Engineer

Observerbarhet

evnen til å forstå et system ut fra dataene det sender ut

Hva det er

Observerbarhet er i hvilken grad et team kan forstå hva et kjørende system holder på med utenfra, bare ut fra dataene systemet selv sender ut, uten å åpne det og uten å vite på forhånd hva som har gått galt. OpenTelemetry-prosjektets innføring beskriver det med spørsmål: Et observerbart system lar deg stille spørsmål om det uten å kjenne de indre detaljene, også spørsmål ingen tenkte på da programvaren ble skrevet. Innføringen oppsummerer spørsmålet systemet skal kunne svare på, som «Hvorfor skjer dette?»

Dermed er observerbarhet en egenskap ved systemet, ikke et produkt. Et verktøy kan lagre og vise data, men dataene må først komme fra applikasjonen. OpenTelemetrys dokumentasjon sier det rett ut: Skal et system være observerbart, må komponentene være instrumentert, det vil si at koden deres sender ut signaler. Den setter også en praktisk målestokk. En applikasjon er skikkelig instrumentert når utviklerne ikke trenger å legge til mer instrumentering for å feilsøke et problem, fordi opplysningene allerede finnes.

Dataene kalles telemetri, og OpenTelemetry deler dem inn i signaler:

- **Spor** (traces) viser veien én forespørsel tar gjennom systemet. Et spor består av span, der hvert span er én arbeidsenhet med et navn, et start- og sluttidspunkt og attributter som HTTP-metoden og ruten. Det første spanet er roten og dekker forespørselen fra start til slutt.
- **Metrikker** er målinger som tas mens systemet kjører, og som samles over tid: feilrate, antall forespørsler per tidsenhet, prosessorbruk.
- **Logger** er tidsstemplede registreringer av hendelser. Mange av dem hører ikke til noen bestemt forespørsel.
- **Baggage** er kontekstopplysninger som sendes med en forespørsel, slik at andre signaler kan bruke dem.

Signalsiden nevner også events, en egen type logg, og profiler, som registrerer ressursbruk på kodenivå, som signaler under utvikling eller på forslagsstadiet.

Det som gjør tre hauger med data til noe et team kan stille spørsmål til, er sammenkobling. OpenTelemetry sender kontekst videre mellom tjenester, slik at et span i én tjeneste vet hvilket spor det hører til, og det kan stemple ID-ene til det aktive sporet og spanet på hver loggpost. Hvert signal bærer også en ressurs: attributter som identifiserer det som produserte signalet, for eksempel tjenestenavnet, verten eller containeren. Med disse felles nøklene kan et team gå fra en metrikk til forespørslene bak den, og fra en forespørsel til logglinjene den etterlot seg.

Observerbarhet og overvåking overlapper, men de er ikke det samme. Overvåking holder øye med forhold som noen på forhånd har bestemt seg for å bry seg om, og sender et varsel når ett av dem inntreffer. Observerbarhet er det teamet har å jobbe med når varselet har gått og årsaken er noe ingen hadde sett for seg.

Bruk begrepet i en beslutning

Bruk «Hva er observerbarhet?» 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

Moderne systemer svikter på måter som er vanskelige å forutsi. En forespørsel kan gå gjennom en gateway, flere tjenester, en kø og en database før den får svar, og en feil i hvilken som helst av dem kan vise seg som treghet et helt annet sted. Innføringen kaller distribuert sporing helt nødvendig for slike systemer, fordi problemene deres ofte er ikke-deterministiske eller for kompliserte til å gjenskape på en utviklers egen maskin. Når en feil ikke lar seg gjenskape, er telemetrien som ble tatt opp mens den skjedde, det eneste beviset som finnes.

Dashbord og varsler svarer på spørsmål noen har forutsett. Det er verdifullt, og det dekker feilene teamet har sett før. De vanskelige driftshendelsene er de nye, der spørsmålet ikke er om feilraten steg, men hvorfor forespørsler fra én region,

gjennom én versjon, til én betalingsleverandør begynte å feile. Hvis svaret krever at man skriver ny kode, ruller den ut og venter på at problemet kommer tilbake, varer nedetiden like lenge som den omveien.

Observerbarhet holder også brukeren i sentrum. Innføringen definerer pålitelighet som et spørsmål om tjenesten gjør det brukerne forventer, og gir et eksempel: En nettbutikk kan være oppe hele tiden og likevel være upålitelig hvis et klikk for å legge svarte sko i handlekurven ikke alltid legger svarte sko i handlekurven. Det ville en måling på servernivå aldri vise. Telemetri fra enkeltforespørsler kan, og det er den tjenestenivåindikatorer bygges av: målinger sett fra brukerens side, som tjenestenivåmål deretter knytter til det virksomheten trenger.

Til slutt endrer det hva en driftshendelse koster. Et team som kan følge en treg forespørsel til akkurat det kallet som holdt den igjen, bruker tiden på å rette i stedet for å gjette, og postmortemen etterpå kan vise til bevis i stedet for erindring.

Slik ser det ut i praksis

Tenk deg en oppdiktet nettbutikk. Siden klokken 14 har en liten andel av utsjekkingene tatt mer enn to sekunder. Ingenting er nede, ingen varsler har gått, og det finnes ikke noe dashbord for akkurat dette. Slik ville et team med sammenkoblede signaler gått fram. Alle navn, ID-er og tall nedenfor er laget for eksempelets skyld.

Første steg er histogrammet over hvor lang tid utsjekkingene tar, målt i sekunder. I tekstformatet til OpenMetrics kan en bucket ha et exemplar etter et `#`: en henvisning til data utenfor metrikken, oftest ID-en til et spor som havnet i den bucketen. To av bucketene ser slik ut:

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

Andre steg følger den spor-ID-en. Sporet viser hvor de 2,3 sekundene ble av, 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 steg henter logglinjene som bærer den samme spor-ID-en. OpenTelemetrys spesifikasjon for logger i andre formater enn prosjektets egen protokoll ber om at ID-ene i JSON skrives som felt på øverste nivå med navnene `trace_id` og `span_id`, med små bokstaver og heksadesimalt kodet:

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

Tre steg, og spørsmålet ingen hadde planlagt for, har fått et svar: Den ene av to betalingsleverandører begynte å svare for sent, og hver utsjekking som brukte den leverandøren, ventet på et nytt forsøk. Den samme attributten, `provider`, kan nå brukes til å gruppere alle nylige spor, slik at mønsteret er bekreftet før noen rører koden.

Ingenting av dette virker hvis tjenestene ikke sier hvem de er. OpenTelemetry-SDK-ene setter som standard ressursattributtet `service.name` til `unknown_service`, og dokumentasjonen anbefaler å sette det eksplisitt, i koden eller med en miljøvariabel. Andre ressursattributter kan legges til på samme måte:

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

På dette nettstedet viser tre porteføljeoppføringer hvordan observerbarhet bygges opp steg for steg. Den ene beskriver et selskaps første observerbarhetsdashbord, som samlet inn systemmålinger som CPU, minne og

disk fra eksterne Linux-servere over SSH; en ny versjon ble senere bygget på Grafana og Prometheus. Arbeidet med å redusere driftsrisikoen med Grafana og Prometheus instrumenterte de viktigste tjenestene og valgte metrikker som betydde noe, framfor tall som bare ser pene ut. En senere oppføring, egen feilovervåking og sporing, la til sporing med OpenTelemetry over OTLP, slik at en treg forespørsel kan følges gjennom frontenden, API-et og databasen i stedet for å bli gjettet på, sammen med en feilpipeline som renser nyttelaster før de lagres.

Hvor det går galt

Å kjøpe det i stedet for å bygge det inn. Et nytt verktøy som ingenting nyttig strømmer inn i, er et tomt rom. Observerbarhet kommer fra instrumentering i koden og bibliotekene, så sett av utviklertid til å legge til span, attributter og loggfelt, ikke bare penger til lisensen.

Tre signaler, tre øyer. Metrikker i ett verktøy, logger i et annet og spor i et tredje, uten en felles nøkkel mellom dem, gjør at teamet ender med å kopiere tidsstempler fra én skjerm til en annen. Skriv spor- og span-ID-ene inn i hver loggpost, legg exemplars på histogrammer over svartid der verktøyene støtter det, og gi alle signaler de samme ressursattributtene.

Alle tjenester heter unknown_service. Det er standardverdien i SDK-en, og en hel flåte av tjenester med samme navn kan ikke skilles fra hverandre. Sett `OTEL_SERVICE_NAME` for hver tjeneste. Ressursen knyttes til når tracer- og metric-providerne opprettes, og kan ikke endres etterpå, så den må være riktig fra oppstart.

Bare kantene er instrumentert. Zero-code-instrumentering er en god start og registrerer hva som skjer i kantene av en applikasjon: i bibliotekene og i miljøet den kjører i. Spørsmålene under en driftshendelse handler som regel om virksomheten: hvilken kundetype, hvilken leverandør, hvilken verdi et feature-flagg hadde. Det krever attributter som legges til i koden. OpenTelemetrys dokumentasjon kaller kodebasert instrumentering et supplement til zero-code, og de to kan kjøre samtidig.

Å instrumentere midt i driftshendelsen. Hvis svaret krevde en ny logglinje og en ny utrulling, var systemet ikke skikkelig instrumentert etter innføringens egen målestokk. Gjør «hva måtte vi legge til?» til et fast spørsmål i hver gjennomgang etter en driftshendelse, og behandle hvert svar som en oppfølgingsoppgave.

Ubegrensede detaljer i feil signal. Bruker-ID-er, fullstendige URL-er og innholdet i forespørsler hører hjemme på span og loggposter, ikke i metrikkenes etiketter, der hver ny verdi skaper en ny tidsserie. OpenMetrics-standarden begrenser etikettnavnene og -verdiene på et exemplar til 128 tegn til sammen, nettopp for å hindre at exemplars brukes til å bære span-data, og den lar en mottaker forkaste exemplars helt. Sjekk at koblingene overlever hele veien fra eksportøren til skjermen før dere stoler på dem.

Telemetri som lekker. Span og logger registrerer hva forespørslene inneholdt, og det kan omfatte adresser, navn og tokener. Bestem hvilke attributter som får registreres, fjern resten før lagring, og beskytt telemetrien like nøye som databasen den beskriver.

Les mer

- Observability primer — OpenTelemetry-prosjektets definisjon, med pålitelighet, SLI-er og SLO-er, span og distribuerte spor.
- Signals — spor, metrikker, logger og baggage, og signalene som fortsatt er under utvikling.
- Instrumentation — kodebasert og zero-code-instrumentering, og hvordan kontekst som sendes videre, binder signalene sammen.
- Resources — ressursattributter, standardverdien `unknown_service` og miljøvariablene som setter dem.
- Trace Context in non-OTLP Log Formats — den stabile spesifikasjonen for hvordan spor- og span-ID-er skrives i syslog, ren tekst og JSON-logger.
- OpenMetrics 1.0 — standarden for eksponeringsformatet, med exemplars og rådene om kardinalitet.

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 observerbarhet?» 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/observability/