Engineer

Forge

en gemensam server för kodrepon och samarbete

Vad det är

En forge, på engelska även software forge eller code forge och i vardagligt tal ofta git-forge, är den server ett team utser till hemvist för sina repon, tillsammans med tjänsterna som körs runt dem. Git kräver ingen sådan server. Varje fullständig klon bär hela historiken, och tekniskt sett är ingen kopia förmer än någon annan. Ett team behöver ändå ett ställe att pusha till, ett ställe som avgör vem som får läsa och skriva och ett ställe där samtalet om koden förs. Det stället är forgen.

Under ytan lagrar en forge bare-repon, alltså repon som bara innehåller historiken och inga utcheckade filer, och den talar ett eller flera av gits överföringsprotokoll. Git-projektets bok räknar upp fyra: det lokala protokollet mot en katalog på samma maskin, HTTP, SSH och gits eget protokoll, en daemon på port 9418 helt utan autentisering. Allt annat byggs ovanpå, och hur mycket som byggs skiljer sig kraftigt.

Forgejos dokumentation visar den fullt utbyggda änden av skalan: ärendehantering, pull requests, en inbyggd wiki, ett paketregister för många språks ekosystem, Forgejo Actions för att köra jobb, spegling av repon, migrering från andra tjänster, skydd av grenar och taggar samt webhookar. Soft Serve står i den andra änden. Projektet kallar sig en git-server för kommandoraden som man hostar själv: en enda binärfil, `soft`, som servar repon över SSH, HTTP eller git-protokollet, styr åtkomsten med publika SSH-nycklar och har stöd för medverkande, privata repon och webhookar. Båda räknas som forge. Det som skiljer dem är hur stor del av teamets arbete de försöker rymma.

Servrarna har också börjat tala med varandra. ForgeFed är ett federationsprotokoll för plattformar som hostar och samordnar mjukvaruprojekt, byggt som en utvidgning av ActivityPub. Det beskriver repon,

ärendehanterare, patchhanterare, projekt och team som aktörer som skickar aktiviteter till varandra. Målet är ett gemensamt språk i ett nätverk av forgar och projektverktyg: en person på en server ska kunna öppna ett ärende mot ett repo på en annan, och ett repo ska kunna flytta mellan servrar medan den gamla kopian arkiveras som skrivskyddad och pekar vidare till sitt nya hem.

Använd begreppet i ett beslut

Använd ”Vad är en forge?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Testade, släppta och möjliga att backa, så att ändringar når kunderna en tisdagseftermiddag i stället för sent en fredagskväll.

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. Automatiserade GIS-driftsättning i GitHub Actions — driftsättning, data och rapportering, helt utan manuellt arbete

Automatiserade GIS-driftsättning i GitHub Actions Hör av er →

Varför det spelar roll

Det är på forgen som en regel går att genomdriva. En regel som bor på en utvecklares dator gäller bara så länge den datorn är rätt uppsatt, medan en hook på servern körs vid varje push. Gits `pre-receive`-hook läser varje pushad referens som ett gammalt id, ett nytt id och ett referensnamn på standard in. Avslutar den med en annan status än noll uppdateras ingen referens alls. Grenskydd, obligatorisk kodgranskning och krav på signerade commits hos de större forgarna är förfiningar av samma idé: den gemensamma kopian säger nej till det som teamet har bestämt att inte ta emot.

Att välja forge är också att välja vem som håller koden. En hostad tjänst tar på sig driften och upptiden, och i gengäld ligger koden, åtkomstlistan och varje diskussion på någon annans hårdvara och på någon annans villkor. Att hosta själv vänder på den avvägningen. Båda vägarna kostar något, och skillnaden väger tyngst för ett företag som berättar för sina kunder var deras data finns.

En forge håller dessutom data som git inte håller. Commits och taggar följer med i varje klon, medan ärenden, granskningstrådar, åtkomstlistor och hookar på serversidan bara finns på forgen. Den gränsen bestämmer vad en säkerhetskopia måste omfatta och vad en flytt mellan två forgar måste få med sig.

Så ser det ut i praktiken

Den här webbplatsens kod ligger på en forge som företaget driver själv, och tre projekt visar vad det krävde.

Källkoden låg tidigare hos en extern värdtjänst. Projektet där företaget driftsatte sin egen git-forge utgår från att den som kör en forge också kör allt som hör till: autentisering, åtkomstkontroll, lagring, säkerhetskopior och ett ansikte utåt. Valet föll på Soft Serve, en enda binärfil som hålls igång av operativsystemet, eftersom den är byggd för SSH först och saknar administrativt webbgränssnitt. Den är privat som standard. Anonym åtkomst nekas, åtkomst utan nyckel nekas och varje deklarerat repo är markerat som privat. SSH-lyssnaren binder bara till loopback-gränssnittet på port 23231 och nås utifrån via en hoppserver. En multiplexer som skulle lägga HTTPS och git över SSH på en och samma publika port är skriven men avstängd, eftersom en lyssnare som ingen använder är attackyta. Resultatet sägs rakt ut: företaget hostar sin egen kod på sin egen hårdvara.

Webbplatsens eget repo bor där. Repots instruktioner pekar ut den egenhostade forgen, Soft Serve över SSH, som kanonisk remote, kallar den äldre hostade kopian en kvarlämnad spegel och konstaterar att det inte finns någon hostad CI. Kvalitetsgrinden körs lokalt och i git-hookar.

En hook på serversidan behöver testas i den mottagande miljön för en riktig Git-push. Därför provkördes hooken mot ett riktigt bare-repo och en riktig push innan den fick förtroende. Provkörningen skapar ett bare-repo, installerar hooken, klonar, committar, pushar och verifierar både vägen där pushen godtas och vägen där den avvisas. Den hittade två fel, och inget av dem syntes i filen. Det andra är det lärorika. När git kör en hook exporterar det en variabel som pekar ut repots katalog, testsviten som hooken startade ärvde den, och nitton tester blev röda på korrekt kod tills miljön rensades vid gränsen. Sidan anger också sin egen begränsning: tvingade uppdateringar och borttagna grenar är pushar som provkörningen inte konstruerar.

Allt detta delar en enda värddator med en kärna och 512 MB minne med allt annat som företaget tillhandahåller publikt, från de sju TLS-domänerna till onion-tjänsten, en Gemini- och en Gopher-server och spärrning av intrångsförsök. Automatisering av säkerhetskopiering har implementerats separat; det är något annat än aktiva säkerhetskopior. En forge med webbpanel skulle ha varit ytterligare en ständigt körande applikation inom den budgeten, vilket är skälet

till att plattformar med webbpanel valdes bort redan i designskedet; på den värddatorn var den lägsta nivån för tillgängligt minne ungefär 310 MB av 464 under ett dokumenterat dygns stickprov.

Var det går fel

Anonyma besökare får läsrätt som standard på vissa forgar. Soft Serves dokumentation anger att den från start ger anonyma anslutningar skrivskyddad åtkomst, styrd av två inställningar, `anon-access` och `allow-keyless`. En privat forge sätter den första till `no-access` och den andra till false, och resultatet är värt att testa från en maskin där ingen nyckel är laddad. Gits egen daemon känner bara till allt eller inget: enligt Git-projektets bok kan ett repo som den servar antingen klonas av alla eller inte servas alls.

En hook som anropar git i en annan katalog träffar fel repo. Under en push kör git de mottagande hookarna inne i bare-repot och exporterar `GIT_DIR` och närliggande variabler, så att git-kommandon i hooken hittar rätt. Allt som hooken startar ärver dem, även en testkörning i en utcheckning någon annanstans. Manualen för githooks säger att variablerna ska rensas innan man arbetar i ett främmande repo. En minimal `pre-receive`-hook som testar varje pushad commit i en tillfällig katalog gör det i ett 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 återställd forge kommer tillbaka med sin kod men utan sina regler. Vilken klon som helst kan fylla en förlorad servers repon igen, men Git-projektets bok varnar för att hookar på serversidan och liknande inställningar stannar på den gamla maskinen, och ärenden, granskningar, nycklar och åtkomstlistor fanns aldrig i klonerna. Att säkerhetskopiera en forge betyder att säkerhetskopiera dess datakatalog och dess konfiguration, att återställa båda på en annan maskin och att kontrollera att en push som hooken ska avvisa faktiskt avvisas.

Varje lyssnare som en forge öppnar måste försvaras. Soft Serves exempelkonfiguration har SSH på port 23231, HTTP på 23232, git-daemonen på 9418 och en statistikserver på 23233. Ett team som bara pushar över SSH kan stänga av resten eller binda dem till loopback, så att de öppna portarna motsvarar det som faktiskt används.

En flytt till en annan forge tar med koden och lämnar samtalet kvar. En spegelpush kopierar varje referens under `refs/`, grenar och taggar inräknade, och mer än så kopierar den inte. Ärenden, granskningstrådar, etiketter och åtkomstlistor kräver den nya forgens migreringsverktyg eller ett protokoll som ForgeFed, som kopierar commits, taggar, ärenden, merge requests och kommentarer och lämnar det gamla repot arkiverat med en hänvisning till det nya.

Läs mer

- ForgeFed-specifikationen — federationsprotokollet för forgar, med aktörerna, aktiviteterna och flytten som arkiverar ett gammalt repo.
- Soft Serve — projektets egen dokumentation för en git-server i en enda binärfil med SSH först, inklusive `anon-access` och `allow-keyless`.
- Forgejo-dokumentationen — handledningar för administratörer och användare av en fullt utbyggd forge som man hostar själv, från installation och säkerhetskopiering till Actions och speglar.
- githooks — alla hookar som git kör, bl.a. `pre-receive`, och miljön som en push-hook ärver.
- The Protocols — de fyra sätt som git flyttar data mellan maskiner på, och vad vart och ett kostar i uppsättning och säkerhet.
- ActivityPub — W3C-rekommendationen som ForgeFed bygger ut.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en forge?” 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/forge/