



git



CONTROL DE VERSIONES CON GIT - GITHUB

Entornos de desarrollo

Índice



- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- 7.- Material de ayuda

Índice

- **1.- ¿Qué es GIT?**
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- 7.- Material de ayuda

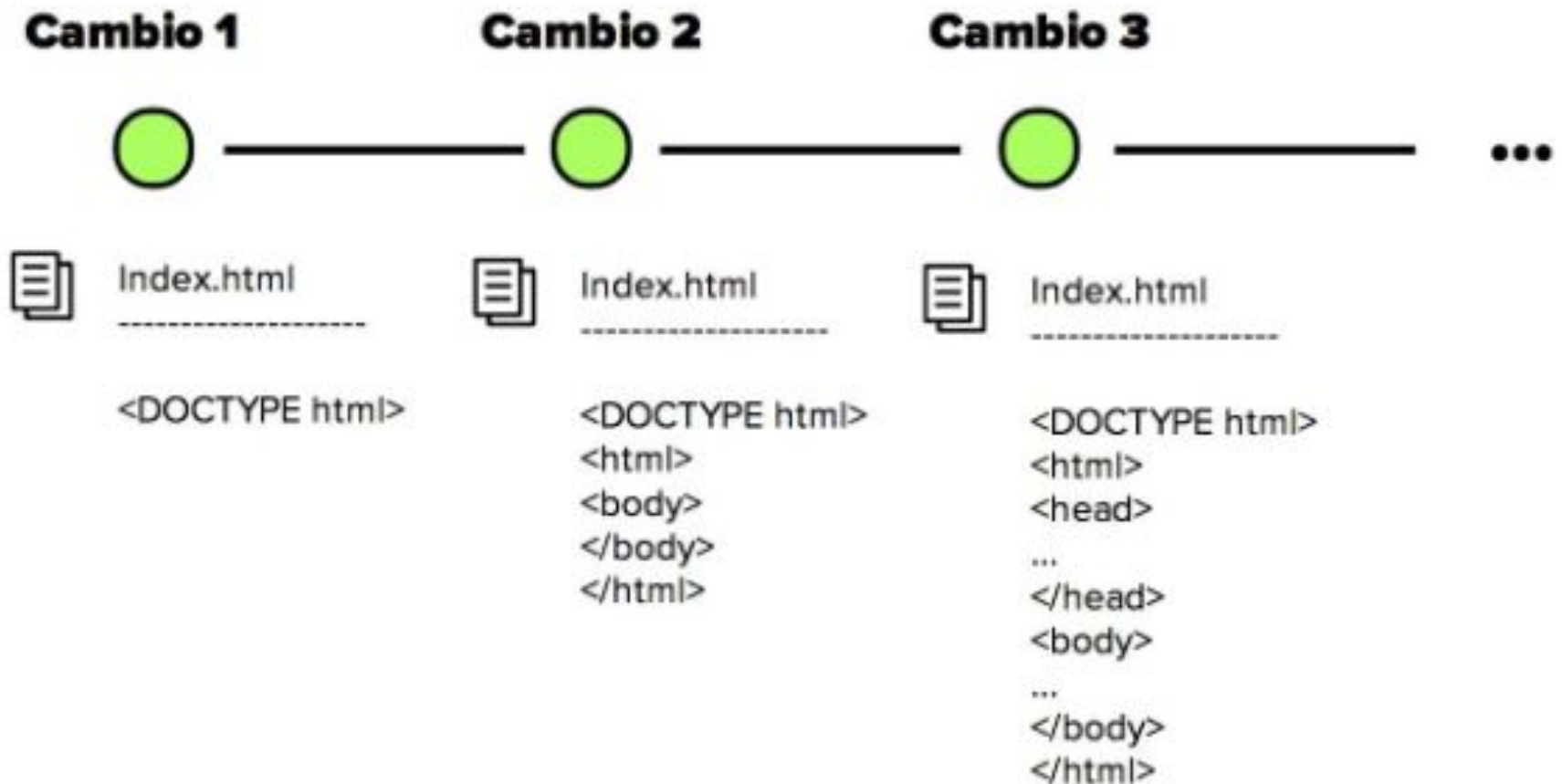
1.- ¿Qué es GIT?

- Es un software controlador de versiones.
- **Detecta todos los cambios que se ejecutan sobre un archivo o carpeta.**
- Cada cambio que hagas en un directorio, GIT se da cuenta y lo registra.
- Imaginemos un archivo que se modifica constantemente:

1.- ¿Qué es GIT?

- Index.html
- **Hacemos un 1° cambio:** Le agregamos una etiqueta `<doctype>`
 - * git está atento y lo registra *
- **2° cambio:** Agregamos etiquetas `<head>` y `<body>`
 - * git lo registra nuevamente. Tenemos 2 cambios. *
- **3° cambio:** Agregamos contenido dentro de las etiquetas.
 - * git siempre lo registra. Tenemos 3 cambios. *

1.- ¿Qué es GIT?



1.- ¿Qué es GIT?

- Cada vez que haces un cambio en tu código, **GIT registra los cambios y los guarda.**
- ¿Cómo sabe en qué momento guardar los cambios? Tú le avisas.
- Es como cuando haces un “Save Game” en un videojuego. Sabes que ya ocurrieron varios cambios porque avanzaste, conseguiste nuevas armas, venciste jefes y necesitas salvar. Las veces que quieras y sean necesarias.



1.- ¿Qué es GIT?

- Ahora bien, ¿qué pasaría si te dijera que el juego te permitiera ver todos tus momentos salvados del juego? Podrías ubicarte en cualquier momento que gustes de la historia.
- Lo mismo ocurre con tus proyectos.
- Irás avanzando, generando cambios (la forma en como “salvas” tus avances) y posteriormente, podrás revisar todo tu proyecto.
- A este conjunto de cambios (que a partir de este momento les llamaremos “**commits**”) se le conoce como **repositorio**.

1.- ¿Qué es GIT?

- **¿Cómo es el proceso técnico? ¿Si hago 10 cambios, GIT guarda 10 veces todos mis archivos? ¿No estaría generando miles de archivos?**
- Este concepto y pregunta es muy normal. **GIT guarda los cambios que haces, no hace copias de los archivos.**
- GIT no clona 10 veces tu proyecto cada vez que salvas, sino que registra cuáles fueron las líneas que modificaste, las encapsula en el registro llamado “commit” y con esto, te permite disfrutar de un historial de avances de tu proyecto.
- Al final, se podría considerar que es **un registro de cambios.**

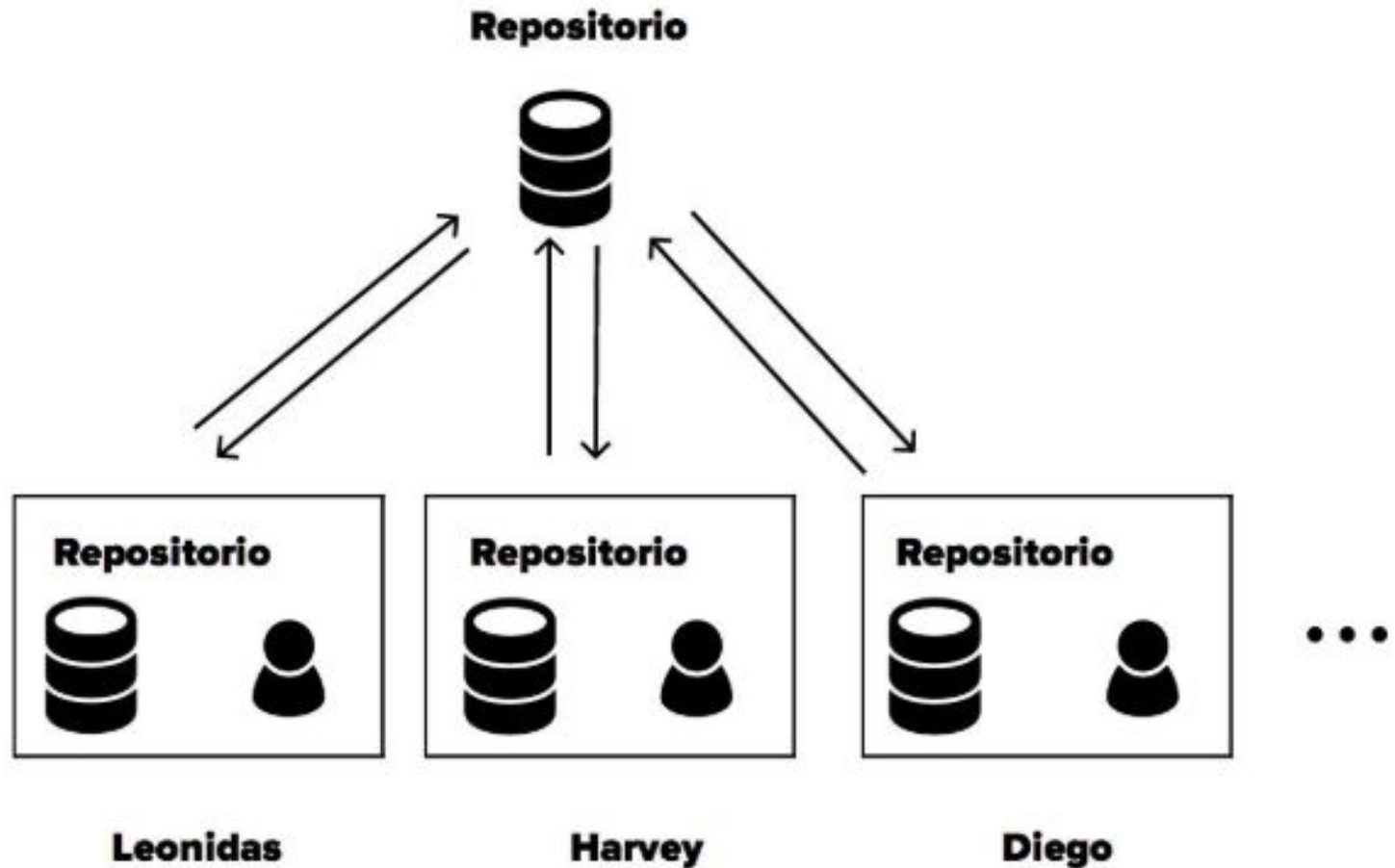
Índice

- 1.- ¿Qué es GIT?
- **2.- Características de GIT**
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- 7.- Material de ayuda

2.- Características de GIT

- **A) Es un sistema de control de versiones distribuido.**
 - Con esto, nos referimos a que GIT clona los proyectos para que cada persona ó miembro de un equipo tenga una copia exacta y completa de todo el código, historial y las personas que estuvieron involucradas.
 - El registro encapsulado de todos los cambios de estos elementos se le conoce como **repositorio**, mencionado anteriormente.
 - Si se llegase a perder el repositorio original, no habría mucho drama porque es probable que existan personas que tienen un clón. De ahí, se puede partir sin problema.

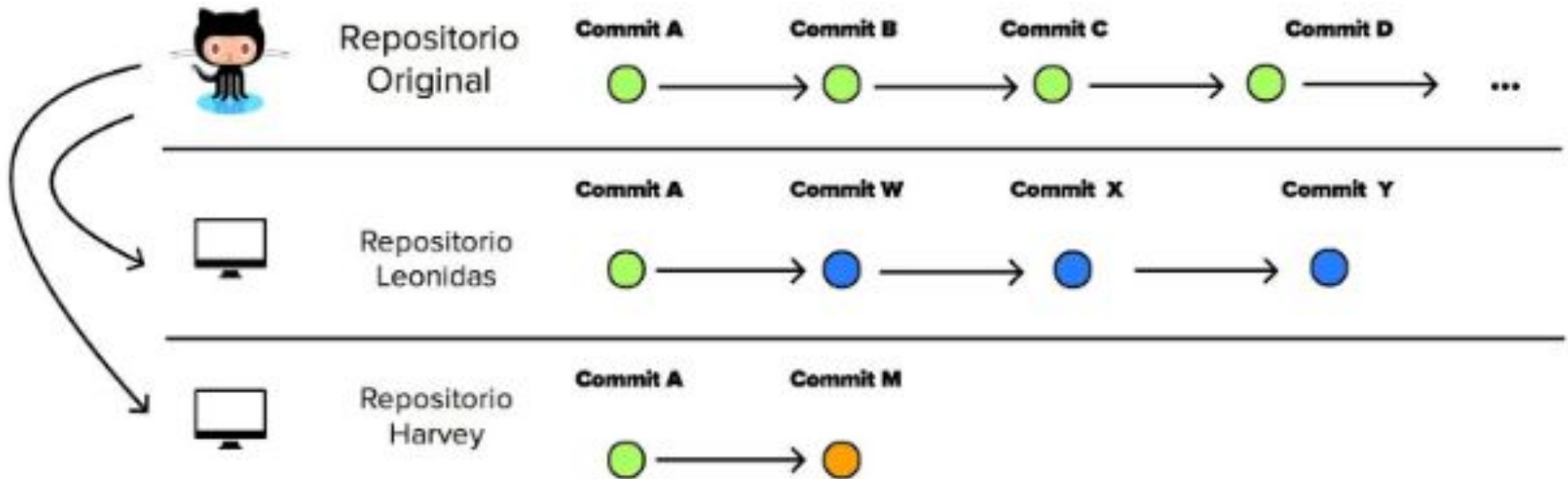
2.- Características de GIT



2.- Características de GIT

- Básicamente, **cada persona (o grupo de personas) mantienen y trabajan sus propios repositorios**, derivados del principal, el cual, con toda la flexibilidad, se pueden fusionar y compartir avances.
- Cada repositorio es independiente. Y aunque al final buscan establecer un repositorio principal para tener orden, ellos tienen sus repositorios totalmente libres para trabajarlos. Si quieren sincronizar, muy bien, si no, no hay lío.

2.- Características de GIT



2.- Características de GIT

- ▣ **b) Es Open Source.**

- ▣ GIT es gratuito, puedes instalarlo en cualquier ordenador o servidor.

- ▣ **c) Puedes utilizarlo offline**

- ▣ Si vas en un avión, puedes seguir trabajando en tu proyecto, en local, para posteriormente cuando te conectes a Internet, puedas subirlo al repositorio principal.

2.- Características de GIT

□ **Conceptos generales**

□ Antes de nada, vamos explicar conceptos generales de los sistemas de control de versiones:

□ **Branches**

- Branches o ramas, son herramientas proporcionadas por los sistemas de control de versiones para poder trabajar paralelamente en distintas versiones del código sin interferir entre sí. Suelen utilizarse ramas de largo recorrido para las principales versiones de la aplicación (master para la aplicación en producción, staging para aplicación en pruebas...) y ramas puntuales para el desarrollo independiente de distintas funcionalidades.

□ **Commits**

- Los commits son los puntos de guardado que se van realizando sobre las ramas en los que fijamos los cambios realizados en el código. Podremos volver a ellos para recuperar el estado del código en un determinado momento.

□ **Merge**

- Un merge es la acciones para juntar una rama sobre otra y unificar los cambios. Por ejemplo para añadir una funcionalidad sobre el proyecto principal. Estas acciones son especialmente críticas cuando trabajan varias personas a la vez, y dependerá de la calidad del sistema de control de versiones que surjan más o menos conflictos a resolver para unificar el código.

□ **Sincronización (Pull/Push)**

- Son los procesos para sincronizar el estado de nuestros sistemas locales con el servidor de código principal, tanto para subir los cambios locales como para descargar las últimas actualizaciones.

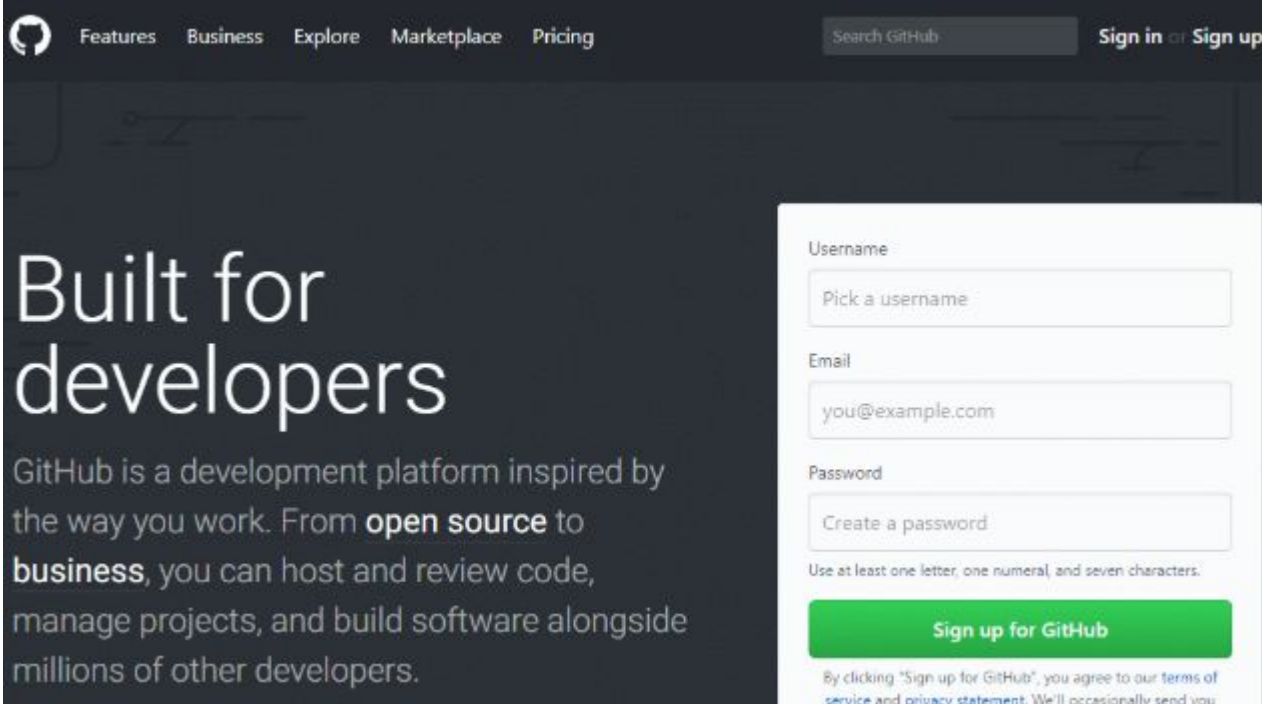
Índice



- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- **3.- ¿Qué es GITHUB?**
- 4.- Instalación
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- 7.- Material de ayuda

3.- ¿Qué es GITHUB?

- GitHub es una compañía que ofrece un servicio de **hosting de repositorios** almacenados en la nube.



The screenshot shows the GitHub homepage with a dark background. At the top, there is a navigation bar with links for Features, Business, Explore, Marketplace, and Pricing. A search bar and links for Sign in or Sign up are also present. The main heading reads 'Built for developers'. Below this, a paragraph describes GitHub as a development platform inspired by the way you work, supporting open source and business projects. On the right side, there is a white sign-up form with fields for Username (with a placeholder 'Pick a username'), Email (with a placeholder 'you@example.com'), and Password (with a placeholder 'Create a password'). A green button labeled 'Sign up for GitHub' is at the bottom of the form. Below the button, a small disclaimer states: 'By clicking "Sign up for GitHub", you agree to our terms of service and privacy statement. We'll occasionally send you'.

Features Business Explore Marketplace Pricing Search GitHub Sign in or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username
Pick a username

Email
you@example.com

Password
Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you

Índice



- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- **4.- Instalación**
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- 7.- Material de ayuda

4.- Instalación

- Instalar git para windows:
 - <https://gitforwindows.org>
- Registrarse en git hub:
 - <https://github.com/>
- Podemos usar cmd en vez de git bash.

Índice



- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- **5.- Repositorio Local**
- 6.- Repositorio Remoto
- 7.- Material de ayuda

5.- Repositorio Local

- Podemos crear un repositorio de dos formas:
 - Botón derecho, seleccionar **git bash here**.
 - Desde la ruta de la carpeta dentro de git bash, escribir **git init**.
 - (crea una carpeta oculta con extensión .git)
- Si queremos cambiar directorio donde crear el repositorio:
 - `pwd` - listar la dir donde te encuentras
 - `cd ruta`
 - `logout`

5.- Repositorio Local

- Configuración inicial desde consola de git:
 - //para inicializar quien realiza los cambios del proyecto
 - `git config --global user.name "NOMBRE"`
 - `git config --global user.email "CORREO"`

5.- Repositorio Local

- Como el famoso "help" en tantas consolas de comandos, en esta consola será "git".
- Ya podemos ver el estado de nuestro proyecto y ficheros mediante:
 - `git status`
 - Nos creamos dentro de nuestro repositorio (carpeta) un fichero `EDD_misiniciales.txt`

5.- Repositorio Local

- Si hacemos `git status` ahora, observamos que nos indica que falta por hacer commit (siempre con el parámetro `-m` para dejar un comentario).
 - `git commit -m "Crear archivo EDD_misiniciales.txt"`
 - `//` también podemos hacerlo sin `-m`
- Si ahora modificamos dicho fichero y volvemos a ver su estado, nos debe de mostrar que ha sido modificado.
 - Deberemos de hacer:
 - `git add --all`
 - `git commit -m "modificaciones"`

5.- Repositorio Local

- **git log** nos permitirá observar los commit realizados y sus etiquetas.
- Posteriormente, si volvemos a realizar un cambio, si usamos el comando "git diff nombreArchivo" nos muestra qué ha cambiado exactamente del código del archivo.

5.- Repositorio Local

□ .gitignore

- Un fichero muy importante a crear es .gitignore, cualquier nombre de fichero que escribamos dentro será ignorado para futuros commit.

□ Ramas:

- **git branch** => nos muestra las diferentes ramas/caminos en la que se encuentra la versión de nuestro código
 - **git branch** nombreRama => Crea una rama nueva
 - **git checkout** nombreRama => cambiar de rama

Índice



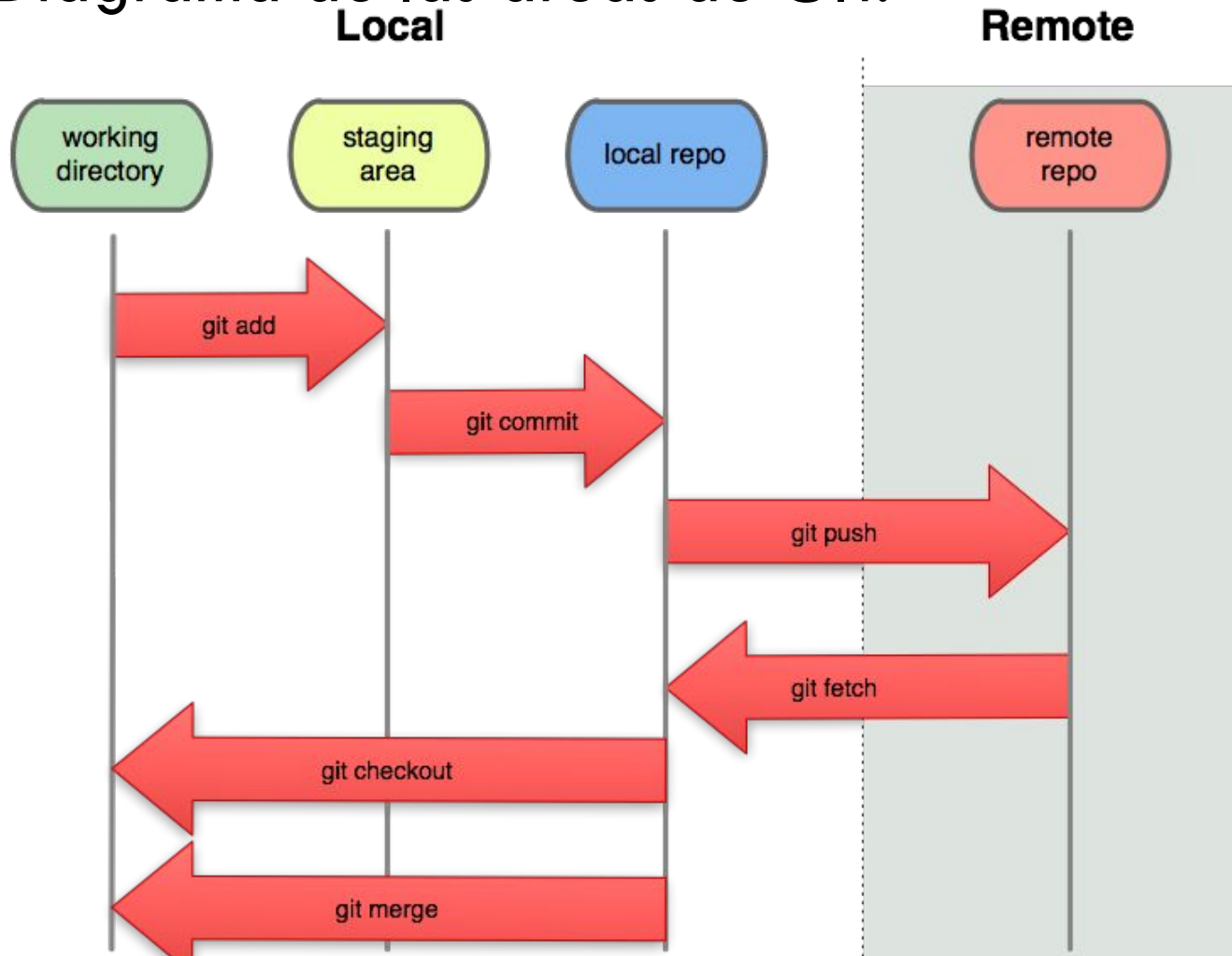
- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- 5.- Repositorio Local
- **6.- Repositorio Remoto**
- 7.- Material de ayuda

6.- Repositorio Remoto

- `git remote add origin «dirección https de tu espacio de github»`
 - `git push -u origin master`
- Importancia de README.txt en github
- Una vez subido nuestro repositorio, podemos clonarlo a otra dirección con:
 - `git clone direccionDelRep.com`

6.- Repositorio Remoto

- Diagrama de las áreas de GIT.



Índice



- 1.- ¿Qué es GIT?
- 2.- Características de GIT
- 3.- ¿Qué es GITHUB?
- 4.- Instalación
- 5.- Repositorio Local
- 6.- Repositorio Remoto
- **7.- Material de ayuda**

7.- Material de ayuda

- Tutorial GIT 23 capítulos
- Tutorial GitHUB

7.- Material de ayuda

□ Resumen de comandos básicos

□ Iniciar un repositorio de git en una carpeta:

■ \$ git init

□ Descargar repositorio de un servidor:

■ \$ git clone <URL>[directory]

□ Comprobar el estado de los ficheros: nuevos, con cambios o pendientes de commit (stage):

■ \$ git status

□ Añadir un fichero o todos (.) al área de stage:

■ \$ git add <file> \$ git add .

□ Guardar los cambios en el repositorio y dejar descripción:

■ \$ git commit -m '<message>'

7.- Material de ayuda

□ Resumen de comandos básicos

- Log de los commits realizados para ver su información (y SHA):
 - \$ git log
- Detalle de un commit:
 - \$ git show <SHA>
- Comparar las diferencias de los cambios pendientes de commit con los del último commit: \$
 - git diff
- Ver ramas:
 - \$ git branch \$ git branch -a
- Cambiar de rama:
 - \$ git checkout <branch>
- Crear una rama:
 - \$ git branch <branch>
- Crear y cambiar a una rama:
 - \$ git checkout -b <branch>

7.- Material de ayuda

□ Resumen de comandos básicos

□ Visor gráfico por defecto con git:

- \$ gitk
- \$ gitk -all

□ Mezclar los commits de una rama con la que estamos:

- \$ git merge [--no-ff] <branch>

□ Mezcla una rama sobre la que estamos, reescribiendo los commits de la rama en la que estamos después del resto (peligroso):

- \$ git rebase <branch>

□ Normalmente usado para ponerse al día con otro rama, siempre que los commits no se hayan compartido ya:

- \$ git rebase master

7.- Material de ayuda

□ Comandos trabajando en remoto

□ Añadir un repositorio remoto (si hemos usado clone, el repositorio remoto **origin** será por defecto el repositorio clonado):

■ \$ git remote add <remote_name> <URL>

□ Subir una rama al repositorio remoto:

■ \$ git push -u <remote_name> <branch>

■ Ejemplo típico:

■ \$ git push -u origin master

■ Con -u

,recordará la rama y el remoto, la siguiente vez puede hacerse git push a secas

□ Recoger una rama del repositorio, por defecto hace un merge y rebase para mezclar los cambios:

□ \$ git pull

7.- Material de ayuda

□ **Cuando todo se rompe.**

□ Arreglar el último commit:

■ `$ git commit --amend -m '<new_message>'`

□ Dejar un fichero como en el último commit:

■ `$ git checkout <file>`

□ Quitar un fichero de stage:

■ `$ git reset HEAD <file>`

□ Revertir un commit (aplicar un nuevo commit inverso a otro):

■ `$ git revert <SHA>`

□ Volver a un estado anterior:

■ `$ git reset <SHA | branch>`

■ Por defecto (`--soft`) mantiene los cambios de los commits eliminados, pero sin aplicar, si queremos eliminarlos totalmente:

■ `$ git reset --hard <SHA | branch>`