Engineer

DevOps

∞ Praktiken, die Softwareentwicklung und Betrieb verbinden

Was es ist

DevOps ist eine Art, Softwarearbeit zu organisieren: Die Menschen, die ein System entwickeln, verantworten auch seinen Build, seine Freigabe und seinen laufenden Betrieb. Die Schritte zwischen dem Schreiben und dem Ausführen von Code werden automatisiert, statt sie von Team zu Team weiterzureichen. Der Name verbindet development und operations, also Entwicklung und Betrieb, die beiden Gruppen, die er zusammenbringen sollte.

NIST, das US-amerikanische National Institute of Standards and Technology, definiert DevOps als Praktiken zur Automatisierung der Prozesse zwischen Entwicklungs- und Betriebsteams, „damit sie Software schneller und zuverlässiger bauen, testen und freigeben können“. Diese Definition ist nützlich, weil sie den entscheidenden Teil benennt: die Prozesse dazwischen. Ein Unternehmen kann hervorragende Entwickler und hervorragende Betreiber haben und trotzdem Tage verlieren, wenn die einen ein Release an die anderen übergeben.

In der Praxis zeigt sich DevOps in einigen Gewohnheiten, die sich gegenseitig verstärken:

- **Gemeinsame Verantwortung für Produktion.** Wer den Code ändert, sieht durch Monitoring, Logs und Alarme, was die Änderung im Betrieb bewirkt, und steht dafür ein.
- **Ein automatisierter Weg zur Freigabe.** Build, Test und Bereitstellung laufen als Pipeline jedes Mal gleich ab, meist auf Grundlage von CI.
- **Infrastruktur in Dateien.** Server, Netzwerke und Datenbanken werden in Vorlagen oder Playbooks beschrieben, die neben dem Code liegen. Eine Umgebung lässt sich so neu aufbauen, statt sie aus dem Gedächtnis wiederherstellen zu müssen.

- **Ein Build, viele Umgebungen.** Das in Staging erfolgreich getestete Artefakt ist dasselbe, das Produktion erreicht. Nur seine Einstellungen ändern sich.
- **Kleine, häufige Releases.** Jedes enthält wenige Änderungen, sodass Fehler leicht zu finden und die Änderungen leicht zurückzunehmen sind.

DevOps ist kein Produkt und nur bedingt eine Berufsbezeichnung. Anbieter verkaufen Werkzeuge unter diesem Namen, und Unternehmen stellen „DevOps Engineers“ ein. Doch beides ist nicht die Praxis selbst. Ein DevOps Engineer baut und betreut üblicherweise die Pipelines und die Infrastruktur, mit denen andere Entwickler selbstständig ausliefern können. Das ist eine Rolle innerhalb der Praxis, kein Ersatz für sie.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist DevOps?“ 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 jeder Übergabe zwischen Teams geht etwas verloren. Der Entwickler weiß, warum eine Änderung vorgenommen wurde und wovon sie abhängt. Der Betreiber weiß, wie die Server aufgebaut sind und was letzten Monat kaputtging. Überquert ein Release als Ticket und Paket die Grenze zwischen beiden, kommt jede Hälfte dieses Wissens unvollständig an. Die Lücken werden bei der Bereitstellung sichtbar, also zum teuersten Entdeckungszeitpunkt.

Die Twelve-Factor App, eine Methodik für den Aufbau von Software-as-a-Service-Anwendungen, benennt drei solcher Lücken genau. Es gibt eine zeitliche Lücke, wenn heute geschriebener Code Wochen später Produktion erreicht. Es gibt eine personelle Lücke, wenn eine Gruppe den Code schreibt und eine andere ihn bereitstellt. Und es gibt eine Werkzeuglücke, wenn auf dem Laptop eine andere Datenbank läuft als in Produktion. DevOps ist zu einem großen Teil der Versuch, alle drei zu schließen: Stunden nach dem Schreiben bereitstellen, die Autoren selbst bereitstellen lassen und alle Umgebungen möglichst ähnlich halten.

Die Releasegröße ist das zweite Argument. Googles Buch über Site Reliability Engineering formuliert es klar: Häufige Releases bedeuten weniger Änderungen zwischen Versionen. Ein Release mit drei Änderungen lässt sich im Fehlerfall leicht überblicken. Eines mit dreihundert bedeutet eine Suche.

Das dritte Argument ist Toil. Dasselbe Buch definiert Toil als Arbeit für den Betrieb eines Dienstes, die manuell, wiederholend, automatisierbar und ohne dauerhaften Wert ist und mit dem Dienst mitwächst. Ein Team, das jeden Dienstag einen Prozess von Hand neu startet, Zertifikate nach einer Kalendererinnerung erneuert und bei einem Alarm die Größe von Festplatten anpasst, verbringt seine Zeit mit Arbeit, die ein Skript zuverlässiger erledigen würde. Durch ihre Automatisierung hält ein kleines Team mit einem wachsenden System Schritt.

Sicherheitsleitlinien sind bei derselben Struktur angekommen. NISTs Leitfaden zu DevSecOps für cloudnative Anwendungen geht von einer Pipeline mit Phasen für Build, Test, Paketierung, Bereitstellung und Betrieb aus. Infrastruktur, Laufzeitrichtlinien und Monitoring gehören dabei zu den Arten von Code, die diese Pipeline beschreibt.

Wie es in der Praxis aussieht

Die Seiten dieser Website zeigen DevOps weniger als Titel denn als Verantwortungsbereich: Dieselbe Ingenieurin entwirft ein System, bringt es in Produktion und hält es dort am Laufen.

Für ein Kartenprodukt umfasste die Arbeit der Gründerin zwei Jahre lang Infrastruktur, Datenverarbeitung und die Kartenanwendung für Kunden. Die Arbeit erstreckte sich über Architektur, Umsetzung und Support dieser Schichten: Konzeption, Aufbau und Pflege der Betriebsumgebung, der Datenverarbeitungsabläufe und der Kartenanwendung. Das verbindet Entwicklung und Betrieb innerhalb derselben Tätigkeit. Der Nachweis belegt Umfang und Dauer dieser Entwicklungs- und Wartungsarbeit, nicht Verfügbarkeits- oder Antwortzeitmessungen.

Bei einer Datenbanklandschaft mit 10 Instanzen von PostgreSQL und MS SQL Server auf VPS mit Ubuntu Linux wurde der Betrieb Schritt für Schritt automatisiert: Sicherungen über `pg_dump` und Jobs des SQL Server Agent, Monitoring in Prometheus und Grafana, regelmäßige Patches, UFW-Firewalls, rollenbasierte Zugriffskontrolle und dokumentierte Verfahren zur

Notfallwiederherstellung. Die Seite berichtet von 99,9 % Verfügbarkeit über alle zehn Server und einer durch Automatisierung um 50 % verringerten manuellen Wartung. Das ist das Toil-Argument, gemessen an einer Datenbanklandschaft.

Auf einer Azure-Plattform mit Entwicklungs-, Staging- und Produktionsumgebung wurde die Bereitstellung durchgängig entlang eines festen Wegs verantwortet. Dazu gehörte jedes Glied dieses Wegs: CI/CD baut die Images, Bicep-Vorlagen sorgen in jeder Phase für eine entsprechende Infrastruktur, und die jeweiligen Einstellungen werden getrennt von Geheimnissen gehalten. Ein gebautes Artefakt wird deshalb von Phase zu Phase weitergereicht, statt für jede neu gebaut zu werden. Änderungen erreichen jede Umgebung vorhersehbar, ohne eine einzige spontane manuelle Bereitstellung.

Wo es schiefgeht

DevOps scheitert leise, meist indem die Begriffe bleiben und die Gewohnheiten verloren gehen.

Der Titel kommt, die Wand bleibt. Ein Betriebsteam wird umbenannt, oder ein „DevOps-Team“ wird zwischen Entwicklern und Produktion eingestellt. Releases durchlaufen weiterhin eine Übergabe; nur der Name auf der anderen Seite ist neu. Der Test ist einfach: Fragen Sie, wer eine Änderung ausliefern kann und wer geweckt wird, wenn sie scheitert. Sind das verschiedene Menschen mit unterschiedlichen Informationen, ist die Praxis noch nicht angekommen.

Bereitstellung bleibt Handarbeit nach Checkliste. Die Pipeline baut und testet. Dann meldet sich jemand auf einem Server an und führt die letzten fünf Schritte aus dem Gedächtnis aus, jedes Mal etwas anders. Nehmen Sie auch die letzten Schritte in die Pipeline auf, geben Sie Releases einen einzigen Weg durch die Umgebungen und behandeln Sie einen manuellen Push in Produktion als erklärungsbedürftigen Vorfall.

Jede Umgebung bekommt einen eigenen Build. Ein Neubau für Staging und ein weiterer für Produktion bedeuten, dass das Getestete nicht das Ausgelieferte ist. Eine beim zweiten Build anders aufgelöste Abhängigkeit zeigt sich als Fehler, den niemand reproduzieren kann. Bauen Sie einmal, kennzeichnen Sie das Artefakt mit seinem Ursprungscommit und reichen Sie dieses Artefakt weiter. Die Einstellungen kommen aus der Umgebung:

```
tag="$(git rev-parse --short HEAD)"
docker build -t "registry.example.com/app:$tag" .
docker push "registry.example.com/app:$tag"
# staging and production run this same tag;
# only the settings they start it with differ
docker run --env-file staging.env "registry.example.com/app:$tag"
```

Staging entfernt sich von Produktion. Eine von Hand auf einem Server geänderte Einstellung, eine zuerst in Produktion aktualisierte Datenbank, eine an einer Stelle fixierte und anderswo ungebundene Bibliotheksversion: Alles macht Staging als Generalprobe etwas weniger nützlich. Beschreiben Sie die Infrastruktur in Vorlagen, wenden Sie dieselben Vorlagen überall an und betreiben Sie in der Entwicklung dieselben unterstützenden Dienste wie in Produktion.

Toil wächst mit dem System. Manuelle Aufgaben, die bei einem Server zehn Minuten pro Woche dauerten, benötigen bei zwanzig Servern Stunden. Ohne bewusst eingeplante Zeit für ihre Automatisierung verbringt das Team schließlich seine Tage damit, den Betrieb aufrechtzuerhalten. Zählen Sie die wiederkehrende manuelle Arbeit und begrenzen Sie ihren erlaubten Zeitanteil.

Weiterführende Lektüre

- DevOps — die NIST-Definition samt den Fachveröffentlichungen, aus denen sie stammt.
- Build, Release, Run — warum ein Build einmal erstellt und mit Konfiguration zu einem Release verbunden wird.
- Gleichheit von Entwicklung und Produktion — die zeitlichen, personellen und werkzeugbezogenen Lücken zwischen Entwicklung und Produktion und wie sie sich schließen lassen.
- Konfiguration — alles, was sich zwischen Bereitstellungen unterscheidet, außerhalb des Codes und in der Umgebung halten.
- Toil beseitigen — Googles Definition von Toil und die Argumente für eine Obergrenze.
- Release Engineering — hermetische Builds, Releases im Selfservice und durch Werkzeuge durchgesetzte Richtlinien.

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 DevOps?“ 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/devops/