**Qué tan importante es aprender este tipo de herramientas y por qué?**

* Te ayuda a trabajar en equipo sin interferir el trabajo de tu compañero/a.
* Podés probar cosas nuevas sin miedo a romper nada.
* Tenés un registro de todo lo que hiciste.
* Rastreabilidad: Git no solo guarda cambios, sino que permite identificar cuándo y por qué se introdujo un error (útil para debugging).
* Integración con flujos de trabajo modernos: Herramientas como CI/CD (Integración Continua/Despliegue Continuo) dependen de Git para automatizar pruebas y despliegues.
* Estándar en la industria: No solo en desarrollo de software, sino también en áreas como ciencia de datos, diseño UX/UI, y documentación técnica.  
    
  Ejemplo práctico: En equipos distribuidos, Git evita el clásico "¿Quién tiene la versión más reciente del archivo?"

Lo usan en casi todos los trabajos relacionados con tecnología

**Qué tipos de archivos deberían subirse a un servidor Git y cuales no?**

**Debería subirse:**

* El código que estás produciendo.
* Archivos que explican o configuran el proyecto (ej, readme)
* Cosas necesarias para que funcione.
* Documentación técnica: Diagramas de arquitectura (en formato vectorial, como .svg), especificaciones de API (OpenAPI/Swagger).
* Scripts de construcción/configuración: Dockerfile, docker-compose.yml, scripts de despliegue.

**NO debería subirse:**

* Datos privados o sensibles (ej. Contraseñas) Incluso si se borran en un commit, quedan en el historial. Usar herramientas como git filter-repo para limpiarlos.
* Archivos pesados (ej. Fotos o videos=
* Archivos que se generan solos (ej. .log.) Archivos generados: Como los .log o binarios compilados. Para esto, el .gitignore es vital (ejemplo para Python):  
    
  plaintext \*.log pycache/ \*.pyc Caso especial:
* Archivos grandes (ej. datasets, videos): Usar Git LFS (Large File Storage) o almacenarlos externamente (AWS S3, Google Drive).

**Qué ventajas y desventajas nos brinda el software libre?**

**Ventajas**

* Es libre
* Se puede modificar
* Hay comunidad y colaboración
* Fomenta el aprendizaje
* No dependés de una empresa
* Coste cero: Ideal para startups o educación.
* Auditoría de seguridad: Al ser abierto, vulnerabilidades se detectan y parchean más rápido (ejemplo: OpenSSL vs Heartbleed).
* Independencia: Evita el vendor lock-in (estar atado a un proveedor, como Microsoft/Adobe).

**Desventajas**

* Dificultad inicial: Proyectos como Linux o Blender tienen curvas de aprendizaje, pero su comunidad ofrece tutoriales (ej: ArchWiki, Stack Overflow).
* Algunos proyectos tienen versiones empresariales con soporte pago (ej: Red Hat para Linux, GitLab Ultimate). Algunos proyectos están poco mantenidos
* Usar estándares abiertos (como ODF en vez de DOCX) reduce este problema.
* Requiere más iniciativa para resolver problemas

Ejemplo ilustrativo:  
  
Ventaja: GIMP (alternativa a Photoshop) es potente y gratis, pero...  
  
Desventaja: Requiere tiempo para acostumbrarse si vienes de herramientas privativas.