Inicialización del repositorio

Podemos comenzar a trabajar de distintas formas: creando un repositorio en local o clonando un repositorio de GitHub

1. Inicio desde repositorio local
2. Abrir git bash
3. Situarse en la carpeta del proyecto
4. Inicializamos el repositorio

git flow init

$ git flow init

Initialized empty Git repository in C:/Desarrollo/golem-box/desarrollo/codigo/java/proyectos/tragaldaba/.git/

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

Branch name for production releases: [master] main

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

How to name your supporting branch prefixes?

Feature branches? [feature/]

Bugfix branches? [bugfix/]

Release branches? [release/]

Hotfix branches? [hotfix/]

Support branches? [support/]

Version tag prefix? [] REL\_

Hooks and filters directory? [C:/Desarrollo/golem-box/desarrollo/codigo/java/proyectos/tragaldaba/.git/hooks]

1. Comprobamos el estado del repositorio local

git status

1. Añadimos todos los ficheros al stage:

git add .

1. Pasamos del stage al repositorio local y ponemos un comentario

git commit –m “Comentario para el commit”

1. Cambiamos al main para sincronizarlo con develop

git checkout main

1. Mergeamos el develop sobre el main

git merge develop

1. Vuelvo al develop

git checkout develop

1. Enganchamos el repositorio local al remoto que hemos creado previamente

git remote add origin https://github.com/marinoqj/wittytool.git

1. Subimos las ramas locales

git branch –M main

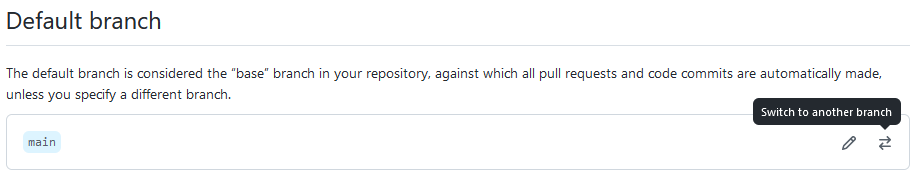
git push –u origin main

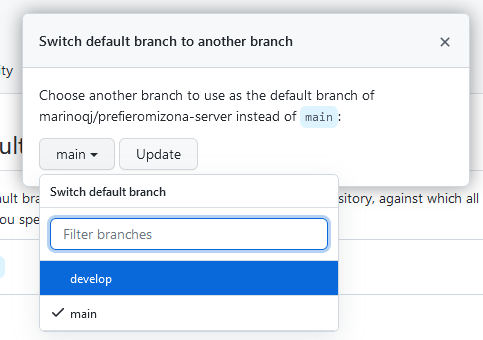
git branch –M develop

git push –u origin develop

1. En Github marcamos la rama develop para que sea la rama por defecto:

Settings -> Branches





1. Inicio desde repositorio remoto
2. Abrir GitBash en la carpeta en la que vamos a clonar el repositorio remoto, normalmente será el directorio de trabajo que tengamos definido en nuestro IDE (por ejemplo: eclipse-workspace)
3. Clonamos el directorio remoto:

git clone <https://github.com/marinoqj/gespracti.git>

1. Cambiamos al directorio recién creado:

cd gespracti/

1. Inicializamos el repositorio para trabajar con git flow:

**Nota Importante:** Al clonar un repositorio existente habrá que indicar que la rama de producción para las releases es ***main*** y para integración se utilizará la rama ***develop***.

**¡Cuidado con las opciones del nombrado de ramas!**! Hay que estar atentos

git flow init

$ git flow init

Which branch should be used for bringing forth production releases?

- develop

Branch name for production releases: [] main

Which branch should be used for integration of the "next release"?

- develop

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

How to name your supporting branch prefixes?

Feature branches? [feature/]

Bugfix branches? [bugfix/]

Release branches? [release/]

Hotfix branches? [hotfix/]

Support branches? [support/]

Version tag prefix? [] REL\_

Hooks and filters directory? [E:/Trabajos/versiones\_funcionando\_20210802/gespracti/.git/hooks]

Creación de nuevas funcionalidades (features)

1. Para trabajar en una nueva rama para una feature:

git flow feature start [nombre de la feature]



…Algún tiempo después…tras concluir la nueva funcionalidad…

1. Añadimos los cambios al stage y al repositorio local:

git add .

git commit –m “Comentario que recoja la finalidad del feature”

1. Mergeamos la rama de la funcionalidad con develop:

git flow feature finish [nombre de la feature]

1. Actualizamos nuestra rama develop local con la remota:

git pull origin develop

1. Resolvemos los conflictos desde Eclipse:

RMB sobre el fichero > Team > Merge Tool

(RMB = right mouse button)

1. Al finalizar el merge habrá que volver a añadir los cambios a nuestro repositorio local. La rama local develop estará entonces “limpia”:

git add .

git commit –m “Volvemos a especificar la finalidad del feature”

**Nota:** En el repositorio remoto quedará este comentario, no el que pusimos al finalizar el feature, que queda sólo en el repositorio local. Por eso es importante que se recoja la finalidad del feature.

1. Subimos los cambios (ya mergeados) al repositorio remoto:

git push

**Nota:** Al realizar el push el nombre de nuestra rama (que se muestra en git bash) cambiará de develop|MERGING a develop