Engineer

Канарковий реліз

випуск зміни спершу для невеликої частки користувачів

Що це таке

Канарковий реліз — це спосіб випускати нову версію сервісу так, щоб спершу вона обслуговувала лише невелику частку справжнього трафіку, а версія, яку вона замінює, тим часом обслуговувала всіх інших. Якщо нова версія працює як слід, частку крок за кроком збільшують, доки вона не обслуговуватиме весь трафік, а стару версію можна буде вивести з роботи. Якщо ні, трафік повертається на стару версію, яка весь цей час працювала, і збій зачепив лише ту невелику частку. Даніло Сато описав цей прийом у блозі-вікі Мартіна Фаулера в червні 2014 року й зазначив, що його ще називають поетапним (phased) або інкрементним (incremental) розгортанням. У світі Kubernetes частіше кажуть canary deployment, а українською — канаркове розгортання. Argo Rollouts, контролер Kubernetes, що додає такі стратегії розгортання, називає відповідне налаштування `canary`.

Назва походить із вугільних шахт. Шахтарі брали з собою під землю канарку в клітці, бо від витоку газу птах слабшав раніше, ніж його вплив відчували люди, і це було для них сигналом вчасно вибиратися нагору. Невелика частка трафіку відіграє ту саму роль: вона першою натрапляє на проблему, поки ще є час діяти.

Склад цієї частки — питання вибору. Сато називає випадкову вибірку користувачів, спершу власних працівників компанії, а вже потім усіх інших, і користувачів, відібраних за профілем чи демографічними ознаками. Великі системи часто ділять трафік за вже наявними межами: спершу один регіон, одне місце або один бренд. За словами Сато, Facebook запускав кілька канаркових груп, і першу з них бачили лише працівники компанії, причому з усіма ввімкненими прапорцями функцій, щоб проблеми з новими можливостями виявлялися якомога раніше.

Від синьо-зеленого розгортання канарковий реліз відрізняється тим, як рухається трафік. Синьо-зелене розгортання будує повну другу копію й переносить увесь трафік одним перемиканням, тримаючи стару копію як шлях назад. Канарковий реліз переносить трафік поступово, тож певний час дві версії обслуговують справжніх користувачів поруч. Поступове оновлення (rolling update) теж якийсь час тримає в роботі дві версії, але замінює екземпляри так швидко, як вони стають готовими, без пауз, щоб оцінити, як справляється нова версія. Канарковий реліз зупиняється на кожному кроці й чекає відповіді.

Застосуйте поняття до рішення

Застосуйте «Що таке канарковий реліз?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Перевірено, випущено і з можливістю відкату, тож зміни доходять до клієнтів у вівторок удень, а не пізно в п'ятницю.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Автоматизувала розгортання GIS SaaS через CI/CD — розгортання, дані та звітність — усе без ручної роботи

Автоматизувала розгортання GIS SaaS через CI/CD Зв'язатися з нами →

Чому це важливо

Деякі збої проявляються лише в продакшні: під справжнім навантаженням, на справжніх даних і з тим набором клієнтів, який є насправді. Середовище staging може бути дуже схожим, але ніколи не буде таким самим. Канарковий реліз це визнає й натомість обмежує шкоду. Якщо до нової версії потрапляє кожен двадцятий запит, поганий реліз шкодить кожному двадцятому запиту доти, доки його не помітять, а щоб це виправити, досить повернути цю частку назад.

Сато вказує ще на одну користь: перевірку потужності. Повільно збільшуючи частку, можна побачити, як нова версія поводиться під зростаючим продакшн-навантаженням, маючи безпечний шлях назад, замість будувати окреме середовище лише для навантажувальних тестів, яке ніколи не буде настільки схожим на продакшн, як сам продакшн.

Заковика в тому, що канарковий реліз вартий рівно стільки, скільки сигнал, за яким його оцінюють. Помилки, затримки й бізнесові показники, що справді мають значення, як-от реєстрації, платежі чи замовлення, мають бути видимі окремо для кожної версії й досить рано, щоб зупинити наступний крок. Без цього канарковий реліз — це просто повільне розгортання з

додатковим очікуванням. Сато зазначає, що IMVU першою застосувала так звану «cluster immune system», тобто імунну систему кластера: стежити за бізнесовими показниками й автоматично відкочувати реліз у разі статистично значущої регресії.

Є й постійні витрати. Дві або більше версій працюють одночасно, і все спільне для них (база даних, кеш, черга, API, який викликають клієнти) має працювати з обома, поки триває розгортання. Сато радить тримати кількість одночасних версій мінімальною. Цей прийом також погано підходить для програм, установлених на власних комп'ютерах чи телефонах користувачів, де постачальник не вирішує, коли відбудеться оновлення. Для таких випадків він радить зберігати на бекенді підтримку обох версій, стежити, які версії клієнтів досі в ужитку, і прибирати стару лише тоді, коли її частка стане достатньо малою.

Як це виглядає на практиці

У найпростішому вигляді канарковий реліз у Kubernetes — це два Service, один з яких обирає стабільні поди, а другий — канаркові, і маршрут, що розподіляє трафік між ними за вагами. У Gateway API ваги задають у 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
```

Ваги — це пропорції, а не відсотки: кожен бекенд отримує свою вагу, поділену на суму ваг у правилі, тож 95 і 5 спрямовують на канаркову версію кожен двадцятий запит. Не вказана вага дорівнює 1, а вага 0 не спрямовує

нічого. Посібник Gateway API з розподілу трафіку також показує правило, яке зіставляє заголовок запиту (у прикладі це `traffic: test`) і спрямовує на нову версію лише такі запити, тож тестувальники можуть дістатися до неї раніше за будь-якого клієнта. Щоб просунути канаркову версію далі, досить змінити два числа, а для відкликання — виставити їй вагу 0.

Argo Rollouts автоматизує ці кроки. `Rollout` займає місце `Deployment` з тим самим шаблоном подів і перелічує кроки, які контролер проходить щоразу, коли шаблон змінюється:

```
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: {}
```

Без маршрутизатора трафіку Argo наближено відтворює вагу кількістю подів: за ваги 10 відсотків десять реплік дають один канарковий под і дев’ять стабільних. Перша пауза триває годину, друга — доки хтось не ухвалить рішення. Реліз і два його можливі завершення виглядають так:

```
kubectl argo rollouts set image web web=registry.example.com/web:1.5.0
# на паузі без терміну: або далі ...
kubectl argo rollouts promote web
# ... або зупинитися й знову віддати весь трафік стабільній версії
kubectl argo rollouts abort web
```

Усе в цій послідовності залежить від того, чи видно під час паузи, що канаркова версія працює гірше за ту, яку замінює. Саме на цьому побудовано послугу Надійність та моніторинг (SRE), що зводиться до сповіщень, які попереджають, поки збій ще малий. Власний моніторинг помилок і трасування 2025 року показує, як виглядає такий сигнал: помилки групуються в повторювані проблеми замість плаского списку, а сплески й регресії виявляються з паузою між сповіщеннями, щоб через одне невдале розгортання чергового не піднімали сорок разів. Саме цього потребує крок канаркового релізу: регресії, що сама дає про себе знати в межах паузи, а не тієї, на яку користувач наткнеться наступного ранку.

Де виникають проблеми

Канаркова версія замала, щоб щось показати. Один под із десяти в тихі години може за свою паузу обслужити лише кілька сотень запитів, і збій, що зачіпає один запит із тисячі, так і не проявиться. Годинник — неправильна мірка; правильна — кількість запитів, які канаркова версія обслужила. Добирайте частку й паузу разом, щоб рідкісна помилка мала шанс проявитися кілька разів, і тримайте крок уночі довше, ніж опівдні.

Дашборди не відрізняють канаркову версію від стабільної. Канаркова версія, чий рівень помилок на два відсоткові пункти вищий, ніж у стабільної, за частки у 5 відсотків додає до загального рівня помилок лише приблизно одну помилку на тисячу запитів. На дашборді з одним рівнем помилок для всього сервісу цього не видно. Дайте канарковим і стабільним подам різні мітки, які Argo Rollouts додає через `canaryMetadata` і `stableMetadata`, перенесіть мітку в показники й порівнюйте канаркову версію зі стабільною, а не з фіксованим порогом. Фоновий `analysis` на об’єкті Rollout перериває оновлення, якщо аналіз провалюється.

Обидві версії пишуть у ті самі дані. Поки триває розгортання, старий і новий код читають і пишуть одну базу даних, а за випадкового розподілу запити одного користувача можуть протягом хвилини потрапити на обидві. Новий стовпець, який стара версія ніколи не заповнює, або формат, якого

вона не вміє прочитати, перетворюються на помилки на стабільному боці, тобто там, де мало бути безпечно. Відповідь Сато — parallel change, паралельна зміна: база даних підтримує обидві версії застосунку протягом усього розгортання, а стару структуру прибирають лише тоді, коли старої версії вже немає.

Переривання зупиняє трафік, а не зміну. `kubectl argo rollouts abort` повертає трафік на стабільний ReplicaSet, але специфікація об'єкта `Rollout` досі вказує на новий образ, і якщо об'єкт вийде з перерваного стану, він знову рушить до цієї версії. Відкат завершено лише тоді, коли шаблон повернуто до попередньої версії. Якщо специфікація зберігається в репозиторії, виправляйте її там, щоб джерело й кластер погоджувалися, що саме має працювати.

Канаркову версію заодно використовують як A/B-тест. Маршрутизація однакова, тож кортить виміряти вплив нової функції на конверсію за допомогою канаркового трафіку. Сато радить тримати ці речі окремо: канарковий реліз шукає проблеми й має завершуватися за хвилини чи години, а експеримент перевіряє гіпотезу й може потребувати даних за кілька днів, щоб результат став статистично значущим. Якщо канаркову версію оцінюють за бізнесовими показниками, ефект, який мав би викликати експеримент, ще й затьмарює вердикт. Випускайте версію через канарковий реліз, а експеримент запускайте вже після її виходу, з окремим розподілом трафіку, наприклад через прапорець функції.

Що почитати далі

- Canary Release — опис, який Даніло Сато 2014 року опублікував у блозі-вікі Мартіна Фаулера: вибір користувачів, перевірка потужності, A/B-тести й база даних.
- Argo Rollouts: Canary — кроки й паузи, ваги, наближені кількістю подів, і масштабування з маршрутизатором трафіку.
- HTTP traffic splitting — посібник Gateway API про бекенди з вагами й канарковий маршрут, що спрацьовує за заголовком.
- kubectl argo rollouts abort — що переривання робить із трафіком і що лишає в специфікації.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке канарковий реліз?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/canary-release/