Engineer

despliegue azul-verde

◐ pasar el tráfico de producción entre dos entornos idénticos

Qué es

El despliegue azul-verde es una forma de publicar software con dos entornos de producción construidos para ser lo más parecidos posible. Uno de ellos, aquí llamado azul, atiende todas las solicitudes. La nueva versión se instala en el otro, el verde, y se prueba allí mientras no recibe tráfico de clientes. Cuando está lista, se cambia el componente situado delante de ambos para que todas las solicitudes nuevas vayan al verde. El azul sigue funcionando sin cambios y, si el verde se comporta mal, el mismo cambio a la inversa devuelve el azul al servicio.

Los colores no tienen ningún significado, y esa es su razón de ser. Nombres como «antiguo» y «nuevo» o «principal» y «de reserva» intercambian su significado después de cada publicación, mientras que dos colores simplemente se turnan: tras el siguiente cambio, el verde está en servicio y el azul queda inactivo. Martin Fowler describió la técnica en su bliki en 2010 y atribuye el nombre, con cierta imprecisión, a Daniel Terhorst-North y Jez Humble. La grafía inglesa habitual lleva un guion, aunque las herramientas la escriben a su manera: Argo Rollouts, un controlador de Kubernetes que añade estrategias de despliegue de este tipo, la llama `blueGreen` en su configuración.

El cambio puede realizarse en cualquier punto único que decida adónde van las solicitudes. La versión de Fowler cambia un enrutador. También puede ser el destino de un balanceador de carga, el servidor de destino de un proxy inverso o, en Kubernetes, un Service cuyo selector de etiquetas elige los Pods que reciben su tráfico. Los dos entornos tampoco tienen por qué ser dos centros de datos: Fowler enumera como opciones suficientes hardware separado, máquinas virtuales separadas o dos zonas de un sistema con sus propias direcciones IP.

Dos técnicas cercanas aclaran la diferencia. Una actualización progresiva sustituye unas pocas instancias cada vez, de modo que las versiones antigua y nueva responden a solicitudes a la vez hasta que desaparece la última instancia antigua, y revertir significa volver a desplegar progresivamente en sentido contrario. Una publicación canario envía primero una pequeña parte del tráfico a la nueva versión y la amplía paso a paso. El despliegue azul-verde lo mueve todo de golpe, pero conserva una copia completa y en funcionamiento de la versión anterior, por lo que volver atrás es un único paso de la misma magnitud que avanzar.

Utilice el concepto en una decisión

Aplique «¿Qué es un despliegue azul-verde?» 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

Publicar una versión se convierte en una operación breve y reversible en vez de una larga y de sentido único. Las partes lentas, copiar archivos, iniciar procesos y preparar cachés, ocurren en el verde mientras el azul sigue atendiendo, por lo que los clientes nunca esperan por ellas. El cambio en sí es rápido, y deshacerlo también: no se reconstruye ni se reinstala nada, porque la versión anterior nunca se detuvo. Eso cambia cómo afronta un equipo una publicación en una tarde cualquiera y la rapidez con la que puede actuar cuando algo parece ir mal.

El verde también se puede comprobar en condiciones muy próximas a producción antes de que lo vea ningún cliente: la misma red, la misma configuración y la misma base de datos. Una prueba básica de funcionamiento contra el verde demuestra mucho más que la misma prueba en un entorno de preproducción que difiere en una docena de pequeños detalles. Argo Rollouts lo formaliza con un Service de vista previa, que dirige el tráfico a la nueva versión antes de promoverla para que se pueda probar sin atender tráfico de producción.

El entorno inactivo tiene una segunda función. Una vez estabilizada una versión, el lado que acaba de dejar de recibir tráfico se convierte en el lugar donde preparar la siguiente y, mientras tanto, sirve de reserva preparada. Fowler señala

que el mismo mecanismo «permite probar el procedimiento de recuperación ante desastres en cada publicación», porque cada cambio es una conmutación por error que alguien está observando.

Los costes son claros. Dos entornos de producción necesitan capacidad para dos entornos, al menos mientras dura una publicación. Y, aunque la capa de aplicación es fácil de duplicar, los datos no lo son: el azul y el verde casi siempre comparten una sola base de datos, lo que significa que cada publicación debe dejar los datos legibles para ambas versiones. Un equipo que no pueda garantizarlo obtiene un cambio rápido hacia delante y un cambio de vuelta que falla.

Cómo se aplica en la práctica

En Kubernetes, la versión funcional más pequeña consta de dos Deployments y un Service. Los Deployments `web-blue` y `web-green` ejecutan la misma aplicación con las etiquetas `app: web` y `slot: blue` o `slot: green`. El Service al que llegan los clientes selecciona uno de los dos espacios de despliegue:

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

Para publicar una versión, se instala la nueva imagen en el lado inactivo, se espera a que esté listo, se prueba mediante un segundo Service que solo selecciona `slot: green` y, después, se cambia el que atiende el tráfico real:

```
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` utiliza de forma predeterminada un parche de fusión estratégica, de modo que solo cambia la clave `slot` del selector y `app: web` permanece donde estaba. El controlador de Service sigue buscando Pods que coincidan con el selector y actualiza los puntos de conexión del Service cuando cambia el conjunto, por lo que las conexiones nuevas llegan al verde mientras el azul sigue preparado con la imagen anterior.

Argo Rollouts automatiza la misma secuencia con un objeto `Rollout` : un `activeService` que dirige el tráfico a los nuevos pods al promover la versión y un `previewService` opcional que permite acceder a ellos antes. Con `autoPromotionEnabled: false` , el despliegue se pausa antes del cambio hasta que alguien ejecuta `kubectl argo rollouts promote web` .

El servicio de DevOps y CI/CD promete cambios probados, publicados y reversibles, y el despliegue azul-verde es una forma de reducir el coste de la reversión. Se apoya en dos hábitos que los despliegues de una plataforma en Azure muestran en la práctica, sin ningún segundo espacio de producción: las plantillas Bicep mantienen idéntica la infraestructura de destino entre entornos y el mismo artefacto compilado se promueve desde desarrollo, pasando por preproducción, hasta producción, adoptando los ajustes de cada entorno al llegar. Esa ruta lleva una versión por los entornos en secuencia. El despliegue azul-verde añade un gemelo al final, y esos mismos dos hábitos, infraestructura idéntica y un artefacto sin cambios, son los que permiten al equipo confiar en que el gemelo lo sea de verdad.

Dónde surgen los problemas

Un cambio de esquema convierte la vuelta atrás en un callejón sin salida. El verde arranca, ejecuta una migración que cambia el nombre de una columna y atiende el tráfico sin problemas. Diez minutos después, un fallo devuelve el tráfico al azul, que sigue solicitando el nombre antiguo de la columna y falla en todas las solicitudes. La reversión se había planificado para la aplicación, no para los datos. El consejo de Fowler de 2015 es separar ambas cosas: publicar un cambio en la base de datos con el que puedan trabajar las dos versiones, después la aplicación y, una vez estabilizada la nueva versión, retirar lo que solo necesitaba la antigua.

El estado escrito en el verde queda aislado al volver atrás. Todo lo que el verde guarda para sí, como archivos subidos a su disco local, un almacén de sesiones en memoria o su propia caché de tareas pendientes, no está en el azul

cuando el tráfico vuelve allí. Los usuarios se encuentran con la sesión cerrada o sin lo que acaban de guardar. Mantenga el estado compartido en almacenes que utilicen ambos colores. Cuando el verde deba tener sus propios datos, Fowler propone enviar las transacciones a los dos entornos o iniciar el verde en modo de solo lectura y pasarlo a lectura y escritura únicamente cuando haya demostrado que funciona.

Se elimina la capacidad de volver atrás antes de que aparezca el fallo. Argo Rollouts reduce las réplicas del ReplicaSet antiguo después del cambio, y el valor predeterminado de `scaleDownDelaySeconds` es 30. Si el fallo aparece al cabo de un cuarto de hora, durante una tarea nocturna o por una fuga lenta, ya no queda ningún entorno azul al que regresar. Ajuste el retraso para cubrir el tiempo que suelen tardar en aparecer los problemas y acepte que el precio es la capacidad:

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

La promoción depende de que esté listo, no de que sea correcto. El valor predeterminado de `autoPromotionEnabled` es true, lo que significa que Argo Rollouts cambia de versión en cuanto los nuevos pods están listos. Una sonda de disponibilidad indica que un proceso responde, no que el proceso de compra funcione. Pause antes de promover, como en el bloque anterior, y pruebe mediante el Service de vista previa, o añada un `prePromotionAnalysis`, que cancela el despliegue si falla, y un `postPromotionAnalysis`, que vuelve a la versión anterior si falla después del cambio.

El lado inactivo se desvía de la configuración. Entre publicaciones, alguien aplica un cambio urgente de configuración solo al entorno en servicio. La siguiente versión llega al lado inactivo, que aún conserva el ajuste antiguo, y el cambio que debía modificar una cosa modifica dos. Construya ambos espacios con las mismas plantillas y la misma fuente de configuración para que un cambio en uno lo sea en ambos, y trate el lado inactivo como el área de preparación de la siguiente versión, tal como describe Fowler.

Lecturas adicionales

- Despliegue azul-verde — la descripción de Martin Fowler de 2010, con su nota de 2015 sobre cambios en la base de datos.
- Argo Rollouts: BlueGreen — Services activo y de vista previa, promoción automática, análisis antes y después del cambio y retraso en la reducción de réplicas.
- Service de Kubernetes — cómo un selector elige los Pods que reciben el tráfico de un Service.
- kubectl patch — parches de fusión estratégica, de fusión y JSON; el comando que cambia el selector en el ejemplo anterior.

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 un despliegue azul-verde?» 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/blue-green-deployment/