Duilio Alejandro Rosciano Montoya

Comandos Básicos de Git

1. Introducción a Git

- **¿Qué es Git?**

Git es un controlador de versiones open source gratuito utilizado para manejar desde pequeños a grande proyectos con alta eficiencia

- **¿Por qué usar Git?**

Git además de ser un controlador de versiones bastante grande y famoso, es bastante fácil de usar y accesible para todo publico, sin contar su alta velocidad al ejecutar sus comandos superando a otros controladores de versiones como CVS, Perforce y ClearCase

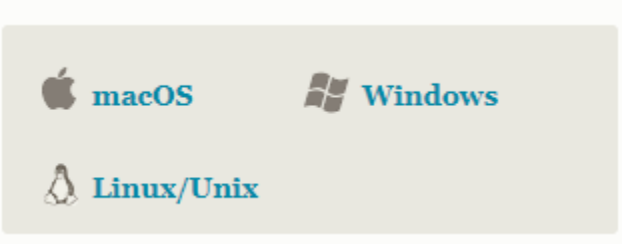
- **Principales conceptos: repositorio, commits, ramas, etc.**
    - **Repositorio:** El repositorio es el área donde estaremos ejecutando git para poder gestionar nuestro flujo de trabajo y dejar un registro, ya sea localmente en nuestra maquina o remotamente en un servidor como GitHub.
    - **GitHub:** Plataforma en línea basado en la nube que emplea el controlador de sistemas Git usado para un uso mas amigable a todas las posibles áreas. Aquí Podemos crear repositorios en la nube que podremos clonar en nuestra maquina, además de poder subir los cambios que queramos.
    - **Clone:** Al decir que clonamos un repositorio estamos diciendo que agarraremos un repositorio guardado en un servidor o la nube (en este caso GitHub) y clonar todo que tiene ese repositorio en nuestra maquina.
    - **Pull:** El comando pull se usa para jalar (pull) todos los cambios realizados de un repositorio de la nube que tengamos en nuestro equipo.
    - **Staging Area:** En esta área es donde tendremos todos los archivos a los cuales les haremos commit en un futuro. Es posible no agregar todos los archivos cambiados al staging area para poder dividir los commit de una manera mas limpia.
    - **Commit:** Este es un termino que usamos para referirnos a los cambios que queramos comprometernos a subir al repositorio, este siempre viene acompañado de un mensaje que especifica el trabajo que se realizó desde el último commit.
    - **Push:** El Push es cuando estamos empujando (pushing) el o los últimos commits que hemos hecho en nuestra maquina para subirlos al servidor o la nube.
    - **Branch:** Una Branch o rama, es un flujo de trabajo alterno usualmente paralelo a la línea principal de trabajo. Puede haber varias ramas simultáneamente para que cada integrante pueda hacer cambios sin afectar el código original.
    - **Merge:** Un Merge es cuando unimos los cambios de una rama en otra, para así poder integrar el trabajo realizado en las distintas ramas.
    - **Head:** Es un puntero que indica la ultima versión en la que se encuentra la rama actual. En cada commit el head se actualiza al nuevo commit.

- **.gitignore:** Este es un archivo que le dice a git que tipos de archivos ignorar y no subir en cada commit. Aquí se ponen todo tipo de archivos temporales generados, etc.

2. Instalación y Configuración Inicial

- **Instalación de Git**

La instalación de Git en una maquina es muy sencillo. Iremos a la pagina https://git-scm.com/downloads y haremos click en el link correspondiente dependiendo de nuestra maquina



- **Configuración global de usuario y correo**
  - `git config --global user.name "Nombre"`
    - Este Comando sirve para dejar un registro de quien ha hecho cambios y commits en el repositorio.
  - `git config --global user.email "correo@example.com"`
    - De la misma manera que en el username aquí dejaremos nuestro correo en el registro de cambios y commits.
- **Verificación de la configuración**
  - `git config --list`
    - Este comando nos ayudara a ver una lista de todas las configuraciones que tenemos.

3. Creación y Clonación de Repositorios

- **Crear un nuevo repositorio**
  - `git init`
    - Este comando le dice a git que en esta locación vamos a crear un repositorio inicializando todos los procesos de git.
- **Clonar un repositorio existente**
  - `git clone <url-del-repositorio>`
    - Aquí podremos clonar un repositorio guardado en la nube o algún servidor (GitHub) y así poder analizarlo o trabajar sobre el en nuestra maquina.
    - Es importante saber que si solo ponemos el url del repositorio, este tiene que ser un repositorio público.

4. Operaciones Básicas con Archivos

- **Ver el estado del repositorio**
  - `git status`
- **Añadir archivos al área de preparación**
  - `git add <archivo>`
    - Aquí añadiremos el o los archivos especificados al staging area.
  - `git add . (para añadir todos los archivos)`
    - De esta forma agregaremos todos los archivos cambiado al staging area.
- **Realizar un commit**
  - `git commit -m "Mensaje del commit"`
- **Ver el historial de commits**
  - `git log`

5. Trabajar con Ramas

- **Crear una nueva rama**
  - `git branch <nombre-de-la-rama>`
- **Cambiar de rama**
  - `git switch <nombre-de-la-rama>`
- **Crear y cambiar a una nueva rama en un solo paso**
  - `git switch -c <nombre-de-la-rama>`
- **Ver las ramas existentes**
  - `git branch -l`
- **Fusionar ramas**
  - `git merge <nombre-de-la-rama>`

6. Actualización y Sincronización con Repositorios Remotos

- **Añadir un repositorio remoto**
  - `git remote add origin <url-del-repositorio>`
- **Ver los remotos configurados**
  - `git remote -v`
- **Obtener cambios del repositorio remoto**
  - `git fetch`
- **Incorporar cambios del remoto a la rama actual**
  - `git pull`
- **Enviar cambios al repositorio remoto**
  - `git push`

7. Otros Comandos Útiles

- **Ver la diferencia entre archivos**
  - `git diff`
- **Ver el estado de los archivos modificados**
  - `git status`
- **Eliminar archivos del repositorio**
  - `git rm <archivo>`

8. Buenas Prácticas y Consejos

- Frecuentar el uso de los commits para dejar un registro de como va avanzando el proyecto.
- Evitar el uso de “commits atómicos” que es cuando hacemos commit únicamente cuando ya todo esta funcional en vez de hacer un commit cada vez que una funcionalidad es implementada
- Acostumbrar el uso de ramas para trabajar y así evitar hacer cambios indeseados o defectuosos en la rama principal.
- Uso de `.gitignore`