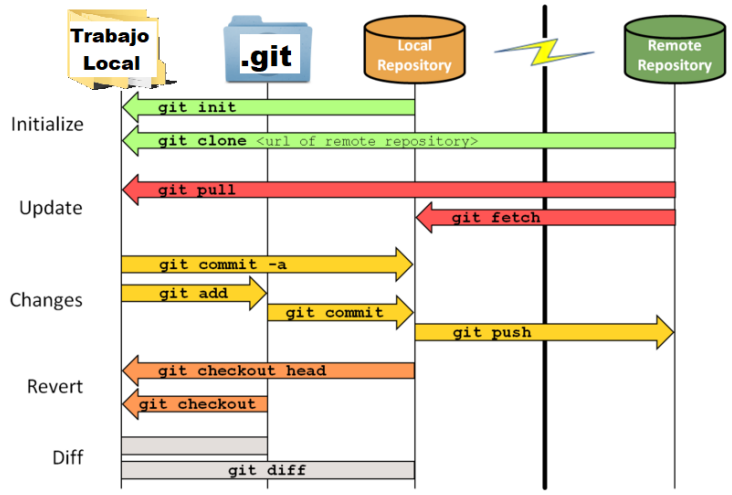
Comandos GIT

Se tiene 3 etapas: **directorio de trabajo(Trabajo Local)| área de ensayo(.gIT) | repositorio local |**



pasosa a seguir:

1-Clip derecho dentro de la carpeta ingresar "Git bash here"

2- Insertar (PUSH) Datos al Repositorio Git.

ingresar siguiente códigos:

✓ **git init**: crea un repositorio nuevo dentro de la carpeta.

-**git status -s** : nos muestra el estado de los archivos o directorios que q no fueron Seleccionados con y el "**??**" se encuentra en el local y el archivo no tiene seguimiento (que sigue en el directorio) Osea q no fue seleccionado.

```
|?? index.html| |?? Estilo.css| |?? JavaScript.js |
```

Si llegáramos a modificar algún archivo y vemos el estado muestra lo sig **Verde** (q no fue modificado) **Rojo** (que hubo modificaciones después de seleccionarlo):

```
|AM Estilos.css||A  JavaScript.js||MM index.html|
```

A(se encuentra seleccionado para hacer un commit) y **M**(se hizo un commit)

-**git add index.html** : hace un seguimiento y le agrega al área de ensayo(.Git)

```
|$ git status -s||A  index.html||?? JavaScript.js|
```

✓ **git add .** : agrega a todos lo q encuentra al ensayo.

```
|A  Estilos.css||A  JavaScript.js||M  index.html|
```

-**git commit -m "comentario"**: Uva ves seguido necesitamos hacer un commit si o si Necesariamente para introducir al repositorio.

```
|[master (root-commit) 4a306c9] comienzo de un proyecto |
      | 1 file changed, 11 insertions(+) |
      | create mode 100644 index.html    |
```

- Al modificar el archivo (agregando más cogido) q se hiso un commit, a través del status visualizamos q se modificó (M en Rojo) y no tomo las últimas modificaciones.
  Y si (M en Verde) tomo las últimas modificaciones pero antes necesito hacer un Seguimiento.

```
| M index.html||?? JavaScript.js|
```

✓ **git commit -am "comentario":** hace un commit de todo seleccionado.

-**git commit –amend :** para editar un commit .

-**git log –oneline :** muestra los commit q se hicieron en el proceso

```
2) | 17f3e77 (HEAD -> master) agregado un párrafos
1) | 4a306c9 comienzo de un proyecto  |
```

-**git reset –hard** 4a306c9 : restauro el código al comienzo donde inicio a través del código de commit se hizo.
Si es ok lanzara el siguiente msn:

```
| HEAD is now at 4a306c9 comienzo de un proyecto |
```

- Con status solo muestra: **|?? JavaScript.js|**
- pero si mostramos lo commit solo nos dará el primero q se hizo **|4a306c9 (HEAD -> master) comienzo de un proyecto |**

Prepara la conexión al repositorio:

✓ **$git remote add origin** https://github.com/nombreDeRepositorio/practica-con-Git.git

✓ **$git push origin master | $git push -u origin master :** envía los archivos al repositorio (master), es posible en el transcurso pida usuario y contraseña, que será el dato con el que se registraron al git.

3- Actualizar (PULL) Datos al Repositorio Git.

✓ **Git pull** https://github.com/nombreDeRepositorio/practica-con-Git.git : actualiza si se hizo cambios en el repositorio. Muestra 4 filas agregadas (-) q se hizo en esos archivos.

```
✓  Estilos.css    | 4 ++--
✓  JavaScript.js | 3 ++-
✓  index.html     | 3 ++-
✓  3 files changed, 6 insertions(+), 4
   deletions(-)
```

Tags: nos permite especificar las versiones de nuestros proyectos para identificar el proceso de nuestros trabajos

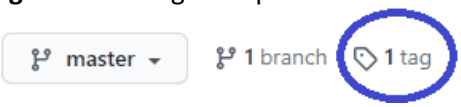
- **$ git tag 01-10-20v1 -m "Versión 1 del proyecto"**

Visualizamos de la siguiente manera a través de log --oneline:

```
a75c6fd (HEAD -> master, tag: 01-10-20v1,
      origin/master) Update JavaScript.js
```

para subir uso el sig comando:

-**git push –tags :** sube el tags al repositorio



Tiene la facilidad de descargar a través de un .zip o tar.gz

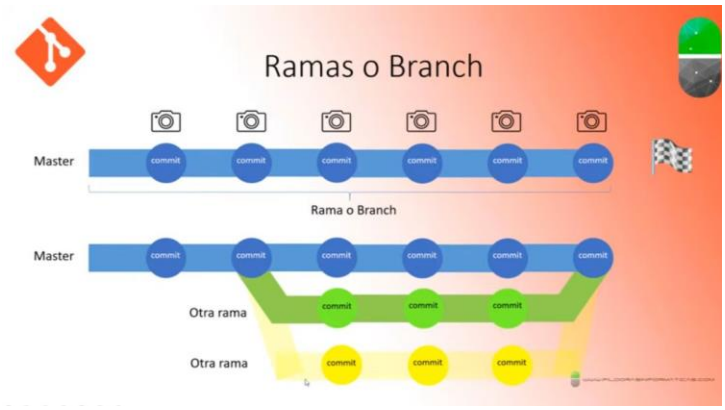
4- Clonar (Clone) Datos al Repositorio Git.

Se realiza una clonación en caso de haber perdido los datos en nuestra compu local u quisiéramos tenerlo en otro. De la siguiente manera:

✓ **Clip derecho dentro de la carpeta ingresar "Git bash here"**

- git clone https://github.com/haguilerts/practica-con-Git.git

5- Ramas o Branch de Git.



Para crear una rama(branch) parto de la rama master:

- git branch MyRama

para mostrar usamos log –oneline:

```
a75c6fd (HEAD -> master, tag: 01-10-20v1,
origin/master, MyRama) Update JavaScript.js
```

- git branch : me posiciono en q rama esto y muestra las otras. En este caso estoy en * master.
  | MyRama | |* master |
- git branch status: muestra en q rama estoy
  | On branch master |
- git checkout MyRama : me cambio a la rama **MyRma.**
- Para verificar uso **git branch** o un msn
  | Switched to branch 'MyRama' |
- Esta es mas sensilla, donde **Creare** Ramas y a ves me cambiare:
  | $ git checkout -b newRama |
- Para **Renombrar** Ramas usamos el sig código:
  | $ git branch -m ramaAntigua ramaNueva |
- Para Borrar Ramas usamos el sig código:
  | $ git branch -d ramaBorrar |
- Y para ver más Opciones q se puede hacer con las Ramas se ejecuta el siguiente comando:
  | $ git branch -h |

Una ves modificado los códigos, lo añadimos a la nueva Rama

- `$ git add .: añadimos a la nueva Rama.`
- `$ git log –oneline : muestra los resultados.`

```
a75c6fd (HEAD -> MyRama), tag: 01-10-20v1,
origin/master, master) Update JavaScript.js
```

finalmente hago el commit y envio al repositorio.

```
$ git commit -am "agregado un vento en html y js"
$ git push origin javaScript
```

a hora volveré a la rama "master" donde desaparecerán los últimos cambios y mostrando la versión anterior.

```
$ git checkout master
```

Otra forma de hacerlo Commit es abriendo un editor de texto y realizar comentarios:

$ git config --global core.editor "Code --wait"

Para finalizar guardamos y cerramos el enlace del commit mostrara el sig msn en la consola de GIT.

```
[newRama 10e08e6] miComentario
1 file changed, 2 insertions(+)
```

Con el sig código visualizaremos todo el estado completo:

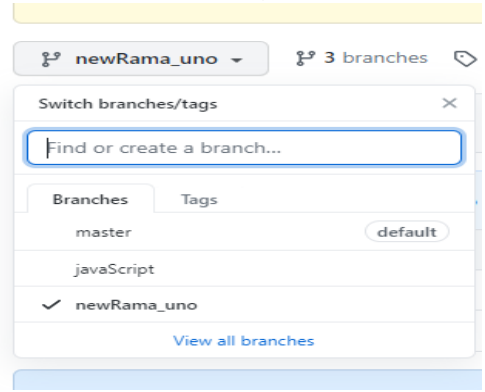
$ git log --oneline --decorate

Mostrando como resultado lo sig:

```
10e08e6 (HEAD -> newRama) miComentario
6ce432d(origin/javaScript,miRama,javaScript,
   MyRama) agregado un vento en html y js
a75c6fd (tag: 01-10-20v1, origin/master,
   master) Update JavaScript.js
```

$ git log --oneline -decorate -all : muestro todo los comit hecho en el git, tanto master y branch

$ git log --oneline -decorate -all -graph : muestra las ramas de sus ramas incluyendo la rama de master.



Finalmente, fusiono las rama master con branch. Posicionándome en la master.

$ git merge RamaBranch