



GIT : MI PRIMER REPOSITORIO

CLASE 2

PROF. NATALIA LUCERO


•CONCEPTOS GENERALES:

- Al ser un sistema de control de versiones, permite ver el estado del proyecto, los cambios realizados en cada archivo, la persona que los realizó, así como poder restaurar una versión anterior.










¿QUÉ ES UNA VERSIÓN?

- Conjunto de nuevas características y funcionalidades de un software disponibles para el usuario final.



A horizontal sequence of seven Windows logos, each corresponding to a version of the operating system. The logos transition from the multi-colored flag design of XP and Vista to the blue four-pane design of Windows 8 and later.

						
Windows XP	Windows Vista	Windows 7	Windows 8	Windows 8.1	Windows 10	Windows 11
2001	2006	2009	2012	2013	2015	2021
5.1	6.0	6.1	6.2	6.3	10.0	10.0

¿CÓMO SABER SI INSTALÉ BIEN GIT?

- En la terminal de Git bash ejecutamos git



```
MINGW64/c/Users/Paloma

Paloma@DESKTOP-CEETV4P MINGW64 ~ (master)
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

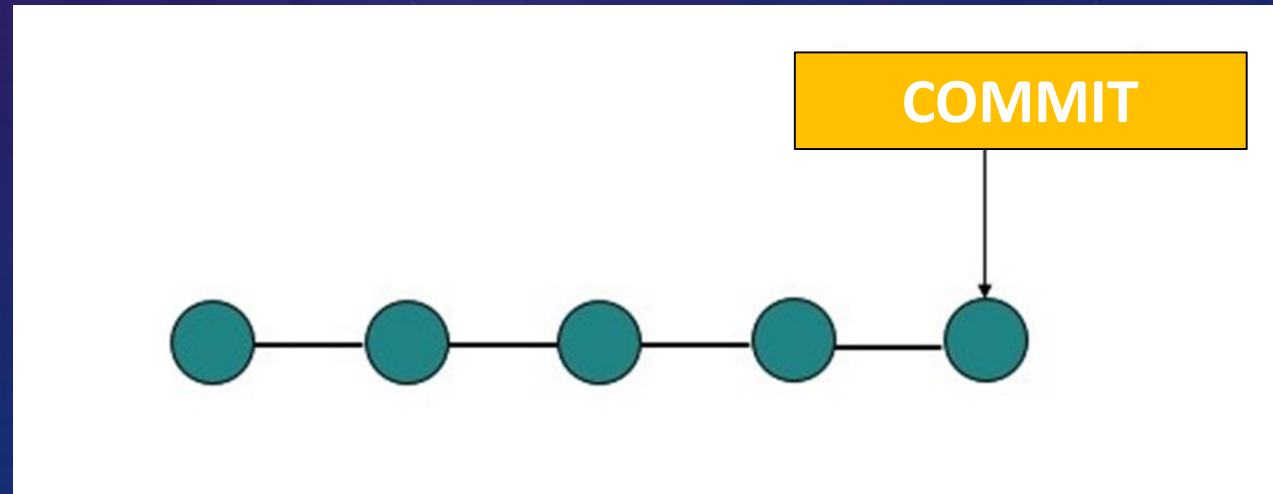
collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

Paloma@DESKTOP-CEETV4P MINGW64 ~ (master)
$ |
```

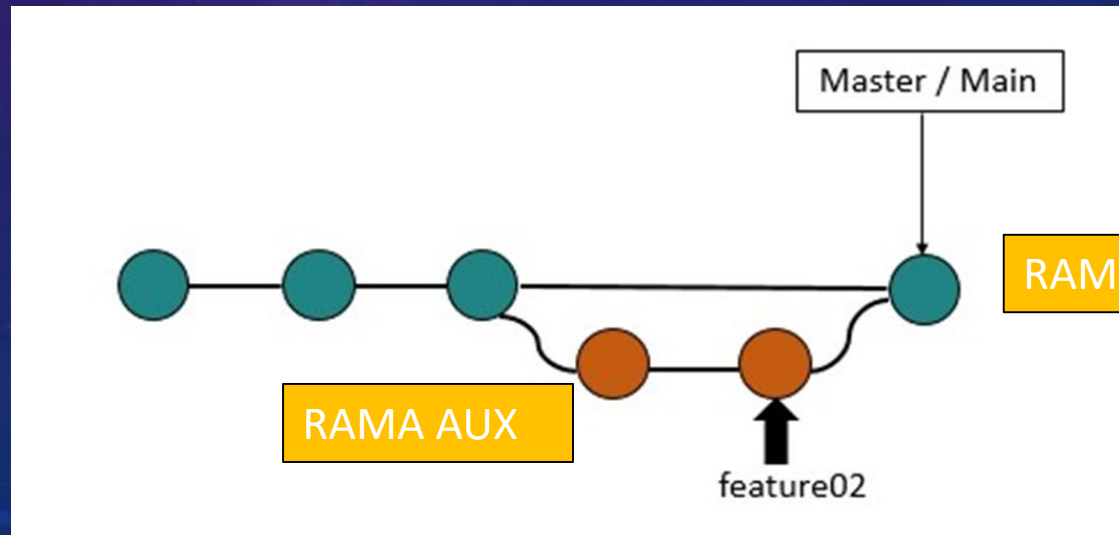
COMMIT:

- Cada vez que se guarda un trabajo, Git crea un Commit (confirmación).
- Una confirmación es una anotación o registro de los cambios realizados en un momento dado
- Si un archivo no ha cambiado de una commit a otro, Git usa el archivo almacenado anteriormente.



¿QUÉ ES UNA RAMA?

- Es un conjunto de commit que se unen entre si y que sufren cambios distintos.
- En cada cambio debe hacerse un commit .
- Las ramas auxiliares son punteros ligeros y van a administrar una separación de la rama principal.
- Una vez que se realizaron los cambios en nuestra rama auxiliar, volvemos a incorporarnos en la rama principal o Master.



- Git nos permite viajar en una línea del tiempo de nuestro proyecto y realizar cambio en el.
- Los commit crean vínculos a otras confirmaciones, formando un gráfico del historial del desarrollo, a esto llamaremos rama.
- Al trabajar en ramas auxiliares, estamos trabajando en un espacio temporal y esto me permite realizar cambios que no necesariamente se van a aplicar en nuestro proyecto.
- Y también proteger nuestro proyecto en caso de que las modificaciones sean fallidas, evitamos cualquier daño a la rama principal.

ESTADOS DE GIT:

- Es un espacio en el que se puede realizar una acción
- Espacio de trabajo.
- Área de preparación.
- Repositorio (carpeta .git).



FLUJO DE TRABAJO EN GIT:

El Flujo de trabajo básico en Git es el siguiente:

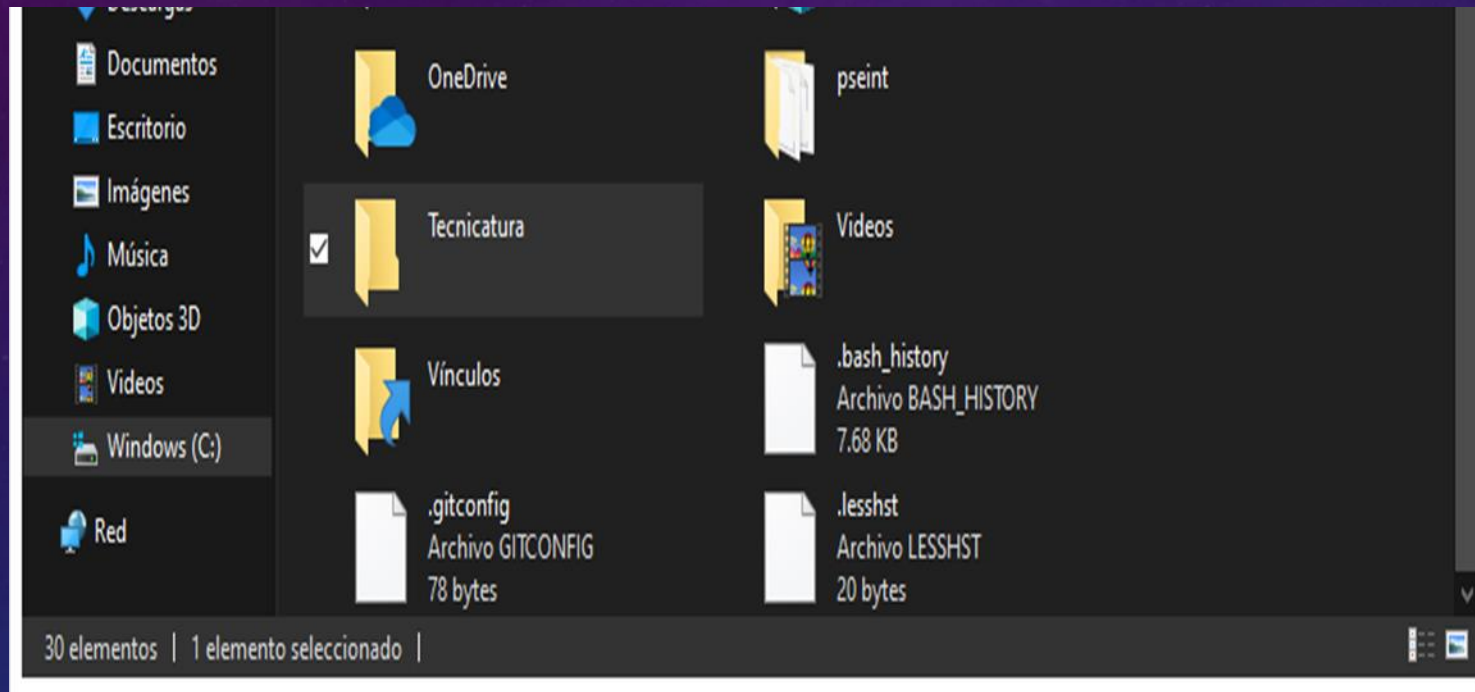
- Editas una serie de archivos en tu directorio de trabajo.
- Preparas los archivos que quieras subir al directorio, añadiéndolos a tu área de preparación.
- Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y guarda esas instantáneas de forma permanente en tu directorio de **Git**.

ESPACIO DE TRABAJO:

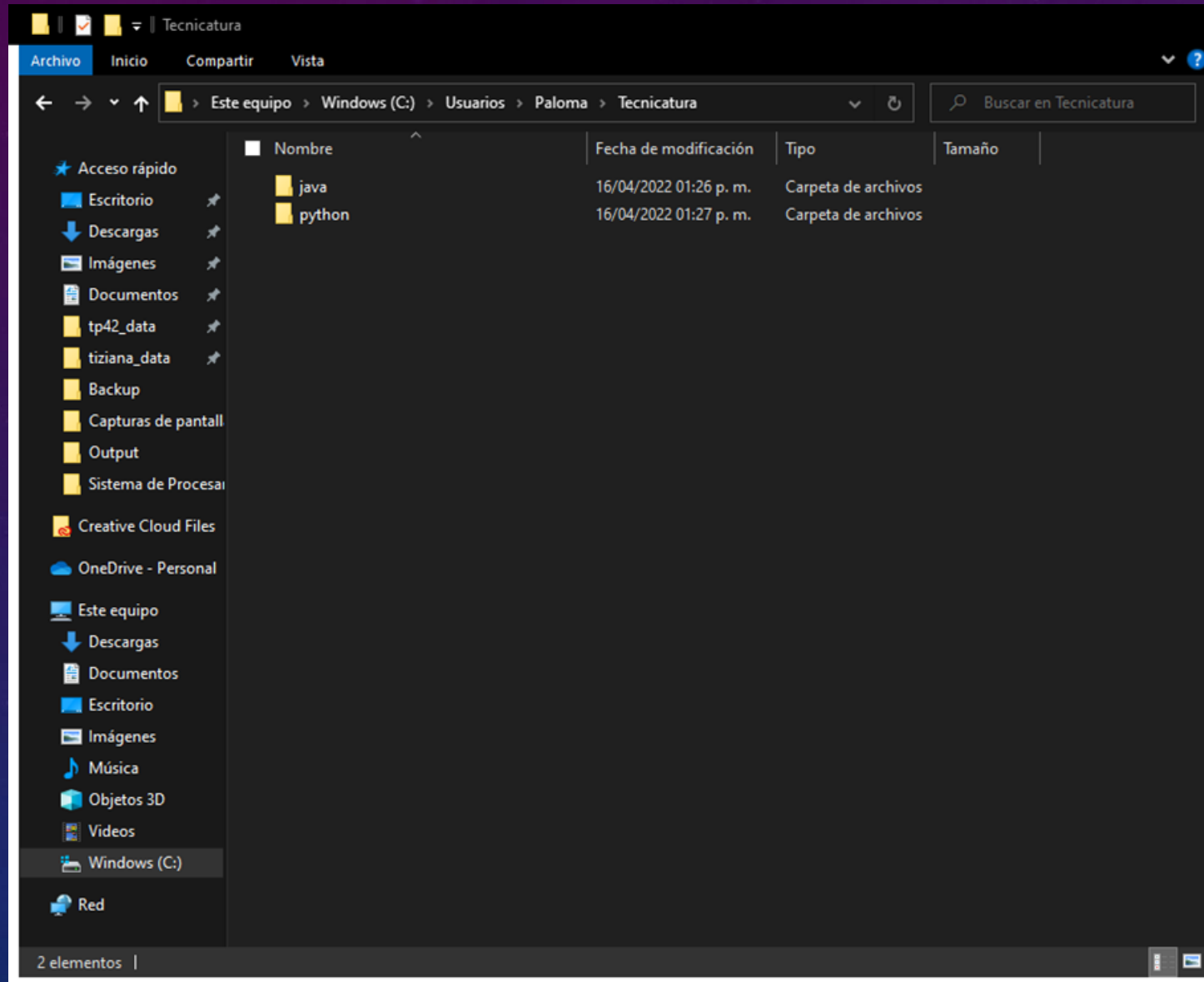
¿Cuál es el espacio de trabajo?

- Se considera como espacio de trabajo, a todos los archivos que se pueden visualizar en carpeta.
- Se ha modificado el archivo pero todavía no lo se ha confirmado en tu base de datos local.
- Git aun no se entera de la existencia de estos archivos, ya que aún no son parte de Git y tampoco es parte del espacio de trabajo.

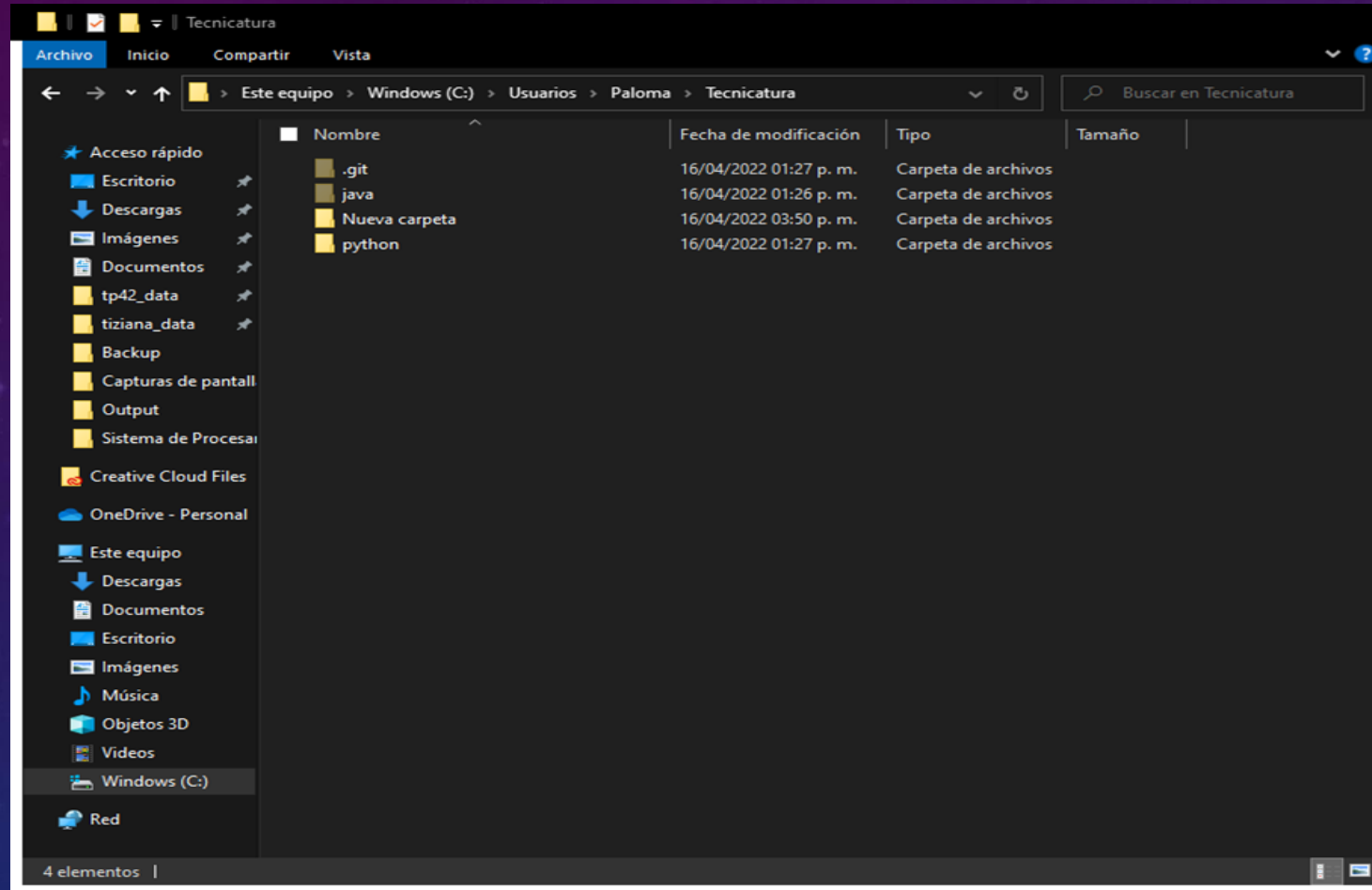
HEMOS CREADO UNA CARPETA EN EL ESPACIO DE TRABAJO O ÁREA DE TRABAJO.



SE PUEDE CREAR CARPETAS DENTRO DE OTRA CARPETA:

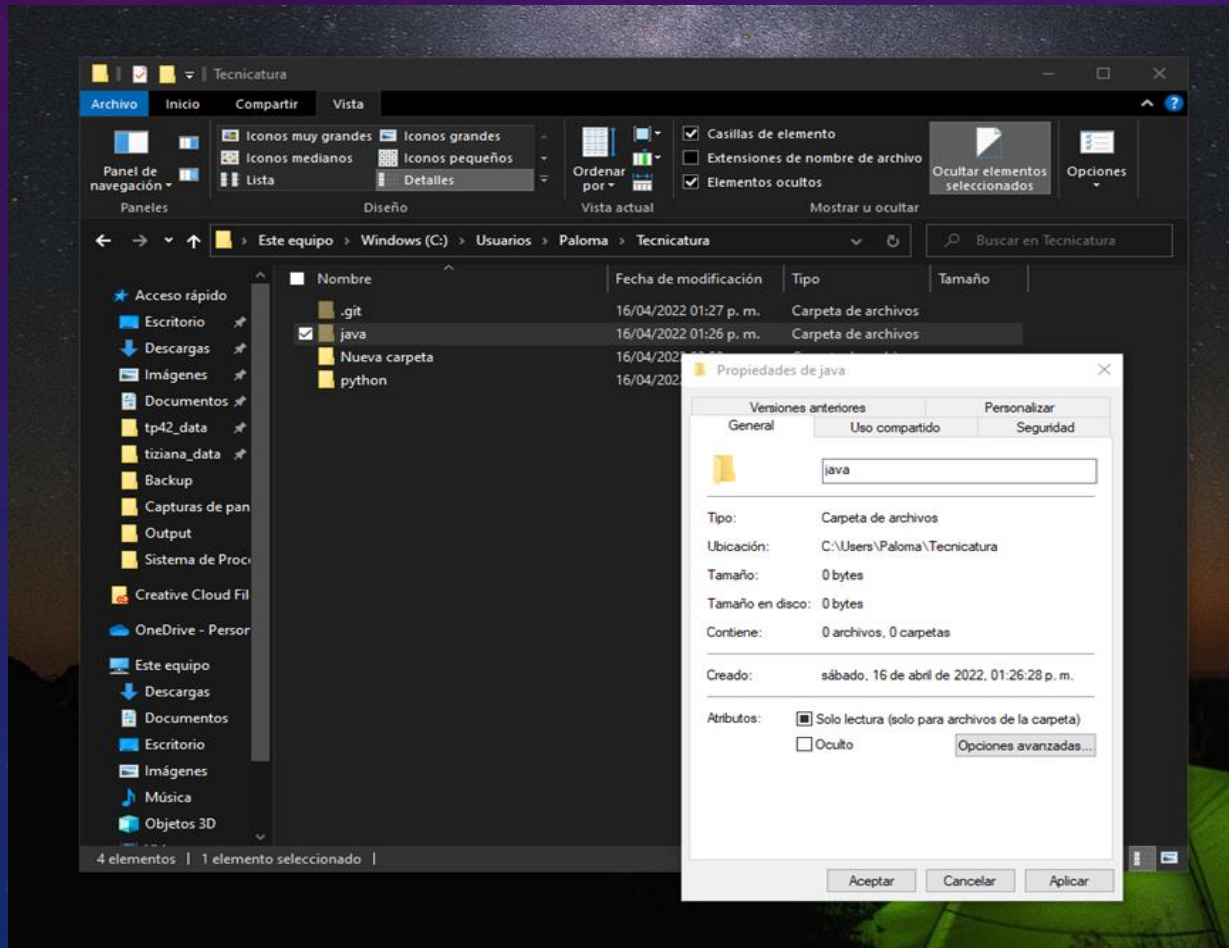


GIT NOS VA A CREAR CARPETAS OCULTAS



¿QUÉ PASA SI NO ME PARECE LAS CARPETAS CREADAS DESDE GIT ?

- *nos vamos a Vista > tildamos elementos ocultos, y nos aparecerán las carpetas ocultas o en propiedades de archivo , destilamos la opción oculta, para que nos aparezca.*



ÁREA DE PREPARACIÓN (STAGING AREA)

- Se ha marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.
- El área de preparación se inicia, cuando se agrega con el comando:
- ***git add***
- Los archivos existentes que todavía no están siendo traqueados por git.
- Acá es donde nosotros vamos a preparar todos los archivos que queremos realizar o mandar al repositorio.
- Eso es un lugar temporal, como una memoria caché donde vamos a tener nuestros archivos.
- Por último, lo pasamos al repositorio.

ÁREA DE REPOSITORIO (CARPETA GIT)

- El repositorio es el lugar en el que se almacenan los datos actualizados e históricos de cambios que realicemos a nuestro proyecto.
- Esta carpeta va a ser . Git
- Los datos del archivo están guardados en tu base de datos local.

RECORDATORIO CLASE 1

- **Paso 1** : Abrimos terminal **Git Bash**
- **Paso 2**: Creamos carpeta **Mkdir Tecnicatura Git**
- **Paso 3**: entrar a la carpeta **Cd Tecnicatura Git**
- **Paso 4**: Crear carpetas **Mkdir Java** y luego **Mkdir Python**
- **Paso 5**: **Git init** (creamos el repositorio)
- **Paso 6**: crear firma **Git config --local user.name**
- **Paso 7**: **Git config --local user.email**
- **Paso 8**: revisar la firma con **git config --list**
- **Paso 9** : salir con q o limpiar terminal o limpiar pantalla con **Clear**
- Luego de haber creado las carpetas de **java y python**

- Luego de haber creado las carpetas de **java y Python**
- Ingresar con el comando en nuestra unidad C desde la terminal de **Git Bash** y abrimos carpetas creadas.
- ***Cd TecnicaturaGit***
- ***Cd /tecnicaturaGit***
- ***Cd java***

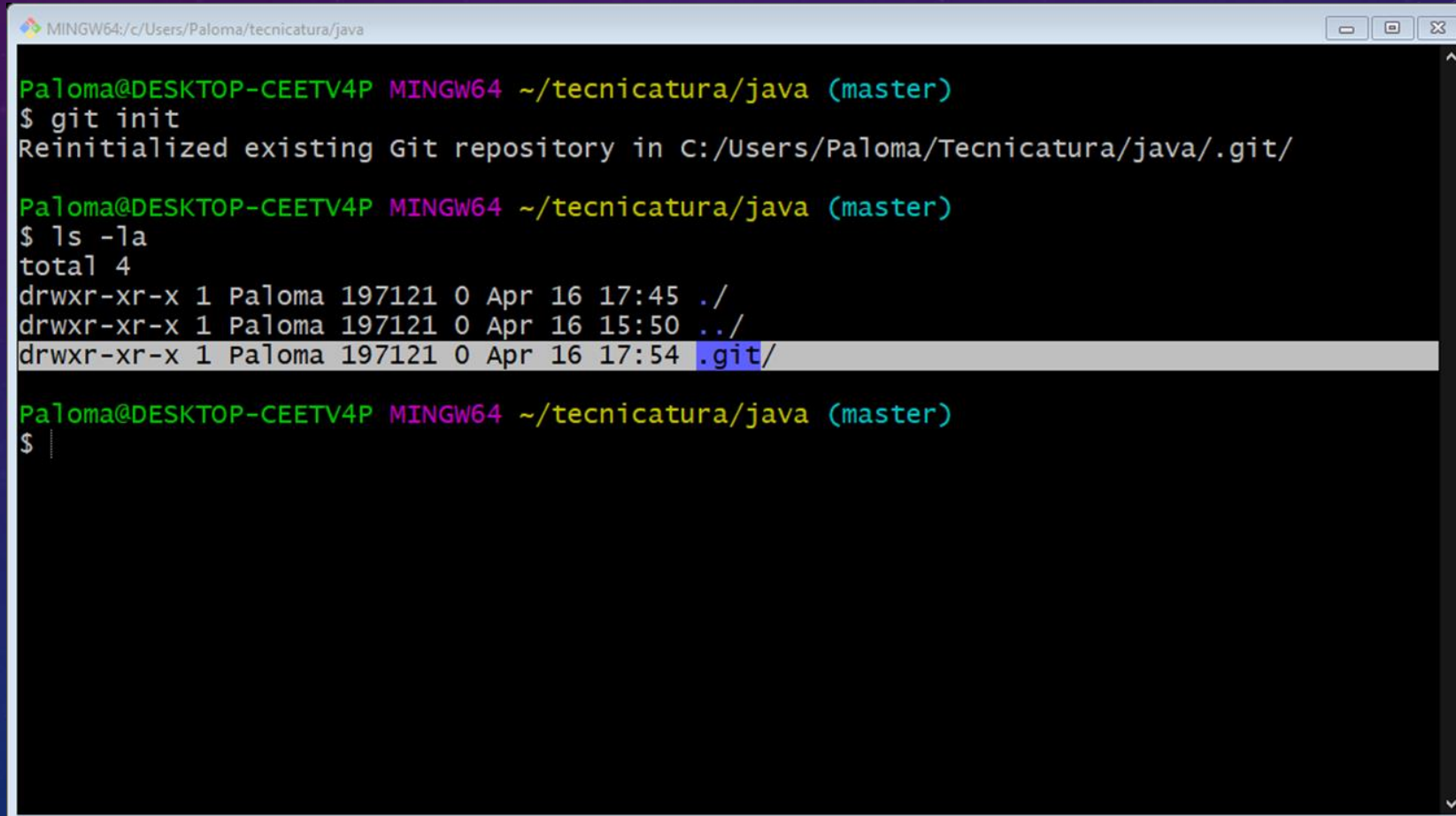
Una vez que ya iniciamos nuestro repositorio .git

- Con el comando **ls -la** (muestra los archivos y directorios ocultos en una carpeta).
- Nos aparecerán todos los archivos ocultos

«**TODOS LOS ARCHIVOS Y CARPETAS QUE EMPIEZAN CON . SON ARCHIVOS O CARPETAS OCULTOS**»

- ***NUESTRA CARPETA . GIT VA A ESTAR OCULTA.***

ESTO VA A HACER QUE NO GENERE NINGUNA MOLESTA AL USUARIO EN EL DESARROLLO DEL PROYECTO .

A screenshot of a terminal window titled 'MINGW64: c:/Users/Paloma/tecnicatura/java'. The terminal shows the following commands and output:
1. Command: `git init`
Output: `Reinitialized existing Git repository in C:/Users/Paloma/Tecnicatura/java/.git/`
2. Command: `ls -la`
Output: `total 4`
`drwxr-xr-x 1 Paloma 197121 0 Apr 16 17:45 ./`
`drwxr-xr-x 1 Paloma 197121 0 Apr 16 15:50 ../`
`drwxr-xr-x 1 Paloma 197121 0 Apr 16 17:54 .git/`
The `.git/` directory is highlighted in blue in the original image.
3. Command: The prompt `$` is shown, indicating the next command is to be entered.
The terminal window has standard Windows window controls (minimize, maximize, close) in the top right corner.

COMANDOS:

•Para limpiar terminal o pantalla:

- Control + L
- Clear
- Q

•Para borrar archivo:

- Rmdir «nombre de la carpeta»

PARA ABRIR MANUAL DE GIT:

Git --help branch

Git --help branch (Manual Git)

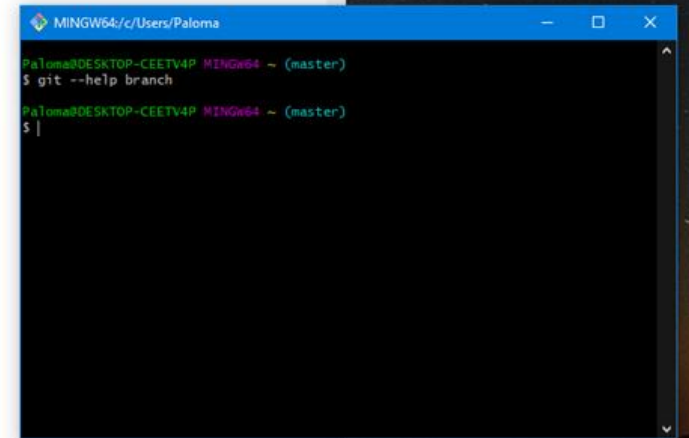
Página de manual de git-branch(1)

NOMBRE

git-branch - Listar, crear o eliminar ramas

SINOPSIS

```
rama git [--color[=<cuando>] | --no-color] [--show-current]
        [-v [--abbrev=<n> | --no-abreviatura]]
        [--columna[=<opciones>] | --sin-columna] [--sort=<clave>]
        [--merged [<commit>]] [--no-merged [<commit>]]
        [--contiene [<commit>]] [--no-contains [<commit>]]
        [--apunta-a <objeto>] [--format=<formato>]
        [(s)l remotes) | (/a)l todos]
```



```
MINGW64/c/Users/Paloma
Paloma@DESKTOP-CEETV4P MINGW64 ~ (master)
$ git --help branch
Paloma@DESKTOP-CEETV4P MINGW64 ~ (master)
$ |
```

PARA PASAR AL ESTADO DE PREPARACIÓN:

git add <archivo>

- Para guardar o aprobar (commit) ficheros en nuestro repositorio Git, primero se necesita añadirlos al área de preparación.
- Los usuarios mueven los cambios desde el espacio de trabajo al área de preparación, antes de aprobarlos/commit hacia el repositorio.
- El comando “**git add .**” permite registrar todos los cambios del directorio de trabajo.
- Si queremos eliminar el archivo de nuestra zona de preparación antes de enviarlo al repositorio.

Git rm --cached "nombre del archivo"

Git reset

- En espacio de preparación ya tenemos un archivo para enviar a nuestro repositorio y ser commiteado.

GIT STATUS:

- Permite visualizar el estado de los cambios en el directorio de trabajo y en el área de preparación comparando con el repositorio. Muestra una lista de los archivos modificados o añadidos.

Aun no tenemos commit y también nos dice que tenemos archivos sin trackear. Los archivos que están listos ser trackeados, nos **aparecen en "rojo"**

Para pasar nuestros archivos desde el área de trabajo al área de preparación utilizaremos el comando. **Git add** y el "nombre del archivo" que vamos a mover

- Luego volvemos a ejecutar. **Git status** para ver el nuevo estado del nuestro archivo

Una vez trackeado nos aparece el archivo de "**color verde**", eso significa que esta listo para ser enviado al repositorio.

New film : archivo

COMMIT:

- Una vez que nuestro archivo ya esta en nuestra área de preparación vamos a pasarlo a nuestra área de repositorio.
- Se inicia los commit (carpeta .git)
- Nos abrirá un editor de texto que se llama
- VIM

EDITOR DE TEXTO (VIM)

Una vez que estamos en el editor de texto, nos va a permitir realizar los comentarios sobre las modificaciones que realicemos a nuestro proyecto.

- En la pantalla del editor nos aparece un mensaje con # , eso quiere decir que todo lo que este en esas líneas será ignorado, solo nos sirve como contexto de lo que estamos realizando.
- Sólo en la primera línea nos aparece sin #, en esa línea escribiremos nuestro primer commit « **mi primer commit con git** » (se inicia el proyecto)
- **IMPORTANTE**: si no escribimos algo no habrá commit.

- Comandos para guardar texto

Ctrl S

- Comando para salir del editor

Ctrl X

NOS APARECE EL PRIMER COMMIT:

« **mi primer commit con git**»(se inicia el proyecto)

Nos menciona la *cantidad de archivos agregados + la cantidad de líneas utilizadas de código*.

Nos menciona *el nombre de los archivos creados*.

GIT LOG

- Con este comando veremos nuestro primer commit.
- •será el cuaderno, la "Bitácora" que mostrará en pantalla todos los commits, todos los mensajes).
- •Nos aparece el ***Autor del commit***(nombre, apellido y el correo)
- •***Fecha y hora del sistema y el mensaje que realizamos en nuestro editor de texto.***
- •Cada commit tiene un ***número identificador*** y que permite de manera ordenara realizar cada commit en mi proyecto, de manera que es imposible que se repitan.

PASOS PARA AÑADIR ARCHIVOS A NUESTRO REPOSITORIO Y CREAR NUESTROS PRIMER COMMIT:

Paso 1: Una vez que tenemos ya creadas las carpetas Tecnicatura Git y las subcarpetas Java y Python con sus respectivos archivos dentro.

Paso 2: Tecleamos ***Git status***

" veremos los archivos listos para ser enviados a nuestra área de preparación, los archivos deben figurar en ***color rojo***.

Paso 3: Tecleamos ***Git add*** .Para añadir lo que contiene ambas carpetas

En caso de querer agregar un archivo específico se debe teclear ***Git add "nombre del archivo"***

Para borrar el archivo del área de preparación ***Git reset*** .

Paso 4: nuevamente tecleamos ***Git Status***

" ahora nos aparecerán los archivos que fueron añadidos en ***color verde*** "

Paso 5 : nuestros archivos están listos para ser commiteados

Git commit

nos abre el editor de texto **Vim**

en la primera línea escribimos " **mi primer commit de git**"

guardamos nuestro mensaje del editor con : **w** o **Ctrol. S**

y **Ctrol. x** para salir del editor .

otra opción es:

Git commit --m "mi primer commit de git" y damos enter

Paso 6 : **Git log** leemos nuestro commit

ACTIVIDAD Nº1:

- *Responder cuestionario en el Aula de Campus*
- *Agregar archivos a la carpetas creadas en Git, de java y python.*

ACTIVIDAD Nº2:

- Realizar cuestionario en el Aula del Campus
- Tarea: crear nuestro primer commit con los archivos agregados a nuestra área de preparación.
- Primer comentario que deben escribir en el editor Vim o editor que tengas por defecto.
- «mi primer comentario en git»