**Luca Mezzalira**

Arquitecto de sistemas en DAZN | Google Developer Expert | London Community Manager de JavaScript



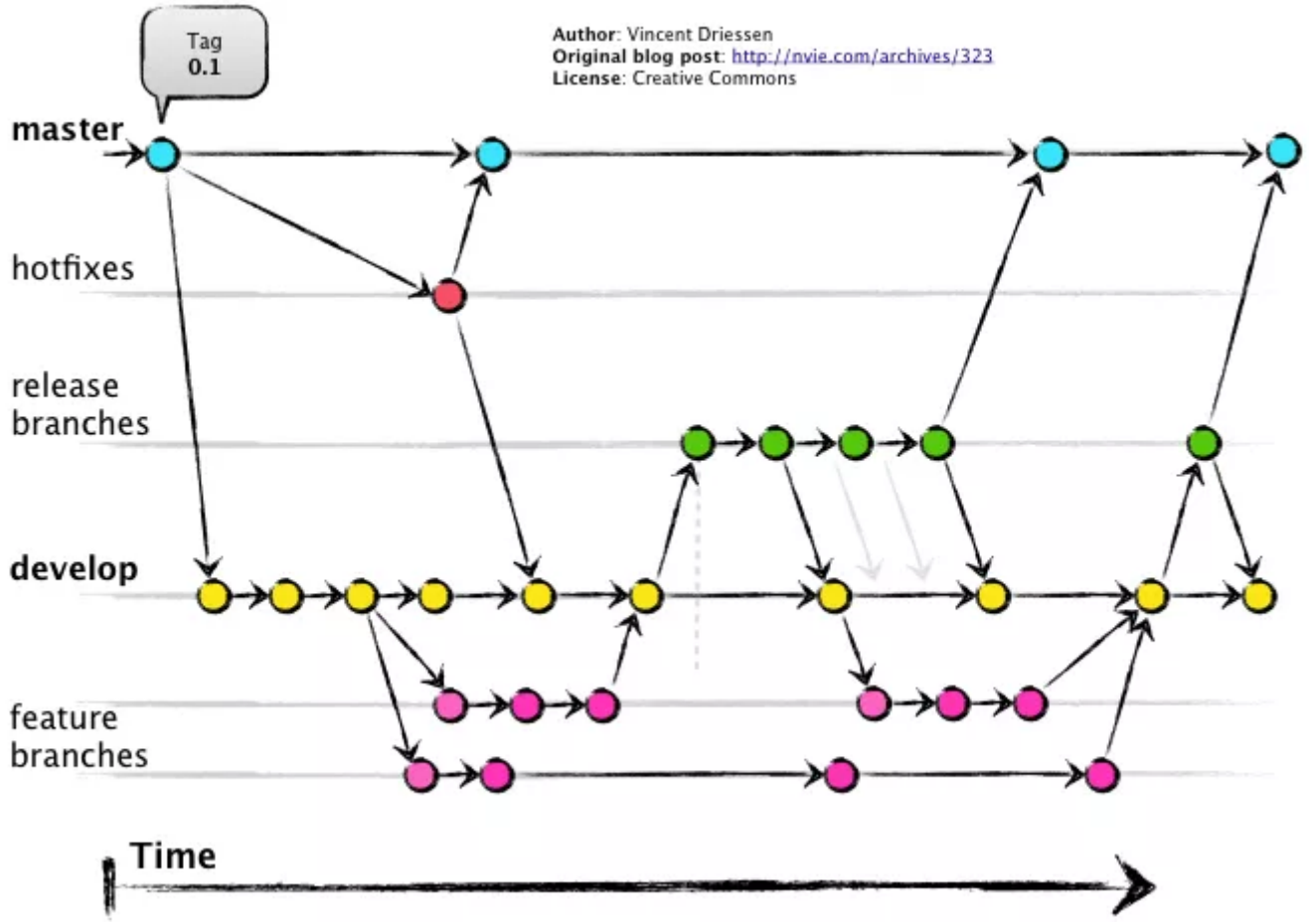
# Git Flow vs Github Flow

lucamezzalira Agile , Recurso , Consejos y Trucos 10 de marzo de 20146 de octubre de 2014 4 minutos

Recientemente, he dedicado tiempo a estudiar una buena forma de administrar proyectos de software con GIT.
Realmente leí una gran cantidad de publicaciones de blog para verificar diferentes puntos de vista y descubrir cuál es la mejor técnica para usar en diferentes situaciones.

Las principales formas de administrar un software en GIT son: Git Flow y Github Flow.
Estos 2 métodos realmente pueden ayudarlo a administrar su proyecto y optimizar su flujo de trabajo en el equipo.
Veamos las diferencias entre ellos.

## Git Flow




*git flow*

Git Flow trabaja con diferentes ramas para administrar fácilmente cada fase del desarrollo de software, se sugiere que se use cuando su software tenga el **concepto de "lanzamiento"** porque, como puede ver en el esquema anterior, no es la mejor decisión cuando trabaja en el entorno de *Entrega continua* o *Despliegue continuo* en el que falta este concepto.
Otro buen punto de este flujo es que se adapta perfectamente cuando trabajas en equipo y uno o más desarrolladores tienen que colaborar con la misma función.
Pero echemos un vistazo más de cerca a este modelo.

Las ramas principales en este flujo son:

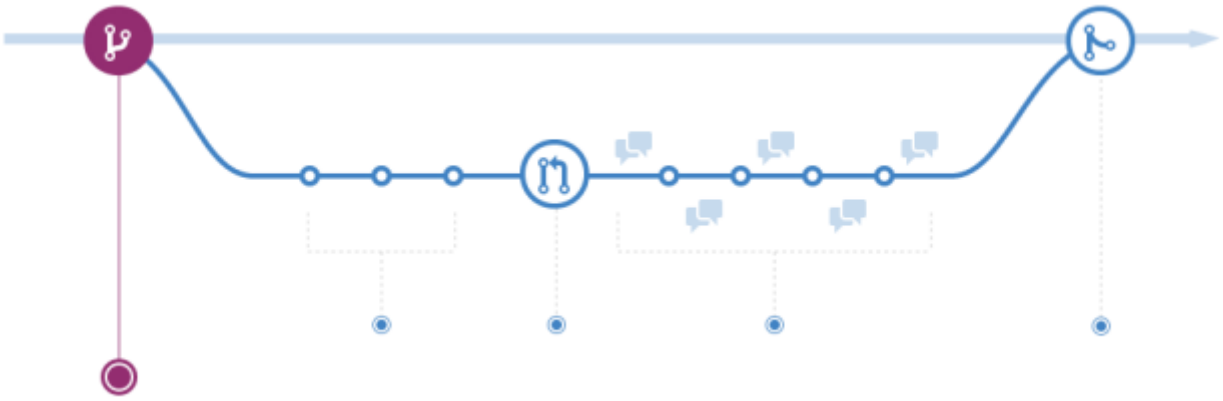
- *dominar*
- *desarrollar*
- *caracteristicas*
- *revisión*
- *lanzamiento*

Cuando clones un repositorio de GIT en tu carpeta local, inmediatamente debes crear una rama desde el maestro llamado desarrollo, esta rama será la principal para el desarrollo y donde todos los desarrolladores de un equipo trabajarán para implementar nuevas características o corregir errores. antes del lanzamiento.
Cada vez que un desarrollador necesita agregar una nueva característica, creará una nueva rama de desarrollo que le permita trabajar adecuadamente en esa función sin comprometer el código para las otras personas en el equipo en la rama de desarrollo.
Cuando la función estará lista y probada, puede ser reestadificada dentro de la rama de desarrollo, nuestro objetivo es tener siempre una versión estable de la rama de desarrollo porque fusionamos el

código solo cuando la nueva característica se completa y está funcionando.
Cuando todas las funciones relacionadas con una nueva versión se implementan en la rama de desarrollo, es hora de derivar el código a la rama de publicación donde allí comenzará a probarse correctamente antes de la implementación final.
Cuando se bifurca el código de la versión de desarrollo a la versión anterior, se debe evitar agregar nuevas características, pero solo se deben corregir los errores dentro del código de la rama de la versión hasta que se cree una rama de versión estable.
Al final, cuando esté listo para implementar en vivo su proyecto en vivo, etiquetará la versión dentro de la rama principal para que pueda tener todas las versiones diferentes que publica cada semana.
Aparentemente puede parecer mucho, pero seguro que es bastante seguro y te ayuda a evitar errores o problemas cuando lanzas, obviamente para lograr todas estas tareas puedes encontrar en línea muchos guiones que podrían ayudarte a trabajar con Git Flow en el línea de comandos o si lo prefiere, puede usar herramientas visuales como SourceTree by Atlassian (http://www.sourcetreeapp.com/) que hacen que el trabajo sucio para usted, por lo que debe seguir las instrucciones dentro del software para administrar todas las ramas, por esta razón también preparé un corto video para ver cómo usar este flujo con SourceTree


Puede profundizar más en este flujo leyendo estos 2 artículos interesantes: nvie blog (http://nvie.com/posts/a-successful-git-branching-model/) o documentación de datasift (http://datasift.github.io/gitflow/IntroducingGitFlow.html) .

# Github Flow



(http://guides.github.com/overviews/flow/)

Entonces, ¿crees que Github está trabajando con Git Flow? ¡Claro que no! (Honestamente, ¡me sorprendió mucho cuando lo leí!)
De hecho, están trabajando con un entorno de despliegue continuo en el que no existe el concepto de "lanzamiento" porque cada vez que terminan de preparar una nueva característica se activan inmediatamente (después del cadena de automatización (http://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff) completa creada en el entorno).
Los conceptos principales detrás del flujo de Github son:

- Cualquier cosa en la `master` rama es desplegable
- Para trabajar en algo nuevo, crear una rama descriptiva llamado fuera de `master` (es decir: `new-oauth2-scopes` )
- Comprométase con esa sucursal localmente y periódicamente envíe su trabajo a la misma sucursal con nombre en el servidor
- Cuando necesite comentarios o ayuda, o si cree que la sucursal está lista para fusionarse, abra una solicitud de extracción
- Después de que otra persona haya revisado y firmado la función, puede fusionarla en principal
- Una vez que se fusiona y se empuja a 'maestro', puede y *debe* implementar de inmediato

Encontré una sorprendente página interactiva (http://guides.github.com/overviews/flow/) donde puedes profundizar en el conocimiento de este método, pero veo que es muy común cuando trabajas en equipos de control de calidad, equipos pequeños o trabajas como autónomo porque es cierto que es un flujo ligero para administrar un proyecto, pero también es bastante claro y seguro cuando desea fusionar su código en la rama principal.
Otro buen recurso sobre Github Flow es la publicación (http://scottchacon.com/2011/08/31/github-flow.html) del blog (http://scottchacon.com/2011/08/31/github-flow.html) hecha por el evangelista de Github Scott Chacon.
También grabé un video sobre cómo usar el flujo de Github con SourceTree:


Si tiene algún otro método para administrar su proyecto en GIT, siéntase libre de compartirlo porque soy bastante interesante para ver cómo suele trabajar con GIT y si hay mejores formas de trabajar con él y si tiene algún otro comentario o pregunta. estoy aquí para ti!



**Tagged:**
Ágil,
rama,
derivación,
entrega continua,

despliegue continuo,
integración continua,
Programación extrema,
git,
git flow,
github,
flujo de github,
mercurial,
desarrollo de software,
svn,
equipo

# Publicado por lucamezzalira

*Soy un arquitecto de soluciones italiano con 13 años de experiencia, un experto en desarrollo de Google en tecnologías web y administrador de la comunidad Javascript de Londres ( www.londonjs.uk) Tuve la oportunidad de trabajar en proyectos de vanguardia para dispositivos móviles (iOS, Android, Blackberry), computadoras de escritorio, web, televisores, decodificadores y dispositivos integrados. Creo que la mejor manera de usar cualquier lenguaje de programación es dominar sus modelos, es por eso que pasé mucho tiempo estudiando e investigando sobre temas como OOP, programación funcional y programación reactiva. Con estas habilidades, puedo intercambiar fácilmente entre diferentes lenguajes de programación aplicando las mejores prácticas aprendidas e impulsando a cualquier equipo al éxito. Soy un líder natural, enfoque en la entrega, solucionador de problemas y cambiador de juego. Utilizo mi pasión en cualquier aspecto de mi trabajo, desde la definición de flujo hasta los procesos de automatización. Intento cubrir cualquier detalle para mejorar los estándares de la compañía, empoderar a los equipos y entregar los resultados esperados. En mi tiempo libre, Escribí para revistas y editores técnicos nacionales e internacionales; también soy revisor técnico de Packt Publishing, Pragmatic Bookshelf y O'Reilly. Fui disertante en: webinars de O'Reilly, O'Reilly Solutions Architect (San Francisco), O'Reilly Oscon (Londres), Voxxed Days (Belgrad), NDC London (Londres), JSDay (Verona), CybercomDev (Łódź) , Jazoon Conference (Berna), JDays (Göteborg), Codemotion (Milán y Amsterdam), FullStack Conference (Londres), React London UG (Londres), Scrum Gathering (Praga), Agile Cymru (Cardiff), Scotch on the rocks (Edimburgo) & London), 360Max (San Francisco), PyCon (Florencia), Lean Kanban Conference (Londres), Flash Camp (Milán), Adobe Creative Suite CS 5.5 - Lanzamiento del evento (Milán), HFWAA (Milán, Turín, Padua, Bari, Florencia), Mobile World Congress (Barcelona) También soy revisor técnico de Packt Publishing, Pragmatic Bookshelf y O'Reilly. Fui disertante en: webinars de O'Reilly, O'Reilly Solutions Architect (San Francisco), O'Reilly Oscon (Londres), Voxxed Days (Belgrad), NDC London (Londres), JSDay (Verona), CybercomDev (Łódź) , Jazoon Conference (Berna), JDays (Göteborg), Codemotion (Milán y Amsterdam), FullStack Conference (Londres), React London UG (Londres), Scrum Gathering (Praga), Agile Cymru (Cardiff), Scotch on the rocks (Edimburgo) & London), 360Max (San Francisco), PyCon (Florencia), Lean Kanban Conference (Londres), Flash Camp (Milán), Adobe Creative Suite CS 5.5 - Lanzamiento del evento (Milán), HFWAA (Milán, Turín, Padua, Bari, Florencia), Mobile World Congress (Barcelona) También soy revisor técnico de Packt Publishing, Pragmatic Bookshelf y O'Reilly. Fui disertante en: webinars de O'Reilly, O'Reilly Solutions Architect (San Francisco), O'Reilly Oscon (Londres), Voxxed Days (Belgrad), NDC London (Londres), JSDay (Verona), CybercomDev (Łódź) , Jazoon Conference (Berna), JDays (Göteborg), Codemotion (Milán y Amsterdam), FullStack Conference (Londres), React London UG (Londres), Scrum Gathering (Praga), Agile Cymru (Cardiff), Scotch on the rocks (Edimburgo) & London), 360Max (San Francisco), PyCon (Florencia), Lean Kanban Conference (Londres), Flash Camp (Milán), Adobe Creative Suite CS 5.5 - Lanzamiento del evento (Milán), HFWAA (Milán, Turín, Padua, Bari, Florencia), Mobile World Congress (Barcelona) Ver todos los mensajes de lucamezzalira*

# 13 pensamientos sobre " Git Flow vs Github Flow "

Pingback: Git Flow contra Github Flow | Aventuras de Umbraco de Ismail
**NoriSte** says:
25 de marzo de 2014 a las 09:26
Thank you Luca for the post!!
I (and I push it to my colleagues) use the first one, it could be a little complicated at the first approach but it ensures you to have everything tracked. I haven't found difficulties because my mind already thinks in the same way. We use SmartGIT and it's embraced the same workflow doing the dirty job for you.
The 'release' approach is also useful in case of publishing tools where you to have to make some steps before publish the project (think about Grunt, the 'build' process is done in the release commit, not in the develop one nor in the master one).
More: the master branch remains clear, tagged, easy to understand and everyone that clones the repo could deploy the latest master commit without worrying about it. But reading the post that approach is moreless the same for the GitHub flow.
The evolution is having GIT installed on the server too, we're working for it 🙂
To give a contest to my comment: we are a web agency with 4/5 developers, usually 2 developers working on the same project.

 Reply
**lucamezzalira** says:
March 25, 2014 at 09:33
Thanks for sharing your thoughts and experience I think it's very useful have this kind of discussions in blogs and forums

 Reply
**derosm2 (@derosm2)** says:
March 2, 2016 at 20:42
Looks like GitHub tweaked github flow a bit so that you deploy your feature branch to production before merging to master. Any thoughts on the changes? Didn't see any place where they explain their reason for the change, or how the implement it.

 Reply
**cemo** says:
May 1, 2016 at 22:43
I think that the main motivation would be a merge into master requires a new build and QA because of being a new build.

 Reply
**daniyal** says:
March 26, 2016 at 10:18
thanx really this article is very helpful !!!

 Reply
**patecone** says:
February 6, 2017 at 12:54

Hi Luca,
In my Company we have adopted a sort of modified GitFlow workflow, becasuse we had wanted to integrate the continuous integration/delivery approach still keeping each development stream isolate into a feature branch. I also started using the Jenkins CI as the build automation tool, specifically the Delivery Pipeline feature, to automate all the building steps from source code to production deployment.
As you are saying Continuous Integration/delivery doesn't really fit with the GitFlow approach, and I must agree on that, although I must say that if you accept lossing something from both approach, it may be working quite well.
One thing that you necessarly loose is testing. I'm not saying that you can't run automated test suites, but you have to focus the testing jobs only on few branches – at least to reduce the CI server's load – like the "develop" or "master" branch.
A second thing that you loose is high fre

 Reply

**patecone** says:
February 6, 2017 at 13:08
Sorry, I posted my reply before reaching the end and reviewing (few typos still there)

As I was saying,

A second thing that you lose is high frequency deployment, because you still need to wait for merging of each feature into the develop branch before start any unit or integration testing session. Finally, of course, what you lose is the release-fixing task, because your automated worklow is no more able to jump-back to an older release to start again the pipeline – I'm not completely sure that "it can't" but in my experience I haven't been able to –

I'm still trying to tune our pipeline two speed but the development-testing-delivery cycle, if someone has suggestion it'll be great to talk about!

Thank you!

 Reply

Pingback: GitHub Flow – Mateusz Mistecki
**panlatent** says:
March 31, 2017 at 03:33
Thank you, this is very helpful to me.

About http://softwareengineering.stackexchange.com/questions/312022/the-trend-of-the-develop-branch-going-away

 Reply

Pingback: Git branching done right with Gitflow & improving code quality with code reviews - Nikola Brežnjak blog
**invntrm** says:
September 20, 2017 at 21:31
You can switch GH default branch from the master to dev and problem solved.

We have adopted GH Flow + Git Flow.
master is a default branch for tags, builds, and our library clients
dev is a default branch for developers and github: CI, merge pull-requests, code reviews, etc.

What am I doing wrong?

 Reply

**lucamezzalira** says:
September 20, 2017 at 21:34
Depende si usted trabaja en implementación continua o con lanzamientos, gitflow es mejor para las versiones porque maneja también las revisiones y la rama de publicación, en cambio el flujo github está más enfocado en la mejora continua y pequeña.

 Respuesta