Engineer

Idempotent

sikkert at køre to gange, fordi anden gang intet ændrer

Hvad det er

En operation er idempotent, når den har samme virkning udført flere gange som udført én gang, og egenskaben kaldes idempotens, på engelsk idempotence eller idempotency. Ordet kommer fra matematikken, hvor en funktion er idempotent, når den lader sit eget resultat uændret, så f(f(x)) er lig med f(x): at runde et tal ned, der allerede er rundet ned, giver det samme tal.

I software er den nyttige prøve en kørsel mere. At sætte en værdi er idempotent, fordi anden kørsel skriver den værdi, der allerede står der. At lægge noget til en værdi er det ikke, fordi anden kørsel lægger til igen:

```
-- idempotent: en kørsel mere sætter den samme værdi
UPDATE accounts SET plan = 'pro' WHERE id = 42;

-- ikke idempotent: en kørsel mere lægger ti til igen
UPDATE accounts SET credits = credits + 10 WHERE id = 42;
```

HTTP giver ordet dets mest præcise definition. RFC 9110 kalder en forespørgselsmetode idempotent, hvis den tilsigtede virkning på serveren af flere ens forespørgsler er den samme som af én enkelt, og nævner PUT, DELETE og de sikre metoder GET, HEAD, OPTIONS og TRACE. To detaljer i definitionen er vigtige. Egenskaben gælder den virkning, klienten bad om, så en server må gerne logge hver forespørgsel for sig. Og svaret må gerne være forskelligt: en anden DELETE af den samme ressource kan svare 404, hvor den første svarede 200, og metoden er stadig idempotent, fordi ressourcen er væk i begge tilfælde. PATCH står uden for listen. RFC 5789 siger, at den hverken er sikker eller idempotent, men en klient kan sende en bestemt PATCH på en idempotent måde ved at gøre den betinget med et stærkt ETag i en If-Match-header, så en gentaget kopi bliver afvist, når den første har ændret ressourcen.

Sikker er det strengeste af de to løfter. En sikker metode i RFC 9110's forstand er i det væsentlige kun læsende, så hver sikker metode er idempotent, mens DELETE er idempotent uden at være sikker.

Brug begrebet i en beslutning

Brug »Hvad betyder idempotent?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Tal, der lander, hvor der er brug for dem, efter en fast plan, uden at nogen kopierer et regneark ved månedens slutning.

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. Én infrastrukturplan til 14 afdelinger — 14 forskningsafdelinger på én infrastrukturplan

Én infrastrukturplan til 14 afdelinger Kontakt os →

Hvorfor det betyder noget

Netværk taber svar. Når en klient sender en forespørgsel, og forbindelsen ryger, før svaret når frem, kan den ikke vide, om serveren aldrig så forespørgslen, så den og stoppede, eller udførte den og mistede svaret. At gentage forespørgslen er den oplagte udvej, og idempotens er det, der gør gentagelsen ufarlig. RFC 9110 bygger sine regler for nye forsøg på netop det. En idempotent forespørgsel kan gentages automatisk efter en kommunikationsfejl. En klient bør kun gentage en ikke-idempotent forespørgsel automatisk, hvis den ved, at forespørgslen i praksis er idempotent, eller kan se, at originalen aldrig blev udført, og en proxy må aldrig gentage en sådan af sig selv.

Køer rejser samme spørgsmål i stor skala. Mange beskedsystemer lover levering mindst én gang: en forbruger, der går ned efter at have gjort arbejdet og før den har kvitteret for beskeden, får den samme besked igen. Levering præcis én gang over et netværk er et langt sværere løfte, så det praktiske svar er en forbruger, hvor en levering mere ikke gør nogen forskel. Konfigurationsstyring afhænger også af egenskaben. Et værktøj, der bringer servere til en erklæret tilstand, kører igen og igen, efter hver ændring og ofte efter en tidsplan, og hver kørsel skal efterlade en korrekt server præcis, som den fandt den.

Den egentlige beslutning er, hvordan man får idempotens, hvor en operation mangler den af natur, og der er tre almindelige svar. Det første er design: udtryk ændringen som en måltilstand frem for et trin, altså »sørg for, at denne linje står i filen« i stedet for »tilføj denne linje«. Det andet er en nøgle: klienten giver forespørgslen et unikt id, serveren gemmer det under en unik begrænsning, og en gentagelse finder posten og får det oprindelige resultat tilbage i stedet for en

handling mere. Det tredje er at tjekke, før man handler, som de fleste Ansible-moduler gør: de sammenligner den nuværende tilstand med den ønskede og afslutter uden at gøre noget, når de to allerede stemmer overens. Hvert svar har sin pris. Design efter måltilstand kræver en klar model af sluttilstanden, nøgler kræver lagerplads og en regel for, hvor længe de gemmes, og at tjekke først kræver en pålidelig måde at læse den nuværende tilstand på.

Sådan ser det ud i praksis

Virksomhedens egne servere konfigureres med Ansible, og i det projekt er idempotens en kontrakt: en konvergeret vært rapporterer nul ændringer. Kørt uden argumenter bringer provisioneringen hver vært i den tilstand, der er erklæret for den, og står værten allerede sådan, skal et play melde nul ændringer; et play, der ikke kan det, regnes for ufærdigt. Dermed bliver idempotens noget, hver kørsel måler, frem for noget, man håber på.

Samme idé anvendt på data ligger bag en test, der kører hele pipelinen i nitten trin to gange. Siden navngiver den slags fejl, testen er til for: en generator, der virker første gang og korrumperer anden gang, gennem trin, der indsætter uden at tjekke, eller afhænger af rækkefølgen i et tidligere trins output. Testen bygger databasen fra en tom fil, tager et øjebliksbillede, kører hele pipelinen igen over resultatet og kræver, at det andet gennemløb lader alt være, ned til antal rækker, indhold og identifikatorer. Testen giver dokumentation for gentagne kørsler med de input og den databasetilstand, den afprøver.

Til salg af et begrænset lager, hvor hver enhed kun må sælges én gang, kombinerer en note om flash-salg en nøgle med betingede skrivninger. Hvert checkout bærer en idempotensnøgle, og reservationstabellen bærer `UNIQUE (tenant_id, idempotency_key)`, så et gentaget klik får den reservation, det allerede har, og ikke en enhed mere. En frigivelse, der kører to gange, fra en timeout og en annullér-knap for eksempel, giver enheden tilbage én gang, fordi den kun giver lageret tilbage, hvis dens egen opdatering ændrede præcis én række. Længere fremme er hver forbruger af beskeder idempotent med ordren som nøgle: beskeder kommer mindst én gang, og idempotens gør det til i praksis én gang.

Hvor det går galt

Et nyt forsøg trækker beløbet to gange. En POST, der opretter en betaling, en ordre eller en e-mail, er ikke idempotent som udgangspunkt, og en klient eller proxy, der gentager den efter en timeout, opretter en til. Giv sådanne forespørgsler en idempotensnøgle, som serveren gemmer under en unik begrænsning, og returnér det gemte resultat, når den samme nøgle kommer igen.

Et svar uden rækker bliver læst som en fejl. I PostgreSQL springer `INSERT ... ON CONFLICT DO NOTHING` en række over, der ville bryde en unik begrænsning, og `RETURNING` giver kun de rækker tilbage, der faktisk blev indsat eller opdateret. En gentaget indsættelse af den samme række returnerer derfor ingen rækker, hvilket er det korrekte udfald af en kørsel mere. Kode, der behandler det svar som en fejl, gør et ufarligt nyt forsøg til en fejlside; den bør slå den række op, der allerede ligger der.

En kommandoopgave ændrer værten ved hver kørsel. Ansibles dokumentation siger, at de fleste moduler tjekker, om den ønskede tilstand allerede er nået, og også, at ikke alle playbooks og ikke alle moduler opfører sig sådan. En opgave, der kører en kommando, er den typiske undtagelse, fordi kommandoen er vilkårlig, og Ansible har ingen måde at vide, hvad den gør. Kommandomodulets `creates` springer trinnet over, når en given fil allerede findes, `removes` kører det kun, mens filen stadig er der, og `changed_when` bestemmer, hvilket udfald der tæller som en ændring. En kommandoopgave uden nogen af dem kører igen ved hver konvergering, uanset hvilken tilstand værten er i.

En tæller tæller det nye forsøg med. Ved levering mindst én gang gør en forbruger, der tæller en tæller op eller sender en notifikation for hver besked, det to gange for en besked, der leveres to gange. Gem beskedens forretningsnøgle sammen med virkningen i samme transaktion, og spring enhver nøgle over, der allerede er gemt.

Anden kørsel ødelægger det, den første byggede. Et trin kan være korrekt på en tom database og forkert på en fuld, fordi det indsætter en dublet, nummererer identifikatorer om eller bygger på en rækkefølge, den første kørsel efterlod. En test, der starter fra ingenting og kører én gang, består alligevel. Kør hele jobbet to gange, og kræv, at anden kørsel lader alt være, som pipelinetesten ovenfor gør.

Læs mere

- RFC 9110, section 9.2.2: Idempotent Methods — HTTP's definition, de idempotente metoder og reglerne for automatiske nye forsøg.
- RFC 5789: PATCH Method for HTTP — hvorfor PATCH hverken er sikker eller idempotent, og hvordan en betinget forespørgsel gør den sikker at gentage.
- Ansible: Introduction to playbooks — ønsket tilstand, idempotens og check mode med Ansibles egne ord.
- Ansible: command module — `creates` og `removes`, de indstillinger, der gør en kommando sikker at køre igen.
- PostgreSQL: INSERT — `ON CONFLICT`, og hvad `RETURNING` giver tilbage, når en række allerede findes.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad betyder idempotent?« 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/idempotent/