Engineer

Forge

en felles server for koderepoer og samarbeid

Hva det er

En forge, også kalt programvareforge, kodeforge eller git-forge, er den serveren et team har blitt enige om å regne som hjemmet til repoene sine, sammen med tjenestene den kjører rundt dem. Hver fullstendige klone av et git-repo bærer hele historikken, så i prinsippet står ingen kopi over en annen. I praksis trenger teamet likevel ett sted å pushe til, ett sted som bestemmer hvem som får lese og skrive, og ett sted der samtalen om koden foregår. Det stedet er forgen.

Under overflaten lagrer en forge nakne repoer, den formen som bare inneholder historikken og ingen utsjekkede filer, og den snakker én eller flere av gits overføringsprotokoller. Boka Pro Git lister opp fire: den lokale protokollen for en mappe på samme maskin, HTTP, SSH og gits egen protokoll, en daemon på port 9418 helt uten autentisering. Alt annet bygges oppå, og hvor mye som bygges, varierer kraftig.

Dokumentasjonen til Forgejo beskriver den fulle enden av skalaen: saksoppfølging, pull requests, en innebygd wiki, et pakkeregister for mange språkøkosystemer, Forgejo Actions for å kjøre jobber, speiling av repoer, migrering fra andre tjenester, beskyttelse av grener og tagger, og webhooks. Soft Serve ligger i motsatt ende. Prosjektet beskriver seg selv som en git-server for kommandolinjen som du kan drifte selv: én binærfil kalt `soft` som serverer repoer over SSH, HTTP eller git-protokollen, styrer tilgang med offentlige SSH-nøkler og støtter samarbeidspartnere, private repoer og webhooks. Begge er forger. Forskjellen ligger i hvor mye av teamets arbeid de prøver å romme.

Forgene har også begynt å snakke med hverandre. ForgeFed er en føderasjonsprotokoll for plattformer der programvareprosjekter ligger og utvikles i fellesskap, bygd som en utvidelse av ActivityPub. Den modellerer repoer, sakssystemer, patch-sporere, prosjekter og team som aktører som sender

aktiviteter til hverandre. Målet er et felles språk på tvers av nettverk av forger og verktøy for prosjektstyring, slik at en person på én server kan opprette en sak mot et repo på en annen, og slik at et repo kan flytte mellom servere mens den gamle kopien arkiveres som skrivebeskyttet og peker videre til det nye hjemmet.

Bruk begrepet i en beslutning

Bruk «Hva er en forge?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Testet, utgitt og mulig å reversere, slik at endringer når kundene en tirsdag ettermiddag i stedet for sent en fredagskveld.

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. Automatiserte GIS-SaaS-utrulling i GitHub Actions — produksjonssetting, data og rapportering, alt uten manuelle grep

Automatiserte GIS-SaaS-utrulling i GitHub Actions Ta kontakt →

Hvorfor det er viktig

Forgen er stedet der regler faktisk kan håndheves. En regel som bor på en utviklers maskin, gjelder bare så lenge den maskinen er satt opp riktig, mens en hook på serveren kjører ved hver eneste push. Gits `pre-receive` -hook leser hver pushet ref som en gammel id, en ny id og et navn på standard input. Avslutter den med en annen status enn null, blir ingen av refene oppdatert. Grenbeskyttelse, påkrevd kodegjennomgang og krav om signerte commits på de større forgene forfiner den samme ideen: den felles kopien avviser det teamet har blitt enige om ikke å ta imot.

Å velge forge er også å velge hvem som sitter på koden. En vertstjeneste tar driftskostnadene og oppetiden, og til gjengjeld ligger koden, tilgangslisten og hver eneste diskusjon på noen andres maskinvare og på noen andres vilkår. Selvhosting snur avveiningen. Begge valgene har en pris, og forskjellen betyr mest for en virksomhet som forteller kundene sine hvor dataene deres befinner seg.

En forge holder dessuten data som git selv ikke holder. Commits og tagger følger med hver klone, mens saker, gjennomgangstråder, tilgangslister og hooks på serversiden hører til forgen alene. Det skillet avgjør hva en sikkerhetskopi må dekke, og hva en flytting fra én forge til en annen må ta med seg.

Slik ser det ut i praksis

Selskapet drifter sin egen forge, og tre prosjekter på dette nettstedet viser hva det krevde.

Kildekoden lå tidligere hos en tredjeparts vertstjeneste, og prosjektet der selskapets egen git-forge ble satt i drift tar utgangspunkt i at å drive en forge betyr å drive en forge: autentisering, tilgangskontroll, lagring, sikkerhetskopier og et offentlig ansikt utad. Forgen er Soft Serve, én binærfil som styres av operativsystemet, valgt fordi den er SSH-først uten noe administrativt webgrensesnitt. Den er privat som standard: anonym tilgang avvises, tilgang uten nøkkel avvises, og hvert deklarerte repo er merket som privat. SSH-lytteren binder seg bare til loopback-grensesnittet på port 23231 og nås utenfra via en jump host. En multiplekser som ville lagt HTTPS og git over SSH på én offentlig port, er skrevet og står avslått, fordi en lytter ingen bruker, er angrepsflate. Resultatet sies rett ut: selskapet hoster sin egen kode på sin egen maskinvare.

Repoet til dette nettstedet bor der. Instruksjonene i det utpeker den selvhostede forgen, Soft Serve over SSH, som kanonisk remote, kaller den eldre kopien hos vertstjenesten et utdatert speil og slår fast at det ikke finnes noen hostet CI. Kvalitetsporten kjører lokalt og i git-hooks.

En hook på serversiden må testes i mottakermiljøet for en ekte Git-push, så hooken ble prøvekjørt mot et ekte nakent repo og en ekte push før den fikk tillit. Prøvekjøringen oppretter et nakent repo, installerer hooken, kloner, committer, pusher og kontrollerer både veien til godkjenning og veien til avvisning. Den fant to feil, og ingen av dem var synlige i selve filen. Den andre er den lærerike: git eksporterer en variabel som peker på repoets mappe når den kjører en hook, testsuiten hooken startet, arvet den, og nitten tester ble røde på korrekt kode helt til miljøet ble tømt ved grensen. Siden nevner også sin egen grense: tvungne oppdateringer og sletting av grener er push-typer prøvekjøringen ikke bygger opp.

Alt dette deler én vert med én kjerne og 512 MB minne med alt annet selskapet gjør tilgjengelig offentlig, fra de sju TLS-domenene til onion-tjenesten, en Gemini-server, en Gopher-server og utestenging av inntrengere. Automatisering av sikkerhetskopiering er implementert separat; det er noe annet enn aktive sikkerhetskopier. En forge med webpanel ville ha vært enda en fast kjørende applikasjon innenfor dette budsjettet. Derfor ble plattformer med webpanel

utelukket på designstadiet; på denne verten nådde tilgjengelig minne et lavpunkt på omtrent 310 MB av 464 i løpet av et dokumentert døgn med stikkprøvemålinger.

Hvor det går galt

Anonyme besøkende får lesetilgang som standard på enkelte forger. Dokumentasjonen til Soft Serve sier at den fra start gir anonyme tilkoblinger skrivebeskyttet tilgang, styrt av to innstillinger, `anon-access` og `allow-keyless`. En privat forge setter den første til `no-access` og den andre til `false`, og resultatet er verdt å teste fra en maskin uten noen nøkkel lastet inn. Gits egen daemon er enda mer kontant: Pro Git påpeker at et repo den serverer, enten kan klones av alle eller ikke serveres i det hele tatt.

En hook som kaller git i en annen mappe, treffer feil repo. Under en push kjører git de mottakende hookene inne i det nakne repoet og eksporterer `GIT_DIR` og beslektede variabler, slik at git-kommandoer i hooken finner det. Alt hooken starter, arver dem, også en testkjøring i en utsjekking et helt annet sted. Manualsiden githooks sier at disse variablene skal tømmes før du arbeider i et fremmed repo. En minimal `pre-receive`-hook som tester hver pushet commit i en midlertidig mappe, gjør det i et subshell:

```
while read -r _ newrev _; do
  git cat-file -e "$newrev^{commit}" 2>/dev/null || continue
  tree=$(mktemp -d)
  git archive "$newrev" | tar -x -C "$tree"
  (
    unset $(git rev-parse --local-env-vars)
    cd "$tree" && ./run-tests
  ) || exit 1
done
```

En gjenopprettet forge kommer tilbake med koden og uten reglene sine. Hvilken som helst klone kan fylle repoene på en tapt server igjen, men Pro Git advarer om at hooks på serversiden og lignende innstillinger blir igjen, og saker, gjennomganger, nøkler og tilgangslister var aldri med i klonene. Å ta sikkerhetskopi av en forge betyr å kopiere både datamappen og konfigurasjonen, så gjenopprette begge et annet sted og sjekke at en push hooken skal avvise, faktisk blir avvist.

Hver lytter en forge åpner, er noe som må forsvares. Eksempelkonfigurasjonen til Soft Serve har SSH på port 23231, HTTP på 23232, git-daemonen på 9418 og en statistikkserver på 23233. Et team som bare pusher over SSH, kan slå av resten eller binde dem til loopback, slik at de åpne portene ikke blir flere enn det som faktisk brukes.

En flytting til en annen forge tar med koden og lar samtalen bli igjen. En speilpush kopierer hver ref under `refs/`, grener og tagger inkludert, og det er alt den kopierer. Saker, gjennomgangstråder, etiketter og tilgangslister krever migreringsverktøyet til den nye forgen, eller en protokoll som ForgeFed, som kopierer commits, tagger, saker, merge requests og kommentarer og lar det gamle repoet stå arkivert med en peker til det nye.

Les mer

- ForgeFed-spesifikasjonen — føderasjonsprotokollen for forger, med aktørene, aktivitetene og flyttingen som arkiverer et gammelt repo.
- Soft Serve — prosjektets egen dokumentasjon for en SSH-først git-server i én binærfil, med innstillingene `anon-access` og `allow-keyless`.
- Forgejo-dokumentasjonen — veiledninger for administratorer og brukere av en fullverdig selvhostet forge, fra installasjon og sikkerhetskopier til Actions og speiling.
- githooks — alle hookene git kjører, blant dem `pre-receive`, og miljøet en push-hook arver.
- The Protocols — de fire måtene git flytter data mellom maskiner på, og hva hver av dem koster i oppsett og sikkerhet.
- ActivityPub — W3C-anbefalingen som ForgeFed utvider.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en forge?» 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/forge/