

## Contenido

Introducción git/github (marzo-22) .....	1
Configuración de git .....	3
Videos de referencia .....	4
Trabajar con proyectos .....	4
Bajar proyecto desde git hub .....	4
Clonar proyecto .....	5
4 subir un nuevo repositorio .....	5
5 mas sobre remotos.....	7
Colaboradores. ....	7
Miscelánea de comandos.....	11
deshacer cambios en una rama .....	11
merge fusiona tu rama con la rama master .....	11
Actividad 1.....	12
Actividad 2.....	14
Ejemplo: aplicación Node para operaciones aritméticas:.....	16
Referencias .....	16

## Introducción git/github (marzo-22)

Git es un software de control de versiones diseñado por Linus Torvalds, para el mantenimiento de versiones de aplicaciones, particularmente útil cuando estas tienen un gran número de archivos de código fuente. Funciona como un conjunto de comandos ejecutables en una *terminal* (como la de **cmd** o **power shell (ps)**). Cuando instala GIT, le da la opción de instalar la interfaz BASH (instálela). Puede instalar git desde:

<https://git-scm.com/downloads>

Y en esta liga tiene un video que explica el funcionamiento de git:

[https://www.youtube.com/watch?v=HiXLkL42tMU&list=PLExyZ7hD-J5bigogqZq1G\\_AfD4K4cWPn8&index=1](https://www.youtube.com/watch?v=HiXLkL42tMU&list=PLExyZ7hD-J5bigogqZq1G_AfD4K4cWPn8&index=1)

Una referencia de instalación de Git la encuentra aquí: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git> .

GNU Bash o simplemente Bash (Bourne-again shell) es una popular interfaz de usuario de línea de comandos, específicamente un **shell** de **Unix**; así como un lenguaje de scripting de **Git**, permite ejecutar comandos de Linux que son muy útiles durante el desarrollo, por ejemplo, el comando **"curl"**. Bash fue originalmente escrito por Brian Fox para el sistema operativo GNU y está disponible para Windows en varias instalaciones, una de ellas es la de GIT.

Instalar la consola Bash

Opción 1: Bash viene integrada al instalar Git.

Opción 2: Bash viene también en otros paquetes de software como MySys2 (Es una colección de herramientas y bibliotecas que le brindan un entorno fácil de usar para crear, instalar y ejecutar software nativo de Windows), puede instalar mySys2 desde: <https://www.msys2.org/>

## Git y GitHub

**\*\*Git\*\*** es un sistema de control de versiones distribuido.

**\*\*Github\*\*** es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

git y github no son lo mismo, pero github utiliza git para gestionar sus proyectos. Básicamente usted lleva el control de sus versiones de proyecto con git y los respalda en la nube con github.

En estas notas usaremos la consola de git (git bash), pero también puede ejecutar los comandos usando Power Shell (PS) o la consola de comandos de DOS.

Para abrir una ventana de bash de clic derecho sobre la carpeta de Windows donde quiera trabajar y seleccione **'git bash here'**.

Para abrir una ventana de **Power Shell**, sobre la carpeta que quiere trabajar presione a tecla **'shift'** de clic derecho y seleccione **'abrir la ventana de Power Shell aquí'**. También puede seleccionar su carpeta en el explorador y elija la opción "Archivo" del menú de su explorador, elija "Abrir Windows Power Shell". Si no ve el menú de su explorador de archivos presione la tecla "Alt".

## Configuración de git

Antes de empezar a trabajar con sus proyectos, debe configurar git para que este enlazado con su cuenta de GitHub, esto lo hará solo una vez al instalar git:

Ejecute los siguientes comandos (las líneas que inician con el carácter # son comentarios):

```
``sh
# configurar su nombre de usuario y password que tiene en github
git config --global user.name "mi nombre en git hub"
git config --global user.email "miemail@mail"
#Opcionalmente puede seleccionar VSCode como el editor x defecto .
git config --global core.editor "code --wait"
#muestra la configuración actual
git config --global -e
#mostrar todas las opciones de configuración
git config -h
``
```

\*En el **git-bash** al escribir el nombre de un archivo basta escribir los primeros caracteres y presionar tab

Ejemplo:

```
$cat nombre_archi<tab>
```

Git tiene 3 áreas de trabajo:

- \* working directory //mi directorio de trabajo
- \* staging area //archivos preparados para respaldo (commit)
- \* git directory(repositorio) //archivos respaldados (committed)

Iniciar un proyecto para mantener un seguimiento con git.

<pre>git init #inicializa el proyecto en rama master #cambiar el nombre de la rama master a main git branch -m main git status #muestra estado de la rama actual git add archivo #agrega archivo del area de trabajo al staging area git add . #agrega todos los archivos pendientes al staging</pre>
---

```
git add -A #agrega todos los archivos pendientes
git restore archivo #regresa archivo desde el staging área("elimina cambios")
git diff archivo #muestra diferencias entre archivo del staging area y el actual
git commit -m "mensaje" #crea snapshot del proyecto (una versión)
git log #muestra los snapshots existentes
git log --oneline # con menos informacion
git log --oneline --graph# con grafica
git log --pretty=oneline #version compacta
#o con información específica
git log --pretty=format:"%h - %an, %ar : %s"
#regresar a un commit específico
git checkout id-commit
```

## Videos de referencia

Instalar git: <https://www.youtube.com/watch?v=RAN58Qjf5uY>  
crear cuenta y configurar: <https://www.youtube.com/watch?v=LzVfVs5n3Gw&t=40s>

git push: [https://www.youtube.com/watch?v=osO\\_eYMIKxM](https://www.youtube.com/watch?v=osO_eYMIKxM)  
git clone, git pull: <https://www.youtube.com/watch?v=IWnW0svZ9JQ&t=79s>

Ramas:  
ramas, basico: <https://www.youtube.com/watch?v=gjKKtQVVCZU>  
diff, merge: <https://www.youtube.com/watch?v=W8rwULnu9nA>

## Trabajar con proyectos

### Bajar proyecto desde git hub

Si desea bajar un repositorio de GitHub, por ejemplo este:  
(<https://github.com/jflorespampano/md>):

```
#arrancar (git bash o ps)
#crear carpeta proyecto
mkdir proyecto
cd proyecto
#inicializar git para esta carpeta
git init
#cambiar el nombre de la rama principal a main
git branch -m main
#agregar un remoto con la dirección de su repo de github
git remote add origin https://github.com/jflorespampano/md
#bajar el proyecto
```

```
git pull origin main
# listo ha bajado el proyecto
# Si tenemos permiso y queremos hacer cambios en el proyecto
# subir los cambios al repo
#agregamos todos los archivos al staging para hacer commit
git add .
# creamos un commit
git commit -m "comentario"
# actualizamos nuestro proyecto por si hubo cambios en el repo
git pull origin main #por si hubo algún cambio en el remoto
git push origin main #cargar mis cambios al remoto
```

## Clonar proyecto

Las tareas anteriores se pueden hacer con un solo comando de la siguiente forma:

```
#lo anterior se puede abreviar clonando un remoto lo que deja inicializado el git y un repo del
remoto además crea una copia completa con todos sus commits
#descarga (clona) un proyecto desde un repositorio
git clone https://github...
```

## 4 subir un nuevo repositorio

1. Creamos nuevo repositorio en GitHub (pej. minuevoRepo)
2. Creamos un nuevo proyecto en tu disco
3. Agregamos el archivo (.gitignore) para indicar los archivos/carpetas que no serán respaldados

Ejemplo de Archivo: “.gitignore”

```
node_modules
```

4. Ir a la carpeta del proyecto que deseamos subir

```
```sh
```

```
# ir a la carpeta del proyecto en el disco
```

```
# abriendo directamente la carpeta con Power Shell (presiona shift + clic derecho/abrir power shell)
```

```
# o abrir git bash sobre esa carpeta
```

```
# crear un archivo README.md si no existe
```

```
echo "# nodeServerBasico" >> README.md
```

```
# inicializar proyecto
```

```
git init
```

```
git add .
```

```
git commit -m "first commit"
```

```
# cambiar nombre de rama a main
```

```
git branch -M main
```

```
# agregar el repositorio remoto creado en github
```

```
git remote add origin https://github.com/jflorespampano/minuevoRepo.git
```

```
# subir el repositorio
```

```
git push origin main
```

```
#listo
```

```
# De hecho cuando crea un nuevo repo en github, el github le da la lista de comandos que tiene que ejecutar para subir su repo.
```

```
#para subir nuevos camios de tu proyecto a tu remoto, hacer:
```

```
git add .
```

```
git commit -m "commit con nuevos cambios"
```

```
git pull origin main #actualiza tu proyecto por si hubo cambios en el remoto hechos por otro programador de tu equipo
```

```
git push origin main #respaldar tu proyecto en el remoto
```

```
```
```

## 5 mas sobre remotos

```
```sh
```

git remote -v #muestra los remotos actuales

git remote rm name #remueve un remoto

git remote rename old-name new-name #renombra un remoto

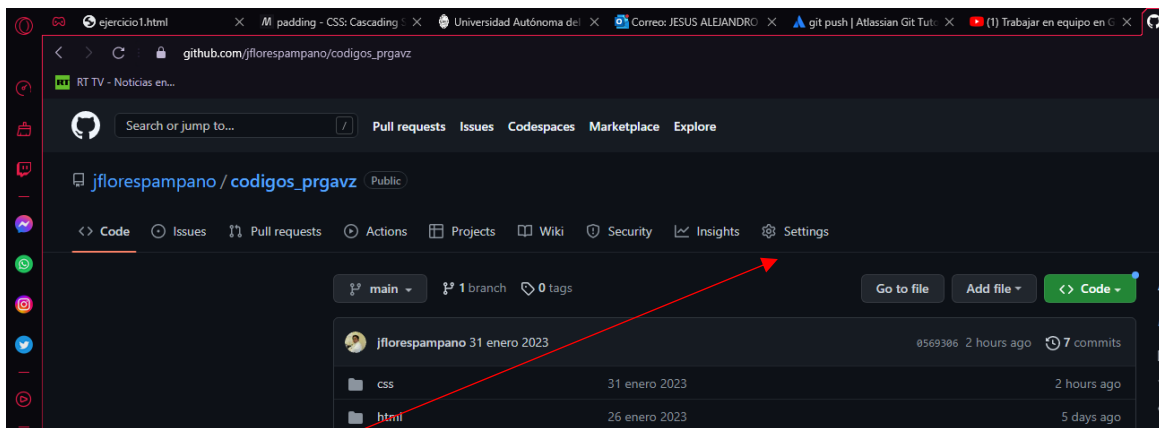
```
```
```

## Colaboradores.

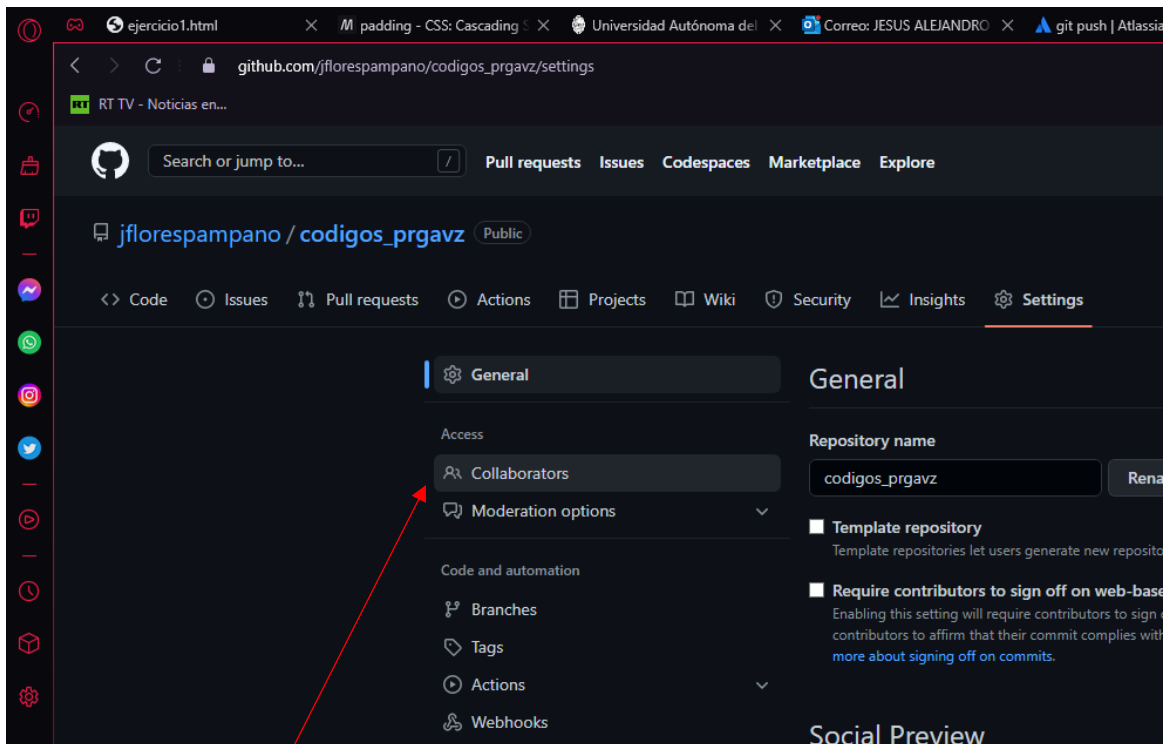
El colaborador debe tener una cuenta en GitHub

Para agregar colaboradores hacer:

1 En mi proyecto:

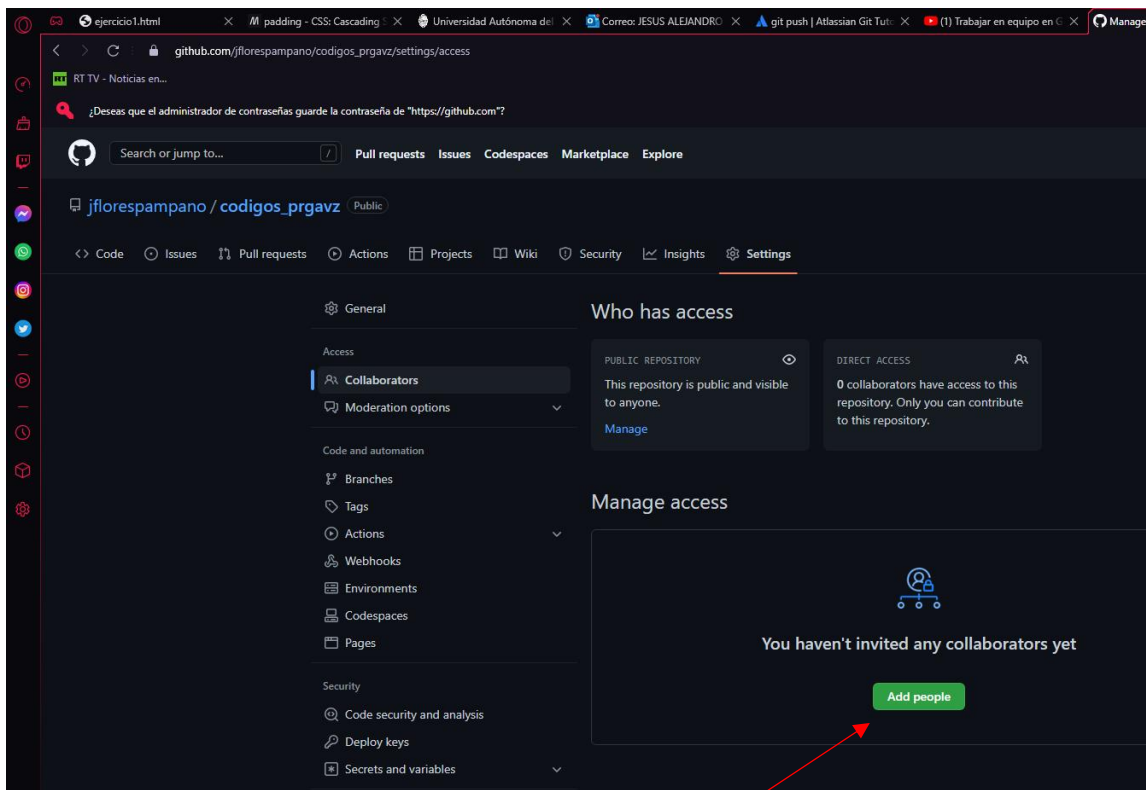


2 Vamos a settings



3 Clic en colaboradores

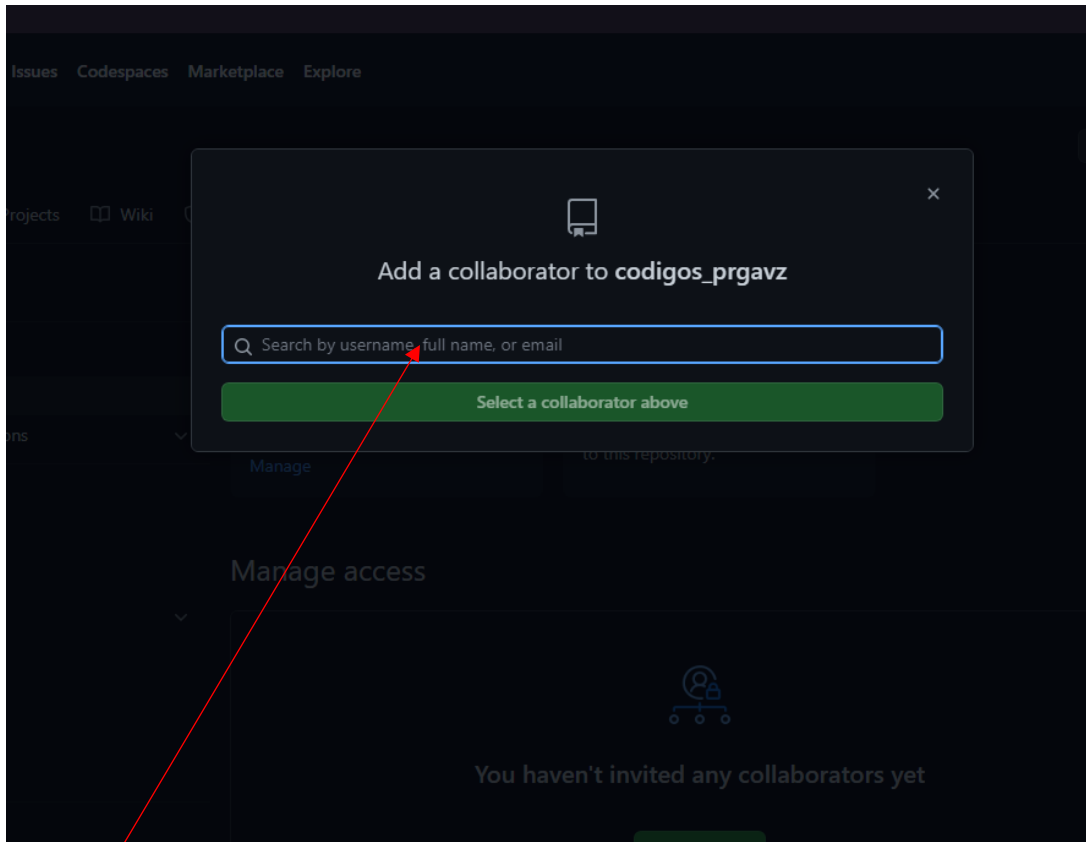
Si solicita password debe dárselo y va a:





4 Clic en *add people*, para agregar el colaborador

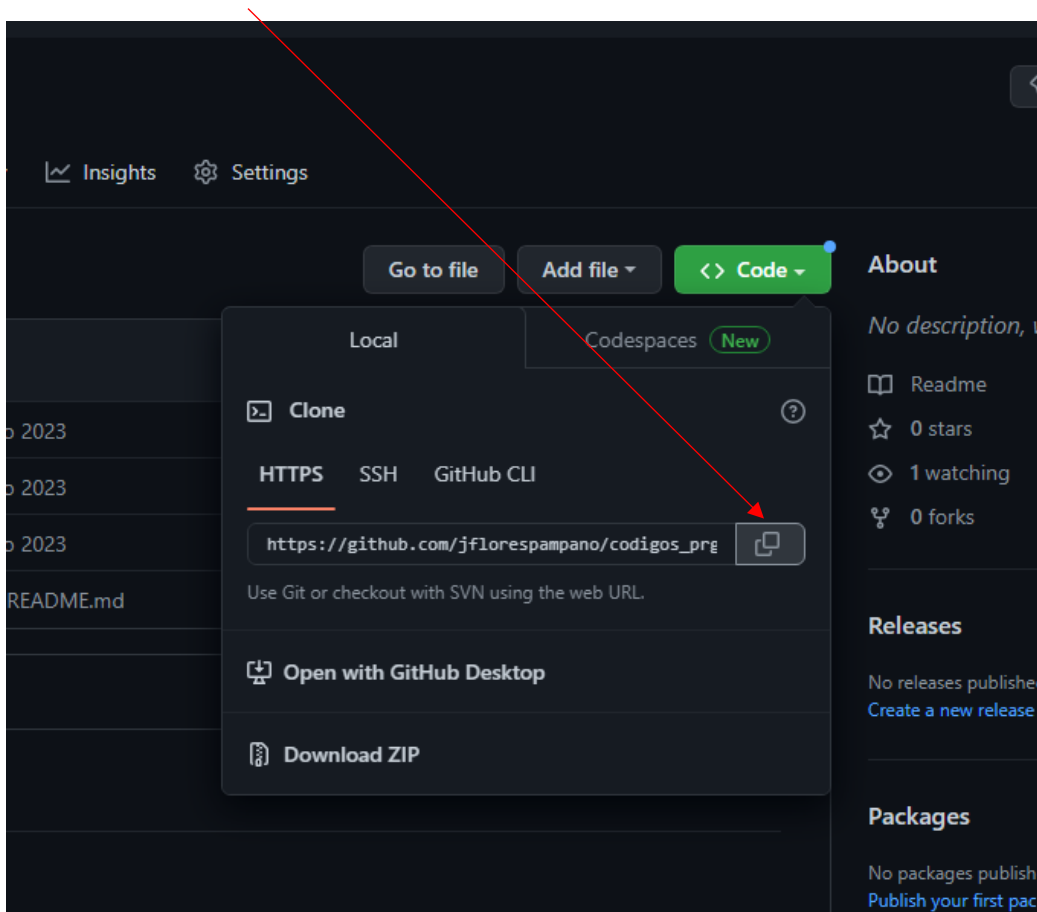
Muestra la ventana



Colocar ahí el correo electrónico con que se registró el colaborador en github.

Ahora la persona invitada puede ver el proyecto dentro de su GitHub.

1 En el github del colaborador, deberá ir a el botón verde code y ahí copiar la dirección http que le pedirá al clonar el repositorio:



2. Debe ir a su equipo y en su git deberá clonar así:

En una ventana de comando:

***\$git remote add origin <dirección http que copio>***

***\$git clone origin***

3 hacer una rama (branch) de trabajo:

***\$git branch nuevarama***

Si ejecutamos: ***git branch <enter>*** mostrara que hay 2 ramas: main y nuevarama

Ahora vamos a la nueva rama

***\$git checkout nuevarama***

Hacemos los cambios o agregamos código o archivos etcétera, después con:

***\$git status*** #para ver cómo va

**\$git add .** #para agregar los cambios al área de trabajo

**\$git commit -m "nuevo commit"** #crea una instantánea del proyecto

**\$git log --oneline** #para mostrar los diferentes commit si es necesario

**\$git diff clave comit1 clave commit2** #para ver las diferencias entre diferentes commit si es necesario

Ahora que termino sus cambios.

1 Ira a la rama main:

**\$git checkout main**

**\$git pull origin main** #para traer la última versión del proyecto a la rama main

**\$git merge nuevarama** # fusiona los cambios de la rama nueva con la rama main

Ahora actualizar la rama ya fusionada con la rama en el origin

**\$git push origin master**

## Miscelánea de comandos.

git restore -- archivo rehace archivo, regresa en los cambios

git diff archivo muestra diferencias

'antes de cambiar de rama debes guardar los cambios de la rama actual

'basicamente hacer git add . y git commit -m "mensaje"

### deshacer cambios en una rama

primero

git log --oneline esto da la rama con su id

git revert id esto deshace el commit

### merge fusiona tu rama con la rama master

recomendación antes:

git checkout master 'cambiar a la rama master

git fetch <remoto> recupera todas las ramas del repositorio

'ahora si

git merge <nombre de la rama a fusionar>

'''

desde github

git clone <https://link...> descarga (clona) un proyecto desde un repositorio

'enviar proyecto a un remoto <nombre remoto yo se lo pongo, es el alias del url>

git remote add <nombre remoto> URLrepositorio.git 'agrega un url al proyecto

git remote -v muestra las url agregadas

git remote rm <nombre remoto> elimina una url remota

'para enviar una rama local a un repositorio

git push <nombre remoto> <nombre rama>

git pull <nombre remoto> recibe actualizaciones del remoto es la combinacion de git fetch y git merge.

## Actividad 1

Objetivo: aprender a trabajar con git y github

Requerimientos:

1. Tener instalad NODE.
2. Tener instalado Git
3. Tener instalada la consola bash
4. Tener instalado el editor de código VSCode

Desarrollo:

1. Lea la introducción a git en estas notas.
2. Cree una carpeta llamada 'operaciones<su nombre>', por ejemplo, si usted se llama alex su carpeta se llamará: 'operacionesalex'.
3. Inicialice su carpeta con git, asegúrese de que la rama principal se llame main.
4. Cree un archivo llamado index.js en su carpeta, donde programará las funciones en Node JS: suma y resta no agregue multiplicación aun (vea ejemplo: aplicación node... al final de estas notas).
5. Haga un commit con message = "operaciones version inicio".
6. Agregue la función multiplicación a su archivo index.js.
7. Muestre la diferencia entre su actual archivo index.js y el que almaceno en el staging area
8. Elimine la función suma
9. Muestre la diferencia entre su actual archivo index.js y el que almaceno en el staging area
10. Recupere la versión anterior de su archivo index.js almacenado en el staging area
11. Modifique su proyecto de manera que las funciones sumas y restar estén en un módulo aparte.
12. Cree un nuevo commit con el resultado del punto 11.
13. Muestre los 2 commit (snap shot).

14. Cambie entre los 2 commit y vea el resultado en su editor de código / carpeta
15. Cree un repositorio llamado 'operacionesalex' (o el nombre que hay puesto a su carpeta) en GitHub y suba su proyecto.
16. Agregue a su proyecto la operación división y actualice su proyecto en GitHub
17. Obtenga la liga a un repositorio de su compañero de clase y baje el repositorio en su disco.

Solución:

|   |   |
|---|---|
| 1 | Leer este contenido   |
| 2 | <p>Crear carpeta en Windows o si lo desea abra una consola de ps o bash y ahí haga:</p> <pre>mkdir operacionesalex cd operacionesalex</pre>   |
| 3 | <p>Inicializar git en la carpeta</p> <pre>git init</pre>  |
| 4 | <p>Crear el archivo index.js:</p> <pre>const suma=(a,b)=&gt;a+b const resta=(a,b)=&gt;a-b  console.log(resta(9,2)) console.log(suma(9,2))</pre>   |
| 5 | <p>Para hacer un commit, debemos primero pasar nuestros archivos al staging área y después hacer commit:</p> <pre>git status #muestra en rojo los archivos que aún no tienen seguimiento, 'index.js' git add . #agrega los archivos pendientes al staging area git commit -m "operaciones version inicio" #hace el commit</pre> |
| 6 | <p>Agregar multiplicación al archivo index.js</p> <pre>const suma=(a,b)=&gt;a+b const resta=(a,b)=&gt;a-b const multiplicacion=(a,b)=&gt;a*b  console.log(resta(9,2)) console.log(suma(9,2))</pre>  |
| 7 | <p>Muestre la diferencia en el index.js en el staging área y el de su carpeta de trabajo</p> <pre>git diff index.js</pre> <p>Observe que en verde pone las líneas que se agregaron a su archivo precedidas por un más.</p>  |
| 8 | <p>Eliminar la función suma</p> <pre>const resta=(a,b)=&gt;a-b const multiplicacion=(a,b)=&gt;a*b const division=(a,b)=&gt;a/b  console.log(resta(9,2)) console.log(suma(9,2))</pre>  |
| 9 | <p>Mostrar nuevamente las diferencias</p> <pre>git diff index.js</pre> <p>Observe que en rojo pone las líneas que se eliminaron de su archivo precedidas por un menos.</p>  |

|    |   |
|----|---|
| 10 | Recuperar la version de index.js almacenada en el staging area<br><i>git restore index.js</i><br>Observe que su Código volvió al estado anterior  |
| 11 | La aplicación quedara con 2 archivos que se deben ver así:<br>index.mjs<br><div> <i>import {suma, resta} from './operaciones.mjs'</i><br/> <i>console.log(resta(9,2))</i><br/> <i>console.log(suma(9,2))</i> </div><br>operaciones.mjs<br><div> <i>const suma=(a,b)=&gt;a+b</i><br/> <i>const resta=(a,b)=&gt;a-b</i><br/> <br/> <i>export {suma, resta}</i> </div> |
| 12 | Crear nuevo commit<br><i>git add .</i><br><i>git commit -m "version con modulos"</i>  |
| 13 | Mostrar los commits<br><i>git log --oneline</i><br>Pruebe Tambien:<br><i>git log --oneline --graph</i><br>Vea que muestra los 2 commit con un identificador para cada uno   |
| 14 | <i>git checkout id-commit</i>   |
| 15 | ...   |

Entregables: muestre al profesor el resultado. Él lo anotara en una lista de cotejo.

## Actividad 2

Objetivo: aprender a trabajar con git y github

Requerimientos:

1. Tener instalad NODE.
2. Tener instalado Git
3. Tener instalada la consola bash
4. Tener instalado el editor de código VSCode

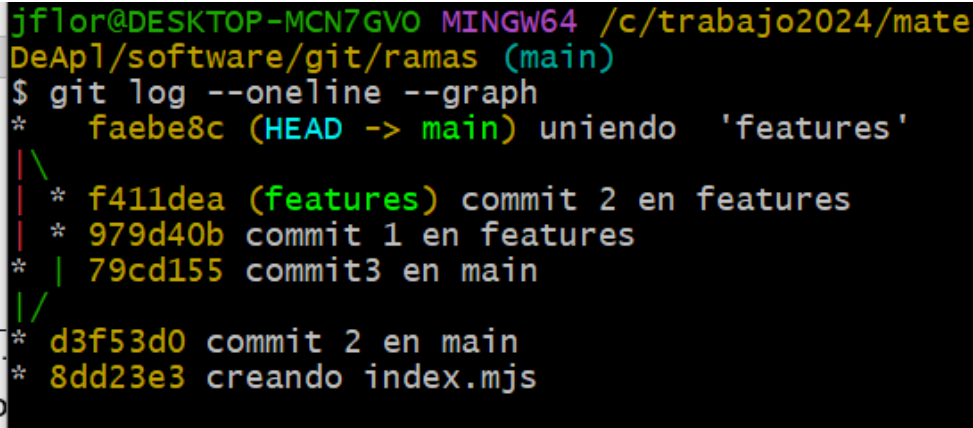
Desarrollo:

1. Cree una nueva carpeta llamada ramas e inicialice con git.
2. Cree un archivo nuevo llamado index.js.
3. Póngalo en el staging area
4. Haga un commit con message="commit 1 en main".
5. Cree un archivo llamado about.js en su carpeta y agréguelo al staging area.
6. Haga un commit con message = "commit 2 en main".
7. Cree una nueva rama llamada features.

8. Regrese a la rama main y agregue el archivo contact.html.
9. Agréguelo al staging área
10. Haga un commit con message="commit 3 en main"
11. Muestre una gráfica del historial de la rama main
12. Regrese a la rama features y muestre el historial (vea que no existe el commit 3)
13. En la rama features agregue el archivo 'change1.html' póngalo en el staging área y haga un commit con message="commit 1 en features".
14. En la rama features agregue el archivo 'change2.html' póngalo en el staging área y haga un commit con message="commit 2 en features".
15. En la rama features muestre el historial.
16. En la rama main muestre el historial grafico.
17. Haga un merge para agregar la rama features a la rama main.
18. Muestre el historial grafico de main.

solución

|      |   |
|------|---|
| 1    | git init  |
| 2    | touch index.js  |
| 3    | git init  |
| 4    | git add .<br>git commit -m "creando main.js"  |
| 5    | touch about.js<br>git add .   |
| 6    | git commit -m "commit 2 en main"<br>veamos los commits:<br>git log --oneline --graph                                      |
| 7    | #Crear nueva rama e ir a ella<br>git checkout -b features<br>#Mostrar la rama en que nos encontramos<br>git branch --list |
| 8    | git checkout main<br>touch contact.html   |
| 9,10 | git add .<br>git commit -m "commit3 en main"  |
| 11   | git log --oneline --graph   |
| 12   | git checkout features<br>git log --oneline --graph  |
| 13   | touch change1.html<br>git add .<br>git commit -m "commit1 en features"  |
| 14   | touch change2.html<br>git add .<br>git commit -m "commit2 en features"  |
| 15   | git log --oneline --graph<br>#observe los commits (cada linea con *)  |
| 16   | git checkout main<br>git log --oneline --graph  |

|    |   |
|----|---|
| 17 | git merge features #llene el commit en el editor si se lo pide  |
| 18 | git log --oneline --graph<br>deve ver algo como esto:<br> <pre> jflor@DESKTOP-MCN7GVO MINGW64 /c/trabajo2024/mate DeAp1/software/git/ramas (main) \$ git log --oneline --graph * faebe8c (HEAD -&gt; main) uniendo 'features'     * f411dea (features) commit 2 en features   * 979d40b commit 1 en features   * 79cd155 commit3 en main  / * d3f53d0 commit 2 en main * 8dd23e3 creando index.mjs </pre> |

Entregables: muestre al profesor el resultado. Él lo anotara en una lista de cotejo.

## Ejemplo: aplicación Node para operaciones aritméticas.

1. Asegúrese de tener instalado Node, de no ser así, instálelo desde ('<https://nodejs.org/en>').
2. Asegúrese de tener instalado Visual Studio Code, de no ser así instálelo desde: '<https://code.visualstudio.com/>'
3. Cree una carpeta, por ejemplo, operaciones.
4. Abra Visual Studio Code en esa carpeta.
5. Cree el archivo operaciones.js
6. Agregue el código:

```

const suma=(a,b)=>a+b
const resta=(a,b)=>a-b
const multiplicacion=(a,b)=>a*b

console.log(multiplicacion(9,2))
console.log(suma(9,2))

```

7. Desde una consola de ps o bash ejecute así:

Node operaciones.js

## Referencias

1. [https://www.youtube.com/watch?v=J\\_UOPpYKPdQ](https://www.youtube.com/watch?v=J_UOPpYKPdQ)



2. <https://git-scm.com/docs>