

PHP-Automatisierung in der Praxis

Leseprobe – Kapitel 1 (Auszug)

Wiederkehrende Aufgaben von Hand erledigen, Deployments per FTP hochladen, Tests nur "wenn Zeit ist" laufen lassen – so verlieren PHP-Teams täglich Stunden und riskieren vermeidbare Fehler. Dieses Buch zeigt Schritt für Schritt, wie sich der komplette PHP-Entwicklungsprozess automatisieren lässt: von zeitgesteuerten Aufgaben mit Cron über CI/CD-Pipelines mit Jenkins, GitLab CI/CD und GitHub Actions bis zu automatisiertem Testen mit PHPUnit und Pest, Zero-Downtime-Deployments mit Deployer und Docker sowie statischer Codeanalyse mit PHPStan und Psalm.

Wenn Klicks entscheiden: warum Automatisierung kein Luxus mehr ist

Drei Uhr nachts. Der Newsletter an 14.000 Empfänger sollte längst raus sein. Stattdessen hängt ein Skript seit zwei Stunden in einer Endlosschleife, weil ein Praktikant vergessen hat, eine Datenbanksicherung vor dem Versand einzuziehen. Am Morgen erklärt der Geschäftsführer, warum diese Aktion „nicht wieder vorkommen darf“. Das ist kein Einzelfall. In PHP-Projekten, die über Jahre gewachsen sind, finden sich solche Zeitzünder an allen Ecken: manuelle Deployments am Freitagabend, lokal getesteter Code, der auf dem Server plötzlich andere Pfade vorfindet, Newsletter, die sporadisch doppelt verschickt werden.

Der ökonomische Treiber für Automatisierung ist so schlicht wie unbestechlich. Jeder händische Schritt kostet Zeit, die ein Entwickler bezahlt bekommt, und produziert Fehler, die im Nachgang teurer werden. Eine Studie nach der anderen zeigt, dass die Fehlerdichte in automatisierten Pipelines um Größenordnungen unter der von Hand ausgeführter Prozesse liegt – der Hebel ist also nicht „ein bisschen bequemer“, sondern strukturell. Wer kontinuierlich liefert, kann kontinuierlich korrigieren, statt in großen, risikoreichen Releases zu denken.

Qualitativ kommt ein zweiter Aspekt hinzu: Code, der nicht durch eine Pipeline geht, ist Code, den niemand außer dem Autor gesehen hat. Selbst eine erfahrene Entwicklerin kann ermüden, abgelenkt sein oder schlicht vergessen, eine Randbedingung zu testen. Ein Werkzeug wird nicht müde. Es prüft die gleiche Regel jedes Mal mit derselben Sorgfalt – und dokumentiert gleichzeitig, was geprüft wurde.

Merke. Manuelle Prozesse sind nicht „nur“ ein Risiko. Sie sind ein laufender Kostentreiber, weil sie bezahlte Arbeitszeit für wiederholbare Tätigkeiten verbrennen und jede Wiederholung eine neue Fehlerquelle darstellt.

Eine kleine Sprachkunde: CI, CD und Continuous Deployment

Im Entwickleralltag werden die drei Begriffe oft munter gemischt. Streng genommen beschreiben sie jedoch unterschiedliche Reifegrade, und ein PHP-Team sollte bewusst entscheiden, auf welcher Stufe es stehen will.

Continuous Integration (CI) bedeutet, dass jeder Push in eine geteilte Codebasis automatisch eine Pipeline anstößt: Tests laufen, statische Analyse wird ausgeführt, die Anwendung wird gebaut. Ziel ist es, Integrationskonflikte früh sichtbar zu machen – nicht erst am Tag vor dem Release. CI ist die Voraussetzung für alles, was danach kommt.

Continuous Delivery (CD im engeren Sinne) geht einen Schritt weiter: Das Ergebnis der Pipeline ist stets ein lauffähiges Artefakt – ein Paket, ein Container-Image, eine versionierte Codebasis –, das jederzeit deployt werden könnte. Der eigentliche Roll-out auf die Produktion erfolgt jedoch weiterhin per Knopfdruck eines Menschen.

Continuous Deployment schließlich hebt auch diese letzte Hürde auf: Jeder grüne Build geht automatisch in die Produktion. Das liegt unter anderem an zwei Verfahren: **Zero-Downtime-Strategien** sorgen dafür, dass während des Roll-outs kein Nutzer eine Fehlermeldung sieht, und **Feature Flags** erlauben es, neue Funktionen zunächst versteckt auszuliefern und nur für einen Teil der Nutzer sichtbar zu schalten.

Lernziel. Nach diesem Kapitel unterscheidest du CI, CD und Continuous Deployment sauber und kannst einordnen, welche Stufe für dein Team realistisch ist.

Wie dieses Buch aufgebaut ist

Wir gehen die Reifegrad-Stufen nicht der Reihe nach durch, weil die meisten Lesenden nicht bei Stufe 1 starten – wer ein gewachsenes PHP-Projekt übernimmt, landet meist mitten in Stufe 2 oder 3 und fragt sich, wie es von dort aus weitergeht. Das Buch greift diese Situation auf und nimmt jede Stufe einzeln auseinander: zuerst die Diagnose, dann die passenden Werkzeuge, dann der Sprung zur nächsten Stufe. Jedes Kapitel schließt mit einem klaren Lernziel und enthält mindestens ein lauffähiges Beispiel, das du in einem kleinen Repository nachstellen kannst.

Kapitel 2 bis 4 widmen sich der Grundlage: Composer, PHPUnit und Pest sowie PHPStan und PHP-CS-Fixer. Kapitel 5 und 6 behandeln zeitgesteuerte Aufgaben mit Cron und Robo. Kapitel 7 bis 9 bilden das Herzstück der Continuous-Integration-Praxis: Jenkins, GitLab CI/CD und GitHub Actions. Kapitel 10 und 11 zeigen, wie mit Deployer und Docker Zero-Downtime-Deployments gelingen. Kapitel 12 und 13 behandeln Betriebsaspekte, Kapitel 14 schließt mit konkreten Migrationspfaden.

Das komplette Buch (14 Kapitel, PDF) gibt es für 9,99 € unter phpautomation.watch-and-alert.de/bestellen.html.