**🧠 Informe: Git y sus Comandos**

**📜 Introducción**

En el mundo del desarrollo de software, llevar un control preciso de los cambios realizados en el código es fundamental. Aquí entra **Git**, un sistema de control de versiones distribuido que ha revolucionado la forma en que los equipos trabajan en colaboración. Git es rápido, eficiente y ofrece un control total sobre el historial de versiones.

**🕰️ Historia de Git**

Git fue creado en **2005** por **Linus Torvalds**, el mismo creador del kernel de **Linux**. Surgió como una alternativa a otros sistemas de control de versiones que no se adaptaban a las necesidades del proyecto Linux, especialmente después de una ruptura con BitKeeper, que era el sistema usado en ese momento.  
Desde entonces, Git se ha convertido en el sistema de control de versiones más utilizado en el mundo, siendo la base de plataformas como **GitHub**, **GitLab** y **Bitbucket**.

**🧾 ¿Qué es Git?**

**Git** es un sistema de control de versiones **distribuido**. Esto significa que cada desarrollador tiene una copia completa del historial del proyecto. Git permite:

* Rastrear cambios realizados en los archivos.
* Trabajar en múltiples ramas de desarrollo.
* Colaborar con otros desarrolladores.
* Revertir fácilmente cambios no deseados.
* Fusionar diferentes líneas de desarrollo.

**🛠️ Comandos Básicos de Git (Con descripción y sintaxis)**

**1. git init**

* **Descripción:** Crea un nuevo repositorio Git.
* **Sintaxis:** git init

**2. git clone**

* **Descripción:** Clona un repositorio remoto.
* **Sintaxis:** git clone [url]

**3. git status**

* **Descripción:** Muestra el estado actual del repositorio.
* **Sintaxis:** git status

**4. git add**

* **Descripción:** Añade archivos al área de staging (preparados para commit).
* **Sintaxis:** git add [archivo] o git add . (todos)

**5. git commit**

* **Descripción:** Guarda los cambios en el historial del repositorio.
* **Sintaxis:** git commit -m "Mensaje del commit"

**6. git log**

* **Descripción:** Muestra el historial de commits.
* **Sintaxis:** git log

**7. git diff**

* **Descripción:** Muestra los cambios entre archivos.
* **Sintaxis:** git diff o git diff [archivo]

**8. git branch**

* **Descripción:** Lista, crea o elimina ramas.
* **Sintaxis:**
  + Ver ramas: git branch
  + Crear: git branch [nombre]
  + Eliminar: git branch -d [nombre]

**9. git checkout**

* **Descripción:** Cambia entre ramas o versiones.
* **Sintaxis:** git checkout [rama]

**10. git switch**

* **Descripción:** Alternativa moderna a checkout para cambiar de rama.
* **Sintaxis:** git switch [rama]

**11. git merge**

* **Descripción:** Une otra rama con la rama actual.
* **Sintaxis:** git merge [rama]

**12. git pull**

* **Descripción:** Descarga y fusiona cambios desde el repositorio remoto.
* **Sintaxis:** git pull

**13. git push**

* **Descripción:** Envía cambios al repositorio remoto.
* **Sintaxis:** git push

**14. git remote**

* **Descripción:** Administra conexiones a repositorios remotos.
* **Sintaxis:**
  + Ver remotos: git remote -v
  + Añadir remoto: git remote add origin [url]

**15. git reset**

* **Descripción:** Reestablece el estado de los archivos o ramas.
* **Sintaxis:**
  + git reset [archivo]
  + git reset --hard [commit]

**16. git stash**

* **Descripción:** Guarda temporalmente los cambios sin hacer commit.
* **Sintaxis:** git stash

**17. git rebase**

* **Descripción:** Reaplica cambios sobre una base diferente.
* **Sintaxis:** git rebase [rama]

**18. git tag**

* **Descripción:** Crea una etiqueta para marcar versiones.
* **Sintaxis:** git tag [nombre]

**19. git show**

* **Descripción:** Muestra información sobre objetos Git (como commits).
* **Sintaxis:** git show [objeto]

**20. git config**

* **Descripción:** Configura opciones de Git (nombre, correo, etc.).
* **Sintaxis:**
  + git config --global user.name "Tu Nombre"
  + git config --global user.email "tu@email.com"

**21. git rm**

* **Descripción:** Elimina archivos del repositorio y del área de staging.
* **Sintaxis:** git rm [archivo]

**22. git mv**

* **Descripción:** Mueve o renombra archivos en el repositorio.
* **Sintaxis:** git mv [archivo\_origen] [archivo\_destino]

**23. git fetch**

* **Descripción:** Descarga los últimos cambios del repositorio remoto sin fusionarlos.
* **Sintaxis:** git fetch

**24. git cherry-pick**

* **Descripción:** Aplica un commit específico a la rama actual.
* **Sintaxis:** git cherry-pick [hash\_commit]

**25. git revert**

* **Descripción:** Revierte un commit creando uno nuevo que deshace los cambios.
* **Sintaxis:** git revert [hash\_commit]

**26. git archive**

* **Descripción:** Crea un archivo comprimido (como .zip o .tar) del proyecto.
* **Sintaxis:** git archive --format=zip --output=nombre.zip HEAD

**27. git bisect**

* **Descripción:** Busca el commit donde se introdujo un error.
* **Sintaxis:**

css

CopiarEditar

git bisect start

git bisect bad

git bisect good [hash\_commit\_bueno]

**28. git blame**

* **Descripción:** Muestra línea por línea quién modificó un archivo.
* **Sintaxis:** git blame [archivo]

**29. git clean**

* **Descripción:** Elimina archivos no rastreados del directorio de trabajo.
* **Sintaxis:** git clean -f (¡Cuidado, no se pueden recuperar fácilmente!)

**30. git log --oneline**

* **Descripción:** Muestra el historial de commits en una línea por commit.
* **Sintaxis:** git log --oneline