Instituto Politécnico Formosa

Desarrollo Colaborativo y Control de Versiones.

Comandos en Git:

- **git init**: inicializa un repositorio local en la carpeta indicada.
- **git config –global user.name “nombre de usuario”**: Se configura el nombre del responsable de los commits realizados.
- **git config --global user.email “emaildeusuario@email.com”**: Se configura el email del responsable de los commits realizados (utilizar las credenciales de GitHub).
- **git add <filename>**: añade un determinado archivo al working directory al staging area para luego realizar el commit.
- **git add –all:** añade todos los archivos del **working directory** al **staging area** para luego realizar el commit (Otra alternativa es **git add .**).
- **git status**: permite verificar el estado de nuestro repositorio local, brinda información sobre la rama en la que nos encontramos trabajando, y si existen archivos en los diferentes estados.
- **git commit -am "Mensaje descriptivo de los cambios realizados"**: Añadir todos los archivos al stagin area y crear el commit.
- **git remote add origin https://url_repositorio_github**: vincula nuestro repositorio local con nuestro respositorio remoto.
- **git remote -v**: Para verificar el vínculo del repositorio local con el remoto
- **git remote remove origin**: En caso de que se haya eliminado el repositorio en la nube, con este comando, podemos deshacer los cambios en el repositorio local.
- **git push -u origin master**: actualizar cambios en la rama master del repositorio en GitHub. En caso de tener varias ramas, podemos reemplazar la palabra “master” por la rama a la que se desea actualizar.
- **git push**: actualizar cambios en el repositorio de GitHub.
- **git pull**: para descargar la última versión del repositorio de GitHub.
- **git branch <branchname>**: para crear una rama.

- **git checkout <branchname>**: para cambiar a otra rama o a otro commit.
- **git checkout -b <branchaname>**: para crear una rama y al mismo tiempo moverse a la misma.
- **git branch –D <branchname>**: para eliminar una rama.
- **git stash**: almacena temporalmente (o guarda en un stash) los cambios que hayas efectuado en el código en el que estás trabajando para que puedas realizar otras acciones.
- **git stash pop**: Restaura los cambios puestos en suspenso.
- **git reset --hard <branchid>**: Permite regresar a una versión anterior de la rama eliminando la actual.
- **git reset HEAD**: permite deshacer los cambios del staging area.
- **git reset HEAD <filename>**: permite quitar un determinado archivo del stagin area.
- **gitk --all --date-order**: este comando abre una GUI (Interfaz Gráfica de Usuario) donde se pueden visualizar la estructura de nuestro proyecto.
- **git commit –amend –m "descripción nueva del commit"**: Para reemplazar la descripción del último commit realizado.
- **git push origin master –f**: Para forzar los cambios en GitHub.
- **git commit –amend**: Abre el archivo de configuración de commits para poder modificar manualmente la descripción de las confirmaciones realizadas.

TAGS ANOTADAS

Las tags anotadas son almacenadas como objetos completos dentro de la base de Git y contienen más información.

- **git tag –a v1.0 –m "Descripción del tag"**

TAGS LIGERAS

Las tags ligeras, son otra forma de crear tags, más simples y con poca información.

- **git tag v0.1**

ESPECIFICACIÓN DE TAGS

Al agregar el código SHA, podemos especificar dónde se va a aplicar una etiqueta.

- **git tag –a v0.1 –m "Descripción del tag" sha (valor hexadecimal).**

COMPARTIENDO TAGS

Cuando creamos un tag, éste no se actualiza con **git push origin master** porque sólo se especifica para el ámbito local. Para poder realizar los cambios en GitHub de la etiqueta creada, debemos utilizar el siguiente comando.

- **git push origin v0.1**: Comando para "subir" los cambios de un determinado tag.
- **git push origin --tags:** A diferencia del comando anterior, este comando nos permite "subir" todos los tags especificados en el repositorio local de nuestro pc.