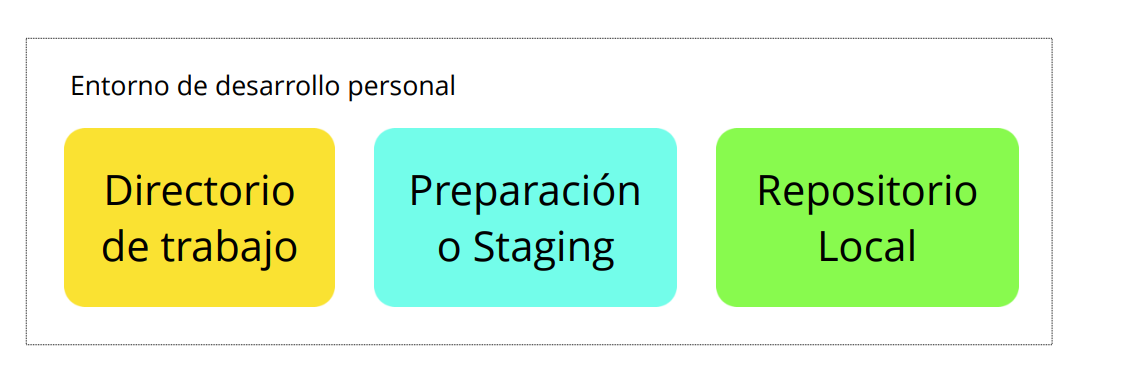
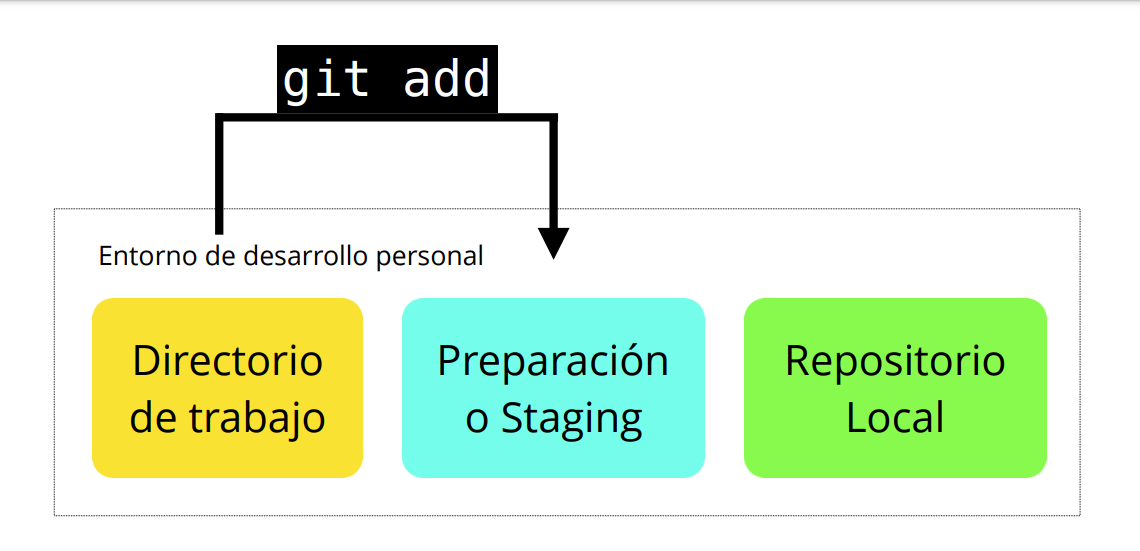
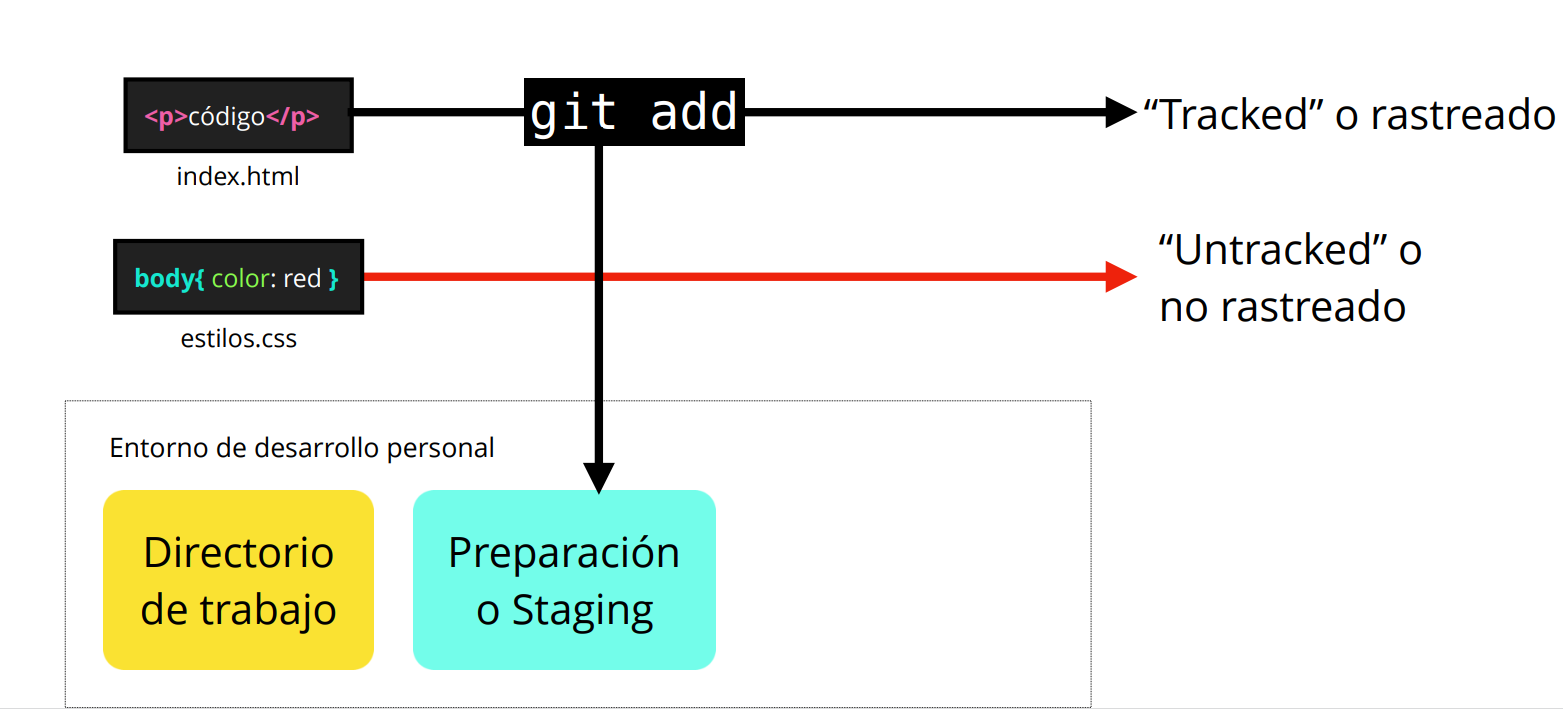
# Comandos de Git

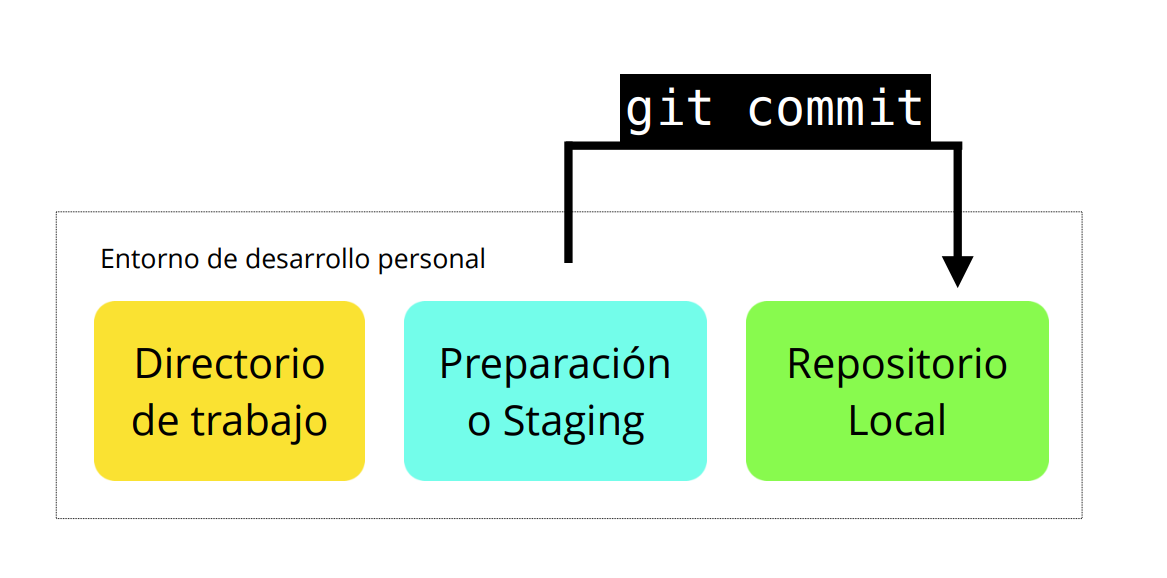
## Introducción

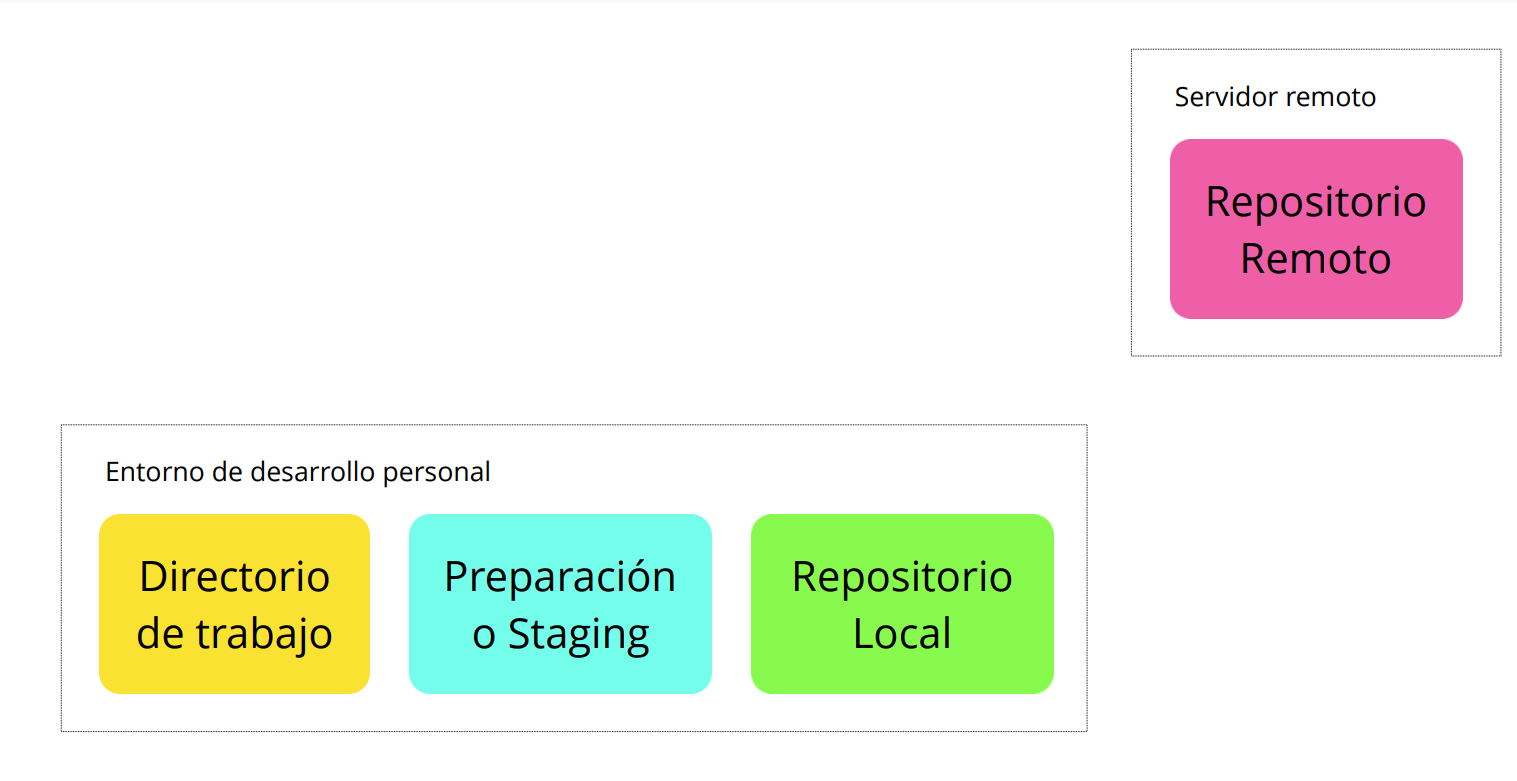
Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. En su lugar GitHub es una forma para alojar proyectos utilizando el sistema de control de versiones Git. GitHub sería la red social de código para los programadores, tu propio curriculum vitae.

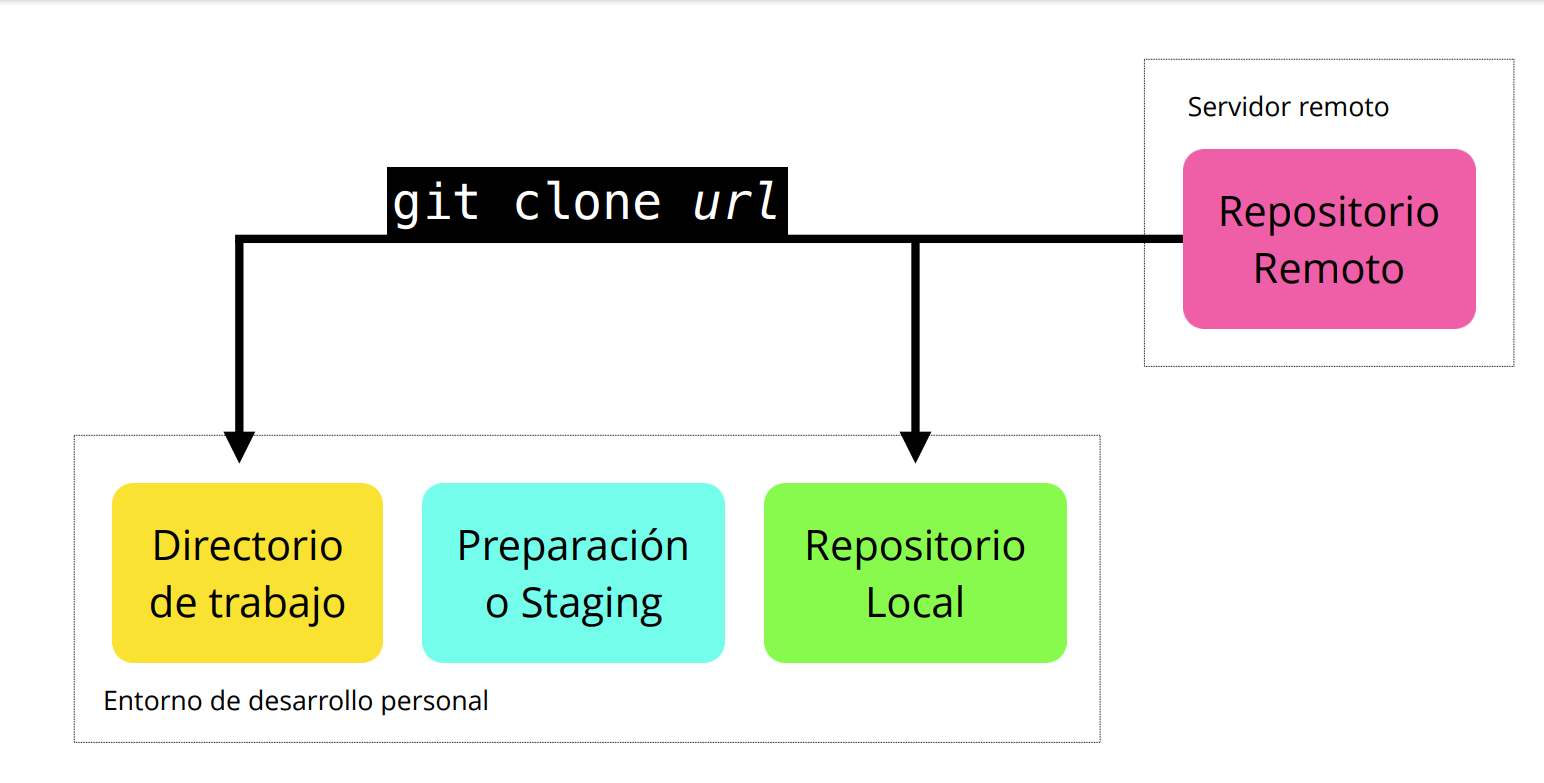


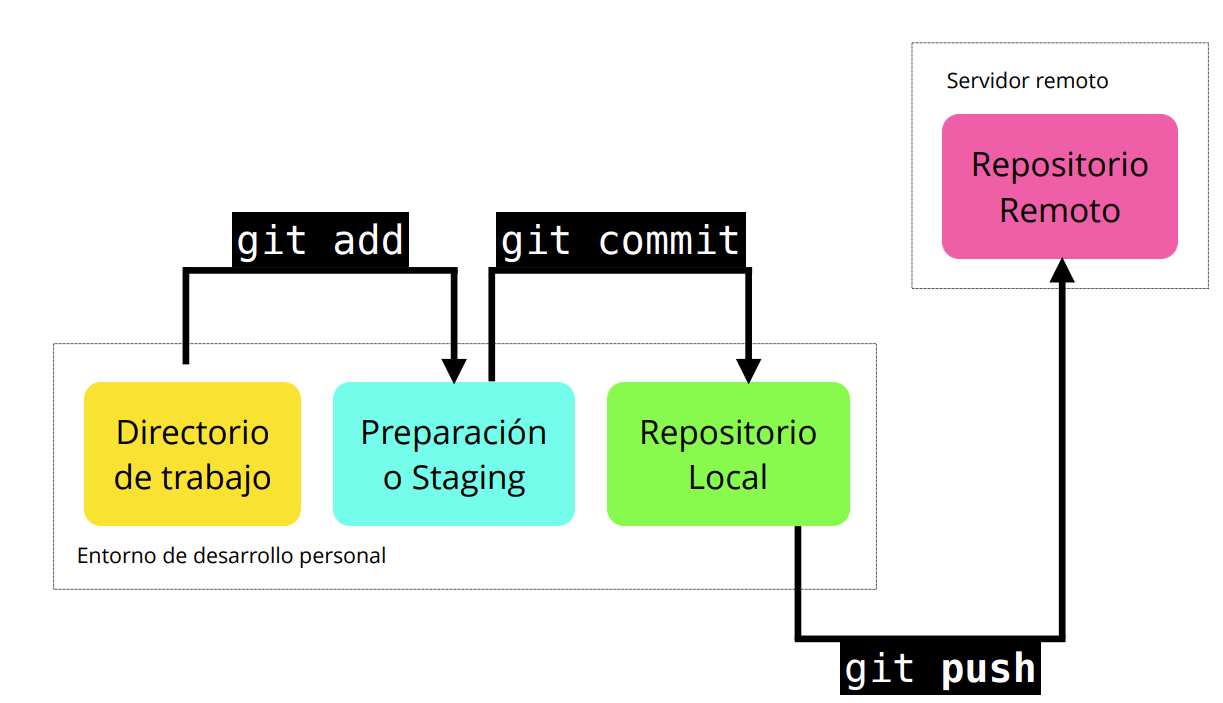


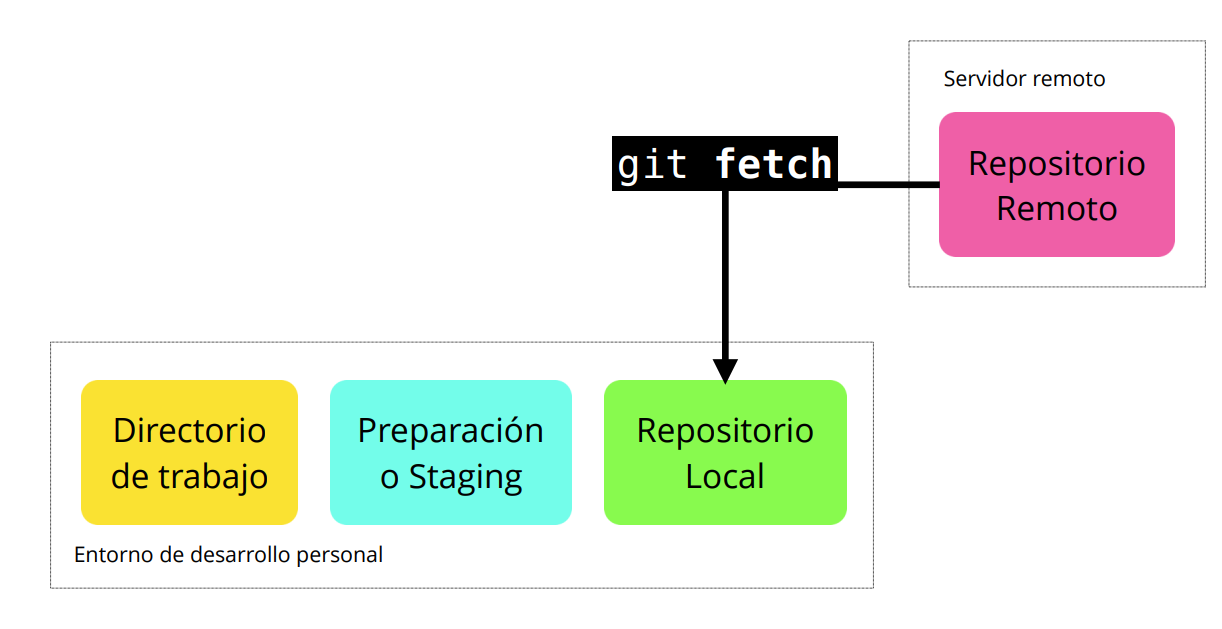


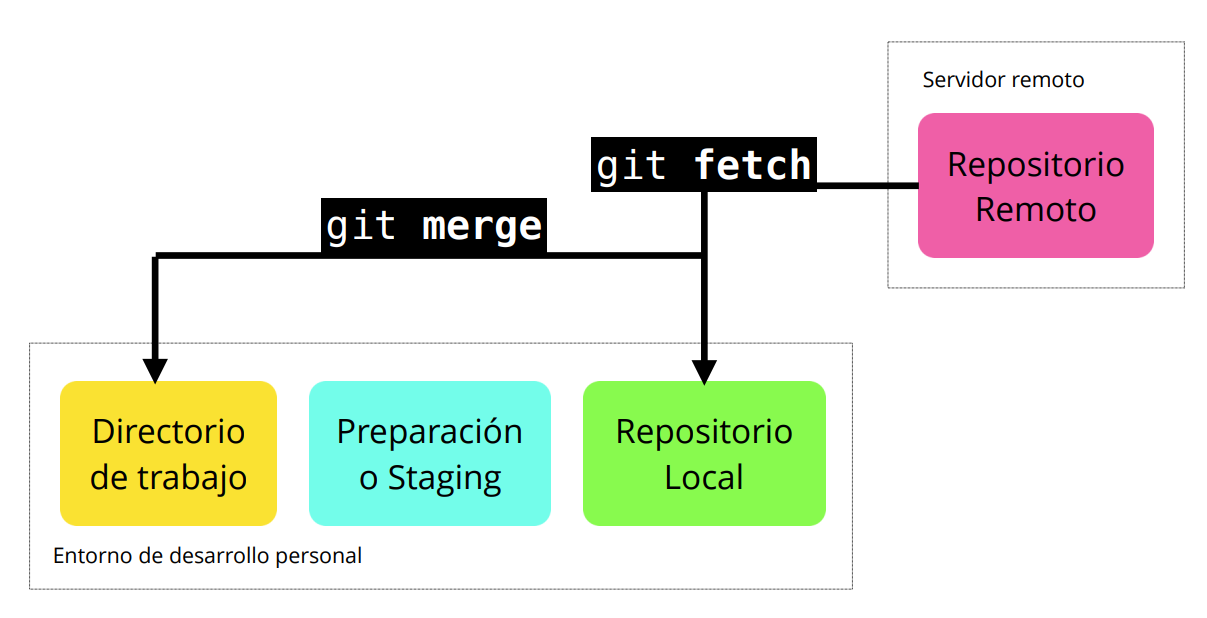


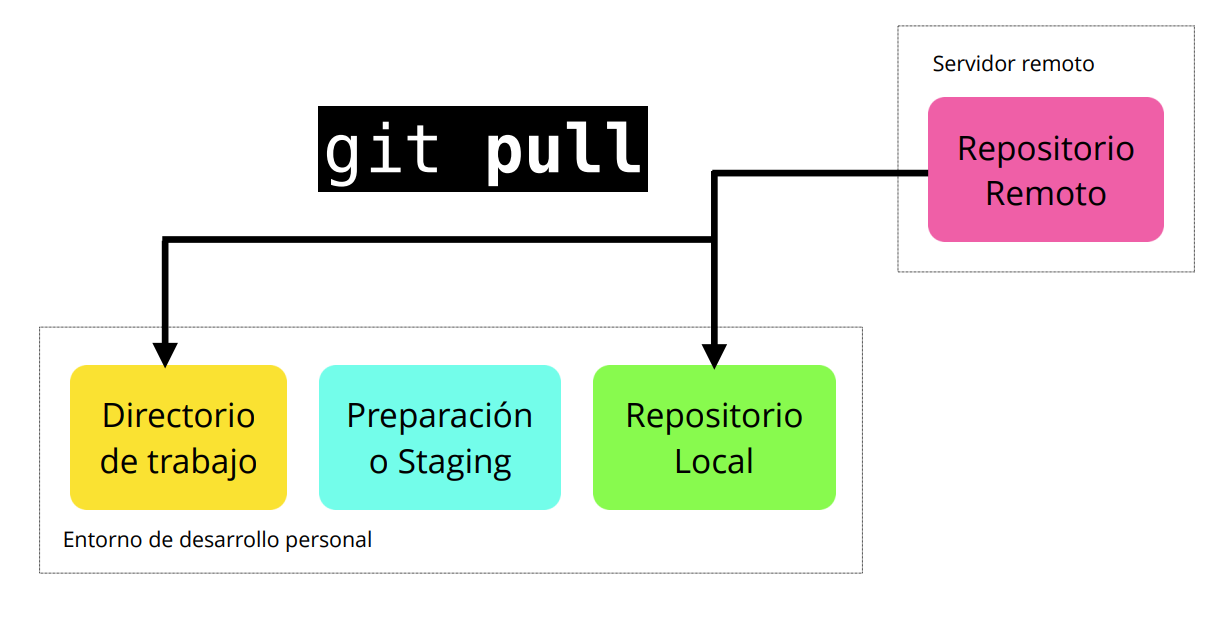


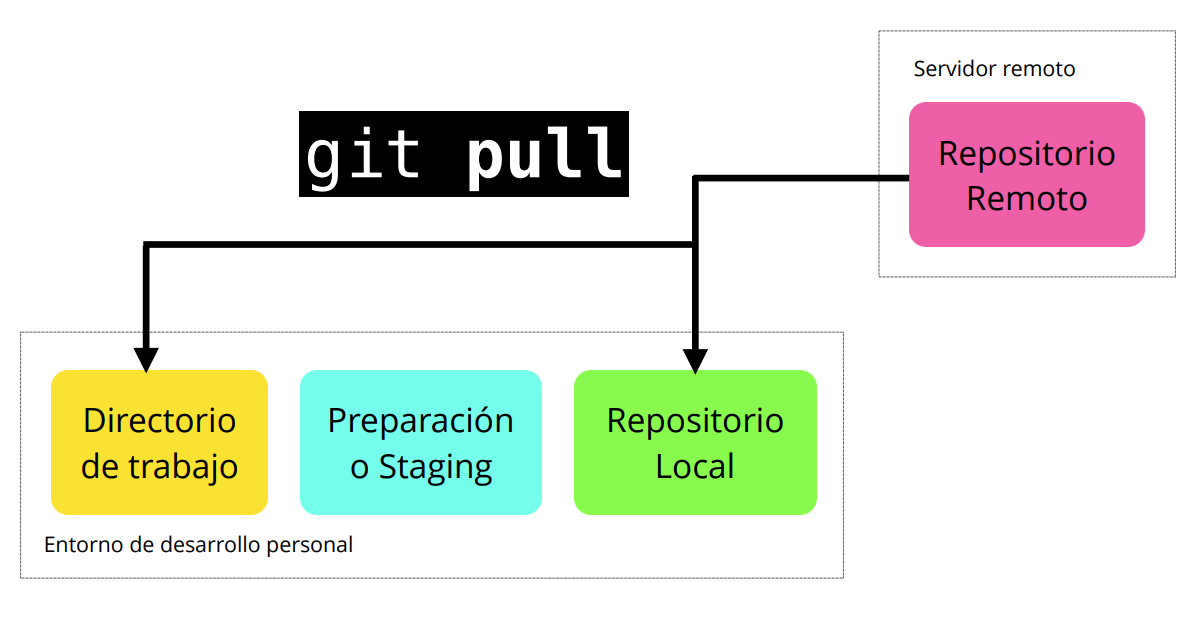


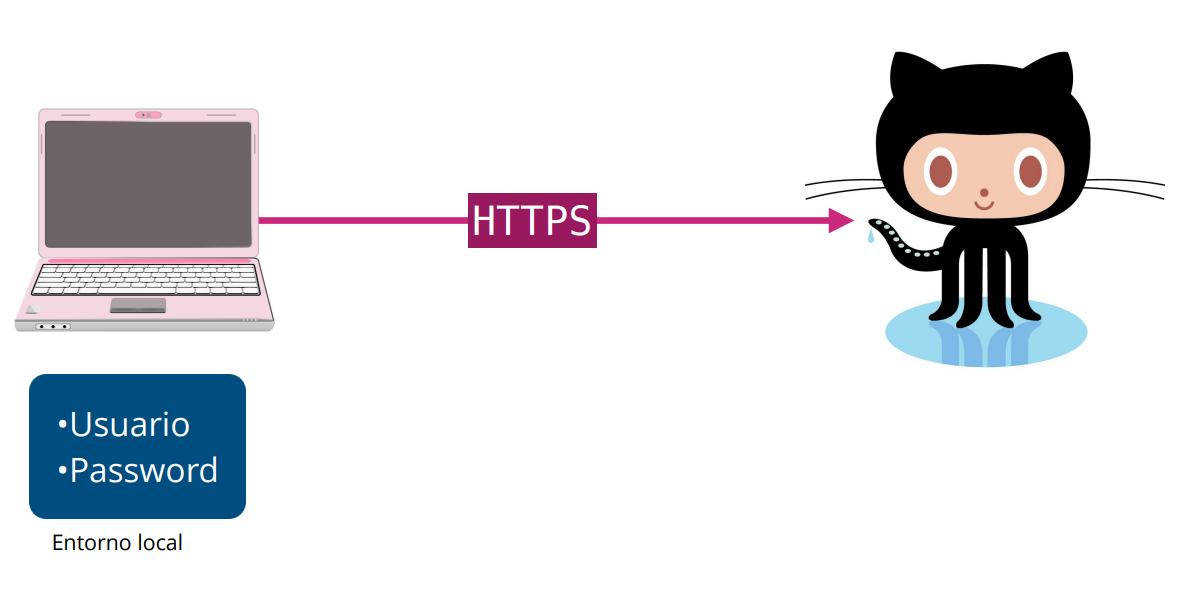


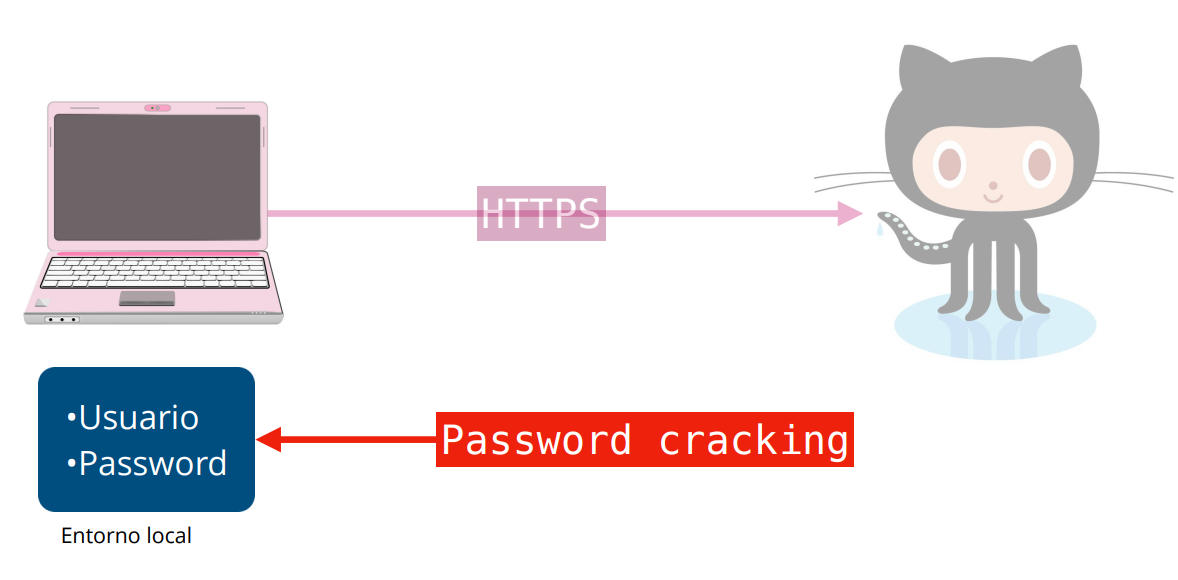




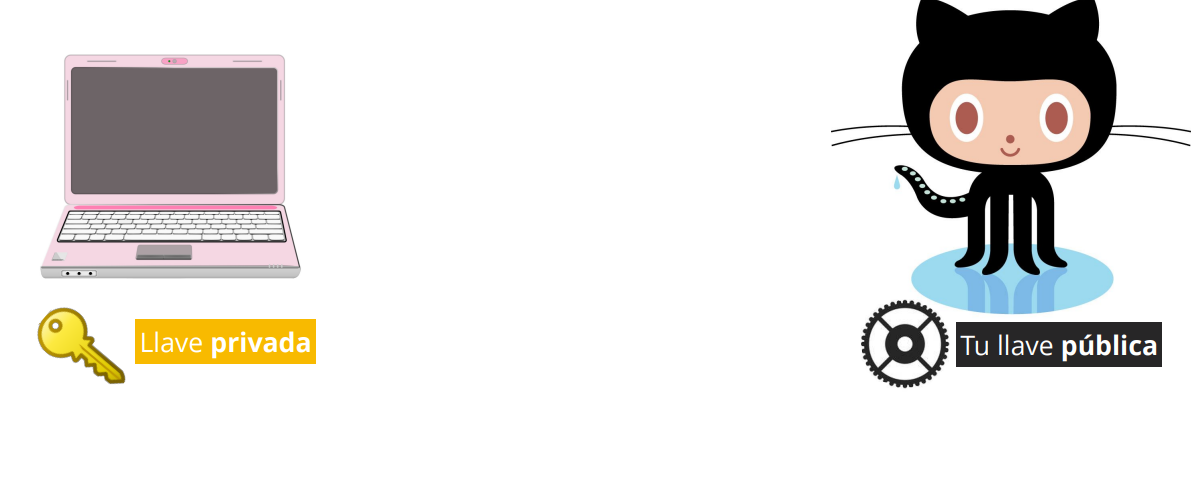


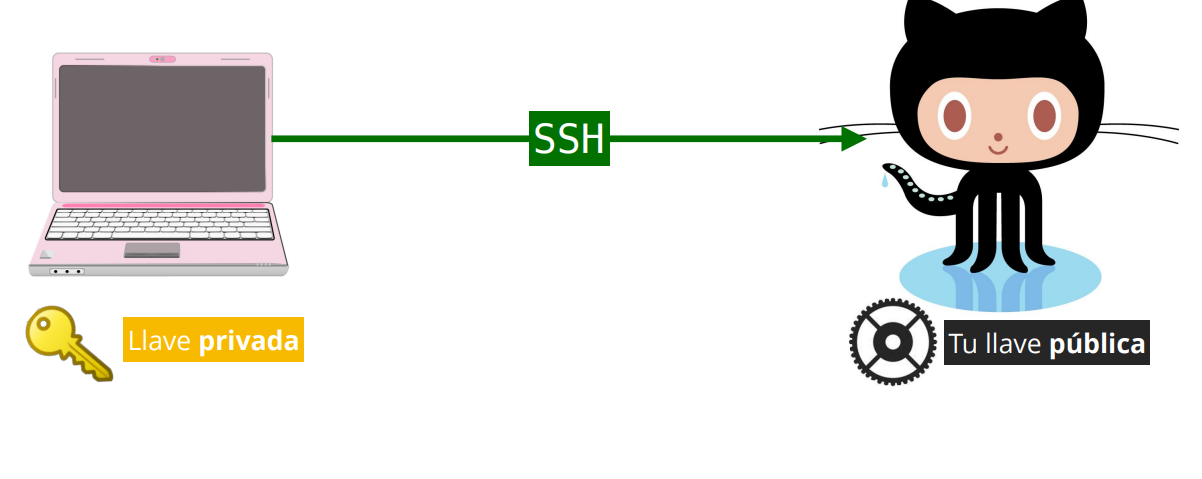


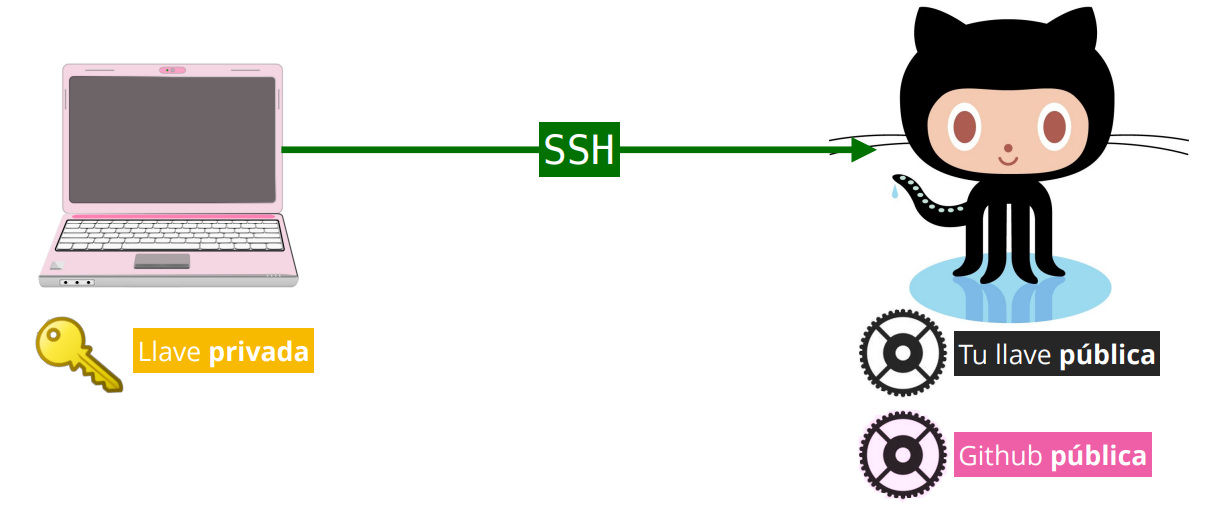


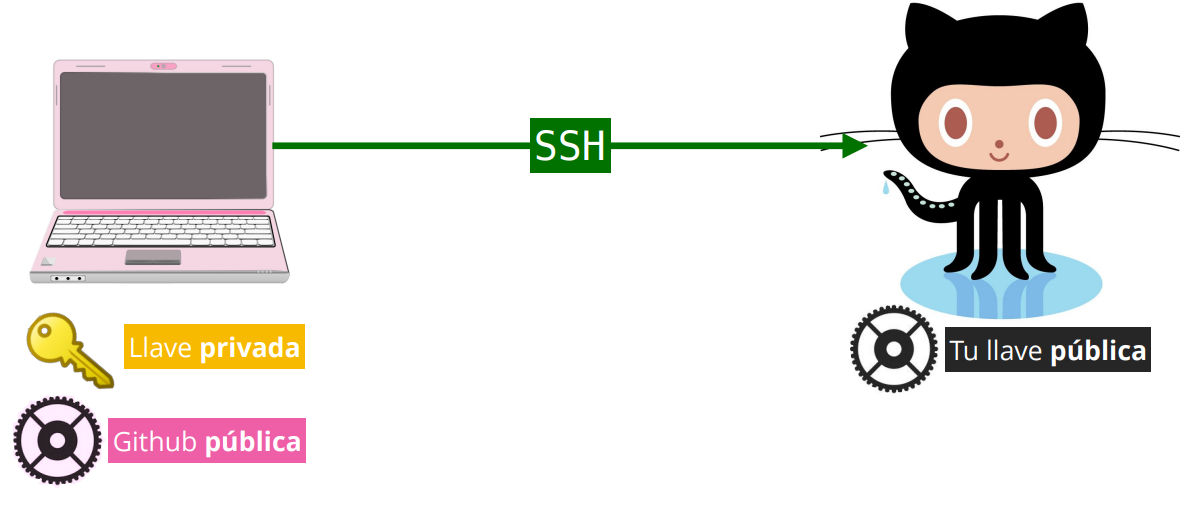


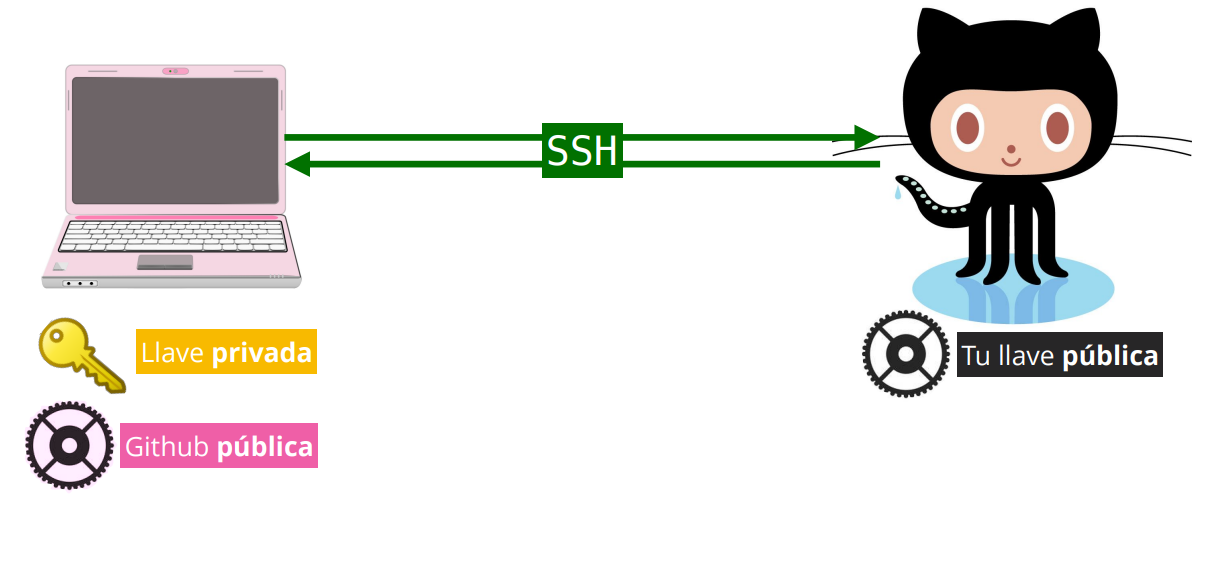












### Inicio de un repositorio (init,add, commit show, log, push, pull)

Un sistema de control de versiones como Git nos ayuda a guardar el historial de cambios y crecimiento de los archivos de nuestro proyecto.

En realidad, los cambios y diferencias entre las versiones de nuestros proyectos pueden tener similitudes, algunas veces los cambios pueden ser solo una palabra o una parte específica de un archivo específico. Git está optimizado para guardar todos estos cambios de forma atómica e incremental, o sea, aplicando cambios sobre los últimos cambios, estos sobre los cambios anteriores y así hasta el inicio de nuestro proyecto.

El comando para iniciar nuestro repositorio, o sea, indicarle a Git que queremos usar su sistema de control de versiones en nuestro proyecto, es

git init.

El comando para que nuestro repositorio sepa de la existencia de un archivo o sus últimos cambios es git add. Este comando no almacena las actualizaciones de forma definitiva, solo las guarda en algo que conocemos como “Staging Area” (no te preocupes, lo entenderemos más adelante).

El comando para almacenar definitivamente todos los cambios que por ahora viven en el staging area es git commit. También podemos guardar un mensaje para recordar muy bien qué cambios hicimos en este commit con el argumento

git commit -m "Mensaje del commit"

Por último, si queremos mandar nuestros commits a un servidor remoto, un lugar donde todos podamos conectar nuestros proyectos, usamos el comando git push.

Git init : inicia el repositorio en git

Git add nombreArchivo : agrega el archivo en el stage para prepararlo para el commit

Git add . : agrega a todo lo archivos dentro del stage

Gir commit –m “mensaje”

Git show muestra todos los cambios que se han realizado

Git log : miestra la historia de los cambios

Git push: enviar todos los cambios al repositorio web

Git pull traer los cambios del repositorio web

### Introducción a la terminal y línea de comandos

Diferencias entre la estructura de archivos de Windows, Mac o Linux.

La ruta principal en Windows es C:\, en UNIX es solo /.

Windows no hace diferencia entre mayúsculas y minúsculas pero UNIX sí.

Recuerda que GitBash usa la ruta /c para dirigirse a C:\ (o /d para dirigirse a D:\) en Windows. Por lo tanto, la ruta del usuario con el que estás trabajando es /c/Users/Nombre de tu usuario

Comandos básicos en la terminal:

pwd: Nos muestra la ruta de carpetas en la que te encuentras ahora mismo.

mkdir: Nos permite crear carpetas (por ejemplo, mkdir Carpeta-Importante).

touch: Nos permite crear archivos (por ejemplo, touch archivo.txt).

rm: Nos permite borrar un archivo o carpeta (por ejemplo, rm archivo.txt). Mucho cuidado con este comando, puedes borrar todo tu disco duro.

cat: Ver el contenido de un archivo (por ejemplo, cat nombre-archivo.txt).

ls: Nos permite cambiar ver los archivos de la carpeta donde estamos ahora mismo. Podemos usar uno o más argumentos para ver más información sobre estos archivos (los argumentos pueden ser -- + el nombre del argumento o - + una sola letra o shortcut por cada argumento).

- ls -a: Mostrar todos los archivos, incluso los ocultos.

- ls -l: Ver todos los archivos como una lista.

cd: Nos permite navegar entre carpetas.

- cd /: Ir a la ruta principal:

- cd o cd ~: Ir a la ruta de tu usuario

- cd carpeta/subcarpeta: Navegar a una ruta dentro de la carpeta donde estamos ahora mismo.

- cd .. (cd + dos puntos): Regresar una carpeta hacia atrás.

- Si quieres referirte al directorio en el que te encuentras ahora mismo puedes usar cd . (cd + un punto).

history: Ver los últimos comandos que ejecutamos y un número especial con el que podemos repetir su ejecución.

! + número: Ejecutar algún comando con el número que nos muestra el comando history (por ejemplo, !72).

clear: Para limpiar la terminal. También podemos usar los atajos de teclado Ctrl + L o Command + L.

Todos estos comandos tiene una función de autocompletado, o sea, puedes escribir la primera parte y presionar la tecla Tab para que la terminal nos muestre todas las posibles carpetas o comandos que podemos ejecutar. Si presionas la tecla Arriba puedes ver el último comando que ejecutamos.

Recuerda que podemos descubrir todos los argumentos de un comando con el argumento --help (por ejemplo, cat --help).

### Configuración del ambiente de trabajo antes de hacer un commit

Recuerda que Git está optimizado para trabajar en equipo, por lo tanto, debemos darle un poco de información sobre nosotros. No debemos hacerlo todas las veces que ejecutamos un comando, basta con ejecutar solo una sola vez los siguientes comandos con tu información:

git config --global user.email [tu@email.com](mailto:tu@email.com)

git config --global user.name "Tu Nombre"