Engineer

GitHub

Gitリポジトリーを預かるMicrosoftのサービス

どのようなものか

GitHubはMicrosoft傘下のホスティングサービスです。Gitリポジトリーを自らのサーバーへ置き、コードと履歴のウェブ表示、変更を提案してレビューするプルリクエスト、作業を追跡するための課題管理、リポジトリーの出来事に応じて処理を自動実行するGitHub Actionsを提供します。

土台は普通のGitです。GitHubから複製したものは全履歴を持つ通常のリポジトリーで、ノートPCとの間は通常のプッシュと取得でやり取りします。加わるものはGitHub側にあります。レビューの議論、課題、ブランチ規則、自動化設定、誰に何を許すかの一覧は、複製の中でなく、GitHub自身のシステムへ保存します。

GitHubでいうプルリクエストは、一ブランチの変更を別のものへマージする提案です。行ごとの差分、コメントと承認、提案コードに対して実行した全自動検査の結果を示します。下書きなら未完成と表示し、レビュー可能と指定するまでマージできません。

GitHub Actionsは組み込みの自動化です。ワークフローは、リポジトリーの `.github/workflows` ディレクトリーに置くYAMLファイルで、プッシュ、新しいプルリクエスト、時刻指定など、起動する出来事と、ステップからなるジョブを定義します。ステップはシェルコマンドか、リポジトリー間で共有できるようパッケージ化された自動処理であるアクションを実行します。ジョブを動かすのはランナーです。GitHubが毎回新しく起動するUbuntu Linux、Windows、macOSの仮想マシンか、自前で登録したセルフホストのマシンを使います。

入ってくるものを監視するサービスが二つあります。Dependabotは `dependabot.yml` を読み、依存関係を更新するプルリクエストと、既知の脆弱性のある依存関係を直す別のプルリクエストを作ります。プッシュ保護はアクセストークンなどの機密情報を検査し、リポジトリーへ入る前に止めます。

概念を判断に生かす

「GitHub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

多くのチームにとって、保存場所だけでなく仕事の場所です。コード、変更の議論、承認者の記録、各検査の結果が並び、一行がなぜあるかを、一ページから調べられることが多いのです。

自動化もホスティングと同じくらい重要です。Actionsがコードの隣にあると、テストを実行して合格した、ビルドが成功した、スキャナーが検出結果を報告したという証拠を提案へ添えられます。保護ブランチは、それを規則にします。プルリクエスト経由だけ、一定数の承認後だけ、指定したステータスチェックの合格後だけに変更を許し、レビューと検査を通る変更だけで主系統を進められます。

コードの入口にあるため、供給網の管理点でもあります。どの第三者アクションをどの正確な版で動かすか、依存関係をどう更新するか、機密情報を送信時に止めるかを決めます。多くはリポジトリーや組織ごとに有効化、調整するため、実際の安全性の大きな部分は、コードレビューでは見えない設定にあります。

これがホスティングサービスの交換条件です。サーバーを動かさずに、一式がそろい保守も行われる環境を得られますが、保護規則は他者の画面で設定し、既定値は自分の事情でなく一般向けに選ばれています。

実際の活用例

自社の仕事の三つの事例が、GitHubのそれぞれ違う役割を示します。

GitHub Actionsによるデリバリーパイプラインは、コード品質ゲート用を一つ、テスト用を一つ、デプロイ用を環境ごとに一つと、ワークフローを分け、CodeQL、依存関係レビュー、ソフトウェア部品表の処理も並行して動かします。コミットからAzureで動くサービスまでをつなぎ、ビルドの昇格を手作業で

行う代わりに、定義した経路で各環境へ進めます。フロントエンドはシェルもパッケージマネージャーもないdistrolessイメージ、Go APIは軽量なAlpineベースで配信します。

継続的なセキュリティ検査は、コードの周囲の管理を使います。第三者のGitHub Actionはすべてタグでなくコミットハッシュへ固定し、タグの移動で実行内容が変わらないようにします。Dependabotは6エコシステムを監視し、Gitleaksは履歴から認証情報を探し、各ビルドはAnchoreとSyftで部品表を作ります。事例は報告量という代償も挙げます。すべてを報告する検査器は無視を招くため、有用な検出結果を保つには、継続的な仕分けが必要です。

GISのSaaSアプリのデプロイでは、テスト、ビルド、デプロイをGitHub Actionsへ移し、担当者が手順を覚えていることに依存しなくしました。定期的なデータ処理とレポートも、Python、Bash、SQLの定時処理へ移し、設定と機密情報を標準化して各環境の動作をそろえました。

ワークフローでの固定は次の形です。長い十六進数はレビューしたリリースの完全なコミット識別子で、コメントには人が読める版を残します。

```
steps:
    - uses: actions/checkout@3d3c42e5aac5ba805825da76410c181273ba90b1 # v7.0.1
```

問題が起きるところ

- **タグへ固定したアクションが、知らせず変わります。** `v4` などのタグは、アクションのリポジトリーを管理する人が移動や削除でき、明日動くコードが今日レビューしたものとは違い得ます。GitHubは完全なコミットSHAへの固定を勧めます。偽装には「有効なGitオブジェクトの内容に対するSHA-1衝突」を作る必要があるためです。
- **トークンが仕事以上の権限を持ちます。** 毎回渡す `GITHUB_TOKEN` の権限で、侵害されたステップが変えられる範囲も決まります。GitHubは、リポジトリー内容の読み取りを既定とし、必要なジョブだけへ書き込みを許すよう勧めます。
- **信頼できない文章がコマンドになります。** プルリクエストの題名を直接インラインスクリプトへ埋めると、入力者が処理の一部を決められます。中間の環境変数へ置けばデータのままです。 `pull_request_target` と `workflow_run` は高権限の処理を起動し、その中で信頼できないプルリクエストをチェックアウトすると、リポジトリーが侵害の危険にさらされると、GitHubは警告します。

- **管理者がブランチ規則を素通りします。** 既定では管理者権限の人を保護規則の制限から除きます。上記設定の回避を許可しないという意味の「Do not allow bypassing the above settings」を有効にすると、管理者にも適用します。
- **機密情報がプッシュ保護をすり抜けます。** 既定で有効なものは、各利用者自身が公開リポジトリーへ送るプッシュを保護します。全貢献者からの送信を守るリポジトリー単位のものにはGitHub Secret Protectionが必要で、有効にするまで無効です。有効でも書き込み権限があれば理由を付けて回避でき、アラートが発生し、監査ログにも記録されます。送った認証情報は全複製にあるため、履歴の書き換えに頼らず交換してください。

参考資料

- GitHub Docs — リポジトリー、プルリクエスト、Actions、セキュリティ機能に関する包括的なドキュメントです。
- GitHub Actionsを理解する — ワークフロー、イベント、ジョブ、ステップ、ランナーの関係です。
- 安全な利用の参照文書 — 固定、トークン権限、スクリプト注入についてのGitHub自身の説明です。
- 保護ブランチについて — 必須レビュー、必須のステータスチェック、それらを回避できる人を説明します。
- Dependabotのバージョン更新 — `dependabot.yml` が更新の提案を動かす仕組みです。
- プッシュ保護 — 二種類が止めるものと、回避の記録です。

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

「GitHub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/github/