



S.E.P. TECNOLÓGICO NACIONAL DE MÉXICO

# INSTITUTO TECNOLÓGICO DE TUXTEPEC

**MERCURIAL**

**NOMBRE DE LOS ALUMNOS:**

CARRILLO OLIVERA WILLIAM JONATHAN  
CRUZ VARGAS MIGUEL ANGEL

**NUMERO DE CONTROL:**

20350258  
20350258

**CARRERA:**

INGENIERIA EN SISTEMAS COMPUTACIONALES

**ASIGNATURA:**

SISTEMAS DE CONTROL DE VERSIONES

**DOCENTE:**

OLIVIA GUADALUPE LOPEZ RUIZ

**FECHA:**

18 de Febrero del 2024

# ÍNDICE DE CONTENIDO

## Contenido

INTRODUCCIÓN .....	3
Breve historia y contexto de Mercurial. ....	3
CONCEPTOS BÁSICOS DE MERCURIAL.....	4
Qué es y cómo funciona en Mercurial. ....	4
Commit (confirmación): explicación de los commits y su importancia. ....	4
Ramas y combinaciones: comprensión de las ramas en Mercurial y cómo se gestionan las fusiones. ....	4
Etiquetas: uso de etiquetas para marcar versiones importantes del proyecto. ....	5
Ventajas y desventajas de utilizar Mercurial. ....	5
INSTALACIÓN Y CONFIGURACIÓN.....	6
Cómo instalar Mercurial en Windows .....	6
FLUJO DE TRABAJO BÁSICO.....	8
Iniciar un nuevo repositorio.....	8
Clonar un repositorio existente. ....	8
Realizar cambios, confirmaciones y visualización del historial. ....	8
INTEGRACIÓN CON OTRAS HERRAMIENTAS.....	16
Integración con servicios de alojamiento de repositorios como Bitbucket o GitHub. ....	16
CASOS DE USO Y EJEMPLOS.....	17
Ejemplos prácticos de cómo Mercurial puede ayudar en el desarrollo de software. ....	17
Casos de estudio de empresas o proyectos que utilizan Mercurial en su flujo de trabajo.....	18
CONCLUSION .....	19

REFERENCIAS.....	20
------------------	----

# INTRODUCCIÓN

## **Breve historia y contexto de Mercurial.**

Mackall hizo pública la existencia de Mercurial el 19 de abril de 2005.<sup>4</sup>El estímulo que llevó a esto fue el anuncio de Bitmover, publicado anteriormente aquel mismo mes, informando que retirarían la versión gratuita de BitKeeper.

Se había estado usando BitKeeper debido a los requisitos de control de versiones del proyecto del núcleo Linux. Mackall decidió escribir un sistema de control distribuido de versiones como sustituto para usarlo con el núcleo Linux. Este proyecto comenzó aproximadamente al mismo tiempo que otro denominado git, iniciado por el propio Linus Torvalds con objetivos similares.

El proyecto Linux decidió usar Git en lugar de Mercurial. Sin embargo, muchos otros proyectos usan este último.

# CONCEPTOS BÁSICOS DE MERCURIAL

## **Qué es y cómo funciona en Mercurial.**

Mercurial es un sistema de control de versiones distribuido (DVCS) diseñado para gestionar el historial de cambios de proyectos de software. En Mercurial, cada usuario tiene una copia completa del repositorio en su propio sistema, lo que permite trabajar de forma independiente y sin conexión a un repositorio central.

El funcionamiento básico implica la creación de commits para guardar cambios en el repositorio local y luego compartir esos commits con otros usuarios a través de la clonación, tirando y empujando cambios a repositorios remotos.

## **Commit (confirmación): explicación de los commits y su importancia.**

Un commit, o confirmación, en Mercurial es una instantánea del estado de los archivos en tu proyecto en un momento dado. Cada commit tiene un mensaje asociado que describe los cambios realizados desde el último commit.

Los commits son fundamentales en Mercurial ya que proporcionan un historial completo de los cambios realizados en el proyecto. Esto facilita la colaboración, la depuración y la gestión de versiones.

## **Ramas y combinaciones: comprensión de las ramas en Mercurial y cómo se gestionan las fusiones.**

Las ramas en Mercurial son líneas de desarrollo independientes que divergen del flujo principal de trabajo. Permiten trabajar en características nuevas o experimentales sin afectar el código principal.

Las combinaciones son el proceso de fusionar cambios de una rama a otra. Mercurial utiliza algoritmos automáticos para fusionar cambios automáticamente cuando es posible, pero pueden surgir conflictos que requieren intervención manual.

## **Etiquetas: uso de etiquetas para marcar versiones importantes del proyecto.**

Las etiquetas en Mercurial son nombres simbólicos que se asignan a commits específicos para marcar versiones importantes del proyecto, como lanzamientos o hitos significativos.

Las etiquetas son útiles para identificar rápidamente versiones estables o importantes en el historial del proyecto y facilitar la gestión del código.

## **Ventajas y desventajas de utilizar Mercurial.**

Ventajas:

- **Distribución:** al ser un sistema de control de versiones distribuido, permite a los usuarios trabajar de forma independiente y sin conexión a un repositorio central.
- **Rendimiento:** Mercurial es conocido por su rendimiento rápido, especialmente en operaciones como clonación y combinaciones.
- **Simplicidad:** Mercurial tiene una curva de aprendizaje más suave que otros sistemas de control de versiones, lo que lo hace ideal para equipos y proyectos pequeños.

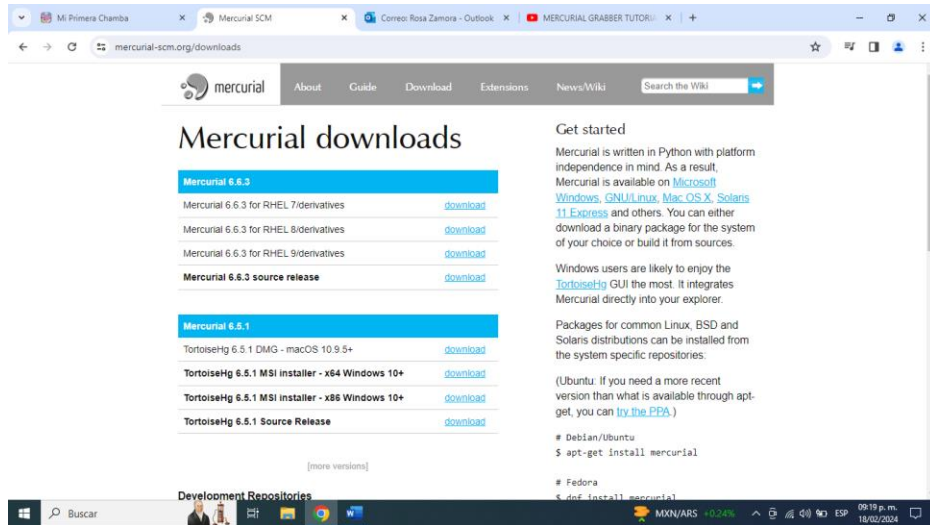
Desventajas:

- **Menos popularidad:** Mercurial es menos popular que Git, lo que puede resultar en menos soporte de la comunidad y herramientas de terceros.
- **Curva de aprendizaje:** Aunque más simple que Git en muchos aspectos, Mercurial aún puede tener una curva de aprendizaje para los usuarios nuevos en el control de versiones.
- **Menos integraciones:** Al ser menos popular, es posible que Mercurial tenga menos integraciones con otras herramientas y servicios de desarrollo.

# INSTALACIÓN Y CONFIGURACIÓN

## Cómo instalar Mercurial en Windows

Ir a la página oficial: <https://www.mercurial-scm.org/downloads>



Para poder usar mercurial t será necesario descargar **TortoiseHg** , lo cual es la versión de Mercurial para Windows y , MacOS

## Mercurial 6.6.3

Mercurial 6.6.3 for RHEL 7/derivatives [download](#)

Mercurial 6.6.3 for RHEL 8/derivatives [download](#)

Mercurial 6.6.3 for RHEL 9/derivatives [download](#)

**Mercurial 6.6.3 source release** [download](#)

## Mercurial 6.5.1

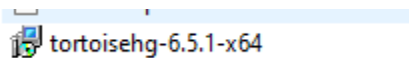
TortoiseHg 6.5.1 DMG - macOS 10.9.5+ [download](#)

**TortoiseHg 6.5.1 MSI installer - x64 Windows 10+** [download](#)

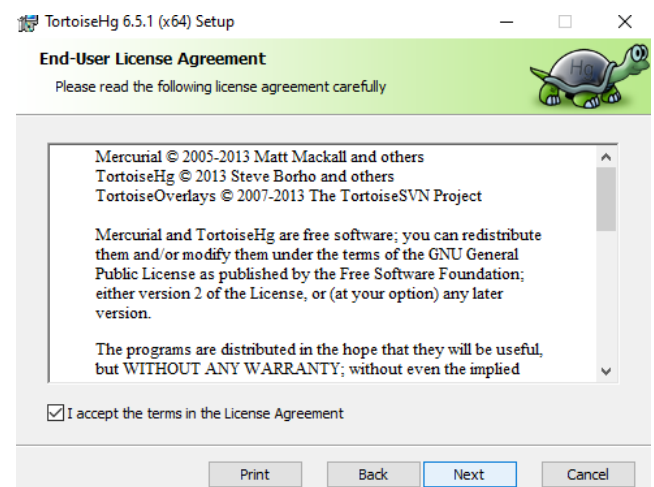
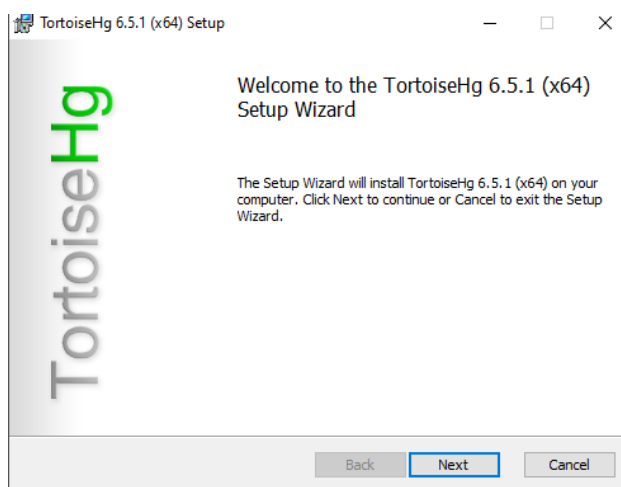
**TortoiseHg 6.5.1 MSI installer - x86 Windows 10+** [download](#)

**TortoiseHg 6.5.1 Source Release** [download](#)

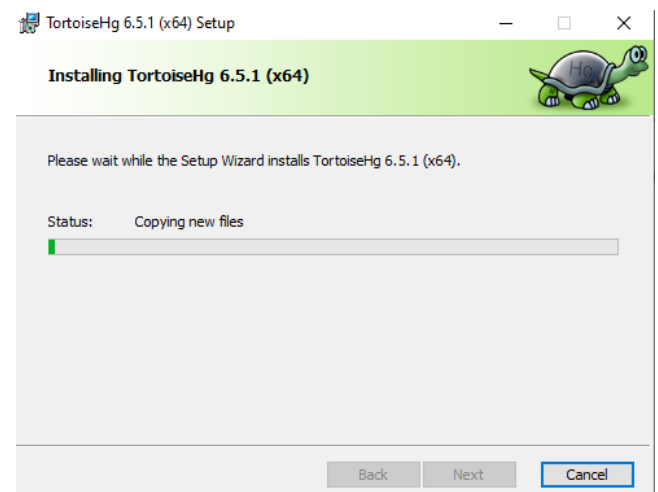
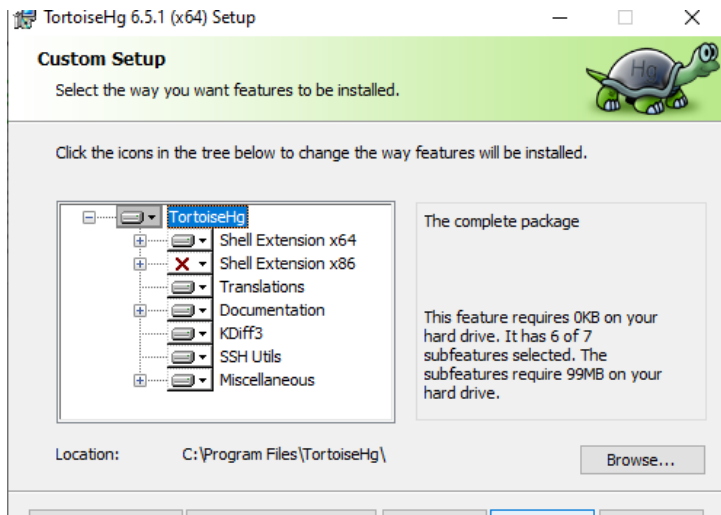
Una vez teniendo el instalador, lo que sigue es instalarlo



Afortunadamente la instalación de este software es muy fácil







## FLUJO DE TRABAJO BÁSICO

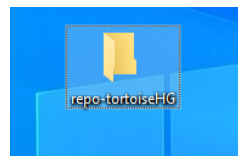
**Iniciar un nuevo repositorio.**

**Clonar un repositorio existente.**

**Realizar cambios, confirmaciones y visualización del historial.**

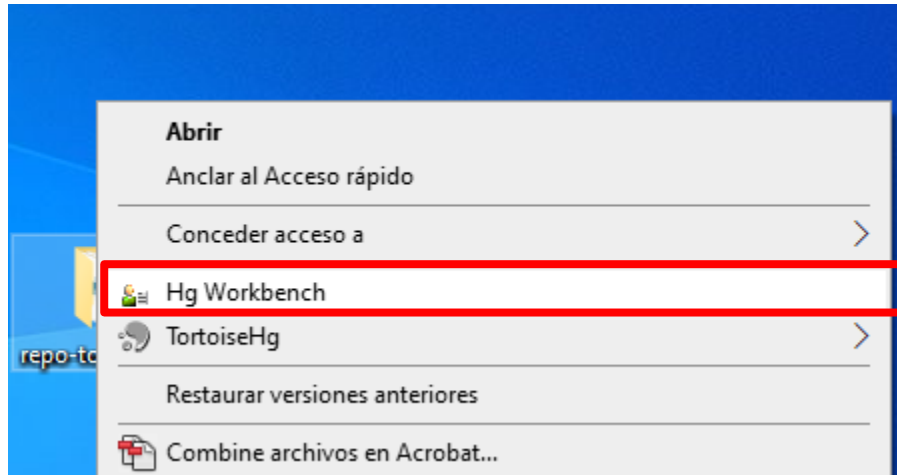
1.- iniciar un nuevo repositorio

Para comenzar un repositorio lo primero es crear una capeta o elegir una existente.



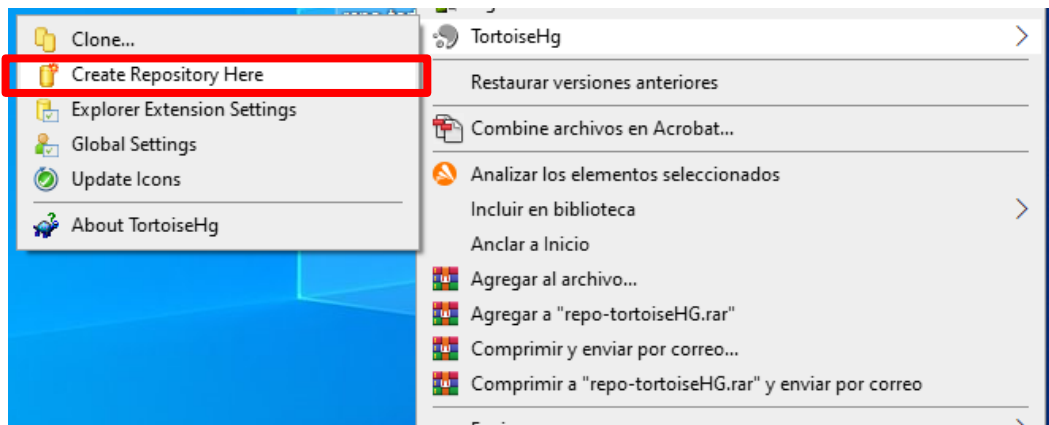
Se deben seguir los siguientes pasos para subir archivos:

Paso 1: dar clic derecho sobre el archivo o carpeta



2. elegimos la segunda opción

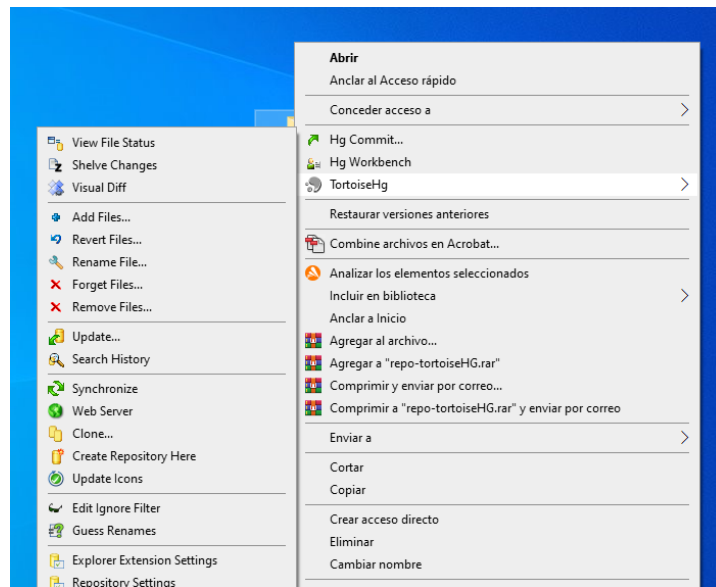
En la cual se desplegará una serie de opciones, clonar, crear...



Nos aparecerá una ventana como esta

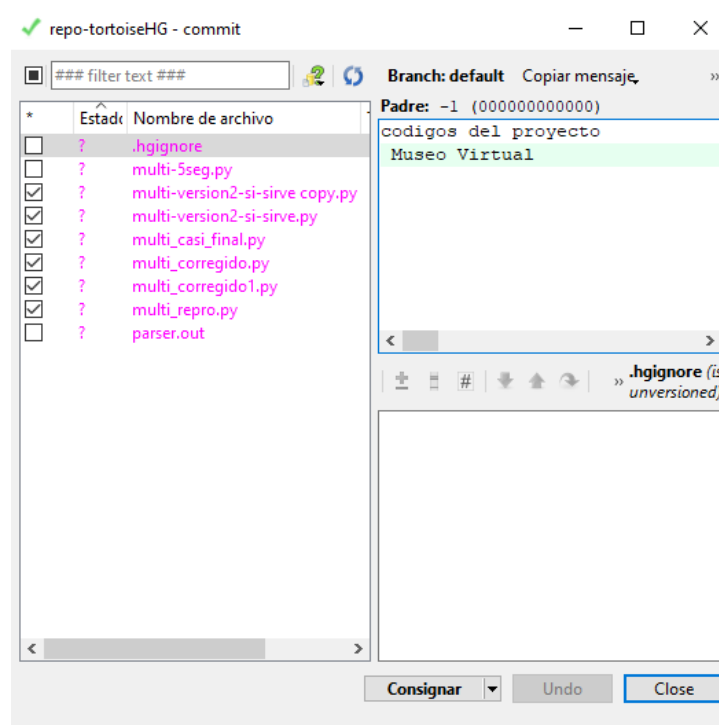


Para añadir archivos o hacer commit basta con darle clic derecho otra vez al archivo y tendremos muchas opciones

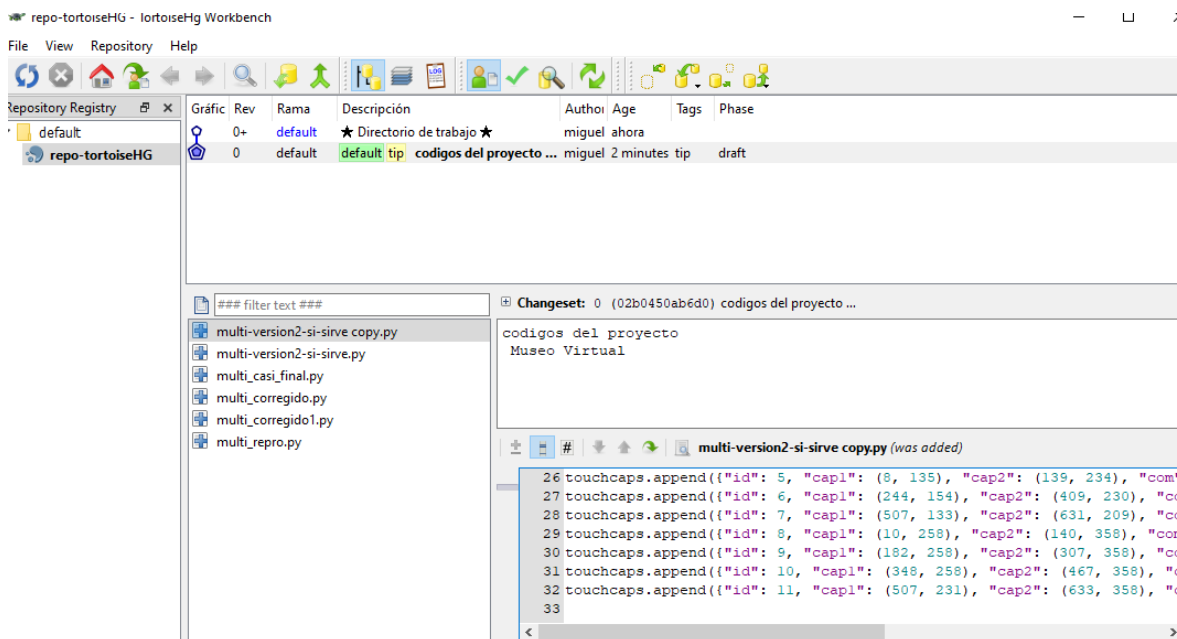
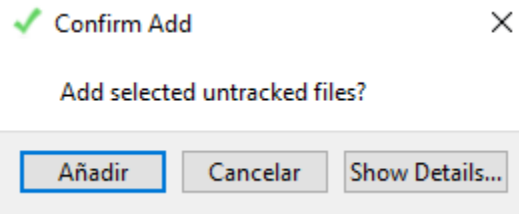
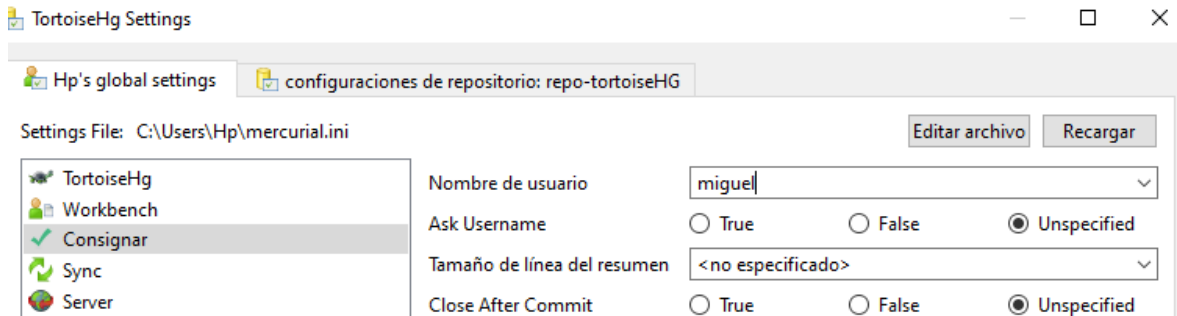


De lado izquierdo nos muestra los archivos por subir y del derecho podremos poner comentarios,

Al igual que en Git Hub está la opción de no subir algunos archivos

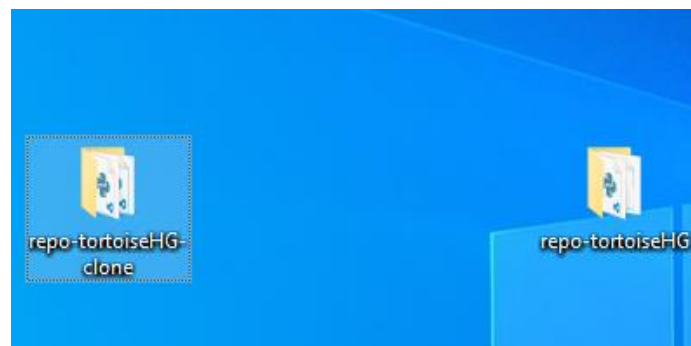
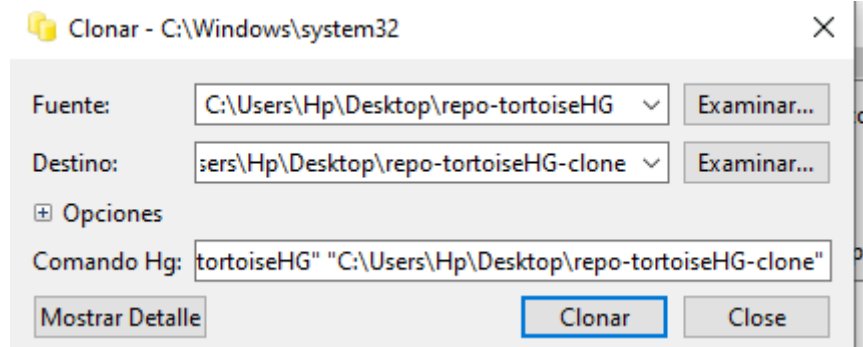
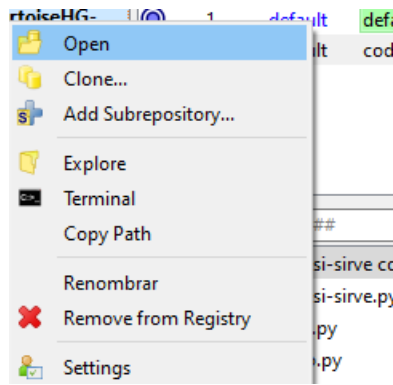


Si es la primera vez que ocupas este software al intentar subir los archivos te pedirá que definas un nombre de usuario o elegir la opción de no especificado



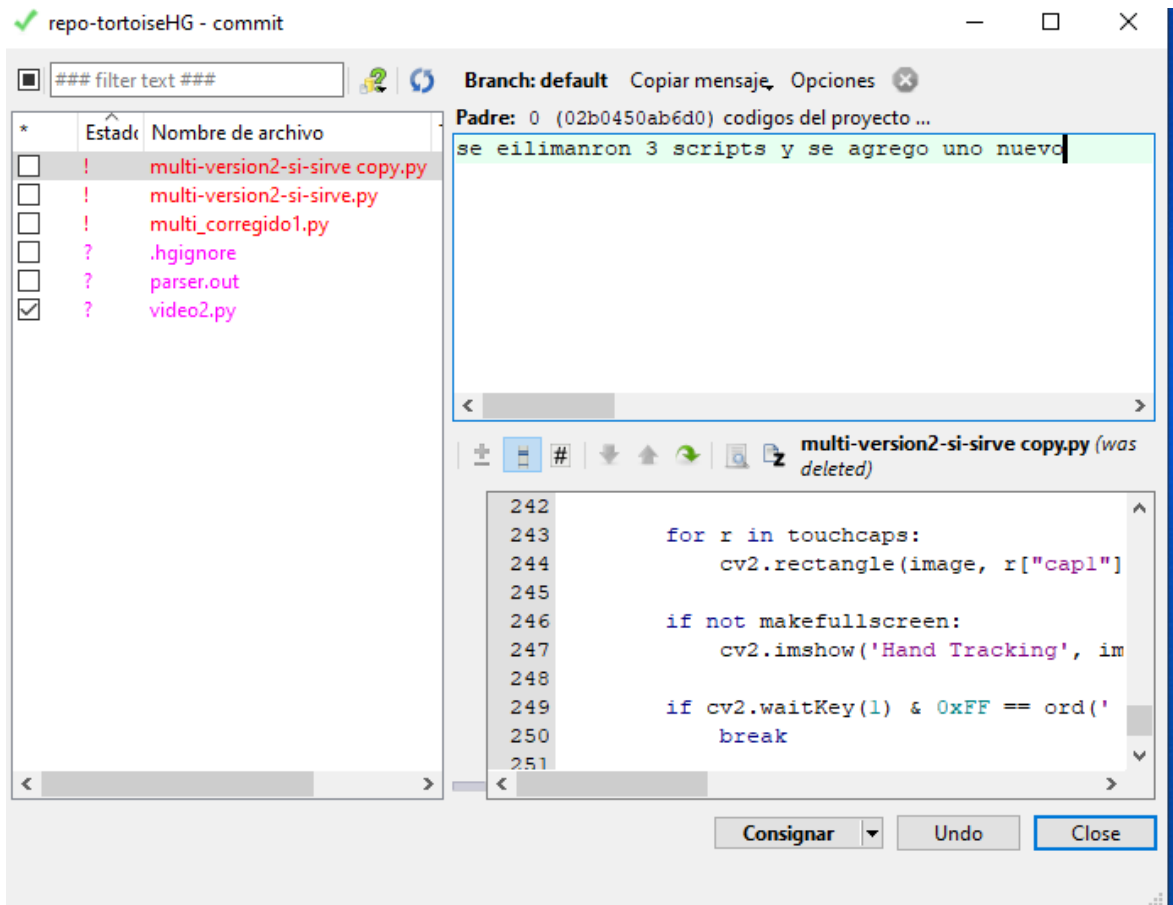
## 2.- clonar un repositorio existente

Para clonar un repositorio en TortoiseHG se puede hacer desde la UI o desde la carpeta, dando clic derecho y eligiendo esa opción

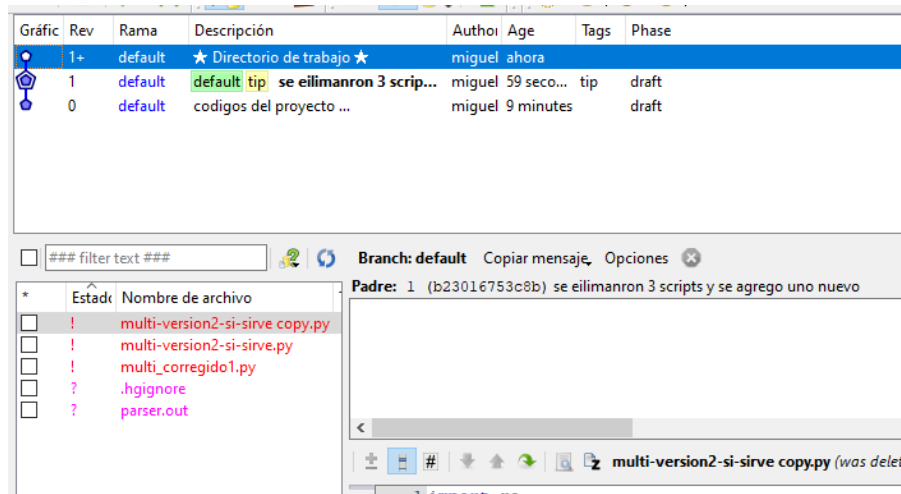


### 3.- Realizar cambios. Confirmaciones y visualización del historial

Si se borra algún archivo o se agrega y posteriormente se realiza un commit para actualizar el repositorio veremos que los archivos borrados se ven en color rojo y los que no están agregados en color rosado



Si abrimos la UI podremos ver el historial de actividades. En este caso estamos viendo el comentario de commit realizado y el nuevo script que se agrego



Para ver de manera gráfica el historial, entonces daremos clic sobre las ramas que tiene el hexágono, podremos ver los archivos borrados, los agregados y los que hay en general en el repositorio

	1+	default	★ Directorio de trabajo ★	miguel ahora			
	1	default	default tip se eilimanron 3 scrip...	miguel 59 seco...	tip	draft	
	0	default	codigos del proyecto ...	miguel 9 minutes		draft	

repo-tortoiseHG - TortoiseHg Workbench

File View Repository Help

Repository Registry

Gráfico	Rev	Rama	Descripción	Author	Age	Tags	Phase
	1+	default	★ Directorio de trabajo ★	miguel ahora			
	1	default	default tip se eilimanron 3 scrip...	miguel 59 seco...	tip	draft	
	0	default	codigos del proyecto ...	miguel 9 minutes		draft	

video2.py

Changeset: 1 (b23016753c0b) se eilimanron 3 scripts y se agrego uno nuevo

Usuario: miguel

Fecha: 2024-02-18 22:24:27 -0600 (67 seconds)

Padre: 0 (02b0450ab6d0) codigos del proyecto ...

Etiquetas: tip

se eilimanron 3 scripts y se agrego uno nuevo

video2.py (was added)

```

1#!/usr/bin/env python
2# -*- coding: utf-8 -*-
3
4import sys

```



# INTEGRACIÓN CON OTRAS HERRAMIENTAS

## Integración con servicios de alojamiento de repositorios como Bitbucket o GitHub.

### Requisitos previos:

- Asegúrate de tener instalados Mercurial, Git y Git LFS en tu sistema (puedes verificarlo ejecutando:
  - `hg --version`, `git --version` y `git lfs --version`).
- También necesitarás Python con el administrador de paquetes `pip`.
- Descarga la última versión de `fast-export` y extrae el archivo.

### Clonar el repositorio de Mercurial:

Ejecuta el siguiente comando para clonar el código fuente de Mercurial en un directorio local:

```
hg clone [^1^][1] mercurial-repo
```

### Crear un repositorio de Git nuevo:

Crea un directorio para tu nuevo repositorio de Git (por ejemplo, `mercurial-git`) y accede a él:

```
mkdir mercurial-git && cd mercurial-git
```

```
git init
```

### Configurar el repositorio de Git:

Para que el repositorio de Git controle el caso de los nombres de archivo de la misma manera que Mercurial, ejecuta:

```
git config core.ignoreCase false
```

### **Mapeo de confirmadores:**

Obtén una lista de los confirmadores del proyecto de Mercurial y almacénala en un archivo llamado `committers.txt`. Puedes hacerlo con el siguiente script:

```
hg log --template "{author}\n" | sort | uniq > committers.txt
```

## **CASOS DE USO Y EJEMPLOS**

### **Ejemplos prácticos de cómo Mercurial puede ayudar en el desarrollo de software.**

#### **Trabajo sin conexión:**

En proyectos donde los desarrolladores se encuentran incomunicados o tienen acceso limitado a Internet, Mercurial permite que los cambios se realicen localmente y luego se sincronicen cuando haya conectividad.

#### **Flexibilidad en la estructura del repositorio:**

Mercurial permite clonar repositorios completos, lo que significa que los clientes pueden actuar como servidores y viceversa. Esto es útil en situaciones donde no hay un servidor central disponible.

#### **Comandos push y pull:**

Mercurial introduce los comandos push y pull. Estos permiten a los usuarios subir cambios locales a un servidor remoto y actualizarse con los cambios realizados por otros usuarios respectivamente.

#### **Gestión de ramas y fusiones:**

Mercurial ofrece capacidades avanzadas de ramificación e integración. Puedes crear ramas para trabajar en características específicas y luego fusionarlas de manera limpia.

**Rendimiento y escalabilidad:**

Mercurial se enfoca en un gran rendimiento y es escalable incluso para proyectos grandes.

**Interfaz web integrada:**

Mercurial incluye una interfaz web que facilita la visualización del historial de cambios y la colaboración entre desarrolladores.

**Casos de estudio de empresas o proyectos que utilizan Mercurial en su flujo de trabajo.**

**Python (antes de 2017):** El lenguaje de programación Python utilizó Mercurial como su sistema de control de versiones antes de migrar a Git.



**OpenOffice.org:** Este proyecto de suite de oficina también empleó Mercurial en su flujo de trabajo.

**Google Code Project Hosting:** Google Code permitía la creación de repositorios Mercurial además de Subversion.



## CONCLUSION

Mercurial es un sistema de control de versiones distribuido que permite a los usuarios gestionar el historial de cambios de sus proyectos de software de manera eficiente y colaborativa. Durante la exposición, hemos explorado los conceptos básicos de Mercurial, incluyendo la creación de commits, el manejo de ramas y fusiones, el uso de etiquetas y las ventajas y desventajas de su utilización. También hemos aprendido cómo instalar Mercurial en sistemas Windows. Además, hemos explorado cómo integrar Mercurial con servicios de alojamiento de repositorios como Bitbucket y GitHub. Mercurial ofrece una solución sólida para el control de versiones en proyectos de software, con su enfoque distribuido, su rendimiento rápido y su relativa facilidad de uso. Si bien existen otras opciones de control de versiones, Mercurial sigue siendo una elección viable para muchos equipos y proyectos, especialmente aquellos que prefieren un enfoque más simple y directo.

## REFERENCIAS


*Importación de un repositorio de Mercurial.* (n.d.).

Lean, P. (2015, September 23). *Top 10 de compañías Lean Manufacturing.*

Progressa Lean. <https://www.progressalean.com/top-10-de-companias-lean-manufacturing/>

*Mercurial (sistema de control de versiones).* (n.d.). Ecured.cu. Retrieved February 18, 2024, from

[https://www.ecured.cu/Mercurial\\_%28sistema\\_de\\_control\\_de\\_versiones%29](https://www.ecured.cu/Mercurial_%28sistema_de_control_de_versiones%29)

Programacion en Castellano, S. L. (n.d.). *Mercurial, un software para la gesti* *n de versiones.* Retrieved February 18, 2024, from

[https://programacion.net/noticia/mercurial-\\_un\\_software\\_para\\_la\\_gestion\\_de\\_versiones\\_1778](https://programacion.net/noticia/mercurial-_un_software_para_la_gestion_de_versiones_1778)

Rodriguez, A. (2014, April 30). *Qué es un sistema de control de versiones y por qué es tan importante.* Hipertextual.

<https://hipertextual.com/2014/04/sistema-control-versiones>

Román, E. (2023, February 17). *Mejores Prácticas de Desarrollo de*

*Software.* Innevo.com. <https://blog.innevo.com/mejores-practicas-desarrollo-software>

Wikipedia contributors. (n.d.). *Mercurial*. Wikipedia, The Free Encyclopedia.

<https://es.wikipedia.org/w/index.php?title=Mercurial&oldid=157684665>

Zitelia. (2024, January 9). *Desarrollo software de empresas: claves y ejemplos de uso*. Zitelia - Diseño web, aplicaciones móviles y control de acceso; Zitelia Soluciones Tecnológicas. <https://www.zitelia.com/desarrollo-software-de-empresas-claves-y-ejemplos-de-uso/>

(N.d.). Theirstack.com. Retrieved February 18, 2024, from

<https://theirstack.com/es/category/mercurial-tools>