Engineer

Continuous Delivery

Software jederzeit sicher veröffentlichen können

Was es ist

Continuous Delivery bedeutet, dass ein Team eine Softwareänderung bei Bedarf sicher veröffentlichen kann. Jede Änderung durchläuft einen wiederholbaren Weg. Die Ergebnisse zeigen, ob die Voraussetzungen für den nächsten Schritt erfüllt sind. Der Wechsel in Produktion kann weiterhin von einer Freigabe, einem vereinbarten Zeitfenster oder betrieblichen Umständen abhängen. Sobald die Entscheidung ansteht, liegen eine geprüfte Version und ein bekannter Installationsweg vor.

Bei Continuous Integration führen die Beteiligten kleine Änderungen häufig zusammen und prüfen den gemeinsamen Code. Das zeigt, ob sich das Programm bauen lässt und ob Tests bestehender Funktionen weiter bestehen. Continuous Delivery umfasst auch die Bereitstellung des Ergebnisses in den vorgesehenen Umgebungen und passende Funktionsprüfungen. Bei Continuous Deployment erreichen geeignete Änderungen die Produktion automatisch. Eine ausdrückliche Freigabe lässt sich mit Continuous Delivery verbinden, wenn sie eine bereits vorbereitete Version betrifft.

Bereitstellung und Freischaltung können an unterschiedlichen Tagen stattfinden. Die neue Version läuft gegebenenfalls schon in Produktion, obwohl ein Schalter die neue Funktion noch deaktiviert. Später lässt sich die Funktion zunächst für wenige Personen freigeben. Beide Schritte benötigen eigene Prüfungen.
Das Team sollte zum Beispiel beobachten können, ob der bisherige Ablauf nach der Installation funktioniert und ob die Freischaltung die gewünschte Gruppe erreicht.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Continuous Delivery?“ 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

Bei einem manuellen Ablauf hängt eine Veröffentlichung womöglich davon ab, ob eine einzige Person ihre Befehle in der richtigen Reihenfolge ausführt. Ein gemeinsamer Weg hält das Ergebnis jedes Teilschritts fest. So lässt sich unterscheiden, ob ein Test fehlgeschlagen ist, eine Umgebung nicht erreichbar war oder jemand noch über die Freigabe entscheiden muss. Erfasst werden sollte auch die Kennung des tatsächlich installierten Pakets. Ein bewegliches Etikett wie `latest` sagt im Nachhinein wenig über den laufenden Stand aus.

Teure Veröffentlichungen verleiten dazu, mehrere unabhängige Änderungen bis zum nächsten Termin zu sammeln. Wenn ein Fehler auftritt, kommen entsprechend mehr Ursachen infrage. Ein eingeübter Weg erleichtert kleinere Lieferungen. Das Team kann sehen, welche einzelne Änderung einen beobachteten Effekt ausgelöst haben könnte. Dieselben Nachweise helfen bei einem bewussten Aufschub, wenn zum Beispiel ein abhängiger Dienst gerade gewartet wird.

Eine Lieferung kann mehrere Repositorien betreffen. Für die Datenbank, Netzwerkregeln oder Anmeldeeinstellungen sind vielleicht andere Teams zuständig als für die Anwendung. Legen Sie fest, welche Versionen und Einstellungen zusammen funktionieren müssen, und prüfen Sie eine repräsentative Kombination. Ein getestetes Programm allein reicht als Aussage für eine ungetestete Umgebung nur begrenzt. Eine zweite Person sollte den Ablauf mit den erwarteten Berechtigungen nachvollziehen können.

Wie es in der Praxis aussieht

Das Fallbeispiel zu CI/CD mit GitHub Actions beschreibt getrennte Abläufe für Codeprüfungen, Tests und Bereitstellung in mehreren Umgebungen. Oberfläche, API und Datenbank nutzten jeweils eine zu ihrer Aufgabe passende Image-Basis. Im Projekt wurde die Veröffentlichung durch einen definierten Weg und sichtbare Kontrollen zur Routine. Andere Anwendungen können andere Pakete oder Freigabeverfahren benötigen.

Stellen Sie sich eine Anwendung vor, die eine Exportfunktion erhält. Der Ablauf hält den Quellstand fest, prüft den Code und erzeugt ein Paket mit eindeutiger Kennung. In einer Testumgebung lädt ein berechtigtes Testkonto eine Datei mit repräsentativen Daten herunter. Der Test prüft Spalten und Zugriffsregeln. Die verantwortliche Person sieht die Kennung und den Nachweis, dass genau dieses Paket getestet wurde. In Produktion wird es mit den dort vorgesehenen Einstellungen installiert. Ein kurzer Funktionstest erfasst den neuen Export und einen bestehenden Ablauf. Das Ergebnis kommt zusammen mit der Paketkennung ins Bereitstellungsprotokoll.

Wenn die Prüfung in der Testumgebung fehlschlägt, bleibt das Paket mit einer nachvollziehbaren Ursache zurück. Bei einer Störung in Produktion lassen sich die vorherige Paketkennung und die erste betroffene Messung nachschlagen. Erproben Sie den Ablauf bewusst mit einer nicht verfügbaren Abhängigkeit. Ist eine Datenbankänderung beteiligt, prüfen Sie, ob die vorige Anwendungsversion auch mit dem Zwischenzustand des Schemas arbeiten kann.

Wo es schiefgeht

Ein neuer Build für jede Umgebung kann verändern, was zuvor geprüft wurde. Selbst zum gleichen Commit können sich Abhängigkeiten oder das Compiler-Image geändert haben. Verwenden Sie in den Zielumgebungen nach Möglichkeit dasselbe identifizierte Paket. Wenn unterschiedliche Plattformen eigene Varianten erfordern, braucht jede Variante eigene Nachweise. Eine gemeinsame Versionsbezeichnung belegt die übereinstimmenden Inhalte einer Installation nicht.

Eine Freigabe kann zu einem zweiten Lieferverfahren heranwachsen. Geben Sie der entscheidenden Person die Paketkennung, die Zielumgebung und die einschlägigen Prüfergebnisse. Wenn sie danach alle technischen Schritte selbst

ausführen muss, ist die letzte Strecke nur schwer übertragbar. Halten Sie fest, welche Entscheidungen menschliches Urteil brauchen. Bauen Sie die Routine so, dass eine andere berechtigte Person sie übernehmen kann.

Eine grüne Prüfung gibt schwache Auskunft, wenn sie nach einem Neustart zufällig besteht. Untersuchen Sie solche Fehler und prüfen Sie, ob ein Signal tatsächlich die fachliche Funktion des Dienstes abbildet. Planen Sie bei riskanten Änderungen ein Beobachtungsfenster. Benennen Sie die Abbruchbedingung und die Person, die sie durchsetzen kann. Nach einer Unterbrechung beginnt der nächste Versuch mit einem bekannten Paket und einem überprüften Zustand.

Weiterführende Lektüre

- DORA zu Continuous Delivery definiert die bedarfsgerechte Veröffentlichungsfähigkeit.
- DORA zu Continuous Integration beschreibt häufige Integration und schnelle Rückmeldung.
- DORA zur Automatisierung von Deployments erklärt wiederholbare Schritte für verschiedene Umgebungen.
- Steuerung von Deployments mit GitHub Actions erläutert Umgebungen, Jobs und Schutzregeln.

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 Continuous Delivery?“ 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/continuous-delivery/