Engineer

Observerbarhet

förmågan att förstå ett system utifrån data det skickar ut

Vad det är

Observerbarhet är i vilken grad ett team kan förstå vad ett körande system gör utifrån, enbart med hjälp av de data som systemet självt skickar ut, utan att öppna det och utan att i förväg veta vad som har gått fel. OpenTelemetry-projektets introduktion beskriver det med frågor: ett observerbart system låter dig ställa frågor om det utan att känna till hur det fungerar inuti, även frågor som ingen tänkte på när koden skrevs. Introduktionen sammanfattar frågan som systemet ska kunna besvara som ”Varför händer det här?”

Därmed är observerbarhet en egenskap hos systemet, inte en produkt. Ett verktyg kan lagra och visa data, men data måste först komma från applikationen. OpenTelemetrys dokumentation säger det rakt ut: för att ett system ska vara observerbart måste dess komponenter vara instrumenterade, alltså måste deras kod skicka ut signaler. Dokumentationen sätter också en praktisk ribba. En applikation är ordentligt instrumenterad när utvecklarna inte behöver lägga till mer instrumentering för att felsöka ett problem, eftersom informationen redan finns där.

Dessa data kallas telemetri, och OpenTelemetry delar in dem i signaler:

- **Spår** (traces) visar vägen som en enskild begäran tar genom systemet. Ett spår består av span, där varje span är en arbetsenhet med ett namn, en start- och sluttid och attribut som HTTP-metoden och routen. Det första spannet är roten och täcker begäran från början till slut.
- **Mätvärden** (metrics) är mätningar som görs medan systemet kör och som aggregeras över tid: felfrekvens, antal begäranden per tidsenhet, processoranvändning.
- **Loggar** är tidsstämplade poster om händelser. Många av dem hör inte till någon särskild begäran.

- **Baggage** är kontextuppgifter som skickas med en begäran så att andra signaler kan använda dem.

Signalsidan tar också upp events, en särskild typ av logg, och profiler, som registrerar resursanvändning på kodnivå, som signaler under utveckling eller på förslagsstadiet.

Det som förvandlar tre högar med data till något ett team kan ställa frågor till är sammankoppling. OpenTelemetry skickar kontext vidare mellan tjänster, så att ett span i en tjänst vet vilket spår det hör till, och det kan stämpla id:na för det aktiva spåret och spannet på varje loggpost. Varje signal bär också en resurs: attribut som identifierar det som producerade signalen, till exempel tjänstens namn, värden eller containern. Med de gemensamma nycklarna kan ett team gå från ett mätvärde till begärandena bakom det, och från en begäran till loggraderna den lämnade efter sig.

Observerbarhet och övervakning överlappar, men de är inte samma sak. Övervakning håller utkik efter förhållanden som någon i förväg har bestämt sig för att bry sig om och slår larm när ett av dem inträffar. Observerbarhet är det teamet har att arbeta med när larmet har gått och orsaken är något som ingen hade räknat med.

Använd begreppet i ett beslut

Använd ”Vad är observerbarhet?” 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

Moderna system fallerar på sätt som är svåra att förutse. En begäran kan passera en gateway, flera tjänster, en kö och en databas innan den får svar, och ett fel i vilken som helst av dem kan visa sig som långsamhet någon helt annanstans. Introduktionen kallar distribuerad spårning nödvändig för sådana system, eftersom deras problem ofta är icke-deterministiska eller för komplicerade för att återskapas på en utvecklares egen dator. När ett fel inte går att återskapa är telemetrin som spelades in medan det hände det enda bevis som finns.

Dashboards och larm besvarar frågor som någon har förutsett. Det är värdefullt, och det täcker de fel som teamet har sett förut. De svåra incidenterna är de nya, där frågan inte är om felfrekvensen steg, utan varför begäranden från en region, genom en version, till en betalleverantör började misslyckas. Om svaret kräver att någon skriver ny kod, driftsätter den och väntar på att problemet ska komma tillbaka, varar avbrottet lika länge som den omvägen.

Observerbarhet håller också användaren i centrum. Introduktionen definierar tillförlitlighet som frågan om tjänsten gör det användarna förväntar sig, och ger ett exempel: en webbutik kan vara uppe hela tiden och ändå vara opålitlig om ett klick för att lägga svarta skor i varukorgen inte alltid lägger svarta skor i varukorgen. Det skulle en mätning på servernivå aldrig visa. Telemetri från enskilda begäranden kan det, och det är den som servicenivåindikatorer byggs av: mätningar sedda från användarens sida, som servicenivåmål sedan knyter till det verksamheten behöver.

Till sist förändrar det vad en incident kostar. Ett team som kan följa en långsam begäran till exakt det anrop som höll upp den lägger tiden på att rätta i stället för att gissa, och postmortem efteråt kan hänvisa till bevis i stället för minnesbilder.

Så ser det ut i praktiken

Tänk dig en påhittad webbutik. Sedan klockan 14 tar en liten andel av köpen mer än två sekunder i kassan. Ingenting ligger nere, inget larm har gått och det finns ingen dashboard för just det här. Så här skulle ett team med sammankopplade signaler gå till väga. Alla namn, id:n och siffror nedan är påhittade för exemplets skull.

Första steget är histogrammet över hur lång tid kassan tar, mätt i sekunder. I textformatet för OpenMetrics kan en bucket bära ett exemplar efter ett `#`: en hänvisning till data utanför mätvärdet, oftast id:t för ett spår som hamnade i den bucketen. Två av bucketarna ser ut så här:

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

Andra steget följer det spår-id:t. Spåret visar vart de 2,3 sekunderna tog vägen, span för 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 steget hämtar loggraderna som bär samma spår-id. OpenTelemetrys specifikation för loggar i andra format än projektets eget ber att id:na i JSON skrivs som fält på översta nivån med namnen `trace_id` och `span_id`, med små bokstäver och hexadecimalt kodade:

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

Tre steg, och frågan som ingen hade planerat för har fått ett svar: den ena av två betalleverantörer började svara för sent, och varje köp som skickades dit väntade på ett nytt försök. Samma attribut, `provider`, kan nu användas för att gruppera alla färska spår, så att mönstret är bekräftat innan någon rör koden.

Inget av detta fungerar om tjänsterna inte talar om vilka de är. OpenTelemetrys SDK:er sätter som standard resursattributet `service.name` till `unknown_service`, och dokumentationen rekommenderar att det sätts uttryckligen, i koden eller med en miljövariabel. Andra resursattribut kan läggas till på samma sätt:

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

På den här webbplatsen visar tre poster i portföljen hur observerbarhet byggs upp steg för steg. Den första beskriver ett företags första observerbarhetsdashboard, som samlade in systemmått som CPU, minne och disk från Linux-servrar på distans via SSH; senare byggdes en andra version med Grafana och Prometheus. Arbetet med att minska driftrisken med Grafana och

Prometheus instrumenterade de viktigaste tjänsterna och valde mätvärden som betydde något framför siffror för syns skull. En senare post, egen felövervakning och spårning, lade till spårning med OpenTelemetry över OTLP, så att ett långsamt anrop kan följas genom frontenden, API:t och databasen i stället för att gissas fram, tillsammans med en felpipeline som sanerar nyttolaster före lagring.

Var det går fel

Att köpa det i stället för att bygga in det. Ett nytt verktyg som inget användbart flödar in i är ett tomt rum. Observerbarhet kommer från instrumentering i koden och biblioteken, så avsätt utvecklartid för att lägga till span, attribut och loggfält, inte bara pengar till licensen.

Tre signaler, tre öar. Mätvärden i ett verktyg, loggar i ett annat och spår i ett tredje, utan någon gemensam nyckel mellan dem, gör att teamet sitter och kopierar tidsstämplar från en skärm till en annan. Skriv in spårets och spannets id:n i varje loggpost, lägg exemplars på histogram över svarstid där verktygen stöder det, och ge alla signaler samma resursattribut.

Alla tjänster heter unknown_service. Det är standardvärdet i SDK:n, och en hel flotta av tjänster med samma namn går inte att skilja åt. Sätt `OTEL_SERVICE_NAME` för varje tjänst. Resursen knyts till när tracer- och metric-providerna skapas och kan inte ändras efteråt, så den måste vara rätt från start.

Bara kanterna är instrumenterade. Zero-code-instrumentering är en bra start och registrerar vad som händer i kanterna av en applikation: i dess bibliotek och i miljön den körs i. Frågorna under en incident handlar oftast om verksamheten: vilken kundnivå, vilken leverantör, vilket värde en funktionsflagga hade. Det kräver attribut som läggs till i koden. OpenTelemetrys dokumentation kallar kodbaserad instrumentering ett komplement till zero-code, och de två kan köras samtidigt.

Att instrumentera mitt under incidenten. Om svaret krävde en ny loggrad och en ny driftsättning var systemet inte ordentligt instrumenterat enligt introduktionens eget mått. Gör ”vad var vi tvungna att lägga till?” till en stående fråga i varje genomgång efter en incident, och behandla varje svar som en uppföljningsuppgift.

Obegränsade detaljer i fel signal. Användar-id:n, fullständiga URL:er och innehållet i begäranden hör hemma på span och loggposter, inte i mätvärdenas etiketter, där varje nytt värde skapar en ny tidsserie. OpenMetrics-standarden begränsar etikettnamnen och etikettvärdena på ett exemplar till 128 tecken sammanlagt, just för att hindra att exemplars används för att bära span-data, och den låter en mottagare kasta exemplars helt. Kontrollera att kopplingarna överlever hela vägen från exportören till skärmen innan ni förlitar er på dem.

Telemetri som läcker. Span och loggar registrerar vad begärandena innehöll, och det kan omfatta adresser, namn och token. Bestäm vilka attribut som får registreras, rensa bort resten före lagring och skydda telemetrin lika noga som databasen den beskriver.

Läs mer

- Observability primer — OpenTelemetry-projektets definition, med tillförlitlighet, SLI:er och SLO:er, span och distribuerade spår.
- Signals — spår, mätvärden, loggar och baggage, och de signaler som fortfarande är under utveckling.
- Instrumentation — kodbaserad och zero-code-instrumentering, och hur kontext som skickas vidare binder ihop signalerna.
- Resources — resursattribut, standardvärdet `unknown_service` och miljövariablerna som sätter dem.
- Trace Context in non-OTLP Log Formats — den stabila specifikationen för hur spår- och span-id:n skrivs i syslog, ren text och JSON-loggar.
- OpenMetrics 1.0 — standarden för exponeringsformatet, med exemplars och råden om kardinalitet.

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 observerbarhet?” 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/observability/