

# Herramientas de Desarrollo

**Semana 05**

**Resolver conflictos en sistema de control de versiones y colaboración con repositorios remotos**



**Universidad  
Tecnológica  
del Perú**

**¿Tienen alguna consulta o duda sobre la clase anterior?**



# Logro de la Unidad

Al finalizar la unidad, el estudiante gestiona los sistemas de control de versiones para el desarrollo de una solución de software.



# Utilidad

- ¿Qué hacer si dos de nuestros colaboradores han realizado cambios en una misma área de codificación?

# contenido

**1** Introducción

**2** ¿Qué es un conflicto en un sistema de control de versiones?

**3** Resolución de Conflictos

**4** Colaboración con Repositorios Remotos

# Transformación

## 1 Introducción



# Introducción

En el desarrollo de software colaborativo, el uso de **sistemas de control de versiones** (SCV) como **Git** se ha vuelto fundamental para gestionar el código fuente, coordinar equipos de trabajo y mantener el historial de cambios.

Estos sistemas permiten que múltiples desarrolladores trabajen simultáneamente en diferentes partes del proyecto sin interferir con el trabajo de otros.

## 2 ¿Qué es un conflicto en un sistema de control de versiones?



# ¿Qué es un conflicto en un sistema de control de versiones?



Un **conflicto** ocurre cuando dos o más colaboradores intentan hacer cambios en la misma parte del código, y esos cambios son incompatibles entre sí. Git no puede decidir automáticamente qué versión del código debe prevalecer, por lo que requiere intervención manual para resolverlo.

## Cómo se originan los conflictos:

**Misma línea del mismo archivo modificada:** Dos personas modifican la misma línea de código en el mismo archivo.

**Renombrado o movimiento de archivos:** Un archivo es renombrado o movido en una rama y, al mismo tiempo, se hace un cambio diferente en ese archivo en otra rama.

# 3 Resolución de Conflictos

# Resolución de Conflictos:



## Pasos básicos para resolver un conflicto en Git:

### **Detectar un conflicto:**

Esto ocurre durante una operación de fusión (merge) o rebase (rebase).

Git marcará los archivos en conflicto con un estado de "conflict" y te indicará qué líneas están en conflicto.

### **Ver archivos en conflicto:**

Puedes usar el siguiente comando para ver los archivos con conflictos:

`git status`

# Resolución de Conflictos:



## ...Pasos básicos para resolver un conflicto en Git:

### Resolver el conflicto manualmente:

Abre los archivos en conflicto. Git coloca marcas dentro del archivo para indicar las secciones conflictivas:

```
<<<<<<< HEAD
```

```
[Tu versión del código]
```

```
=====
```

```
[La versión que viene de la otra rama]
```

```
>>>>>>> feature/otra-rama
```

Edita el archivo para resolver el conflicto, eliminando las marcas <<<<<<<, =====, y >>>>>>>, y dejando el código que desees conservar.

# Resolución de Conflictos:



## ...Pasos básicos para resolver un conflicto en Git:

### Marcar el conflicto como resuelto:

Después de editar el archivo, marca el conflicto como resuelto con:

```
git add <archivo>
```

### Finalizar la fusión:

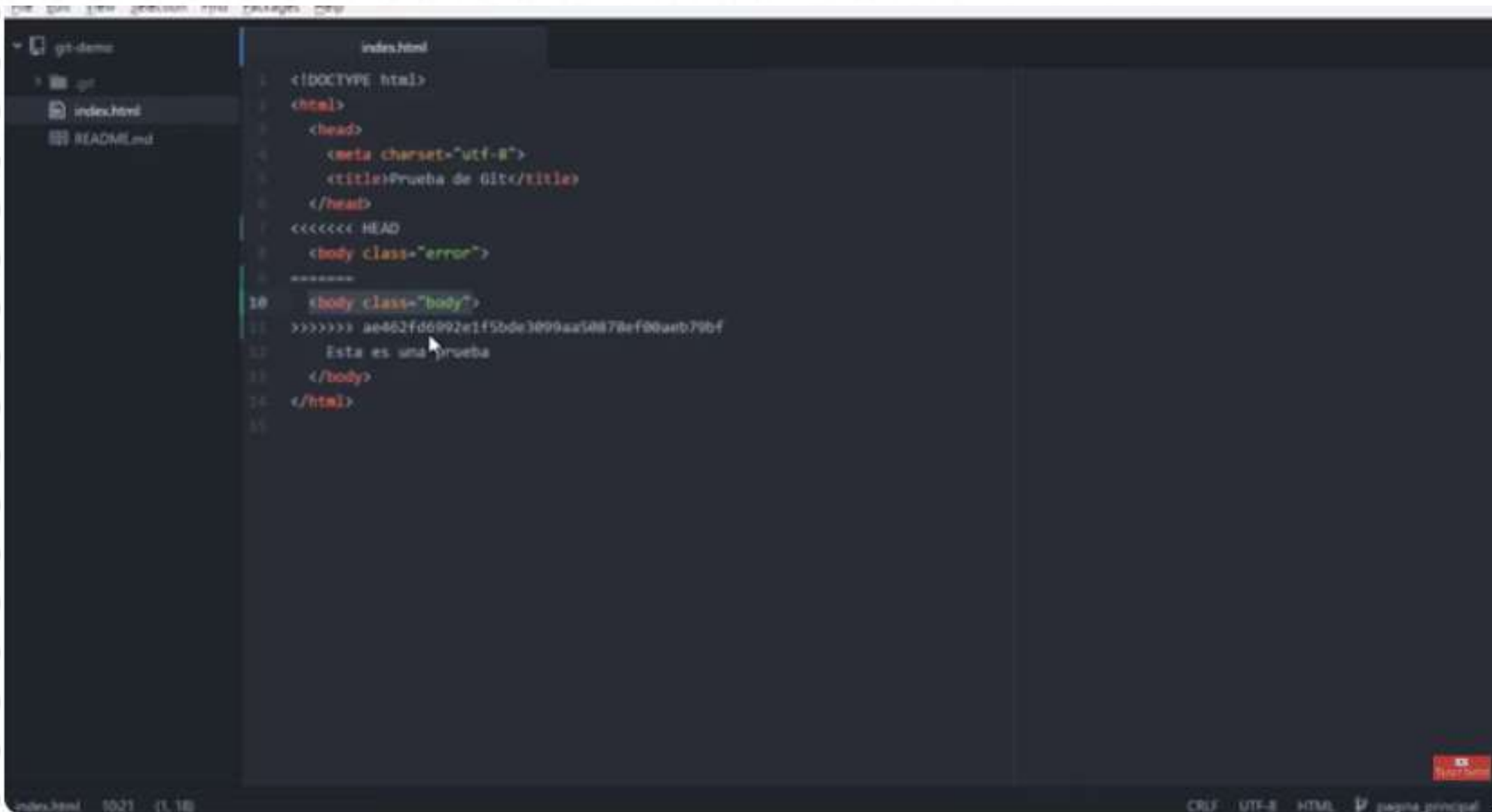
Si estás haciendo un merge, simplemente realiza un commit para completar la fusión:

```
git commit
```

### Envío de cambios al repositorio remoto:

Si el conflicto se ha resuelto y ya has hecho el commit de la fusión, envía tus cambios al repositorio remoto con:

```
git push origin <rama>
```



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Prueba de Git</title>
6   </head>
7   <body class="error">
8     <div>
9       <div>
10        <div class="body">
11          >>>>>> ae462fd6992e1f5bde3099aa50878ef00a0b79bf
12          Esta es una prueba
13        </div>
14      </div>
15    </body>
16  </html>
```

Corregir un error de conflicto en git "CONFLICT (content): Merge conflict"

<https://www.youtube.com/watch?v=hxEuxdCzrxE>



# 4 Colaboración con Repositorios Remotos



# Colaboración con Repositorios Remotos:



El trabajo con **repositorios remotos** permite a múltiples desarrolladores colaborar de manera eficiente en un proyecto. Las acciones principales son **clone**, **pull**, **push**, y **fetch**.

## Clonar un repositorio remoto:

Se usa para crear una copia local de un repositorio remoto.

```
git clone https://github.com/usuario/repositorio.git
```

# Colaboración con Repositorios Remotos:



## Obtener cambios del repositorio remoto:

**git fetch** descarga los cambios desde el repositorio remoto, pero no los aplica a tu rama actual.

**git pull** descarga y fusiona los cambios de la rama remota con tu rama actual.

```
git fetch origin
```

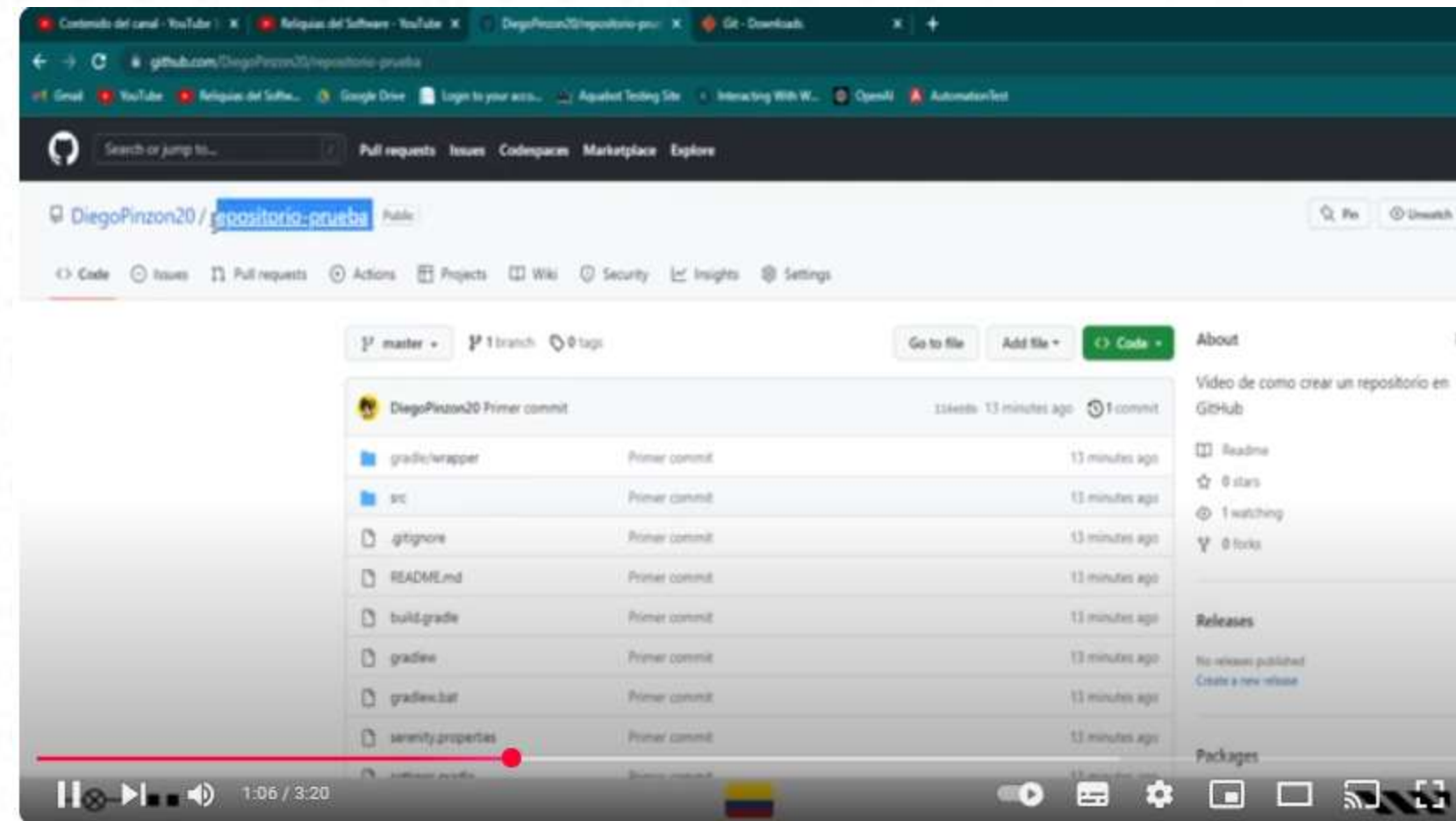
```
git pull origin main
```

## Enviar cambios al repositorio remoto:

**git push** se utiliza para enviar tus commits locales a un repositorio remoto.

```
git push origin feature/mi-rama
```

# Como clonar un repositorio remoto



Como clonar un repositorio remoto - GIT

<https://www.youtube.com/watch?v=bz-Vr7g8Yao>

# Git Fetch y git pull Cómo funcionan



Git Fetch y git pull 🚀 Cómo funcionan | Diferencias | GitHub | Repositorio remoto | Git ✅

<https://www.youtube.com/watch?v=A7cSX3ZBCws>

# Práctica

**Verificar si tu repositorio local está desactualizado respecto al remoto**

**¿Qué git debo utilizar para saber qué archivos están en conflicto?.**

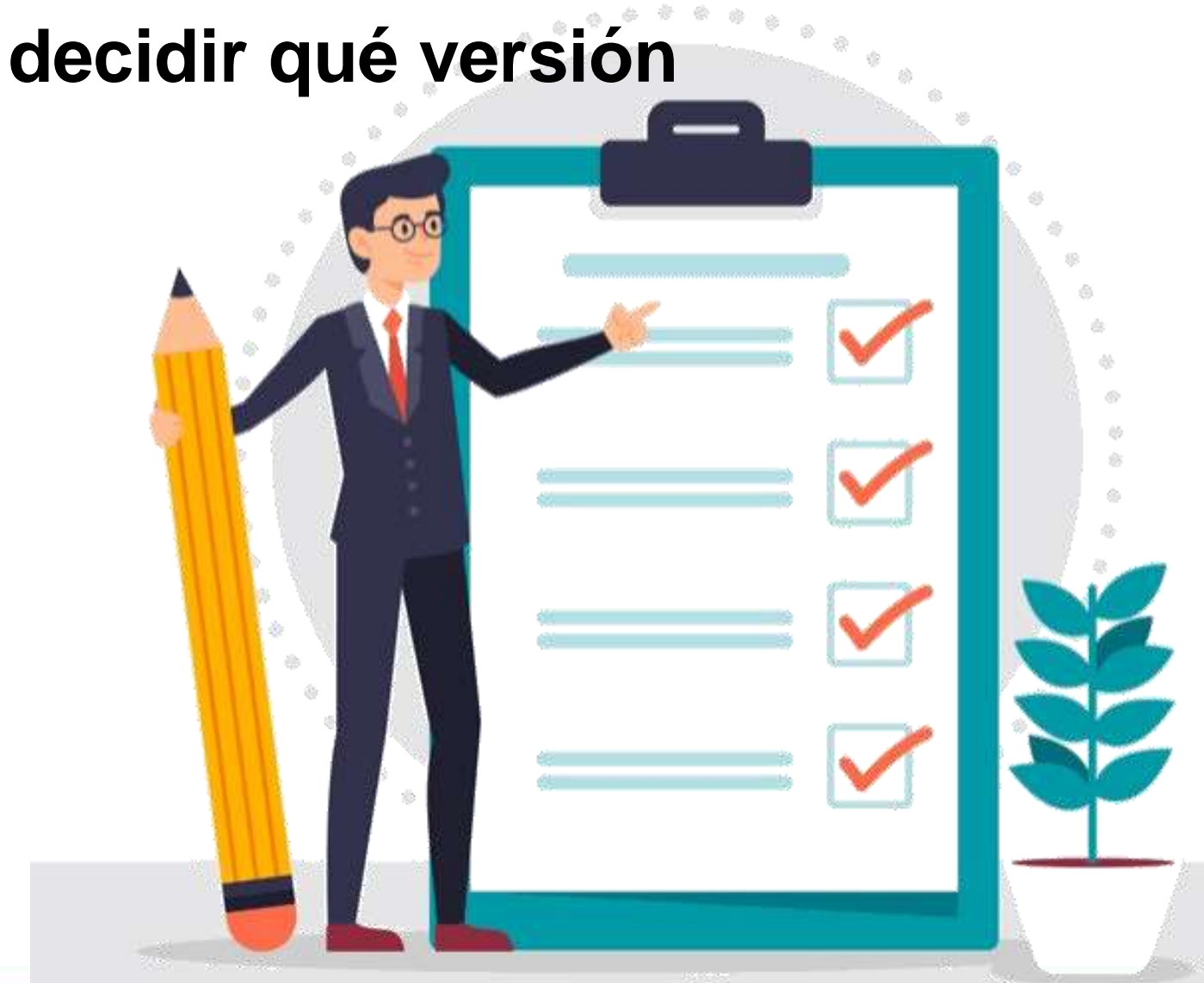
**Clonar un repositorio remoto a tu máquina local.**

**Enviar tus cambios al repositorio remoto y verificar**



# Cierre

1. ¿Qué es un conflicto de fusión en Git?
2. ¿Cuál es la diferencia entre git pull y git fetch?
3. ¿Qué significa el estado "conflict" en Git?
4. ¿Cómo puedes resolver un conflicto si no puedes decidir qué versión del código mantener?



# Bibliografía

Hernández Bejarno, Miguel. *Ciclo de vida de desarrollo ágil de software seguro*. Fundación Universitaria Los Libertadores. <https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36016>

Guillamón Morales, Alicia. (). *Manual desarrollo de elementos software para gestión de sistemas*.

Editorial CEP, S.L. <https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=34982>

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.  
Enlace: <https://git-scm.com/book/es/v2>

Poulton, N. (2017). *Docker Deep Dive*. Independently published.  
Enlace: <https://www.nigelpoulton.com/dvd/>





**Universidad  
Tecnológica  
del Perú**