</> ByteNova

Git y GitHub para principiantes

Guía práctica desde cero

Aprenderás control de versiones, repositorios, commits, ramas, GitHub, clone, pull, push, README y un flujo de trabajo seguro para tus proyectos.

Nivel: principiante

Creado por ByteNova · 2026

Contenido de la guía

Cómo aprovechar esta guía

Practica en una carpeta nueva. Ejecuta un comando, revisa el resultado con git status y continúa solamente cuando comprendas qué cambió.

1. Control de versiones, Git y GitHub

El control de versiones registra los cambios realizados en los archivos de un proyecto. Permite comparar versiones, recuperar estados anteriores y colaborar con otras personas sin crear copias desordenadas.

Git

Programa instalado en la computadora que administra el historial del proyecto.

GitHub

Plataforma en Internet que aloja repositorios Git y facilita la colaboración.

Repositorio

Carpeta de proyecto cuyo historial es administrado por Git.

Commit

Versión guardada acompañada de un mensaje descriptivo.

Idea clave

Git controla versiones. GitHub almacena y comparte repositorios Git. Puedes usar Git sin GitHub.

2. Instalación y configuración inicial

Después de instalar Git, abre una terminal y comprueba la versión:

```
git --version
```

Configura el nombre y el correo que aparecerán en tus commits:

```
git config --global user.name "Tu nombre"
git config --global user.email "correo@ejemplo.com"
```

Consulta la configuración actual:

```
git config --list
```

Privacidad

El correo puede quedar registrado en el historial. Utiliza uno apropiado para proyectos públicos.

3. Crear un repositorio local

Abre la terminal dentro de la carpeta del proyecto y ejecuta:

```
cd mi-proyecto
git init
```

Git crea una carpeta oculta llamada .git. Allí conserva la información del repositorio.

Importante

No elimines la carpeta .git. Si la borras, perderás la conexión con el historial local.

El comando más útil para conocer la situación del repositorio es:

```
git status
```

4. Preparar archivos y crear commits

Git separa el trabajo en tres estados: archivos modificados, área de preparación y versiones guardadas.

Modificado

El archivo cambió, pero todavía no está preparado.

Preparado

El cambio fue seleccionado con git add.

Confirmado

El cambio fue guardado mediante un commit.

Sin cambios

El directorio coincide con la última versión guardada.

Preparar un archivo específico:

```
git add index.html
```

Preparar todos los cambios de la carpeta actual:

```
git add .
```

Crear un commit:

```
git commit -m "Agregar sección de contacto"
```

Mensajes útiles

Describe qué cambio contiene la versión. “Corregir menú móvil” es más claro que “cambios”.

5. Historial y diferencias

Consulta el historial completo o una vista resumida:

```
git log
git log --oneline
```

Antes de preparar los archivos, puedes revisar las diferencias:

```
git diff
```

Para ver los cambios ya preparados:

```
git diff --staged
```

Buena práctica

Revisa git status y git diff antes de cada commit. Así reduces la posibilidad de guardar archivos no deseados.

6. Ramas y fusiones

Una rama permite desarrollar una función separada de la versión principal.

```
git switch -c nueva-seccion
```

Después de completar y guardar el trabajo, regresa a la rama principal y fusiona los cambios:

```
git add .
git commit -m "Crear nueva sección"
git switch main
git merge nueva-seccion
```

Consulta las ramas disponibles:

```
git branch
```

Conflictos

Si dos ramas modifican la misma parte de un archivo, Git puede pedirte que elijas qué contenido conservar. Revisa el archivo, resuelve las marcas y crea un nuevo commit.

7. Repositorios remotos en GitHub

En GitHub puedes crear un repositorio público o privado. Antes de publicar, revisa que el proyecto no contenga contraseñas, claves, archivos personales ni datos sensibles.

Conecta el repositorio local con la URL que GitHub proporciona:

```
git remote add origin URL-DEL-REPOSITORIO
git branch -M main
git push -u origin main
```

Comprueba los remotos configurados:

```
git remote -v
```

Seguridad

Nunca subas archivos .env, claves privadas, tokens ni credenciales. Si un secreto fue publicado, elimínalo del historial y reemplázalo inmediatamente.

8. Clone, pull y push

git clone

Descarga una copia completa de un repositorio por primera vez.

git pull

Obtiene e integra cambios del repositorio remoto.

git push

Envía los commits locales al repositorio remoto.

origin

Nombre común utilizado para el remoto principal.

```
git clone URL-DEL-REPOSITORIO
cd nombre-del-repositorio
git pull
git push
```

Orden recomendado

En proyectos compartidos, descarga los cambios recientes antes de enviar los tuyos. Resuelve cualquier conflicto antes de volver a hacer push.

9. README y .gitignore

README.md explica qué hace el proyecto, cómo utilizarlo y cómo está organizado.

```
# Mi proyecto

Página creada para practicar HTML, CSS y JavaScript.

## Funciones
- Diseño responsive
- Modo oscuro
- Formulario de contacto
```

El archivo .gitignore indica qué archivos o carpetas no deben añadirse al repositorio:

```
node_modules/
.env
*.log
archivos-temporales/
```

Atención

Agregar un archivo a .gitignore no elimina versiones que ya fueron guardadas en commits anteriores.

10. Flujo de trabajo recomendado

Un flujo sencillo para trabajar de forma ordenada:

- Actualiza el repositorio con git pull cuando corresponda.
- Crea una rama para una función o corrección importante.

- Realiza cambios pequeños y relacionados.
- Revisa con git status y git diff.
- Prepara solamente los archivos necesarios.
- Crea un commit con un mensaje descriptivo.
- Prueba el proyecto antes de fusionar.
- Envía los commits mediante git push.

```
git pull
git switch -c mejorar-menu
# realizar cambios
git status
git diff
git add index.html style.css
git commit -m "Mejorar menú móvil"
git push -u origin mejorar-menu
```

11. Errores frecuentes y seguridad

- Ejecutar comandos dentro de la carpeta equivocada.
- Olvidar guardar el archivo antes de crear el commit.
- Utilizar git add . sin revisar qué se incluirá.
- Crear commits demasiado grandes o con mensajes vagos.
- Trabajar siempre directamente en la rama principal.
- Subir credenciales, archivos privados o datos sensibles.
- Eliminar accidentalmente la carpeta .git.
- Hacer push sin descargar cambios recientes.

El mensaje “fatal: not a git repository” normalmente indica que la terminal no está dentro de un repositorio Git:

```
pwd
ls
cd carpeta-correcta
git status
```

No copies comandos destructivos

Evita comandos de recuperación o eliminación que no comprendas. Antes de reescribir el historial, crea una copia de seguridad.

12. Práctica guiada

Crea una carpeta nueva y ejecuta los comandos uno por uno:

```
mkdir practica-git
cd practica-git
git init
echo "# Práctica de Git" > README.md
git status
git add README.md
git commit -m "Crear README"
git log --oneline
```

Agrega un archivo HTML y una rama de estilos:

```
touch index.html
git add index.html
git commit -m "Agregar página principal"
git switch -c agregar-estilos
touch style.css
git add style.css
git commit -m "Agregar hoja de estilos"
git switch main
git merge agregar-estilos
```

Próximo paso

Crea un repositorio de práctica en GitHub, conéctalo como remoto y envía tu rama principal. No utilices información privada.

Conclusión

Git permite registrar versiones, comparar cambios, trabajar con ramas y recuperar estados anteriores. GitHub permite almacenar esos repositorios en Internet, compartirlos y colaborar.

Domina primero git status, git add, git commit y git log. Después practica ramas, repositorios remotos, pull y push.

Resumen final

Trabaja en cambios pequeños, revisa antes de guardar, escribe mensajes claros y nunca publiques credenciales.

Continúa aprendiendo en ByteNova

Visita https://bytenova.app/ para consultar artículos, rutas, retos y otras guías gratuitas.

Creado por ByteNova · 2026