Engineer

GitLab

en kodeplatform, man også kan køre selv

Hvad det er

GitLab er ét samlet program bygget op om git-repositories: det opbevarer koden og lægger merge requests til gennemgang, issues til planlægning og et indbygget CI/CD-system, der bygger, tester og udruller fra samme sted, oven på. Det, der skiller det ud, er valget af, hvor det kører. Det samme produkt findes som GitLab.com, den tjeneste GitLab selv driver; som GitLab Self-Managed, som man installerer på sine egne servere; og som GitLab Dedicated, en single-tenant-instans, som GitLab hoster og vedligeholder for én kunde i den AWS-region, kunden foretrækker.

Nedenunder ligger helt almindelig git. Hvert projekt er et git-repository, der kan klones, skubbes til og hentes fra med de sædvanlige kommandoer, og historikken i en klon er komplet. Alt det, GitLab lægger ovenpå, fra tråde i en gennemgang og issues til pipelineresultater og rettigheder, opbevarer GitLab selv uden for repositoryet, og en klon har intet af det med.

En merge request er GitLabs navn for en foreslået ændring, der venter på at blive flettet ind, den samme idé, som andre platforme kalder en pull request. Dokumentationen beskriver den som et centralt sted til at gennemgå kode, føre diskussionen og følge ændringen. Én side viser beskrivelsen, kodeændringerne med kommentarer på de enkelte linjer, status for CI/CD-pipelines, rapporterne om, hvorvidt ændringen kan flettes, og listen over commits. En merge request kan knyttes til et issue og lukke det automatisk, når den bliver flettet. Som standard er projektets standardgren, som regel main, beskyttet, og kun personer med rollen Maintainer eller højere må flette ind i den.

CI/CD-systemet sættes op i en fil ved navn `.gitlab-ci.yml` i roden af projektet. Den definerer de stages, job og scripts, en pipeline består af: stages bestemmer rækkefølgen, for eksempel build, så test og så deploy, og job er de opgaver, der

udføres i hver stage. Jobbene køres af GitLab Runner, et separat program skrevet i Go og leveret som én binær fil. En runner forbinder sig til en GitLab-instans, venter på job, kører hvert af dem direkte på maskinen, i en Docker-container eller på en fjern server over SSH, og melder resultatet tilbage.

Brug begrebet i en beslutning

Brug »Hvad er GitLab?« 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

Det meste af det, et softwareteam gør mellem at skrive kode og at sende den ud, kan foregå i én GitLab-instans: koden, gennemgangen, pipelinen der tester den, det issue, der bad om den, og optegnelsen af, hvem der godkendte hvad. Ét program betyder ét sæt konti og rettigheder, ét sted at søge og forbindelser, der holder mellem et issue, den merge request, der løste det, og den pipeline, der tjekkede det.

Muligheden for selv at drive det er grunden til, at mange organisationer vælger GitLab i første omgang. Når kode skal blive på virksomhedens egen infrastruktur, på grund af en kontrakt, en myndighed eller en intern politik, er en hostet tjeneste udelukket, og GitLab kan installeres inde i det netværk, der skal beholde koden. Dokumentationen behandler det som en fuldgyldig brug: den udgiver referencearkitekturer til installationer dimensioneret til mellem 1.000 og 50.000 brugere, ved siden af Linux-pakken og et Helm-chart til Kubernetes.

Den kontrol har en pris, og den betales i drift og ikke kun i licenser. At drive GitLab selv betyder at dimensionere en server til det, holde PostgreSQL-databasen og Redis- eller Valkey-lageret sunde, stille maskiner til rådighed for runnerne, tage sikkerhedskopier, der faktisk kan gendannes, og gå gennem opgraderinger i den rækkefølge, projektet kræver. De hostede varianter overlader det arbejde til GitLab; den selvdrevne beholder det hos en selv.

Sådan ser det ud i praksis

Virksomhedens eget arbejde har mødt GitLab fra begge sider.

Gennem flere år med dataengineering erstattede CI/CD-praksisser bygget på GitHub og GitLab, med Bash og Python til selve automatiseringen nedenunder, manuelle trin i både dataanalysen og softwareudviklingen. Måden, arbejdet bevægede sig fra udvikling til levering, blev standardiseret, så den var den samme hver gang i stedet for at blive opfundet på ny i hvert projekt, og analysesiden, der længe var blevet behandlet som en separat og mere manuel verden, kom ind i den samme pipeline som udviklingsarbejdet. Siden fortæller om omkring 4.000 sparede timer og om arbejde, der blev leveret mere ensartet, fordi det ikke længere blev gjort en smule forskelligt hver gang.

Virksomhedens egen kode viser den anden ende af at drive tingene selv. Da den flyttede fra en tredjeparts hostingtjeneste, gik den til en forge, virksomheden selv driver, der er én binær fil, SSH-først og uden administrativ webgrænseflade, valgt ud fra princippet om, at jo mindre grænseflade der er, jo mindre er der at forsvare. Den forge deler en vært til offentligt tilgængelige tjenester med 512 MB hukommelse, hvoraf omkring 464 MB kan bruges, med en webserver for syv domæner, en Tor-onion-tjeneste, en Gemini-server, en Gopher-server og blokering af indtrængningsforsøg. Automatisering af sikkerhedskopiering er implementeret separat; implementeringen er noget andet end aktive sikkerhedskopier.

Ingen af de to sider nævner GitLab, og sammenligningen herefter er artiklens egen regning. GitLab angiver 8 GB hukommelse som det mindste, en installation på én node kan køre med; hele den vært har en sekstendedel af det. At kunne drive GitLab selv betyder at have en server dimensioneret til GitLab, og en lille forge med ét formål er den anden vej til at holde koden på egne maskiner.

En minimal pipelinefil viser formen på CI/CD-opsætningen, med to stages, der kører i rækkefølge, og ét job i hver:

```
stages:
    - build
    - test

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

```
        - make build

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

Hvor det går galt

- **En lille server gør hele instansen langsom.** GitLabs dokumentation angiver 16 GB hukommelse som udgangspunkt for en installation på én node og 8 GB som det mindste, den kan køre med i et miljø med begrænset hukommelse. Den advarer også om, at swap kan give et betydeligt fald i ydelsen under belastning, og beder om, at swap slås fra, hvor det er muligt, eller at der er så meget hukommelse, at GitLab aldrig bruger den.
- **En sikkerhedskopi gendannes til en instans, der ikke kan læse sine egne data.** Backupkommandoen udelader konfigurationsmappen, `/etc/gitlab`, og dermed `gitlab-secrets.json`, nøglerne, der dekrypterer krypterede værdier i databasen. Uden den fil forbliver de værdier låst i en gendannet database. Dokumentationen beder om, at konfigurationen sikkerhedskopieres for sig, som det mindste `gitlab-secrets.json` og `gitlab.rb`.
- **En opgradering springer et påkrævet stop over.** Opgraderingsstierne har påkrævede stop, versioner, der skal installeres før enhver senere version, og fra GitLab 17.5 ligger de ved x.2, x.5, x.8 og x.11. Baggrundsmigreringerne for hvert stop skal være færdige, før den næste opgradering begynder, så en instans, der er kommet flere versioner bagud, må indhente dem ét stop ad gangen.
- **Et administreret datalager viser sig at være uden for det understøttede.** PostgreSQL er den eneste understøttede database. Redis eller Valkey rummer sessioner og baggrundsopgaver og skal være en selvstændig instans: Redis Cluster og serverless-varianter af Redis og Valkey understøttes ikke. En administreret cloudtjeneste er tilladt, så længe den er af den selvstændige slags.
- **Pipelines venter på maskiner, som ingen har planlagt.** GitLab.com stiller færdige runners til rådighed, men på en selvdrevet instans skal administratoren levere og drive den infrastruktur, CI/CD-job kører på, og det omfatter at sørge for, at runnerne har kapacitet nok til organisationens arbejdsmængde. Et travlt team med én lille runner står i kø bag den.

Læs mere

- GitLab Docs — hele dokumentationen om at bruge, installere og administrere GitLab.

- Installation requirements — hukommelse, CPU, lager og de understøttede versioner af PostgreSQL og Redis.
- Back up GitLab — hvad en sikkerhedskopi indeholder, hvad den udelader, og hvordan hemmelighederne gemmes ved siden af.
- Upgrade paths — de påkrævede opgraderingsstop, og hvorfor baggrundsmigreringer skal blive færdige imellem dem.
- Get started with GitLab CI/CD — filen `.gitlab-ci.yml`, stages, job og pipelines.
- GitLab Runner — programmet, der kører jobbene, og de måder, det kan køre dem på.

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 GitLab?« 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/gitlab/