Ejemplo práctico

- Veremos temas como uso de: branch's, merge, commit, status, log, etc.
- Al final hay un resumen de cada punto tocado.
- Se mostrará como crear un repositorio central con git.

Recomendaciones

- Traer siempre la última versión con un **pull**, y correr los test's, para evitar **conflictos de archivos** e incosistencia en el proyecto.
- Nunca dejar **cambios sin guardar** al termino del horario de trabajo.
- Realizar commit's con **comentarios significativos**, a fin de mantener el repositorio limpio y ordenado.
- Resolver todos los **conflictos** al hacer un **merge** antes de enviar los cambios con **push**.

Resumen de lo que se realizará en la práctica

- Primero se crea un **repositorio git local** "myproject" y se guardaron revisiones.
- Luego se crea un **branch** llamado "nueva_rama" y se guardaron revisiones.
- **Se fusinan** los cambios del branch "nueva_rama" con "master".
- Creamos un **repositorio central** llamado "mirepo.git" a partir de "myproject".
- **Clonamos** "mirepo.git" con nombre "newproject".
- Se hacen cambios, **guardamos** y luego enviamos al repositorio central "mirepo.git".
- **Clonamos** otra vez de "mirepo.git" llamado "miproyecto" y nos fijamos el **historial de revisiones**.
- Creamos una nueva rama, hacemos cambios, **comparamos y fusionamos**, finalmente enviamos los cambios al repositorio "mirepo.git".
- Por último vamos a "newproject" a **actualizar** con pull.

Práctica

Crear carpeta vacía "myproject", e iniciamos

```
$ git init
```

Crear archivo1.txt y archivo2.txt y agregarlos

```
$ git add .
```

Guardando la primera revisión

```
$ git commit -m "Iniciando proyecto"
```

Crear archivo3.txt y archivo4.txt, ver estado

```
$ git status
```

Agregando los nuevos archivos

```
$ git add archivo3.txt archivo4.txt
```

Guardamos una nueva revisión

```
$ git commit -m "Segunda revisión"
```

Ver el historial detallado

```
$ git log --stat
```

Crear un nueva rama

```
$ git checkout -b nueva_rama
```

Crear carpeta demo y dentro archivo5.txt y agregarlos

```
$ git add demo/
```

Guardar cambios

```
$ git commit -m "Tercera revisión"
```

Regresar a master

```
$ git checkout master
```

Fusionar con nueva_rama

```
$ git merge nueva_rama
```

Salimos de la carpeta y creamos un repo

```
$ git clone --bare myproject mirepo.git
```

Clonamos otro proyecto a partir del repo

```
$ git clone mirepo.git newporject
```

Entramos en newproject, creamos archivo6.txt y lo agregamos

```
$ git add archivo6.txt
```

Guardamos los cambios

```
$ git commit -m "Agregado archivo6.txt"
```

Nos aseguramos de tener la última versión del repo

```
$ git pull origin master
```

Ahora enviaremos nuestros cambios al repo

```
$ git push origin master
```

Salimos de newproject y clonamos otro

```
$ git clone mirepo.git miproyecto
```

Nos fijamos el historial

```
$ git log
```

Creamos una nueva rama

```
$ git checkout -b nuevos_cambios
```

Creamos archivo7.txt, lo agregamos y guardamos

```
$ git add archivo7.txt

$ git commit -m "Agregando archivo7.txt"
```

Regresamos a master y comparamos

```
$ git checkout master

$ git diff master nuevos_cambios
```

Hacemos merge y enviamos los cambios

```
$ git merge nuevos_cambios

$ git push origin master
```

Eliminamos la rama que ya no nos sirve

```
$ git branch -d nuevos_cambios
```

Nos vamos a newproject y traemos los nuevos cambios

```
$ git pull origin master
```

Otros comandos útiles

Por ejemplo para revisar el trabajo realizado:

Ver el historial de cambios.

```
$ git log
```

Ver solo las últimas 3 revisiones.

```
$ git log -n 3
```

Ver cambios de forma detallada.

```
$ git log --stat --sumary
```

Comandos para corregir errores

Descartar todos los cambios hechos desde la última revisión

```
$ git reset --hard
```

Descartar cambios en eun archivo individual (actualizar a la última revisión conocida)

```
$ git checkout archivo
```

Corrigiendo la descripción de la última revisión (el último commit)

```
$ git commit --amend
```

Incluir algo que debió estar en la última revisión creada

```
$ git reset --soft HEAD^

$ git add olvidado1 olvidado2

$ git commit
```

Trabajando con un repositorio remoto

Crear un “clon” de un repositorio remoto

```
$ git clone git://servidor.com/ruta/proyecto.git
```

Agregar el origen remoto

```
$ git remote add origin git://servidor.com/ruta/proyecto.git
```

Hacer cambios y agregar los archivos modificados a una nueva revisión

```
$ git add archivo1 archivo2 carpeta1 carpeta2

$ git commit -m "Nueva revision"
```

Enviar los cambios de vuelta al repositorio remoto

```
$ git push origin master
```

Conclusiones

- **Git** es una herramienta **rápida, eficiente y moderna** para el **control de versiones** de proyectos con archivos digitales que sean o no código fuente.
- **Git** permite **flujos de trabajo distribuidos** en los que no tiene porque existir un repositorio central.