

UNIVERSIDAD DE CUNDINAMARCA

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

COMANDOS DE GIT EN INTELLIJ

TRABAJO PRESENTADO POR:

Laura Daniela Hoyos Gordillo

ASIGNATURA:

Programación 2

DOCENTE:

William Alexander Matallana Rojas

FECHA:

20 de febrero de 2025

SEDE:

Chía

Tabla de contenido

Uso de IntelliJ IDEA con GitHub	3
Objetivos	3
Desarrollo.....	4
Creación del repositorio	4
Git init	4
Git Clone	5
Git status	5
Git add.....	6
Git commit	6
Git push.....	6
Git switch.....	7
Git Branch.....	7
Git Pull	7
Conclusiones	8
Referencias.....	9

Uso de IntelliJ IDEA con GitHub

IntelliJ IDEA facilita la gestión de proyectos con Git, permitiendo a los desarrolladores administrar su código sin salir del entorno de desarrollo. Entre sus principales funcionalidades se encuentran:

- **Clonación de repositorios:** Los usuarios pueden importar proyectos directamente desde plataformas como GitHub sin complicaciones.
- **Registro y envío de cambios:** Las modificaciones realizadas en el código pueden registrarse con un mensaje descriptivo y enviarse a un servidor remoto.
- **Gestión de ramas:** IntelliJ IDEA permite crear, cambiar, fusionar y eliminar ramas, lo que facilita la organización del flujo de trabajo.
- **Organización de cambios antes de su confirmación:** Con la herramienta Changelists, es posible agrupar modificaciones antes de guardarlas, mejorando el control sobre el código.
- **Comparación de versiones:** Los desarrolladores pueden visualizar las diferencias entre archivos, comparar cambios entre ramas y restaurar versiones anteriores cuando sea necesario.

Objetivos

- Comprender cómo se integra Git en IntelliJ IDEA para facilitar la gestión de versiones dentro del entorno de desarrollo.
- Explorar las herramientas de IntelliJ IDEA que permiten el uso de Git de manera eficiente.
- Analizar las funciones esenciales de Git en IntelliJ IDEA, como la clonación de repositorios, la confirmación de cambios, la gestión de ramas y la resolución de conflictos.

Desarrollo

Lo primero que debemos hacer en IntelliJ IDEA es realizar una configuración inicial. es necesario establecer el nombre del usuario en Git ejecutando el siguiente comando “`git config --global user.name` “Tu Nombre” en la terminal, también usamos “`git config --global user.email`” para guardar el correo del usuario, y por ultimo usamos un “`git config --global --list`” para comprobar que todo nos haya quedado bien.

```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\ADMIN\IdeaProjects\untitled1> git config --global user.name "Laura Daniela Hoyos Gordillo"
PS C:\Users\ADMIN\IdeaProjects\untitled1> git config --global user.email laura7713h@gmail.com
PS C:\Users\ADMIN\IdeaProjects\untitled1> git config --global --list
user.name=Laura Daniela Hoyos Gordillo
user.email=laura7713h@gmail.com
```

Creación del repositorio

Para crear un repositorio después de haber creado el proyecto vamos a GitHub para crear un nuevo repositorio, Escribe un nombre para tu repositorio y configura la visibilidad (público o privado). Luego copiamos la siguiente información en el terminal

```
... Terminal Local x + v
user.email=laura7713h@gmail.com
PS C:\Users\ADMIN\IdeaProjects\untitled1> echo "# ejemplotareAa" >> README.md
PS C:\Users\ADMIN\IdeaProjects\untitled1> git init
Reinitialized existing Git repository in C:/Users/ADMIN/IdeaProjects/untitled1/.git/
PS C:\Users\ADMIN\IdeaProjects\untitled1> git add README.md
PS C:\Users\ADMIN\IdeaProjects\untitled1> git commit -m "first commit"
[master (root-commit) 5e87fef] first commit
 8 files changed, 78 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 README.md
 create mode 100644 src/Main.java
 create mode 100644 untitled1.iml
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch -M main
PS C:\Users\ADMIN\IdeaProjects\untitled1> git remote add origin https://github.com/hk-laurah/ejemplotareAa.git
PS C:\Users\ADMIN\IdeaProjects\untitled1> git push -u origin main
```

Con esto, nuestro proyecto local ya está sincronizado con el repositorio en GitHub, permitiéndonos gestionar cambios y colaborar fácilmente

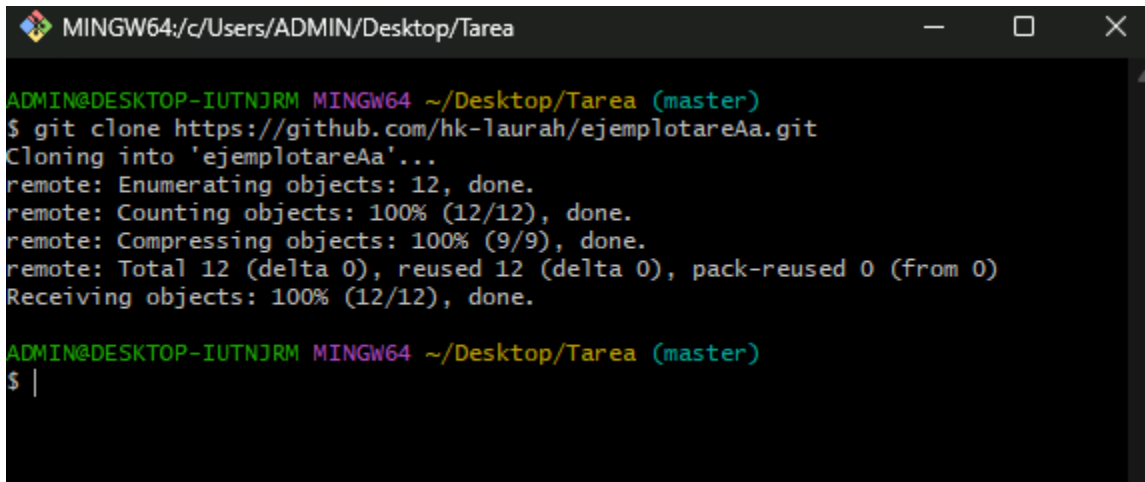
Git init

El comando Git init puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git init
Reinitialized existing Git repository in C:/Users/ADMIN/IdeaProjects/untitled1/.git/
PS C:\Users\ADMIN\IdeaProjects\untitled1>
```

Git Clone

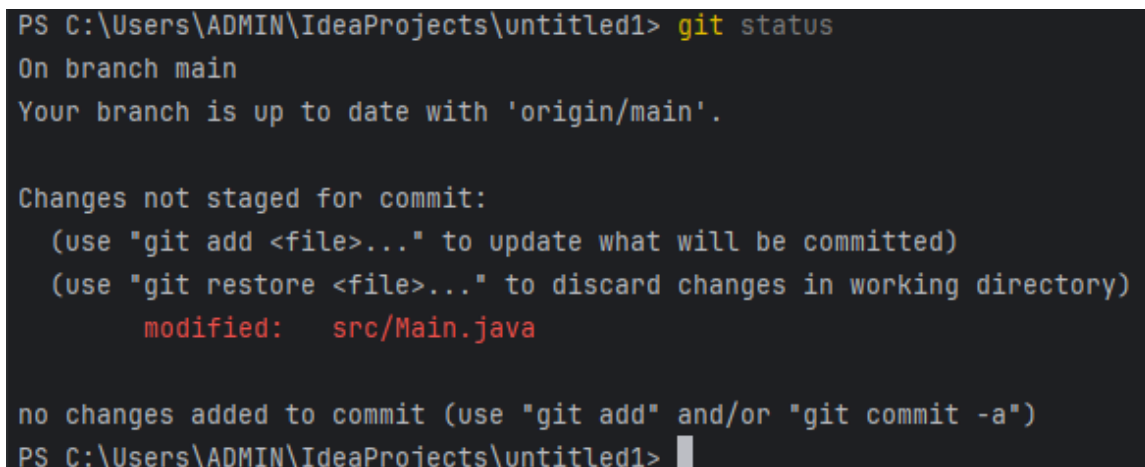
Git clone se utiliza para copiar un repositorio remoto, Para usar Git clone, primero necesitamos la dirección del repositorio que queremos copiar. Por ejemplo, si el proyecto está en GitHub, debemos entrar al repositorio, hacer clic en el botón "Code" y copiar la url que aparece. Luego, abrimos la terminal y abrimos en la carpeta donde queremos descargar el proyecto y ejecutamos el comando junto con la url copiada. Esto descargará todos los archivos y el historial del repositorio, permitiéndonos trabajar en él localmente.



```
MINGW64:/c/Users/ADMIN/Desktop/Tarea
ADMIN@DESKTOP-IUTNJRM MINGW64 ~/Desktop/Tarea (master)
$ git clone https://github.com/hk-laurah/ejemplotareAa.git
Cloning into 'ejemplotareAa'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (12/12), done.
ADMIN@DESKTOP-IUTNJRM MINGW64 ~/Desktop/Tarea (master)
$ |
```

Git status

Este comando se utiliza para verificar el estado actual de un repositorio en Git, Se usa antes de hacer un commit o un push, para asegurarnos de qué cambios se van a guardar o enviar al repositorio remoto. También es útil para confirmar si olvidamos agregar algún archivo.



```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/Main.java

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\ADMIN\IdeaProjects\untitled1> |
```

Git add

El comando Git add se usa para preparar archivos antes de hacer un commit.

Básicamente, le indica a Git qué cambios queremos guardar en la próxima confirmación commit.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git add .
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\ADMIN\IdeaProjects\untitled1> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/Main.java
```

Git commit

El comando git commit se usa para guardar los comentarios en el historial del repositorio de forma permanente

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git commit -m "se agrega el comentario"
[main 95bff23] se agrega el comentario
 1 file changed, 1 insertion(+), 11 deletions(-)
PS C:\Users\ADMIN\IdeaProjects\untitled1>
```

Git push

El comando Git push se usa para enviar los cambios de un repositorio local a un repositorio remoto como GitHub. Después de hacer git commit, los cambios solo están guardados en nuestra computadora. Con Git push, los subimos al servidor remoto para que otros puedan verlos y colaborar en el proyecto.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 392 bytes | 392.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/hk-laurah/ejemplotareAa.git
 5e87fef..95bff23  main -> main
```

Git switch

El comando Git switch se usa para cambiar entre ramas en un repositorio de Git. Una rama es una versión paralela de nuestro proyecto, y con Git switch podemos movernos de una rama a otra fácilmente.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git switch -c Ramaprueba
Switched to a new branch 'Ramaprueba'
PS C:\Users\ADMIN\IdeaProjects\untitled1> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Git Branch

El comando Git Branch se usa para gestionar y visualizar las ramas en un repositorio de Git. Las ramas permiten desarrollar nuevas funcionalidades sin afectar el código principal, facilitando el trabajo en equipo y la organización del proyecto.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch PruebaRama2
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch PruebaRama3
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch PruebaRama4
```

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch
  PruebaRama2
  PruebaRama3
  PruebaRama4
  Ramaprueba
* main
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch -m Rama4.0
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch
  PruebaRama2
  PruebaRama3
  PruebaRama4
* Rama4.0
PS C:\Users\ADMIN\IdeaProjects\untitled1> git branch -d PruebaRama3
Deleted branch PruebaRama3 (was 95bff23).
PS C:\Users\ADMIN\IdeaProjects\untitled1>
```

Git Pull

El Git Pull se usa para actualizar tu repositorio local con los cambios más recientes del repositorio remoto.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git pull origin main
From https://github.com/hk-laurah/ejemplotareAa
* branch          main          -> FETCH_HEAD
Already up to date.
PS C:\Users\ADMIN\IdeaProjects\untitled1>
```

Git Revert

El comando Git revert permite deshacer cambios de un commit anterior de una manera segura. En lugar de eliminar commits del historial (Como lo hace git reset), git revert crea un nuevo commit que invierte los cambios introducidos por el commit especificado. Esto es ideal para trabajar en equipo, ya que no altera el historial existente y evita conflictos. Es una herramienta esencial para corregir sin perder rastro de lo que ha sucedido en el proyecto

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git revert
usage: git revert [--no-edit] [-n] [-m <parent-number>] [-s] [-S[<keyid>]] <commit>...
or: git revert (--continue | --skip | --abort | --quit)
```

Git fetch --all

Git fetch --all se utiliza en Git para obtener todos los cambios de todos los repositorios remotos asociados a nuestro proyecto, pero sin unirlos automáticamente con nuestra rama local. Es una forma segura de actualizar nuestro repositorio local con la información más reciente del repositorio remoto.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git fetch
```

Git log

Es una herramienta fundamental en Git que te permite ver el historial de commits de un repositorio. Muestra información detallada sobre cada commit, como el autor, la fecha, el mensaje del commit y el identificador único (hash) del commit.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git log
commit 95bfff23a22e5b4646eaae7d60b41a4a9750e0cf2 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2)
Author: Laura Daniela Hoyos Gordillo <laura7713h@gmail.com>
Date: Thu Feb 20 15:27:11 2025 -0500

    se agrega el comentario

commit 5e87feffa8ea276d2ab4dcabb63c8beb8820c485
Author: Laura Daniela Hoyos Gordillo <laura7713h@gmail.com>
Date: Thu Feb 20 14:42:37 2025 -0500

    first commit
```


Git reflog

El comando Git reflog es una herramienta que te permite ver el historial de todas las acciones realizadas en nuestro repositorio local, incluyendo cambios de ramas, resets, merges, commits y más, git reflog registra todas las operaciones que afectan a las referencias (como ramas y head).

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git reflog
95bfff23 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2) HEAD@{0}: Branch: renamed refs/heads/main to refs/heads/Rama4.0
95bfff23 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2) HEAD@{2}: checkout: moving from Ramaprueba to main
95bfff23 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2) HEAD@{3}: checkout: moving from main to Ramaprueba
95bfff23 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2) HEAD@{4}: commit: se agrega el comentario
5e87fef HEAD@{5}: Branch: renamed refs/heads/main to refs/heads/main
5e87fef HEAD@{7}: Branch: renamed refs/heads/main to refs/heads/main
5e87fef HEAD@{9}: Branch: renamed refs/heads/master to refs/heads/main
5e87fef HEAD@{11}: commit (initial): first commit
PS C:\Users\ADMIN\IdeaProjects\untitled1>
```

Git log --oneline

Es una versión simplificada y más compacta de Git log. Muestra el historial de commits en un formato mas resumido, donde cada commit se representa en una sola línea. Esto es especialmente útil cuando queremos revisar rápidamente el historial de un repositorio sin detalles adicionales.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> Git log --oneline
95bfff23 (HEAD -> Rama4.0, origin/main, origin/HEAD, Ramaprueba, PruebaRama4, PruebaRama2) se agrega el comentario
5e87fef first commit
```

Git merge+ (--Rebase)

git merge y git rebase, que son dos formas diferentes de integrar cambios de una rama a otra en Git. Ambos tienen propósitos similares, pero funcionan de maneras distintas y se usan en diferentes situaciones.

```
PS C:\Users\ADMIN\IdeaProjects\untitled1> git merge --rebase "RamaPrueba4"
```

Git push origin --delete rama

Se utiliza para eliminar una rama en un repositorio remoto. Esto es útil cuando ya no necesitamos una rama remota (por ejemplo, después de fusionar sus cambios en la rama principal o cuando la rama ya no es relevante).

Conclusiones

Git es una herramienta esencial que permite a los desarrolladores trabajar en equipo de manera segura y eficiente, ya que cada modificación queda registrada en el historial y puede ser revertida si es necesario. Gracias a su integración con IntelliJ IDEA, se facilita la gestión del control de versiones mediante herramientas visuales para clonar

repositorios, administrar ramas y comparar cambios sin necesidad de usar la línea de comandos. Desde la inicialización de un repositorio con ``git init`` hasta la sincronización con servidores remotos mediante “`git push`” y “`git pull`”, cada comando juega un papel crucial en la gestión del código fuente. Además, con “`git branch`” y “`git switch`”, los desarrolladores pueden trabajar en nuevas funcionalidades en ramas separadas sin afectar la versión estable del código, lo que garantiza un flujo de trabajo estructurado y seguro

.Referencias

- <https://www.atlassian.com/git/tutorials/saving-changes>
- <https://git-scm.com/book/es/v2/Ap%C3%A9ndice-C:-Comandos-de-Git-Seguimiento-B%C3%A1sico>
- <https://www.atlassian.com/es/git/tutorials/syncing/git-pull>
- <https://git-scm.com/book/es/v2/Fundamentos-de-Git-Deshacer-Cosas>
- <https://git-scm.com/docs/git-revert>