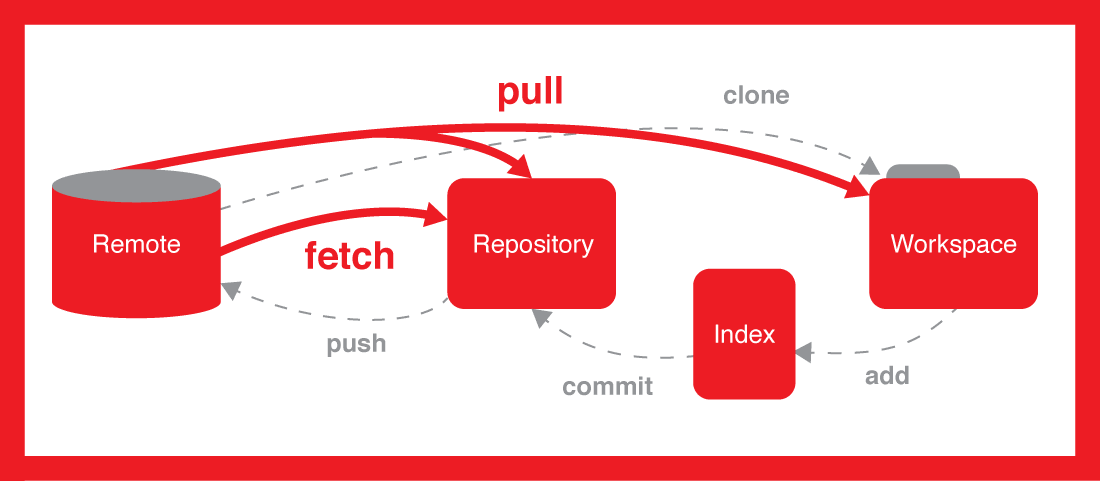
# FLUJO COMPLETO SIN CONFLICTOS

1. Entra en GitHub
2. Crea un repositorio VACIO: Repo\_iniciales
3. Sincronizar en local mediante CLONE
4. Crea el fichero “fichero.txt” de texto en local: y añade el texto: “soy un fichero”
5. Subirlo a GitHub:
   1. Git add: añadirlo al tracking, “stage”
   2. Git commit: confirmar cambios
   3. Git push: subir el fichero al servidor
6. Confirmar en la Web de GitHub el fichero subido
7. Desde GitHub; crea el fichero fichero2.txt con el contenido: “soy el fichero2”
8. Ejecuta git pull para descargar los nuevos cambios del repositorio
9. Comprueba que el repositorio se ha sincronizado correctamente
10. Ejecuta git stauts.
11. Cuántos ficheros aparecen? En que estado se encuentran?
12. Sube un fichero cualquiera, desde tu ordenador a GitHub con la opción de “UploadFiles” o arrastrando el fichero al repositorio.
13. En el repositorio web de GitHub debes tener tres ficheros; pero en local, sólo tienes dos ficheros.
14. Ejecuta git fetch
15. ¿cuántos ficheros tienes ahora en local?
16. Ejecuta git merge: para sincronizar el workspace con el repository
17. ¿cuántos ficheros tienes ahora en local?

A screenshot of a cell phone

Description automatically generated



# OTRA FORMA DE HACER LA PRIMERA SINCRONIZACIÓN

1. Borra la carpeta local del repositorio anterior: Repo\_iniciales
2. Crea una carpeta que se llame Repo\_iniciales\_manual
3. Entra en la carpeta
4. Inicializa la carpeta como git: git init
5. Sincroniza con el repositorio remoto: git remote add origin <https://github.com/>....
6. Revisa la carpeta local, ¿hay algo?
7. Git pull
8. ¿se ha sincronizado el repositorio en local? Que dice
9. Git pull origin master (git pull <remote> <branch>
10. Comprueba tu directorio local

# CONFLICTOS:

1. Situación de partida:
   1. Repositorio remoto en “versión\_1”
   2. Repositorio local en “versión\_1”
2. En local, en tu PC, edita el fichero “fichero.txt” añadiendo una nueva línea
3. Situación actual:
   1. Repositorio remoto en “versión\_1”
   2. Repositorio local en “versión\_1” + fichero modificado localmente.
4. ¿Qué ocurre si ejecutamos git push?
5. Prepara todo para hacer commit del fichero
   1. Git add
   2. Git commit
6. Sube los cambios al repositorio remoto: Git push origin master
7. Comprueba en github que se ha subido el fichero con los cambios
8. Situación actual:
   1. Repositorio remoto en “versión\_2”
   2. Repositorio local en “versión\_2”
9. Modifica el fichero en GitHub añadiendo una nueva línea
10. Situación actual:
    1. Repositorio remoto en “versión\_2”
    2. Repositorio local en “versión\_1”
11. En local, vuelve a modificar el mismo fichero añadiendo una nueva línea.
12. Situación actual:
    1. Repositorio remoto en “versión\_2”
    2. Repositorio local en “versión\_1” + fichero modificado localmente
13. Prepara todo para hacer commit, y push
14. ¿qué ha ocurrido?

A picture containing clock

Description automatically generated

1. Git ha detectado que los cambios de una copia se solapan con los hechos en la otra, y no nos deja “destruir” el trabajo. ¿Cómo lo podemos solucionar
2. Mediante pull traer los cambios desde gitHubt, posteriormente, unirlos a la copia en la que estamos trabajando y luego hacer push.
3. Limpiad la pantalla
4. Git pull origin master
5. Indica que hay un conflicto -- y el merge automático ha fallado.
6. Git status – indica que hay un fichero que ha sido modificado.
7. Visualizamos el contenido del fichero
   1. El cambio en el HEAD (es decir, el cambio de local)
   2. Un separador ======
   3. El contenido del fichero que hay en github (con el marcador que identifica el commit)
8. Editamos el fichero con lo que entendemos que debería ser su contenido final.
9. Git status (está igual que el anterior)
10. Git add
11. Git status
12. Git commit -m “merge de cambios”
13. Sincronizamos con el repositorio remote: git push origin master
14. Situación actual:
    1. Repositorio remoto en “versión\_3”
    2. Repositorio local en “versión\_3

--ahora con un fichero nuevo

1. Añade un fichero al repositorio (a través de la web)
2. Modifica un fichero en local, e intenta subirlo al repositorio (add, commit, push)
3. Ha dado un conflicto
4. Git pull para comenzar su resolución
5. Que ha ocurrido?
6. En qué estado está ahora?
7. Git push

--borramos un fichero

1. Borra un fichero del repositorio
2. Git status
3. ¿se podría hacer un commit?
4. Que comandos tengo que hacer para sincronizar con el repositorio remoto?
5. Sincroniza con el repositorio remoto y mira lo que ocurre.

--borra un fichero del repositorio remoto y sincroniza; comprueba lo que ha ocurrido localmente.

1. Crea un nuevo fichero local, y súbelo al repositorio.
2. En el repositorio, borra el fichero recién creado
3. En local, modifica uno de los ficheros.
4. Ejecuta git push
5. Resuleve el conflicto:
6. Git pull.
7. Leer detenidamente para ver los conflictos.
8. Git status para ver el estado
9. Git commit -a -m “cambios”
10. Observar el mensaje que aparece.
11. Subir los cambios al repositorio.

# Colaborar en un proyecto ajeno

1. Busca el repositorio de un compañero
2. Hacer click en fork
3. Vete a tu perfil
4. ¿Qué ha ocurrido?
5. Introduce una modificación, por ejemplo, crear un nuevo fichero
6. Haz un pull request
7. ¿Qué ha ocurrido?
8. Acepta el pull request
9. Comprueba tu repositorio local

El merge del pull request, se puede hacer con la misma rama o con distintas ramas.

USO DE RAMAS.

https://styde.net/ramas-y-resolucion-de-conflictos-en-git/

1. Colocate en un repositorio
2. Ejecuta git branch
3. Cuantas ramas te salen?
4. Crea una nueva rama llamada: “nueva” git branch nueva
5. Ejecuta git branch
6. Git checkout nueva para moverte a la rama creada
7. Ejecuta git branch para comprobar la rama en la que estas
8. Modifica uno de los ficheros. Y confrmialos (commit)
9. Muévete a la rama master: git checkout master
10. Comprueba el fichero. ¿estan los cambios que has metido?
11. Como estamos en la rama master (la que va a absorver los cambios) ejecuta:
    1. Git merge <rama\_creada>
12. Ejecuta git branch para asegurar la rama en la que estas
13. Veriffica el fichero. ¿tiene los cambios?
14. ¿Dónde están estos cambios?
15. Envialos al repositorio remoto:
16. Git push origin master
17. En GitHub, ¿Cuántas ramas te aparecen?
18. Git push origin <nueva\_rama> para que subir también la rama que se ha creado

Otro ejercicio:

1. Crear nueva rama (git checkout -b feature)
2. Realizar cambios
3. Guardar cambios (git commit -m “cambios realizados”)
4. Hacer merge de los cambios (git checkout master)+ Fusionamos los cambios (git merge featur)
5. Elimianr rama (git branch -d rama-a-eliminar)

https://styde.net/ramas-y-resolucion-de-conflictos-en-git/

**REPASANDO**

**¿Qué es Git ?**

**¿Qué es Github?**

**COMANDOS**

**1.git ini**

**2. git add**

**3. git commit**

**Push**

**Pull**

**CLONE**

**git remote add origin [URL DEL REPOSITORIO EN GITHUB]**

**git push origin master**

**¿Qué es un Branch ?**

**¿Qué es un Merge?**

**Utilizando el comando Branch**

**git push origin [Nombre de la Rama]**

A close up of a logo

Description automatically generated

**Ejemplo de Merge**

1. Situarse en la rama de producción en este caso master: **git checkout master.**

3. Realizar un merge con la rama que se desea incluir : **git merge portada**

5. Publicarlo : **git push origin master**

**Subir un proyecto.**

1. Situarse en la carpeta donde esta nuestro contenido

2. Realizar un **git init**

3. Realizar un add ( **git add .** )

5. git commit

6. git remote add origin para conectar nuestro repositorio local con el github

7. Realizamos un push