Engineer

Blue-Green-Bereitstellung

◐ Live-Verkehr zwischen zwei identischen Umgebungen umschalten

Was es ist

Eine Blue-Green-Bereitstellung veröffentlicht Software mithilfe zweier möglichst gleich aufgebauter Produktionsumgebungen. Eine davon, hier Blau genannt, bedient jede Anfrage. Die neue Version wird auf der anderen, Grün, installiert und dort getestet, während noch kein Kundenverkehr sie erreicht. Sobald sie bereit ist, wird die vorgeschaltete Komponente so umgestellt, dass jede neue Anfrage an Grün geht. Blau läuft unverändert weiter. Verhält sich Grün fehlerhaft, bringt dieselbe Änderung in umgekehrter Richtung Blau wieder in Betrieb.

Die Farben haben keine eigene Bedeutung, und genau darum geht es. Bezeichnungen wie „alt“ und „neu“ oder „primär“ und „Standby“ tauschen ihre Bedeutung bei jeder Veröffentlichung. Zwei Farben wechseln sich dagegen einfach ab: Nach dem nächsten Umschalten ist Grün live und Blau untätig. Martin Fowler beschrieb das Verfahren 2010 in seinem Bliki und schreibt den Namen mit einer gewissen Unsicherheit Daniel Terhorst-North und Jez Humble zu. Die übliche englische Schreibweise enthält einen Bindestrich. Werkzeuge schreiben sie allerdings auf ihre eigene Weise: Argo Rollouts, ein Kubernetes-Controller für Bereitstellungsstrategien dieser Art, verwendet in seiner Konfiguration `blueGreen` .

Umschalten lässt sich an jeder einzelnen Stelle, die über das Ziel von Anfragen entscheidet. Fowlers Variante stellt einen Router um. Ebenso kann das Ziel eines Load Balancers, das Upstream-Ziel eines Reverse Proxys oder in Kubernetes ein Service geändert werden, dessen Labelselektor die Pods für seinen Verkehr auswählt. Die beiden Umgebungen müssen auch keine zwei Rechenzentren sein: Fowler nennt getrennte Hardware, getrennte virtuelle Maschinen oder zwei Zonen auf einem System mit jeweils eigenen IP-Adressen als gleichermaßen ausreichend.

Zwei benachbarte Verfahren verdeutlichen den Unterschied. Ein Rolling Update ersetzt jeweils wenige Instanzen. Alte und neue Version beantworten dabei nebeneinander Anfragen, bis die letzte alte Instanz verschwunden ist. Ein Rollback bedeutet, denselben schrittweisen Austausch in die Gegenrichtung auszuführen. Ein Canary Release schickt zunächst einen kleinen Verkehrsanteil an die neue Version und erhöht ihn schrittweise. Blue-Green verlagert alles auf einmal, hält aber eine vollständige, laufende Kopie der vorherigen Version bereit. Der Rückweg besteht deshalb aus einem Schritt, genauso groß wie der Hinweg.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine Blue-Green-Bereitstellung?“ 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

Eine Veröffentlichung wird zu einem kurzen, umkehrbaren Ereignis statt zu einem langen Vorgang in nur eine Richtung. Die langsamen Teile, Dateien kopieren, Prozesse starten und Caches aufwärmen, geschehen auf Grün, während Blau weiter Anfragen bedient. Kunden warten deshalb nicht darauf. Das Umschalten selbst geht schnell, ebenso die Rücknahme: Nichts wird neu gebaut oder installiert, weil die vorherige Version nie angehalten wurde. Das verändert die Bereitschaft eines Teams, an einem gewöhnlichen Nachmittag zu veröffentlichen, und die Geschwindigkeit, mit der es auf Auffälligkeiten reagieren kann.

Grün lässt sich außerdem unter nahezu produktionsgleichen Bedingungen prüfen, bevor Kunden es zu sehen bekommen: dasselbe Netzwerk, dieselbe Konfiguration, dieselbe Datenbank. Ein Smoke-Test gegen Grün beweist deutlich mehr als derselbe Test in einer Staging-Umgebung, die sich in einem Dutzend Kleinigkeiten unterscheidet. Argo Rollouts bildet dies mit einem Preview-Service ausdrücklich ab. Er leitet vor der Freigabe an die neue Version weiter, damit diese ohne Produktionsverkehr getestet werden kann.

Die untätige Umgebung hat eine zweite Aufgabe. Sobald sich eine Veröffentlichung bewährt hat, dient die gerade still gewordene Seite zur Vorbereitung der nächsten. Dazwischen ist sie ein betriebsbereiter Standby. Fowler weist darauf hin, dass derselbe Mechanismus „es Ihnen erlaubt, Ihr Verfahren zur Notfallwiederherstellung bei jeder Veröffentlichung zu testen“, weil jedes Umschalten ein beobachteter Failover ist.

Der Aufwand ist klar. Zwei Produktionsumgebungen benötigen die Kapazität zweier Umgebungen, zumindest während einer Veröffentlichung. Die Anwendungsschicht lässt sich leicht verdoppeln, die Daten dagegen nicht: Blau und Grün teilen fast immer eine Datenbank. Jede Veröffentlichung muss deshalb einen Datenstand hinterlassen, den beide Versionen lesen können. Ein Team, das dies nicht gewährleisten kann, erhält einen schnellen Hinweg und einen defekten Rückweg.

Wie es in der Praxis aussieht

Auf Kubernetes besteht die kleinste funktionsfähige Variante aus zwei Deployments und einem Service. Die Deployments `web-blue` und `web-green` führen dieselbe Anwendung mit den Labels `app: web` und `slot: blue` beziehungsweise `slot: green` aus. Der von Kunden erreichte Service wählt einen Slot aus:

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  selector:
    app: web
    slot: blue
  ports:
    - port: 80
      targetPort: 8080
```

Eine Veröffentlichung bringt das neue Image auf die untätige Seite, wartet auf deren Bereitschaft und testet sie über einen zweiten Service, der ausschließlich `slot: green` auswählt. Danach wird der aktive Service umgestellt:

```
kubectl set image deployment/web-green web=registry.example.com/web:1.5.0
kubectl rollout status deployment/web-green --timeout=5m
# smoke tests against the web-green preview Service go here
kubectl patch service web -p '{"spec":{"selector":{"slot":"green"}}}'
# the way back is the same command with the other colour
kubectl patch service web -p '{"spec":{"selector":{"slot":"blue"}}}'
```

`kubectl patch` verwendet standardmäßig einen Strategic Merge Patch. Deshalb ändert sich nur der Schlüssel `slot` des Selektors, während `app: web` erhalten bleibt. Der Service-Controller sucht fortlaufend nach Pods, die zum Selektor passen, und aktualisiert die Endpunkte des Services, wenn sich diese Menge ändert. Neue Verbindungen landen dadurch auf Grün, während Blau mit dem vorherigen Image bereitsteht.

Argo Rollouts automatisiert dieselbe Choreografie mit einem `Rollout` -Objekt: einem `activeService` , den es bei der Freigabe auf die neuen Pods richtet, und einem optionalen `previewService` , der sie schon vorher erreicht. Mit `autoPromotionEnabled: false` hält der Rollout vor dem Umschalten an, bis jemand `kubectl argo rollouts promote web` ausführt.

Die Leistung DevOps und CI/CD verspricht getestete, veröffentlichte und rückgängig zu machende Änderungen. Eine Blue-Green-Bereitstellung ist eine Möglichkeit, die Rücknahme mit wenig Aufwand zu ermöglichen. Sie beruht auf zwei Gewohnheiten, die sich bei Bereitstellungen einer Plattform nach Azure bereits zeigen, dort ohne einen zweiten Produktionsslot: Bicep-Vorlagen halten die Zielinfrastruktur von einer Umgebung zur nächsten identisch. Dasselbe gebaute Artefakt wird von Entwicklung über Staging in die Produktion übernommen und erhält am jeweiligen Ziel dessen Einstellungen. Dieser Weg führt eine Version nacheinander durch die Umgebungen. Blue-Green ergänzt am Ende einen Zwilling. Dieselben beiden Gewohnheiten, identische Infrastruktur und ein unverändertes Artefakt, erlauben es einem Team, sich auf die tatsächliche Gleichheit dieses Zwillings zu verlassen.

Wo es schiefgeht

Eine Schemaänderung macht den Rückweg zur Sackgasse. Grün startet, führt eine Migration zur Umbenennung einer Spalte aus und bedient den Verkehr problemlos. Zehn Minuten später führt ein Fehler den Verkehr zurück zu Blau. Diese Version fragt weiterhin nach dem alten Spaltennamen und scheitert bei jeder Anfrage. Der Rollback wurde für die Anwendung geplant, aber nicht für die

Daten. Fowlers Rat von 2015 trennt beides: Zuerst eine Datenbankänderung ausliefern, mit der beide Versionen arbeiten können, dann die Anwendung. Was nur die alte Version brauchte, wird entfernt, sobald sich die neue bewährt hat.

Auf Grün geschriebener Zustand bleibt beim Zurückschalten zurück. Alles, was Grün für sich behält, etwa auf die lokale Platte hochgeladene Dateien, ein Sitzungsspeicher im Arbeitsspeicher oder ein eigener Cache anstehender Aufträge, fehlt auf Blau, wenn der Verkehr dorthin zurückkehrt. Nutzer sind plötzlich abgemeldet oder finden gerade Gespeichertes nicht mehr. Halten Sie gemeinsamen Zustand in Speichern, die beide Farben verwenden. Muss Grün eigene Daten besitzen, schlägt Fowler vor, Transaktionen an beide Umgebungen zu liefern oder Grün zunächst nur lesend zu starten und erst nach seiner Bewährung auf Lesen und Schreiben umzustellen.

Der Rückweg wird wegskaliert, bevor der Fehler sichtbar wird. Argo Rollouts skaliert das alte ReplicaSet nach dem Umschalten herunter. `scaleDownDelaySeconds` hat den Standardwert 30. Ein Fehler, der nach einer Viertelstunde auftritt, ein nächtlicher Auftrag oder ein schleichendes Leck, findet kein Blau mehr vor, zu dem zurückgekehrt werden könnte. Stellen Sie die Verzögerung so ein, dass sie die übliche Zeit bis zum Sichtbarwerden von Problemen abdeckt. Akzeptieren Sie dabei, dass die benötigte Kapazität der Preis dafür ist:

```
strategy:
  blueGreen:
    activeService: web
    previewService: web-preview
    autoPromotionEnabled: false
    scaleDownDelaySeconds: 3600
```

Die Freigabe folgt der Bereitschaft statt der Korrektheit. `autoPromotionEnabled` steht standardmäßig auf true. Argo Rollouts schaltet deshalb um, sobald die neuen Pods bereit sind. Eine Readiness-Probe sagt, dass ein Prozess antwortet, nicht dass der Bezahlvorgang funktioniert. Pausieren Sie vor der Freigabe wie im obigen Block und testen Sie über den Preview-Service. Alternativ ergänzen Sie eine `prePromotionAnalysis`, die den Rollout abbricht, wenn sie fehlschlägt, und eine `postPromotionAnalysis`, die zur vorherigen Version zurückwechselt, wenn sie nach dem Umschalten fehlschlägt.

Die untätige Seite driftet ab. Zwischen Veröffentlichungen spielt jemand eine dringende Konfigurationsänderung nur in die aktive Umgebung ein. Die nächste Veröffentlichung landet auf der untätigen Seite mit der alten Einstellung. Das Umschalten, das eine Sache ändern sollte, ändert nun zwei. Bauen Sie beide Slots aus denselben Vorlagen und derselben Konfigurationsquelle, damit eine Änderung an einer Seite beide betrifft. Behandeln Sie die untätige Seite als Vorbereitungsbereich für die nächste Veröffentlichung, wie Fowler es beschreibt.

Weiterführende Lektüre

- Blue Green Deployment — Martin Fowlers Beschreibung von 2010 mit seinem Hinweis von 2015 zu Datenbankänderungen.
- Argo Rollouts: BlueGreen — aktive und Preview-Services, automatische Freigabe, Analyse vor und nach dem Umschalten sowie die Verzögerung beim Herunterskalieren.
- Kubernetes Service — wie ein Selektor die Pods auswählt, die den Verkehr eines Services erhalten.
- kubectl patch — Strategic Merge, Merge und JSON Patches sowie der Befehl, der den Selektor im obigen Beispiel umstellt.

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 Blue-Green-Bereitstellung?“ 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/blue-green-deployment/