Engineer

Спостережуваність

здатність зрозуміти систему за даними, які вона надсилає

Що це таке

Спостережуваність — це міра того, наскільки команда здатна зрозуміти, що робить працююча система, лише з даних, які вона сама надсилає назовні: не розбираючи її і не знаючи заздалегідь, що саме зламалося. Вступ до спостережуваності від проєкту OpenTelemetry формулює це через запитання: спостережувана система дозволяє ставити їй запитання, не знаючи її внутрішньої будови, зокрема такі, про які ніхто не думав, коли писав код. Головне запитання, на яке вона має відповідати, вступ формулює як «Чому це відбувається?»

Отже, спостережуваність — це властивість системи, а не продукт. Інструмент може зберігати й показувати дані, але спершу ці дані має надіслати застосунок. Документація OpenTelemetry каже про це прямо: щоб система була спостережуваною, її компоненти мають бути інструментовані, тобто їхній код має надсилати сигнали. Далі вона задає практичну планку: застосунок інструментовано як слід, коли розробникам не потрібно додавати нове інструментування, щоб розібратися з проблемою, бо потрібні відомості вже є.

Ці дані називають телеметрією, і OpenTelemetry поділяє їх на сигнали:

- **Трейси** показують шлях одного запиту крізь систему. Трейс складається зі спанів; кожен спан — це одна одиниця роботи з назвою, часом початку й завершення та атрибутами, як-от HTTP-метод і маршрут. Перший спан кореневий і охоплює запит від початку до кінця.
- **Метрики** — це вимірювання, зроблені під час роботи й агреговані в часі: частка помилок, кількість запитів, завантаження процесора.
- **Логи** — це записи подій із позначкою часу. Багато з них не пов’язані з жодним конкретним запитом.

- **Baggage** — це контекстні відомості, які передаються разом із запитом, щоб ними могли скористатися інші сигнали.

Сторінка про сигнали також згадує події (events), окремий різновид логів, і профілі, що фіксують використання ресурсів на рівні коду, як сигнали, які ще розробляються або перебувають на стадії пропозиції.

Перетворює три купи даних на щось, чому можна ставити запитання, саме зв’язок між ними. OpenTelemetry передає контекст між сервісами, тож спан в одному сервісі знає, до якого трейсу належить, а до кожного запису в логах можна додати ідентифікатори активного трейсу й спану. Кожен сигнал також несе ресурс: атрибути, що визначають, хто його створив, наприклад назву сервісу, хост або контейнер. Завдяки цим спільним ключам команда може перейти від метрики до запитів, що за нею стоять, а від запиту — до рядків логів, які він залишив.

Спостережуваність і моніторинг перетинаються, але це не одне й те саме. Моніторинг стежить за умовами, які хтось заздалегідь вирішив вважати важливими, і піднімає тривогу, коли одна з них настає. Спостережуваність — це те, з чим команда працює, коли тривога вже спрацювала, а причина виявилася такою, якої ніхто не передбачав.

Застосуйте поняття до рішення

Застосуйте «Що таке спостережуваність?» 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

Сучасні системи ламаються так, що це важко передбачити. Запит може пройти крізь шлюз, кілька сервісів, чергу й базу даних, перш ніж отримає відповідь, і збій у будь-якій ланці може проявитися як повільність зовсім деінде. Вступ називає розподілене трасування необхідним для таких систем, бо їхні проблеми часто недетерміновані або надто складні, щоб відтворити їх на машині розробника. Коли збій не вдається відтворити, телеметрія, записана під час нього, — єдиний доказ, який є.

Дашборди й тривоги відповідають на запитання, які хтось передбачив. Це цінно, і це покриває збої, які команда вже бачила. Найважчі інциденти — нові, коли питання не в тому, чи зросла частка помилок, а в тому, чому почали падати запити з одного регіону, через одну версію, до одного платіжного провайдера. Якщо для відповіді треба написати новий код, викотити його й чекати, поки проблема повториться, простій триває рівно стільки, скільки цей обхідний шлях.

Спостережуваність також тримає в полі зору користувача. Вступ визначає надійність як те, чи робить сервіс те, чого від нього очікують користувачі, і наводить приклад: інтернет-магазин може бути доступним весь час і все одно ненадійним, якщо натискання «додати чорні черевики до кошика» не завжди додає до кошика чорні черевики. Метрика сервера цього ніколи не покаже. Телеметрія окремих запитів покаже, і саме з неї будують індикатори рівня обслуговування: вимірювання з погляду користувача, які цілі рівня обслуговування потім пов’язують із тим, що потрібно бізнесу.

Нарешті, вона змінює ціну інциденту. Команда, яка може простежити повільний запит до конкретного виклику, що його затримав, витрачає час на виправлення, а не на здогадки, а постмортем після інциденту спирається на докази, а не на спогади.

Як це виглядає на практиці

Візьмімо вигаданий інтернет-магазин. Починаючи з 14:00 невелика частка оформлень замовлення триває понад дві секунди. Нічого не лежить, жодна тривога не спрацювала, і дашборда саме для цього немає. Ось як із цим упоралася б команда, у якої сигнали пов’язані між собою. Усі назви, ідентифікатори й числа нижче вигадані для прикладу.

Перший крок — гістограма тривалості оформлення замовлення в секундах. У текстовому форматі OpenMetrics бакет може нести exemplar після знака `#`: посилання на дані поза метрикою, найчастіше ідентифікатор трейсу, який потрапив у цей бакет. Два бакети виглядають так:

```
checkout_bucket{le="1.0"} 95
checkout_bucket{le="2.5"} 97 # {trace_id="3f1c9a77e2b04d18a6c5e0b9d4127f60"} 2.3
```

Другий крок — перейти за цим ідентифікатором трейсу. Трейс показує, куди пішли 2,3 секунди, спан за спаном:

```
POST /checkout                      2.30 s   service checkout
  SELECT cart                       0.04 s   service checkout
  POST /authorise                   2.21 s   service payments
    event: provider timeout, retrying
  INSERT order                      0.03 s   service checkout
```

Третій крок — знайти рядки логів із тим самим ідентифікатором трейсу. Специфікація OpenTelemetry для логів у форматах, відмінних від власного протоколу проєкту, просить записувати ідентифікатори в JSON як поля верхнього рівня з назвами `trace_id` і `span_id`, малими літерами в шістнадцятковому вигляді:

```
{
    "timestamp": 1793110051.402,
    "severity": "WARN",
    "body": "provider timeout after 2000 ms, retrying",
    "provider": "acquirer-b",
    "trace_id": "3f1c9a77e2b04d18a6c5e0b9d4127f60",
    "span_id": "a91c04e7d2f3b658"
}
```

Три кроки — і запитання, якого ніхто не планував, має відповідь: один із двох платіжних провайдерів почав перевищувати час очікування відповіді, і кожне замовлення, надіслане до нього, чекало на повторну спробу. Тепер за тим самим атрибутом `provider` можна згрупувати всі недавні трейси й підтвердити закономірність, перш ніж хтось торкнеться коду.

Нічого з цього не працює, якщо сервіси не називають себе. SDK OpenTelemetry за замовчуванням задають ресурсному атрибуту `service.name` значення `unknown_service`, і документація радить задавати його явно, у коді або через змінну середовища. Інші атрибути ресурсу додають так само:

```
env OTEL_SERVICE_NAME=checkout \
    OTEL_RESOURCE_ATTRIBUTES=deployment.environment.name=production \
    ./checkout
```

На цьому сайті три роботи з портфоліо показують, як спостережуваність нарощують крок за кроком. Перша описує перший дашборд спостережуваності в компанії, який збирав системні показники, як-от процесор, пам'ять і диск, з віддалених Linux-серверів через SSH; згодом другу версію побудували на Grafana і Prometheus. Робота, що знизила операційний ризик за допомогою Grafana і Prometheus, інструментувала ключові сервіси й обрала метрики, які щось означають, замість чисел для краси. Пізніша робота, власні моніторинг помилок і трасування, додала трасування OpenTelemetry через OTLP, щоб повільний запит можна було простежити крізь фронтенд, API і базу даних, а не вгадувати, а поруч із ним працює конвеєр помилок, який очищує вміст перед збереженням.

Де виникають проблеми

Купити замість вбудувати. Новий інструмент, у який не надходить нічого корисного, — це порожня кімната. Спостережуваність походить з інструментування в коді й бібліотеках, тож закладайте час розробників на те, щоб додати спани, атрибути й поля логів, а не лише гроші на ліцензію.

Три сигнали — три острови. Метрики в одному інструменті, логи в іншому, трейси в третьому, без спільного ключа між ними, — і команда переписує позначки часу з одного екрана на інший. Додавайте ідентифікатори трейсу й спану до кожного запису в логах, прикріплюйте exemplars до гістограм тривалості там, де це підтримують ваші інструменти, і давайте всім сигналам однакові атрибути ресурсу.

Усі сервіси звуться unknown_service. Це значення SDK за замовчуванням, і цілий парк сервісів з однаковою назвою неможливо розрізнити. Задайте `OTEL_SERVICE_NAME` для кожного сервісу. Ресурс прив'язується під час створення провайдерів трейсерів і метрик і згодом не змінюється, тож він має бути правильним із самого запуску.

Інструментовано лише краї. Інструментування без коду (zero-code) — добрий початок: воно фіксує, що відбувається на краях застосунку, у його бібліотеках і в середовищі, де він працює. Але запитання під час інциденту зазвичай про бізнес: який тип клієнта, який провайдер, яке значення мав прапорець функції. Для цього потрібні атрибути, додані в коді. Документація OpenTelemetry називає інструментування в коді доповненням до zero-code, і обидва можуть працювати одночасно.

Інструментувати посеред інциденту. Якщо для відповіді знадобилися новий рядок у логах і нове розгортання, система не була інструментована як слід за мірками самого вступу. Зробіть «що нам довелося додати?» постійним запитанням кожного розбору інциденту, а кожну відповідь — окремим завданням на потім.

Необмежені подробиці не в тому сигналі. Ідентифікатори користувачів, повні URL і тіла запитів належать до спанів і записів у логах, а не до міток метрик, де кожне нове значення створює ще один часовий ряд. Стандарт OpenMetrics обмежує сукупну довжину назв і значень міток в exemplar 128 символами саме для того, щоб exemplars не використовували для перенесення даних спанів, і дозволяє приймачеві взагалі відкидати exemplars. Перш ніж покладатися на ці зв'язки, перевірте, що вони доживають до екрана на всьому шляху від експортера.

Телеметрія, що витікає. Спани й логи фіксують, що було в запитах, а там можуть бути адреси, імена й токени. Вирішіть, які атрибути можна записувати, відкидайте решту перед збереженням і захищайте телеметрію так само ретельно, як базу даних, яку вона описує.

Що почитати далі

- Observability primer — визначення від проєкту OpenTelemetry разом із надійністю, SLI і SLO, спанами та розподіленими трейсами.
- Signals — трейси, метрики, логи й baggage, а також сигнали, які ще розробляються.
- Instrumentation — інструментування в коді та без коду і те, як передавання контексту пов'язує сигнали між собою.
- Resources — атрибути ресурсу, значення `unknown_service` за замовчуванням і змінні середовища, що їх задають.
- Trace Context in non-OTLP Log Formats — стабільна специфікація того, як записувати ідентифікатори трейсу й спану в syslog, звичайний текст і логи у форматі JSON.
- OpenMetrics 1.0 — стандарт формату експонування метрик, зокрема exemplars, з порадами щодо кардинальності.

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 спостережуваність?» 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/observability/