Engineer

Forge

Plattform zum Hosten, Besprechen und Teilen von Code

Was es ist

Eine Forge, auch Software-Forge, Code-Forge oder Git-Forge genannt, ist der Server, den ein Team als Heimat seiner Repositories festlegt, samt den Diensten, die es darum herum betreibt. Jeder vollständige Klon eines Git-Repositorys enthält die gesamte Historie. Grundsätzlich ist daher keine Kopie einer anderen übergeordnet. Praktisch braucht ein Team einen Ort für Pushes, einen Ort, der über Lese- und Schreibrechte entscheidet, und einen Ort für die Gespräche über den Code. Dieser Ort ist die Forge.

Im Kern speichert eine Forge Bare-Repositories, die reine Historie ohne ausgecheckte Dateien, und spricht eines oder mehrere der Übertragungsprotokolle von Git. Das Git-Buch nennt vier: das lokale Protokoll für ein Verzeichnis auf demselben Rechner, HTTP, SSH und das eigene Git-Protokoll mit einem Daemon auf Port 9418, ganz ohne Authentifizierung. Alles Weitere baut darauf auf, in sehr unterschiedlichem Umfang.

Forgejos Dokumentation beschreibt das umfangreiche Ende des Spektrums: Issue-Verwaltung, Pull Requests, ein integriertes Wiki, eine Paketregistry für viele Sprachökosysteme, Forgejo Actions zum Ausführen von Jobs, Repository-Spiegel, Migration aus anderen Diensten, Branch- und Tag-Schutz sowie Webhooks. Soft Serve steht am anderen Ende. Es beschreibt sich als „selbst hostbarer Git-Server für die Kommandozeile“: eine einzelne Binärdatei namens `soft`, die Repositories über SSH, HTTP oder das Git-Protokoll ausliefert, den Zugriff mit öffentlichen SSH-Schlüsseln steuert und Mitwirkende, private Repositories und Webhooks unterstützt. Beides sind Forges. Sie unterscheiden sich darin, wie viel Teamarbeit sie aufnehmen wollen.

Forges beginnen auch, miteinander zu sprechen. ForgeFed ist ein auf ActivityPub aufbauendes Föderationsprotokoll für Plattformen zum Hosten und gemeinsamen Bearbeiten von Softwareprojekten. Es modelliert Repositories, Ticket- und Patch-Tracker, Projekte und Teams als Akteure, die einander Aktivitäten senden. Ziel ist eine gemeinsame Sprache für Netzwerke aus Forges und Projektmanagementwerkzeugen. Eine Person auf einem Server soll so ein Ticket für ein Repository auf einem anderen eröffnen können. Ein Repository soll zwischen Servern umziehen können, während die alte Kopie schreibgeschützt archiviert bleibt und auf die neue Heimat verweist.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine Forge?“ 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

Auf der Forge werden Regeln durchsetzbar. Eine Regel auf einem Entwicklungsrechner gilt nur, solange dieser richtig eingerichtet ist. Ein Hook auf dem Server läuft dagegen bei jedem Push. Gits Hook `pre-receive` liest jede übertragene Referenz als alte ID, neue ID und Namen von der Standardeingabe. Endet er mit einem Status ungleich null, wird keine der Referenzen aktualisiert. Branch-Schutz, verpflichtende Reviews und Prüfungen signierter Commits auf größeren Forges verfeinern dieselbe Idee: Die gemeinsame Kopie lehnt ab, was das Team vereinbart hat, nicht anzunehmen.

Die Wahl einer Forge entscheidet auch, wer den Code hält. Ein gehosteter Dienst übernimmt Betriebskosten und Verfügbarkeit. Dafür liegen Code, Zugriffsliste und sämtliche Diskussionen auf fremder Hardware unter fremden Bedingungen. Eigenbetrieb kehrt den Tausch um. Keine Wahl ist kostenlos. Der Unterschied zählt besonders für ein Unternehmen, das Kunden sagt, wo ihre Daten liegen.

Eine Forge hält außerdem Daten, die Git nicht enthält. Commits und Tags reisen mit jedem Klon. Issues, Review-Diskussionen, Zugriffslisten und serverseitige Hooks gehören dagegen allein zur Forge. Diese Trennung bestimmt, was eine

Sicherung abdecken und ein Umzug zwischen Forges mitnehmen muss.

Wie es in der Praxis aussieht

Dieses Unternehmen betreibt eine eigene Forge. Drei Projekte auf dieser Website zeigen, was dafür nötig war.

Der Quellcode des Unternehmens lag früher bei einem fremden Hostingdienst. Das Projekt zur Bereitstellung einer eigenen Git-Forge beginnt mit der Feststellung, dass eine Forge zu betreiben heißt, eine Forge zu betreiben: Authentifizierung, Zugriffskontrolle, Speicher, Sicherungen und eine öffentliche Anlaufstelle dafür. Die Forge ist Soft Serve, eine einzelne vom Betriebssystem überwachte Binärdatei. Ausschlaggebend waren der Vorrang von SSH und der Verzicht auf eine administrative Weboberfläche. Standardmäßig ist sie privat: Anonymer Zugriff wird abgelehnt, ebenso Zugriff ohne Schlüssel, und jedes deklarierte Repository ist als privat markiert. Ihr SSH-Listener bindet nur an die Loopback-Schnittstelle auf Port 23231 und ist von außen über einen Sprungserver erreichbar. Ein Multiplexer, der HTTPS und Git über SSH auf einem öffentlichen Port zusammenführen würde, ist geschrieben, bleibt aber ausgeschaltet. Ein ungenutzter Listener ist Angriffsfläche. Das Ergebnis wird klar benannt: Das Unternehmen hostet seinen Code auf eigener Hardware.

Auch das Repository dieser Website liegt dort. Seine Anweisungen nennen die selbst gehostete Forge, Soft Serve über SSH, als kanonischen Remote. Sie bezeichnen die ältere gehostete Kopie als Legacy-Spiegel und halten fest, dass es keine gehostete CI gibt: Die verbindliche Qualitätsprüfung läuft lokal und in Git-Hooks.

Ein serverseitiger Hook muss in der Empfangsumgebung eines echten Git-Pushs getestet werden. Daher wurde er an einem echten Bare-Repository und einem echten Push erprobt, bevor man ihm vertraute. Der Probelauf erzeugt ein Bare-Repository, installiert den Hook, klont, committet, pusht und prüft sowohl Annahme als auch Ablehnung. Er fand zwei Defekte, die beide in der Datei nicht sichtbar waren. Der zweite ist besonders lehrreich: Git exportiert beim Ausführen eines Hooks eine Variable mit dem Repository-Verzeichnis. Die vom Hook gestartete Testsuite erbte sie. Neunzehn Tests wurden bei korrektem Code rot, bis die Umgebung an der Grenze bereinigt wurde. Die Seite nennt auch die Grenze des Probelaufs: Er erzeugt keine erzwungenen Aktualisierungen und keine Branch-Löschungen, die ebenfalls Pushes sind.

All das teilt sich einen Einkern-Host mit 512 MB Arbeitsspeicher mit allen anderen öffentlich angebotenen Diensten des Unternehmens, von sieben TLS-Domains über den Onion-Dienst bis zu Gemini- und Gopher-Servern sowie Angreifersperren. Die Sicherungsautomatisierung ist separat implementiert; das ist von aktiven Sicherungen zu unterscheiden. Eine Forge mit Webpanel wäre auf diesem Budget eine weitere dauerhaft laufende Anwendung gewesen. Deshalb wurden solche Plattformen bereits beim Entwurf ausgeschlossen. Der niedrigste Wert des verfügbaren Arbeitsspeichers lag auf diesem Host in einer dokumentierten eintägigen Messreihe bei etwa 310 MB von 464 MB.

Wo es schiefgeht

Auf manchen Forges erhalten anonyme Besucher standardmäßig Lesezugriff. Soft Serves Dokumentation beschreibt die Voreinstellung als schreibgeschützten Zugriff für anonyme Verbindungen, gesteuert durch `anon-access` und `allow-keyless`. Eine private Forge setzt Ersteres auf `no-access` und Letzteres auf false. Ein Test von einem Rechner ohne geladenen Schlüssel ist sinnvoll. Gits eigener Daemon ist noch gröber: Laut Git-Buch ist ein ausgeliefertes Repository entweder für alle klonbar oder wird überhaupt nicht ausgeliefert.

Ein Hook, der Git in einem anderen Verzeichnis aufruft, bearbeitet das falsche Repository. Bei einem Push führt Git die empfangenden Hooks im Bare-Repository aus und exportiert `GIT_DIR` und verwandte Variablen, damit Git-Befehle im Hook es finden. Alles, was der Hook startet, erbt sie, auch ein Testlauf in einem Checkout an anderer Stelle. Das githooks-Handbuch weist darauf hin, diese Variablen vor der Arbeit in einem fremden Repository zu entfernen. Ein minimaler Hook `pre-receive`, der jeden übertragenen Commit in einem Wegwerfverzeichnis prüft, tut das in einer 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
```

Eine wiederhergestellte Forge hat ihren Code zurück, aber ihre Regeln nicht. Jeder Klon kann die Repositories eines verlorenen Servers neu befüllen. Das Git-Buch warnt jedoch, dass serverseitige Hooks und ähnliche Einstellungen zurückbleiben. Issues, Reviews, Schlüssel und Zugriffslisten waren nie in den Klonen. Eine Forge zu sichern heißt, ihr Datenverzeichnis und ihre Konfiguration zu sichern, beides irgendwo wiederherzustellen und zu prüfen, ob ein vom Hook abzulehnender Push tatsächlich abgelehnt wird.

Jeder von einer Forge geöffnete Listener muss verteidigt werden. Die Beispielkonfiguration von Soft Serve hat SSH auf Port 23231, HTTP auf 23232, den Git-Daemon auf 9418 und einen Statistikserver auf 23233. Ein Team, das nur über SSH pusht, kann den Rest abschalten oder an Loopback binden. Offen bleibt dann nur, was tatsächlich genutzt wird.

Ein Umzug zu einer anderen Forge nimmt den Code mit und lässt die Gespräche zurück. Ein Mirror-Push kopiert jede Referenz unter `refs/`, einschließlich Branches und Tags, und sonst nichts. Issues, Review-Diskussionen, Labels und Zugriffslisten brauchen das Migrationswerkzeug der neuen Forge oder ein Protokoll wie ForgeFed. Dieses kopiert Commits, Tags, Tickets, Merge Requests und Kommentare und hinterlässt das alte Repository archiviert mit einem Verweis auf das neue.

Weiterführende Lektüre

- ForgeFed-Spezifikation — das Föderationsprotokoll für Forges mit seinen Akteuren, Aktivitäten und dem Umzugsablauf, der ein altes Repository archiviert.
- Soft Serve — die eigene Dokumentation des Projekts zu einem SSH-orientierten Git-Server aus einer Binärdatei, einschließlich `anon-access` und `allow-keyless`.
- Forgejo-Dokumentation — Anleitungen für Administration und Nutzung einer voll ausgestatteten selbst gehosteten Forge, von Installation und Sicherungen bis zu Actions und Spiegeln.
- githooks — jeder von Git ausgeführte Hook, einschließlich `pre-receive` und der von einem Push-Hook geerbten Umgebung.
- Die Protokolle — die vier Wege der Datenübertragung zwischen Rechnern mit Git und ihr Aufwand für Einrichtung und Sicherheit.
- ActivityPub — die W3C-Empfehlung, die ForgeFed erweitert.

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 eine Forge?“ 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/forge/