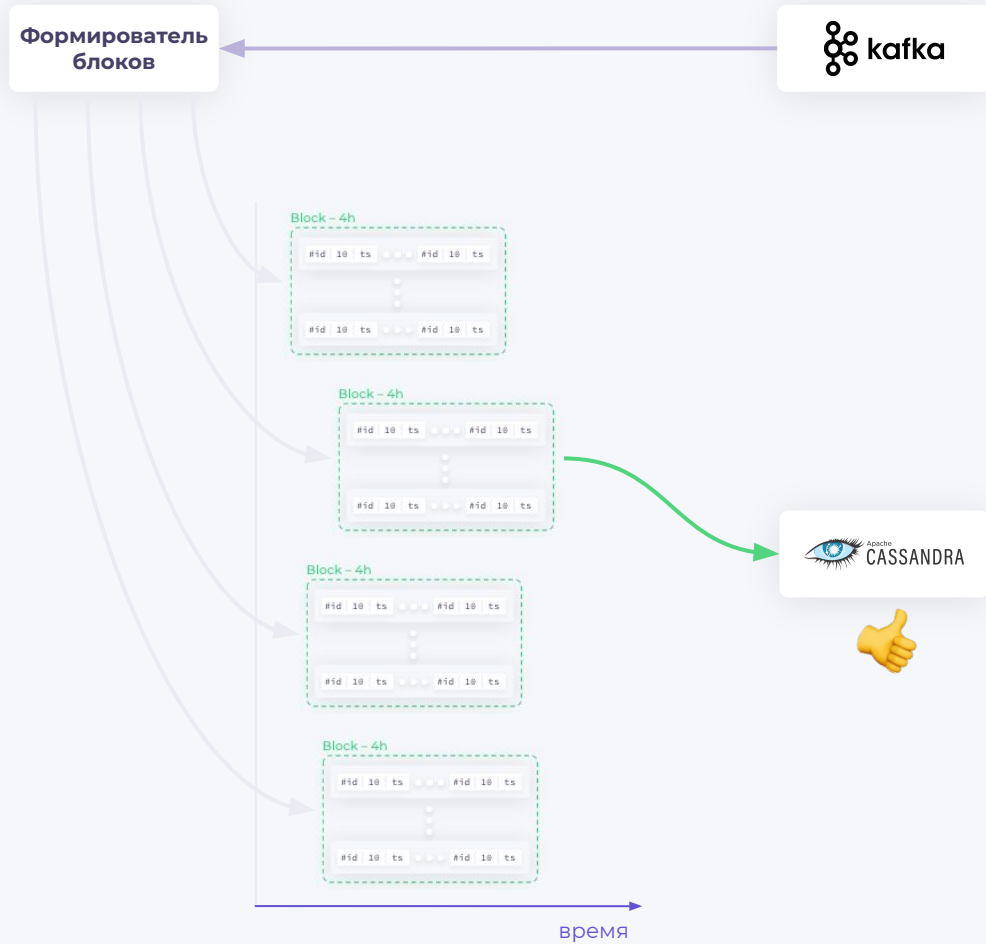
Block - 4h



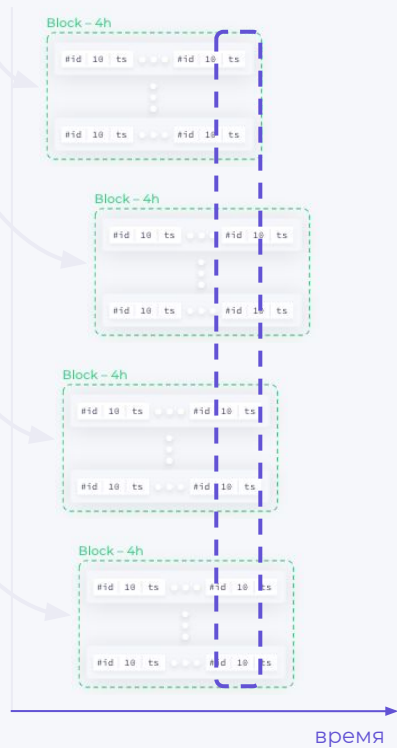
время

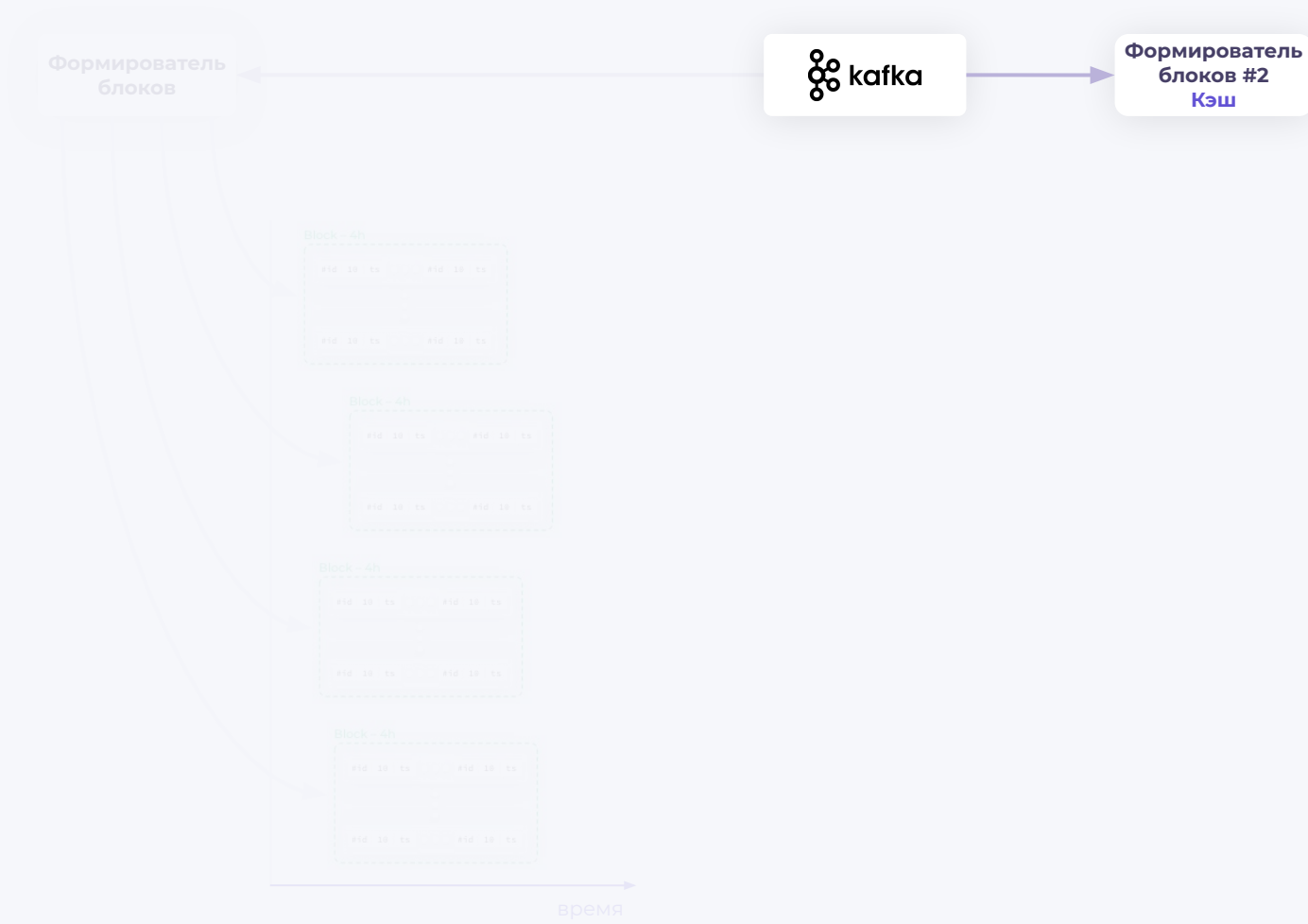


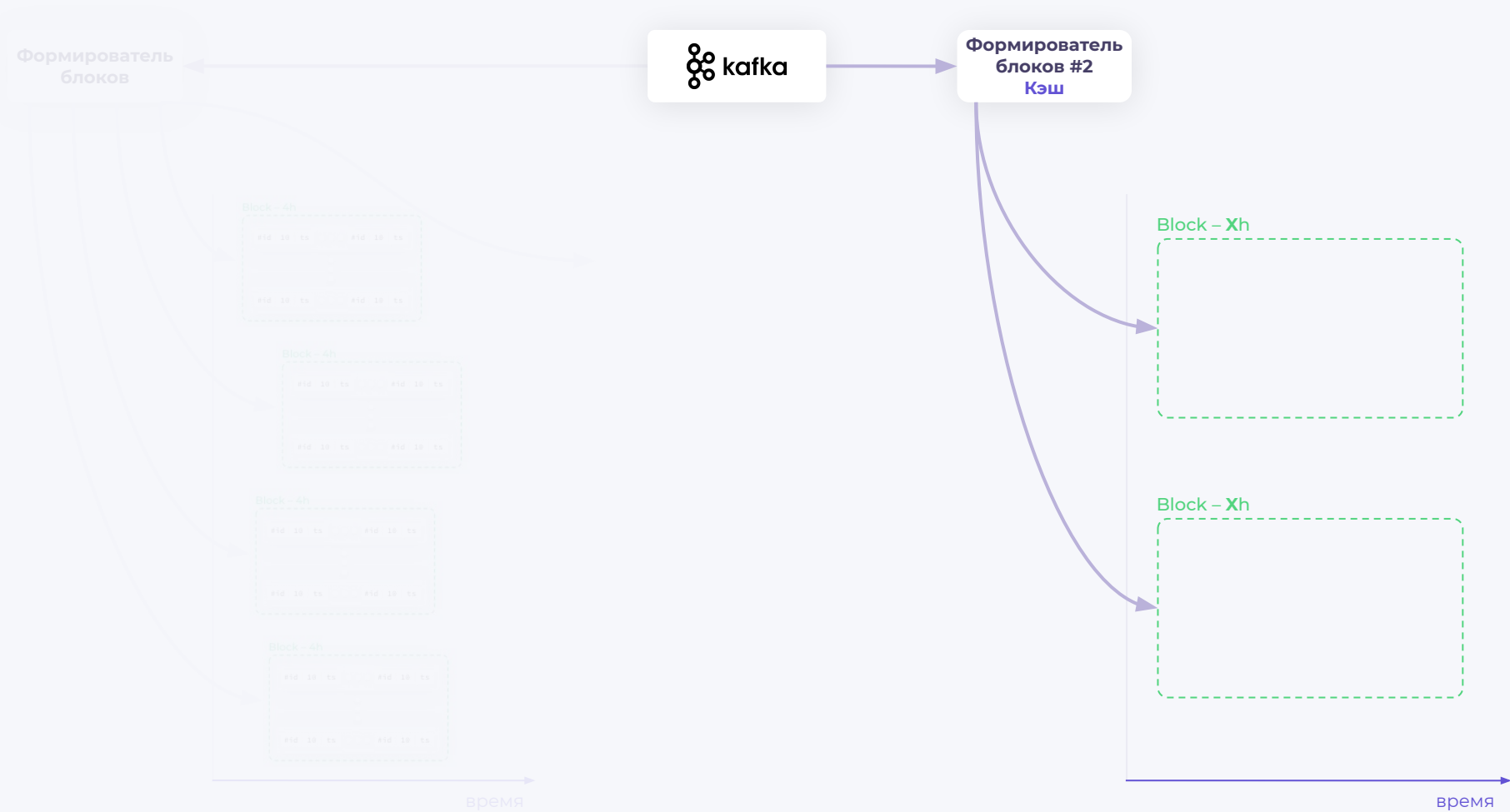


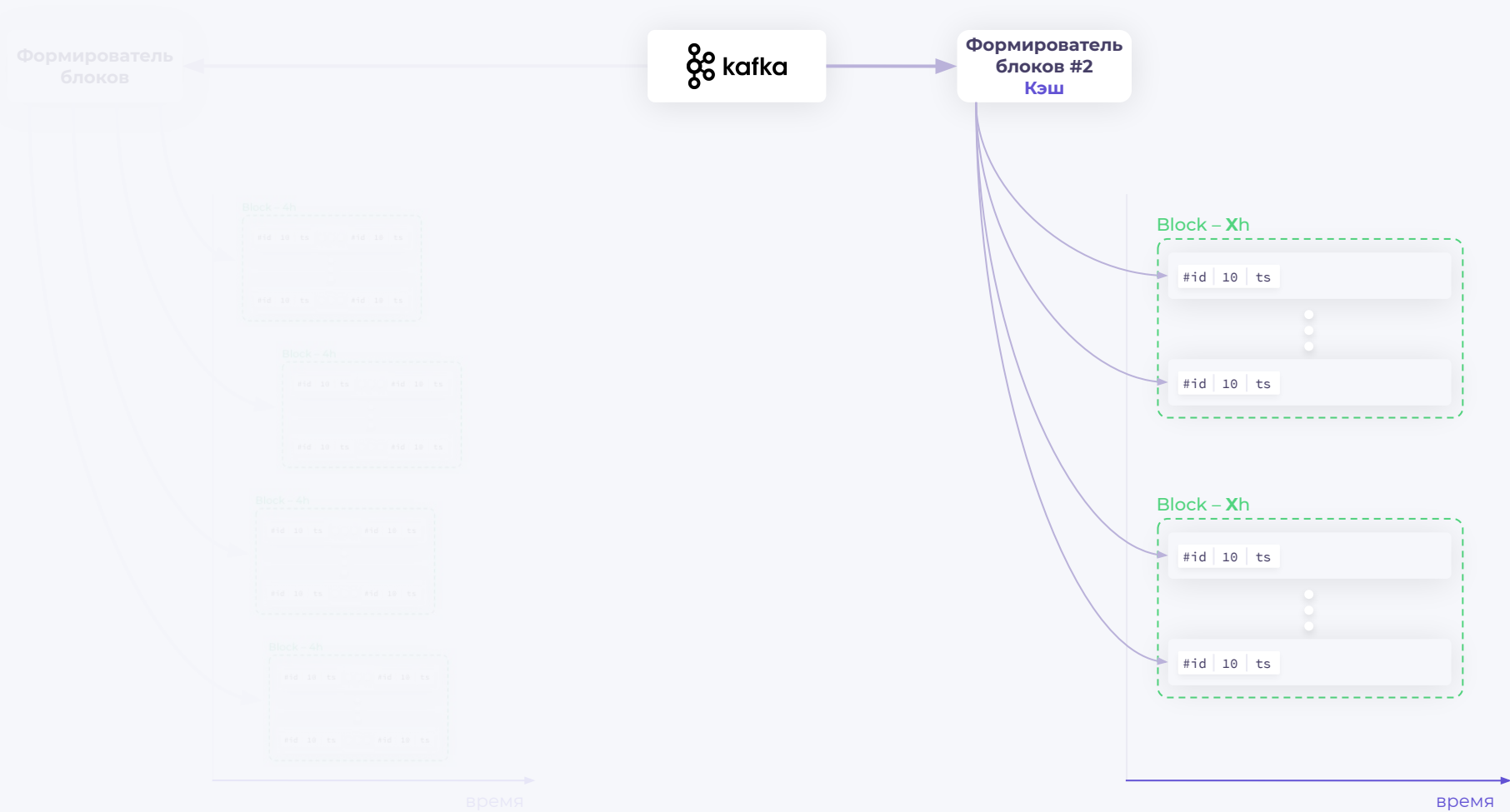


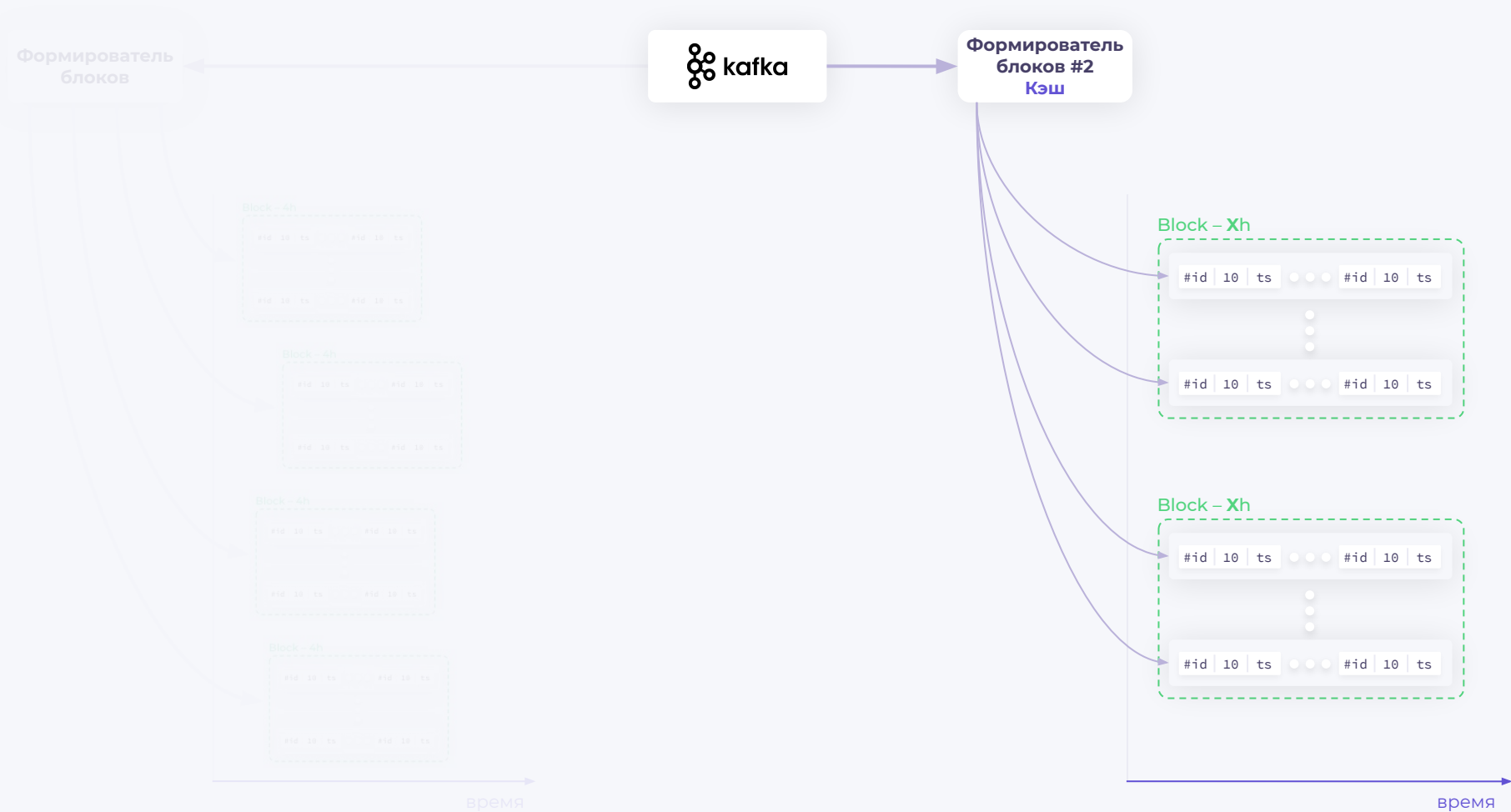
Формирователь
блоков



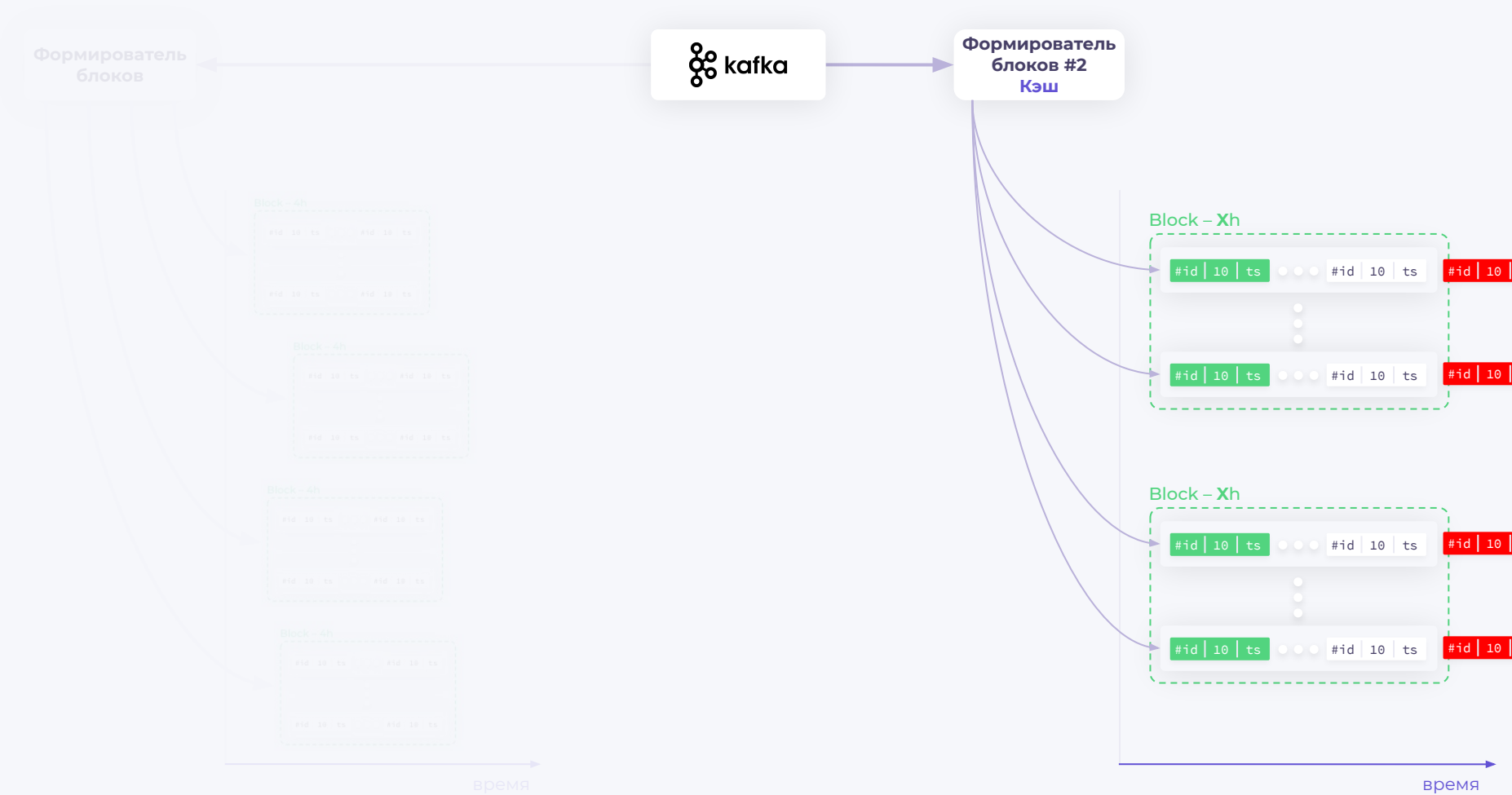




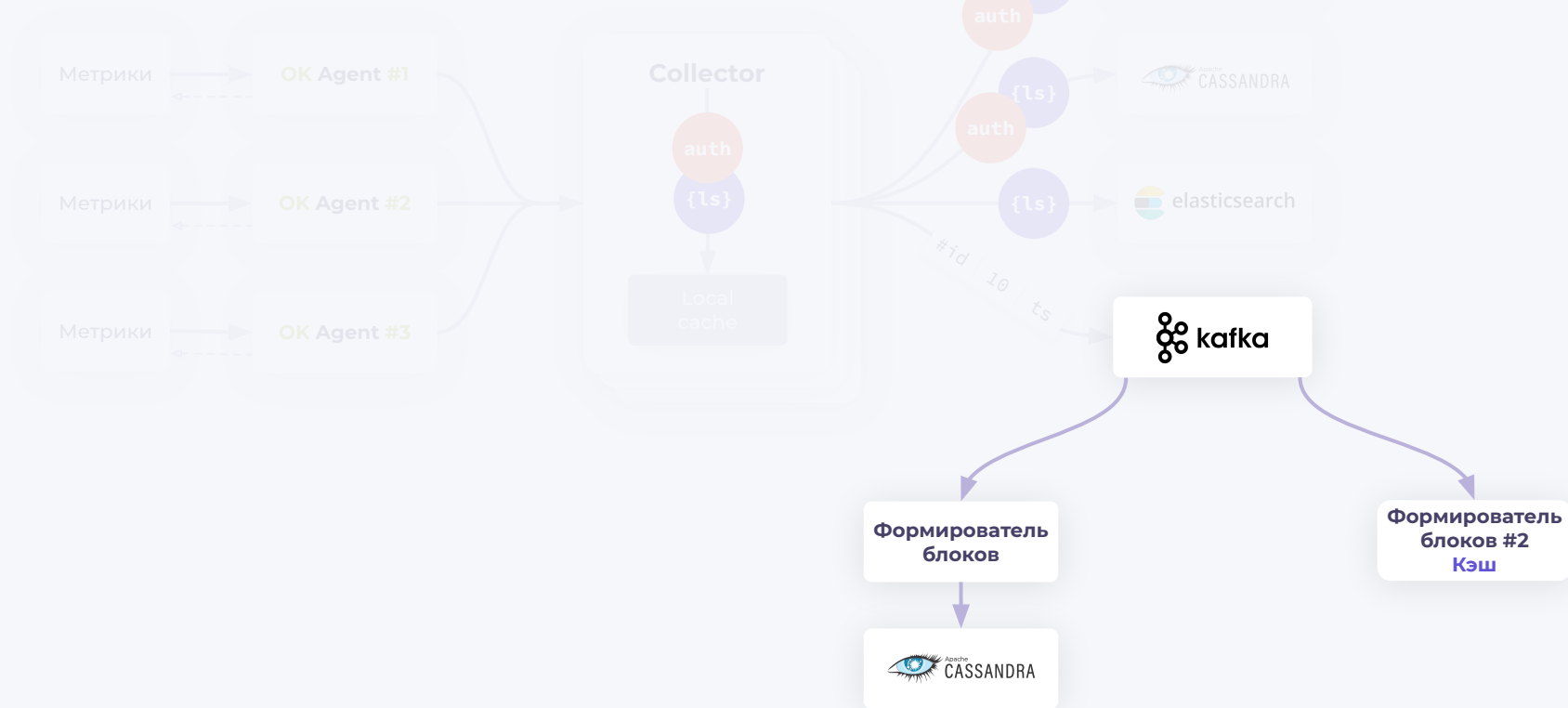




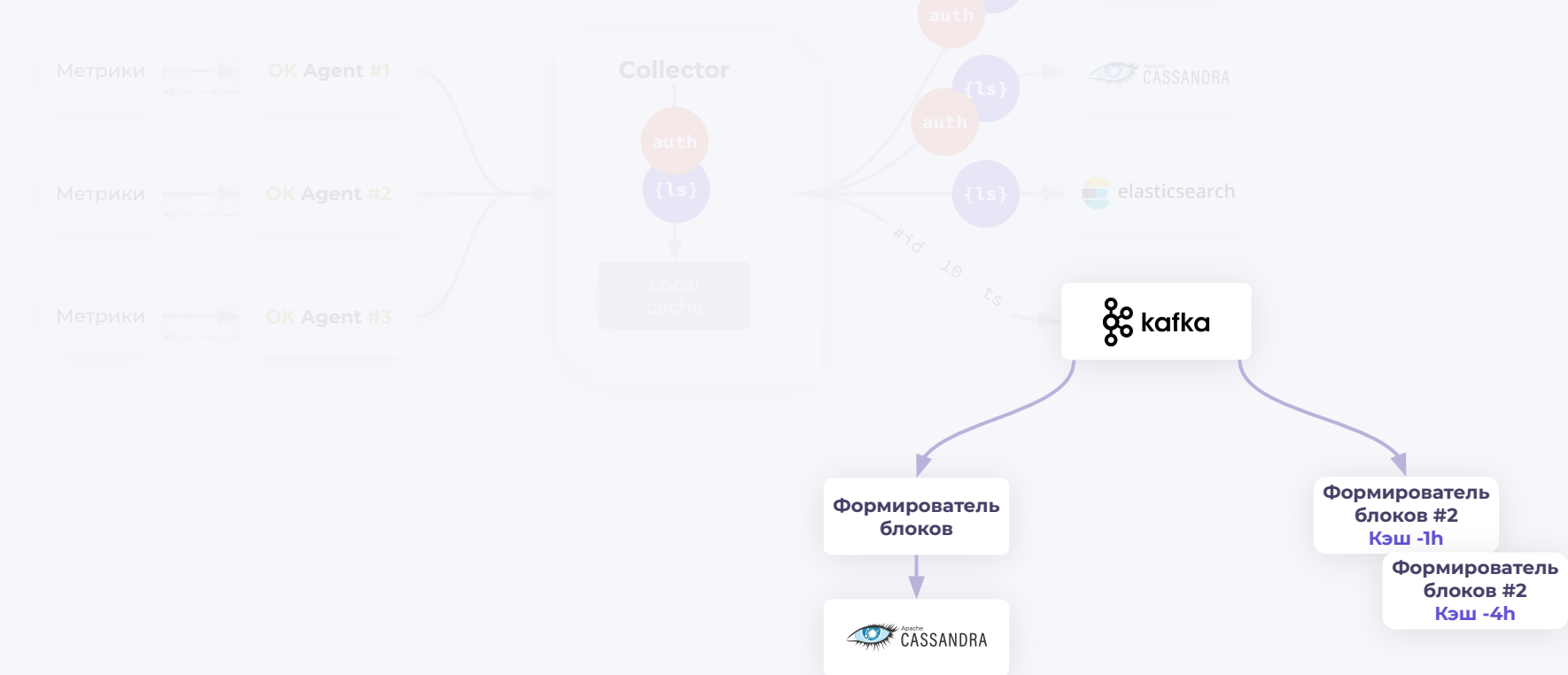




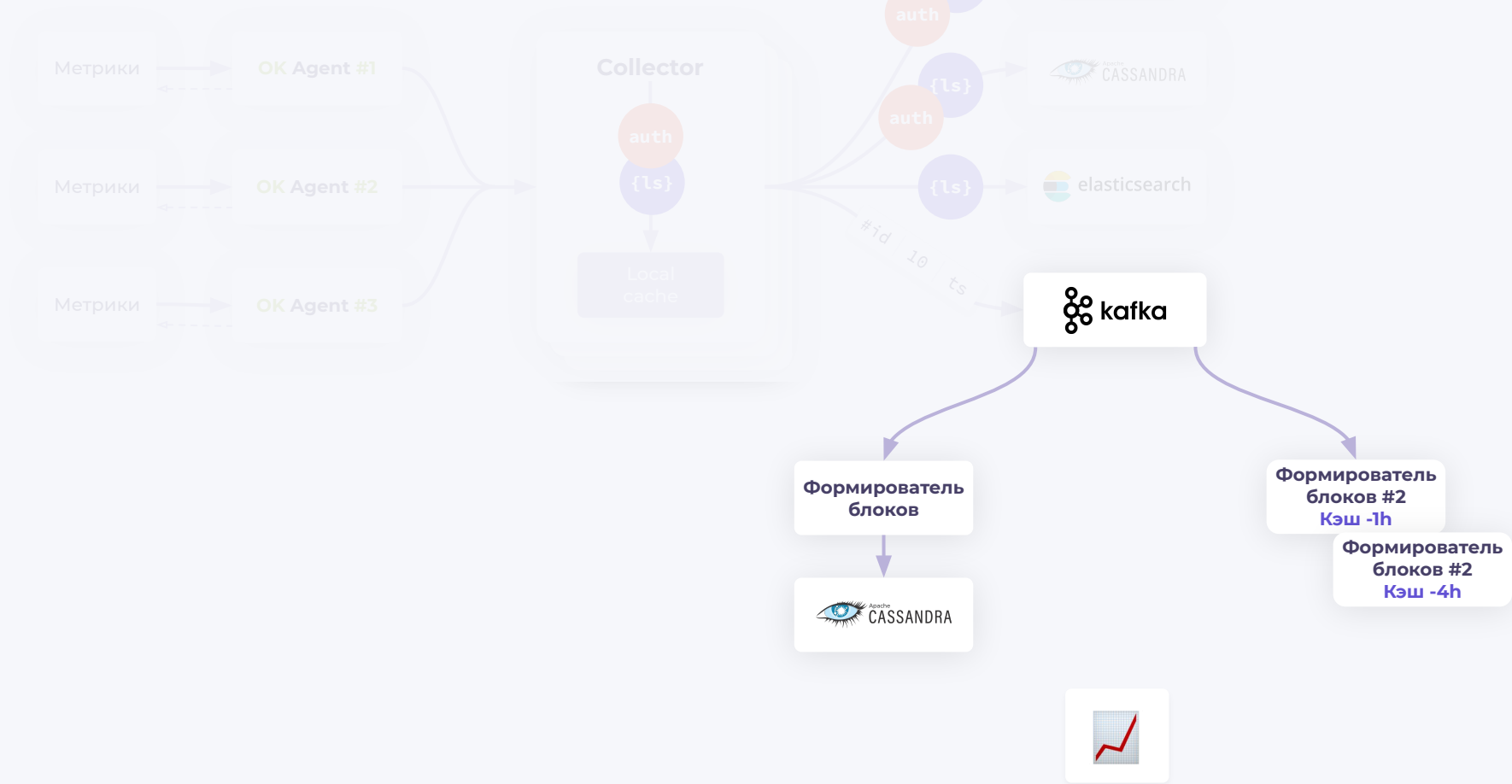
Okmeter – собственный путь



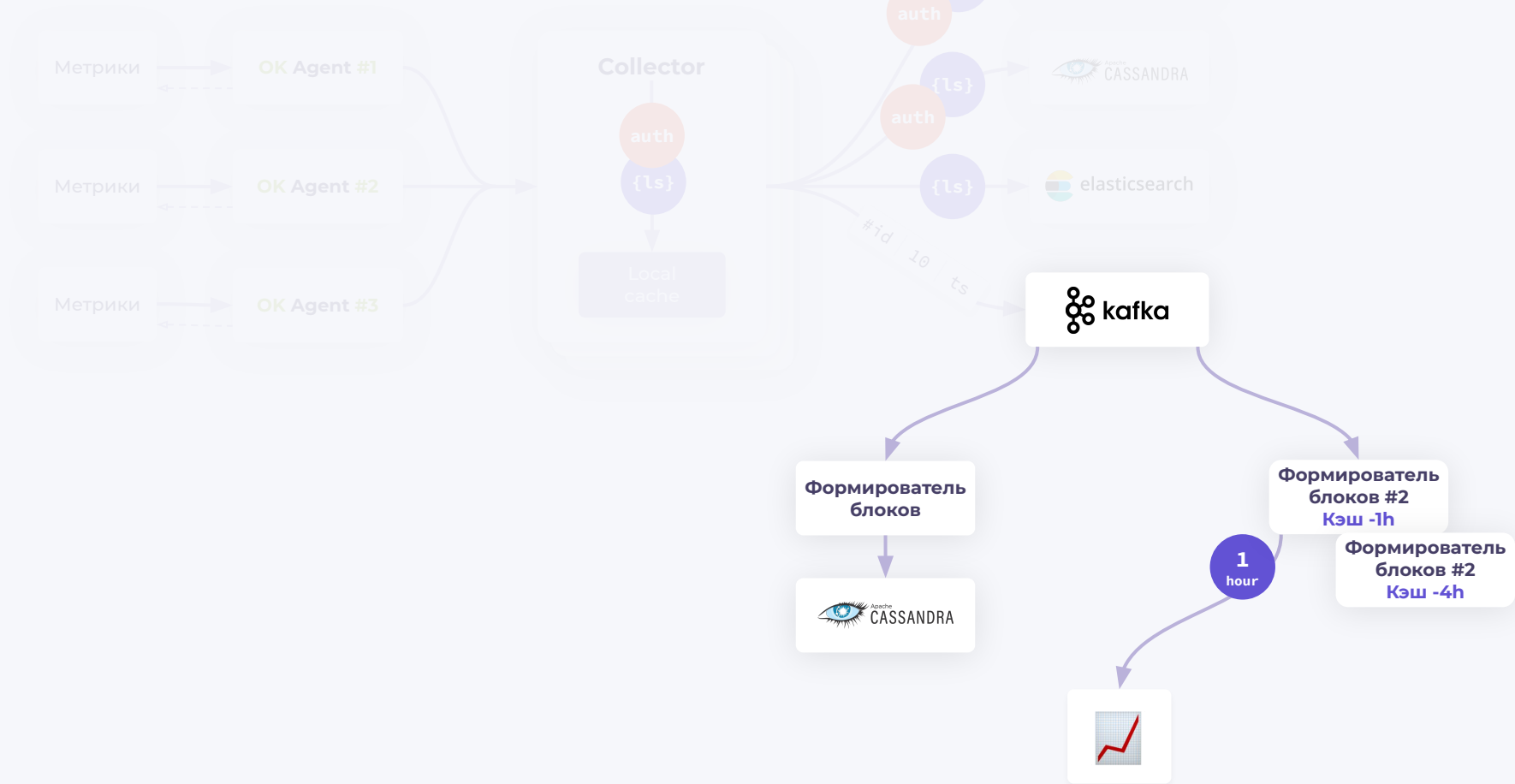
Okmeter – собственный путь



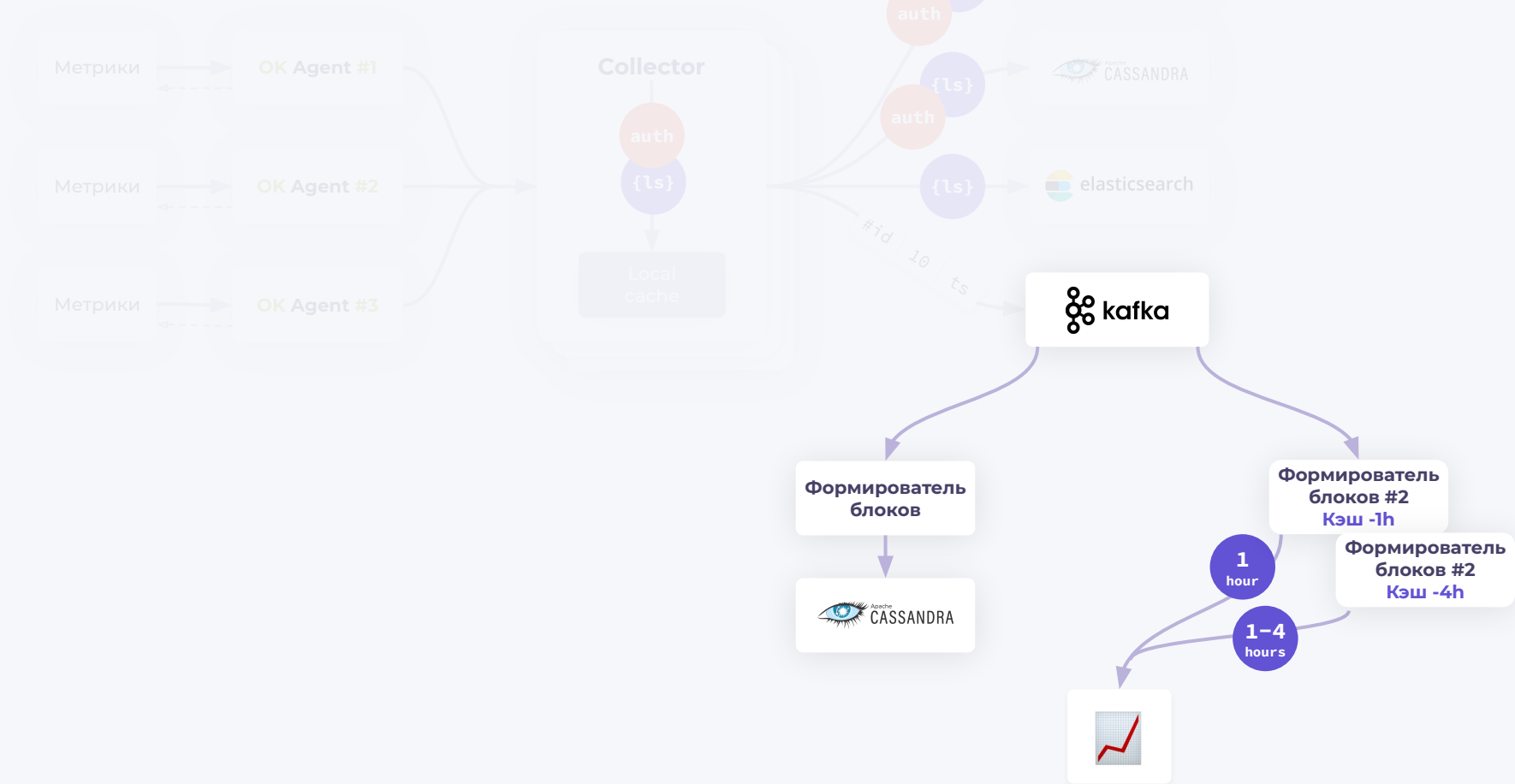
Okmeter – собственный путь



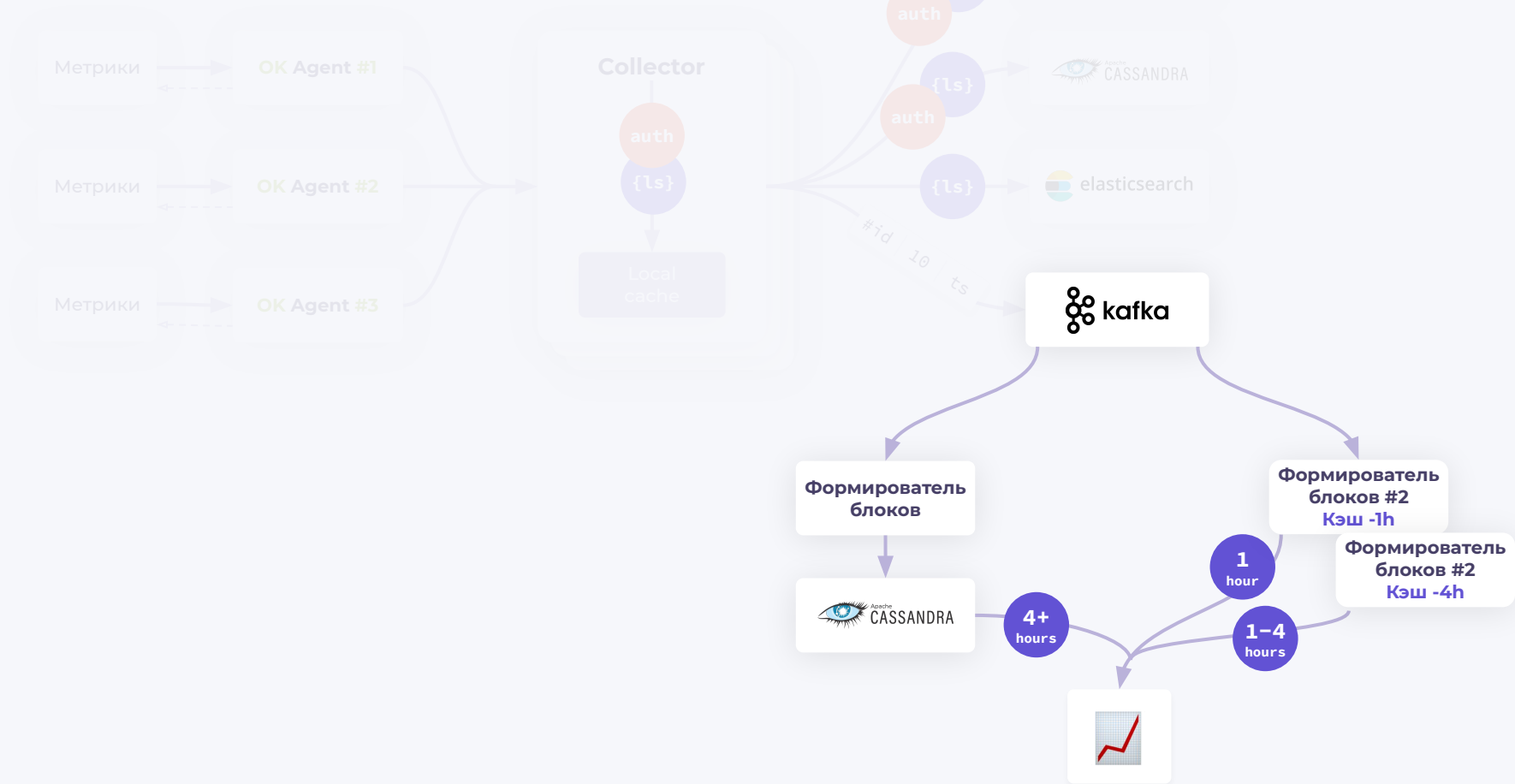
Okmeter – собственный путь



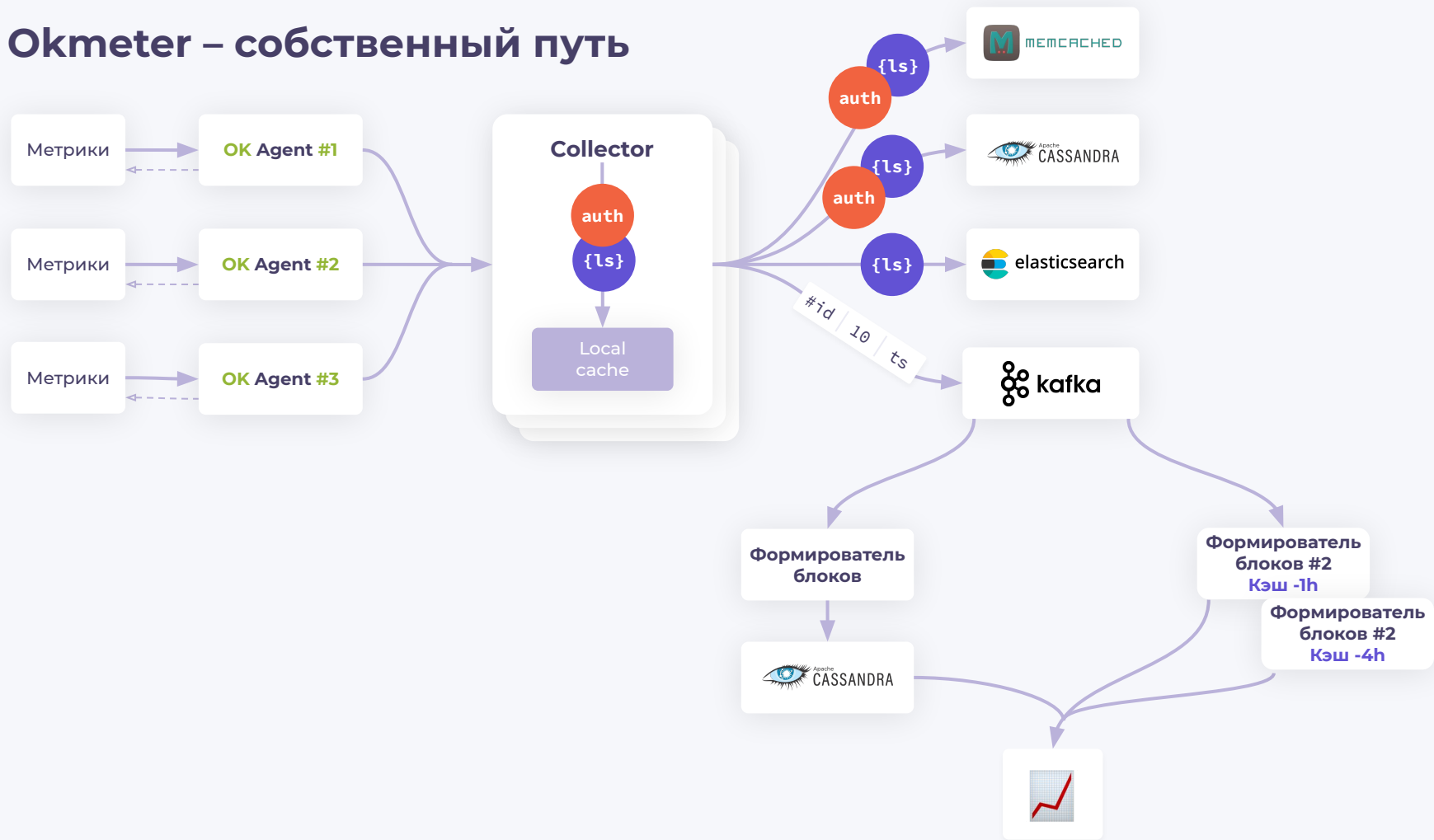
Okmeter – собственный путь



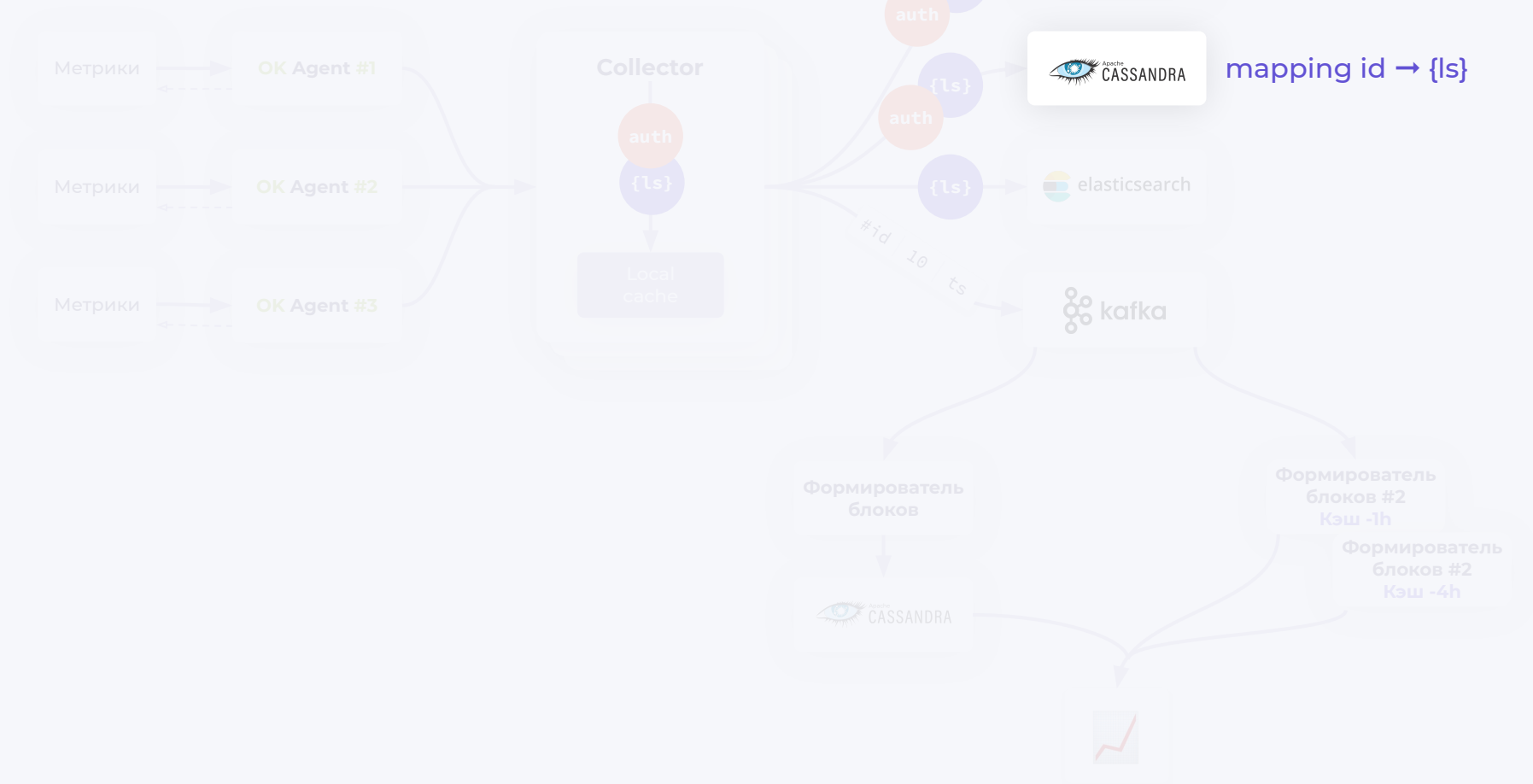
Okmeter – собственный путь



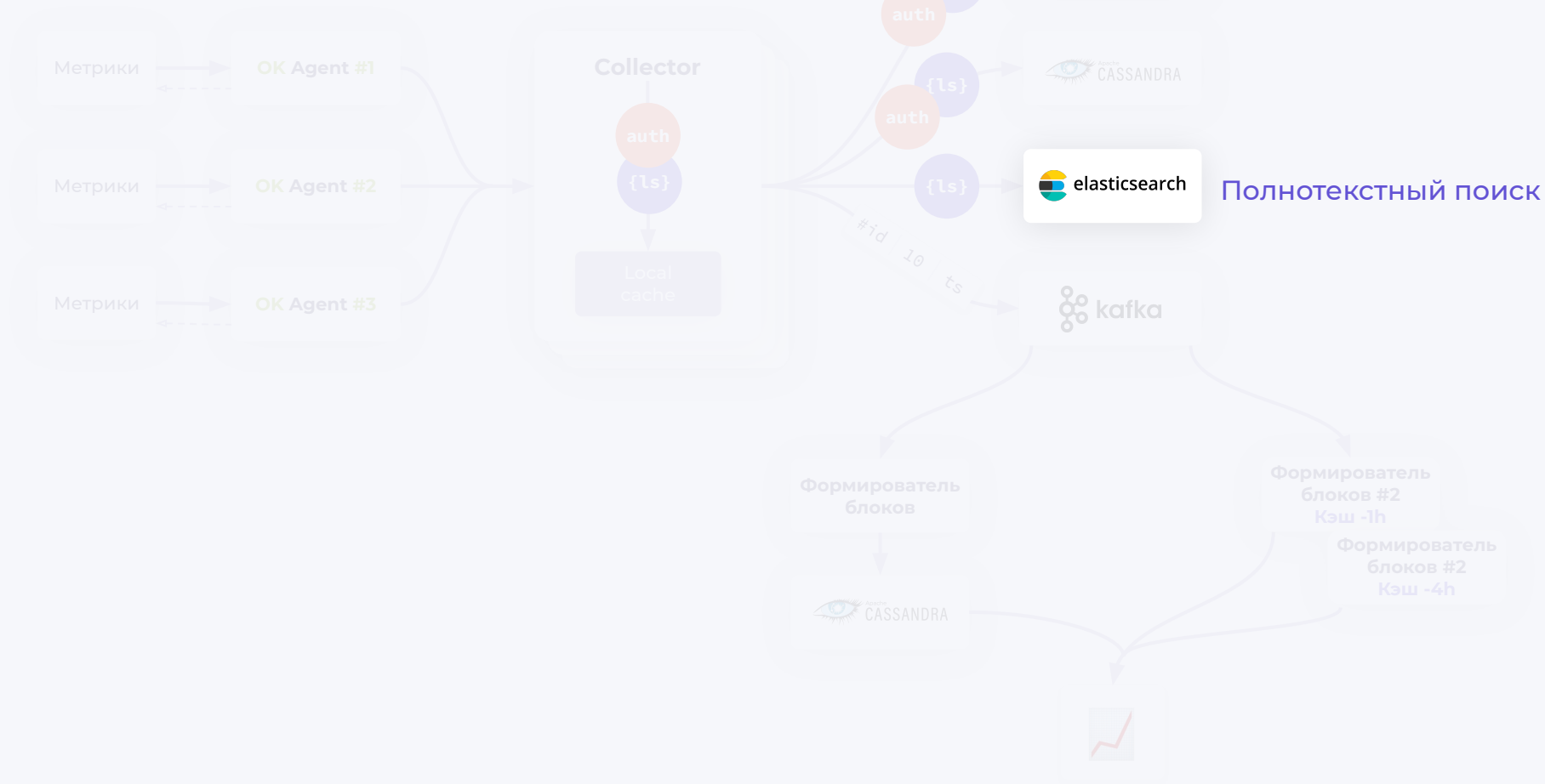
Okmeter – собственный путь



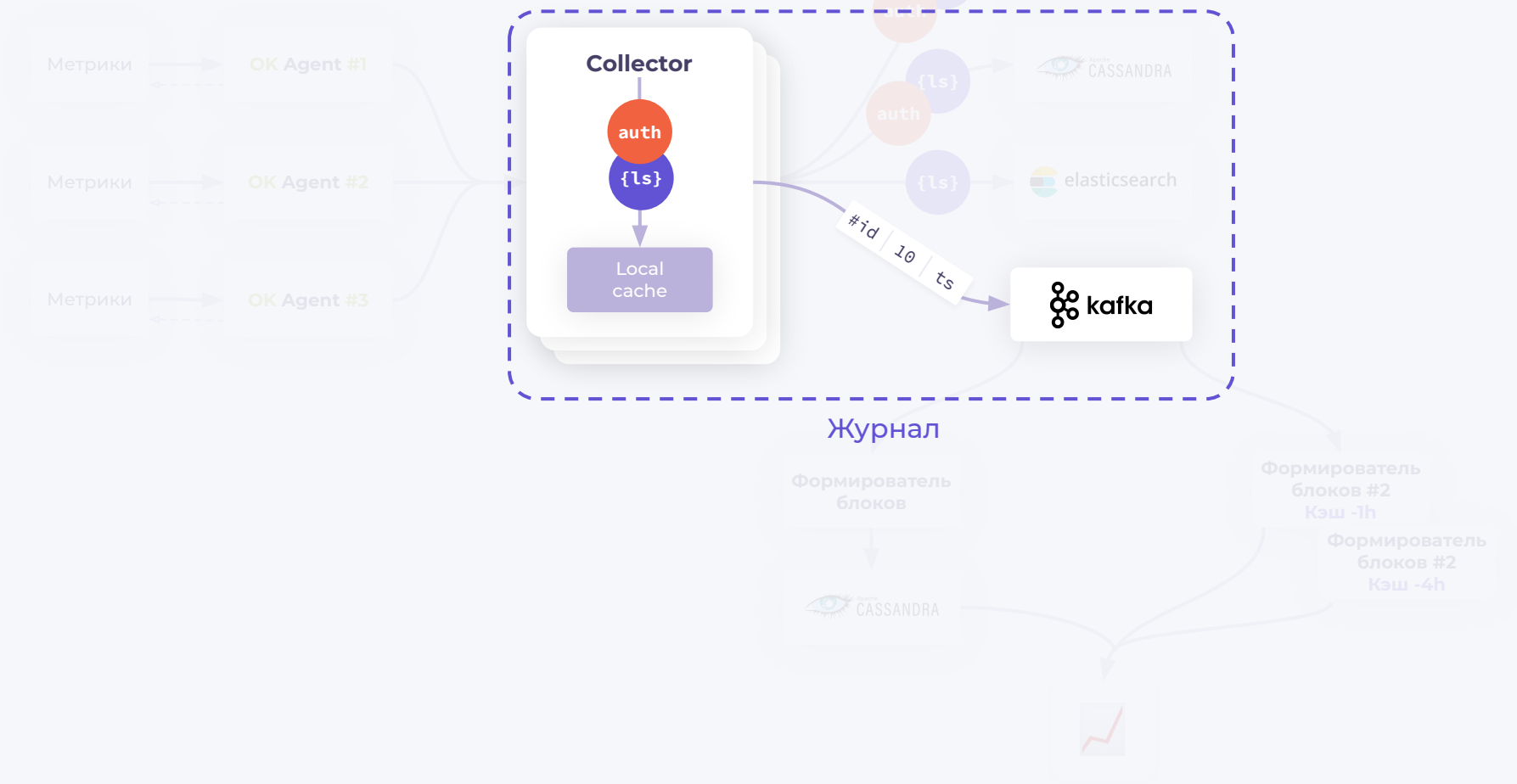
Okmeter – собственный путь



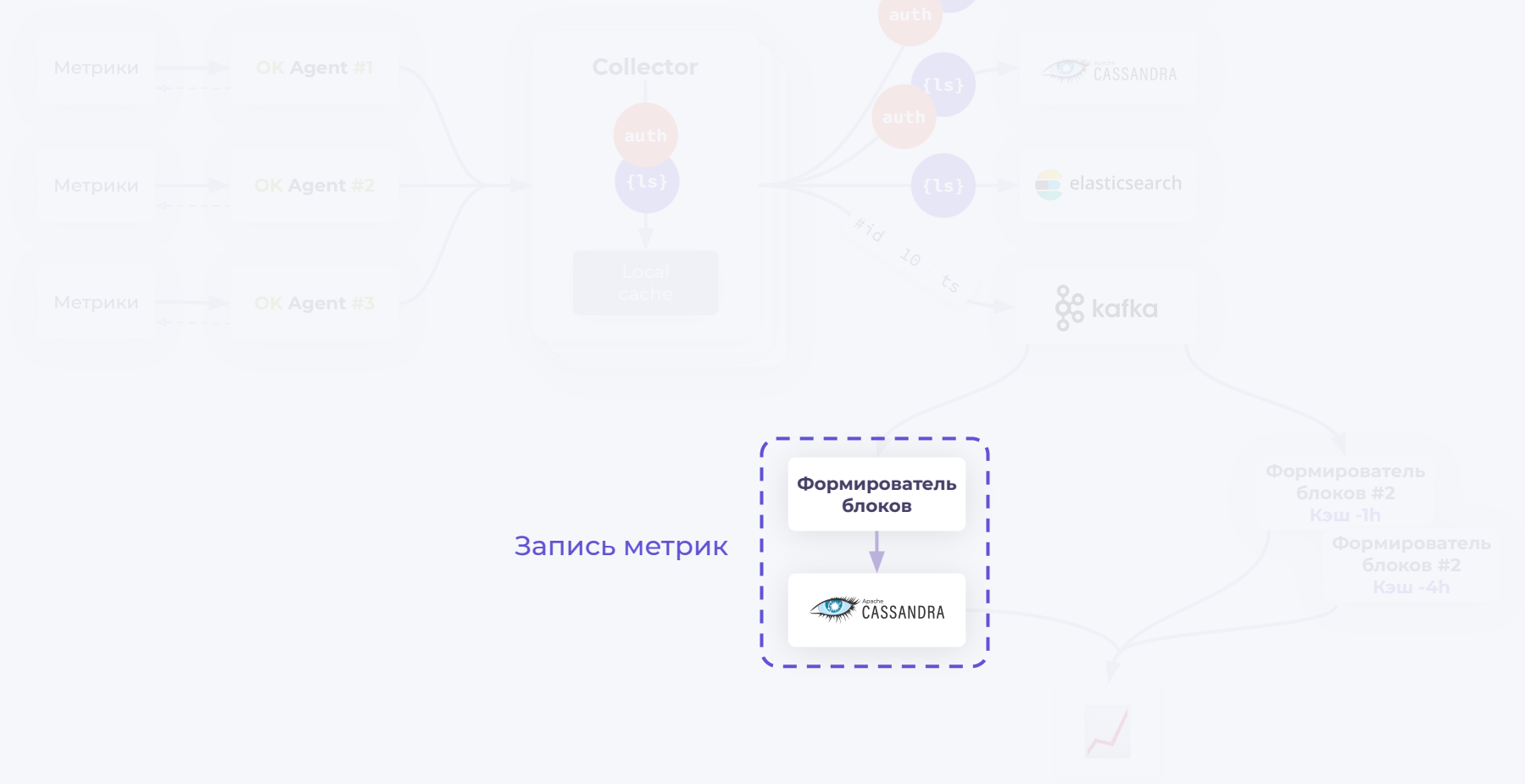
Okmeter – собственный путь



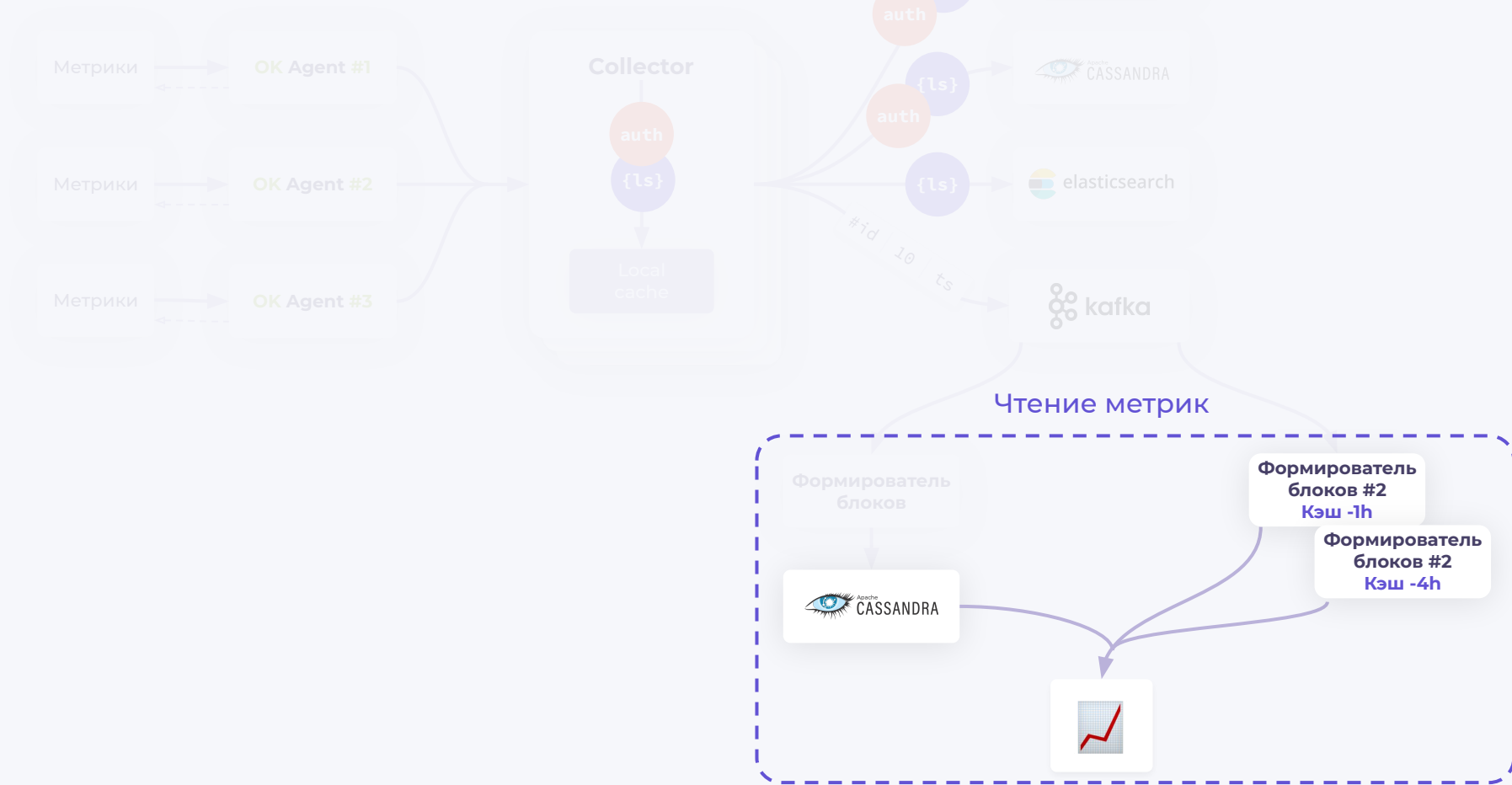
Okmeter – собственный путь



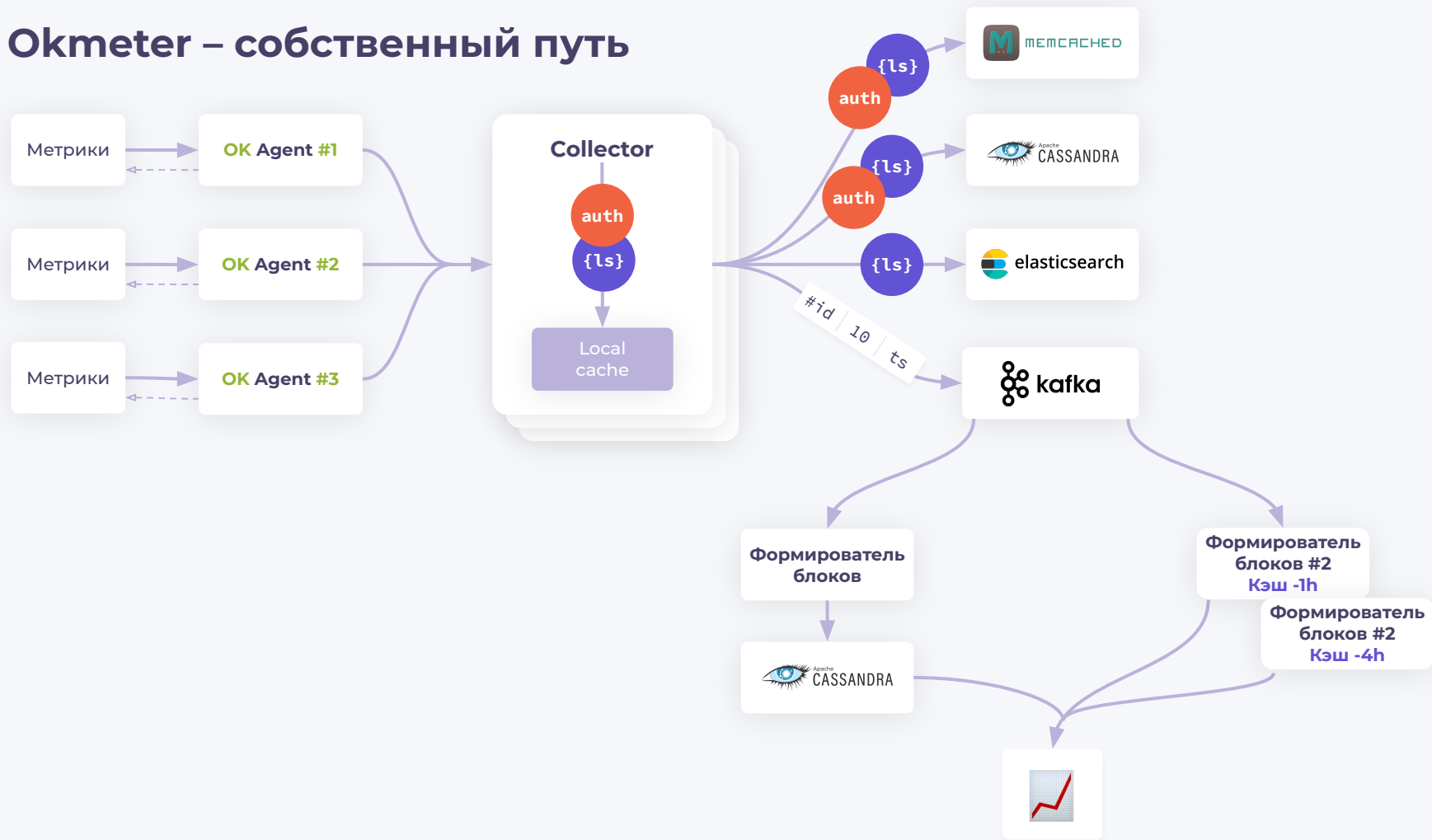
Okmeter – собственный путь



Okmeter – собственный путь



Okmeter – собственный путь



Что такое надёжность?

Что такое надежность?

- Данные не теряются

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)
- Доступны данные за 4 часа (смотрим графики)

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)
- Доступны данные за 4 часа (смотрим графики)
- Доступны исторические данные

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)
- Доступны данные за 4 часа (смотрим графики)
- Доступны исторические данные
- Низкое время ответа

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)
- Доступны данные за 4 часа (смотрим графики)
- Доступны исторические данные
- Низкое время ответа

А для этого нужно обеспечить, чтобы

- Можно много и часто выкатывать обновления
- После сбоя система восстанавливается самостоятельно
- Поддерживалась масштабируемость

Что такое надежность?

- Данные не теряются
- Доступны данные за час (получаем алерты)
- Доступны данные за 4 часа (смотрим графики)
- Доступны исторические данные
- Низкое время ответа

А для этого нужно обеспечить, чтобы

- Можно много и часто выкатывать обновления
- После сбоя система восстанавливается самостоятельно
- Поддерживалась масштабируемость
- Падения не приводят к проблемам у пользователей



Что будет, если чихнёт человек?

А-а-а-пчи!

Будь здоров!



Что будет, если чихнёт человек?



Что будет, если чихнёт **okmeter**?

А-а-а-пчхи!

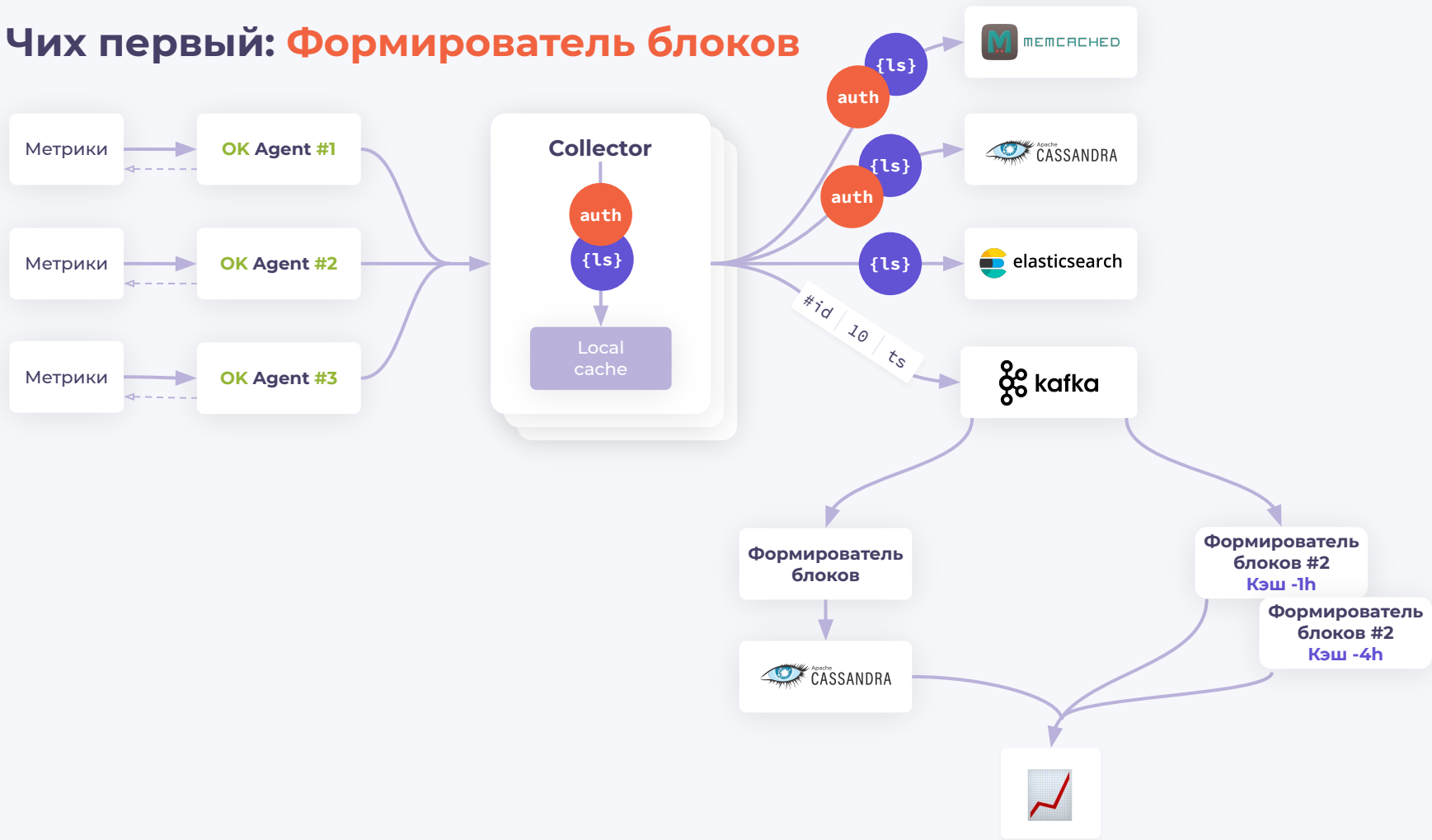
Бро... у тебя всё в порядке?

Что будет, если чихнёт okmeter?

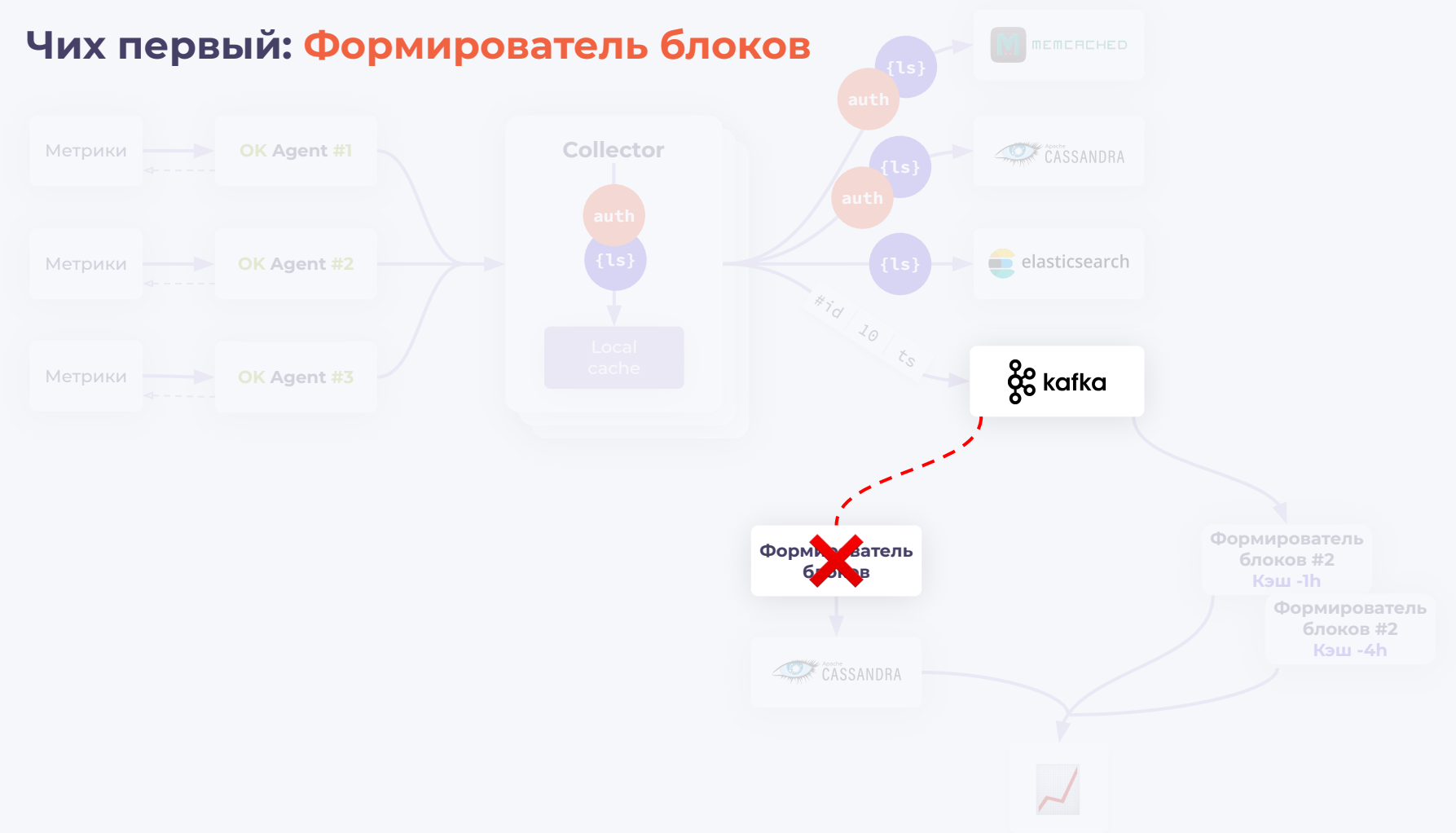


Чих первый:
Формирователь блоков

Чих первый: Формирователь блоков



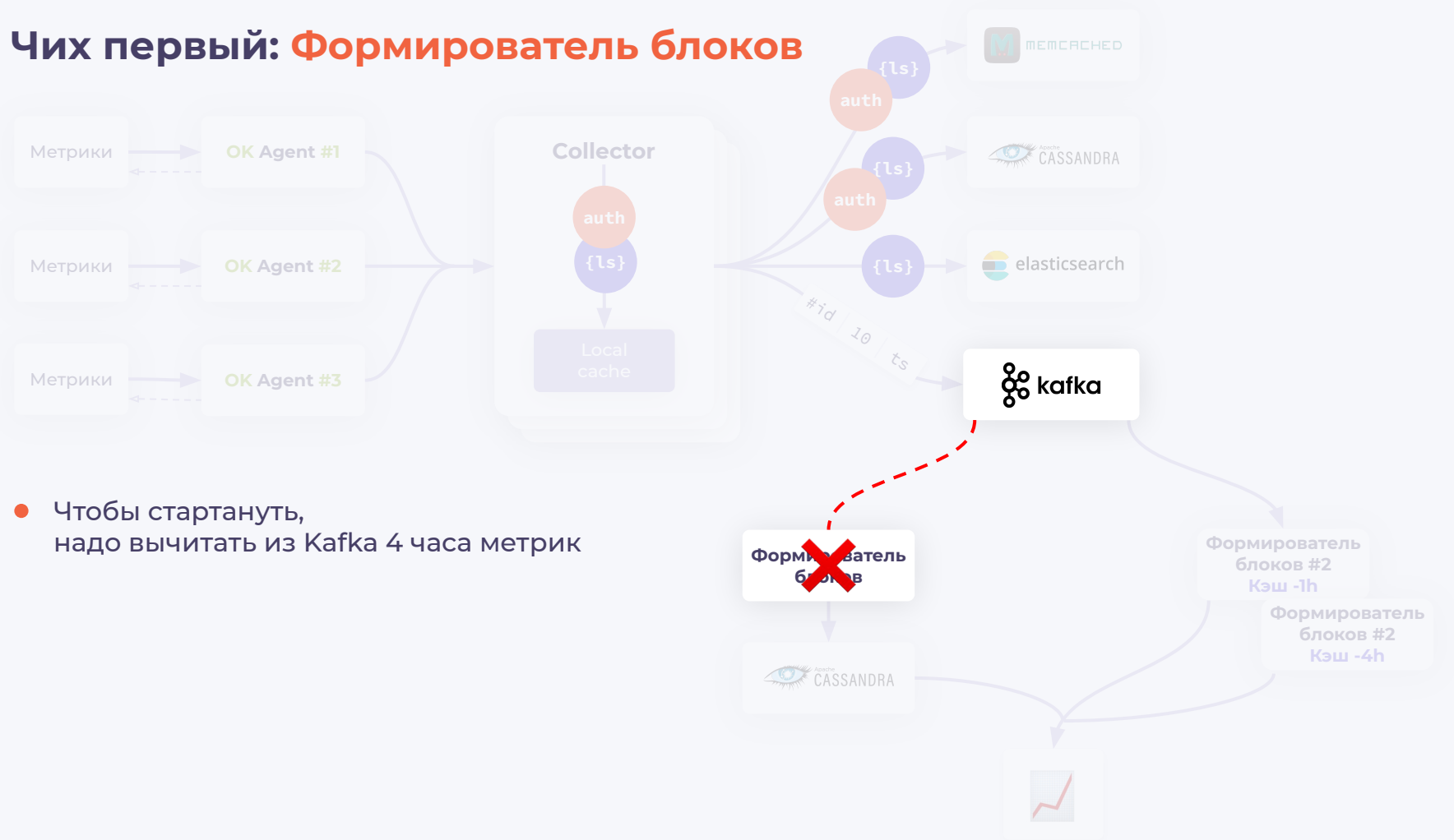
Чих первый: Формирователь блоков



Чих первый: Формирователь блоков

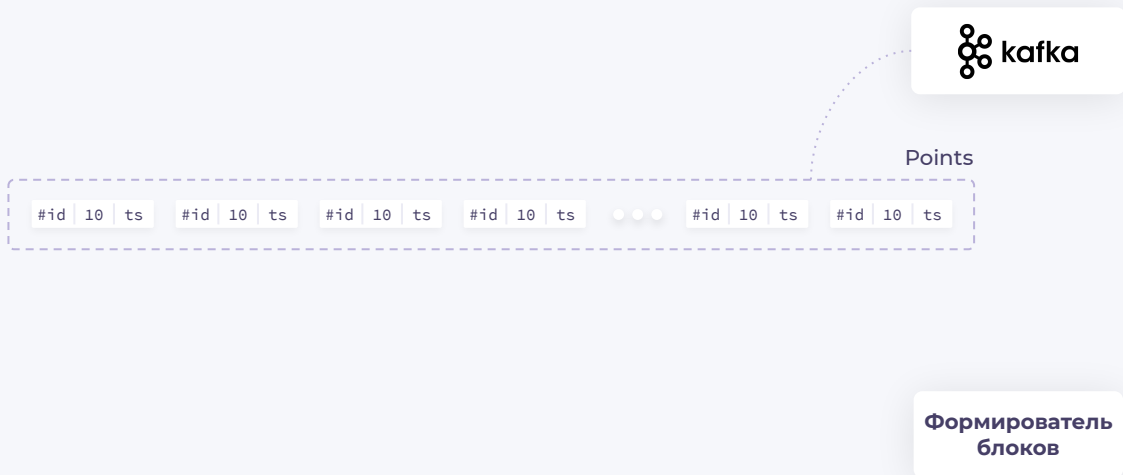


Чих первый: Формирователь блоков



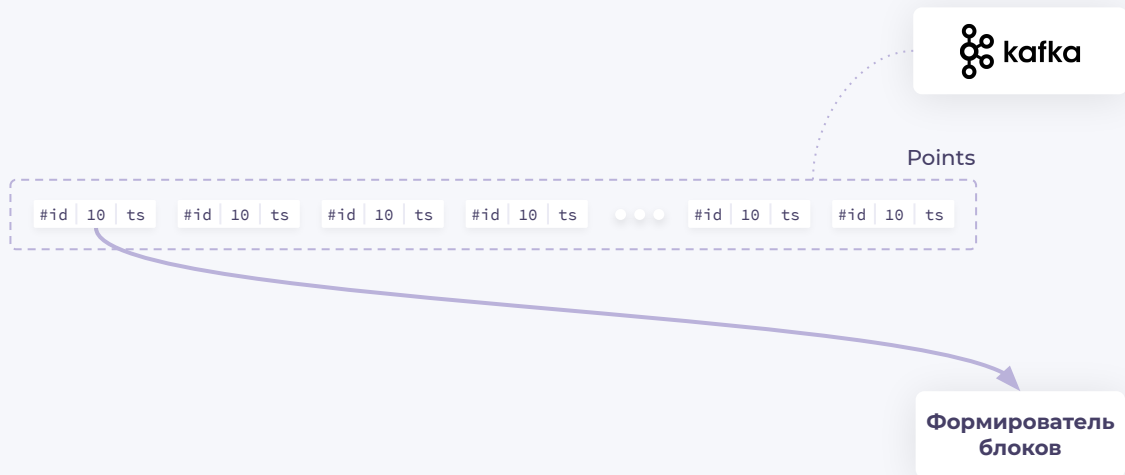
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



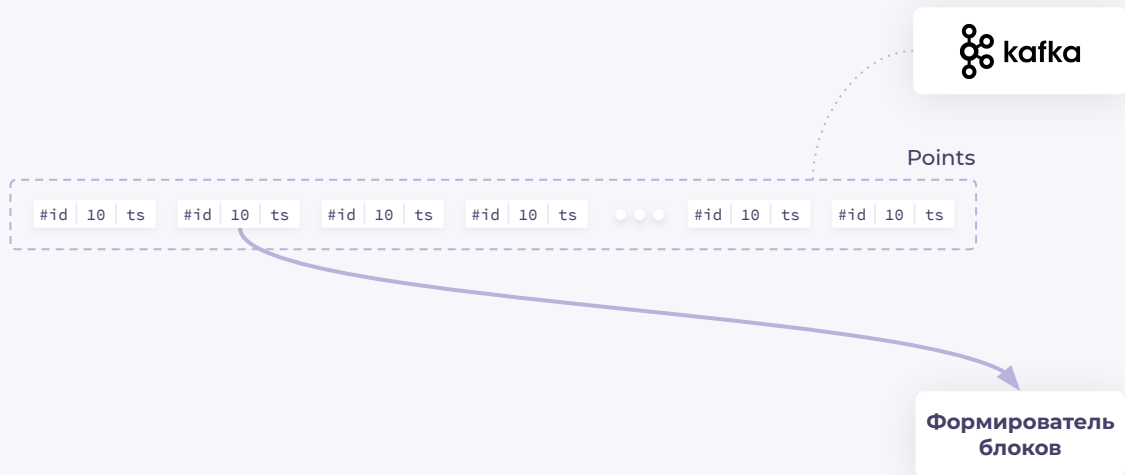
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



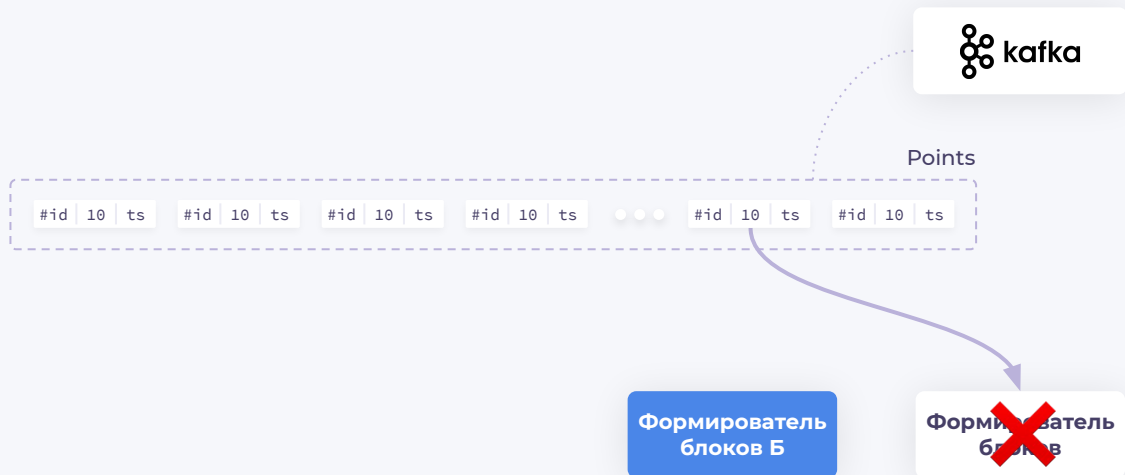
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



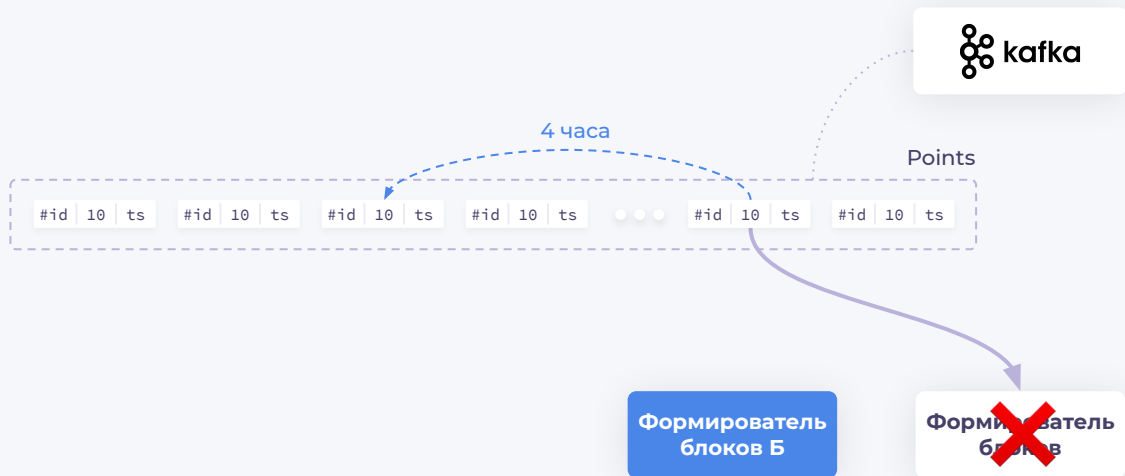
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



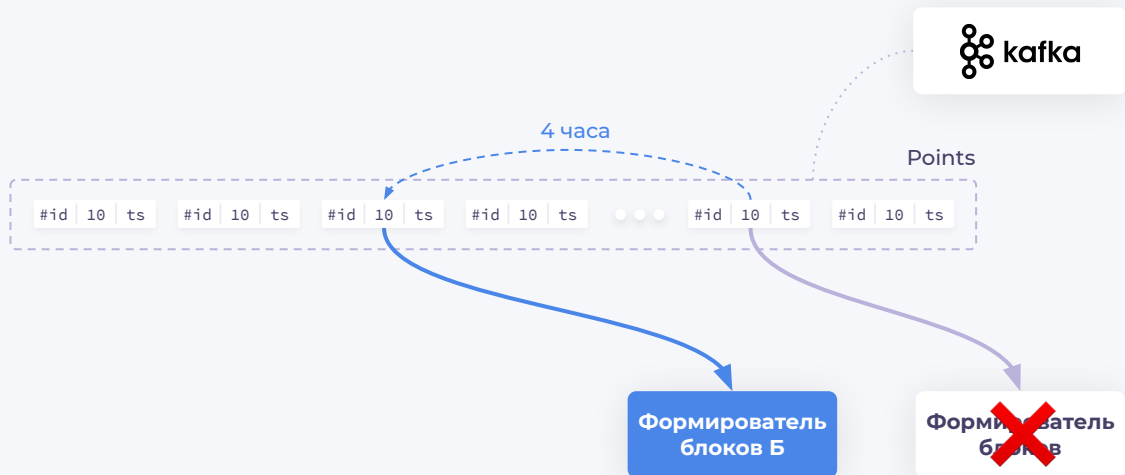
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



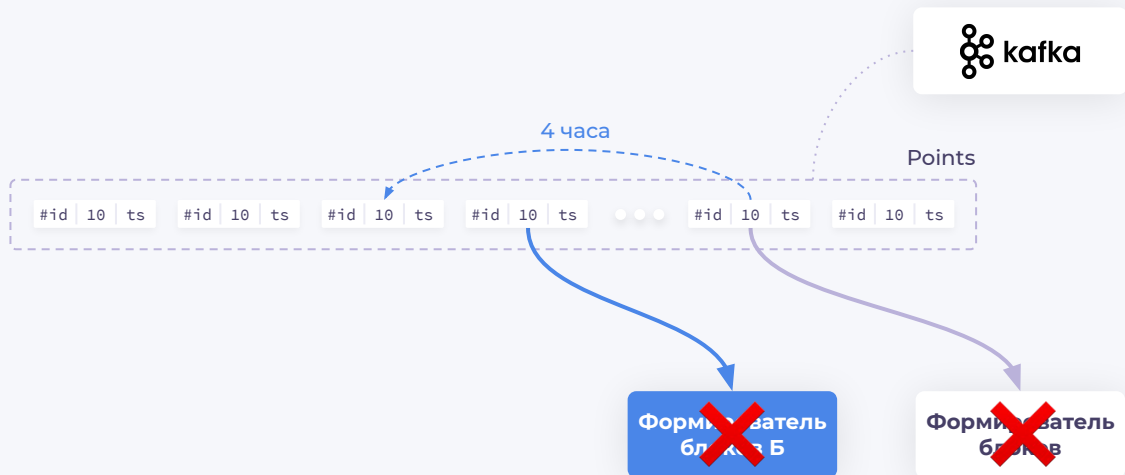
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



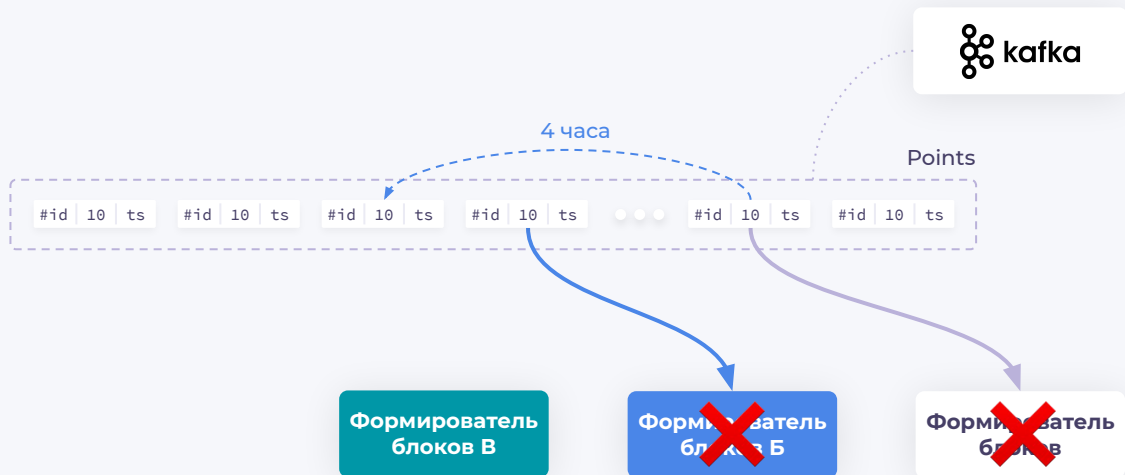
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



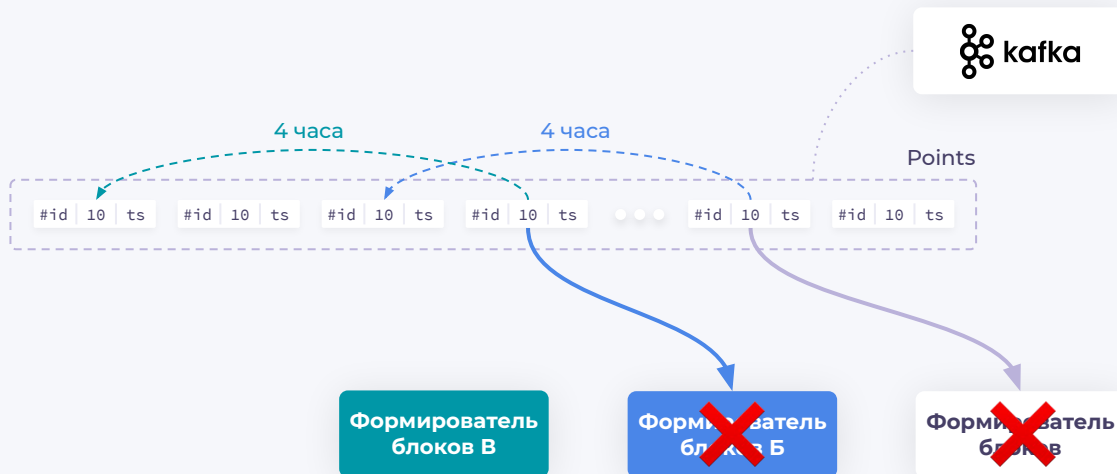
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



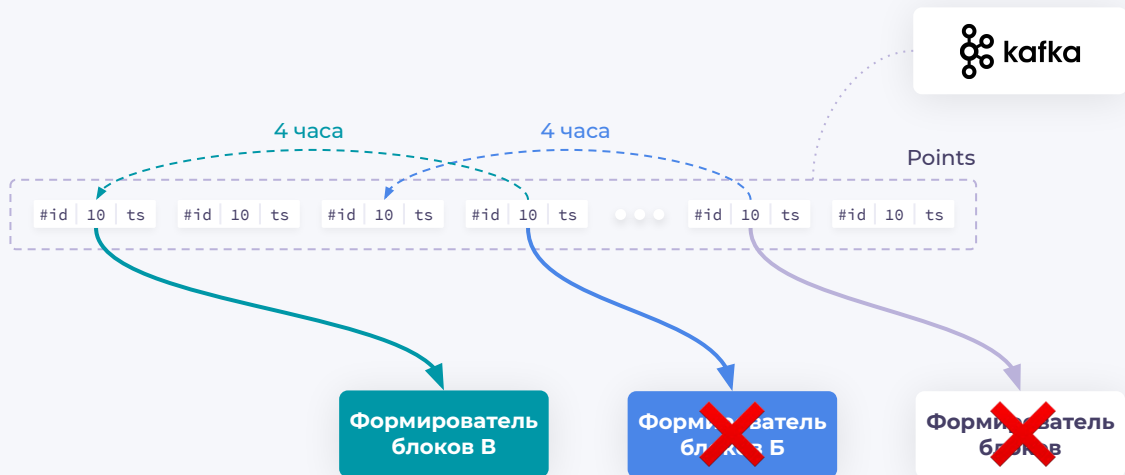
Чих первый: **Формирователь блоков**

Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



Чих первый: **Формирователь блоков**

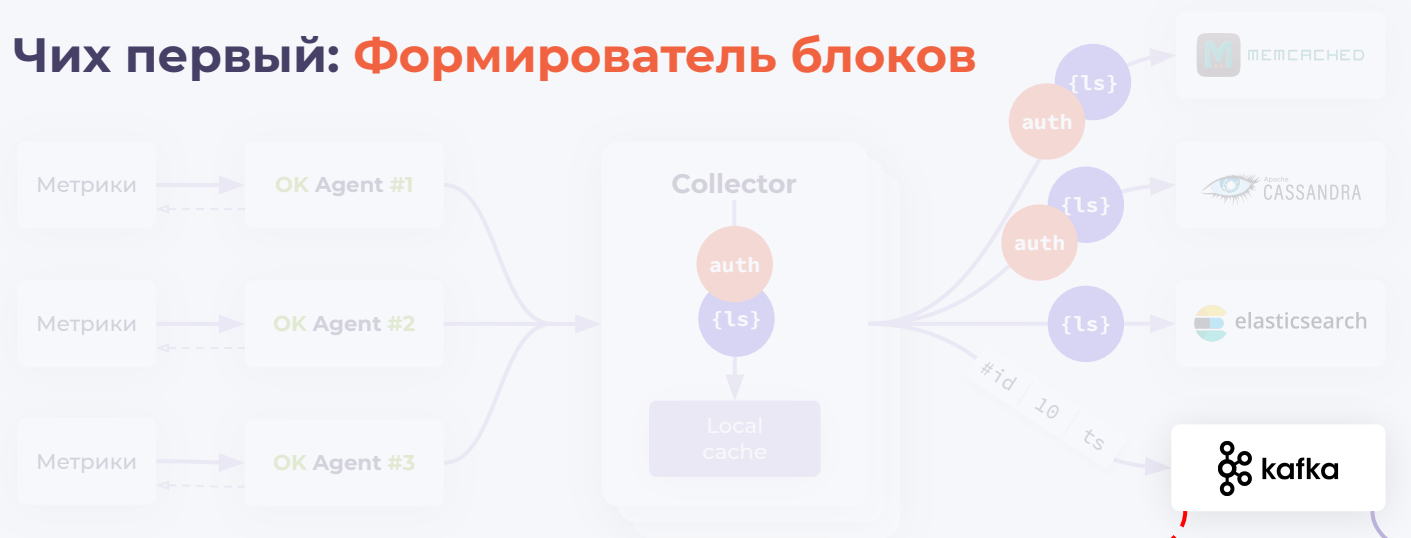
Чтобы стартануть, надо вычитать из Kafka 4 часа метрик



Чих первый: Формирователь блоков



Чих первый: Формирователь блоков



- Чтобы стартануть, надо вычитать из Kafka 4 часа метрик
- Если перезапустить несколько раз, то время каждый раз смещается на 4 часа в прошлое
- Процесс старта может занимать более 4 часов, в кэше данных **уже нет**, а в Cassandra **еще нет**

~~Формирователь
блоков~~

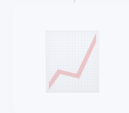


Ещё нет данных

Формирователь
блоков #2
Кэш -1h

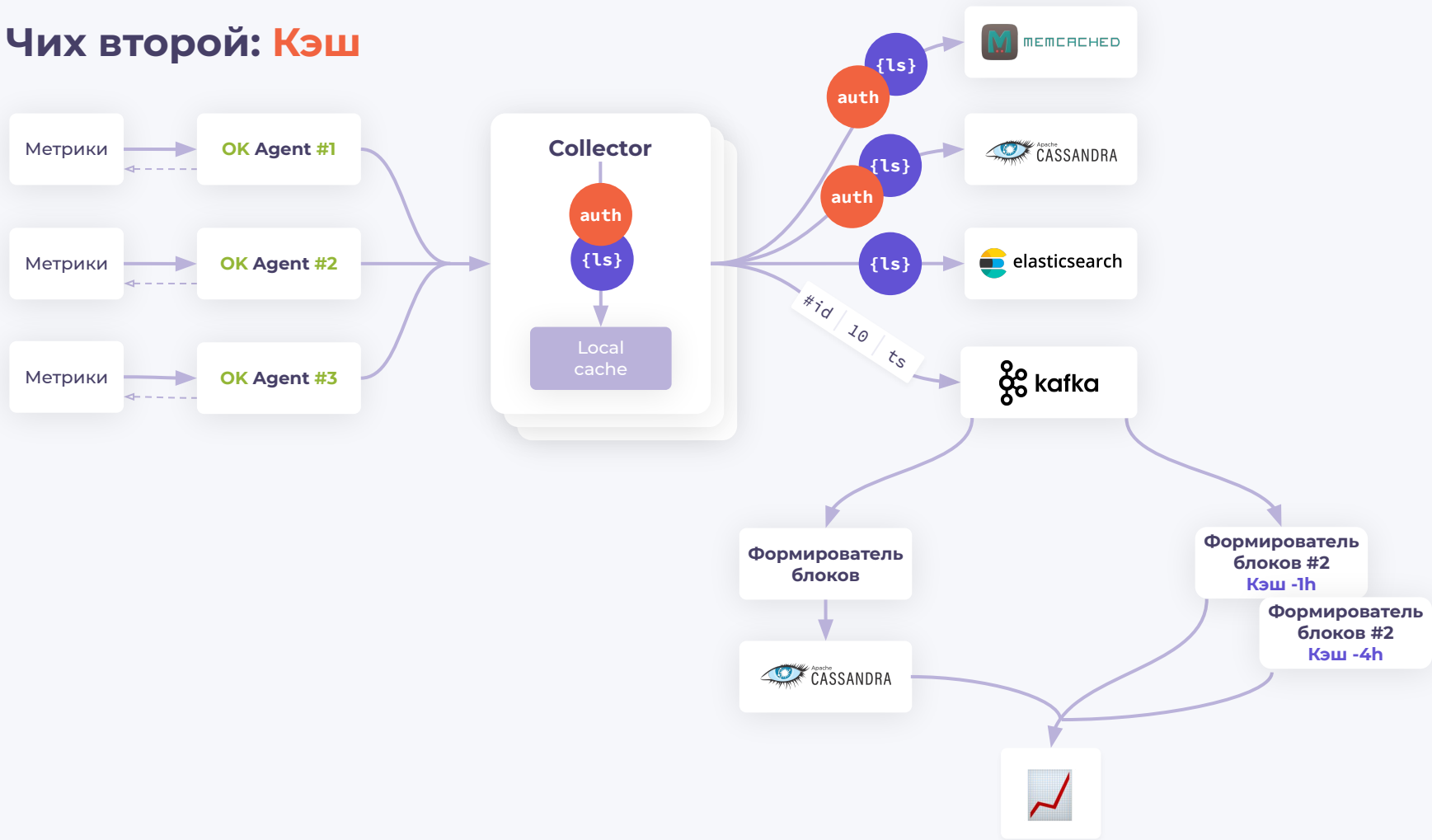
Формирователь
блоков #2
Кэш -4h

Уже нет данных

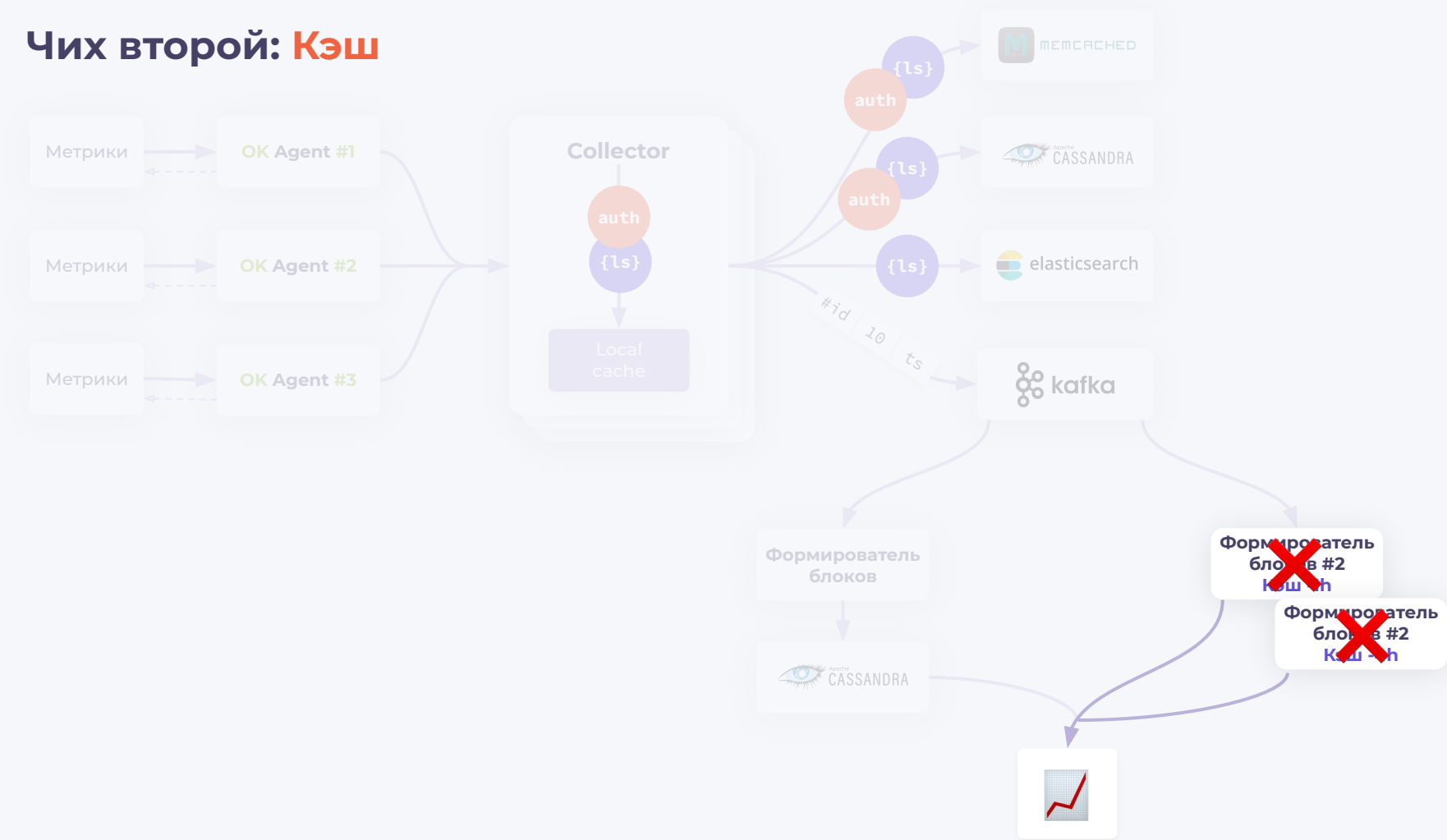


Чих второй:
Кэш

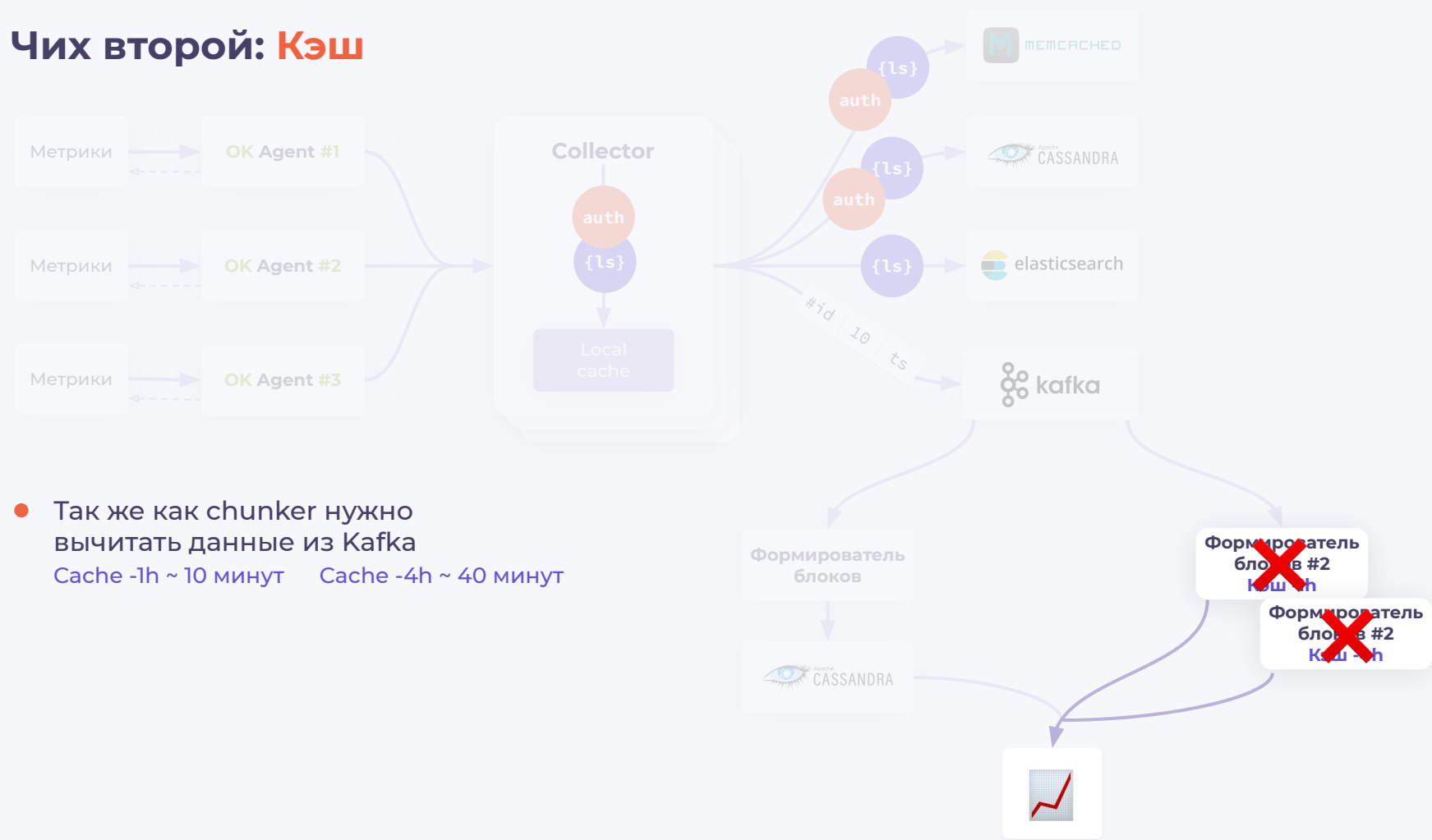
Чих второй: Кэш



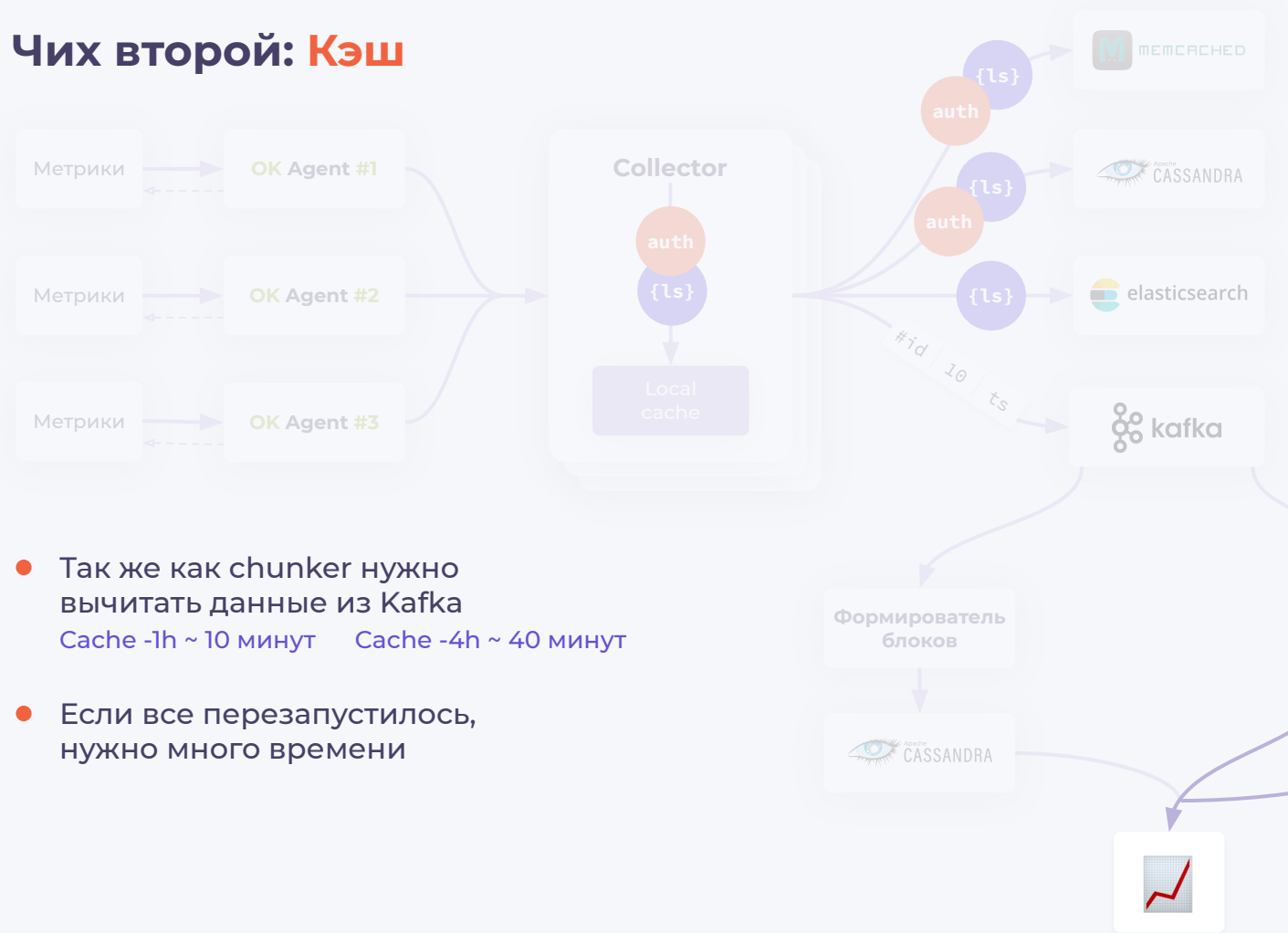
Чих второй: Кэш



Чих второй: Кэш



Чих второй: Кэш



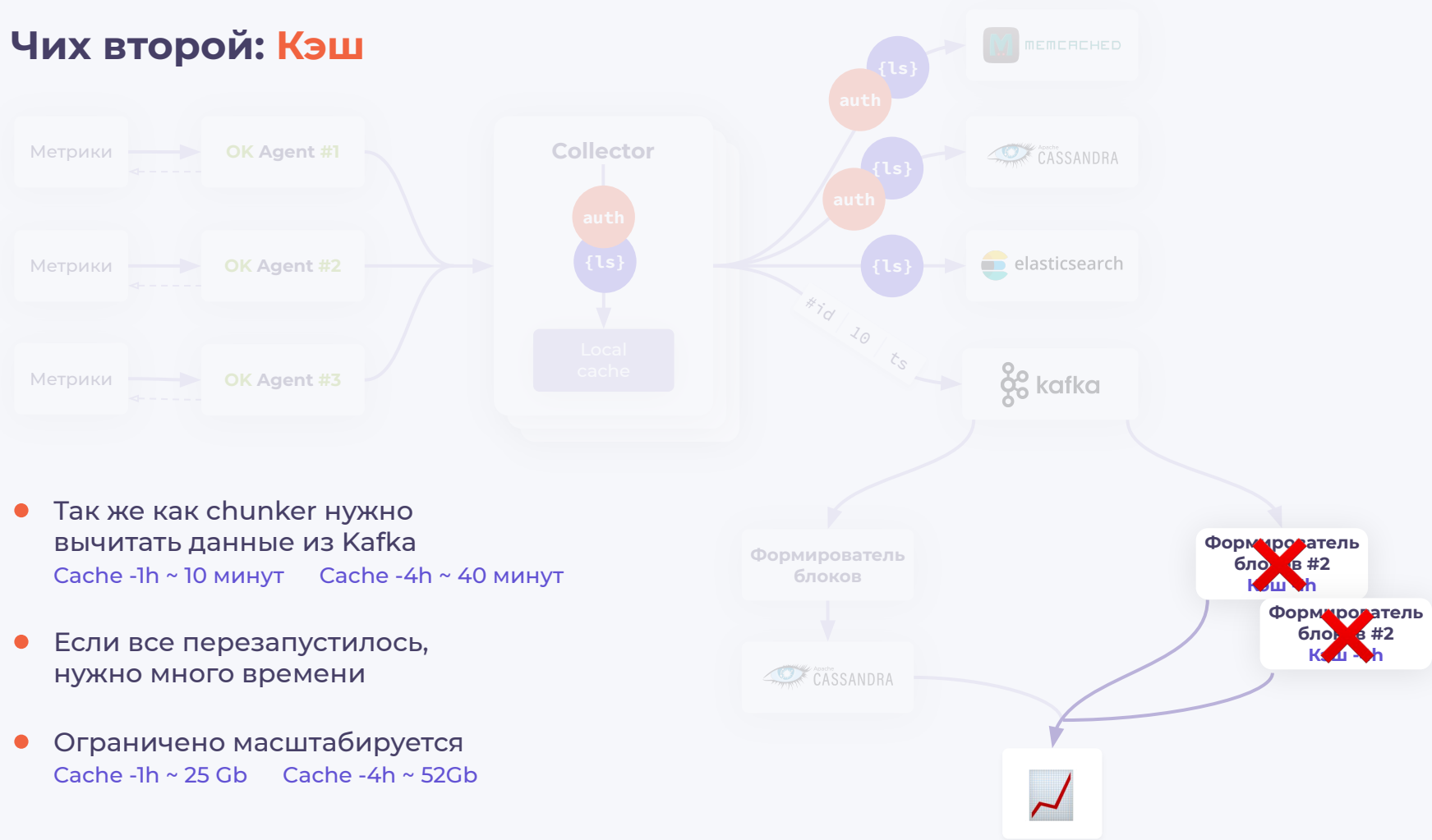
Чих второй: Кэш



- Так же как chunker нужно вычитать данные из Kafka
Cache -1h ~ 10 минут Cache -4h ~ 40 минут
- Если все перезапустилось, нужно много времени

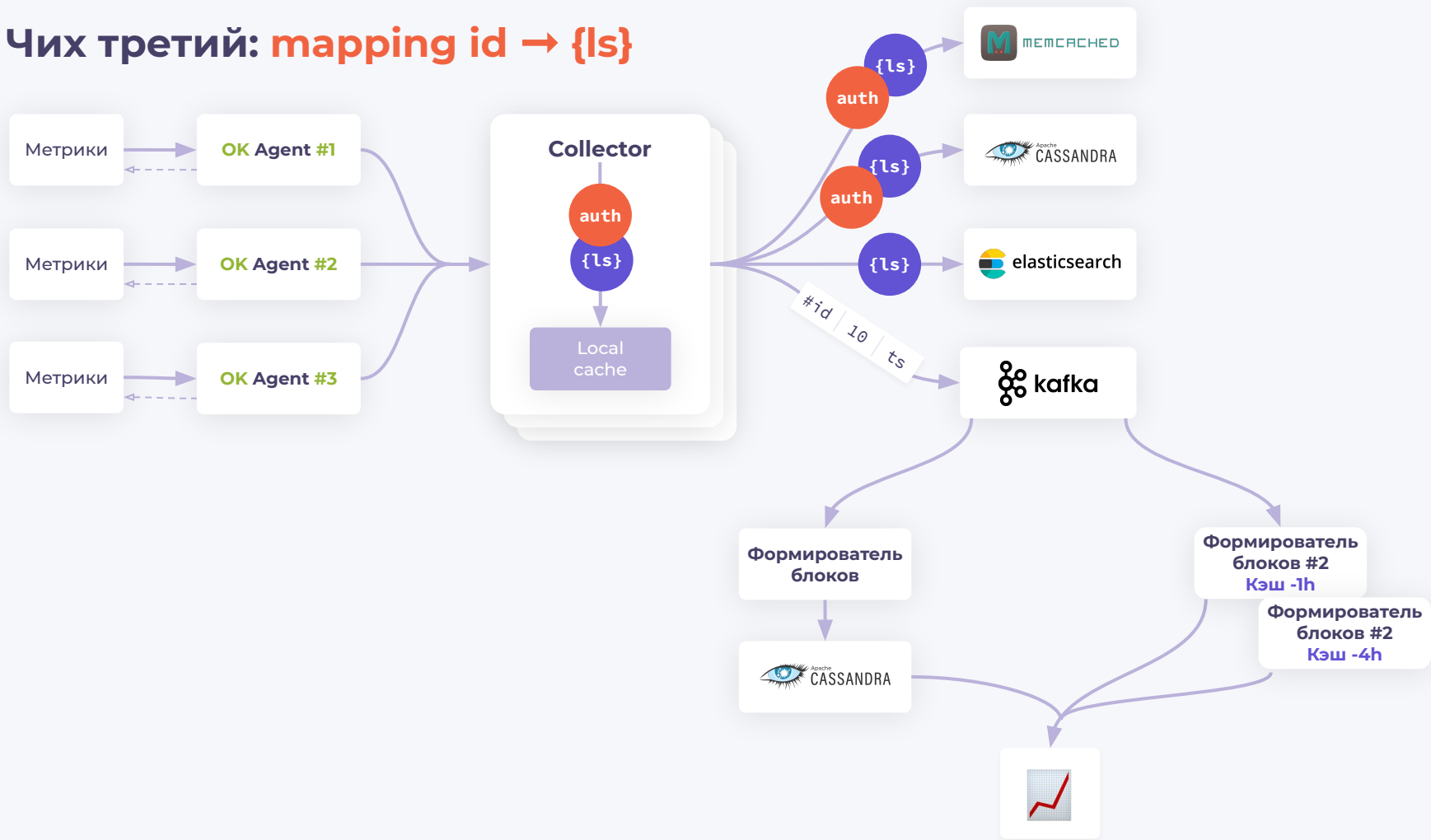


Чих второй: Кэш

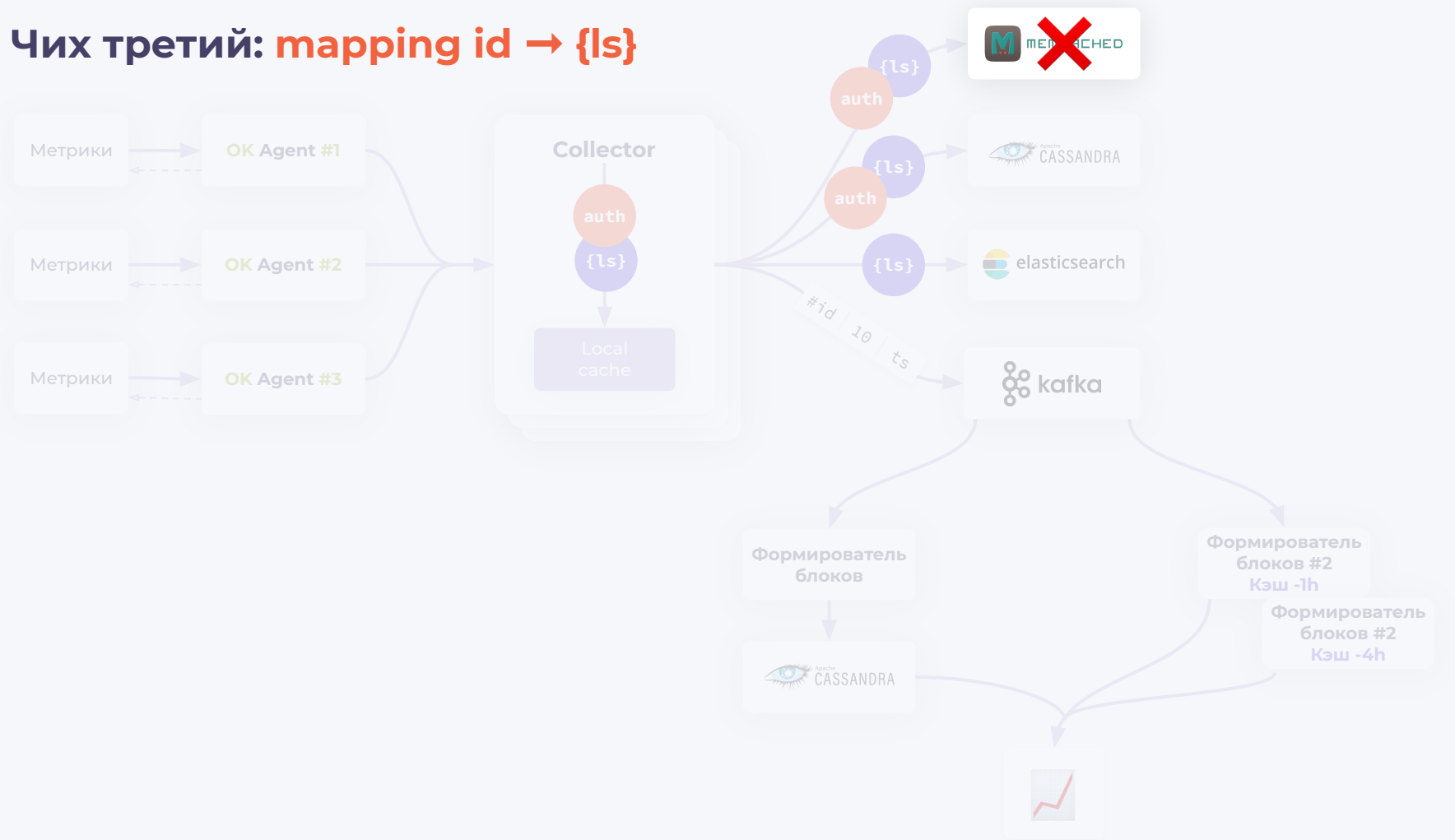


Чих третий:
mapping id $\rightarrow$ {ls}

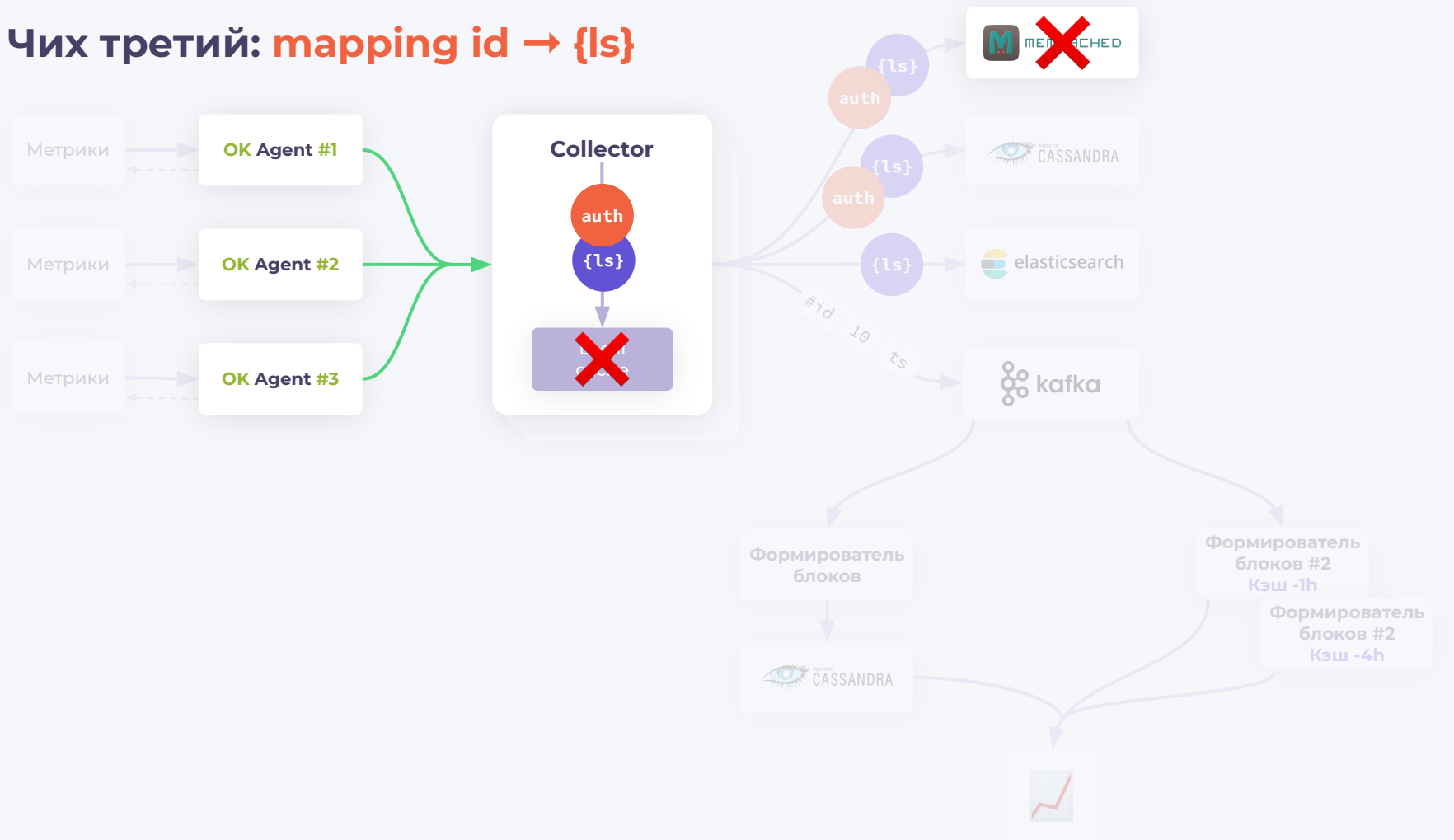
Чих третий: mapping id → {ls}



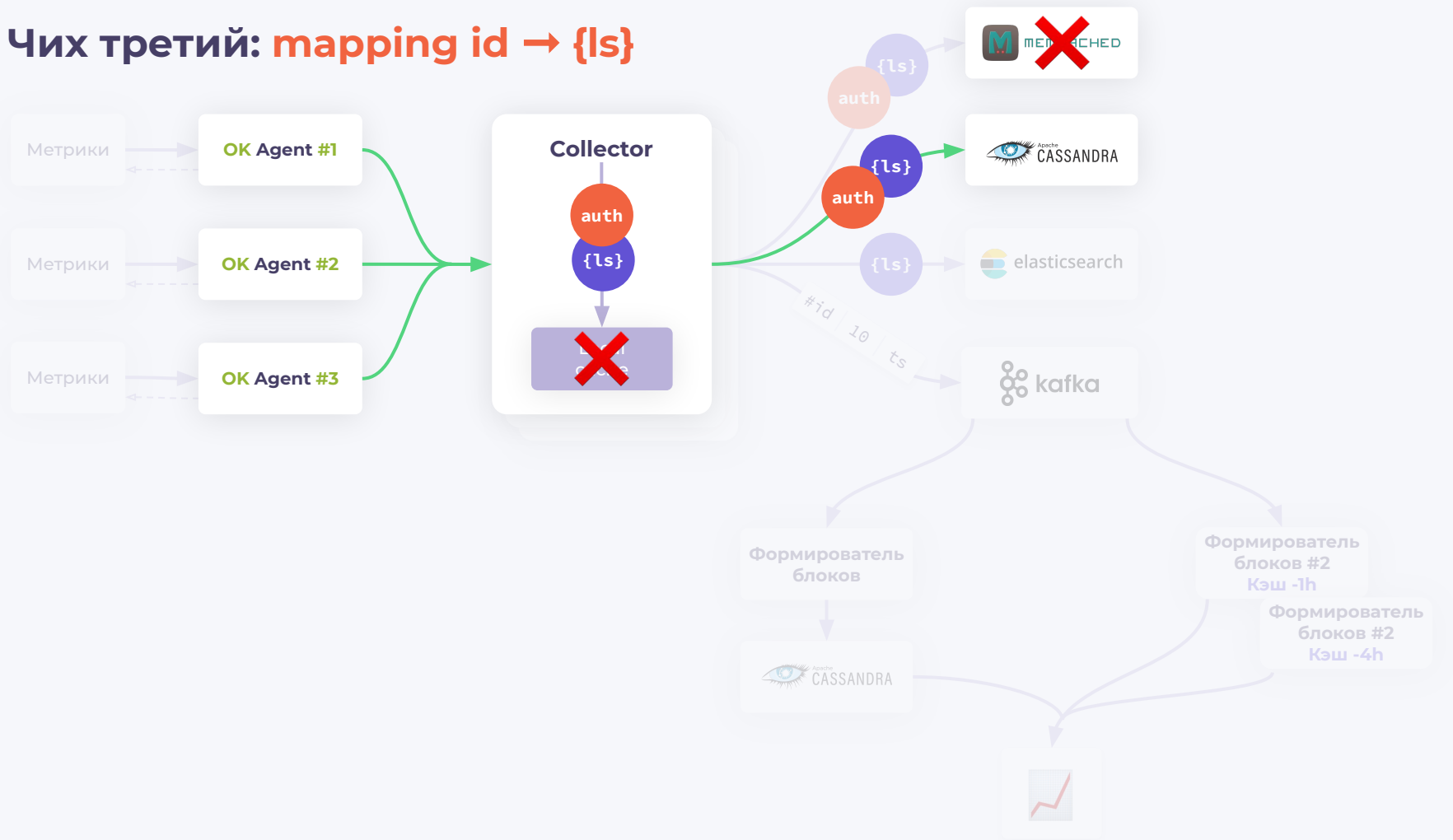
Чих третий: mapping id → {ls}



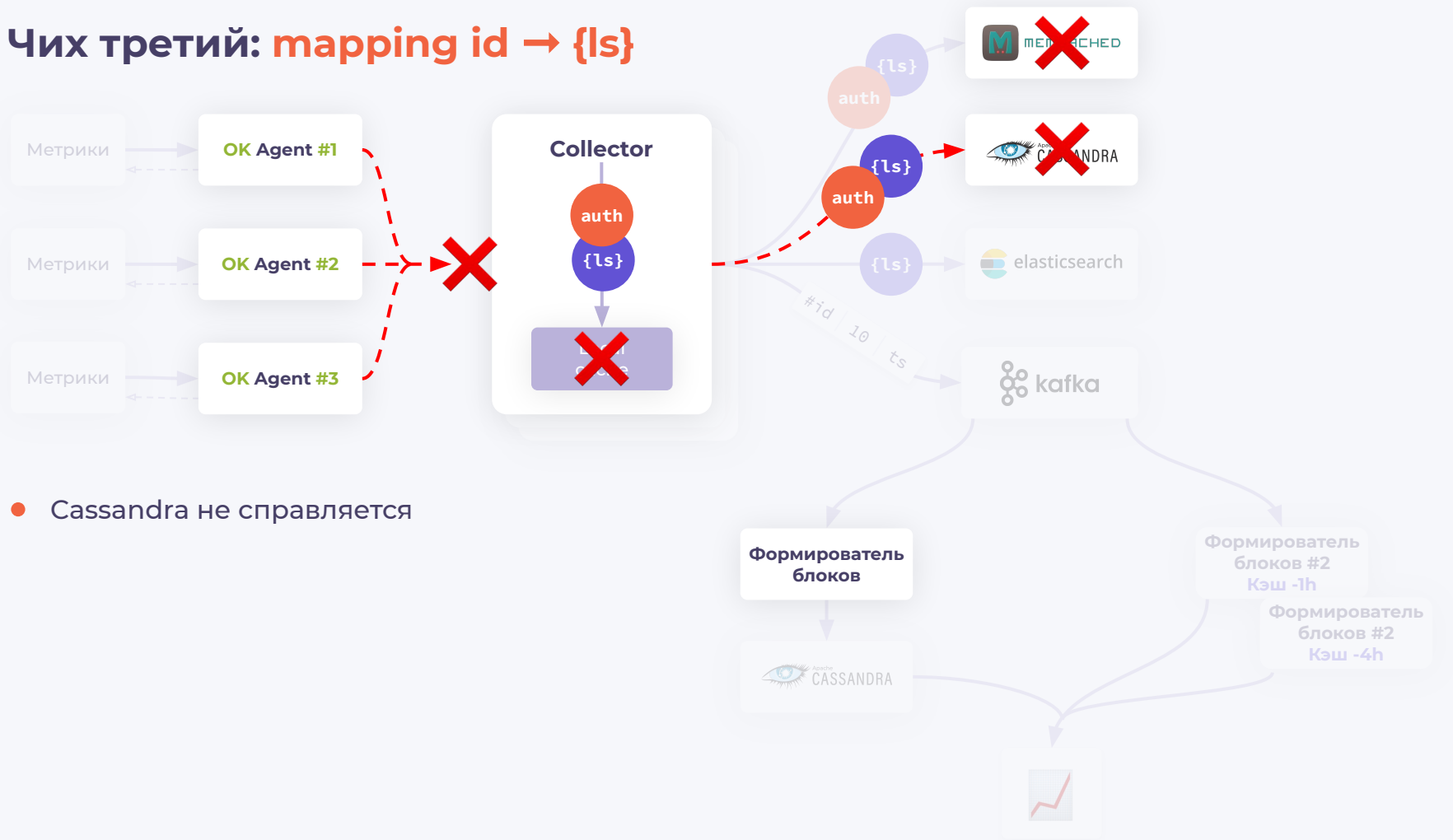
Чих третий: mapping id → {ls}



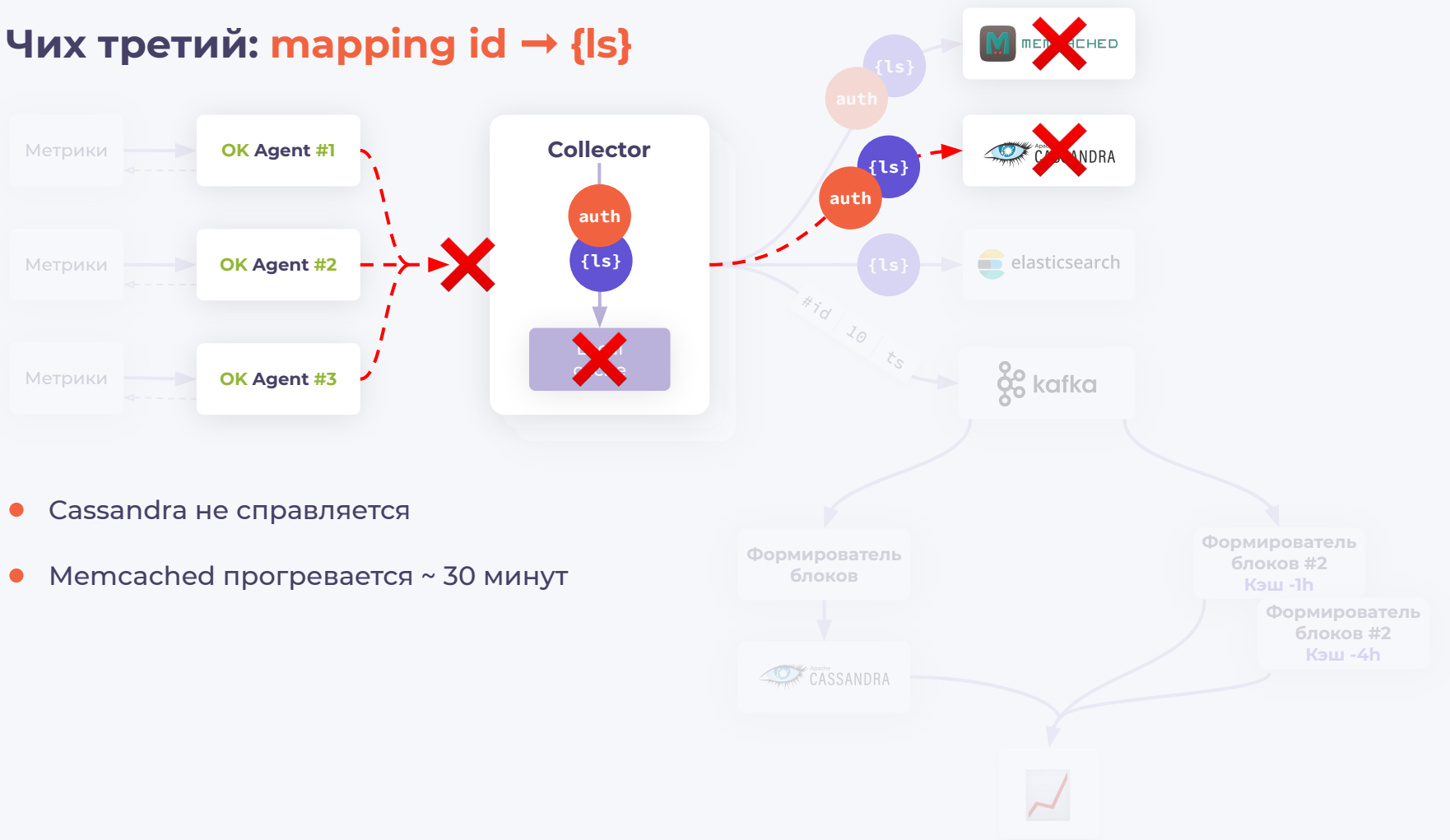
Чих третий: mapping id → {ls}



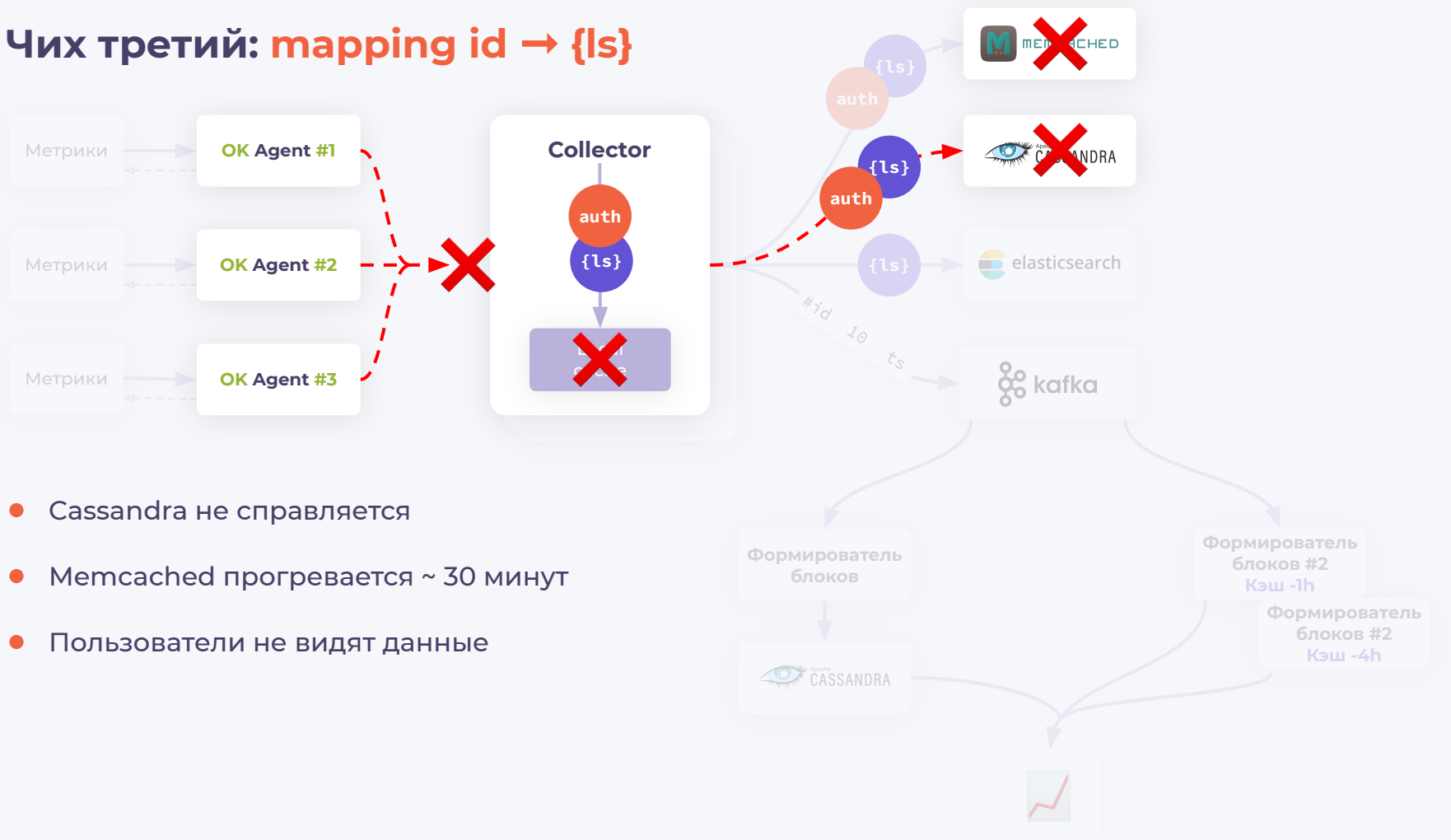
Чих третий: mapping id → {ls}



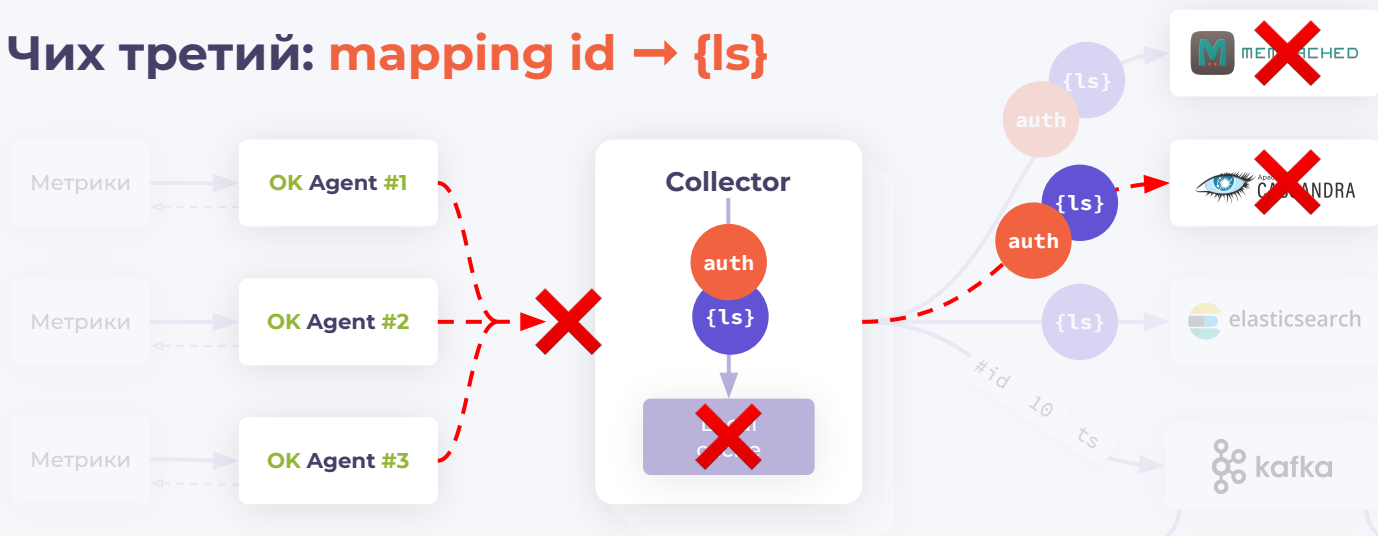
Чих третий: mapping id → {ls}



Чих третий: mapping id → {ls}



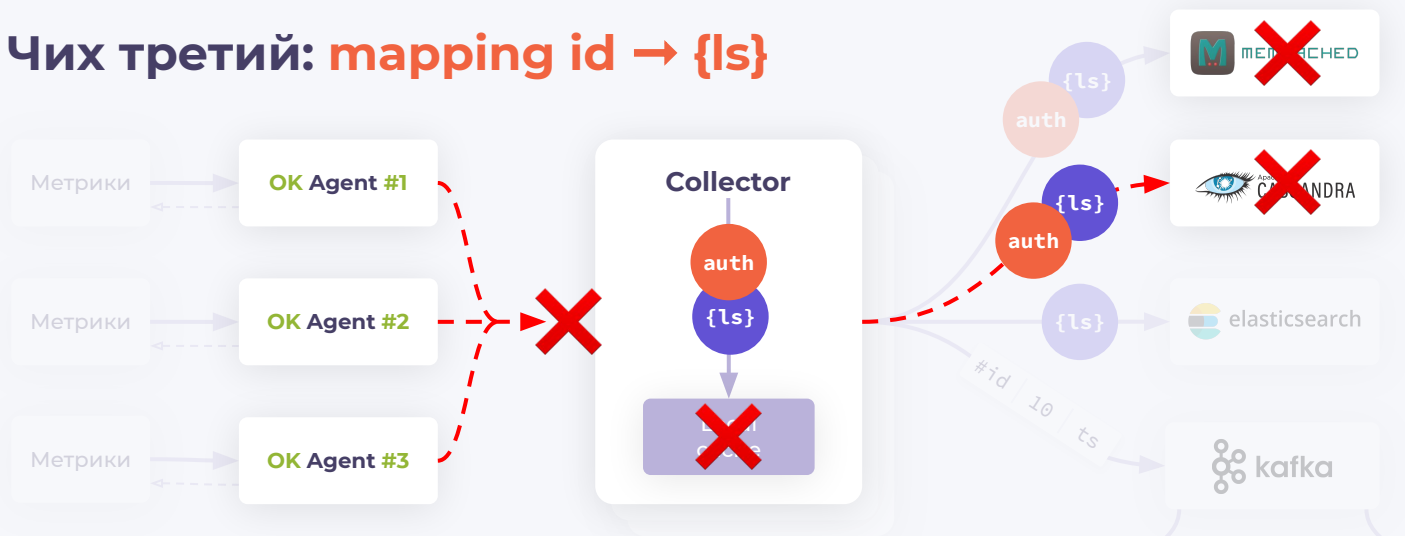
Чих третий: mapping id → {ls}



- Cassandra не справляется
- Memcached прогревается ~ 30 минут
- Пользователи не видят данные



Чих третий: mapping id → {ls}



- Cassandra не справляется
- Memcached прогревается ~ 30 минут
- Пользователи не видят данные
- В итоге все хорошо,
и исторические данные тоже доезжают

okmeter классный!
НО СЛИШКОМ МНОГО
ДВИЖУЩИХСЯ ЧАСТЕЙ



okmeter классный!
но слишком много
движущихся частей



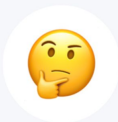
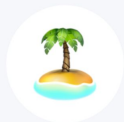
**Вводим режим
карантина!**



Как мы здесь оказались и какой у нас план?

Возможность приобрести крутой продукт

okmeter



Сложилась “база”



Уже было

200+

кластеров Kubernetes

120 млн

метрик Prometheus



В результате к нам “приехало”

23 млн

метрик

от
4400+
серверов

от
290+
проектов

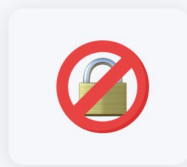
Сделать круто и расширить аудиторию



«Домашние
тапочки»



Перспективный
стек



Флант против
vendor lock-in

“Переворот” на 90°



```
{ls} → #t_id → Временной ряд
```



Okmeter – собственный путь

Архитектура и сильные стороны



Проблемы и слабые места

А-а-а-пчхи!

Будь здоров!



Что будет, если чихнёт человек?

А что будет дальше?



А что будет дальше?



Coming soon!

Продолжение у нас на YouTube

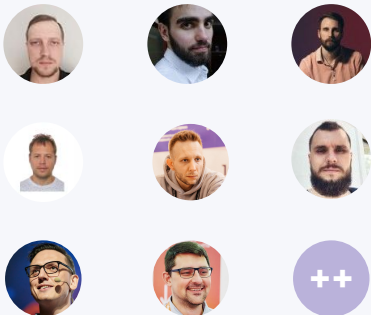
youtube.com/c/Флант



Acknowledgements

Команда okmeter

За разработку и поддержку



Антон Климов

За помощь с презентацией



*Моргни два раза, если они
держат тебя в подвале*

Флант

За текущие и будущие успехи



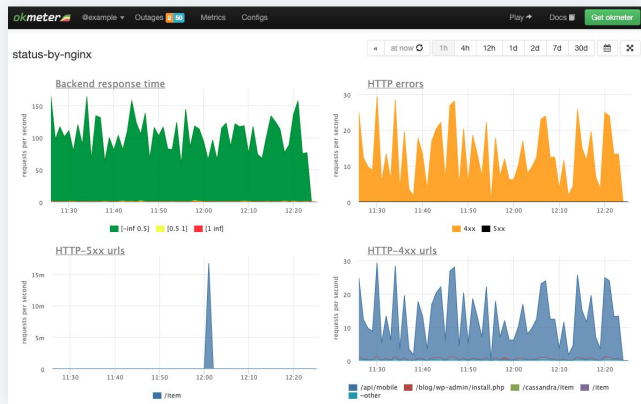
Демонстрация мониторинга Kubernetes



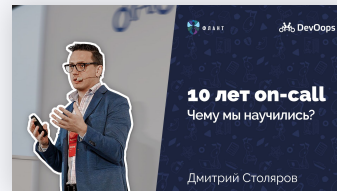
grafana-demo.deckhouse.io



Демонстрация Okmeter



okmeter.io/example



youtube.com/watch?v=7ZWGFngHSLU



youtube.com/watch?v=zi6SlzzBRaA



Спасибо!



Владимир Гурьянов

Ведущий инженер

 vladimir.guryanov@flant.com

 t.me/Magvai69



Андрей Колаштов

Руководитель продукта okmeter

 andrey.kolashtov@flant.com

 t.me/kolashtov



Дмитрий Столяров

CTO & Co-founder

 dmitry.stolyarov@flant.com

 t.me/distol

okmeter

Будь готов к любым сбоям на серверах

Очень детализированный мониторинг

okmeter.io

Демо-версия: okmeter.io/example



Флант

DevOps и Kubernetes, обслуживание 24×7

 flant.ru

 t.me/flant_ru

 youtube.com/c/Флант

 habr.com/ru/company/flant