

ACADEMÍA JAVA XIDERAL MONTERREY, N.L



SEMANA 1 INTRODUCCIÓN A GIT

**Realizado por:
Luis Miguel Sánchez Flores**

INTRODUCCIÓN

Git es un software considerado como un Sistema de Control de Versiones (VCS, por sus siglas en inglés) que mantiene un registro de los cambios realizados en los archivos a lo largo del tiempo, lo que genera un historial de los datos que puede ser recuperado cuando sea necesario.

Git se convirtió en la herramienta *de facto* para mantener el control de versiones de los archivos gracias a su naturaleza distribuida que permite a los equipos trabajar en un proyecto de codificación simultáneamente de una manera organizada y eficaz, aprovechando sus potentes funcionalidades que se adaptan a un flujo de trabajo no lineal. Es idóneo para realizar la ramificación, y el etiquetado como procesos prioritarios.

Hoy en día, Git va más allá de proyectos de programación, ya que está siendo utilizado por diseñadores, escritores e incluso instituciones gubernamentales para mantener su progreso del proyecto.

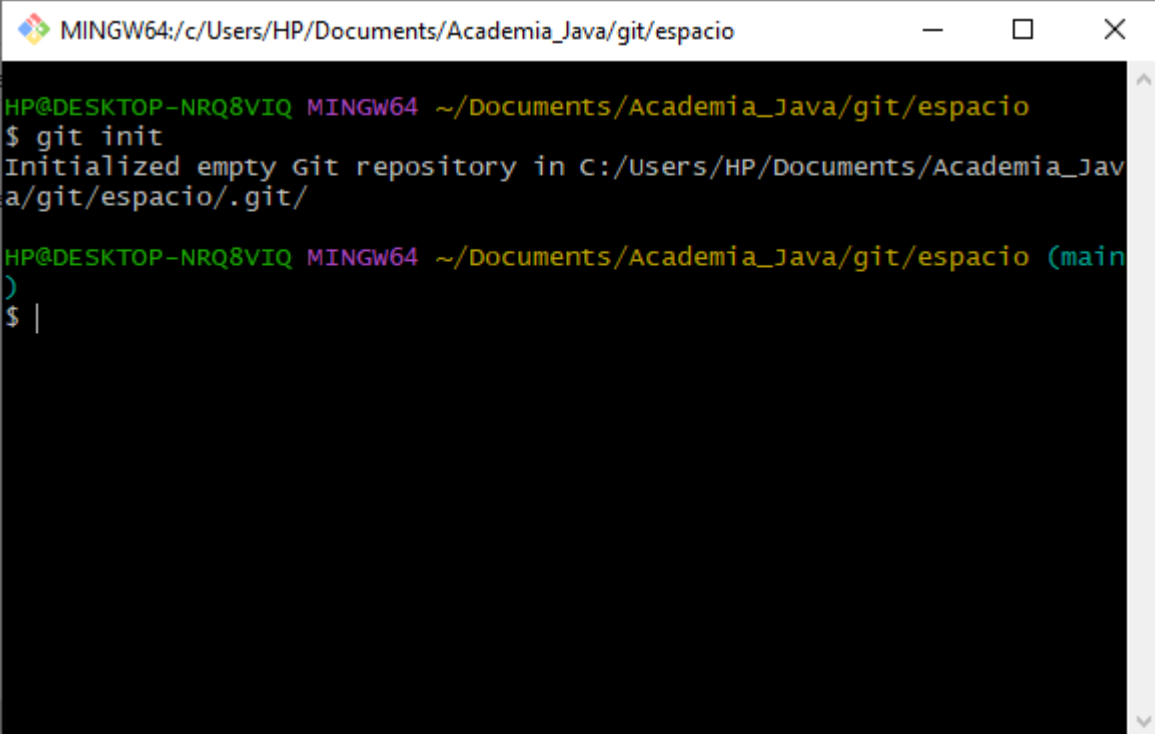
Coincidentemente, Git fue creado por Linus Torvalds en el año 2005, durante el desarrollo de sistema operativo Linux, después de que los VCS existentes fueran incómodos de usar para el equipo, estuvieran detrás de un muro de pago o simplemente no cumplieran las expectativas del equipo, tomando el asunto en sus propias manos para crear un sistema potente, de código abierto, rápido y que soportara procesos de desarrollo no lineales. Tras semanas de intenso trabajo, la primera versión de Git fue lanzada en abril del año 2005.

CONCEPTOS PRINCIPALES

GIT INIT

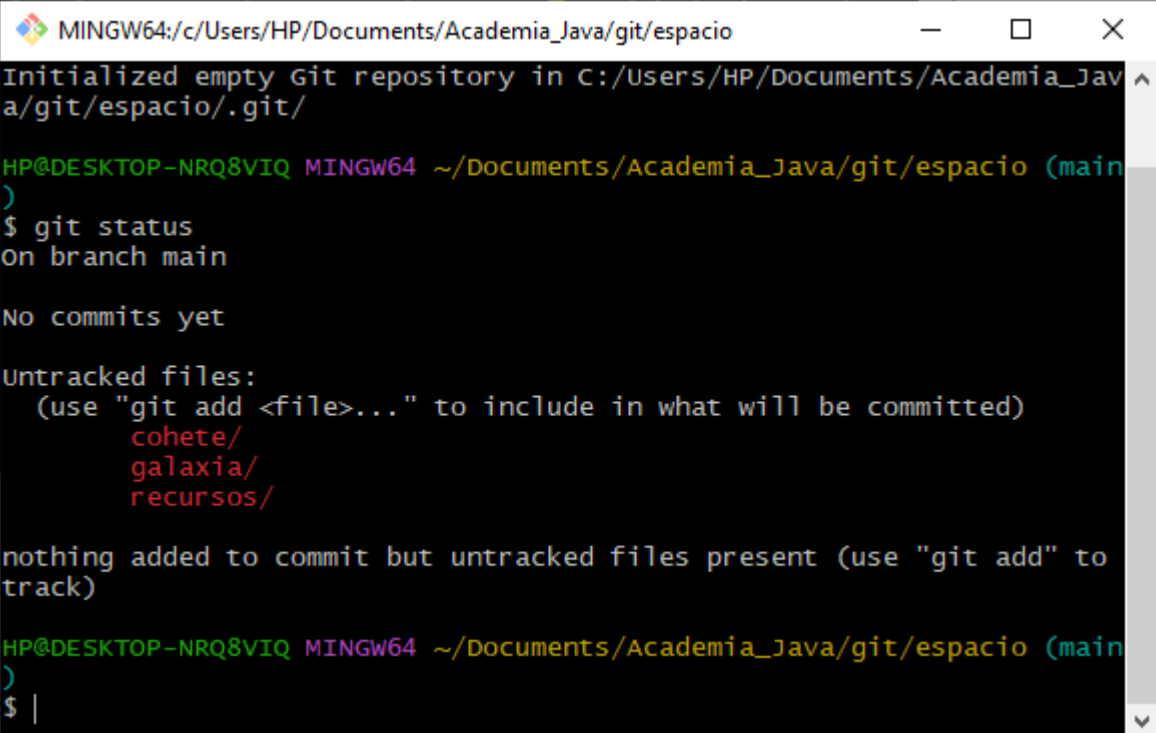
Git trabaja con un **repositorio**, que esencialmente es la carpeta del proyecto completo, que incluye una carpeta `.git` oculta, **permitiendo al programa rastrear los cambios en los contenidos / archivos correspondientes**.

Para inicializar el repositorio, basta con llamar a **git init** en la carpeta raíz del proyecto. Ahora con el acceso del resto de los comandos, podemos aprovechar el poder de Git dentro del proyecto.

A screenshot of a terminal window titled 'MINGW64:/c:/Users/HP/Documents/Academia_Java/git/espacio'. The terminal shows the command '\$ git init' being entered, followed by the output 'Initialized empty Git repository in c:/Users/HP/Documents/Academia_Java/git/espacio/.git/'. Below this, the prompt changes to 'HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)' and the user enters '\$ |'.

GIT STATUS

Uno de los comandos más sencillos de Git es **git status**, que proporciona un breve resumen de los archivos y cambios que Git va a tomar en cuenta, y es útil para los usuarios quienes desean revisar el contenido que se va a rastrear:

A screenshot of a Windows terminal window titled "MINGW64: c:/Users/HP/Documents/Academia_Java/git/espacio". The terminal shows the output of the 'git status' command. It indicates that an empty Git repository has been initialized in the specified path. The user is on the 'main' branch, and there are no commits yet. Three untracked files are listed: 'cohete/', 'galaxia/', and 'recursos/'. The terminal also shows the prompt '\$ |' at the bottom.

```
MINGW64: c:/Users/HP/Documents/Academia_Java/git/espacio
Initialized empty Git repository in C:/Users/HP/Documents/Academia_Java/git/espacio/.git/

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git status
on branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        cohete/
        galaxia/
        recursos/

nothing added to commit but untracked files present (use "git add" to track)

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ |
```

GIT ADD

Para comenzar el rastreo de los cambios deseados, hay que incluirlos en el área de preparación de Git (también conocido como el *staging area*) mediante el comando **git add**.

El *staging area*, también conocida como el índice, es un espacio intermedio donde Git incluye toda la nueva información que va a guardar, pero también permite al usuario revisar y formatear dicho contenido antes de hacerlo.

El área de preparación diferencia a Git de otros sistemas de control de versiones, permitiendo al usuario incluir archivos específicos en lugar de todos, o incluso solamente incluir partes específicas de un archivo particular.

Podemos utilizar el comando **git add ARCHIVO** para incluir un solo archivo específico al *staging area*, o simplemente podemos aplicar el comando de **git add** . para incluir todos los archivos que aún no están siendo rastreados por Git.

```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git add galaxia/planetas/tierra.txt

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git status
on branch main

No commits yet

changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   galaxia/planetas/tierra.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    cohete/
    galaxia/facciones/
    galaxia/planetas/jupiter.txt
    galaxia/planetas/marte.txt
```

```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git add .

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git status
on branch main

No commits yet

changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   cohete/inventario/comida.txt
    new file:   cohete/inventario/dispositivos.txt
    new file:   cohete/inventario/gas.txt
    new file:   cohete/inventario/medicinas.txt
    new file:   galaxia/facciones/aliens.txt
    new file:   galaxia/facciones/humanos.txt
    new file:   galaxia/facciones/piratas.txt
    new file:   galaxia/planetas/jupiter.txt
```

Con los archivos incluidos en el *staging area*, ahora se puede emplear uno de los comandos más fundamentales de Git: **git commit**

GIT COMMIT

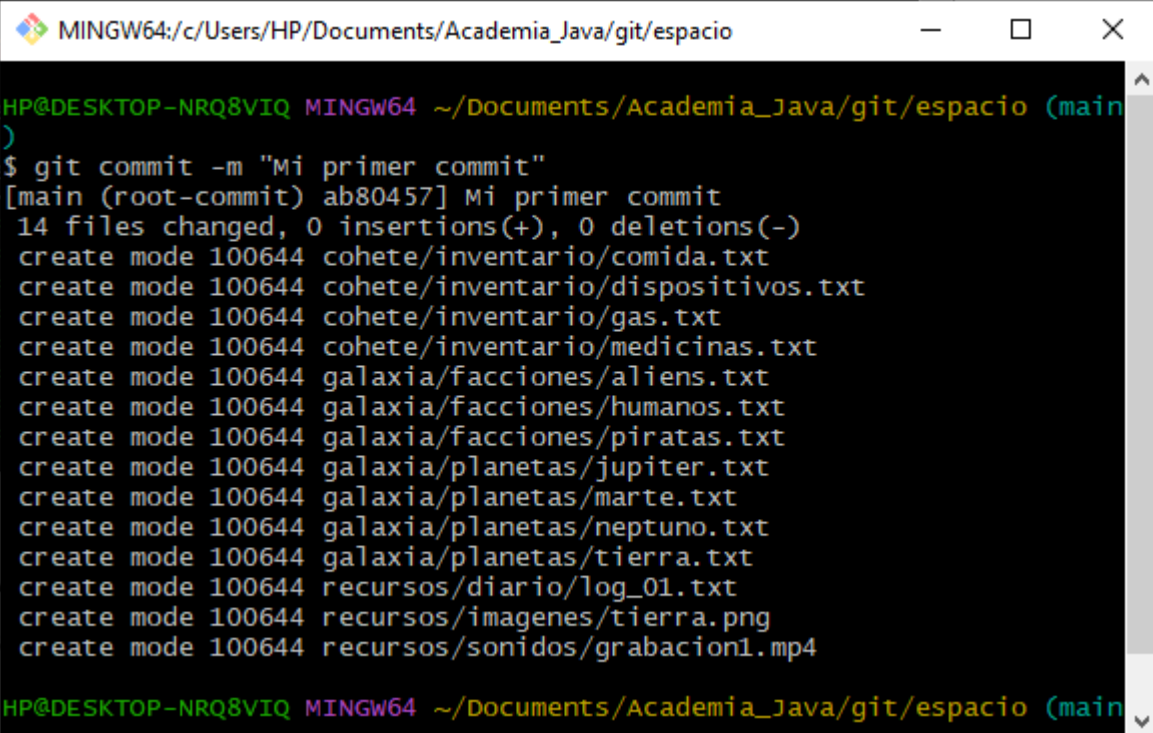
Se puede considerar el comando commit como la acción de tomar una captura instantánea de la información en un determinado momento: Git registra todo el contenido incluido en el *staging area* y lo introduce en el historial de commits realizados hasta ahora, ayudando a mantener un conjunto de versiones del proyecto y dando a conocer la evolución de la misma hasta la versión más reciente.

Además del contenido, **los commits pueden contener un mensaje clave** que puede escribir el usuario para describir los cambios realizados en ese momento, lo que ayuda a los demás colaboradores a entender el propósito de dichos cambios, así como mantener una referencia de lo que se desarrolló para el futuro.

Para los mensajes se recomienda escribirlos de forma concisa, proporcionar el suficiente contexto sobre los cambios realizados y, si se aplica, seguir el formato determinado.

Además del mensaje, el commit puede contener metadatos adicionales, como el autor (nombre y correo) que generó el commit, y la fecha en que se produjo.

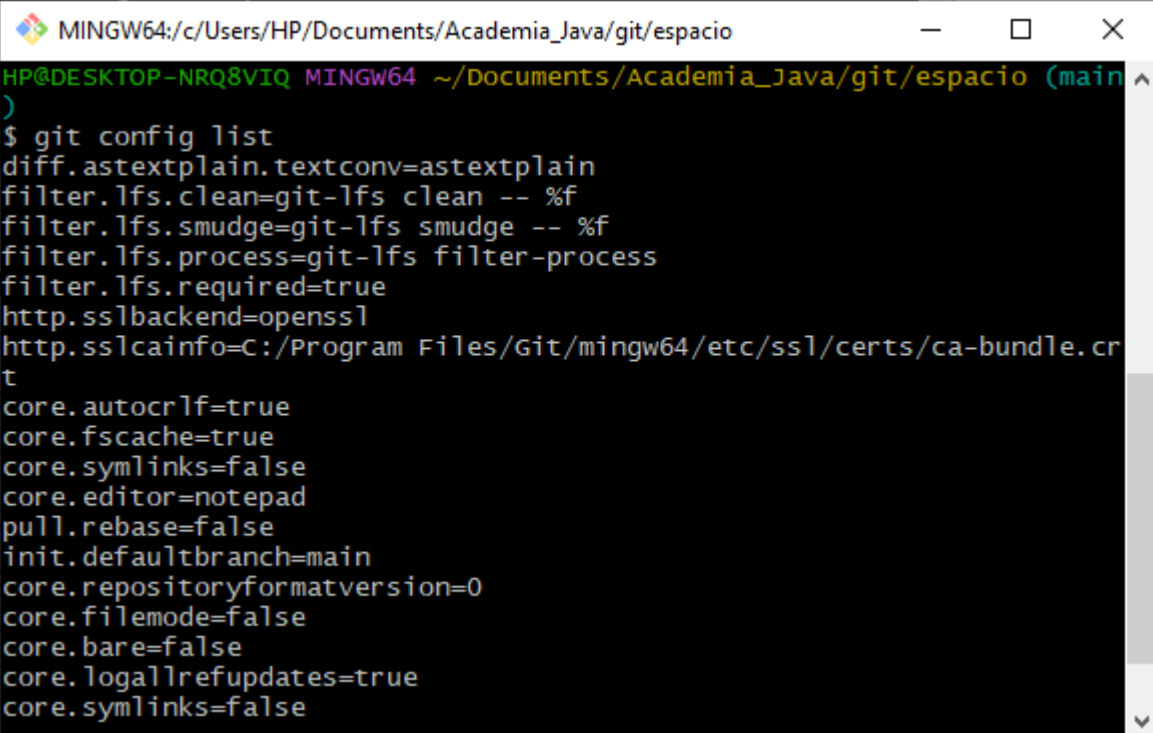
El formato común para efectuar un commit es **git commit -m MENSAJE**, como se demuestra a continuación:



```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git commit -m "Mi primer commit"
[main (root-commit) ab80457] Mi primer commit
14 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 cohete/inventario/comida.txt
create mode 100644 cohete/inventario/dispositivos.txt
create mode 100644 cohete/inventario/gas.txt
create mode 100644 cohete/inventario/medicinas.txt
create mode 100644 galaxia/facciones/aliens.txt
create mode 100644 galaxia/facciones/humanos.txt
create mode 100644 galaxia/facciones/piratas.txt
create mode 100644 galaxia/planetas/jupiter.txt
create mode 100644 galaxia/planetas/marte.txt
create mode 100644 galaxia/planetas/neptuno.txt
create mode 100644 galaxia/planetas/tierra.txt
create mode 100644 recursos/diario/log_01.txt
create mode 100644 recursos/imagenes/tierra.png
create mode 100644 recursos/sonidos/grabacion1.mp4
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
```

.gitignore y .gitconfig

También es importante dar a conocer sobre archivos especiales que afectan las operaciones de los comandos. Uno de ellos es **.gitconfig**, el cual permite configurar parámetros (ya sean de forma local o global) para establecer preferencias que mejor se adapten al flujo de trabajo del usuario. Por ejemplo, mediante **.gitconfig** se configura el nombre y correo electrónico del usuario que se utilizan al momento de realizar un comando commit:

A screenshot of a terminal window titled 'MINGW64: c:/Users/HP/Documents/Academia_Java/git/espacio'. The prompt is 'HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)'. The command '\$ git config list' has been executed, displaying a list of Git configuration settings. The settings include diff, filter, http, and core options, such as 'diff.astextplain.textconv=astextplain', 'filter.lfs.clean=git-lfs clean -- %f', 'http.sslbackend=openssl', 'http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt', 'core.autocrlf=true', 'core.fscache=true', 'core.symlinks=false', 'core.editor=notepad', 'pull.rebase=false', 'init.defaultbranch=main', 'core.repositoryformatversion=0', 'core.filemode=false', 'core.bare=false', 'core.logallrefupdates=true', and 'core.symlinks=false'.

```
MINGW64: c:/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ 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/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.editor=notepad
pull.rebase=false
init.defaultbranch=main
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
```

También se encuentra el archivo **.gitignore** que establece una “lista negra” de archivos del proyecto que no deben ser rastreados por Git al momento de capturar los cambios. El **.gitignore** es bastante útil para dejar fuera archivos ya sean redundantes o que puedan contener información sensible.

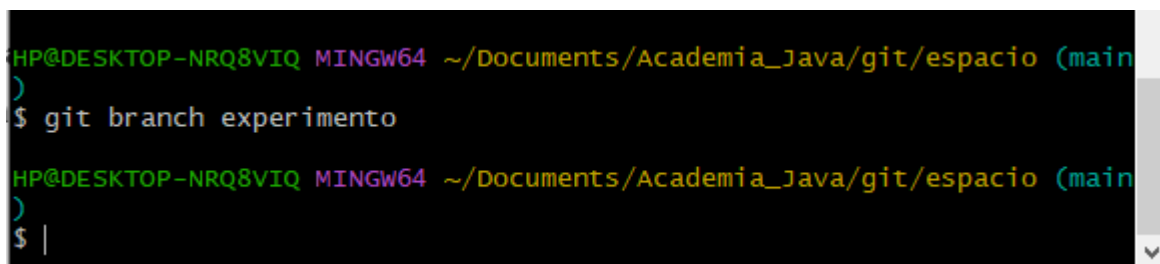
TRABAJANDO CON RAMAS

Durante la fase de desarrollo del proyecto, a los usuarios les gusta encapsular algunos cambios antes de introducirlos al trabajo, como una especie de borrador, ya sea para experimentar cosas y/o corregir errores. Para un equipo enorme que trabaja en el proyecto de forma simultánea, manipulando los mismos archivos a la vez, la organización y la colaboración se vuelven críticos para garantizar el desarrollo exitoso del proyecto e incluir el contenido adecuado en la versión final.

Git puede agilizar este proceso de encapsulación mediante el **uso de ramas**. Las ramas funcionan sobre un commit (generalmente en el más reciente hasta el momento) y generan un espacio de trabajo aislado del proyecto, lo que permite a los usuarios a experimentar y realizar cambios en los archivos sin afectar a la **rama principal (conocida como master o main)**.

La rama principal es creada por defecto cuando se inicializa un repositorio, y normalmente es utilizada para guardar los archivos completos, libres de errores y estables del proyecto, mientras que se generan ramas adicionales para trabajar en cambios experimentales o en borradores hasta conseguir una versión apropiada para la rama principal.

Las ramas son creadas mediante el comando **git branch NOMBRE**



```
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git branch experimento

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ |
```

Aunque se haya creado la rama, Git no transfiere al usuario a la nueva rama. Sigue en la misma rama que antes. Para ello, podemos utilizar el comando **git checkout RAMA** o el comando **git switch RAMA** para ubicarnos en la nueva rama:

```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git checkout experimento
Switched to branch 'experimento'

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (experimento)
$ |
```

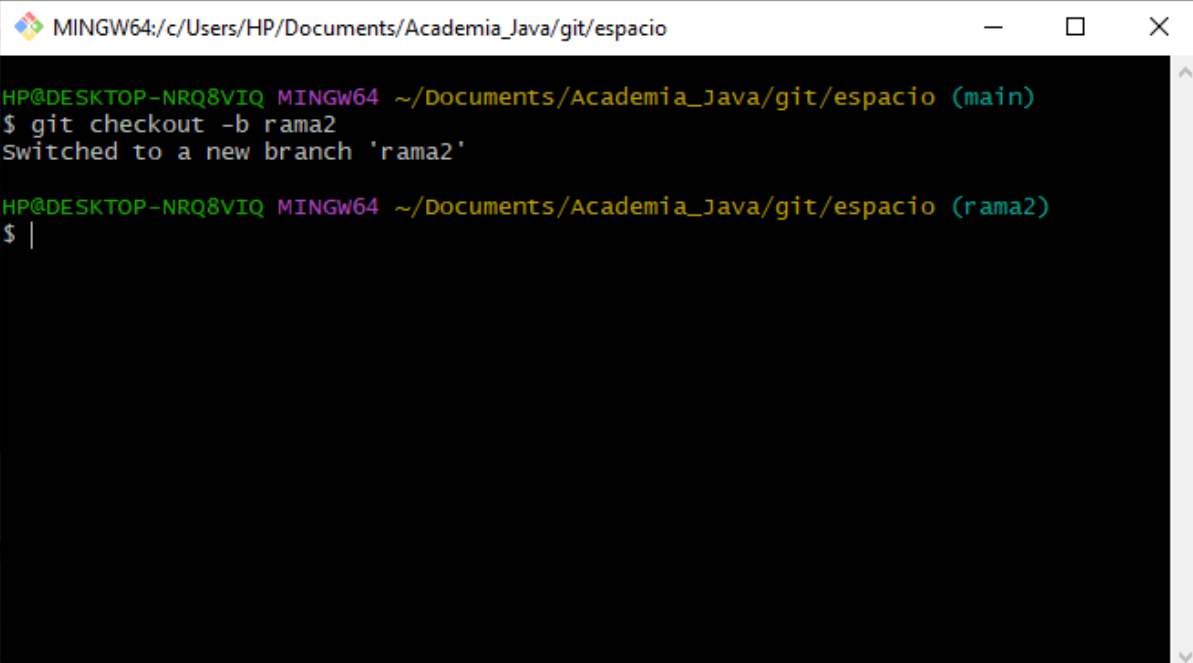
Se puede desplegar todas las ramas disponibles en el repositorio mediante la **opción -a del comando branch**, así como desplegar la rama en el que el usuario se encuentra ahora mediante la **opción --show-current**:

```
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (experimento)
$ git branch -a
* experimento
  main

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (experimento)
$ git branch --show-current
experimento

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (experimento)
$ |
```

Git también ofrece atajos en las opciones del comando **checkout** para crear una rama y cambiar a ella de forma automática. Para ello, podemos aplicar **git checkout -b RAMA** para generar y cambiarse a la nueva rama:

A terminal window titled 'MINGW64; c:/Users/HP/Documents/Academia_Java/git/espacio'. The prompt is 'HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)'. The user enters '\$ git checkout -b rama2', and the output is 'Switched to a new branch 'rama2''. The prompt changes to 'HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (rama2)'. The user enters '\$ |' and the cursor is at the end of the line.

```
MINGW64; c:/Users/HP/Documents/Academia_Java/git/espacio
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (main)
$ git checkout -b rama2
Switched to a new branch 'rama2'
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio (rama2)
$ |
```

INTEGRANDO LOS NUEVOS CAMBIOS

En la nueva rama, se ejecutan los commits necesarios y finalmente se decide que los cambios finales están listos para ser introducidos a la rama principal. Pero, ¿cómo se integran estos cambios a otra rama?

Git nos ofrece dos formas para integrar los cambios de una rama a otra: La primera es con el comando de **git merge**.

Como el nombre lo indica, el comando merge combina los cambios de la rama especificada a la rama actual, generando un nuevo commit. Esto ayuda a preservar la historia de ambas ramas y es una forma segura para integrar los cambios.

Por ejemplo, si se tiene cambios en una rama que se desea integrar a la rama principal:

```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio/galaxia/planetas
commit 1dc5404406fc374e34e5c33c3cc7eb98f2e70117 (HEAD -> experimento)
Author: Pepe Gonzalez <pepito@gmail.com>
Date: Fri Aug 16 22:14:58 2024 -0600

    Modificado neptuno.txt

commit ec071758042eb45ca60ecf08977e2347069b67ee
Author: Pepe Gonzalez <pepito@gmail.com>
Date: Fri Aug 16 22:14:20 2024 -0600

    Agregado información a jupiter.txt

commit 713565ee2d68810490f76740a25475b03f1040b8
Author: Pepe Gonzalez <pepito@gmail.com>
Date: Fri Aug 16 22:12:38 2024 -0600

    Agregado informacion a tierra.txt

commit ab80457457d87c42ae1c10edb2a56191c9e29686 (rama2, main)
Author: Pepe Gonzalez <pepito@gmail.com>
Date: Fri Aug 16 21:04:24 2024 -0600
:|
```

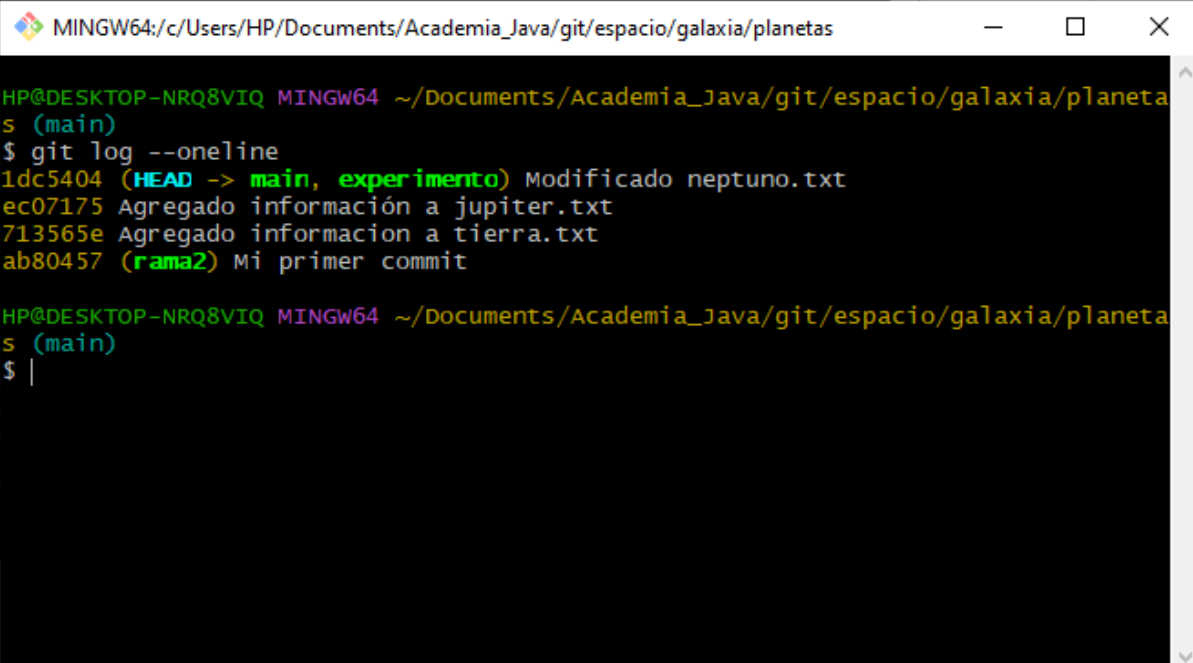
Es necesario introducirse a la rama principal mediante el comando **checkout**, y dentro de la rama, ejecutar el comando **merge** para integrar todos los nuevos cambios de la rama de experimento:

```
MINGW64:/c/Users/HP/Documents/Academia_Java/git/espacio/galaxia/planetas
commit ab80457457d87c42ae1c10edb2a56191c9e29686 (rama2, main)
Author: Pepe Gonzalez <pepito@gmail.com>
Date: Fri Aug 16 21:04:24 2024 -0600

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio/galaxia/planetas (experimento)
$ git checkout main
Switched to branch 'main'

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio/galaxia/planetas (main)
$ git merge experimento
Updating ab80457..1dc5404
Fast-forward
 galaxia/planetas/jupiter.txt | 1 +
 galaxia/planetas/neptuno.txt | 1 +
 galaxia/planetas/tierra.txt  | 1 +
 3 files changed, 3 insertions(+)

HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio/galaxia/planetas (main)
$ |
```



```
MINGW64; c:/Users/HP/Documents/Academia_Java/git/espacio/galaxia/planetas
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio/galaxia/planetas
$ git log --oneline
1dc5404 (HEAD -> main, experimento) Modificado neptuno.txt
ec07175 Agregado información a jupiter.txt
713565e Agregado información a tierra.txt
ab80457 (rama2) Mi primer commit
HP@DESKTOP-NRQ8VIQ MINGW64 ~/Documents/Academia_Java/git/espacio/galaxia/planetas
$ |
```

En cambio, existe el comando de **git rebase**, que en cambio sobrescribe el historial de commits al mover aquellos de la rama original a la rama destinada, dando la ilusión de que la rama original no existía en primer lugar y ordenado el historial de commits de forma lineal.

Sin embargo, el usuario debe utilizar el comando con precaución, ya que su mal uso puede dar lugar a complicaciones graves que pueden ser difíciles de resolver.