

Дашборд, которому можно верить

Очередной рассказ
об ускорении сборок



Шико Мстоян

~10 лет в разработке — из них

7 мес X5 Tech

мобильная инфраструктура

2 года Яндекс Еда

мобильная инфраструктура

5 лет Яндекс Маркет

команда промо-механик

X5 Tech / MOBIUS 2026 AUTUMN



О чём буду рассказывать

- | | | |
|----|--------------------------|---|
| 01 | Инструмент | что выбрали и как внедряли |
| 02 | Сначала починили графики | как метрики обманывали нас самих и как мы научили их говорить правду |
| 03 | Что оптимизировали | неработающий Configuration Cache, медленный синк и разработчики на старом develop |
| 04 | Итоги | что не сработало, сколько сэкономили и что бы я сделал по-другому |

С чего начинали

01

BuildMetrics — готовый Gradle-плагин метрик, взяли за основу

внутренняя разработка X5 Tech,
автор — Кирилл Старостин



02

**14 репозиториев
мобильной разработки**

03

**Mac mini 2018 года
на Intel**

04

**Много
энтузиазма**



Почему именно BuildMetrics?

А лично для меня у всего
остального софта один
фатальный недостаток —
его разрабатываем не мы

Подробно —
в докладе



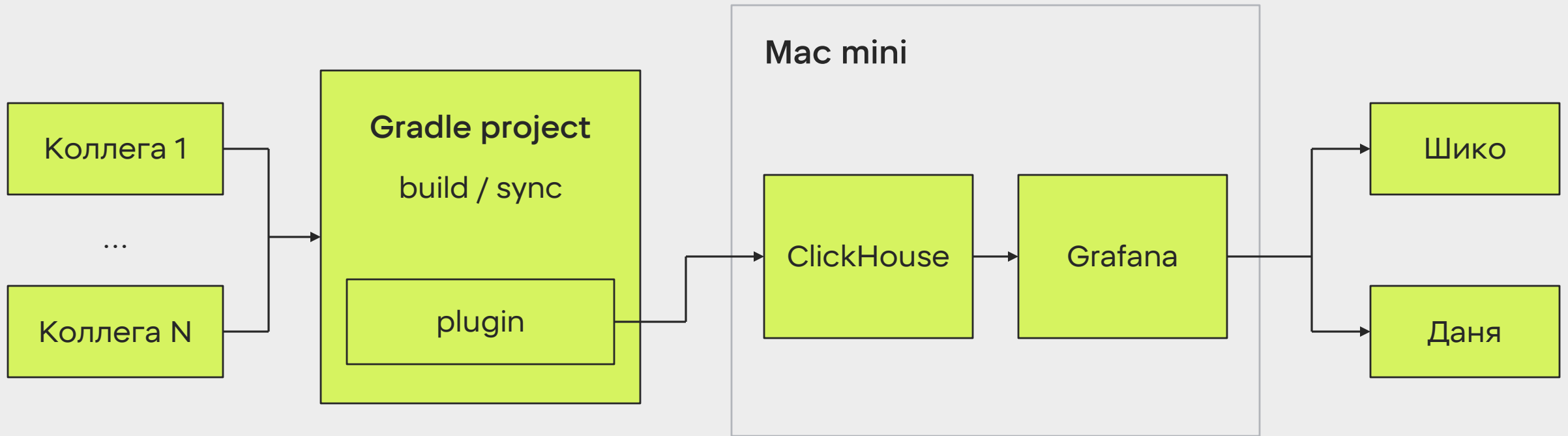
Кирилл Старостин

Мониторим здоровье Gradle-сборки
KMP проекта



Как все работает?

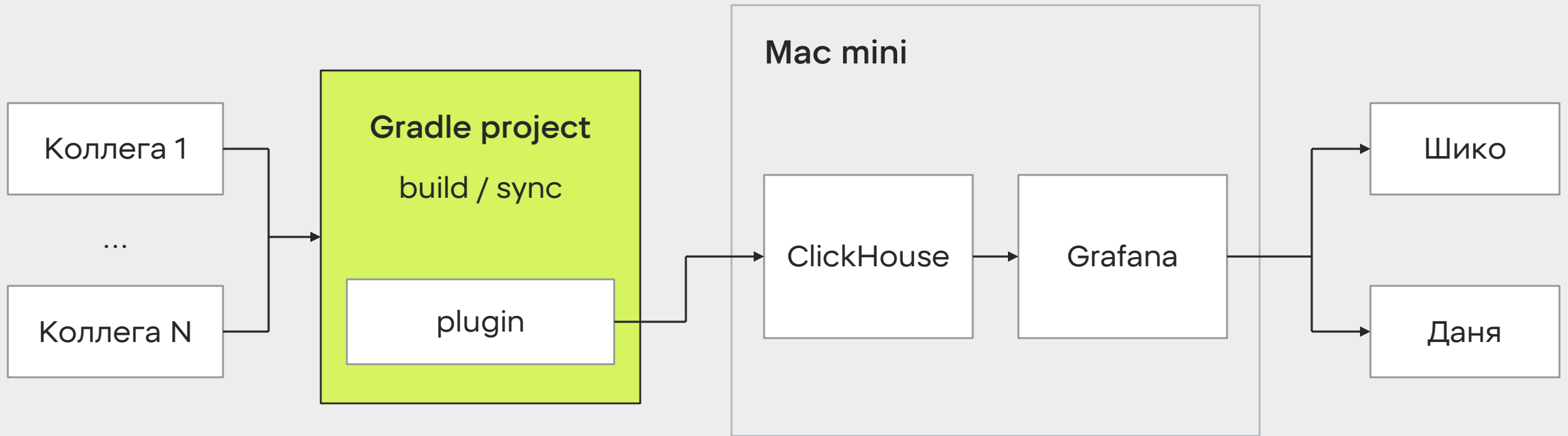
06



Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как все работает?

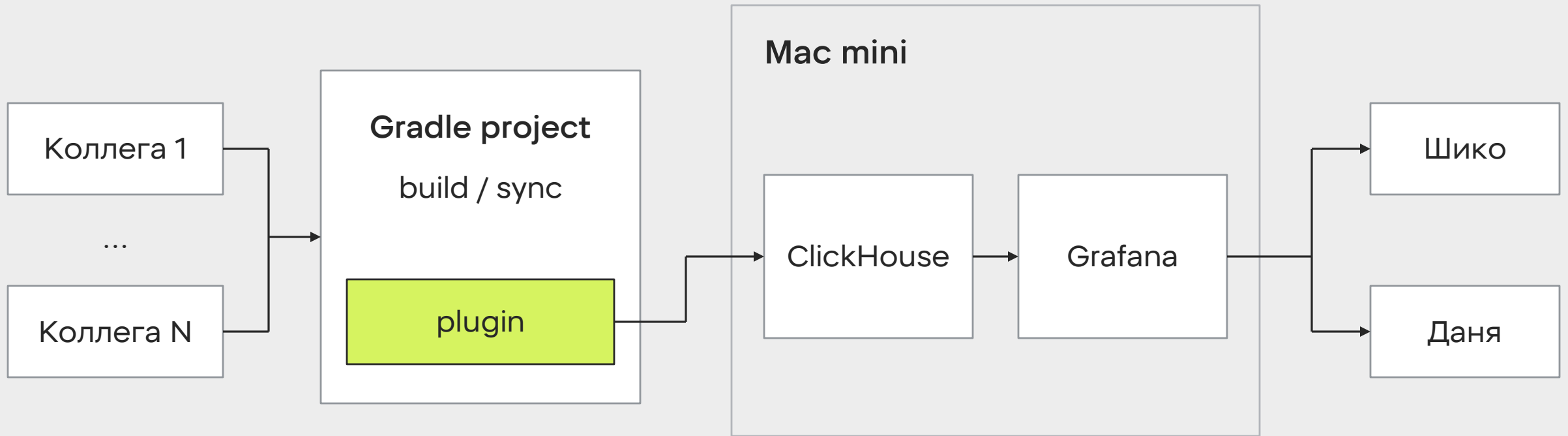
07



Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как все работает?

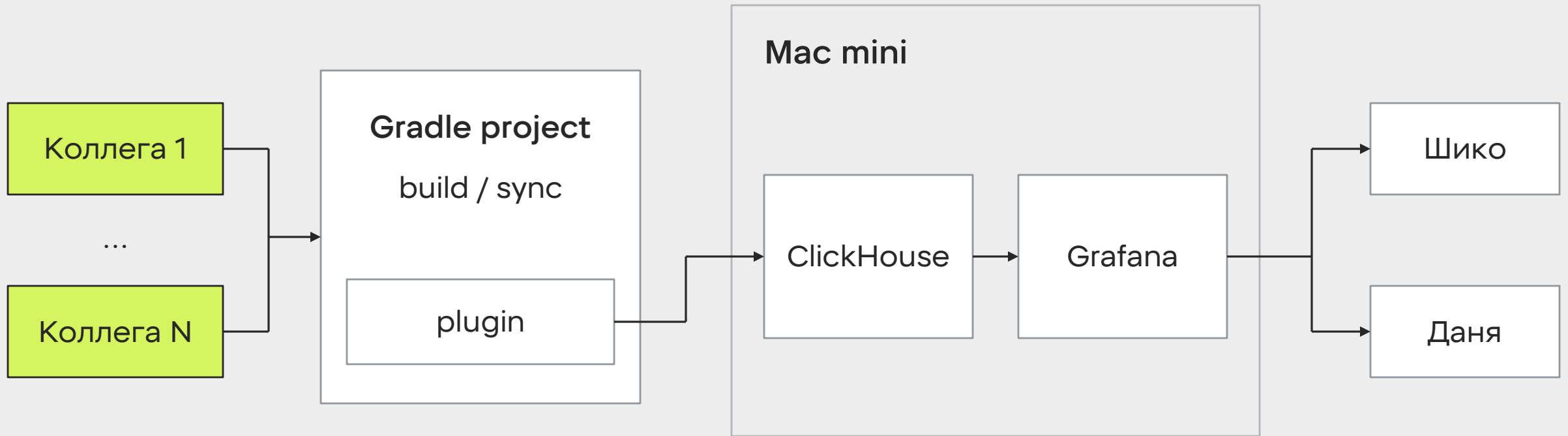
08



Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как все работает?

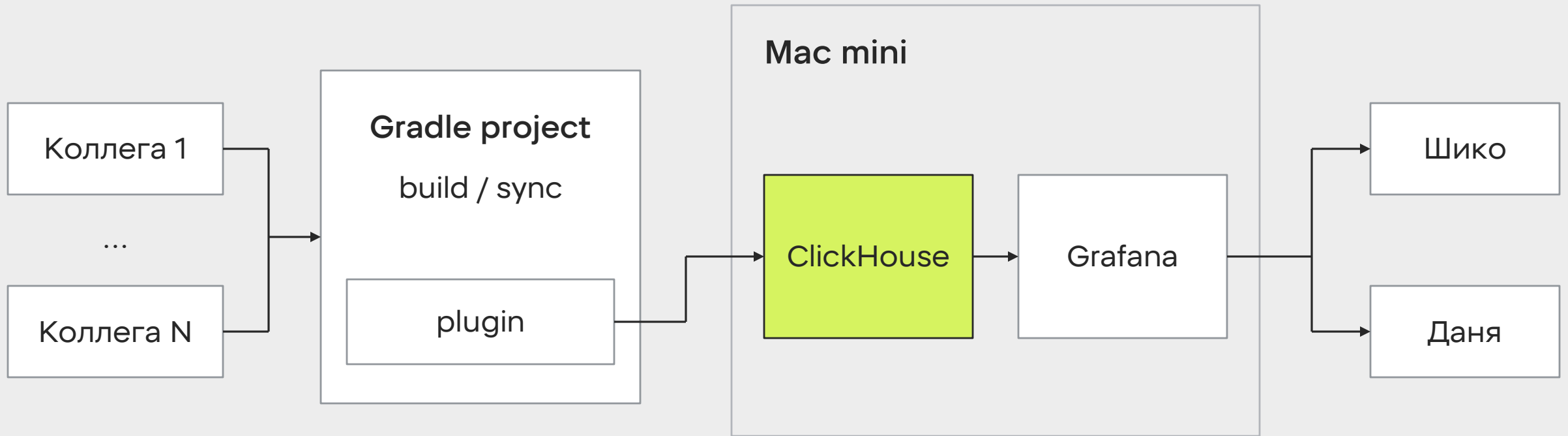
09



Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как все работает?

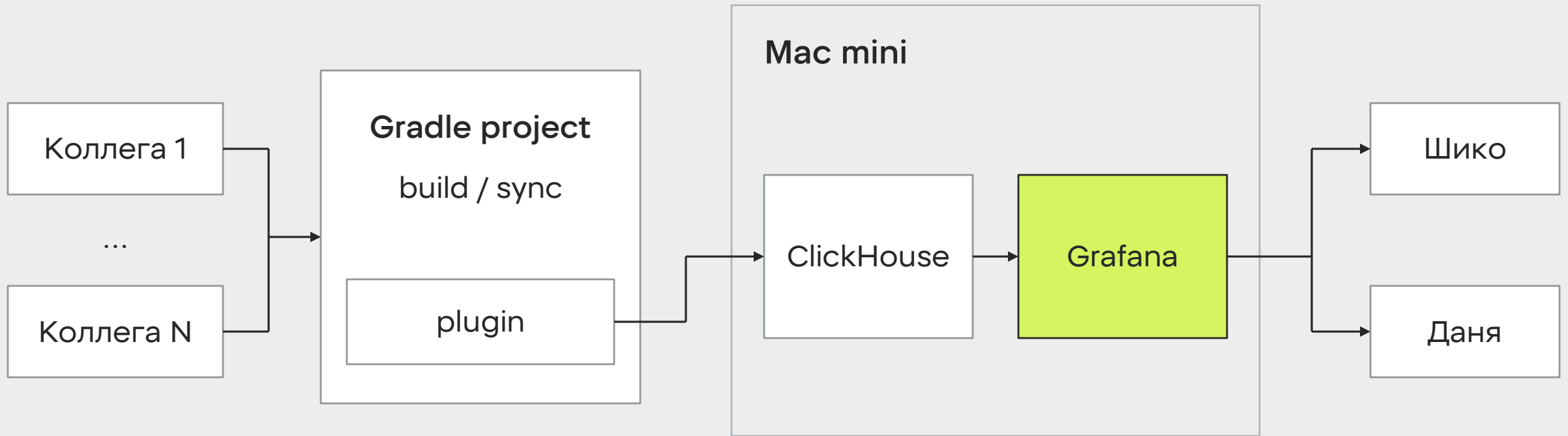
10



Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

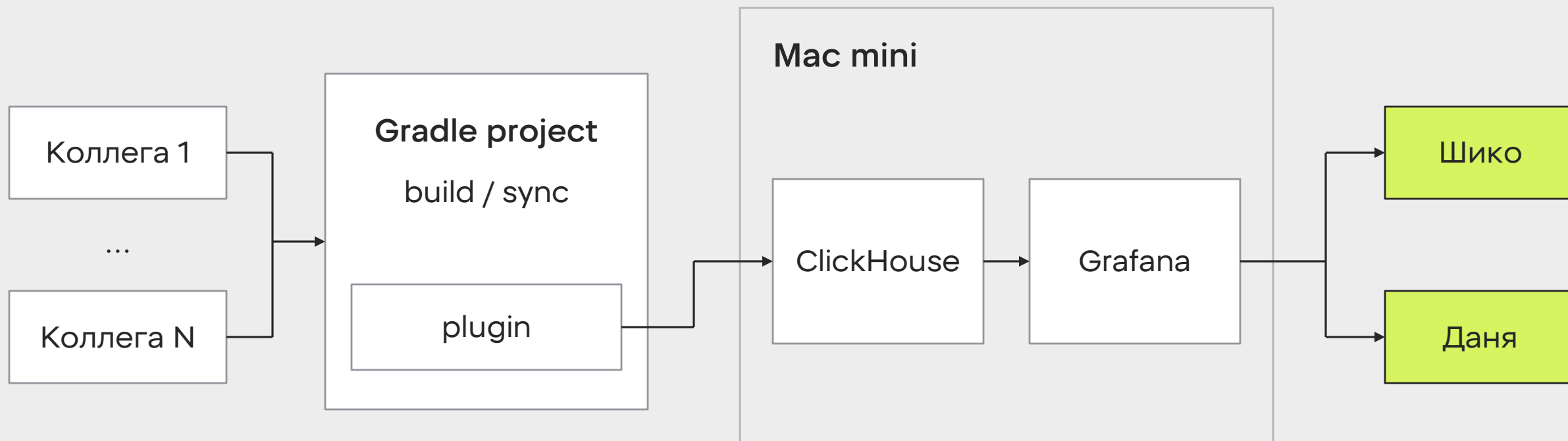
Как все работает?

11



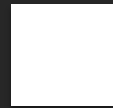
Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как все работает?

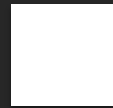


Плагин шлёт данные в ClickHouse, Grafana рисует эти данные из ClickHouse. Всё.

Как подключали плагин



14 репозиторий + быстро меняющийся плагин →
динамическая версия, чтобы не гонять 14 MR на каждый релиз



Репозитории живут на разных версиях Gradle, Kotlin и AGP — перед релизом плагин прогоняется на двух крайних стеках: самом старом и самом свежем

ВСЁ ПОДКЛЮЧЕНИЕ В РЕПОЗИТОРИИ — ДВЕ ПРАВКИ

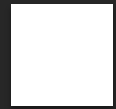
```
// settings.gradle.kts — где искать плагин
pluginManagement {
    repositories {
        maven { url = uri("https://art.x5.ru/artifactory/maven-mobile-local-release/") }
    }
}
// build.gradle.kts — версия динамическая: новый релиз приезжает сам
plugins {
    id("ru.x5.mobile.infra.buildmetrics") version "1.+"
}
```

Что замеряем?

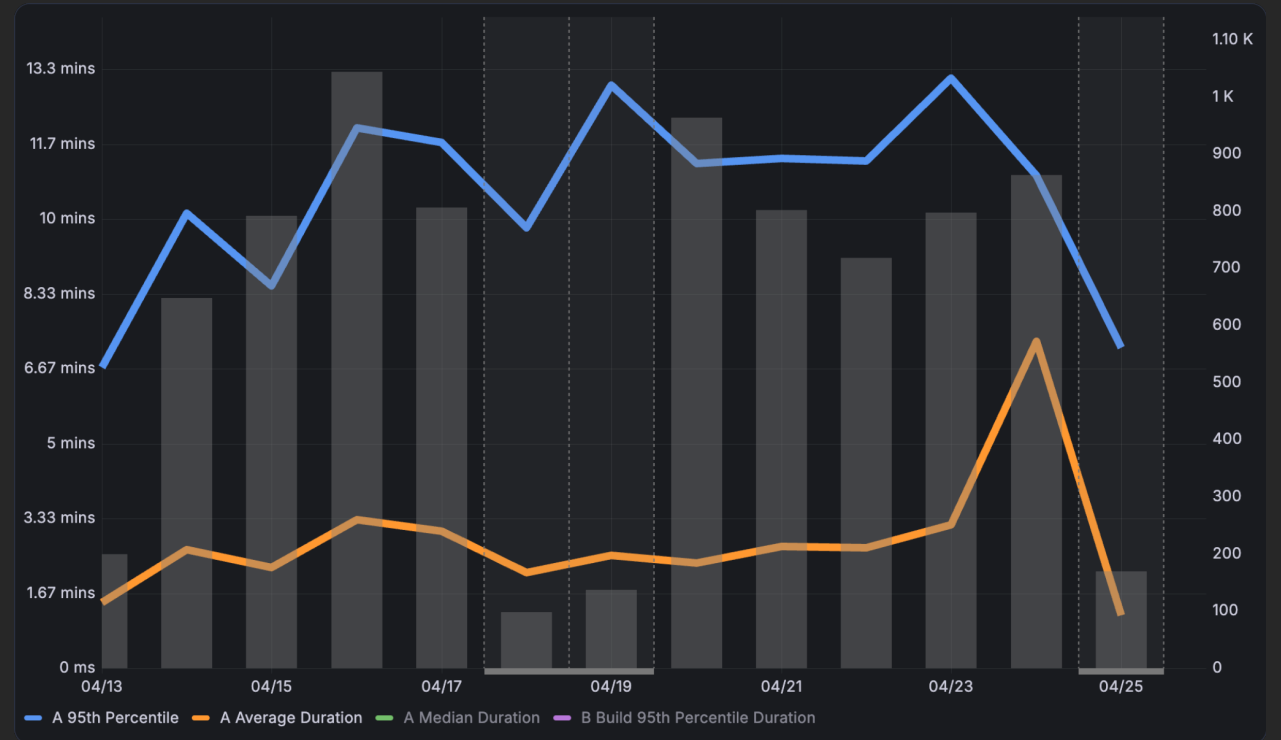
14



Замеряем локальные сборки разработчиков

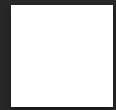


Берём 95-й перцентиль от всех сборок

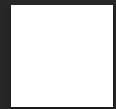


Что замеряем?

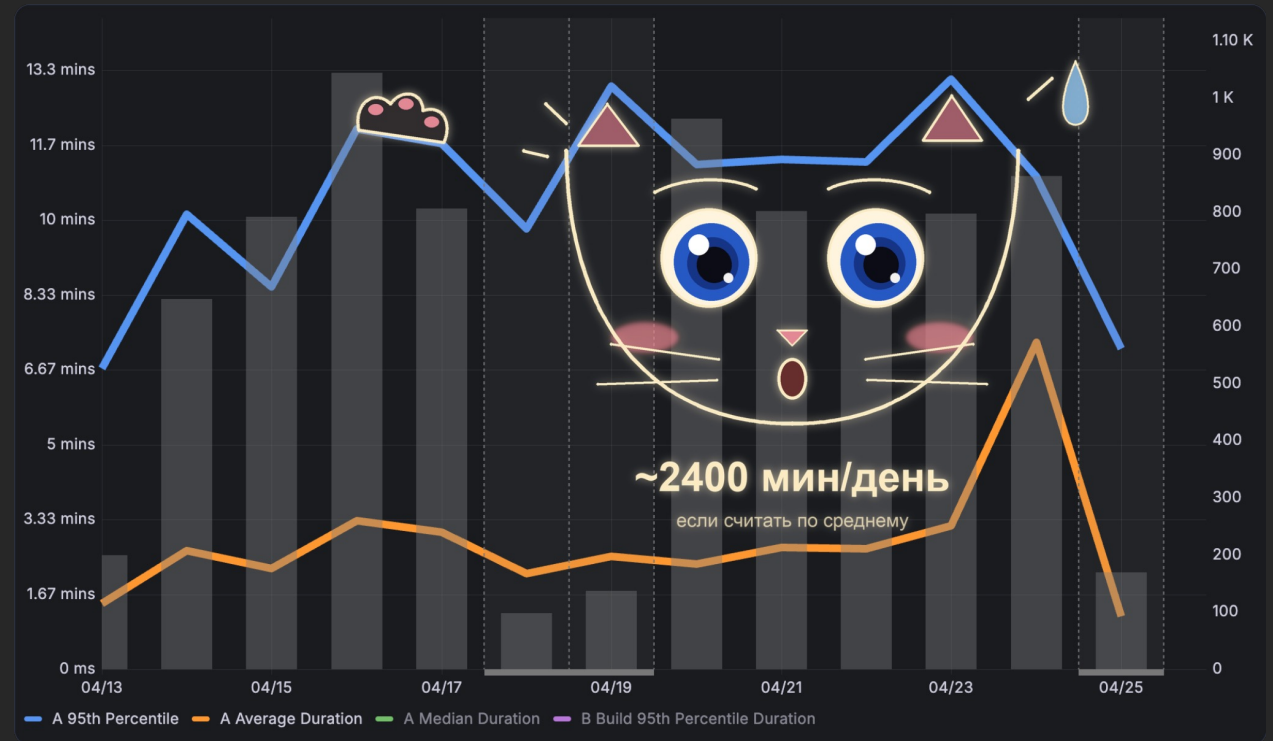
15



Замеряем локальные сборки разработчиков

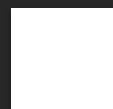


Берём 95-й перцентиль от всех сборок



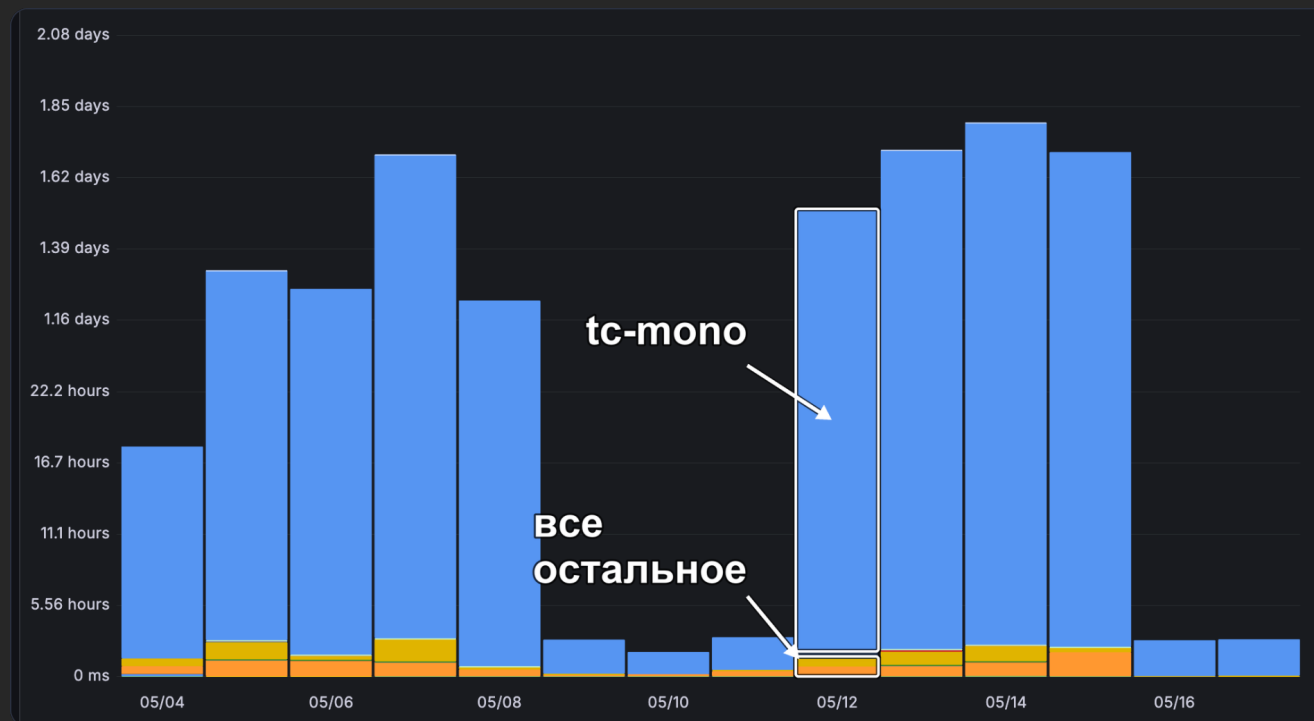
Что будем оптимизировать?

16



Сразу увидели: `tc-mono` занимает 95%+ суммарного времени сборки

→ берёмся за него



Проект tc-mono

Монорепо из 9 composite-сборок

Было

30 марта · влили
плагин метрик

1,57M

строк кода

320

gradle-модулей

Стало

12 июля · конец окна
замера

2,05M

строк кода

491

gradle-модулей

Проект tc-mono

Монорепо из 9 composite-сборок

Было

30 марта · влили
плагин метрик

1,57M

строк кода

320

gradle-модулей

Стало

12 июля · конец окна
замера

2,05M

строк кода

491

gradle-модулей

ОДНО РЕПО — 4 ПРИЛОЖЕНИЯ

Пятёрочка tc5

Перекрёсток tcx

Чижик tc4

Секретный проект скоро

↑ ↑ ↑ ↑

общий код · tc-shared

Проект tc-mono

Монорепо из 9 composite-сборок

Было

30 марта · влили
плагин метрик

1,57M

строк кода

320

gradle-модулей

Стало

12 июля · конец окна
замера

2,05M

строк кода

491

gradle-модулей

Всё это время

темп разработки

+30K

строк кода в неделю

~100

разработчиков

ОДНО РЕПО — 4 ПРИЛОЖЕНИЯ

Пятёрочка

tc5

Перекрёсток

tcx

Чижик

tc4

Секретный проект

скоро

↑ ↑ ↑ ↑

общий код · tc-shared

Проект tc-mono

Монорепо из 9 composite-сборок

Подробнее —
в докладе



Денис Александров

A mono project
is all you need



Прежде чем чинить сборки — пришлось чинить графики

Данные ввали, сквозная метрика прятала фазы, дубли сдвигали p95, Grafana гасла, а эффект любого фикса приезжал через неделю

01

Обновили Kotlin / Gradle / AGP → всё замедлилось

22

Подняли версии Kotlin и библиотек, ждали ускорения

Сначала всё замедлилось

Затем получили ускорение
но выглядит недостаточно

Разделили метрику:
отдельно синки,
отдельно сборки

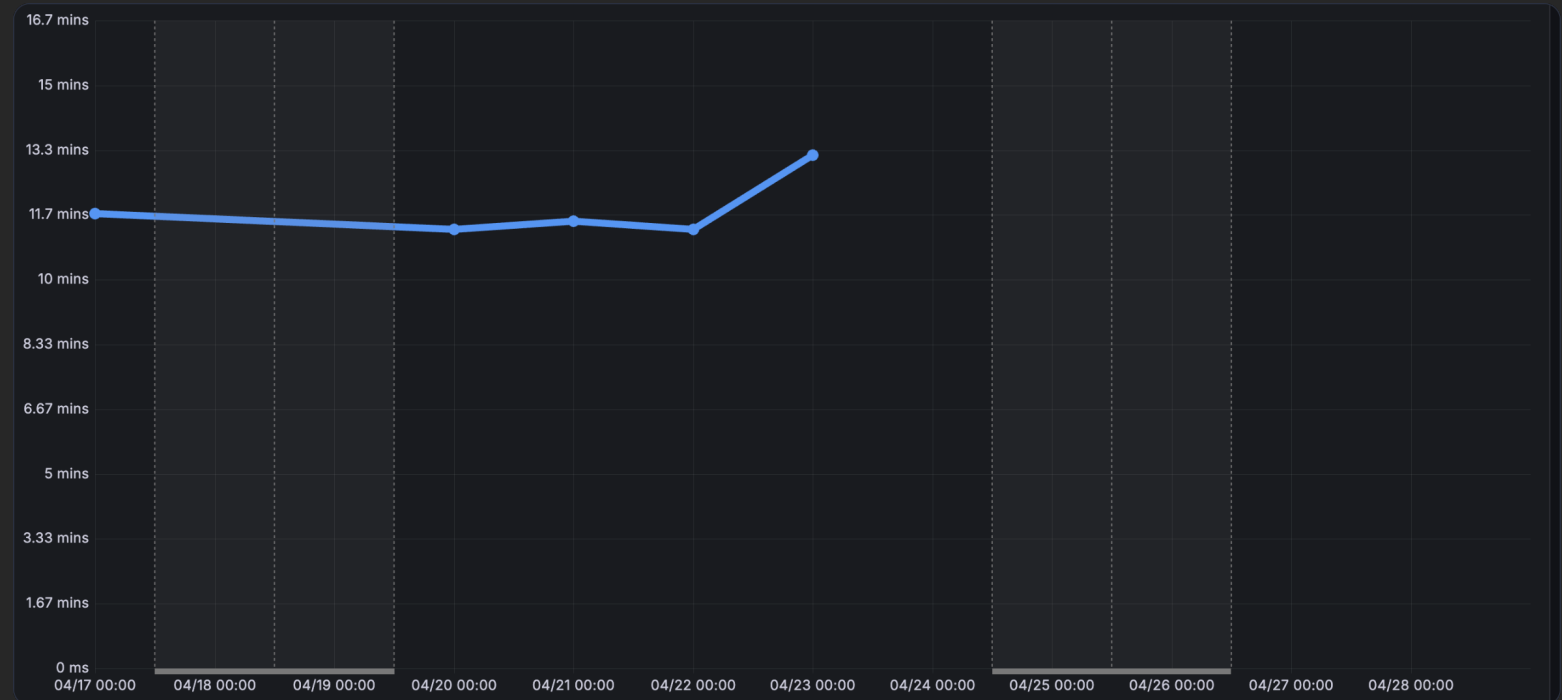


Обновили Kotlin / Gradle / AGP → всё замедлилось

23

Подняли версии Kotlin и библиотек, ждали ускорения

Сначала всё замедлилось



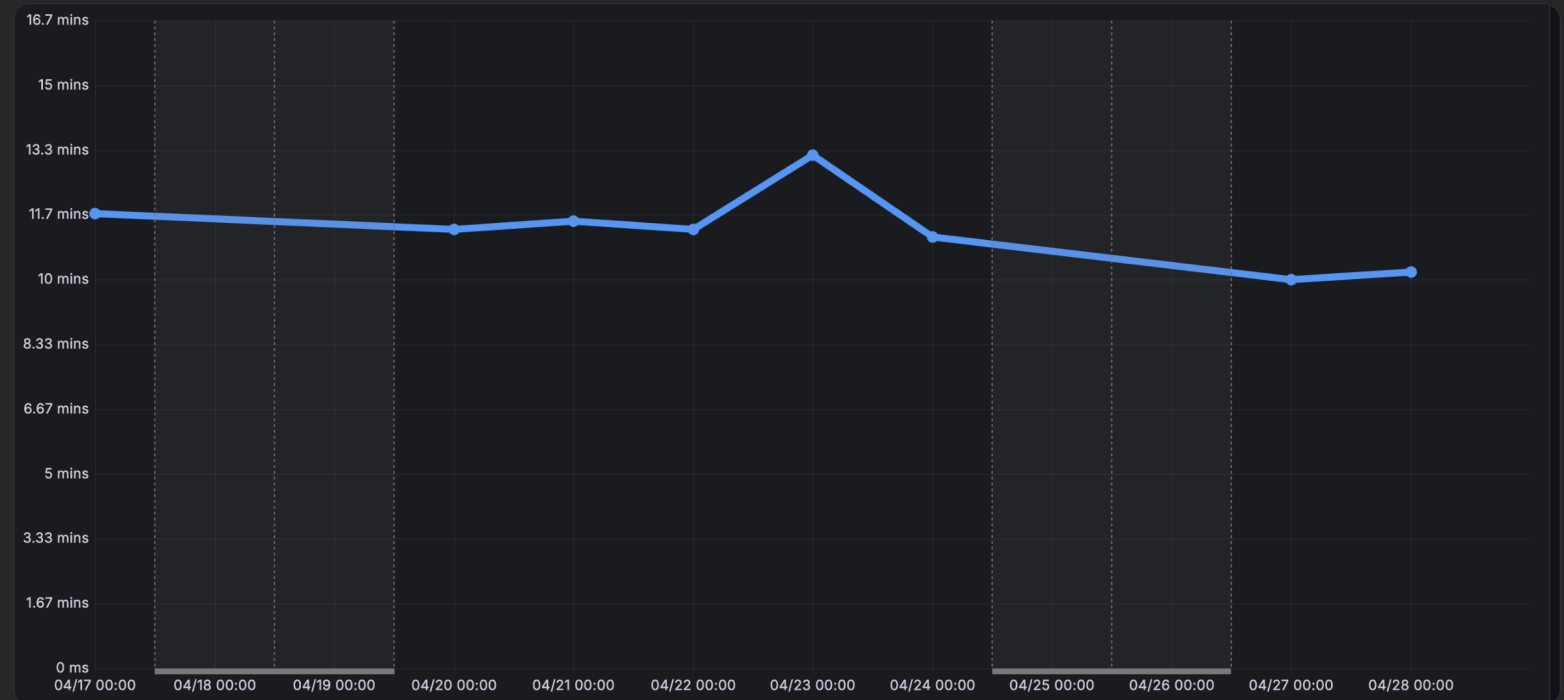
Обновили Kotlin / Gradle / AGP → всё замедлилось

24

Подняли версии Kotlin и библиотек, ждали ускорения

Сначала всё замедлилось

Затем получили ускорение, но выглядит недостаточно



Обновили Kotlin / Gradle / AGP → всё замедлилось

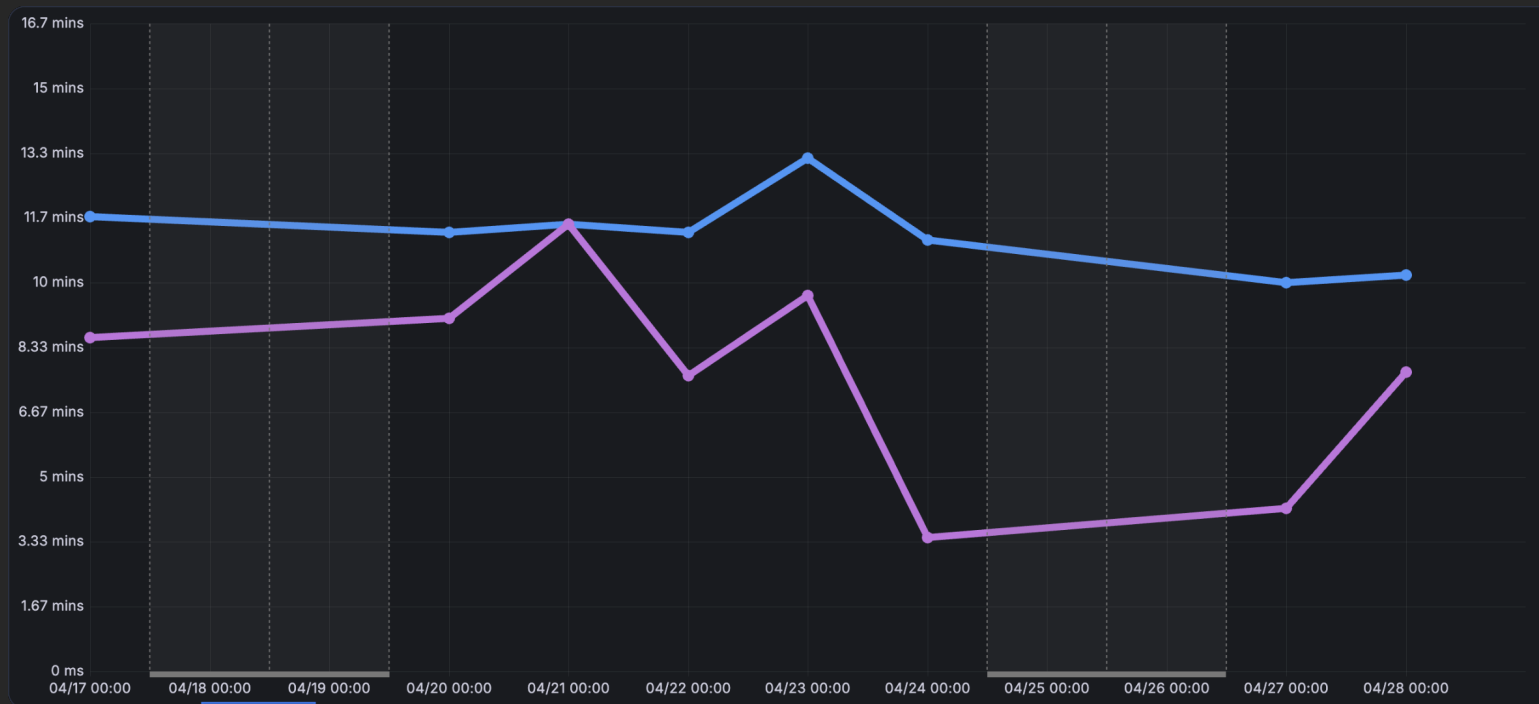
25

Подняли версии Kotlin и библиотек, ждали ускорения

Сначала всё замедлилось

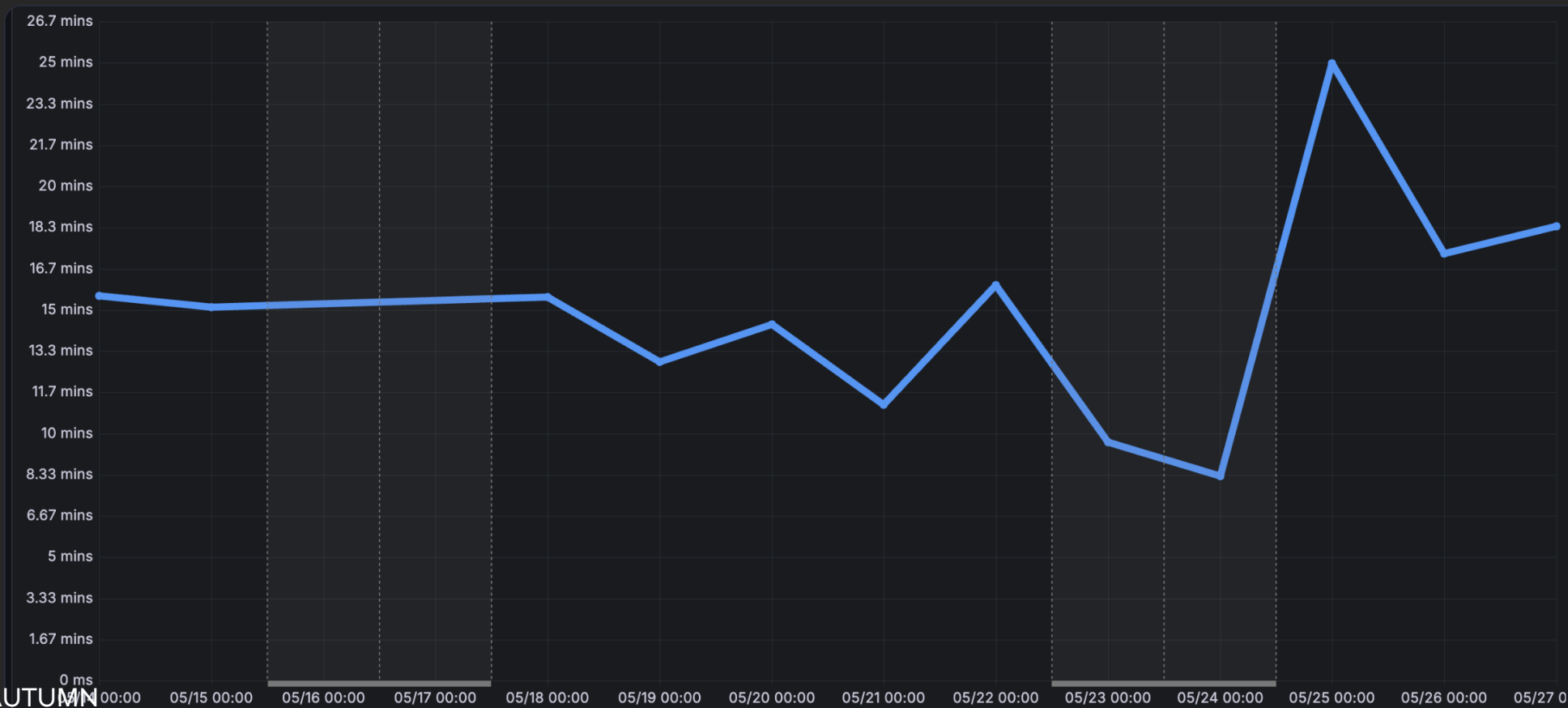
Затем получили ускорение, но выглядит недостаточно

Разделили метрику: отдельно синки, отдельно сборки



Графики выросли. Опять?

Линии поползли вверх — что мы сломали на этот раз?



Графики выросли. Опять?

Линии поползли вверх — что мы сломали на этот раз?

Плагин в `try/catch` блоке несколько раз пытался отправить данные

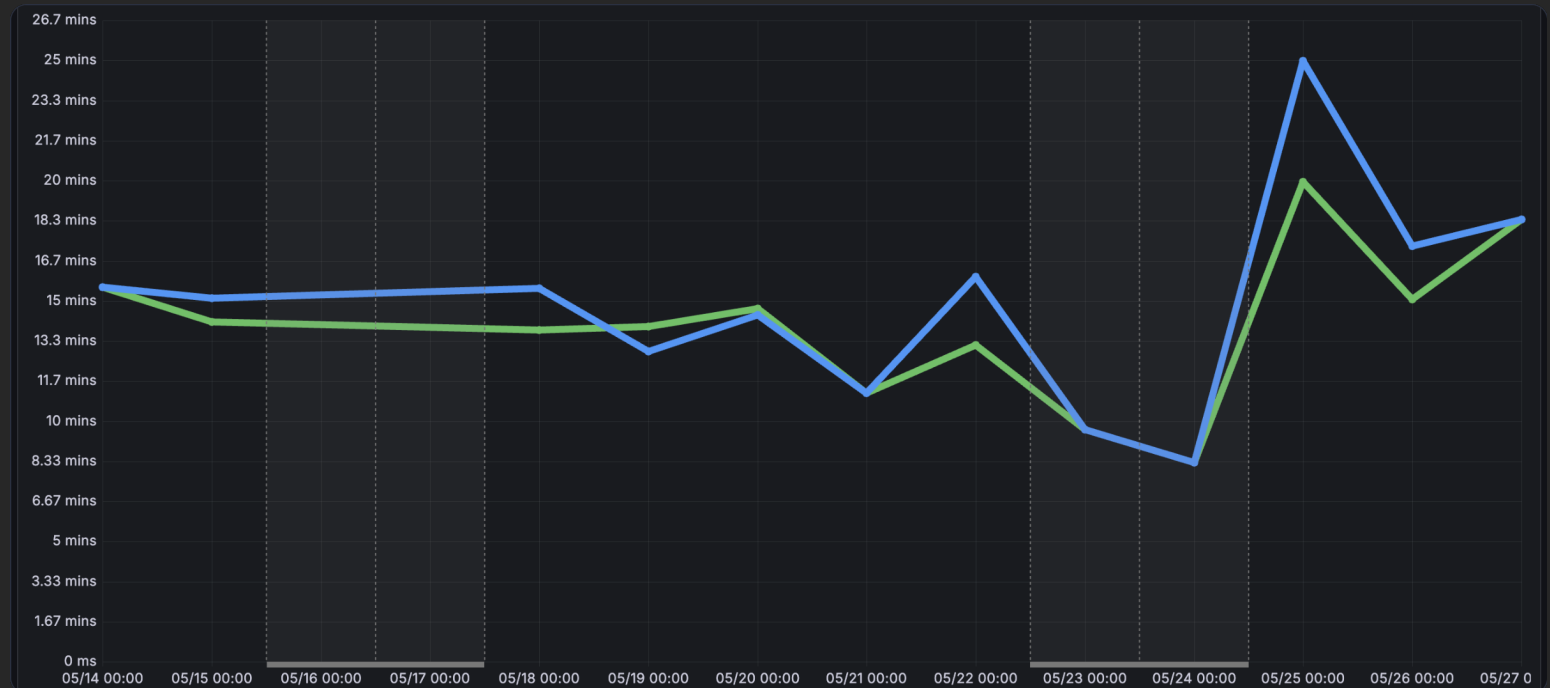
```
// SenderProcessor.uploadOne — до фикса
while (true) {
    attempt++
    try {
        uploader.upload(payload) // вставка прошла, а ответ потерялся → исключение
        return
    } catch (ex: Exception) {
        if (attempt >= maxAttempts) {
            return
        }
        sleeper(backoff) // и отправляем тот же отчёт ещё раз
        backoff *= 2
    }
}
```

Графики выросли. Опять?

28

Линии поползли вверх — что мы сломали на этот раз?

Плагин в `try/catch` блоке несколько раз пытался отправить данные



Графики выросли. Опять?

Линии поползли вверх — что мы сломали на этот раз?

Плагин в `try/catch` блоке несколько раз пытался отправить данные

Включили дедупликацию с двух сторон — окно в ClickHouse и идиempотентный токен в отправщике → графики вернулись к правде

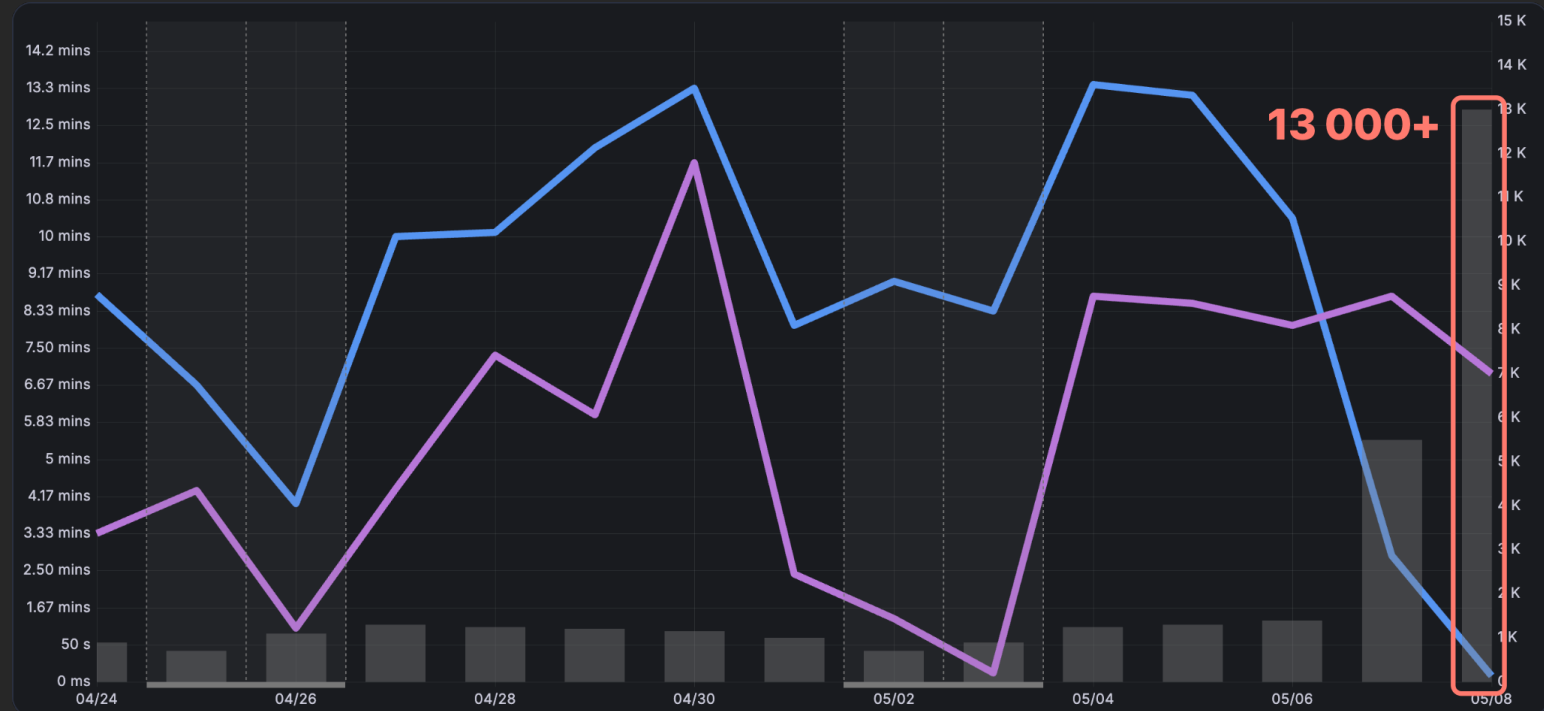
-- 1. ClickHouse: в DDL каждой таблицы включаем окно дедупликации
`SETTINGS non_replicated_deduplication_window = 1000` -- по умолчанию 0, и токен ничего не делает

// 2. отправщик: стабильный токен на логическую запись — `<таблица>:<build_id>`
`val settings = InsertSettings().serverSetting("insert_deduplication_token", "$table:$buildId")`
`client.insert(table, listOf(record), settings).get()`

13 000+ сборок за день, среднее – 6 секунд



В один из дней сборок стало больше 13К, среднее время – 6 секунд.
Ускорение века?

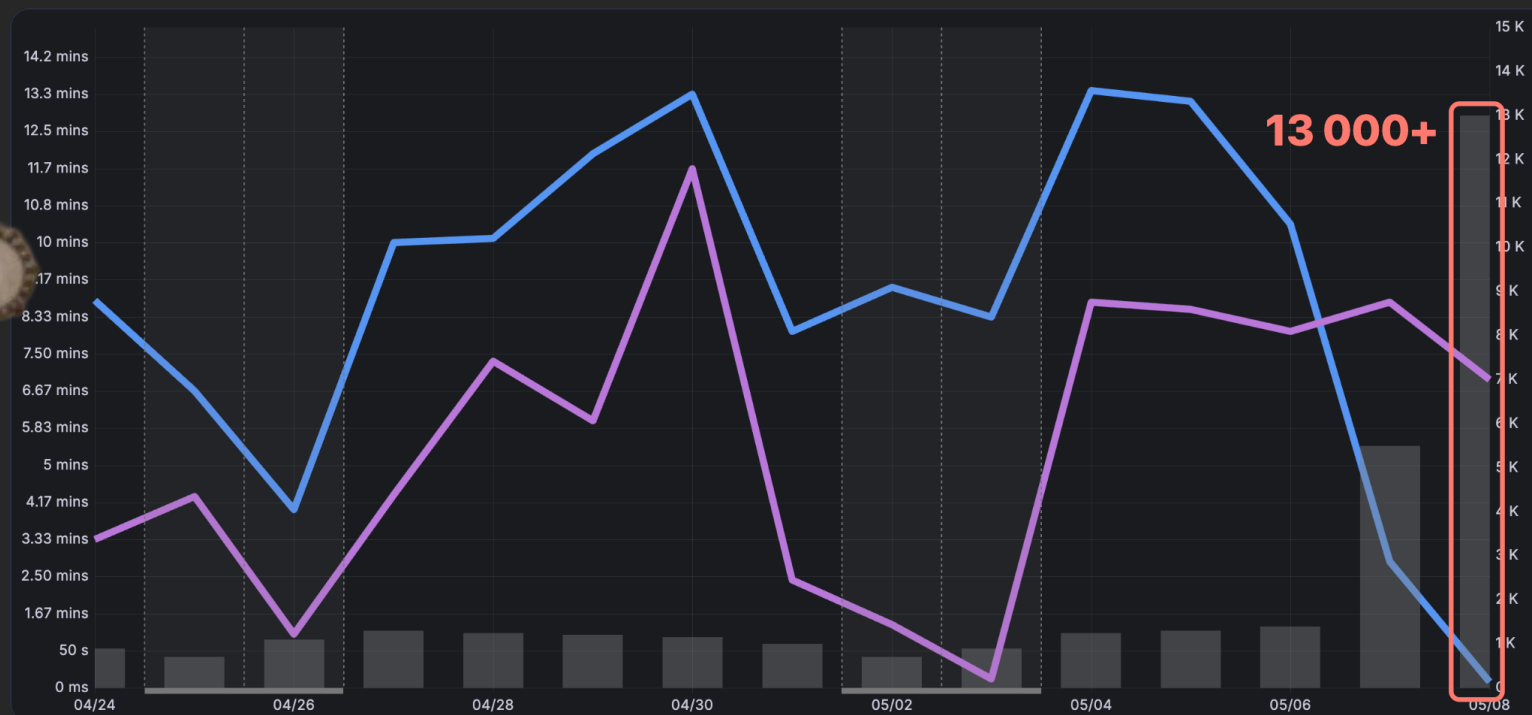


13 000+ сборок за день, среднее – 6 секунд

В один из дней сборок стало больше 13К, среднее время – 6 секунд. Ускорение века?

Нет: коллега сломал нам метрики – его окружение спамило сборками

Починили ему логи, выпилили лишнее из истории

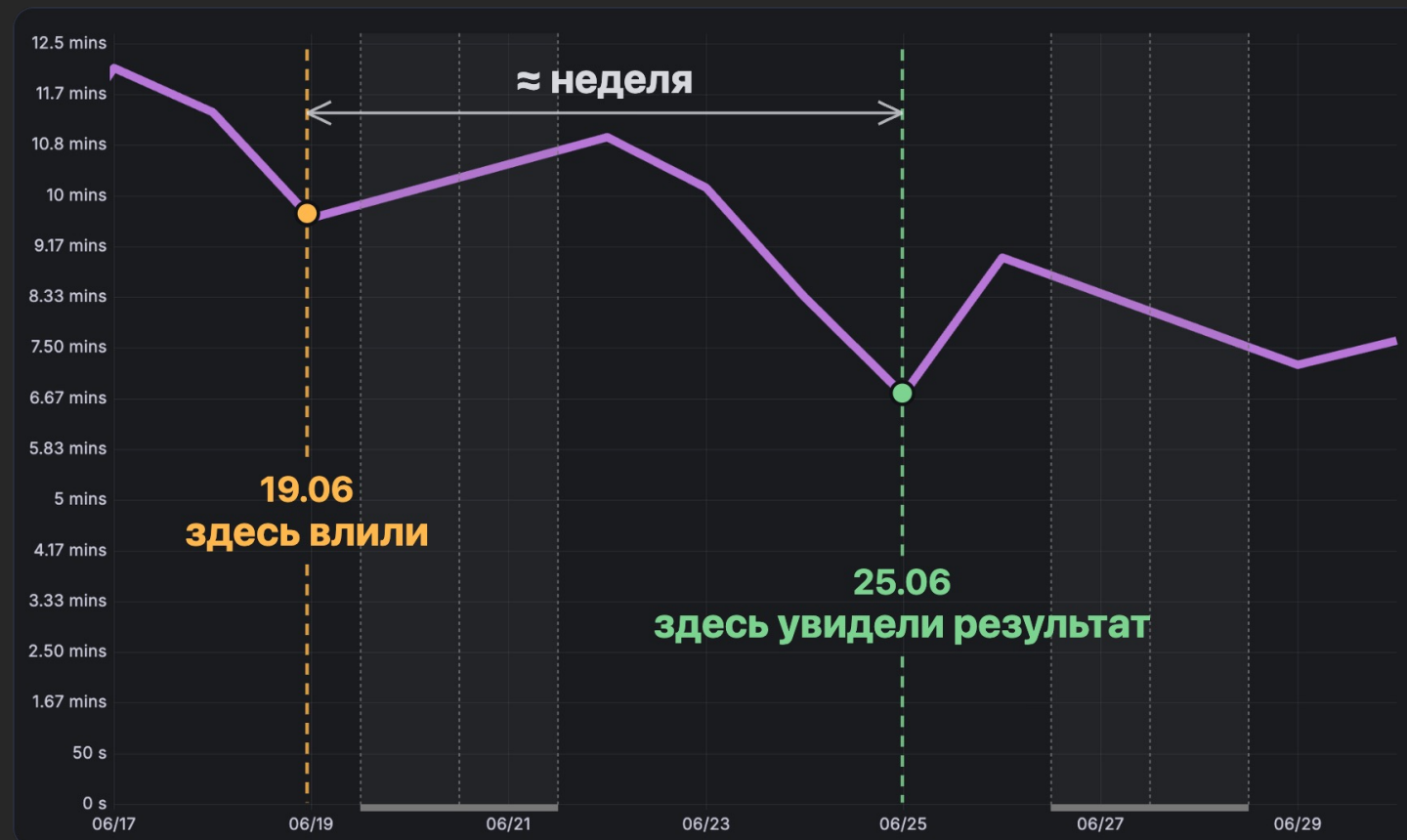


Задолбало ждать неделю

32

После любого улучшения ~неделя, пока все разработчики подтянут его к себе – и только потом видно эффект

Сделали график от даты, когда правка попала в основную ветку (develop), а не от даты сборки

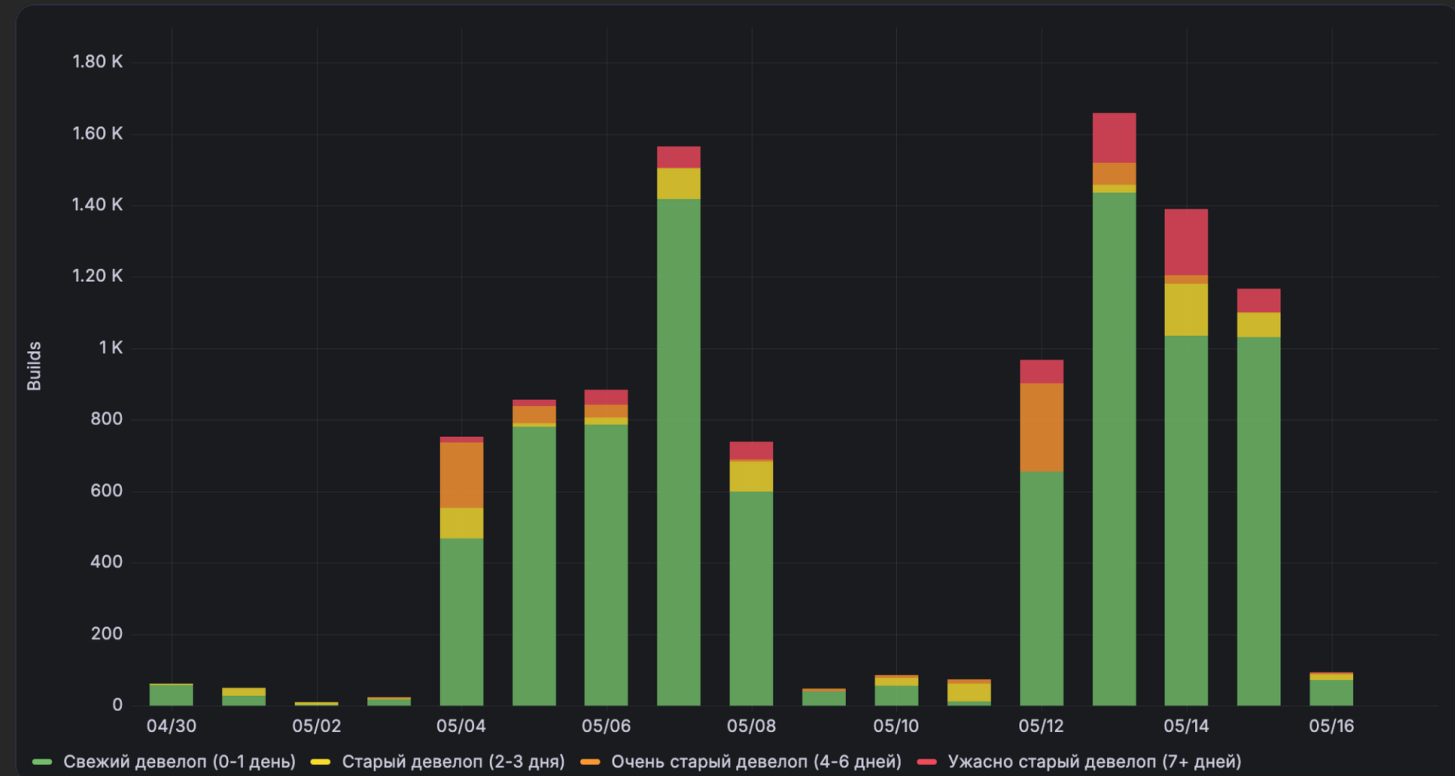


Задолбало ждать неделю

После любого улучшения ~неделя, пока все разработчики подтянут его к себе – и только потом видно эффект

Сделали график от даты, когда правка попала в основную ветку (develor), а не от даты сборки

Заодно видно, насколько у каждого свежий код: кто отстал и насколько



Сходимость графиков

Обычный график (от даты сборки)

■ Свежие дни дозаполняются новыми сборками — картина «дорисовывается в будущее»

■ ±Реальную картину видишь через неделю

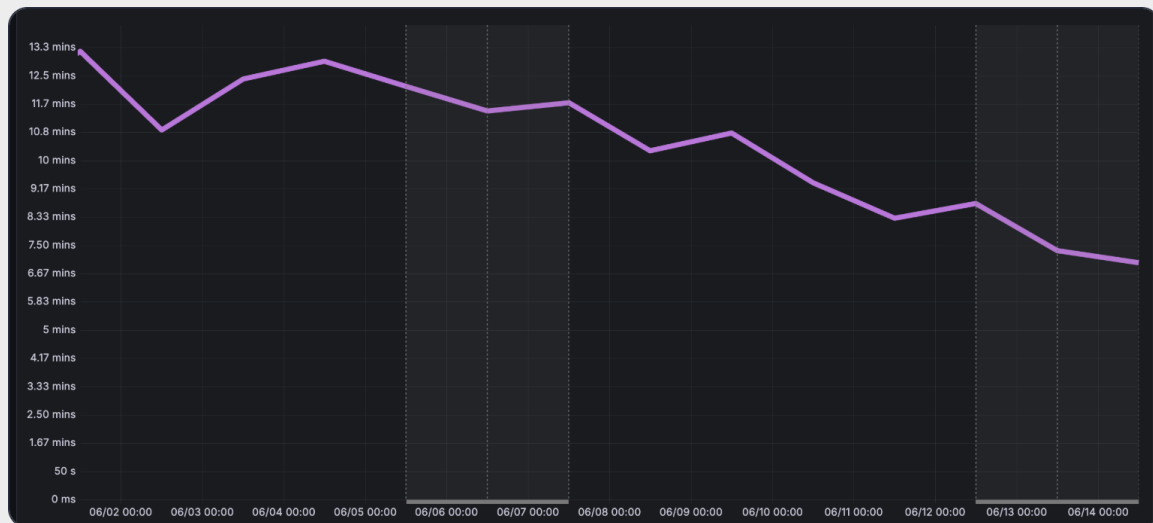
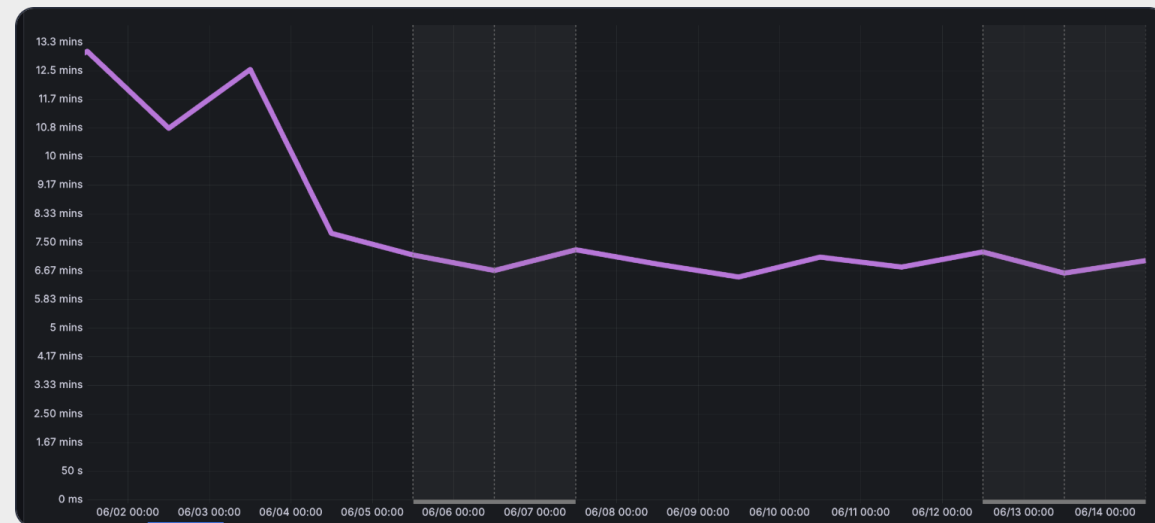


График от develop

■ Старые точки дозаполняются «в прошлое» — на старый коммит ещё прилетают сборки

■ Но чёткая картина — уже за 2–3 дня



Grafana отключилась



В один из дней мониторинг просто погас;
данные за простой терялись



Стали делать **резервные копии данных** —
история метрик больше не пропадает,
даже если инфра умерла
совсем

Страница не найдена



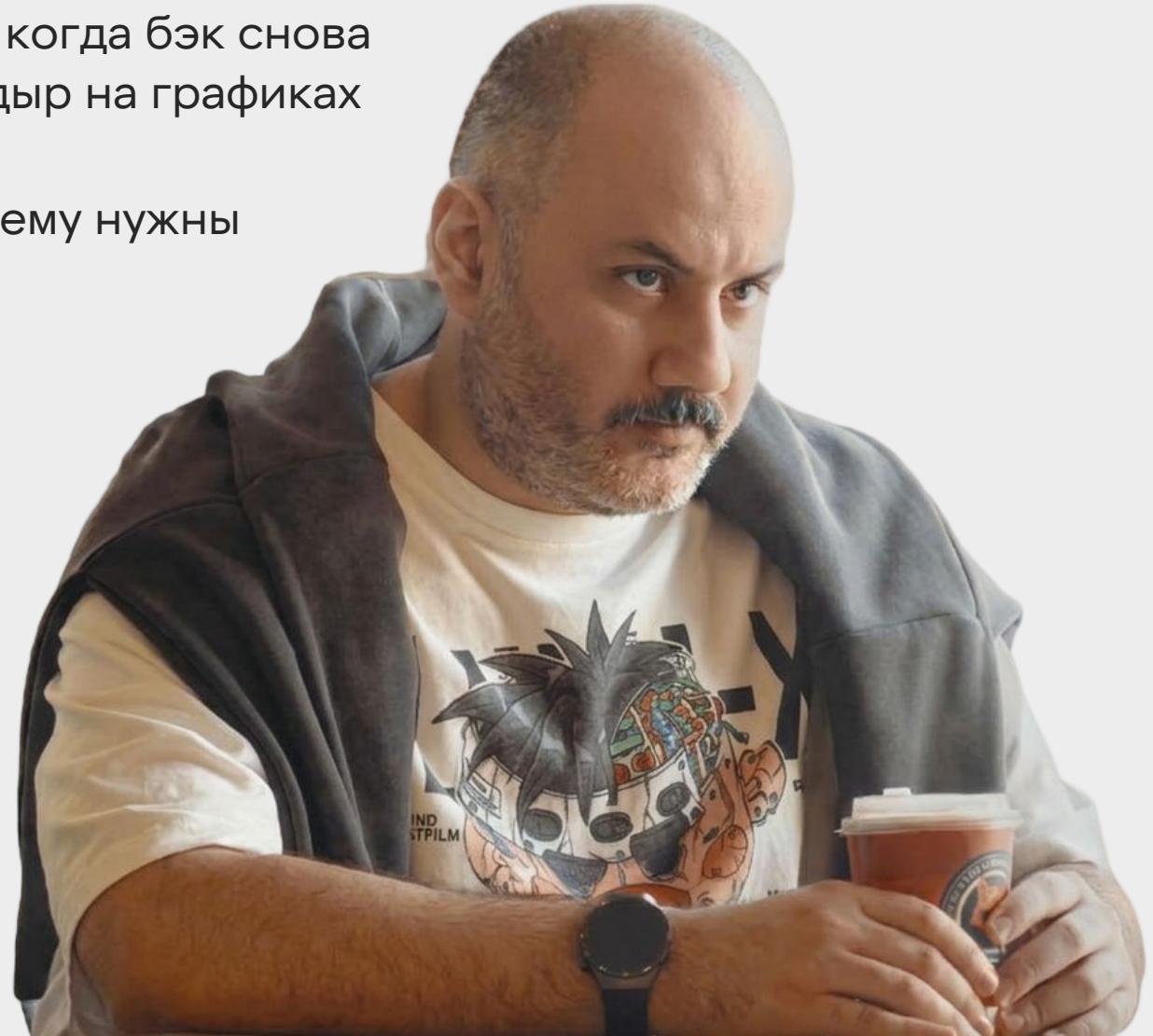
Если вы искали на ней что-то конкретное,
спросите об этом Яндекс:

[Техническая информация](#)

Через неделю — ещё раз

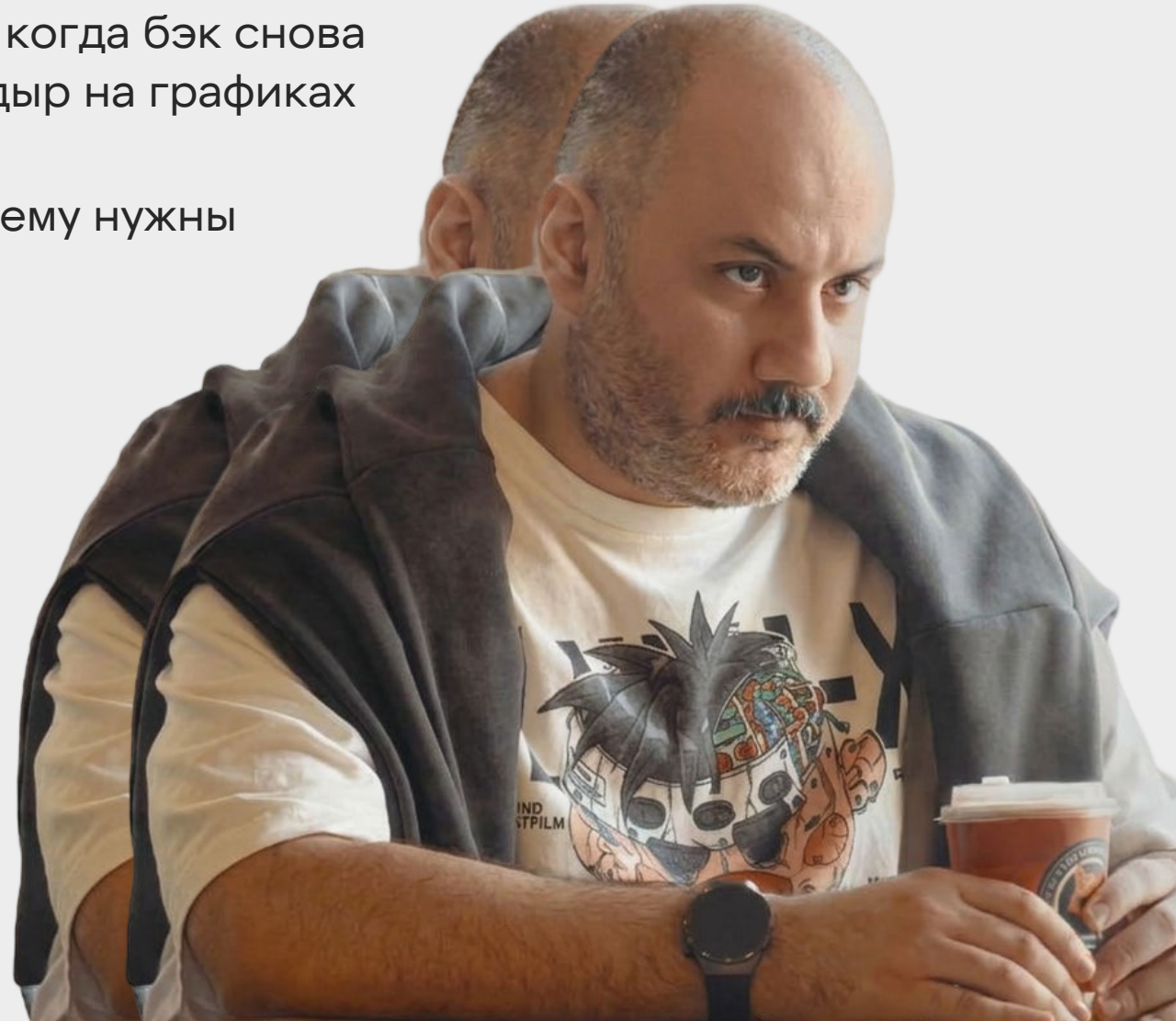
■ Плагин **дозаливает** накопившиеся данные, когда бэк снова доступен — простой больше не оставляет дыр на графиках

■ Observability-конвейер — тоже продакшен: ему нужны свои ретраи, свой backfill и свой алертинг



Через неделю — ещё раз

- Плагин **дозаливает** накопившиеся данные, когда бэк снова доступен — простой больше не оставляет дыр на графиках
- Observability-конвейер — тоже продакшен: ему нужны свои ретраи, свой backfill и свой алертинг
- Побочка: теперь и график **по дате сборки** дополняется задним числом — как график по оси develop, вчерашняя точка может сдвинуться сегодня



Через неделю — ещё раз

■ Плагин **дозаливает** накопившиеся данные, когда бэк снова доступен — простой больше не оставляет дыр на графиках

■ Observability-конвейер — тоже продакшен: ему нужны свои ретраи, свой backfill и свой алертинг

■ Побочка: теперь и график **по дате сборки** дополняется задним числом — как график по оси develop, вчерашняя точка может сдвинуться сегодня

■ ...а через месяц этот самый backfill **сам начал плодить дубли** — те самые, с которых мы чинили графики

Теперь перед досылкой плагин **проверяет**, что уже лежит на бэке, и доливает только недостающее



Теперь графикам можно верить. Что они показали

Каждый шаг: что увидели
на графике → что сделали
→ что стало с графиком

02

Композитный плагин

Решили отдельным плагином настраивать remote build cache...

...и пришли к идее **композитного плагина**:
один плагин позволяет подключить
все общие оптимизации разом

Каждая общая оптимизация
опциональна, сначала проверяем
стабильность на 1–2 проектах

Composite plugin

Build
metrics

Studio
sync fix

Remote
build cache

Remote build cache: как настраивали

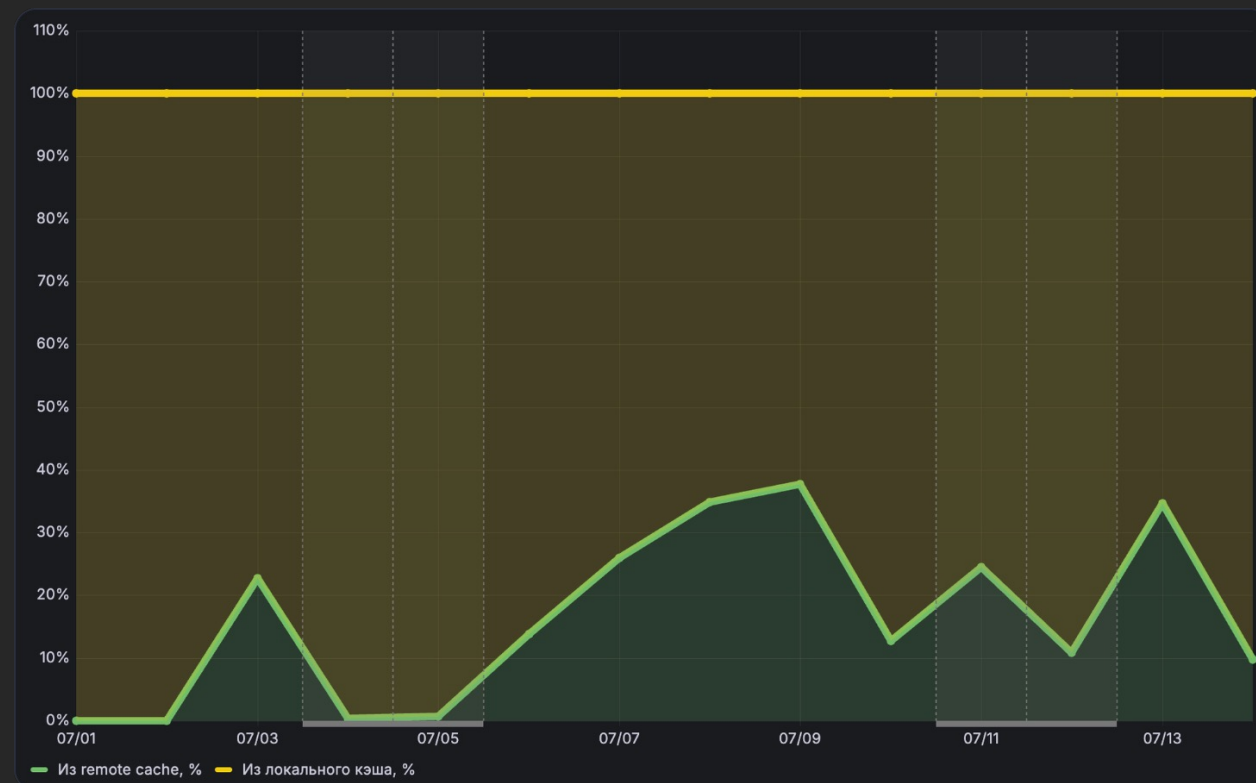
■ Поначалу прогревали кэш руками, пока не донастроили на CI

■ Уменьшили количество inputs в задачах — иначе ключи кэша не совпадают

■ Настроили общую Java на CI и у пользователей — чтобы задачи брались из кэша

■ Сделали плагин, который досылает инфу по кэшу: что берётся из локального, что из remote

■ Протестировали два варианта хранилища: **Artifactory vs миник**

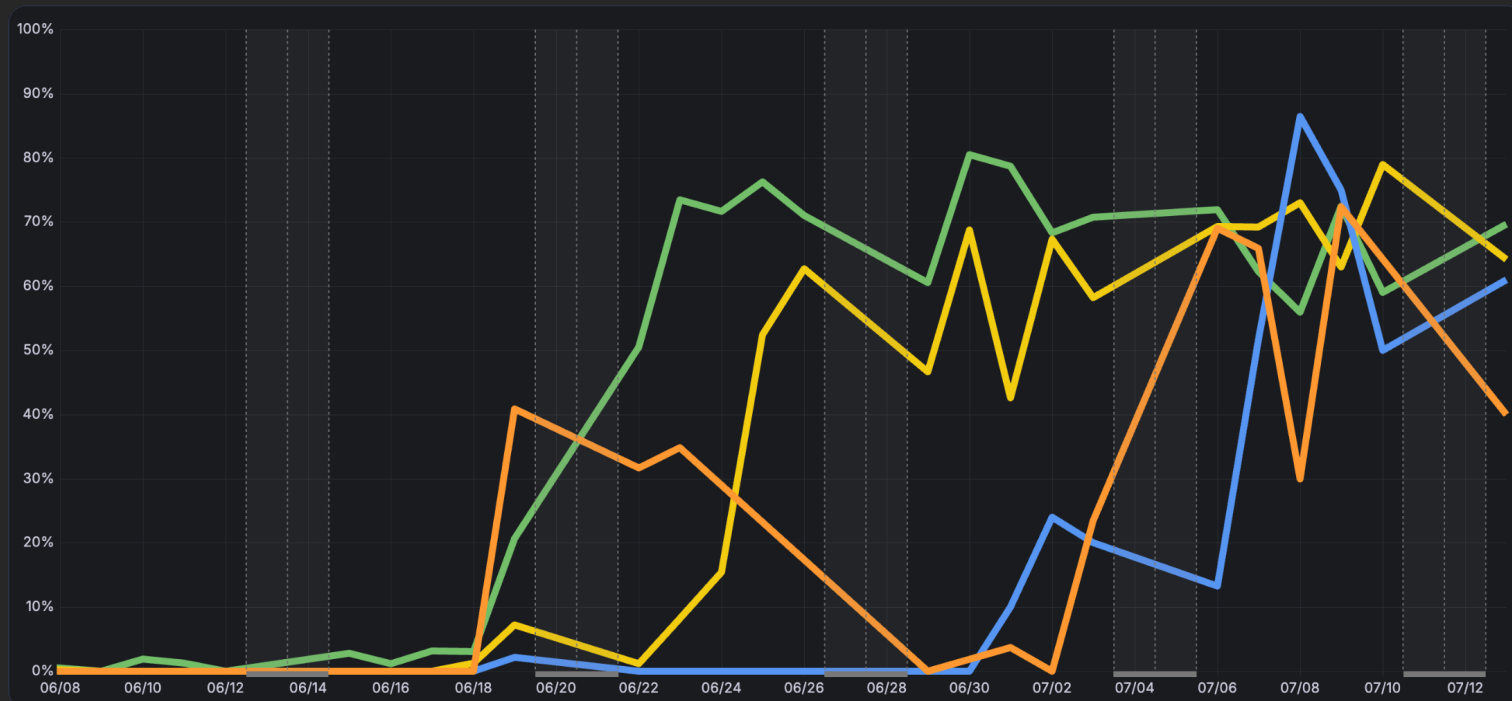


Configuration Cache не работает. Вообще

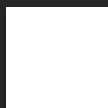
■ CC не работает для всех сборок tc-моно из корня –
hit $\approx 2\%$, при том что standalone тот же таргет даёт $\sim 67\%$

■ Вынули апы из includeBuild
в subproject'ы – каждый
в свой день

- tc5-android · 18.06
- tcx-android · 24.06
- tc4 · 01.07
- секрет · 03.07



Как чинили: includeBuild → subproject'ы



Было

```
includeBuild("tc5-android")
```

Отдельный Gradle build.
Студия + композит = СС не
переиспользуется



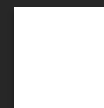
Стало

```
include(":tc5-android:app")
include(":tc5-android:mode")
// ...и так все модули аппа
```

Обычные subproject'ы корня.
Один build вместо двух

ВСЕ ВХОД В КОРНЕВОМ SETTINGS.GRADLE.KTS
– ТРИ СТРОКИ

```
extra["inlineApps"] = listOf("tc5-android", "tcx-android", "tc4", "secret")
// сканер + include + projectDir
apply(from = "../custom.include.modules.settings.gradle.kts")
```



Контейнер :tc5-android
уводим на пустую папку-
заглушку – иначе Gradle
прочитает tc5-
android/build.gradle.kts как build-
скрипт контейнера, а он
написан под корень
отдельного билда



rootProject.name = префикс
инлайна → typesafe-
аксессуары и пути задач в CI
не меняются, ап по-
прежнему открывается
standalone

Пользователи бастуют

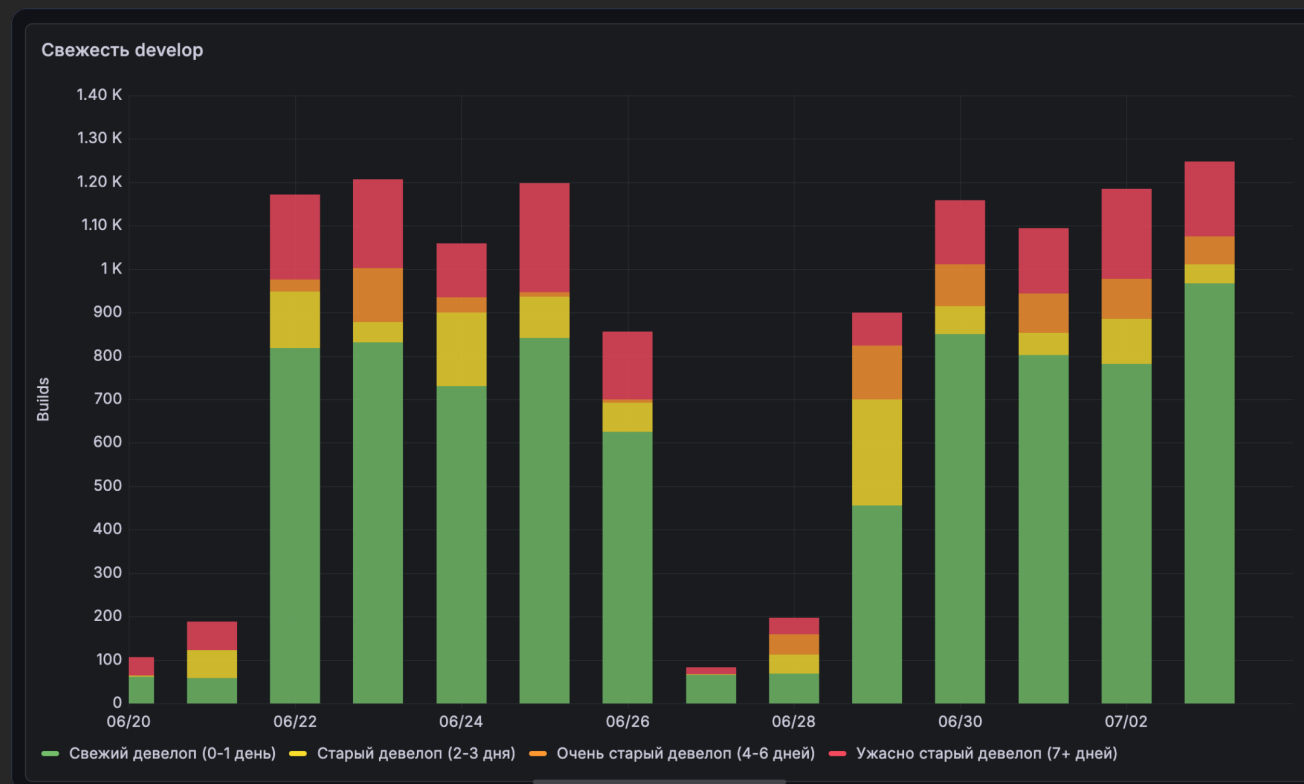
44



Посмотрели на свежесть develop у сборок – и увидели неожиданное:



...разработчики сидят на старом develop

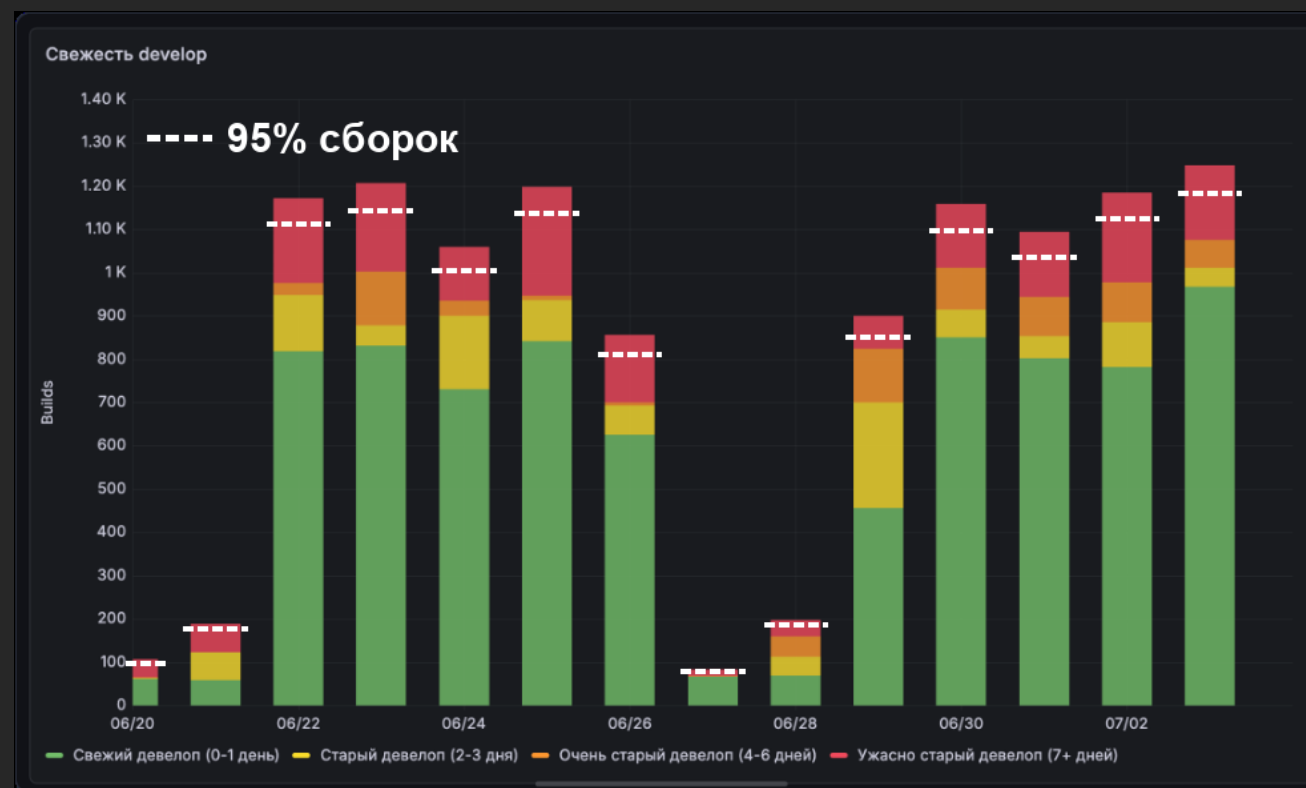


Пользователи бастуют

45

Посмотрели на свежесть develop у сборок — и увидели неожиданное:

...разработчики сидят
на старом develop



Пока мы ускоряли сборку, люди страдали на синке

Мы убиваем время на сборку, а у пользователей медленный sync — и он болит чаще



Кирилл постучался в личку

В прошлом докладе
он ускоряд сборку

Подробно —
в докладе



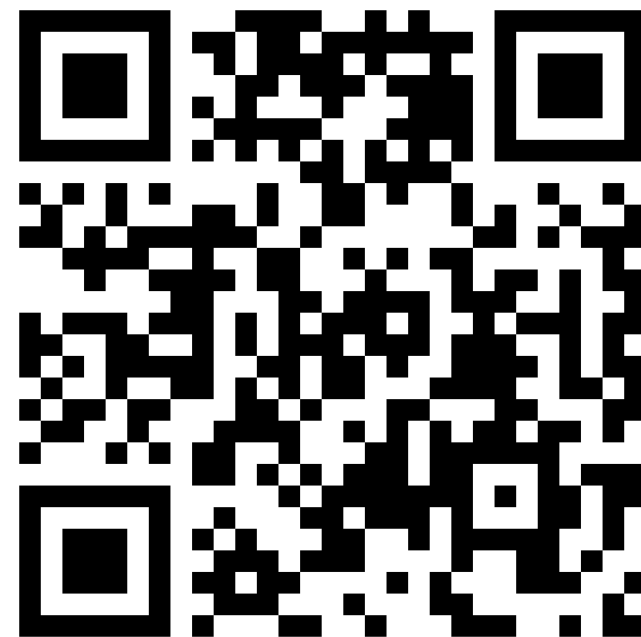
Кирилл Старостин

Мониторим здоровье Gradle-сборки
KMP проекта

Ускорили сборку в 2 раза

Вышла новая Студия —
и сборка замедлилась
в 2 раза

Баланс сил
во Вселенной сохранён



Кирилл постучался в личку

И ВОТ ЧТО ОН НАШЁЛ



Проблемы в телеметрии плагина Kotlin Multiplatform — com.jetbrains.kmm

Два слушателя статистики на каждый модуль рекурсивно обходят всё дерево `DataNode` проекта — синхронно, на потоке уведомлений `sync'a`

```
// плагин регистрирует два слушателя статистики  
KmpStatisticsComposeGradleListener  
KmpStatisticsProjectGradleListener
```

```
// на каждый модуль — рекурсивный обход всего дерева проекта,  
// чтобы посчитать, сколько модулей используют Ktor и Compose  
ExternalSystemApiUtil.findAllRecursively(projectDataNode)
```

```
// O(модули × dataNodes), синхронно, на потоке уведомлений sync'a  
// → sync не завершится, пока обход не закончится
```

Кирилл постучался в личку ФИКС



Хирургический патч плагина: из двух XML внутри jar удаляются две строки регистрации листенеров

Раскатили через композитный плагин — сразу на всех пользователей

скрипт снимает регистрацию листенеров в двух местах

```
<plugins>/kmm-plugin/lib/kmm-plugin.jar  
└─ META-INF/plugin.xml ← минус строка
```

```
<plugins>/kmm-plugin/lib/modules/intellij.kmm.statistics.jar  
└─ intellij.kmm.statistics.xml ← минус строка
```

Полегчало — но синк всё равно плохой

50

Линия сошла с пика и вернулась в коридор 12–16 минут

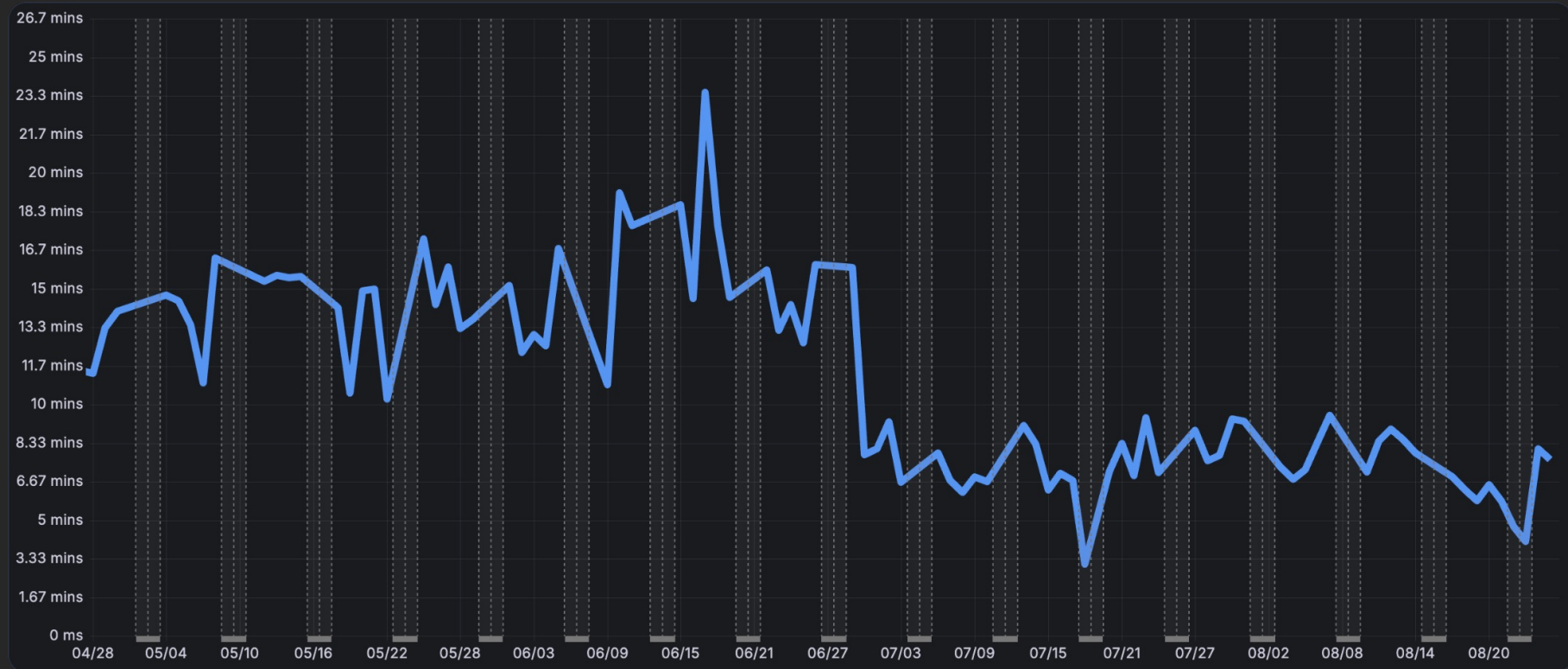


Ускорили синк

51

Ускорили холодный и горячий синк – убрали лишние таргеты из sync, сократили сетевые походы

Пользователи вернулись на нормальный develop – порочный круг «старый develop ↔ дорогой переход» разорван



Проверили антивирус на машинах разработчиков

52

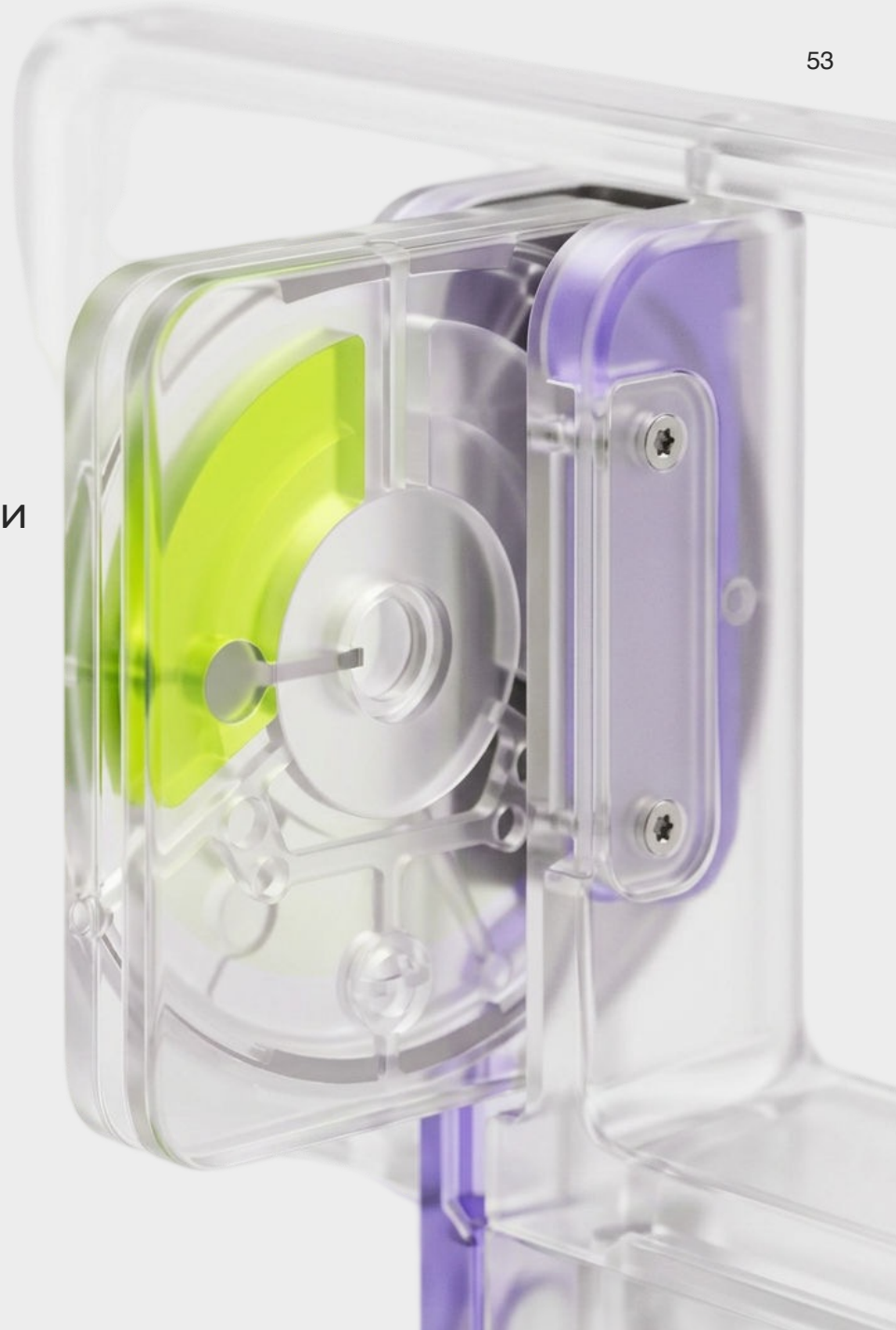
Прошлись по настройкам антивируса у всех коллег

■ У половины — политики по умолчанию: антивирус сканирует сборочные каталоги и кэши Gradle

■ Выставили облегчённые политики с исключениями → графики стабилизировались, шума в данных стало меньше

Что ещё делали

- kart → KSP — ушла генерация Java-стабов, обработка аннотаций быстрее
- Выпилили jetifier — больше не переписываем библиотеки в AndroidX на каждой сборке
- Выпилили material-icons-extended — тысячи иконок шли через компиляцию, а использовали одну



Что не сработало

■ Параллельная конфигурация (`org.gradle.configuration-cache.parallel`) — замедлила этап конфигурации

■ Разделение design system / core-модуля — эффекта не дало

Каждый провал обошёлся дёшево — именно потому, что эффект было чем измерить



Денежки: СКОЛЬКО СЭКОНОМИЛИ

×2.45

быстрее — с поправкой на рост кода
(без поправки ×2.06)

317 ч

в неделю ожидания
≈ 7.9 разработчика

~27 млн ₽

в год

Период оценки: 27.04 – 12.07.2026.

Неделя «до»: 27.04–03.05,
неделя «после»: 06.07–12.07.

Только dev-сборки tc-mono

Как считали

Среднее время сборки до: 27.04–03.05 → 3.38 мин
За 10 недель код вырос на +19% → без оптимизаций
сегодня было бы 4.02 мин
Среднее время сборки после: 06.07–12.07 → 1.64 мин
Экономия на сборке: $4.02 - 1.64 = 2.38$ мин
 $2.38 \times 8000 \text{ сборок/нед} \div 60 = 317 \text{ ч}$
 $317 \div 40 = 7.9 \text{ FTE} \times 280\,000 \text{ ₽} \times 12$

280 000 ₽ — медианная зарплата мобильного разработчика,
Хабр Карьера, I полугодие 2026

Две поправки

Сравниваем не «было → стало», а «что было бы, если бы не трогали»: кодовая база росла всё это время, сборка сама по себе стала бы медленнее.

Метрики покрывают ~85% коммитеров монорепо →
оценка занижена примерно на 15%, реально ~31 млн ₽/год.

Ускорение синка сюда не входит — считали только сборки.

Цифра своя у каждого. Важнее формула:
(ожидаемое без оптимизаций – реальное) × число сборок

Что бы я сделал по-другому

01

Предупреждать команды
об изменениях

56



Что бы я сделал по-другому

01

Предупреждать команды
об изменениях

02

Сделать плагин
максимально незаметным



Что бы я сделал по-другому

01

Предупреждать команды
об изменениях

02

Сделать плагин
максимально незаметным

03

Сразу покрыть
алертами



Что бы я сделал по-другому

01

Предупреждать команды
об изменениях

02

Сделать плагин
максимально незаметным

03

Сразу покрыть
алертами

04

Вливать оптимизации по одной
и смотреть на результат



Спасибо!

Вопросы?

