Engineer

OpenTelemetry

ett leverantörsoberoende sätt att skicka telemetri

Vad det är

OpenTelemetry är en standard med öppen källkod, med bibliotek och verktyg som hör till, för att ta fram spår, systemmätvärden och loggar från programvara som körs och lämna dem vidare till det system som lagrar och visar dem. De flesta ingenjörer säger OTel. Projektet hör till Cloud Native Computing Foundation och bildades när två tidigare satsningar, OpenTracing och OpenCensus, slogs ihop till en.

Det mest användbara att veta om det är vad det lämnar utanför. OpenTelemetry är ingen backend för observerbarhet: det lagrar inga data, ritar inga dashboards och skickar inga larm. Jaeger, Prometheus och de kommersiella plattformarna sköter den delen. Frågan ”OpenTelemetry eller Prometheus?” jämför alltså ett rör med en destination, och det vanliga svaret är båda: OpenTelemetry där datan uppstår, en backend som teamet själv väljer där den förvaras.

Projektet består av flera delar som passar ihop:

- en **specifikation** som definierar API:et, SDK:t och datan, för varje språk;
- **OTLP**, protokollet som bär telemetri över gRPC eller HTTP, kodad med Protocol Buffers;
- **semantiska konventioner**, gemensamma namn för gemensamma saker, så att en HTTP-rutt eller en databassats märks på samma sätt i Go, Java och Python;
- ett **API och SDK per språk**, med instrumenteringsbibliotek och automatisk instrumentering för populära ramverk;
- **Collector**, en separat process som tar emot telemetri, bearbetar den och exporterar den vidare.

Av de signaler det bär är spår det som kräver mest förklaring. Ett spår följer ett anrop genom varje tjänst det passerar. Varje steg är ett span: ett namn, en starttid, en varaktighet, en uppsättning attribut och identiteten för det span som orsakade det. När anropet går från en tjänst till nästa följer identiteten med i ett huvudfält, som standard det som W3C Trace Context definierar:

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

De fyra fälten är formatversionen, spår-ID:t som delas av varje span i anropet, ID:t för det anropande spannet och en uppsättning flaggor, där ett avslutande `01` markerar att spåret samplas. Mätvärden och loggar fullbordar bilden, och en fjärde signal, baggage, bär små fakta i form av nyckel och värde längs med anropet, så att en tjänst tre steg bort fortfarande kan se dem.

Använd begreppet i ett beslut

Använd ”Vad är OpenTelemetry?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Byggde företagets första observerbarhetsdashboard — företagets första överblick över de egna systemen

Byggde företagets första observerbarhetsdashboard Hör av er →

Varför det spelar roll

Lagringen är den synliga kostnaden för observerbarhet. Den mer seglivade är instrumenteringen: någon måste lägga in anropen i varje tjänst, i varje språk, och hålla dem aktuella när koden ändras. När varje leverantör levererade sin egen agent och sitt eget API innebar ett leverantörsbyte att göra allt det en gång till. Med OpenTelemetry anropar koden ett enda API, och valet av backend flyttar in i konfigurationen, antingen som en exporter i SDK:t eller som en rutt i Collector. Projektet säger principen rakt ut: den data som ett system genererar tillhör den som driver det.

Det ägandet är lika mycket praktiskt som avtalsmässigt. Collector står mellan applikationerna och backend, vilket gör den till platsen där attribut filtreras, samplas, döps om eller tas bort innan något lämnar organisationen, och där

samma ström kan skickas till två backends medan en migrering pågår. Att byta från en leverantör till en annan blir en ändring i konfigurationen av Collector i stället för i hundra tjänster.

Volymen är kostnaden att planera för. Varje anrop kan ge dussintals span, och vid verklig trafik kan räkningen för att behålla alla växa sig större än tjänsten som övervakas. Standardsamplern i SDK:t, `ParentBased(root=AlwaysOn)`, samplar varje nytt spår och följer anroparens beslut för resten, så från start behålls allt. Hur mycket som ska behållas är ett beslut som ett team fattar med avsikt.

Mognaden skiljer sig mellan signaler och mellan språk. OTLP är stabilt för spår, mätvärden och loggar, medan SDK:t för varje språk rör sig i sin egen takt, och projektets statussida visar var vart och ett står. Läs den innan en tjänst skriven i ett mindre vanligt språk byggs på det.

Och det är inte alltid värt det. En enskild applikation med strukturerade loggar och en handfull mätvärden vinner lite på distribuerad spårning. OpenTelemetry betalar sig när ett anrop korsar processgränser, genom en frontend, ett API, en kö och en databas, och frågan ”vart tog tiden vägen?” kräver ett möte för att besvaras.

Så ser det ut i praktiken

På en plattform byggdes övervakning i egen regi i stället för att köpas, och OpenTelemetry är dess spårningshalva. Spårningen går över OTLP, så att ett långsamt anrop följs genom frontenden, API:t och databasen i stället för att gissas fram. Bredvid den finns en egenutvecklad felpipeline, en errmon-tjänst och en ingest-tjänst, som sanerar nyttolaster innan de lagras, grupperar fel till återkommande problem, upptäcker toppar och regressioner med en karenstid så att en dålig driftsättning inte väcker någon fyrtio gånger, symboliserar minifierade stackspår från frontenden och kör ett syntetiskt hjärtslag för att bevisa att själva pipelinen lever.

Resultaten hamnar i plattformens egen databas som felhändelser, felgrupper, en inkorg, API-latens och stackprov. Operatörerna arbetar i 11 vyer över den datan, en av dem för servicenivåmål. Sidan är rak om priset: att bygga kostade verklig tid och gav ännu en sak att underhålla, och en köpt felspårare hade varit igång samma eftermiddag. I gengäld stannar känsliga data på plattformen. Även larmreglerna skrevs för den här plattformen och inte för en generisk.

Var det går fel

Spåret stannar vid en gräns. Kontexten tar sig lätt över HTTP-anrop, eftersom instrumenteringsbiblioteken skriver och läser huvudfältet på båda sidor. En meddelandekö, ett schemalagt jobb eller en proxy som tar bort okända huvudfält bryter kedjan om ingen bär kontexten över för hand. Spåret delas i två, och den långsamma halvan ser ut att sakna samband med anropet som orsakade den. Lägg kontexten i meddelandets metadata, och kontrollera att varje tjänst kör samma propagator.

Attribut läcker eller exploderar. Ett attribut på ett span är lätt att lägga till, och de lättaste värdena att lägga till är fel värden: e-postadresser, fullständiga URL:er med token i frågesträngen, SQL med de bokstavliga värdena kvar. För mätvärden är faran kardinaliteten. Ett attribut med miljontals olika värden, till exempel ett användar-ID, multiplicerar antalet tidsserier och räkningen med det. Följ de semantiska konventionerna, håll identifierare borta från mätvärden och maskera i Collector som en andra försvarslinje.

Samplingen sker av en slump. Head sampling beslutar i början av ett anrop, innan någon vet om det blir långsamt eller misslyckas, så den som behåller 1 % av spåren behåller också ungefär 1 % av de sällsynta långsamma. Tail sampling beslutar efter att ha sett alla span i ett spår tillsammans och kan behålla varje fel och varje långsamt anrop, men komponenten som gör det måste hålla all den datan medan den väntar. Det är en tjänst med tillstånd och med egen kapacitet att planera och övervaka.

Pipelinen får blind tillit. OTLP kvitterar leveransen ett steg i taget; leverans hela vägen fram ligger utanför dess omfång. En Collector som får slut på minne eller halkar efter med sin exportkö kastar telemetri, och applikationerna som skickar till den fortsätter som om allt vore väl. Bevaka de mätvärden som Collector själv exponerar, kör mer än en och behandla en lucka i spåren som ett fel att förklara snarare än en lugn timme.

Läs mer

- What is OpenTelemetry? — projektets egen definition, dess komponenter och vad det medvetet inte är.
- OTLP Specification — protokollet på tråden: transporter, kodning, komprimering och kvittering.

- Trace Context — rekommendationen från W3C bakom huvudfältet `traceparent`.
- Context propagation — hur ett spår följer ett anrop över processgränser.
- Semantic conventions — de gemensamma namnen för attribut, mätvärden och resurser.
- Sampling — head sampling och tail sampling, och vad var och en kostar att driva.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är OpenTelemetry?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/opentelemetry/