Gestión de Proyectos WEB

Sistemas de control de versiones

Presentación 1

Trabajo local

Prof. Luis Eduardo Fagúndez

Qué es GIT?

Herramienta de control de versiones distribuida

Herramienta de código abierto que los desarrolladores instalan localmente para gestionar el código fuente

Plataforma basada en la nube

Servicio en línea al que los desarrolladores que utilizan Git pueden conectarse y cargar o descargar recursos

Configuraciones iniciales

- **$ git config --global user.name "Nombre de usuario"**
  - Configura el nombre de usuario
- **$ git config --global user.email correo_electronico@gmail.com**
  - Configura el correo por defecto
- **$ git config --global clore.editor editor_por_defecto**
  - Configura el IDE por defecto

Configuraciones iniciales

- **$ git config --global init.defaultBranch main**
  - Agrega la rama por defecto al crear un repositorio
- **$ git config --global core.excludesfile ~/.gitignore**
  - Indica la ubicación del archivo .gitignore
- **$ touch ~/.gitignore**
  - Crea el archivo .gitignore
- **$ git config --list**
  - Verifica todas las configuraciones de mi GIT

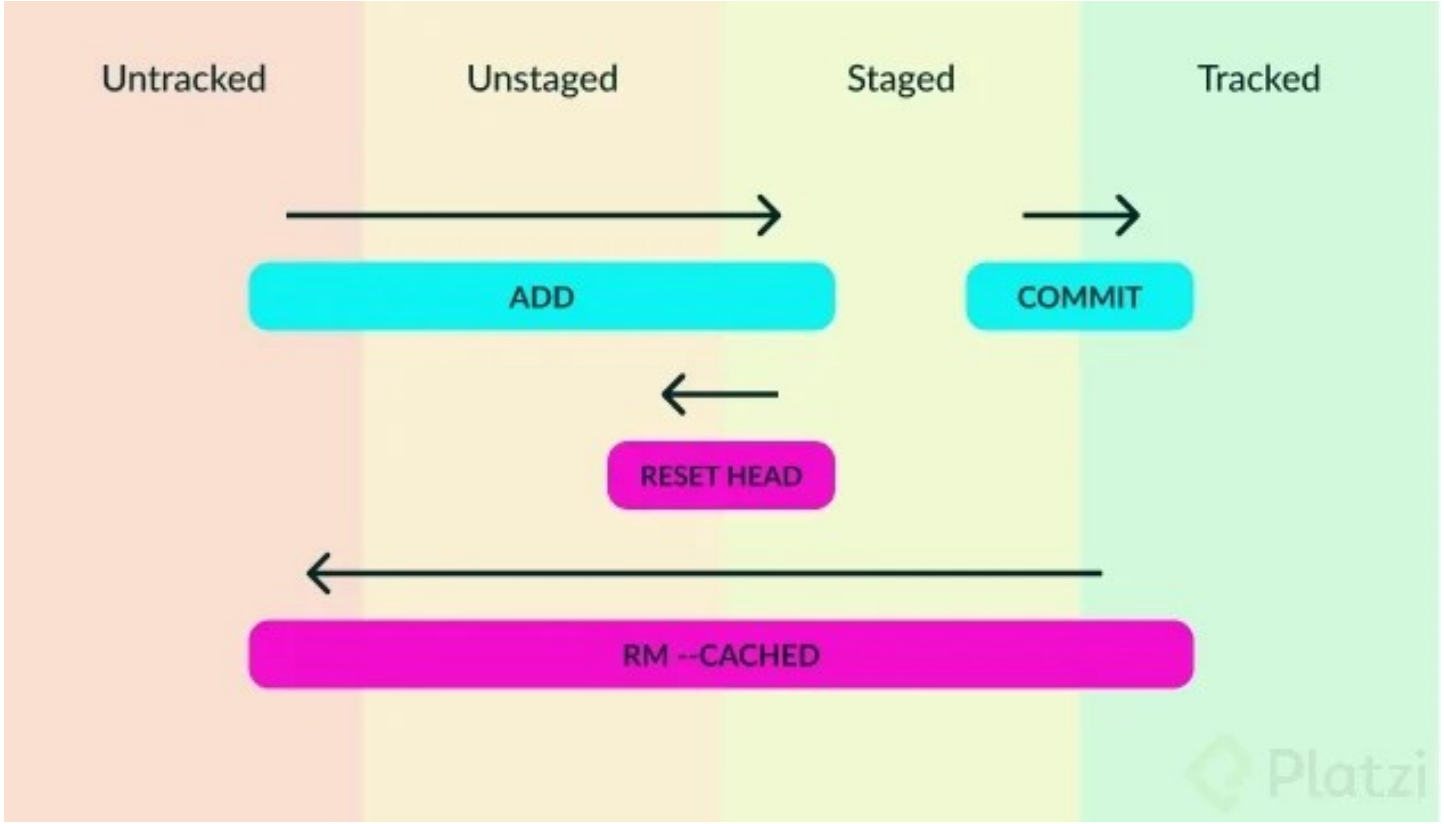
Comandos básicos

- **$ git init**
  - Inicializa el repositorio para comenzar a trabajar
- **$ git status**
  - Ver el estado de un repositorio
- **$ git add . Ó también: $ git add -A**
  - Actualiza todas las modificaciones a la lista de cambios.
- **$ git commit -am <mensaje>**
  - Si ya se hizo un add previamente se puede hacer add y commit al mismo tiempo.

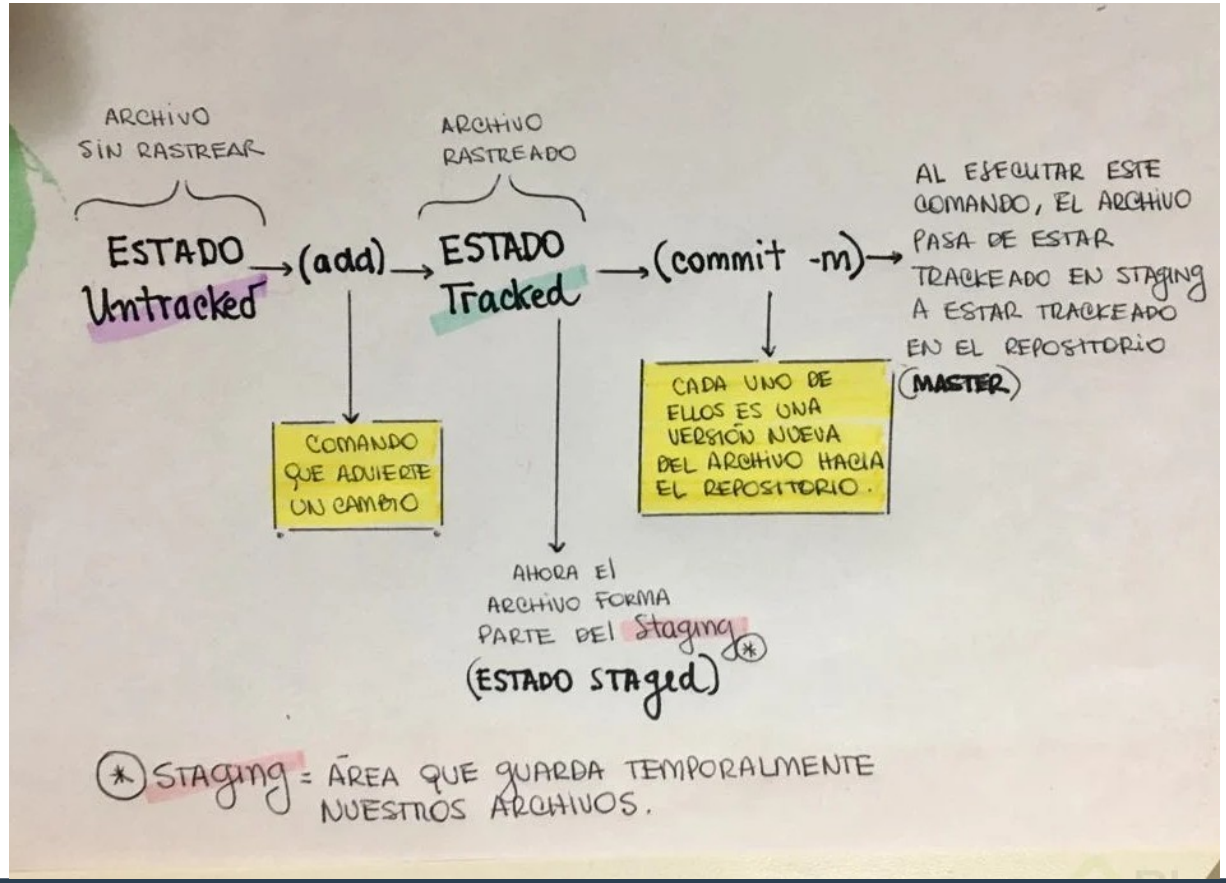
Comandos básicos

- **$ git commit -m <Mensaje>**
  - Confirma las modificaciones al repositorio
- **$ git log**
  - Ver el historial de cambios de mi repositorio
- **$ git log --stat**
  - Ver el historial de cambios desglosado
- **$ git show <archivo>**
  - Ver las modificaciones de un archivo

Estados del commit



Estados del commit



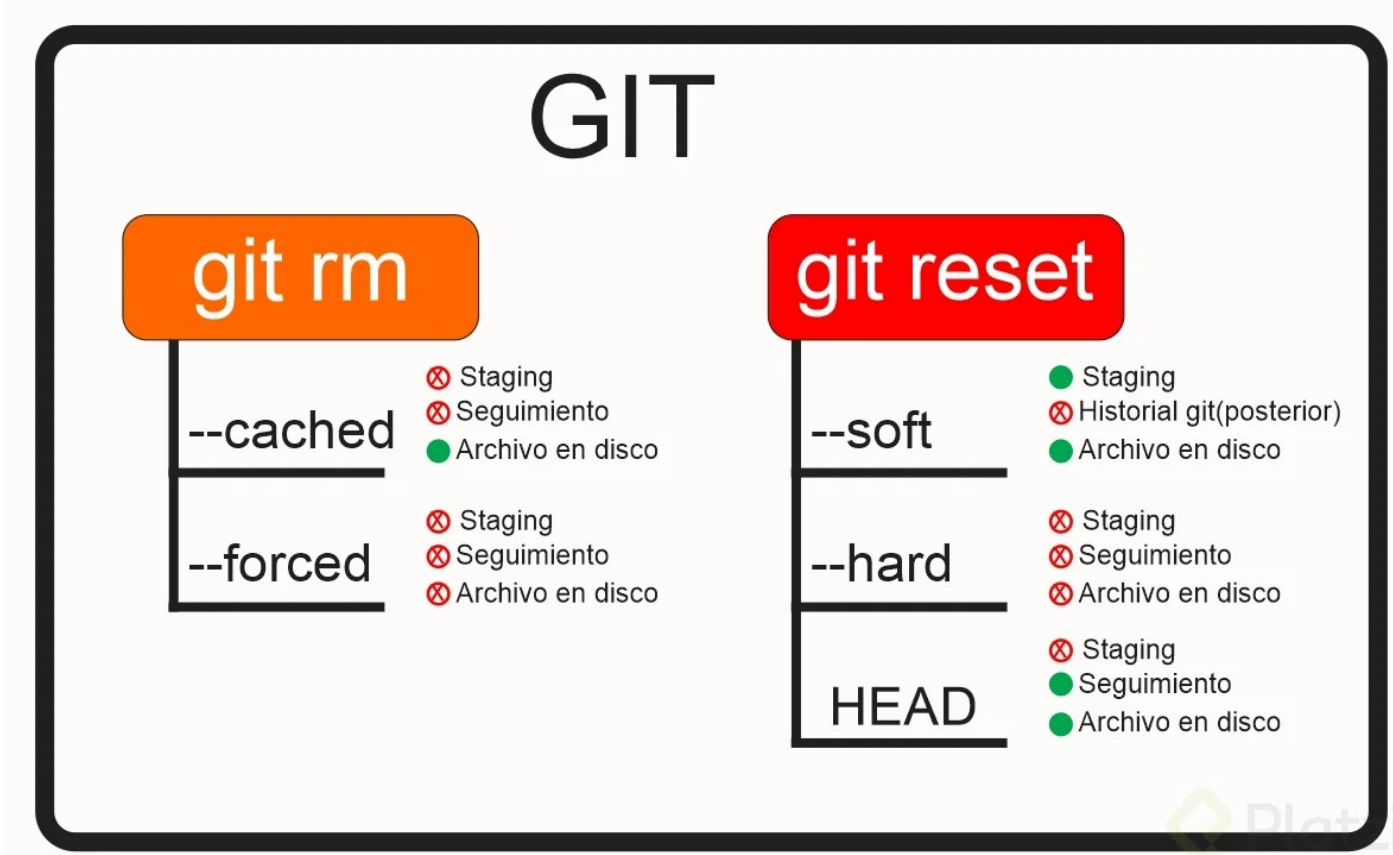
Comandos básicos

- **$ git branch**
  - Verifica las ramas disponibles en mi repositorio
- **$ git checkout -b <nueva-rama> <rama-actual>**
  - Crea una rama nueva y copia el contenido de la actual
- **$ git checkout <rama>**
  - Cambia de rama de trabajo

Comandos básicos

- **$ git checkout <código del git log> <archivo>**
  - Me permite volver atrás un archivo especifico
- **$ git reset <código del git log> --hard**
  - Vuelve todo hacia atrás.
- **$ git reset <código del git log> --soft**
  - Vuelve hacia atrás pero los staging aún están disponibles para actualizar.

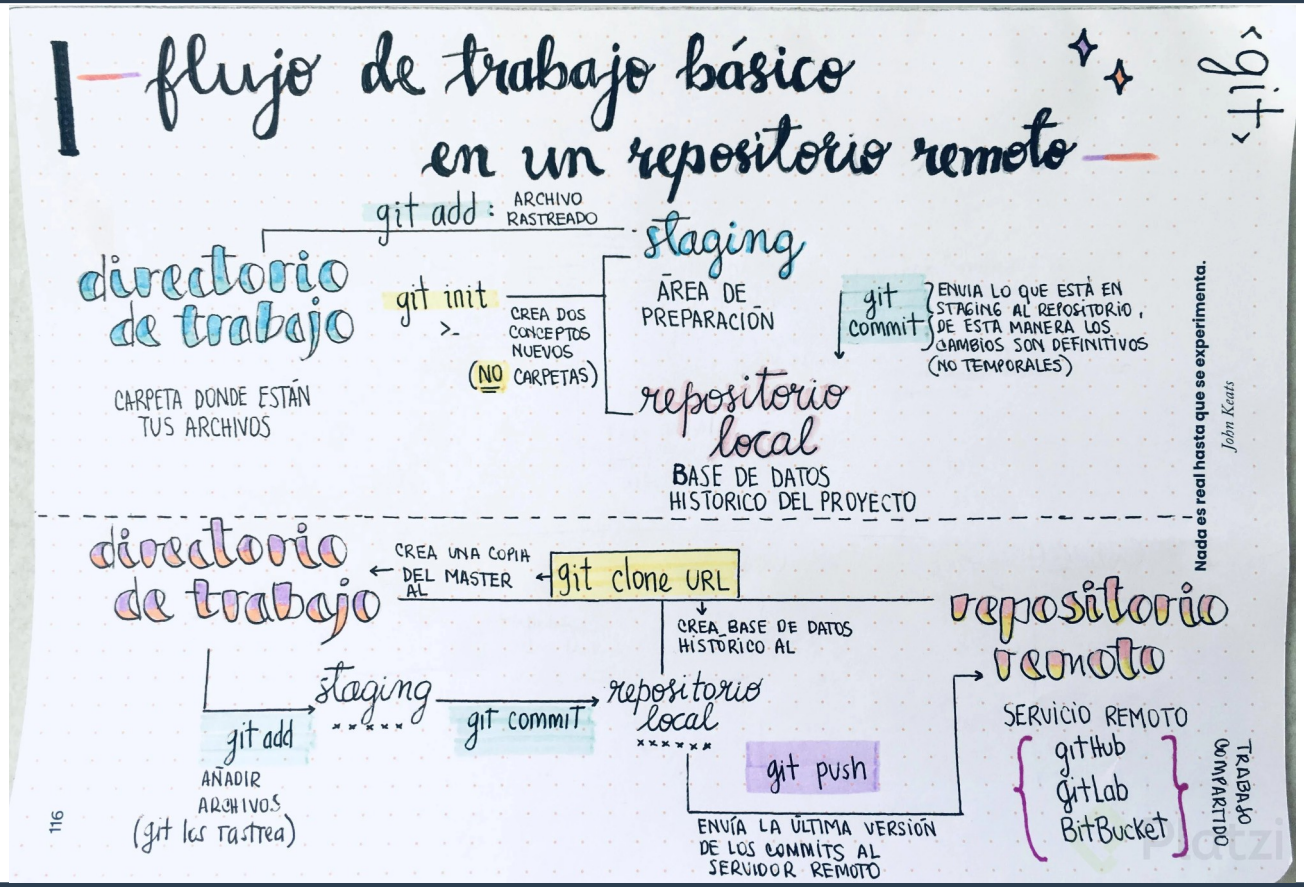
Git reset


GIT
git rm
--cached
⊗ Staging
⊗ Seguimiento
● Archivo en disco
--forced
⊗ Staging
⊗ Seguimiento
⊗ Archivo en disco
git reset
--soft
● Staging
⊗ Historial git(posterior)
● Archivo en disco
--hard
⊗ Staging
⊗ Seguimiento
⊗ Archivo en disco
HEAD
⊗ Staging
● Seguimiento
● Archivo en disco

git rm

- **Este comando nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones. Esto quiere decir que si necesitamos recuperar el archivo solo debemos “viajar en el tiempo” y recuperar el último commit antes de borrar el archivo en cuestión.**
  - $ git rm --cached
    - Elimina los archivos de nuestro repositorio local y del área de staging, pero los mantiene en nuestro disco duro
  - $ git rm --force
    - Elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos, de modo que podremos recuperarlos si es necesario

Flujo de trabajo básico



Fusión de ramas

- **Para fusionar las ramas debemos de estar parados en la rama que queremos recibir los cambios, en el caso de contar con dos ramas “main” y “testing”, si los cambios fueron realizados en testing deberíamos de usar $ *git checkout main* para dirigirnos a nuestra rama main y poder hacer la fusión con el comando marge.**
- **$ git merge testing**
  - Realiza una fusión de la rama testing a la rama que estamos situados actualmente.

Fusión de ramas: Conflictos

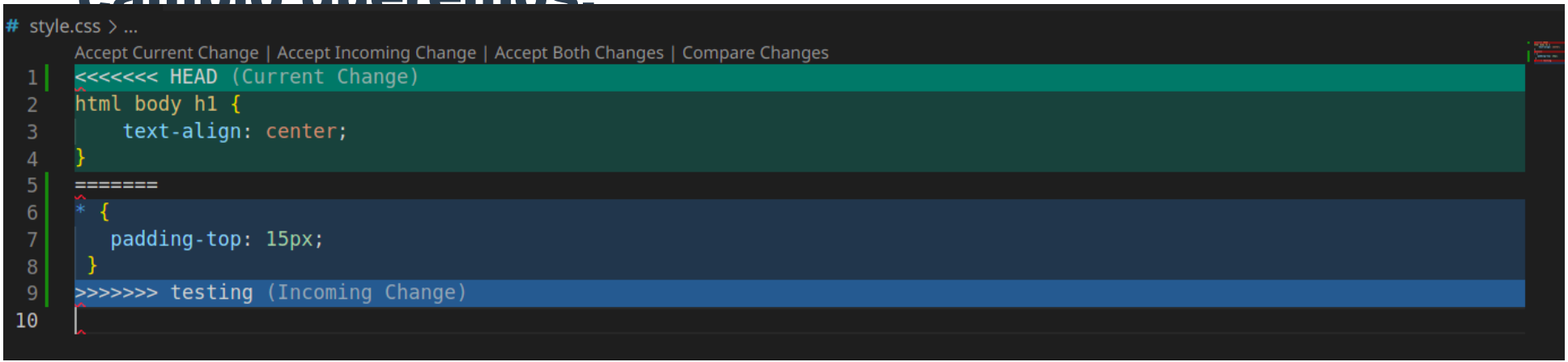
Fusión de ramas: Conflictos

- **En el caso de que dos desarrolladores estuvieran trabajando sobre el mismo archivo, al realizar un merge, visualizaremos el siguiente mensaje de error:**

```
→  sigd git:(main) git merge testing
Auto-fusionando style.css
CONFLICTO (agregar/agregar): Conflicto de fusión en style.css
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
```

Fusión de ramas: Conflictos

- **Desde Visual Studio Code vamos a poder ver automáticamente una opción para aceptar cual cambio queremos:**



Fusión de ramas: Conflictos

- **Al fusionar los dos usando la opción accept both changes nos quedará de la siguiente forma:**