Engineer

Servicenivåmål

◎ ett mätbart tillförlitlighetsmål som en tjänst ska uppfylla

Vad det är

Ett servicenivåmål är ett krav på hur väl en tjänst ska fungera, formulerat så exakt att vem som helst kan kontrollera om det uppfylldes: en andel förfrågningar som ska lyckas, eller besvaras tillräckligt snabbt, under en angiven period. På engelska heter det service level objective, och nästan alla förkortar det till SLO, även i svenska team. Googles bok om Site Reliability Engineering definierar det som ett målvärde eller ett värdeintervall för en servicenivå, mätt med en indikator.

Den indikatorn är servicenivåindikatorn, eller SLI:n: ett noga definierat mått på en aspekt av den tjänst användarna faktiskt får. Latens för förfrågningar är det vanliga förstahandsvalet, tätt följt av felfrekvens och genomströmning. SRE Workbook rekommenderar att varje SLI skrivs som en kvot, goda händelser delat med alla händelser, så att den går från 0 %, där ingenting fungerar, till 100 %, där ingenting är trasigt. Målet är då ett streck på den skalan: indikatorn ska hålla sig på rätt sida om en gräns, eller mellan en undre och en övre.

Ett fullständigt mål anger allt som behövs för att kontrollera det. Bokens exempel lyder: 99 % av Get RPC-anropen, i genomsnitt över en minut, slutförs på under 100 ms, mätt över alla backend-servrar. En indikator kan ha flera mål samtidigt, till exempel 90 % av anropen under 1 ms, 99 % under 10 ms och 99,9 % under 100 ms, eftersom en enda percentil sällan beskriver en hel fördelning. Workbook skiljer också mellan specifikationen, det utfall som spelar roll för användarna, och implementationen, platsen där det mäts: en serverlogg, en sond som kör en webbläsare eller kod på själva sidan. Varje plats ser olika fel, och valet hör hemma i definitionen.

Ett SLO är inte ett SLA. Ett servicenivåavtal (service level agreement) är ett kontrakt med användarna, uttalat eller underförstått, som knyter konsekvenser till om målen i det nås, oftast en rabatt eller ett vite. Bokens test är att fråga vad

som händer när målet missas: följer ingen uttalad konsekvens är det ett SLO. En fotnot tillägger att den som talar om ett brutet SLA nästan alltid menar ett missat SLO, eftersom ett verkligt brott mot ett SLA kan sluta i domstol som kontraktsbrott. Google Search, påpekar boken, har inget SLA gentemot allmänheten, men ett avbrott där kostar ändå anseende och annonsintäkter.

Använd begreppet i ett beslut

Använd ”Vad är ett servicenivåmål?” 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

Utan ett uttalat mål hittar användarna på ett eget. SRE-boken beskriver båda sätten det går fel på: somliga litar mer på tjänsten än de borde, och tänkbara användare tror att den är ostadigare än den är och håller sig borta. Exemplet är Chubby, Googles låstjänst för distribuerade system. Globala avbrott var så sällsynta att andra tjänster i det tysta började räkna med att den aldrig föll, och de tjänsterna gick sönder de få gånger den gjorde det. Lösningen var att låta den globala instansen nå sitt SLO utan att överträffa det med mycket: varje kvartal där verkliga fel inte hade pressat tillgängligheten under målet tog SRE-teamet ned systemet med flit.

Målet avgör också diskussionen om hur tillförlitligt som är tillförlitligt nog. SRE Workbook kallar 100 % tillförlitlighet fel mål, och skälen är praktiska. Komponenter fallerar samtidigt, hur de än är byggda. Kedjan av nät och enheter mellan tjänsten och kunderna fallerar också, så kunderna skulle aldrig se de 100 procenten. Varje extra nia kostar mer än den förra medan värdet för användarna närmar sig noll. Och eftersom förändringar är den vanligaste orsaken till avbrott kan en tjänst som aldrig får fallera heller aldrig uppdateras. När målet ligger under 100 % blir glappet mellan målet och det perfekta ett utrymme att lägga på releaser, experiment och planerat underhåll. Det utrymmet kallas felbudgeten (error budget), och den är helt enkelt 100 % minus målet. Någon måste äga avvägningen mellan nya funktioner och stabil drift; Workbook pekar ut CTO:n i en liten organisation och produktägaren eller produktchefen i en större.

Till sist gör målet övervakning till beslut. Boken beskriver en slinga: mäta indikatorerna, jämföra dem med målen, avgöra om något behöver göras och göra det. Om latensen stiger och kommer att korsa strecket inom några timmar kontrollerar teamet om servrarna är CPU-bundna och sätter in fler. Utan målet, påpekar boken, kan ingen säga om man ska agera, eller när.

Så ser det ut i praktiken

Ta ett checkout-API och ett mål skrivet så som Workbook föreslår, med indikatorn skild från platsen där den mäts, fönstret angivet och motiveringen bredvid:

```
Tjänst:       checkout-API
SLI:          andel POST /checkout-förfrågningar besvarade utan
              5xx-status, räknade i serverns förfrågningslogg
Mål:          99,9 % över ett rullande fönster på 28 dagar
Motivering:   observerat, avrundat nedåt från fyra veckors data
Ägare:        produktchef tillsammans med jourhavande utvecklare
Genomgång:    månadsvis tills målet har satt sig, sedan kvartalsvis
```

Fönstret väger lika tungt som procentsatsen. Workbook föredrar ett rullande fönster framför en kalendermånad, eftersom användarna inte glömmer ett avbrott den sista dagen i månaden bara för att en ny börjar, och ett helt antal veckor, så att varje fönster rymmer lika många helger; fyra veckor är dess val för allmänt bruk. Det fönstret är 28 × 24 × 60 = 40 320 minuter, och 0,1 % av det är drygt 40 minuter av fullständigt avbrott. För en månad på 30 dagar ger samma 99,9 % utrymme för 43,2 minuter, siffran i bokens tillgänglighetstabell.

Tid är ändå det grövre sättet att räkna. SRE-boken förklarar att Google mäter tillgänglighet som andelen lyckade förfrågningar snarare än som drifttid (uptime), eftersom en tjänst som är utspridd över hela världen sällan är helt uppe eller helt nere. Räknat i förfrågningar lyder Workbooks exempel: med 99,9 % lämnar 3 miljoner förfrågningar under fyra veckor utrymme för 3 000 fel, och ett avbrott som orsakar 1 500 av dem förbrukar halva budgeten. Att räkna förfrågningar fångar också de partiella fel som drifttiden missar. Vid jämn trafik kostar 20 minuter då hälften av alla förfrågningar misslyckas lika mycket som 10 minuter av fullständigt avbrott, och det är ungefär en fjärdedel av budgeten för 28 dagar.

I Prometheus, med en räknare för förfrågningar märkt med status, är indikatorn över fönstret en enda kvot, byggd efter Workbooks eget exempel:

```
sum(rate(http_requests_total{job="checkout", status!~"5.."}[28d]))
  /
sum(rate(http_requests_total{job="checkout"}[28d]))
```

På den här webbplatsen står sidan om servicenivå på avtalets sida av strecket. För ett system som företaget driver åt en kund skrivs svarstiden vid en incident, vilka timmar som täcks, den tillgänglighet systemet ska hålla och när underhåll får ske in i avtalet för just det systemet. Villkoren bestäms system för system, eftersom en nattlig rapport och en betaltjänst behöver olika svar. Den egna felövervakningen och spårningen som grundaren byggde i sitt pågående arbete hos NextMariner presenteras i en rad operatörsvyer, däribland en för servicenivåmål. Och anteckningen om att sälja den sista biljetten exakt en gång gör en lagerinvariant till ett sådant mål: kör kontrollen schemalagt, visa den i ett diagram och larma vid varje avvikelse, så att noll översäljningar blir ett mål med ett larm bakom sig, inte en förhoppning.

Var det går fel

Målet är kopierat från förra månadens dashboard. Att ta det systemet klarar i dag känns tryggt, men SRE-boken varnar för att ett mål som väljs utan eftertanke kan låsa ett team vid heroiska insatser för att fortsätta nå det, på ett system som inte kan bli bättre utan att byggas om. Utgå från vad användarna behöver, och skriv motiveringen bredvid siffran: Workbook ber om resonemanget, och om siffrorna kommer från experiment eller observation, även när de är rent ad hoc. Gå igenom dokumentet varje månad i början och varje kvartal när det har satt sig.

Målet är 100 %, eller en rad nior som ingen har räknat på. Ett system som alltid är tillgängligt, eller skalar utan gräns, är dyrt att bygga och driva, och bättre än användarna kommer att märka. Börja med ett löst mål och skärp det i takt med att ni lär känna systemet; boken föredrar det framför ett strikt mål som senare måste mjukas upp för att det visade sig ouppnåeligt.

Ett medelvärde står i stället för det användarna märker. I bokens exempel tar en typisk förfrågan ungefär 50 ms medan 5 % av förfrågningarna tar tjugo gånger så lång tid, och övervakning av enbart medellatensen skulle inte visa någon förändring under dagen. Skriv latensmål som percentiler, eller som en andel förfrågningar under en gräns, till exempel 300 ms, aldrig som ett medelvärde.

Det finns fler mål än någon kan försvara. Ett dussin SLO:er per tjänst späder ut dem alla. Bokens test är rakt på sak: om ett mål aldrig kan användas för att vinna en diskussion om prioriteringar är det troligen inte värt att ha. Behåll de få som täcker det användarna faktiskt gör.

Ett missat mål ändrar ingenting. Workbook kallar detta ett SLO utan tänder: efterlevnaden blir bara ett KPI till i en rapport i stället för ett skäl att besluta något. Kom överens i förväg om vad som händer när budgeten är förbrukad, och låt produktchefen, utvecklarna och SRE-teamet skriva under. Workbooks exempel på en felbudgetpolicy stoppar alla ändringar utom buggfixar med högsta prioritet och säkerhetsfixar tills tjänsten åter håller sitt mål, och säger rent ut att policyn inte är ett straff.

SLO och SLA är samma siffra. När det interna målet är lika med kontraktet är avtalsbrottet det första teamet hör om problemet. Boken rekommenderar ett striktare internt mål än det som utlovas utåt, som en buffert mot kroniska problem. Det motsatta misstaget skadar också: en tjänst som går långt bättre än sitt mål lär användarna att räkna med det, så boken föreslår planerade driftstopp då och då, strypning (throttling) eller ett system som inte är snabbare under låg last.

Fönstret är längre än datan. Ett SLO på 28 dagar som räknas ut på en Prometheus-server som behåller 15 dagars data, standardvärdet när ingen lagringstid är satt, mäter i det tysta 15 dagar och rapporterar det som fyra veckor. Sätt lagringstiden längre än fönstret innan du litar på siffran:

```
prometheus --storage.tsdb.retention.time=35d
```

Läs mer

- Service Level Objectives — kapitel 4 i Googles bok om Site Reliability Engineering: indikatorer, mål och avtal, och hur man väljer ett mål.
- Implementing SLOs — kapitlet i SRE Workbook om SLI:er som kvoter, felbudgetar, fönster och hur målet skrivs ned.
- Example Error Budget Policy — en fullständig policy för en påhittad speltjänst: när releaser stoppas och vem som avgör en oenighet.
- Embracing Risk — varför Google mäter tillgänglighet med lyckade förfrågningar snarare än drifttid, och vad varje steg i tillförlitlighet kostar.

- Availability Table — hur mycket driftstopp varje tillgänglighetsnivå tillåter per år, kvartal, månad, vecka, dag och timme.
- Storage — hur Prometheus lagrar data på disk och vilka flaggor för lagringstid som avgör hur långt bakåt en SLO-fråga kan se.

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 ett servicenivåmål?” 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/service-level-objective/