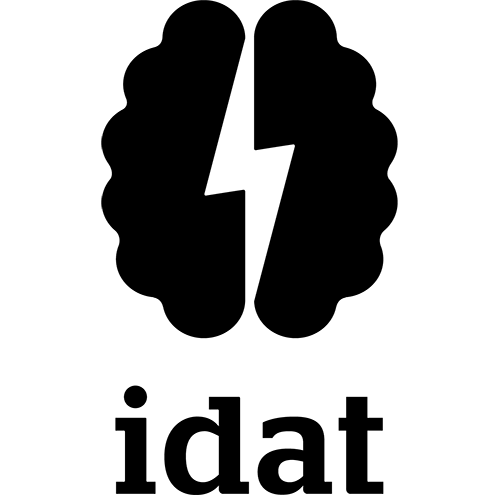
**INSTITUTO IDAT**

**ESCUELA DE CODING**

****

**INFORME ACADÉMICO**

“COMANDOS ESENCIALES DE GIT”

**ESTUDIANTE**

CASTILLO HUAMANÍ, ARNOLD HANS

**CURSO**

PROGRAMACIÓN FRONT-END

**DOCENTE**

ING. PINEDA LOPEZ, ROBERTO DAVID

**VIRTUAL – PERÚ**

**2025**

1. **INTRODUCCIÓN**

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar cambios en el código fuente de manera eficiente. Esta herramienta fundamental en proyectos de desarrollo de software, fue creado por Linus Torvalds en 2005. Su creación facilita la colaboración, seguimiento de cambios y mantenimiento del historial del proyecto. Por otro lado, Torvalds es el creador del núcleo del sistema operativo Linux, y su principal objetivo fue desarrollar Git para gestionar el código fuente del sistema operativo. En este informe, se describen los comandos más utilizados en Git, su funcionamiento y utilidad.

1. **COMANDOS BÁSICOS DE GIT**
   1. **Git Init**

* Inicializa un nuevo repositorio Git en el directorio actual
* Cuando quieres comenzar a gestionar un proyecto con Git, ejecutas git init. Esto crea una carpeta oculta “.git” que contiene toda la estructura necesaria para gestionar el repositorio.
  1. **Git Clone**
* Clona un repositorio existente en tu máquina local
* Para obtener una copia exacta de un repositorio existente, ejecuta git clone <URL>. Esto descarga todos los archivos y el historial del proyecto.
  1. **Git Status**
* Muestra el estado de los archivos en el directorio de trabajo y en el área de staging
* Es útil para verificar qué archivos han sido modificados, añadidos o eliminados.
  1. **Git Add**
* Añade archivos al área de staging para preparar su commit
* Para incluir cambios en el próximo commit, ejecuta git add <archivo> o “git add .” para ñadir todos los cambios en el directorio actual
  1. **Git Commit**
* Registra los cambios en el repositorio con descripción
* Después de agregar archivos al área de staging, ejecuta git commit -m “mensaje” para crear un nuevo punto en el historial con una descripción del cambio.
  1. **Git Push**
* Envía los commits locales al repositorio remoto
* Para compartir tus cambios con otros, ejecuta git push origin <rama>.
  1. **Git Pull**
* Descarga y fusiona los cambios del repositorio remoto a la rama actual.
* Para actualizar tu copia local con los cambios del equipo, ejecuta git pull.
  1. **Git Branch**
* Lista, crea o elimina ramas
* Ejecuta git Branch para listar ramas, git Branch <nombre> para crear una nueva rama
  1. **Git Checkout**
* Cambia de rama o restaura archivos a un estado anterior
* Para cambiar a otra rama, ejecuta git checkout <rama>. Para restaurar archivos a su estado en el último commit, usa git chekout -- <archivo>
  1. **Git Merge**
* Fusiona otra rama en la rama actual
* Para integrar cambios de otra rama, ejecuta git merge <rama>.
  1. **Git Log**
* Muestra el historial de commits
* Ejecuta git log para ver los commits realizados en el repositorio.
  1. **Git Diff**
* Muestra las diferencias entre archivos o commits.
* Para ver cambios no staged, usa git diff. Para ver cambios staged, usa git diff –cached.
  1. **Git Reset**
* Deshace cambios en el área de staging del archivo.
* Para quitar archivos del área de staging, ejecuta git reset <archivo>- Para deshacer commits, usa git reset --soft o git reset --hard.
  1. **Git RM**
* Elimina archivos del repositorio y del directorio de trabajo
* Ejecuta git rm <archivo> para eliminar archivos y prepararlos para el commit.
  1. **Git Fetch**
* Obtiene los cambios del remoto sin fusionarlos
* Ejecuta git fetch para obtener actualizaciones sin modificar tu rama actual.
  1. **Git Stash**
* Guarda temporalmente cambios no confirmados para limpiar el área del trabajo.
* Ejecuta Git stash para guardar cambiso no commitados y git stash pop para recuperarlos.
  1. **Git Tag**
* Marca un punto en la historia del proyecto con una etiqueta
* Para crear nueva etiqueta, ejecuta git tag <nombre>.
  1. **Git Rebase**
* Reaplica commits sobre otra base, para mantener un historial lineal
* Para rebasar tu rama sobre otra, ejecuta git rebase <rama>.
  1. **Git Cherry-Pick**
* Aplica un commit específico en la rama actual
* Ejecuta git cherry-pick <commit> para copiar un commit particular.
  1. **Git Config**
* Configura opciones de Git, como tu nombre o correo
* Para establecer tu nombre, ejecuta git config –global user.name “Tu Nombre”. Para establecer tu correo, git config – global user.email “tuemail@example.com”.