



# Desarrollo colaborativo con Git & Github

CLASE 3

# 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"`

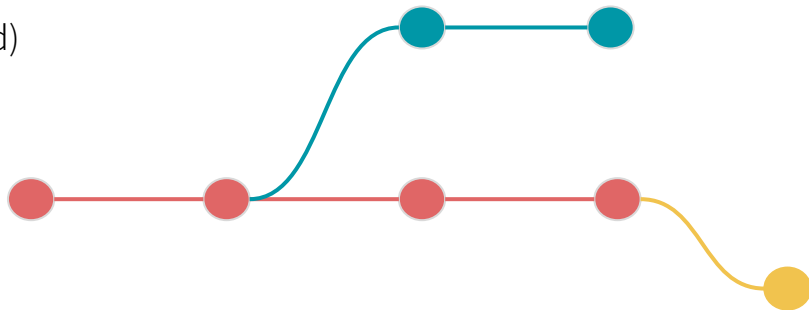
`git log`

`git push`



(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>
```

# OBJETIVOS DE LA CLASE 3

- ❑ ¿Qué es el área de stash?
- ❑ ¿Opciones alternativas del comando git merge?
- ❑ ¿Que es un tag?
- ❑ ¿Qué es un proceso de bisect?



## 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 stash
- ☐ git stash list
- ☐ git stash apply
- ☐ git bisect start
- ☐ git bisect bad
- ☐ git bisect good
- ☐ git bisect reset
- ☐ git tag

# 02

STASH





**¿Qué pasa si  
cambiamos de rama  
sin commitear  
antes?**



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

A

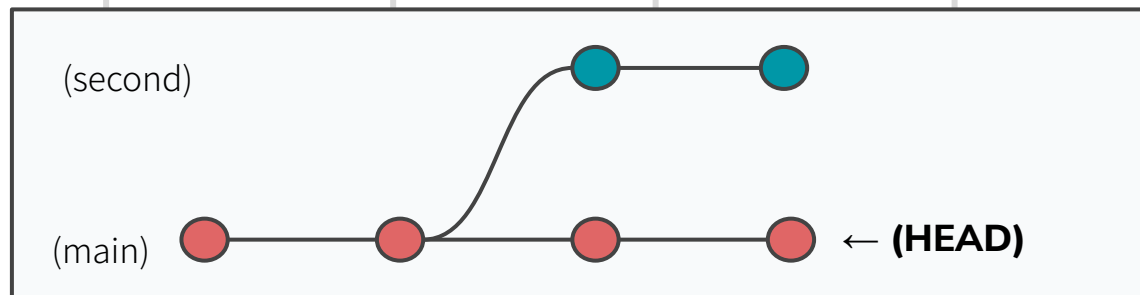
B

C

(second)

(main)

← (HEAD)



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

A

B

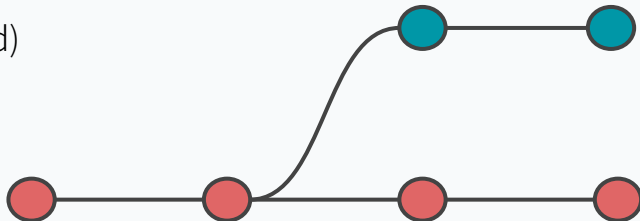
C

El área de Working y Staging son únicos para todas las ramas, por lo que no está en un commit se mueve de rama conmigo.

(second)

← (HEAD)

(main)





# El area de STASH

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

A

B

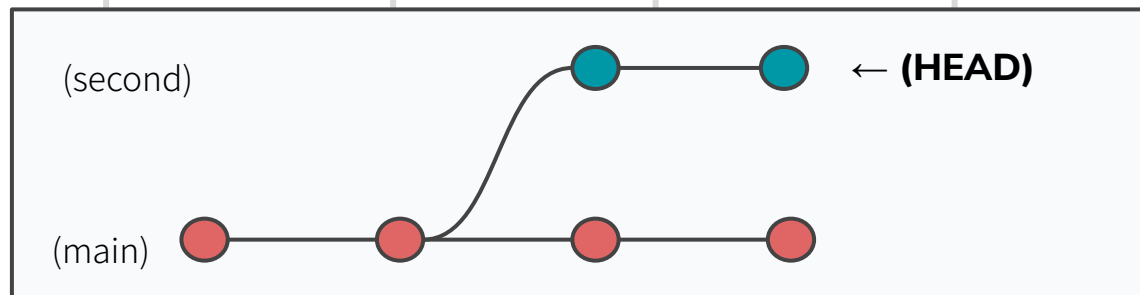
C

(second)

← (HEAD)

(main)

AREA DE STASH



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git stash`

`git status`

Stash 1

A

B

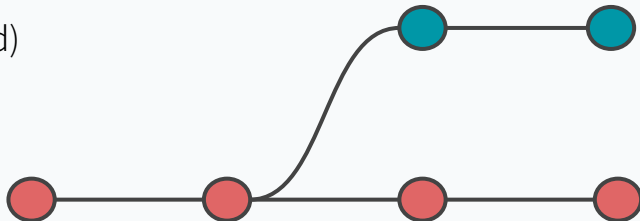
C


AREA DE STASH

(second)

← (HEAD)

(main)





**¿Cómo veo los archivos  
en el área de stash?**

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git stash`

`git status`

`git stash list`

Stash 1

A

B

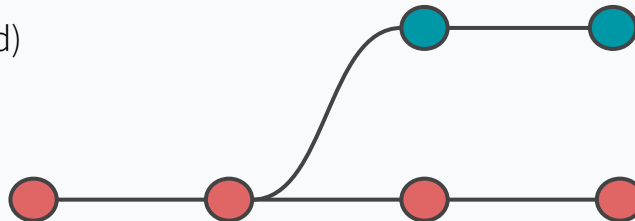
C

AREA DE STASH

(second)

(main)

← (HEAD)





**¿Cómo recupero los  
archivos del área de  
stash?**



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

A

B

C

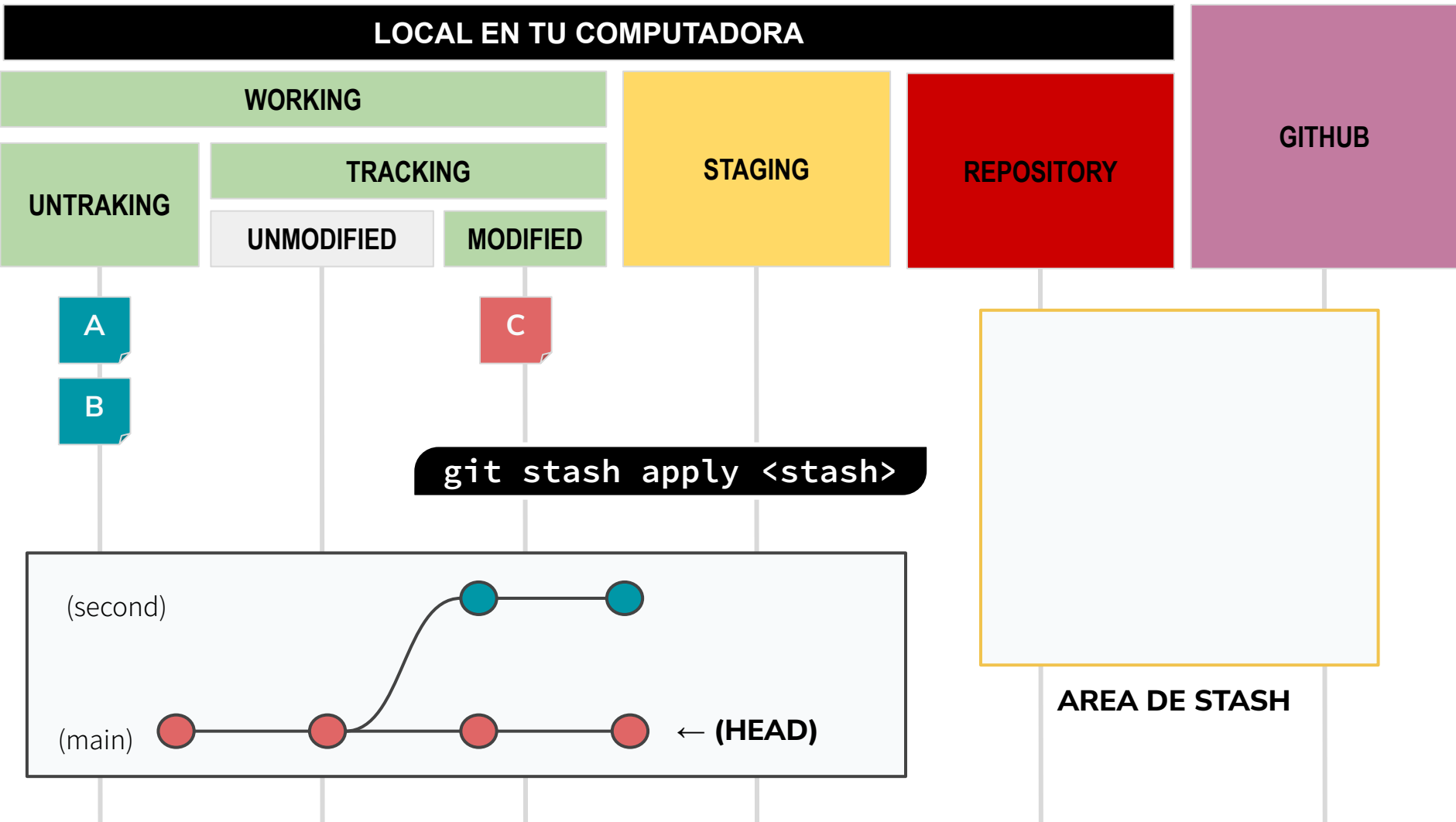
`git stash apply <stash>`

(second)

(main)

← (HEAD)

AREA DE STASH



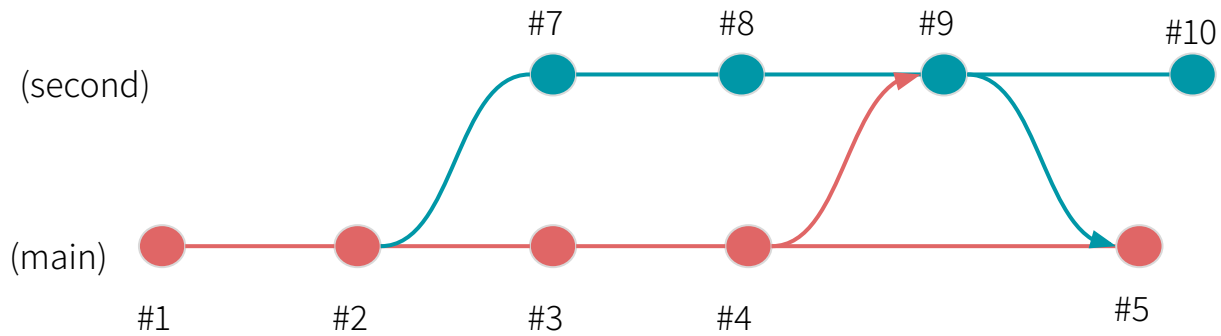
# 03

## COMPARACIÓN ENTRE COMMITS





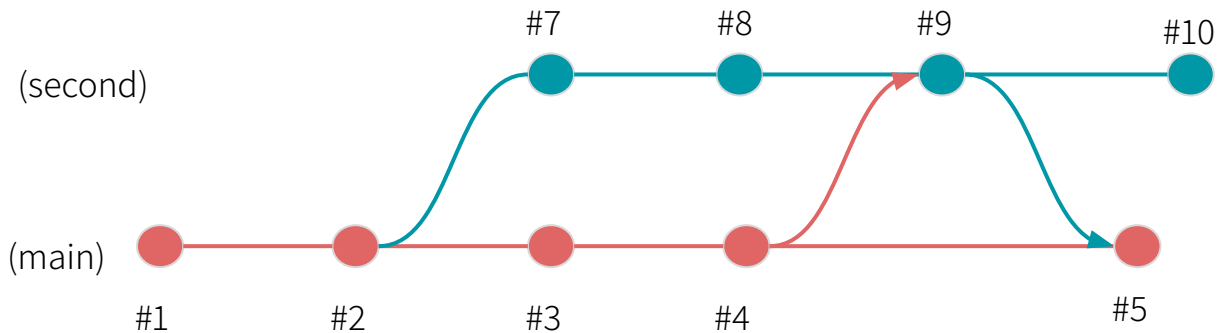
**¿Cómo puedo comparar  
dos commits?**



### USO SENCILLO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```



### USO SENCILLO

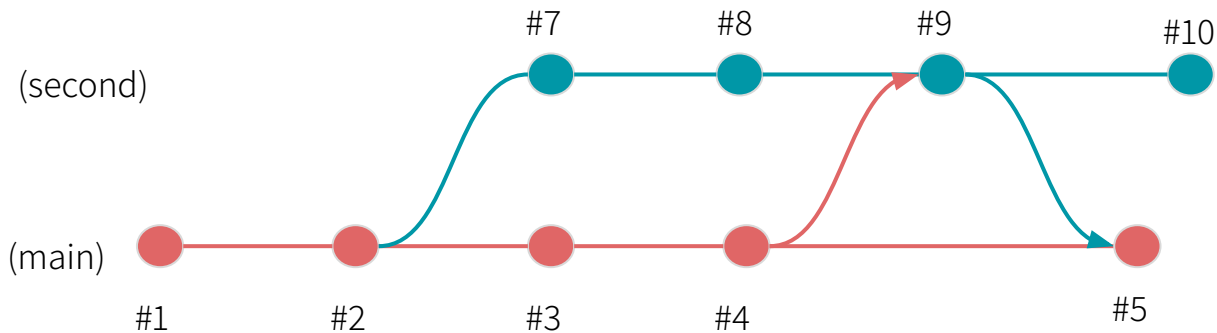
El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```

### USO INVERTIDO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_B> <#commit_A>
```



### USO SENCILLO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```

### USO SENCILLO

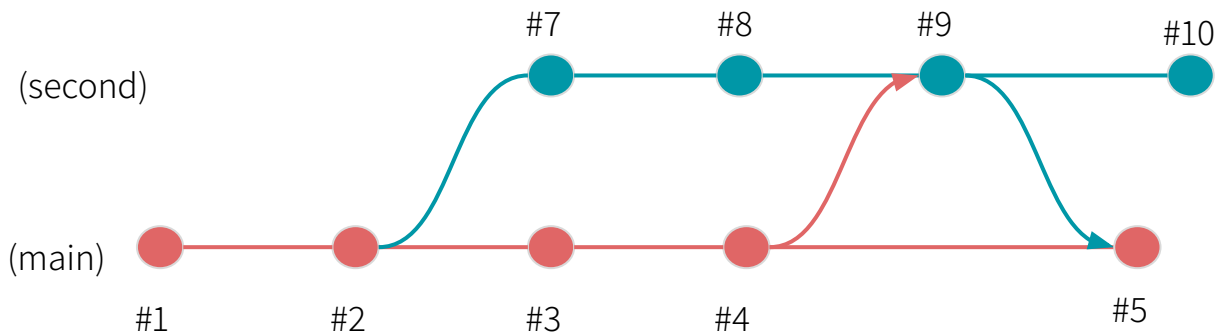
Si solo especifico un commit, entonces Git supondrá que estamos comparando con el HEAD.

```
git diff <#commit_A>
```

### USO INVERTIDO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_B> <#commit_A>
```



### USO SENCILLO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```

### USO POR DEFAULT

Si solo especifico un commit, entonces Git supondrá que estamos comparando con el HEAD.

```
git diff <#commit_A>
```

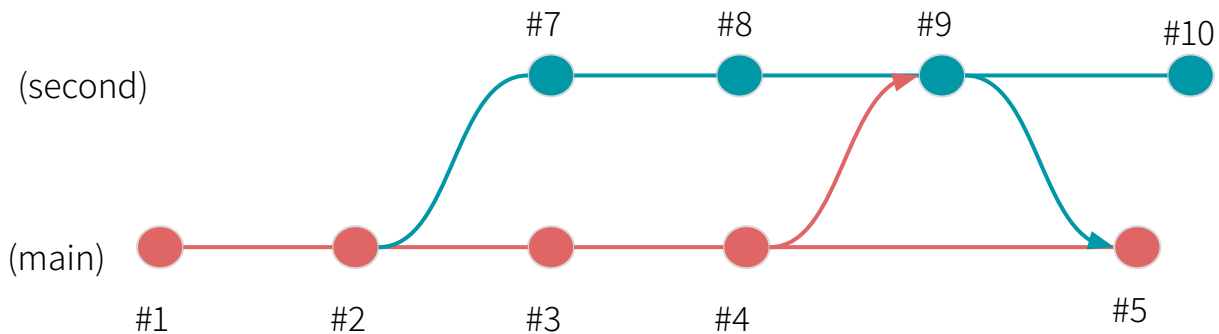
### USO INVERTIDO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_B> <#commit_A>
```

### USO SENCILLO

También puedo comparar commits de ramas distintas.



### USO SENCILLO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```

### USO POR DEFAULT

Si solo especifico un commit, entonces Git supondrá que estamos comparando con el HEAD.

```
git diff <#commit_A>
```

### USO INVERTIDO

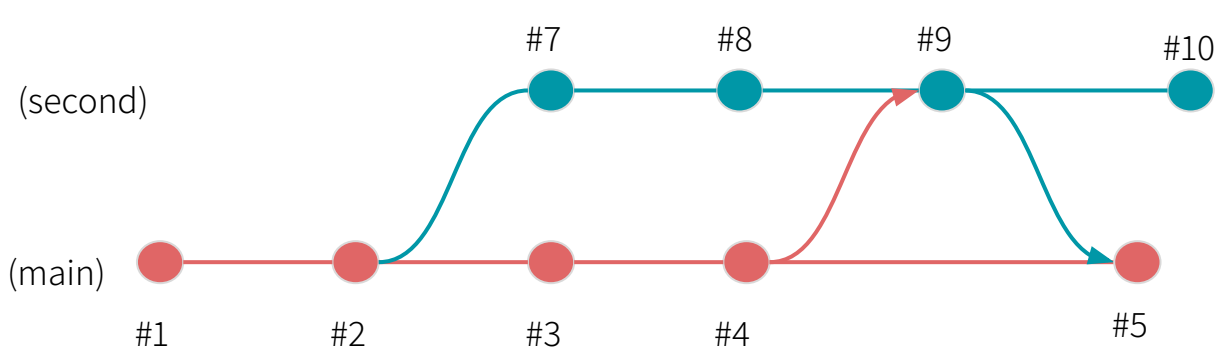
El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_B> <#commit_A>
```

### USO SENCILLO

También puedo comparar commits de ramas distintas.





### USO SENCILLO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_A> <#commit_B>
```

### USO POR DEFAULT

Si solo especifico un commit, entonces Git supondrá que estamos comparando con el HEAD.

```
git diff <#commit_A>
```

### USO INVERTIDO

El comando diff compara los archivos del commit\_A con los archivos del commit\_B.

```
git diff <#commit_B> <#commit_A>
```


### USO SENCILLO

También puedo comparar commits de ramas distintas.

# 01

ELIMINAR O COMPENSAR





**¿Cómo elimino  
errores que solo  
existen en mi  
computador? (No  
hiciste push)**

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

```
git reset --soft HEAD~1
```

NO HICISTE PUSH  
DE ESOS  
COMMITTS.

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

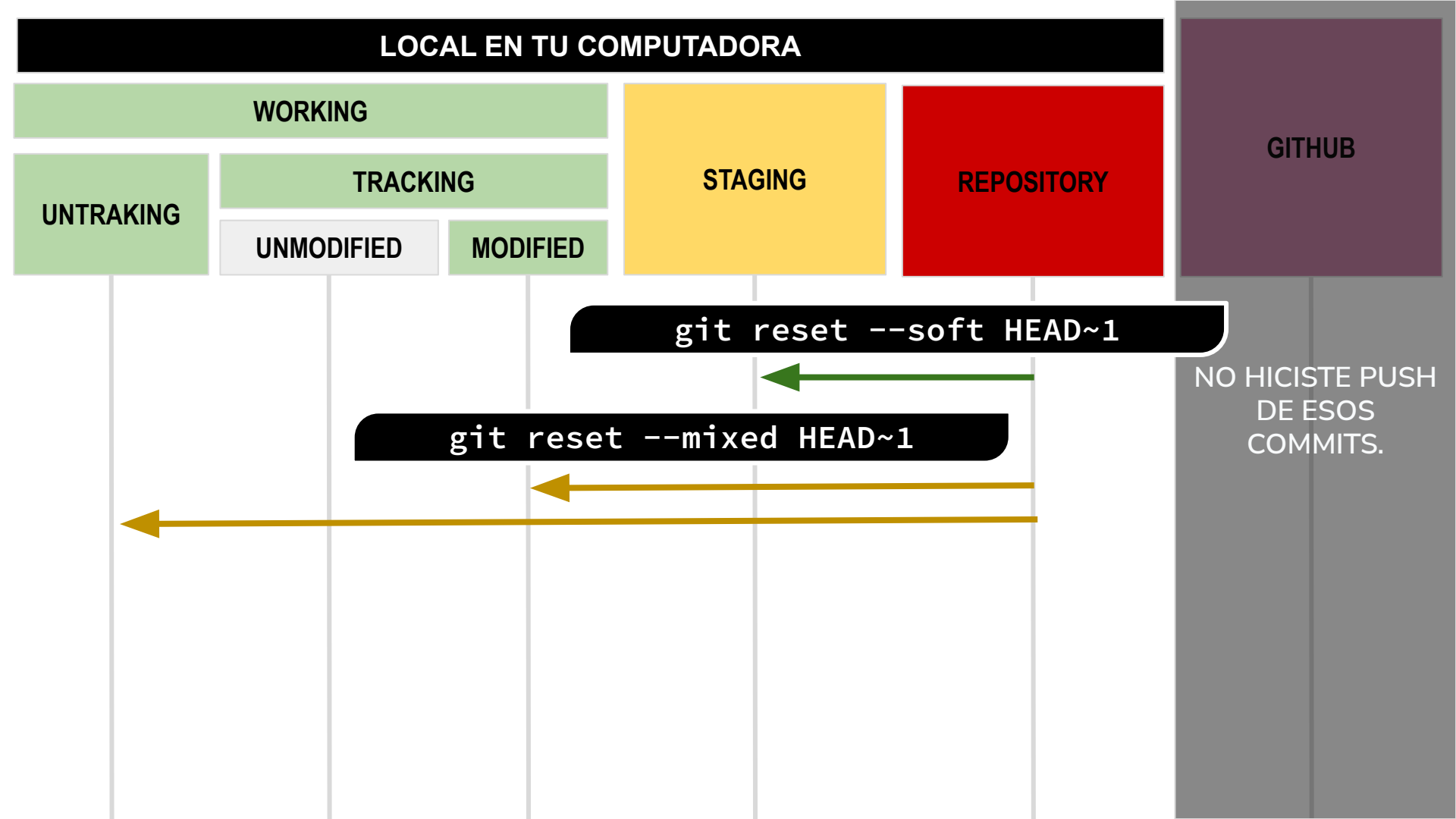
REPOSITORY

GITHUB

`git reset --soft HEAD~1`

`git reset --mixed HEAD~1`

NO HICISTE PUSH  
DE ESOS  
COMMITTS.



## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git reset --soft HEAD~1`

`git reset --mixed HEAD~1`

`git reset --hard HEAD~1`

NO HICISTE PUSH  
DE ESOS  
COMMITTS.



## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

```
git reset --soft <#commit>
```


```
git reset --mixed <#commit>
```

```
git reset --hard <#commit>
```

NO HICISTE PUSH  
DE ESOS  
COMMITTS.



NOTA: Se puede especificar un número de commit al cual llegar y borrar todo lo que existe después de ese commit, pero la realidad es un comando riesgoso.



**¿Que hago si ya subí  
el error? (Hiciste  
push)**



## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git revert <#commit>`

`git push`


(main) ● — ● — ● — ●<sup>A</sup> — ●<sup>A'</sup> ← (HEAD)

NOTA : No olvidar subir la compensación al repositorio o es lo mismo que nada.

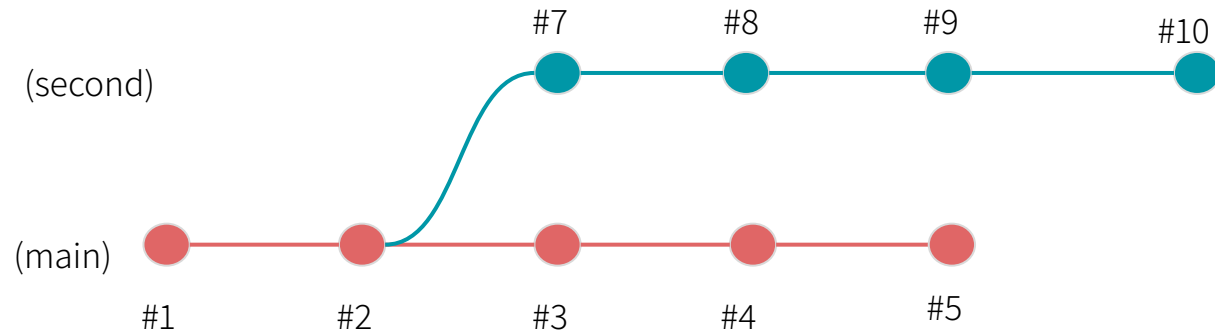
# 02

## OPCIONES ALTERNATIVAS A MERGE

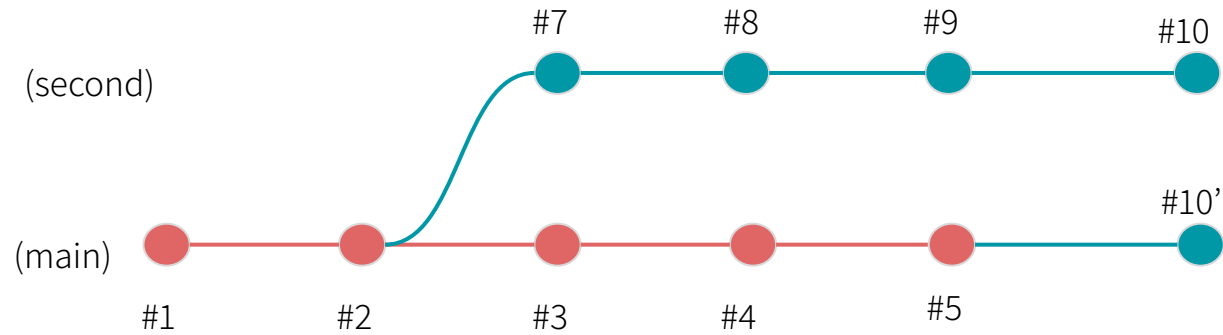




**Cherry-pick : Solo un  
commit, no toda la  
rama.**



```
git cherry-pick <#commit>
```



```
git cherry-pick <#commit>
```



## FIN DE LA TEORÍA

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

Asunto : CURSO/DIA/TURNO