JAVAACADEMY

GIT

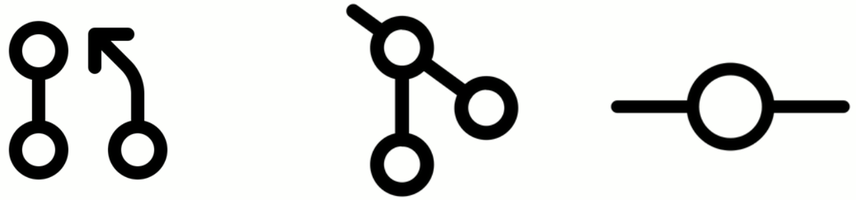
BASICS

Que es GIT?

Git es un sistema de control de versiones distribuido que permite a los desarrolladores rastrear y gestionar los cambios en el código fuente a lo largo del tiempo. Es como una "máquina del tiempo" para el código, permitiendo volver a versiones anteriores, colaborar con otros y mantener un historial de cambios.

Diferencias Entre GIT Y GITHUB

Git es la herramienta que utilizas para gestionar el código localmente, mientras que GitHub es una plataforma en la nube que permite almacenar y compartir esos repositorios Git de manera remota. Piensa en GitHub como una red social para desarrolladores, donde puedes colaborar en proyectos con otros usuarios.



Configuracion inicial

Configurar el nombre de usuario y correo electrónico:

Antes de empezar a usar Git, debes configurarlo para que sepa quién eres. Esto se hace estableciendo tu nombre de usuario y correo electrónico, los cuales se usarán para identificarte en cada commit.
Esto se logra con los sig comandos:

git config --global user.name "Tu Nombre"
git config --global user.email "tuemail@example.com"

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (master)
$ git config --global user.email "mkermz@gmail.com"

HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (master)
$ git config --global user.name "Miguel Angel Ramirez"
```

COMANDOS COMUNES

GIT INIT

convierte cualquier carpeta en un repositorio de Git, es decir, empieza a rastrear los cambios en esa carpeta. Este es el primer paso para comenzar a usar Git en un proyecto.

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c
$ cd AcademyJavaRepository

HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository
$ git init
Initialized empty Git repository in C:/AcademyJavaRepository/.git/
```

GIT ADD

agrega todos los archivos y cambios en la carpeta actual al "staging area" o área de preparación, que es como una zona de espera antes de confirmar (commit) los cambios.

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (main)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the
 next time Git touches it
```

GIT COMMIT

guarda un snapshot o captura de los archivos en la historia del proyecto. Es un paso crucial para versionar tu código.

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (main)
$ git commit -m " First commit "
[main (root-commit) 5411f63]  First commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

COMANDOS COMUNES

GIT BRANCH

crea una nueva rama en tu proyecto. Las ramas permiten trabajar en diferentes características o fixes sin afectar el código principal

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (main)
$ git branch TESTBRANCH

HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (main)
$ git checkout TESTBRANCH
Switched to branch 'TESTBRANCH'
```

GIT CHECKOUT

te permite cambiar a otra rama. Esto es útil para cambiar entre diferentes flujos de trabajo.

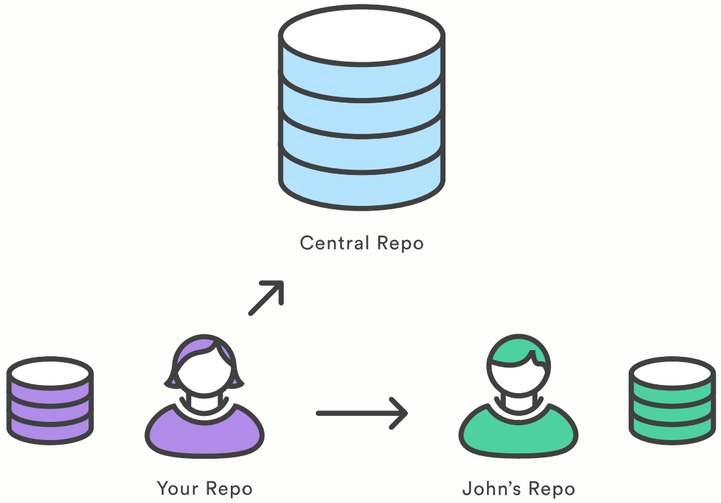
GIT MERGE

combina los cambios de una rama en otra, generalmente la rama principal. Es como unir el trabajo realizado en diferentes caminos.

```
HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (TESTBRANCH)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

HP@DESKTOP-NRQ8VIQ MINGW64 /c/AcademyJavaRepository (main)
$ git merge TESTBRANCH
Updating 5411f63..8ea8d10
Fast-forward
 TESTBRANCH - copia (2).txt | 1 +
 TESTBRANCH - copia (3).txt | 1 +
 TESTBRANCH - copia.txt     | 1 +
 TESTBRANCH.txt             | 1 +
 4 files changed, 4 insertions(+)
 create mode 100644 TESTBRANCH - copia (2).txt
 create mode 100644 TESTBRANCH - copia (3).txt
 create mode 100644 TESTBRANCH - copia.txt
 create mode 100644 TESTBRANCH.txt
```

REPOSITORIO

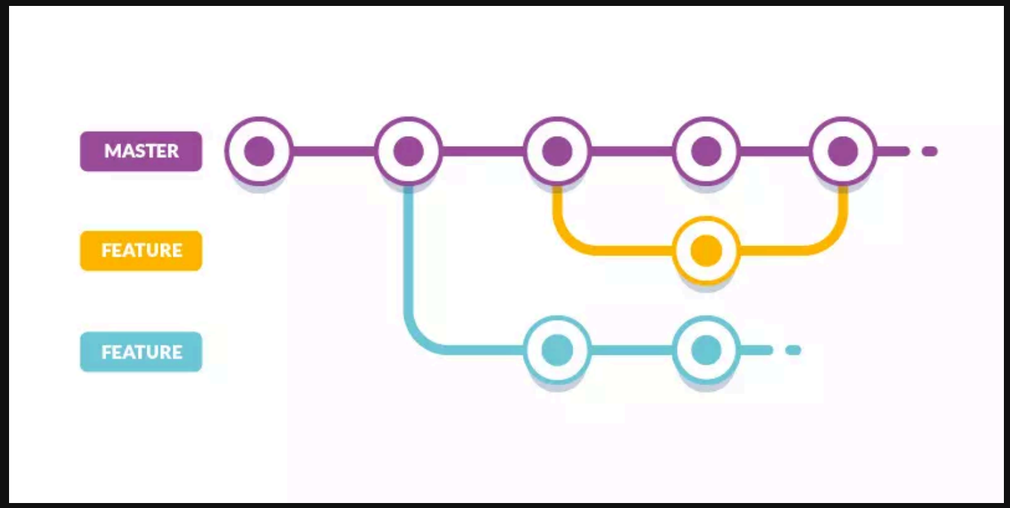


Un repositorio es básicamente un "almacén" para tu proyecto de código. Contiene todos los archivos del proyecto, además de un historial de todos los cambios que se han realizado en esos archivos. Cada vez que haces un commit, estás guardando un nuevo "punto en el tiempo" en el repositorio.

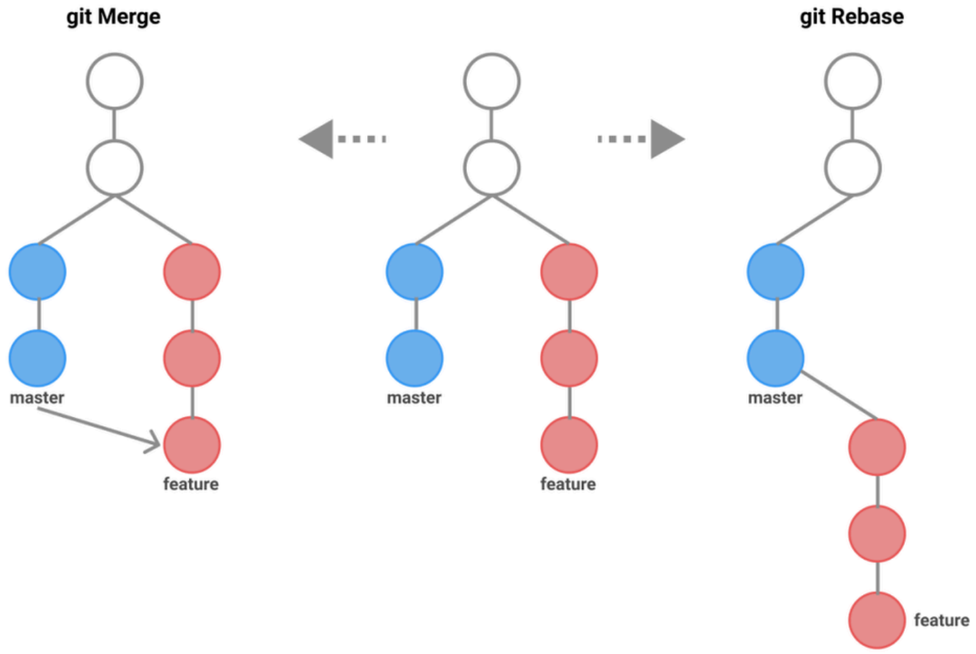
Piensa en un repositorio como una biblioteca digital para tu proyecto, donde cada commit es un libro que contiene un capítulo de la historia del desarrollo de tu software.

BRANCHES

Una rama es una "línea paralela de desarrollo" dentro de tu proyecto. Imagina que estás trabajando en una nueva característica o solución de un problema. En lugar de hacer cambios directamente en la línea principal del proyecto (que suele llamarse main), puedes crear una nueva rama y trabajar en esa característica de manera aislada. Esto te permite experimentar y hacer cambios sin afectar la rama principal.



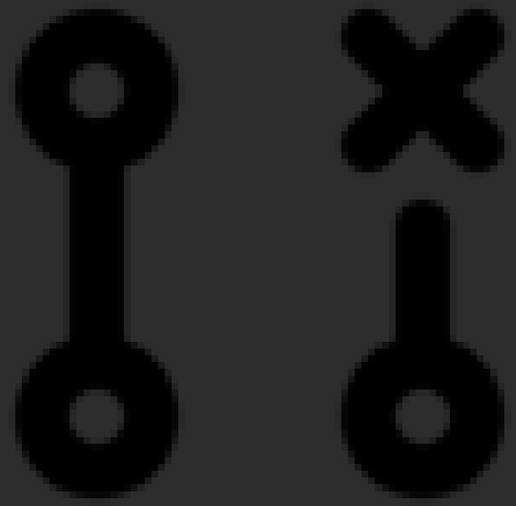
MERGE



Un repositorio es básicamente un "almacén" para tu proyecto de código. Contiene todos los archivos del proyecto, además de un historial de todos los cambios que se han realizado en esos archivos. Cada vez que haces un commit, estás guardando un nuevo "punto en el tiempo" en el repositorio.

Piensa en un repositorio como una biblioteca digital para tu proyecto, donde cada commit es un libro que contiene un capítulo de la historia del desarrollo de tu software.

CONFLICTOS



Un conflicto en Git ocurre cuando intentas combinar dos ramas (o commits) que tienen cambios contradictorios en el mismo lugar de un archivo. Git no puede decidir automáticamente qué versión del código conservar y requiere intervención manual para resolver el problema.

Ejemplo de Conflicto:

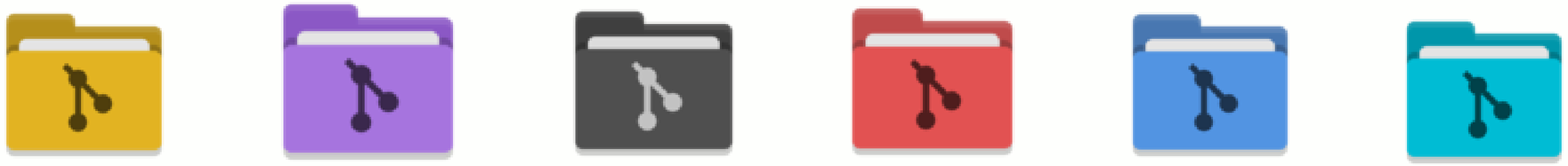
Imagina que dos desarrolladores están trabajando en el mismo proyecto:

- Desarrollador A modifica la línea 10 del archivo main.java en la rama feature-A.
- Desarrollador B también modifica la línea 10 del mismo archivo en la rama feature-B.

Cuando intentas hacer un merge de feature-B en main, Git detecta que hay dos versiones diferentes de la línea 10 y no sabe cuál debería conservar. Esto genera un conflicto.

JAVAACADEMY

CONCLUSION



Git es una herramienta esencial en el desarrollo de software moderno, permitiendo a los equipos trabajar de manera eficiente y colaborativa en proyectos de cualquier tamaño. Su capacidad para gestionar versiones, rastrear cambios, y facilitar la integración de múltiples líneas de desarrollo a través de ramas y fusiones, lo hace indispensable para cualquier desarrollador.

Al aprender a usar Git, no solo adquieres una habilidad técnica clave, sino que también adoptas prácticas que promueven la organización, la colaboración y la eficiencia en el desarrollo de software. Resolver conflictos, manejar repositorios remotos, y trabajar con ramas se vuelven procesos naturales que te permiten concentrarte en la creatividad y la innovación en tu trabajo, con la seguridad de que siempre puedes revertir cambios o experimentar sin riesgo.