Git es un sistema de control de versiones.
Consta de 3 areas:

- Work Tree (directorio de trabajo)
- Index o Staging area (es una zona intermedia)
- Repositorio o HEAD (que apunta al ultimo commit realizado)

https://www.youtube.com/watch?v=6PNP1NmEDLU&list=PLzSFZWTjelbJenxKBbxhc8C-2qq3YeAwR

**CONFIGURACION INICIAL** (se hace por unica vez cuando se instala Git)

Hay que indicarle a Git nuestro nombre de usuario y nuestro email, ya que los commits (son los cambios de versiones que vamos haciendo sobre nuestro proyecto) deben estar asociados a un autor.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

**COMANDOS**

// Iniciar un nuevo repositorio (este comando se debe ejecutar estando en el directorio del proyecto)

```
git init
```

// Agregar un archivo o directorio al staging area

```
git add nombre_de_archivo_o_carpeta
```

// Agregar todos los archivos modificados al staging area

```
git add -A
```

// Grabar lo que esta en el staging area en el repositorio

```
git commit -m "descripcion del commit o version"
```

// Ver el estado de los archivos

```
git status
```

// Para ver las versiones (los commits realizados)

```
git log
```

// Otra forma de usar git log es enviando el contenido que muestra en pantalla hacia un archivo

```
git log > archivo_log.txt
```

// Para evitar que el archivo archivo_log.txt dentro de nustro directorio sea agregado al commit creo un archivo llamado .gitignore. Y dentro de este archivo escribir el nombre de los archivos que git ignorará.
El archivo .gitignore no debe ser ignorado por git, ya que muchos proyectos incluyen archivos de configuracion y esos archivos son agregados al archivo .gitignore

// Para volver a una version anterior o posterior (Permite moverse entre commits o versiones del proyecto)

```
git checkout id_del_commit
```

//Borrar un archivo (lo borra del Work Tree y del Staging area)

```
git rm -f "nombre del archivo"
```

//Borra archivo solo del staging area

```
git rm --cached "nombre del archivo"
```

// Borrar un commit ya realizado (dejan de existir tambien los commit posteriores).
Cuanta con tres formas: soft, mixed y hard.

```
git reset --soft id_de_commit_al_que_quiero_volver
```

--soft // borra el commit, pero no afecta (no modifica) el Work Tree ni el Staging area
--mixed // Resetea el Index pero no modifica el work tree
--hard // borra el commit y afecta el Work Tree y el Index, es decir borra las modificaciones.

Git permite la creación de ramas (inicialmente solo trabajamos en la rama principal llamada Master)
Esto permite implementar una caracteristica en particular, luego esa rama independiente se puede fusionar a la rama master.

// Lista las ramas del proyecto

```
git branch
```

// Crea una nueva rama

```
git branch nombre_de_rama_nueva
```

// Elimina una rama

```
git branch -d nombre_de_rama
```

//nos permite realizar fusiones entre ramas, debemos estar situados en la rama que absorvera los cambios de la otra al momento de usar este comando.

```
git marge nombre_de_la_rama_a_fusionar
```

// checkout ademas de permitirnos movernos entre commit, permite moverse entre ramas

```
git checkout nombre_de_la_rama
```