Aprende GIT

Información y experiencias sobre el uso de git

¿Qué es git-flow?

Empezamos una serie de artículos sobre git-flow, conjunto de extensiones de git que facilitan la gestión de ramas y flujos de trabajo.

Si quieres seguir esta serie, debes disponer de una máquina con git instalado:

- Windows: msysgit que puedes descargar de este enlace
- Mac: a través de homebrew o macports
- Linux: a través del gestor de paquetes de tu distribución

Flujos de trabajo

Hace unos días participé en el Open Space de Calidad del Software organizado en Madrid este mes de febrero. En la reunión se abordaron varios temas que iban desde responder a preguntas como ¿qué se calidad del software? ¿cuánto cuestan los tests funcionales? o ¿cómo hacer testing de desarrollos para dispositivos móviles? pasando por otros tan exóticos como el Pirata Roberts, llegando incluso a plantearse hasta la eliminación de los responsables de calidad de la faz de la tierra.

En casi todas las conversaciones en las que tuve la oportunidad de participar había un denominador común: las ramas. Se hablaba de ramas para hacer hot-fixes urgentes, ramas para desarrollar nuevas versiones separadas de las ramas maestras donde está la versión en producción. Ramas para probar nuevas versiones, ramas y repositorios para trabajar con proveedores externos, ramas para hacer pruebas en pre-producción, ramas para que los departamentos de calidad hagan sus pruebas antes de liberar nuevas versiones. Con git podemos crear ramas “como churros” y ese fin de semana tuve la oportunidad de compartir con varios colegas de profesión cómo utilizar las ramas para hacer el bien. Sin embargo, esta facilidad para crear ramas también se puede utilizar para hacer el mal y sembrar el terror. Más de una vez he visto ramas creadas sin ningún criterio, sin ningún flujo de información detrás que las sustente. Esta situación suele llevar al repositorio al caos más absoluto.

Para no acabar en el caos, debemos establecer unas “reglas del juego” que todo el equipo debe respetar. Aunque a grandes rasgos casi todos los proyectos pueden utilizar unas reglas de base comunes, las reglas deben ser flexibles para adaptarse a los cambios que puedan surgir en el tablero de juego; al fin y al cabo, las necesidades y particularidades de cada equipo, empresa o proyecto no son las mismas.

¿Y cuáles son estas reglas base comunes? En enero de 2010 Vincent Driessen publicó en su blog un artículo en el que compartía con la comunidad un flujo de trabajo que a él le estaba funcionando: “A successful Git

branching model". Como él mismo cuenta en el artículo (te recomiendo encarecidamente que lo leas) Vincent propone una serie de "reglas" para organizar el trabajo del equipo.

Ramas master y develop

producción

- Rama develop: rama en la que está el código que conformará la siguiente versión planificada del proyecto

Cada vez que se incorpora código a master, tenemos una nueva versión.

Además de estas dos ramas, Se proponen las siguientes ramas auxiliares:

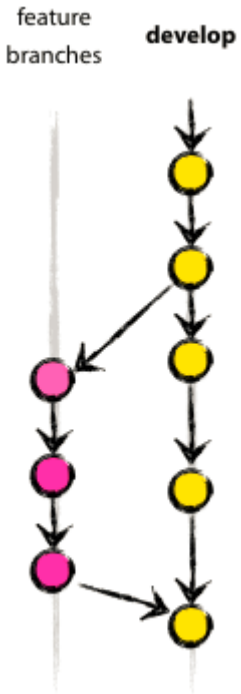
- Feature
- Release
- Hotfix

Cada tipo de rama, tiene sus propias reglas, que resumimos a continuación.

Feature or topic branches

- Se originan a partir de la rama develop.
- Se incorporan siempre a la rama develop.
- Nombre: cualquiera que no sea master, develop, hotfix-* o release-*

Estas ramas se utilizan para desarrollar nuevas características de la aplicación que, una vez terminadas, se incorporan a la rama develop.



fuente: nvie
http://nvie.com/posts/a-successful-git-branching-model/

Release branches

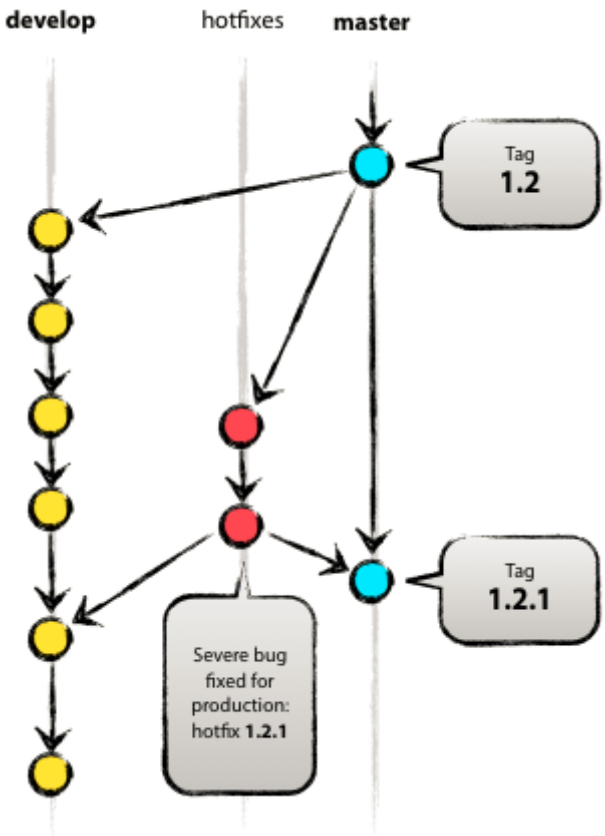
- Se originan a partir de la rama develop

últimos ajustes y se corrigen los últimos bugs antes de pasar el código a producción incorporándolo a la rama master.

Hotfix branches

- Se origina a partir de la rama master
- Se incorporan a la master y develop
- Nombre: hotfix-*

Esas ramas se utilizan para corregir errores y bugs en el código en producción. Funcionan de forma parecida a las Releases Branches, siendo la principal diferencia que los hotfixes no se planifican.



fuente http://nvie.com/posts/a-successful-git-branching-model/

¿Qué es git-flow?

Si queremos implementar este flujo de trabajo, cada vez que queramos hacer algo en el código, tendremos que crear la rama que corresponda, trabajar en el código, incorporar el código donde corresponda y cerrar la rama. A lo largo de nuestra jornada de trabajo necesitaremos ejecutar varias veces al día los comandos git, merge, push y pull así como hacer checkouts de diferentes ramas, borrarlas, etc. Git-flow son un conjunto de extensiones que nos ahorran bastante trabajo a la hora de ejecutar todos estos comandos, simplificando la gestión de las ramas de nuestro repositorio.

La flexibilidad de git...y el sentido común

Las “reglas” que Vincent plantea en su blog son un ejemplo de cómo git nos permite implementar un flujo de trabajo para nuestro equipo. Estas no son reglas absolutas, bien es cierto que pueden funcionar en un gran número de proyectos, aunque no siempre será así. Por ejemplo ¿qué pasa si tenemos que mantener dos o tres versiones diferentes de una misma aplicación? digamos que tenemos que mantener la versión 1.X, la 2.X y la 3.X. El tablero de juego es diferente así que necesitaremos ampliar y adaptar estas reglas

sentido común será nuestra mejor aliada para responder las preguntas que nos surjan durante el camino.

Referencias:

- A successful branching model
- git-flow en github

Esta entrada se publicó en workflow y está etiquetada con git-flow en 28/02/2013 [http://aprendegit.com/que-es-git-flow/] por alfonso.

---

14 pensamientos en “¿Qué es git-flow?”

Pingback: git-flow: la rama develop y uso de feature branches | Aprende GIT

Pingback: git-flow: Resumen y conclusiones | Aprende GIT

Pingback: ¿Qué es git-flow? | Aprende GIT |...

Ramiro
22/08/2014 en 12:20

Buen artículo.

Viendo que esta página está muy bien posicionada en Google, te propondría un cambio para no generar confusión.

Cuando hablas de las ramas release-* y hotfix-* dices que “Se incorporan a la master o develop” cuando creo que deberías decir “Se incorporan a la master **y** develop”.

En ambos casos es obvio que se incorporan a master porque son para producción, pero también hay que hacerlo a develop para incorporar a desarrollo los posibles cambios que se hayan producido en la propia

rama.

Hola Ramiro:

Gracias por el gazapo, efectivamente se incorporan a master y develop. Lo corrijo ahora mismo.

Un saludo,

Alfonso

Jose Maria
07/09/2014 en 19:32

Yo también propondría un cambio a esta web: cuando partes una palabra en sílabas mediante un guión intenta que el guión no parta una sílaba en dos: cóm-o, cód-igo, chec-kouts.
Da muy mala imagen a la web el no usar las reglas del castellano correctamente, y más cuando no hay necesidad real de partir palabras existiendo el 'texto justificado'.

¡Un saludo!

admin
08/09/2014 en 08:29

Hola Jose María:

Te agradezco la apreciación, pondré más cuidado en ese detalle.

Muchas gracias por el feedback,

Alfonso

Pingback: Usando Git con git-flow | Aprende GIT

Pingback: Estrategia de branching y release con Git flow | Coda

Pingback: Desarrollo Front-end (web y dispositivos) » Uso de repositorios Git con SourceTree

Manuel Aguilar
14/10/2016 en 23:37

Buen blog, muy claro

Pingback: Entrevista a Mauricio Gelves

Pingback: Entrevistas a profesionales de WordPress: Mauricio Gelves