



# Desarrollo colaborativo con Git & Github

CLASE 2

# 01

REPASO



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git status`

`git clone <url>`

`git add .`

`git status`

`git commit -m "mensaje"`

`git log`

`git push`



# OBJETIVOS DE LA CLASE 2

- ❑ ¿Que es una rama?
- ❑ Buenas prácticas
- ❑ gitignore
- ❑ repositorio remoto
- ❑ Fork



## COMANDOS DE LA CLASE DE HOY

Los alumnos pueden utilizar esta lista para guiarse en qué comando se explicó en cada clase, para saber si corresponde ver este video grabado o no. Si no llega a estar en esta clase, seguro en la siguiente.

- ❑ `git remote`
- ❑ `git branch <name>`
- ❑ `git branch -l`
- ❑ `git branch -l -a`
- ❑ `git branch -d <branch_name>`
- ❑ `git branch -D <branch_name>`
- ❑ `git pull`
- ❑ `git checkout <name_branch>`
- ❑ `git merge <name_branch>`

# 02

REMOTE





**¿Cómo se crea un  
repositorio?**

## EXISTEN DOS CAMINOS



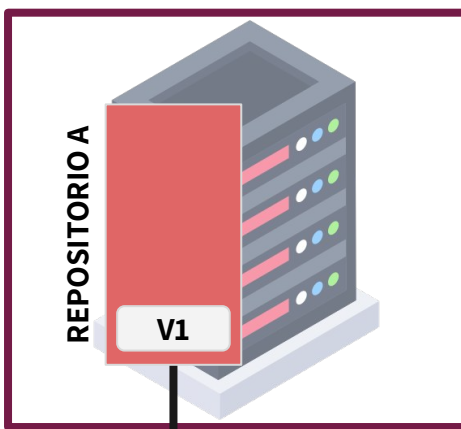
Este lo vimos la clase pasada, ahora toca el otro.



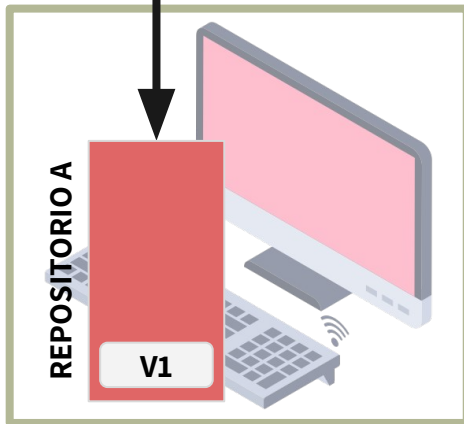


# **Primer camino - Clonando un repositorio**

Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### CUARTO PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### QUINTO PASO - En Consola

Ejecutamos el siguiente comando con la url copiada de Github.

```
git clone <url>
```

### SEXTO PASO - En Consola

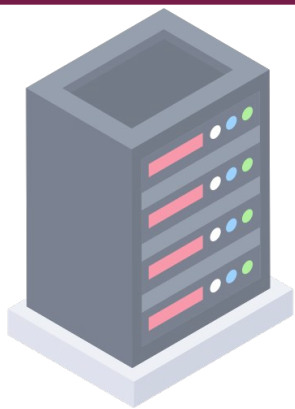
Abrimos el repositorio en Visual studio code para comenzar a trabajar.

```
code <nombre_repositorio>
```



**Segundo camino -  
Creando un repositorio  
local y conectando a  
otro remoto**

Github (En la nube)



## PASOS PARA UN REPOSITORIO REMOTO

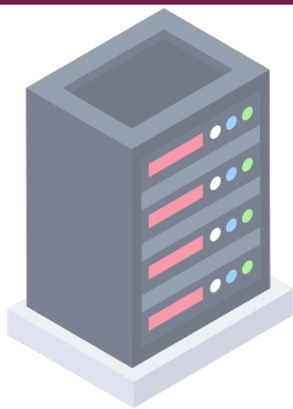
### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

Tu compu (Local)



Github (En la nube)



## PASOS PARA UN REPOSITORIO REMOTO

### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

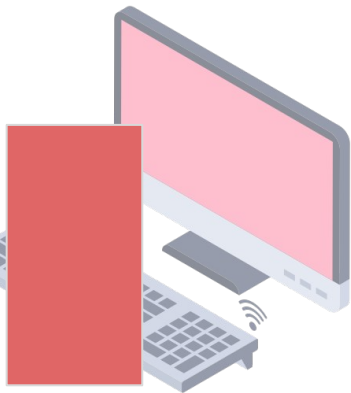
### SEGUNDO PASO - En consola

Ejecutamos el siguiente comando para crear una nueva carpeta que sea un repositorio.

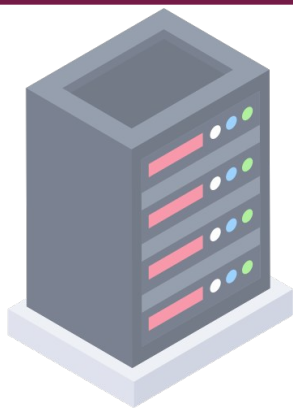
```
git init <nombre_repositorio>
```

Tu compu (Local)

REPOSITORIO A



Github (En la nube)



## PASOS PARA UN REPOSITORIO REMOTO

### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

### SEGUNDO PASO - En consola

Ejecutamos el siguiente comando para crear una nueva carpeta que sea un repositorio.

```
git init <nombre_repositorio>
```

### TERCER PASO - En consola

Abrir en Visual studio code para poder trabajar con el siguiente comando.

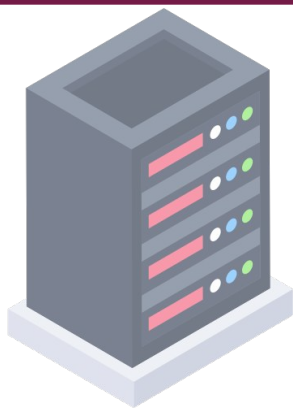
```
code <nombre_repositorio>
```

Tu compu (Local)

REPOSITORIO A



Github (En la nube)



## PASOS PARA UN REPOSITORIO REMOTO

### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

### SEGUNDO PASO - En consola

Ejecutamos el siguiente comando para crear una nueva carpeta que sea un repositorio.

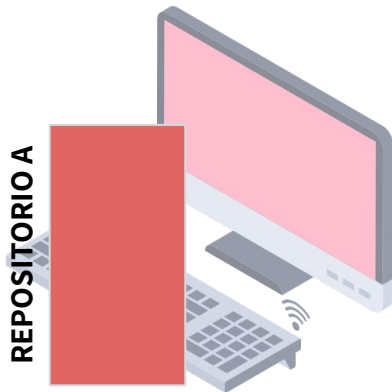
```
git init <nombre_repositorio>
```

### CUARTO PASO - En visual

Hay que crear una primera rama con el nombre de main utilizando el siguiente comando.

```
git branch -M main
```

Tu compu (Local)

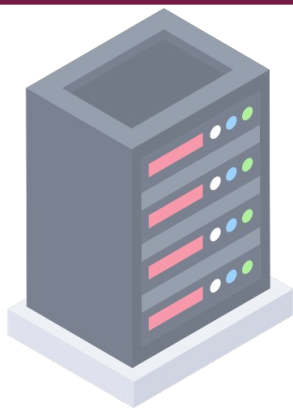


### TERCER PASO - En consola

Abrir en Visual studio code para poder trabajar con el siguiente comando.

```
code <nombre_repositorio>
```

Github (En la nube)



## PASOS PARA UN REPOSITORIO REMOTO

### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

### SEGUNDO PASO - En consola

Ejecutamos el siguiente comando para crear una nueva carpeta que sea un repositorio.

```
git init <nombre_repositorio>
```

### TERCER PASO - En consola

Abrir en Visual studio code para poder trabajar con el siguiente comando.

```
code <nombre_repositorio>
```

### CUARTO PASO - En visual

Hay que crear una primera rama con el nombre de main utilizando el siguiente comando.

```
git branch -M main
```

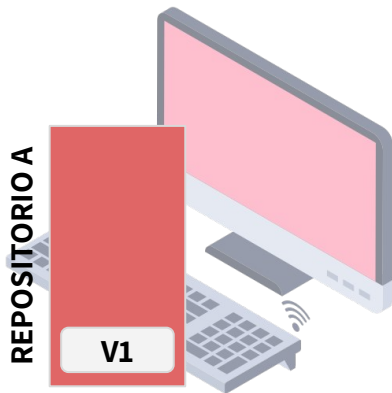
### QUINTO PASO - En visual

Crear un archivo de README.md y crear el primer commit con los siguientes comandos.

```
git add .
```

```
git commit -m "mensaje"
```

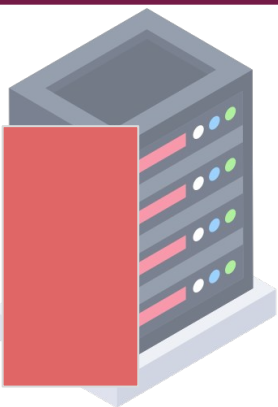
Tu compu (Local)





Github (En la nube)

REPOSITORIO B



## PASOS PARA UN REPOSITORIO REMOTO

### PRIMER PASO - En consola

Abrir la consola ubicada donde se quiere el repositorio.

### SEGUNDO PASO - En consola

Ejecutamos el siguiente comando para crear una nueva carpeta que sea un repositorio.

```
git init <nombre_repositorio>
```

### TERCER PASO - En consola

Abrir en Visual studio code para poder trabajar con el siguiente comando.

```
code <nombre_repositorio>
```

### CUARTO PASO - En visual

Hay que crear una primera rama con el nombre de main utilizando el siguiente comando.

```
git branch -M main
```

### QUINTO PASO - En visual

Crear un archivo de README.md y crear el primer commit con los siguientes comandos.

```
git add .
```

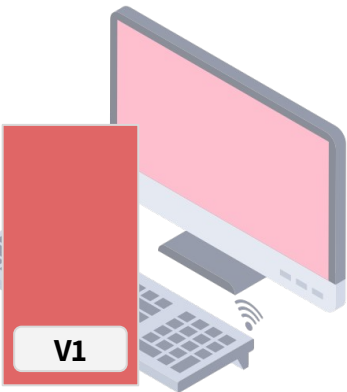
```
git commit -m "mensaje"
```

### SEXTO PASO - En Github

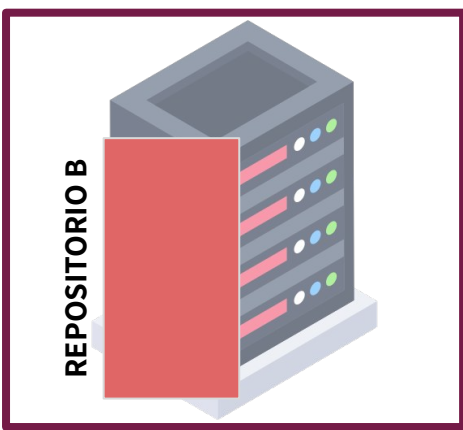
Crear un nuevo repositorio en Github sin cambiar los valores por default. El nombre puede ser el mismo u otro.

Tu compu (Local)

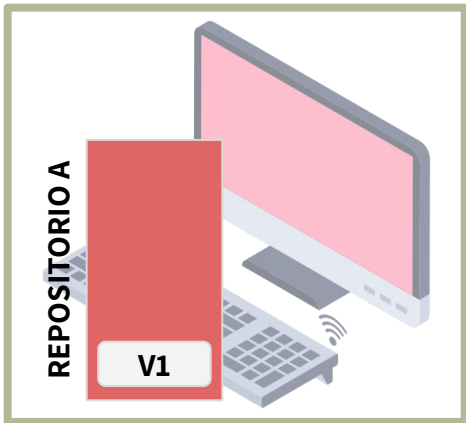
REPOSITORIO A



Github (En la nube)



Tu compu (Local)



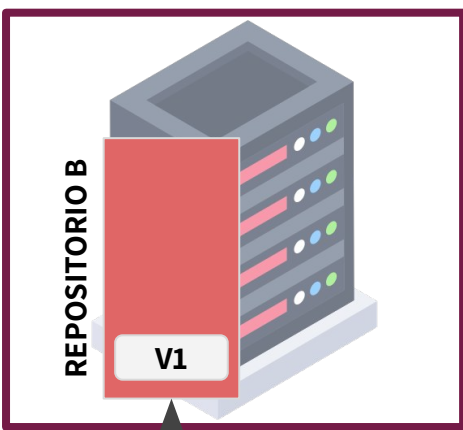
## PASOS PARA UN REPOSITORIO REMOTO

### SÉPTIMO PASO - En consola

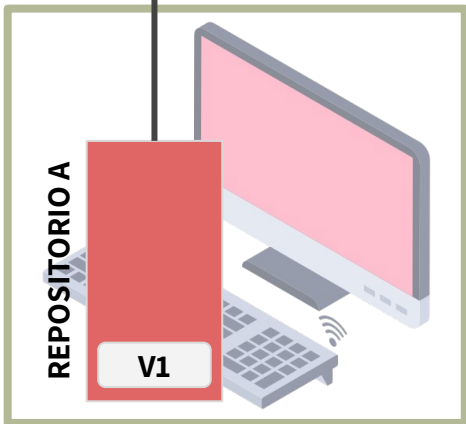
Agregamos un nuevo repositorio remoto de nombre “origin”.

```
git remote add origin <url>
```

Github (En la nube)



Tu compu (Local)



## PASOS PARA UN REPOSITORIO REMOTO

### SÉPTIMO PASO - En consola


Agregamos un nuevo repositorio remoto de nombre "origin".

```
git remote add origin <url>
```

### OCTAVO PASO - En consola

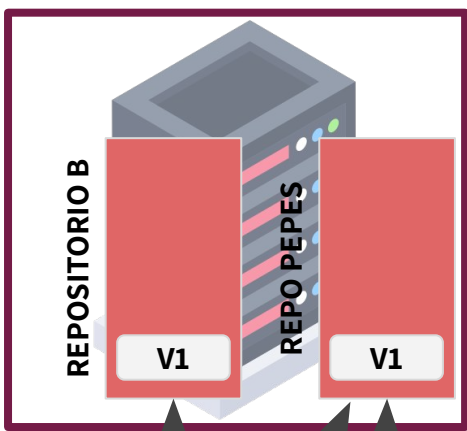
Conecto con el repositorio remoto en Github utilizando la siguiente configuración para push por única vez.

```
git push -set-upstream origin main
```

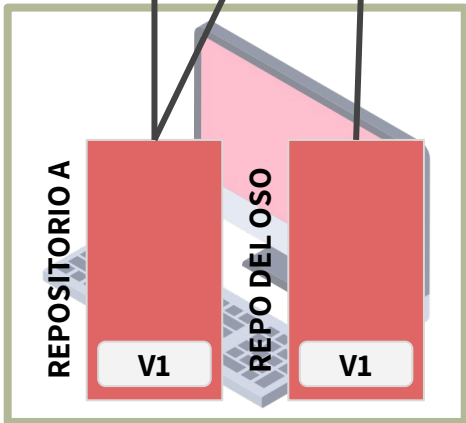


**¿Se pueden tener varios  
repositorios remotos  
simultáneamente?**

Github (En la nube)



Tu compu (Local)



## PASOS PARA UN REPOSITORIO REMOTO

### SÉPTIMO PASO - En consola

Agregamos un nuevo repositorio remoto de nombre “origin”.

```
git remote add <nombre_remoto> <url>
```

### OCTAVO PASO - En consola

Conecto con el repositorio remoto en Github utilizando la siguiente configuración para push por única vez.

```
git push -set-upstream <nombre_remoto> main
```

# 02

FORK





**¿Que es un fork?**

**Un FORK es una función que permite realizar una copia completa de un repositorio en mi cuenta. No es lo mismo que clonarlo, esta copia no tiene ninguna relación con la original.**



A large white chevron graphic pointing to the right, located on the left side of the slide.

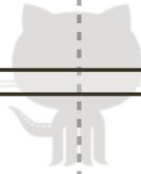
**¿Para qué sirve un  
fork?**

## **ACER UNA COPIA NO CONFIDENCIAL**

s permite hacer una copia de la línea de tiempo en  
estro proyecto, pero queda constancia. Suele  
lizarse para crear un software nuevo en base a un  
ftware que ya existe.



Them



You



origin

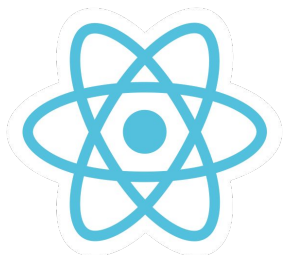


not your  
problem



"fork and clone"

## Ejemplo de repositorios



**ReactJS**

[CLICK ME](#)



**VueJS**

[CLICK ME](#)



**Angular**

[CLICK ME](#)

# 03

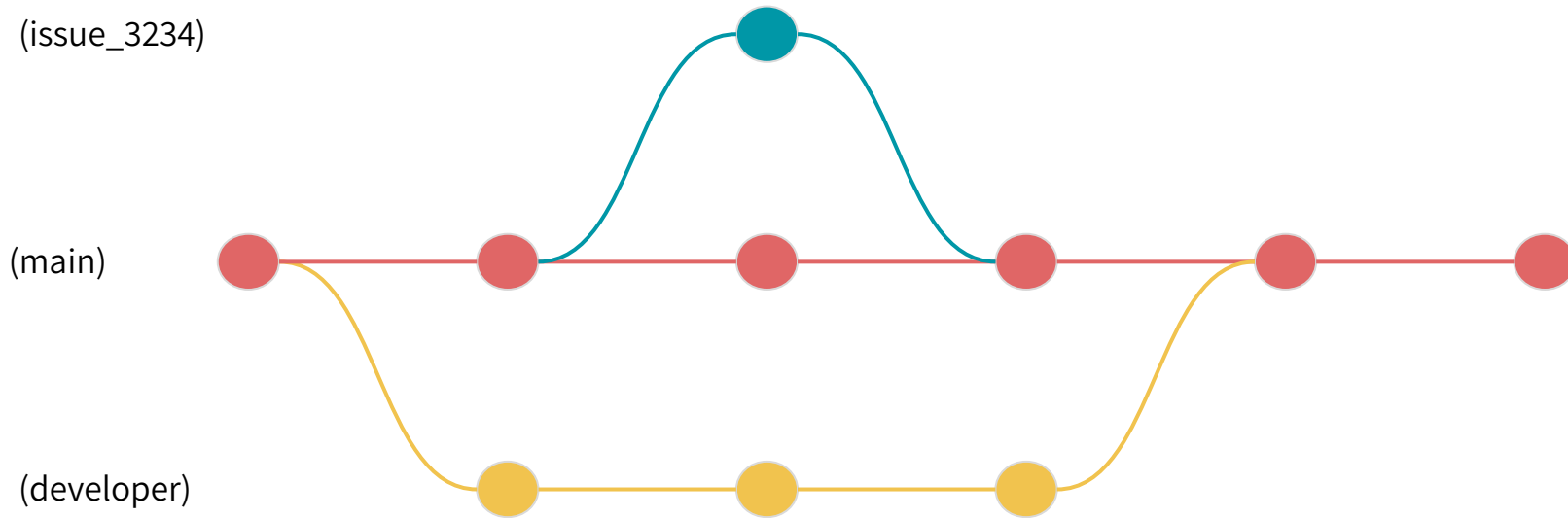
RAMAS





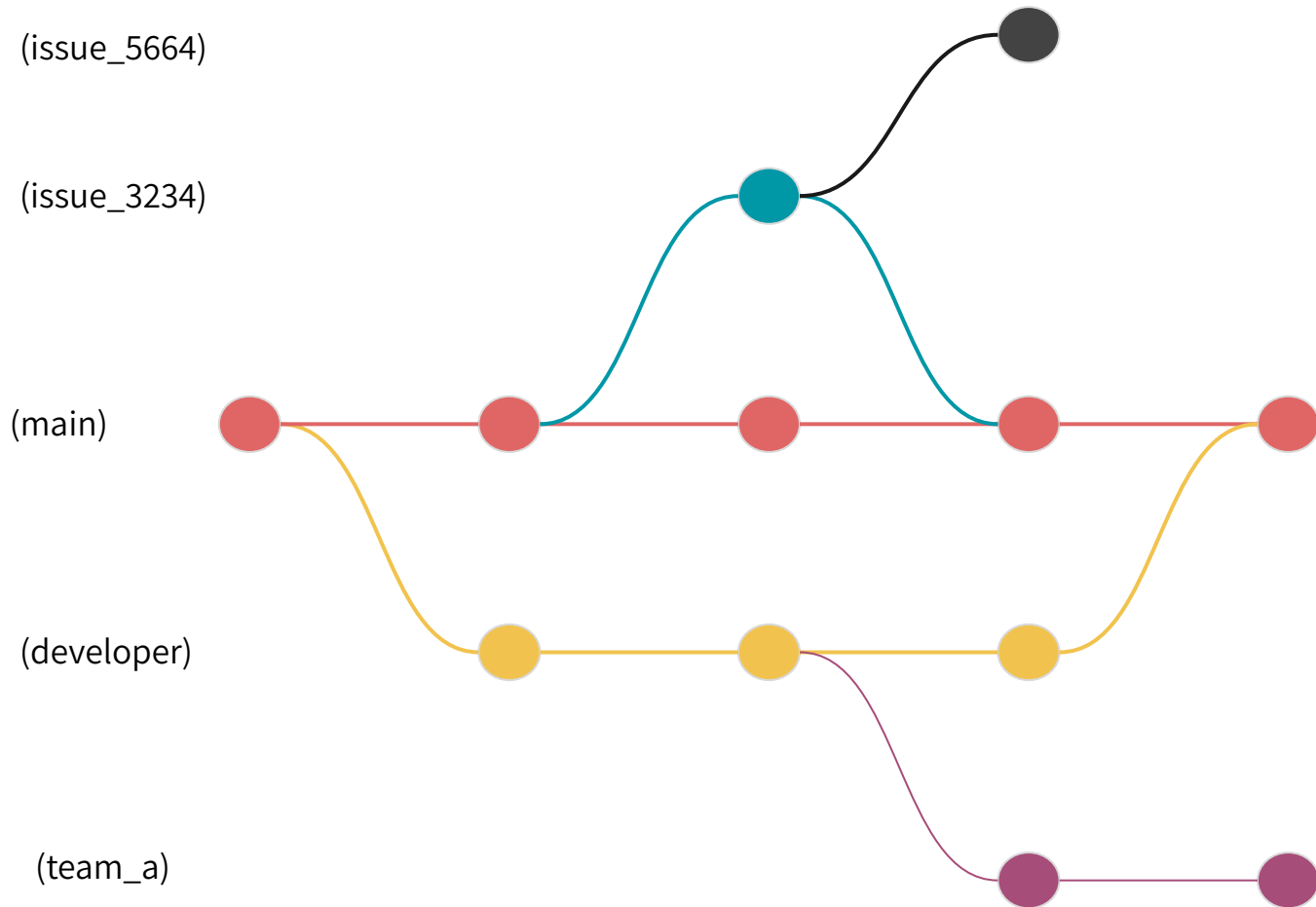
**¿Qué son las ramas?**

**Una RAMA es una línea de tiempo  
paralela a la original que permite traer  
los cambios a otra rama para obtener un  
único proyecto.**



NOTA: Las versiones viejas de Git, la rama “main” se llamaba “master”. Si lees en un foro o ves en un video de Youtube que hablan de “master”, probablemente el material está desactualizado.

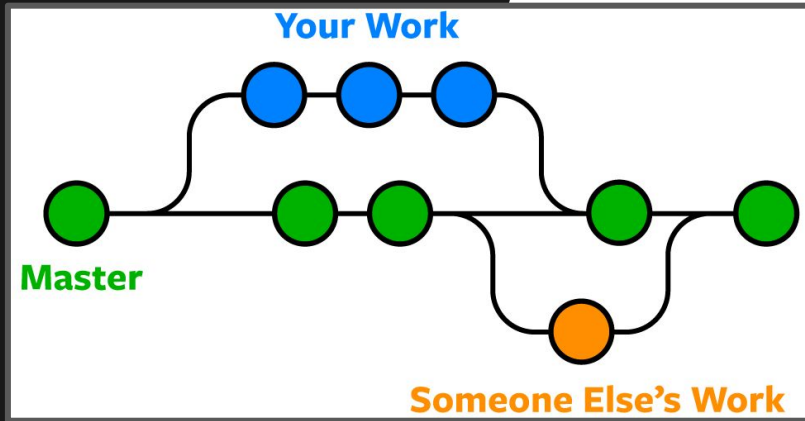







**¿Qué uso tienen las  
ramas?**

# USO DE LA RAMAS



- ❑ Permite organizar el trabajo del proyecto. Una rama para cada desarrollador, equipos, test, developer y versiones estables.
- ❑ Realizar pruebas sin estorbar al resto.



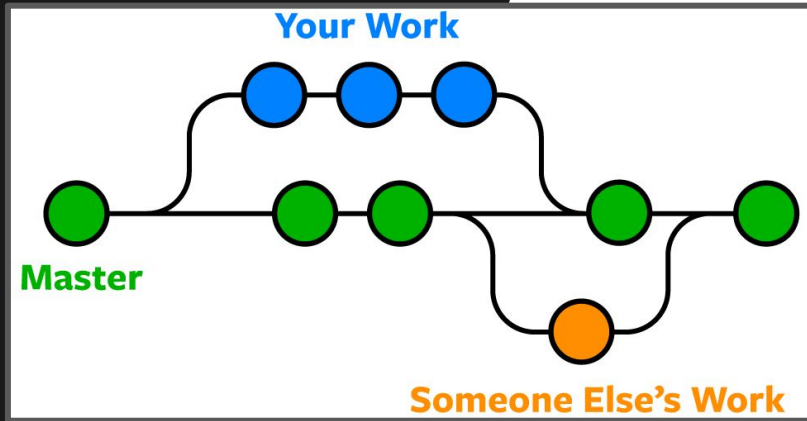
**¿Cuáles son las  
reglas de  
convivencia en las  
ramas?**

# REGLAS DE CONVIVENCIA

- ❑ No tocas una rama ajena.
- ❑ No mergear o fusionar ramas que no perteneces sin autorización.
- ❑ Utilizar nombres claros que permitan reconocer el contenido de la rama.



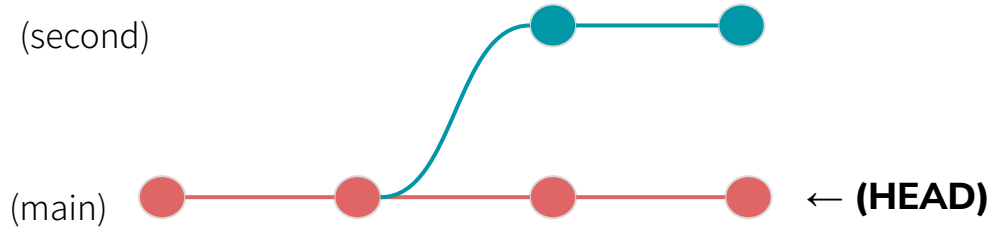
# REGLAS DE CONV



- ❑ Permite organizar el trabajo del proyecto. Una rama para cada desarrollador, equipos, test, developer y versiones estables.
- ❑ Realizar pruebas sin estorbar al resto.



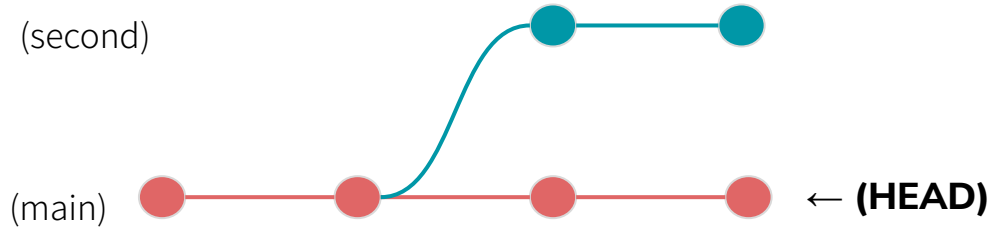
# **Operaciones básicas con ramas**



¿Cómo creó una nueva rama?

```
git branch <nombre>
```



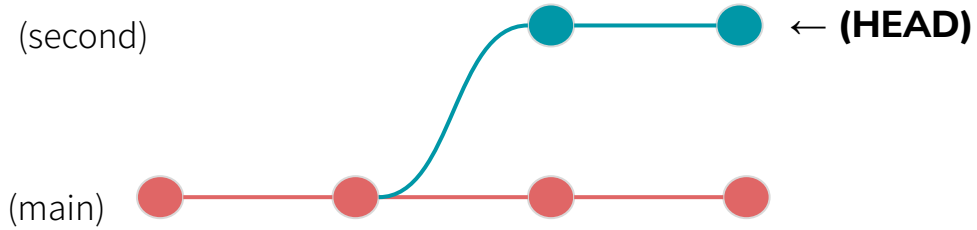


¿Cómo creó una nueva rama?

```
git branch <nombre>
```

¿Cómo veo las ramas?

```
git branch
```



¿Cómo creó una nueva rama?

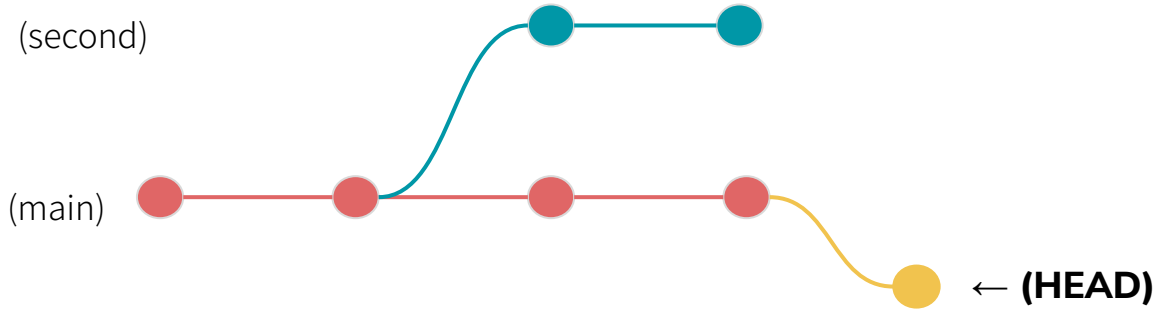
```
git branch <nombre>
```

¿Cómo veo las ramas?

```
git branch
```

¿Cómo me muevo de rama?

```
git checkout <nombre>
```



¿Cómo creó una nueva rama?

```
git branch <nombre>
```

¿Cómo crear y me muevo a esa rama?

```
git checkout -b <nombre>
```

¿Cómo veo las ramas?

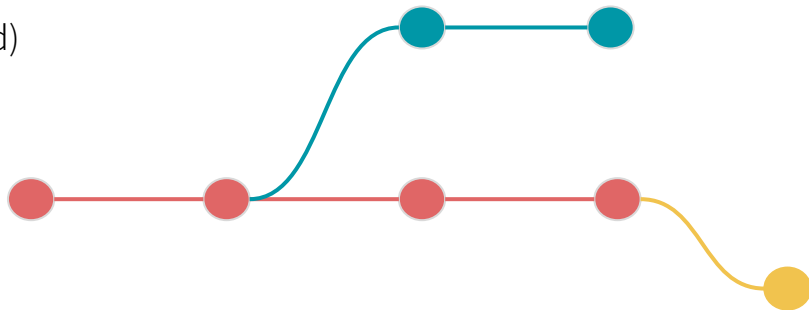
```
git branch
```

¿Cómo me muevo de rama?

```
git checkout <nombre>
```

(second)

(main)



¿Cómo veo las ramas?

```
git log --oneline --graph --all
```

¿Cómo creó una nueva rama?

```
git branch <nombre>
```

¿Cómo crear y me muevo a esa rama?

```
git checkout -b <nombre>
```

¿Cómo veo las ramas?

```
git branch
```

¿Cómo eliminar una rama?


```
git branch -d <nombre>
```

¿Cómo me muevo de rama?

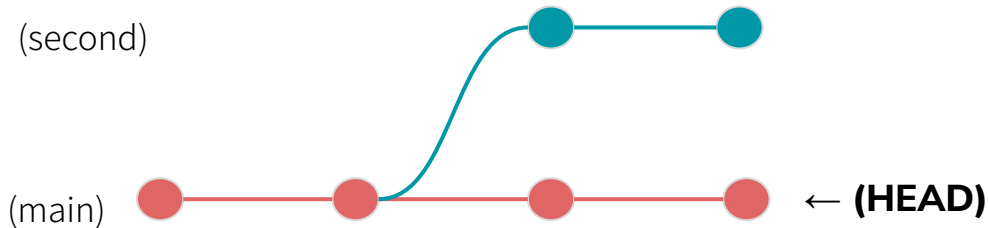
```
git checkout <nombre>
```

¿Cómo fuerzo eliminar una rama?

```
git branch -D <nombre>
```



**¿Cómo se combinan  
dos ramas?**



## PASOS PARA MERGEAR

### PRIMER PASO - En visual

Nos posicionamos en la rama donde queremos que el resultado de la unión se guarde.

```
git checkout main
```

### SEGUNDO PASO - En visual

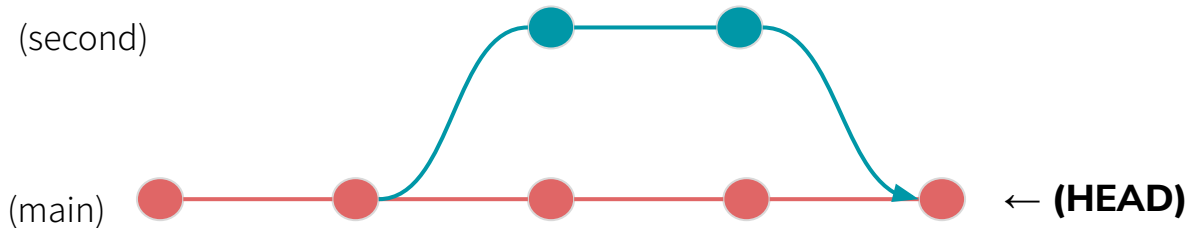
Me traigo los cambios de la otra rama con el siguiente comando.

```
git merge second
```

### TERCER PASO - En visual

Observó el resultado de la combinación. Se creó el commit correctamente, ¿Hubo conflictos?, ¿Sigue funcionando el programa?

```
git log -2
```



## PASOS PARA MERGEAR

### PRIMER PASO - En visual

Nos posicionamos en la rama donde queremos que el resultado de la unión se guarde.

```
git checkout main
```

### SEGUNDO PASO - En visual

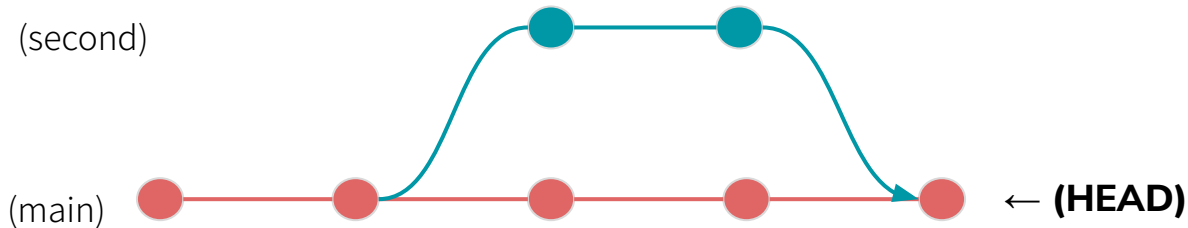
Me traigo los cambios de la otra rama con el siguiente comando.

```
git merge second
```

### TERCER PASO - En visual

Observó el resultado de la combinación. Se creó el commit correctamente, ¿Hubo conflictos?, ¿Sigue funcionando el programa?

```
git log -2
```



## PASOS PARA MERGEAR

### PRIMER PASO - En visual

Nos posicionamos en la rama donde queremos que el resultado de la unión se guarde.

```
git checkout main
```

### SEGUNDO PASO - En visual

Me traigo los cambios de la otra rama con el siguiente comando.

```
git merge second
```

### TERCER PASO - En visual


Observó el resultado de la combinación. Se creó el comité correctamente, ¿Hubo conflictos?, ¿Sigue funcionando el programa?

```
git log -2
```

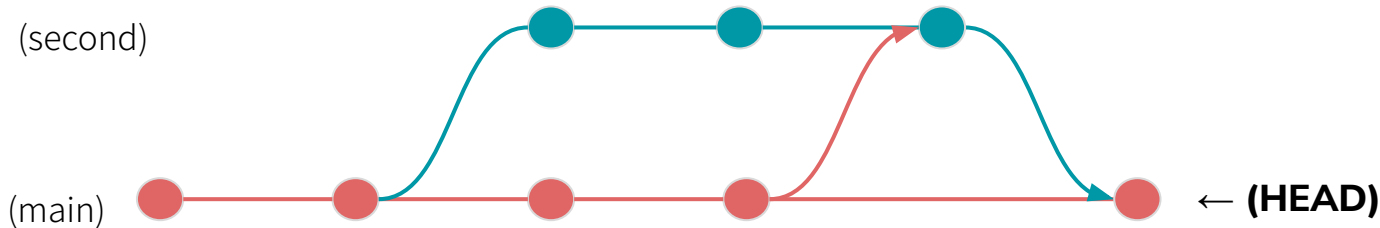
### ADVERTENCIA

Lamentablemente un merge no es tan sencillo. Esto no se hizo respetando las buenas prácticas, por lo que nos pudimos generar un gran inconveniente a nuestros compañeros.





**¿Cómo se combinan  
dos ramas con  
buenas practicas?**



## PASOS PARA MERGEAR

### PRIMER PASO - En visual

Nos posicionamos en la rama secundaria.

```
git checkout second
```

### SEGUNDO PASO - En visual

Traemos los cambios de la rama principal.

```
git merge main
```

### TERCER PASO - En visual

Confirmar si se creó el nuevo commit en la rama secundaria, y si hay conflictos lo resuelvo.

```
git log -3
```

### CUARTO PASO - En visual

Nos posicionamos en la rama principal y nos traemos los cambios.

```
git checkout main
```

```
git merge second
```



## FIN DE LA TEORÍA

[rapa.matias.e@gmail.com](mailto:rapa.matias.e@gmail.com)

Asunto : CURSO/DIA/TURNO