GIT

Qué es Git?

SCM

- Software de Gestión de Cambios
- Controlar los cambios sobre archivos
- Mantener un correcto versionado de nuestro sistema

Historia

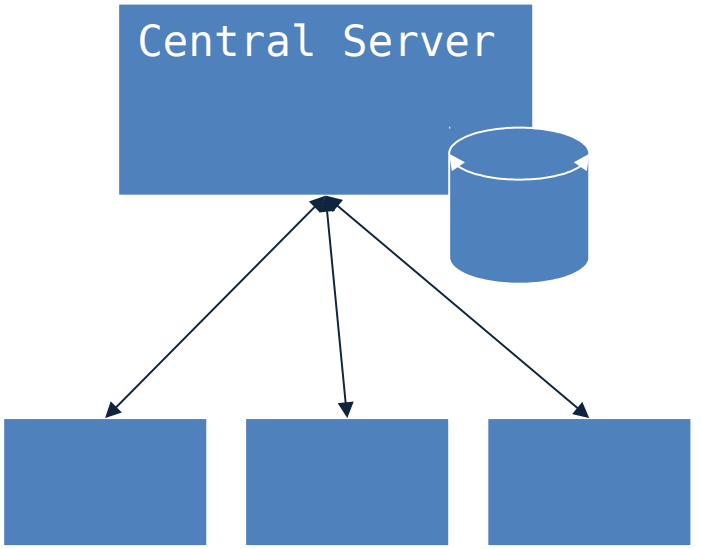
- Kernel de Linux utilizaba BitKeeper
- A partir del año 2005 la relación se rompió y se debía pagar por su uso
- Fue así como Linus Torvalds y equipo comenzaron a desarrollarlo

Empresas


Companies & Projects Using Git
Google
facebook
Microsoft
twitter
LinkedIn
NETFLIX
PostgreSQL
android
RAILS
Qt
GNOME
eclipse

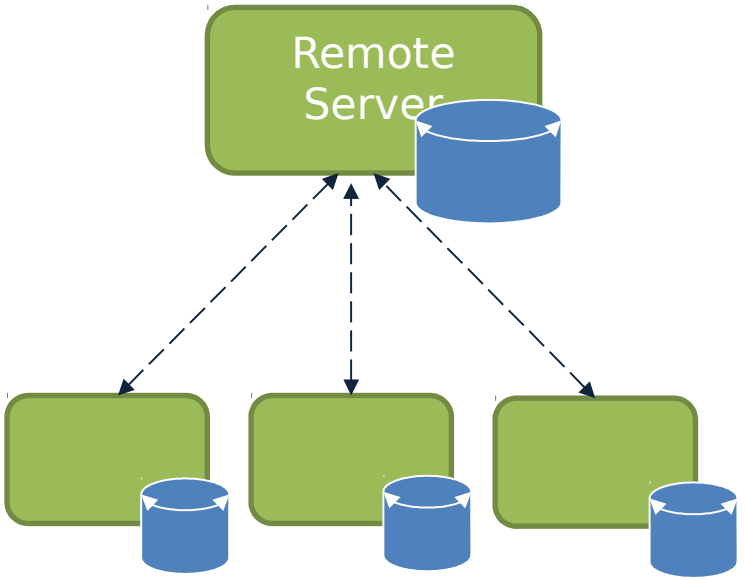
Centralizado Vs Distribuido

- Centralizado (por ej. Subversion)
  - Un solo servidor con todas las versiones
  - Todos saben en cada momento que está haciendo cada uno
  - Único punto de falla



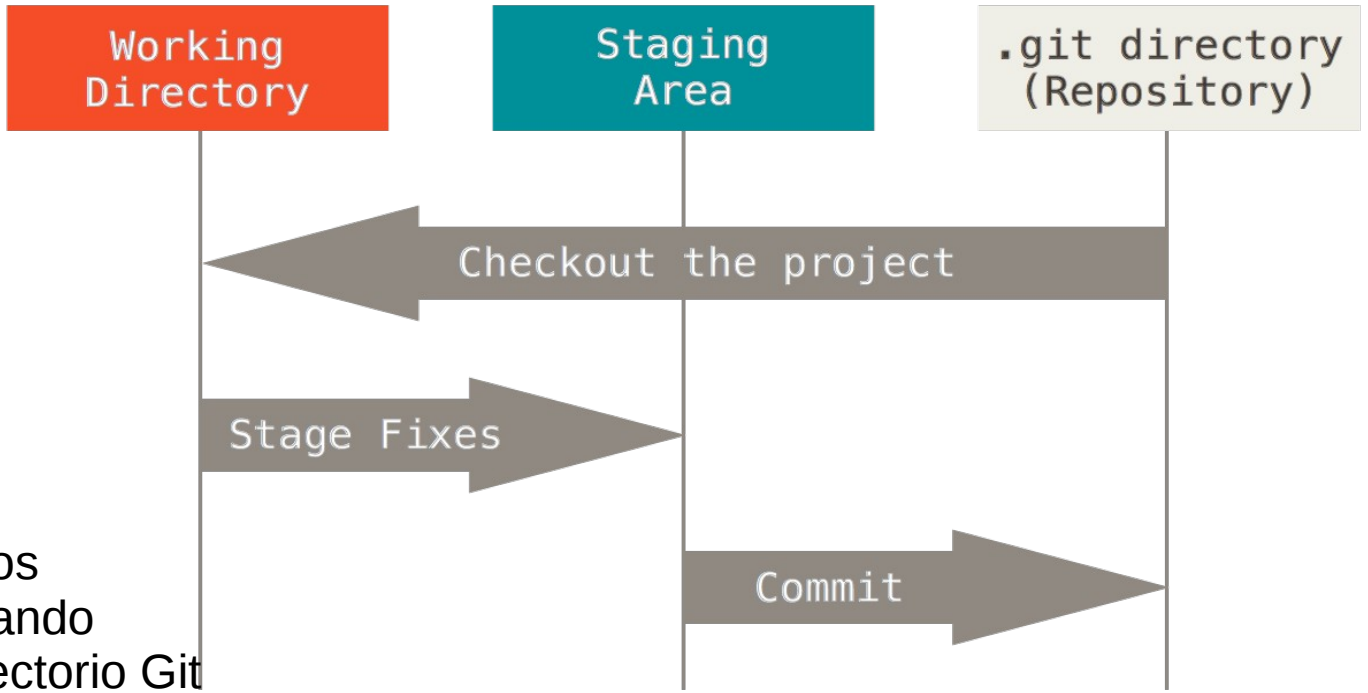
Centralizado Vs Distribuido

- “Distribuido” (por ej. Git)
  - Varias copias en cada usuario
  - Cada **clone** del repositorio es un backup de todos los datos
  - No hay un único punto de falla
  - Más dificil de saber en que están trabajando cada uno de los integrantes



Los 3 Estados

- Estado de archivos: **Commiteados, Modificados y Preparados (o Staged)**
- Secciones de Git
- Workflow
  - Modificamos archivos
  - Los preparamos
  - Commiteamos archivos preparados, almacenando un snapshot en el directorio Git



Ciclo de trabajo

- Configurar identidad

```
$ git config --global user.name "John Doe"


$ git config --global user.email johndoe@example.com
```

¡¡¡IMPORTANTE!!!

- Porque permite llevar control de nuestro trabajo.

Clonar un repositorio

```
$ git clone git@gitlab.fing.edu.uy:proyecto.git
```

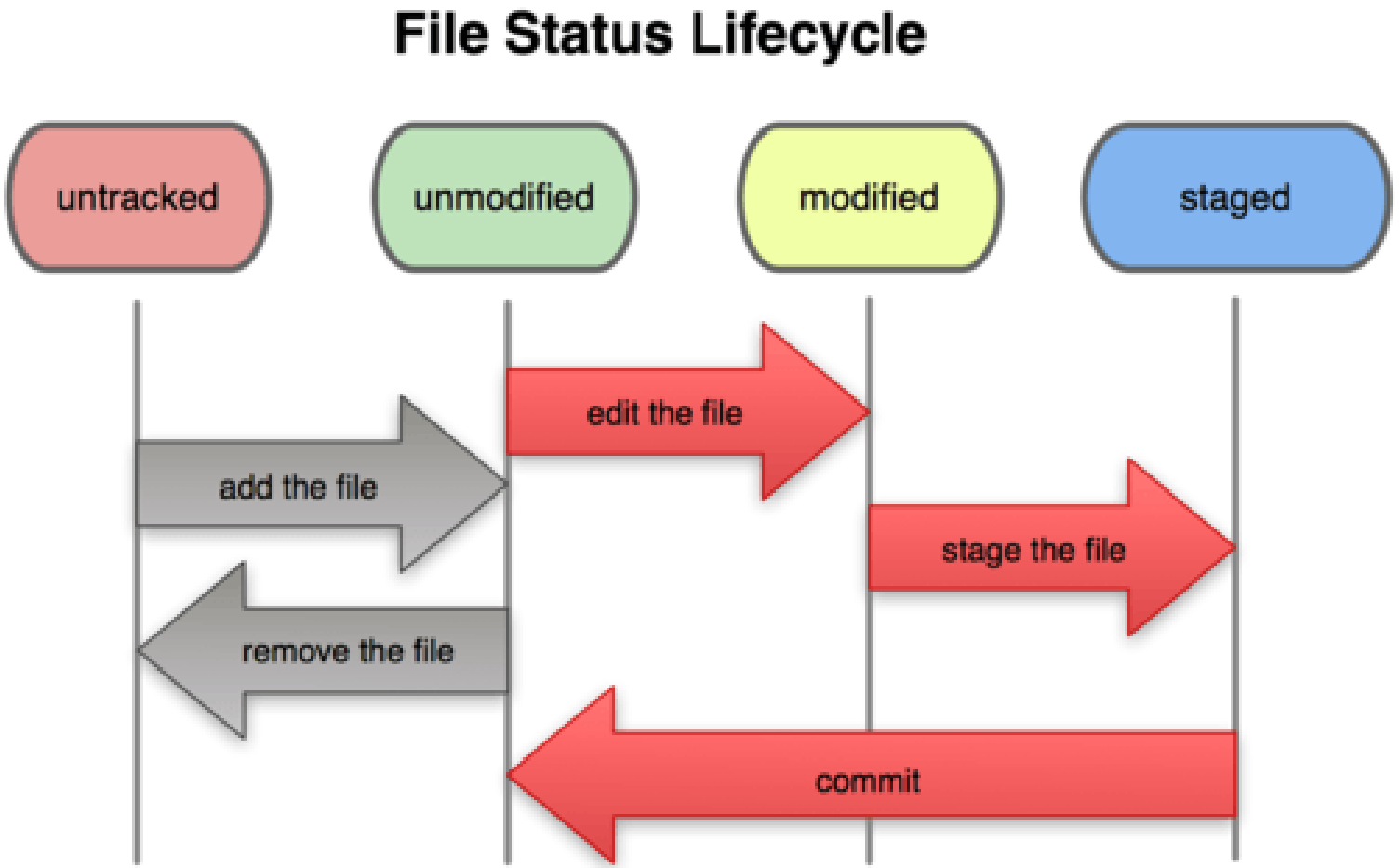
- Recibimos copia completa de nuestros datos del servidor
- El protocolo a utilizar es ssh y no https

  Utilizar ssh tiene mayores dificultades a la hora de configurar inicialmente pero simplifica el trabajo posterior.

Agregando cambios

- Los archivos pueden estar Trackeados y No Trackeados
- Trackeados: Archivos incluidos en el último snapshot. Pueden estar sin modificar, modificados o preparados
- No Trackeados: El resto de los archivos. No son utilizados por git
- Realizar las transiciones de estado para los archivos implica la ejecución de comandos git

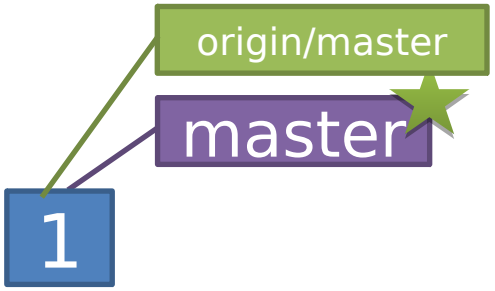
Agregando cambios



Branchs (ramas)

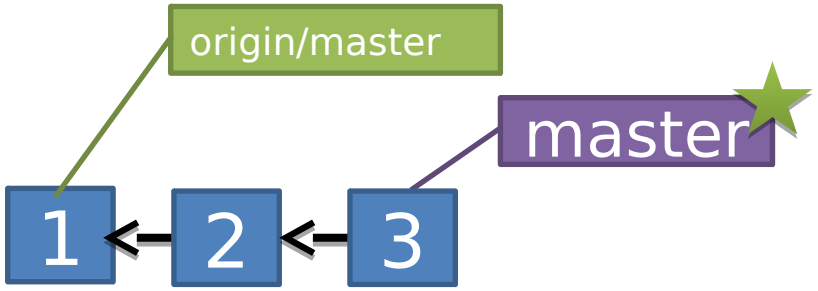
- Línea separada de código con su propia historia
- Es un puntero a un commit
- El branch principal y por defecto es master
- Las funcionalidades se deben desarrollar en un branch para luego incorporarlas a master

Branchs




```
$ git commit –m ‘mi primer commit’
```

Branchs

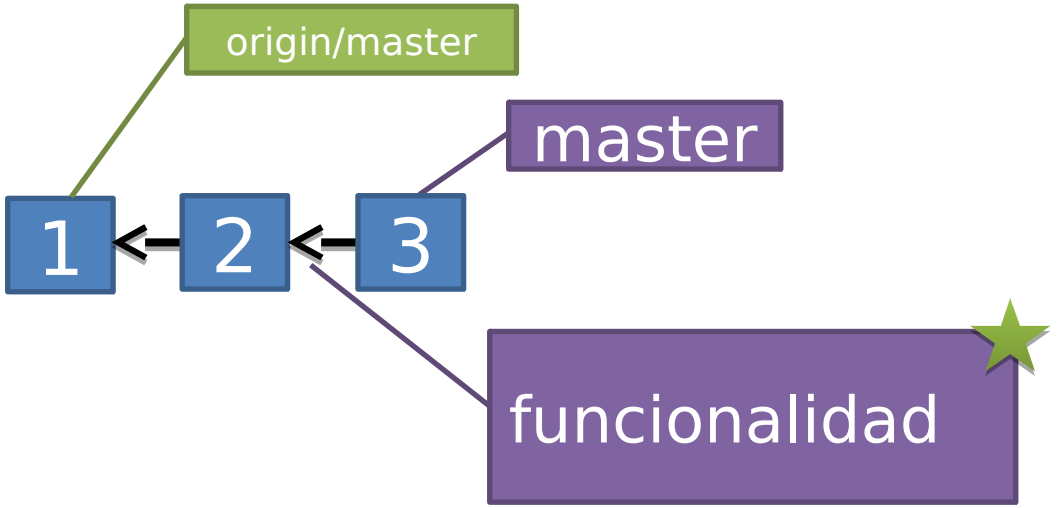


Realizamos dos commit y nuestro árbol tiene la estructura anterior

```
$ git commit –m (x2) # Realizamos dos commit
```

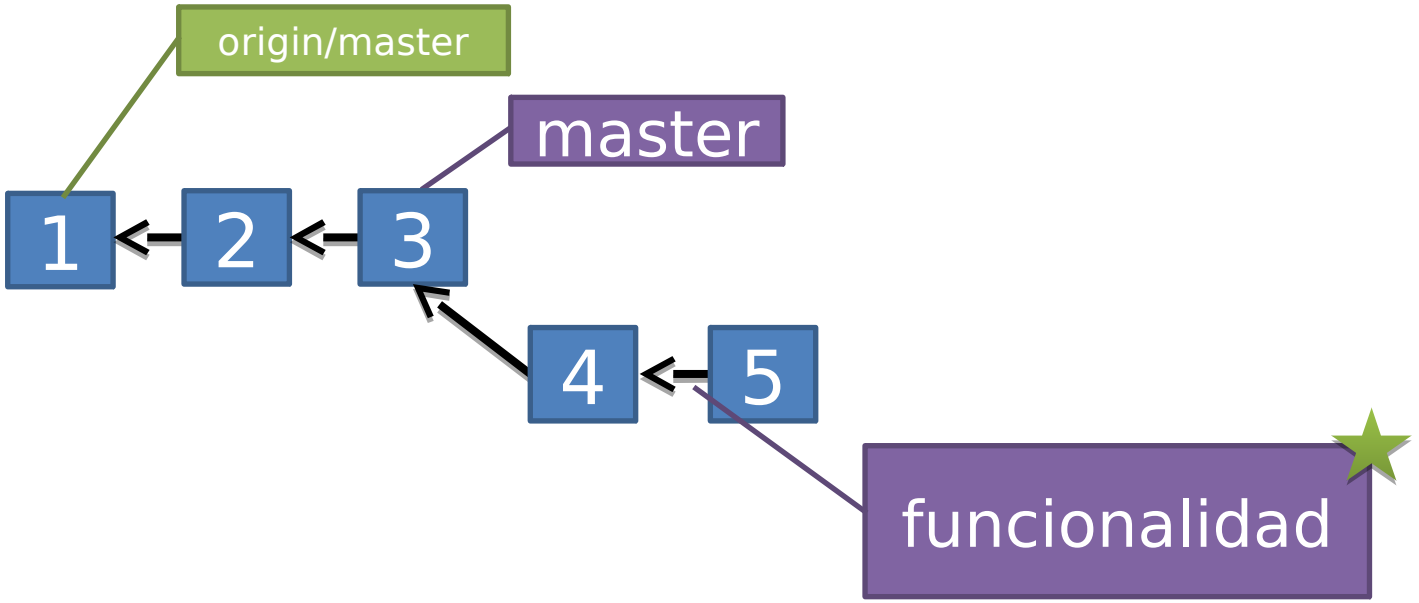
Branchs

- Creamos un branch “funcionalidad”



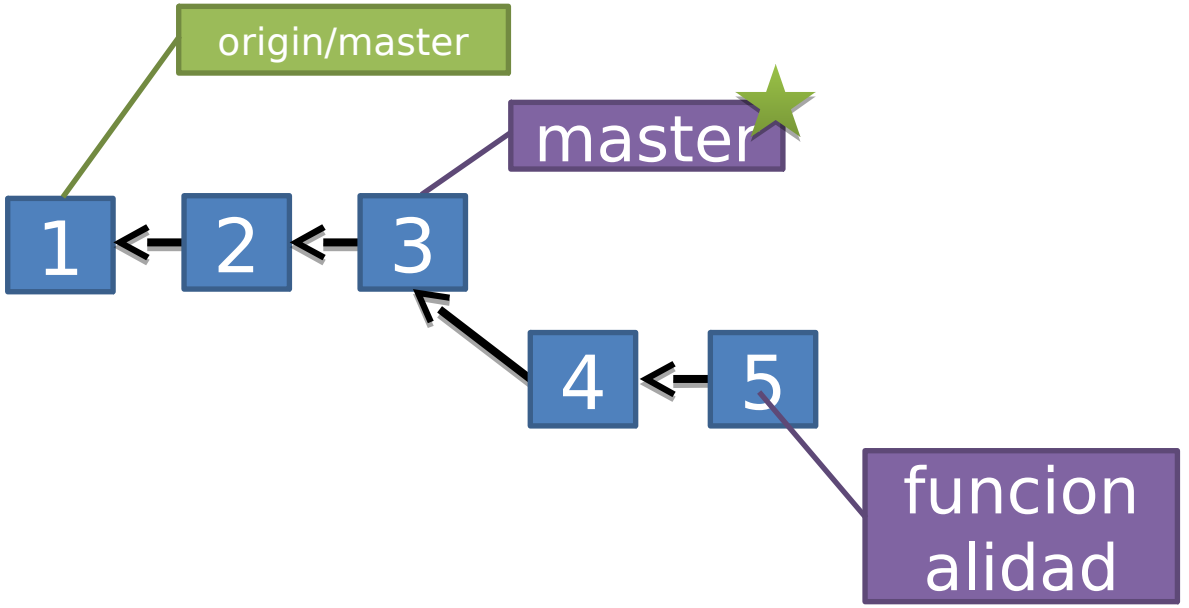

```
$ git checkout –b funcionalidad
```

Branchs




```
$ git commit (x2)
```

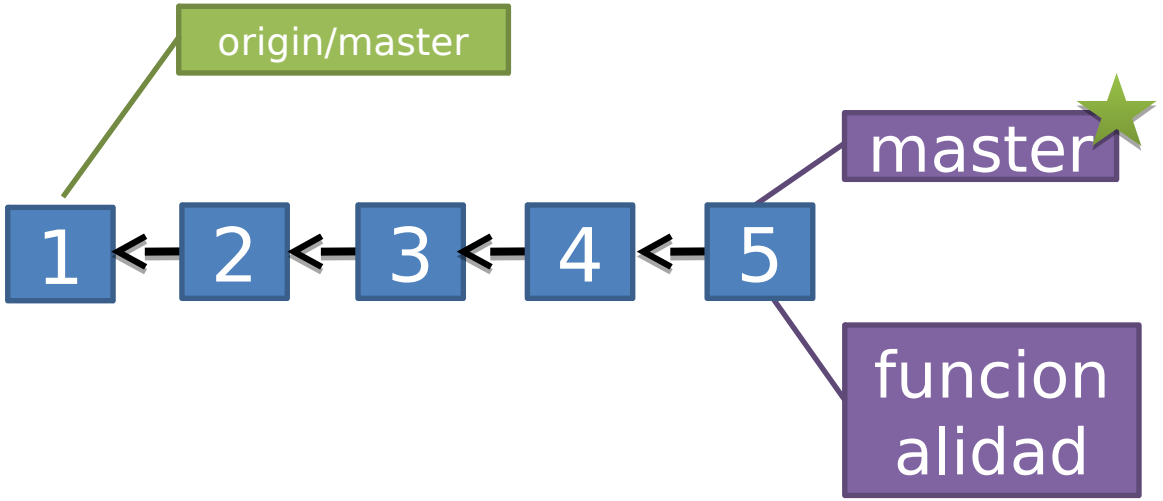
Merge




```
$ git checkout master
```

Merge

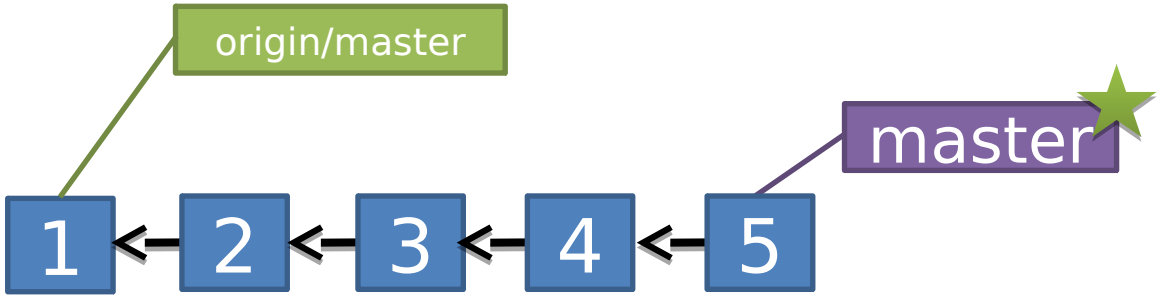
- Este merge no resulta en conflicto.




```
$ git merge funcionalidad
```

Merge

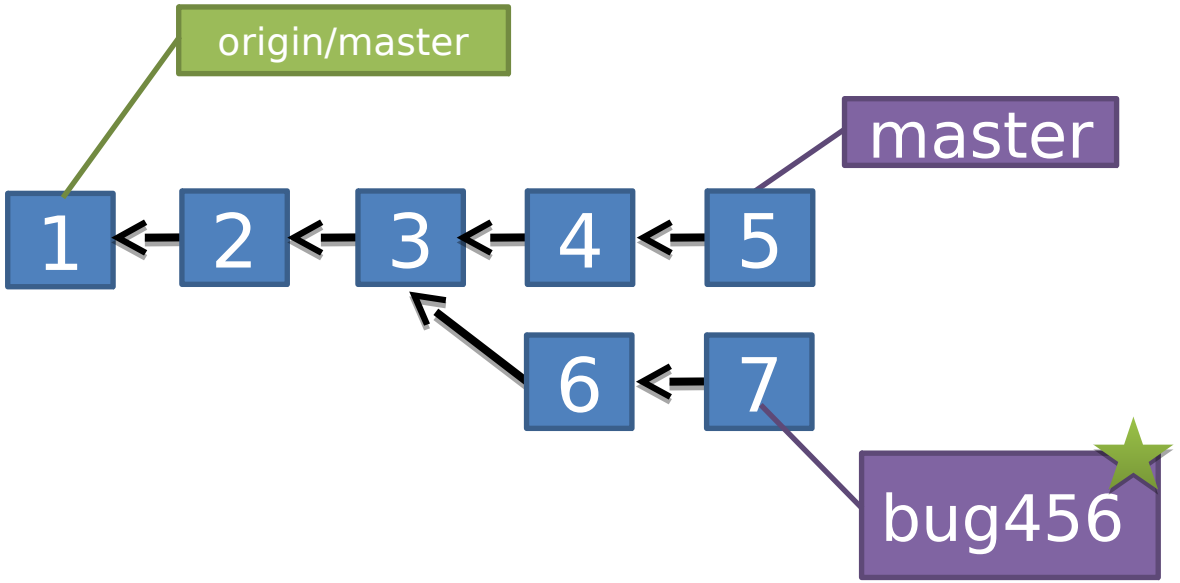
- Eliminamos el branch luego de incorporarlo a master.




```
$ git branch -d funcionalidad
```

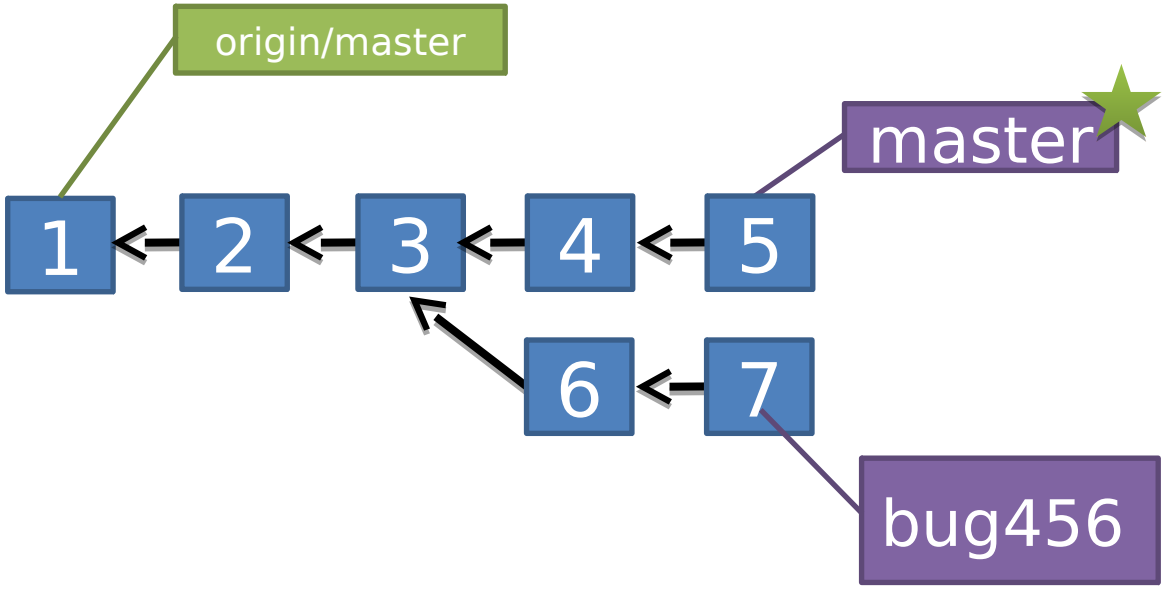
Merge

- Suponer escenario donde anteriormente otro integrante crea un branch a partir del commit 3.



- Los commit 6 y 7 realizan modificaciones en los mismos archivos que los commit 4 y 5.

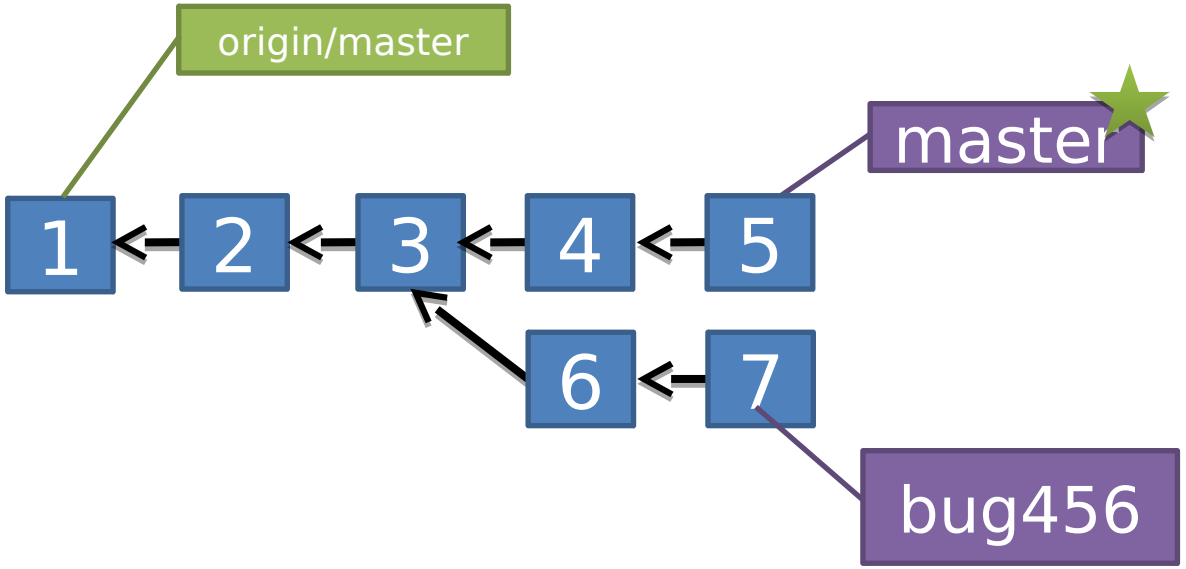
Merge




```
$ git checkout master
```

Merge

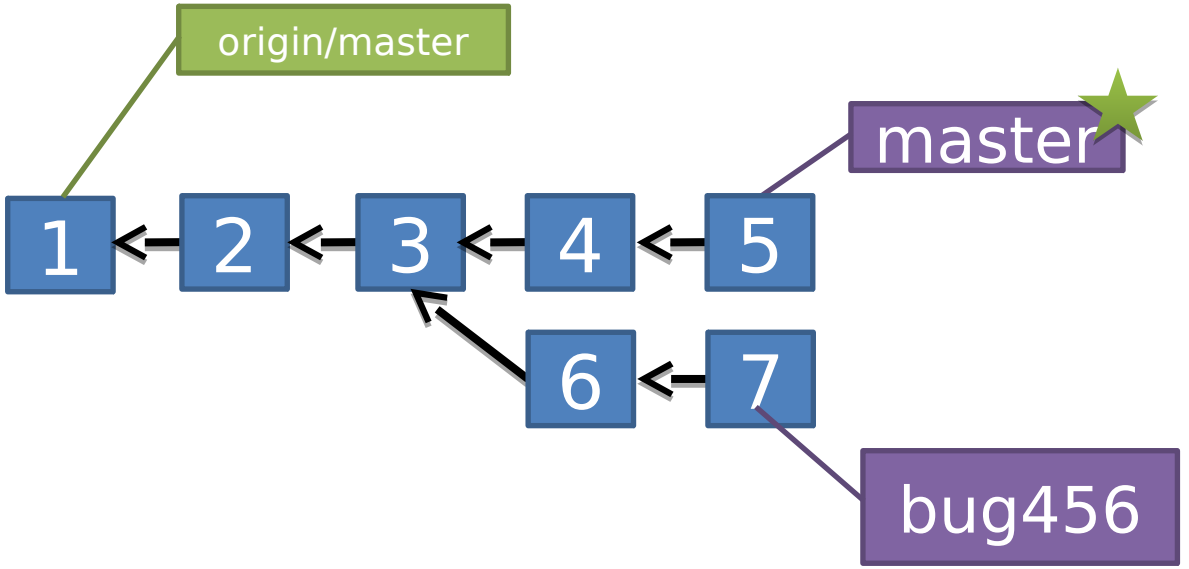
```
$ git merge bug456
```



- Este merge resultará con conflicto.

Merge

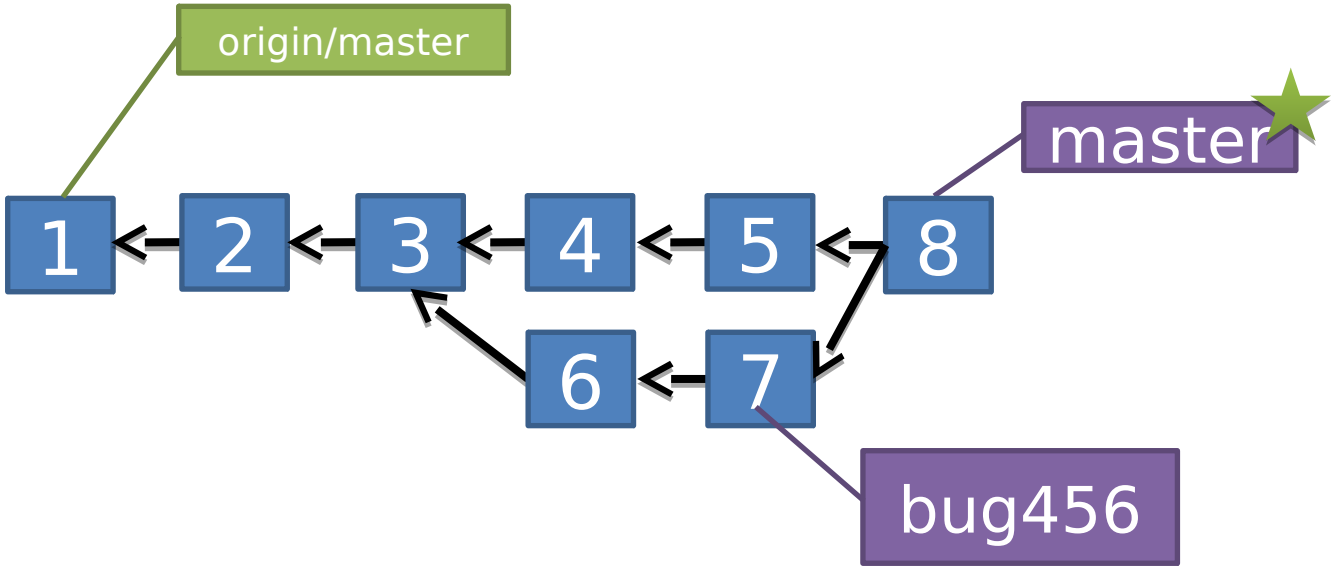
- Para resolverlo debemos ejecutar el siguiente comando...




```
$ git mergetool
```

Merge

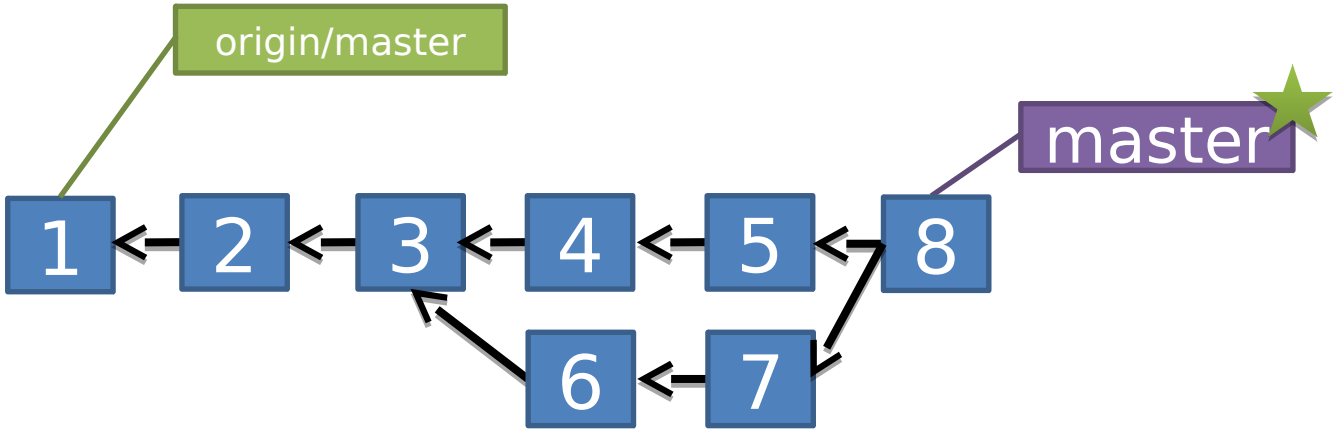
- ...y resolver los conflictos con meld. Luego debemos realizar el commit correspondiente a las modificaciones.




```
$ git commit -m “Resolucion de conflicto”
```

Merge

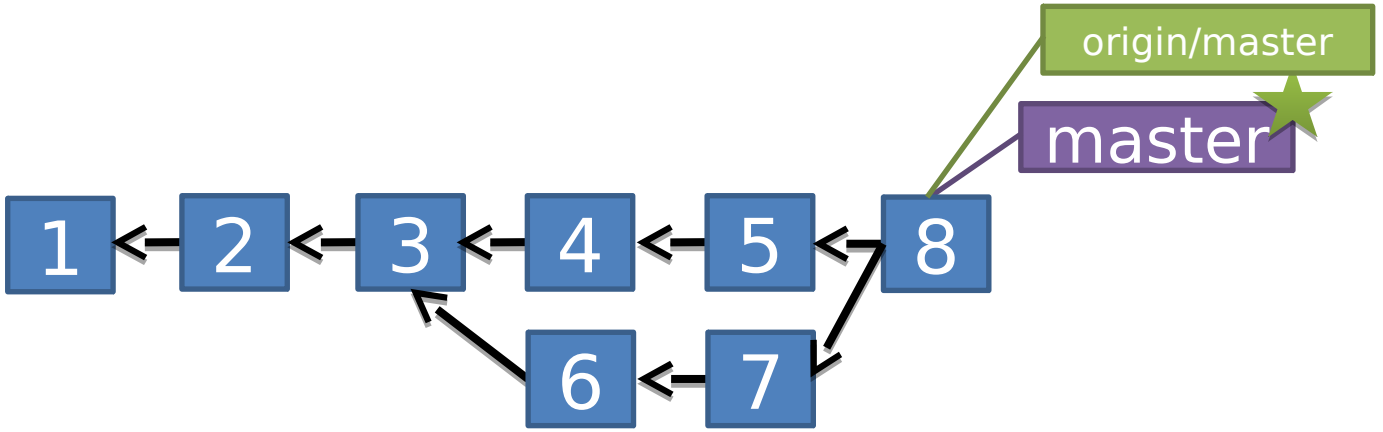
- Eliminamos el branch luego de la incorporación de los cambios a master.




```
$ git branch -d bug456
```

Actualizando el repositorio

- Estamos en condiciones de actualizar el repositorio remoto (origin/master) para que el resto del equipo tenga nuestras modificaciones




```
$ git pull origin master
```

Traemos cambios remotos

```
$ git push origin master
```

Enviamos cambios al servidor remoto

En resumen

$ **git clone url** # inicializa repositorio con el contenido del repositorio en url

$ **git init** # inicializa un repositorio vacío

$ **git add** archivos # agrega uno o más archivos para ser commiteados

$ **git commit -m “mensaje”** # crea un nuevo comit con todos los archivos agregados

$ **git status** # muestra el estado del repositorio

$ **git push origin master** # envia el estado de la rama master a la rama remota origin

$ **git pull origin master** # trae el estado de la rama remota origin a la rama local master

En resumen

Utilizar **gitk** para ver el estado de las ramas, controlar commits y visualizar el repositorio de manera global y ordenada.

Es importante tener un buen editor para resolver conflictos, recomiendo meld, y se configura así: **git config mergetool meld.** Los conflictos se dan cuando realizamos un **pull**

Por si no fui claro, **git status** es el mejor amigo del usuario git

Git en FIng

- GitLab así como GitHub son manejadores web de repositorios git.
- GitLab es la instalación central de Git en la Facultad de Ingeniería (https://gitlab.fing.edu.uy/)
- Se debe configurar una clave pública para acceder
- GitHub ofrece una herramienta gratuita para aprender a usar git (https://try.github.io/levels/1/challenges/1)