Engineer

フォージ

コードのリポジトリーと共同作業のための共有サーバー

どのようなものか

フォージは、ソフトウェアフォージ、コードフォージ、Gitフォージとも呼ばれる、チームがリポジトリーの拠点と決めるサーバーと、その周りのサービスです。Gitの完全なクローンには、全履歴があるため、原理上、どの複製も他より上位ではありません。実際のチームには、プッシュする場所、読み書きできる人を決める場所、コードについて話す場所が必要です。それが、フォージです。

フォージの内部には、チェックアウトしたファイルを持たず、履歴だけを保持するベアリポジトリーを置きます。Gitの転送プロトコルの、一つ以上に対応します。Gitの本では、同じマシンのディレクトリーを使うローカルプロトコル、HTTP、SSH、そしてポート9418のデーモンで動き、認証を一切行わないGit独自のプロトコルの四つを挙げています。それ以外の機能はこの土台の上に追加され、どこまで備えるかはフォージによって大きく異なります。

Forgejoの文書は、機能の多い側を示します。課題管理、プルリクエスト、統合Wiki、多くの言語のエコシステム向けパッケージレジストリー、ジョブを実行するForgejo Actions、リポジトリーのミラー、他サービスからの移行、ブランチとタグの保護、ウェブフック です。Soft Serveは、反対側です。自らを「自前でホストできるコマンドライン向けのGitサーバー」と説明します。`soft` という単一のバイナリーが、SSH、HTTP、Gitプロトコルでリポジトリーを配信し、SSH公開鍵で権限を管理し、共同開発者、非公開リポジトリー、ウェブフックに対応します。どちらもフォージで、チームの仕事をどこまで引き受けるかが違います。

相互に通信する動きもあります。ActivityPubを拡張したForgeFedは、ソフトウェアプロジェクトのホスティングと共同作業のプラットフォームを、連合させるプロトコルです。リポジトリー、チケットやパッチの管理ツール、プロジェクト、チームを、活動を送り合うアクターとして表します。フォージとプロジェク

ト管理ツールのネットワークに、共通の言葉を作るのが目的です。あるサーバーの人が、別のサーバーのリポジトリーへチケットを出せ、リポジトリーも移動できます。古い複製は、読み取り専用で保存し、新しい拠点を示します。

概念を判断に生かす

「フォージとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 変更を検証し、リリースし、元に戻せる仕組みを整え、金曜の深夜ではなく火曜の午後にお客様へ届けられるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GitHub ActionsでGIS SaaSのデプロイを自動化 — デプロイ、データ、報告を手作業から解放

GitHub ActionsでGIS SaaSのデプロイを自動化 お問い合わせ →

なぜ重要なのか

フォージは、方針を強制できる場所です。開発者のマシンの規則は、その設定が正しい間しか効きませんが、サーバーのフックは、全プッシュで動きます。Gitの `pre-receive` は、各参照の古いID、新しいID、名前を、標準入力から読みます。ゼロ以外で終了すれば、どの参照も更新しません。大きいフォージのブランチ保護、必須レビュー、署名付きコミットの検査も、同じ考えを細かくしたものです。チームが受け入れないと決めたものを、共有の複製が拒みます。

フォージを選ぶことは、誰がコードを保持するかを決めることでもあります。ホスティングサービスは、運用費と稼働の維持を引き受ける代わりに、コード、権限一覧、議論が、他人のハードウェアと条件の下に置かれます。自前でホストする場合は、コードを自社の管理下に置く代わりに、運用の費用と責任を自社で負います。どちらも無料ではなく、顧客へデータの置き場所を説明する会社には、特に重要です。

Gitにないデータも持ちます。コミットとタグは、全クローンと一緒に移りますが、課題、レビューのスレッド、権限一覧、サーバー側フックは、フォージだけにあります。その違いが、バックアップの対象と、移行で運ぶものを決めます。

実際の活用例

この会社も、自社のフォージを運用します。このサイトの三つのプロジェクトが、必要な仕事を示します。

以前は、会社のソースコードを第三者のサービスへ置いていました。会社のGitフォージを配備するプロジェクトは、フォージを運用するとは、認証、権限管理、ストレージ、バックアップ、公開の窓口まで運用することだ、という認識から始まります。Soft Serveという、OSが稼働を管理する一つのバイナリーを使います。SSHを第一とし、管理用ウェブ画面を持たないため、選びました。既定で非公開で、匿名も鍵なしも拒み、宣言した全リポジトリーを非公開にします。SSHは、ポート23231のループバックだけへバインドし、外からは踏み台を通ります。HTTPSとSSH上のGitを、一つの公開ポートへまとめるマルチプレクサーも実装しましたが、無効のままです。使わない待ち受けも、攻撃面になるためです。結果は、会社のコードを会社のハードウェアでホストすることだと、明記します。

このサイトのリポジトリーも、そこにあります。指示書は、SSH経由のSoft Serveを、正本のリモートとし、以前のサービスの複製を、旧来のミラーとします。ホストされたCIはなく、品質ゲートは、ローカルとGitフックで動くとも明記します。

サーバー側フックは、実際のGit pushを受信する環境でテストする必要があります。そのため、信頼する前に、本物のベアリポジトリーとプッシュで予行しました。ベアリポジトリーを作り、フックを入れ、クローン、コミット、プッシュを行い、受け入れと拒否の両方を検査します。ファイルでは見えない二つの欠陥が見つかりました。二つ目が、教訓になります。Gitがフックを動かす際、リポジトリーのディレクトリーを示す変数をエクスポートし、そこから起動したテストが引き継ぎました。処理の境界でこれらの環境変数を除去するまで、正しいコードでも十九件のテストが失敗しました。限界も明記します。強制更新とブランチ削除というプッシュは、この予行では作りません。

これらはすべて、会社が公開するほかのサービスとともに、512 MBの単一コアのホストで稼働します。七つのTLSドメイン、onionサービス、Gemini、Gopher、不正なアクセス元の遮断などです。バックアップの自動化は別に実装されています。ただし、自動化の実装と、バックアップが実際に稼働していることは別です。ウェブ画面を持つフォージなら、この予算で動く常駐アプリが、もう一つ増えます。そのため、設計時点で除外しました。記録した一日分のサンプルでは、このホストの使用可能メモリーの最小値は、464 MB中、約310 MBでした。

問題が起きるところ

一部のフォージは、匿名の訪問者へ、既定で読み取り権限を与えます。Soft Serveの文書も、初期状態では、 `anon-access` と `allow-keyless` で管理する、匿名の読み取り専用接続を許すとします。非公開なら、最初を `no-access` 、二つ目をfalseにし、鍵を読み込んでいないマシンから試す価値があります。Git自身のデーモンは、さらに単純です。Gitの本は、全員がクローンできるか、まったく配信しないかの、どちらかだと述べます。

別のディレクトリーでGitを呼ぶフックが、違うリポジトリーを操作します。プッシュ中、受信フックはベアリポジトリー内で動き、Gitは `GIT_DIR` などを渡して、フック内のGitコマンドが、そのリポジトリーを見つけられるようにします。フックが起動するすべての処理に、これらの環境変数が引き継がれます。別の場所にチェックアウトしたコードのテストも例外ではありません。githooksのマニュアルは、別のリポジトリーを使う前に、これらの変数を除去するよう述べます。プッシュされた各コミットを、使い捨てのディレクトリーで試す、最小限の `pre-receive` フックでは、サブシェルの中で行います。

```
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
```

復元したフォージに、コードは戻っても、規則が戻りません。どのクローンでも、失ったサーバーのリポジトリーを埋め直せます。しかし、Gitの本は、サーバー側フックなどの設定が元の側に残り、課題、レビュー、鍵、権限一覧は、最初からクローンにないと警告します。バックアップとは、データディレクトリーと設定を保存し、どこかへ両方を復元し、拒むべきプッシュを、実際に拒むか確かめることです。

開く待ち受けごとに、防御が必要です。Soft Serveの設定例には、SSHが23231、HTTPが23232、Gitデーモンが9418、統計サーバーが23233とあります。SSHでしかプッシュしないなら、ほかを止めるか、ループバックへ限定し、使う分だけを開けておけます。

別のフォージへ移ると、コードは運べても、議論が残ります。ミラープッシュがコピーするのは、ブランチとタグを含む、`refs/` 以下の全参照だけです。課題、レビューのスレッド、ラベル、権限一覧には、新しいフォージの移行ツールか、ForgeFedなどが必要です。ForgeFedはコミット、タグ、チケット、マージリクエスト、コメントを運び、古いリポジトリーをアーカイブして、新しい場所を示します。

参考資料

- ForgeFed仕様 — アクター、活動、古いリポジトリーを保存する移行の流れを含む、フォージ間の連合プロトコルです。
- Soft Serve — `anon-access` と `allow-keyless` の設定も含めて、SSHを重視した単一バイナリーのGitサーバーを説明する、プロジェクト自身の文書です。
- Forgejoの文書 — インストールとバックアップからActionsとミラーまで、機能のそろった自前のフォージの、管理者と利用者向けの案内です。
- githooks — `pre-receive` と、プッシュのフックが引き継ぐ環境を含む、Gitが実行する全フックです。
- プロトコル — Gitがマシン間でデータを移す四つの方法と、設定とセキュリティの負担です。
- ActivityPub — ForgeFedが拡張する、W3Cの勧告です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「フォージとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/forge/