Engineer

publicación canario

probar una versión con una parte pequeña del tráfico real

Qué es

Una publicación canario presenta primero una versión nueva de un servicio a una pequeña parte de su tráfico real, mientras la versión que sustituye sigue atendiendo al resto. Si la recién llegada funciona bien, su proporción aumenta por etapas hasta que recibe todo el tráfico y se retira la versión antigua. Si no, el tráfico vuelve a la versión anterior, que nunca dejó de funcionar, y solo aquella pequeña parte habrá visto el fallo. Danilo Sato describió la técnica en el bliki de Martin Fowler en junio de 2014 y señaló que también se conoce como despliegue por fases o despliegue incremental. Las herramientas de Kubernetes suelen hablar de despliegue canario: Argo Rollouts, un controlador de Kubernetes que añade estrategias de publicación de este tipo, denomina `canary` a la opción.

El nombre procede de la minería del carbón. Los mineros bajaban un canario en una jaula porque una fuga de gas incapacitaría al ave antes de alcanzar a las personas, y sus señales de malestar eran el aviso para salir. La pequeña parte del tráfico cumple esa misma función: se encuentra antes con el problema, cuando todavía hay tiempo para actuar.

Elegir quién entra en esa parte es una decisión. Sato enumera una muestra aleatoria de usuarios, el personal de la propia empresa antes que el público y usuarios seleccionados por perfil o características demográficas. Los sistemas grandes suelen dividir primero por límites que ya existen: una región, una ubicación o una marca. Según el relato de Sato, Facebook utilizaba varios canarios; el primero solo era visible para sus propios empleados y tenía activadas todas las banderas de funcionalidad, de modo que los problemas con funciones nuevas aparecían pronto.

La diferencia respecto al despliegue azul-verde está en la forma de hacer el cambio. El azul-verde construye una segunda copia completa y mueve todo el tráfico de una vez, conservando la copia antigua como vía de regreso. Un canario mueve el tráfico de forma gradual, por lo que durante un tiempo dos versiones atienden a usuarios reales en paralelo. Una actualización progresiva también mantiene dos versiones activas a la vez, pero sustituye las instancias en cuanto están listas, sin detenerse para evaluar cómo funciona la nueva versión. Un canario se detiene en cada etapa y espera una respuesta.

Utilice el concepto en una decisión

Aplique «¿Qué es una publicación canario?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cambios probados, publicados y reversibles, para que lleguen a sus clientes un martes por la tarde en vez de un viernes por la noche.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Despliegue GIS SaaS con GitHub Actions — despliegue, datos e informes automatizados

Despliegue GIS SaaS con GitHub Actions Póngase en contacto →

Por qué importa

Algunos fallos solo aparecen en producción: con carga real, datos reales y la combinación real de clientes. Un entorno de preproducción puede aproximarse, pero nunca es idéntico. Un canario lo acepta y limita los daños. Si una de cada veinte solicitudes llega a la versión nueva, una publicación defectuosa perjudica a una de cada veinte durante el tiempo que se tarde en detectarla, y la solución es devolver esa parte a la anterior.

Sato señala un segundo uso: las pruebas de capacidad. Aumentar la proporción lentamente muestra cómo se comporta la versión nueva bajo una carga de producción creciente, con una vuelta atrás segura, en lugar de construir un entorno separado solo para pruebas de carga, que nunca puede parecerse tanto a producción como la propia producción.

La dificultad es que un canario solo resulta tan útil como la señal que lo evalúa. Los errores, la latencia y las cifras de negocio relevantes, como registros, pagos o pedidos, deben ser visibles por versión y con suficiente antelación para detener la etapa siguiente. Sin eso, un canario es un despliegue lento con esperas añadidas. Sato señala que IMVU fue pionera en lo que se conoce como

un «sistema inmunitario del clúster»: observar las métricas de negocio y revertir automáticamente una publicación cuando hay una regresión estadísticamente significativa.

También hay costes operativos. Dos versiones o más están en servicio a la vez, y todo lo que comparten, la base de datos, una caché, una cola o la API a la que llaman los clientes, tiene que funcionar con ambas durante ese tiempo. Sato aconseja mantener al mínimo el número de versiones activas simultáneamente. La técnica también encaja mal con software instalado en los propios ordenadores o teléfonos de los usuarios, donde el proveedor no decide cuándo se actualiza. En ese caso, propone mantener el backend capaz de atender a ambas versiones, observar qué versiones del cliente siguen utilizándose y retirar la antigua solo cuando su proporción haya caído lo suficiente.

Cómo se aplica en la práctica

En su forma más sencilla, un canario en Kubernetes consta de dos Services, uno que selecciona los pods estables y otro que selecciona los pods canario, y una ruta que reparte el tráfico entre ambos mediante pesos. Con Gateway API, los pesos se definen en un 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
```

Los pesos son proporciones, no porcentajes: cada backend recibe su peso dividido por la suma de los pesos de esa regla, por lo que 95 y 5 envían una de cada veinte solicitudes al canario. Un peso omitido cuenta como 1, y un peso de 0 no envía nada. La guía de Gateway API sobre reparto de tráfico también muestra

una regla que coincide con una cabecera de solicitud, `traffic: test` en su ejemplo, y envía solo esas solicitudes a la versión nueva, de modo que quienes prueban puedan acceder antes que cualquier cliente. Hacer avanzar el canario consiste en editar dos números; retirarlo, en poner su peso a 0.

Argo Rollouts automatiza las etapas. Un `Rollout` sustituye a un `Deployment`, con la misma plantilla de pod, y enumera los pasos que recorre el controlador cada vez que cambia esa plantilla:

```
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: {}
```

Sin un enrutador de tráfico, Argo aproxima el peso mediante el número de pods: diez réplicas al 10 % significan un pod canario y nueve estables. La primera pausa espera una hora; la segunda espera hasta que alguien decida. Una publicación y sus dos finales posibles tienen este aspecto:

```
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
```

Todo en esa secuencia depende de saber, durante la pausa, si el canario funciona peor que la versión a la que sustituyó. Ese es el trabajo que hay detrás del servicio de fiabilidad del sitio y monitorización, basado en alertas que avisan cuando un fallo todavía es pequeño. La solución propia de monitorización de errores y trazado, realizada en el marco de un puesto desempeñado desde 2025, muestra cómo es una señal de ese tipo: los errores se agrupan en problemas recurrentes en lugar de enumerarse uno a uno, y los picos y las regresiones se detectan con un período de espera entre avisos, para que un despliegue defectuoso no genere cuarenta avisos a una persona de guardia. Una etapa canario necesita exactamente eso: una regresión que se haga visible durante la pausa, no una con la que un usuario tropiece a la mañana siguiente.

Dónde surgen los problemas

El canario es demasiado pequeño para aportar información. Un pod de cada diez en una hora tranquila puede atender unos pocos cientos de solicitudes durante su pausa, y un fallo que afecta a una de cada mil nunca aparece. El reloj es la medida equivocada; la correcta es el número de solicitudes que ha atendido el canario. Dimensione conjuntamente la proporción y la pausa para que un error poco frecuente tenga la oportunidad de aparecer varias veces, y mantenga una etapa más tiempo por la noche que al mediodía.

Los paneles no distinguen el canario de la versión estable. Un canario con un fallo adicional por cada cincuenta solicitudes respecto de la versión estable, con una proporción del 5 %, añade aproximadamente una solicitud fallida por cada mil al total. En un panel con una sola tasa de errores para todo el servicio, eso resulta invisible. Asigne etiquetas distintas a los pods canario y estables, que Argo Rollouts añade mediante `canaryMetadata` y `stableMetadata`, traslade la etiqueta a las métricas y compare el canario con la versión estable en lugar de con un umbral fijo. Un `analysis` en segundo plano del Rollout cancela la actualización cuando falla.

Ambas versiones escriben en los mismos datos. Mientras dura el despliegue, el código antiguo y el nuevo leen y escriben en la misma base de datos y, con enrutamiento aleatorio, las solicitudes de un mismo usuario pueden llegar a ambos en un minuto. Una columna nueva que la versión antigua nunca rellena, o un formato que no puede leer, produce errores en el lado estable, la parte que

debía ser segura. La respuesta de Sato es el cambio en paralelo: la base de datos admite ambas versiones de la aplicación durante todo el despliegue, y la estructura antigua solo se retira cuando desaparece la versión antigua.

Cancelar detiene el tráfico, no el cambio. `kubectl argo rollouts abort` devuelve el tráfico al ReplicaSet estable, pero la especificación del Rollout sigue nombrando la imagen nueva y reanudar el Rollout vuelve a convertir esa imagen en el destino. La reversión solo está completa cuando la plantilla se devuelve a la versión anterior. Si la especificación se conserva en un repositorio, haga allí el cambio para que la fuente y el clúster coincidan sobre lo que debe estar ejecutándose.

El canario se utiliza también como prueba A/B. El enrutamiento se parece, por lo que resulta tentador medir el efecto de una función nueva sobre las conversiones con el tráfico del canario. Sato aconseja separar ambas cosas: un canario busca problemas y debería terminar en minutos u horas, mientras que un experimento pone a prueba una hipótesis y puede necesitar días de datos para alcanzar significación estadística. Si el canario se evalúa con métricas de negocio, el efecto que el experimento pretende provocar también enturbia el veredicto. Publique con el canario y después ejecute el experimento con su propia división, como una bandera de funcionalidad, una vez publicada la versión.

Lecturas adicionales

- Publicación canario — la descripción de Danilo Sato de 2014 en el bliki de Martin Fowler: selección de usuarios, pruebas de capacidad, pruebas A/B y la base de datos.
- Argo Rollouts: canario — etapas y pausas, pesos aproximados mediante el número de pods y escalado con un enrutador de tráfico.
- Reparto de tráfico HTTP — la guía de Gateway API sobre backends ponderados y una ruta canario que coincide con una cabecera.
- kubectl argo rollouts abort — qué hace una cancelación con el tráfico y qué deja en la especificación.



Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una publicación canario?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/canary-release/