¿Qué es Git?

Git es un sistema de control de versiones que permite llevar un historial de cambios en proyectos de software, facilitando el trabajo en equipo y la recuperación de versiones anteriores. Se utiliza en desarrollo de software, especialmente en colaboración con plataformas como GitHub, GitLab y Bitbucket.

Lista de Comandos Básicos de Git

1. Configuración Inicial

Comando	Descripción
`git config --global user.name "Tu Nombre"`	Configura tu nombre en Git.
`git config --global user.email "tuemail@example.com"`	Configura tu correo en Git.
`git config --list`	Muestra la configuración actual de Git.

2. Inicializar y Clonar un Repositorio

Comando	Descripción
`git init`	Inicializa un nuevo repositorio en la carpeta actual.
`git clone URL_DEL_REPOSITORIO`	Clona un repositorio remoto en tu computadora.

3. Trabajando con Cambios

Comando	Descripción
`git status`	Muestra el estado actual del repositorio (archivos modificados, sin seguimiento, etc.).
`git add archivo.txt`	Agrega un archivo específico al área de preparación.
`git add .`	Agrega todos los archivos modificados al área de preparación.

`git commit -m "Mensaje del commit"`	Guarda los cambios en el historial de Git con un mensaje descriptivo.

4. Trabajando con Ramas (Branches)

Comando	Descripción
`git branch`	Muestra todas las ramas disponibles.
`git branch nueva-rama`	Crea una nueva rama llamada nueva-rama.
`git checkout nueva-rama`	Cambia a la rama nueva-rama.
`git checkout -b nueva-rama`	Comando avanzado para información del sistema
`git branch -d rama`	Crea y cambia a la nueva rama en un solo comando.
`git push origin --delete rama`	Elimina una rama local.
`git push origin --delete rama`	Elimina una rama en el repositorio remoto.

5.Sincronización con GitHub

Comando	Descripción
`git remote add origin URL_DEL_REPOSITORIO`	Enlaza el repositorio local con GitHub
`git pull origin main`	Descarga cambios del repositorio remoto a la rama actual.
`git push origin main`	Sube los cambios al repositorio remoto.
`git push -u origin main`	Sube los cambios y establece la rama principal como predeterminada.

6. Deshacer Cambios

Comando	Descripción

`git reset archivo.txt`	Quita el archivo del área de preparación sin perder los cambios.
`git checkout -- archivo.txt`	Restaura un archivo al último commit guardado.
`git reset --hard HEAD`	Revierte todos los cambios locales a la última versión confirmada.
`git revert ID_DEL_COMMIT`	Revierte un commit sin perder el historial.

7.Ver Historial y Diferencias

Atajo	Acción
`git log`	Muestra el historial de commits.
`git log --oneline`	Muestra el historial en una sola línea por commit.
`git diff`	Muestra las diferencias entre archivos modificados y la última versión guardada.

Guía de Buenas Prácticas en Git y Unity

Para trabajar de manera eficiente con Git y Unity, sigue estas buenas prácticas:

Organización del Repositorio

- Ignorar archivos innecesarios con .gitignore:
  Evita subir archivos generados automáticamente por Unity como Library/, Temp/, Logs/ y obj/.
  Usa el siguiente .gitignore para Unity:

```
# Carpetas y archivos innecesarios de Unity
[Llibrería/](Library/)
[Temp/](Temp/)
[Logs/](Logs/)
[Obj/](obj/)
[*.csproj](*.csproj)
[*.sln](*.sln)
```

Estructura organizada:

Mantén un orden lógico en Assets/, separando scripts, modelos, texturas, etc.

Uso de Ramas

Usa ramas para nuevas funcionalidades en lugar de trabajar en main:

```
git checkout -b feature-nueva-funcionalidad
```

Fusiona cambios en main solo cuando la funcionalidad esté terminada y probada:

```
git checkout main
git merge feature-nueva-funcionalidad
```

Commits Claros y Frecuentes

Usa mensajes de commit descriptivos

```
git commit -m "Añadida animación de salto para el personaje principal"
```

Evita mensajes vagos como "cambios" o "actualización".

Commits pequeños y frecuentes → No esperes a hacer grandes cambios sin registrar avances.

Sincronización Regular con el Repositorio Remoto

Antes de empezar a trabajar, sincroniza con el repositorio remoto:

git pull origin main

Flujo de trabajo en Git

De manera general el **Flujo de Trabajo en Git** funciona de la siguiente manera:

1. **Inicializar o Clonar un Repositorio**

Si el proyecto es nuevo, se inicializa con: **git init**

Si el proyecto ya existe en GitHub, se clona con:git clone URL_DEL_REPOSITORIO

2. **Crear o Modificar Archivo:**

Se crean o editan archivos dentro del proyecto

3. **Verificar el Estado de los Cambios**

Se usa el siguiente comando para ver qué archivos han sido modificados o agregados: **git status**

4. **Añadir Cambios al Área de Preparación**

Se agregan archivos específicos al área de preparación (staging area): **git add archivo.txt**

Para agregar todos los archivos modificados: **git add .**

5. **Realizar un Commit**

Se guarda un punto en el historial con una descripción de los cambios: **git commit -m "Descripción del cambio realizado"**

6. **Subir Cambios al Repositorio Remoto**

Se envían los cambios al repositorio en GitHub: **git push origin main**

7. **Descargar Cambios del Repositorio Remoto**

Antes de hacer nuevos cambios, se recomienda actualizar la versión local: **git pull origin main**

8. **Trabajar con Ramas (Opcional)**

Se crea una nueva rama para trabajar en una nueva funcionalidad sin afectar main: **git checkout -b nueva-funcionalidad**

Una vez terminados los cambios, se integra la rama en main:

git checkout main

git merge nueva-funcionalidad