



Universidad de
CUNDINAMARCA
FACULTAD DE INGENIERÍA - EXTENSIÓN CHÍA

Línea de profundización III

Aactividad – Comandos de Git

Est: Julian David Herrera Romero

Prof. Willian Alexander Matallana

Febrero - 2025

Contenido

Introducción	2
Objetivos	2
Practica	3
Conclusiones	9
Referencias	10

Introducción

¿Qué tan importante en el trabajo colaborativo en la industria del desarrollo?, muy importante, ya que posibilita la integración organizada de los aportes individuales en un proyecto común, en la presente práctica a continuación vamos a trabajar la herramienta para versionamiento de código llamada GIT, empezaremos a trabajar desde las configuraciones básicas de la herramienta, mas adelante, nos encaminaremos por los comandos principales y mas usados a la hora de colaborar con un equipo de desarrollo y así mismo, aprenderemos las buenas practicas para así poder resolver conflictos, generar merge y pull request, con el fin de fortalecer las competencias necesarias para desenvolverse de manera efectiva en un entorno laboral real.

Objetivos

1. Familiarizarse con términos como repositorio, commit, branch, merge y pull request.
2. Crear, clonar, inicializar y administrar repositorios locales y remotos.
3. Realizar commits para registrar cambios en el proyecto con mensajes claros y descriptivos.
4. Utilizar ramas y pull requests para trabajar en paralelo y fusionar cambios de manera efectiva.
5. : Identificar y resolver conflictos que puedan surgir al combinar cambios de diferentes ramas.

Practica

Git config –list: Se utiliza para mostrar la lista con las configuraciones de git

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
:...skipping...
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
```

Git config –global user.email

Sirve para establecer el correo electrónico a nivel global de Git

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git config --global user.email "davher.ext@let tiempo.com"
```

Git config –global user.name

Sirve para establecer el nombre de usuario a nivel global de Git

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git config --global user.name "davher.ext"
```

Git config –global –unset user.email

Sirve para desasociar el correo electrónico a nivel global de git

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git config --global --unset user.email
```

Git config –global –unset user.name

Sirve para desasociar el nombre a nivel global de git

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git config --global --unset user.name
```

Git init

Sirve para inicializar el proyecto en el repositorio de git y crea el archivo .git

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git init
Initialized empty Git repository in C:/Users/User/Desktop/Curso Kotlin/practicaGit/.git/
```

echo "# practicaGit"

Crear un archivo README.md con una línea de título practicaGit

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> echo "# practicaGit" >> README.md
```

git add README.md

Empaqueta el archivo README.md

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git add README.md
```

git commit -m "first commit"

Toma el paquete y realiza el commit y lo asigna con el nombre que se le establece

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git commit -m "first commit"
```

git branch -M main

renombra la rama actual como la main

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git branch -M main
```

git remote add origin <https://github.com/elyulian/practicaGit.git>

Sirve para conectar tu repositorio local con un repositorio remoto en Git

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git remote add origin https://github.com/elyulian/practicaGit.git
```

git push -u origin main

Sirve para enviar todos los commits que están en la rama local al repositorio remoto

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 120.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/elyulian/practicaGit.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Git status

Se utiliza para ver el estado actual del repositorio (Que archivos están empaquetados y cuales no)

```
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    .idea/
    pom.xml
    src/
```

Git add .

Sirve para empaquetar todos los archivos

```
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git add .
```

Git commit -m "MensajeCommit"

Crea el commit que empaqueto y lo prepara para subirlo

```
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git commit -m "Hola Mundo"
[main 06b8a85] Hola Mundo
7 files changed, 99 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/encodings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/vcs.xml
create mode 100644 pom.xml
create mode 100644 src/main/java/org/example/Main.java
```

Git push origin nombreRama

Sirve para subir el commit y los cambios en la rama que se le especifica

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git push origin main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (15/15), 2.16 KiB | 245.00 KiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/elyulian/practica6it.git
   c8b4e4e..06b8a85  main -> main
PS C:\Users\User\Desktop\Curso Kotlin\practica6it>
```

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git push origin DEV
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (8/8), 542 bytes | 542.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'DEV' on GitHub by visiting:
remote:   https://github.com/elyulian/practica6it/pull/new/DEV
remote:
```

Git pull origin

Sirve para bajar todos los cambios asociados

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> Git pull origin
Already up to date.
```

Git switch -c rama

Sirve para crear una rama nueva y cambiarse a ella

```
PS C:\Users\User\Desktop\Curso Kotlin\practica6it> git switch -c "DEV"
Switched to a new branch 'DEV'
```

Git branch -D rama

Sirve para borrar la rama de manera local

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git branch -d DEV
Deleted branch DEV (was e8b77da).
```

git push origin --delete DEV

Sirve para borrar la rama en el repositorio de manera remota

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git push origin --delete DEV
To https://github.com/elyulian/practicaGit.git
- [deleted]          DEV
```

Git fetch --all

Sirve para actualizar toda la información de las ramas

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git fetch --all
Fetching origin
```

Git branch -r

Sirve para ver todas las ramas que hay creadas en el proyecto

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git branch -r
origin/main
```

Git log

Sirve para mostrar el historial de commits del repositorio actual

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git log
commit e8b77da5d952988b4d5246cf4767dd98ea9f27c0 (HEAD -> DEV, origin/DEV)
Author: davher.ext <davher.ext@let tiempo.com>
Date: Tue Feb 18 16:06:27 2025 -0500

    Bucle

commit 06b8a8571ca874b506183d69dfbe38e2c6af9f34 (origin/main, main)
Author: davher.ext <davher.ext@let tiempo.com>
Date: Tue Feb 18 16:01:33 2025 -0500

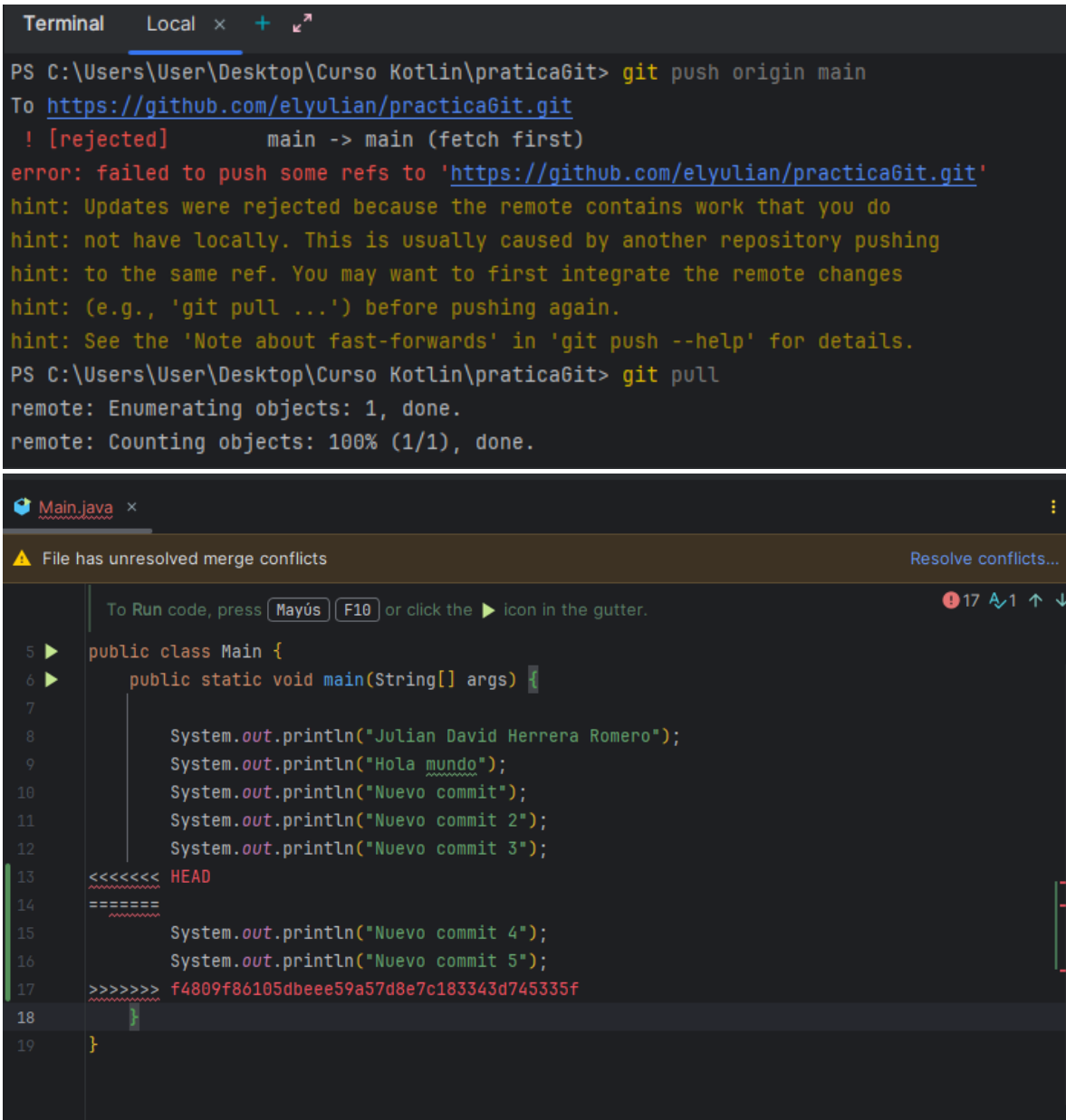
    Hola Mundo
```

Git merge rama

Sirve para realizar un merge de la rama actual a la rama que digites

```
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git merge DEV
Updating 06b8a85..e8b77da
Fast-forward
 src/main/java/org/example/Main.java | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit>
```

Resolver conflictos



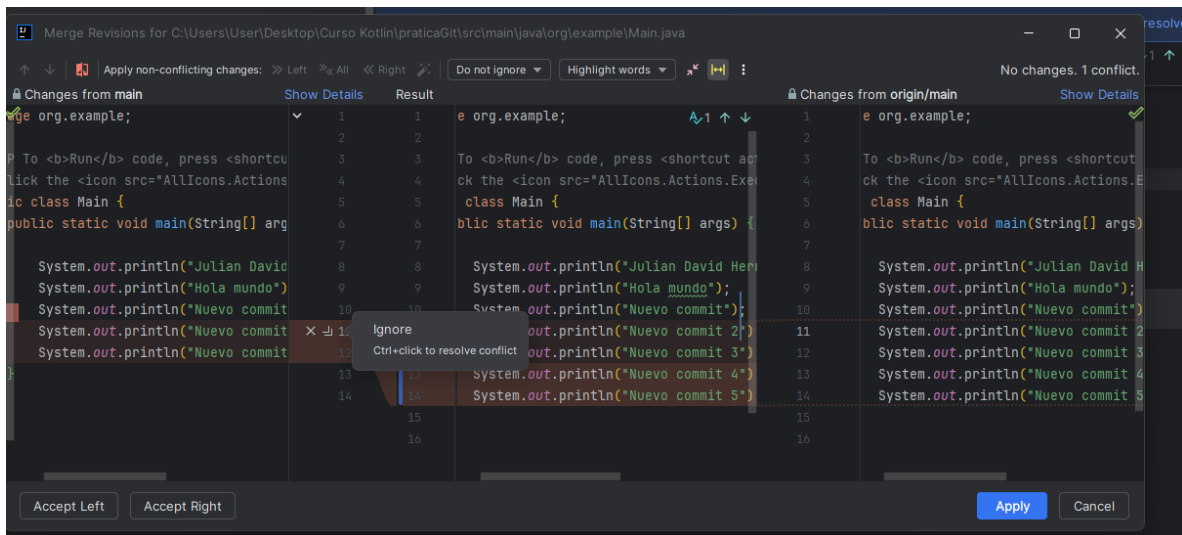
The image shows a terminal window and an IDE window. The terminal window displays the output of a `git push` command, which is rejected because the remote contains work that is not locally available. It then shows the output of a `git pull` command, which successfully fetches the remote changes. The IDE window shows the `Main.java` file with a merge conflict. The conflict is between the local `HEAD` and the remote `f4809f86105dbeee59a57d8e7c183343d745335f`. The conflict is resolved by keeping the local changes.

```
Terminal Local x + ↗
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git push origin main
To https://github.com/elyulian/practicaGit.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/elyulian/practicaGit.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
PS C:\Users\User\Desktop\Curso Kotlin\practicaGit> git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.

Main.java x
⚠ File has unresolved merge conflicts Resolve conflicts...

To Run code, press Mayús F10 or click the ▶ icon in the gutter. 17 ↶ ↷ ↵ ↴

5 ▶ public class Main {
6 ▶     public static void main(String[] args) {
7
8         System.out.println("Julian David Herrera Romero");
9         System.out.println("Hola mundo");
10        System.out.println("Nuevo commit");
11        System.out.println("Nuevo commit 2");
12        System.out.println("Nuevo commit 3");
13        <<<<<< HEAD
14        =====
15        System.out.println("Nuevo commit 4");
16        System.out.println("Nuevo commit 5");
17        >>>>>> f4809f86105dbeee59a57d8e7c183343d745335f
18
19    }
```

```
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git add .
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git commit -m "Fixed"
[main ac40c12] Fixed
PS C:\Users\User\Desktop\Curso Kotlin\praticaGit> git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 667 bytes | 333.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
```

Conclusiones

1. Importancia del control de versiones ya que Git registra todo el historial de modificaciones en un proyecto, lo que facilita rastrear, recuperar versiones anteriores y detectar fallos.
2. Trabajo colaborativo más organizado ya que el uso de ramas, se pueden crear nuevas ramas de manera independiente sin modificar el proyecto principal, lo que ayuda mucho a los entornos de CI/CD

3. Resolución de conflictos ya que pueden presentarse conflictos cuando hay múltiples colaboradores, pero Git proporciona herramientas efectivas para solucionarlos y preservar la calidad del código
4. Buenas practicas como desarrollador

Referencias

Castellanos, E. (2022). *Free Code Camp*. Obtenido de <https://www.freecodecamp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/>

Git. (s.f.). Obtenido de Git: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git>

Worsley, V. (s.f.). *Data Camp*. Obtenido de Data Camp: <https://www.datacamp.com/blog/all-about-git>