

Pruebas con Git

Nicolas Cardona Ramirez

24 de enero de 2020

1. Video 2: Primeros pasos en Git

Para iniciar el Git se utiliza lo siguiente:

1. `git config --global user.name` seguido del usuario entre comillas
2. `git config --global user.email` seguido del email entre comillas
3. `git config --global color.ui true` para habilitar los colores dentro del código

2. Video 3: Iniciar el monitoreo

- Con el comando `git init`, para iniciar el rastreo del proyecto
- Con `git status` se muestran las modificaciones del proyecto
- Para añadir los archivos a la sincronización se utiliza el comando `git add`
- Para agregar todo al proyecto se utiliza `git add -A`
- Para agregar se utiliza el comando `git commit -m` y un mensaje entre comillas para identificar la modificación
- Para ver una lista con los comentarios se utiliza el comando `git log`
- Ahora para viajar entre versiones, se crea un último commit, se utiliza el `git log` para copiar el código del commit y se utiliza el comando `git checkout` con el código del commit. Para ver el código en su totalidad se utiliza `git checkout master` para devolver a la versión deseada
- Para “Matar los commit” se utiliza el comando `git reset` las el código del commit, el cuál se subdivide en `git reset --soft`, el cuál no se mete con el código, esta el `git reset --mixed` y está el comando `git reset --hard` que borra absolutamente todo
- El comando `git log > commits.txt` crea un documento de texto con cada uno de los commits

- El comando `git help` da un pequeño manual para el uso de git, si se quiere ser mas específico se utiliza `git help + el nombre del comando`
- Con el comando `git diff` se muestran todas las modificaciones creadas en los archivos sin ser añadidos aún
- Para ignorar un archivo o carpeta de archivos se crea un fichero llamado `.gitignore` y dentro de él se escribe el nombre de los ficheros o carpeta de ficheros que se van a ignorar a la hora de subir los archivos

3. Video 4: Ramas y Funciones

Head = Actual commit

- Para crear una rama se utiliza el comando `git branch + nombre de la rama`
- Para moverse entre ramas se utiliza el comando `git checkout + el nombre de la rama`
- Para absorber una rama se utiliza el comando `git merge` para absorber todos los commits de la rama de prueba
- Fast-Forward solo hace la fusion sin preguntar nada
- Manual Merge los cambios tienen que pasar por nosotros
- Para crear y cambia automaticamente a una nueva rama se utiliza el comando `git checkout -b nueva rama`

4. Video 5: GitHub

- El comando `git clone` toma un proyecto de GitHub y lo pasa a la computadora
- El comando `git remote add origin + url` vincula el proyecto local con nuestro proyecto remoto
- Para comprobar el vinculo se utiliza el comando `git remote -v`
- Para desvincular se utiliza `git remote remove origin`
- Para sincronizar se utiliza `git push origin + la rama a pasar`
- Para hacer una prueba con las ramas en GitHub

4.1. Issues, Milestones y Labels

- Los **Issues** se utilizan para arreglar errores o algo que nos hace falta
- Los **Milestones** son grupos de Issues que se aplican para un proyecto, característica o periodo de tiempo
- Los **Labels** son etiquetas para los Issues

5. Video 6: Tags

- Los **Tags** son etiquetas para identificar la versión del archivo o historia en nuestro proyecto
- Para modificar el nombre de un commit se utiliza el comando `git commit -amend -m "nuevo/nombre"`
- El comando `git push origin master -f` obliga a GitHub a tomar las modificaciones en los commit

5.1. Tags

- Las **Tags anotadas** son almacenadas como objetos completos dentro de la base de Git y contienen más información con el comando `git tag -a -m "`
- Las **Tags Ligeras** contienen menos información
- Para agregar Tags a commits dentro del proyecto se utiliza el comando `git tag -a v0.7 -m "Version 0.7" + el código SHA`
- Para compartir las tags en GitHub se utiliza el comando `git push origin + el nombre del tag`
- Para agregar todos los tags de golpe se utiliza el comando `git push origin -tags`

6. Video 7: WorkFlows

- Crear organizaciones para el trabajo en equipo
- Para agregar una persona, se busca su nick o su correo y se le otorgan permisos de edición
- La otra persona clona el proyecto de la misma manera
- Para ver las ramas ocultar y observar la rama origin/master se utiliza el comando `git branch -a`

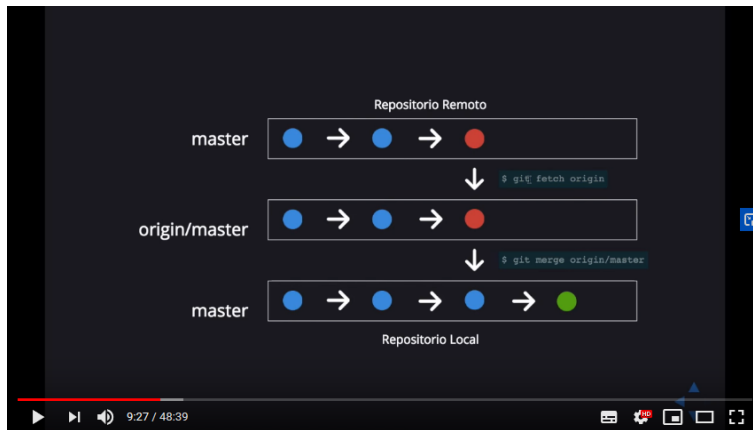


Figura 1: Comando Fetch

- Para pasar los commit de Omar y de Filli se utiliza el comando `git fetch` (Ver Fig. 1). El flujo es el siguiente: Con el comando `fetch origin` se sube todo al `origin/master` y con el comando `merge origin/master` se absorben esos cambios; una vez se tienen los cambios, se sube de nuevo al `origin master` con el comando `push`
- Se debe recordar que se debe cambiar el tipo de conexión remota, del usuario del repositorio del chest al repositorio de la organización con el comando `git remote remove -v` y se verifica con el comando `git remote -v` y se pone de nuevo `git remote add origin + código de la organización`

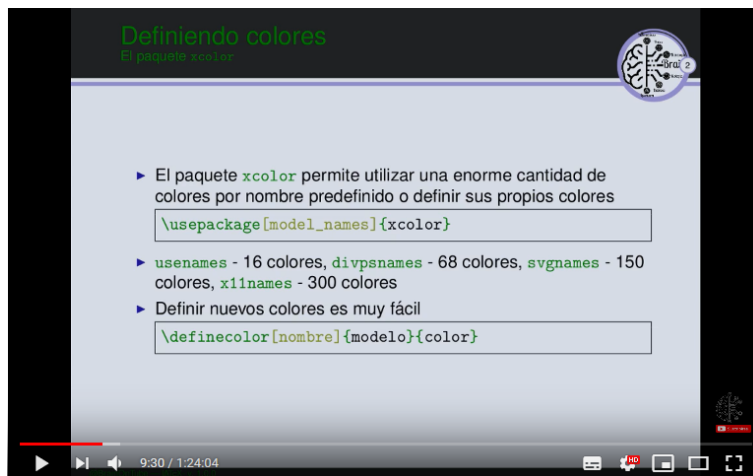


Figura 2: Colores \LaTeX