Repaso de Git

1. Introducción

Git es un sistema de control de versiones distribuido que permite a múltiples desarrolladores trabajar en el mismo proyecto de manera colaborativa sin sobrescribir el trabajo de los demás.

Beneficios principales:

- **Historial Completo:** Cada cambio se registra, permitiendo revertir a versiones anteriores.
- **Colaboración Eficaz:** Fusiona cambios de diferentes personas sin conflictos.
- **Gestión de Ramas:** Permite experimentar sin afectar la rama principal.
- **Respaldo y Recuperación:** El código está seguro y recuperable.

---

2. Configuración Inicial de Git

Ejecuta en tu terminal:

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu.email@ejemplo.com"
```

---

3. Primeros Pasos: Inicializando un Repositorio

Si queremos crear un nuevo repositorio para comenzar un proyecto:

3.1. Creando un Nuevo Proyecto

```
cd /ruta/a/tu/proyecto
git init
```

Si queremos trabajar en el repositorio de un proyecto que ya existe, en cambio:

3.2. Clonando un Proyecto Existente

```
git clone <URL_del_repositorio_remoto>
```

---

4. El Ciclo de Vida de los Cambios: Add, Commit, Push

4.1. Ver Estado

```
git status
```

4.2. Añadir Cambios

```
git add nombre_del_archivo.txt
# o para todos los cambios
git add .
```

4.3. Confirmar Cambios

```
git commit -m "Mensaje descriptivo"
```

4.4. Subir Cambios

```
git push origin some_branch
```

---

5. Ramas (Branches)

5.1. Ver branches

```
git branch
```

5.2. Crear Rama

```
git branch new_feature
```

5.3. Cambiar de Rama

```
git checkout new_feature
```

5.4. Crear y Cambiar a Rama (Atajo)

```
git checkout -b new_feature
```

5.5. Borrar Rama

```
git branch -d new_feature
```

---

6. Actualizando Repositorio Local

```
git pull origin nombre_de_tu_rama
```

---

7. Deshaciendo Cambios

7.1. Descartar Cambios No Confirmados

```
git checkout -- nombre_del_archivo.txt
```

7.2. Deshacer Último Commit

```
git reset HEAD~1
# o con hard reset (con precaución)
git reset --hard HEAD~1
```

---

8. Revisando el Historial (git log)

```
git log
# Opciones útiles
git log --oneline
git log --graph --oneline --decorate
git log -p
git log --author="Tu Nombre"
```

---

9. Tags

Los *tags* en Git son referencias que marcan puntos específicos en la historia de tu repositorio, comúnmente usados para identificar versiones de lanzamiento o hitos importantes.

9.1. Crear Tag

```
git tag v1.0.0
```

9.2. Crear Tag Anotado

```
git tag -a v1.0.0 -m "Versión inicial"
```

9.3. Listar Tags

```
git tag
```

9.4. Subir Tags

```
git push origin --tags
```

---

10. .gitignore

El archivo `.gitignore` se utiliza para indicar a Git qué archivos o carpetas no deben ser trackeados, como serian archivos temporales, archivos de contraseñas, configuraciones locales o dependencias generadas automáticamente.

Ejemplo de archivo:

```
*.log
```

```
node_modules/
mi_archivo_secreto.env
.DS_Store
```

Ejercicio Práctico

1. Crear un directorio nuevo.
2. Inicializar un repositorio Git.
3. Crear un archivo `README.md` y hacer commit.
4. Crear y cambiar a la rama `feature-suma`.
5. Agregar un archivo `calculadora.py` con una función de suma.
6. Hacer commit de `calculadora.py`.
7. Taggear el último commit como “v0.1”
8. Pushear y revisar que el tag este visible en GitHub.
