**Informe: Comandos de Git**

**Introducción**

Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Permite rastrear cambios en el código, colaborar en equipo y gestionar proyectos de manera eficiente. A continuación, se presentan los comandos esenciales de Git.

**1. Configuración Inicial**

Antes de usar Git, es necesario configurar el usuario y el correo electrónico:

b

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

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

Para ver la configuración actual:

git config --list

**2. Iniciar un Repositorio**

Para crear un nuevo repositorio o clonar uno existente:

* **Inicializar un repositorio local:**

git init

* **Clonar un repositorio remoto:**

git clone <URL-del-repositorio>

**3. Comandos Básicos de Trabajo**

**A. Verificar el Estado del Repositorio**

git status

**B. Agregar Cambios al Área de Preparación (Staging)**

* Agregar un archivo específico:

git add <nombre-del-archivo>

* Agregar todos los archivos modificados:

git add .

**C. Confirmar Cambios (Commit)**

git commit -m "Mensaje descriptivo del cambio"

**D. Ver el Historial de Commits**

git log

**4. Ramas (Branches)**

Git permite trabajar en ramas paralelas para desarrollar nuevas funciones sin afectar la rama principal.

* **Listar ramas:**

git branch

* **Crear una nueva rama:**

git branch <nombre-de-la-rama>

* **Cambiar de rama:**

git checkout <nombre-de-la-rama>

* **Fusionar ramas (merge):**

git merge <nombre-de-la-rama>

**5. Trabajo con Repositorios Remotos**

**A. Conectar un Repositorio Remoto**

git remote add origin <URL-del-repositorio>

**B. Subir Cambios (Push)**

git push -u origin <nombre-de-la-rama>

**C. Actualizar el Repositorio Local (Pull)**

git pull origin <nombre-de-la-rama>

**D. Traer Cambios sin Fusionar (Fetch)**

git fetch

**6. Deshacer Cambios**

* **Restaurar un archivo modificado:**

git checkout -- <nombre-del-archivo>

* **Eliminar cambios del área de staging:**

git reset HEAD <nombre-del-archivo>

* **Revertir un commit:**

git revert <hash-del-commit>

**Conclusión**

Git es una herramienta poderosa para el control de versiones. Dominar estos comandos básicos permite gestionar proyectos de manera eficiente, colaborar en equipo y mantener un historial de cambios organizado.