Engineer

OpenTelemetry

💓 én leverandørnøytral måte å sende telemetri på

Hva det er

OpenTelemetry er en åpen standard, med tilhørende biblioteker og verktøy, for å produsere spor (traces), metrikker og logger fra programvare som kjører, og levere dem til det systemet som lagrer og viser dem. De fleste ingeniører kaller det OTel. Det er et prosjekt under Cloud Native Computing Foundation (CNCF), og det ble til da to tidligere initiativer, OpenTracing og OpenCensus, slo seg sammen.

Det nyttigste å vite om OpenTelemetry er hva det lar være. Det er ikke en backend for observerbarhet: det lagrer ingen data, tegner ingen dashbord og sender ingen varsler. Den delen tar Jaeger, Prometheus og de kommersielle plattformene seg av. Spørsmålet «OpenTelemetry eller Prometheus?» sammenligner derfor transporten med bestemmelsesstedet, og det vanlige svaret er begge deler: OpenTelemetry der dataene oppstår, og en backend teamet selv velger, der de lagres.

Prosjektet består av flere deler som passer sammen:

- en **spesifikasjon** som definerer API-et, SDK-en og dataene, på tvers av alle språk;
- **OTLP**, protokollen som frakter telemetri over gRPC eller HTTP, kodet med Protocol Buffers;
- **semantiske konvensjoner**, felles navn på felles ting, så en HTTP-rute eller en databasesetning merkes likt i Go, Java og Python;
- et **API og en SDK per språk**, med instrumenteringsbiblioteker og automatisk instrumentering for populære rammeverk;
- **Collector**, en egen prosess som tar imot telemetri, behandler den og eksporterer den videre.

Av signalene prosjektet frakter, er det spor som krever mest forklaring. Et spor følger én forespørsel gjennom alle tjenestene den berører. Hvert steg er et spenn (span): et navn, et starttidspunkt, en varighet, et sett attributter og identiteten til spennet som utløste det. Når forespørselen går fra én tjeneste til den neste, følger den identiteten med i en header, som standard den W3C Trace Context definerer:

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

De fire feltene er formatversjonen, spor-ID-en som alle spennene i forespørselen deler, ID-en til spennet som kalte, og et sett flagg, der en avsluttende `01` markerer at sporet er samplet. Metrikker og logger fullfører bildet, og et fjerde signal, baggage, bærer små nøkkel–verdi-opplysninger med forespørselen, så en tjeneste tre ledd unna fortsatt kan se dem.

Bruk begrepet i en beslutning

Bruk «Hva er OpenTelemetry?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Overvåking og varsler som sier fra mens en feil fortsatt er liten, og tallene som viser hvor ofte den holder seg liten.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Bygde selskapets første observerbarhetsdashbord — selskapets første innsyn i egne systemer

Bygde selskapets første observerbarhetsdashbord Ta kontakt →

Hvorfor det er viktig

Lagring er den synlige kostnaden ved observerbarhet. Den som henger lenger ved, er instrumentering: noen må legge inn kallene i hver tjeneste, i hvert språk, og holde dem oppdatert mens koden endrer seg. Da hver leverandør leverte sin egen agent og sitt eget API, betydde et leverandørbytte å gjøre alt dette på nytt. Med OpenTelemetry kaller koden ett API, og valget av backend flyttes inn i konfigurasjonen, enten som en eksportør i SDK-en eller som en rute i Collector. Prosjektet sier prinsippet rett ut: dataene et system genererer, tilhører den som drifter det.

Det eierskapet er praktisk, ikke bare kontraktsmessig. Collector står mellom applikasjonene og backenden, og det gjør den til stedet der attributter filtreres, samples, får nye navn eller fjernes før noe forlater huset, og der den samme

strømmen kan sendes til to backender mens et leverandørbytte pågår. Å flytte fra én leverandør til en annen blir da en endring i én Collector-konfigurasjon i stedet for i hundre tjenester.

Volum er kostnaden dere må planlegge for. Hver forespørsel kan gi dusinvis av spenn, og ved ekte trafikk kan regningen for å beholde alle sammen vokse seg større enn tjenesten som overvåkes. SDK-ens standardsampler, `ParentBased(root=AlwaysOn)`, sampler hvert nytt spor og følger kallerens avgjørelse for resten, så uten videre oppsett beholdes alt. Hvor mye som skal beholdes, er en beslutning et team må ta med vilje.

Modenheten varierer med signal og språk. OTLP er stabilt for spor, metrikker og logger, mens SDK-en i hvert språk går i sitt eget tempo, og prosjektets statusside viser hvor hver av dem står. Les den før dere binder dere til OpenTelemetry for en tjeneste skrevet i et mindre vanlig språk.

Og det er ikke alltid verdt det. Én enkelt applikasjon med strukturerte logger og en håndfull metrikker har lite å hente på distribuert sporing. OpenTelemetry betaler seg når en forespørsel krysser prosessgrenser, gjennom en frontend, et API, en kø og en database, og spørsmålet «hvor ble tiden av?» krever et møte for å bli besvart.

Slik ser det ut i praksis

På én plattform ble overvåkingen bygd i stedet for kjøpt, og OpenTelemetry er sporingshalvdelen av den. Sporingen går over OTLP, så en treg forespørsel følges på tvers av frontenden, API-et og databasen i stedet for å gjettes på. Ved siden av går en egenutviklet feilpipeline, en errmon-tjeneste og en innsamlingstjeneste, som renser nyttelaster før de lagres, grupperer feil i gjentakende problemer, oppdager topper og regresjoner med en avkjølingsperiode så én dårlig produksjonssetting ikke varsler noen førti ganger, symboliserer minifiserte kallstakker fra frontenden og kjører et syntetisk hjerteslag for å bevise at selve pipelinen er i live.

Resultatene havner i plattformens egen database som feilhendelser, feilgrupper, en innboks, API-latens og stakkprøver. Operatørene jobber fra 11 visninger over de dataene, én av dem for tjenestenivåmål. Siden er tydelig på prisen: å bygge kostet reell tid og ga én ting til å vedlikeholde, og en kjøpt feilsporer ville vært i

drift samme ettermiddag. Til gjengjeld blir sensitive data værende på plattformen. Også varslingsreglene ble skrevet for akkurat denne plattformen i stedet for en generisk en.

Hvor det går galt

Sporet stopper ved en grense. Konteksten krysser HTTP-kall uten problemer, fordi instrumenteringsbibliotekene skriver og leser headeren på begge sider. En meldingskø, en planlagt jobb eller en proxy som fjerner ukjente headere, bryter kjeden med mindre noen bærer konteksten over for hånd. Sporet deles i to, og den trege halvdelen ser ut til å være uten sammenheng med forespørselen som forårsaket den. Legg konteksten i meldingens metadata, og sjekk at alle tjenestene bruker den samme propagatoren.

Attributtene lekker eller eksploderer. Et attributt på et spenn er lett å legge til, og de enkleste verdiene å legge til er de gale: e-postadresser, hele URL-er med tokener i spørrestrengen, SQL med de bokstavelige verdiene fortsatt i seg. For metrikker er faren kardinalitet. Et attributt med millioner av ulike verdier, f.eks. en bruker-ID, mangedobler antallet tidsserier, og regningen med dem. Følg de semantiske konvensjonene, hold identifikatorer unna metrikker, og masker opplysninger i Collector som en ekstra forsvarslinje.

Samplingen skjer ved et uhell. Head sampling avgjør ved starten av en forespørsel, før noen vet om den blir treg eller feiler, så den som beholder 1 % av sporene, beholder også bare ca. 1 % av de sjeldne trege. Tail sampling avgjør etter å ha sett spennene i et spor samlet og kan beholde hver feil og hver treg forespørsel, men komponenten som gjør det, må holde på alle de dataene mens den venter. Det er en tjeneste med tilstand, med egen kapasitet som må planlegges og overvåkes.

Pipelinen får blind tillit. OTLP kvitterer for levering ett ledd om gangen; levering fra ende til ende ligger utenfor det protokollen dekker. En Collector som går tom for minne eller kommer på etterskudd med eksportkøen, mister telemetri, og applikasjonene som sender til den, fortsetter som om alt var i orden. Følg med på Collectors egne metrikker, kjør mer enn én, og behandle et hull i sporene som en feil som må forklares, ikke som en rolig time.

Les mer

- What is OpenTelemetry? — prosjektets egen definisjon, komponentene og hva det med vilje ikke er.
- OTLP Specification — protokollen på ledningen: transport, koding, komprimering og kvittering.
- Trace Context — W3C-anbefalingen bak headeren `traceparent`.
- Context propagation — hvordan et spor følger en forespørsel over prosessgrenser.
- Semantic conventions — de felles navnene på attributter, metrikker og ressurser.
- Sampling — head og tail sampling, og hva hver av dem koster å drifte.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er OpenTelemetry?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/opentelemetry/