



Git Y GitHub

Por Ariel Parra.

Git

GitHub

Git es una herramienta de código abierto sobre el control de versión distribuido mientras que GitHub es una plataforma y servicio basado en la nube de Microsoft para el desarrollo de software y el control de versiones utilizando Git, básicamente un servidor donde almacenas tu código y lo controlas con Git.

Para conectar git con github:

- Primero crearemos una cuenta de github <https://github.com/signup>
- Despues creamos un repositorio (proyecto) en <https://github.com/new>
- Luego descargamos git con algun pacakge manager o desde la pagina oficial <https://git-scm.com/>
- También comprobamos que esta instalado el clinte de [OpenSSH](#) en Windows.
- En una terminal creamos nuestro usuario local de git (puede ser distinto al de Github):

```
git config --global user.name "Usuario"  
git config --global user.email correo@ejemplo.com  
git config --global --list
```

- En la misma terminal creamos una llave pública de ssh:

```
ssh-keygen -t ed25519 -C "al-ID-@edu.uaa.mx"
```

- Copiamos los datos del archivo generado en C:\Users\"Usuario\"\.ssh\id_ed25519.pub (en windows es el que tiene icono de MS Publisher) ó /home/"usuario"/.ssh/id_ed25519.pub (puedes abrirlo con bloc de notas) y lo conectamos con nuestra cuenta de github atravez de <https://github.com/settings/ssh/new>, dandole un nombre (el de tu computadora) y pegando la los datos copiados (llave).

Conectar al repositorio de Github

- Crear un Repo de manera local

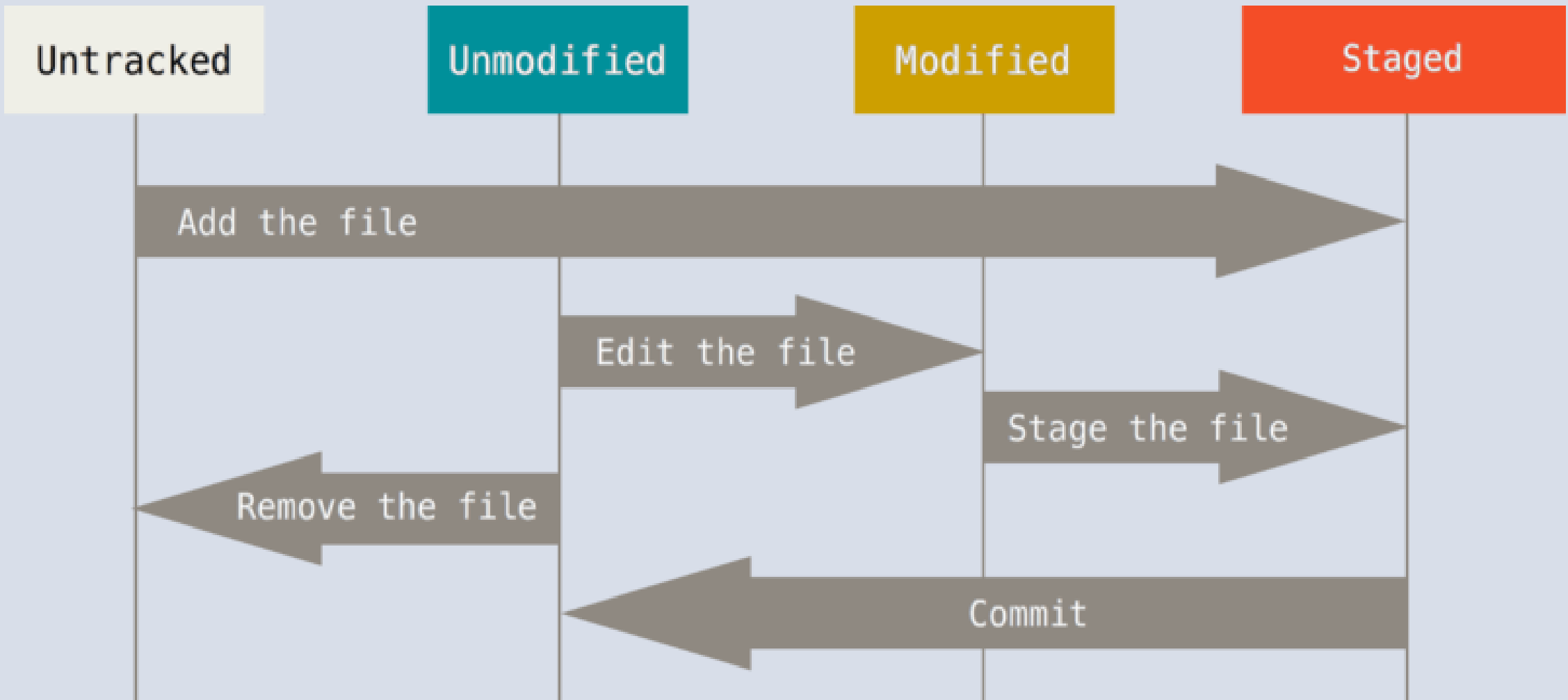
```
mkdir proyecto/           #Crear la carpeta
/cd proyecto/             #entrar a la carpeta
git init                   #Crear los archivos de .git
echo '#README' >> README.md #Crear el Readme con contenido
git add .                  #Agrega todos los archivos de la carpeta
git status                 #Lista los archivos modificados
git commit -m "Primer commit" #Commiteas localmente
git remote add origin git@github.com:usuario/Proyecto.git
git remote -v              #Muestra los repos conectados
git push -u origin master  #Pusheas a la rama principal
```

- Descargarlo

(importante que sea el git@ en lugar de https para poder hacer cambios)

```
git clone git@github.com:"Usuario de GitHub"/"Proyecto".git
```

Estados de proyecto



Staged (preparado):

Propósito: Los archivos en el área "staged" (preparado) son aquellos que están listos para ser confirmados en el próximo commit. En otras palabras, son los cambios que has seleccionado específicamente para incluir en tu próximo commit.

Momento de uso: Se utiliza antes de realizar un commit. Debes utilizar el comando `git add` para agregar cambios al área "staged". Los archivos en el área "staged" serán parte del próximo commit que realices.

Uso típico: Cuando deseas seleccionar y confirmar un conjunto específico de cambios en lugar de confirmar todos los cambios en tu directorio de trabajo. Esto te permite realizar commits más enfocados y organizados.

Stash (almacenar en Git):

Propósito: El comando git stash se utiliza para guardar temporalmente los cambios locales en tu directorio de trabajo sin la necesidad de realizar un commit. Esto es útil cuando estás en medio de una tarea en una rama y necesitas cambiar de rama o realizar otras operaciones sin comprometer tus cambios locales.

Momento de uso: Se utiliza antes de realizar un commit. Te permite guardar temporalmente cambios que aún no están preparados para ser confirmados y, por lo tanto, no están en el área de "staged" (preparados).

Uso típico: Cuando trabajas en una rama y necesitas cambiar a otra rama, pero no deseas confirmar tus cambios locales o deseas moverte entre ramas sin llevar contigo los cambios locales.

```
git stash          #archivos en staged pasan a stash
git stash pop      #Recupera y aplica los cambios stash más reciente
git stash clear    #archivos del área de stash serán eliminados
git stash list     #lista de todas las entradas del stash
```

Archivos comunes

- `.gitignore`

```
# ignora todos los archivos .a a travez del wildcard *
*.a

# excepcion a lib.a
!lib.a

# solo ignora a TODO , pero no a subdir/TODO/
/TODO

# ignora los archivos en el directorio build/
build/

# ignora doc/notes.txt, pero no doc/server/arch.txt
doc/*.txt

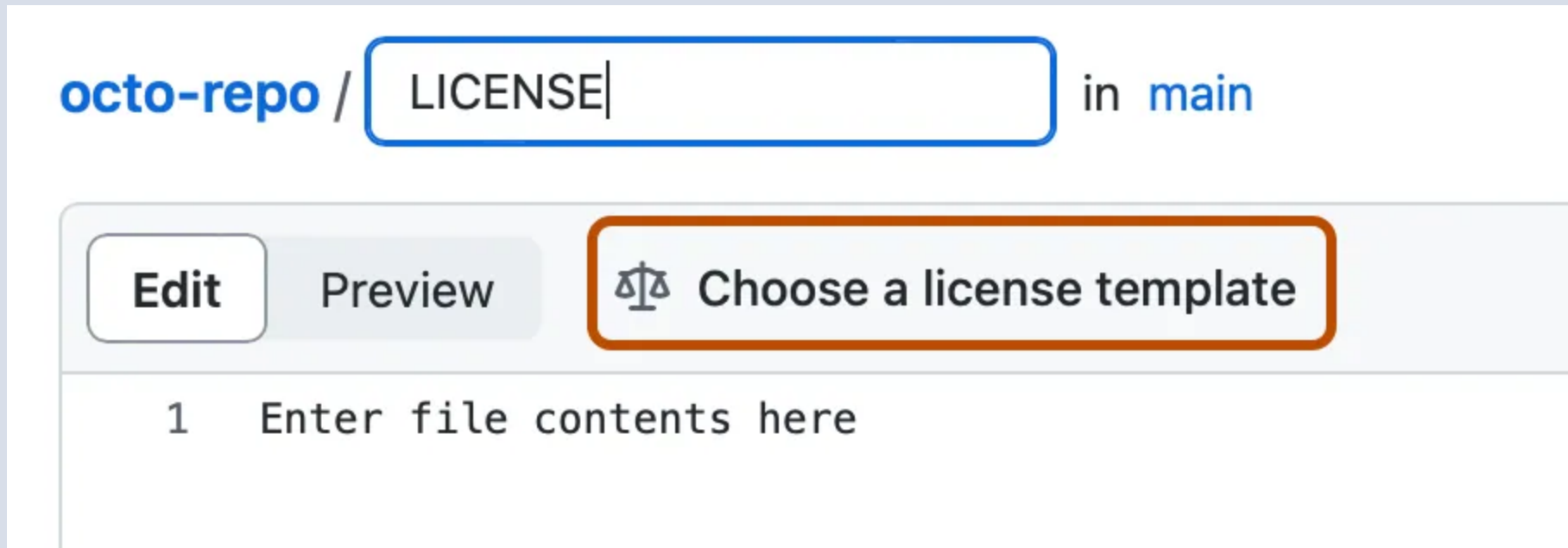
# ignora los .pdf en doc/ y sus subdirectorios
doc/**/*.pdf
```

- README.md

Es un documento en formato Markdown que se utiliza para proporcionar información sobre un proyecto de desarrollo de software. Contiene una descripción del proyecto, instrucciones de instalación, detalles de uso, configuración, pautas de contribución, licencia, etc.

- LICENCE o LICENSE.md

Es un documento largo para mostrar la licencia junto con sus libertades y restricciones, github te da una manera facil de agregarlas:



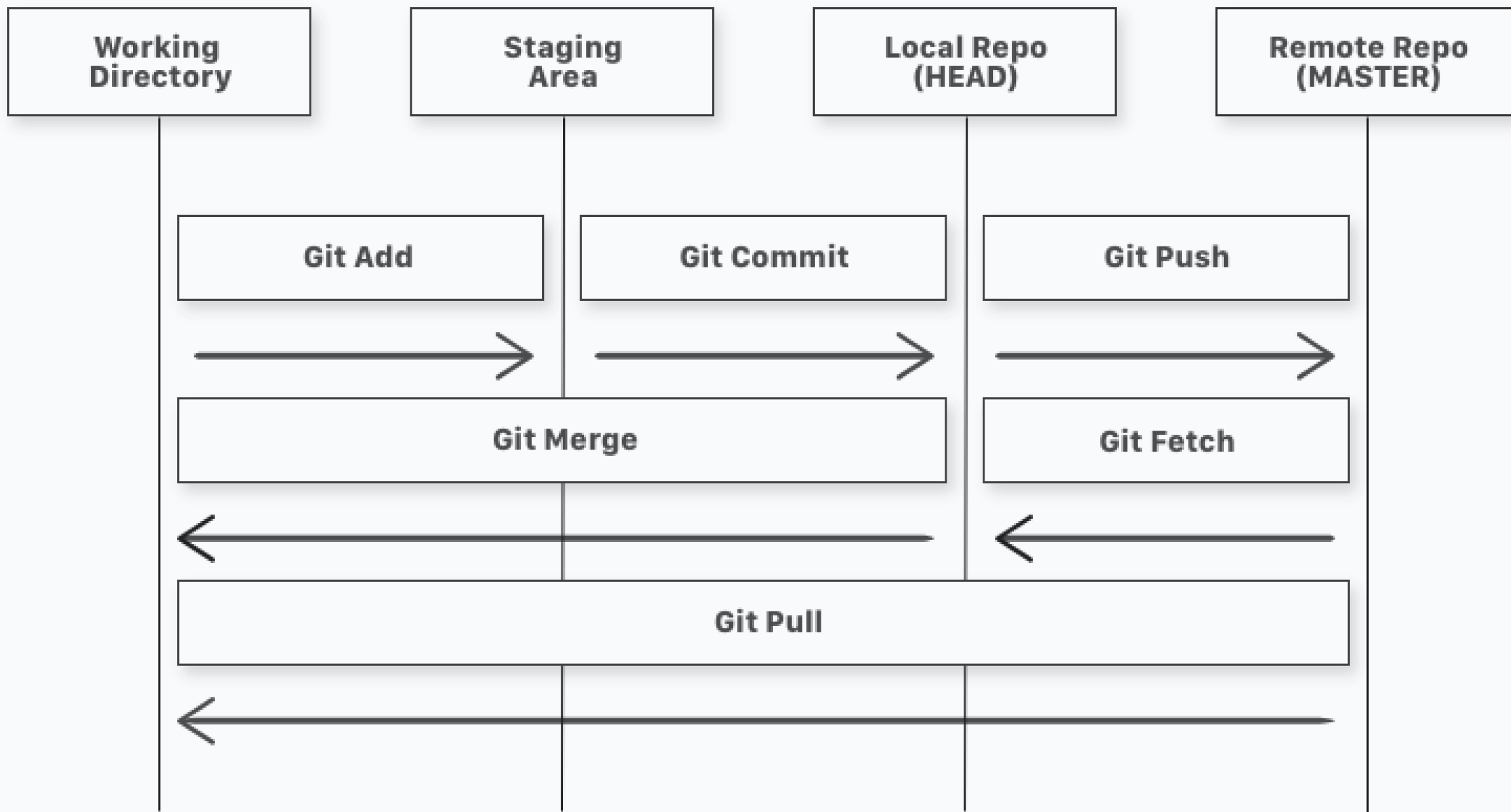
Ramas

Cuando las ideas divergen, pueden hacerse dos ramas de desarrollo

```
git branch ramaNueva      #crear rama  
git checkout rama Nueva   #Cambiar al nueva rama
```

```
git checkout -b rama Nueva #0 ambos comandos a la vez
```

Git Workflow



Commit

Commit History

```
git log
```

Retractarse

```
git commit -m 'Mensaje'  
git add forgotten_file  
git commit --amend      # agrega al anterior commit  
git reset HEAD~1        # 0 Remueve el ultimo commit
```

Commit junto con git add .

```
git commit -a -m "Mensaje del commit"
```


Push

git push es un comando en Git que se utiliza para cargar o enviar tus cambios locales a un repositorio remoto. Esto permite compartir tus cambios con otros colaboradores y mantener sincronizado el repositorio remoto con tus últimas actualizaciones. Aquí está cómo funciona git push y algunos de sus comandos comunes:

```
git push          # el push a la rama actual
git push nombre-remoto nombre-rama-local:nombre-rama-remota
git push --all    # pushear todas las ramas
git push -u nombre-remoto nombre-rama-local
```

Fetch

`git fetch` es un comando en Git que se utiliza para recuperar información sobre cambios en un repositorio remoto, pero no realiza la fusión automática de estos cambios en tu rama local. En otras palabras, `git fetch` descarga las actualizaciones del repositorio remoto y actualiza la información de seguimiento, como las ramas remotas, sin aplicar los cambios directamente a tu rama de trabajo local.

El repositorio remoto se suele llamar `origin`

```
git fetch origin
```

Vscode tiene una configuración para [auto fetch](#)

Merge

git merge es un comando en Git que se utiliza para combinar cambios desde una rama en otra rama. Es comúnmente utilizado para incorporar cambios de una rama de características (feature branch) en una rama principal, como la rama "master" o "main". Aquí te explico cómo se usa git merge:

```
git checkout nombre-de-la-rama-destino      # usualmente main
git merge nombre-de-la-rama-a-fusionar      # rama divergente a converger
git commit -m "fusión"
git push
```

Pull

El comando `git pull` es utilizado para recuperar los cambios del repositorio remoto y fusionarlos automáticamente con la rama local actual.

En los términos más simples, `git fetch` seguido de un `git merge` es equivalente a un `git pull`. Pero, ¿por qué existen estas dos opciones?

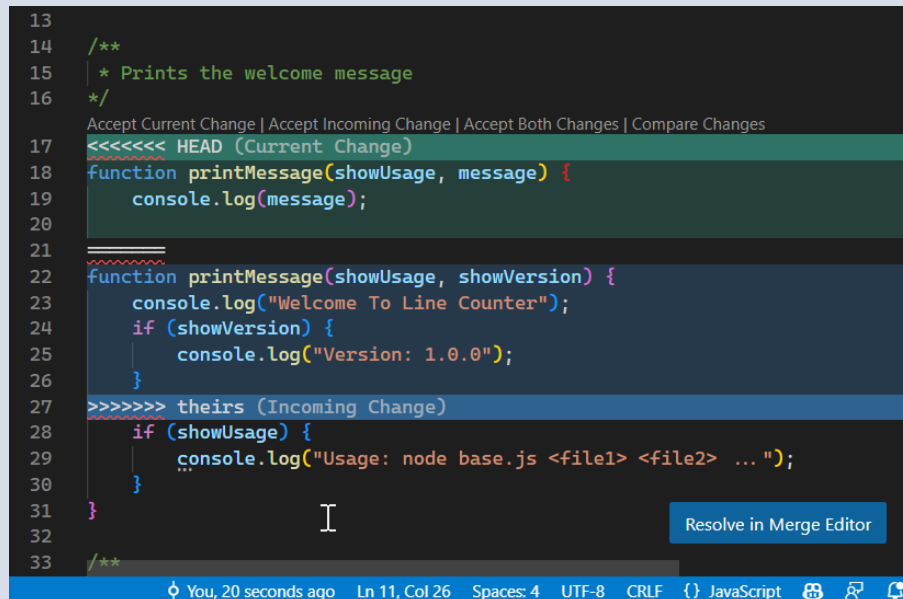
```
git checkout nombre-de-la-rama-destino  
git pull
```

Problemas de Merge o Pull

Para ver de quien fue el ultimo cambio

```
git blame
```

Vscode puedes ver graficamente los problemas



```
13
14 /**
15  * Prints the welcome message
16  */
17 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
18 <<<<<< HEAD (Current Change)
19 function printMessage(showUsage, message) {
20     console.log(message);
21 }
22
23 function printMessage(showUsage, showVersion) {
24     console.log("Welcome To Line Counter");
25     if (showVersion) {
26         console.log("Version: 1.0.0");
27     }
28 }
29 >>>>>> theirs (Incoming Change)
30 if (showUsage) {
31     console.log("Usage: node base.js <file1> <file2> ...");
32 }
33 /**
```

Rebase

git rebase es un comando en Git que se utiliza para reorganizar o limpiar la historia de commits de una rama. En lugar de fusionar (merge) cambios como lo hace git merge, git rebase mueve, en esencia, tus cambios locales a la punta de otra rama, lo que resulta en un historial de commits más lineal y limpio.

Comandos POSIX de git

Para eliminar un archivo de Git, debes eliminarlo de tus track files en el area de staging (preparacion) y luego hacer un commit.

```
git rm archivo.txt # o git rm --cached archivo.txt #para el area de staging
```

¿Qué has cambiado pero aún no staged?" y "¿Qué has staged y estás a punto de commit?"

```
git diff
```

si quieres ver lo que has staged, que ira al proximo commit,

```
git diff --staged
```

Si quieres renombrar o mover

```
git mv file_from file_to
```


Git kraken

GitKraken (gratis): Soporte para Git básico, Integración con repositorios públicos en GitHub y GitLab, Interfaz de usuario gráfica para operaciones de Git, Historial de commit y visualización de ramas, Herramientas de resolución de conflictos básicas, Seguimiento de archivos y cambios, Integración con cuentas de correo electrónico.

GitKraken Pro (de pago): Soporte avanzado para Git, incluyendo integración con GitFlow, Integración con repositorios privados en GitHub y GitLab, Herramientas de seguimiento de errores y solicitudes de extracción integradas, Características de seguridad y gestión de acceso, Capacidades avanzadas de resolución de conflictos, Capacidades de revisión de código, Soporte y actualizaciones prioritarios, Integración con servicios de terceros como Jira, Trello, Slack y más.

Git Lens

GitLens (gratis): Visualización de información en línea, Exploración de historial de Git, Búsqueda de cambios, Blame Annotations, Comparación de revisiones.

GitLens Pro (de pago): Exploración avanzada del historial, Búsqueda avanzada, Comparación de ramas y commits, Integración con GitHub, Soporte y actualizaciones prioritarios.

Por suerte ambas versiones de pago son gratis mientras estudies, para esto tienes que aplicar para el [Github Student Developer Kit](#).

Referencias

<https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet>

<https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/>

<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>

<https://git-scm.com/docs>

<https://www.geeksforgeeks.org/git-tutorial/>

<https://www.geeksforgeeks.org/ultimate-guide-git-github/>

<https://learn.microsoft.com/en-us/visualstudio/version-control/?view=vs-2022>

<https://code.visualstudio.com/docs/sourcecontrol/overview>