Engineer

Despliegue

llevar un cambio al lugar donde las personas lo utilizan

Qué es

El despliegue es la secuencia de pasos que saca una versión probada de una aplicación del sistema de compilación y la pone en servicio: colocar los archivos compilados o la imagen de contenedor en las máquinas o la plataforma que la ejecutarán, entregarle la configuración de ese entorno, iniciarla y dirigir hacia ella las peticiones reales. La palabra designa tanto el acto como el resultado, así que un equipo habla de un despliegue detenido en el paso relativo a la base de datos y de los despliegues que se ejecutan actualmente en cada entorno. Lo desplegado puede ser un sitio web, una API, una tarea programada, un cambio de base de datos o el backend de una aplicación móvil.

Desplegar y lanzar suelen mantenerse separados. Desplegar pone el código nuevo donde puede ejecutarse; lanzar decide quién lo ve. Una función oculta tras un indicador puede desplegarse el lunes y ponerse a disposición de los clientes el jueves sin volver a tocar los servidores. Separar ambas cosas permite entregar código en pasos pequeños y frecuentes, mientras las decisiones de producto sobre el momento siguen en manos de quienes tienen esa responsabilidad.

Muchos despliegues actuales entregan una imagen de contenedor. Open Container Initiative, creada bajo Linux Foundation en junio de 2015, publica tres especificaciones al respecto: cómo se organiza una imagen, cómo la ejecuta un entorno de ejecución una vez desempaquetada y cómo los registros almacenan y distribuyen imágenes. Una imagen se identifica por su digest, un hash de su contenido, de modo que el mismo digest siempre significa los mismos bytes. Una etiqueta como `1.4.2` es un puntero legible a una imagen, y volver a publicar bajo esa etiqueta hace que apunte a algo nuevo.

Twelve-Factor App, una metodología muy citada para construir servicios web, divide el camino del código al servicio en ejecución en tres etapas estrictamente separadas. La etapa de compilación convierte un commit en un paquete ejecutable. La de lanzamiento combina ese paquete con la configuración de un entorno. La de ejecución inicia procesos de un lanzamiento elegido. Cada lanzamiento recibe un identificador único y nunca se cambia una vez creado; cualquier cambio produce un lanzamiento nuevo, lo que convierte la reversión en elegir uno anterior.

La forma de cambiar de versión depende de lo que pueda tolerar el servicio. Un despliegue con recreación detiene la versión anterior y después inicia la nueva, aceptando un intervalo sin servicio. Un despliegue gradual sustituye unas pocas instancias cada vez, de modo que ambas versiones atienden tráfico durante un tiempo; es el predeterminado para un Deployment de Kubernetes. Un despliegue azul-verde ejecuta la versión nueva junto a la anterior y mueve todo el tráfico de una vez, manteniendo la anterior lista para recibirlo de vuelta. Un canario envía primero una pequeña parte del tráfico a la nueva versión y la amplía mientras las tasas de error y la latencia se mantengan en valores saludables.

Utilice el concepto en una decisión

Aplique «¿Qué es un despliegue?» 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

El despliegue es el momento en que un cambio empieza a ayudar a alguien y también aquel en que un error empieza a llegar a los usuarios. Los equipos que despliegan pocas veces tienden a reunir muchos cambios en cada lanzamiento, por lo que cada despliegue lleva más riesgo y, cuando falla, más posibles causas. Los despliegues pequeños y frecuentes mantienen cada paso fácil de entender y de revertir.

Un despliegue a mano solo es tan consistente como quien lo ejecuta. Se omiten pasos, se hacen en otro orden o un compañero que cubre unas vacaciones los ejecuta de forma ligeramente distinta. Un flujo de procesamiento hace los

mismos pasos cada vez y deja un registro del commit, el artefacto y el entorno, con una marca de tiempo. Ese registro es donde empieza la revisión de un incidente y, a menudo, la forma más rápida de responder qué cambió.

La práctica más sólida es compilar una vez y promover el resultado. El artefacto probado en preproducción debe ser, byte por byte, el que llega a producción; solo difiere la configuración, que el entorno suministra cuando llega el artefacto. Registrar el digest al compilar y desplegar por digest permite comprobar esa promesa:

```
podman build -t registry.example.com/web:1.4.2 .
podman push --digestfile web.digest registry.example.com/web:1.4.2
kubectl -n staging set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
# after staging checks pass, the same digest goes to production
kubectl -n production set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
```

La reversión depende de la misma disciplina. Cuando los lanzamientos son inmutables y están identificados, volver atrás significa apuntar al anterior, algo que tarda segundos. Cuando cada entorno vuelve a compilar desde el código fuente, volver atrás significa compilar un commit antiguo y esperar que sus dependencias sigan resolviéndose de la misma manera.

Cómo se aplica en la práctica

Los despliegues de una plataforma en Azure se gestionaron de principio a fin, con todos los cambios siguiendo la misma ruta fija de desarrollo a preproducción y a producción, sin que nadie publicara directamente en un entorno activo. CI/CD compila las imágenes y las entrega, mientras las plantillas Bicep mantienen la misma infraestructura de destino en cada entorno, de modo que cada promoción llega a una infraestructura equivalente. Los ajustes que difieren por entorno se mantienen separados de los secretos, lo que permite que una sola compilación atraviese todas las etapas y recoja los ajustes de cada entorno al llegar, sin volver a compilar durante el recorrido. Los cambios llegan a cada entorno por ese camino definido, sin despliegues manuales improvisados, y el lanzamiento pasó a ser un paso controlado que se ejecuta igual cada vez.

Un trabajo anterior muestra cuánto vale esa consistencia en horas. Automatizar la compilación, las pruebas y el despliegue ahorró unas 4.000 horas al estandarizar el camino de un cambio hasta producción y eliminar los pasos manuales que variaban según quién los realizara. Los lanzamientos se volvieron más rápidos y más fiables.

En la propia plataforma de la empresa, el despliegue se mantiene separado del aprovisionamiento. Una capa de contenedores construida con Podman y Quadlet requiere que root prepare la ruta de despliegue una vez. Después, un usuario sin privilegios vuelve a desplegar ejecutando un único script fijo mediante una regla de alcance restringido que permite únicamente ese comando sin argumentos. El script compila la imagen y reinicia el servicio, de modo que un lanzamiento rutinario no necesita una sesión de shell de root ni comandos arbitrarios.

Dónde surgen los problemas

La mayoría de los problemas de despliegue proceden de una diferencia entre lo comprobado y lo que cambió realmente.

Producción ejecuta algo que preproducción nunca vio. Cuando cada entorno compila su propia copia, los intervalos de dependencias se resuelven en versiones más nuevas, una imagen base cambia entre compilaciones o difiere una opción de compilación, y el artefacto que superó las pruebas no es el que atiende a los clientes. Compile una vez, registre el digest, promueva ese digest y mantenga todo lo específico de cada entorno en la configuración, en lugar de en el artefacto.

Una etiqueta cambia y la reversión llega a la versión equivocada. Un despliegue que nombra `latest` o reutiliza una etiqueta de versión después de volver a compilar depende de a qué apunta la etiqueta el día en que se ejecuta. La documentación de Kubernetes desaconseja `:latest` en producción porque dificulta saber qué versión está ejecutándose y revertir correctamente. Utilice una etiqueta de versión que nunca se reutilice y fije el digest donde la plataforma lo permita.

El flujo informa de éxito antes de que funcione la versión nueva. Muchos comandos de despliegue terminan en cuanto la plataforma acepta la petición, mientras las instancias nuevas todavía están iniciándose y podrían fallar un minuto después. Un paso que espera a que la versión nueva indique que está lista

y después envía unas cuantas peticiones reales y comprueba las respuestas convierte un fallo silencioso en una compilación fallida mientras el lanzamiento anterior sigue a mano.

El código nuevo y el esquema antiguo se encuentran a mitad del despliegue. Durante un despliegue gradual o canario, las versiones antigua y nueva atienden tráfico a la vez contra una base de datos. Una migración que renombra o elimina una columna rompe las instancias antiguas que siguen ejecutándose y hace insegura la reversión, porque el código antiguo espera lo que se eliminó. Reparta esos cambios entre despliegues: añada primero la estructura nueva, entregue código que funcione con ambas y elimine la estructura antigua solo cuando ninguna versión en ejecución la necesite.

Lecturas adicionales

- Compilación, lanzamiento y ejecución — los argumentos de Twelve-Factor App a favor de tres etapas separadas y lanzamientos que nunca se cambian una vez creados.
- Open Container Initiative — el proyecto de Linux Foundation responsable de las especificaciones de imagen, ejecución y distribución en las que se apoyan la mayoría de los despliegues de contenedores.
- Especificación del formato de imagen OCI — manifiestos, capas, configuración y los digests del contenido que identifican una imagen.
- Deployments de Kubernetes — actualizaciones graduales, estrategia de recreación, plazo de progreso y reversiones.
- Imágenes de Kubernetes — etiquetas, digests, políticas de descarga y por qué `:latest` dificulta las reversiones.

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?» 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/deployment/