codehouse
{ACADEMY}

codehouse
{ACADEMY}

3

Control de versiones con Git

codehouse
{ACADEMY}

Agenda

1. ¿Qué es el control de versiones?
2. ¿Por qué lo utilizamos?
3. Conceptos clave del control de versiones
4. Introducción a Git

1. ¿Qué es el control de versiones?

Es la práctica de registrar y gestionar cambios en el código.

Es un historial vivo de cada cambio en nuestro código, junto con la información del autor del cambio, fecha y más información importante.

Protegemos así el código de cambios sobre los que estamos trabajando.

Evitamos así tener múltiples copias del mismo código, pudiendo perder información o ficheros de gran valor.

Permite tener múltiples versiones de un código sin que estas se vean afectadas.

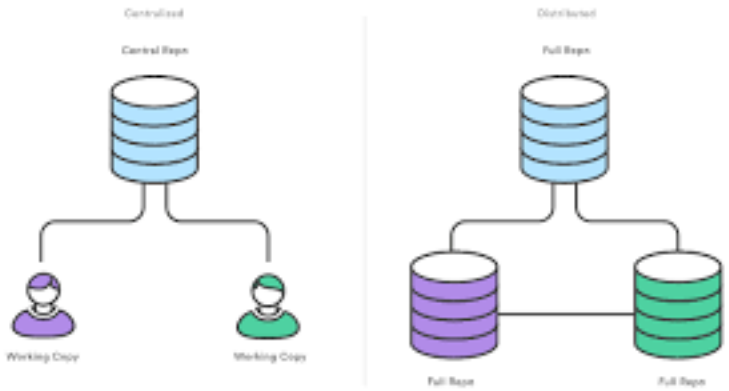
1.1. Tipos de control de versiones

- **Centralizado**

Una única copia para la colaboración de todos los programadores en el equipo. Todos los cambios se registran en la copia única.

- **Distribuido**

Cada programador se crea una copia del código original y trabaja sobre ella, solo visible para el programador. Después ofrece sus cambios para incorporarlos al código original, donde es visible por todos.



1.1. Tipos de control de versiones

La industria ha popularizado **el modelo distribuido**, que es el que aprenderemos.

Tendremos entonces:

- El código original donde todo el código se almacena todo código producido
- Una copia de todo el código para cada programador

2. ¿Por qué lo utilizamos?

Beneficios de usar control de versiones:

- Reusabilidad
- Seguimiento
- Gestión
- Eficiencia
- Colaboración
- Aprendizaje

3.1 Conceptos clave

Repositorios

Un repositorio es el directorio donde almacenamos el código. Puede ser en remoto o en nuestro equipo, de forma local.

Commit

Un conjunto de cambios agrupados bajo el nombre del autor, la fecha y la descripción de los cambios que hemos realizado.

***Branch*, ramas**

Las versiones con las que podemos trabajar.

3.2. Conceptos clave

Clonado

La copia en local de un repositorio que se encuentra en remoto.

Staging

Un estado intermedio que determina los cambios que van a formar parte de un *commit*.

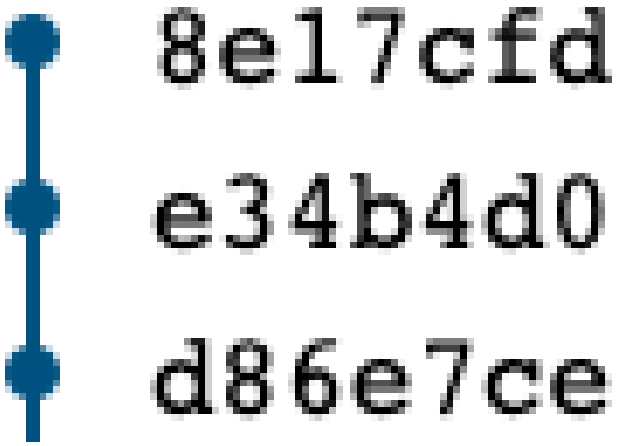
Pull request

La propuesta de incorporación de cambios de una rama a otra.

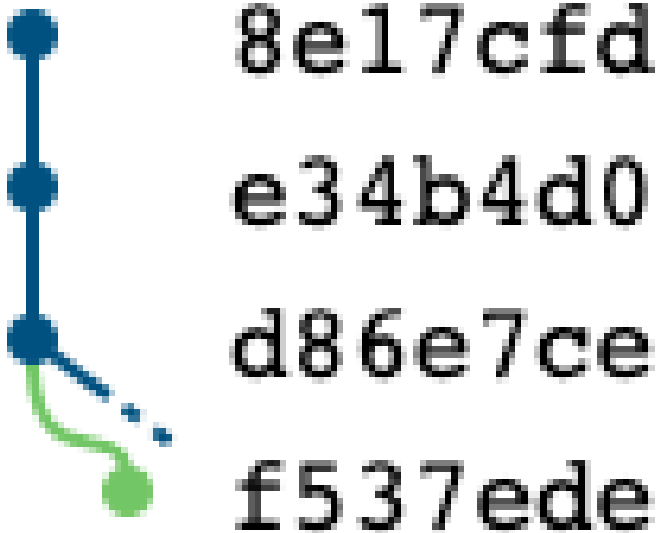
Merge

La fusión de los cambios de una rama hacia otra, generalmente cuando un *pull requests* es validado.

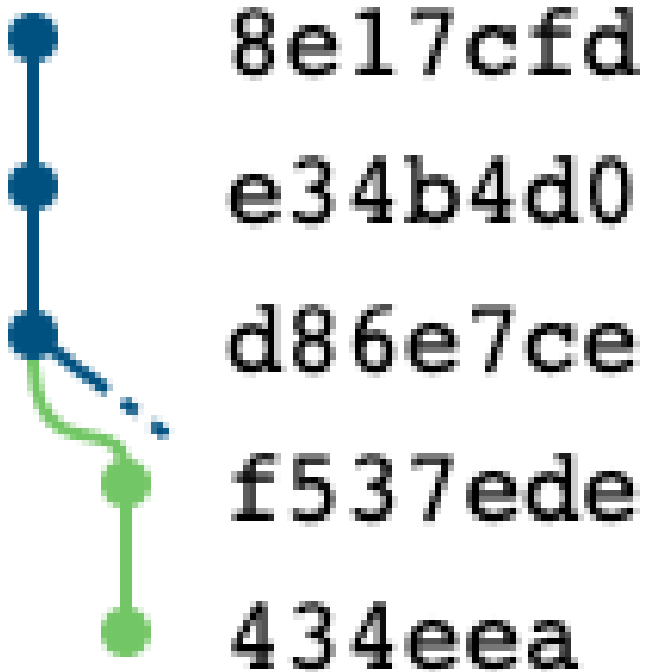
codehouse
{ACADEMY}



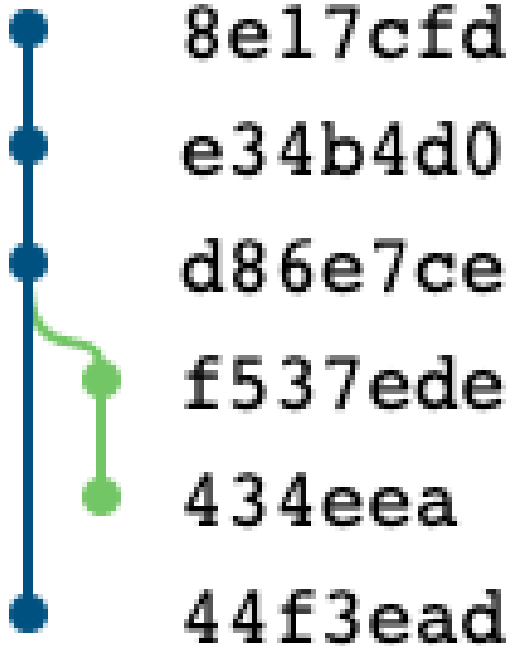
Control de versiones con Git



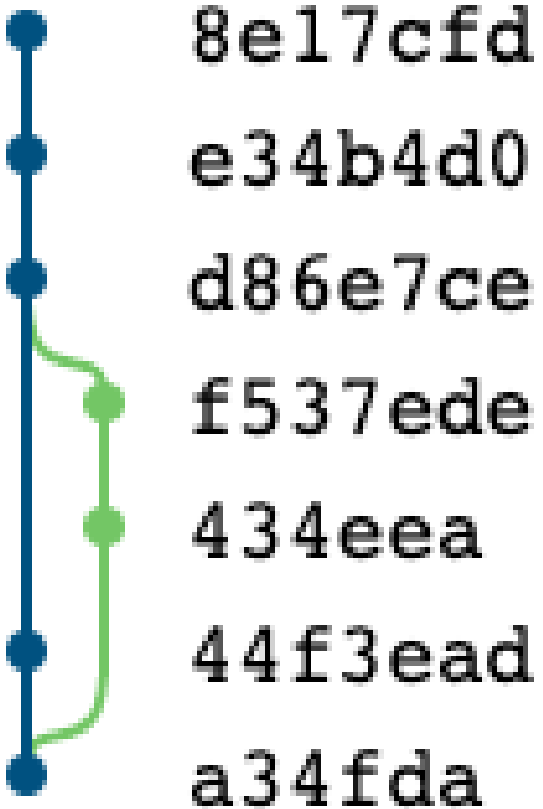
codehouse
{ACADEMY}



Control de versiones con Git



Control de versiones con Git



4. Introducción a Git

Realizamos el control de versiones de nuestros proyectos con Git.

Fue creado en 2005 por Linus Torvalds, creador del sistema operativo GNU/Linux.

Es la alternativa más popularizada en la actualidad, y su conocimiento es fundamental en el mundo del desarrollo.

A través del terminal, ejecutamos los comandos para controlar las versiones de nuestros proyectos. Con Git podemos:

- Crear repositorios
- Gestionar *branches*
- Crear *commits*

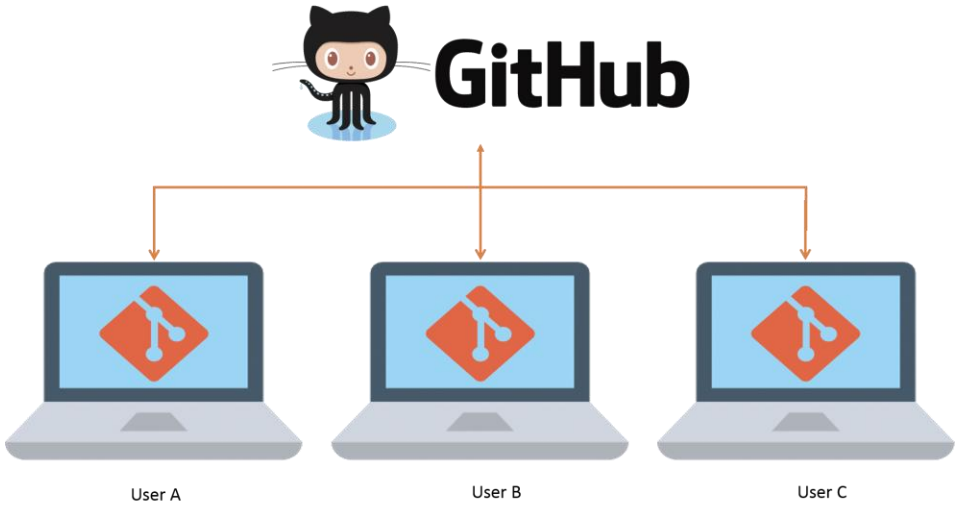
Y todas las operaciones convenientes para un control de versiones efectivo.

4.1. Git y GitHub

GitHub es una plataforma online donde hospedar tus repositorios.

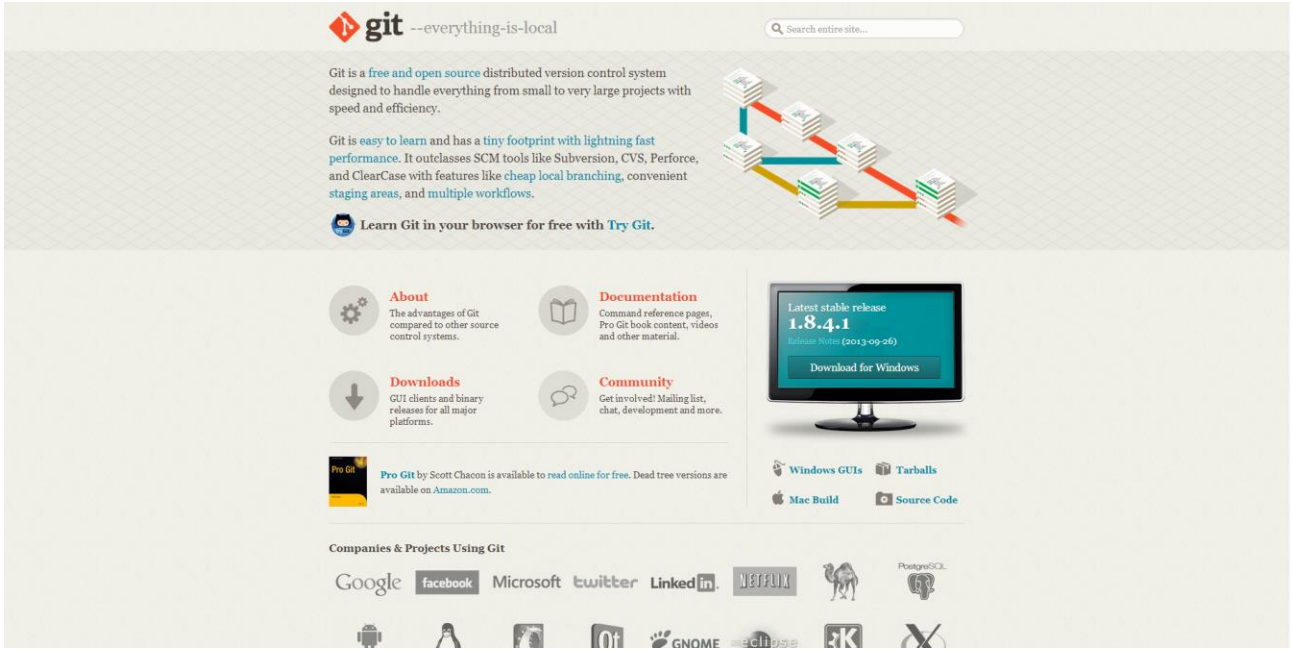
Es común confundir Git con GitHub, pero GitHub solo hospeda repositorios con Git.

Sirve como red social y como *portfolio* de proyectos de desarrollo.



4.2. Instalando Git

Instalando Git en tu equipo a través de [git-scm.com](git-scm.com)



codehouse
{ACADEMY}