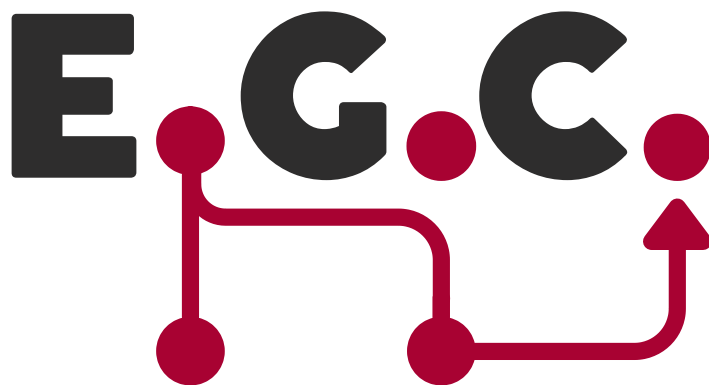




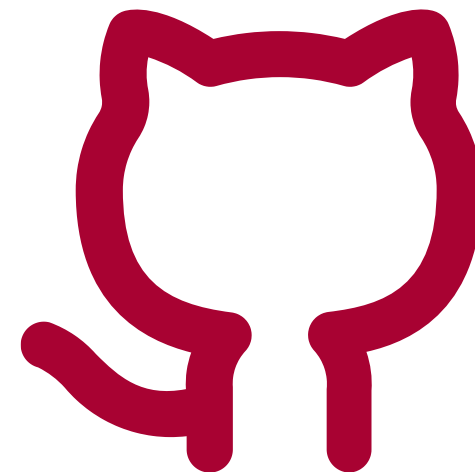
Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

Práctica 3 Gestión del código fuente

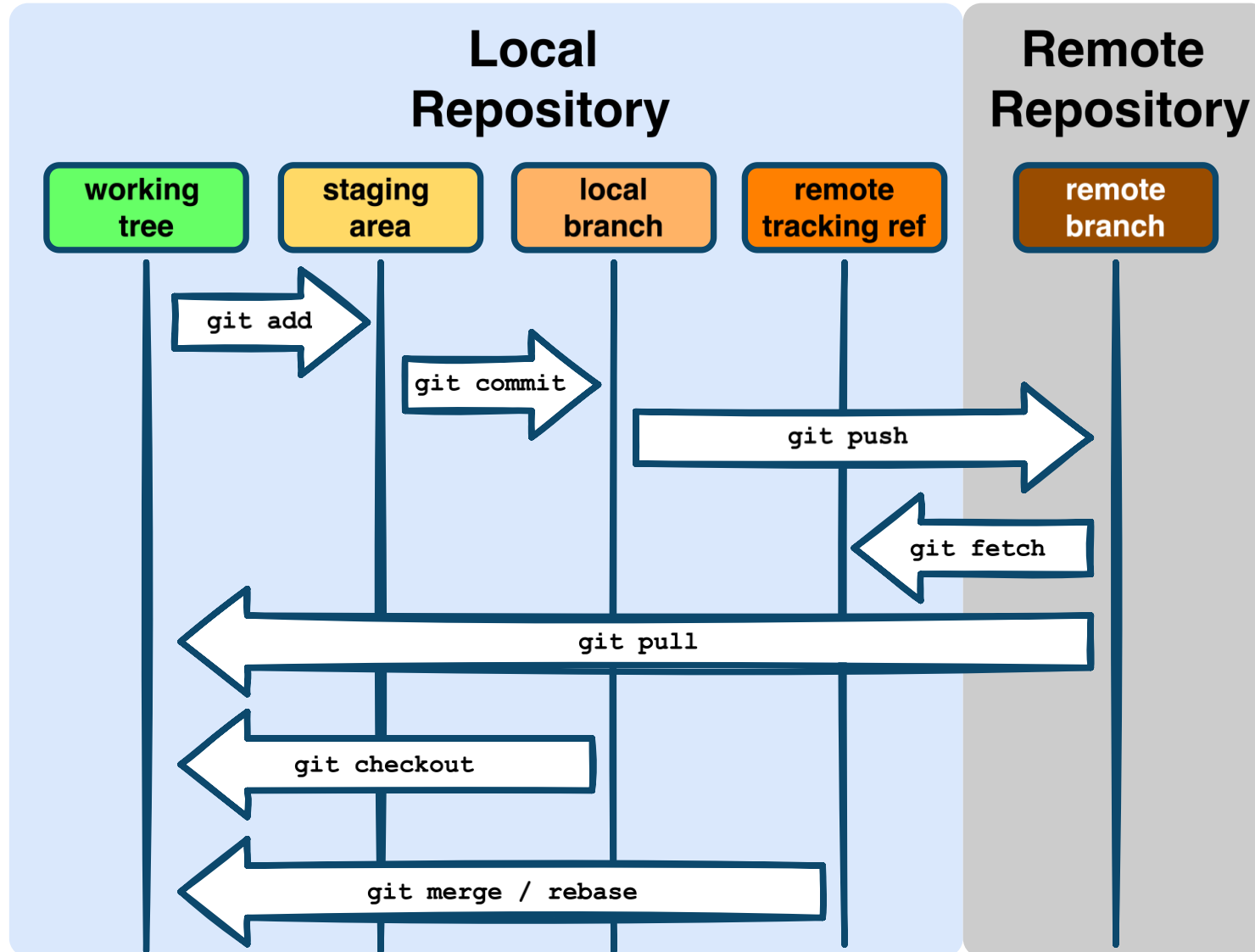


1. **Flujo de trabajo**
 2. **Ramas**
 3. **Conceptos avanzados de Git**
 4. **Para ayudarte a entrenar con Git**
 5. **Y yo, ¿qué puedo hacer en mi proyecto?**
 6. **Ejercicio práctico: first *commit* dates**
- 

1. Flujo de trabajo



IMPORTANT

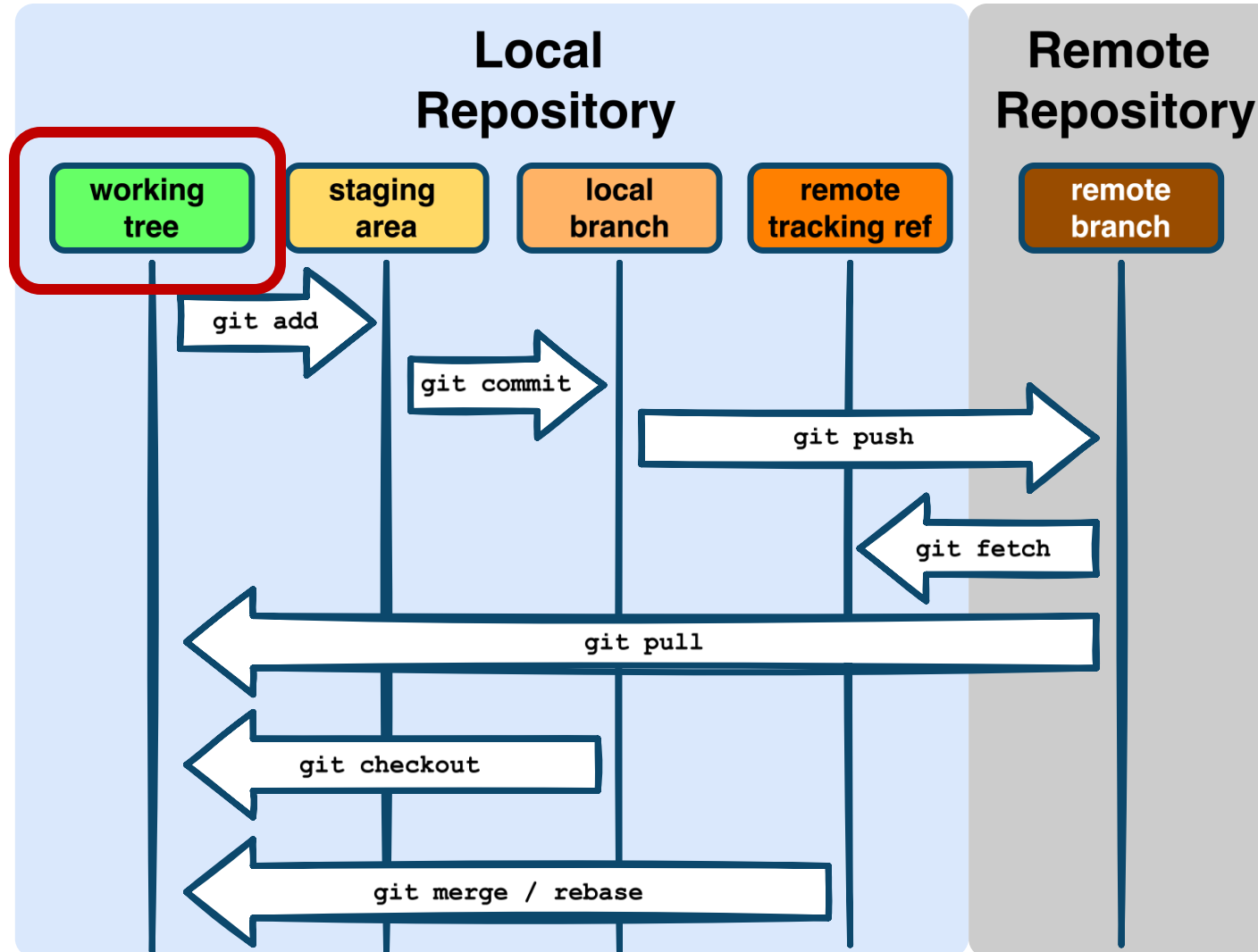


IMPORTANT

IMPORTANT

IMPORTANT

1. Flujo de trabajo

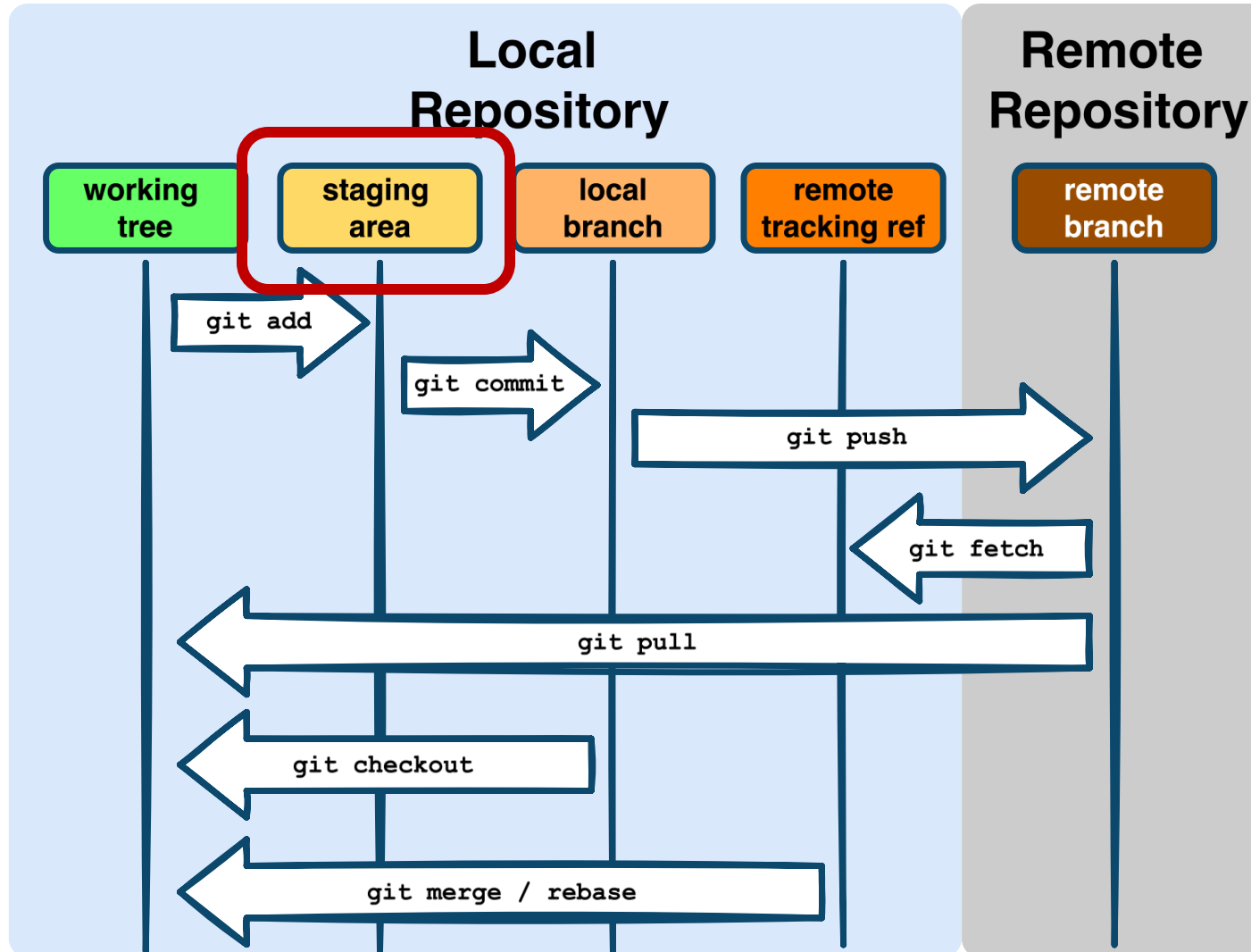


Working tree (árbol de trabajo)

Trabajas directamente con tus archivos, es decir, donde haces cambios en tu proyecto

Esto es lo que ves y editas

1. Flujo de trabajo

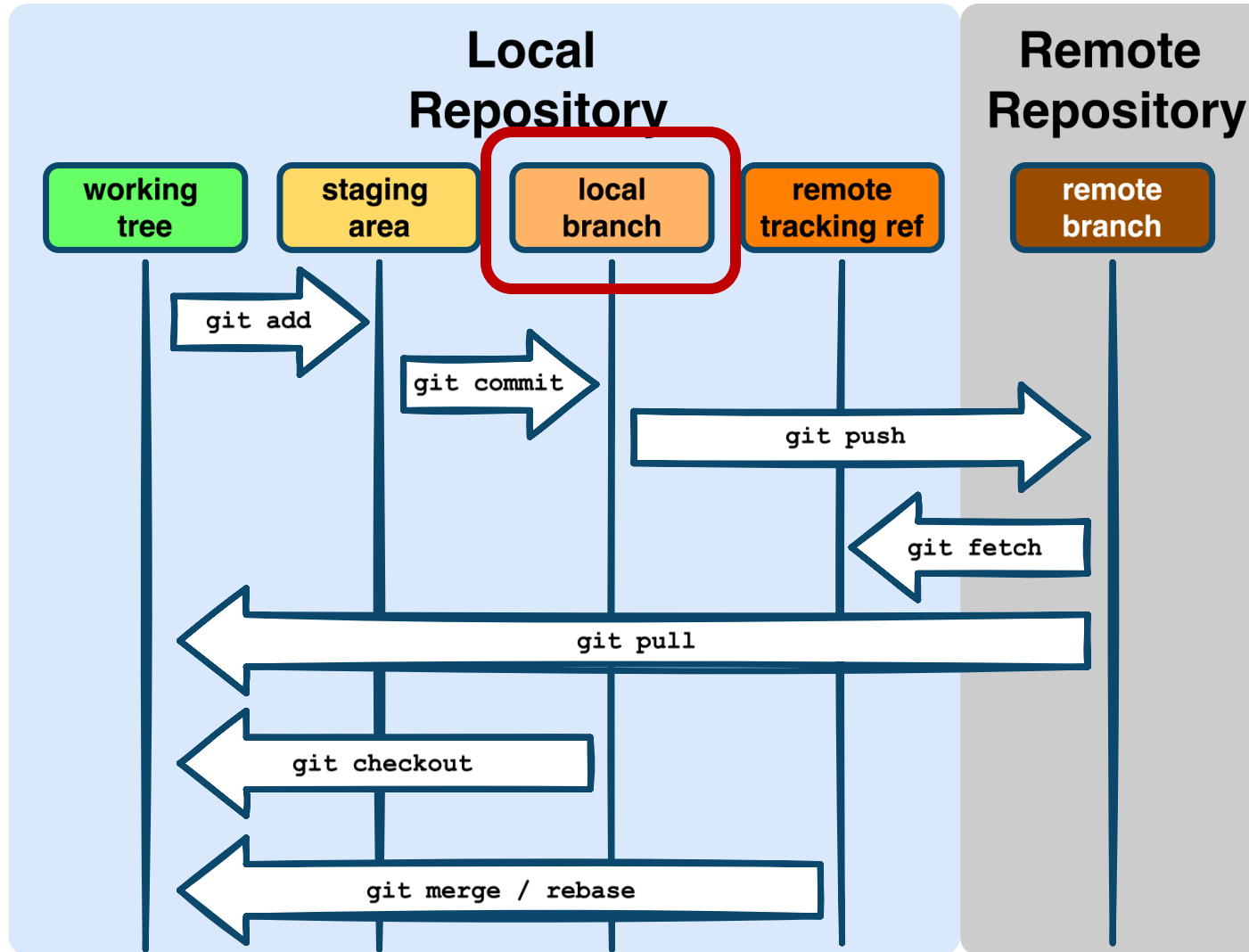


Staging area (área de preparación)

Es una "sala de espera" donde colocas los cambios que quieres confirmar

Es el proceso anterior a guardar tus cambios

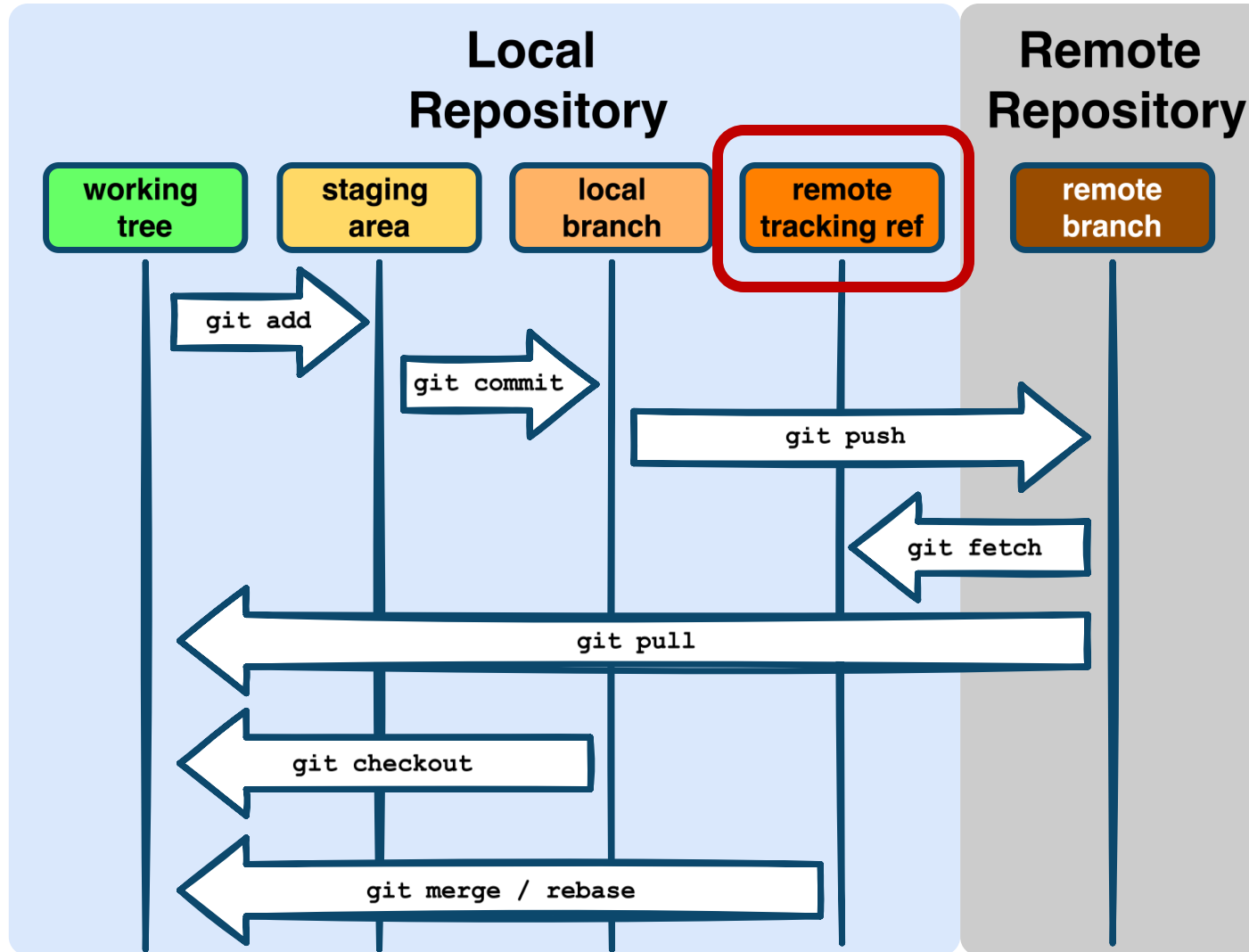
1. Flujo de trabajo



Local branch (rama local)

Guardo los cambios de manera permanente

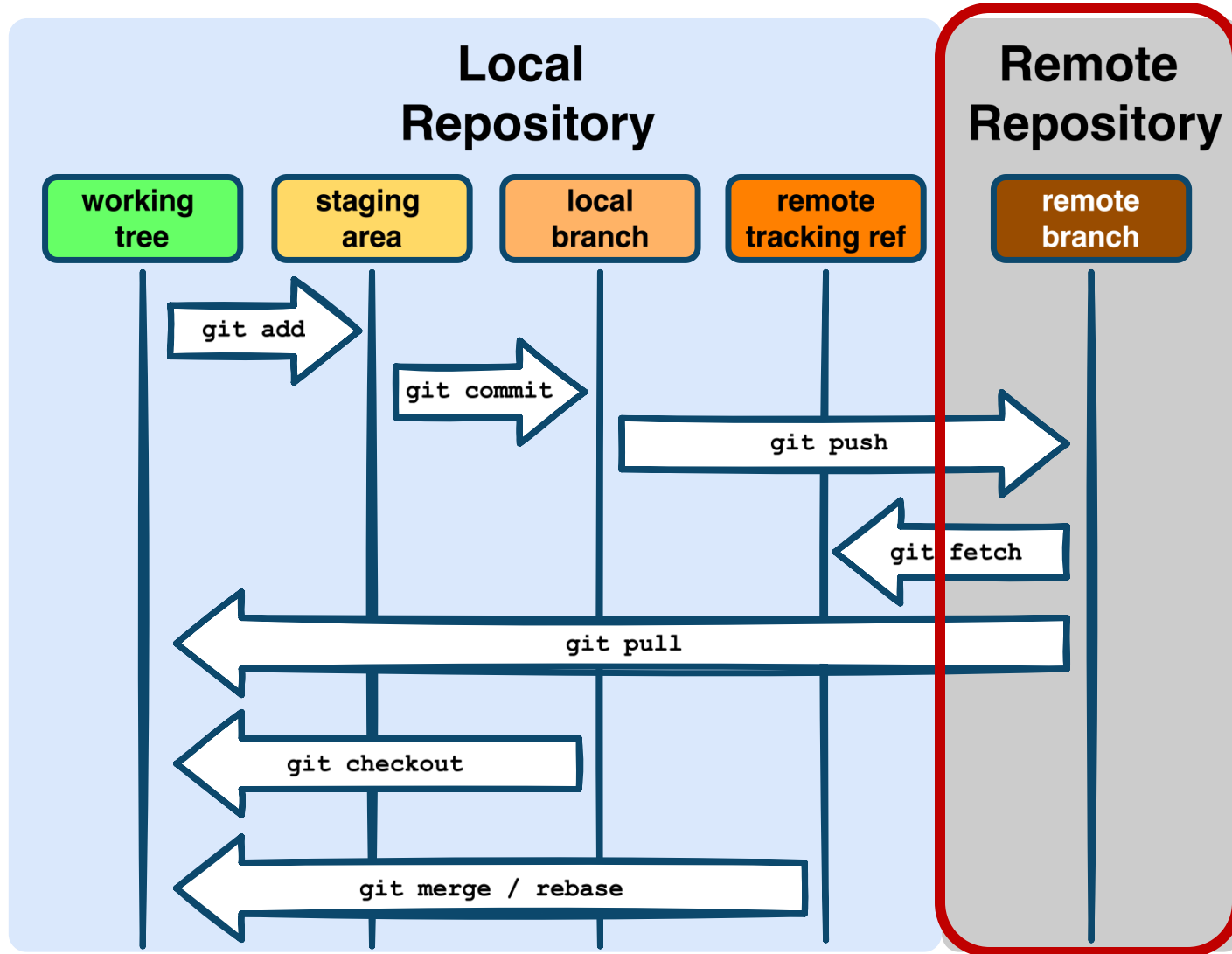
1. Flujo de trabajo



Remote tracking ref (referencia de seguimiento remoto)

Rastrea el estado de las ramas remotas sin mezclar directamente los cambios con tu rama local

1. Flujo de trabajo

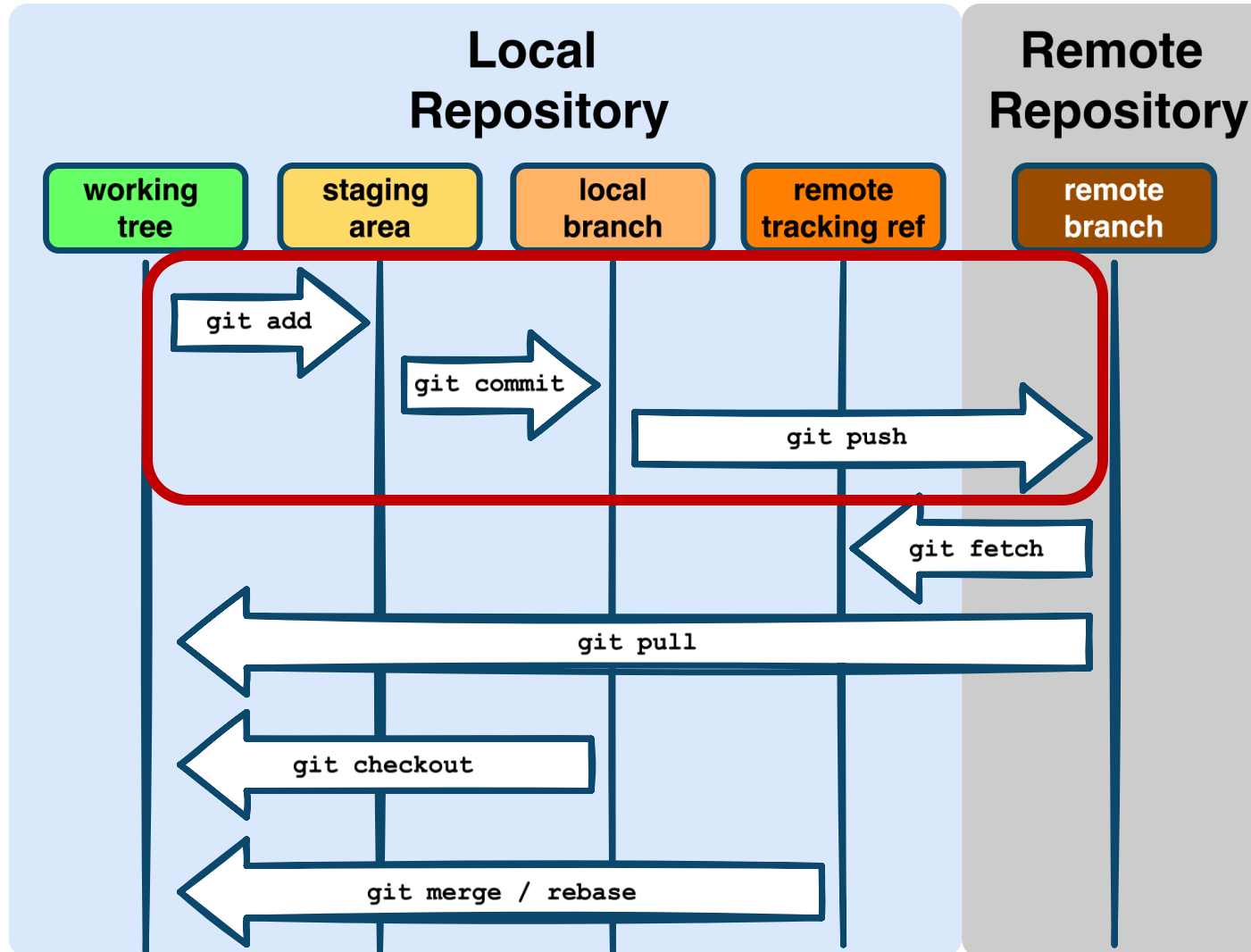


Remote repository (repositorio remoto)

Es una copia de tu proyecto en un servidor como GitHub

Permite la colaboración entre otros miembros

1. Flujo de trabajo



git add

añade cambios al área de preparación

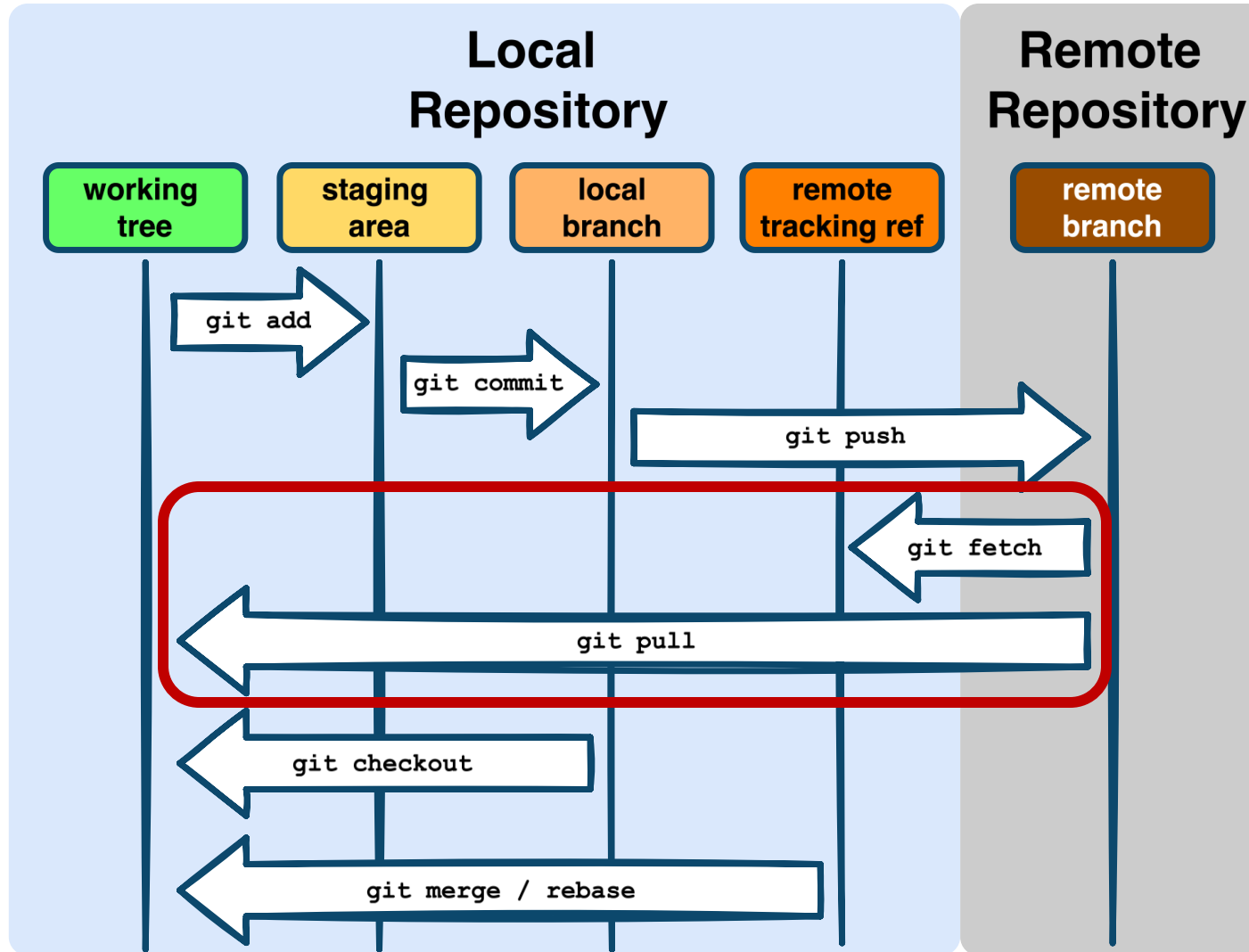
git commit

guarda esos cambios en el historial

git push

envía los commits al repositorio remoto

1. Flujo de trabajo



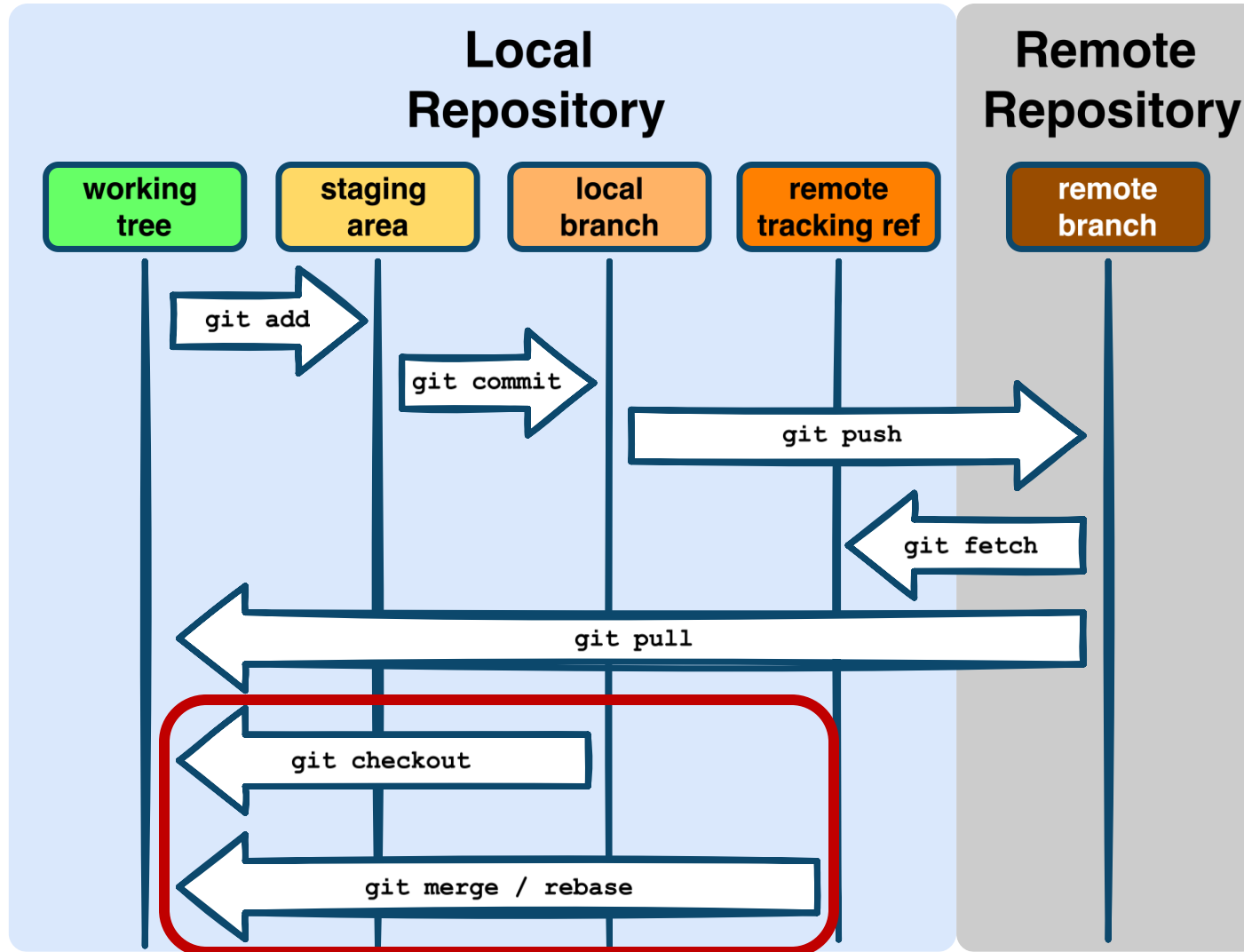
git fetch

descarga los cambios sin fusionarlos con tu código

git pull

descarga los cambios y los fusiona con tu código

1. Flujo de trabajo



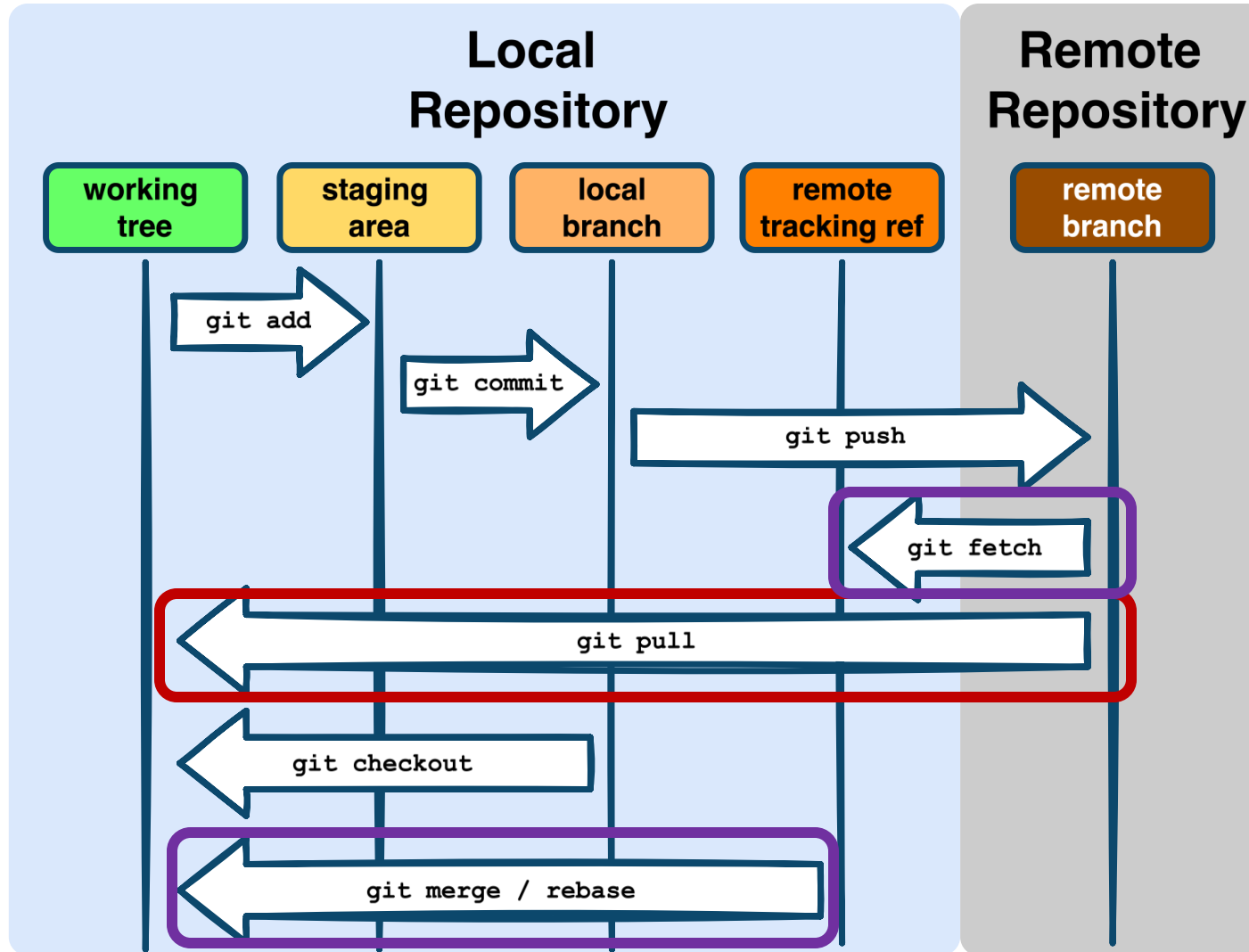
git checkout

- cambiar de rama
- restaurar archivos a un estado anterior
- cambiar a un commit específico

git merge / rebase

combinar ramas

1. Flujo de trabajo



git pull = git fetch + git merge

primero descarga los cambios del repositorio remoto y luego los fusiona

1. Flujo de trabajo
 2. **Ramas**
 3. Conceptos avanzados de Git
 4. Para ayudarte a entrenar con Git
 5. Y yo, ¿qué puedo hacer en mi proyecto?
 6. Ejercicio práctico: first *commit* dates
- 

2. Ramas

Comandos básicos

Crear rama

```
git branch <nombre_rama>
```

Saltar a la rama

```
git checkout <nombre_rama>
```



```
git switch <nombre_rama>
```

Crear y saltar a la rama (en un solo comando)

```
git checkout -b <nombre_rama>
```



2. Ramas

Comandos básicos

Subir rama a repositorio remoto

```
git push -u origin <nombre_rama>
```

-u = upstream = referencia de seguimiento, es la conexión entre la rama local y la rama remota



Listar ramas locales

```
git branch
```

Listar ramas remotas

```
git branch -r
```

Listar ramas locales y remotas

```
git branch -a
```

2. Ramas

Comandos básicos

Eliminar rama local (si la rama ya ha sido fusionada con otra)

```
git branch -d <nombre_rama>
```

Eliminar rama local (forzar eliminación)

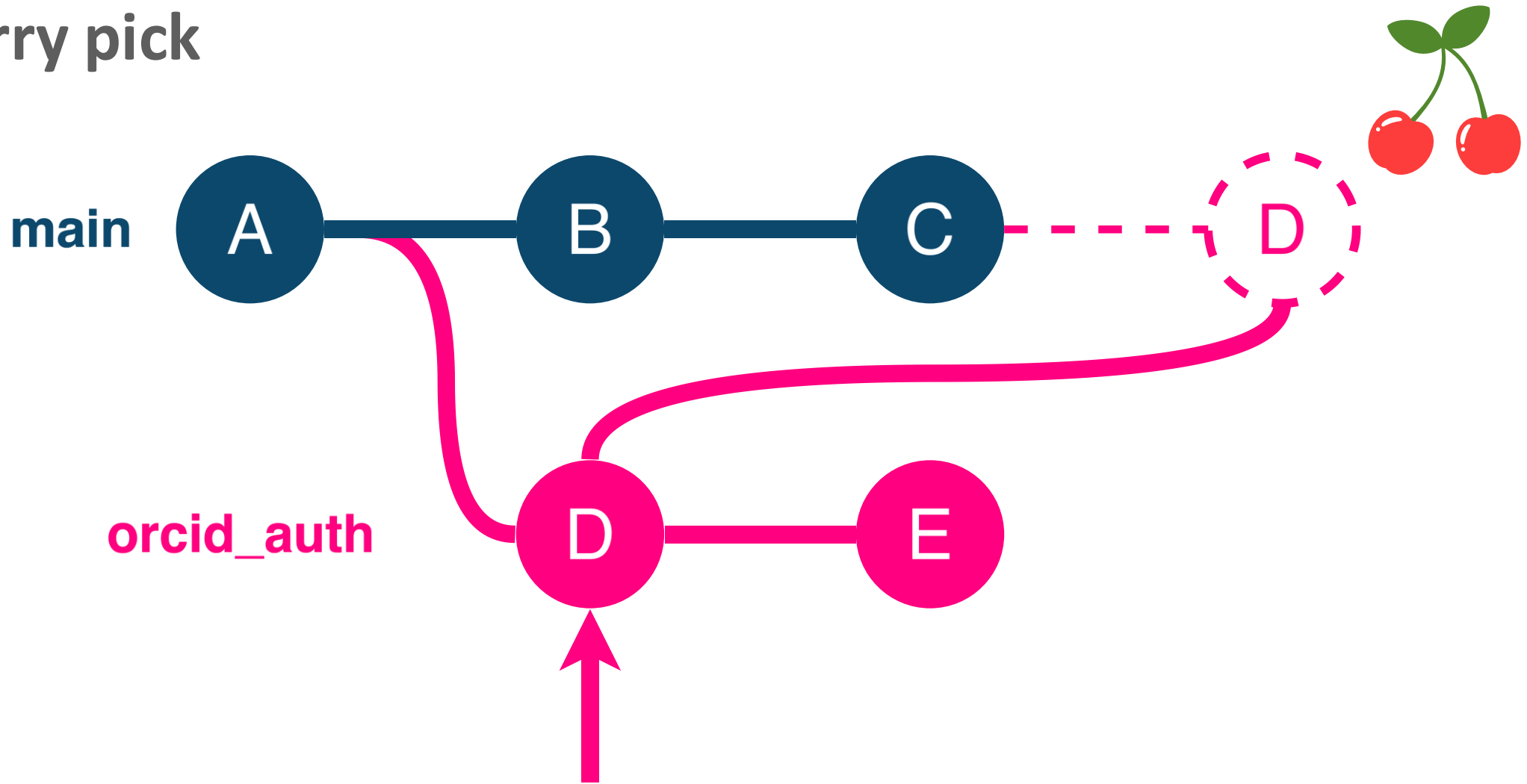
```
git branch -D <nombre_rama>
```

Eliminar rama remota

```
git push origin --delete <nombre_rama>
```

2. Ramas

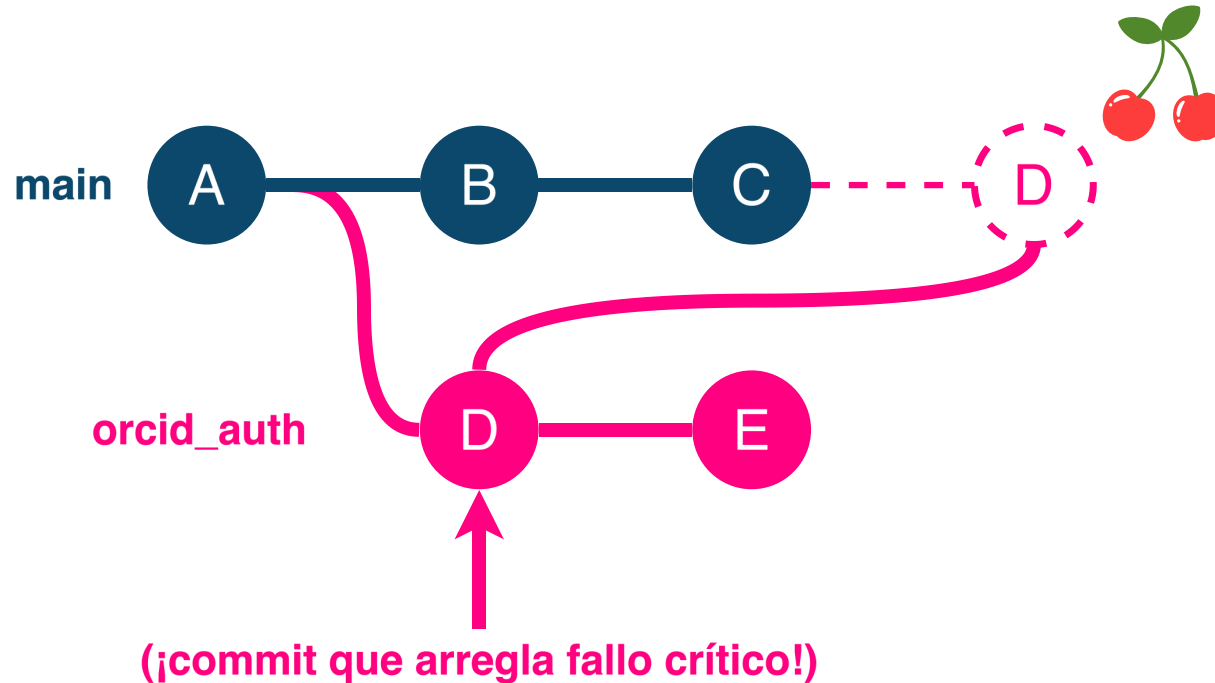
Cherry pick



(¡commit que arregla fallo crítico!)

2. Ramas

Cherry pick



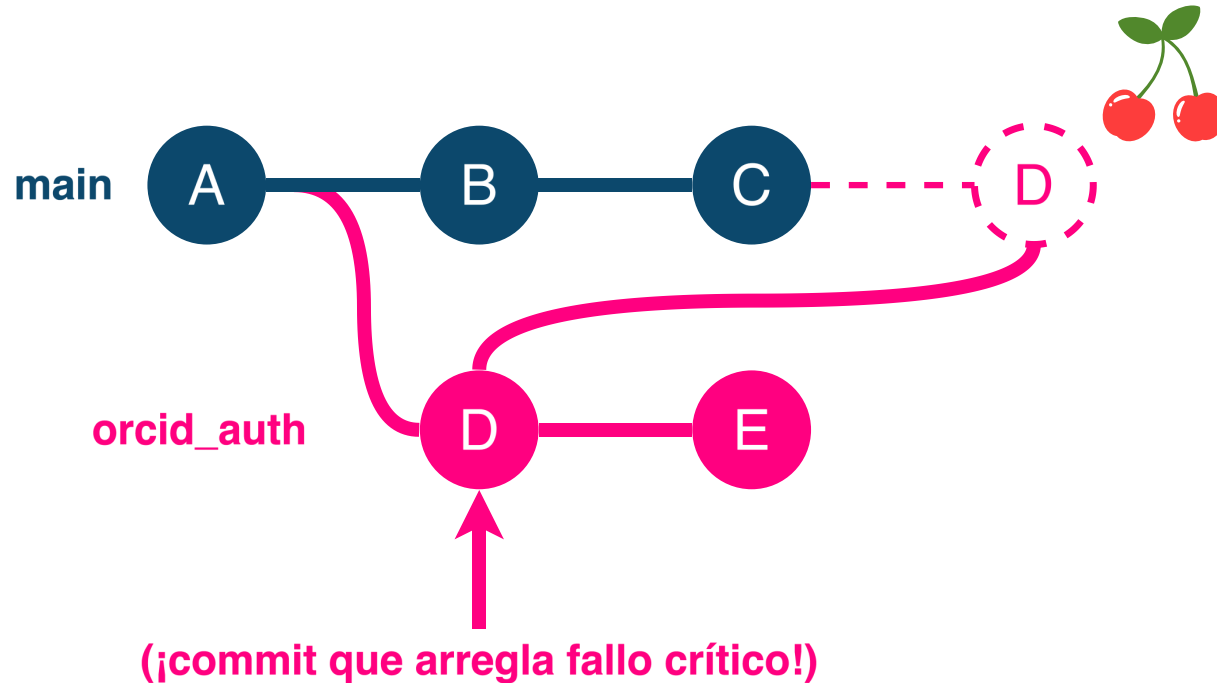
Aplicamos **un cambio específico** (un commit) de una rama a otra, sin fusionar toda la rama

Traerse el commit (estando en la rama *main*)

```
git cherry-pick <commit_hash>
```

2. Ramas

Cherry pick



git cherry-pick --continue
en caso de conflicto y
resolverlo manualmente

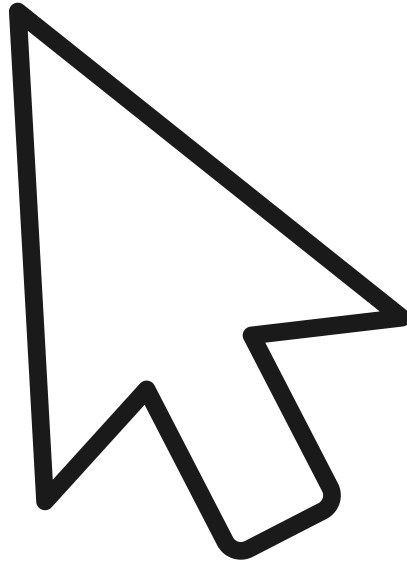
git cherry-pick --abort
abortar proceso

Traerse varios commits (estando en la rama *main*)

```
git cherry-pick <commit_hash1> <commit_hash2> ...
```

1. Flujo de trabajo
 2. Ramas
 3. **Conceptos avanzados de Git**
 4. Para ayudarte a entrenar con Git
 5. Y yo, ¿qué puedo hacer en mi proyecto?
 6. Ejercicio práctico: first *commit* dates
- 

3. Conceptos avanzados de Git



REFS (referencias)

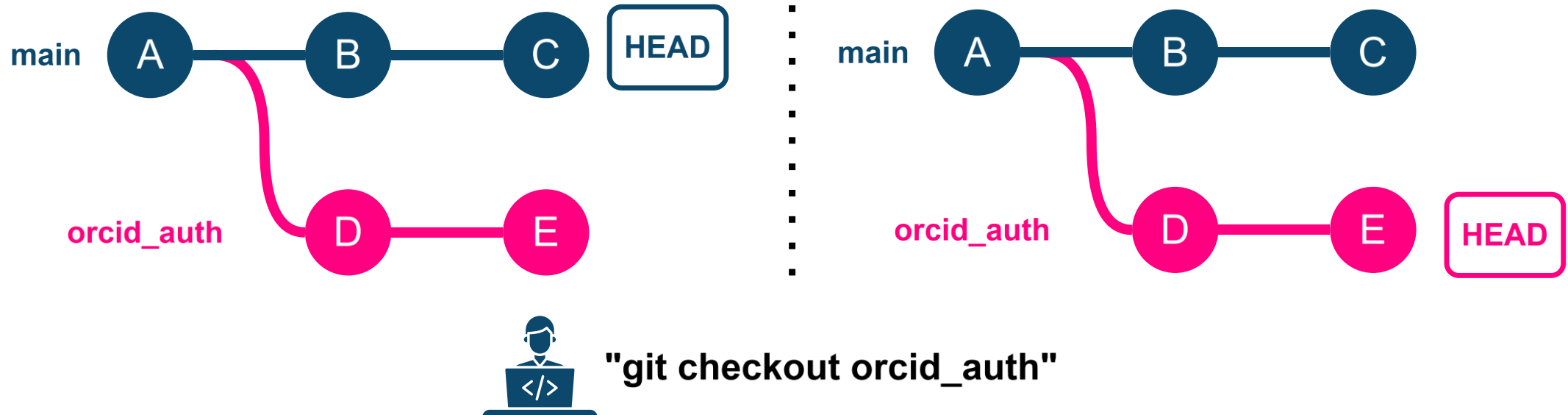
punteros o etiquetas que apuntan
a commits específicos

3. Conceptos avanzados de Git

Refs (referencias)

HEAD (*refs/heads/*)

Puntero que siempre señala último commit **en la rama local en la que estás trabajando actualmente**



3. Conceptos avanzados de Git

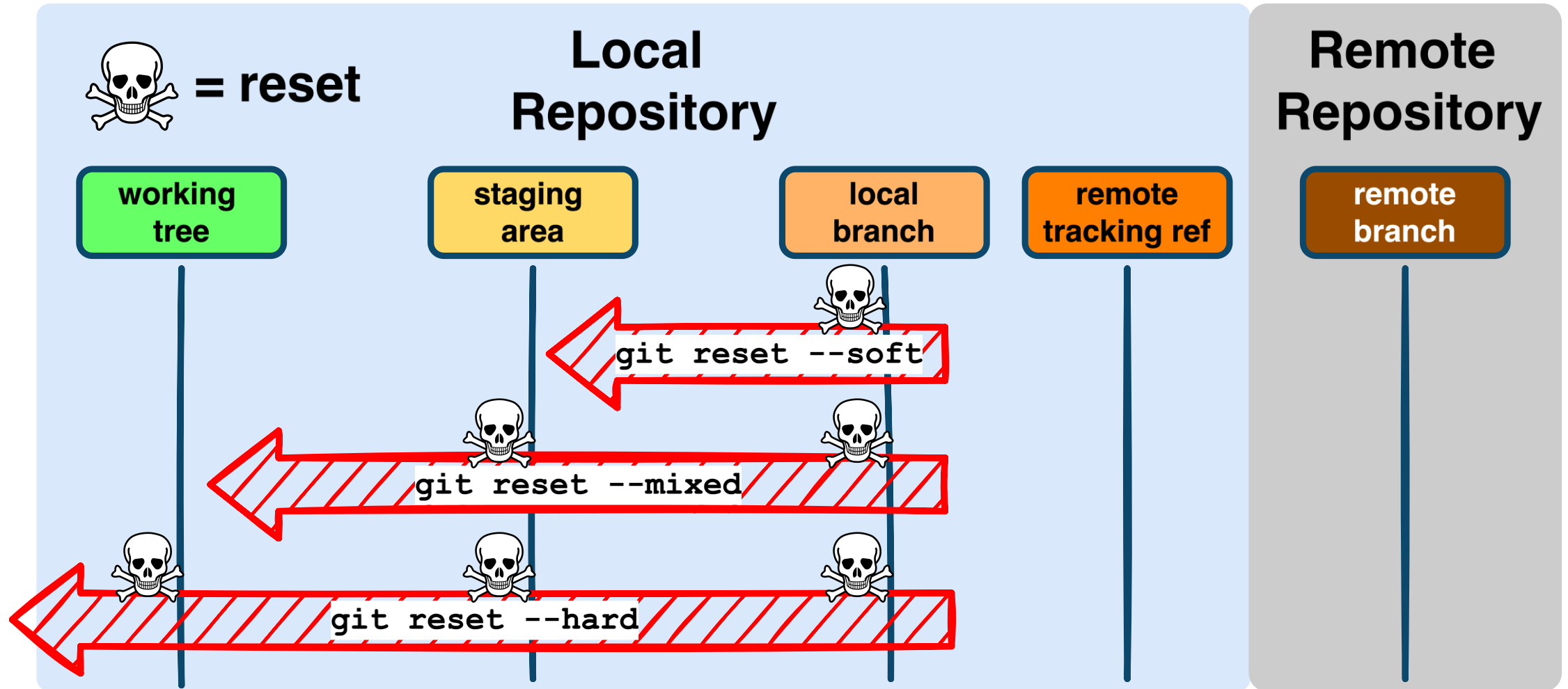


RESETEO

comando que deshace cambios moviendo la referencia de una rama a un commit anterior

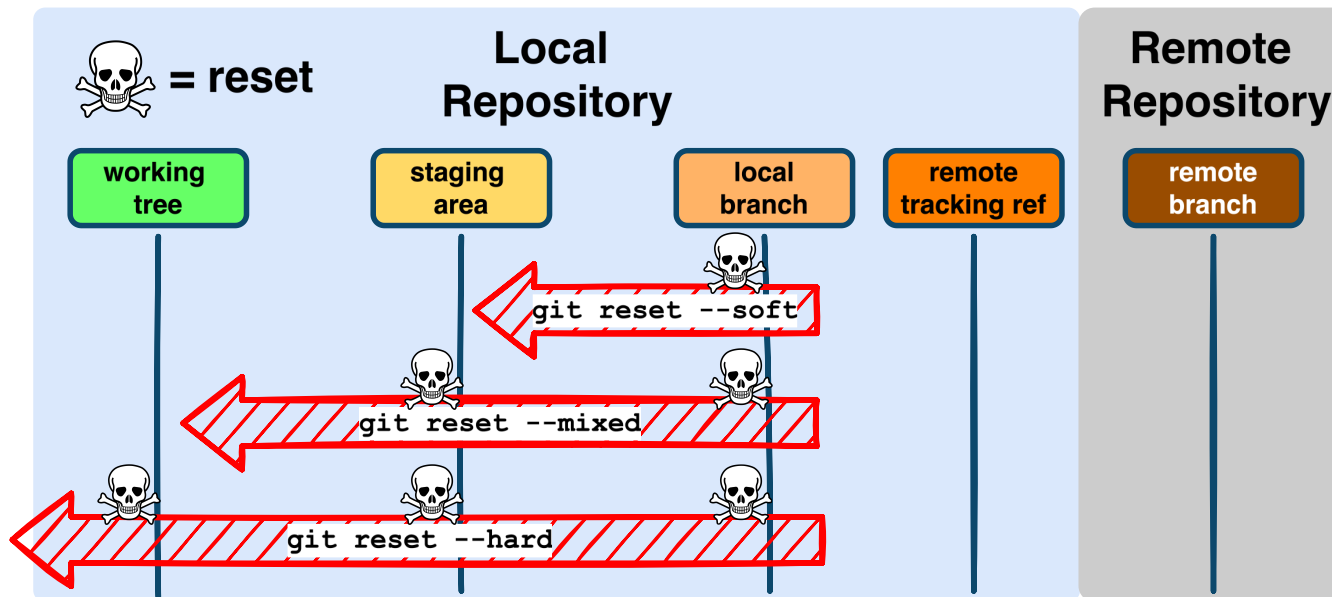
3. Conceptos avanzados de Git

Reseteo



3. Conceptos avanzados de Git

Reseteo



`git reset --hard HEAD`



`git reset --soft`

mueve la rama local al commit anterior (Staging y Working Tree sin cambios)

`git reset --mixed` (por defecto)

mueve la rama local al commit anterior y quita los cambios del Staging (Working Tree sin cambios)

`git reset --hard`

mueve la rama local al commit anterior y borra los cambios tanto del Staging como del Working Tree

3. Conceptos avanzados de Git

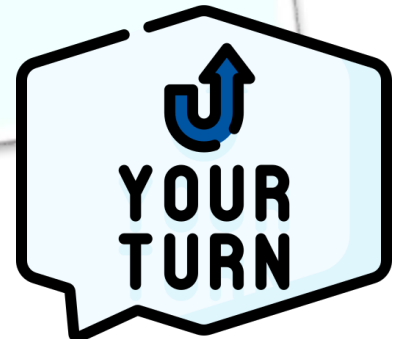


¡SOCORRO!

He hecho git reset –hard <hashX>
y lo he perdido todo

3. Conceptos avanzados de Git

Git Reflog:
Rescuing Lost Work
in Git's Timeline



3. Conceptos avanzados de Git



HOOKS

scripts útiles para automatizar tareas o validar cambios
antes o después de acciones específicas

3. Conceptos avanzados de Git

Hooks (.git/hooks)



Se ejecutan en el **lado del usuario** que realiza acciones locales, como *commits* o *merges*

pre-commit

se ejecuta antes de que se realice un *commit*, útil para revisar el código

post-commit

se ejecuta después de que se ha realizado un *commit*, útil para acciones de notificación o registro

Client-Side Hooks

3. Conceptos avanzados de Git

Hooks (.git/hooks)



Se ejecutan en el **lado del servidor** que aloja el repositorio cuando se realizan acciones como *push* o *pull*

pre-receive

se ejecuta antes de que el servidor acepte un *push*

post-receive

se ejecuta después de recibir un *push* en el servidor, útil para desencadenar integraciones continuas o notificaciones

Server-Side Hooks

3. Conceptos avanzados de Git



¡SOCORRO!

Es habitual que surjan problemas técnicos durante el uso de
Git

3. Conceptos avanzados de Git

Algunos comandos útiles

log/revert

Ver el histórico de commits

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

Deshacer un commit (crea uno nuevo deshaciendo el anterior)

```
git revert <commit_hash>
```

Deshacer un commit (igual, pero sin editar el mensaje)

```
git revert --no-edit <commit_hash>
```

3. Conceptos avanzados de Git

Algunos comandos útiles

diff

Ver cambios no confirmados

```
git diff
```

Ver cambios en el área de preparación

```
git diff --staged
```

Comparar commits

```
git diff <commit_hash1> <commit_hash2>
```

Comparar ramas

```
git diff <rama_1> <rama_2>
```

3. Conceptos avanzados de Git

Algunos comandos útiles

stashing

Guarda temporalmente los cambios no confirmados

```
git stash
```

Muestra una lista de todos los stashes almacenados

```
git stash list
```

Aplica cambio guardado (sin sacarlo del stash)

```
git stash apply
```

Aplica cambio guardado (eliminándolo del stash)

```
git stash pop
```



pila *stash*

1. Flujo de trabajo
 2. Ramas
 3. Conceptos avanzados de Git
 - 4. Para ayudarte a entrenar con Git**
 5. Y yo, ¿qué puedo hacer en mi proyecto?
 6. Ejercicio práctico: first *commit* dates
- 

4. Para ayudarte a entrenar con Git



ohmygit.org



learngitbranching.js.org



w3schools.com/git/

1. Flujo de trabajo
 2. Ramas
 3. Conceptos avanzados de Git
 4. Para ayudarte a entrenar con Git
 5. **Y yo, ¿qué puedo hacer en mi proyecto?**
 6. Ejercicio práctico: first *commit* dates
- 

5. Y yo, ¿qué puedo hacer en el proyecto?



**¡Diseña tu propia
gestión de código!**

Diseñar tu política de
gestión de **commits**

Diseñar tu política de
gestión de **issues**

Diseñar tu política de
gestión de **ramas**

1. Flujo de trabajo
 2. Ramas
 3. Conceptos avanzados de Git
 4. Para ayudarte a entrenar con Git
 5. Y yo, ¿qué puedo hacer en mi proyecto?
 6. **Ejercicio práctico: first *commit* dates**
- 

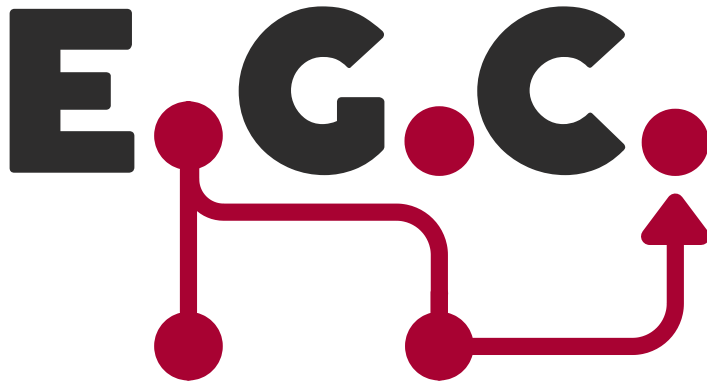
6. Ejercicio práctico: first *commit* dates



p3_usuario_A.pdf



p3_usuario_B.pdf



Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

¡Gracias!

*“La vida es como un repositorio
de Git, nunca sabes qué
conflicto te va a tocar.”*

- Forrest Dump