Engineer

Canary Release

eine Änderung erst einem kleinen Nutzeranteil bereitstellen

Was es ist

Ein Canary Release setzt eine neue Version eines Dienstes zunächst einem kleinen Anteil seines echten Verkehrs aus. Die bisherige Version bedient währenddessen alle anderen. Bewährt sich die neue Version, wächst der Anteil schrittweise, bis sie alles übernimmt und die alte Version außer Betrieb geht. Bewährt sie sich nicht, geht der Verkehr zurück zur alten Version, die nie aufgehört hat zu laufen. Nur der kleine Anteil hat den Fehler gesehen. Danilo Sato beschrieb das Verfahren im Juni 2014 in Martin Fowlers Bliki und nannte auch die Bezeichnungen Phased Rollout und Incremental Rollout. Kubernetes-Werkzeuge sprechen meist von Canary Deployment: Argo Rollouts, ein Kubernetes-Controller für Veröffentlichungsstrategien dieser Art, nennt die Einstellung `canary`.

Der Name stammt aus dem Kohlebergbau. Bergleute nahmen einen Kanarienvogel im Käfig mit unter Tage, weil austretendes Gas den Vogel früher überwältigte als die Menschen. Die Not des Vogels war die Warnung, hinauszugehen. Der kleine Verkehrsanteil übernimmt dieselbe Rolle: Er trifft zuerst auf das Problem, solange noch Zeit zum Handeln bleibt.

Welche Nutzer zu diesem Anteil gehören, lässt sich auswählen. Sato nennt eine zufällige Auswahl von Nutzern, die eigenen Mitarbeiter eines Unternehmens vor der Öffentlichkeit und nach Profil oder demografischen Merkmalen ausgewählte Nutzer. Große Systeme teilen stattdessen häufig entlang bestehender Grenzen auf: zuerst eine Region, ein Standort oder eine Marke. Facebook verwendete laut Sato mehrere Canarys. Der erste war nur für eigene Mitarbeiter sichtbar und hatte sämtliche Funktionsflags eingeschaltet, damit Probleme neuer Funktionen früh auffielen.

Der Unterschied zur Blue-Green-Bereitstellung liegt im Ablauf. Blue-Green baut eine vollständige zweite Kopie und verlagert den gesamten Verkehr mit einem Umschalten dorthin, während die alte Kopie als Rückweg bleibt. Ein Canary verlagert den Verkehr allmählich. Eine Zeit lang bedienen deshalb zwei Versionen nebeneinander echte Nutzer. Auch ein Rolling Update hat zwei Versionen gleichzeitig in Betrieb, ersetzt Instanzen aber so schnell, wie sie bereit werden, ohne Pause zur Bewertung der neuen Version. Ein Canary hält bei jedem Schritt an und wartet auf eine Antwort.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Canary Release?“ 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

Manche Fehler zeigen sich erst in Produktion: unter echter Last, mit echten Daten und hinter der tatsächlichen Mischung von Clients. Eine Staging-Umgebung kann nahe herankommen, ist aber nie identisch. Ein Canary akzeptiert das und begrenzt stattdessen den Schaden. Erreicht eine von zwanzig Anfragen die neue Version, beeinträchtigt eine fehlerhafte Veröffentlichung eine von zwanzig Anfragen, bis sie bemerkt wird. Die Abhilfe besteht darin, diesen Anteil zurückzuschicken.

Sato nennt einen zweiten Nutzen: Kapazitätstests. Den Anteil langsam zu erhöhen zeigt, wie sich die neue Version unter wachsender Produktionslast verhält, mit einem sicheren Rückweg. Dafür muss keine eigene Lasttestumgebung gebaut werden, die nie ganz so produktionsnah sein kann wie die Produktion selbst.

Der Haken: Ein Canary ist nur so gut wie das Signal, das ihn bewertet. Fehler, Latenz und wichtige Geschäftszahlen wie Registrierungen, Zahlungen oder Bestellungen müssen pro Version sichtbar sein, und zwar früh genug, um den nächsten Schritt zu stoppen. Ohne das ist ein Canary ein langsamer Rollout mit zusätzlichem Warten. Sato merkt an, dass IMVU beim sogenannten „Cluster-

Immunsystem“ Pionierarbeit leistete: Geschäftsmetriken beobachten und eine Veröffentlichung bei einer statistisch signifikanten Verschlechterung automatisch zurückrollen.

Hinzu kommen Betriebskosten. Zwei oder mehr Versionen sind gleichzeitig im Einsatz. Alles, was sie teilen, die Datenbank, ein Cache, eine Warteschlange und die von Clients aufgerufene API, muss währenddessen mit beiden funktionieren. Sato rät, die Zahl gleichzeitig aktiver Versionen möglichst klein zu halten. Das Verfahren passt auch schlecht zu Software, die auf den eigenen Rechnern oder Telefonen der Nutzer installiert ist. Dort bestimmt der Anbieter nicht den Zeitpunkt einer Aktualisierung. Sato empfiehlt, das Backend weiter für beide Versionen bereitzuhalten, die noch verwendeten Clientversionen zu beobachten und die alte Version erst aufzugeben, wenn ihr Anteil weit genug gefallen ist.

Wie es in der Praxis aussieht

In der einfachsten Form besteht ein Canary auf Kubernetes aus zwei Services. Einer wählt die stabilen Pods aus, der andere die Canary-Pods. Eine Route verteilt den Verkehr mit Gewichten zwischen beiden. Bei der Gateway API stehen diese Gewichte in einer HTTPRoute:

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: web
spec:
  parentRefs:
    - name: prod-web
  hostnames:
    - "www.example.com"
  rules:
    - backendRefs:
        - name: web-stable
          port: 8080
          weight: 95
        - name: web-canary
          port: 8080
          weight: 5
```

Gewichte sind Verhältnisse, keine Prozentwerte: Der Verkehrsanteil jedes Backends entspricht seinem Gewicht geteilt durch die Summe der Gewichte dieser Regel. 95 und 5 schicken deshalb eine von zwanzig Anfragen an den Canary. Ein ausgelassenes Gewicht zählt als 1, ein Gewicht von 0 schickt nichts.

Die Anleitung der Gateway API zur Verkehrsaufteilung zeigt außerdem eine Regel, die einen Anfrageheader abgleicht, im Beispiel `traffic: test`, und nur diese Anfragen zur neuen Version leitet. Tester erreichen sie so vor allen Kunden. Den Canary voranzubringen heißt, zwei Zahlen zu ändern. Ihn zurückzunehmen heißt, sein Gewicht auf 0 zu setzen.

Argo Rollouts automatisiert die Schritte. Ein `Rollout` ersetzt ein `Deployment`, mit derselben Pod-Vorlage. Er führt die Schritte auf, die der Controller bei jeder Änderung dieser Vorlage durchläuft:

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: web
spec:
  replicas: 10
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
  strategy:
    canary:
      maxSurge: "25%"
      maxUnavailable: 0
      steps:
        - setWeight: 10
        - pause:
            duration: 1h
        - setWeight: 20
        - pause: {}
```

Ohne Verkehrsrouter nähert Argo das Gewicht über die Zahl der Pods an: Zehn Replikate bei 10 Prozent bedeuten einen Canary-Pod und neun stabile Pods. Die erste Pause dauert eine Stunde, die zweite so lange, bis jemand entscheidet. Eine Veröffentlichung und ihre beiden möglichen Ausgänge sehen so aus:

```
kubectl argo rollouts set image web web=registry.example.com/web:1.5.0
# at the open-ended pause, either carry on ...
kubectl argo rollouts promote web
# ... or stop and put the stable version back in front of everyone
kubectl argo rollouts abort web
```

Die gesamte Folge hängt davon ab, während der Pause zu wissen, ob der Canary schlechter arbeitet als sein Vorgänger. Dahinter steht die Leistung Site Reliability & Monitoring, die auf Alarmen beruht, die sich melden, solange ein Fehler noch klein ist. Die plattformeigene Lösung für Fehlermonitoring und Tracing aus der seit 2025 laufenden Tätigkeit zeigt ein solches Signal: Fehler werden zu wiederkehrenden Problemen zusammengefasst statt einzeln aufgelistet. Spitzen und Regressionen werden mit einer Abklingzeit erkannt, damit eine schlechte Bereitstellung niemanden vierzigmal alarmiert. Genau das braucht ein Canary-Schritt: Eine Verschlechterung muss innerhalb der Pause sichtbar werden, nicht erst am nächsten Morgen zufällig einem Nutzer auffallen.

Wo es schiefgeht

Der Canary ist zu klein, um etwas auszusagen. Ein Pod von zehn kann zu einer ruhigen Stunde während der Pause nur wenige Hundert Anfragen bedienen. Ein Fehler, der eine von tausend trifft, zeigt sich dabei nie. Die Uhr ist das falsche Maß; die Zahl der vom Canary bedienten Anfragen ist das richtige. Stimmen Sie Anteil und Pause gemeinsam ab, damit ein seltener Fehler die Chance bekommt, mehrmals aufzutreten. Halten Sie einen Schritt nachts länger als mittags.

Die Dashboards unterscheiden Canary und stabile Version nicht. Eine Canary-Version, bei der eine von fünfzig Anfragen mehr scheitert als bei der stabilen Version, erhöht bei einem Anteil von 5 Prozent die Gesamtzahl um ungefähr eine fehlgeschlagene Anfrage pro tausend. Auf einem Dashboard mit nur einer Fehlerrate für den ganzen Dienst bleibt das unsichtbar. Geben Sie Canary- und stabilen Pods unterschiedliche Labels. Argo Rollouts fügt sie über `canaryMetadata` und `stableMetadata` hinzu. Übernehmen Sie das Label in die Metriken und vergleichen Sie den Canary mit der stabilen Version statt mit einem festen Grenzwert. Eine im Hintergrund laufende `analysis` des Rollouts bricht die Aktualisierung ab, wenn diese Analyse fehlschlägt.

Beide Versionen schreiben dieselben Daten. Während des gesamten Rollouts lesen und schreiben alter und neuer Code dieselbe Datenbank. Bei zufälligem Routing können die Anfragen eines Nutzers binnen einer Minute bei beiden

landen. Eine neue Spalte, die die alte Version nie füllt, oder ein Format, das sie nicht lesen kann, verursacht Fehler auf der stabilen Seite, die eigentlich sicher sein sollte. Satos Antwort heißt parallele Änderung: Die Datenbank unterstützt während des gesamten Rollouts beide Anwendungsversionen. Die alte Form verschwindet erst, wenn die alte Version weg ist.

Ein Abbruch stoppt den Verkehr, nicht die Änderung. `kubectl argo rollouts abort` verschiebt den Verkehr zurück zum stabilen ReplicaSet. Die Spezifikation des Rollouts nennt aber weiterhin das neue Image. Wird der Rollout fortgesetzt, wird dieses Image erneut zum Ziel. Der Rollback ist erst vollständig, wenn die Vorlage wieder auf die vorherige Version gesetzt ist. Liegt die Spezifikation in einem Repository, ändern Sie sie dort, damit Quelle und Cluster darüber übereinstimmen, was laufen soll.

Der Canary dient zugleich als A/B-Test. Das Routing sieht gleich aus. Es liegt deshalb nahe, mit dem Canary-Verkehr die Wirkung einer neuen Funktion auf Conversions zu messen. Sato empfiehlt, beides zu trennen: Ein Canary sucht Probleme und sollte nach Minuten oder Stunden abgeschlossen sein. Ein Experiment prüft dagegen eine Hypothese und braucht womöglich tagelange Daten, um statistisch signifikant zu sein. Wird der Canary anhand von Geschäftsmetriken beurteilt, trübt die vom Experiment beabsichtigte Wirkung auch das Urteil. Veröffentlichen Sie mit dem Canary und führen Sie das Experiment nach der Veröffentlichung mit einer eigenen Verkehrsaufteilung aus, etwa hinter einem Funktionsflag.

Weiterführende Lektüre

- Canary Release — Danilo Satos Beschreibung von 2014 in Martin Fowlers Bliki: Nutzerauswahl, Kapazitätstests, A/B-Tests und die Datenbank.
- Argo Rollouts: Canary — Schritte und Pausen, durch Podzahlen angenäherte Gewichte und Skalierung mit einem Verkehrsrouter.
- HTTP-Verkehr aufteilen — die Anleitung der Gateway API zu gewichteten Backends und einer headerabhängigen Canary-Route.
- kubectl argo rollouts abort — was ein Abbruch mit dem Verkehr macht und was er in der Spezifikation belässt.

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 ein Canary Release?“ 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/canary-release/