GITHUB FLOW

Flujo de trabajo estándar del proyecto

Ian Mejías

5 de noviembre 2015

Grupo de estándares

Tabla de contenidos

INTRODUCCIÓN

Antes de continuar...

- ¿Que es Git?

Antes de continuar...

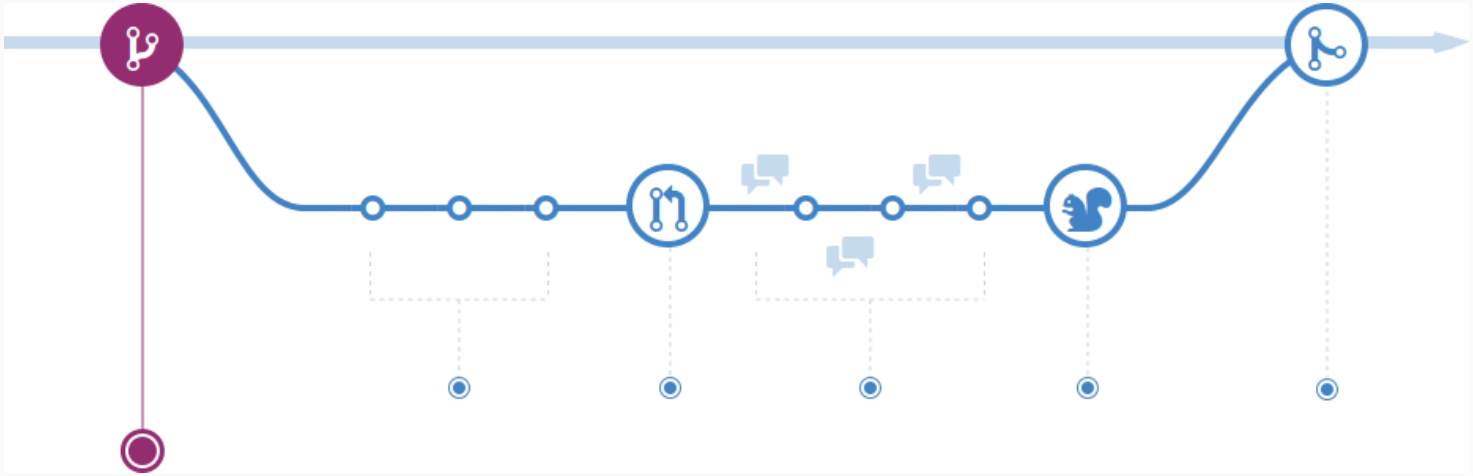
- ¿Que es Git?
- Git no es GitHub

Antes de continuar...

- ¿Que es Git?
- Git no es GitHub
- ¿Que es GitHub flow?

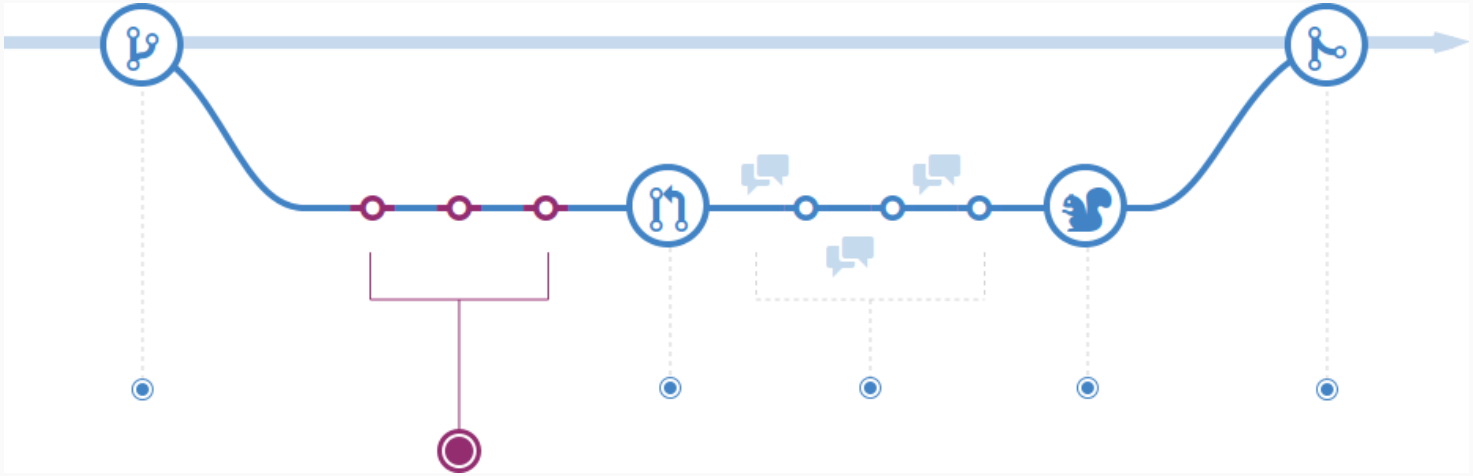
FLUJO DE TRABAJO

Pasos para contribuir



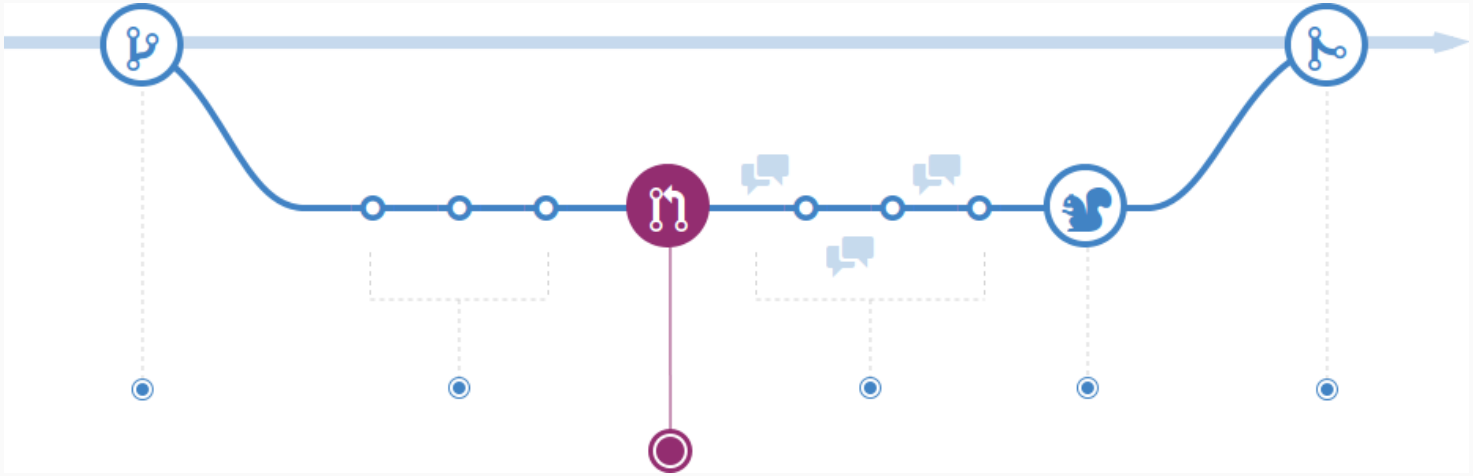
1. Crear una rama sobre algún tópico

Pasos para contribuir



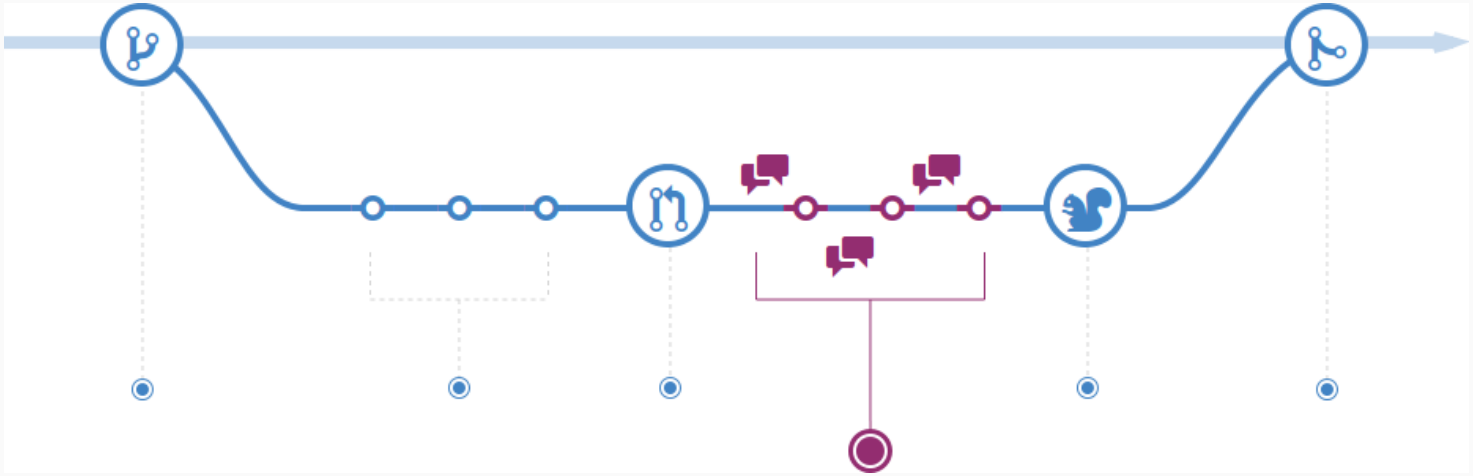
1. Crear una rama sobre algún tópico
2. Hacer commits en la rama

Pasos para contribuir



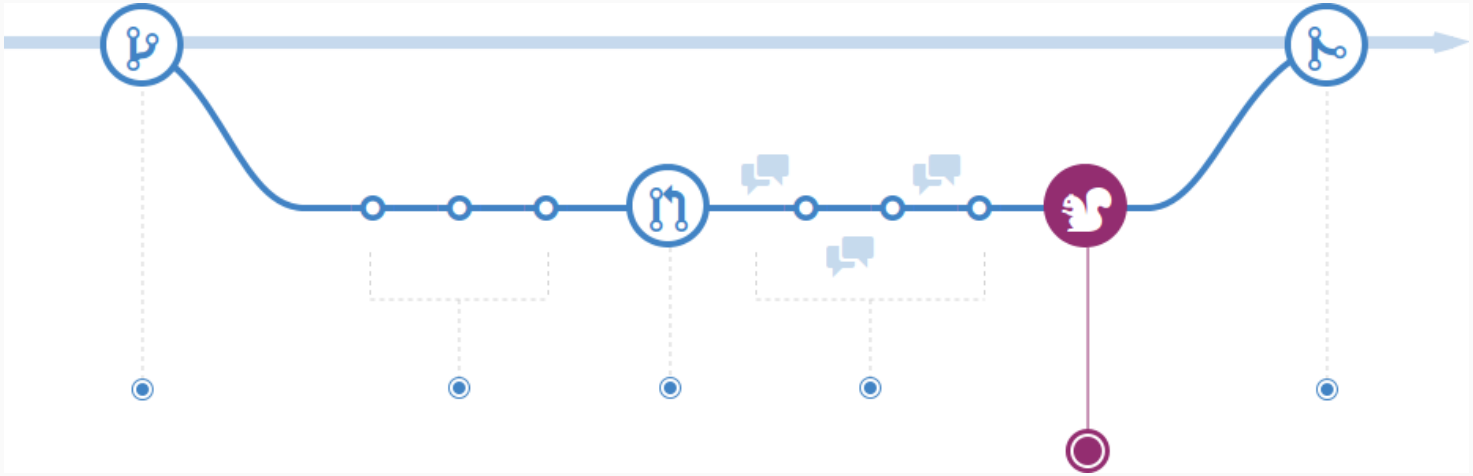
1. Crear una rama sobre algún tópico
2. Hacer commits en la rama
3. Abrir una petición de integración

Pasos para contribuir



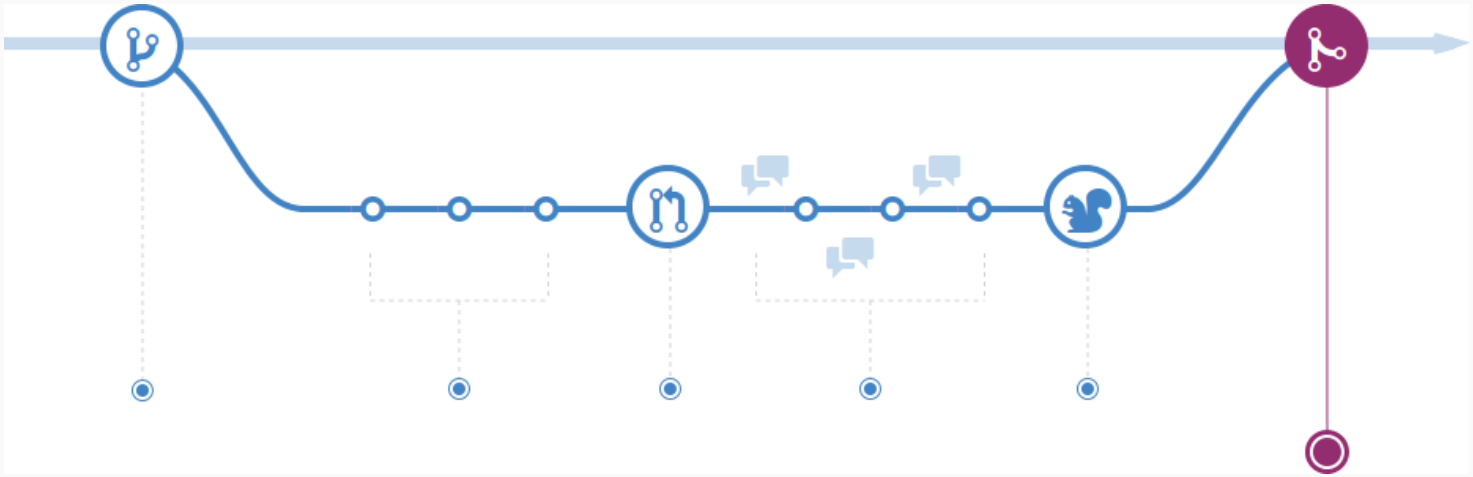
1. Crear una rama sobre algún tópico
2. Hacer commits en la rama
3. Abrir una petición de integración
4. Discutir y revisar el código. Puede que existan cosas que tengas que corregir antes de mezclar con `master`.

Pasos para contribuir



1. Crear una rama sobre algún tópico
2. Hacer commits en la rama
3. Abrir una petición de integración
4. Discutir y revisar el código. Puede que existan cosas que tengas que corregir antes de mezclar con `master`.
5. Probar los cambios implementados.

Pasos para contribuir



1. Crear una rama sobre algún tópico
2. Hacer commits en la rama
3. Abrir una petición de integración
4. Discutir y revisar el código. Puede que existan cosas que tengas que corregir antes de mezclar con `master`.
5. Probar los cambios implementados.
6. Si estamos seguros de que funciona y es estable → mezclar los cambios a la rama `master`

Crear nueva rama

Crear rama I

```
$ git checkout -b <topico> master
```

Para crear una rama con el nombre `topico` y saltar inmediatamente a ella, o

```
$ git branch <topico> master
```

para crearla pero quedarse en la rama actual.

Cuando se crea una nueva rama en tu proyecto, estas creando un ambiente en donde puedes experimentar nuevas ideas. Los cambios que hagas en tu rama no afectan a `master`, así que puedes estar tranquilo sabiendo que solo será mezclada con `master` cuando haya sido revisada por el resto del equipo de trabajo.

OJO: Hay una sola regla que se debe respetar siempre: todo lo que esté en `master` está listo para salir a producción. Por este motivo

Crear rama II

cada vez que se quiera trabajar en algo se debe crear la rama desde `master`.

Las ramas deben tener nombres descriptivos como por ejemplo `application-navbar`, `perfil-profesor`. Así todos pueden ver en que se está trabajando.

Hacer cambios en la rama

Agregar archivos a la confirmación

Luego de hacer algunos cambios se deben agregar los archivos que se quieren incluir en el commit.

```
$ git add <archivo>
$ git add <carpeta> #agrega todos los archivos de la carpeta
$ git add *.js      #agrega todos los archivos que terminen en .js
...
```

Los commits son considerados como unidades de cambio independientes permitiendonos hacer un *rollback* de los cambios hechos. Es por esta razón que no se recomienda agregar todos los archivos cambiados con:

```
$ git add .
```

sino solamente los archivos que tengan que ver con el cambio que vas a hacer.

Confirmar cambios

Existen dos formas de confirmar los cambios

Confirmar cambios

Existen dos formas de confirmar los cambios

1. Escribiendo el mensaje en la misma linea de comandos

   ```
   $ git commit -m <mensaje>
   ```

   El mensaje debe ser autoexplicativo. Se debe preceder con `-m`

Confirmar cambios

Existen dos formas de confirmar los cambios

1. Escribiendo el mensaje en la misma linea de comandos

   ```
   $ git commit -m <mensaje>
   ```

   El mensaje debe ser autoexplicativo. Se debe preceder con `-m`
2. Abriendo algún editor de texto para explicar con mayor detalle el commit

   ```
   $ git commit
   ```

   El mensaje que se escribe en el commit debe tener un cierto formato.

Formato del mensaje

```
Concisa, no mas de 50 carácteres.

Texto explicativo mas detallado (si es necesario) de aproximadamente 72
carácteres de ancho. En algunos contextos, la primera linea es tratada
como el asunto de un correo electrónico y el resto del texto como el
cuerpo. La línea en blanco que separa el sumario del cuerpo siempre
debe existir, a menos que no se escriba el cuerpo.

Los párrafos se separan por una linea en blanco.

Escribe el mensaje del commit en tercera persona tiempo presente. Se
debe escribir: "Se arregla bug del botón enviar encuesta" y no "Arreglé
el bug del botón enviar encuesta" o "Se arregló el bug..."

 - Las viñetas empiezan con un guión '-' o asterisco '*', para
 expresar una lista de cosas que abarca el commit.

 - Y deben separarse por una linea en blanco.
```

Pull request

Crear un pull request

Las peticiones de integración inician una discusión acerca de los commits que hiciste, como entradas de un foro. Puedes abrir un *pull request* en cualquier etapa del proceso de desarrollo:

Crear un pull request

Las peticiones de integración inician una discusión acerca de los commits que hiciste, como entradas de un foro. Puedes abrir un *pull request* en cualquier etapa del proceso de desarrollo:

- Cuando no tienes nada de código y quieres compartir ideas o mostrar maquetas

Crear un pull request

Las peticiones de integración inician una discusión acerca de los commits que hiciste, como entradas de un foro. Puedes abrir un *pull request* en cualquier etapa del proceso de desarrollo:

- Cuando no tienes nada de código y quieres compartir ideas o mostrar maquetas
- Cuando estés atascado y necesites ayuda o consejos.

Crear un pull request

Las peticiones de integración inician una discusión acerca de los commits que hiciste, como entradas de un foro. Puedes abrir un *pull request* en cualquier etapa del proceso de desarrollo:

- Cuando no tienes nada de código y quieres compartir ideas o mostrar maquetas
- Cuando estés atascado y necesites ayuda o consejos.
- O cuando hayas terminado y quieres que alguien revise tu trabajo

Crear un pull request

Las peticiones de integración inician una discusión acerca de los commits que hiciste, como entradas de un foro. Puedes abrir un *pull request* en cualquier etapa del proceso de desarrollo:

- Cuando no tienes nada de código y quieres compartir ideas o mostrar maquetas
- Cuando estés atascado y necesites ayuda o consejos.
- O cuando hayas terminado y quieres que alguien revise tu trabajo

Usando el sistema de *@menciones* puedes pedir *feedback* a un integrante o equipo en específico.

Ejemplo de pull requests

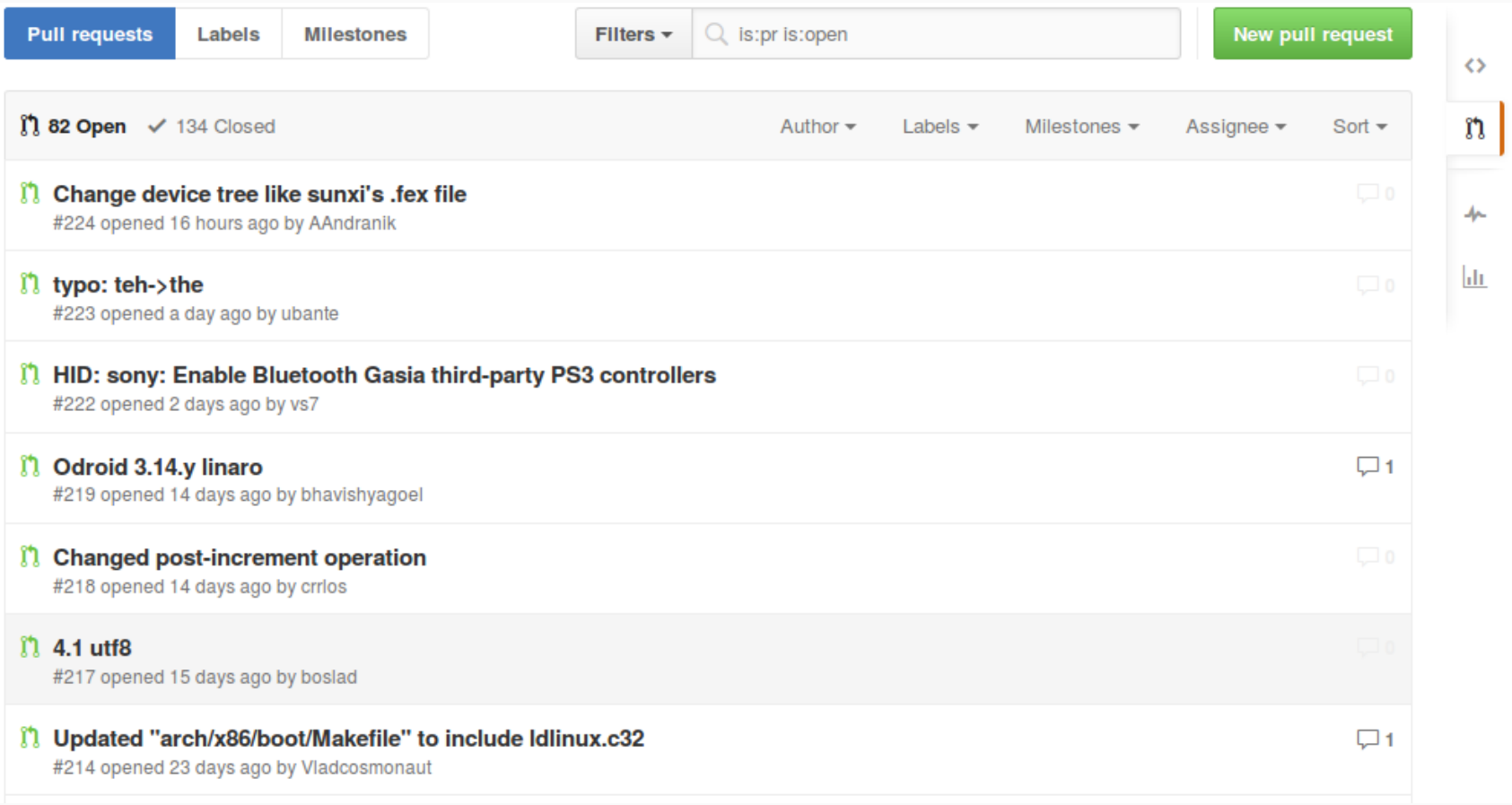


Figura 1: Lista de pull request del kernel de Linux

Recibir feedback

Discusión

Una vez que ha sido abierto un pull request, el equipo o persona que revisa tu código puede tener dudas o comentarios.

Discusión

Una vez que ha sido abierto un pull request, el equipo o persona que revisa tu código puede tener dudas o comentarios.

Puede que el código no encaje con el estilo del proyecto

Discusión

Una vez que ha sido abierto un pull request, el equipo o persona que revisa tu código puede tener dudas o comentarios.

Puede que el código no encaje con el estilo del proyecto … o que falten hacer pruebas unitarias

Discusión

Una vez que ha sido abierto un pull request, el equipo o persona que revisa tu código puede tener dudas o comentarios.

Puede que el código no encaje con el estilo del proyecto … o que falten hacer pruebas unitarias … o quizás está todo perfecto y listo para integrar.

Discusión

Una vez que ha sido abierto un pull request, el equipo o persona que revisa tu código puede tener dudas o comentarios.

Puede que el código no encaje con el estilo del proyecto … o que falten hacer pruebas unitarias … o quizás está todo perfecto y listo para integrar.

Los *pull request* están diseñados para incentivar y capturar este tipo de conversaciones.

Verificar

Verificar los cambios en producción

Básicamente es probar los cambios, como si su rama fuera la rama principal. En nuestro caso muy puntual sería correr el server (verificando previamente que estamos en nuestra rama)

```
$ rails s
```

Y probar las nuevas funcionalidades que implementamos.

Mezclar

Mezclar los cambios hacia `master`

Luego que tus cambios han sido probados en producción, es hora de mezclar tu código con la rama `master`.

Una vez que se ha mezclado, los *pull request* guardan un registro de los cambios históricos a tus commits.

TIP: Incorporando ciertas palabras claves en el texto de tu *pull request* puedes asociar incidencias al código. Cuando un *pull request* es mezclado, las incidencias (*issues*) asociadas, también son cerradas. Por ejemplo escribiendo `Closes #32` cerraría la incidencia numero 32 en el repositorio. Para más información ver *Closing an issue in the same repository* en la ayuda de GitHub.

¿Preguntas?

BIBLIOGRAFÍA

Referencias

- Chacon, S. (2011). Github flow. [Online; accedido el 16 de noviembre de 2015]. Recuperado desde http://scottchacon.com/2011/08/31/github-flow.html
- GitHub. (2013). Understanding the GitHub Flow. [Online; accedido el 16 de noviembre de 2015]. Recuperado desde https://guides.github.com/introduction/flow/