




Git Avanzado

 2025-01   25 min.

Comandos

<https://github.com/joshnh/Git-Commands/blob/master/READMEes.md>

★ Concepto 7: Clonar

Acto de hacer una copia local de un repositorio remoto.

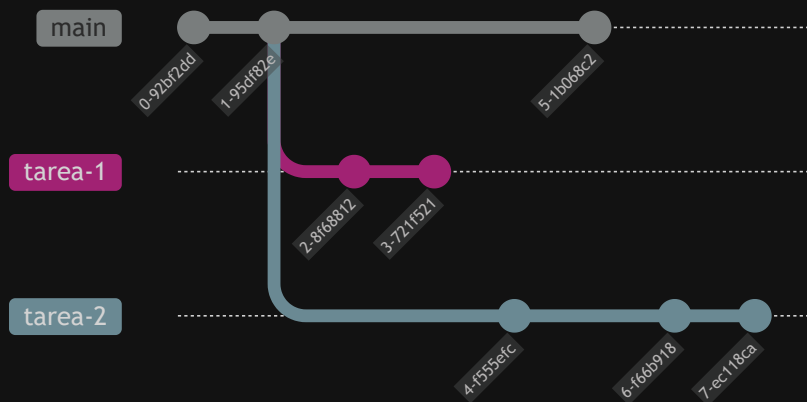
Actividad: Clonen su repositorio de talleres.

```
$ git clone [url]
```

★ Concepto 7+1: Branches

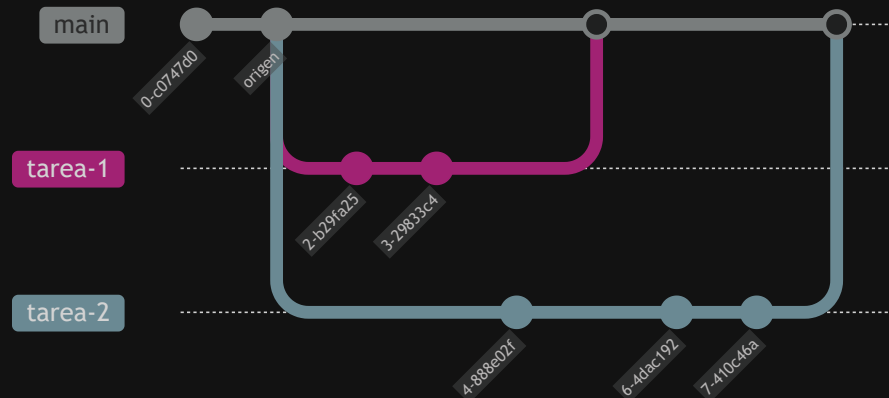
Una "branch", o rama, es una línea de desarrollo independiente al desarrollo principal.

Las ramas permiten el desarrollo simultaneo de varias tareas sin dañar el producto presentado al cliente, ni el trabajo de los compañeros.



★ Concepto 7+1: Branches

Las ramas parten de un commit "origen", y luego suelen juntarse a otra rama.



★ Concepto 7+1: Branches

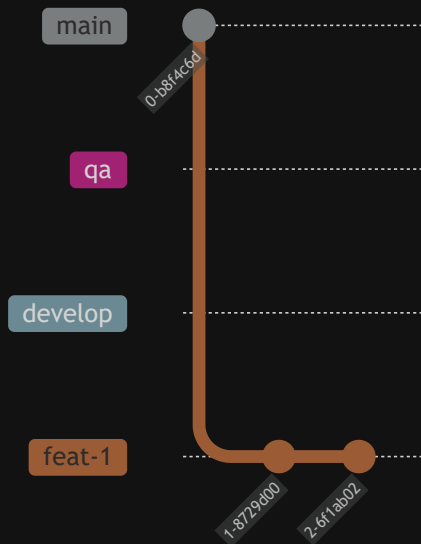
Actividad: Creen una branch en el local de su repositorio.

```
$ git branch [nombre]
```

Setup normal en produccion y flujo de trabajo

Por lo general, se manejan las siguientes ramas:

1. main / production / prod
2. staging / testing / qa
3. development / develop / dev
4. features / fixes / refactorors / chores / etc

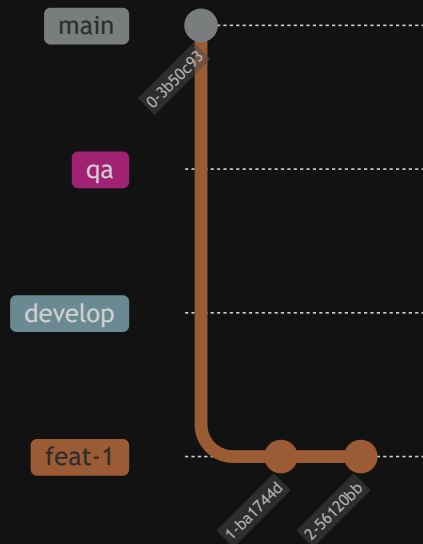


1. main / production / prod

Esta rama es la que usa el cliente final.

Es la versión actual desplegada, funcional y estable.

A esta rama no se permiten commits directos*.



* A MENOS QUE SEA 200% NECESARIO, LO HACE UN SENIOR DEVELOPER, Y ES LO QUE SE LE LLAMA UN HOTFIX.

2. staging / testing / qa

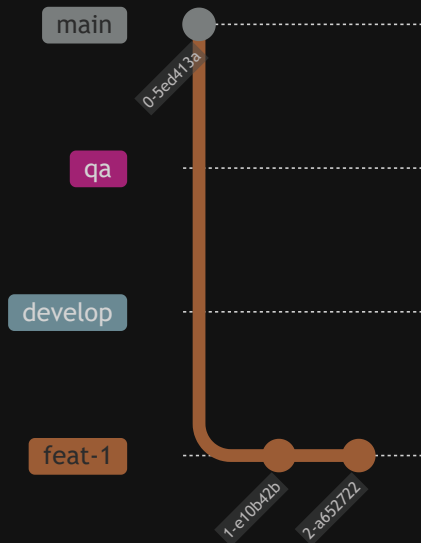
Esta rama se usa internamente para probar o demostrar la aplicación afuera del equipo de desarrollo.

Está desplegada en un link interno.

Es funcional y estable.

Al ser aprobada, todos los cambios de staging se mandan a main.

A esta rama no se permiten commits directos*.



* EN CASOS MUY EXTRAÑOS.

3. development / develop / dev

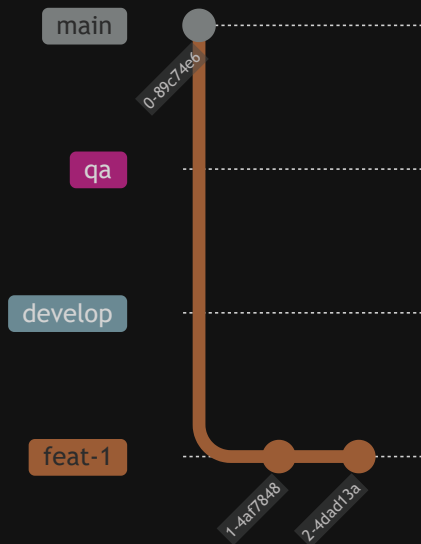
Esta rama se usa internamente para verificar que "hasta ahora todo está bien", usualmente adentro del equipo de desarrollo.

Está desplegada en un link interno.

A veces funciona. Estabilidad es pedir mucho.

Al ser aprobada, todos los cambios de dev se mandan a staging.

A esta rama no se permiten commits directos*.



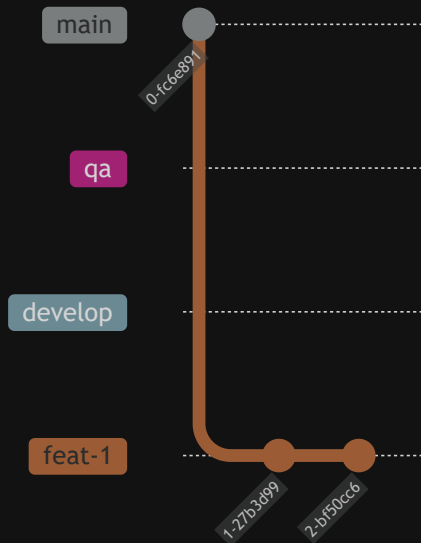
* PORQUE PARA QUE?

4. features / fixes / refactors / chores / etc

No es una sola rama, son varias.

Cada una de estas ramas representa una tarea en proceso de un desarrollador.

Al ser aprobada, la rama se junta con dev.



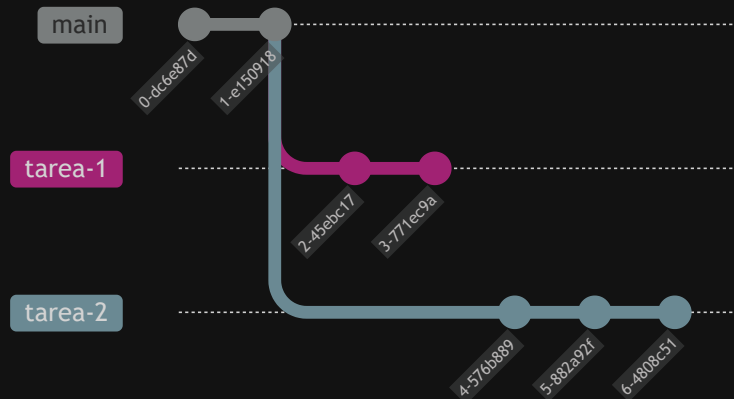
★ Concepto 9: Check Out

Acción de moverse entre 2 ramas.

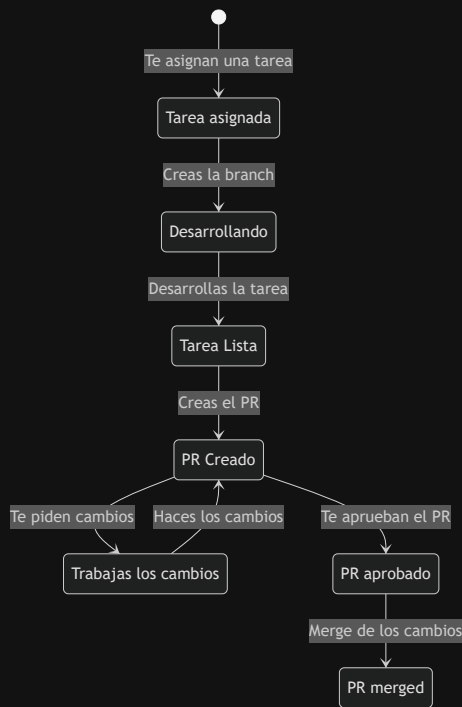
Al crear una branch, no nos movemos a la branch creada automáticamente.

Estoy en rama 1, hago check out a rama 2.

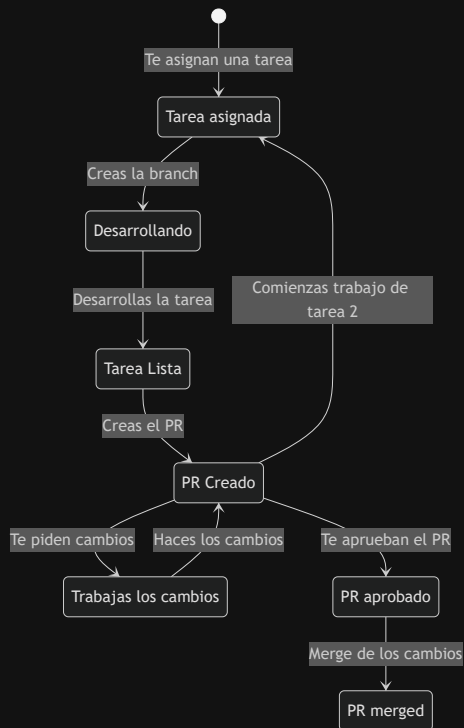
```
$ git checkout [nombre de rama]
```



Flujo de trabajo normal de un dev



Flujo de trabajo normal de un dev



★ Concepto 10: Stash

Guardar unos cambios en proceso para mas tarde.

Stash vs commit?

1. No se puede hacer checkout con cambios pendientes.
2. A veces un repositorio no permite commits con errores.
3. Quizá hiciste los cambios en la branch que no es.

★ Concepto 11: Merge

Merge es la acción de juntar una rama con otra, y con ella, sus cambios.

Actividad: Realicen cambios en la rama actual, hagan checkout a main, y hagan merge de los cambios.

- Agrega cambios a staging area

```
$ git add .
```

- Realiza commit de los cambios

```
$ git commit -m "mensaje de commit"
```

- Realiza un merge de una rama a la rama actual

```
$ git merge [nombre de la rama]
```

Quien decide los merges?

1. De feature -> dev, el líder del equipo de desarrollo, o cualquier persona que sepa lo que hace.
2. De dev -> staging, el líder del equipo de desarrollo, o el product owner/project manager
3. De staging -> production, el product owner/project manager

★ Concepto 12: Stash

Puedes crear un stash con los cambios que tengas, y hacer **pop**, es decir tomar los cambios y destruir la stash, o **apply**, es decir tomar los cambios y dejar la stash ahí.

Apply permite aplicar unos mismos cambios de una stash a varias branches, por ejemplo.

★ Concepto 12+1: Blame

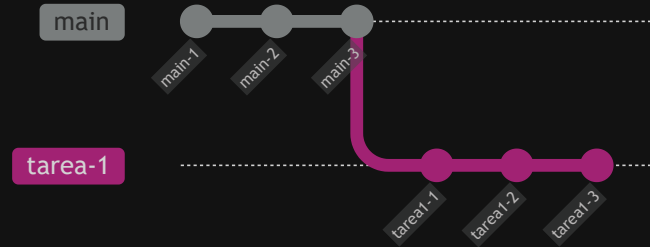
Accion que muestra el autor de un cambio.

★ Concepto 14: Cherry Pick

Accion de copiar un commit de un branch, o sus cambios, (con el proposito de aplicarlo de nuevo).

★ Concepto 15: Rebase

Accion de cambiar el commit origen de una branch/rama.



★ Concepto 16: Reset

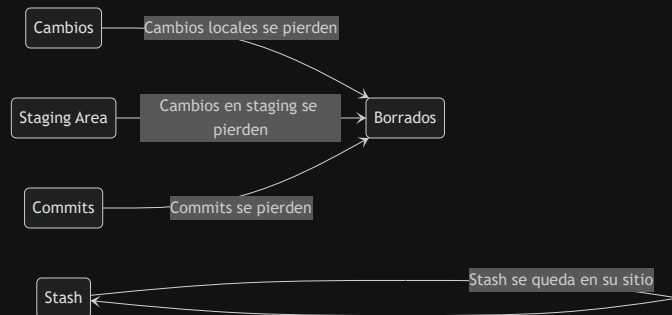
Un reset borra los commits de una branch desde el ultimo commit, hasta el commit seleccionado.

Los stashes no se ven afectados.

★ Concepto 16: Reset: Hard Reset

En un hard reset, toda la información de los commits a borrar se pierde.

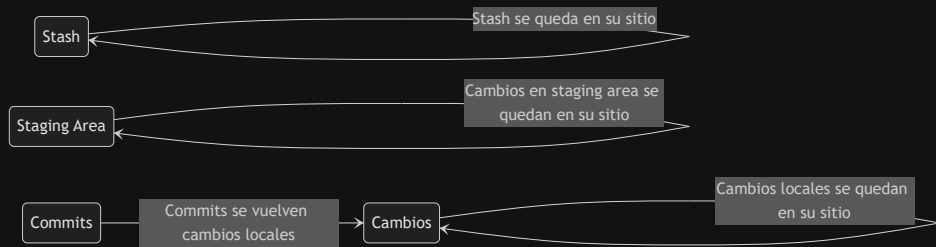
Cualquier cambio que se tenía en local, o en staging, se pierde.



★ Concepto 16: Reset: Soft Reset

En un soft reset, toda la información de los commits a borrar se vuelven cambios actuales.

Cualquier cambio que se tiene en local, o en staging, se queda donde estaba.

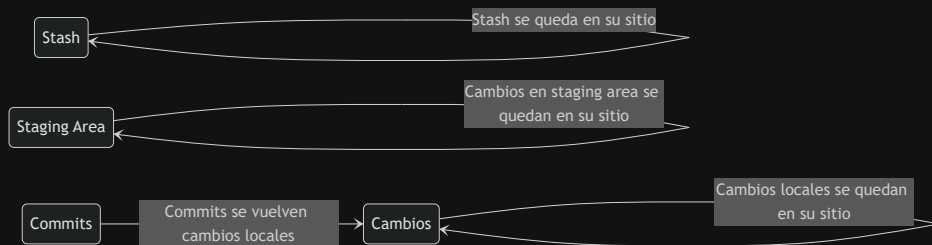


★ Concepto 16: Resets

Soft

En un soft reset, toda la información de los commits a borrar se vuelven cambios actuales.

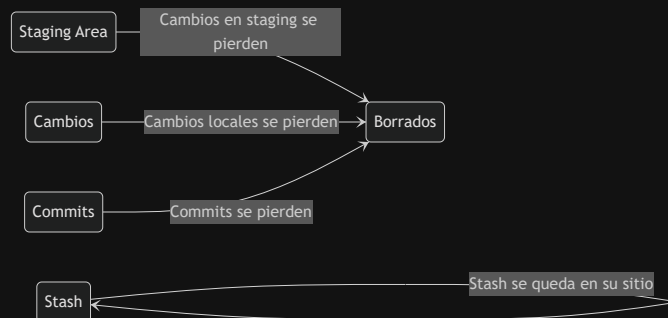
Cualquier cambio que se tiene en local, o en staging, se queda donde estaba.



Hard

En un hard reset, toda la información de los commits a borrar se pierde.

Cualquier cambio que se tenía en local, o en staging, se pierde.



★ Concepto 17: Conflictos

Sucede cuando dos branches que se intentan merge tienen cambios en la misma ubicacion (aproximadamente).

Para terminar el merge, esos conflictos deben resolverse.

En el codigo, los conflictos se ven asi

```
// <<<<<< HEAD
// codigo1
// =====
// codigo2
// >>>>>> new_branch_to_merge_later
```

★ Concepto 17+1: Revisiones y Aprobaciones

En un ambiente profesional, las ramas no se les hacen *merge* así como si nada.

Otro desarrollador lo debe revisar y lo aprobar.

Tras ser aprobado, se hace *merge*.

Usualmente no se rechazan, solo se siguen trabajando.



Saben Git!