FABLAB.UV

TALLER de GIT HUB

versiona como los dioses

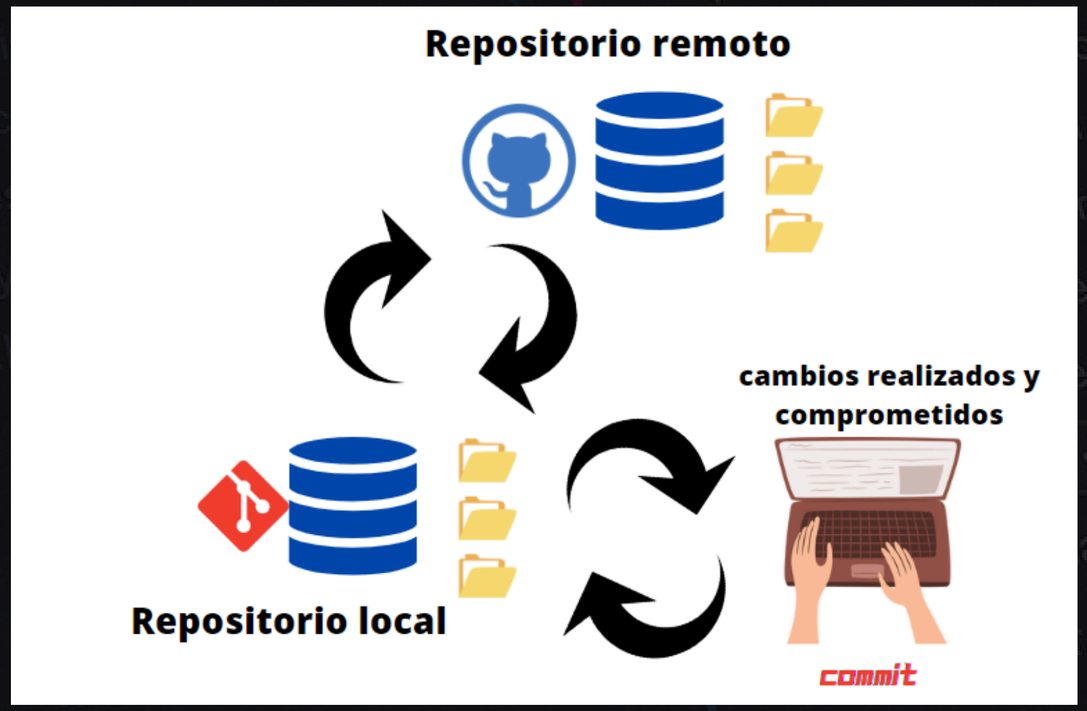
ALEJANDRO DIAZ CANTILLANO

EST. INGENIERÍA CIVIL INFORMATICA

Facultad de Ingeniería UV

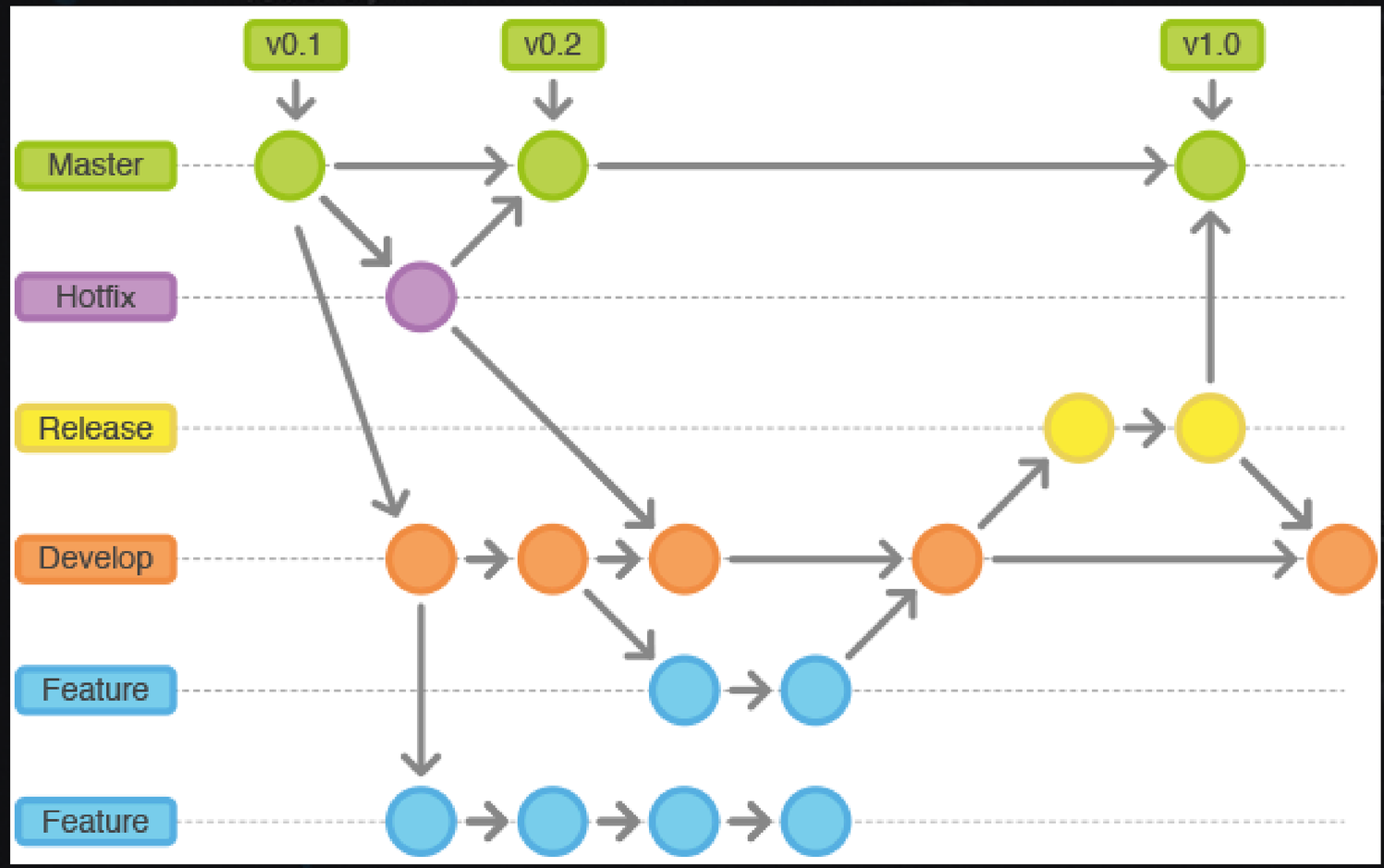
Universidad de Valparaíso CHILE

Github / Git

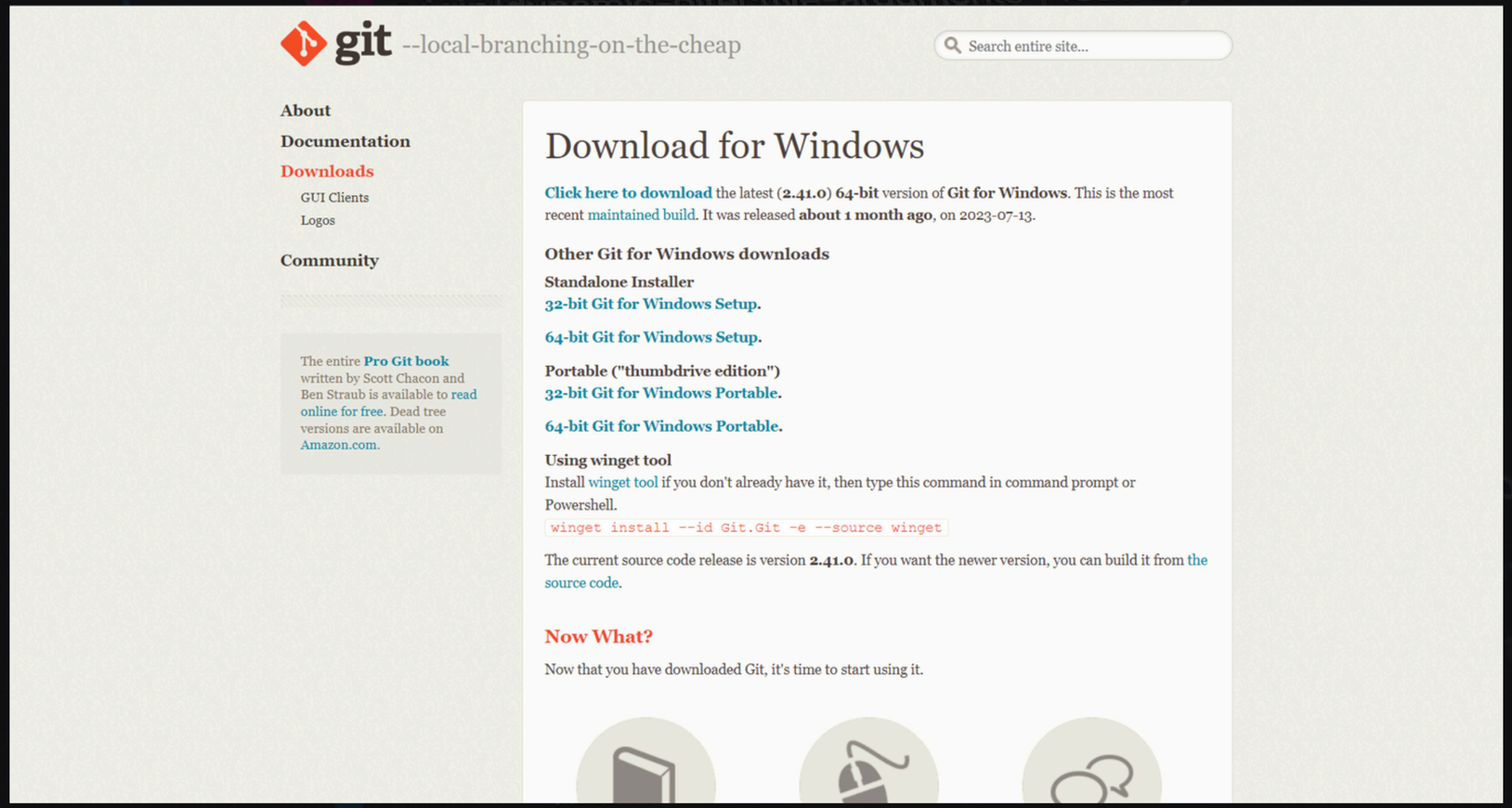


"Git controla versiones, y GitHub es donde se alojan y colaboran esos proyectos en la nube."

¿Que es Git?



Instalación y configuración básica



Configuraciones iniciales

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
```

Configurar tu usuario

```
git config --global init.defaultBranch main
```

Configurar nombre de la rama principal

Nuestro primer repositorio

```
# Inicializar repositorio
git init

# Agregar elementos al stash
git add .

# Realizar un commit
git commit -am "Nombre del commit"
```

Inicializa repositorio Git.

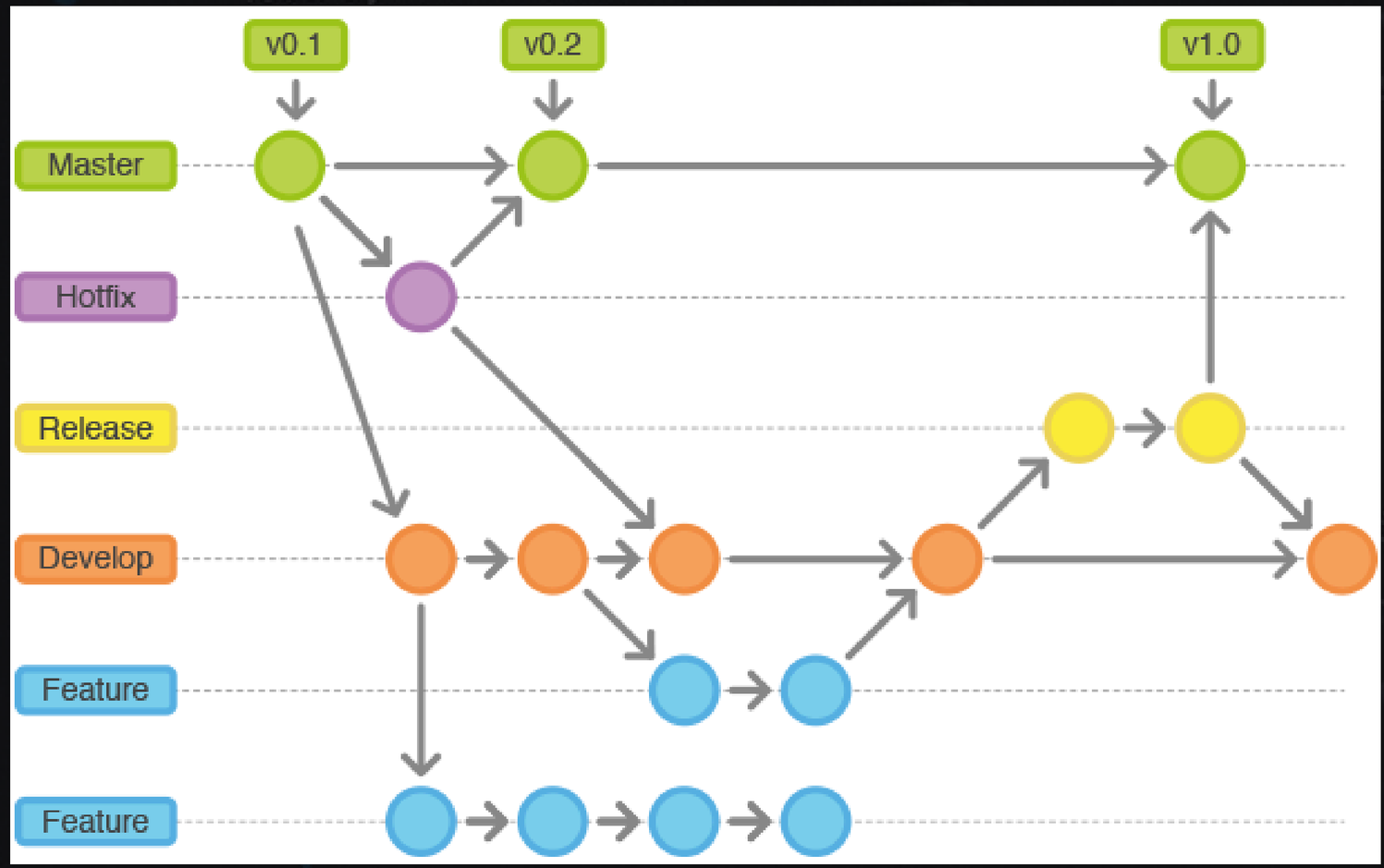
Añade todos los archivos actuales al área de preparación para el próximo commit.

Realiza commit y agrega cambios rastreados automáticamente.

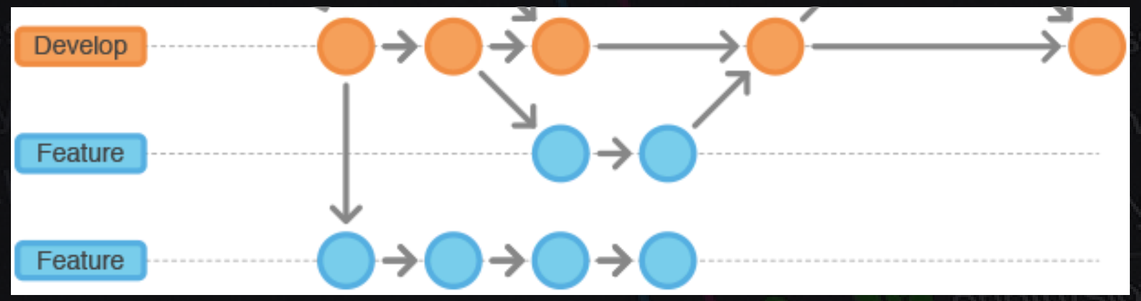
Comandos de utilidad

```
# Muestra el estado actual del repositorio.
git status

# Muestra el historial de commits.
git log

# Muestra el historial de movimientos de
HEAD, es decir, todas las acciones
realizadas en el repositorio.
git reflo
```

¿Que son las ramas?

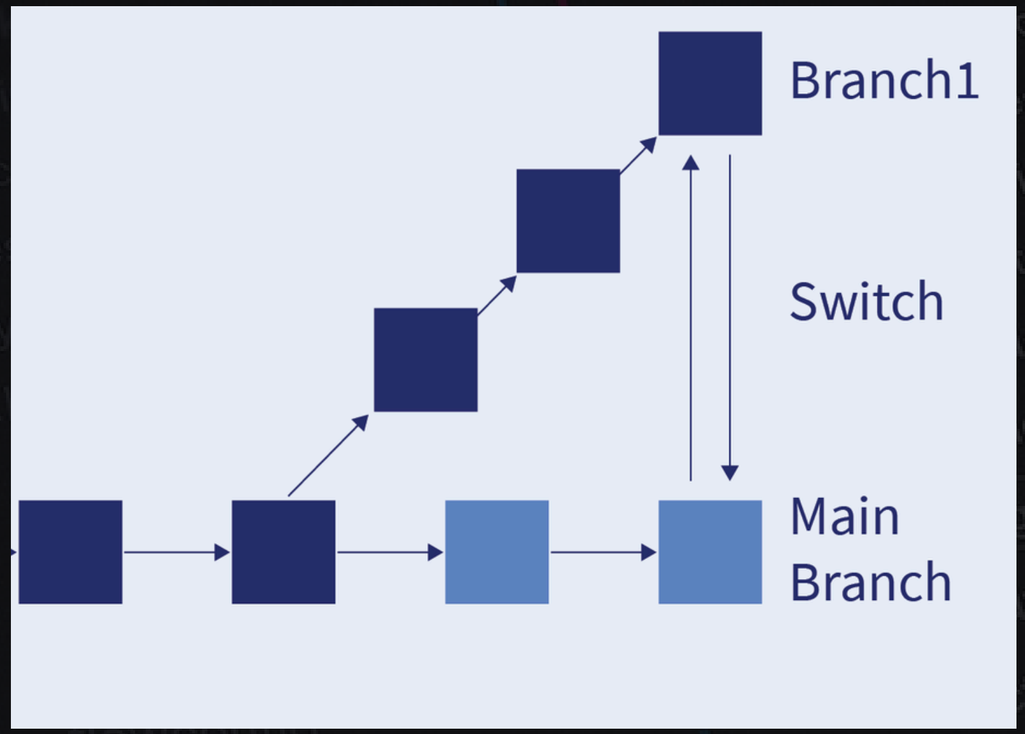


Gestion de ramas



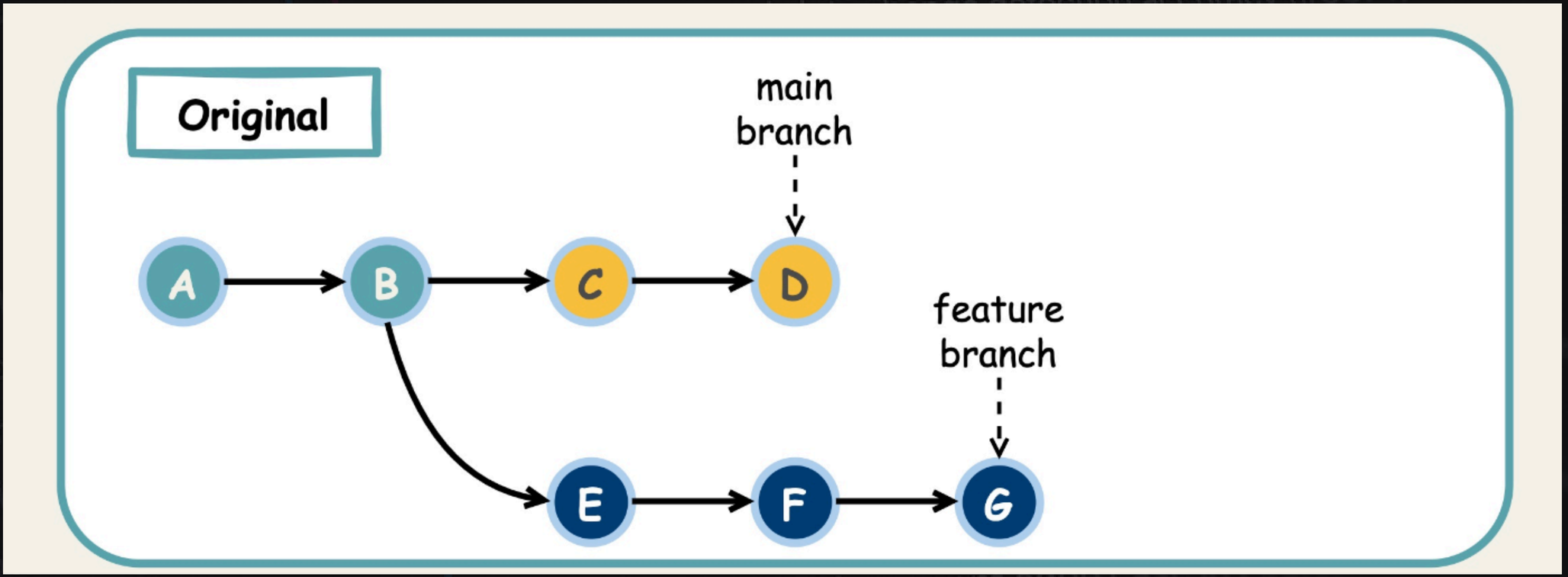

```
# Visualizar ramas locales
git branch

# Crear una rama
git branch

# EIiminar una rama ( -D para forma forzada)
git branch -d <nombre_rama>

# Renombrar rama actual
git branch -m <nuevo_nombre>
```

Gestion de ramas: Movimiento en ramas

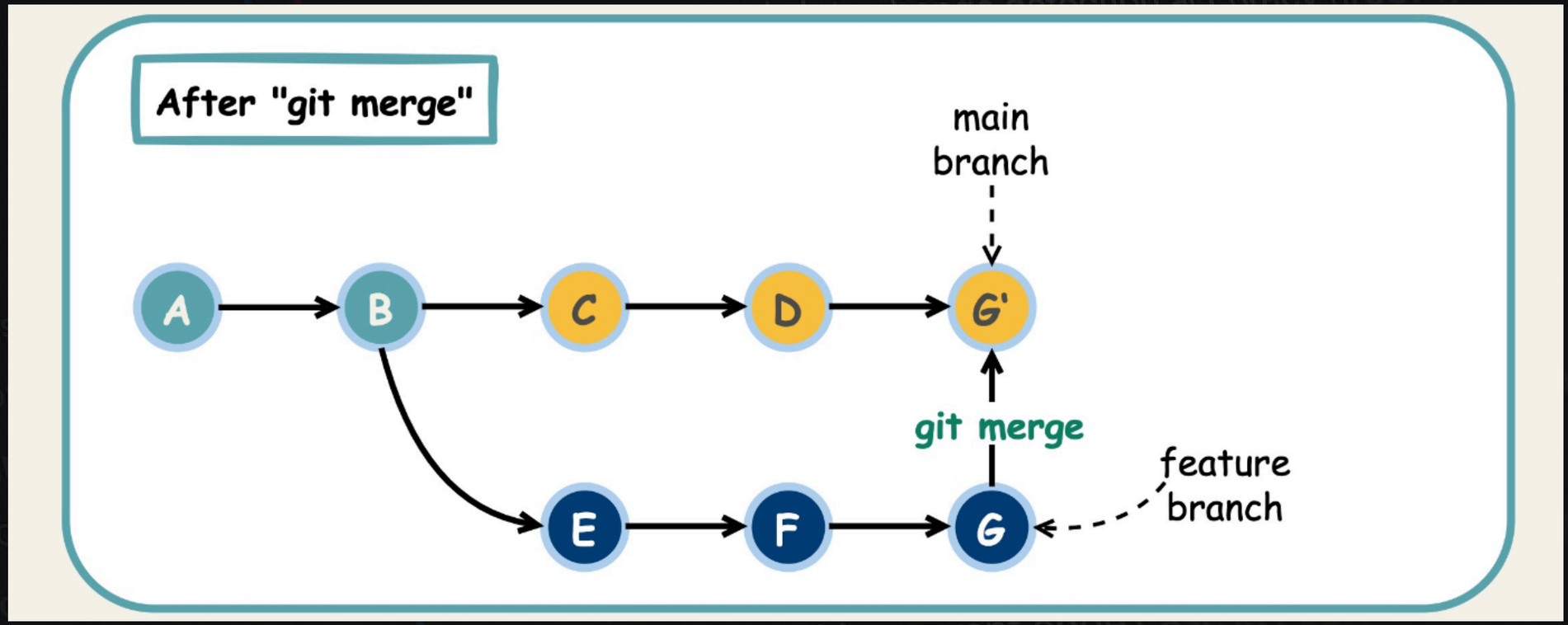



```
# Cambiar a otra rama
git switch <nombre_rama>

# Crea una nueva rama y cambia a ella al
mismo tiempo.
git switch -c <nombre_rama>
```

Gestion de ramas: Unión de ramas



Gestion de ramas: Unión de ramas - Merge

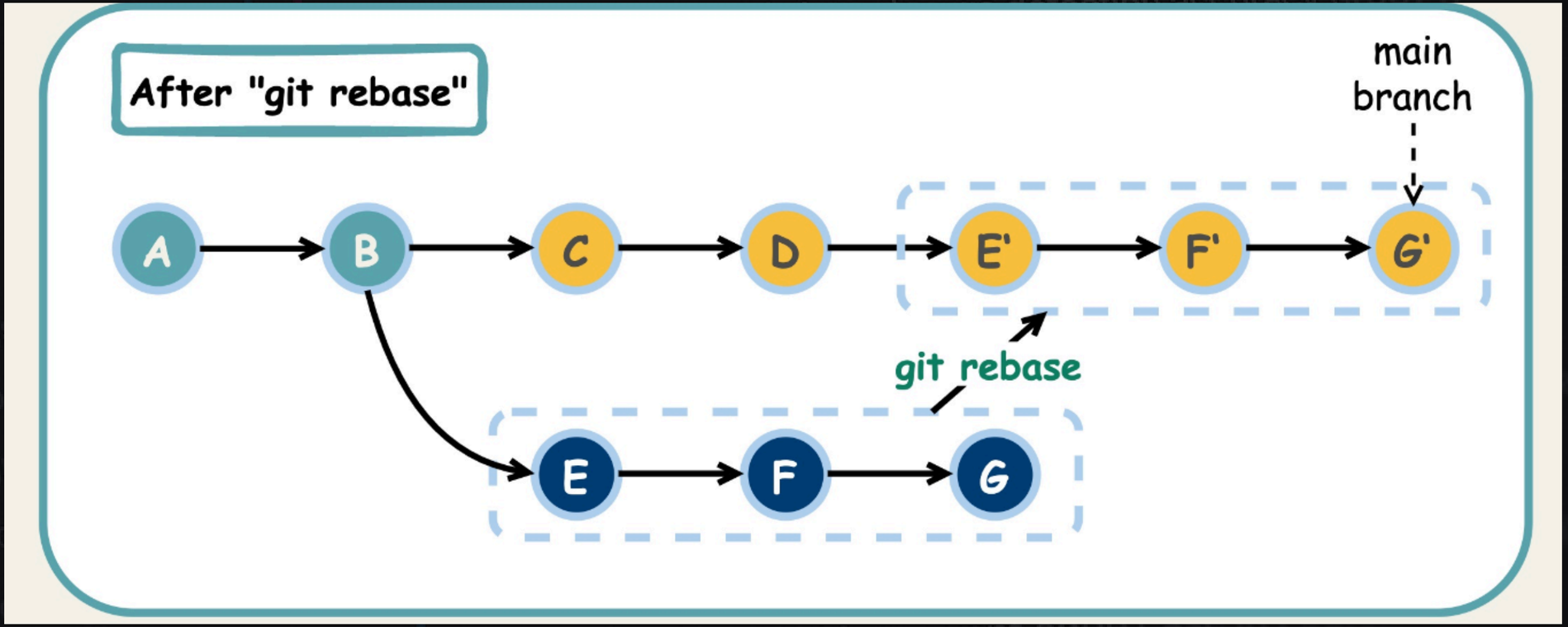


Gestion de ramas: Unión de ramas - Merge

```
# Cambios de la rama feature se llevan a
la rama main usando merge

# (1) Cambia a la rama main, donde
traeremos los cambios de feature.
git checkout main

# (2) Fusiona los cambios de feature en
la rama main.
git merge feature
```

git merge combina los cambios de dos ramas, creando un commit de fusión que une sus historiales. Mantiene un registro completo de cuándo y cómo se integraron, ideal para ramas principales como main o develop, conservando la evolución del proyecto.

Gestion de ramas: Unión de ramas - Rebase



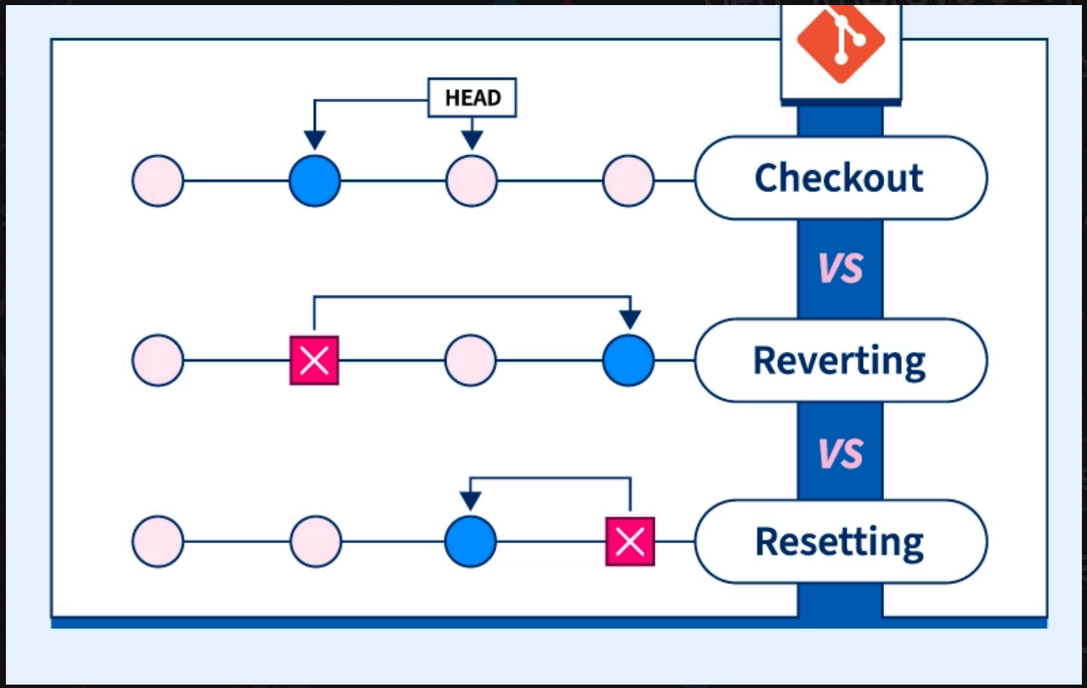
Gestion de ramas: Unión de ramas - Rebase

```
# Para llevar los cambios de feature a main usando rebase

# (1) Cambia a la rama feature donde se quieren llevar los cambios de main
git checkout feature

# (2) Reaplica los commits de feature sobre main
git rebase main
```

git rebase mueve los commits de una rama sobre otra, creando un historial lineal y limpio. Reaplica los cambios como si hubieran ocurrido después de la otra rama. Es útil para simplificar el historial antes de fusionar, pero debe usarse con cuidado para evitar conflictos, especialmente si la rama ya se ha compartido con otros.

Gestion de ramas: Unión de ramas - Conflictos

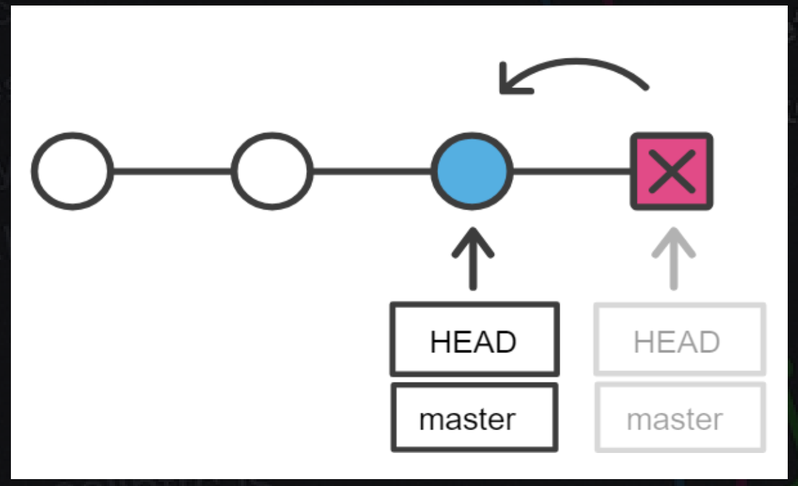
Revertir cambios



Git ofrece tres métodos para retroceder cambios: reset, revert y checkout.

- **Reset** elimina commits,
- **Revert** deshace cambios manteniendo el historial, y
- **Checkout** restaura archivos o cambia entre commits/ramos.

Revertir cambios: Reset

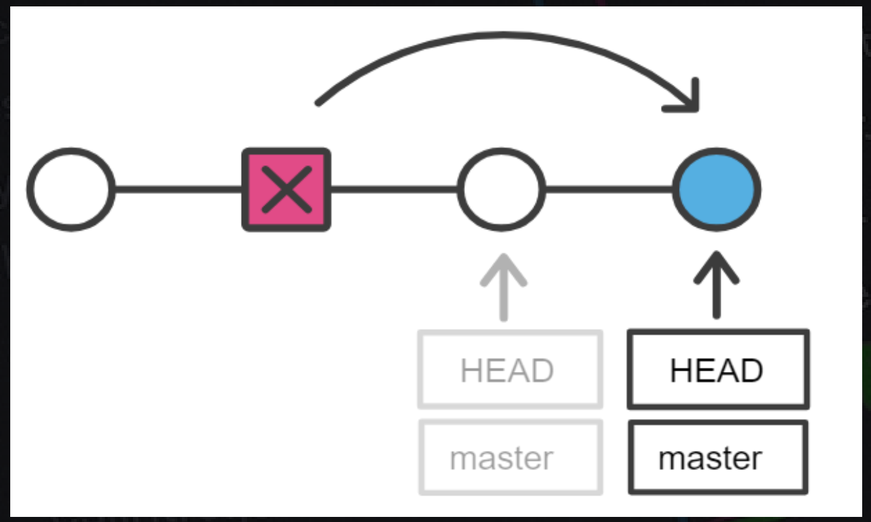



```
# Deshace el commit, pero mantiene todos los cambios en el área de preparación.
git reset --soft <commit_id>

# Elimina el commit y los cambios se mantienen en el directorio de trabajo, pero salen del área de preparación.
git reset --mixed <commit_id>

# Elimina el commit y borra todos los cambios del área de preparación y del directorio de trabajo.
git reset --hard <commit_id>
```

Revertir cambios: Reset - Ejemplos de uso

```
# Descartar todos los cambios locales y volver al último commit confirmado
git reset --hard

# Eliminar varios commits recientes, manteniendo los cambios en el directorio de trabajo
git reset --mixed HEAD~3

# Combinar varios commits en uno solo
git reset --soft HEAD~2

# Deshacer el último commit manteniendo los cambios listos para editar
git reset --soft HEAD~1
```

Revertir cambios: Revert




```
# Crea un nuevo commit que deshace los cambios realizados
por el commit especificado, No afecta otros commits ni los
cambios en archivos diferentes.
git revert <commit_id>

# Deshace los cambios realizados por el commit
especificado, pero no crea un commit inmediatamente.
git revert --no-commit <commit_id>
```

Revertir cambios: Revert - Ejemplos de uso

Ejemplo: Deshacer un commit específico

Supongamos que tienes los siguientes commits:

* Commit A: Modifica index.html
* Commit B: Modifica styles.css
* Commit C: Modifica index.html nuevamente

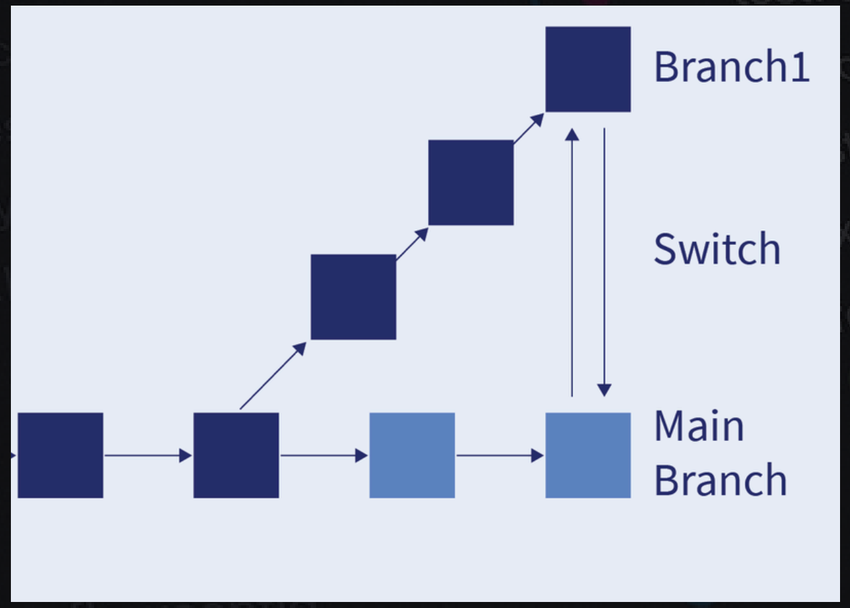
Si quieres deshacer los cambios realizados por el commit B que afectaron styles.css:

```
git revert B
```

Esto creará un nuevo commit que revierte exclusivamente los cambios realizados por el commit B en styles.css.

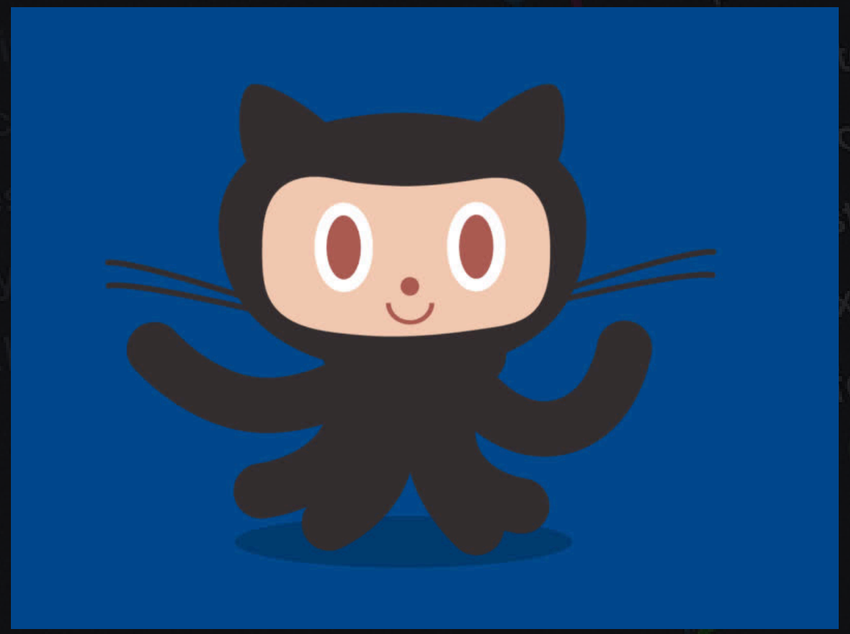
Los cambios en index.html realizados por los commits A y C no se verán afectados.

Revertir cambios: Checkout



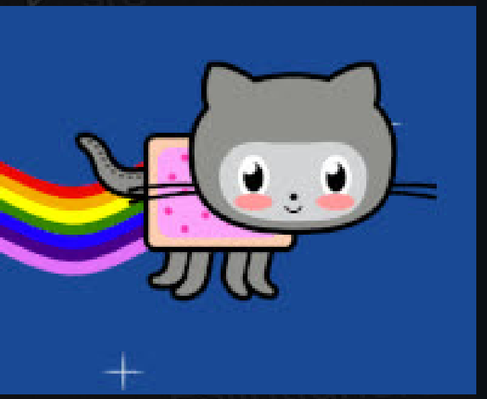

```
# Crea una nueva rama llamada <nueva_rama> basada en un commit específico
identificado por <commit_id> y cambia automáticamente a esa nueva rama.
git checkout -b <nueva_rama> <commit_id>

# Cambiarnos a un estado en especifico
git checkout <commit_id>
```

De lo local a lo remoto: ¿Qué es GitHub?



GitHub es una plataforma de desarrollo colaborativo que aloja repositorios Git, permitiendo a desarrolladores gestionar, compartir, versionar y colaborar en proyectos de software de manera eficiente y organizada.

De lo local a lo remoto: Creación de un repositorio en GitHub



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Owner * ixyz022 /

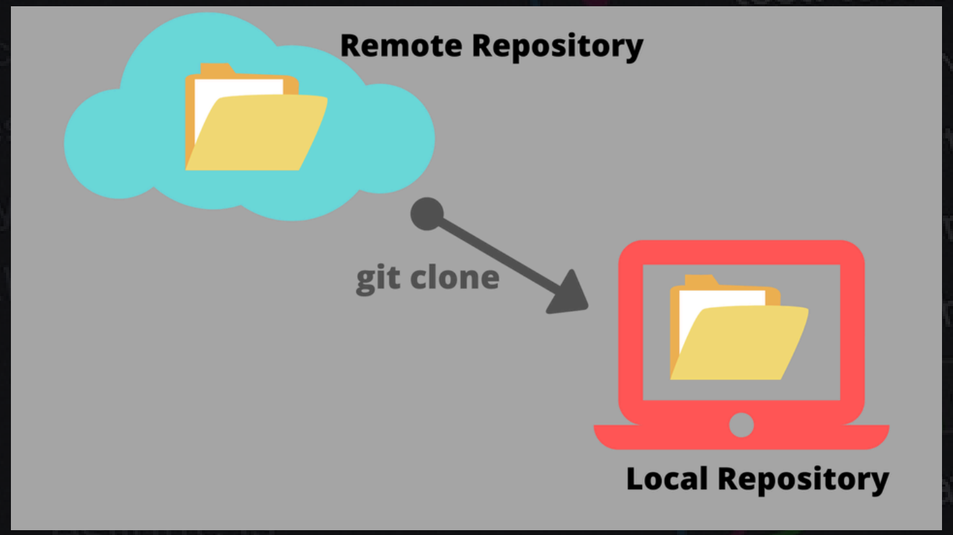
Repository name *

Great repository names are short and memorable. Need inspiration? How about refactored-tribble ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

De lo local a lo remoto: Clonar un repositorio remoto

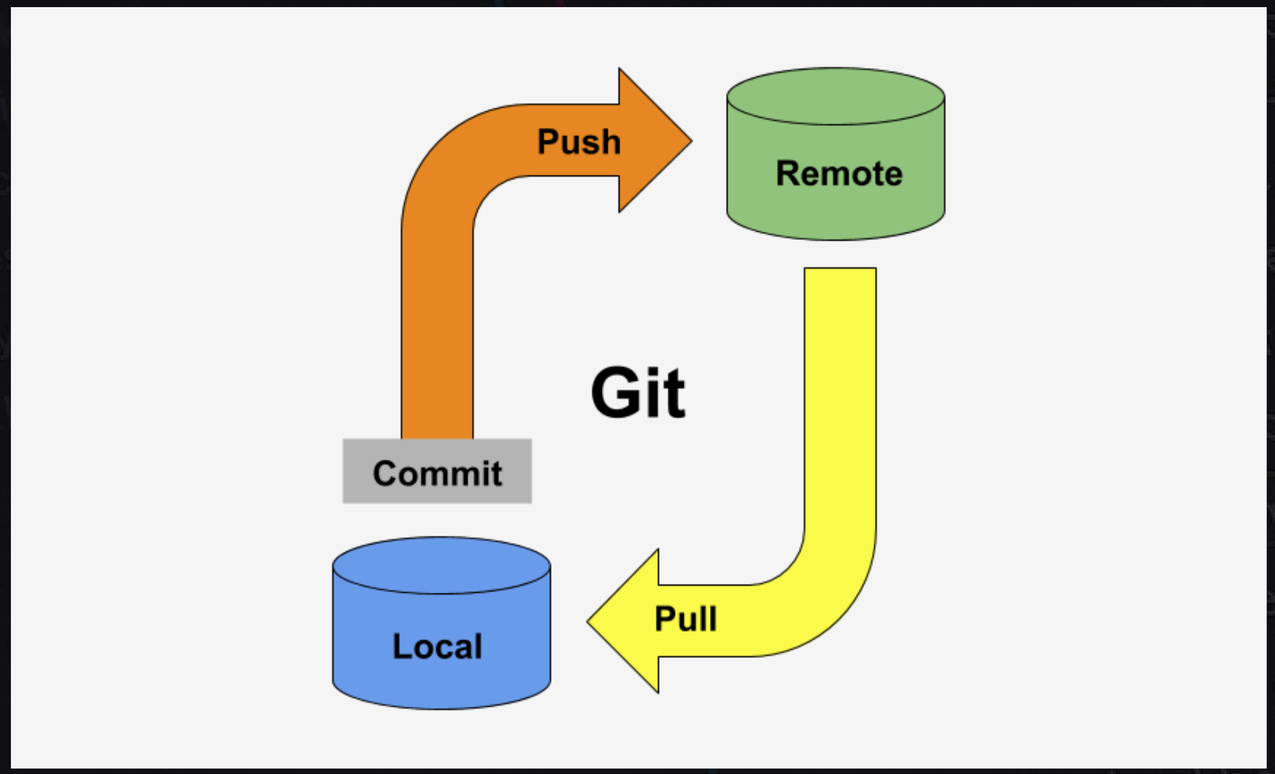



```
# Descarga el repositorio remoto de GitHub a tu máquina local, creando una carpeta llamada proyecto con todos los archivos y el historial.
git clone <url_del_repositorio>
```

Gestion de ramas: Ramas remotas

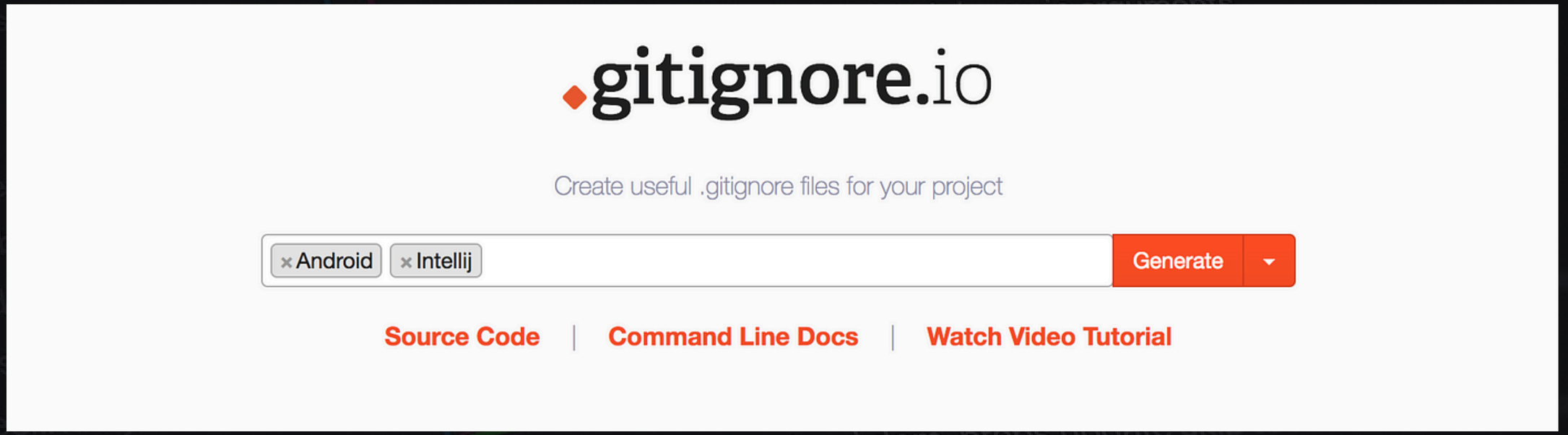
```
# Descarga actualizaciones sin aplicarlas localmente.
git fetch

# Visualizar ramas locales y remotas
git branch -a

# Descarga la rama remota, luego simplemente nos cambiamos
git fetch origin <nombre_rama_remota>

# Crea y cambia a nueva rama en base a una rama remota
git switch -c <nombre_rama_local> origin/<nombre_rama_remota>
```

De lo local a lo remoto: Sincronizar cambios

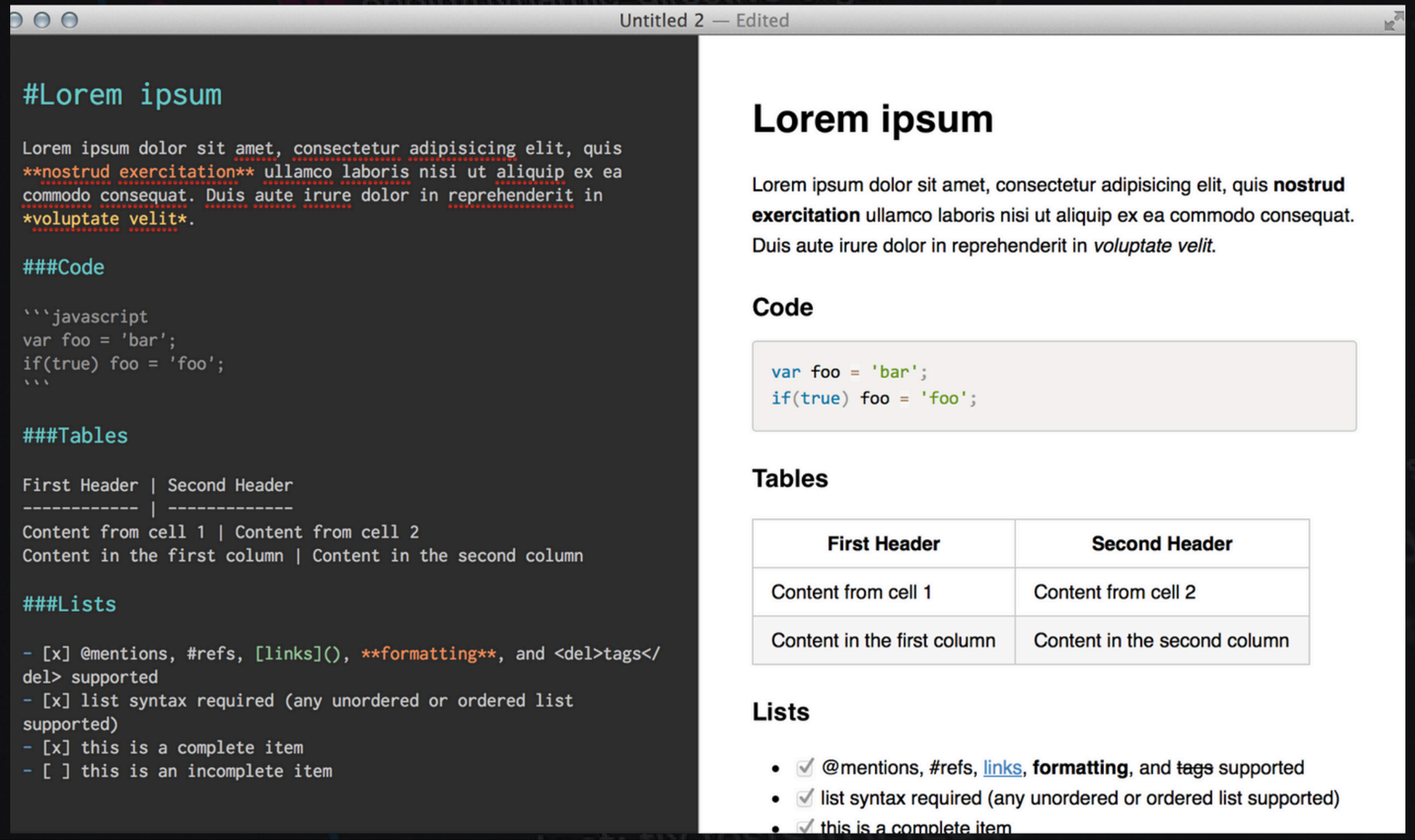



```
# Envía cambios locales al
repositorio remoto.
git push

# Trae y fusiona los cambios del
repositorio remoto a tu rama local.
git pull
```

Uso de .gitignore para ignorar archivos no deseados.



Documentación de tu repositorio



Dominando el Control de Versiones

FABLAB.UV

Universidad de Valparaíso CHILE

FABLAB.UV

TALLER de GIT HUB

versiona como los dioses

ALEJANDRO DIAZ CANTILLANO

EST. INGENIERÍA CIVIL INFORMATICA

Facultad de Ingeniería UV

Universidad de Valparaíso CHILE

Dominando el Control de Versiones

FABLAB.UV

Universidad de Valparaíso CHILE