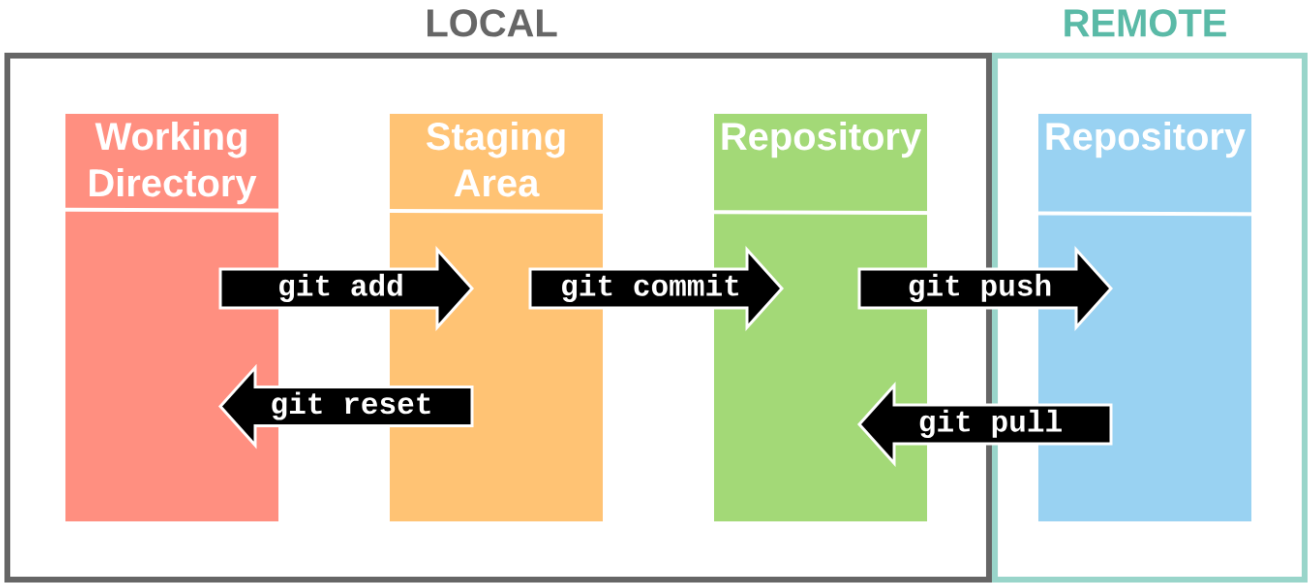
Guia de GIT

Git es un software de gestión de datos. Con esto podemos llevar un registro de los avances en proyectos, donde en caso de necesitar podremos “volver en el tiempo” en nuestro código. Por ejemplo. Imaginemos que estamos programando una aplicación como Instagram. Nos encargamos de programar la gestión de la GUI, que es lo mismo que la interfaz gráfica o también llamado front-end. Los cambios que vamos a realizar son muy grandes, nos llevara algunos meses y trabajamos con un equipo de 3 programadores.

Durante el desarrollo de un programa siempre surgen problemas, cosas que antes funcionaban y dejan de función por algún motivo. O incluso en códigos muy grandes borramos cosas que consideramos inútiles y al final resulta que son útiles. Imagínense que ustedes borran algo que parecía inútil en una aplicación grande y a los meses se dan cuenta que si era útil. Con git se puede revertir esos errores, ya sean por problemas o por necesidad de revisar versiones anteriores del código.



Esta Imagen representan la estructura interna de GIT. Existen dos partes lo que llamaremos entornos LOCAL y REMOTO.

→ Centrémonos en el entorno LOCAL. A su vez este se divide en 3 partes. Working Directory, Staging Area y el Repository.

- **Working Directory:** Esta sección es la que nosotros manipulamos constantemente. En esta parte nosotros podemos crear nuevos archivos, carpetas, modificarlos borrarlos. En resumen, es lo que ustedes manipulan directamente desde el editor de texto que estén utilizando.
- **Staging Area:** Esta sección lo que me permite es preparar mis archivos para ser enviados al repositorio. Todos los archivos que quiera subir a mi repositorio

tienen que pasar por la Staging Area antes. En este lugar preparo mis archivos para luego ser comentados y enviarlos al **repositorio.**

- **Repositorio:** Este lugar no es más que un almacén de mis archivos. Pero no es un almacén cualquiera, sino que es un almacén que guarda absolutamente todo lo hubo antes, es decir es un almacén que tiene tu proyecto desde el día 0 hasta el día 100, y si en algún momento queremos recuperar versiones anteriores de nuestros códigos podemos hacerlo de una forma muy simple.

→ Centrémonos en el entorno REMOTO. Este entorno es simplemente una copia exacta de mi Repositorio del ámbito global, pero en la nube es decir en internet.

- Al ser algo que existe en internet necesitaremos algo o alguien que nos permita guardar esas cosas en algún servidor. Existen diversos Host, un host es un servicio o una plataforma que te almacena esa información, algunos de ellos pueden ser: GitKraken, SmartGit, GitCola, Aurees, BitBuket, **Github**.

Elegimos GitHub por ser gratuito fácil de utilizar, y porque tiene una comunidad inmensa, además de ser el más utilizado en el mundo con esta herramienta.

Es importante comprender que a trabajar en un proyecto grande. No podemos enviar constantemente el código de un compañero a otro por Gmail o WhatsApp porque es un código que está siendo modificado simultáneamente por diferentes programadores. Para eso se utiliza GitHub o cualquiera de los otros Host que utilicen git. Con esto tengo una herramienta que permite subir un código para ser modificado y compartido de una forma más simple y ordenada por un grupo de desarrolladores, así como para el público.

Esta es la estructura general de git, es mucho más complejo que esto y tiene muchas más funcionalidades. Sin embargo, no entraremos tanto en detalles. Pueden ver estos videos:

Explicacion ---------- Explicaion2

Lista de comandos:

Configuración inicial:

Esto es solo necesario si vamos a utilizar un repositorio Remoto, como es el caso daremos por hecho que ya tienen creada una cuenta en GitHub. Así que abriremos cualquier terminal y escribiremos estos comandos:

Si es un pc particular:

- git config --global user.name “username”
- git config --global user.email “mail@gmail”

Si es un pc que utilizan varios programadores:

##

IMPORANTE: TIENE QUE HACERCE ESTE COMANDO EN UN REPOSITORIO LOCAL YA INICIADO (MAS ABAJO)

##

- git config user.email “mail@gmail”
- git config user.name “username”

Abrir un repositorio (Esto se hace una unica vez):

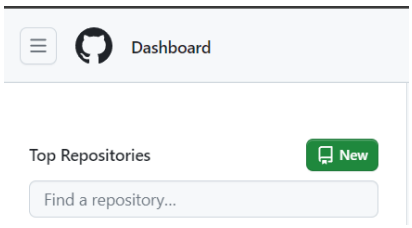
Con cualquier terminal (cmd, wsl, git bash,VisualStudioTerminal), ubicarse utilizando los comandos ‘cd’, ‘cd ..’, ubicarse en la carpeta donde queremos inicializar un repositorio.

- git init (inicializa un entorno local con el repositorio vacío)

Además de inicializar el Repositorio local hay que configurarlo en caso de queramos utilizar un repositorio remoto (Utilizaremos GitHub así que vamos a configurarlo).

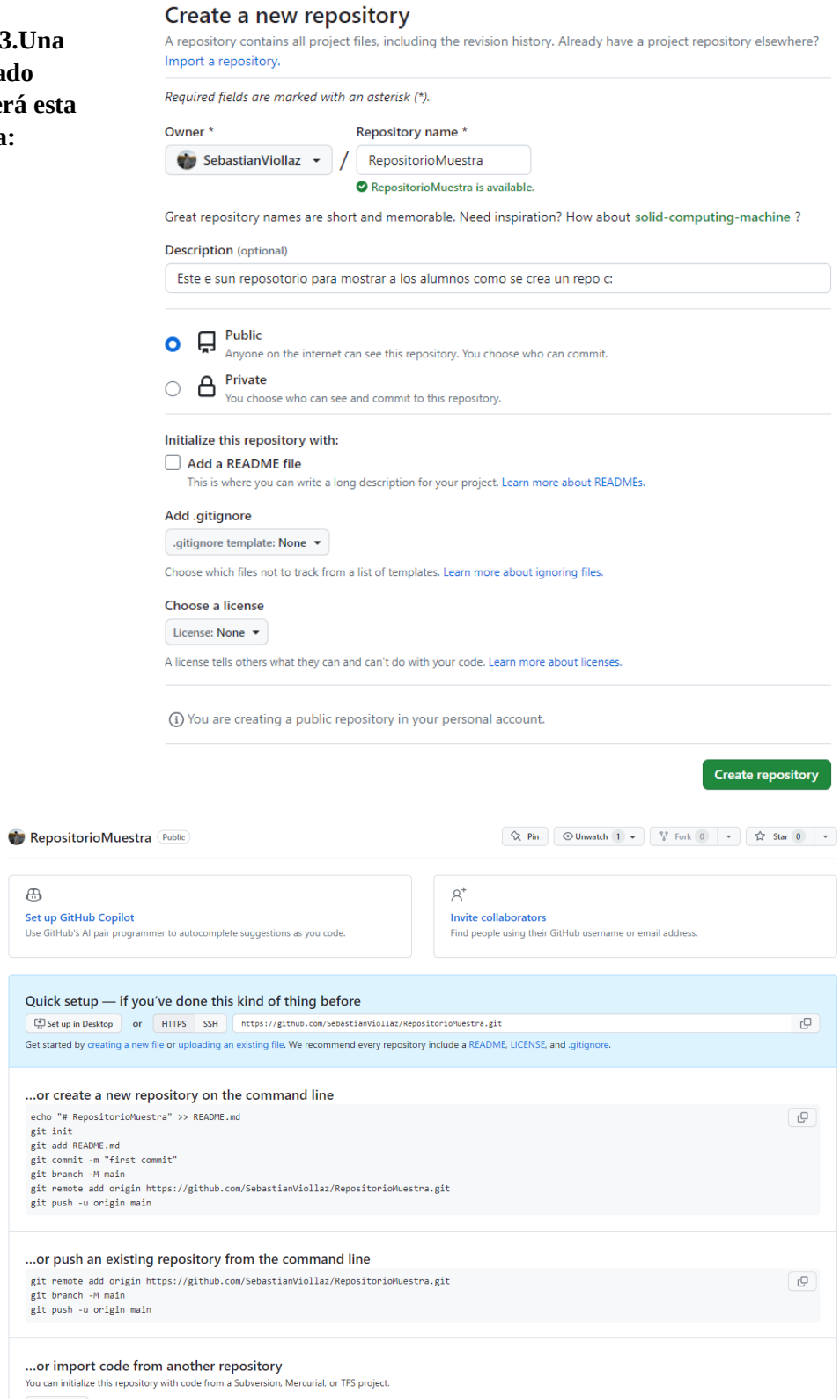
Obviamente si queremos utilizar un repositorio remoto hay que crear uno. Por lo tanto, vamos a crearlo en Github:

1.Apretar New en la página principal arriba a la derecha:



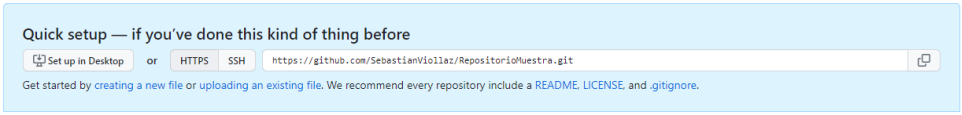
2.Detallan su repositorio, nombre, descripción y si va a ser público a privado, para fines del curso hagan sus repositorios públicos.

3.Una vez creado aparecerá esta pantalla:



Una vez creado un repositorio remoto tenemos que obtener su URL, este es muy importante porque es lo que nos ayudara a conectar nuestro repositorio en el ámbito local con el del remoto.

Este URL se encuentra aquí:



Una vez tengamos ese link tendremos que ir a la terminal de nuestra carpeta en la que inicializamos nbuestro repositorio y correr el siguiente comando:

- git remote add <ETIQUETA> <URL>

Este comando lo que hace es agregar en una etiqueta esa url, es como guardar en una variable con el noombre que nosotros queramos esa url. La forma tradicional de hacer este comand o es:

- git remote add origin <URL>

Entocnes cada vez que nosotros escribamos origin en comando posteriores estaremos haciendo referencia a ese url de ese repositorio.

PUEDO TENER VARIAS ETIQUETAS QUE HAGAN REFERENCIA A DISITNTOS REPOSITORIOS SIEMPRE Y CUENDO ESAS ETIQUETAS NO TENGAN NOMBRES IGUALES

Esos serian todos los pasos para configurar por primera vez nuestro repositorio.

Flujo de trabajo

El flujo de trabajo es el detallado en la foto de arriba.

Resumen: En nuestro WorkingDirectory modificaremos nuestros archivos, agregaremos otros y borraremos otros sin embargo estos cambios no se guardan de forma automática en nuestros repositorios, tendremos que correr una serie de comando para que esto pase.

- git add <RutadeArchivo>

Este comando lo que hace es agregar a la stagin área un archivo que queramos podemos agregar cuantos queramos, aunque no es muy recomendable hacerlo de ese modo.

- git commit –m “Mensaje que quieren”

Este comando lleva los archivos que estén en mi stagin area al Repositorio LOCAL, y además les agrega un comentario para que luego podamos saber que cambios hicimos en ese archivo (por eso esta bueno poner un mensaje que explico de forma muy corta lo que se hizo en el archivo”

- git push <ETIQUETA> <RAMA>

Este comando lleva TODO lo que este en mi repositorio LOCAL a mi repositorio REMOTO. Para el comando en <ETIQUETA> referenciaremos a la etiqueta que creamos por primera vez con el comando git remote add <ETIQUETA> <URL> y en <RAMA> haremos referencia a la rama de nuestro repositorio REMOTO a la cual queremos llevar estos cambios. Si no sabes lo que son las ramas y te interesa en la lista de videos de arriba hay 2 explicaciones, si no te interesa referencia a la rama master.