Engineer

OpenTelemetry

én leverandørfri måde at sende telemetri

Hvad det er

OpenTelemetry er en åben standard med tilhørende biblioteker og værktøjer, der får kørende software til at udsende spor, målinger og logs og afleverer dem til det system, der gemmer og viser dem. I daglig tale hedder det OTel. Projektet hører under Cloud Native Computing Foundation og opstod, da to ældre projekter, OpenTracing og OpenCensus, blev slået sammen.

Det vigtigste at vide om det er, hvad det lader ligge. OpenTelemetry er ikke en observability-backend: det gemmer ingen data, tegner ingen dashboards og sender ingen alarmer. Den del klarer Jaeger, Prometheus og de kommercielle platforme. Spørgsmålet »OpenTelemetry eller Prometheus?« sammenligner derfor et rør med en destination, og svaret er som regel begge dele: OpenTelemetry der, hvor data opstår, og en backend efter eget valg der, hvor de opbevares.

Projektet består af flere dele, der passer sammen:

- en **specifikation**, der fastlægger API, SDK og dataformat på tværs af alle sprog;
- **OTLP**, protokollen der fragter telemetrien over gRPC eller HTTP, kodet med Protocol Buffers;
- **semantiske konventioner**, fælles navne for fælles ting, så en HTTP-rute eller en databaseforespørgsel får samme betegnelse i Go, Java og Python;
- et **API og et SDK til hvert sprog** med instrumenteringsbiblioteker og automatisk instrumentering af udbredte frameworks;
- **Collector**, en selvstændig proces, der modtager telemetri, bearbejder den og sender den videre.

Af signalerne kræver spor mest forklaring. Et spor, på engelsk en trace, følger én forespørgsel gennem alle de tjenester, den kommer forbi. Hvert trin er et span: et navn, et starttidspunkt, en varighed, en række attributter og identiteten på det span, der satte trinnet i gang. Når forespørgslen går fra én tjeneste til den næste, rejser identiteten med i en header, som standard den, W3C Trace Context definerer:

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

De fire felter er formatets version, det spor-ID, som alle forespørgslens spans deler, ID'et på det kaldende span og en række flag, hvor et afsluttende `01` betyder, at sporet er udvalgt til at blive gemt. Målinger og logs fuldender billedet, og et fjerde signal, baggage, fører små nøgle-værdi-oplysninger med forespørgslen, så en tjeneste tre led længere nede stadig kan se dem.

Brug begrebet i en beslutning

Brug »Hvad er OpenTelemetry?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Overvågning og alarmer, der siger til, mens fejlen stadig er lille, og tallene, der viser, hvor tit den bliver ved med at være det.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har bygget det første observerbarhedsdashboard — virksomhedens første overblik over sine egne systemer

Har bygget det første observerbarhedsdashboard Kontakt os →

Hvorfor det betyder noget

Lagring er den synlige udgift ved observability. Den mere klæbrige er instrumenteringen: nogen skal tilføje kaldene i hver tjeneste og hvert sprog og holde dem ved lige, efterhånden som koden ændrer sig. Dengang hver leverandør havde sin egen agent og sit eget API, betød et leverandørskifte, at hele arbejdet skulle gøres om. Med OpenTelemetry kalder koden ét API, og valget af backend flytter ud i konfigurationen, enten som en exporter i SDK'et eller som en rute i Collector. Projektet siger princippet ligeud: de data, et system skaber, tilhører den, der driver det.

Ejerskabet er praktisk og ikke kun kontraktligt. Collector står mellem applikationerne og backenden og er derfor stedet, hvor man filtrerer, sampler, omdøber eller fjerner attributter, før noget forlader huset. Det er også her,

den samme strøm kan sendes til to backends, mens en migrering står på. At skifte leverandør bliver en ændring i én Collector-konfiguration frem for i hundrede tjenester.

Mængden er den udgift, man skal planlægge efter. Hver forespørgsel kan give snesevis af spans, og ved rigtig trafik kan regningen for at gemme dem alle vokse sig større end den tjeneste, der overvåges. SDK'ets standardsampler, `ParentBased(root=AlwaysOn)`, udvælger hvert nyt spor og følger kalderens beslutning for resten, så i standardopsætningen gemmes alt. Hvor meget der skal beholdes, er en beslutning, man må træffe bevidst.

Modenheden svinger med signal og sprog. OTLP er stabil for spor, målinger og logs, mens hvert sprogs SDK bevæger sig i sit eget tempo, og projektets statusside viser, hvor hvert enkelt står. Læs den, før en tjeneste skrevet i et mindre udbredt sprog bindes til det.

Og det er langtfra altid umagen værd. En enkelt applikation med strukturerede logs og en håndfuld målinger får ikke meget ud af distribueret sporing. OpenTelemetry tjener sig ind, når en forespørgsel krydser procesgrænser, gennem en frontend, et API, en kø og en database, og spørgsmålet »hvor blev tiden af?« kræver et møde at besvare.

Sådan ser det ud i praksis

På én platform blev overvågningen bygget i stedet for købt, og OpenTelemetry står for sporingsdelen. Sporingen kører over OTLP, så en langsom forespørgsel kan følges gennem frontend, API og database i stedet for at blive gættet på. Ved siden af kører en hjemmebygget fejlpipeline, en errmon-tjeneste og en ingest-tjeneste. Den renser payloads, før de gemmes, samler fejl i tilbagevendende problemer og opdager spidser og regressioner med en cooldown, så ét dårligt deploy ikke kalder nogen ud fyrre gange. Den symbolicerer også minificerede stack traces fra frontenden og kører et syntetisk hjerteslag, der beviser, at pipelinen selv er i live.

Resultaterne havner i platformens egen database som fejlhændelser, fejlgrupper, en indbakke, API-latenstid og stack-udsnit. Driften arbejder ud fra 11 visninger oven på de data, heraf én til service level objectives. Siden lægger ikke skjul på prisen: det kostede reel tid at bygge og blev én ting mere at

vedligeholde, og en købt fejltracker ville have kørt samme eftermiddag. Til gengæld bliver følsomme data på platformen. Alarmreglerne er desuden skrevet til netop denne platform og ikke til en generisk.

Hvor det går galt

Sporet stopper ved en grænse. Kontekst krydser HTTP-kald uden besvær, fordi instrumenteringsbibliotekerne skriver og læser headeren i begge ender. En beskedkø, et planlagt job eller en proxy, der fjerner ukendte headers, bryder kæden, medmindre nogen bærer konteksten over i hånden. Sporet deles i to, og den langsomme halvdel ser ud til at være uden forbindelse til den forespørgsel, der udløste den. Læg konteksten i beskedens metadata, og tjek, at alle tjenester bruger samme propagator.

Attributter lækker eller eksploderer. En attribut på et span er nem at tilføje, og de nemmeste værdier er de forkerte: e-mailadresser, fulde URL'er med tokens i forespørgselsstrengen, SQL med de konkrete værdier stadig i. For målinger hedder faren kardinalitet. En attribut med millioner af forskellige værdier, for eksempel et bruger-ID, ganger antallet af tidsserier op og regningen med. Følg de semantiske konventioner, hold identifikatorer ude af målinger, og rens data i Collector som en ekstra sikring.

Sampling sker ved et uheld. Head sampling beslutter ved forespørgslens start, før nogen ved, om den bliver langsom eller fejler. Den, der beholder 1 % af sporene, beholder altså også kun omkring 1 % af de sjældne langsomme. Tail sampling beslutter, når et spors spans er set samlet, og kan beholde hver fejl og hver langsom forespørgsel, men komponenten, der gør det, må holde alle de data, mens den venter. Den er en tjeneste med tilstand og med sin egen kapacitet, der skal planlægges og overvåges.

Pipelinen får blind tillid. OTLP kvitterer for levering ét led ad gangen; levering hele vejen igennem ligger uden for protokollens område. En Collector, der løber tør for hukommelse eller sakker bagud med sin eksportkø, taber telemetri, mens applikationerne, der sender til den, fortsætter, som om alt var i orden. Overvåg Collectors egne målinger, kør mere end én, og behandl et hul i sporene som en fejl, der skal forklares, og ikke som en stille time.

Læs mere

- What is OpenTelemetry? — projektets egen definition, dets dele, og hvad det bevidst ikke er.
- OTLP Specification — selve protokollen: transport, kodning, komprimering og kvittering.
- Trace Context — W3C-anbefalingen bag `traceparent` -headeren.
- Context propagation — hvordan et spor følger en forespørgsel på tværs af procesgrænser.
- Semantic conventions — de fælles navne for attributter, målinger og ressourcer.
- Sampling — head og tail sampling, og hvad hver af dem koster at drive.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er OpenTelemetry?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/opentelemetry/