Guía Completa: Git y GitHub para Proyectos LaTeX

Toribio de J. Arrieta F.

17 de octubre de 2025

Índice general

1 Introducción

1.1 ¿Qué es Git?

Git es un sistema de control de versiones distribuido que permite:

- Guardar el historial completo de cambios de tus archivos
- Volver a versiones anteriores cuando lo necesites
- Trabajar sin internet (todo se guarda localmente)
- Colaborar con otras personas en el mismo proyecto

1.2 ¿Qué es GitHub?

GitHub es una plataforma en la nube que:

- Aloja tus repositorios Git en internet
- Sirve como respaldo de tu trabajo
- Permite acceder a tus proyectos desde cualquier computadora
- Facilita la colaboración y el código abierto

1.3 ¿Por qué usar Git + GitHub para LaTeX?

Beneficios para proyectos LaTeX

1. **Nunca más perderás trabajo**: Cada versión queda guardada
2. **Recupera versiones antiguas**: Si borras algo por error, puedes recuperarlo
3. **Respaldo en la nube**: Si tu computadora se daña, tus archivos están seguros
4. **Trabajo offline**: Git funciona sin internet, solo necesitas conexión para sincronizar
5. **Historial completo**: Ves qué cambiaste, cuándo y por qué

2 Configuración Inicial

2.1 Instalación de GitHub CLI

El GitHub CLI (`gh`) es la herramienta oficial para interactuar con GitHub desde la terminal.

```
# Instalar con Homebrew
brew install gh
```

2.2 Autenticación con GitHub

```
# Iniciar autenticación
gh auth login
```

Pasos que verás:

1. Te dará un código de un solo uso (ej: `BC0B-57BD`)

2. Te pedirá abrir: https://github.com/login/device
3. Pega el código en la página web
4. Autoriza GitHub CLI
5. ¡Listo! Ya estás conectado

2.3 Verificar autenticación

```
# Ver estado de autenticación
gh auth status
```

Deberías ver algo como:

```
github.com
   Logged in to github.com account toribio99 (keyring)
  - Active account: true
```

3 Conceptos Fundamentales

3.1 Vocabulario básico

Repositorio (repo) Carpeta que contiene tu proyecto y todo su historial de cambios

Commit "Fotografía" o punto de guardado de tu proyecto en un momento específico

Staging Area Área de preparación donde pones archivos antes de hacer commit

Push Subir tus commits a GitHub (requiere internet)

Pull Descargar cambios de GitHub a tu computadora

Clone Copiar un repositorio de GitHub a tu computadora

Remote Conexión entre tu repositorio local y GitHub

3.2 El flujo de trabajo básico

Ciclo típico de trabajo

1. **Editas** tus archivos .tex normalmente
2. **git add** — Agregas los cambios al área de preparación
3. **git commit** — Guardas los cambios con un mensaje descriptivo
4. **git push** — Subes a GitHub (cuando tengas internet)

4 Comandos Esenciales

4.1 Ver el estado actual

```
# Ver qué archivos cambiaron
git status
```

Posibles salidas:

- `modified: archivo.tex` — Archivo modificado
- `Untracked files` — Archivos nuevos que Git no rastrea
- `Changes to be committed` — Archivos listos para guardar
- `nothing to commit` — Todo está guardado

4.2 Agregar archivos al área de preparación

```
# Agregar UN archivo específico
git add archivo.tex

# Agregar TODOS los archivos modificados
git add .

# Agregar múltiples archivos específicos
git add archivo1.tex archivo2.tex
```

4.3 Crear un commit (guardar cambios)

```
# Commit con mensaje descriptivo
git commit -m "Descripción breve del cambio"
```

Mensajes de commit efectivos

Buenos ejemplos:

- `"Agregada sección sobre funciones trigonométricas"`
- `"Corregido error en la ecuación de la línea 45"`
- `"Actualizada bibliografía con 3 nuevas referencias"`

Malos ejemplos:

- `"cambios"` (muy vago)
- `"asdf"` (sin sentido)
- `"arreglado"` (¿qué arreglaste?)

4.4 Subir cambios a GitHub

```
# Subir todos los commits al repositorio remoto
git push
```

Nota importante

`git push` requiere conexión a internet. Puedes hacer múltiples commits sin internet y luego hacer un solo `push` cuando te conectes.

4.5 Ver el historial de cambios

```
# Ver historial completo
git log

# Ver historial compacto (una línea por commit)
git log --oneline

# Ver últimos 5 commits
git log --oneline -5
```

Ejemplo de salida:

```
e72c314 Agregada nota final al taller
aa37424 Versión inicial del proyecto
```

4.6 Ver diferencias entre versiones

```
# Ver qué cambió en archivos modificados (no staged)
git diff

# Ver cambios en archivos en staging area
git diff --staged

# Ver cambios en un archivo específico
git diff archivo.tex
```

5 Trabajar con Repositorios de GitHub

5.1 Clonar un repositorio desde GitHub

```
# Clonar usando GitHub CLI
gh repo clone usuario/nombre-repositorio

# Ejemplo:
gh repo clone toribio99/LaTeX-Varios
```

Esto crea una carpeta con todo el proyecto y su historial completo.

5.2 Crear un nuevo repositorio en GitHub

```
# Crear repositorio público
gh repo create nombre-repositorio --public \
  --description "Descripción del proyecto"

# Ejemplo:
gh repo create MiTesisLatex --public \
  --description "Tesis de maestría en LaTeX"
```

5.3 Ver repositorios remotos

```
# Ver a dónde está conectado tu repositorio
git remote -v
```

Salida típica:

```
origin  https://github.com/toribio99/LaTeX-Varios.git (fetch)
origin  https://github.com/toribio99/LaTeX-Varios.git (push)
```

6 Flujo de Trabajo Completo: Ejemplo Práctico

6.1 Escenario: Editaste un archivo

Supongamos que editaste el archivo:
`TallerTrianguloRectangulo.tex`

Ubicación del proyecto:
`~/Documents/LaTeX-GitHub/LaTeX-Varios/`

6.2 Paso 1: Navega al directorio

```
cd ~/Documents/LaTeX-GitHub/LaTeX-Varios
```

6.3 Paso 2: Verifica qué cambió

```
git status
```

Salida:

```
Changes not staged for commit:
  modified:   Clases De Sheyra/Geometría analíca/TallerTrianguloRectangulo.
     tex
```

6.4 Paso 3: Agrega el archivo modificado

```
# Opción 1: Agregar ese archivo específico
git add "Clases De Sheyra/Geometría analíca/TallerTrianguloRectangulo.tex"

# Opción 2: Agregar todos los cambios
git add .
```

6.5 Paso 4: Verifica el estado nuevamente

```
git status
```

Salida:

```
Changes to be committed:
  modified:   Clases De Sheyra/Geometría analíca/TallerTrianguloRectangulo.
     tex
```

6.6 Paso 5: Crea el commit

```
git commit -m "Agregada nota final al taller de geometría analítica"
```

Salida:

```
[main e72c314] Agregada nota final al taller de geometría analítica
 1 file changed, 8 insertions(+), 4 deletions(-)
```

6.7 Paso 6: Sube a GitHub

```
git push
```

Salida:

```
To https://github.com/toribio99/LaTeX-Varios.git
   aa37424..e72c314  main -> main
```

¡Listo!

Tu cambio ahora está guardado en:

- Tu computadora (Mac)
- GitHub (en la nube)

Puedes verlo en: https://github.com/toribio99/LaTeX-Varios

7 Trabajar Sin Internet

7.1 Git funciona 100% local

Lo que SÍ puedes hacer sin internet:

- Editar tus archivos .tex
- Compilar con LaTeX
- `git add` (agregar archivos)
- `git commit` (guardar cambios)
- `git status` (ver estado)
- `git log` (ver historial)
- `git diff` (ver diferencias)

Lo que NO puedes hacer sin internet:

- `git push` (subir a GitHub)
- `git pull` (descargar de GitHub)
- `gh repo create` (crear repositorio en GitHub)
- Clonar repositorios

7.2 Flujo de trabajo sin internet

Escenario: Trabajas offline

Lunes (sin internet):

```
# Editas archivo1.tex
git add archivo1.tex
git commit -m "Agregué introducción"
```

```
# Editas archivo2.tex
git add archivo2.tex
git commit -m "Completé capítulo 2"
```

Martes (con internet):

```
# Subes TODOS los commits que hiciste offline
git push
```

Resultado: Los 2 commits se suben a GitHub de una vez.

8 Recuperar Versiones Anteriores

8.1 Ver historial

```
# Ver lista de commits
git log --oneline
```

Ejemplo de salida:

```
e72c314 Agregada nota final al taller
aa37424 Versión inicial del proyecto
9d82076 Primer commit
```

8.2 Recuperar un archivo específico

```
# Recuperar archivo de un commit específico
git checkout aa37424 archivo.tex
```

¡Atención!

Este comando **sobrescribe** el archivo actual con la versión antigua. Asegúrate de hacer commit de tus cambios actuales primero, o los perderás.

8.3 Ver un archivo sin modificarlo

```
# Ver contenido de archivo en un commit específico
git show aa37424:ruta/al/archivo.tex
```

Esto solo **muestra** el contenido, no modifica tu archivo.

9 Archivo .gitignore para LaTeX

9.1 ¿Qué es .gitignore?

Un archivo especial que le dice a Git qué archivos **NO** rastrear.

9.2 ¿Por qué necesitamos .gitignore en LaTeX?

LaTeX genera muchos archivos auxiliares (`.aux`, `.log`, `.synctex.gz`, etc.) que:

- Cambian cada vez que compilas
- No son necesarios para el control de versiones
- Hacen que Git muestre cambios innecesarios

9.3 Contenido típico de .gitignore para LaTeX

```
# Archivos auxiliares de LaTeX
*.aux
*.log
*.out
*.toc
*.lof
*.lot
*.fls
*.fdb_latexmk
*.synctex.gz
*.bbl
*.blg
*.idx
*.ilg
*.ind
*.listing

# PDFs (opcional - descomenta si quieres versionar PDFs)
# *.pdf

# Archivos de sistema
.DS_Store
*~
*.swp
__MACOSX/

# Otros
general.idx
*.zip
```

9.4 Crear .gitignore

```
# Crear archivo .gitignore en la raíz del proyecto
cd ~/Documents/LaTeX-GitHub/MiProyecto
nano .gitignore
# (pega el contenido de arriba y guarda)
```

10 Solución de Problemas Comunes

10.1 Olvidé hacer commit antes de editar

Problema: Editaste un archivo pero quieres volver a la versión anterior.
Solución:

```
# Descartar cambios no guardados
git restore archivo.tex
```

10.2 Hice commit pero no quería hacerlo

Problema: Hiciste commit de cambios que no querías guardar.
Solución (antes de push):

```
# Deshacer el último commit (mantiene cambios)
git reset --soft HEAD~1

# Deshacer el último commit (descarta cambios)
git reset --hard HEAD~1
```

¡Cuidado con –hard!

`git reset --hard` **elimina permanentemente** tus cambios. Úsalo solo si estás seguro.

10.3 Error al hacer push

Problema: `git push` falla con error.
Posibles causas y soluciones:

1. **Sin conexión a internet**

```
# Espera a tener internet y vuelve a intentar
git push
```

2. **Cambios en GitHub que no tienes localmente**

```
# Descargar cambios primero
git pull
# Luego subir
git push
```

10.4 No sé en qué estado está mi repositorio

Solución:

```
# Ver estado completo
git status

# Ver últimos commits
git log --oneline -5

# Ver diferencias no guardadas
git diff
```

11 Comandos de Referencia Rápida

12 Flujo de Trabajo Recomendado

12.1 Rutina diaria

Recomendación: Commits frecuentes

Al empezar a trabajar:

```
cd ~/Documents/LaTeX-GitHub/MiProyecto
git status  # Ver si hay cambios pendientes
git pull    # Actualizar (si trabajas desde varias computadoras)
```

Durante el trabajo:

Comando	Descripción
`git status`	Ver estado actual del repositorio
`git add .`	Agregar todos los cambios
`git add archivo.tex`	Agregar archivo específico
`git commit -m "msg"`	Guardar cambios con mensaje
`git push`	Subir commits a GitHub
`git pull`	Descargar cambios de GitHub
`git log --oneline`	Ver historial compacto
`git diff`	Ver cambios no guardados
`git restore archivo`	Descartar cambios no guardados
`gh auth status`	Ver estado de autenticación
`gh repo create`	Crear repositorio en GitHub
`gh repo clone`	Clonar repositorio

Tabla 1: Comandos Git y GitHub más usados

- Haz commits cada vez que completes una sección importante
- No esperes a que todo esté "perfecto"
- Un commit por cada cambio lógico

Al terminar la sesión:

```
git add .
git commit -m "Descripción de lo que hiciste hoy"
git push  # Si tienes internet
```

12.2 Frecuencia de commits

Buenas prácticas:

- **Muy frecuente**: Cada cambio significativo (ej: completar una sección)
- **Moderado**: Al final de cada sesión de trabajo
- **Mínimo**: Al menos una vez al día si trabajaste

No recomendado:

- Una vez por semana (demasiado tiempo entre commits)
- Solo cuando terminas el proyecto completo
- Cada pequeño cambio de una palabra

13 Organización de Repositorios LaTeX

13.1 Estructura recomendada

Para proyectos LaTeX grandes, organiza por temas:

```
LaTeX-GitHub/
 LaTeX-Varios/              # Trabajos misceláneos
 LaTeX-Libros-Terminados/   # Libros completos
 LaTeX-Practicas/           # Ejercicios y prácticas
 LaTeX-Fisica/              # Proyectos de física
 LaTeX-Matematicas/         # Proyectos de matemáticas
```

13.2 Cada repositorio debe contener

- `.gitignore` — Lista de archivos a ignorar
- Archivos `.tex` — Tus documentos fuente
- Carpetas de figuras/imágenes
- `README.md` (opcional) — Descripción del proyecto

14 Conclusión

14.1 Beneficios de usar Git + GitHub

¡Nunca más perderás trabajo!

Con Git y GitHub:

- Tu trabajo está seguro en la nube
- Puedes recuperar cualquier versión anterior
- Trabajas con confianza sabiendo que todo está respaldado
- Colaboras fácilmente con otras personas
- Accedes a tus proyectos desde cualquier computadora

14.2 Los 3 comandos que usarás el 90% del tiempo

```
git add .
git commit -m "Descripción del cambio"
git push
```

14.3 Recursos adicionales

- Documentación oficial de Git: https://git-scm.com/doc
- GitHub CLI: https://cli.github.com
- Tu perfil en GitHub: https://github.com/toribio99

¡Feliz versionado!

Documento creado el 17 de octubre de 2025

Toribio de J. Arrieta F.