Laboratorio corrección de conflictos

Laboratorio #1 sencillo

Paso 1: Configurar el entorno

1. **Crear un nuevo repositorio:**
   Si no tienes un repositorio creado, inicia uno nuevo:

   ```
   git init git-laboratorio
   cd git-laboratorio
   ```

2. **Crear un archivo y hacer el primer commit:**
   Crea un archivo `archivo.txt` y añádele algo de contenido:

   ```
   echo "Esta es la línea original" > archivo.txt
   ```

   Agrega el archivo al repositorio y realiza el primer commit:

```
git add archivo.txt
git commit -m "Primer commit: archivo.txt creado con una línea inicial"
```

Paso 2: Crear ramas para provocar el conflicto

1. **Crear la rama `rama1`:**

   ```
   git checkout -b rama1
   ```

2. **Modificar `archivo.txt` en `rama1`:**
   Edita el archivo en `rama1`:

   ```
   echo "Línea agregada en rama1" >> archivo.txt
   ```

   Haz commit del cambio:

   ```
   git add archivo.txt
   git commit -m "Agregada línea en rama1"
   ```

3. **Volver a la rama principal (`main` o `master`):**

   ```
   git checkout main
   ```

4. **Crear la rama `rama2`:**

   ```
   git checkout -b rama2
   ```

5. **Modificar `archivo.txt` en `rama2`:**
   Edita el archivo en `rama2` de manera que provoque un conflicto:

   ```
   echo "Línea agregada en rama2" >> archivo.txt
   ```

   Haz commit del cambio:

```
git add archivo.txt
git commit -m "Agregada línea en rama2"
```

Paso 3: Provocar el conflicto

1. **Volver a la rama principal (`main`):**

   ```
   git checkout main
   ```

2. **Hacer merge de `rama1` en `main`:**

   ```
   git merge rama1
   ```

   Esto no debería causar ningún conflicto ya que `rama1` y `main` no tienen cambios que se superpongan.

3. **Intentar hacer merge de `rama2` en `main`:**
   Aquí es donde ocurrirá el conflicto.

   ```
   git merge rama2
   ```

   Git te informará de un conflicto en `archivo.txt`.

Paso 4: Resolver el conflicto

1. **Revisar el estado del conflicto:**
   Git te informará de los archivos en conflicto:

   ```
   git status
   ```

   Abre el archivo `archivo.txt` y verás algo como esto:

   ```
   Esta es la línea original
   <<<<<<< HEAD
   Línea agregada en rama1
   =======
   Línea agregada en rama2
   >>>>>>> rama2
   ```

2. **Resolver el conflicto:**
   Edita el archivo para resolver el conflicto, elige la versión que quieras mantener o combina ambas líneas:

```
Esta es la línea original
Línea agregada en rama1
Línea agregada en rama2
```

3. **Agregar el archivo resuelto:**
   Una vez resuelto el conflicto, añade el archivo al área de staging:

```
git add archivo.txt
```

4. **Completar el merge:**
   Finalmente, termina el merge:

```
git commit -m "Conflicto resuelto entre rama1 y rama2"
```

Paso 5: Verificación

Puedes usar `git log --graph --oneline` para visualizar cómo han quedado las ramas después de los merges y ver el historial.

Paso 6: Limpiar (opcional)

Si quieres eliminar las ramas después de la práctica:

```
git branch -d rama1
git branch -d rama2
```

Con este laboratorio, tendrás la oportunidad de experimentar con la resolución de conflictos en Git de una manera práctica y segura.

Laboratorio #2 avanzado

Vamos a resolver conflictos pero esta vez con múltiples conflictos en un solo archivo

Paso 1: Configurar el entorno

1. **Crear un nuevo repositorio:**
   Si no tienes un repositorio creado, inicia uno nuevo:

   ```
   git init git-laboratorio-avanzado
   cd git-laboratorio-avanzado
   ```

2. **Crear un archivo y hacer el primer commit:**
   Crea un archivo
   `archivo-complejo.txt` con varias líneas:

   ```
   echo -e "Línea 1: Este es el inicio\nLínea 2: Trabajo in
   icial\nLínea 3: Pendiente de cambios\nLínea 4: Más traba
   jo pendiente\nLínea 5: Fin del archivo" > archivo-comple
   jo.txt
   ```

   Haz el primer commit:

   ```
   git add archivo-complejo.txt
   git commit -m "Primer commit: archivo-complejo.txt cread
   o con varias líneas"
   ```

Paso 2: Crear las ramas y hacer los cambios

1. **Crear la primera rama `ramaA`:**

   ```
   git checkout -b ramaA
   ```

2. **Modificar `archivo-complejo.txt` en `ramaA`:**
   Vamos a modificar varias líneas en
   `ramaA`:

   ```
   sed -i 's/Trabajo inicial/Trabajo realizado en ramaA/' a
   rchivo-complejo.txt
   sed -i 's/Pendiente de cambios/Trabajo completado en ram
   aA/' archivo-complejo.txt
   ```

Haz commit de los cambios:

```
git add archivo-complejo.txt
git commit -m "Modificado archivo en ramaA"
```

3. **Volver a la rama principal (`main`):**

```
git checkout main
```

4. **Crear la segunda rama `ramaB`:**

```
git checkout -b ramaB
```

5. **Modificar `archivo-complejo.txt` en `ramaB`:**
   Ahora vamos a hacer modificaciones en las mismas líneas en `ramaB`, para provocar múltiples conflictos:

```
sed -i 's/Trabajo inicial/Trabajo realizado en ramaB/' a
rchivo-complejo.txt
sed -i 's/Pendiente de cambios/Trabajo completado en ram
aB/' archivo-complejo.txt
sed -i 's/Más trabajo pendiente/Tarea adicional completa
da en ramaB/' archivo-complejo.txt
```

   Haz commit de los cambios:

```
git add archivo-complejo.txt
git commit -m "Modificado archivo en ramaB"
```

Paso 3: Provocar múltiples conflictos

1. **Volver a la rama principal (`main`):**

```
git checkout main
```

2. **Hacer merge de `ramaA` en `main`:**
   Esto no causará conflictos, ya que solo estamos combinando `ramaA` con `main`.

```
git merge ramaA
```

3. **Intentar hacer merge de `ramaB` en `main`:**
   Aquí es donde se producirán varios conflictos.

   ```
   git merge ramaB
   ```

   Ahora deberías ver algo como:

   ```
   CONFLICT (content): Merge conflict in archivo-complejo.t
   xt
   Automatic merge failed; fix conflicts and then commit th
   e result.
   ```

Paso 4: Revisar y resolver los conflictos

1. **Verificar el estado del conflicto:**
   Ejecuta
   `git status` para ver los archivos en conflicto.

   ```
   git status
   ```

   Git te informará que el archivo `archivo-complejo.txt` tiene conflictos.

2. **Abrir el archivo para ver los conflictos:**
   Abre
   `archivo-complejo.txt` y verás algo así:

   ```
   Línea 1: Este es el inicio
   <<<<<<< HEAD
   Línea 2: Trabajo realizado en ramaA
   Línea 3: Trabajo completado en ramaA
   =======
   Línea 2: Trabajo realizado en ramaB
   Línea 3: Trabajo completado en ramaB
   Línea 4: Tarea adicional completada en ramaB
   ```

```
>>>>>>> ramaB
Línea 5: Fin del archivo
```

3. **Resolver los conflictos manualmente:**
   Puedes decidir cómo combinar los cambios. Por ejemplo, puedes combinar ambas versiones de las líneas de esta forma:

```
Línea 1: Este es el inicio
Línea 2: Trabajo realizado en ramaA y ramaB
Línea 3: Trabajo completado en ramaA y ramaB
Línea 4: Tarea adicional completada en ramaB
Línea 5: Fin del archivo
```

4. **Agregar el archivo resuelto:**
   Una vez que hayas resuelto los conflictos, agrega el archivo al área de staging:

```
git add archivo-complejo.txt
```

5. **Completar el merge:**
   Termina el merge con un commit:

```
git commit -m "Conflictos resueltos entre ramaA y ramaB"
```

Paso 5: Verificación y limpieza

1. **Revisar el historial:**
   Puedes usar
   `git log --graph --oneline` para ver cómo quedó el historial de las ramas después de los merges.

2. **Limpiar las ramas (opcional):**
   Si has terminado, puedes eliminar las ramas:

```
git branch -d ramaA
git branch -d ramaB
```

Conclusión:

Este laboratorio simula un escenario real en el que múltiples desarrolladores hacen cambios en distintas partes de un archivo y cómo resolver los conflictos resultantes. Los conflictos están distribuidos en varias líneas, lo que hace que sea más interesante y cercano a situaciones del mundo real.

Este laboratorio te ayudará a fortalecer tus habilidades para manejar conflictos más complejos en Git.