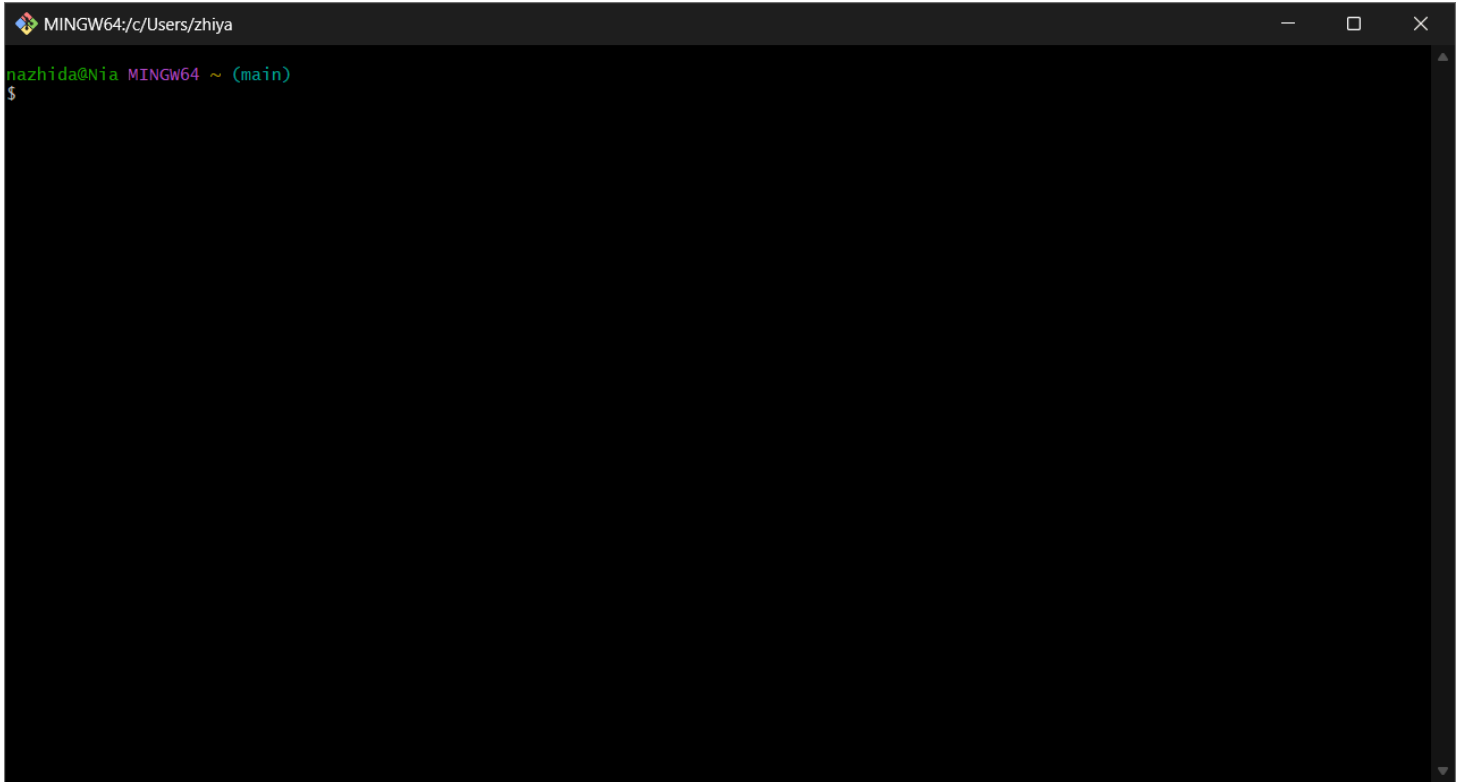


Tareas del Proyecto Git

1. Iniciar el software Git Bash. Utilizar la Shell Mingw64 siguiendo los manuales de descarga e instalación proporcionados. Mostrar un pantallazo de su pantalla de inicio.



2. Ejecutar por pantalla algún comando Linux. Probar que funciona. Escribir un comando que borre el contenido de la pantalla. Capturar el momento antes y después.

```
MINGW64:/c/Users/zhiya
nashida@Nia MINGW64 ~ (main)
$ ls -l
total 27528
drwxr-xr-x 1 nashida 197121      0 Apr 10 08:37  AppData/
drwxr-xr-x 1 nashida 197121      0 May 30 10:35  'Cisco Packet Tracer 8.2.2'/
lrwxrwxrwx 1 nashida 197121    28 Apr 10 08:36  'Configuración local' -> /c/Users/zhiya/AppData/Local/
drwxr-xr-x 1 nashida 197121      0 Apr 10 10:08  Contacts/
lrwxrwxrwx 1 nashida 197121    58 Apr 10 08:36  Cookies -> /c/Users/zhiya/AppData/Local/Microsoft/windows/INetCookies/
drwxr-xr-x 1 nashida 197121      0 May  9 17:11  'Creative Cloud Files Personal Account kitkat.starun@gmail.com AF4F2CBB5C530F500A495E15@A
dobeID' /
lrwxrwxrwx 1 nashida 197121    30 Apr 10 08:36  'Datos de programa' -> /c/Users/zhiya/AppData/Roaming/
drwxr-xr-x 1 nashida 197121      0 Jun 18 22:50  Desktop/
drwxr-xr-x 1 nashida 197121      0 Jun  4 09:43  Documents/
drwxr-xr-x 1 nashida 197121      0 Jun 18 22:50  Downloads/
lrwxrwxrwx 1 nashida 197121    66 Apr 10 08:36  'Entorno de red' -> '/c/Users/zhiya/AppData/Roaming/Microsoft/windows/Network Shortcuts'/
drwxr-xr-x 1 nashida 197121      0 Apr 21 12:53  Favorites/
drwxr-xr-x 1 nashida 197121      0 Apr 26 12:43  IdeaProjects/
drwxr-xr-x 1 nashida 197121      0 Apr 27 19:49  IdeaSnapshots/
lrwxrwxrwx 1 nashida 197121    66 Apr 10 08:36  Impresoras -> '/c/Users/zhiya/AppData/Roaming/Microsoft/Windows/Printer Shortcuts'/
-rw-r--r-- 1 nashida 197121 5666516 Apr  9 11:12  JetBrainsMono.tar.xz
drwxr-xr-x 1 nashida 197121      0 Apr 10 10:08  Links/
lrwxrwxrwx 1 nashida 197121    59 Apr 10 08:36  'Menú Inicio' -> '/c/Users/zhiya/AppData/Roaming/Microsoft/Windows/Start Menu'/
lrwxrwxrwx 1 nashida 197121    24 Apr 10 08:36  'Mis documentos' -> /c/Users/zhiya/Documents/
drwxr-xr-x 1 nashida 197121      0 Apr 10 10:08  Music/
-rw-r--r-- 1 nashida 197121 16252928 Jun 19 01:09  NTUSER.DAT
-rw-r--r-- 1 nashida 197121    65536 Apr 10 08:36  NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TM.b1f
-rw-r--r-- 1 nashida 197121    524288 Apr 10 08:36  NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TMContainer00000000000000000001.regtran
s-ms
-rw-r--r-- 1 nashida 197121    524288 Apr 10 08:36  NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TMContainer00000000000000000002.regtran
s-ms
drwxr-xr-x 1 nashida 197121      0 Mar  6 12:41  OneDrive/
drwxr-xr-x 1 nashida 197121      0 May 22 12:13  'OneDrive - IMF Smart Education'/
drwxr-xr-x 1 nashida 197121      0 May 22 12:13  OneDriveCloudTemp/
drwxr-xr-x 1 nashida 197121      0 Nov 29 2023  Oracle/
drwxr-xr-x 1 nashida 197121      0 Apr 10 10:08  Pictures/
lrwxrwxrwx 1 nashida 197121    58 Apr 10 08:36  Plantillas -> /c/Users/zhiya/AppData/Roaming/Microsoft/Windows/Templates/
drwxr-xr-x 1 nashida 197121      0 Jun  6 11:49  Postman/
```

```
MINGW64:/c/Users/zhiya
nashida@Nia MINGW64 ~ (main)
$ clear
```

```
nazhida@Nia MINGW64 ~ (main)
$ |
```

3. Visualiza por pantalla varias combinaciones útiles del comando CTRL. Se proporciona una lista de las combinaciones de las teclas más usadas. L C U K A E. Probarlas todas.

- Algunas combinaciones útiles de teclas CTRL incluyen:
 - CTRL + L : Limpia la pantalla.

A screenshot of a MINGW64 terminal window. The title bar shows the path 'MINGW64:/c/Users/zhiya'. The terminal content shows the prompt 'nazhida@Nia MINGW64 ~ (main)' followed by the command '\$ sdfasdfsdfsf' entered on the next line. The rest of the terminal is empty.

- CTRL + c : Termina el comando actual.



```
MINGW64/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ adfdfad^C
nazhida@Nia MINGW64 ~ (main)
$
```

- CTRL + U : Borra la línea de comando actual.



```
MINGW64/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ sdfasdfsdfs^C
nazhida@Nia MINGW64 ~ (main)
$ fasdfs
```

- CTRL + K : Borra desde el cursor hasta el final de la línea.



A terminal window titled 'MINGW64/c/Users/zhiya' with standard window controls. The prompt is 'nazhida@Nia MINGW64 ~ (main)'. The first line shows the command 'sdfasdfsdf^C\sd^C' where the cursor is at the end. The second line shows the same command 'sdfasdfsdf^C\sd^C' but the cursor has moved to the beginning of the line.

```
MINGW64/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ sdfasdfsdf^C\sd^C
nazhida@Nia MINGW64 ~ (main)
$ sdfasdfsdf^C\sd^C|
```

- CTRL + A : Mueve el cursor al inicio de la línea.



A terminal window titled 'MINGW64/c/Users/zhiya' with standard window controls. The prompt is 'nazhida@Nia MINGW64 ~ (main)'. The command 'sdfasdfsdf^C\sd^C' is shown with the cursor at the end of the line.

```
MINGW64/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ sdfasdfsdf^C\sd^C
```

- CTRL + E : Mueve el cursor al final de la línea.

A terminal window titled 'MINGW64:/c/Users/zhiya' with standard window controls. The prompt is 'nazhida@Nia MINGW64 ~ (main)'. The command '\$ sdfasdfsdfsf^C\sd^C]' is entered, with the second line being a truncated version of the first.

```
MINGW64:/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ sdfasdfsdfsf^C\sd^C]
```

4. Averiguar la versión de Git que se está usando actualmente en vuestro PC. Para ello ejecutar un comando Git. Y luego opcionalmente actualizar la versión a la última.

- Para verificar la versión actual de Git, usa:

```
git --version
```

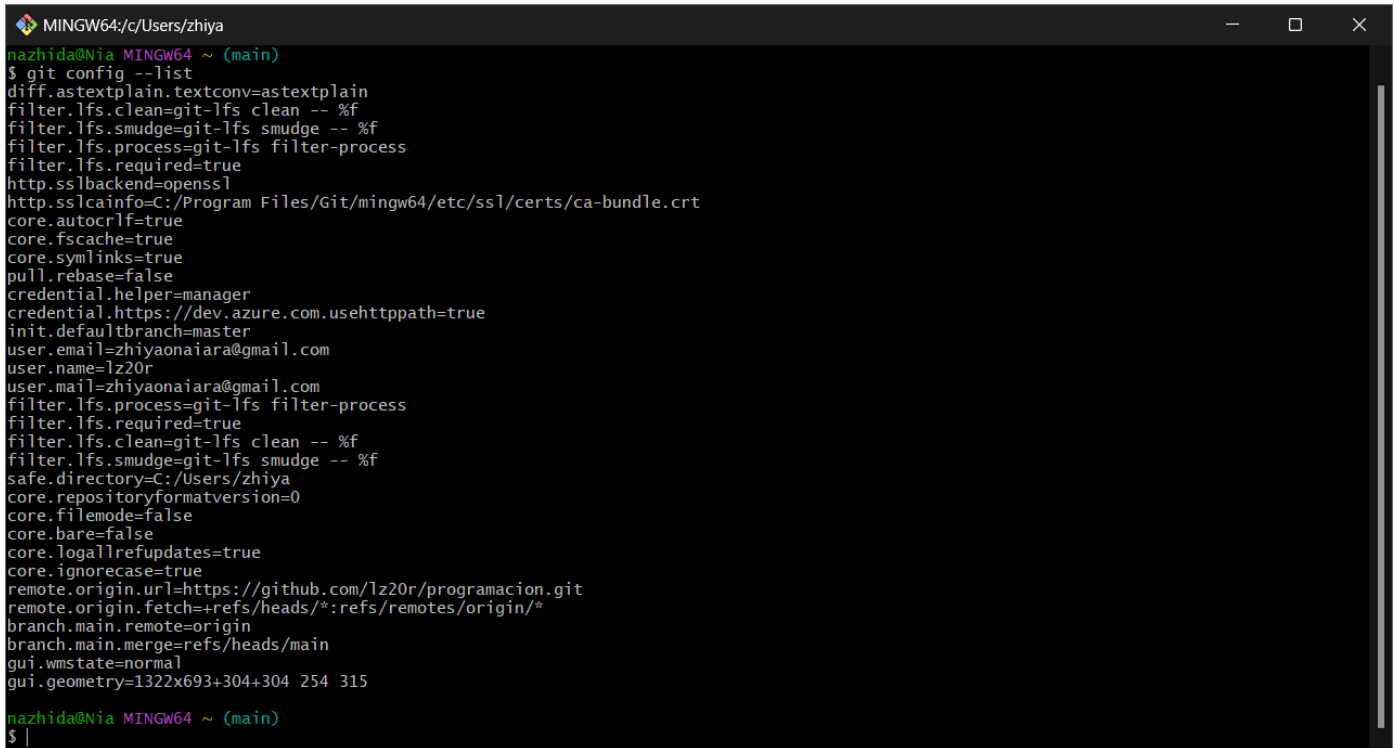
A terminal window titled 'MINGW64:/c/Users/zhiya' with standard window controls. The prompt is 'nazhida@Nia MINGW64 ~ (main)'. The command '\$ git --version' is entered, and the output 'git version 2.45.1.windows.1' is displayed. The prompt '\$' is shown again on the next line.

```
MINGW64:/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ git --version
git version 2.45.1.windows.1
nazhida@Nia MINGW64 ~ (main)
$
```

5. Mostrar por pantalla la lista de configuración de la sesión actual de Git. Todo lo que se pide es por comandos. Nos servirá para ver la información inicial preconfigurada.

- Para ver la configuración actual de Git, usa:

```
git config --list
```

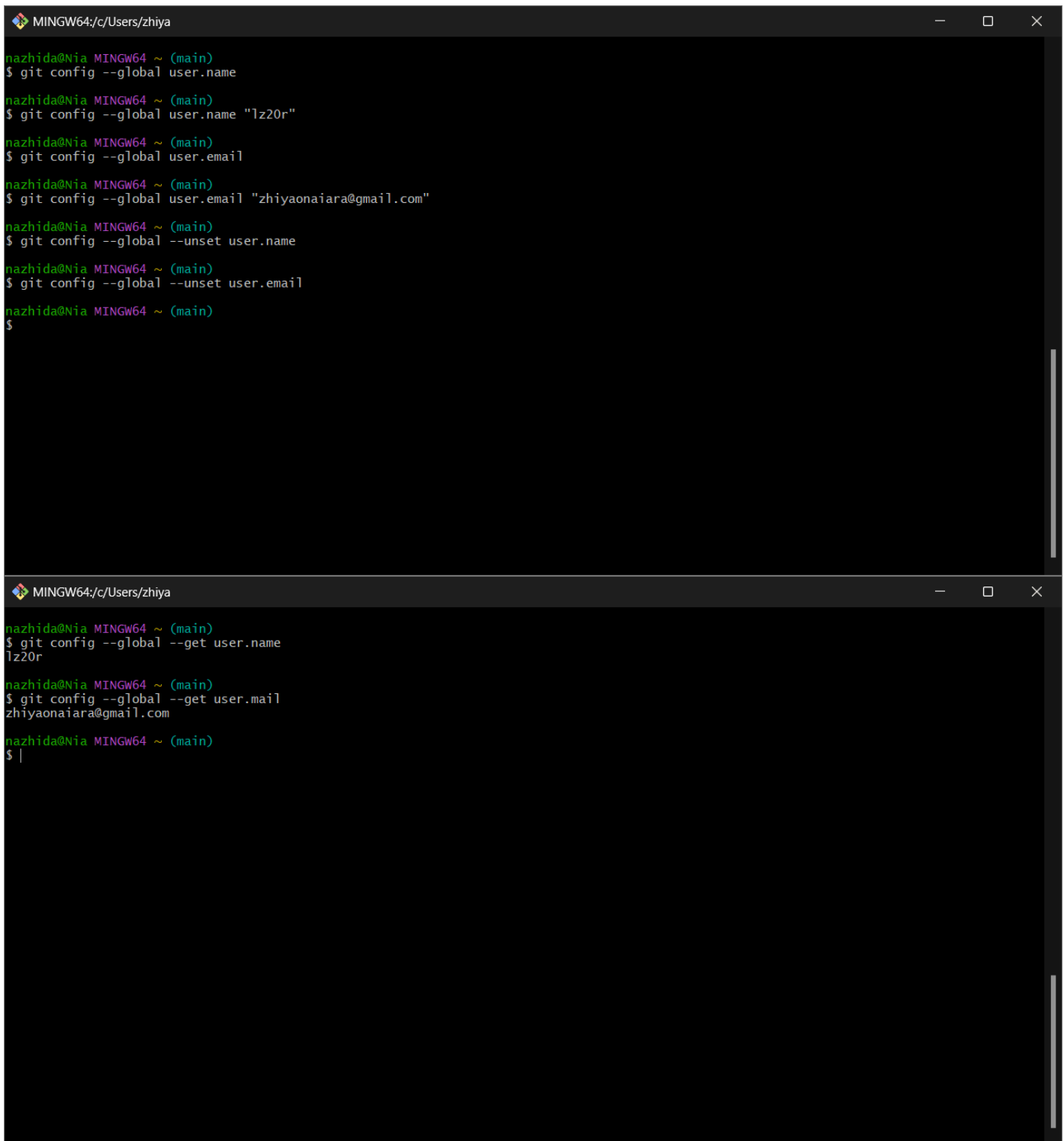
A screenshot of a terminal window titled 'MINGW64: c:/Users/zhiya'. The prompt is 'nazhida@Nia MINGW64 ~ (main)'. The command '\$ git config --list' has been executed, resulting in a long list of configuration variables and their values. The output includes settings for diff, filter.lfs, http.sslbackend, http.sslcainfo, core.autocrlf, core.fscache, core.symlinks, pull.rebase, credential.helper, credential.https, init.defaultbranch, user.email, user.name, user.mail, filter.lfs.process, filter.lfs.required, filter.lfs.clean, filter.lfs.smudge, safe.directory, core.repositoryformatversion, core.filemode, core.bare, core.logallrefupdates, core.ignorecase, remote.origin.url, remote.origin.fetch, branch.main.remote, branch.main.merge, gui.wmstate, and gui.geometry. The prompt returns to '\$ |' at the bottom.

```
MINGW64: c:/Users/zhiya
nazhida@Nia MINGW64 ~ (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=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=zhiyaonaiara@gmail.com
user.name=lz20r
user.mail=zhiyaonaiara@gmail.com
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
safe.directory=C:/Users/zhiya
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
remote.origin.url=https://github.com/lz20r/programacion.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
gui.wmstate=normal
gui.geometry=1322x693+304+304 254 315
nazhida@Nia MINGW64 ~ (main)
$ |
```

6. Verificar en la configuración inicial que no hay nombre de usuario. Y que tampoco hay correo electrónico asociado al usuario. Configurar de forma global un usuario y un email asociado. Verificar consultando la lista de configuración que se han creado ambos. Probar luego a borrar tanto usuario como email. Luego volver a crear los mismos de nuevo.

- Configurar globalmente un usuario y un correo:

```
git config --global user.name "TuNombre"
git config --global user.email "tuemail@example.com"
git config --global --unset user.name      # Para borrar el usuario
git config --global --unset user.email     # Para borrar el email
```



```
MINGW64:/c/Users/zhiya

nazhida@Nia MINGW64 ~ (main)
$ git config --global user.name

nazhida@Nia MINGW64 ~ (main)
$ git config --global user.name "lz20r"

nazhida@Nia MINGW64 ~ (main)
$ git config --global user.email

nazhida@Nia MINGW64 ~ (main)
$ git config --global user.email "zhiyaonaiara@gmail.com"

nazhida@Nia MINGW64 ~ (main)
$ git config --global --unset user.name

nazhida@Nia MINGW64 ~ (main)
$ git config --global --unset user.email

nazhida@Nia MINGW64 ~ (main)
$

MINGW64:/c/Users/zhiya

nazhida@Nia MINGW64 ~ (main)
$ git config --global --get user.name
lz20r

nazhida@Nia MINGW64 ~ (main)
$ git config --global --get user.mail
zhiyaonaiara@gmail.com

nazhida@Nia MINGW64 ~ (main)
$ |
```

7. Ejecutar un comando de Linux para mostrar por pantalla la dirección del directorio actual

- Para mostrar la ruta del directorio actual:

`pwd`


```
MINGW64:/c/Users/zhiya

nazhida@Nia MINGW64 ~ (main)
$ pwd
/c/Users/zhiya

nazhida@Nia MINGW64 ~ (main)
$
```

8. Listar con un comando Linux (nunca uséis para ningún apartado instrucciones de CMD) el directorio actual, es decir, mostrar todos los archivos, carpetas, etc.

- Para listar todos los archivos y carpetas del directorio actual:

```
ls
```

```
MINGW64:/c/Users/zhiya

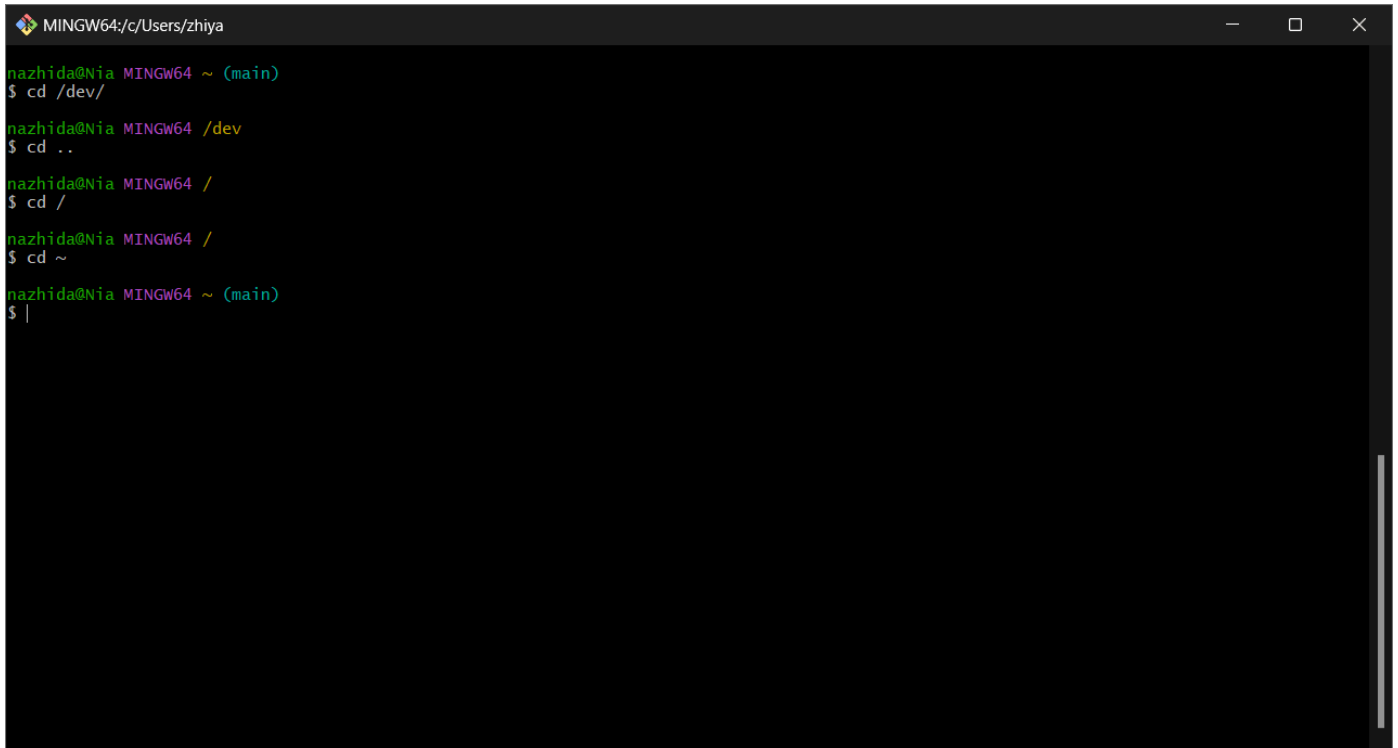
nazhida@Nia MINGW64 ~ (main)
$ ls
AppData/
'Cisco Packet Tracer 8.2.2'/
'Configuración local'@
Contacts/
Cookies@
'Creative Cloud Files Personal Account kitkat.starun@gmail.com AF4F2CBB5C530F500A495E15@AdobeID'/
'Datos de programa'@
Desktop/
Documents/
Downloads/
'Entorno de red'@
Favorites/
IdeaProjects/
IdeaSnapshots/
Impresoras@
JetBrainsMono.tar.xz
Links/
'Menú Inicio'@
'Mis documentos'@
Music/
NTUSER.DAT
NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TM.blf
NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TMContainer00000000000000000001.regtrans-ms
NTUSER.DAT{a93e4df7-f70c-11ee-b72c-cbfcd1de25d3}.TMContainer00000000000000000002.regtrans-ms
OneDrive/
'OneDrive - IMF Smart Education'/
OneDriveCloudTemp/
Oracle/
Pictures/
Plantillas@
Postman/
ProjectGenerator/
README.md
Reciente@
'Saved Games'/
Searches/
SendTo@
Videos/
'VirtualBox VMs'/
anse1/
bounce/
cinammonbilling/
composer.json
composer.lock
deportes/
desktop.ini
dir
eclipse/
iCloudDrive/
inst.ini
node_modules/
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini
nuuid.ini
output.txt
outut.txt
package-lock.json
package.json
prueba/
userid.ini
vendor/
vmlogs/
welcome-to-docker/

nazhida@Nia MINGW64 ~ (main)
$
```

9. Moverse a través de los directorios de tu disco duro principal o unidad C. Tanto a la derecha en el árbol de directorio. Como luego a la izquierda volviendo al directorio inicial.

- Para moverse a través de los directorios:

```
cd path/to/directory  
cd ..
```

A screenshot of a MINGW64 terminal window. The title bar shows the path 'MINGW64:/c/Users/zhiya'. The terminal content shows a series of commands and their outputs: 'nazhida@Nia MINGW64 ~ (main)' followed by '\$ cd /dev/' which outputs 'nazhida@Nia MINGW64 /dev', then '\$ cd ..' which outputs 'nazhida@Nia MINGW64 /', then '\$ cd /' which outputs 'nazhida@Nia MINGW64 /', then '\$ cd ~' which outputs 'nazhida@Nia MINGW64 ~ (main)', and finally '\$ |' which shows a cursor. The background is black and the text is green and white.

```
MINGW64:/c/Users/zhiya  
nazhida@Nia MINGW64 ~ (main)  
$ cd /dev/  
nazhida@Nia MINGW64 /dev  
$ cd ..  
nazhida@Nia MINGW64 /  
$ cd /  
nazhida@Nia MINGW64 /  
$ cd ~  
nazhida@Nia MINGW64 ~ (main)  
$ |
```

10. Utilizar un comando que te lleve a la raíz del usuario. Suele ser la misma en la que inicia el software Bash. Y otro comando que te lleve a la raíz del sistema. Suele ser la unidad C.

- Para ir a la raíz del usuario:

```
cd ~
```

- Para ir a la raíz del sistema:

```
cd /
```

```
MINGW64:/c/Users/zhiya
nazhida@Nia MINGW64 ~ (main)
$ cd /dev/
nazhida@Nia MINGW64 /dev
$ cd ..
nazhida@Nia MINGW64 /
$ cd /
nazhida@Nia MINGW64 /
$ cd ~
nazhida@Nia MINGW64 ~ (main)
$ |
```

11. Dirígete a la raíz del sistema operativo, para Windows es disco donde este instalado, y crea una carpeta que se llame como tu correo electrónico, sin arroba, y sin el texto a la derecha de arroba. Dentro de esa carpeta crea una subcarpeta que se llame Git en minúsculas. Y ejecuta una instrucción de Linux que muestre todos los elementos que contengan la palabra Git.

- Para ir a la raíz del usuario:

```
cd ~
```

- Para ir a la raíz del sistema:

```
cd /
```

- Para crear un directorio:

```
mkdir [directorio]
```

- Para encontrar un archivo e imprimirlo

```
find ./ -name "*nombre del archivo" -print
```

```
MINGW64:/naiara

nazhida@Nia MINGW64 ~ (main)
$ cd /

nazhida@Nia MINGW64 /
$ mkdir naiara

nazhida@Nia MINGW64 /
$ ls
LICENSE.txt      bin/   dev/   git-bash.exe*  mingw64/  proc/  unins000.dat  unins000.msg
ReleaseNotes.html cmd/   etc/   git-cmd.exe*   naiara/   tmp/   unins000.exe*  usr/

nazhida@Nia MINGW64 /
$ cd naiara/

nazhida@Nia MINGW64 /naiara
$ mkdir git

nazhida@Nia MINGW64 /naiara
$ ls
git/

nazhida@Nia MINGW64 /naiara
$ find ./ -name "*git" -print
./git

nazhida@Nia MINGW64 /naiara
$ |
```

12. Borra el directorio correspondiente a la carpeta Git. En todos los casos de este trabajo, cuando borramos algún elemento que luego necesitamos, repetimos los pasos para poder crearlo nuevamente. De modo que podamos continuar con el desarrollo de este proyecto

- Para borrar el directorio:

```
rm -rf ./git/
```

```
MINGW64:/naiara

nazhida@Nia MINGW64 /naiara
$ rm -rf ./git/

nazhida@Nia MINGW64 /naiara
$ ls

nazhida@Nia MINGW64 /naiara
$ mkdir git

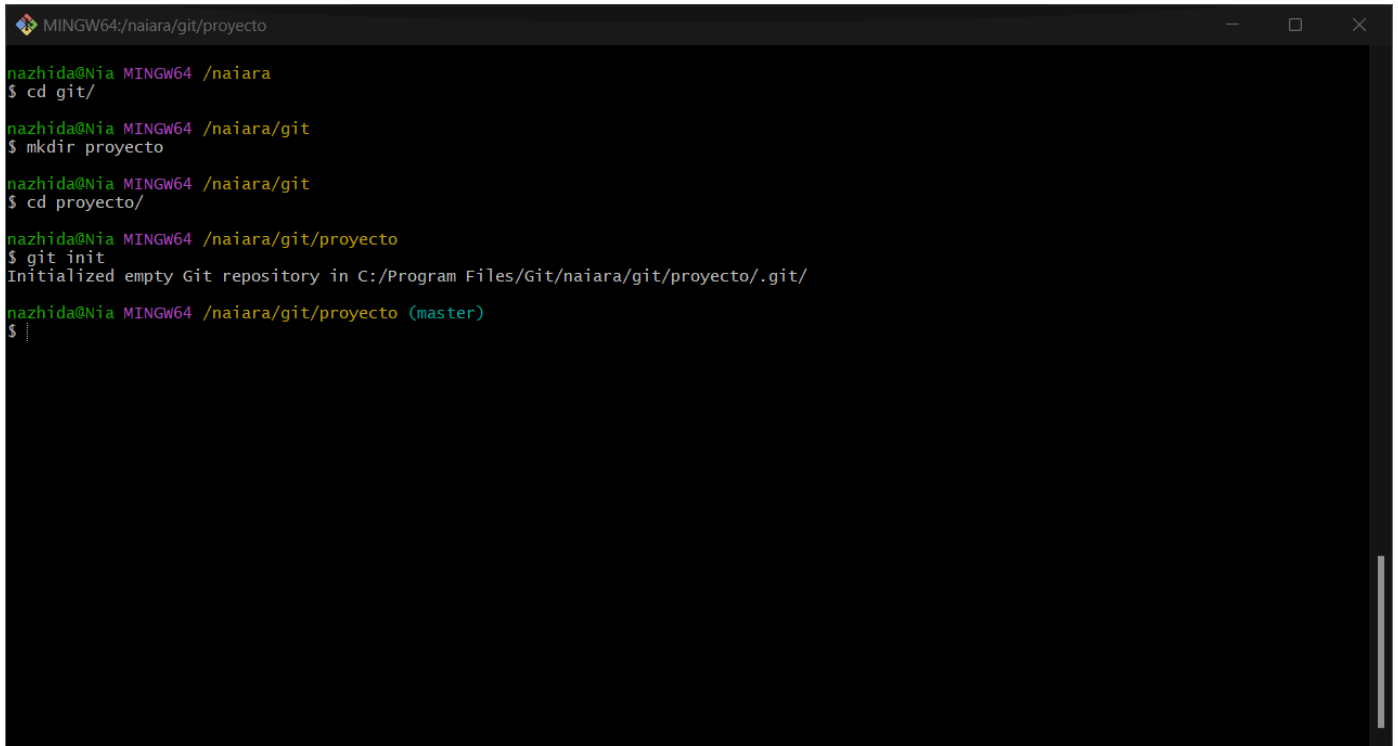
nazhida@Nia MINGW64 /naiara
$ ls
git/

nazhida@Nia MINGW64 /naiara
$ |
```

13. Entra en la carpeta Git. Y genera una subcarpeta que se llame proyecto. Una vez dentro de esta subcarpeta ejecuta la orden que inicializar un repositorio Git en la ruta actual.

- Para inicializar un repositorio Git:

```
cd  
git init
```



```
MINGW64:/naiara/git/proyecto  
nashida@Nia MINGW64 /naiara  
$ cd git/  
nashida@Nia MINGW64 /naiara/git  
$ mkdir proyecto  
nashida@Nia MINGW64 /naiara/git  
$ cd proyecto/  
nashida@Nia MINGW64 /naiara/git/proyecto  
$ git init  
Initialized empty Git repository in C:/Program Files/Git/naiara/git/proyecto/.git/  
nashida@Nia MINGW64 /naiara/git/proyecto (master)  
$ |
```

14. Ahora borra de forma completa dicho repositorio. Hazlo con una única instrucción. Y asegure luego de que ya no existe. Recuerda como ya se dijo luego debes iniciarlo de nuevo.

- Para borrar un repositorio Git completamente:

```
rm -rf ./git/
```

```
MINGW64:/naiara/git/proyecto
nashida@Nia MINGW64 /naiara
$ rm -rf ./git/
nashida@Nia MINGW64 /naiara
$ ls
nashida@Nia MINGW64 /naiara
$ mkdir git
nashida@Nia MINGW64 /naiara
$ cd git/
nashida@Nia MINGW64 /naiara/git
$ mkdir proyecto
nashida@Nia MINGW64 /naiara/git
$ cd proyecto/
nashida@Nia MINGW64 /naiara/git/proyecto
$ git init
Initialized empty Git repository in C:/Program Files/Git/naiara/git/proyecto/.git/
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

15. Muestra con una orden el estado actual del repositorio. Debe de aparecer un mensaje de respuesta indicando que no hay ningún commit hecho. Muestra el mismo estado, pero añade un modificador a la orden para que la respuesta sea resumida, respecto a la anterior.

- Para una respuesta resumida:

```
git status -s
```

- Para mostrar el estado actual del repositorio:

```
git status
```

```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status -s

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

16. Visualiza la información actual del historial o registro de tu repositorio. Se llama log

- Para ver el historial o log:

```
git log
```

```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
fatal: your current branch 'master' does not have any commits yet

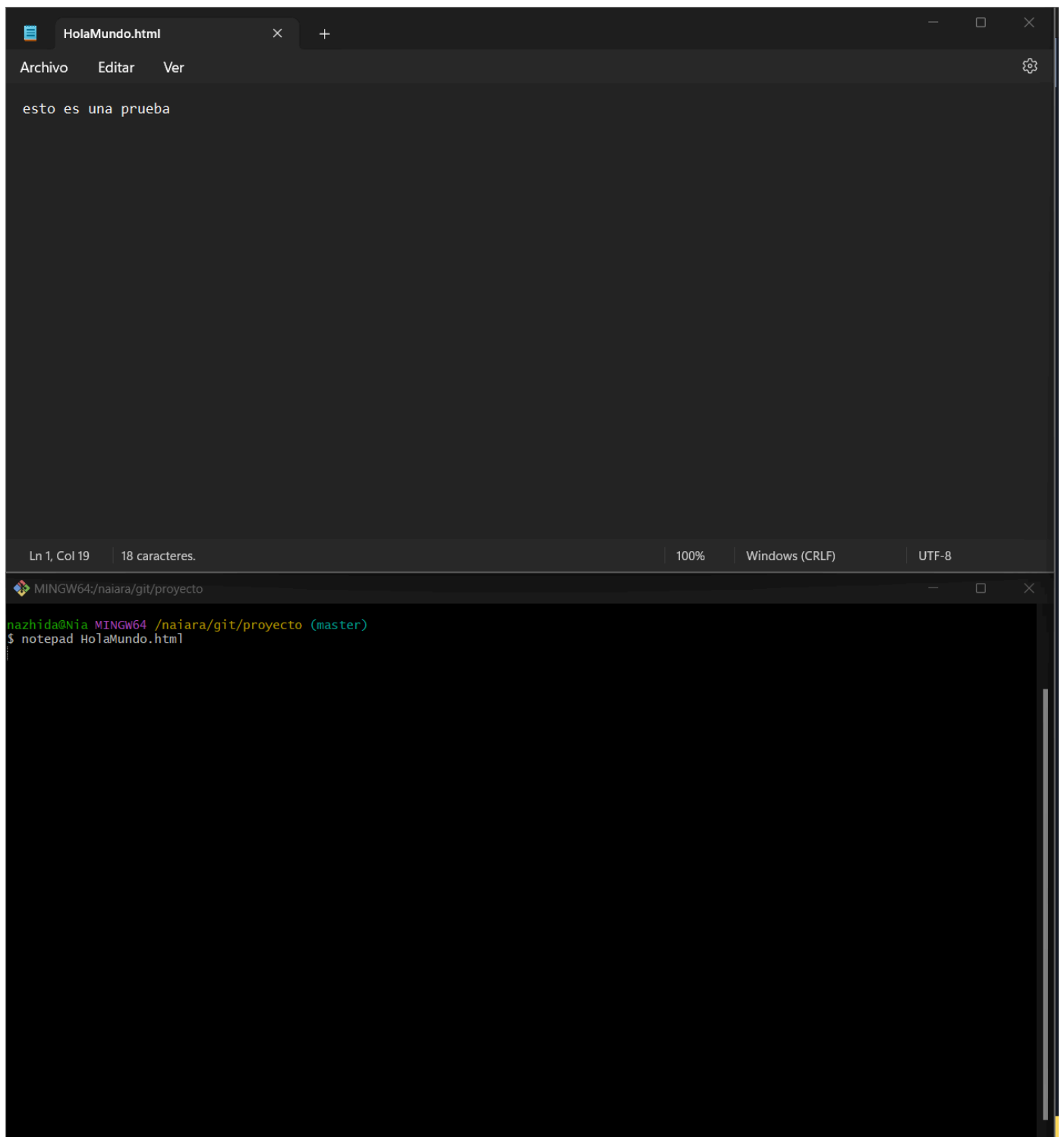
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

17. Ejecuta un editor de código desde línea de comandos. Puede ser alguno que tengas instalado en tu maquina por defecto. Por ejemplo, el bloc de notas. Una vez abierto crea un documento que sea una página web Hola Mundo. Salva el documento. Y comprueba existe. Luego bórralo, y recuerda, siempre que efectúas una acción en Git se comprueba si se hizo.

- Abre un editor y crea un archivo "Hola Mundo":

```
notepad hola_mundo.html
```

- Guarda el archivo y verifica que exista.



```
ls
rm hola_mundo.html
ls
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ notepad HolaMundo.html
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ ls
HolaMundo.html
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ rm HolaMundo.html
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ ls
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

18. Con el fichero de la página web generado, repitiendo los pasos. Comprueba el estado de tu repositorio. Debes obtener una respuesta acerca de tu fichero indicando que no es parte del seguimiento. Haz también la comprobación del estado, pero de forma resumida.

```
git status
git status -s
```

```
MINGW64:/naiara/git/proyecto
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ notepad HolaMundo.html
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        HolaMundo.html

nothing added to commit but untracked files present (use "git add" to track)
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status -s
?? HolaMundo.html
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

19. Añade el fichero al seguimiento del repositorio Git. Se puede añadir el fichero de forma individual o colectiva. Utiliza las dos instrucciones si esto fuera posible. Si no lo es, utiliza una instrucción y explica la otra.

- Para añadir el archivo al seguimiento:

```
git add hola_mundo.html
```

- Para añadir todos los archivos:

```
git add .
```

```
MINGW64:/naiara/git/proyecto
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git add HolaMundo.html
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git add .
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

20. Seguidamente visualiza el estado de tu repositorio, de forma extendida y abreviada, tras añadir el fichero. Aparecerá un mensaje de respuesta marcando los cambios para commit.

- Verificar el estado extendido y abreviado:

```
git status
git status -s
```

```
MINGW64:/naiara/git/proyecto
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   HolaMundo.html

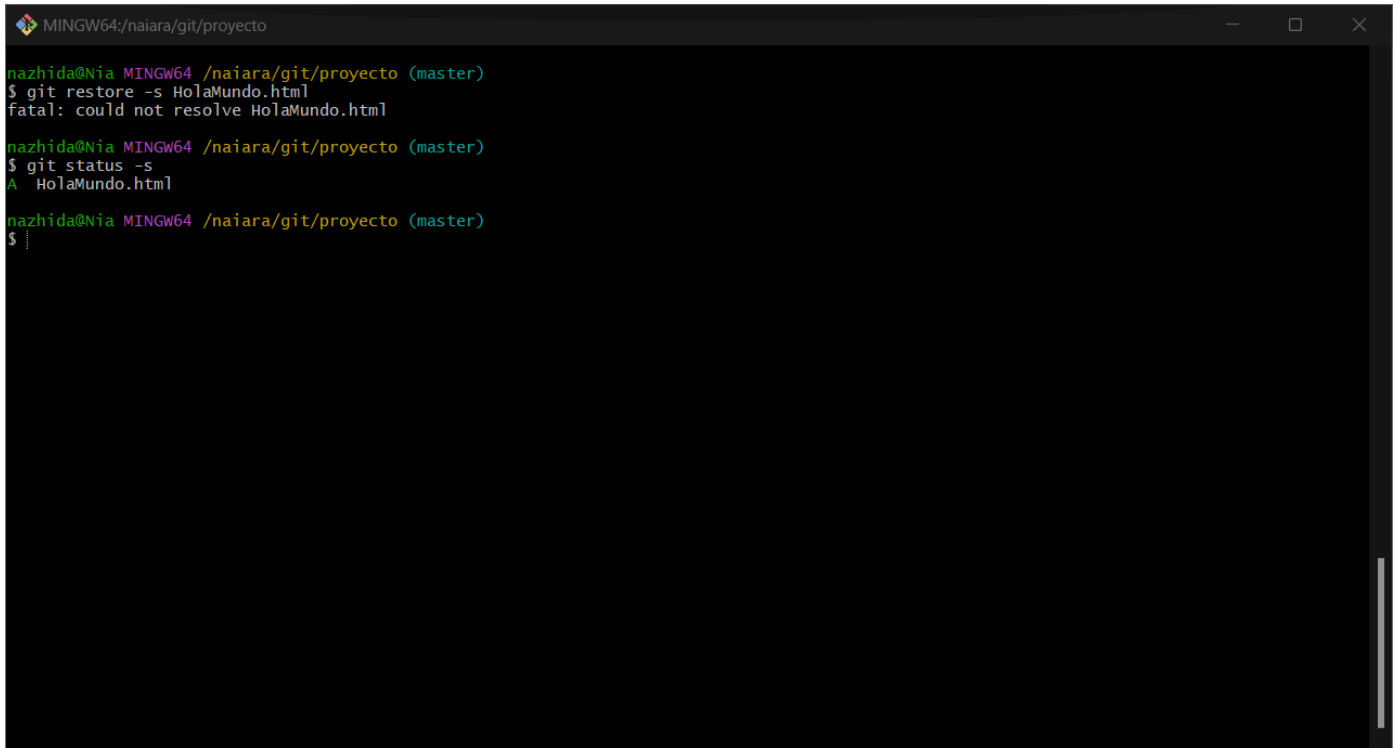
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status -s
A  HolaMundo.html

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

21. Ahora revierte los cambios, es decir, saca del seguimiento a ese fichero. Verifica el estado de tu repositorio para ver que el fichero esta fuera del seguimiento.

- Para sacar el archivo del seguimiento:

```
git reset hola_mundo.html
```

A screenshot of a terminal window titled 'MINGW64:/naiara/git/proyecto'. The terminal shows the following commands and output:

```
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git restore -s HolaMundo.html
fatal: could not resolve HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status -s
A  HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

22. Realiza un commit del fichero, que debes previamente haber añadido al seguimiento Git, especificando un mensaje descriptivo, por ejemplo, primer lanzamiento. Verifica siempre el estado tras realizar una acción relevante. Y por último muestra el log.

- Para hacer un commit con un mensaje descriptivo:

```
git commit -m "Mi primer guardado"
```

- Verificar el estado y el log después del commit.

```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git commit -m "Mi primer guardado"
[master (root-commit) d1fff2f] Mi primer guardado
1 file changed, 1 insertion(+)
create mode 100644 HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
nothing to commit, working tree clean

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
commit d1fff2f86690435847af052c0239536d5924e061 (HEAD -> master)
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 14:58:55 2024 +0200

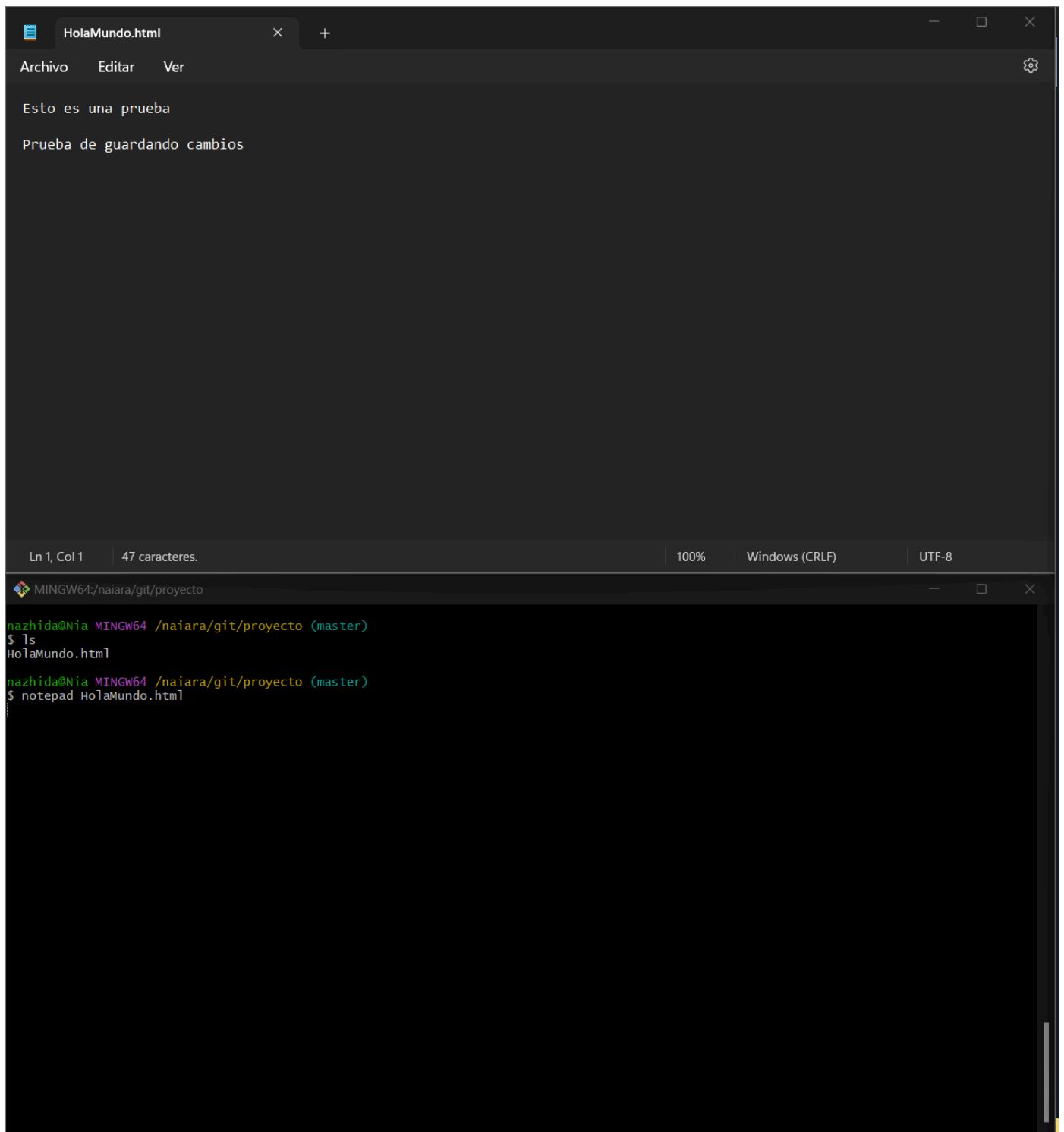
    Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

23. Lista el directorio actual donde se encuentra el fichero. Abre dicho fichero con un editor. Realiza un cambio en la página web que contiene el fichero. Salvo todos los cambios antes de cerrar el fichero. Y comprueba el estado de Git viendo si hay modificaciones. Para finalizar da marcha atrás con una orden a dichas modificaciones. Y comprueba el estado final.

- Abre el archivo, realiza cambios y verifica el estado:

```
notepad hola_mundo.html
```



- Verificar si hay modificaciones y revertir:

```
git status
git checkout -- hola_mundo.html
```

```
MINGW64:/naiara/git/proyecto
nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ ls
HolaMundo.html

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ notepad HolaMundo.html

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
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:   HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git restore HolaMundo.html

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
nothing to commit, working tree clean

nashida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

24. Repite los pasos anteriores. Lista el directorio, abre el fichero, realiza un cambio, una vez salvado y cerrado comprueba el estado del repositorio. La respuesta debe ser que de nuevo ha habido modificaciones, pero ahora debes añadirlas al seguimiento, y comprobar su estado.

- Realiza un cambio, guarda, añade al seguimiento, y verifica el estado:

```
notepad hola_mundo.html
git add hola_mundo.html
git status
```



```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ notepad HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
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:   HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git add .

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

25. Antes de continuar verifica el log por pantalla, debería haber solo un lanzamiento, es decir primer lanzamiento. Verifica el estado largo y corto, para luego proceder a realizar otro commit, con nombre y orden correctos por ejemplo segundo lanzamiento. Comprueba que el mensaje de respuesta para el estado es similar al lanzamiento anterior, pero que el log ha aumentado.

- Verifica el log y realiza otro commit:

```
git log
git status
git commit -m "Segundo lanzamiento"
git log
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
commit d1fff2f86690435847af052c0239536d5924e061 (HEAD -> master)
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 14:58:55 2024 +0200

    Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git commit -m "Mi segundo guardado"
[master a199ede] Mi segundo guardado
1 file changed, 3 insertions(+), 1 deletion(-)

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
commit a199ede14623600143e6b30582c786942e098701 (HEAD -> master)
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 15:08:36 2024 +0200

    Mi segundo guardado

commit d1fff2f86690435847af052c0239536d5924e061
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 14:58:55 2024 +0200

    Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

26.Muestra la ayuda de todos los comandos Git. Una vez dentro de la ayuda sal introduciendo la letra q. Ahora muestra la ayuda de un comando en particular que tu elijas.

- Para mostrar la ayuda de todos los comandos Git:

```
git help
```

- Para la ayuda de un comando en particular:

```
git help <comando>
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git help
usage: git [-v | --version] [-h | --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>]
          [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

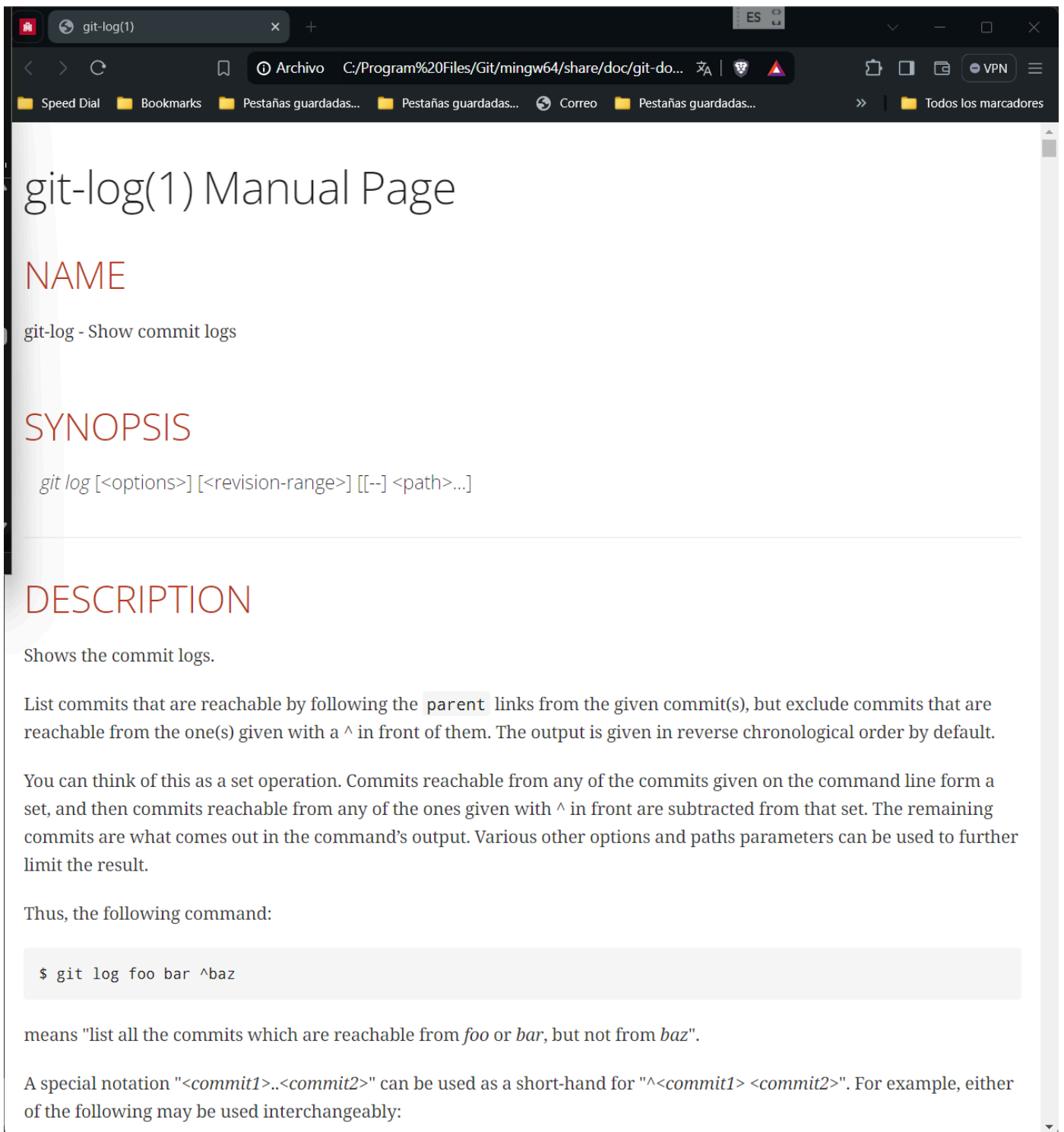
start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

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

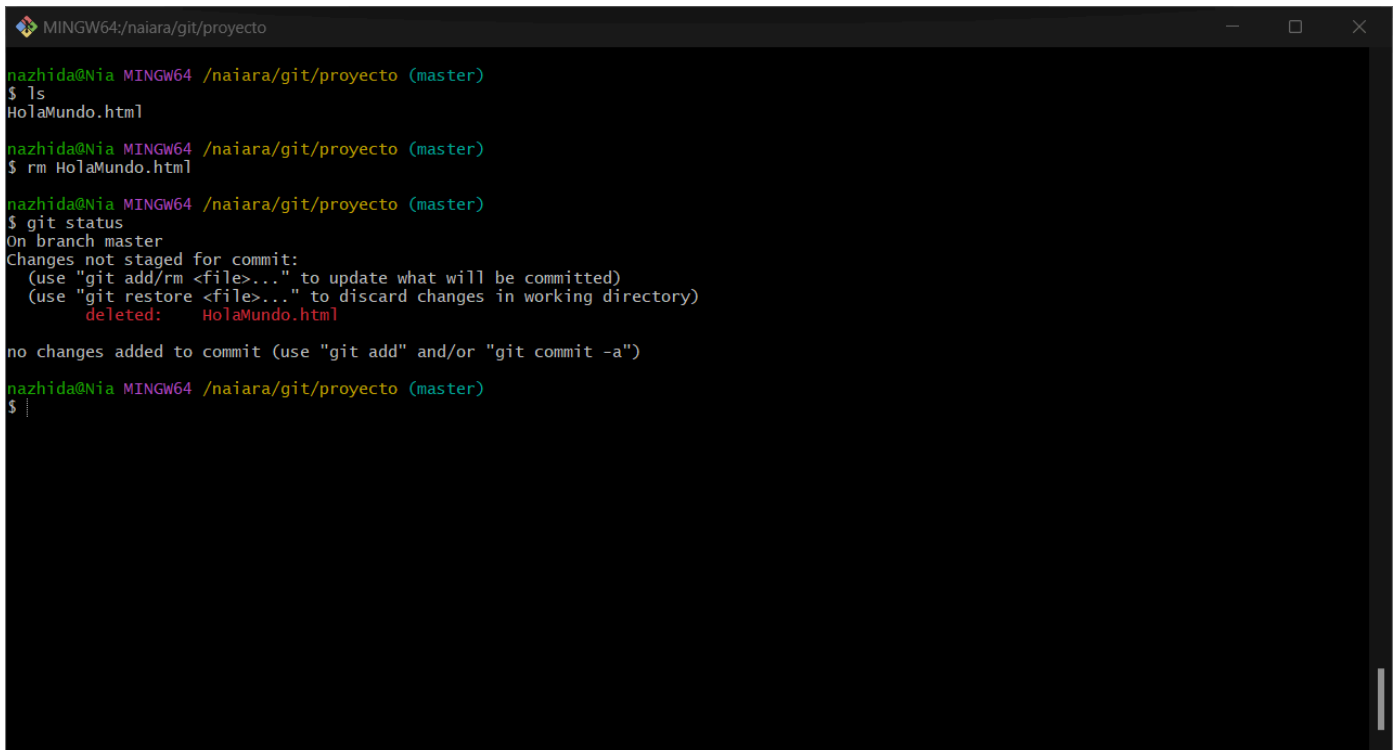
collaborate (see also: git help workflows)
```



27. Lista el directorio de tu proyecto Git. Verifica que el archivo sobre el que trabajamos esta físicamente. Y elimínalo. Lista a continuación la ruta para ver que no aparece. Y ejecuta una orden de Git para verificar si realmente el mismo fichero sigue estando en el repositorio.

- Verifica que el archivo esté presente y elimínalo:

```
ls
rm hola_mundo.html
ls
git status
```



```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ ls
HolaMundo.html

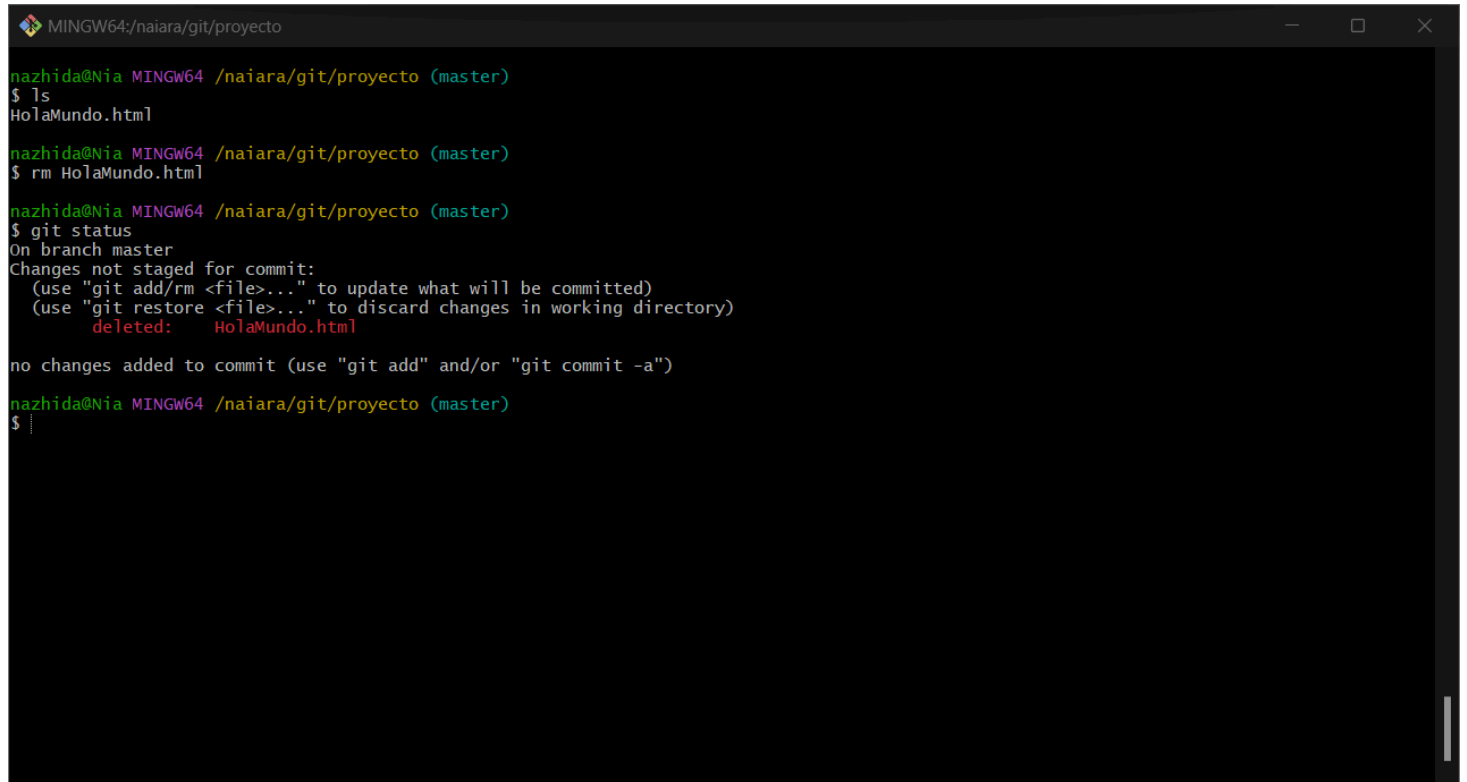
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ rm HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

28. Muestra el estado de tu carpeta asociada a tu repositorio. La respuesta debe de indicar que al fichero ha sido eliminado. Restaura el fichero con un

comando Git. Y luego usa una orden Linux para listar el directorio, verificando se recuperó físicamente el fichero original.

A screenshot of a terminal window titled 'MINGW64:/naiara/git/proyecto'. The terminal shows a series of commands and their outputs. First, the user runs 'ls' and sees 'HolaMundo.html'. Then, they run 'rm HolaMundo.html'. Finally, they run 'git status', which shows that 'HolaMundo.html' has been deleted. The terminal text is as follows:

```
minghida@Nia MINGW64 /naiara/git/proyecto (master)
$ ls
HolaMundo.html

minghida@Nia MINGW64 /naiara/git/proyecto (master)
$ rm HolaMundo.html

minghida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")

minghida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

29. Visualiza el log de tu repositorio. Además, visualízalo también en un formato de una línea, para ello, utiliza la misma orden con un modificador concreto. Deben de aparecer ordenados los dos lanzamientos efectuados. Y los códigos que los identifican.

- Mostrar el log en una línea:

```
git log --oneline
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
commit a199ede14623600143e6b30582c786942e098701 (HEAD -> master)
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 15:08:36 2024 +0200

    Mi segundo guardado

commit d1fff2f86690435847af052c0239536d5924e061
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 14:58:55 2024 +0200

    Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
a199ede (HEAD -> master) Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

30. Ejecuta una orden para revertir el segundo lanzamiento en el log. Has de emplear el código asociado a dicho lanzamiento. Luego vuelve a ver el log en el formato de una única línea. Debe aparecer una línea extra de información en el log.

- Revertir usando el código del log:

```
git revert <código-del-segundo-lanzamiento>
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log
commit d6b21ee024895993b44168469bd534779ea11bda (HEAD -> master)
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 15:19:19 2024 +0200

    Revert "Mi segundo guardado"

    This reverts commit a199ede14623600143e6b30582c786942e098701.

commit a199ede14623600143e6b30582c786942e098701
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 15:08:36 2024 +0200

    Mi segundo guardado

commit d1fff2f86690435847af052c0239536d5924e061
Author: lz20r <zhiyaonaiara@gmail.com>
Date:   Wed Jun 19 14:58:55 2024 +0200

    Mi primer guardado

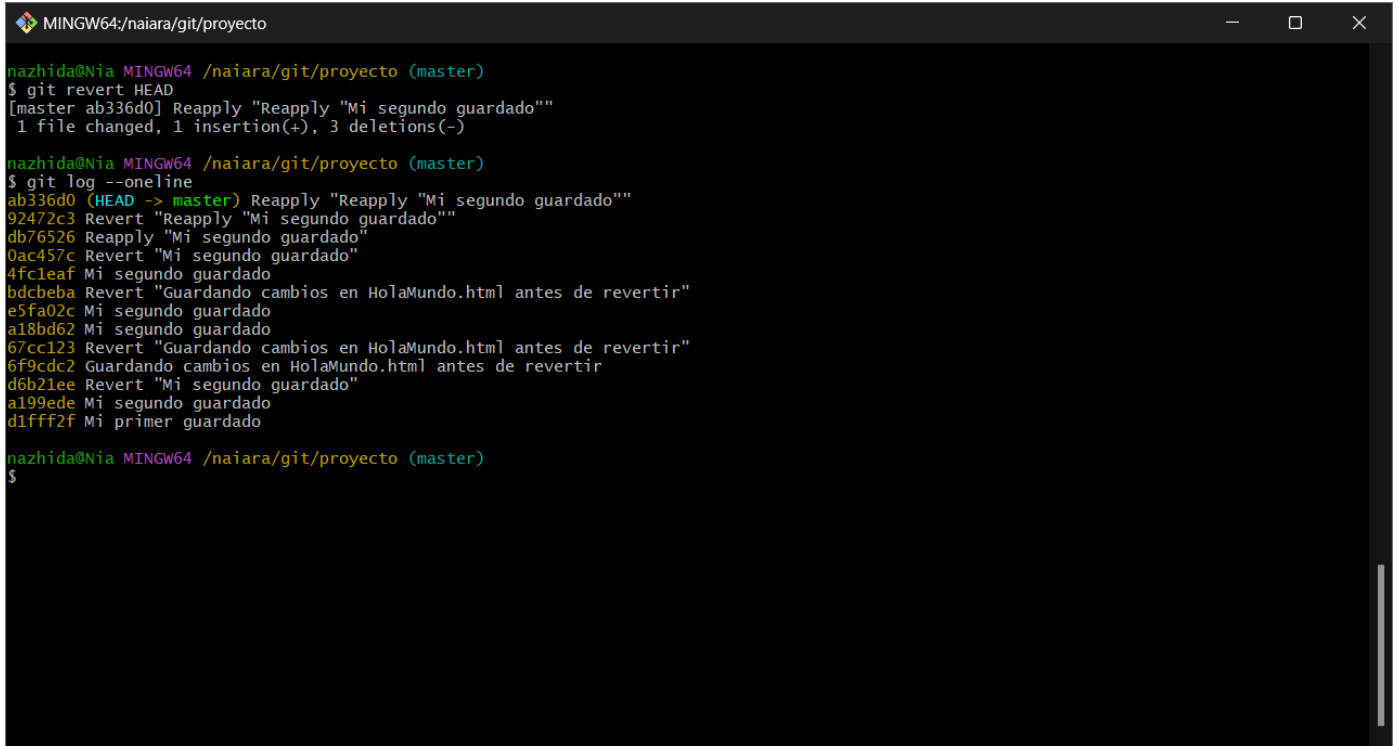
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
d6b21ee (HEAD -> master) Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado
```

31. Repite justo los mismos pasos que en el apartado anterior. Pero en vez de indicar el código, esta vez lo sustituyes por la palabra reservada

cabecera en inglés HEAD. Veras que la respuesta es la misma que en el apartado anterior. Da una explicación a esta coincidencia.

- Revertir usando HEAD:

```
git revert HEAD
```



```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git revert HEAD
[master ab336d0] Reapply "Reapply "Mi segundo guardado""
1 file changed, 1 insertion(+), 3 deletions(-)

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
ab336d0 (HEAD -> master) Reapply "Reapply "Mi segundo guardado""
92472c3 Revert "Reapply "Mi segundo guardado""
db76526 Reapply "Mi segundo guardado"
0ac457c Revert "Mi segundo guardado"
4fc1eaf Mi segundo guardado
bdcbeba Revert "Guardando cambios en HolaMundo.html antes de revertir"
e5fa02c Mi segundo guardado
a18bd62 Mi segundo guardado
67cc123 Revert "Guardando cambios en HolaMundo.html antes de revertir"
6f9cdc2 Guardando cambios en HolaMundo.html antes de revertir
d6b21ee Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

32. Muestra el log en formato única línea. Ahora resetea con una orden Git el código referido al segundo lanzamiento, es decir, el del medio. Y vuelve a mostrar el log. Tendrá que aparecer la misma información, que la primera vez que utilizaste el log, o sea dos registros del historial.

- Resetear al segundo lanzamiento:

```
git reset <código-del-segundo-lanzamiento>
git log
```



```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
ab336d0 (HEAD -> master) Reapply "Reapply "Mi segundo guardado""
92472c3 Revert "Reapply "Mi segundo guardado""
db76526 Reapply "Mi segundo guardado"
0ac457c Revert "Mi segundo guardado"
4fc1eaf Mi segundo guardado
bdcbeba Revert "Guardando cambios en HolaMundo.html antes de revertir"
e5fa02c Mi segundo guardado
a18bd62 Mi segundo guardado
67cc123 Revert "Guardando cambios en HolaMundo.html antes de revertir"
6f9cdc2 Guardando cambios en HolaMundo.html antes de revertir
d6b21ee Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git reset --hard 92472c3
HEAD is now at 92472c3 Revert "Reapply "Mi segundo guardado""

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
92472c3 (HEAD -> master) Revert "Reapply "Mi segundo guardado""
db76526 Reapply "Mi segundo guardado"
0ac457c Revert "Mi segundo guardado"
4fc1eaf Mi segundo guardado
bdcbeba Revert "Guardando cambios en HolaMundo.html antes de revertir"
e5fa02c Mi segundo guardado
a18bd62 Mi segundo guardado
67cc123 Revert "Guardando cambios en HolaMundo.html antes de revertir"
6f9cdc2 Guardando cambios en HolaMundo.html antes de revertir
d6b21ee Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

33. Muestra el estado del repositorio de nuevo. Debe aparecer el archivo modificado. Restaura dicho archivo usando un comando Git. Y verifica a continuación el estado del repositorio. No debe aparecer ningún archivo que pueda hacerse commit.

- Para restaurar un archivo modificado:

```
git restore <archivo>
```

```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
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:   HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git restore HolaMundo.html

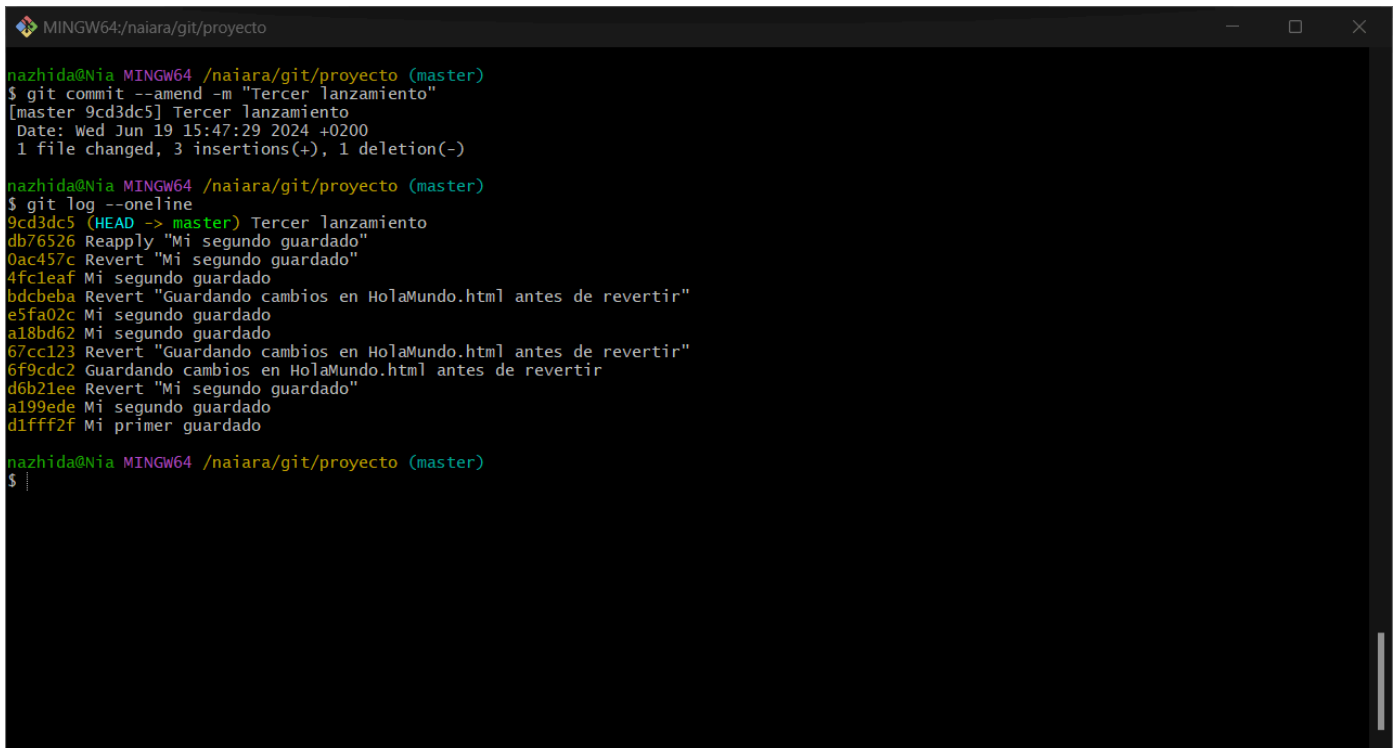
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
nothing to commit, working tree clean

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

34. Muestra el log de única línea. Veras el primer y segundo lanzamiento y sus mensajes. Con una orden Git debes corregir o rectificar el último lanzamiento escribiendo un nuevo mensaje que sea tercer lanzamiento. Comprueba ha cambiado del segundo al tercero. Es decir, que cuando ejecutes el log de única línea, veras que aparecen el primero y el tercero.

- Para corregir el último mensaje del commit:

```
git commit --amend -m "Tercer lanzamiento"
```

A screenshot of a terminal window titled 'MINGW64:/naiara/git/proyecto'. The user 'nazhida@Nia' is in the directory '/naiara/git/proyecto' on the 'master' branch. They execute the command 'git commit --amend -m "Tercer lanzamiento"', which results in a new commit '9cd3dc5' with the message 'Tercer lanzamiento'. The date is 'Wed Jun 19 15:47:29 2024 +0200' and it shows '1 file changed, 3 insertions(+), 1 deletion(-)'. Then, they run 'git log --oneline', which displays a list of commits from '9cd3dc5' (HEAD -> master) 'Tercer lanzamiento' down to 'd1fff2f' 'Mi primer guardado'.

```
mingw64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git commit --amend -m "Tercer lanzamiento"
[master 9cd3dc5] Tercer lanzamiento
Date: Wed Jun 19 15:47:29 2024 +0200
1 file changed, 3 insertions(+), 1 deletion(-)

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
9cd3dc5 (HEAD -> master) Tercer lanzamiento
db76526 Reapply "Mi segundo guardado"
0ac457c Revert "Mi segundo guardado"
4fc1eaf Mi segundo guardado
bdcbeba Revert "Guardando cambios en HolaMundo.html antes de revertir"
e5fa02c Mi segundo guardado
a18bd62 Mi segundo guardado
67cc123 Revert "Guardando cambios en HolaMundo.html antes de revertir"
6f9cdc2 Guardando cambios en HolaMundo.html antes de revertir
d6b21ee Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

35. Abre el fichero con tu editor. Realiza un cambio en él diferente a los que ya hayas realizado antes. Guárdalo y ciérralo. Comprueba el estado del repositorio para verificar que el fichero ha sido modificado. Vuelve a hacer justo lo mismo del apartado anterior, es decir, debes corregir o rectificar el último lanzamiento, sustituyendo el mensaje de tercero por cuarto. Por último, muestra el log en una única línea. Veras el primero y el cuarto lanzamiento.

- Realiza un cambio y guarda el archivo, luego rectifica el commit:

```
notepad hola_mundo.html
git commit --amend -m "Cuarto lanzamiento"
```

```
MINGW64:/naiara/git/proyecto

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ notepad HolaMundo.html

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git status
On branch master
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:   HolaMundo.html

no changes added to commit (use "git add" and/or "git commit -a")

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git commit --amend
[master 5c278fd] Mi cuarto guardado
Date: Wed Jun 19 15:47:29 2024 +0200
1 file changed, 3 insertions(+), 1 deletion(-)

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git log --oneline
5c278fd (HEAD -> master) Mi cuarto guardado
db76526 Reapply "Mi segundo guardado"
0ac457c Revert "Mi segundo guardado"
4fc1eaf Mi segundo guardado
bdcbeba Revert "Guardando cambios en HolaMundo.html antes de revertir"
e5fa02c Mi segundo guardado
a18bd62 Mi segundo guardado
67cc123 Revert "Guardando cambios en HolaMundo.html antes de revertir"
6f9cdc2 Guardando cambios en HolaMundo.html antes de revertir
d6b21ee Revert "Mi segundo guardado"
a199ede Mi segundo guardado
d1fff2f Mi primer guardado

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$
```

36. Visualiza la estructura de ramas de Git. Recordar la rama principal se llama master. Crea para Git una nueva rama de tu repositorio. La nueva rama se va a llamar apprentice . Una vez creada ejecuta una orden Git para visualizar de nuevo todas las ramas de tu repositorio.

- Crear una nueva rama llamada apprentice :

```
git branch apprentice
git branch
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git branch
* master

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git branch apprentice

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git branch
  apprentice
* master

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

37. Borra la rama apprentice que acabas de crear. Y comprueba que se ha borrado. Debe de aparecer únicamente la rama principal llamada master.

- Borrar la rama master :

```
git branch -d apprentice
git branch
```

```
MINGW64:/naiara/git/proyecto
nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git branch -d apprentice
Deleted branch apprentice (was 5c278fd).

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ git branch
* master

nazhida@Nia MINGW64 /naiara/git/proyecto (master)
$ |
```

38. Siempre que borramos algo del proyecto luego lo creamos de nuevo para continuar. Por tanto, visualizar que existan ambas ramas `master` y `apprentice`. Situados en la rama `master`, lo que hacemos es cambiar a la rama `apprentice`, ejecutando una orden Git. Y mostramos el log donde deberás ver tus dos lanzamientos, pero desde la perspectiva de tu rama actual.

- Crear y cambiar entre ramas:

```
git branch apprentice
git checkout apprentice
git log
```

39. Lista desde tu rama `apprentice` los elementos del directorio asociado a tu repositorio. Abre el archivo, modifícalo añadiendo texto a la página web, guárdalo y ciérralo. Visualiza el estado actual del repositorio. Veras ha sido modificado el archivo, añádelo al seguimiento, muestra el estado de nuevo, luego realiza en Git un commit llamado quinto lanzamiento, y termina con la ejecución del comando log en una única línea. Veras que aparecen tres lanzamientos.

- Modificar archivo y realizar commit:

```
notepad hola_mundo.html
git add hola_mundo.html
git commit -m "Quinto lanzamiento"
git log --oneline
```

40. Muestra las ramas existentes. Estando en la rama `apprentice` cambia a la rama `master`. Ahora cambia a una nueva rama que denominamos `emergency`. En ella lista los elementos del directorio, abre el editor, modifica el fichero, guárdalo y ciérralo.

- Crear y modificar archivo en la rama `emergency` :

```
git branch emergency
git checkout emergency
notepad hola_mundo.html
```

41. Verifica por pantalla el estado actual del repositorio. Veras el archivo modificado. Añade dicho archivo al seguimiento. Comprueba de nuevo su estado. Realiza un commit con nombre por ejemplo sexto lanzamiento. Y

muestra el log en una única línea resumido. Deberías de ver tres lanzamientos en total.

- Realizar commit en `emergency` :

```
git add hola_mundo.html
git commit -m "S
git log --oneline
```

42. Visualiza todas las ramas. Estando en la rama `emergency` cambia a la rama `master`. Fusiona todos los cambios de la rama actual, es decir, fusiona la rama principal con la rama que se especifica en el comando fusionar o sea la rama `emergency` anteriormente creada. Por último, elimina la rama `emergency` . Debe dar un aviso de la acción efectuada. Muestra todas las ramas por pantalla para asegurarte.

- Fusionar ramas y borrar `emergency` :

```
git checkout master
git merge emergency
git branch -d emergency
git branch
```

43. Estando en la rama principal denominada `master`. Cambia a la rama `apprentice` . Lista los elementos del directorio, abre el editor con un comando, modifica, guarda y cierra el fichero.

- Modificar archivo en `apprentice` :

```
git checkout apprentice
notepad hola_mundo.html
```

44. Estando en la rama `apprentice` verifica el estado de tu repositorio. Debe aparecer el fichero como modificado. Añádalo al seguimiento, comprueba que cambio el estado en Git, y luego realiza un commit indicando en el mensaje séptimo lanzamiento. Muestra el log, donde has de comprobar, aparecen un total de cuatro lanzamientos.

- Realizar commit:

```
git add hola_mundo.html
git commit -m "Séptimo lanzamiento"
git log
```

45. Desde la rama actual o sea `apprentice` . Lista todas las ramas. Y luego cambia a la rama `master` de tu repositorio. Efectúa una unión de la rama actual `master` con la rama que tú le indiques en la instrucción Git que será la rama `apprentice` . Esto debe de generar un conflicto a través de un mensaje por pantalla. Para resolver el conflicto debes abortar la unión con otro comando Git. Y verificar que en el estado final está todo correcto sin errores.

- Generar un conflicto y abortar:

```
git checkout master
git merge apprentice
git merge --abort
git status
```

46. Vamos a realizar un nuevo intento. Desde la rama actual o sea `apprentice`. Mostramos todas las ramas. Cambiamos a la rama `master`. Comprobamos el estado actual para verificar que todo está correcto. Efectuamos una fusión de la rama actual `master` con la rama que le indiquemos que será `apprentice`. La respuesta debe ser un mensaje de conflicto.

- Resolver el conflicto:

```
git checkout master
git merge apprentice
```

47. Vamos a resolver el conflicto actual. Mostramos el estado del repositorio donde aparecerá el fichero modificado. Añadimos los cambios del fichero al seguimiento. Volvemos a mostrar el estado actual. Y hacemos un commit con nombre octavo lanzamiento. Para finalizar se muestra el log con el modificador única línea. Veremos un total de seis lanzamientos.

- Realizar commit del conflicto:

```
git add hola_mundo.html
git commit -m "Octavo lanzamiento"
git log --oneline
```

48. Desde la actual rama principal o `master`, visualizamos todas las ramas. A continuación, borramos la rama `apprentice` . Intentamos cambiar a la rama `apprentice` . Aparecerá un aviso de que no existe. Mostramos nuevamente

todas las ramas solo quedara la rama master. Y además se representa por pantalla el estado actual deberá que estar todo correcto.

- Borrar y verificar:

```
git branch -d apprentice
git checkout apprentice
```

49. Busca información sobre comandos en Git. Ejecuta una instrucción útil que todavía no se haya usado hasta ahora. Y explica brevemente su funcionamiento como comando.

- Buscar un comando nuevo:

```
git reflog
```

50. Ejecuta un comando para salir de la Shell, pero antes de ello realiza un pantallazo. Con ello debe cerrarse el programa. Si no ejecuta los comandos necesarios para terminar. The end.

- Salir de Git Bash:

```
exit
```

51. A continuación se van a emplear varios comandos Linux para un mejor manejo de la shell. Usar el comando find con los modificadores maxdepth de tamaño uno y type referido a directorios. Hacer lo mismo, pero negando el modificador type.

- Para listar directorios con maxdepth y type :

```
find . -maxdepth 1 -type d
```

- Para negar el modificador type :

```
find . -maxdepth 1 ! -type d
```

52. Ejecutar en una misma línea de comando, la orden ls solo para ficheros, y grep para encontrar la barra mayúscula siete. Hacer lo mismo, pero negando el modificador grep .

- Listar solo ficheros y encontrar la barra mayúscula siete:


```
ls -p | grep -v /  
ls -p | grep 7
```

- Para negar el modificador `grep` :

```
ls -p | grep -v 7
```

53. Listar con una orden Linux el directorio del repositorio. Mostrar por la consola directamente el contenido del fichero. No es válido usar ningún editor de código.

- Para listar el directorio del repositorio:

```
ls
```

- Para mostrar el contenido del fichero:

```
cat nombre_del_fichero
```

54. Listar la ruta actual del repositorio. Mover el fichero a una ruta más a la izquierda. Sin cambiar de ruta listar la ruta donde se movió el fichero.

- Mover el fichero a una ruta más a la izquierda:

```
mv nombre_del_fichero ../nueva_ruta/
```

- Listar la ruta donde se movió el fichero:

```
ls ../nueva_ruta/
```

55. Listar la ruta del proyecto de nuevo. Ejecutar un comando que muestre por pantalla la ruta absoluta del directorio donde te encuentras. Cambiar al directorio donde moviste el fichero es decir un lugar a la izquierda. Desde esta nueva ubicación mover el fichero a la ruta donde estaba antes o sea la ruta de la carpeta del proyecto. Lista dicha ruta desde la ruta actual. Por último, cambiar a la ruta de la carpeta del proyecto donde ahora está el fichero y listarlo

- Mostrar la ruta absoluta:

```
pwd
```

- Moverse y listar directorios:

```
cd ../nueva_ruta/  
mv nombre_del_fichero ../ruta_original/  
cd ../ruta_original/  
ls
```

56. Listar la ruta verificando se encuentra el archivo. Copiar dicho archivo en la ruta llamada Git que se genero al inicio. Comprobar que ahora el archivo existe en ambas rutas. Dirigirse a la nueva ruta donde se copio y borrarlo. Volver a la ruta original y listar la misma de nuevo.

- Verificar y copiar el archivo:

```
cp nombre_del_fichero ~/Git/  
ls ~/Git/  
rm ~/Git/nombre_del_fichero  
ls ~/Git/
```

57. Estando en la carpeta raíz del proyecto. Cambiamos a una carpeta más a la izquierda. Copiamos toda la carpeta de forma recursiva a otra ubicación. Cambiamos a dicho lugar, listando todas las carpetas que contengan parte de la palabra proyecto. Una vez encontrado, eliminamos de forma recursiva esta copia del proyecto y verificamos que desapareció. Por último, volvemos a la ruta estándar de nuestro repositorio y comprobamos está el proyecto.

- Copiar carpeta recursivamente:

```
cp -r carpeta_origen carpeta_destino  
cd carpeta_destino  
ls | grep proyecto  
rm -r carpeta_destino  
cd carpeta_origen
```

58. Listamos todos los elementos del repositorio. Aparecerá nuestro fichero. Renombramos el fichero a otro nombre. Verificamos listando el directorio el nombre ha cambiado. De nuevo renombramos este nuevo fichero a su nombre original. Y lo volvemos a listar por pantalla.

- Renombrar fichero:

```
mv nombre_del_fichero nuevo_nombre
ls
mv nuevo_nombre nombre_del_fichero
ls
```

59. Listamos otra vez el repositorio Git, pero ahora lo listamos mostrando en columnas los detalles referentes al fichero, empleando un modificador junto al comando listar. Ejecuta un comando para ver el directorio actual por pantalla. Cambia al directorio que esta justo una posición a la izquierda. Y lista este directorio como antes con todos los detalles, pero ahora del directorio Git. Para ello usa el mismo modificador del comando listar de nuevo.

- Listar con detalles:

```
ls -l
cd ..
ls -l Git
```

60. Para el fichero de nuestro repositorio. Cambia los permisos de este a 000 y 755. Antes y

- Cambiar permisos:

```
chmod 000 nombre_del_fichero
ls -l
chmod 755 nombre_del_fichero
ls -l
```

61. Realizar las siguientes tareas en una única línea de comando. Listar directorio, cambiar al directorio a la izquierda del actual, listar el nuevo directorio, volver a la ubicación del proyecto un lugar más a la derecha, y listar nuevamente sus elementos

- Listar, cambiar, listar, volver y listar:

```
ls && cd .. && ls && cd - && ls
```

62. Muestra todos los procesos activos del sistema. Finaliza uno de ellos. Sal de Bash. The end.

- Mostrar procesos activos:

```
ps aux
```

- Finalizar un proceso:

```
kill PID    # Reemplaza PID con el ID del proceso a terminar
```

- Salir de Bash:

```
exit
```