Engineer

Verbindliche Qualitätsprüfung

Prüfungen, die Code vor dem nächsten Schritt bestehen muss

Was es ist

Eine verbindliche Qualitätsprüfung ist eine Reihe von Bedingungen, die eine Änderung erfüllen muss, bevor sie zur nächsten Stufe übergehen darf. Am Ende gibt es genau ein Ergebnis: bestanden oder durchgefallen. Die Stufe kann ein Commit, eine Zusammenführung in den Hauptbranch, eine Veröffentlichung oder eine Bereitstellung sein. Die Bedingungen können alles betreffen, was eine Maschine entscheiden kann: einen sauberen Lint-Lauf, bestandene Tests, eine Abdeckungszahl oder einen erfolgreich abgeschlossenen Build.

Das entscheidende Wort ist verbindlich. Eine Kontrolle meldet; eine verbindliche Prüfung entscheidet. Durchgesetzt wird die Entscheidung von dem, was den nächsten Schritt steuert. Ein Linter, der vierzig Warnungen in ein Log schreibt, ist eine Kontrolle. Derselbe Linter, so eingebunden, dass die Schaltfläche zum Zusammenführen bis zu einer sauberen Ausgabe deaktiviert bleibt, ist Teil einer verbindlichen Prüfung.

SonarQube Cloud, wo der Begriff eine benannte Funktion bezeichnet, beschreibt eine verbindliche Qualitätsprüfung als Bedingungen, an denen der Code während der Analyse gemessen wird. Jede Bedingung verbindet eine Metrik mit einem Vergleichsoperator und einem Fehlerwert. „Zuverlässigkeitsprobleme größer als 0“ ist daher eine Bedingung; sobald eine Bedingung erfüllt ist, schlägt die Prüfung fehl. Eine Bedingung gilt entweder für neuen Code oder für den Code als Ganzes.

Die Hostingplattformen bauen dieselbe Idee in ihren Zusammenführungsablauf ein:

- **GitHub** bietet die Branch-Schutzeinstellung „Require status checks before merging“, die Statusprüfungen vor dem Zusammenführen verlangt. Eine vorgeschriebene Prüfung muss erfolgreich, übersprungen oder neutral melden, bevor Änderungen den geschützten Branch erreichen.
- **GitLab** hat eine Zusammenführungsprüfung namens „Pipelines must succeed“, die erfolgreiche Pipelines verlangt. Ist sie gewählt, kann ein Merge Request weder mit fehlschlagender Pipeline noch ganz ohne Pipeline zusammengeführt werden.
- **Auf dem eigenen Entwicklungsrechner** installiert pre-commit Kontrollen als Git-Hooks, die beim Commit oder Push laufen. Ein Taskrunner wie Task gibt der ganzen Sammlung einen Namen.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine verbindliche Qualitätsprüfung?“ 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

Ein Standard, der nur in einem Dokument lebt, wird Ausnahme für Ausnahme verhandelt. Ein Standard, der den Build scheitern lässt, wird von allen befolgt, auch von der Person, die ihn geschrieben hat, an einem Nachmittag, an dem sie den Standard lieber umgehen würde. Eine verbindliche Prüfung macht vereinbarte Regeln ausführbar. Ihre Einhaltung hängt damit nicht mehr von Gedächtnis und Laune ab.

Sie verändert auch die Aufgabe menschlicher Reviews. Wenn eine Maschine bereits Formatierungsfehler, ungenutzte Imports, defekte Links und offengelegte Geheimnisse gefunden hat, kann sich die prüfende Person auf Absicht und Entwurf konzentrieren, über die eine Maschine nicht urteilen kann. Der Maßstab ist zudem für alle Autoren derselbe.

Die Beschränkung auf neuen Code macht eine solche Prüfung auch in einer alten Codebasis einführbar. Sonar way, die Standardprüfung in SonarQube Cloud, konzentriert sich auf neuen Code statt auf die Bereinigung alten Codes, mit

Bedingungen wie mindestens 80,0 % Testabdeckung und höchstens 3,0 % Duplikation im neuen Code. Auf einem Pull Request gelten nur Bedingungen für neuen Code. Ein Projekt mit Tausenden bestehender Probleme kann die Prüfung einschalten und jede neue Änderung an diesem Maßstab messen, ohne zuerst den gesamten Rückstand abzuarbeiten.

Die Geschwindigkeit entscheidet, ob die Prüfung eingeschaltet bleibt. Task führt die Abhängigkeiten einer Aufgabe gleichzeitig aus. Eine aus vielen unabhängigen Kontrollen bestehende Prüfung kann sie deshalb nebeneinander statt nacheinander laufen lassen. Auch eine Aufteilung nach Aufwand hilft: die günstige Hälfte bei jedem Commit, die ganze Sammlung, bevor Code den Rechner verlässt.

Diese Website hat ihre eigene verbindliche Prüfung: `task check`, ausgeführt auf dem Rechner, auf dem die Website geschrieben wird, statt bei einem gehosteten Builddienst. Jeder Befund zählt als Fehler. Der Seitengenerator läuft mit `--panicOnWarning`, sodass selbst ein Abkündigungshinweis den Build stoppt. Inline-Unterdrückungen wie `# noqa` oder `stylelint-disable` sind verboten: Die Ursache eines Fehlers wird behoben, statt den Befund stummzuschalten. Ein Pre-commit-Hook führt bei jedem Commit die dateibezogenen Prüfungen aus; ein Pre-push-Hook führt den vollständigen `task check` aus, der Minuten statt Sekunden braucht. Ein erfolgreicher Lauf aus einem sauberen Arbeitsstand versieht diesen Stand mit einer Prüfmarke; das Bereitstellungsskript lehnt einen Stand ohne Marke ab. Seine einzige Umgehungsmöglichkeit ist eine Umgebungsvariable, die deutlich warnt. Wer die Prüfung im Notfall auslässt, muss dies also kenntlich machen.

Wie es in der Praxis aussieht

Die Credits-Seite der Website nennt die Werkzeuge dieser Prüfung: vnu für HTML, Stylelint, axe-core auf einer Seite je Vorlage, Lighthouse, lychee für Links, Prettier, markdownlint, ESLint, Ruff, basedpyright, yamllint, ShellCheck, codespell und gitleaks. Task führt alles unter einem Namen pro Aufgabe aus, und pre-commit wiederholt bei jedem Commit die dateibezogene Hälfte. Die Seite nennt auch, was außerhalb bleibt: zwei von Puppeteer gesteuerte Browserprüfungen, auf über die Ansichtsfläche hinausragende Inhalte und auf visuelle Regression, die eine Person bei Bedarf startet.

Ein Einpersonenprojekt hat überhaupt keinen Reviewer. Davon geht eine Portfolioseite aus. Die Standards mussten ausführbar sein, damit ein Verstoß den Commit scheitern lässt, statt auf eine zufällige Entdeckung zu warten. Im dokumentierten Implementierungsstand vom August 2026 war die Antwort eine Commit-Prüfung mit 22 Lintern ohne Warnstufe, ergänzt um projektspezifische Kontrollen, zu denen kein Standardwerkzeug eine Meinung hat. Dazu gehört, dass der bereitgestellte Dateibaum weder private Pfade noch private Hostnamen enthält und die Texte in drei Sprachen ihre Tonregeln einhielten. Ein bereits vorhandener Fehler wird im selben Durchgang behoben, statt akzeptiert. Die Seite nennt den Preis deutlich: Ein Commit ist langsam, und die verbindliche Prüfung ist der einzige Reviewer des Projekts.

Eine Prüfung kann sich auch selbst widersprechen. Eine andere Seite beschreibt ein Projekt mit Commit-Hook und vollständiger Prüfung, einem Dokument, laut dem der Hook diese Prüfung ausführt, und zwei von Hand in zwei Dateien gepflegten Prüflisten. Ein Test, der beide Listen auf die tatsächlich aufgerufenen Prüfungen auflöst, zeigte fünf Kontrollen, die nur in der vollständigen Prüfung liefen. Genau sie lasen die Texte. Jede den Code schützende Kontrolle lief automatisch; jede die Texte schützende Kontrolle lief nur, wenn jemand daran dachte. Die fünf wanderten in den Hook. Zwei wurden schnell genug gemacht, um in dessen Zeitbudget zu passen, und der Vergleichstest blieb erhalten, um Unterschiede zwischen den Listen zu erkennen.

Wo es schiefgeht

Die meisten verbindlichen Prüfungen versagen, indem sie Fehler durchlassen und trotzdem grün zeigen.

Warnungen sammeln sich hinter einem bestandenen Ergebnis. Eine Prüfung, die nur bei Fehlern scheitert, lässt Warnungen anwachsen, bis das Log zu lang zum Lesen ist. Dann erscheint ein echtes Problem als Warnung zweihunderteins. Schaffen Sie die Warnstufe ab: Entweder rechtfertigt ein Regelverstoß das Scheitern der Prüfung, oder die Regel wird abgeschaltet.

Eine übersprungene Kontrolle zählt als bestanden. Bei GitHub erfüllt ein übersprungener Status eine vorgeschriebene Prüfung. Eine durch ihre `if:` -Bedingung übersprungene Aufgabe meldet Erfolg. Auch eine von einer fehlgeschlagenen Aufgabe abhängige Aufgabe wird übersprungen und blockiert

die Zusammenführung möglicherweise nicht. GitHub empfiehlt `always()` mit `needs`: Verlangen Sie eine Aufgabe, die immer läuft und nur besteht, wenn alle ihre Abhängigkeiten erfolgreich waren.

```
# .github/workflows/check.yml (excerpt)
jobs:
    gate:
        if: always()
        needs: [lint, test, build]
        runs-on: ubuntu-latest
        steps:
            - run: |
                  for result in ${{ join(needs.*.result, ' ') }}; do
                      test "$result" = success || exit 1
                  done
```

Auch die umgekehrte Falle gibt es: Ein durch Pfad- oder Branchfilter übersprungener vorgeschriebener Workflow lässt seine Prüfungen ausstehend und blockiert jede Zusammenführung. Vermeiden Sie vorgeschriebene Workflows, die übersprungen werden können.

Zwei Listen driften auseinander. Der Hook führt eine Sammlung aus, der Server eine andere, und ein Dokument behauptet ihre Gleichheit. Definieren Sie die Prüfungen einmal und rufen Sie diese Definition überall auf, oder behalten Sie einen Test, der die Listen vergleicht und bei Abweichungen scheitert.

Die Prüfung wird langsam und wird umgangen. Eine Option zum Überspringen der Hooks ist schnell zur Hand. Bei GitHub gilt der Branch-Schutz standardmäßig nicht für Repositoryadministratoren; „Do not allow bypassing the above settings“ schließt diese Ausnahme. Messen Sie den Zeitverbrauch, führen Sie unabhängige Prüfungen gleichzeitig aus, halten Sie den Teil beim Commit klein und verschieben Sie eine tatsächlich langsame Kontrolle in einen dokumentierten manuellen Schritt.

Eine Kontrolle besteht, weil sie nichts geprüft hat. Ein Linter, dessen Dateimuster keine einzige Datei trifft oder dessen Ausschlussmuster ein ganzes Verzeichnis verschluckt, beendet sich fehlerfrei. Ein fehlerfreies Ende sieht genauso aus wie fehlerfreier Code. Testen Sie die Prüfung selbst: Geben Sie jeder Kontrolle eine bekanntermaßen fehlerhafte Datei und erwarten Sie, dass sie scheitert.

Weiterführende Lektüre

- Einführung in verbindliche Qualitätsprüfungen — Bedingungen, die Standardprüfung Sonar way und die Konzentration auf neuen Code.
- Geschützte Branches — vorgeschriebene Statusprüfungen und die Personen, die sie umgehen dürfen.
- Probleme mit vorgeschriebenen Statusprüfungen beheben — als bestanden zählende Statuswerte und das Verhalten übersprungener Workflows und Aufgaben.
- Automatisches Zusammenführen — GitLabs Prüfung „Pipelines must succeed“ und ihr Umgang mit übersprungenen Pipelines.
- pre-commit — einmal deklarierte Prüfungen, die als Git-Hooks für Commit und Push installiert werden.
- Abhängigkeiten in Task — wie Task die Abhängigkeiten einer Aufgabe gleichzeitig und ihre Befehle nacheinander ausführt.

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 verbindliche Qualitätsprüfung?“ 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/quality-gate/