Engineer

GitLab

en git-plattform som du också kan köra själv

Vad det är

GitLab är en enda applikation byggd runt git-repon: den lagrar koden och lägger till merge requests för granskning, ärenden för planering och ett inbyggt CI/CD-system som bygger, testar och driftsätter från samma ställe. Det som skiljer GitLab från andra är valet av var det körs. Samma produkt erbjuds som GitLab.com, tjänsten som GitLab själva driver, som GitLab Self-Managed, som du installerar på egna servrar, och som GitLab Dedicated, en instans som bara en kund använder och som GitLab driftar och underhåller åt den kunden i den AWS-region som kunden föredrar.

Därunder finns vanligt git. Varje projekt är ett git-repo som går att klona, pusha till och hämta från med de vanliga kommandona, och historiken i en klon är komplett. Allt som GitLab lägger till ovanpå, från granskningstrådar och ärenden till resultat från pipelines och behörigheter, hålls av GitLab självt, utanför repot, och en klon bär inget av det med sig.

En merge request är GitLabs namn på en föreslagen ändring som väntar på att slås ihop, samma idé som andra plattformar kallar en pull request. Dokumentationen beskriver den som en central plats för att granska kod, föra diskussionen och följa ändringen. En och samma sida visar beskrivningen, kodändringarna med granskningskommentarer direkt i koden, läget för CI/CD-pipelines, rapporterna om huruvida ändringen kan slås ihop och listan över commits. En merge request kan kopplas till ett ärende och stänga det ärendet automatiskt när den slås ihop. Som standard är projektets standardgren, oftast main, skyddad, och bara personer med rollen Maintainer eller högre får slå ihop ändringar in i den.

CI/CD-systemet konfigureras i en fil som heter `.gitlab-ci.yml` i projektets rot. Filen definierar de steg (stages), jobb och skript som tillsammans utgör en pipeline: stegen bestämmer ordningen, t.ex. bygge, sedan test, sedan driftsättning, och jobben är de uppgifter som utförs inom varje steg. Jobben körs av GitLab Runner, ett separat program skrivet i Go som levereras som en enda binärfil. En runner ansluter till en GitLab-instans, väntar på jobb, kör varje jobb på själva maskinen, i en Docker-container eller på en fjärrserver via SSH, och rapporterar tillbaka resultatet.

Använd begreppet i ett beslut

Använd ”Vad är GitLab?” 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 mesta som ett mjukvaruteam gör mellan att koden skrivs och att den levereras kan ske inuti en enda GitLab-instans: koden, granskningen, den pipeline som testar den, ärendet som bad om den och uppgiften om vem som godkände vad. En applikation betyder en uppsättning konton och behörigheter, ett ställe att söka på, och länkar som håller mellan ett ärende, den merge request som besvarade det och den pipeline som kontrollerade den.

Möjligheten att driva det själv är skälet till att många organisationer väljer GitLab över huvud taget. När koden inte får lämna företagets egen infrastruktur, på grund av ett avtal, en tillsynsmyndighet eller helt enkelt en policy, är en värdtjänst utesluten, och GitLab kan installeras inne i det nätverk som måste behålla koden. Dokumentationen behandlar detta som ett fullvärdigt användningsfall: den publicerar referensarkitekturer för installationer dimensionerade för 1 000 till 50 000 användare, vid sidan av Linux-paketet och ett Helm-chart för Kubernetes.

Den kontrollen har ett pris, och det betalas i drift snarare än bara i licensavgifter. Att driva GitLab själv betyder att dimensionera en server för det, hålla dess PostgreSQL-databas och dess lager i Redis eller Valkey friska, ordna maskiner till

runners, ta säkerhetskopior som faktiskt går att återställa och ta sig igenom uppgraderingar i den ordning som projektet kräver. De driftade alternativen lämnar över det arbetet till GitLab, medan Self-Managed låter det stanna hos dig.

Så ser det ut i praktiken

Företagets eget arbete har mött GitLab från båda håll.

Under flera år av dataingenjörsarbete ersatte CI/CD-praxis byggd på GitHub och GitLab, med Bash och Python som skötte automatiseringen under ytan, manuella steg i både dataanalysen och mjukvaruutvecklingen. Sättet som arbetet rörde sig från utveckling till leverans på standardiserades så att det blev likadant varje gång i stället för att uppfinnas på nytt för varje projekt, och analyssidan, som länge hade behandlats som en separat och mer manuell värld, fördes in i samma pipeline som utvecklingsarbetet. Sidan noterar ungefär 4 000 sparade timmar, och arbete som levererades mer konsekvent eftersom det inte längre gjordes på ett lite annorlunda sätt varje gång.

Företagets egen kod visar den andra änden av att driva sitt eget. När koden flyttades från en värdtjänst som drevs av tredje part hamnade den på en egen forge i en enda binärfil, byggd i första hand för SSH och utan administrativt webbgränssnitt, vald enligt principen att ju mindre gränssnitt det finns, desto mindre finns det att försvara. Den forgen delar en värddator för publika tjänster med 512 MB minne, varav ungefär 464 MB går att använda, med en webbserver för sju domäner, en Tor-onion-tjänst, en Gemini-server, en Gopher-server och spärrning av intrångsförsök. Automatisering av säkerhetskopiering har implementerats separat; implementeringen är något annat än aktiva säkerhetskopior.

GitLab nämns på ingen av sidorna, och jämförelsen som följer är den här artikelns egen räkning. GitLab dokumenterar 8 GB minne som det minsta en installation på en enda nod kan köras med, och hela den värden har en sextondel av det. Att kunna driva GitLab själv betyder att ha en server som är dimensionerad för GitLab, och en liten forge med ett enda syfte är det andra sättet att behålla koden på egna maskiner.

En minimal pipelinefil visar formen på CI/CD-konfigurationen, med två steg som körs i ordning och ett jobb i varje:

```
stages:
    - build
    - test

build-site:
    stage: build
    script:
        - make build

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

Var det går fel

- **En liten server gör hela instansen långsam.** GitLabs dokumentation anger 16 GB minne som utgångspunkt för en installation på en enda nod och 8 GB som det minsta den kan köras med i en miljö med begränsat minne. Den varnar också för att swap kan ge en betydande försämring av prestandan under belastning, och ber om att swap stängs av där det går eller att det finns så mycket minne att GitLab aldrig behöver röra den.
- **En säkerhetskopia återställs till en instans som inte kan läsa sina egna data.** Kommandot för säkerhetskopiering utelämnar konfigurationskatalogen, `/etc/gitlab`, och därmed `gitlab-secrets.json`, nycklarna som dekrypterar krypterade värden i databasen. En återställd databas håller de värdena låsta så länge den filen saknas. Dokumentationen ber om att konfigurationen säkerhetskopieras separat, åtminstone `gitlab-secrets.json` och `gitlab.rb`.
- **En uppgradering hoppar över ett obligatoriskt stopp.** Uppgraderingsvägarna innehåller obligatoriska stopp, versioner som måste installeras före alla senare versioner, och sedan GitLab 17.5 ligger de på x.2, x.5, x.8 och x.11. Bakgrundsmigreringarna i varje stopp måste bli klara innan nästa uppgradering startar, så en instans som har hamnat flera versioner efter kommer ikapp ett stopp i taget.
- **Ett hanterat datalager visar sig sakna stöd.** PostgreSQL är den enda databas som stöds. Redis eller Valkey håller sessioner och bakgrundsuppgifter och måste vara en fristående instans: Redis Cluster och serverlösa varianter av Redis och Valkey stöds inte. En hanterad molntjänst är tillåten, så länge den är av det fristående slaget.

- **Pipelines väntar på maskiner som ingen planerade för.** GitLab.com erbjuder färdiga runners, men på en instans som du driver själv är det administratören som tillhandahåller och sköter infrastrukturen där CI/CD-jobben körs, vilket innefattar att se till att runners har tillräcklig kapacitet för organisationens arbetsbörda. Ett upptaget team med en enda liten runner hamnar i kö bakom den.

Läs mer

- GitLab Docs — hela dokumentationen för att använda, installera och administrera GitLab.
- Installation requirements — minne, processor, lagring och de versioner av PostgreSQL och Redis som stöds.
- Back up GitLab — vad en säkerhetskopia innehåller, vad den utelämnar och hur hemligheterna förvaras bredvid den.
- Upgrade paths — de obligatoriska uppgraderingsstoppen och varför bakgrundsmigreringar måste bli klara mellan dem.
- Get started with GitLab CI/CD — filen `.gitlab-ci.yml` , steg, jobb och pipelines.
- GitLab Runner — programmet som kör jobben, och de olika sätt som det kan köra dem på.

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 GitLab?” 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/gitlab/