Engineer

Bandera de funcionalidad

interruptor que activa o desactiva código sin recompilar

Qué es

Una bandera de funcionalidad es un punto de decisión del código que lee un ajuste que el equipo puede cambiar sin volver a compilar la aplicación. Tras ella hay una ruta de código nueva que ya está en producción, pero permanece inactiva hasta que el ajuste la activa, para todos o para un grupo de usuarios elegido. Pete Hodgson describió la técnica en detalle en el sitio de Martin Fowler, por entregas desde enero de 2016 y con una revisión en octubre de 2017. Las llama feature toggles y enumera feature flags, feature bits y feature flippers como otros nombres de lo mismo.

Hodgson divide una bandera en partes. El punto de conmutación es el lugar del código que pregunta si la ruta nueva está activada. El enrutador de conmutación responde a partir de la configuración de conmutación y, cuando la respuesta depende de quién pregunta, de un contexto de conmutación: normalmente el usuario actual o una cookie o cabecera HTTP especial que lo representa. Advierte contra acoplar el punto de conmutación y el enrutador. El código del punto de conmutación debería formular una pregunta concreta, como si la confirmación de este pedido incluye el nuevo enlace de cancelación; la regla que lo decide vive en un único lugar detrás del enrutador, para poder cambiar sin tocar cada punto de llamada.

Las banderas se diferencian por cuánto viven y con qué frecuencia cambia su respuesta. Hodgson las clasifica en cuatro tipos. Las de publicación permiten llevar a producción código inacabado y sin probar como código latente; dice que «por lo general no deberían permanecer mucho más de una o dos semanas». Las de experimentación dividen a los usuarios en grupos para una prueba A/B y asignan siempre a cada usuario al mismo grupo. Las de operaciones permiten cambiar el comportamiento del sistema en producción. Muchas duran poco, pero algunos interruptores de emergencia duraderos permiten recortar funciones prescindibles cuando la carga es inusualmente alta, como un panel de

recomendaciones costoso de generar; Hodgson los compara con un disyuntor gestionado manualmente. Las de permisos deciden qué funciones ve cada usuario, por ejemplo, funciones prémium para clientes de pago o funciones alfa para el personal de la empresa, y pueden permanecer años.

Una bandera de funcionalidad no es una publicación canario, aunque ambas suelen combinarse. Un canario ejecuta dos versiones de un servicio en paralelo y dirige parte del tráfico a la nueva; una bandera distribuye una versión con las dos rutas y elige entre ellas en cada solicitud. Una rama duradera en el control de versiones también separa el trabajo nuevo, pero el código solo llega a producción al integrar la rama, todo de una vez.

Utilice el concepto en una decisión

Aplique «¿Qué es una bandera de funcionalidad?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Un equipo que lleva una idea del boceto a un producto al que sus clientes pueden acceder, sin pasarla entre tres proveedores.

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. Ingresos multiplicados por 10 con mapas GIS — 10x crecimiento de ingresos impulsado desde el producto cartográfico

Ingresos multiplicados por 10 con mapas GIS Póngase en contacto →

Por qué importa

El principal motivo para usar banderas es separar el despliegue de la publicación. Hodgson describe las de publicación como la forma más habitual de aplicar un principio de entrega continua: publicar una función es un paso distinto de desplegar su código. El código puede llegar a producción cada día desde una única rama principal mientras una función que requiere un mes sigue oculta hasta estar lista. Eso hace viable el desarrollo basado en tronco.

Como una bandera puede dar respuestas distintas según el usuario, una publicación puede ampliarse por etapas. Hodgson describe activar primero las funciones nuevas para usuarios internos, lo que llama Champagne Brunch, porque la empresa bebe primero su propio champán. Después llega un grupo canario: su ejemplo es un 1 por ciento aleatorio de usuarios, elegido mediante una operación de módulo con el identificador de usuario para mantener a las mismas personas en el grupo, mientras el otro 99 por ciento conserva el comportamiento antiguo. Si ese grupo comunica problemas, se desactiva la bandera y nadie tiene que revertir un despliegue.

Todo tiene un precio. Cada bandera añade una rama al código y una combinación que probar, y su configuración pasa a formar parte de la producción y puede cambiar sin compilar. Hodgson aconseja guardar la configuración de conmutación en el control de versiones y cambiarla mediante nuevos despliegues siempre que lo permita la bandera, para que recorra el mismo flujo que el código y sus cambios se revisen y registren. La configuración en ejecución justifica su coste en las banderas que deben cambiar en minutos, como las de operaciones.

Cómo se aplica en la práctica

flagd es un motor de evaluación de banderas de código abierto que sigue la especificación OpenFeature. No tiene interfaz de usuario ni almacena nada propio: lee definiciones de un archivo JSON o de otra fuente y responde a solicitudes, como sidecar, dentro del proceso de la aplicación o como servicio central. Una bandera que envía aproximadamente a uno de cada veinte usuarios a un proceso de compra nuevo tiene este aspecto:

```
{
  "$schema": "https://flagd.dev/schema/v0/flags.json",
  "flags": {
    "new-checkout": {
      "state": "ENABLED",
      "variants": {
        "on": true,
        "off": false
      },
      "defaultVariant": "off",
      "targeting": {
        "fractional": [
          ["on", 5],
          ["off", 95]
        ]
      },
      "metadata": {
        "owner": "checkout",
        "expires": "2027-01-31"
      }
    }
  }
}
```

Cada bandera tiene un estado y un conjunto de variantes que deben ser todas del mismo tipo; la variante predeterminada es opcional. La regla `fractional` divide a los usuarios en grupos con pesos relativos, por lo que 5 y 95 sitúan

aproximadamente a uno de cada veinte en `on`. De forma predeterminada, el grupo se calcula a partir de la clave de la bandera y del `targetingKey` del contexto de evaluación. Por tanto, un mismo usuario recibe siempre la misma respuesta y renombrar la bandera redistribuye a todos. `metadata` contiene lo que el equipo quiera adjuntar, aquí un responsable y la fecha para la que debería haberse eliminado. Poner `state` en `DISABLED` hace que flagd responda sin valor alguno y que quien llama recurra a su propio valor predeterminado.

Ejecutar flagd con el archivo y pedirle una respuesta por HTTP:

```
flagd start --uri file:./flags.json
curl -X POST "localhost:8013/flagd.evaluation.v1.Service/ResolveBoolean" \
  -H "Content-Type: application/json" \
  -d '{"flagKey":"new-checkout","context":{"targetingKey":"user-4711"}}'
```

La respuesta contiene el valor, la variante y el motivo `TARGETING_MATCH`. En la aplicación, la llamada pasa por un cliente OpenFeature con la clave de la bandera, un valor predeterminado de `false` y el contexto de evaluación; el punto de conmutación solo ve verdadero o falso. Subir el porcentaje del 5 al 20 por ciento es un cambio en el archivo de banderas, no en la compilación de la aplicación.

Los espacios de trabajo de contratación y carrera marítima, desarrollados en un puesto desempeñado desde 2025, muestran una bandera de publicación en la instantánea de implementación documentada: esta describe la parte de contratación como activa y el espacio de carrera profesional como controlado por una bandera y publicado de forma incompleta. Es una instantánea histórica, no una declaración del estado actual del despliegue. El back-office del operador, también desarrollado en ese puesto, es donde se cambian banderas así. Los operadores las conmutan y activan una función solo para algunos usuarios desde una pantalla, en lugar de que un ingeniero edite a mano la base de producción; cada cambio se autoriza y registra como cualquier otra acción allí.

Dónde surgen los problemas

Banderas que nunca se van. Una bandera de publicación que ya hizo su trabajo es coste puro: una rama innecesaria, una ruta que nadie prueba y un ajuste que alguien puede cambiar por error. Hodgson propone tratarlas como un inventario que tiene un coste y mantenerlo pequeño. Entre sus remedios están añadir una tarea de eliminación al trabajo pendiente en cuanto se introduce la bandera,

ponerle una fecha de caducidad, una bomba de tiempo que haga fallar una prueba o impida arrancar la aplicación al superar esa fecha y un límite fijo de banderas, de modo que añadir una obligue a quitar otra. Eliminar una bandera exige borrar también la ruta de código descartada, además de la comprobación.

Una bandera antigua adquiere otro significado. El 1 de agosto de 2012, SMARS, el enrutador automático de órdenes de Knight Capital, envió millones de órdenes al mercado durante 45 minutos y la empresa perdió más de 460 millones de dólares. La resolución de la SEC explica cómo. El código nuevo del Retail Liquidity Program de la Bolsa de Nueva York reutilizó una bandera incluida en las órdenes entrantes que antes activaba Power Peg, una función que Knight dejó de usar en 2003 pero nunca eliminó. El despliegue omitió uno de los ocho servidores, donde la bandera reutilizada despertó la función antigua. Cuando el personal retiró el código nuevo de los otros siete, la bandera también activó allí Power Peg. Nunca dé un significado nuevo a una bandera retirada, elimine el código muerto junto con la bandera que lo protegía y compruebe que todos los servidores ejecutan la misma compilación antes de activarla.

Combinaciones que nadie probó. Dos banderas producen cuatro combinaciones y diez superan las mil; Hodgson lo llama explosión combinatoria. Rara vez es posible probarlas todas, así que pruebe las que pueden llegar a producción: la configuración actual con las banderas a punto de publicarse activadas, la alternativa con esas banderas desactivadas y, como hacen muchos equipos, todo activado. Ayuda una convención fija: desactivada siempre significa el comportamiento existente y activada el nuevo.

Un valor predeterminado que nadie eligió. La especificación OpenFeature dice que evaluar banderas no debe lanzar excepciones. Si algo falla, la llamada devuelve el valor predeterminado recibido, la forma detallada del resultado contiene un código de error y el cliente tampoco debería escribir en el registro, porque una aplicación desplegada antes de definir su bandera lo inundaría de `FLAG_NOT_FOUND`. Eso mantiene la aplicación funcionando, pero también convierte silenciosamente una errata en la clave en el valor predeterminado para todos. Pase el comportamiento antiguo y seguro como valor predeterminado y añada un hook, donde la especificación sitúa el registro, para contar evaluaciones que terminan en error y generar una alerta cuando aparezcan.

Valores forzados que cualquiera puede fijar. Es cómodo dejar que los probadores fuercen la activación de una bandera mediante una cookie especial, un parámetro en la URL o una cabecera HTTP. Hodgson señala que usuarios

curiosos o maliciosos pueden fijar el mismo valor forzado y cambiar para sí el estado de la bandera. Respete esos valores forzados solo en tráfico interno o fírmelos criptográficamente. Eso alivia la preocupación, pero, como señala Hodgson, añade su propia complejidad y superficie de ataque.

Lecturas adicionales

- Feature Toggles (también llamadas Feature Flags) — El artículo de Pete Hodgson en el sitio de Martin Fowler: partes de una bandera, cuatro categorías, pruebas y cómo mantener pequeño el inventario.
- API de evaluación de banderas — La especificación OpenFeature para evaluarlas, incluidos valores predeterminados, códigos de error y por qué la evaluación nunca lanza excepciones.
- flagd: definiciones de banderas — El formato JSON que lee flagd: estado, variantes, variante predeterminada, selección de usuarios y metadatos.
- flagd: operación fraccionaria — Repartos porcentuales deterministas y qué decide en qué grupo termina un usuario.
- En el asunto de Knight Capital Americas LLC — La resolución de la SEC de 2013 sobre la pérdida bursátil de agosto de 2012, incluida la bandera reutilizada.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una bandera de funcionalidad?» 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/feature-flag/