Documentation LaTeX + GitHub + Overleaf

Sebastián Palavecino

29 de septiembre de 2025

Índice

1. Introducción

Esta documentación tiene como objetivo centralizar, en un único lugar, toda la información necesaria para crear, mantener y compartir el material del proyecto en LaTeX. Está pensada para un inicio rápido durante la pasantía: compilar el PDF en la computadora, versionar los cambios en GitHub y (si corresponde) editar también en Overleaf.

¿Qué vas a encontrar?

- Una estructura mínima de archivos (`main.tex`, `chapters/`, `figures/`, `bib/`).
- Pasos simples para compilar en Windows usando `latexmk`.
- Un flujo básico de trabajo con GitHub: ramas, commits, Pull Requests y Releases.

- Dos formas de sincronizar con Overleaf (URL de Git o proyecto creado desde GitHub).

¿Por qué así? Porque mantener la documentación en texto plano (LaTeX) y con control de versiones (GitHub) permite:

- Historial claro de cambios (quién, cuándo y qué).
- Trabajo en equipo sin pisarse (ramas y revisiones).
- Reproducibilidad: cualquiera puede compilar el mismo PDF con los mismos pasos.

A quién va dirigida. A miembros del equipo que necesiten consultar o actualizar la documentación, aun con experiencia limitada en LaTeX y GitHub.

Cómo usar este documento. Leé la sección de GitHub para entender el flujo de trabajo, seguí los pasos de *LaTeX en local* para generar `main.pdf` y, si editás online, revisá *Sincronizar con Overleaf*. Cuando cierres una entrega, publicá un Release con el PDF.

2. Alcance

- Explicar cómo trabajar con este repositorio (estructura mínima).
- Pasos simples para compilar LaTeX en Windows.
- Flujo básico con GitHub: ramas, commits, Pull Requests y Releases.
- Dos formas de usar Overleaf: URL de Git o crear proyecto desde GitHub.

Esta primera versión no cubre temas avanzados de LaTeX ni automatizaciones de CI.

3. GitHub: qué es y cómo lo usamos

¿Qué es GitHub?

GitHub es un sitio web donde guardamos el contenido del proyecto (en este caso, la documentación en LaTeX) de forma segura y con historial. Permite trabajar en equipo: cada cambio queda registrado, se puede revisar, comentar y volver atrás si hace falta.

¿Para qué lo usamos en este proyecto?

- Tener un **lugar único** con la documentación.
- **Guardar el historial** de cambios (quién, cuándo y qué cambió).
- **Colaborar** sin pisarnos: cada persona trabaja en su rama y luego propone el cambio.
- **Publicar versiones** con el PDF final (Releases).

Conceptos básicos (sin vueltas)

- **Repositorio (repo):** la carpeta del proyecto en GitHub con todos los archivos.
- **Commit:** un paquete de cambios con un mensaje corto (ej.: `docs: agrega alcance`).
- **Rama (branch):** una línea de trabajo paralela para no romper la principal.
- **Pull Request (PR):** una propuesta de cambio desde una rama hacia la principal para revisar y aprobar.
- **Issue:** una tarea o problema por resolver; sirve para anotar pendientes y asignar trabajo.
- **Release:** una versión publicada del proyecto (acá adjuntamos el PDF).

Estructura mínima del repo (lo que vas a ver)

- `main.tex`: documento principal que incluye capítulos.
- `chapters/`: capítulos sueltos (uno por tema).
- `figures/`: imágenes (si hacen falta).
- `bib/references.bib`: bibliografía (opcional al inicio).
- `README.md`: cómo compilar y cómo sincronizar con Overleaf.

Flujo de trabajo recomendado (paso a paso)

1. **Traer el repo** a tu PC (clonar) y abrirlo en VS Code.
2. **Crear una rama** por cada cambio (ej.: `docs/intro`).
3. **Editar** los archivos necesarios (capítulos, figuras).
4. **Hacer commits** chicos con mensajes claros.
5. **Subir la rama** a GitHub y abrir un **Pull Request**.
6. **Revisar** (aunque seas vos mismo) y hacer **merge** a `main`.
7. **Publicar un Release** cuando quieras "congelar" una versión y adjuntar el `main.pdf`.

Buenas prácticas

- Un cambio concreto = un commit con mensaje claro.
- Usar ramas con nombres simples: `docs/<tema>` o `fix/<detalle>`.
- No subir archivos temporales ni PDFs al repo (el PDF va en Releases).
- Mantener el `README.md` corto pero actualizado.

Errores comunes y cómo salir rápido

- **"No me aparecen mis cambios en GitHub"**: falta *push*. Hacer `git push origin <rama>`.
- **"No veo los cambios del otro"**: traer lo último con `git pull origin main`.
- **"Me confundí de rama"**: guardar, cambiar de rama con VS Code (o `git checkout`), y volver a aplicar el cambio.
- **Conflicto al hacer merge**: abrir el archivo marcado, elegir qué parte dejar, guardar y hacer commit del merge.

Publicar una versión con el PDF

1. Compilar `main.pdf` en tu PC (ver sección *LaTeX en local*).
2. En GitHub: *Releases* → *New release*.
3. Tag (por ejemplo): `v0.1.0`. Adjuntar `main.pdf` y publicar.

4. LaTeX en local

Objetivo

Generar el PDF (`main.pdf`) en tu computadora y dejar listo el flujo para trabajar junto con Overleaf si hace falta.

Requisitos (Windows)

- MiKTeX instalado (distribución LaTeX).
- **Opción A (simple):** usar `pdflatex`.
- **Opción B (cómoda):** usar `latexmk` (requiere tener Perl instalado).

Compilar con `pdflatex` (opción simple)

Desde la carpeta del repositorio:

```
pdflatex -interaction=nonstopmode -file-line-error main.tex
pdflatex -interaction=nonstopmode -file-line-error main.tex
```

(Se corre dos veces para actualizar índice). El PDF queda como `main.pdf`.

Compilar con `latexmk` (opción cómoda)

Si falla por "falta Perl", instalar Strawberry Perl y reabrir la terminal. Luego:

```
latexmk -pdf -interaction=nonstopmode -file-line-error main.tex
```

Para limpiar archivos temporales:

```
latexmk -c
```

Estructura mínima esperada

```
main.tex
chapters/01-introduccion.tex
chapters/02-alcance.tex
chapters/03-github.tex
chapters/04-latex-local.tex
chapters/05-overleaf.tex
```

Errores comunes (y solución rápida)

- **No se reconoce `pdflatex`**: MiKTeX no está en PATH. Reinstalar/abrir MiKTeX Console y reiniciar la terminal.
- **Faltan paquetes** (ventana de MiKTeX): aceptar instalar paquetes automáticamente.
- **`latexmk` falla**: instalar Perl y volver a correr (o usar `pdflatex` dos veces).

Checklist

- Compilar sin errores y abrir `main.pdf`.
- Hacer commit + push de los *.tex* (no subir archivos temporales).
- (Opcional) Publicar el PDF en un Release de GitHub.

5. Sincronizar con Overleaf

¿Qué es Overleaf y para qué lo usamos?

Overleaf es un editor LaTeX online. Permite escribir desde el navegador y ver el PDF sin instalar nada en tu PC. En este proyecto lo usamos sólo si queremos editar en la nube y mantener sincronía con el repositorio de GitHub.

Formas de conectarlo con este repositorio

Hay dos opciones. Elegí una (no hace falta usar las dos).

Opción A — Usar la URL de Git de Overleaf (sirve con cualquier plan)

1. En el proyecto de Overleaf, abrí el menú **Git** y copiá la URL (formato `https://git.overleaf.com/<id>`).
2. En tu repositorio local, agregá Overleaf como remoto:

   ```
   git remote add overleaf https://git.overleaf.com/<id>
   ```

3. Subir tus cambios locales a Overleaf:

   ```
   git push overleaf master
   ```

4. Traer cambios hechos en Overleaf a tu PC:

   ```
   git pull overleaf master
   ```

Notas: (1) La rama por defecto en Overleaf suele ser `master`. (2) Si Overleaf ya tiene contenido y querés conservarlo, conviene clonar primero desde Overleaf y luego agregar GitHub como segundo remoto.

Opción B — Crear el proyecto Overleaf desde GitHub (si tu plan lo permite)

1. En Overleaf: **New Project** $\rightarrow$ **From GitHub**.
2. Elegí este repositorio `documentation`.
3. Cuando edites online, usá los botones **Pull from GitHub** (traer) y **Push to GitHub** (enviar).

Flujo de trabajo recomendado

1. **PC:** hacé cambios en una rama (por ejemplo `docs/intro`) y probá compilar.
2. **Subí** la rama a GitHub y abrí un Pull Request hacia `main`.
3. **Merge** aprobado $\rightarrow$ actualizá Overleaf:
   - Opción A: `git push overleaf master` desde tu PC (o `git pull overleaf master` si editaron online).
   - Opción B: en Overleaf, **Pull from GitHub** para traer lo último de `main`.
4. Cuando quieras entregar una versión, compilá el PDF y publicá un *Release* en GitHub adjuntando `main.pdf`.

Problemas comunes y soluciones rápidas

- **Acceso restringido en Overleaf:** pedí permiso de lectura/escritura al propietario del proyecto.
- **Ramas diferentes (`main` vs `master`):** en Overleaf la rama suele ser `master`. Alineá con:

  ```
  git checkout main
  git push overleaf main:master
  ```

- **Conflictos al hacer pull/push:** primero traé cambios, resolvé archivos marcados, hacé commit y volvé a empujar.
- **Overleaf no actualiza el PDF:** guardá todos los archivos (`Ctrl+S`) y forzá recompilación en Overleaf.
- **Credenciales:** si pide usuario/clave, usá los datos de Overleaf (Opción A) o reconectá tu GitHub (Opción B).

Checklist rápido

- Elegí A (URL de Git) o B (From GitHub).
- Probá una ida y vuelta: editar una línea en local $\rightarrow$ enviar a Overleaf; editar una línea en Overleaf $\rightarrow$ traer a local.
- Confirmá que el PDF se compila en ambos (PC y Overleaf).

Comandos útiles (resumen)

```
# agregar remoto Overleaf (Opción A)
git remote add overleaf https://git.overleaf.com/<id>

# enviar de local a Overleaf
git push overleaf master

# traer de Overleaf a local
git pull overleaf master
```

Puente local → Overleaf y vuelta

- **Desde local a Overleaf (Opción A)**:

  ```
  git push overleaf master
  ```

- **Desde Overleaf a local (Opción A)**:

  ```
  git pull overleaf master
  ```

- **Si tu rama principal es `main`** y Overleaf usa `master`:

  ```
  git checkout main
  git push overleaf main:master
  ```