

Use Git de manera más eficiente: un flujo de trabajo simple de Git



Negar Jamalifard

6 de junio · 4 min de lectura ★



Git sin duda es una de las herramientas esenciales para todo desarrollador. Por útil que sea Git, no seguir las mejores prácticas lo hará completamente inútil. Es por eso que hay diferentes flujos de trabajo estándar para aprovechar al máximo Git y sus increíbles funciones.

Durante mucho tiempo, mi inicio de Git y GitHub fue una herramienta que respalda los códigos y nos permite acceder a ellos en línea; Lo que no estaba completamente mal, pero hay mucho más sobre ellos que no supe hasta que comencé mi trabajo como desarrollador en YasnaTeam .

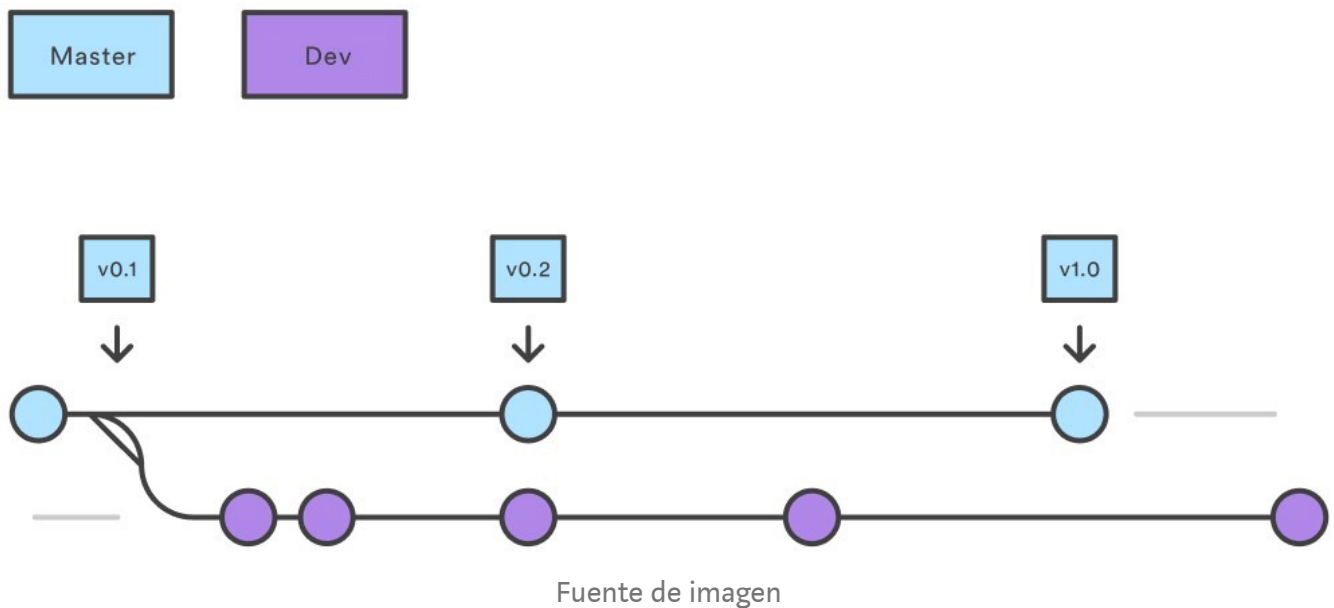
En nuestro equipo, unos 10 desarrolladores están trabajando activamente en el mismo repositorio. Tener el mismo hábito y la llamada cultura es necesario para evitar

conflictos y caos. Con más de 100 confirmaciones por día, las cosas podrían salirse de control muy pronto. Para evitar tales problemas, implementamos una versión modificada de Gitflow .

. . .

Cómo funciona

En lugar de tener una sola `master` rama, tenemos otra rama adicional llamada `dev` . Estas dos ramas están bloqueadas y nadie puede empujarlas directamente.



`master` está reservado solo para producción. Después de probar nuevas funciones, el propietario del repositorio se actualiza `master` antes de cada lanzamiento.

`dev` es la rama predeterminada desde la cual todos los desarrolladores crean una rama y la fusionan nuevamente. `dev` siempre está por delante del maestro y es la única rama en la que se fusionará `master` , con frecuencia.

Nombramiento de ramas

Tenemos un patrón para nombrar nuestras ramas.

```
[Nombre del módulo] - [Tipo] - [Detalle]
```

```
// Ejemplo
users-fix-attach-roles-issue # 765
```

Dado que tenemos una estructura modular en nuestra base de código, los cambios relacionados con cada módulo deben confirmarse en una rama separada con el nombre del módulo. Esta parte podría eliminarse si no tiene esa estructura en su proyecto.

Según el tipo de tarea, los desarrolladores deben elegir entre uno de estos tipos:

- El tipo `hotfix` es para soluciones urgentes en la rama maestra y no se utilizará con tanta frecuencia.

El detalle es una breve descripción de lo que la sucursal entregará.



Fuente de imagen

Estas reglas de ramificación nos obligan a separar cada tarea en trozos significativos y fusionarnos con frecuencia `dev` y siempre tener ramas frescas. Un gran efecto secundario de este enfoque es minimizar los conflictos.

Confirmar mensajes

Para tener una lista de confirmación limpia y descriptiva, también tenemos una regla de nomenclatura para los mensajes de confirmación (¡me encantan las reglas!).

```
[[Nombre del módulo]] [Mensaje de confirmación]
```

```
// Ejemplo
```

```
[usuarios] agregar punto final de usuarios de búsqueda
```

Tener el nombre del módulo en cada mensaje de confirmación puede parecer redundante ya que tenemos el nombre en nuestra rama, pero la verdad es que después de fusionar todas las ramas en `dev` (o `master`), mirando el historial de confirmaciones no puede decir a qué rama pertenecía cada una de las confirmaciones. Porque en este momento todos pertenecen a la `dev` (o `master`) rama. Para filtrar fácilmente todos los commits, necesitamos tener el nombre del módulo en algún lugar dentro del mensaje de commit.

El mensaje de confirmación en sí mismo debe completar la oración " *Esta confirmación ...* " gramaticalmente.

Solicitudes de extracción

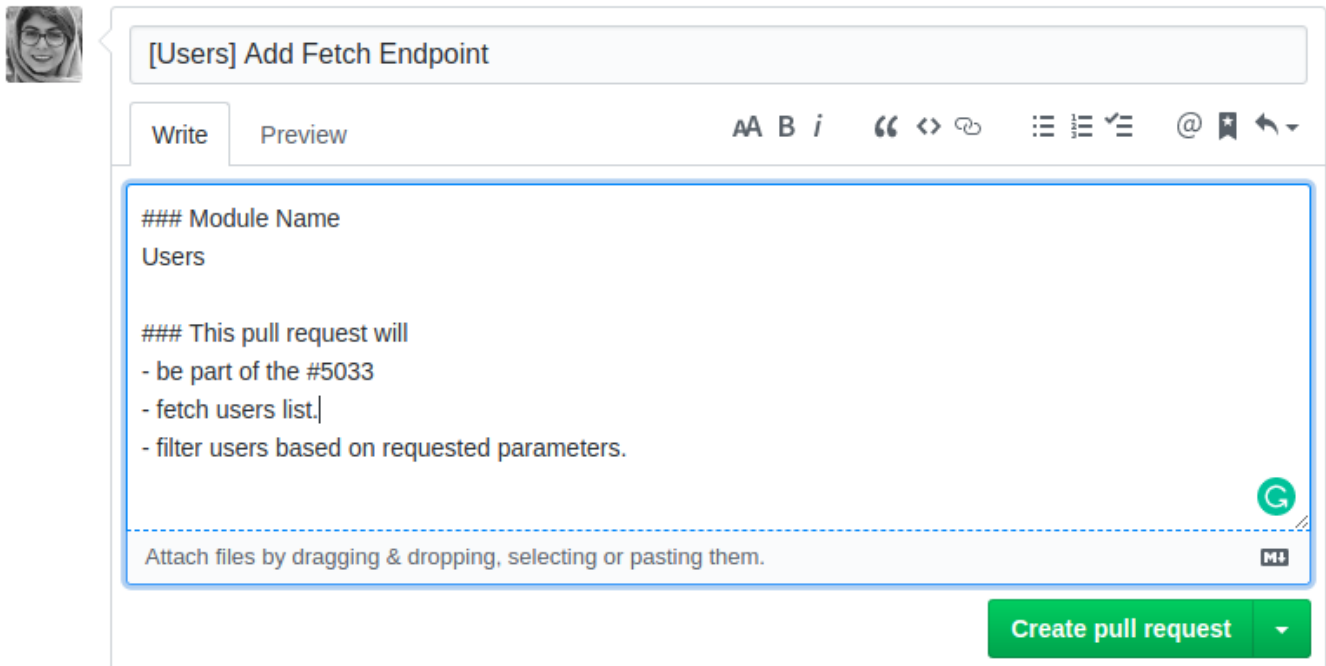
Título

El título de la solicitud de extracción se completa automáticamente con el nombre de la sucursal. En caso de que el nombre de la sucursal sea demasiado genérico, se espera agregar más detalles.

Cuerpo

El cuerpo debe contener detalles sobre la solicitud de extracción. En nuestro flujo de trabajo, las solicitudes de extracción generalmente están relacionadas con un problema existente. Creamos un problema para cada error o característica. Por lo tanto,

vinculamos el problema relacionado dentro del cuerpo de la solicitud de extracción. También mencionamos los aspectos más destacados de los cambios que se realizan dentro de la extracción mediante viñetas. Hemos creado una plantilla para nuestras solicitudes de extracción , para facilitar el proceso.



The screenshot shows a GitHub pull request template interface. At the top, there's a header with a user profile picture and the title "[Users] Add Fetch Endpoint". Below the header, there are two tabs: "Write" and "Preview". The "Write" tab is active. The main text area contains a template for a pull request. It starts with "### Module Name" followed by "Users". Then, it has "### This pull request will" followed by a bulleted list: "- be part of the #5033", "- fetch users list.", and "- filter users based on requested parameters." At the bottom of the text area, there's a dashed line and the text "Attach files by dragging & dropping, selecting or pasting them." To the right of the text area, there's a green circular icon with a white 'G' and a small icon with the letters 'M'. At the bottom right of the form, there's a green button that says "Create pull request" with a dropdown arrow.

revisión

La revisión de código es una de las partes más importantes de nuestro trabajo diario. Una vez que se crea una solicitud de extracción, el creador debe elegir dos pares para revisar el código en función de algunos criterios previamente especificados.

Una vez que los revisores aprueban la extracción, el creador puede fusionarse en dev y eliminar la rama.

. . .

Hemos estado utilizando este flujo de trabajo durante más de un año y lo actualizamos con frecuencia según nuestras necesidades. ¿Utiliza otros flujos de trabajo de git en su equipo? Déjame saber cómo están.

