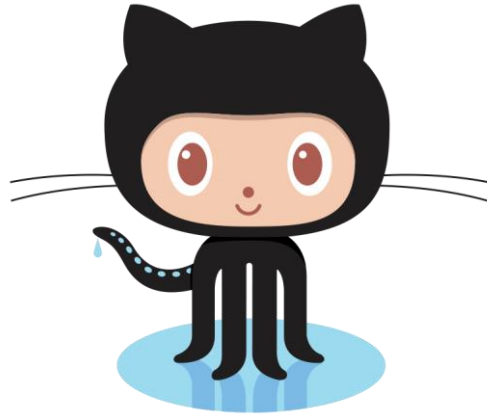
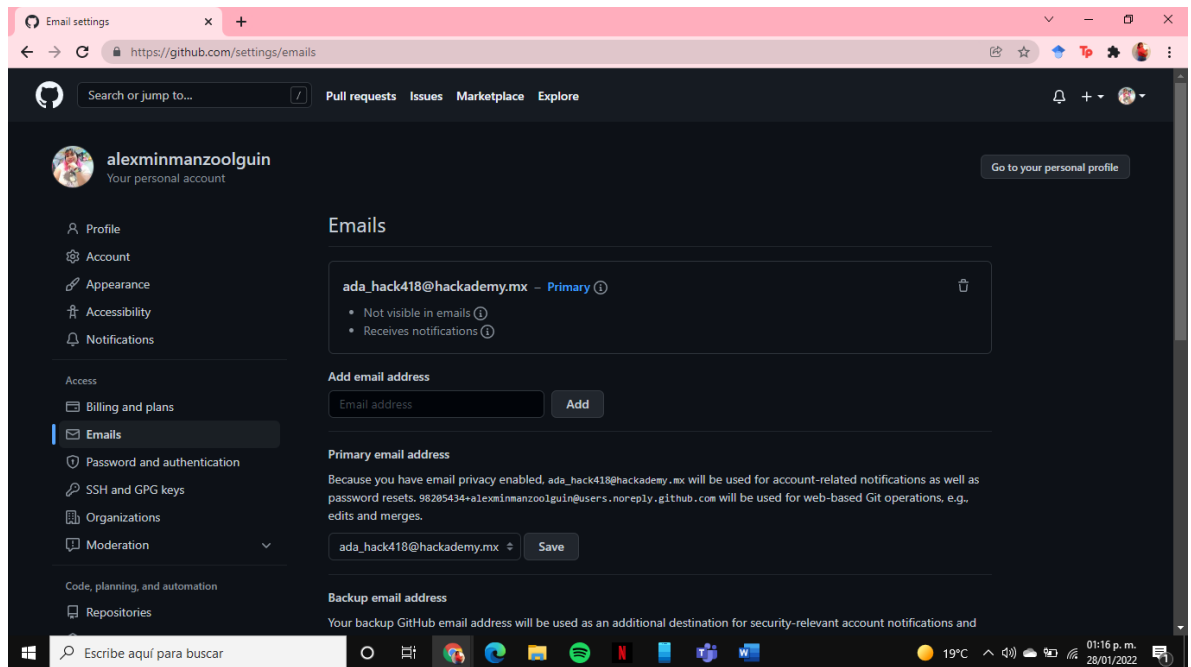


ACTIVIDAD DE CORREO

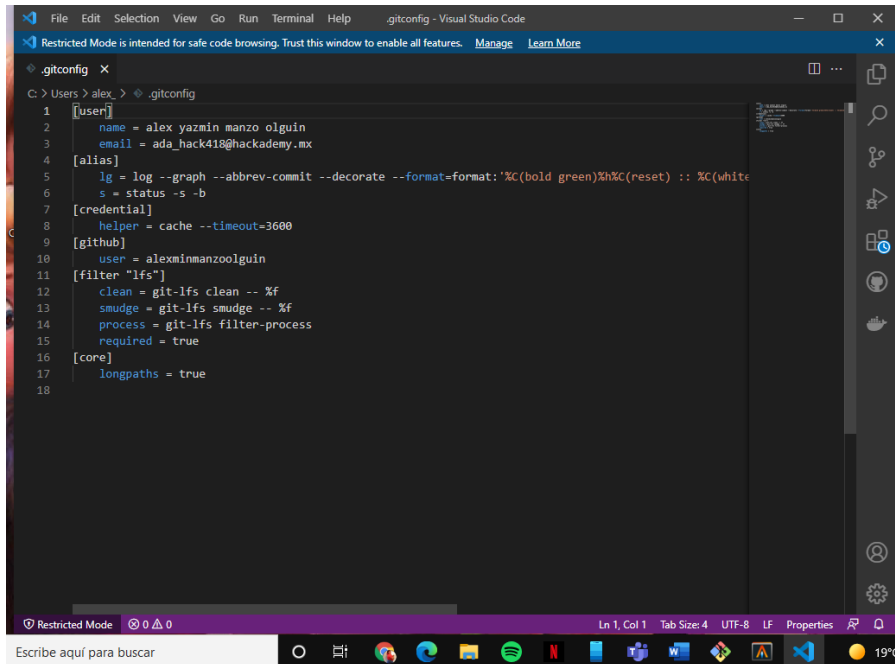


Los 5 puntos importantes para la memoria de GitHub

1. Tener una cuenta en GitHub donde el correo principal sea el de hackademy



2. Tener instalado Git en mi computadora para trabajar en la línea de comandos



```
1 [user]
2   name = alex yazmin manzo olguin
3   email = ada_hack418@hackademy.mx
4 [alias]
5   lg = log --graph --abbrev-commit --decorate --format=format:%C(bold green)%h%C(reset) :: %C(white)s
6   s = status -s -b
7 [credential]
8   helper = cache --timeout=3600
9 [github]
10  user = alexinmanzooolguin
11 [filter "lfs"]
12  clean = git-lfs clean -- %f
13  smudge = git-lfs smudge -- %f
14  process = git-lfs filter-process
15  required = true
16 [core]
17   longpaths = true
18
```

3. Prepararme con los comando básicos GIT: init, commit, status, checkout

Configuración git

\$git config

- Establecer el nombre del usuario
\$ git config --global user.name "colocar su nombre"
- Establecer el correo de usuario
\$ git config --global user.email "colocar su correo"
- Activar el coloreado de la salida
\$ git config --global color.ui auto
- Mostrar el estado original en los conflictos
\$ git config --global merge.conflictstyle diff3
- Mostrar configuración
\$ git config --list

Creación de repositorios

Creación de un repositorio nuevo

\$git init

- \$ git init<nombre-repositorio> crea un nuevo repositorio con el nombre <nuevo repositorio>
- Este comando crea una nueva carpeta con el nombre del repositorio que a su vez contiene otra carpeta oculta llamada .git que contiene la base de datos donde se registran los cambios en el repositorio.

Añadir cambios al repositorio

\$git commit

- `$git commit -m "mensaje"` confirma todos los cambios de la zona de intercambio temporal añadiéndolos al repositorio y creando una nueva versión del proyecto. "Mensaje" es un breve mensaje describiendo los cambios realizados que se asociara a la nueva versión del proyecto.
- `$git commit --amend -m "mensaje"` cambia el mensaje del último commit por el nuevo mensaje.

Estado e historia de un repositorio

Mostrar el estado de un repositorio

\$git status

- Muestra el estado de los cambios en el repositorio desde la última versión guardada. En particular, muestra los ficheros con cambios en el directorio de trabajo que no se han añadido a la zona de intercambio temporal y los ficheros en la zona de intercambio temporal que no se han añadido al repositorio.

Deshacer cambios

Elimina cambios del directorio de trabajo o volver a una versión anterior

\$git checkout

- `$git checkout <commit> --<file>` actualiza el fichero <file> a la versión correspondiente al commit. Suele utilizarse para eliminar los cambios en un fichero que no han sido guardados aun en la zona de intercambio temporal, mediante el comando `git checkout HEAD -- <file>`.

4. Prepararme que ¿Qué es un repositorio? ¿Qué es clonar un repositorio?

¿Qué es un repositorio?

un repositorio es una ubicación central de almacenamiento de archivos. Es utilizado por control de versiones sistemas para almacenar múltiples versiones de archivos. Si bien un repositorio se puede configurar en una máquina local para un solo usuario, a menudo se almacena en un servidor, al que pueden acceder varios usuarios.

Un repositorio contiene tres elementos principales: un tronco, ramas y etiquetas. Esto puede incluir múltiples código fuente archivos, así como otros recursos utilizados por el programa.

Las ramas se utilizan para almacenar nuevas versiones del programa. Un desarrollador puede crear una nueva sucursal cada vez que realiza revisiones sustanciales al programa. Las etiquetas se utilizan para guardar versiones de un proyecto, pero no están destinadas al desarrollo activo.

¿Qué es clonar un repositorio?

significa bajarse una copia completa del mismo a nuestro ordenador. significa que después de clonar un repositorio en la computadora tendrás toda la historia del repositorio, completa.

Se podrá:

- Navegar por la historia del proyecto, viendo cómo ha evolucionado
- Ver los cambios que han ocurrido en el código
- Saber quién y cuándo ha modificado cada cosa
- Acceder a las ramas del proyecto y crear las vuestras propias
- Empezar a modificar el código, guardar los cambios y añadir vuestro trabajo al que ya existe

5. Prepararme ¿Cuál es la diferencia entre Git y GitHub?

Git es una herramienta de control de versiones distribuida que puede gestionar el historial de código fuente de un proyecto de desarrollo, mientras que GitHub es una plataforma basada en la nube construida alrededor de la herramienta Git.



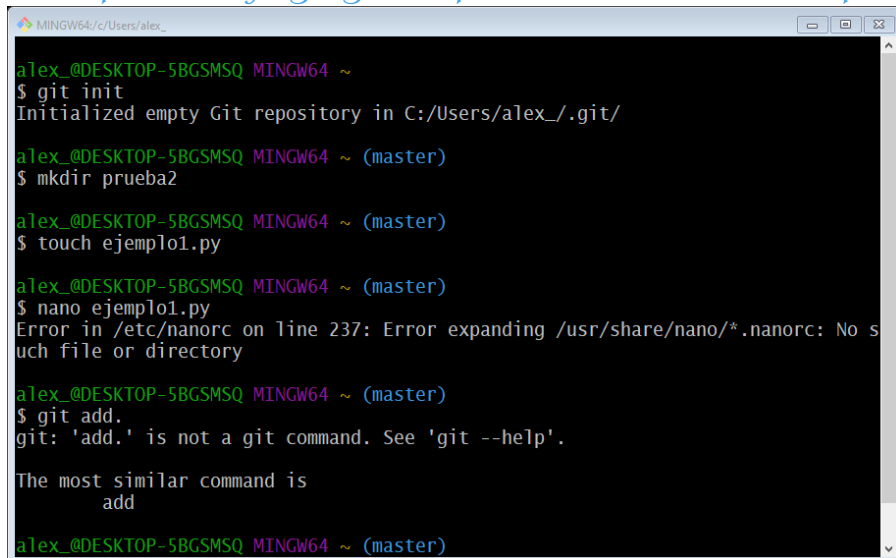
La diferencia principal entre Git y GitHub es que Git es una herramienta de código abierto que los desarrolladores instalan localmente para gestionar el

código fuente, mientras que GitHub es un servicio en línea al que los desarrolladores que utilizan Git pueden conectarse y cargar o descargar recursos.

En resumen, Git es el sistema de control de versiones y GitHub es un servicio de alojamiento para los repositorios de Git.

5 ACTIVIDADES PARA PRACTICAR DE GIT

1. Crear un repo local y agregar mi primer archivo a este repo



```
alex@DESKTOP-5BGSMSQ MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/alex/.git/

alex@DESKTOP-5BGSMSQ MINGW64 ~ (master)
$ mkdir prueba2

alex@DESKTOP-5BGSMSQ MINGW64 ~ (master)
$ touch ejemplo1.py

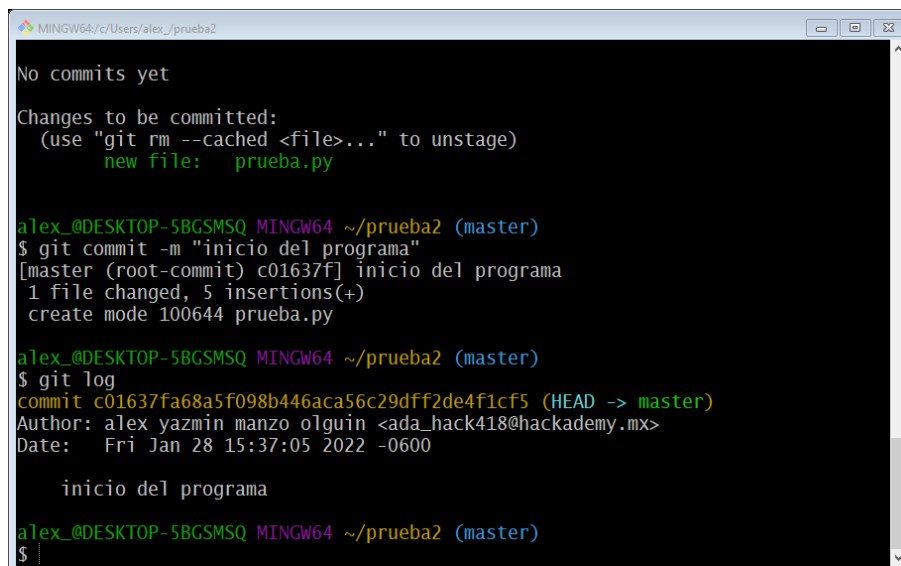
alex@DESKTOP-5BGSMSQ MINGW64 ~ (master)
$ nano ejemplo1.py
Error in /etc/nanorc on line 237: Error expanding /usr/share/nano/*.nanorc: No such file or directory

alex@DESKTOP-5BGSMSQ MINGW64 ~ (master)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
    add

alex@DESKTOP-5BGSMSQ MINGW64 ~ (master)
```

2. Hacer mi primer commit en mi repo local y validar el estatus de este commit



```
No commits yet

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

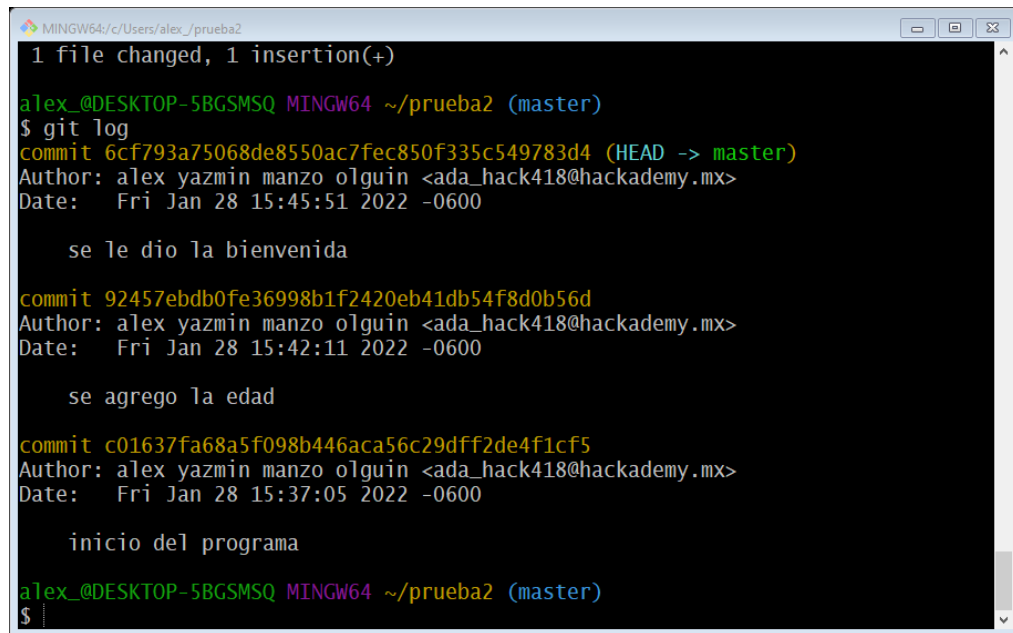
alex@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git commit -m "inicio del programa"
[master (root-commit) c01637f] inicio del programa
 1 file changed, 5 insertions(+)
 create mode 100644 prueba.py

alex@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git log
commit c01637fa68a5f098b446aca56c29dff2de4f1cf5 (HEAD -> master)
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:37:05 2022 -0600

    inicio del programa

alex@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$
```

3. Crear una rama y trabajar en ella haciendo 2 commits en la misma rama



```
MINGW64/c/Users/alex/_prueba2
1 file changed, 1 insertion(+)

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git log
commit 6cf793a75068de8550ac7fec850f335c549783d4 (HEAD -> master)
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:45:51 2022 -0600

    se le dio la bienvenida

commit 92457ebdb0fe36998b1f2420eb41db54f8d0b56d
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:42:11 2022 -0600

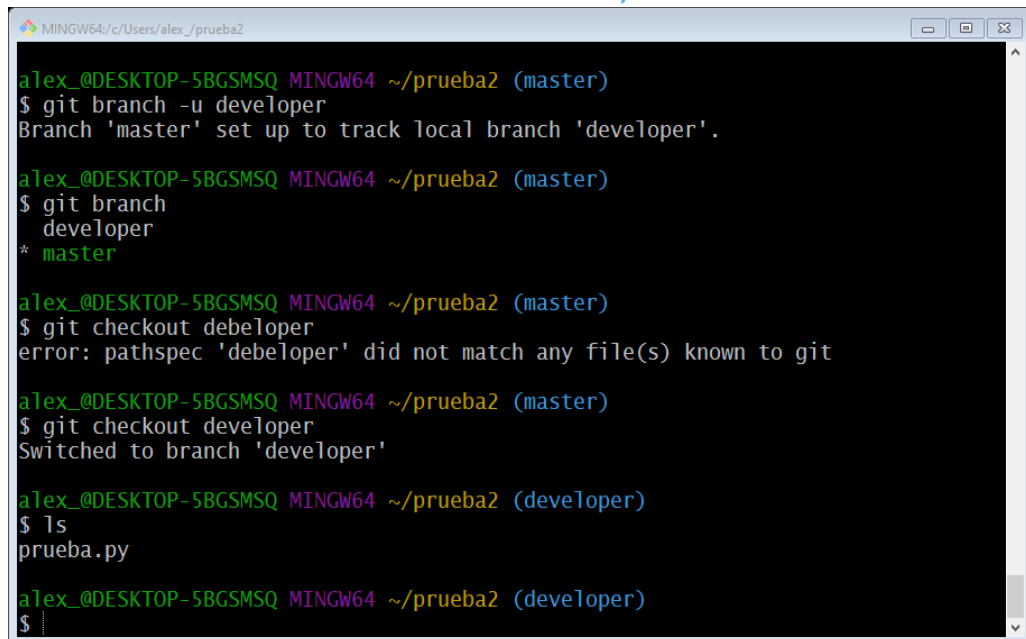
    se agrego la edad

commit c01637fa68a5f098b446aca56c29dff2de4f1cf5
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:37:05 2022 -0600

    inicio del programa

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$
```

4. Unir la rama creada con la rama develop o main



```
MINGW64/c/Users/alex/_prueba2

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git branch -u developer
Branch 'master' set up to track local branch 'developer'.

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git branch
  developer
* master

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git checkout debeloper
error: pathspec 'debeloper' did not match any file(s) known to git

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (master)
$ git checkout developer
Switched to branch 'developer'

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (developer)
$ ls
prueba.py

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (developer)
$
```

5. Ver la lista de las ramas que hemos creado en nuestro repositorio local.

```
alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (developer)
$ git log
commit 6cf793a75068de8550ac7fec850f335c549783d4 (HEAD -> developer, master, c016
37fa68a5f098b446aca56c29dff2de4f1cf5)
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:45:51 2022 -0600

    se le dio la bienvenida

commit 92457ebdb0fe36998b1f2420eb41db54f8d0b56d (commit)
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:42:11 2022 -0600

    se agrego la edad

commit c01637fa68a5f098b446aca56c29dff2de4f1cf5
Author: alex yazmin manzo olguin <ada_hack418@hackademy.mx>
Date:   Fri Jan 28 15:37:05 2022 -0600

    inicio del programa

alex_@DESKTOP-5BGSMSQ MINGW64 ~/prueba2 (developer)
$ |
```