Engineer

GitHub

alojamiento de Microsoft para repositorios Git

Qué es

GitHub es un servicio alojado, parte de Microsoft, que guarda repositorios Git en sus servidores y construye a su alrededor un lugar de trabajo: una vista web del código y su historial, pull requests para proponer y revisar cambios, incidencias para seguir el trabajo y GitHub Actions para ejecutar tareas automáticamente cuando ocurre algo en un repositorio.

Por debajo hay Git sin más. Un repositorio clonado de GitHub es un repositorio Git normal con todo su historial, y el trabajo viaja entre un portátil y GitHub con los push y fetch habituales. Lo que añade GitHub vive en su lado de la conexión: discusiones de revisión, incidencias, reglas de ramas, ajustes de automatización y la lista de quién puede hacer qué se guardan en los sistemas de GitHub, no en el repositorio clonado.

Pull request es el nombre que GitHub da a una propuesta de integrar los cambios de una rama en otra. Muestra las diferencias línea por línea, reúne comentarios y aprobaciones y presenta el resultado de cada comprobación automática ejecutada sobre el código propuesto. Un pull request abierto como borrador queda marcado como inacabado y no se puede fusionar hasta que alguien lo marque como listo para revisión.

GitHub Actions es la automatización integrada en el servicio. Un flujo de trabajo es un archivo YAML en el directorio `.github/workflows` del repositorio. Nombra los sucesos que lo inician, como un push, un nuevo pull request o una programación horaria, y enumera trabajos compuestos por pasos; cada paso ejecuta una orden del shell o una acción, una pieza de automatización empaquetada que se puede compartir entre repositorios. Los trabajos se

ejecutan en runners: máquinas virtuales que GitHub inicia de nuevo para cada ejecución en Ubuntu Linux, Windows o macOS, o máquinas propias registradas como runners autoalojados.

Otros dos servicios vigilan lo que entra en un repositorio. Dependabot lee un archivo `dependabot.yml` y abre pull requests que actualizan dependencias; por separado, abre otros para dependencias con vulnerabilidades conocidas. La protección de push inspecciona los pushes entrantes en busca de secretos como tokens de acceso y bloquea uno que los contenga antes de que llegue al repositorio.

Utilice el concepto en una decisión

Aplique «¿Qué es GitHub?» 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

Para muchos equipos, GitHub es donde se hace el trabajo, no solo donde se guarda. El código, la discusión de cada cambio, el registro de quién lo aprobó y el resultado de cada comprobación están juntos, de modo que el motivo de que exista una línea suele poder averiguarse en una sola página.

La automatización importa tanto como el alojamiento. Con Actions junto al código, un pull request puede llevar su propia evidencia: las pruebas se ejecutaron y pasaron, la compilación funcionó y los analizadores comunicaron sus hallazgos. Las ramas protegidas convierten esa evidencia en una regla. Una rama se puede configurar para admitir cambios solo mediante un pull request, solo tras un número determinado de revisiones favorables y solo después de superar comprobaciones de estado concretas. Así, por la línea principal solo pasan cambios revisados y comprobados.

Como GitHub está en el punto donde el código entra al proyecto, también alberga los controles de la cadena de suministro: qué acciones de terceros puede ejecutar un flujo y en qué versión exacta, cómo se actualizan las dependencias y

si un secreto se frena al hacer push. Esos controles suelen activarse y ajustarse por repositorio u organización. Por eso, gran parte de la seguridad real de un proyecto de GitHub vive en ajustes que nadie ve durante una revisión de código.

Esa es la contrapartida de un servicio alojado. Se obtiene un entorno completo y mantenido sin operar servidores; a cambio, las reglas que protegen el código se configuran en la interfaz de otro, con valores predeterminados elegidos para el caso general y no para el propio.

Cómo se aplica en la práctica

Tres páginas del trabajo de la empresa muestran distintas funciones de GitHub.

Una cadena de entrega construida con GitHub Actions divide el trabajo en flujos separados: uno para el control de calidad del código, otro para las pruebas y uno por entorno para los despliegues, con CodeQL, revisión de dependencias y un paso de inventario de componentes de software en paralelo. El recorrido va desde un commit hasta un servicio en ejecución en Azure, y la promoción lleva una compilación por los entornos siguiendo una ruta definida, en vez de hacerlo a mano. El frontend se entrega sobre una imagen distroless sin shell ni gestor de paquetes; la API de Go, sobre una base Alpine ligera.

Un programa continuo de análisis de seguridad utiliza los controles que rodean al código en GitHub. Cada GitHub Action de terceros está fijada a un hash de commit y no a una etiqueta, de modo que nadie puede mover una etiqueta y cambiar lo que ejecuta un flujo. Dependabot vigila 6 ecosistemas, Gitleaks analiza el historial en busca de credenciales y cada compilación produce un inventario de componentes de software con Anchore y Syft. La página también nombra el coste real: el volumen de hallazgos. Un analizador que lo comunica todo acostumbra a la gente a ignorarlo, así que mantener útil la señal exige clasificar y priorizar continuamente.

El despliegue de una aplicación SIG de software como servicio trasladó su ciclo de pruebas, compilación y despliegue a flujos de GitHub Actions, por lo que un lanzamiento dejó de depender de que alguien recordase los pasos. El procesamiento recurrente de datos y los informes pasaron a tareas programadas en Python, Bash y SQL, y la configuración y los secretos se estandarizaron para que todos los entornos se comportasen igual.

Así se ve la fijación a un commit en un flujo. La larga cadena hexadecimal es el identificador completo del commit de la versión revisada, y el comentario mantiene a su lado la versión legible por una persona:

```
steps:
    - uses: actions/checkout@3d3c42e5aac5ba805825da76410c181273ba90b1 # v7.0.1
```

Dónde surgen los problemas

- **Una acción fijada a una etiqueta cambia sin avisar.** Una etiqueta como `v4` puede moverla o eliminarla quien controla el repositorio de la acción, así que el código que ejecute mañana un flujo puede diferir del revisado hoy. GitHub recomienda fijar la acción a un SHA de commit completo, que un atacante solo podría imitar produciendo «una colisión SHA-1 para el contenido de un objeto Git válido».
- **Un token del flujo tiene más poder del necesario.** Cada ejecución recibe un `GITHUB_TOKEN`, y sus permisos determinan qué podría cambiar un paso comprometido. GitHub recomienda que el acceso de lectura al contenido del repositorio sea el valor predeterminado y conceder escritura solo a los trabajos que la necesiten.
- **Texto no fiable se ejecuta como orden.** Un flujo que escribe el título de un pull request directamente en un script en línea permite que quien lo redactó elija parte del script. Pasar el valor por una variable de entorno intermedia lo mantiene como datos. Los desencadenantes `pull_request_target` y `workflow_run` inician flujos privilegiados, y GitHub advierte de que hacer checkout del código de un pull request no fiable dentro de uno de esos flujos expone el repositorio a un ataque que lo comprometa.
- **Los administradores pasan de largo ante las reglas de ramas.** De forma predeterminada, una regla de protección de ramas exime de sus restricciones a quienes tienen permisos de administrador en el repositorio. Activar «Do not allow bypassing the above settings», que impide omitir los ajustes anteriores, hace que las reglas también les obliguen.
- **Un secreto se cuela por la protección de push.** La modalidad activada de forma predeterminada protege los pushes propios de cada usuario a repositorios públicos. La que protege un repositorio frente a cualquier colaborador requiere GitHub Secret Protection y permanece desactivada hasta que alguien la habilita. Incluso entonces, cualquiera con acceso de escritura puede saltarse un bloqueo dando un motivo, lo que genera una alerta y queda en el registro de auditoría. Una vez enviada una credencial, vive en cada clon: cámbiela en lugar de confiar en una reescritura del historial.

Lecturas adicionales

- Documentación de GitHub — La documentación completa de repositorios, pull requests, Actions y funciones de seguridad.
- Entender GitHub Actions — Flujos, eventos, trabajos, pasos y runners, y cómo encajan entre sí.
- Referencia de uso seguro — Fijación de acciones, permisos de tokens e inyección de scripts, explicados por el propio GitHub.
- Acerca de las ramas protegidas — Revisiones y comprobaciones de estado obligatorias y quién puede omitirlas.
- Actualizaciones de versiones de Dependabot — Cómo un archivo `dependabot.yml` dirige los pull requests de actualización.
- Protección de push — Sus dos modalidades, qué bloquea cada una y cómo se registra una omisión.

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 GitHub?» 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/github/