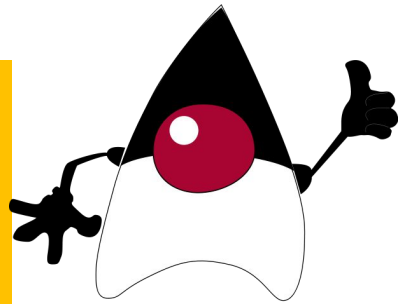
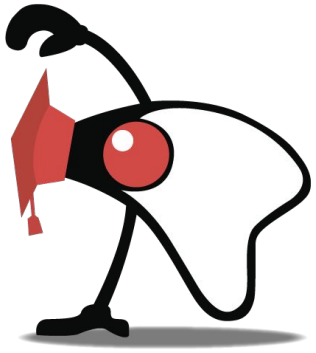
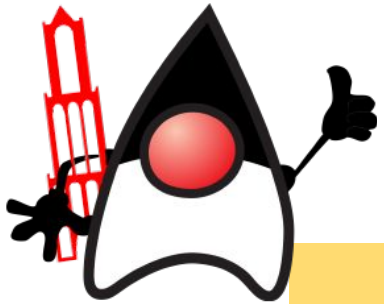
Curso FullStack Python

Codo a Codo 4.0

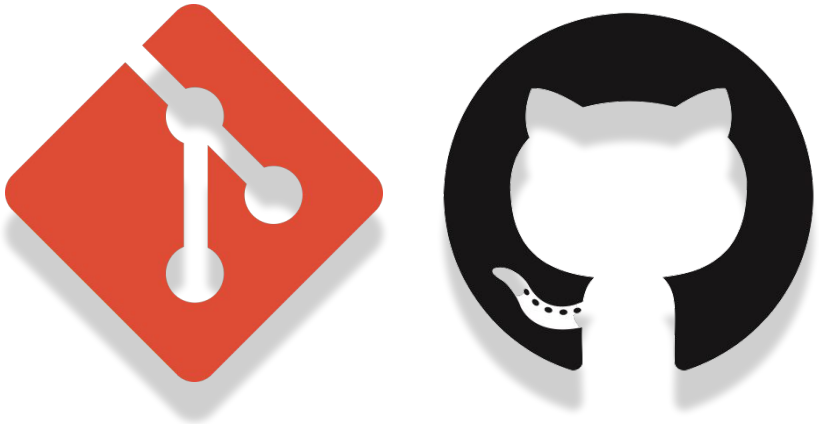






GIT

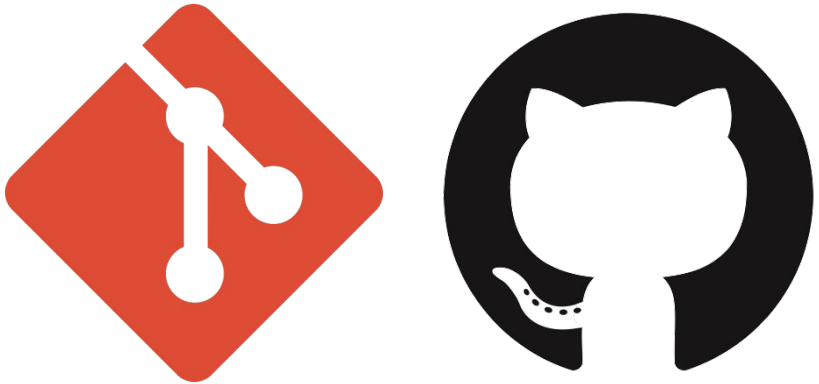
GIT y GitHub



¿Qué es GIT?

Es un software de control de versiones, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

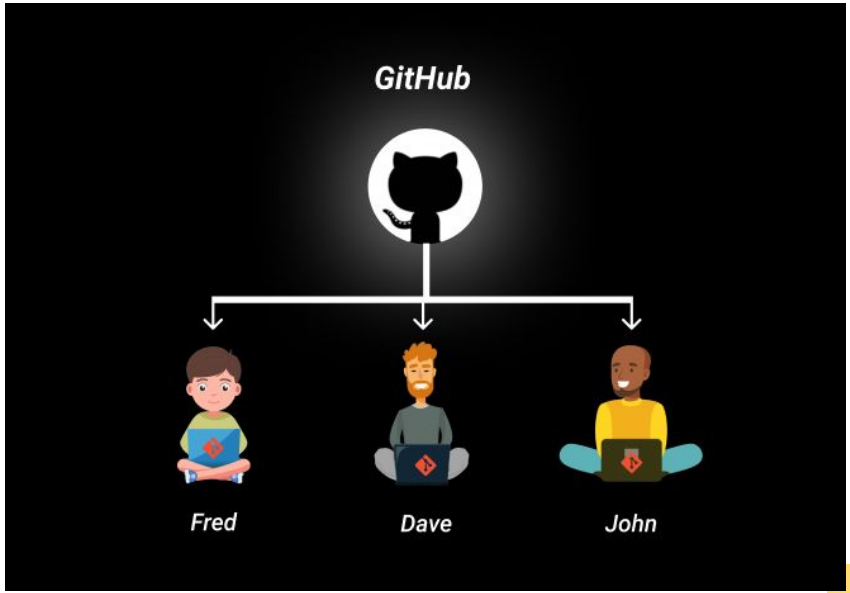
Existe la posibilidad de trabajar de forma remota y una opción es **GitHub**.



¿Qué es GitHub?

Es una plataforma de desarrollo colaborativo para alojar proyectos (en la “nube”) utilizando el sistema de control de versiones Git.

Además cuenta con una herramienta muy útil que es GitHub Pages donde podemos publicar nuestros proyectos estáticos (HTML, CSS y JS) de forma gratuita.



GIT | Definición

Un sistema que ayuda a organizar el código, el historial y su evolución, funciona como una máquina del tiempo que permite navegar a diferentes versiones del proyecto y si queremos agregar una funcionalidad nueva nos permite crear una rama (branch) para dejar intacta la versión estable y crear un ambiente de trabajo en el cual podemos trabajar en una nueva funcionalidad sin afectar la versión original.

GIT | ¿Qué permite?

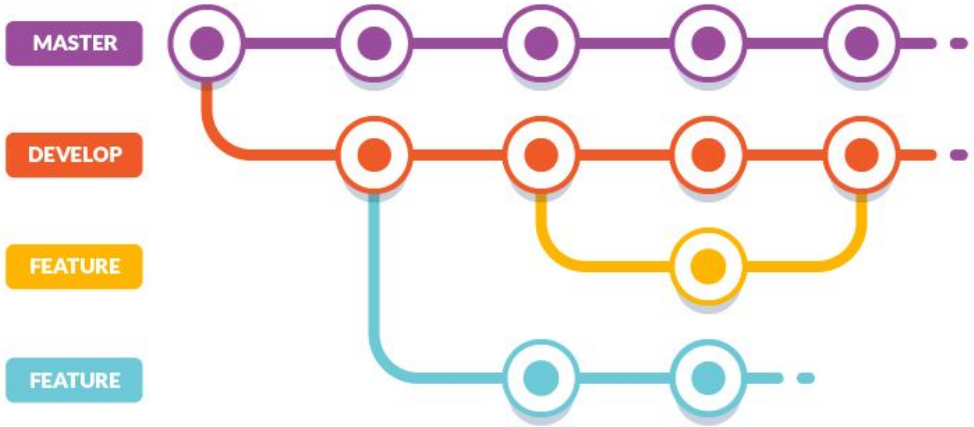
Mediante el control de versiones distribuido permite:

- Manejar distintas versiones del proyecto.
- Guardar el historial o guardar todas las versiones de los archivos del proyecto.
- Trabajar simultáneamente sobre un mismo proyecto.

Para ampliar:
Guía rápida:
https://docs.github.com/en/get-started/quickstart/set-up-git

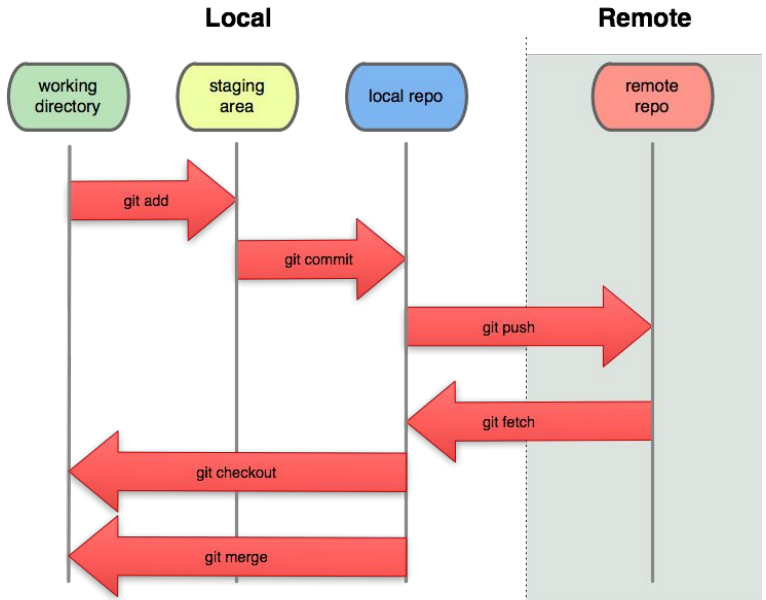
GIT | ¿Cómo funciona?

Git almacena instantáneas de un mini sistema de archivos, cada vez que confirmamos un cambio lo que Git hace es tomar una "foto" del aspecto del proyecto en ese momento y crea una referencia a esa instantánea, si un archivo no cambió Git no almacena el nuevo archivo sino que crea un enlace a la imagen anterior idéntica que ya tiene almacenada.

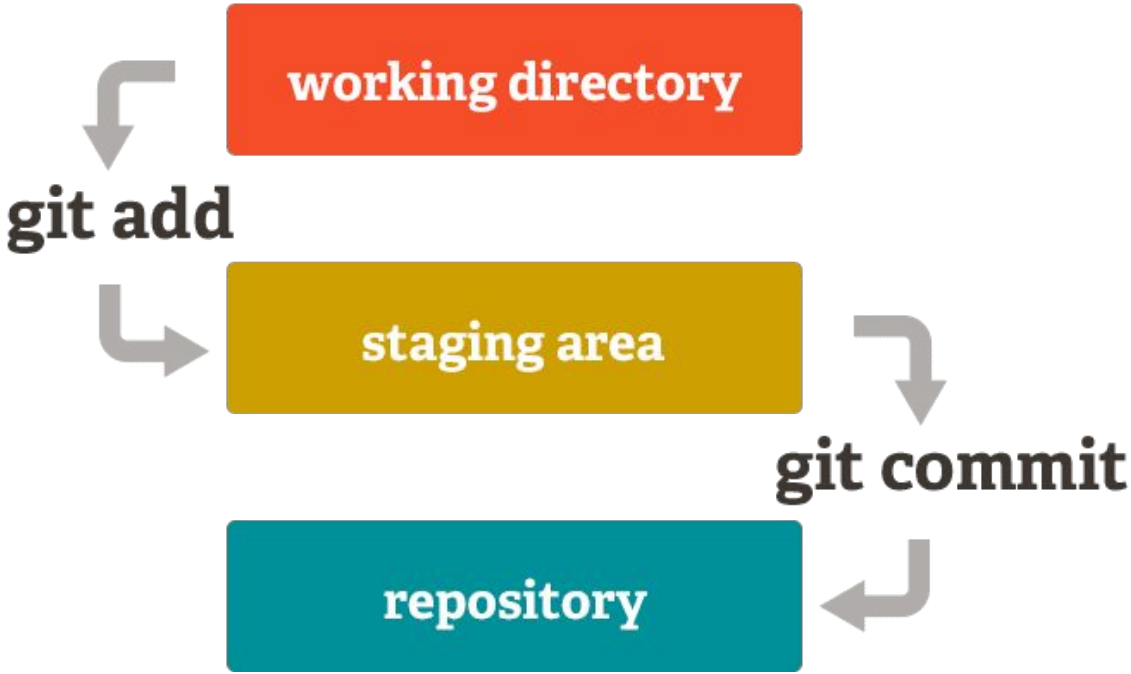


GIT | Flujo de trabajo

Tenemos nuestro directorio local (una carpeta en nuestra PC) con muchos archivos, Git nos irá registrando los cambios de archivos o códigos cuando nosotros le indiquemos, así podremos viajar en el tiempo retrocediendo cambios o restaurando versiones de código, ya sea en Local o de forma Remota (servidor externo).

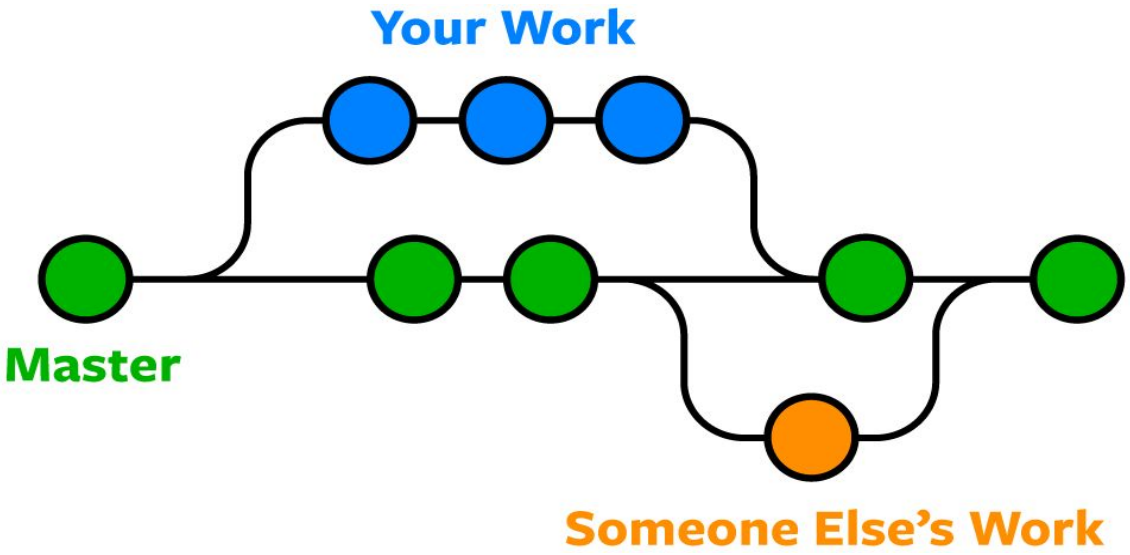


GIT | Estados


working directory
git add
staging area
git commit
repository

GIT | Ramas (*branch* y *merge*)

- **Crear** una rama: git branch nombreBranch
- **Unir** la rama a Master: git merge nombreBranch
- **Mostrar** en qué rama nos encontramos: git branch
- **Cambiar** a una rama determinada: git checkout nombreBranch



GIT | Terminología

- **Repositorio:** es la carpeta principal donde se encuentran almacenados los archivos que componen el proyecto. El directorio contiene metadatos gestionados por Git, de manera que el proyecto es configurado como un repositorio local.
- **Commit:** un commit es el estado de un proyecto en un determinado momento de la historia del mismo, imaginemos esto como punto por punto cada uno de los cambios que van pasando. Depende de nosotros determinar cuántos y cuales archivos incluirá cada commit.
- **Rama (branch):** una rama es una línea alterna del tiempo, en la historia de nuestro repositorio. Funciona para crear features, arreglar bugs, experimentar, sin afectar la versión estable o principal del proyecto. La rama principal por defecto es **master**.
- **Pull Request:** en proyectos con un equipo de trabajo, cada persona puede trabajar en una rama distinta pero llegado el momento puede pasar que dicha rama se tenga que unir a la rama principal, para eso se crea un **pull request** donde comunicas el código que incluye tu cambio y usualmente revisa tu código, se agregan comentarios y por último lo aprueban para darle **merge**. En el contexto de GIT, merge significa unir dos trabajos, en este caso tu **branch** con **master**.

GIT | Instalación

1) Descargar desde: https://git-scm.com/downloads.
2) Una vez descargado, se emplea la interfaz de línea de comando del sistema operativo para interactuar con GIT:
    - En Windows: abrir la aplicación Git Bash que se instaló junto con GIT.
    - En Mac: abrir la terminal mediante el finder.
    - En Linux: abrir la consola bash.
3) Para verificar si está instalado, podemos ejecutar el comando: git --version.
    - Si obtenemos respuesta nos indicará la versión de Git que tenemos instalada.
    - En caso de que no poder, ver las instrucciones acá: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git

GIT | Comandos básicos

- Una vez realizada la configuración básica, hay que utilizar la línea de comando para ubicarnos en la carpeta que queremos hacer control de versión.
    - **Ejemplo:** cd /Users/*Alumno*/Desktop/proyecto y una vez parados en esa carpeta, inicializamos git mediante el comando **git init** que también va a crear un archivo oculto llamado .git que se va a encargar de llevar el control de todas las modificaciones que se hagan en esa carpeta.
- Con el comando **git status** se van a mostrar en rojo todos los archivos que git sabe que existen en la carpeta pero que todavía no están registrados.
- Para que esos cambios queden registrados tenemos que añadirlos con el comando **git add** . el punto indica que queremos añadir todos los archivos.
- Si hacemos **git status** todos los archivos van a aparecer en verde.

GIT | Comandos básicos

GitLab vs. GitHub

- Actividad Práctica: investigar las diferencias.
  - Referencia: https://www.redeszone.net/2019/01/10/github-vs-gitlab-diferencias/

GIT y GitHub | Tutoriales

- **Fundamentos de GIT:** https://bluuweb.github.io/tutorial-github/guia/fundamentos.html
- **GitHub:** https://bluuweb.github.io/tutorial-github/guia/github.html

GIT y GitHub | Material multimedia

- **Videos del Profesor Alejandro Zapata (Coordinador y Docente de Codo a Codo):** https://www.youtube.com/watch?v=ptXiQwE535s&list=PLoCpUTIZIYORkDzYwdunkVf-KIqGjyoot
- **GIT y GitHub (tutorial en español). Inicio Rápido para Principiantes:** https://www.youtube.com/watch?v=hWgIK8nWh60&list=PLPl81lqbj-4I8i-x2b5_MG58tZfgKmJls
- **¿Cómo trabajar con Git desde Visual Studio Code?:** https://youtu.be/AYbgqmyg7dk
  - Nota: con Visual Studio Code también se puede hacer commits, push, resolver conflictos, crear ramas y mucho más. Prácticamente todo lo que se hace desde la línea de comandos lo podés hacer desde una interfaz gráfica. En el video se indica cómo hacerlo. Al final se recomienda un plugin que hay que instalar si se quiere trabajar con Git desde Visual Studio Code.
  - *Importante: se puede utilizar una interfaz gráfica para trabajar con Git, pero es importante saber qué es lo que pasa detrás de cada clic que uno hace. Por ese motivo antes, hay que aprender los fundamentos de GIT.*