**Guía Completa de Git y GitHub para Equipos de Desarrollo**

**1. Conceptos Fundamentales**

Git es un sistema de control de versiones distribuido. Esto significa que cada desarrollador tiene una copia completa del historial del proyecto en su máquina local. GitHub es una plataforma basada en la nube que utiliza Git, proporcionando un lugar centralizado para almacenar repositorios y colaborar.

**Repositorio (Repo):** Una carpeta que Git rastrea. Contiene todos los archivos del proyecto y su historial de versiones.

**Commit:** Una "instantánea" del repositorio en un momento dado. Cada commit tiene un mensaje que describe los cambios realizados.

**Rama (Branch):** Un puntero a un commit. Por defecto, la rama principal se llama main (o master). Las ramas permiten trabajar en nuevas funcionalidades o correcciones de errores de forma aislada.

**HEAD:** Un puntero que indica la rama en la que estás actualmente.

**2. Configuración Inicial de Git**

Antes de empezar, es crucial configurar tu identidad en Git. Estos datos se adjuntarán a cada uno de tus commits.

git config --global user.name "Tu Nombre"

git config --global user.email "tu.email@ejemplo.com"

**3. Flujo de Trabajo Básico (Local)**

**3.1. Creación de un Repositorio**

Para iniciar un nuevo repositorio en tu máquina local:

# Navega a la carpeta de tu proyecto

cd /ruta/a/mi/proyecto

# Inicializa Git en la carpeta

git init

**3.2. Realización de Commits**

Un commit es el paso fundamental para guardar tus cambios.

1. **Estado de los archivos:** Git categoriza los archivos en tres estados principales:
   * **Untracked:** Archivos nuevos que Git no ha visto.
   * **Unmodified:** Archivos que no han cambiado desde el último commit.
   * **Modified:** Archivos que han sido modificados desde el último commit.
   * **Staged:** Archivos modificados que has preparado para el próximo commit.
2. **Preparar los cambios (Staging):** Selecciona los archivos que quieres incluir en el commit.
3. # Para añadir un solo archivo
4. git add nombre\_del\_archivo.js
5. # Para añadir todos los archivos modificados y nuevos en la carpeta actual
6. git add .
7. **Hacer el Commit:** Guarda los cambios con un mensaje descriptivo.
8. git commit -m "Descripción concisa de los cambios"

**4. Colaboración con GitHub**

**4.1. Creación de un Repositorio en GitHub**

1. Ve a GitHub.com y haz clic en "New" o el signo + en la esquina superior derecha.
2. Dale un nombre al repositorio y una descripción.
3. Elige si será público o privado.
4. No selecciones Add a README file ni .gitignore si ya tienes archivos locales.

**4.2. Enlazar el Repositorio Local con GitHub**

Conecta tu repositorio local al remoto en GitHub.

git remote add origin [https://github.com/tu-usuario/nombre-del-repo.git](https://github.com/tu-usuario/nombre-del-repo.git)

# O si la rama principal se llama `master` en tu repositorio local

git branch -M main

**4.3. Subir Cambios (Push)**

Sube los commits de tu repositorio local al repositorio remoto en GitHub.

# Sube la rama 'main' (o 'master') al remoto 'origin'

git push -u origin main

El flag -u (upstream) configura el enlace para futuros push y pull.

**4.4. Descargar Cambios (Pull)**

Obtén los cambios más recientes del repositorio remoto.

# Descarga y fusiona los cambios de la rama remota 'main'

git pull origin main

**5. Flujo de Trabajo en Equipo (Branches y Pull Requests)**

**5.1. Flujo de Trabajo de Ramas**

El uso de ramas es crucial para el trabajo en equipo. La rama main debe ser siempre estable y funcional.

1. **Crear una rama para tu trabajo:**
2. # Crea una nueva rama y cámbiate a ella inmediatamente
3. git checkout -b feature/nueva-funcionalidad
4. **Realizar commits en tu rama:** Trabaja y haz commits como lo harías normalmente.
5. git add .
6. git commit -m "Implementación de la nueva funcionalidad"
7. **Subir la rama al remoto:**
8. git push -u origin feature/nueva-funcionalidad

**5.2. Pull Request (PR)**

Un Pull Request es una solicitud para fusionar los cambios de tu rama con otra rama (normalmente main). Es el momento de la revisión de código.

1. Una vez que subes tu rama, GitHub te ofrecerá la opción de "Open a Pull Request".
2. Escribe un título claro y una descripción detallada de los cambios.
3. Asigna revisores (miembros del equipo).
4. Cuando los revisores aprueban los cambios, puedes fusionar la PR.

**5.3. Fusión de Ramas (Merge)**

Cuando los cambios son aprobados, puedes fusionarlos en la rama principal. Esto se hace a través del botón "Merge pull request" en la interfaz de GitHub.

También puedes hacerlo localmente (evitar si se usa PR):

# Cambia a la rama principal

git checkout main

# Descarga los cambios más recientes

git pull origin main

# Fusiona tu rama de funcionalidad

git merge feature/nueva-funcionalidad

**5.4. Resolución de Conflictos**

Los conflictos ocurren cuando dos desarrolladores modifican la misma línea de código.

1. **Identifica los conflictos:** Git te informará cuando un pull o merge no pueda completarse debido a un conflicto.
2. **Edita los archivos:** Abre los archivos en conflicto. Verás marcadores de Git (<<<<<<<, =======, >>>>>>>) que indican las versiones en conflicto.
3. **Elige la versión correcta:** Elimina los marcadores y edita el código para reflejar la versión deseada.
4. **Marca el conflicto como resuelto:**
5. git add nombre\_del\_archivo\_en\_conflicto.js
6. **Completa el merge o pull:**
7. git commit -m "Resolución de conflictos"

**6. Comandos Avanzados para Equipos**

**Clonar un repositorio:** Para obtener una copia completa de un repositorio remoto.

git clone [https://github.com/usuario/mi-proyecto.git](https://github.com/usuario/mi-proyecto.git)

**Ver el estado del repositorio:**

git status

**Ver el historial de commits:**

git log --oneline

--oneline muestra una versión resumida.

**Deshacer cambios (con precaución):**

* **Volver a un commit anterior (temporal):**
* git reset --hard HEAD~1

Esto borra el último commit y sus cambios. **No usar si ya lo has subido.**

* **Deshacer un commit sin borrar los cambios:**
* git reset HEAD~1
* **Crear un nuevo commit que revierte los cambios de otro:**
* git revert <hash-del-commit-a-revertir>

Esto es seguro para commits que ya se han subido.

**Squashing de commits:** Combina varios commits en uno solo para mantener un historial limpio.

git rebase -i HEAD~N

N es el número de commits que quieres combinar.

**7. Consejos para un Flujo de Trabajo Sin Conflictos**

* **Trabaja en ramas separadas:** Nunca trabajes directamente en main.
* **Haz pull a menudo:** Descarga los cambios de main antes de empezar a trabajar y antes de fusionar tu rama.
* **Commits atómicos:** Cada commit debe contener un solo cambio lógico.
* **Mensajes de commit claros:** Describe lo que hiciste y por qué.
* **Revisión de código:** Fomenta la revisión de código por pares a través de Pull Requests.
* **Utiliza .gitignore:** Ignora archivos generados automáticamente (logs, node\_modules, etc.) para no subirlos al repositorio.