Control de Versiones

¿Qué es el control de versiones?

Los **sistemas de control de versiones** son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo. A medida que los entornos de desarrollo se aceleran, los sistemas de control de versiones ayudan a los equipos de software a trabajar de forma más rápida e inteligente.

Ventajas

- Un completo historial de cambios a largo plazo de todos los archivos.
- Creación de ramas y fusiones.
- Trazabilidad.

Arquitecturas

Podemos clasificar los sistemas de control de versiones atendiendo a la arquitectura utilizada para el almacenamiento del código:

- **Distribuidos:** cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Ejemplos: **Git** y Mercurial.
- **Centralizados:** existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable. Algunos ejemplos son CVS, Subversion(SVN) o Team Foundation Server (TFS).

GIT

Git es un **sistema de control de versiones distribuido** de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

Los Tres Estados

Git tiene tres estados principales en los que se pueden encontrar tus archivos:

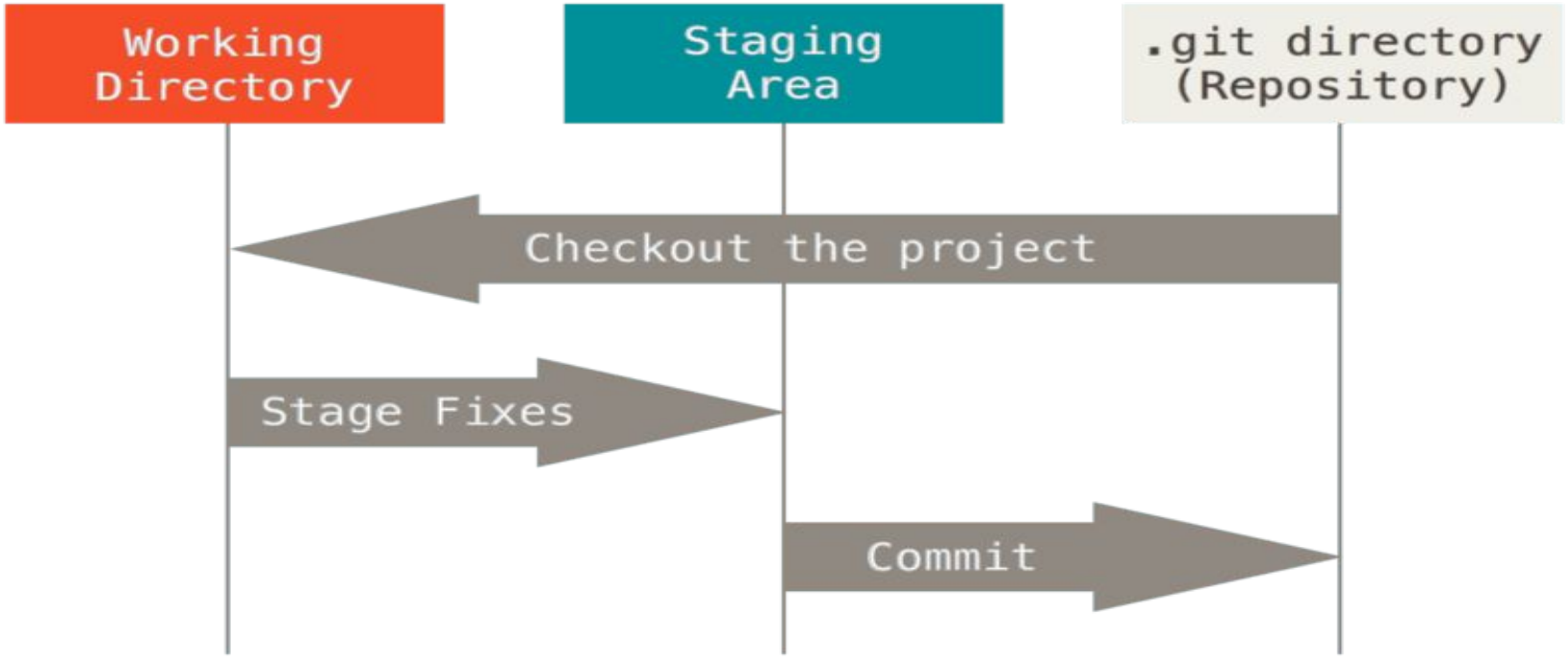
- Modificado (modified): significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
- Preparado (staged): significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.
- Confirmado (committed): significa que los datos están almacenados de manera segura en tu base de datos local.

Los Tres Estados

Esto nos lleva a las tres secciones principales de un proyecto de Git:

- El directorio de Git (Git directory),
- el directorio de trabajo (working directory),
- y el área de preparación (staging area).

Los Tres Estados



Los Tres Estados

- El **directorio de Git** es donde se almacenan los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando clonas un repositorio desde otra computadora.

- El **directorio de trabajo** es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

- El **área de preparación** es un archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice (“index”).

¿Que es un repositorio remoto?

Los **repositorios remotos** son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. De esta manera podrás colaborar con otras personas en la gestión de un proyecto enviando y trayendo datos de estos, cada vez que se necesite.

Ejemplos:

- Github
- Gitlab
- Bitbucket
- etc..

Comandos Básicos

git init: Inicializa un nuevo repositorio de GIT (se realiza una vez)

git clone: Clona un nuevo repositorio de GIT (se realiza una vez)

git add: Se usa para agregar archivos al área de preparación (staging area)

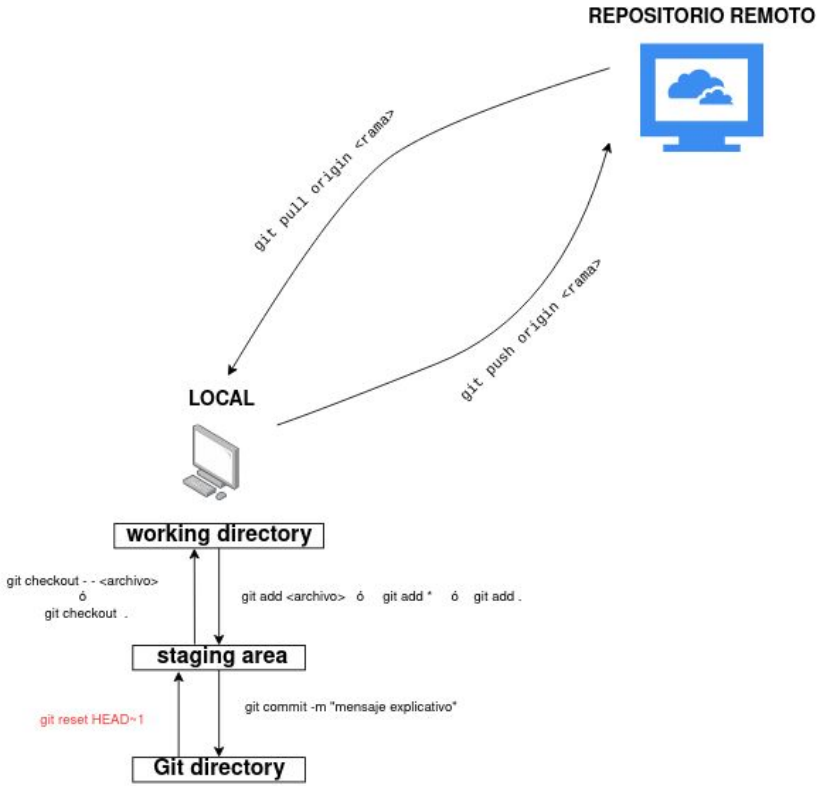
git commit: Creará una instantánea de los cambios y la guardará en el directorio git (Git directory)

git pull origin <rama>: Descarga y fusiona todos los cambios que se han hecho en el repositorio remoto con el directorio de trabajo local

git push origin <rama>: se usa para enviar confirmaciones locales a la rama seleccionada del repositorio remoto.

git status: muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser preparados o confirmados.

Flujo Básico de trabajo



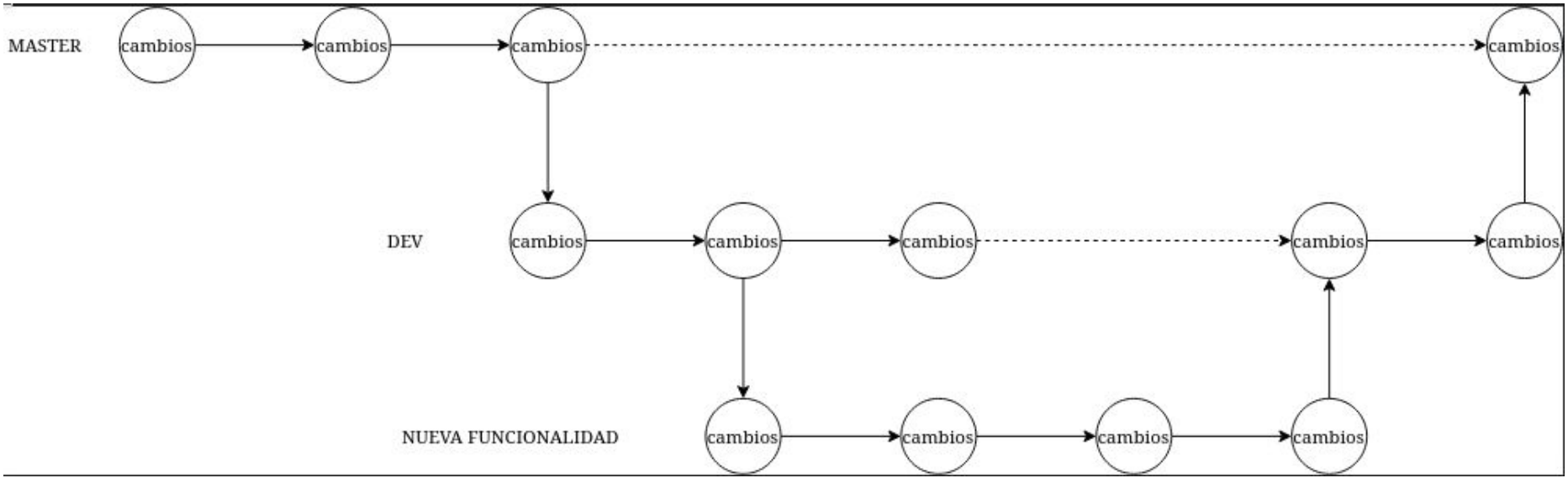
1. Modificas una serie de archivos en tu **directorio de trabajo**.
2. Preparas los archivos, añadiéndolos a tu **área de preparación**.
3. Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu **directorio de Git**.
4. En caso de que lo tengas, subir los cambios al repositorio remoto.

¿Qué son las ramas?

En GIT el flujo de trabajo principal se desarrolla sobre una rama, llamada “master” (o “main” en las nuevas versiones). Esta rama puede bifurcarse tantas veces sea necesario para poder trabajar y desarrollar tareas que no afecten al correcto funcionamiento de la rama principal.

Las nuevas confirmaciones se van a registrar en el historial de la rama actual y son las que hacen que distintas ramas vayan cambiando una de otras.

¿Qué son las ramas?



Comandos Básicos

git branch: Listar todas las ramas presentes en el repositorio.

git branch <nueva_rama>: Crea una nueva rama pero se queda parado en la rama actual.

git branch -d <rama>: Eliminar una rama.

git checkout <rama>: Para cambiar de una rama a otra.

git checkout -b <nueva_rama>: Crea una nueva rama y cambia a la rama recién creada.

git merge <rama>: se usa para fusionar una rama con otra rama activa.

git fetch: descarga los archivos sin hacer el merge (a diferencia del pull, que mergea automáticamente)

Flujo de trabajo

Lo recomendable para el desarrollo es tener 2 ramas principales: master(main) o dev (desarrollo). La primera se usará para tener la versión final del producto software, la segunda va a ser de desarrollo y testing.

Cada desarrollador va a desprender ramas desde la rama de desarrollo para trabajar sobre una nueva funcionalidad y una vez lista se hará un pull request (o merge request) para que un desarrollador experimentado ó un líder técnico revise, apruebe y fusione o no los cambios realizados a la rama de desarrollo.