Engineer

Migrering

en planlagt overgang til en ny versjon eller plattform

Hva det er

En migrering er en planlagt flytting av noe som allerede er i bruk, over til en ny versjon eller en ny plattform. Den skjer i trinn, og til slutt blir det gamle faset ut. Det som flyttes, kan være en hel arbeidslast, en applikasjon, en mengde data eller strukturen dataene ligger i. Det som skiller en migrering fra en gjenoppbygging, er at systemet er i bruk hele veien: brukere, poster og integrasjoner blir med over, og den gamle versjonen går videre til den nye har tatt over.

Ordet brukes om arbeid i to størrelser, og ofte sier ingen hvilken av dem det gjelder.

I stor skala er det en flytting til en ny plattform eller en ny hovedversjon. Microsofts Cloud Adoption Framework beskriver det å flytte arbeidslaster til Azure, fra egne servere eller fra en annen sky, som fem trinn: planlegg, klargjør arbeidslastene, gjennomfør, optimaliser og fas ut kilden. Esris kunder flytter kart og data fra ArcMap til ArcGIS Pro. En Next.js-applikasjon går fra Pages Router til App Router, der `getServerSideProps` viker for datahenting inne i Server Components. En PostgreSQL-klynge går over til en ny hovedversjon, som ofte endrer oppbyggingen av systemtabellene, så dataene må føres over med `pg_upgrade` eller med en dump og en gjenoppretting.

I liten skala er en migrering én versjonert endring av strukturen i en database, lagret som en fil i repoet og kjørt av et verktøy i fast rekkefølge. Django kaller migreringene sine et versjonskontrollsystem for databaseskjemaet: hver fil beskriver en endring i modellene, blir committet sammen med koden og kjøres i samme rekkefølge på utviklerens maskin, på staging og i produksjon. GitLab leverer batchvise bakgrunnsmigreringer som skriver om store tabeller litt om gangen etter en oppgradering, for eksempel ved å gjøre heltallskolonner om til

`bigint` før verdiene renner over. En datamigrering hører også hjemme her: den endrer radene i stedet for kolonnene, for eksempel ved å fylle et nytt felt ut fra to gamle.

Dette er én idé i to størrelser. Begge har en kjent starttilstand, en ordnet rekke trinn, et mål og en vei tilbake hvis et trinn går galt. En plattformflytting rommer som regel skjemamigreringer, og en lang rekke skjemamigreringer er med årene en plattform som byttes ut bit for bit under dem som bruker den.

Bruk begrepet i en beslutning

Bruk «Hva er en migrering?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Data hentet ut av noe som er i ferd med å gå ut på dato, og flyttet over til noe som fortsatt støttes, mens virksomheten går som vanlig.

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. Gjorde GIS-kart 10 ganger raskere på PostgreSQL — 10x raskere kart etter overgangen fra MSSQL

Gjorde GIS-kart 10 ganger raskere på PostgreSQL Ta kontakt →

Hvorfor det er viktig

Programvare lever lenger enn det den er bygget på. Skyavtaler blir reforhandlet, rammeverk bytter modell, og leverandører faser ut produkter. Et system som skal leve i ti år, blir migrert flere ganger i løpet av de ti årene, så hvor godt et team migrerer, ender med å bety mer enn hvor godt det valgte den første plattformen.

- **Mellomtilstanden er der regningen havner.** Så lenge en del av systemet kjører på den gamle plattformen og en del på den nye, må begge driftes, overvåkes og holdes i takt. Cloud Adoption Framework råder til å holde den perioden kort og til å flytte komponenter som avhenger direkte av hverandre, i samme bølge.
- **En database kan ikke bare produksjonssettes på nytt.** Kode kan bygges på nytt og byttes ut; radene i produksjon kan ikke det. Skjemamigreringer gjør en strukturendring i en database i drift om til en fil som kan gjennomgås, testes mot en kopi, kjøres i rekkefølge og for de fleste endringer rulles tilbake. Alternativet er et produksjonsskjema som ser ut slik den siste personen med en konsoll etterlot det.
- **En planlagt flytting kan angres.** Cloud Adoption Framework ber om at kriterier og prosedyrer for tilbakerulling ligger fast før en migrering begynner, at tilbakerullingen er testet, og at det finnes go/no-go-kontrollpunkter underveis.

Avveiningene ligger i selve omleggingen. Cloud Adoption Framework skiller mellom migrering med nedetid og migrering med nesten null nedetid, der data replikeres løpende og byttet skjer til slutt. For en ny hovedversjon av PostgreSQL korter `pg_upgrade` ned omleggingen: verktøyet hopper over dumpen og gjenopprettingen en slik oppgradering vanligvis krever, ved å gjenbruke de eksisterende datafilene, og med `--link`, som ikke kopierer noen filer, går det enda mye raskere.

Slik ser det ut i praksis

Tre prosjekter på dette nettstedet viser begrepet i tre størrelser, fra en hel plattform ned til rekkefølgen på filene i ett repo.

Én migrering var en flytting bort fra en skyleverandør. En kundes AWS-oppsett hadde vært overdimensjonert, og arbeidet begynte med å gjennomgå hva som faktisk ble brukt: unødvendige tjenester ble fjernet, og den gjenværende kapasiteten ble tilpasset. Deretter ble arbeidslastene flyttet bort fra AWS. Dokumentasjonen av oppdraget underbygger disse infrastrukturendringene. Å kutte ubrukte tjenester før flyttingen betydde at det var mindre å flytte; det fastslår ikke et kostnadsforhold for sammenlignbare oppsett, et produktivitetsresultat eller en migrering uten driftsforstyrrelser.

Det andre var et bytte av databasemotor med et nytt design i den andre enden. Databasen til en geografisk kartapplikasjon gikk fra MSSQL til PostgreSQL med PostGIS-utvidelsen, og målet var ikke en kopi av den gamle strukturen. Det nye skjemaet ble designet for romlige data, med GiST- og SP-GiST-indekser og strenge fremmednøkler og CHECK-regler som håndhever valideringen av romlige koordinater. Over 50 millioner romlige poster ble flyttet inn via ETL-pipelines som transformerte dem til EPSG:4326. En flytting til et nytt design er sjansen til å skrive datareglene inn i selve skjemaet. Siden melder om 10 ganger bedre ytelse for romlige spørringer og at CPU-belastningen i backend falt med 65 %.

Det tredje er den lille skalaen, holdt ved like i årevis. Skjemaet til én plattform er summen av 1 022 migreringer, nummerert fra 036 til 1102, og siden kaller disiplinen rundt rekkefølgen uglamorøs og ikke til forhandling. Historikken i repoet er en beskrivelse av produksjon, og den stemmer bare så lenge enhver database, fra en nyopprettet utviklerkopi til en gjenoppretting fra sikkerhetskopi, kan bygges opp igjen ved å spille historikken av fra første til siste fil.

Hvor det går galt

Omleggingen får planleggingen, mens problemene gjerne samler seg i ukene rundt den, i fire gjenkjennelige former.

To systemer går side om side lenger enn planlagt. Flyttingen stopper opp halvveis, bitene som gjenstår, er de vanskelige, og teamet drifter begge plattformene i månedsvis. Brukerne kjenner skjøten: Next.js-veiledningen påpeker at en overgang mellom en side under Pages Router og en side under App Router er en hard navigasjon, og at `next/link` ikke prefetcher på tvers av de to. Hold bølgene små, og gjør hvert neste trinn billig. Én frontend her flytter sidene sine over på en felles fabrikk som henter hver side med én rundtur, slik at hver side som flyttes, arver oppførselen.

En omlegging uten vei tilbake. Noen trinn endrer kilden i tillegg til målet. Med `pg_upgrade --link` har den nye klyngen, straks den er startet, skrevet til filer som den gamle deler, og den gamle klyngen må da gjenopprettes fra sikkerhetskopi. Esri advarer om at et datasett som er oppgradert for ArcGIS Pro, kanskje ikke lenger kan brukes i ArcMap, og ber brukerne ta en kopi først. I Django gir en `RunSQL` -operasjon uten omvendt SQL en `IrreversibleError` i det øyeblikket noen migrerer bakover. Fastsett kriteriene for tilbakerulling før første trinn, øv på veien tilbake, og gjør den til en kontroll: ett repo her har en kontrollmotor der validatorene blant annet sjekker at en migrering kan rulles tilbake.

En migrering som allerede har kjørt, blir redigert. Migreringsverktøy registrerer hvilke filer som er kjørt, og hopper over dem fra da av. Endrer du en fil etter at den har nådd produksjon, beholder produksjon den gamle utgaven, mens hver ny database får den nye: to skjemaer under én historikk. La kjørte migreringer være, og rett dem med en ny; et bygg som spiller av hele historikken fra en tom database og sammenligner resultatet med en migrert kopi, fanger avviket.

Den oppgraderte databasen er treg første dag. `pg_upgrade` fører over det meste av planleggerens statistikk, men ikke alt, og planleggeren gjetter på resten til den er bygget opp igjen. Kontrollmodusen `--check` endrer ingenting og kan kjøre mens den gamle serveren fortsatt betjener brukere, så kjør den først, og bygg opp den manglende statistikken rett etter oppgraderingen før du samler inn resten:

```
pg_upgrade --check -b "$OLD/bin" -B "$NEW/bin" -d "$OLD/data" -D "$NEW/data"
pg_upgrade -b "$OLD/bin" -B "$NEW/bin" -d "$OLD/data" -D "$NEW/data"
vacuumdb --all --analyze-in-stages --missing-stats-only
vacuumdb --all --analyze-only
```

Les mer

- Plan your migration — Microsofts metode for å flytte arbeidslaster til Azure: bølger, valg av omlegging og planlegging av tilbakerulling.
- Migrations — Djangos veiledning i versjonerte skjemaendringer, tilbakerulling av dem og datamigreringer.
- pg_upgrade — oppgraderingsverktøyet til PostgreSQL, med kontrollmodus, link-modus, tilbakeføring og statistikktrinnene etterpå.
- How to migrate from Pages to the App Router — slik flytter du en Next.js-applikasjon til den nye routeren én side om gangen.
- Migrate existing work — hva som følger med fra ArcMap til ArcGIS Pro, hva som må konverteres, og hva du bør kopiere først.
- Check migrations before upgrade — hvordan GitLabs batchvise bakgrunnsmigreringer skriver om store tabeller etter en oppgradering, og hvordan du ser at de er ferdige.

Ordliste

Velg ut fra deres faktiske rammer

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en migrering?» 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/migration/