Engineer

Gitフック

⚓ Gitの所定の処理時点で自動実行されるプログラム

どのようなものか

Gitフックは、コミットを記録する前、メッセージを書いた後、プッシュがマシンを出る前、サーバーが受け取るときなど、所定の時点でGit自身が起動するプログラムです。ファイル名は `pre-commit` 、 `commit-msg` 、 `pre-push` など、その時点を表し、拡張子はありません。既定ではリポジトリー内の `.git/hooks` を探し、 `core.hooksPath` で別の場所を指定できます。実行ビットがなければ無視されるため、実行可能である必要がありますが、マシンで動く言語なら何で書いてもかまいません。

クライアント側フックは開発者の手元のクローンで動き、コミットやマージなどで実行されます。サーバー側は、フォージなど、プッシュを受け取るマシンで、送られたコミットの到着時に動きます。

フックを通知用の仕組みから、処理を止められる関門へ変えるのが終了ステータスです。 `pre-commit` は引数を取らず、ゼロ以外で終わるとコミットを中止します。 `commit-msg` は提案されたメッセージのファイルパスを受け取り、拒否や編集ができます。 `pre-push` はリモートの名前と場所を引数に、参照ごとの一行を標準入力に受け、ゼロ以外なら何も送る前にプッシュを止めます。サーバーの `pre-receive` はプッシュごとに一回動き、各参照の古いオブジェクト名、新しいオブジェクト名、参照名を読み、ゼロ以外なら、どの参照も更新しません。 `post-commit` のように後で動くものは通知用で、結果を変えられません。

`git init` がリポジトリーを作成すると、フック用ディレクトリーに、名前が `.sample` で終わるサンプルスクリプトを配置します。接尾辞を外して名前を変えたものだけをGitが実行します。

概念を判断に生かす

「Gitフックとは？」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

覚えておく必要のある検査を、負担の最も小さいタイミングで自動的に行えます。コミット時のコード整形や機密情報の検出なら、作者が変更の内容をまだ覚えているうちに、問題を見つけられます。数日後のレビューや、共有後のCI実行まで待つ必要がありません。

フックを使えば、誰がコミットするかに左右されず、リポジトリー自体に規則を組み込めます。貢献手順書の決まりは読んでもらえる間しか効きませんが、`commit-msg` に入れれば、どのコミットにも適用されます。

ただし、保証できる範囲は二種類で異なります。Pro Gitは、リポジトリーをクローンしてもクライアント側フックはコピーされないと指摘し、方針をサーバー側で強制するよう勧めます。`pre-receive` などは、手元の導入状況にかかわらず、誰のプッシュにも適用されます。代償は時間です。毎回のコミットやプッシュで動くため、遅い検査は回避を招きます。

実際の活用例

このサイトのリポジトリーは、pre-commitフレームワークを使い、`.pre-commit-config.yaml` で設定し、`task setup` が `pre-commit install --install-hooks` を実行して導入します。三つのGitフックを設定します。pre-commit段階では、Prettier、markdownlint、stylelint、yamllint、codespell、Ruffのlintと整形、gitleaksによる機密情報の検出、絵文字検査、語調検査を実行します。commit-msg段階では、規約に沿った、絵文字のない形式を強制するスクリプトを一つ動かします。pre-push段階では、全体の品質ゲート `task check` を動かします。秒でなく分単位ですが、プッシュは少ないため許容するとコメントにあります。設定ファイルの冒頭は `--no-verify` で回避しないよう明記し、このフックのコメントも、待てないプッシュは準備できていないと、その方針を守ります。

コミットメッセージのフックは、四つのリポジトリーを一つの履歴基準へそろえる役割も持ちます。規約に沿った接頭辞とスコープ、上限未満の件名、絵文字なし、帰属を示す末尾行なしという規則です。仕事を置くすべてのリポジトリーで違反を拒否するため、基準は人でなくリポジトリーに属します。

自社フォージのサーバー側フックを、本番のプッシュを受ける前に予行演習で検証しました。作業ツリーのないベアリポジトリーで、フォージの設定した環境の下、標準入力から参照を読むため、ノートPCで手動実行しても証明できることはわずかです。予行演習ではベアリポジトリーを作り、フックを導入し、複製、コミット、プッシュを行い、受理と拒否の両方を確かめます。ファイルを読むだけでは見えなかった二つの欠陥を見つけました。どちらも後で説明します。

あるプロジェクトでは、文書には同じゲートを動かすとあったのに、コミットフックと品質ゲートの検査が異なりました。今は、両方の設定をたどって実際に呼ぶ検査の集合を特定し、差があれば失敗するテストがあります。ゲートだけにあった五つはCVの文章を読む検査で、コードは毎回守っても、文章は人が覚えているときだけでした。

クライアント側フックにフレームワークは不要です。次を `.git/hooks/commit-msg` へ保存し、実行可能にすると、件名が規約に沿った種別で始まらないコミットを拒否します。

```
#!/bin/sh
# Gitは、提案されたコミットメッセージのファイルパスを$1で渡す
pattern='^(feat|fix|docs|chore)(\([a-z0-9-]+\))?: .+'
if head -n 1 "$1" | grep -Eq "$pattern"; then
    exit 0
fi
echo "commit-msg: 'fix(nav): ...'のように、件名を種別で始めてください" >&2
exit 1
```

問題が起きるところ

- **新しい複製にはフックがありません。** それぞれで導入する必要があり、このサイトでは `task setup` が担います。 `Taskfile.yml` のコメントは、静かな失敗も説明します。マシンに `pre-commit` がなければ、 `task setup` は何も導入せず、どのフックも、気付かれないまま実行されなくなります。検査に合格した状態と区別が付きません。

- `--no-verify` **が素通りさせます。** `git commit --no-verify` は `pre-commit` と `commit-msg` を、`git push --no-verify` は `pre-push` を回避します。`core.hooksPath` を `/dev/null` にすれば、すべてを一度に止められます。必ず守る規則は、サーバー側にも置いてください。
- **Gitの環境が別のリポジトリーまで付いてきます。** Gitは `GIT_DIR` などを環境変数として渡し、フック内のコマンドが正しいリポジトリーを見つけられるようにします。githooks(5)は、別のリポジトリーで作業するときには、これらの環境変数を解除するよう勧めます。フォージの予行演習で、まさにこの問題が起きました。ゲートが送られた版をチェックアウトしてテストを動かすと、ベアリポジトリーへのポインターを引き継ぎ、正しいコードで十九のテストが失敗しました。
- **標準入力が例外的なケースを運びます。** もう一つは、対象の参照がないと、Gitが解決できない仮の値へ置き換える処理でした。標準入力から来るため、ずっと先で無関係に見えるエラーになりました。事例は、強制更新やブランチ削除もフォージに来るのに、予行演習では再現していないとも記します。githooks(5)では、削除する参照は `pre-push` へ `(delete)` と、全桁ゼロのオブジェクト名で届きます。
- **遅いフックは回避されます。** 五つの文章検査を入れるには、二つをpre-commitフックの許容実行時間に収める高速化が必要でした。内容が増えると実行時間も増え、遅すぎれば回避される点があると、事例も明記します。
- **形式は通っても、意味を検査しません。** 四つのリポジトリーの事例が述べるように、調べるのは形式で、内容ではありません。何も説明しない件名でも、形式さえ合えば、変更を説明する件名と同じように通ります。

参考資料

- githooks文書 — 対応する全フック、実行時点、入力、終了ステータスが決めるものです。
- Pro Git：Gitフック — クライアント側とサーバー側、導入方法、複製に付かない理由です。
- Pro Git：Gitによる方針の強制例 — メッセージ形式とアクセス規則を強制するサーバー側 `update` と、送る前に警告するクライアント側フックです。
- git config — `.git/hooks` の外へ移したり、すべて無効にしたりする `core.hooksPath` 設定です。
- git commit — `--no-verify` と、回避する二つのフックです。
- git push — `pre-push` と、それを回避する `--no-verify` 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

「Gitフックとは？」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/git-hook/