Engineer

GitLab

en git-plattform du også kan drifte selv

Hva det er

GitLab er én applikasjon bygd rundt git-repoer. Den holder koden og legger til merge requests for gjennomgang, saker for planlegging og et innebygd CI/CD-system som bygger, tester og produksjonssetter fra det samme stedet. Det som skiller GitLab ut, er valget av hvor det kjører. Det samme produktet tilbys som GitLab.com, tjenesten GitLab drifter selv; som GitLab Self-Managed, som du installerer på dine egne servere; og som GitLab Dedicated, en instans med én enkelt leietaker som GitLab drifter og vedlikeholder for én kunde, i den AWS-regionen kunden foretrekker.

Under ligger vanlig git. Hvert prosjekt er et git-repo som kan klones, pushes til og hentes fra med de vanlige kommandoene, og historikken i en klone er komplett. Alt GitLab legger oppå, fra tråder i gjennomganger og saker til resultater fra pipelines og rettigheter, oppbevares av GitLab selv, utenfor repoet, og en klone tar ikke med seg noe av det.

En merge request er GitLabs navn på en foreslått endring som venter på å bli flettet, den samme ideen som andre plattformer kaller en pull request. Dokumentasjonen beskriver den som et sentralt sted for å gjennomgå kode, holde diskusjonen og følge endringen. Én side viser beskrivelsen, kodeendringene med kommentarer rett i linjene, tilstanden til CI/CD-pipelinene, rapportene om hvorvidt endringen kan flettes, og listen over commits. En merge request kan kobles til en sak og lukke saken automatisk når den blir flettet. Som standard er prosjektets standardgren, vanligvis main, beskyttet, og bare folk med rollen Maintainer eller høyere kan flette inn i den.

CI/CD-systemet konfigureres i en fil som heter `.gitlab-ci.yml`, øverst i prosjektet. Filen definerer stadiene, jobbene og skriptene som til sammen utgjør en pipeline: stadiene bestemmer rekkefølgen, for eksempel først bygging, så

testing og til slutt produksjonssetting, og jobbene er oppgavene som utføres i hvert stadium. Jobbene kjøres av GitLab Runner, et eget program skrevet i Go og levert som én binærfil. En runner kobler seg til en GitLab-instans, venter på jobber, kjører hver av dem på selve maskinen, i en Docker-container eller på en ekstern server over SSH, og rapporterer resultatet tilbake.

Bruk begrepet i en beslutning

Bruk «Hva er GitLab?» 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

Det meste av det et programvareteam gjør mellom å skrive kode og å levere den, kan skje inne i én GitLab-instans: koden, gjennomgangen, pipelinen som tester den, saken som ba om den, og oversikten over hvem som godkjente hva. Én applikasjon betyr ett sett kontoer og rettigheter, ett sted å søke, og lenker som holder mellom en sak, merge requesten som svarte på den, og pipelinen som sjekket den.

Muligheten til å drifte GitLab selv er grunnen til at mange organisasjoner velger det i det hele tatt. Når koden må bli værende i selskapets egen infrastruktur, enten på grunn av en kontrakt, en tilsynsmyndighet eller rett og slett interne regler, er en vertstjeneste utelukket, og GitLab kan installeres inne i det nettverket som skal holde på koden. Dokumentasjonen behandler dette som et fullverdig bruksområde: den publiserer referansearkitekturer for installasjoner dimensjonert for alt fra 1 000 til 50 000 brukere, ved siden av Linux-pakken og et Helm-chart for Kubernetes.

Den kontrollen har en pris, og den betales i drift og ikke bare i lisensavgifter. Å drifte GitLab selv betyr å dimensjonere en server for det, holde PostgreSQL-databasen og Redis- eller Valkey-lageret friske, skaffe maskiner til runnerne, ta sikkerhetskopier som faktisk lar seg gjenopprette, og gå gjennom oppgraderinger i den rekkefølgen prosjektet krever. Med de hostede variantene tar GitLab seg av det arbeidet. Med Self-Managed blir det liggende hos deg.

Slik ser det ut i praksis

Selskapets eget arbeid har møtt GitLab fra begge kanter.

I flere år med dataingeniørarbeid tok CI/CD-praksis bygget på GitHub og GitLab over manuelle steg i både dataanalysen og programvareutviklingen, med Bash og Python til automatiseringen i bunnen. Måten arbeidet gikk fra utvikling til leveranse på, ble standardisert slik at den var lik hver gang i stedet for å bli funnet opp på nytt for hvert prosjekt. Analysesiden, som lenge var blitt behandlet som en egen og mer manuell verden, kom inn i den samme pipelinen som utviklingsarbeidet. Siden oppgir omtrent 4 000 sparte timer, og arbeid som ble levert mer konsistent fordi det sluttet å bli gjort på en litt annen måte hver gang.

Selskapets egen kode viser den andre enden av det å drifte selv. Da selskapet flyttet bort fra en tredjeparts vertstjeneste, gikk det til en forge det drifter selv, som er én binærfil, SSH-først og uten administrativt webgrensesnitt, valgt etter prinsippet om at jo mindre grensesnitt, desto mindre å forsvare. Denne forgen deler en vert for offentlig tilgjengelige tjenester med 512 MB minne, hvorav omtrent 464 MB er brukbart, med en webserver for sju domener, en Tor-onion-tjeneste, en Gemini-server, en Gopher-server og utestenging av inntrengere. Automatisering av sikkerhetskopiering er implementert separat; implementeringen er noe annet enn aktive sikkerhetskopier.

GitLab nevnes ikke på noen av de to sidene, så sammenligningen som følger, er denne artikkelens eget regnestykke. GitLab oppgir 8 GB minne som det minste en installasjon på én node kan kjøre med, og hele den verten har en sekstendedel av det. Å kunne drifte GitLab selv betyr å ha en server dimensjonert for GitLab, og en liten forge med ett eneste formål er den andre måten å holde koden på egne maskiner.

En minimal pipelinefil viser formen på CI/CD-konfigurasjonen, med to stadier som kjører etter hverandre og én jobb i hvert:

```
stages:
    - build
    - test

build-site:
    stage: build
    script:
```

```
        - make build

run-tests:
    stage: test
    script:
        - make test
```

Hvor det går galt

- **En liten server gjør hele instansen treg.** GitLabs dokumentasjon oppgir 16 GB minne som utgangspunkt for en installasjon på én node, og 8 GB som det minste den kan kjøre med i et miljø med knapt minne. Den advarer også om at swap kan gi betydelig dårligere ytelse under last, og ber om at swap slås av der det lar seg gjøre, eller om så mye minne at GitLab aldri rører den.
- **En gjenopprettet sikkerhetskopi gir en instans som står uten nøklene til sine egne data.** Kommandoen for sikkerhetskopiering utelater konfigurasjonskatalogen, `/etc/gitlab`, og med den `gitlab-secrets.json`, nøklene som dekrypterer krypterte verdier i databasen. En gjenopprettet database holder de verdiene låst så lenge den filen mangler. Dokumentasjonen ber om at konfigurasjonen sikkerhetskopieres for seg, i det minste `gitlab-secrets.json` og `gitlab.rb`.
- **En oppgradering hopper over et påkrevd stopp.** Oppgraderingsstiene har påkrevde stopp, versjoner som må installeres før noen senere versjon, og siden GitLab 17.5 ligger de på x.2, x.5, x.8 og x.11. Bakgrunnsmigreringene i hvert stopp må bli ferdige før neste oppgradering starter, så en instans som har sakket flere versjoner akterut, tar igjen ett stopp om gangen.
- **Et administrert datalager viser seg å falle utenfor det som støttes.** PostgreSQL er den eneste støttede databasen. Redis eller Valkey holder økter og bakgrunnsoppgaver og må være en frittstående instans: Redis Cluster og serverløse varianter av Redis og Valkey støttes ikke. En administrert skytjeneste er tillatt, så lenge den er av den frittstående typen.
- **Pipelinene venter på maskiner ingen planla for.** GitLab.com tilbyr ferdige runnere, men på en Self-Managed-instans er det administratoren som skaffer og forvalter infrastrukturen CI/CD-jobbene kjører på, og det omfatter å sørge for at runnerne har nok kapasitet til organisasjonens arbeidsmengde. Et travelt team med én liten runner blir stående i kø bak den.

Les mer

- GitLab Docs — den fullstendige dokumentasjonen for å bruke, installere og administrere GitLab.
- Installation requirements — minne, prosessor, lagring og de støttede versjonene av PostgreSQL og Redis.

- Back up GitLab — hva en sikkerhetskopi inneholder, hva den utelater, og hvordan du oppbevarer hemmelighetene ved siden av den.
- Upgrade paths — de påkrevde stoppene for oppgradering, og hvorfor bakgrunnsmigreringene må bli ferdige mellom dem.
- Get started with GitLab CI/CD — filen `.gitlab-ci.yml`, stadier, jobber og pipelines.
- GitLab Runner — programmet som kjører jobbene, og de ulike måtene det kan kjøre dem på.

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 GitLab?» 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/gitlab/