Engineer

GitHub

Microsofts Hosting für Git-Repositorys

Was es ist

GitHub ist ein Hostingdienst von Microsoft, der Git-Repositorys auf eigenen Servern speichert und einen Arbeitsplatz darum aufbaut: eine Webansicht des Codes und seiner Historie, Pull Requests zum Vorschlagen und Prüfen von Änderungen, Issues zum Verfolgen von Aufgaben und GitHub Actions zum automatischen Ausführen von Jobs, sobald etwas in einem Repository geschieht.

Darunter liegt gewöhnliches Git. Ein von GitHub geklontes Repository ist ein normales Git-Repository mit vollständiger Historie. Zwischen Laptop und GitHub bewegt sich die Arbeit mit den üblichen Push- und Fetch-Vorgängen. Was GitHub hinzufügt, liegt auf seiner Seite der Verbindung: Review-Diskussionen, Issues, Branch-Regeln, Automatisierungseinstellungen und die Liste, wer was tun darf, bleiben in GitHubs eigenen Systemen, außerhalb des geklonten Repositorys.

Ein Pull Request ist GitHubs Bezeichnung für den Vorschlag, die Änderungen eines Branches in einen anderen zusammenzuführen. Er zeigt die Unterschiede Zeile für Zeile, sammelt Kommentare und Freigaben und zeigt das Ergebnis jeder automatischen Prüfung des vorgeschlagenen Codes. Ein als Entwurf eröffneter Pull Request ist als unfertig markiert. Er kann erst zusammengeführt werden, nachdem ihn jemand als prüfbereit markiert hat.

GitHub Actions ist die eingebaute Automatisierung des Dienstes. Ein Workflow ist eine YAML-Datei im Verzeichnis `.github/workflows` des Repositorys. Er benennt seine auslösenden Ereignisse, etwa einen Push, einen neuen Pull Request oder einen Zeitplan, und listet Jobs aus einzelnen Schritten auf. Jeder Schritt führt einen Shell-Befehl oder eine Action aus, einen als Paket bereitgestellten Automatisierungsbaustein, der sich zwischen Repositorys teilen lässt. Jobs laufen

auf Runnern: virtuellen Maschinen mit Ubuntu Linux, Windows oder macOS, die GitHub für jeden Lauf neu startet, oder eigenen Rechnern, die als selbst gehostete Runner registriert sind.

Zwei weitere Dienste überwachen, was in ein Repository gelangt. Dependabot liest eine Datei namens `dependabot.yml` und eröffnet Pull Requests, die Abhängigkeiten aktualisieren. Unabhängig davon eröffnet es Pull Requests für Abhängigkeiten mit bekannten Sicherheitslücken. Push Protection prüft eingehende Pushes auf Geheimnisse wie Zugriffstoken und blockiert einen Push, der eines enthält, bevor er das Repository erreicht.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist GitHub?“ 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

Für viele Teams wird die Arbeit auf GitHub erledigt, nicht nur gespeichert. Der Code, die Diskussion jeder Änderung, die Aufzeichnung, wer sie freigegeben hat, und das Ergebnis jeder Prüfung stehen nebeneinander. Warum eine Zeile existiert, lässt sich daher meist auf einer Seite beantworten.

Die Automatisierung ist genauso wichtig wie das Hosting. Mit Actions neben dem Code kann ein Pull Request seine eigenen Belege mitbringen: Die Tests liefen und bestanden, der Build funktionierte, die Scanner meldeten ihre Funde. Geschützte Branches machen aus diesen Belegen eine Regel. Ein Branch lässt sich so einstellen, dass er Änderungen nur über einen Pull Request, nur nach einer festgelegten Anzahl zustimmender Reviews und erst nach bestandenen, namentlich festgelegten Statusprüfungen annimmt. Die Hauptlinie nimmt damit ausschließlich geprüfte und freigegebene Änderungen auf.

Weil GitHub dort sitzt, wo Code in ein Projekt gelangt, liegen dort auch die Kontrollen der Softwarelieferkette: welche Actions Dritter ein Workflow in welcher genauen Version ausführen darf, wie Abhängigkeiten aktuell bleiben und ob ein Geheimnis beim Push gestoppt wird. Diese Kontrollen werden meist je Repository oder Organisation eingeschaltet und abgestimmt. Ein großer Teil der tatsächlichen Sicherheit eines GitHub-Projekts liegt deshalb in Einstellungen, die niemand bei einem Code-Review sieht.

Darin liegt der Kompromiss bei einem Hostingdienst. Sie erhalten eine vollständige, gepflegte Umgebung, ohne Server zu betreiben. Dafür werden die Regeln zum Schutz Ihres Codes in der Oberfläche eines anderen konfiguriert, mit Vorgaben für den allgemeinen Fall statt für Ihren eigenen.

Wie es in der Praxis aussieht

Drei Seiten aus der Arbeit des Unternehmens zeigen die unterschiedlichen Aufgaben, die GitHub übernimmt.

Eine auf GitHub Actions aufgebaute Auslieferungspipeline teilt die Arbeit in getrennte Workflows: einen für die verbindliche Codequalitätsprüfung, einen für die Tests und einen je Umgebung für Bereitstellungen. Daneben laufen CodeQL, die Prüfung von Abhängigkeiten und ein Schritt für eine Softwarestückliste. Der Weg führt vom Commit zu einem in Azure laufenden Dienst. Die Promotion führt einen Build auf einem festgelegten Weg durch die Umgebungen, statt ihn von Hand weiterzureichen. Das Frontend wird in einem Distroless-Image ohne Shell und Paketverwaltung ausgeliefert, die Go-API auf einer schlanken Alpine-Basis.

Ein kontinuierliches Programm für Sicherheitsprüfungen nutzt die Kontrollen um den Code auf GitHub. Jede GitHub Action eines Drittanbieters ist an einen Commit-Hash statt an ein Tag gebunden. Niemand kann daher ein Tag verschieben und damit ändern, was ein Workflow ausführt. Dependabot überwacht 6 Ökosysteme, Gitleaks durchsucht die Historie nach Zugangsdaten, und jeder Build erzeugt mit Anchore und Syft eine Softwarestückliste. Die Seite nennt auch die tatsächlichen Kosten: die Menge der Funde. Ein Scanner, der alles meldet, gewöhnt Menschen daran, ihn zu ignorieren. Damit das Signal nutzbar bleibt, müssen die Funde fortlaufend bewertet und priorisiert werden.

Die Bereitstellung einer GIS-Software-as-a-Service-Anwendung verlagerte ihren Test-, Build- und Bereitstellungszyklus in Workflows von GitHub Actions. Ein Release hing damit nicht mehr davon ab, dass jemand an die Schritte dachte.

Wiederkehrende Datenverarbeitung und Berichte wanderten in zeitgesteuerte Python-, Bash- und SQL-Jobs. Konfiguration und Geheimnisse wurden vereinheitlicht, damit sich jede Umgebung gleich verhielt.

So sieht die Bindung an einen Commit in einem Workflow aus. Die lange Hexadezimalzeichenfolge ist der vollständige Commit-Bezeichner des geprüften Releases. Der Kommentar hält daneben die für Menschen lesbare Version fest:

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

Wo es schiefgeht

- **Eine an ein Tag gebundene Action ändert sich ohne Warnung.** Ein Tag wie `v4` kann von der Person, die das Repository der Action kontrolliert, verschoben oder gelöscht werden. Der morgen ausgeführte Code kann deshalb vom heute geprüften abweichen. GitHub selbst empfiehlt die Bindung an einen vollständigen Commit-SHA. Ein Angreifer könnte ihn nur nachahmen, indem er „eine SHA-1-Kollision für die Nutzdaten eines gültigen Git-Objekts“ erzeugt.
- **Ein Workflow-Token hat mehr Rechte, als sein Job braucht.** Jeder Lauf erhält ein `GITHUB_TOKEN`. Seine Berechtigungen bestimmen, was ein kompromittierter Schritt ändern könnte. GitHub empfiehlt, standardmäßig nur Lesezugriff auf Repository-Inhalte zu gewähren und Schreibzugriff nur den Jobs zu geben, die ihn brauchen.
- **Nicht vertrauenswürdiger Text wird zum Befehl.** Schreibt ein Workflow den Titel eines Pull Requests direkt in ein eingebettetes Skript, darf dessen Verfasser damit einen Teil des Skripts bestimmen. Eine zwischengeschaltete Umgebungsvariable hält den Wert als Daten fest. Die Trigger `pull_request_target` und `workflow_run` starten privilegierte Workflows. GitHub warnt, dass das Auschecken eines nicht vertrauenswürdigen Pull Requests darin das Repository einer Kompromittierung aussetzt.
- **Administratoren gehen geradewegs an den Branch-Regeln vorbei.** Standardmäßig nimmt eine Branch-Schutzregel Personen mit Administratorrechten am Repository von ihren Beschränkungen aus. Die Einstellung „Do not allow bypassing the above settings“ (Umgehen der obigen Einstellungen nicht zulassen) bindet auch sie an die Regeln.

- **Ein Geheimnis rutscht durch Push Protection.** Die standardmäßig aktive Variante schützt die eigenen Pushes jedes Nutzers zu öffentlichen Repositorys. Die Variante, die ein Repository gegen alle Mitwirkenden schützt, braucht GitHub Secret Protection und bleibt bis zur Aktivierung abgeschaltet. Selbst dann kann jeder mit Schreibzugriff eine Blockierung mit einer Begründung umgehen. Das löst einen Alarm aus und wird im Audit-Log erfasst. Einmal gepushte Zugangsdaten leben in jedem Klon. Tauschen Sie sie deshalb aus, statt auf ein Umschreiben der Historie zu vertrauen.

Weiterführende Lektüre

- GitHub-Dokumentation — die gesamte Dokumentation zu Repositorys, Pull Requests, Actions und Sicherheitsfunktionen.
- GitHub Actions verstehen — Workflows, Ereignisse, Jobs, Schritte und Runner und ihr Zusammenspiel.
- Referenz zur sicheren Verwendung — Bindung von Actions an Commits, Token-Berechtigungen und Skript-Injection, von GitHub selbst erläutert.
- Geschützte Branches — erforderliche Reviews und Statusprüfungen sowie die Personen, die sie umgehen dürfen.
- Dependabot-Versionsaktualisierungen — wie eine Datei `dependabot.yml` die Aktualisierungs-Pull-Requests steuert.
- Push Protection — die beiden Varianten, was jede blockiert und wie eine Umgehung erfasst wird.



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 GitHub?“ 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/github/