Engineer

Git

versionsstyringen, dette sites historik lever i

Hvad det er

Git er et distribueret versionsstyringssystem: et program, der gemmer tilstanden af en samling filer som en kæde af øjebliksbilleder, så enhver tidligere tilstand kan ses, sammenlignes med nutiden eller hentes tilbage. Projektet skriver navnet Git og kommandoen `git`, og begge betegner det samme værktøj.

Hvert øjebliksbillede er en commit. En commit gemmer, hvordan hver fulgt fil så ud i det øjeblik, hvem der lavede ændringen og hvornår, en besked om hvorfor, og en henvisning til den eller de commits, der kom før. En fil, git allerede kender, gemmes kun én gang: er den uændret fra én commit til den næste, peger det nye øjebliksbillede på den kopi, der allerede ligger på disken, så en lang historik fylder langt mindre, end en fuld kopi pr. commit ville gøre.

Alt indeni er navngivet efter sit indhold. Før git gemmer en fil, en mappeoversigt eller en commit, beregner det en kontrolsum over bytene, i det klassiske format en SHA-1-hash skrevet som 40 hexadecimale tegn, og den kontrolsum bliver objektets navn. Ændres en enkelt byte, ændres navnet, og det samme gør navnet på hver commit, der er bygget ovenpå. Sådan opdager git beskadigede data, og derfor peger et commit-id stadig på præcis den samme kode mange år senere.

Distribueret betyder, at hver kopi er komplet. En kloning henter hele historikken ned og ikke kun de nyeste filer, så det at læse loggen, sammenligne to versioner eller lave en commit foregår på den lokale disk uden nogen server. At dele arbejde er et særskilt og bevidst skridt: man skubber sine commits til en anden kopi med push eller henter commits fra en med fetch og fletter dem ind. Enhver kopi kan spille serverens rolle; i de fleste teams gør én aftalt kopi det, ofte hos en hostingtjeneste, efter aftale og ikke fordi værktøjet kræver det.

Arbejdet bevæger sig gennem tre steder. Arbejdstræet rummer de filer, man redigerer. Staging-området, også kaldet indekset, rummer det, den næste commit skal indeholde, så man kan committe en del af sine ændringer og gemme resten til senere. Git-mappen, `.git` øverst i projektet, rummer objektdatabasen og alle referencer. En gren er blot et flytbart navn, der peger på én commit, og derfor tager det ingen tid at oprette en.

Git blev skrevet i 2005 af Linus Torvalds og fællesskabet omkring Linux-kernen, efter at BitKeeper, det proprietære værktøj de havde brugt, mistede sin gratis status. Målene var fart, et enkelt design, stærk understøttelse af tusindvis af parallelle grene, fuld distribution og evnen til at håndtere et projekt på kernens størrelse.

Brug begrebet i en beslutning

Brug »Hvad er git?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Testet, udgivet og til at rulle tilbage, så ændringer når kunderne en tirsdag eftermiddag i stedet for sent en fredag aften.

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. Har automatiseret GIS-SaaS-udrulning med CI/CD — udrulning, data og rapportering, alt sammen uden manuelt arbejde

Har automatiseret GIS-SaaS-udrulning med CI/CD Kontakt os →

Hvorfor det betyder noget

Versionsstyring svarer på de spørgsmål, der melder sig, så snart mere end én person, eller én person over mere end en uge, rører de samme filer: hvad blev ændret, hvornår, af hvem og hvorfor. Holdt i hånden lever svarene i hukommelsen, i filnavne som `endelig-v3` og i mailtråde. Med git ligger de ved siden af koden, knyttet til hver ændring, hvor alle kan søge i dem.

Den distribuerede opbygning ændrer et teams hverdag. Hver klon rummer hele historikken, så ingen venter på en central server for at læse loggen eller committe, og et nedbrud på serveren lader alle arbejde videre. Grene er billige, så hver funktion, rettelse eller hvert eksperiment kan få sin egen, blive gennemgået for sig og flettes ind, når den er klar, mens hovedlinjen hele tiden er klar til at blive sendt ud.

Samme opbygning gør git forsigtigt med data. Næsten alle handlinger tilføjer kun til objektdatabasen, så arbejde, der er committet, er svært at miste: en gren slettet ved en fejl, en rebase der gik skævt, eller en reset til den forkerte commit

kan som regel rulles tilbage. Sikkerhedsnettet er refloggen, en lokal optegnelse af, hvor hver gren pegede og hvornår, som standard gemt i 90 dage for commits, der stadig kan nås, og i 30 dage for dem, der ikke kan.

Git er også det fælles fundament under de fleste andre værktøjer i en leverancekæde. Forges, værktøjer til kodegennemgang, byggepipelines, deployscripts og statiske sitegeneratorer læser alle fra det, og dermed bliver historikken en kilde til fakta ud over selve koden: dagen en side sidst blev ændret, den commit en udgivelse blev bygget fra, den ændring en fejl kom ind med.

Sådan ser det ud i praksis

Virksomhedens egen kode viser tre sider af det.

Kildekoden lå tidligere hos en tredjeparts hostingtjeneste. Nu ligger den på en git-server, som virksomheden kører på sin egen hardware: en enkelt binær, overvåget af styresystemet og valgt, fordi den er SSH-først uden administrativ webgrænseflade. For den, der kloner og pusher, betyder det git over SSH med en nøgle: anonyme forsøg og forsøg uden nøgle bliver afvist, hvert erklæret repository er markeret privat, og forgens port 23231 lytter kun på loopback, så forbindelsen udefra går via en jump host. Siden lægger ikke skjul på prisen: at køre sin egen forge betyder også at eje autentificering, adgangskontrol, lagring og sikkerhedskopier.

Fulde kopier af historikken på flere maskiner er lette at forveksle med en sikkerhedskopi. Virksomhedens implementering af sikkerhedskopiering og gendannelsestest blev udformet til at udvide gendannelsesberedskabet ud over et udbydersnapshot. Dens restic-scripts og systemd-enheder understøtter et databasedump, hvor det er konfigureret, krypterede, deduplikerede snapshots over SFTP til lager hos en anden udbyder, udtynding efter opbevaringspolitikken og integritetstjek. Et separat månedligt workflow gendanner en kendt fil til et midlertidigt arbejdsområde og sammenligner dens bytes. Den test skal faktisk køres, før den dokumenterer et gendannelsesberedskab: implementeringsbeskrivelsen dateret 2026-10-27 angiver, at timerne var deaktiveret uden et konfigureret backuplager, og dokumenterer ingen gennemført gendannelse i drift. Selv mange kloner på bærbare computere leverer ikke den manglende dokumentation.

Historikken fødes også ind i selve sitet. Sitets kolofon fortæller, at W3C's feed-validator advarer om, at de fleste indlæg i feedsene deler ændringsdato, og at advarslen er rigtig: datoerne kommer fra git-historikken, og de fleste af siderne blev faktisk skrevet i en håndfuld commits. Sitet viser hellere advarslen end at opdigte en dato. Historikken fortæller, hvad der skete, også at meget af det skete på én gang.

Én kommando læser sådan et faktum tilbage. Den viser det korte id, commit-datoen i ISO 8601-form og emnelinjen for den seneste commit, der rørte én bestemt fil:

```
git log -1 --format='%h %cI %s' -- README.md
```

Hvor det går galt

- **En hemmelighed, der er committet én gang, bliver i historikken.** Slettes filen i den næste commit, forsvinder den fra det nyeste øjebliksbillede, men bliver i det forrige, som hver klon allerede har. Betragt nøglen som lækket og skift den; at omskrive historikken bagefter begrænser spredningen, men kopier lavet i mellemtiden beholder de gamle commits.
- **Et force push erstatter en kollegas arbejde.** `git push --force` slår det tjek fra, der sikrer, at fjerngrenen er en forfader til det, man skubber, så commits, en kollega har skubbet i mellemtiden, falder af grenen. `--force-with-lease` opdaterer kun fjerngrenen, så længe den stadig peger, hvor man sidst så den.
- **Kloner bliver behandlet som sikkerhedskopien.** En kopi på en bærbar halter bagefter, rummer kun de grene, ejeren tilfældigvis har hentet, og er aldrig blevet gendannet fra med vilje. Issues, adgangsregler og resten af en forges egne data ligger helt uden for repositoryet.
- **En stor binær fil gør hver klon tungere for altid.** Git beholder hver version af hver fil, og hver klon henter dem alle, så en video, der kom med ved en fejl og blev slettet dagen efter, følger med til hver maskine, der kloner projektet bagefter.
- **Arbejde, der kun ligger ét sted, forsvinder med det sted.** Refloggen, der redder en dårlig reset, findes på én maskine og udløber. Commits, der aldrig er skubbet, findes ingen andre steder, og en gren, der får lov at ligge i måneder, glider så langt væk, at det bliver et projekt i sig selv at flette den ind igen.

Læs mere

- Git documentation — projektets egen referencemanual, bogen og videoerne samlet ét sted.
- Pro Git — hele bogen online, fra den første commit til objektdatabasen nedenunder.
- What is Git? — øjebliksbilleder, lokale handlinger, kontrolsummer og de tre tilstande i ét kapitel.
- A Short History of Git — hvorfor kernens fællesskab skrev sit eget værktøj i 2005, og hvad det skulle kunne.
- git-reflog — sådan finder man, hvor en gren pegede før fejlen, og hvor længe den optegnelse holder.
- git-push — force-tilvalgene, og hvad hvert af dem beskytter.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er git?« 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/git/