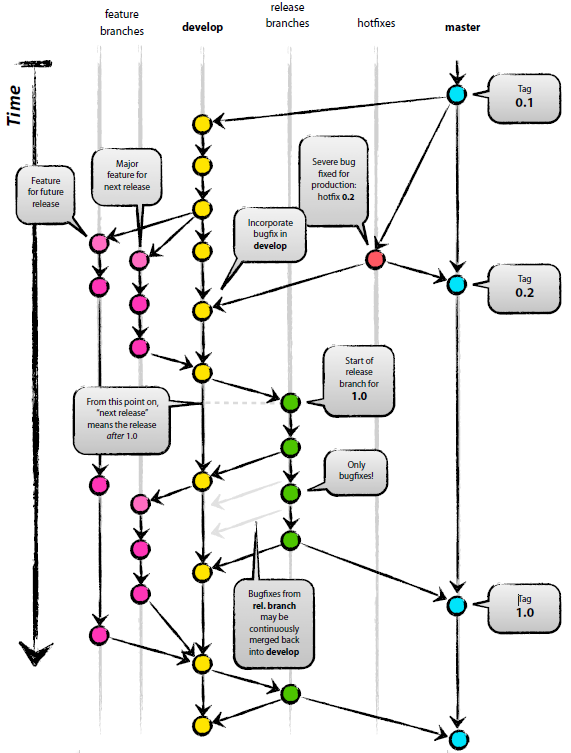
**Configurar GIT para usar Git-flow**

**Git-flow** es un conjunto de extensiones para **git** que proveen comandos de alto nivel para operar repositorios basado en un modelo de ramificaciones con una estructura para el optimo trabajo de distintas ramas, este modelo fue propuesto por Vincent Driessen. **Git-flow** no es más que un conjunto de procedimientos que cada miembro del equipo tiene que seguir para llegar a un proceso de desarrollo de software administrado.



**Master**: Es la rama (*branch*) que tiene la última versión productiva del código. Cada vez que se realice un merged de los cambios en master, por definición se trata de una nueva versión de producción.

**Release**: Es la rama que contiene los nuevos **features** terminados que se van desarrollando para el siguiente lanzamiento (*release*) de forma que al iniciar uno nuevo puedas descargar todos los anteriores por si tienen alguna dependencia. Los **Release** se crean desde la rama de **Develop**. Por ejemplo digamos que la versión actual productiva es la 1.1.5 y tenemos varios cambios a subir y hemos decidido que se convertirá en la versión 1.2 (en lugar de 1.1.6 o 2.0) . Así que damos a la rama **Release** un nombre que refleje el nuevo número de versión.

**Develop**: Es la rama que contiene las características (*features*) en desarrollo en una iteración, esta rama será posteriormente parte de Release mediante un *pull request*. Esta rama podría entenderse como la "rama de integración".

**Feature**: Es la rama que contiene el *feature* en el que estás trabajando personalmente (varios desarrolladores pueden trabajar en un feature), éste debe ser enviado a develop mediante un *full request*, por lo general aprobado por el líder técnico.

**Hotfix**: Es la rama que contiene cambios urgentes sobre master que permiten corregir un bug o resolver un error, éste debe ser enviado a master y se debe notificar a todos los desarrolladores para que puedan actualizar sus ramas.

**Descargar Git**

Descargar git para Windows: [**https://git-scm.com/**](https://git-scm.com/)

Usaremos Git Bash para trabajar con git:

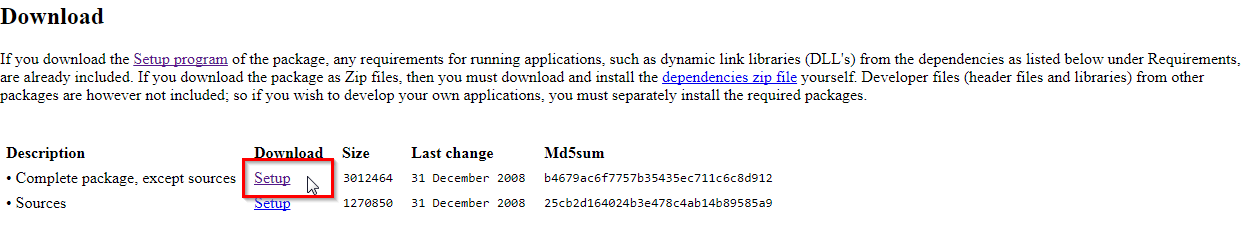


**Instalar Git-Flow en Windows**

Para poder instalar Git-Flow primero se necesita tener instalado **WGET** y **util-Linux**.

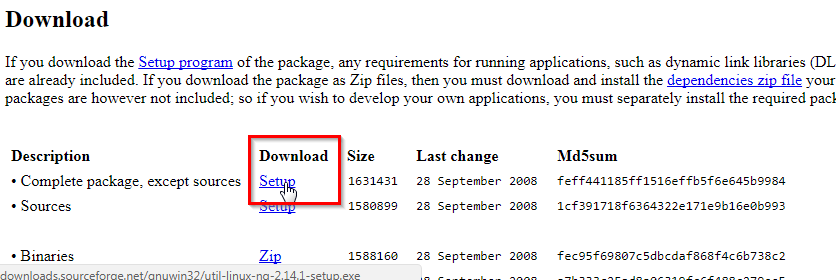
**Wget**

Descargar Wget: [**http://gnuwin32.sourceforge.net/packages/wget.htm**](http://gnuwin32.sourceforge.net/packages/wget.htm)



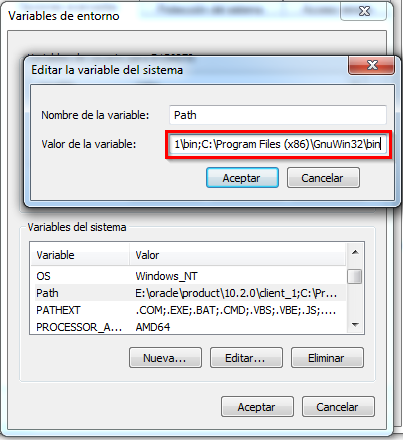
**Util-linux**

Descargar util-linux: [**http://gnuwin32.sourceforge.net/packages/util-linux-ng.htm**](http://gnuwin32.sourceforge.net/packages/util-linux-ng.htm)

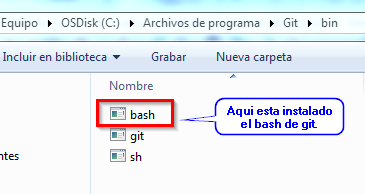


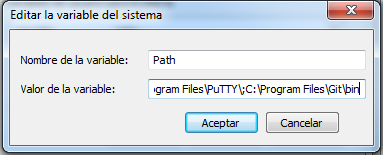
Después de instalar ambos agregar a la variables de entorno del Path: la ruta bin donde se instalo Wget:

Copiar la ruta y pegarla al final del path:



También va ser necesario agregar la ruta donde esta instalado el bash de git (después puedes quitar esta ruta de las variables de entorno, solo es para hacer la instalación de Git-flow )



Copiar la ruta y agregarla a la variable de entorno path (recuerda que después la puedes eliminar):

Una vez hecho esta configuración abrimos una consola de Windows y corremos el siguiente comando:

wget -q -O - --no-check-certificate https://raw.github.com/petervanderdoes/gitflow-avh/develop/contrib/gitflow-installer.sh install stable | bash

Con esto finaliza la instalación de git-flow ahora se podrá usarlo en cualquier proyecto git.

**Usar Git-flow**

Abrimos una consola Git Bash y nos posicionamos en la carpeta donde vamos a tener nuestro repositorio.

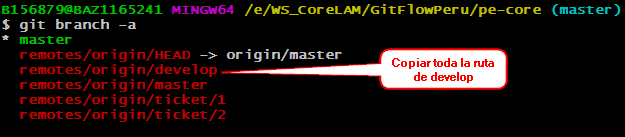
Para comenzar a usar git-flow se necesita inicializar desde dentro de un repositorio git existente.

Los siguientes comandos son para **un repositorio recién clonado y con git-flow ya trabajando**:



Una vez clonado posicionarse en el repositorio y consultamos todas las ramas locales y remotas del repositorio:





El siguiente comando nos moverá a la referencia de la rama remota de develop:



Creamos una rama local con referencia a la rama develop del repositorio remoto. Este comando nos creara la rama local develop y nos posiciona en ella :

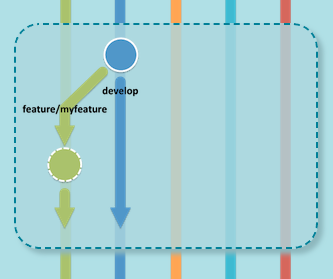


Inicializamos git-flow:



La terminal nos pedirá las configuraciones a utilizar (nombre del branch maestro, desarrollo, prefijo branch feautre, etc.) siempre se recomienda dejar los valores por defecto dando enter.

**Realizar un feature**

****

Con la terminal en el directorio de nuestro repositorio y nuestra área de trabajo limpia.

Si no estamos en la rama local develop nos movemos en ella:



Bajamos los cambios del repositorio remoto de la rama develop:



Iniciamos el feature (el nombre del feature puede ser algo descriptivo de los que se vaya a desarrollar):



Esta acción creara una nueva rama derivada de develop y nos posiciona a esta, estableciéndola como rama de trabajo actual.

Se hacen los cambios pertinentes como vamos a usar los tickets de gitBlit es necesario bajar los cambios antes.

Ahora si se hace commit de nuestros cambios





Cerramos el feature:

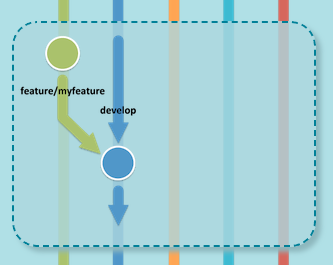


Finaliza el desarrollo de un feature. Realiza lo siguiente:

Fusiona "ejemploFeature" en develop.

Borra la rama "ejemploFeature"

Salta a la rama develop, estableciéndola como una rama de trabajo actual.

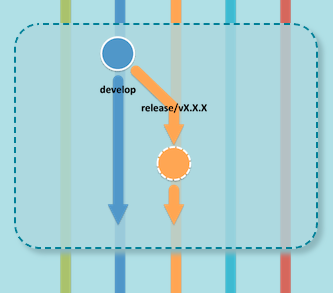


Subimos los cambios a la rama remota develop, en nuestro caso será a través de un ticket que tendremos que esperar a que sea autorizado por el integrador:



Con esto finaliza nuestro feature.

**Comenzar un Release**

****

**NOTA:** En nuestro caso el Release solo lo realizara el integrador ya que el va ser el único que impacte a master, de cualquier forma lo integro para que sea de su conocimiento.

Con la terminal en nuestro repositorio y nuestra área de trabajo limpia.

Nos posicionamos en la rama local develop:



Bajamos los cambios del repositorio remoto de la rama develop:



Nos posicionamos en la rama local master:



Bajamos los cambios del repositorio remoto de la rama master:



Bajamos los tags del repositorio remoto de la rama master:



Consultamos los tags ya utilizados para definir la versión siguiente:



Regresamos la rama local develop:



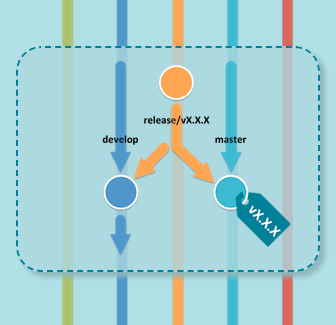
Iniciamos el Release: git flow release start \*tag de la versión siguiente\*. Lo cual crea una rama Release derivada de la rama develop:



De ser necesario se ralizan los cambios para el release.

Finalizar el Release: git flow release finish \*tag de la versión siguiente\*. Al cerrar un Release realiza varias acciones:

* Fusiona la rama release con la rama master.
* Etiqueta el cambio con su nombre.
* Vuelve a fusionar release con la rama develop.
* Borra la rama release.



Nos posicionamos en la rama local master:



Subimos los cambios y tags a la rama remota master:



Nos posicionamos en la rama local develop:

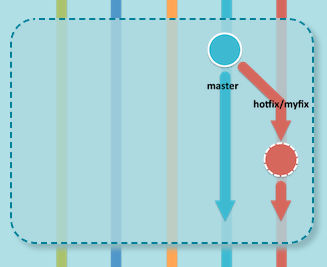


Subimos los cambios a la rama remota develop, en nuestro caso será a través de un ticket que tendremos que esperar a que se autorizado por el integrador:



Con esto finaliza nuestro Release.

**Realizar un hotfix**

****

Con la terminal posicionada en nuestro repositorio y nuestra área de trabajo limpia

Nos posicionamos en la rama local develop:



Bajamos los cambios del repositorio remoto de la rama develop:



Nos posicionamos en la rama local master:



Bajamos los cambios del repositorio remoto de la rama master:



Bajamos los tags del repositorio remoto de la rama master:



Consultamos los tags ya utilizados para definir la versión siguiente:



Iniciamos el hotfix:



Se realizan los cambios para solucionar el problema

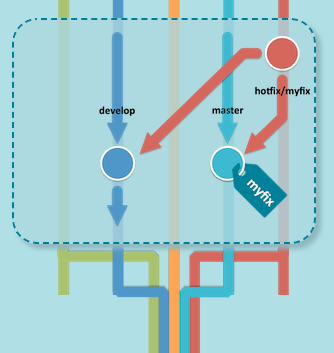
Se realiza el commit de nuestros cambios:





Cerramos el hotfix, al cerrar esta rama se fusiona en las ramas develop y master, luego el cambio en master es etiquetado con el nombre del hotfix:





Nos posicionamos a la ramo local master:



Subimos los cambios y tags a la rama remota master:



Nos posicionamos en la rama local develop:

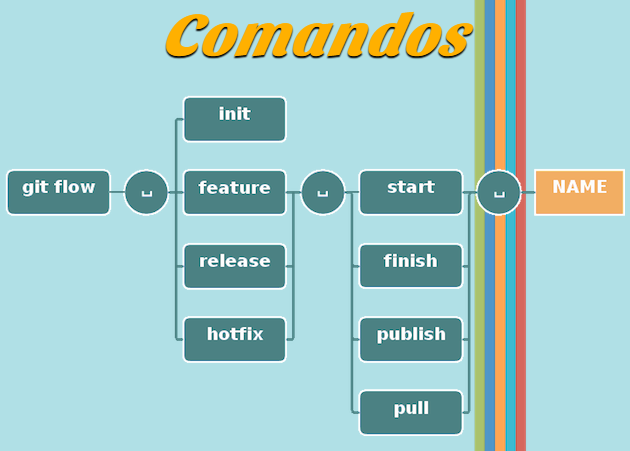


Subimos los cambios a la rama remota develop, en nuestro caso será a través de un ticket que tendremos que esperar a que se autorizado por el integrador:



Consto finaliza nuestro hotfix.

**Anexos**



**Bibliografía**

[**http://nvie.com/posts/a-successful-git-branching-model/**](http://nvie.com/posts/a-successful-git-branching-model/)[**http://danielkummer.github.io/git-flow-cheatsheet/index.es\_ES.html**](http://danielkummer.github.io/git-flow-cheatsheet/index.es_ES.html)[**https://github.com/KarmaPulse/git-flow-recipes**](https://github.com/KarmaPulse/git-flow-recipes)