Engineer

OpenTelemetry

єдиний незалежний спосіб надсилати телеметрію

Що це таке

OpenTelemetry — це відкритий стандарт разом із бібліотеками та інструментами, який дає змогу програмам під час роботи видавати трейси, метрики й логи та передавати їх системі, що їх зберігає і показує. Інженери часто скорочують назву до OTel. Проєкт належить до Cloud Native Computing Foundation і виник зі злиття двох давніших ініціатив, OpenTracing та OpenCensus.

Найважливіше — розуміти межі цього стандарту. OpenTelemetry не є бекендом спостережуваності: він не зберігає даних, не малює дашбордів і не надсилає сповіщень. Цим займаються Jaeger, Prometheus і комерційні платформи. Тож питання «OpenTelemetry чи Prometheus?» порівнює трубу з пунктом призначення, і відповідь зазвичай — обидва: OpenTelemetry там, де дані виникають, і бекенд на вибір команди там, де вони зберігаються.

Проєкт складається з кількох частин, які доповнюють одна одну:

- **специфікація**, що визначає API, SDK і формат даних для всіх мов;
- **OTLP**, протокол, який передає телеметрію через gRPC або HTTP у кодуванні Protocol Buffers;
- **семантичні конвенції**, тобто спільні назви для спільних речей, щоб маршрут HTTP чи запит до бази даних мали однакові позначення в Go, Java і Python;
- **API та SDK для кожної мови**, а з ними бібліотеки інструментування й автоматичне інструментування для поширених фреймворків;
- **Collector**, окремий процес, який приймає телеметрію, обробляє її та пересилає далі.

З усіх сигналів найбільше пояснень потребують трейси. Трейс — це шлях одного запиту крізь усі сервіси, яких він торкається. Кожен крок на цьому шляху є спаном: назва, час початку, тривалість, набір атрибутів та ідентифікатор спану, що спричинив цей крок. Коли запит переходить від одного сервісу до наступного, ідентифікатор мандрує разом із ним у заголовку, типово в тому, який визначає W3C Trace Context:

```
traceparent: 00-a0892f3577b34da6a3ce929d0e0e4736-f03067aa0ba902b7-01
```

Чотири поля — це версія формату, ідентифікатор трейсу, спільний для всіх спанів запиту, ідентифікатор спану, який зробив виклик, і прапорці; кінцеве `01` означає, що трейс відібрано для збереження. Метрики й логи доповнюють картину, а четвертий сигнал, baggage, несе разом із запитом невеликі пари «ключ — значення», щоб сервіс за три переходи далі теж міг їх побачити.

Застосуйте поняття до рішення

Застосуйте «Що таке OpenTelemetry?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Моніторинг і сповіщення, які попереджають, поки збій ще малий, і цифри, що показують, як часто так і залишається.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Запустила перший дашборд спостережуваності — перший огляд власних систем компанії

Запустила перший дашборд спостережуваності Зв'язатися з нами →

Чому це важливо

Зберігання — найпомітніша стаття витрат на спостережуваність. Значно важче позбутися іншої: інструментування. Хтось має додати виклики в кожен сервіс, кожною мовою, і підтримувати їх у міру того, як змінюється код. Коли кожен постачальник мав власного агента й власний API, зміна постачальника означала зробити всю цю роботу заново. З OpenTelemetry код звертається до одного API, а вибір бекенду переходить у конфігурацію: експортер у SDK або маршрут у Collector. Проєкт формулює принцип прямо: дані, які створює система, належать тому, хто її експлуатує.

Ця власність практична, а не лише договірна. Collector стоїть між застосунками та бекендом, тож саме тут зручно фільтрувати, семплювати, перейменовувати чи вилучати атрибути, перш ніж щось вийде назовні. Тут

же той самий потік можна надсилати у два бекенди, поки триває переїзд. Перехід від одного постачальника до іншого стає зміною однієї конфігурації Collector, а не сотні сервісів.

Обсяг — ось витрата, яку варто планувати. Кожен запит може породити десятки спанів, і на справжньому трафіку рахунок за зберігання їх усіх здатен перерости вартість самого сервісу, за яким стежать. Типовий семплер SDK, `ParentBased(root=AlwaysOn)`, відбирає кожен новий трейс і повторює рішення викликача для решти, тож у стандартному налаштуванні зберігається все. Скільки залишати, команда має вирішити свідомо.

Зрілість залежить від сигналу та мови. OTLP стабільний для трейсів, метрик і логів, але SDK кожної мови рухається у своєму темпі, а сторінка статусу проєкту показує, на якому етапі кожен із них. Перегляньте її, перш ніж прив'язувати до OTel сервіс, написаний менш поширеною мовою.

І потрібен він далеко не завжди. Окремий застосунок зі структурованими логами та кількома метриками мало що виграє від розподіленого трасування. OpenTelemetry окуповується тоді, коли запит перетинає межі процесів, проходячи крізь фронтенд, API, чергу й базу даних, а щоб відповісти на питання «куди подівся час?», доводиться збирати нараду.

Як це виглядає на практиці

На одній платформі моніторинг збудували самі замість купити, і OpenTelemetry відповідає в ньому за трасування. Трасування працює через OTLP, тож повільний запит можна простежити крізь фронтенд, API і базу даних, а не вгадувати, де він застряг. Поруч працює власний конвеєр помилок із двох сервісів, errmon та ingest. Він очищує дані перед збереженням, групує помилки в повторювані проблеми й виявляє сплески та регресії з періодом охолодження, щоб один невдалий деплой не піднімав нікого сорок разів. Ще він відновлює символічні імена в мініфікованих стеках викликів із фронтенду й запускає синтетичний heartbeat, який доводить, що сам конвеєр живий.

Результати потрапляють у власну базу даних платформи як події помилок, групи помилок, вхідна черга, затримка API та зразки стеків. Оператори працюють з 11 представленнями над цими даними, і одне з них присвячене цілям рівня обслуговування. Сторінка чесно називає ціну: побудова забрала реальний час і додала ще одну систему, яку треба підтримувати, а куплений

трекер запрацював би того ж дня. Натомість чутливі дані залишаються на платформі. А правила сповіщень написано саме під цю платформу, а не під якусь типову.

Де виникають проблеми

Трейс обривається на межі. Контекст легко проходить HTTP-виклики, бо бібліотеки інструментування записують і читають заголовок з обох боків. Черга повідомлень, заплановане завдання чи проксі, що відкидає невідомі заголовки, розривають ланцюжок, якщо контекст ніхто не перенесе вручну. Трейс розпадається надвоє, і повільна половина виглядає не пов'язаною із запитом, який її спричинив. Кладіть контекст у метадані повідомлення й перевіряйте, що всі сервіси використовують той самий пропагатор.

Атрибути витікають або розростаються. Атрибут спану додати легко, і найлегше додати якраз не ті значення: адреси електронної пошти, повні URL із токенами в рядку запиту, SQL з підставленими значеннями. Для метрик небезпека інша — кардинальність. Атрибут із мільйонами різних значень, як-от ідентифікатор користувача, множить кількість часових рядів, а з ними й рахунок. Дотримуйтеся семантичних конвенцій, не додавайте ідентифікатори до метрик і чистьте дані в Collector як другу лінію захисту.

Семплювання стається випадково. Head sampling ухвалює рішення на початку запиту, ще до того, як відомо, чи буде він повільним або завершиться помилкою. Тож хто зберігає 1% трейсів, той зберігає й лише близько 1% рідкісних повільних. Tail sampling вирішує, коли бачить спани трейсу разом, і може залишити кожну помилку й кожен повільний запит, але компонент, який це робить, мусить тримати всі ці дані, поки чекає. Це сервіс зі станом, і його потужність теж треба планувати та моніторити.

Конвеєру довіряють наосліп. OTLP підтверджує доставку по одному переходу за раз; наскрізна доставка лежить поза межами протоколу. Collector, якому забракло пам'яті або який не встигає за власною чергою експорту, губить телеметрію, а застосунки, що надсилають у нього дані, працюють далі так, наче все гаразд. Стежте за власними метриками Collector, запускайте більше одного екземпляра й сприймайте прогалину в трейсах як збій, який треба пояснити, а не як тиху годину.

Що почитати далі

- What is OpenTelemetry? — власне визначення проєкту, його складники і те, чим він свідомо не є.
- OTLP Specification — сам протокол: транспорти, кодування, стиснення й підтвердження.
- Trace Context — рекомендація W3C, на якій стоїть заголовок `traceparent`.
- Context propagation — як трейс іде за запитом крізь межі процесів.
- Semantic conventions — спільні назви для атрибутів, метрик і ресурсів.
- Sampling — head і tail sampling і скільки коштує кожен із них в експлуатації.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке OpenTelemetry?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/opentelemetry/