
**Institución:** Universidad de Colima

**Escuela:** Facultad de Ingeniería Mecánica y Eléctrica

**Carrera:** Ingeniería en Computación Inteligente

**Materia:** Cómputo en la Nube

**Profesor:** Carrillo Zepeda Oswaldo

**Alumno:** Herrera Escareño Kevin Alejandro

**Grado y grupo:** 6°B

**Nombre de la actividad:** Git y GitHub Desktop

**Fecha:** 17 de Mayo de 2022

**Lugar:** Colima, Col.

## Git

Git es un sistema de control de versiones distribuido de código abierto desarrollado por Linus Torvalds, el creador de Linux. El control de versiones se refiere al proceso de guardar diferentes archivos, o "versiones", a lo largo de las diferentes etapas de un proyecto. Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden revertir algunos de los cambios hechos.


**Figura 1.** Logotipo de Git.

La principal diferencia entre Git y cualquier otro sistema de control de versiones (VCS) es la forma en la que manejan sus datos. Conceptualmente, la mayoría de los VCS almacenan la información como una lista de cambios en los archivos. Estos sistemas manejan la información que almacenan como un conjunto de archivos y las modificaciones hechas a cada uno de ellos a través del tiempo.

Git maneja sus datos como un conjunto de copias instantáneas (snapshots) de un sistema de archivos miniatura. Cada vez que se confirma un cambio, o se guarda el estado de un proyecto en Git, él básicamente toma una foto del aspecto de todos los archivos en ese momento y guarda una referencia a esa copia instantánea. Para ser eficiente, si los archivos no se han modificado Git no almacena el archivo de nuevo, sino un enlace al archivo anterior idéntico que ya tiene almacenado. Git maneja sus datos como una secuencia de copias instantáneas.

Git ofrece ramas de características. Esto significa que cada miembro del equipo puede dividir una rama de características que proporcionará un repositorio local aislado para hacer cambios en el código. Las ramas de características no afectan a la rama maestra, que es donde se encuentra el código original del proyecto. Una vez que se hayan realizado los cambios y el código actualizado esté listo, la rama de características puede fusionarse de nuevo con la rama maestra, que es la forma en que se harán efectivos los cambios en el proyecto. Todas las modificaciones subidas se guardan en versiones independientes, no sobrescribiendo en el archivo original.

### Ventajas

Git ofrece varias ventajas frente a otros sistemas tradicionales:

- **Sistema distribuido**, sin un punto central de fallo, que permite el trabajo incluso sin conexión.
- **Rápido y ligero**, optimizado para hacer operaciones de control muy rápidas.

- **Crear ramas y mezclarlas** es rápido y poco propenso a problemas, al contrario que en otros sistemas tradicionales.
- **La integridad de la información está asegurada** gracias a su modelo de almacenamiento, que permite predecir este tipo de problemas.
- **Permite flujos de trabajo muy flexibles.**
- **El concepto de área de preparación** o staging permite versionar los cambios de manera personalizada, no todo o nada.
- **Es gratis** y de código abierto.

## Desventajas

Git presenta los siguientes inconvenientes:

- **Es más complejo** que los sistemas centralizados tradicionales, ya que entran en juego más repositorios, más operaciones y más posibilidades para trabajar en equipo.
- **La curva de aprendizaje es pronunciada** y la documentación es tan compleja que muchas veces no resulta de ayuda.
- **Los comandos y algunos conceptos** que usa pueden llegar a ser **confusos**, al igual que algunos mensajes que muestra.
- Por defecto, **se lleva mal con archivos binarios muy grandes**, como vídeos o documentos gráficos muy pesados.

## Extensiones de archivos compatibles con Git

Es posible utilizar Git para crear, abrir o editar el archivo con estas extensiones:

- **GITIGNORE**: Git Ignore File.
- **GITATTRIBUTES**: Git Attributes File.
- **GITKEEP**: Git Keep File.
- **SNAG**: SnagIt Capture File.
- **SNAGPROJ**: Snagit Project File.
- **SNAGSTYLES**: Snagit Style Archive.
- **SNAGITSTAMPS**: Snagit Stamp Archive.
- **ZGT**: Oren Scientific Word Processor.
- **GITCONFIG**: Git Configuration.
- **GITEXT**: Git Extension.
- **GITMODULES**: Git Module.
- **NPMIGNORE**: Git Data.
- **SNAGACC**: SnagIt File.
- **SNAGPROF**: SnagIt File.
- **SNAGUNDO**: SnagIt Data.
- **SSF**: SnagIt File.

## Instalación

### Instalar GIT en Windows

En Windows, sólo se tiene que descargar el instalador y ejecutarlo. Sigua estos sencillos pasos para hacerlo:

1. Descargar el instalador a través del siguiente enlace: http://git-scm.com/download/win.
2. Una vez que descargado el instalador, hacer doble clic sobre el ejecutable para que comience el proceso de instalación y sigua las instrucciones que aparecerán en pantalla. Al igual que cualquier otro programa, se tendrá que dar “Next” (siguiente) en varias ocasiones hasta que aparezca la opción “Finish” (terminar) para completar la instalación.
3. Ahora es necesario abrir el símbolo de sistema y escribir los siguientes comandos en la terminal:
    a. git config --global user.name "nombre".
    b. git config --global user.email "ejemplo@email.com".

### Instalar GIT en MacOS

Es necesario seguir los siguientes paso:

1. Descargar el instalador a través del siguientes enlace: http://git-scm.com/download/mac.
2. Seguir las instrucciones que aparecerán en el programa de instalación.
3. Ahora ejecute los siguientes comandos en la terminar para configurar su correo y el nombre de usuario que están asociados a la cuenta GIT:
    a. git config --global user.name "nombre".
    b. git config --global user.email "ejemplo@email.com".

### Instalar GIT en Linux (Ubuntu/Debian)

1. Abrir la terminal y ejecutar los siguientes comandos:
    a. sudo apt-get update
    b. sudo apt-get install git
2. Verificar que la instalación se haya logrado correctamente usando el comando: git –version.
3. A continuación, ejecutar los siguientes comandos en la terminal para poder configurar el correo y el nombre de usuario que están asociados a la cuenta GIT:
    a. git config --global user.name "nombre".
    b. git config --global user.email "ejemplo@email.com".

### Instalar GIT en Linux (Fedora)

1. Abrir la terminal y ejecutar los siguientes comandos:
    a. sudo dnf install git
    b. sudo yum install git
2. A continuación, ejecutar los siguientes comandos en la terminal para poder configurar el correo y el nombre de usuario que están asociados a la cuenta GIT:
    a. git config --global user.name "nombre".
    b. git config --global user.email "ejemplo@email.com".

## Funcionamiento

Para utilizar Git, los desarrolladores utilizan comandos específicos para copiar, crear, cambiar y combinar el código. Estos comandos pueden ejecutarse directamente desde la línea de comandos o utilizando una aplicación como GitHub Desktop. Aquí algunos comandos comunes para utilizar Git:

- **git init** inicializa un repositorio nuevo de Git y comienza a rastrear el directorio existente. Este agrega una subcarpeta oculta dentro del directorio existente que hospeda la estructura de datos interna que se requiere para el control de versiones.
- **git clone** crea una copia local de un proyecto que ya existe remotamente. El clon incluye todos los archivos, historial y ramas del proyecto.
- **git add** prueba un cambio. Git rastrea los cambios que se hacen a la base del código de un desarrollador, pero es necesario probarlos y tomar una captura de pantalla de ellos para incluirla en el historial del proyecto. Este comando realiza pruebas, la primera parte de este proceso de dos pasos. Cualquier cambio que se pruebe, se convertirá en parte de la siguiente captura de pantalla y también del historial del proyecto. Las pruebas y confirmaciones por separado otorgan a los desarrolladores el control completo sobre el historial y sobre el proyecto sin cambiar la forma en la que codifican y trabajan.
- **git commit** guarda la captura de pantalla del historial del proyecto y completa el proceso de rastreo de cambios. En resumen, una confirmación funciona tal como el tomar una fotografía. Todo lo que se pruebe con **git add** se convertirá en parte de la captura de pantalla con **git commit**.
- **git status** muestra el estado de los cambios como "sin rastrear", "modificados" o "probados".
- **git branch** muestra las ramas en las que se está trabajando localmente.
- **git merge** fusiona las líneas de desarrollo juntas. Este comando habitualmente se utiliza para combinar los cambios que se realizan en dos ramas distintas.
- **git pull** actualiza la línea de desarrollo local con actualizaciones de sus contrapartes remotas. Los desarrolladores utilizan este comando si un compañero de equipo hizo confirmaciones en una rama en un repositorio remoto y quieren reflejarlos en su ambiente local.
- **git push** actualiza el repositorio remoto con cualquier confirmación que se haga localmente a una rama.

## GitHub Desktop

GitHub Desktop es, de hecho, una Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés) diseñada para facilitar el uso de Git. Es posible utilizar GitHub Desktop para completar la mayoría de los comandos de Git desde una computadora de escritorio con confirmaciones visuales para los cambios. Permite subir, extraer y clonar repositorios remotos y utilizar herramientas colaborativas tales como atribuir confirmaciones y crear solicitudes de extracción. GitHub Desktop es un proyecto de código abierto, por lo tanto, es posible ver el itinerario, contribuir con el proyecto, o abrir un informe de problemas para proporcionar retroalimentación o solicitudes de características.


**Figura 2.** Logotipo de GitHub Desktop.

## Ventajas

GitHub Desktop cuenta con las siguientes ventajas:

- La GUI le permite al usuario interactuar con el programa a través de un dispositivo visual que reemplaza la línea de comandos.
- Se reduce la curva de aprendizaje de Git.
- Búsqueda muy rápida en las estructura de los repositorios.
- Amplia comunidad y facilidad para encontrar ayuda.
- Ofrece herramientas de cooperación prácticas y una buena integración con Git.
- Fácil integración con otros servicios de terceros.
- Trabaja también con TFS, HG y SVN.
- Servicio gratuito.

## Desventajas

GitHub Desktop presenta los siguientes inconvenientes:

- Algunos servicios son de pago.
- Tiene limitaciones de espacio, ya que no se puede exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis.

## Extensiones de archivos compatibles

Archivos compatibles:

- PNG (.png)
- GIF (.gif)
- JPEG (.jpg)
- SVG (.svg)
- Archivos de registro (.log)
- Documentos de Microsoft Word (.docx), PowerPoint (.pptx) y Excel (.xlsx)
- Archivos de texto (.txt)
- PDF (.pdf)

- ZIP (.zip, .gz)
- Video (.mp4, .mov)

El tamaño máximo de archivo es:

- 10MB de imágenes y gifs.
- 10MB para videos que se suban a un repositorio que pertenezca a un usuario u organización en un plan gratuito de GitHub.
- 100MB para videos que se suban a los repositorios que pertenezcan a un usuario u organización de un plan de pago de GitHub.
- 25MB para el resto de los archivos.

## Compatibilidad de lenguajes

La mayoría de las características de GitHub funcionan sin importar en qué lenguaje esté escrito un código. Es posible buscar código o habilitar el resaltado de sintaxis con base en cualquier lenguaje conocido en GitHub.

Algunos productos de GitHub tienen características que solo son compatibles actualmente para un subconjunto de lenguajes de programación. Los lenguajes centrales para las características de GitHub incluyen a C, C++, C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala y TypeScript.

| Lenguaje | Navegación de código | Escaneo de código | Gráficas de dependencias, alertas del dependabot, actualizaciones de seguridad del dependabot | Actualizaciones de versión del dependabot | GitHub Actions | Registro del paquete de GitHub |
|---|---|---|---|---|---|---|
| **C** | ✓ | × | ✓ | × | × | ✓ |
| **C++** | ✓ | × | ✓ | × | ✓ | ✓ |
| **C#** | ✓ | ✓ | ✓ | ✓ dotnet CLI | ✓ dotnet CLI | ✓ |
| **Go** | ✓ | ✓ | ✓ | ✓ Go modules | ✓ Go modules | ✓ |
| **Java** | ✓ | ✓ | ✓ | ✓ Maven | ✓ Maven, Gradle | ✓ |
| **JavaScript** | ✓ | ✓ | ✓ | ✓ npm, Yarn | ✓ npm | ✓ |
| **PHP** | ✓ | ✓ | ✓ | ✓ Composer | ✓ Composer | ✓ |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Python** | ✓ | ✓ precise | ✓ | ✓ pip | ✓ pip | ✓ |
| **Ruby** | ✓ | ✓ | ✓ | ✓ RubyGems | ✓ RubyGems | ✓ |
| **Scala** | ✓ | × | ✓ | ✓ Maven | ✓ Maven, Gradle | ✓ |
| **TypeScript** | ✓ | ✓ | ✓ | ✓ npm, Yarn | ✓ npm | ✓ |

**Figura 3.** Características compatibles con los administradores de paquetes.

### Instalación

1. Para instalar GitHub Desktop, visite la página de descargas: https://desktop.github.com/.
2. Seguir las instrucciones que aparecerán en el programa de instalación.
3. Después de que haya instalado GitHub Desktop, puede autenticar la aplicación con su cuenta en GitHub o en GitHub Enterprise. Esta autenticación le permite conectarse remotamente a los repositorios en GitHub o en GitHub Enterprise.
    a. En el menú desplegable de archivo, da clic en Opciones. En la ventana de opciones, da clic en Cuentas y sigue los pasos para iniciar sesión.

### Funcionamiento

GitHub hospeda repositorios de Git y proporciona a los desarrolladores las herramientas para generar un código mejor mediante características de línea de comandos, propuestas (debates en hilo), solicitudes de cambio, revisión de código o el uso de un conjunto de apps gratuitas y de pago en GitHub Marketplace. Con las capas de colaboración tales como el flujo de GitHub, una comunidad de 15 millones de desarrolladores y un ecosistema con cientos de integraciones, GitHub cambia la forma en la que se crea el software.

GitHub crea una colaboración directamente en el proceso de desarrollo. El trabajo se organiza en los repositorios donde los desarrolladores pueden describir los requisitos o la dirección y marcar las expectativas para los miembros de los equipos. Posteriormente, utilizando el flujo de GitHub, los desarrolladores simplemente crean una rama para trabajar en las actualizaciones, confirmar cambios para guardarlos, abrir una solicitud de cambios para proponerlos y debatirlos y fusionar solicitudes de cambio una vez que todos estén de acuerdo.

## Diferencias entre Git y GitHub

Git es un software de VCS local que permite a los desarrolladores guardar instantáneas de sus proyectos a lo largo del tiempo. Generalmente es mejor para uso individual.

GitHub es una plataforma basada en la web que incorpora las características de control de versiones de git para que puedan ser utilizadas de forma colaborativa. También incluye características de gestión de proyectos y equipos, así como oportunidades para la creación de redes y la codificación social.

# Bibliografía


Alarcón, J. M. (2020, 1 junio). *Qué es Git, ventajas e inconvenientes y por qué deberías aprenderlo (bien)*. campusMVP.es. Recuperado 14 de mayo de 2022, de https://www.campusmvp.es/recursos/post/que-es-git-ventajas-e-inconvenientes-y-por-que-deberias-aprenderlo-bien.aspx

B., G. (2021a, marzo 8). ¿Qué es GitHub y Cómo Usarlo? Tutoriales Hostinger. Recuperado 14 de mayo de 2022, de https://www.hostinger.mx/tutoriales/que-es-github

B., G. (2021b, noviembre 25). Cómo instalar GIT en Windows, MacOS y Linux. Tutoriales Hostinger. Recuperado 14 de mayo de 2022, de https://www.hostinger.mx/tutoriales/instalar-git-en-distintos-sistemas-operativos

GitHub. (s. f.-a). Microsoft. GitHub Docs. Recuperado 14 de mayo de 2022, de https://docs.github.com/es/get-started/learning-about-github/github-language-support

GitHub. (s. f.-b). *Microsoft*. GitHub Docs. Recuperado 14 de mayo de 2022, de https://docs.github.com/es/get-started/writing-on-github/working-with-advanced-formatting/attaching-files

GitHub. (s. f.-c). Microsoft. GitHub Docs. Recuperado 14 de mayo de 2022, de https://docs.github.com/es/get-started/using-git/about-git

GitHub. (s. f.-d). Microsoft. GitHub Docs. Recuperado 14 de mayo de 2022, de https://docs.github.com/es/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop

Kinsta. (2020, 29 diciembre). Git vs Github: ¿Cuál es la Diferencia y cómo Empezar? Recuperado 14 de mayo de 2022, de https://kinsta.com/es/base-de-conocimiento/git-vs-github/