

**GA7-220501096-AA1-EV03 identifica herramientas de versionamiento**

**DANIEL ESTEBAN SANDOVAL CASTAÑEDA**

**APRENDIZ SENA**

**TECNOLOGIA EN ANALISIS Y DESARROLLO DE SOFTWARE**

**FICHA: 2721402**

**ABANED MANRIQUE CALDERON**

**INSTRUCTOR VOCERO**

**CENTRO DE FORMACION AGROINDUSTRIAL LA ANGOSTUR**

**2023**

## Contenido

Evidencia .....	3
Introducción .....	3
git.....	4
Github.....	10
Conclusiones .....	12

## Evidencia

- Evidencia de conocimiento: GA7-220501096-AA1-EV03 identifica herramientas de versionamiento

Tomando como referencia el componente formativo “Integración continua”, realice una tabla con las diferencias

entre el sistema de control de versionamiento git local y git remoto.

Elementos para tener en cuenta en el documento:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe

tener como mínimo una portada, introducción, objetivo, tablas con diferencias y características y comandos

de git local y git remoto.

- Realice una tabla con la descripción de los comandos básicos de git remoto y git local

## Introducción

Git es un sistema de control de versiones distribuido que permite rastrear cambios en archivos a lo largo del tiempo, facilitando la colaboración entre desarrolladores en proyectos de software.

GitHub, por otro lado, es una plataforma en línea que utiliza Git para alojar proyectos de código abierto y facilitar la colaboración entre desarrolladores. Permite a los equipos trabajar juntos en proyectos, realizar seguimiento de problemas, realizar solicitudes de extracción y mucho más, todo dentro de un entorno colaborativo.



Aspecto	Git	GitHub
Definición	Sistema de control de versiones distribuido.	Plataforma en línea para alojar proyectos y colaboración en Git.
Funcionalidad	Gestiona versiones locales de archivos y permite la colaboración en un proyecto.	Proporciona funciones de alojamiento de repositorios, seguimiento de problemas, solicitudes de extracción, gestión de colaboradores, entre otros.
Utilización	Se utiliza en la línea de comandos mediante comandos como git init, git add, git commit, etc.	Se accede a través de una interfaz web y se integra con Git para proporcionar funcionalidades adicionales.
Almacenamiento	Almacena versiones de archivos y metadatos de forma local en el sistema de archivos del usuario.	Almacena repositorios de Git en servidores remotos accesibles a través de Internet.
Colaboración	Permite la colaboración entre desarrolladores en un mismo repositorio o proyecto.	Facilita la colaboración en proyectos al permitir a los desarrolladores compartir código, revisar y fusionar cambios, y colaborar en el desarrollo de software.

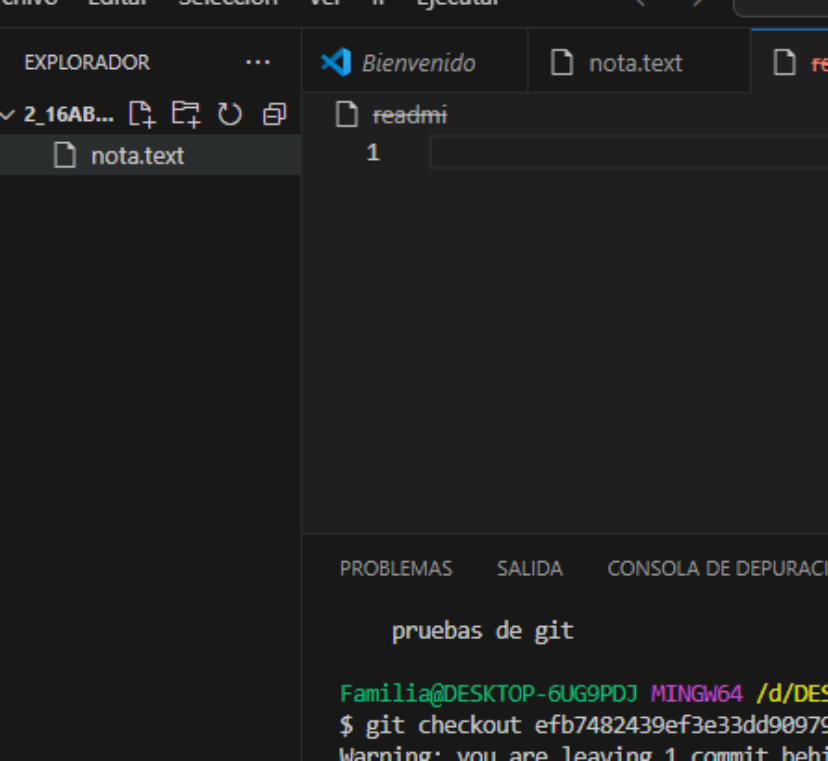
git

información	comandos	
Para saber que versión de git tengo instalado	git -v o git --version	<pre> ● PS D:\DESAROLLO\FELIPE\2_16abril&gt; git --version git version 2.44.0.windows.1 ○ PS D:\DESAROLLO\FELIPE\2_16abril&gt; </pre>
Limpiar consola	clear	
Muestra los comandos de git	git --help	
Es para dar el nombre de nuestro identificador por lo general se hace una vez	git config --global user.name "Daniel Sandoval"	<pre> &gt; git config --global user.name "Daniel Sandoval" &gt; </pre>

	git config --global user.email "correo"	<pre>git config --global user.email "desandoval62@gmail.com"</pre>
Muestra todas las configuraciones de mi git	git config --list	
Para saber el user name y el correo	\$ git config user.name \$ git config user.email	<pre>\$ git config user.name Daniel Sandoval</pre>
Idealizar la carpeta con GIT Lo que hace es amistar todas las versiones de nuestro proyecto. Se deja quieto y es una carpeta oculta	git init	<pre>\$ git init Initialized empty Git repository in D:/DESAROLLO/FELIPE</pre> 
Es para hacer seguimiento La herramienta principal para determinar qué archivos están en qué estado es el	git add y nombre de archivo	

comando git status.		
Nos muestra que archivos si tiene segumien to y cuales no	\$ git status	<pre> \$ git status On branch master  No commits yet  Changes to be committed:   (use "git rm --cached &lt;file&gt;..." to unstage)         new file:   nota.text         new file:   nota2.text         new file:   nota3.text  Changes not staged for commit:   (use "git add/rm &lt;file&gt;..." to update what will be commi   (use "git restore &lt;file&gt;..." to discard changes in worki         deleted:    nota2.text  Untracked files:   (use "git add &lt;file&gt;..." to include in what will be comm         otro.text </pre>
Es para que tenga segumien to el archivo	\$ git add	<pre> Familia@DESKTOP-6UG9PDJ \$ git add otro.text </pre>
SI EN DADO CASO SE ME BORRA UN ARCHIVO PUEDO RECUPER ARLO recuerda volver a verificar con GIT STATUS	git restore	<pre> Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16ab \$ git restore nota2.text  Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16ab \$ git status On branch master  No commits yet  Changes to be committed:   (use "git rm --cached &lt;file&gt;..." to unstage)         new file:   nota.text         new file:   nota2.text         new file:   nota3.text         new file:   otro.text </pre>

<p>escribir el mensaje de confirmación directamente en el comando commit utilizando la opción -m</p> <p>Es guardar una copia con una nota</p>	<p>git commit -m</p>	<pre>\$ git commit -m "pruebas de git" [master (root-commit) 27e61c2] pruebas de git 4 files changed, 3 insertions(+) create mode 100644 nota.text create mode 100644 nota2.text create mode 100644 nota3.text create mode 100644 otro.text</pre>
<p>Para saber en que estado esta nuestros archivos usamos el comando</p>	<p>git status -s</p>	<pre>\$ git status -s A o.text</pre>
<p>Es para hacer todo el seguimiento de todos los archivos</p>	<p>git add .</p>	

<p>Es como un historial de todos los commit que hemos hecho</p> <p>Lo amarillos son los identificadores</p>	<p>git log</p>	<pre>Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16abril \$ git log commit 69d14d46b59b5c4d615410b7fde8d55aa95fafaf (HEAD -&gt; master) Author: Daniel Sandoval &lt;desandoval62@gmail.com&gt; Date: Fri Apr 19 01:15:23 2024 -0500      mire las letras  commit 27e61c27424cf28b28e5e0d51a7bed4d455205b0 Author: Daniel Sandoval &lt;desandoval62@gmail.com&gt; Date: Fri Apr 19 00:44:36 2024 -0500      pruebas de git  Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16abril \$</pre>
<p>Esto nos permite volver el código al estado que queramos dependiendo del identificador que seleccionemos</p> <p>Si hay algo creado se nos borra o queda marcado de la siguiente manera</p>	<p>git checkout identificador</p>	 <pre> Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16abril \$ git checkout efb7482439ef3e33dd90979f Warning: you are leaving 1 commit behind (HEAD -&gt; master). To discard the tracked changes, use:   git reset --hard To discard the tracked changes and return to the original state, use:   git checkout --discard-changes </pre>




Para saber en que foto estoy	git branch	<pre>\$ git branch * (HEAD detached at efb7482) master  Familia@DESKTOP-6UG9PDJ MINGW64 /d/DESAROLLO/FELIPE/2_16ab \$</pre>
Me deja el proyecto como lo estaba trabajando	git checkout 'master'	
Una vez que identifique el commit en el que se eliminaron los archivos, ver los cambios específicos realizados en ese commit. Esto te mostrará qué archivos fueron eliminados.	git show <hash_del_commit>	
Para eliminar el identificador	Git rm identificador	<pre>\$ git rm a34d53452107c9b81fd0973d61f02</pre>


## Github

Required fields are marked with an asterisk (\*).

**Owner \*** **Repository name \***


 DannielS / PRUEBA\_SENA\_ABRIL24 **Le damos un nombre al proyecto**


✔ PRUEBA\_SENA\_ABRIL24 is available.

Great repository names are short and memorable. Need inspiration? How about [curly-octo-couscous](#) ?

**Description** (optional)

**Le podemos dar una descripción**

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit. **Lo dejamos público o privado depende el caso**

☐  **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

.gitignore template: **None** **Aquí es como lo idealicemos lo podemos hacer manual o desde aqui**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**


License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

---

**Create repository**

## Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/Dannnie15/PRUEBA_SENA_ABRIL24.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## ...or create a new repository on the command line

```
echo "# PRUEBA_SENA_ABRIL24" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Dannnie15/PRUEBA_SENA_ABRIL24.git
git push -u origin main
```

Si ya hicimos todo lo de lo  
De git pasamos de una vez a  
este paso

## ...or push an existing repository from the command line

```
git remote add origin https://github.com/Dannnie15/PRUEBA_SENA_ABRIL24.git
git branch -M main
git push -u origin main
```

Si no podemos usar este ruta  
Debemos aceptar

## ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

```
Famillia@DESKTOP-8XGPP3 MINGW64 /d/DESAROLLO/FELIPE/2_16abr11 (main)
$ git remote add origin https://github.com/Dannnie15/PRUEBA_SENA_ABRIL24.git
error: remote origin already exists.
```

```
Famillia@DESKTOP-8XGPP3 MINGW64 /d/DESAROLLO/FELIPE/2_16abr11 (main)
$ git branch -M main
```

```
Famillia@DESKTOP-8XGPP3 MINGW64 /d/DESAROLLO/FELIPE/2_16abr11 (main)
$ git push -u origin main
Info: please complete authentication in your browser...
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (13/13), 1.05 KiB | 30.00 KiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Dannnie15/PRUEBA_SENA_ABRIL24.git
 * [new branch]    main -> main
branch 'main' set up to track 'origin/main'.
```



## Como clonar un repositorio de git

Cuando nos añadan a un repositorio lo que debemos es dar clip en code y copiar el HTTPS del repositorio luego de esos nos vamos a visualstudio y le vamos a dar el comando **GIT CLONE Y PEGAMOS EL CODE** y luego le damos **CD NOMBRE DEL PROYECTO TAL CUAL ESTA ESCRITO** y para que lo inicie con un nuevo Visual CODE .

CUANDO YA HAGAMOS UNA MODIFICACION EN EL CODIGO HACEMOS LOS COMADOS DE GIT  
Haciendo el `git add git status git commit - m` y asi

Luego lo de tener todo el git bien lo vamos a subir a GIT HUB con el comando

### git push origin main

Cuando hace una modificación o sube mas codigo lo que pasa es que en el git hub se vera en verde por la modificacion

Si yo quiero descargar las modificaciones

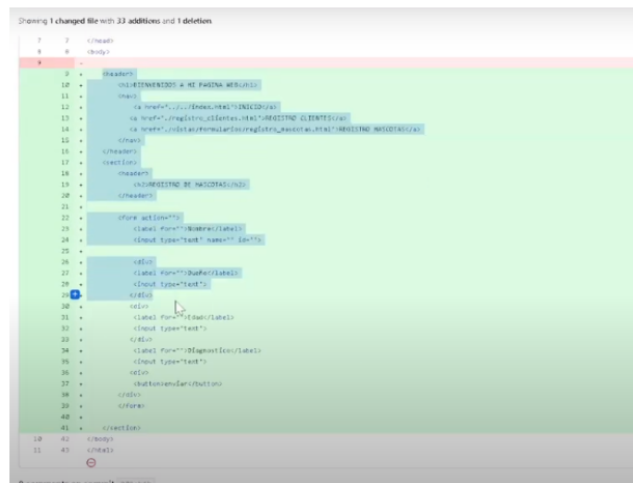
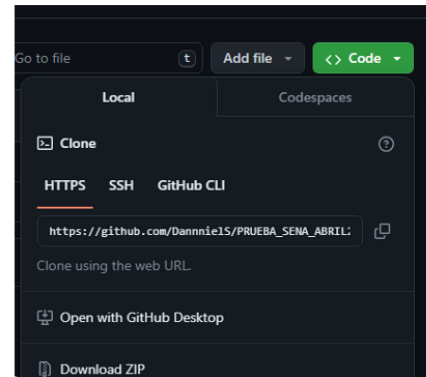
Uso en la terminal el comando

### Git fetch origin main

Es para que observe si hay modificaciones en la nube

### GIT MERGE

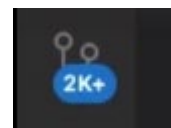
Este comando es para que lo una con lo que tenemos en forma local



## Conclusiones

Git git hub pensé que era lo mismo peor me doy cuenta que son dos herramientas completamente diferentes peor entre las dos son una gran ayuda.

En cursos pasados tocamos un poco el tema peor en verdad siempre fue un problema y en verdad lo vi muy incensario y no lo entendía incluso mi computador tubo un error que hasta el dia de hoy no e logrado muy bien comprender peor lo pude solucionar pero debí borrar lo que quería subir en un repositorio.



Pero en la primera clase de git logre comprender muy bien todas las ayudas de git y github y la forma de trabajar en grupo o en un equipo