Engineer

Migrering

🚚 en planerad övergång till en ny version eller plattform

Vad det är

En migrering är en planerad flytt av något som redan används, till en ny version eller en ny plattform. Den sker i steg, och till sist avvecklas det gamla. Det som flyttas kan vara en hel arbetslast, en applikation, en mängd data eller den struktur som datan ligger i. Det som skiljer en migrering från en ombyggnad är att systemet används hela vägen: användare, poster och integrationer följer med, och den gamla versionen fortsätter arbeta tills den nya har tagit över.

Ordet används om arbete i två storlekar, och ofta säger ingen vilken av dem som avses.

I stor skala är det en flytt till en ny plattform eller en ny huvudversion. Microsofts Cloud Adoption Framework beskriver att flytta arbetslaster till Azure, från egna servrar eller från ett annat moln, som fem steg: planera, förbered arbetslasterna, genomför, optimera och avveckla källan. Esris kunder flyttar sina kartor och data från ArcMap till ArcGIS Pro. En Next.js-applikation går från Pages Router till App Router, där `getServerSideProps` ger vika för datahämtning inuti Server Components. Ett PostgreSQL-kluster går över till en ny huvudversion, som ofta ändrar uppbyggnaden av systemtabellerna, så datan måste föras över med `pg_upgrade` eller med en dump och en återställning.

I liten skala är en migrering en enda versionerad ändring av strukturen i en databas, sparad som en fil i repot och körd av ett verktyg i en fast ordning. Django kallar sina migreringar ett versionshanteringssystem för databasschemat: varje fil beskriver en ändring i modellerna, committas tillsammans med koden och körs i samma ordning på utvecklarens dator, i staging och i produktion. GitLab levererar batchvisa bakgrundsmigreringar som skriver om stora tabeller en del i taget efter en uppgradering, till exempel när heltalskolumner görs om till

`bigint` innan värdena slår i taket. En datamigrering hör också hemma här: den ändrar raderna i stället för kolumnerna, till exempel genom att fylla ett nytt fält utifrån två gamla.

Det här är en idé i två storlekar. Båda har ett känt utgångsläge, en ordnad följd av steg, ett mål och en väg tillbaka om ett steg går snett. En plattformsflytt innehåller oftast schemamigreringar, och en lång rad schemamigreringar är med åren en plattform som byts ut bit för bit under dem som använder den.

Använd begreppet i ett beslut

Använd ”Vad är en migrering?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Data som flyttas ur något åldrande och in i något som fortfarande underhålls, medan verksamheten fortsätter som vanligt under tiden.

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. Gjorde GIS-kartan 10 gånger snabbare i PostgreSQL — 10x snabbare kartor, efter flytten från MSSQL

Gjorde GIS-kartan 10 gånger snabbare i PostgreSQL Hör av er →

Varför det spelar roll

Programvara lever längre än det den är byggd på. Molnavtal omförhandlas, ramverk byter modell, och leverantörer avvecklar produkter. Ett system som ska leva i tio år migreras flera gånger under de tio åren, så hur väl ett team migrerar betyder till slut mer än hur väl det valde den första plattformen.

- **Mellanläget är där kostnaden hamnar.** Så länge en del av systemet körs på den gamla plattformen och en del på den nya måste båda driftas, övervakas och hållas i takt. Cloud Adoption Framework uppmanar teamen att hålla den perioden kort och att flytta komponenter som är direkt beroende av varandra i samma våg.
- **En databas går inte att driftsätta på nytt.** Kod kan byggas om och bytas ut; raderna i produktion kan inte det. Schemamigreringar gör en strukturändring i en databas i drift till en fil som kan granskas, testas mot en kopia, köras i ordning och för de flesta ändringar rullas tillbaka. Alternativet är ett produktionsschema som ser ut så som den senaste personen med en konsol lämnade det.
- **En planerad flytt går att ångra.** Cloud Adoption Framework kräver att kriterier och rutiner för återgång ligger fast innan en migrering börjar, att återgången är testad och att det finns go/no-go-kontrollpunkter längs vägen.

  Avvägningarna ligger i själva omställningen. Cloud Adoption Framework skiljer mellan migrering med driftstopp och migrering med nästan noll driftstopp, där data replikeras löpande och bytet sker i slutet. För en ny huvudversion av

PostgreSQL kortar `pg_upgrade` omställningen: verktyget hoppar över den dump och återställning som en sådan uppgradering vanligtvis kräver genom att återanvända de befintliga datafilerna, och med `--link`, som inte kopierar några filer, går det ännu mycket snabbare.

Så ser det ut i praktiken

Tre projekt på den här webbplatsen visar begreppet i tre storlekar, från en hel plattform ned till ordningen på filerna i ett enda repo.

En migrering var en flytt bort från en molnleverantör. En kunds AWS-miljö var överdimensionerad, och arbetet började med en genomgång av vad som faktiskt användes: onödiga tjänster togs bort och återstående kapacitet anpassades. Därefter flyttades arbetslasterna bort från AWS. Beskrivningen av uppdraget ger belägg för de infrastrukturändringarna. Att ta bort oanvända tjänster före flytten innebar att det fanns mindre att flytta; beskrivningen fastställer inte ett kostnadsförhållande på jämförbara grunder, ett produktivitetsresultat eller en störningsfri migrering.

Det andra var ett byte av databasmotor med en ny design i andra änden. Databasen i en geografisk kartapplikation gick från MSSQL till PostgreSQL med tillägget PostGIS, och målet var inte en kopia av den gamla strukturen. Ett nytt schema utformades för spatiala data, med GiST- och SP-GiST-index och strikta främmande nycklar och CHECK-villkor som upprätthåller valideringsregler för spatiala koordinater. Över 50 miljoner spatiala poster flyttades in via ETL-pipelines som transformerade dem till EPSG:4326. En flytt till en ny design är chansen att skriva in datans regler i själva schemat. Sidan anger att prestandan för spatiala databasfrågor blev 10 gånger bättre och att CPU-belastningen i backend minskade med 65 %.

Det tredje är den lilla skalan, underhållen i flera år. En plattforms schema är summan av 1 022 migreringar, numrerade från 036 till 1102, och sidan kallar ordningsdisciplinen kring dem oglamorös och inte förhandlingsbar. Historiken i repot är en beskrivning av produktion, och den stämmer bara så länge varje databas, från en nyskapad utvecklarkopia till en återställning från säkerhetskopia, kan byggas upp igen genom att historiken spelas upp från första till sista fil.

Var det går fel

Omställningen får planeringen, medan problemen gärna samlas i veckorna runt den, i fyra igenkännbara former.

Två system går sida vid sida längre än planerat. Flytten stannar halvvägs, bitarna som återstår är de svåra, och teamet driftar båda plattformarna i månader. Användarna känner skarven: Next.js-guiden påpekar att en övergång mellan en sida under Pages Router och en sida under App Router är en hård navigering, och att `next/link` inte förhämtar mellan de två. Håll vågorna små, och gör varje nästa steg billigt. En frontend här flyttar sina sidor till en gemensam fabrik som laddar varje sida med en enda rundtur, så att varje sida som flyttas ärver beteendet.

En omställning utan väg tillbaka. Vissa steg ändrar källan lika väl som målet. Med `pg_upgrade --link` har det nya klustret, så snart det har startats, skrivit till filer som det gamla delar, och det gamla klustret måste då återställas från säkerhetskopia. Esri varnar för att en datamängd som har uppgraderats för ArcGIS Pro kanske inte längre går att använda i ArcMap, och ber användarna ta en kopia först. I Django ger en `RunSQL` -operation utan omvänd SQL ett `IrreversibleError` i samma ögonblick som någon migrerar bakåt. Bestäm kriterierna för återgång före första steget, öva vägen tillbaka och gör den till en kontroll: ett repo här har en kontrollmotor vars validerare bland annat verifierar att en migrering går att rulla tillbaka.

En migrering som redan har körts redigeras. Migreringsverktyg registrerar vilka filer som har körts och hoppar över dem från och med då. Ändrar du en fil efter att den har nått produktion behåller produktion den gamla versionen, medan varje ny databas får den nya: två scheman under en och samma historik. Låt körda migreringar vara och rätta dem med en ny; ett bygge som spelar upp hela historiken från en tom databas och jämför resultatet med en migrerad kopia fångar avvikelsen.

Den uppgraderade databasen är långsam första dagen. `pg_upgrade` för över det mesta av planerarens statistik, men inte allt, och planeraren gissar på resten tills den har byggts upp igen. Kontrolläget `--check` ändrar ingenting och kan köras medan den gamla servern fortfarande betjänar användare, så kör det först, och bygg upp den saknade statistiken direkt efter uppgraderingen innan du samlar in 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
```

Läs mer

- Plan your migration — Microsofts metod för att flytta arbetslaster till Azure: vågor, val av omställning och planering av återgång.
- Migrations — Djangos guide till versionerade schemaändringar, hur de rullas tillbaka och datamigreringar.
- pg_upgrade — uppgraderingsverktyget i PostgreSQL, med kontrolläge, länkläge, återgång och statistikstegen efteråt.
- How to migrate from Pages to the App Router — så flyttar du en Next.js-applikation till den nya routern en sida i taget.
- Migrate existing work — vad som följer med från ArcMap till ArcGIS Pro, vad som måste konverteras och vad du bör kopiera först.
- Check migrations before upgrade — hur GitLabs batchvisa bakgrundsmigreringar skriver om stora tabeller efter en uppgradering och hur du ser att de är klara.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en migrering?” 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/migration/