educación IT
CODE YOUR FUTURE

Git:
Desarrollo colaborativo

Módulo 1

Repositorios

Repositorios

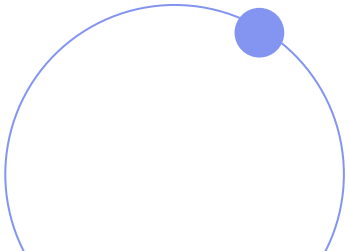
Un **repositorio Git** es la carpeta **.git /** dentro de un proyecto.

Rastrea todos los cambios realizados en los archivos de un proyecto y crea un historial a lo largo del tiempo. Es decir, **si se elimina la carpeta .git /, entonces desaparece el historial del proyecto**.

Un directorio .git tiene una **estructura** similar a la siguiente:

Repositorio Git - Directorios

- **objects/**
  En este directorio se **almacenan los datos de los objetos Git:** todo el contenido de los archivos que se hayan registrado, los *commits*, *branches* y *tags.*



  - **objects/[0-9a-f][0-9a-f]**
    Un objeto recién creado se almacena en su propio archivo. Los objetos se colocan en subdirectorios 256 utilizando los dos primeros caracteres del nombre de objeto SHA1 para mantener el número de entradas de directorio en los objetos en un número manejable.

    Los objetos que se encuentran aquí a menudo se denominan ***objetos desempaquetados*** o ***sueltos.***

- **objects/pack**
  Los archivos que **almacenan muchos objetos en forma comprimida, junto con los archivos de índice** para permitir el acceso aleatorio, se encuentran en este directorio.
- **objects/info**
  La **información adicional** sobre el objeto almacenado se coloca en este directorio.

- **refs**
  Las referencias se almacenan en los subdirectorios de este directorio. El comando **`git prune`** sabe que debe preservar los objetos accesibles a partir de las referencias que se encuentran aquí y en los subdirectorios.
  - **refs/heads/**
    Contiene objetos de `commit`.
  - **refs/tags**
    Contiene cualquier nombre de objeto.
  - **refs/remotes**
    Contiene los objetos de confirmación de las ramas copiadas desde un repositorio remoto.

- **Archivo HEAD**
  Contiene una referencia al ***branch* en el que se encuentra actualmente.** Esto le dice a Git qué usar como padre de su próximo `commit`.

- **Archivo de configuración config**
  Es el archivo de **configuración principal de Git.** Mantiene opciones específicas para el proyecto, como sus controles remotos, configuraciones de inserción, branches de seguimiento y más. Su configuración se cargará primero desde este archivo, luego desde un archivo ***~ / .gitconfig*** y luego desde un archivo ***/etc/gitconfig***, si existe.

- **hooks/**
  Este directorio contiene *scripts* de shell que se invocan después de los comandos Git correspondientes. Por ejemplo, después de ejecutar un `commit`, Git intentará ejecutar el *script* posterior a la confirmación.

- **Archivo de índice index**
  El index Git se utiliza como *Stage Area* entre el *Working Directory* y el repositorio. Se puede utilizar para **crear un conjunto de cambios que se desean confirmar juntos.** Cuando se crea un `commit`, se confirma lo que se encuentra actualmente en el *index*, no lo que está en su directorio de trabajo. Es un **archivo binario que contiene una lista ordenada de nombres de ruta, cada uno con permisos y el SHA-1 de un objeto blob.**

- **info/**
  La **información adicional sobre el repositorio** se registra en este directorio.

- **logs/**
  Almacena los cambios realizados en las referencias en el repositorio.

  - **logs/refs/heads/**
    Registra todos los cambios realizados en las diferentes puntas de rama.

  - **logs/refs/tags/**
    Registra todos los cambios realizados en las diferentes etiquetas.

- **modules/**
  Contiene los repositorios Git de los submódulos.

Consideraciones

Casi todas las acciones que se realizan en Git solo agregan al registro de cambios el estado en que se encuentra actualmente el proyecto. Es muy difícil lograr que el sistema haga algo que, luego, no sea posible deshacer o que borre los datos.

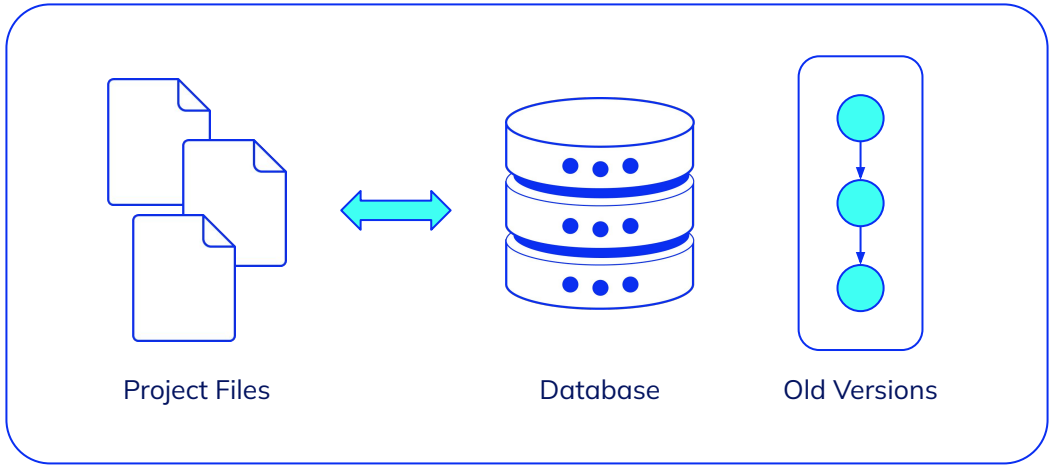
Como en cualquier SCV, se pueden perder o alterar los cambios que aún no se han confirmado pero, después de confirmar un snapshot en Git, es muy difícil que se pierda, especialmente si se realiza un *push* regularmente de la base de datos a otro repositorio.

Tipos de repositorios

Repositorios locales

Estrictamente hablando, este tipo de control de versiones, si bien puede tener una API flexible, es **el menos conveniente porque se presenta en la computadora de cada desarrollador. No ofrece la posibilidad de compartir el código** en caso de que se elija trabajar en modo de Desarrollo Colaborativo.

Repositorio local



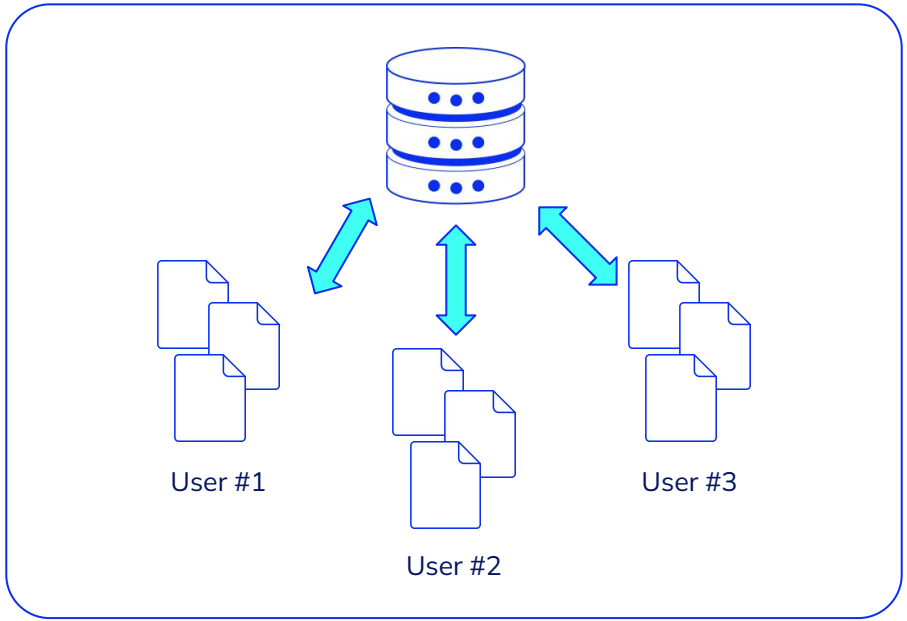
Local version control

Repositorios centralizados

Los sistemas de control de versiones fueron evolucionando para facilitar la colaboración de múltiples desarrolladores en un solo proyecto. **En lugar de almacenar los cambios y versiones en el disco duro, pasaron a guardarlas en un servidor.** Aunque el avance frente a los sistemas de control de versiones locales fue enorme, los sistemas centralizados trajeron nuevos retos como múltiples usuarios trabajando en un mismo archivo al mismo tiempo.

Por otro lado, los sistemas centralizados solían usar como fuente de verdad un repositorio almacenado en algún servidor. **Si se cortaba la conexión a Internet o al servidor no se podía seguir trabajando.**

Repositorio centralizado

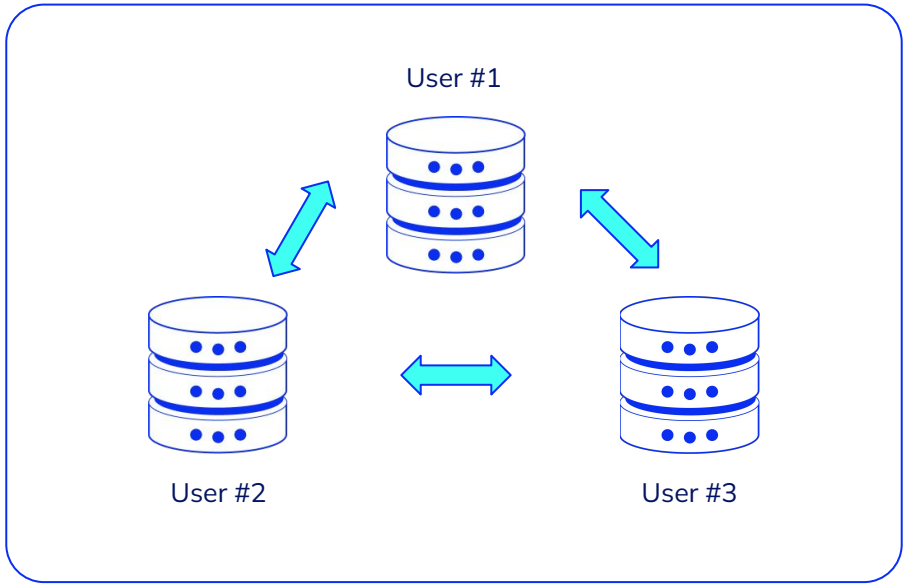


Centralized version control

Repositorios distribuidos

La siguiente generación de sistemas de control de versiones se alejó de la idea de un solo repositorio centralizado y optó por **darle a cada desarrollador una copia local de todo el proyecto.** De esta manera **se construyó una red distribuida de repositorios, en la que cada desarrollador podía trabajar de manera aislada** pero teniendo un mecanismo de resolución de conflictos mucho más elegante que en su versión anterior.

Repositorio distribuido



Distributed version control

Estados de un repositorio Git

Esto es lo más importante que se debe recordar:

Git tiene tres estados principales en los que sus archivos pueden residir: **committed**, **modified** y **staged**.

Committed

Significa que los datos se almacenan de forma segura en su base de datos local.

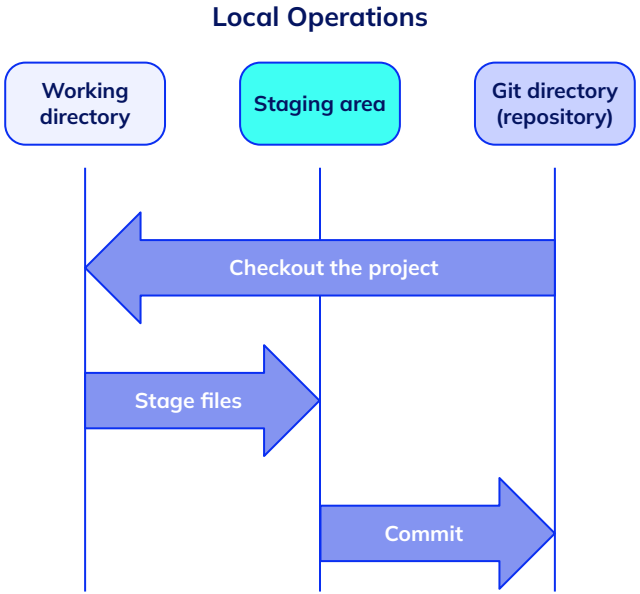
Modified

Significa que se ha cambiado el archivo pero aún no se lo ha confirmado en su base de datos.

Staged

Significa que se ha marcado un archivo modificado en su versión actual para pasar a su próximo snapshot de confirmación.

Esto nos lleva a las **tres secciones principales de un proyecto Git:** el **directorio .git**, el ***working directory*** y el ***staging area***:



¡Sigamos trabajando!