Engineer

GitLab

Git-Plattform, die sich auch selbst betreiben lässt

Was es ist

GitLab ist eine einzige Anwendung rund um Git-Repositorys: Sie hostet den Code und ergänzt Merge Requests zur Prüfung, Issues zur Planung und ein eingebautes CI/CD-System, das am selben Ort baut, testet und bereitstellt. Ein Unterscheidungsmerkmal ist die Wahl des Betriebsorts. Dasselbe Produkt gibt es als GitLab.com, den von GitLab selbst betriebenen Dienst, als GitLab Self-Managed zur Installation auf eigenen Servern und als GitLab Dedicated, eine von GitLab für einen Kunden gehostete und gepflegte Single-Tenant-Instanz in dessen bevorzugter AWS-Region.

Darunter liegt gewöhnliches Git. Jedes Projekt ist ein Git-Repository, das sich mit den üblichen Befehlen klonen lässt und Pushes und Fetches unterstützt. Die Historie eines Klons ist vollständig. Alles, was GitLab darüber hinaus ergänzt, von Review-Diskussionen und Issues bis zu Pipeline-Ergebnissen und Berechtigungen, speichert GitLab selbst außerhalb des Repositorys. Ein Klon enthält nichts davon.

Ein Merge Request ist GitLabs Bezeichnung für eine vorgeschlagene Änderung, die auf das Zusammenführen wartet. Andere Plattformen nennen dieselbe Idee Pull Request. Die Dokumentation beschreibt ihn als zentralen Ort, um Code zu prüfen, die Diskussion zu führen und die Änderung zu verfolgen. Eine Seite zeigt die Beschreibung, die Codeänderungen mit Review-Kommentaren direkt an den betreffenden Zeilen, den Zustand der CI/CD-Pipelines, die Berichte zur Zusammenführbarkeit und die Liste der Commits. Ein Merge Request lässt sich mit einem Issue verknüpfen und kann es beim Zusammenführen automatisch schließen. Standardmäßig ist der Standardbranch des Projekts, meist main, geschützt. Nur Personen mit der Rolle Maintainer oder höher dürfen Änderungen in ihn zusammenführen.

Das CI/CD-System wird in der Datei `.gitlab-ci.yml` im Projektwurzelverzeichnis konfiguriert. Sie deklariert Stages, Jobs und Skripte, die zusammen eine Pipeline bilden. Stages bestimmen die Reihenfolge, etwa Bauen, dann Testen, dann Bereitstellen. Jobs sind die Aufgaben innerhalb jeder Stage. Ausgeführt werden sie von GitLab Runner, einem gesonderten, in Go geschriebenen und als einzelne Binärdatei ausgelieferten Programm. Ein Runner verbindet sich mit einer GitLab-Instanz, wartet auf Jobs, führt jeden auf dem Rechner selbst, in einem Docker-Container oder über SSH auf einem entfernten Server aus und meldet das Ergebnis zurück.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist GitLab?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Geprüft, veröffentlicht und rückgängig zu machen: Änderungen erreichen Ihre Kunden am Dienstagnachmittag statt spät am Freitagabend.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. GIS-SaaS-Deployments mit GitHub automatisiert — Bereitstellung, Daten und Berichte ohne Handarbeit

GIS-SaaS-Deployments mit GitHub automatisiert Kontakt aufnehmen →

Warum es wichtig ist

Das meiste, was ein Softwareteam zwischen dem Schreiben und Ausliefern von Code tut, kann in einer GitLab-Instanz stattfinden: der Code, das Review, die prüfende Pipeline, das anfordernde Issue und die Aufzeichnung, wer was freigegeben hat. Eine Anwendung bedeutet einen Satz von Konten und Berechtigungen, einen Ort zum Suchen und beständige Verknüpfungen zwischen einem Issue, dem zugehörigen Merge Request und der prüfenden Pipeline.

Der mögliche Eigenbetrieb ist für viele Organisationen überhaupt der Grund, GitLab zu wählen. Darf Code die eigene Infrastruktur wegen eines Vertrags, einer Aufsichtsbehörde oder schlicht einer Richtlinie nicht verlassen, scheidet ein Hostingdienst aus. GitLab lässt sich dann innerhalb des Netzes installieren, in dem der Code bleiben muss. Die Dokumentation behandelt das als vollwertigen Anwendungsfall: Neben dem Linux-Paket und einem Helm-Chart für Kubernetes veröffentlicht sie Referenzarchitekturen für Installationen mit 1.000 bis 50.000 Nutzern.

Diese Kontrolle hat ihren Preis. Er wird im Betrieb bezahlt, nicht nur mit Lizenzgebühren. Wer GitLab selbst betreibt, muss einen Server passend auslegen, die PostgreSQL-Datenbank und den Redis- oder Valkey-Speicher funktionsfähig halten, Rechner für die Runner bereitstellen, tatsächlich wiederherstellbare Sicherungen anlegen und Upgrades in der vom Projekt vorgeschriebenen Reihenfolge durchlaufen. Die gehosteten Angebote übertragen diese Arbeit an GitLab. Beim Eigenbetrieb bleibt sie bei Ihnen.

Wie es in der Praxis aussieht

Die Arbeit des Unternehmens liefert Beispiele für beide Seiten von GitLab.

Über mehrere Jahre im Data Engineering ersetzten CI/CD-Verfahren auf Basis von GitHub und GitLab mit Bash und Python als Grundlage der Automatisierung manuelle Schritte sowohl in der Datenanalyse als auch in der Softwareentwicklung. Der Weg von der Entwicklung zur Auslieferung wurde vereinheitlicht, damit er jedes Mal gleich ablief und nicht je Projekt neu erfunden wurde. Die Analyse, lange als getrennte und stärker manuelle Welt behandelt, kam in dieselbe Pipeline wie die Entwicklungsarbeit. Die Seite verzeichnet etwa 4.000 ersparte Stunden und verlässlicher ausgelieferte Arbeit, weil sie nicht mehr jedes Mal etwas anders erledigt wurde.

Der eigene Code des Unternehmens zeigt das andere Ende des Eigenbetriebs. Beim Umzug von einem fremden Hostingdienst ging er zu einer selbst betriebenen Forge aus einer einzelnen Binärdatei, mit vorrangigem SSH-Zugang und ohne administrative Weboberfläche. Sie wurde nach dem Grundsatz gewählt: Je weniger Oberfläche es gibt, desto weniger muss verteidigt werden. Diese Forge teilt sich einen Host für öffentliche Dienste mit 512 MB Arbeitsspeicher, von denen etwa 464 MB nutzbar sind, mit einem Webserver für sieben Domains, einem Tor-Onion-Dienst, einem Gemini-Server, einem Gopher-Server und der Sperrung von Angreifern. Die Sicherungsautomatisierung ist gesondert implementiert. Ihre Implementierung ist von aktiven Sicherungen zu unterscheiden.

Keine der beiden Seiten erwähnt GitLab. Der folgende Vergleich ist eine eigene Rechnung dieses Artikels. GitLab nennt 8 GB Arbeitsspeicher als Minimum für eine Installation auf einem einzelnen Knoten. Der gesamte Host hat ein Sechzehntel davon. GitLab selbst betreiben zu können bedeutet, einen passend für GitLab ausgelegten Server zu haben. Eine kleine Forge für einen einzigen Zweck ist der andere Weg, Code auf eigenen Rechnern zu halten.

Eine minimale Pipeline-Datei zeigt den Aufbau der CI/CD-Konfiguration: zwei nacheinander laufende Stages mit je einem Job:

```
stages:
    - build
    - test

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

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

Wo es schiefgeht

- **Ein kleiner Server macht die ganze Instanz langsam.** GitLabs Dokumentation nennt 16 GB Arbeitsspeicher als Ausgangspunkt für eine Installation auf einem einzelnen Knoten und 8 GB als Minimum bei eingeschränktem Speicher. Sie warnt außerdem, dass Swap die Leistung unter Last erheblich verschlechtern kann, und fordert, ihn möglichst abzuschalten oder genügend Arbeitsspeicher vorzuhalten, damit GitLab ihn nie verwendet.
- **Eine Sicherung wird in eine Instanz eingespielt, die ihre eigenen Daten nicht lesen kann.** Der Sicherungsbefehl lässt das Konfigurationsverzeichnis `/etc/gitlab` aus und damit `gitlab-secrets.json` : die Schlüssel zum Entschlüsseln verschlüsselter Datenbankwerte. Ohne diese Datei bleiben die Werte einer wiederhergestellten Datenbank gesperrt. Die Dokumentation fordert eine gesonderte Sicherung der Konfiguration, mindestens von `gitlab-secrets.json` und `gitlab.rb` .
- **Ein Upgrade überspringt einen vorgeschriebenen Zwischenstand.** Upgrade-Pfade enthalten Pflichtstopps: Versionen, die vor jeder späteren installiert werden müssen. Seit GitLab 17.5 liegen sie bei x.2, x.5, x.8 und x.11. Die Hintergrundmigrationen jedes Zwischenstands müssen fertig sein, bevor das nächste Upgrade beginnt. Eine mehrere Versionen zurückliegende Instanz holt daher einen Pflichtstopp nach dem anderen auf.
- **Ein verwalteter Datenspeicher wird nicht unterstützt.** PostgreSQL ist die einzige unterstützte Datenbank. Redis oder Valkey hält Sitzungen und Hintergrundaufgaben und muss als eigenständige Instanz laufen. Redis Cluster und serverlose Varianten von Redis und Valkey werden nicht unterstützt. Ein verwalteter Clouddienst ist zulässig, sofern er zu den eigenständigen Varianten gehört.

- **Pipelines warten auf Rechner, die niemand eingeplant hat.** GitLab.com bietet fertige Runner an. Bei einer selbst verwalteten Instanz stellt und verwaltet der Administrator dagegen die Infrastruktur für die CI/CD-Jobs. Dazu gehört, den Runnern genügend Kapazität für die Arbeitslast der Organisation zu geben. Ein ausgelastetes Team mit einem kleinen Runner steht dahinter Schlange.

Weiterführende Lektüre

- GitLab-Dokumentation — die gesamte Dokumentation zur Nutzung, Installation und Administration von GitLab.
- Installationsanforderungen — Arbeitsspeicher, CPU, Speicherplatz und unterstützte PostgreSQL- und Redis-Versionen.
- GitLab sichern — was eine Sicherung enthält, was sie auslässt und wie die Geheimnisse daneben aufzubewahren sind.
- Upgrade-Pfade — die vorgeschriebenen Zwischenstände und warum die Hintergrundmigrationen dazwischen fertig werden müssen.
- Einstieg in GitLab CI/CD — die Datei `.gitlab-ci.yml` , Stages, Jobs und Pipelines.
- GitLab Runner — das Programm zur Ausführung der Jobs und seine Ausführungsarten.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist GitLab?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/gitlab/