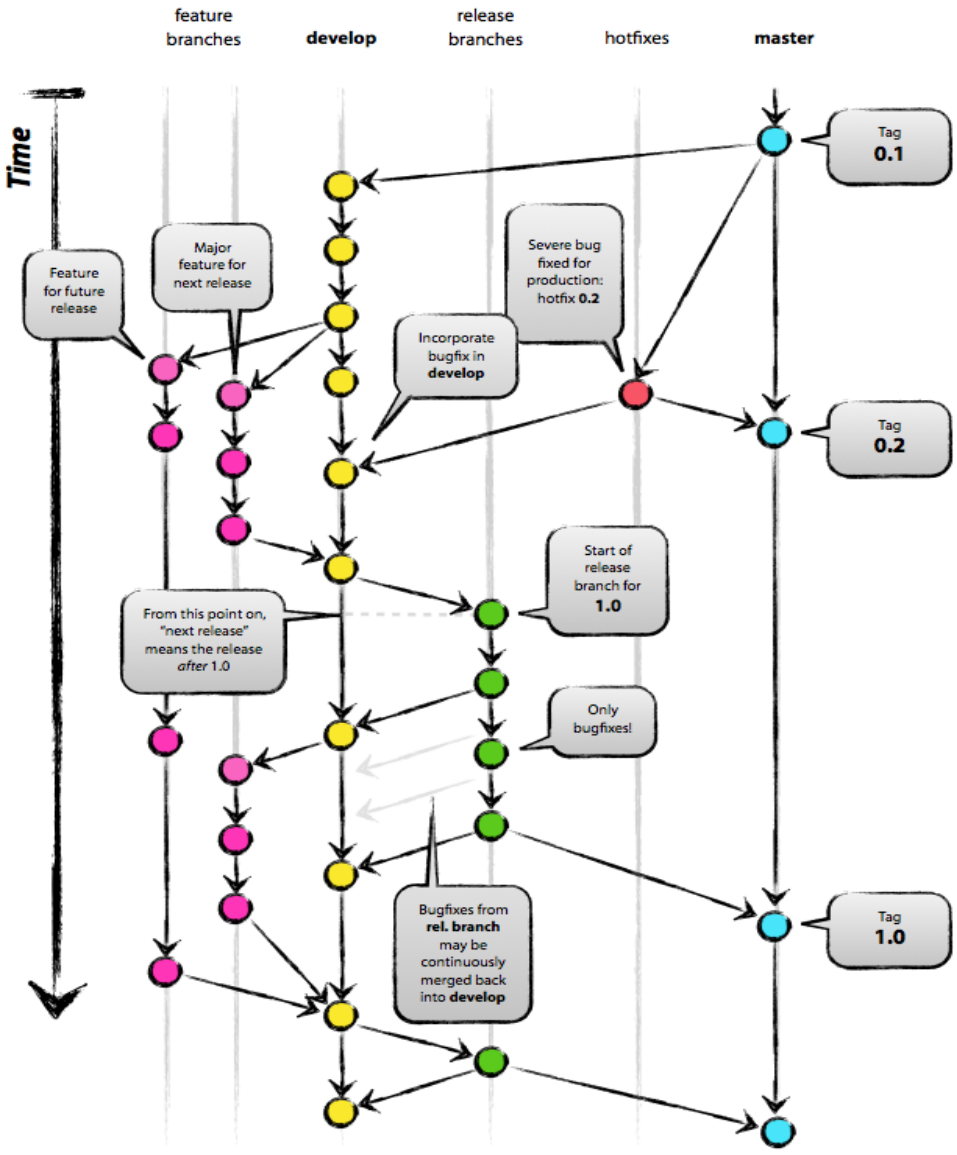
Tema 5 Sistemas de Control de Versiones

1. Flujo de trabajo con Git (git flow)

En la introducción vimos los diferentes esquemas de organización externa de los repositorios (es decir, en lo relativo a los usuarios que componen el equipo de trabajo).

Pero el repositorio en sí también tiene su esquema de organización.

En los ejemplos hemos visto que usábamos una rama máster y creábamos ramas para añadir funcionalidades que luego integrábamos. Es una forma de trabajar de las muchas que hay propuestas, posiblemente la más simple, pero tiene el inconveniente de dejar la rama máster a expensas de una mala actualización y quedarnos sin una rama estable. Por eso, hay otras propuestas mejores que permiten separar el trabajo de desarrollo con el mantenimiento de las versiones estables. Una de las más conocidas es la propuesta por Vincent Driessen y que podemos ver en la figura siguiente.



1.1 Las ramas principales

En este esquema hay dos ramas principales con un tiempo de vida indefinido:

- master (*origin/master*): el código apuntado por *HEAD* siempre contiene un estado listo para producción.
- develop (*origin/develop*): el código apuntado por *HEAD* siempre contiene los últimos cambios desarrollados para la próxima versión del software. También se le puede llamar *rama de integración*. No es necesariamente estable.

Cuando el código de la rama de desarrollo es lo suficientemente estable, se integra con la rama master y una nueva versión es lanzada.

1.2 Las ramas auxiliares

Para labores concretas, pueden usarse otro tipo de ramas, las cuales tienen un tiempo de vida definido. Es decir, cuando ya no son necesarias se eliminan:

- Ramas de funcionalidad (*feature branches*)
- Ramas de versión (*release branches*)
- Ramas de parches (*hotfix*[1] *branches*).

Feature branches

- Pueden partir de: develop
- Deben fusionarse con: develop
- Convención de nombres: feature/NUMissue-*.

Release branches

- Pueden partir de: develop
- Deben fusionarse con: develop y master
- Convención de nombres: reléase/*

Hotfix branches

- Pueden partir de: master
- Deben fusionarse con: develop y master
- Convención de nombres: hotfix/*

[1] Un *hotfix* es una corrección rápida y urgente que se aplica directamente a una versión en producción para solucionar un problema crítico.

2. La extensión flow de Git

Una de las ventajas de Git es que, además, es extensible. Es decir, se pueden crear nuevas órdenes como si de plugins se tratara. Una de las más usadas es gitflow.

2.1 Instalación

Aunque la fuente original de la extensión es del mismo autor del artículo, el código no se encuentra ya muy actualizado y hay un fork bastante más activo en petervanderdoes/gitflow. En el wiki del repositorio están las instrucciones de instalación para distintos sistemas. Una vez instalados tendremos una nueva órden: git flow.

2.2 Uso

Para cambiar a las ramas master y develop, seguiremos usando git checkout, pero para trabajar con las ramas antes indicadas gitflow nos facilita las siguientes órdenes:

git flow init

Inicializa el espacio de trabajo. De forma automática, crea las ramas que necesitamos y permite configurar el nombre de las mismas.

```
$ git flow init

Initialized empty Git repository in ~/project/.git/

No branches exist yet. Base branches must be created now.

Branch name for production releases: [master]

Branch name for "next release" development: [develop]


How to name your supporting branch prefixes?

Feature branches? [feature/]

Release branches? [release/]

Hotfix branches? [hotfix/]

Support branches? [support/]

Version tag prefix? []


$ git branch

* develop

 master
```

Podemos ver que por defecto (usando intro en vez de escribir nada) pone nombres por defecto a cada rama. Con git branch comprobamos que ramas existen y en cual nos encontramos.

git flow feature

Permite crear y trabajar con ramas de funcionalidades.

```
$ git flow feature start feature_branch
```

Así creamos una rama *'feature/feature_branch'* y nos mueve automáticamente a ella. En esta haremos los cambios que queramos en nuestro repositorio. Cuando queramos acabar de usar la rama, haremos un commit y la finalizaremos:

```
$ git flow feature stop feature_branch
```

Esto finaliza nuestra rama y la integra automáticamente a la rama *develop*. Si queremos seguir cambiando nuestro repositorio abriremos una nueva rama *feature*.

git flow release

Permite crear y trabajar con ramas de versiones. Cuando entendemos que después de todas las funcionalidades (features, cambios en nuestro repositorio) nuestro trabajo está listo para ser publicado, abriremos una rama release, que nacerá de nuestra rama develop.

```
$ git flow release start 0.1.0
```

```
Switched to a new branch 'release/0.1.0'
```

Usaremos un tag para identificar de que release se trata. Ahora podemos hacer los cambios que estimemos oportuno para integrar todas las features que el repositorio ha sufrido hasta el momento. Tras hacer commit a todo el proceso, podemos cerrar la rama release.

```
$git flow release finish '0.1.0'
```

Esto la integrará de forma automática con master (con esto finalizamos el proceso de 'subir a producción' nuestro código) y con la rama develop, para que las futuras features estén al día.

git flow hotfix

Permite crear y trabajar con ramas de parches. Esto lo usaremos para hacer cambios rapidos que no puedan esperar a la próxima integración de una release.

```
$ git flow hotfix start hotfix_branch
```

Tras hacer commit finalizamos la rama hotfix. Esta se fusionará con nuestra rama master y con nuestra rama develop para que esta también esté al día de los últimos cambios.

```
$ git flow hotfix finish hotfix_branch
```