**Curso profesional del GIT y GITHub**

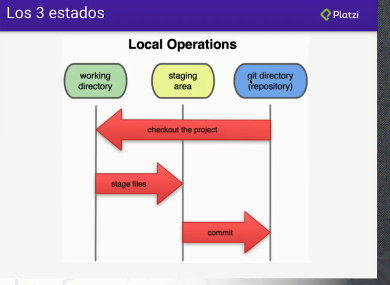
Descarga: <https://git-scm.com/>

Tipos

Sistema local control Z

Centralizado – sevidor del repositorio

Distribuidos GIT



Git herramienta de verisionamiento

GitHub Herramienta de la nube para colaborar, red social para distribución.

Organiza tareas



Saber la versión de GIT



Asociar email



Asociar nombre de usuario



Activar Colores



Crear archivo en el repositorio



Borra un archivo



Borrar directorio



Editar Archivo

Vim

Guardar del archivo y salir



**Primer Repositorio**

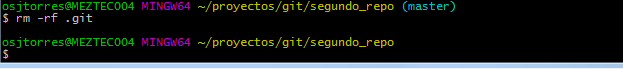
Iniciar repositorio en GIT



Iniciar Git en directorio ya existente



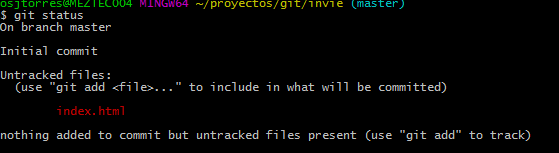
Borrar proyecto iniciado en Git



**git add | rm | status (Agregando, quitando y viendo el estatus de archivos)**

Crear archivos y pasar del working directory a staging

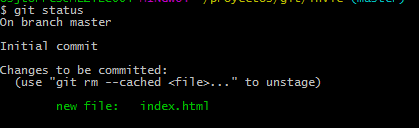
Cuando aparece en rojo está en nuestro working directory



Agregar a staging



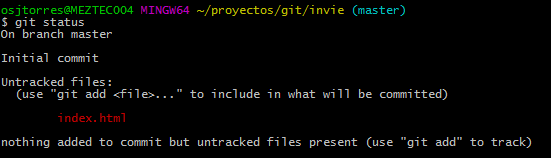
Cuando está en staging aparece en verde



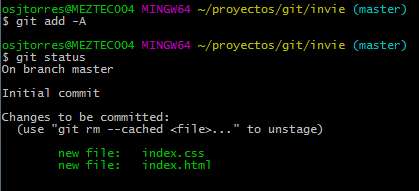
Devolver del staging (unstage)



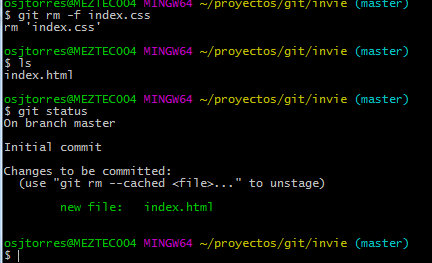
Resultado



Agregar varios archivos a staging



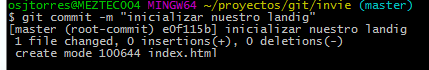
Quitar archivos de Staging y del Working Directory



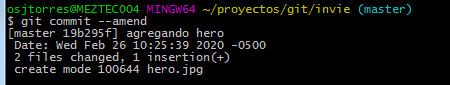
Borrar archivo del working Directory

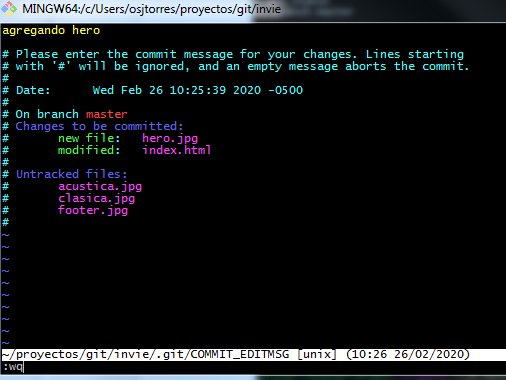


**git commit (Confirmando cambios)**

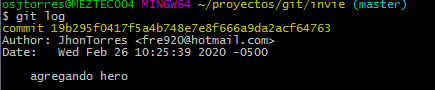
****

Unir archivos a un commit ya realizado





Ver el log de commit realizado en el repo



**git tag (Etiquetando confirmaciones)**

Ligeras “Version” git tag 0.5

Anotadas “versión con mensage” git tag –a 0.5 –m ‘version estable del proyecto’



Consultar lista de las etiquetas

****

Borrar tag



Renombrar Tag



**git log (Revisando la historia de nuestro proyecto)**

Log Resumido

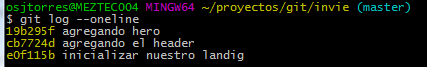
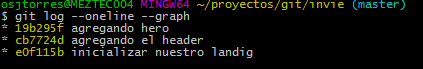
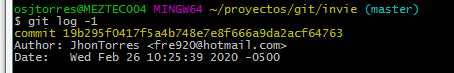


Grafico log del proyecto

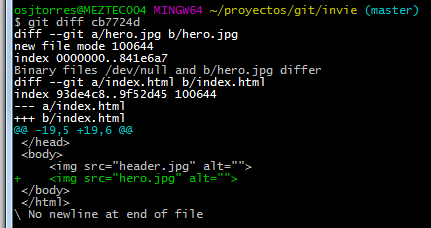


Ver solo el último commit



**git diff (Revisando los cambios entre versiones)**

Comparar estado inicial con la versión – primero busco el sha con git log --oneline



Comprar dos commit



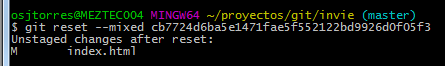
**git reset – soft**

Quitar un commit y mantener los cambios en staging



**git reset – mixed**

Quita el commit y también descarta los cambios del staging



**git reset – hard**

Devolver hasta el commit que le indiquemos

Quitar los archivos de Working Directory y staging

Git-reset –hard #sha

Devolverse

Git-reset –hard #sha borrado

Configurar VS con GIT



**git branch (Múltiples variantes del repositorio)**

Hasta el momento hemos estado trabajando en la rama principal de git. Ahora nos vamos a centrar en las posibles ramas de nuestro proyecto.  
Comenzamos con ““Master”” que es la rama principal, sin embargo, podemos crear ramas para que el equipo pueda trabajar en el proyecto. Lo ideal es que siempre tengas ramas adicionales para que si hacemos cambios podamos hacer modificaciones. Para crear una rama o un branch usamos el comando git branch.  
Si necesitamos borrar una rama definitivamente, debemos usar -D. Para renombrar una rama vamos a usar el comando -m."

Crear rama:



Listar ramas



git branch " "  
para listar las ramas: git branch -l  
para borrar la rama: git branch -d  
si tiene cambios la rama: git branch -D  
para renombrar rama: git branch -m "Nombre\_viejo " “nuev”

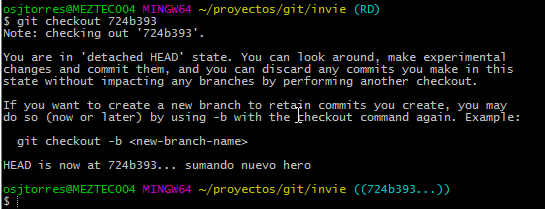
Cambiar de rama



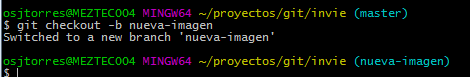
Crear rama



Moverse y crear una rama virtual para verificar en cual se incluyó algún bug



Crea la rama y la selecciona

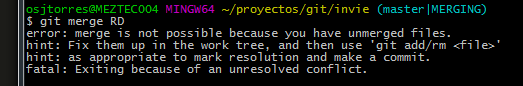


**Trabajando entre ramas**

**git merge (Mezclando ramas y resolviendo conflictos)**

En este material nos vamos a concentrar en mezclar ramas y resolver conflictos. Lo primero que tengo que hacer para mezclar ramas es pararme sobre la rama que quiero mezclar.  
Para mezclar los cambios deberíamos usar el comando **git merge.** Las instrucciones sería pararnos desde la rama que queremos mezclar y hacer un **git merge** con la rama que pretendemos usar.

**Haciendo merge de la rama master con RD**

****

**git stash (Guardando cambios temporalmente)**

¿Qué tal si aún no estás listo para confirmar ningún cambio? **Stash** es un estado que tenemos como intermedio. Para esto debemos ir a alguna de nuestras ramas y usando el comando **git stash** que nos permite hacer cambios, pero no confirmarlos.

$ git stash guarda cambios temporales

$ git stash list Consulta la lista

$ git stash drop “codigo” elimina un cambio

$ git stash appply devulve el último cambio

$ git stash appply “codigo” devulve el ultimo cambio

**Cherry pick eligiendo commits selectivamente**

Si estás trabajando en una rama, pero de repente notas que hiciste un cambio en la rama que no debías, para esto podemos usar **cherry pick**. Este comando nos puede salvar la vida, ya que nos permite sacar cambios específicos de una rama y mezclarlos en otra.

$ git cherry-pick 3d8e410 Pasa un commit a otra rama

**git clone/fork (Clonando repositorios remotos)**

Vamos a proceder a crear nuestro primer proyecto en **GitHub.**

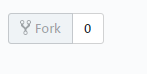
* Nuevo repositorio
* Asigna un nombre a tu repositorio
* Asigna una descripción a tu repositorio
* Puedes adicionar un ““read me””
* Crea el repositorio

En la parte derecha encuentras una opción que se llama clone or download si no es nuestra intención compartirlo.  
Un fork va a clonar el proyecto y nos va a crear un nuevo proyecto en el **GitHub.**

Ubicado en la ruta del proyecto

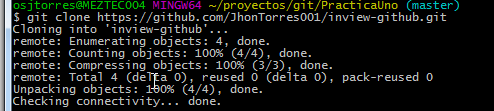
$ git clone https://github.com/JhonTorres001/inview-github.git

Para copiar un repo existente

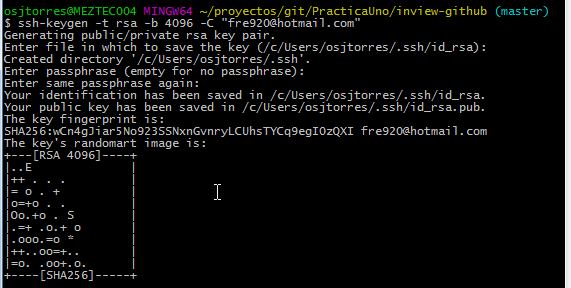


**Añadiendo una llave ssh a GitHub**

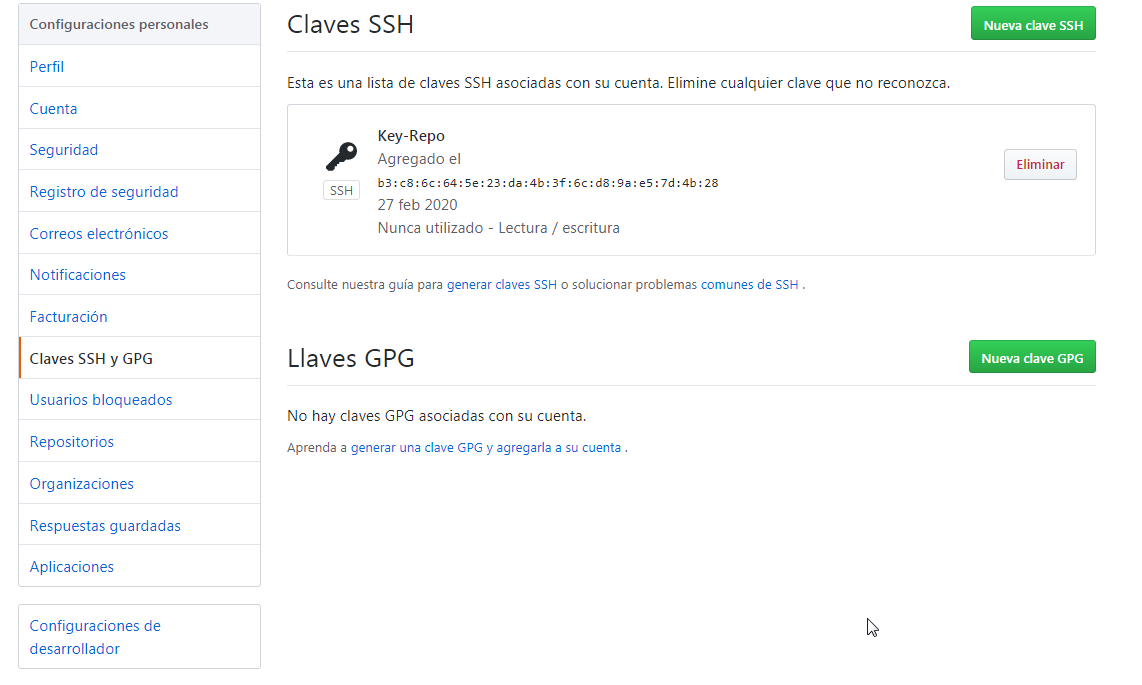
**Clonando repo**

****

**Añadiendo Key**

****

**c:/Users/osjtorres/.ssh/**



**git remote (Añadiendo un repositorio remoto a uno local)**

Tenemos que crear una conexión entre tu repositorio local y tu repositorio remoto para que puedas compartirlo con la comunidad. Para esto vamos a usar un nuevo comando que en este caso es **git remote.** Por convención lo nombramos “origin”. Para borrarlo podemos usar **git remote remove** y el nombre del repositorio.

$ git remote add origin <https://github.com/JhonTorres001/inview-github.git>

$ git remote –v Consultar

$ git remote remove origin Eliminar

**git pull/fetch (Trayendo cambios desde el repositorio remoto)**

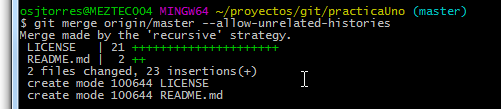
Vamos a compartir nuestro código con el repositorio remoto. Antes de poder enviar cambios tengo que tener los mismos cambios en el repositorio remoto que en el repositorio local.  
Para traer los cambios del repositorios remoto vamos a usar el comando **git fetch.** Este comando recibe dos parámetros 1. desde donde y hacia donde van.  
Luego de traer los cambios, debemos mezclarlos con la rama que estamos usando en local. Para esto usamos git merge.

Traer los datos del repo a la rama master



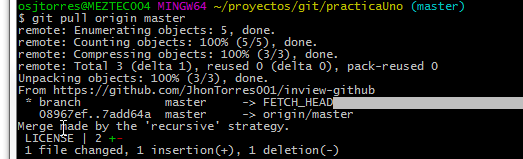
$ git fetch origin master

Hacer merge con el repo local y la rama remota



Actualizar el repo

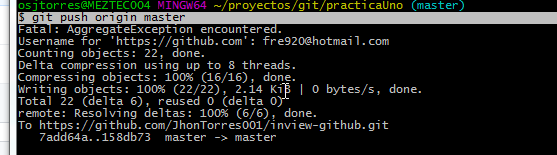
$ git pull origin master



**git push (Enviando cambios al repositorio remoto)**

Para enviar nuestro código al repositorio remoto vamos a usar el comando git push. Así logramos que otras personas tengan acceso a nuestro proyecto.  
Cuando esto termine tendremos acceso a nuestro trabajo desde la plataforma de **GitHub.**  
Lo primero que debemos hacer es enviar nuestros tags en el proyecto.  
también recordemos que podemos eviar ramas, lo que también posdemos hacer a través del uso del comando git push [nombre de la rama].

$ git push origin master



Enviar tags

$ git push origin --tags



Enviar las ramas

$ git push origin Develoment

