Engineer

Observerbarhed

evnen til at forstå et system ud fra de data, det sender ud

Hvad det er

Observerbarhed er graden af, hvor godt et team kan forstå, hvad et kørende system foretager sig, udelukkende ud fra de data, systemet selv sender ud, uden at skille det ad og uden på forhånd at vide, hvad der er gået galt. OpenTelemetry-projektets introduktion formulerer det som et spørgsmål om spørgsmål: Et observerbart system lader dig spørge ind til det uden at kende dets indre, også med spørgsmål, som ingen tænkte på, da softwaren blev skrevet. Introduktionen opsummerer det spørgsmål, systemet skal kunne besvare, som »Hvorfor sker det her?«

Dermed er observerbarhed en egenskab ved systemet, ikke et produkt. Et værktøj kan gemme og vise data, men data skal først komme fra applikationen. OpenTelemetrys dokumentation siger det ligeud: For at et system kan være observerbart, skal dets komponenter være instrumenteret, det vil sige, at deres kode sender signaler ud. Den sætter også en praktisk målestok. En applikation er ordentligt instrumenteret, når udviklerne ikke behøver at tilføje mere instrumentering for at fejlsøge et problem, fordi oplysningerne allerede er der.

Data kaldes telemetri, og OpenTelemetry deler dem op i signaler:

- **Spor** (traces) viser den vej, én forespørgsel tager gennem systemet. Et spor består af spans, som hver er en enkelt arbejdsenhed med et navn, et start- og sluttidspunkt og attributter som HTTP-metoden og ruten. Det første span er roden og dækker forespørgslen fra start til slut.
- **Metrikker** er målinger, der tages under kørslen og samles over tid: fejlrate, antal forespørgsler pr. tidsenhed, processorforbrug.
- **Logs** er tidsstemplede registreringer af hændelser. Mange af dem hører ikke til en bestemt forespørgsel.
- **Baggage** er kontekstoplysninger, der sendes med en forespørgsel, så andre signaler kan bruge dem.

Signalsiden nævner desuden events, en særlig type log, og profiler, der registrerer ressourceforbrug på kodeniveau, som signaler, der er under udvikling eller på forslagsstadiet.

Det, der gør tre bunker data til noget, et team kan stille spørgsmål til, er sammenkædning. OpenTelemetry sender kontekst videre mellem tjenester, så et span i én tjeneste ved, hvilket spor det hører til, og det kan stemple id'erne for det aktive spor og span på hver logpost. Hvert signal bærer også en ressource: attributter, der identificerer det, der har produceret signalet, for eksempel tjenestens navn, værten eller containeren. Med de fælles nøgler kan et team gå fra en metrik til forespørgslerne bag den og fra en forespørgsel til de loglinjer, den efterlod.

Observerbarhed og overvågning overlapper hinanden, men de er ikke det samme. Overvågning holder øje med forhold, som nogen på forhånd har besluttet at interessere sig for, og slår alarm, når et af dem opstår. Observerbarhed er det, teamet har at arbejde med, når alarmen er gået, og årsagen er noget, ingen havde forudset.

Brug begrebet i en beslutning

Brug »Hvad er observerbarhed?« 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

Moderne systemer fejler på måder, der er svære at forudsige. En forespørgsel kan passere en gateway, flere tjenester, en kø og en database, før den får svar, og en fejl i én af dem kan vise sig som langsomhed et helt andet sted. Introduktionen kalder distribueret sporing afgørende for den slags systemer, fordi deres problemer ofte er ikke-deterministiske eller for komplicerede til at genskabe på en udviklers egen maskine. Når en fejl ikke kan genskabes, er den telemetri, der blev optaget, mens den skete, det eneste bevis, der findes.

Dashboards og alarmer besvarer spørgsmål, som nogen har forudset. Det er værdifuldt, og det dækker de fejl, teamet har set før. De svære driftshændelser er de nye, hvor spørgsmålet ikke er, om fejlraten steg, men hvorfor forespørgsler fra én region, gennem én version, til én betalingsudbyder begyndte at fejle. Hvis svaret kræver, at man skriver ny kode, udruller den og venter på, at problemet kommer igen, varer nedbruddet lige så længe som den omvej.

Observerbarhed holder også brugeren i fokus. Introduktionen definerer pålidelighed som et spørgsmål om, hvorvidt tjenesten gør det, brugerne forventer, og giver et eksempel: En webbutik kan være oppe hele tiden og alligevel være upålidelig, hvis et klik for at lægge sorte sko i kurven ikke altid lægger sorte sko i kurven. Det ville en måling på serverniveau aldrig vise. Telemetri fra de enkelte forespørgsler kan, og det er den, serviceniveauindikatorer bygges af: målinger set fra brugerens side, som serviceniveaumål derefter knytter til det, forretningen har brug for.

Endelig ændrer det, hvad en driftshændelse koster. Et team, der kan følge en langsom forespørgsel til netop det kald, der holdt den tilbage, bruger tiden på at rette og ikke på at gætte, og den efterfølgende postmortem kan henvise til beviser i stedet for erindringer.

Sådan ser det ud i praksis

Forestil dig en opdigtet webbutik. Siden klokken 14 tager en lille andel af forløbene i kassen mere end to sekunder. Intet er nede, ingen alarm er gået, og der findes intet dashboard til netop dette. Sådan ville et team med sammenkædede signaler gribe det an. Alle navne, id'er og tal nedenfor er opfundet til eksemplet.

Første skridt er histogrammet over svartiden i kassen, målt i sekunder. I OpenMetrics' tekstformat kan en bucket have et exemplar efter et `#` : en henvisning til data uden for metrikken, oftest id'et på et spor, der landede i den bucket. To af dens buckets ser sådan ud:

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

Andet skridt følger det spor-id. Sporet viser, hvor de 2,3 sekunder gik hen, span for span:

```
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
```

Tredje skridt henter de loglinjer, der bærer samme spor-id. OpenTelemetrys specifikation for logs i andre formater end projektets egen protokol beder om, at id'erne i JSON skrives som felter på øverste niveau med navnene `trace_id` og `span_id`, med små bogstaver og hex-kodet:

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

Tre skridt, og det spørgsmål, ingen havde planlagt, har et svar: Den ene af to betalingsudbydere begyndte at svare for sent, og hvert forløb i kassen, som brugte den udbyder, ventede på et nyt forsøg. Den samme attribut, `provider`, kan nu bruges til at gruppere alle nylige spor, så mønstret er bekræftet, før nogen rører koden.

Intet af det virker, hvis tjenesterne ikke siger, hvem de er. OpenTelemetry-SDK'erne sætter som standard ressourceattributten `service.name` til `unknown_service`, og dokumentationen anbefaler at sætte den eksplicit, i koden eller via en miljøvariabel. Andre ressourceattributter kan tilføjes på samme måde:

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

På dette site viser tre porteføljeopgaver, hvordan observerbarhed bygges op i trin. Den ene beskriver en virksomheds første observerbarhedsdashboard, som samlede systemmålinger som CPU, hukommelse og disk ind fra fjerne Linux-servere over SSH; en anden version blev senere bygget på Grafana og

Prometheus. Arbejdet med at mindske driftsrisikoen med Grafana og Prometheus instrumenterede de vigtigste tjenester og valgte metrikker, der betød noget, frem for tal, der kun ser flotte ud. En senere opgave, egen fejlovervågning og tracing, tilføjede sporing med OpenTelemetry over OTLP, så en langsom forespørgsel kan følges på tværs af frontenden, API'et og databasen i stedet for at blive gættet på, sammen med en fejlpipeline, der renser indholdet, før det gemmes.

Hvor det går galt

At købe det i stedet for at bygge det ind. Et nyt værktøj, som intet brugbart løber ind i, er et tomt rum. Observerbarhed kommer fra instrumentering i koden og bibliotekerne, så sæt udviklertid af til at tilføje spans, attributter og logfelter, ikke kun penge til licensen.

Tre signaler, tre øer. Metrikker i ét værktøj, logs i et andet og spor i et tredje, uden en fælles nøgle imellem, efterlader teamet med at kopiere tidsstempler fra den ene skærm til den anden. Skriv spor- og span-id'erne i hver logpost, sæt exemplars på histogrammer over svartid, hvor værktøjerne understøtter det, og giv alle signaler de samme ressourceattributter.

Alle tjenester hedder unknown_service. Det er SDK'ens standard, og en flåde af tjenester med samme navn kan ikke skelnes fra hinanden. Sæt `OTEL_SERVICE_NAME` for hver tjeneste. Ressourcen knyttes til, når tracer- og metric-providerne oprettes, og kan ikke ændres bagefter, så den skal være rigtig fra opstarten.

Kun kanterne er instrumenteret. Zero-code-instrumentering er en god start og registrerer, hvad der sker i kanten af en applikation: i dens biblioteker og i det miljø, den kører i. Spørgsmålene under en driftshændelse handler som regel om forretningen: hvilken kundetype, hvilken udbyder, hvilken værdi et feature-flag havde. Det kræver attributter, der tilføjes i koden. OpenTelemetrys dokumentation kalder kodebaseret instrumentering et supplement til zero-code, og de to kan køre samtidig.

At instrumentere midt i driftshændelsen. Hvis svaret krævede en ny loglinje og en udrulning, var systemet ikke ordentligt instrumenteret efter introduktionens egen målestok. Gør »hvad måtte vi tilføje?« til et fast spørgsmål i hver gennemgang efter en driftshændelse, og behandl hvert svar som en opfølgningsopgave.

Ubegrænsede detaljer i det forkerte signal. Bruger-id'er, fulde URL'er og indholdet af forespørgsler hører hjemme på spans og logposter, ikke i metrikkernes labels, hvor hver ny værdi skaber endnu en tidsserie. OpenMetrics-standarden begrænser et exemplars labelnavne og -værdier til 128 tegn i alt, netop for at forhindre, at exemplars bruges til at bære span-data, og den tillader en modtager helt at kassere exemplars. Kontrollér, at koblingerne overlever hele vejen fra eksportøren til skærmen, før I regner med dem.

Telemetri, der lækker. Spans og logs registrerer, hvad forespørgsler indeholdt, og det kan omfatte adresser, navne og tokens. Beslut, hvilke attributter der må registreres, fjern resten, før det gemmes, og beskyt telemetrien lige så omhyggeligt som den database, den beskriver.

Læs mere

- Observability primer — OpenTelemetry-projektets definition med pålidelighed, SLI'er og SLO'er, spans og distribuerede spor.
- Signals — spor, metrikker, logs og baggage samt de signaler, der stadig er under udvikling.
- Instrumentation — kodebaseret og zero-code-instrumentering, og hvordan kontekst, der sendes videre, binder signalerne sammen.
- Resources — ressourceattributter, standardværdien `unknown_service` og de miljøvariabler, der sætter dem.
- Trace Context in non-OTLP Log Formats — den stabile specifikation for, hvordan spor- og span-id'er skrives i syslog, almindelig tekst og JSON-logs.
- OpenMetrics 1.0 — standarden for eksponeringsformatet, herunder exemplars og dens råd om kardinalitet.

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 observerbarhed?« 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/observability/