Engineer Plattform

Tests und Qualitätssicherung

28 Leistungen

Die Teile Ihres Systems verbinden

Die unter „Tests und Qualitätssicherung“ gesammelten Arbeiten zeigen einen Teil eines funktionierenden Systems. Wir können helfen, diesen Teil mit dem Rest Ihres Betriebs zu verbinden. Geprüft, veröffentlicht und rückgängig zu machen: Änderungen erreichen Ihre Kunden am Dienstagnachmittag statt spät am Freitagabend.

Unsere Leistungen entdecken Kontakt aufnehmen →

Eine vergleichbare Herausforderung finden

Folgen Sie einem Beispiel, das Ihrer Situation ähnelt. Die Rückschau beschreibt die Entscheidungen hinter dem Ergebnis und geht über die bloße Technologiebezeichnung hinaus. GIS-SaaS-Deployments mit GitHub automatisiert — Bereitstellung, Daten und Berichte ohne Handarbeit

GIS-SaaS-Deployments mit GitHub automatisiert Kontakt aufnehmen →

CI/CD vereinfacht und 4.000 Stunden gespart

CI/CD-Prozesse verbessert und durch die Automatisierung von Softwareentwicklungspipelines 4.000 Stunden eingespart.

Den API-Vertrag aus der Datenbank generiert

Den API-Vertrag von der Datenbank aus nach außen generiert — OpenAPI, einen typisierten TypeScript-Client mit 44.076 Zeilen, 61 Mock-Handler und die von der UI durchgesetzten Grenzen — mit einer Prüfung an jedem Übergang, die Abweichungen zurückweist.

GitHub-Actions-CI/CD-Pipelines aufgebaut

GitHub-Actions-CI/CD-Pipelines mit einem Distroless-Frontend-Image für den Produktivbetrieb und der Übernahme von Releases über mehrere Umgebungen entwickelt.

578 go-task-Automatisierungsziele verfasst

578 go-task-Automatisierungsziele für native, Docker- und HTTPS-Entwicklungsmodi, Linting, Tests, Datenbank und Bereitstellung verfasst.

Warnungsfreiheit für sechs Sprachen durchgesetzt

Für sechs Sprachen — Go, TypeScript, SQL, Python, Shell und Markdown — einen durch Pre-Commit-Hooks durchgesetzten Qualitätsstandard ohne Warnungen festgelegt.

Testsuite für Go, SQL und Browser aufgebaut

Eine mehrschichtige automatisierte Testsuite aufgebaut — 981 Go-Tests, 543 Frontend- und Browserspezifikationen, 494 verhaltensprüfende SQL-Tests — mit Mutationstests, eigenschaftsbasierten Tests und einer verbindlichen Barrierefreiheitsprüfung.

Repository-Prüfengine mit 268 Checks entwickelt

Die Prüfengine des Repositorys entwickelt — 268 registrierte Commit-Prüfungen, 277 Lint-Regeln und 15 eigene ESLint-Regeln — sowie 146 Tests der Prüfungen selbst, damit der Build den Standard durchsetzt und nicht erst das Review.

1.022 Migrationen mit Schemaabweichungsprüfung

Die Konsistenz des Schemas über 1.022 Migrationen mit einer CI-Prüfung abgesichert, die die Datenbank auf zwei Wegen aufbaut — Neuinstallation sowie Installation mit sämtlichen Migrationen — und bei Unterschieden fehlschlägt.

SSH mit 24 geprüften Direktiven gehärtet

SSH auf 24 geprüfte Direktiven gehärtet und dreistufig validiert — Kandidatendatei, zusammengesetzte Konfiguration und anschließend die Rückmeldung des Daemons selbst — nachdem diese gezeigt hatte, dass der laufende Server zwei der vierundzwanzig Direktiven stillschweigend überschrieb.

Angriffssperren bei jedem Lauf nachgewiesen

Den Pfad zur Angriffssperre bei jedem Härtungslauf durchgängig nachgewiesen: eine reservierte Testadresse gesperrt, die entstandene Kernelregel ausgelesen und die Sperre in einem garantierten Aufräumblock aufgehoben, sodass eine ausgefallene Jail den Lauf scheitern lässt, statt einen gesunden Zustand zu melden.

Firewallregeln nach Position geprüft

Firewallregeln nach ihrer Position statt bloß ihrer Existenz geprüft, anhand der nummerierten Regelliste und der aktiven Paketfilterkette, denn eine vorhandene Regel wird nicht zwingend von einem Paket erreicht.

Ansibles Prüfmodus aussagekräftig gemacht

Den Prüfmodus auf der gesamten Plattform aussagekräftig gemacht, nachdem sechs Sonden anhand eines vom Host nie gelieferten Werts entschieden hatten: Ansibles command-Modul meldet unter --check Erfolg, obwohl es den Befehl vollständig überspringt.

Sekunden-Preflight für Geheimnisse ergänzt

Ein Preflight-Play ergänzt, das denselben Code wie der Konvergenzlauf mit den lokal beim Betreiber liegenden Geheimnissen in etwa einer Sekunde ausführt, nachdem ein teilweise durchgelaufener Produktivlauf bei Aufgabe neun abbrach, während die Swap-Einstellungen bereits auf dem Produktivhost geschrieben waren.

Alle systemd-Units stärker isoliert

Die Angriffsfläche der systemd-Sandboxen für jede von der Plattform installierte Unit verringert — einen Totmannschaltungsdienst von 9,6 UNSAFE auf 1,5, eine internetseitig erreichbare Git-Forge von 8,3 EXPOSED auf 1,5 — und eine Parserprüfung beim Konvergenzlauf ergänzt, nachdem eine falsch geschriebene Direktive in drei Unit-Vorlagen stillschweigend ignoriert worden war.

Commit-Prüfung mit 22 Lintern aufgebaut

Eine verbindliche Commit-Prüfung aus 22 einzeiligen Lintern und fünf ausführlicheren Prüfungen aufgebaut, ohne Warnungsstufe und ohne Inline-Unterdrückungen, für HTML, CSS, JavaScript, Python, YAML, Markdown, Shell, Links, Rechtschreibung, Geheimnisse und Typografie.

Ein Sicherheitsnetz, das keine halbe Stunde mehr kostet

Die browsergestützte Qualitätsprüfung der Website von 1.636 auf 615 Sekunden verkürzt, indem die Prüfungen in einem auf vier parallele Ausführungen begrenzten Worker-Pool nach längster Laufzeit zuerst eingeplant werden, nachdem die alphabetische Reihenfolge 320 statt 224 Sekunden benötigt hatte.

Sitemap mit einheitlichem Zeitstempel korrigiert

Eine Sitemap korrigiert, in der 172 von 176 URLs denselben Änderungszeitpunkt trugen, indem das Datum nach der Feststellung aus der Git-Historie gelesen wurde, dass der Export jede Datei bei jedem Lauf neu schreibt.

Qualitätsprüfung mit 10.242 Zeilen selbst geprüft

10.242 Zeilen JavaScript der Qualitätsprüfung unter Formatter und Linter gebracht, nachdem sie sich als größter und zugleich ungeprüfter Codebestand des Repositorys erwiesen hatten; 13 Befunde behoben und keinen unterdrückt.

Generator mit 981 Tests und 92 % Abdeckung

Den Generator mit 981 Testfällen und mindestens 92 % Zweigabdeckung geprüft, Warnungen als Fehler behandelt und Idempotenz durch zwei vollständige Pipeline-Durchläufe aus einer leeren Datei nachgewiesen, wobei der zweite nichts ändern darf.

Jede Lint-Regel als Fehler aktiviert

Sämtliche Regeln des Python-Linters als Fehler aktiviert und 1.815 Befunde bis auf null abgearbeitet; jede der wenigen Ausnahmen schriftlich begründet und zwei davon durch eine Prüfung statt nur einen Kommentar abgesichert.

Die Prüfwerkzeuge selbst mit Tests geprüft

Tests für die Prüfwerkzeuge selbst geschrieben, nachdem klar war, dass ein nur mit fehlerfreien Eingaben getestetes Werkzeug irgendwann auch nach dem Lesen von nichts Erfolg meldet — einen Tippfehler eingebaut, um die Rechtschreibprüfung zu prüfen, und einen ID-Bereich aus der Datenbank statt einer Testkonstante übernommen.

Abweichende Hooks und Qualitätsprüfung gefunden

Durch den Vergleich beider Listen in einem Test festgestellt, dass Commit-Hooks und Qualitätsprüfung unterschiedliche Prüfungen ausführten, obwohl ein Dokument Gleichheit versprach — die fünf ausschließlich von Hand ausgeführten Prüfungen waren gerade diejenigen für die Lebenslauftexte.

Forge-CI-Hook geprobt und 2 Fehler gefunden

Den CI-Hook auf der Forge probeweise ausgeführt und zwei beim bloßen Lesen unerreichbare Fehler gefunden: einen Fallback mit einem nicht auflösbaren Eingabeargument und eine Git-Umgebungsvariable, die der Qualitätsprüfung in den Checkout folgte und 19 Tests scheitern ließ.

Eine begrenzte Ereignispipeline für eine macOS-App

Ereignispuffer und Protokollierung nach einem dokumentierten macOS-Vorfall mit 41 MB Wachstum pro Sekunde und 111 GB komprimierten Speicherseiten auf einem Rechner mit 36 GB begrenzt; ein dokumentierter vollständiger Testlauf reduzierte Kernprotokollzeilen von 610.996 auf 1.411.

Parser zur Prüfung aller FFI-Aufrufe geschrieben

Einen Parser geschrieben, der den echten C-Header mit 7.308 Zeilen liest und jede Aufrufstelle, jede Enum-Konstante sowie final für jede pointerbesitzende Klasse prüft, nachdem ein handgeschriebener Platzhalterheader Aufrufe dreier entfernter Funktionen kompilieren und linken ließ, die zur Laufzeit abstürzten.

Testsuite auf 8,9 Sekunden beschleunigt

Die Testsuite von sechs Tests jenseits eines Limits von sechzig Sekunden auf 135 erfolgreiche Tests in 8,9 Sekunden gebracht, indem der Hauptthread profiliert und die zwei Aufrufe entfernt wurden, in denen er bei 3.989 von 4.017 Stichproben festhing.

Die ungenutzte Hälfte des Kerns erschlossen

Die zuvor ungenutzte Hälfte des Messaging-Kerns erschlossen — Sicherungsübertragung, verschwindende Nachrichten, Nachrichtenbearbeitung und erneutes Senden, verifizierte Einladungen, Proxys und Verschlüsselungsrichtlinien — und jeden Test ohne Mocks gegen die echte Bibliothek ausgeführt.

Sechs Sammler weisen übersprungene Eingaben aus

Sechs Berichtssammler um Angaben zu ungelesenen Eingaben ergänzt — ungeöffnete Dateien, nicht installierte Sonden, unbekannte Journalzeilen — und Schwellenwerte anhand von 116 gemessenen Stichproben bestimmt, nachdem eine blinde Erfassung und ein gesunder Host dieselbe Seite ergeben hatten.

Die technische Arbeit gezielt einsetzen

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 →

Das Thema in Ihren Kontext einordnen

Erzählen Sie uns, welche Rolle „Tests und Qualitätssicherung“ in Ihrem eigenen System spielt. Wir können helfen, die passende Leistung auszuwählen und die damit verbundene Arbeit zu definieren.

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://platform.engineer.company/de/categories/testing/