Git Avanzado

📝 2024-03 — ⏱️ 25 min.

★ Concepto 7: 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: Branches

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

Que otra rama?

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 / refactors / 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 8: Merge

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

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

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 9: Check Out

Accion de moverse entre 2 ramas.

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

Flujo de trabajo normal de un dev

1. Creas una rama
2. Trabajas la tarea asignada, separando el progreso en commits
3. Al terminar la tarea, creas un pull request o merge request (es lo mismo)
4. Si te piden cambios, los realizas.
5. Te aprueban la tarea, y tu branch se merge con la dev branch.

Flujo de trabajo normal de un dev

1. Creas la rama para la tarea #1
2. Trabajas la tarea
3. Al terminar la tarea, creas el PR
4. Mientras que te revisan la branch de la tarea, te devuelves al paso 1 pero con la tarea # i+1

★ 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 10: 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 ahi.

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

★ Concepto 11: Blame

Accion que muestra el autor de un cambio.

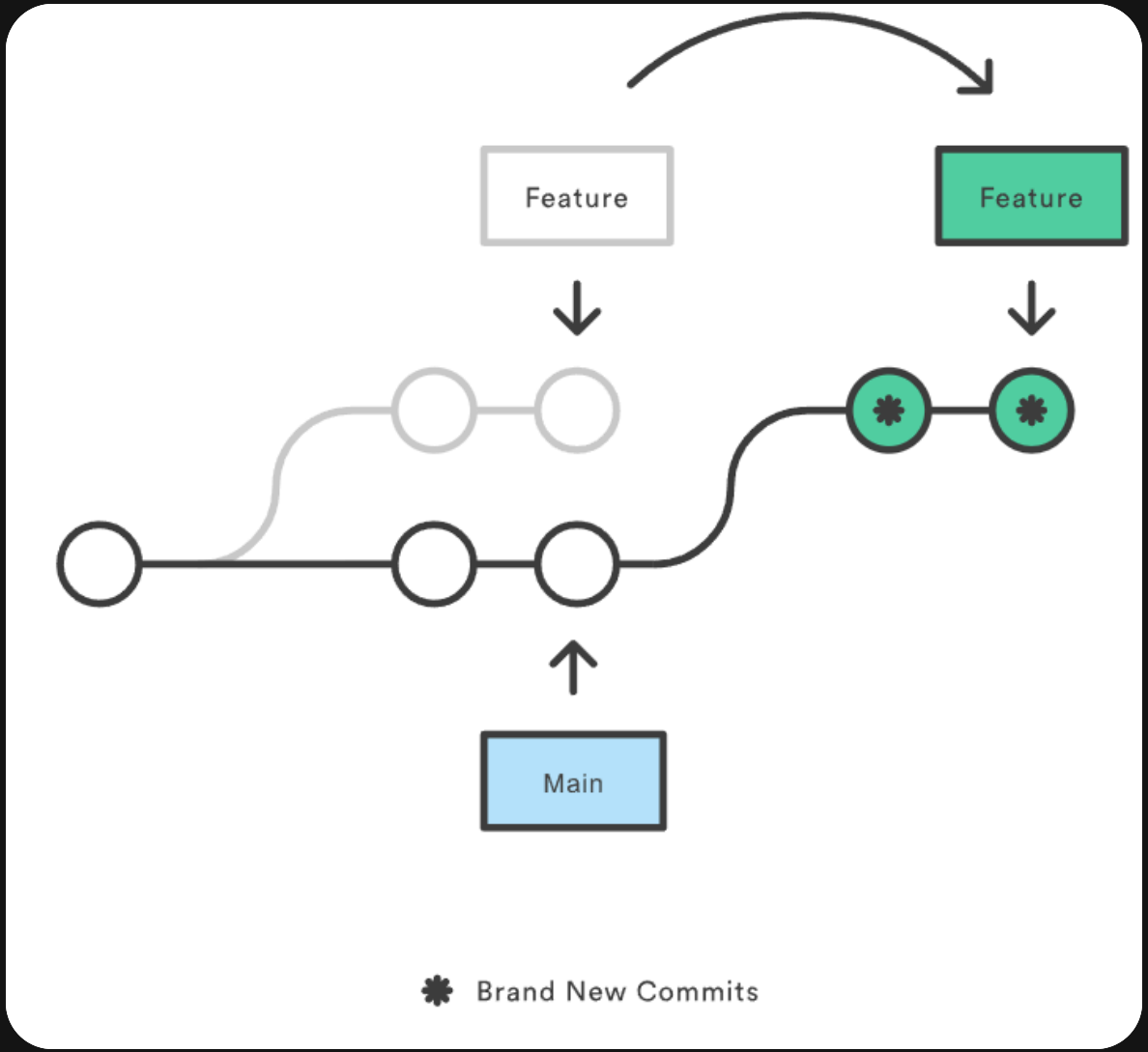
★ Concepto 12: Cherry Pick

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

★ Concepto 12+1: Rebase

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

★ Concepto 12+1: Rebase



★ Concepto 14: Reset

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

Los stashes no se ven afectados.

★ Concepto 14: Reset: Hard Reset

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

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

★ Concepto 14: Reset: Soft Reset

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

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

★ Concepto 15: Conflictos

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

Para terminar el merge, esos conflictos deben resolverse.

★ Concepto 15: Conflictos

En el codigo, los conflictos se ven asi

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

★ Concepto 16: Revisiones y Aprobaciones

En un ambiente profesional, las ramas no se merge asi como si nada.

Otro desarrollador lo debe revisar y lo aprobar.

Tras ser aprobado, se hace merge.

Usualmente no se niegan, solo se siguen trabajando.

Felicidades!

Saben Git!