**Universidad Americana**

Managua-Nicaragua



**Documentación de Git y GitHub**

**Autores:**

Hendrix Devadip Aguirre Zamora

**Profesor:**

Armando José López López

**Fecha:**

20/10/2025

**Clonar un Repositorio (git clone)**

* **En qué consiste:** "Clonar" es el proceso de crear una copia local, idéntica y completamente funcional de un repositorio que existe remotamente (en GitHub). A diferencia de una "descarga" simple, la clonación trae no solo los archivos actuales, sino **todo el historial de versiones** (commits) del proyecto. Esto vincula tu copia local con la remota.
* **Pasos (Se realiza una sola vez por proyecto):**
  1. **En GitHub:**
     + Navega a la página principal del repositorio.
     + Haz clic en el botón verde < > Code.
     + Copia la URL HTTPS proporcionada (termina en .git).
  2. **En tu Escritorio (o ubicación deseada):**
     + Haz clic derecho en un espacio vacío.
     + Selecciona "Open Git Bash Here".
  3. **En la Terminal de Git Bash:**
     + Escribe el comando git clone, deja un espacio y pega la URL copiada (Shift + Insert o clic derecho > Paste).
     + Comando de ejemplo: git clone https://github.com/usuario/proyecto.git
     + Presiona Enter.
  4. **Resultado:** Se creará una nueva carpeta en tu escritorio con el nombre del repositorio, lista para trabajar.

**Sincronizar Cambios (El Flujo de Trabajo Diario)**

Este es el ciclo que repites cada vez que trabajas.

**1. Actualizar tu Repositorio Local (git pull)**

* **En qué consiste:** Usar git pull (jalar) es la acción de **actualizar tu copia local**. Revisa el repositorio remoto (GitHub) y descarga todos los cambios nuevos que otros miembros del equipo hayan subido desde la última vez que te conectaste. (**Importante:** Haz esto siempre **antes** de empezar a modificar archivos).
* **Pasos:**
  1. Navega hasta la **carpeta del proyecto** (la que clonaste).
  2. Haz clic derecho dentro de esa carpeta.
  3. Selecciona "Open Git Bash Here". (La terminal debe indicar que estás dentro del proyecto).
  4. Escribe el comando git pull y presiona Enter.

**2. Trabajar en tus Archivos**

* **En qué consiste:** Este no es un comando de Git, es simplemente el acto de hacer tu trabajo.
* **Pasos:**
  1. Minimiza la terminal de Git Bash.
  2. Abre los archivos del proyecto (Word, Excel, código, etc.) con tus programas habituales.
  3. Modifica y guarda los archivos (Ctrl + S) como lo harías normalmente.

**3. Subir tus Cambios (El Proceso de 3 Pasos)**

(Haz esto **después** de haber guardado tus archivos y cuando estés listo para compartir tu avance).

**Paso 3.1: Preparar Cambios (git add)**

* **En qué consiste:** git add (añadir) es como poner tus archivos modificados en un "área de preparación" o una "caja de envío". Le estás diciendo a Git: "Quiero que estos cambios específicos se incluyan en mi próximo guardado (commit)". El comando git add . (con un punto) es un atajo para añadir todos los archivos que hayas modificado.
* **Pasos:**
  1. Regresa a la terminal de Git Bash (que sigue abierta dentro de la carpeta del proyecto).
  2. Escribe git add . y presiona Enter.

**Paso 3.2: Confirmar Cambios Localmente (git commit)**

* **En qué consiste:** git commit (confirmar) es el acto de "tomar la foto" o **guardar permanentemente** los cambios que preparaste (con git add) en el historial de **tu computadora local**. Cada "commit" requiere un mensaje (-m) donde explicas qué hiciste.
* **Pasos:**
  1. En la misma terminal, escribe git commit -m "Tu mensaje descriptivo aquí".
  2. Ejemplo: git commit -m "Corregí errores de ortografía en la introducción"
  3. Presiona Enter. Tus cambios ya están guardados en el historial de tu PC.

**Paso 3.3: Subir Cambios a GitHub (git push)**

* **En qué consiste:** git push (empujar) es la acción de **enviar** todos esos "commits" (guardados locales) que has hecho, desde tu computadora hacia el repositorio remoto en GitHub. Solo después de hacer "push", tu equipo podrá ver y descargar tus cambios.
* **Pasos:**
  1. En la misma terminal, escribe git push y presiona Enter.
  2. **Resultado:** Tus cambios ahora están seguros en la nube y disponibles para todos.

**Diferencias entre Git y GitHub**

Es fundamental distinguir entre estos dos conceptos, ya que, aunque están intrínsecamente relacionados, cumplen funciones diferentes.

* **Git:** Es el software en sí. Se define como un **Sistema de Control de Versiones Distribuido (VCS)**. Su función principal es ejecutarse de manera local en la computadora del usuario (en la terminal o línea de comandos) para rastrear y gestionar los cambios en los archivos de un proyecto a lo largo del tiempo. Permite guardar "instantáneas" (commits) del proyecto, crear ramas de desarrollo y revertir a versiones anteriores.
* **GitHub:** Es una plataforma o servicio de alojamiento web. Utiliza Git como su tecnología subyacente, pero añade una capa de colaboración y una interfaz gráfica basada en la nube. GitHub es el "lugar" en internet donde se almacenan los proyectos (repositorios) gestionados con Git, permitiendo que múltiples desarrolladores colaboren, revisen el código, gestionen incidencias y administren el proyecto de forma centralizada.

En resumen: Git es la herramienta de control de versiones, y GitHub es la plataforma para alojar y colaborar en los repositorios de Git.