SISTEMAS DE CONTROL DE VERSIONES

Unidad 1

Herramientas útiles para el desarrollo web

Módulo **Afondamento nas Competencias Profesionais (GS)**

Ciclo Desenvolvemento de aplicacións Web a distancia 2025-2026

XUNTA DE GALICIA
CENTRO INTEGRADO DE FORMACIÓN PROFESIONAL A CARBALLEIRA-MARCOS VALCÁRCEL

FP FORMACIÓN PROFESIONAL

SISTEMAS DE CONTROL DE VERSIONES

Sistemas de Control de Versiones

Los Sistemas de Control de Versiones permiten gestionar los cambios realizados en el código fuente de programas o documentos.

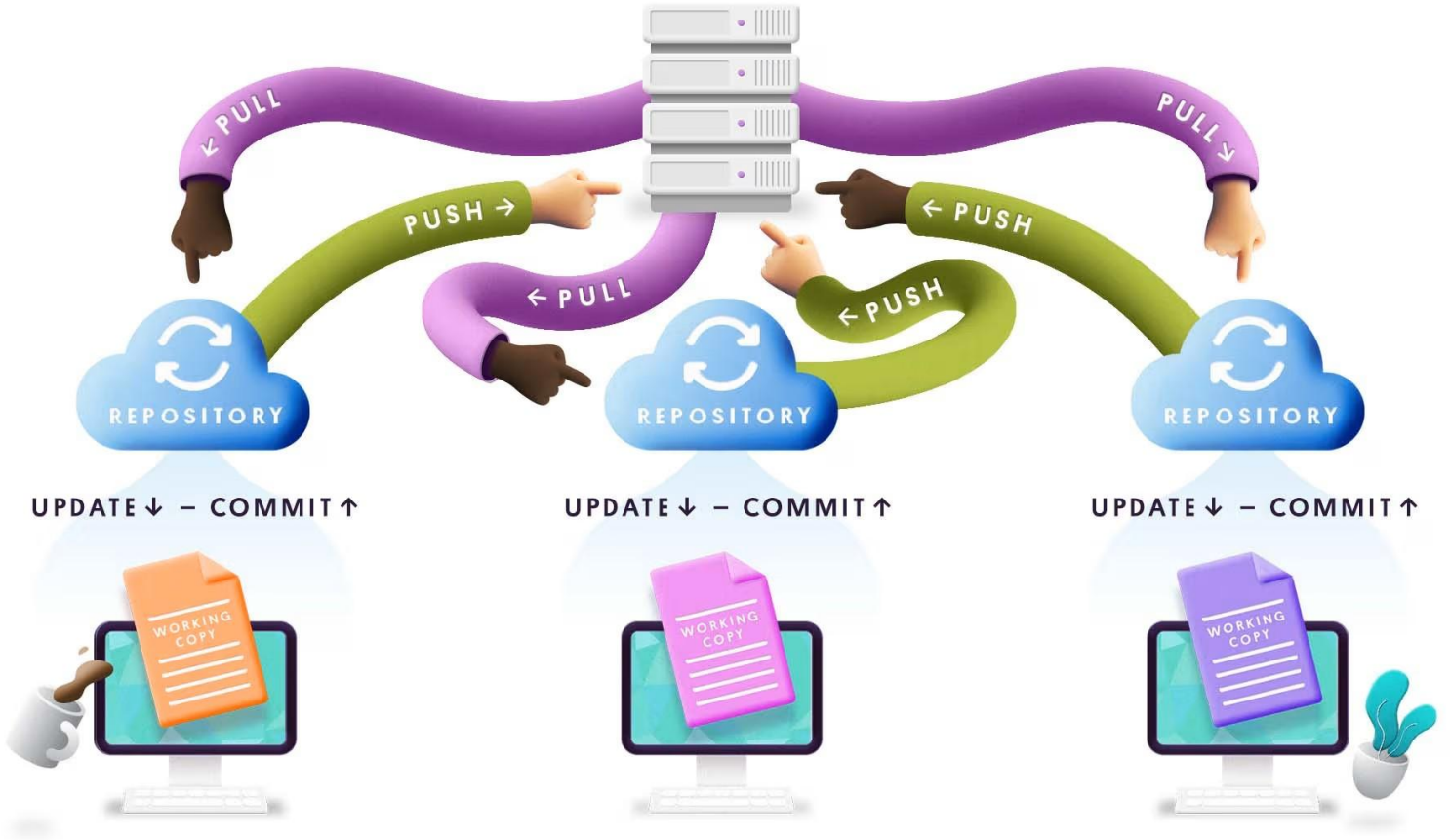
En los años 90 la mayoría de VCS era sistemas centralizados, en el que un servidor central guarda toda la información y los clientes se conectan al servidor.

En el mundo del software libre, el sistema más popular fue **CVS** (Concurrent Versions System), sustituido a partir del 2004 por **Subversion**.

SISTEMAS DE CONTROL DE VERSIONES

En el año 2000 apareció **BitKeeper**, un sistema de control de versiones distribuido, en el que cada cliente mantiene su propia copia completa del repositorio y puede trabajar sin estar conectado al servidor.
BitKeeper era un programa comercial, pero permitía su uso en proyectos de software libre.
Entre 2002 y 2005, BitKeeper se utilizó en el desarrollo del kernel Linux, pero en 2005 BitKeeper revocó la licencia que había concedido a los programadores del kernel.
Aunque para entonces ya había sistemas de control de versiones distribuidos libres (Monotone, darcs), Linus Torvalds decidió crear un nuevo programa, **Git**, que se publicó en abril de 2005.

SISTEMAS DE CONTROL DE VERSIONES



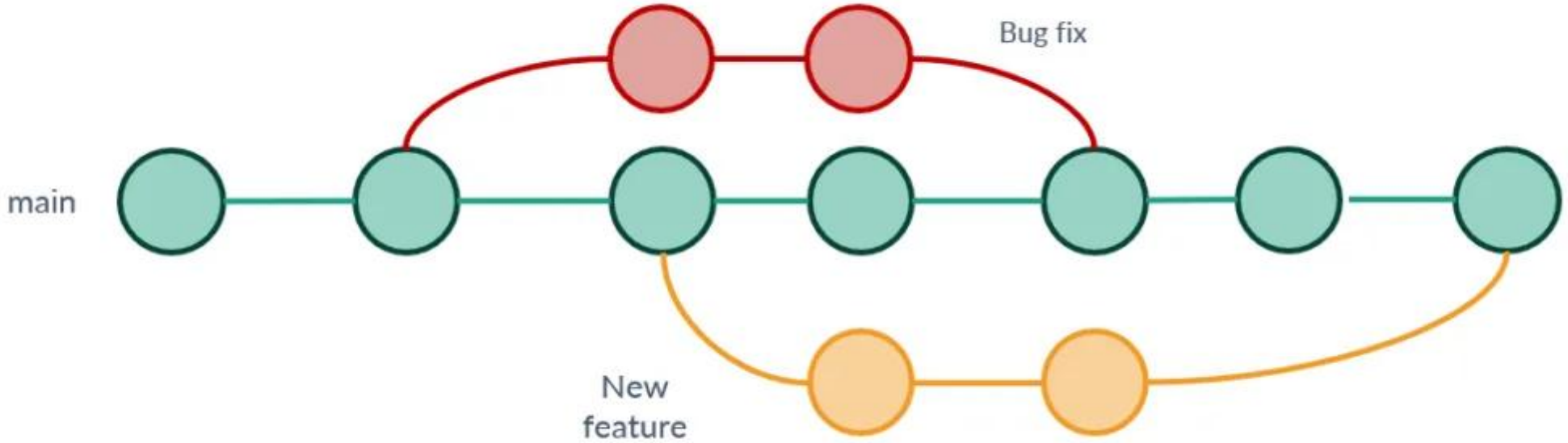
SISTEMAS DE CONTROL DE VERSIONES

¿Para qué se utiliza un Control de Versiones?

Permite realizar las siguientes tareas:

1. **Control de cambios:** Permite realizar un seguimiento de las modificaciones realizadas en los archivos de código fuente.
2. **Gestión de ramas (Branching)**: Permite crear ramas separadas del código para desarrollar nuevas características o solucionar problemas sin afectar la rama principal (conocida como rama "main" o "master").
3. **Fusión de cambios (Merging)**: Permite combinar las modificaciones realizadas en diferentes ramas y sincronizar el código entre ellas.
4. **Revertir cambios**: Permite deshacer o revertir cambios anteriores en caso de que se haya introducido un error o se requiera volver a una versión anterior del código.
5. **Gestión de conflictos**: Si varios desarrolladores están trabajando en el mismo archivo y realizan modificaciones concurrentes, un sistema de control de versiones puede ayudar a detectar y resolver los conflictos que puedan surgir al fusionar esos cambios.

SISTEMAS DE CONTROL DE VERSIONES



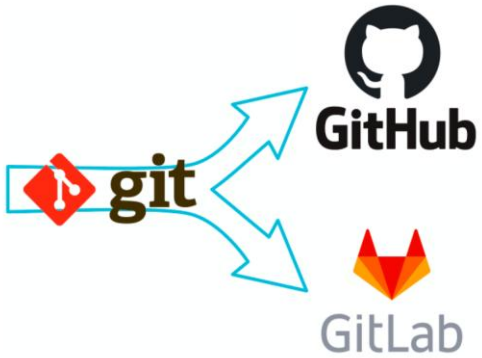
SISTEMAS DE CONTROL DE VERSIONES

Algunos de los sistemas de control de versiones que más se emplean actualmente son:

- **Git**
- Subversion (SVN)
- Mercurial
- CVS
- Bazaar

Complementando a los sistemas de control de versiones:

- **Github**
- Gitlab



GIT

Git es un **sistema de control de versiones** o VCS (Version Control System) que ayuda a los equipos de desarrollo a **gestionar los cambios en el código fuente a lo largo del tiempo**.

Este sistema nos permite almacenar un histórico de la evolución del desarrollo y nos ayuda al control de conflictos cuando hay varias personas trabajando sobre el mismo fichero.

Se ha convertido en el sistema de control de versiones más utilizado.

Fue creado por Linus Torvalds en 2005.

GIT

La función principal de GIT es rastrear cambios en el código fuente durante el desarrollo de software, lo que permite a los equipos trabajar de manera colaborativa y mantener un historial detallado de todas las modificaciones realizadas en el código.

Las características clave de GIT incluyen la capacidad de ramificar y fusionar fácilmente, lo que facilita el desarrollo paralelo de funciones o correcciones de errores sin afectar al código principal.

También ofrece un rendimiento rápido y eficiente, incluso en proyectos grandes.

Documentación oficial: https://git-scm.com/docs

GIT

Un proyecto Git cuenta con tres áreas de trabajo principales: **el directorio de trabajo**, **el área de preparación** y **el directorio repositorio (local o remoto)**.
En un proyecto Git los ficheros pueden tener los siguientes estados:

- **Untracked** (Sin seguimiento)
- **Tracked** (Bajo seguimiento)
- **Staged** (Preparado para confirmación)
- **Modified** (Modificado)
- **Deleted** (Elminado)

*Recuerda que Git y GitHub no son lo mismo, **Git** es nuestro **sistema local de gestión de versiones** y **GitHub** nos permite **publicar repositorios de código** en remoto. De esta forma podemos trabajar con un **sistema de control de versiones** en la nube.*

GITHUB

GitHub es una **plataforma en la nube** que utiliza el sistema de control de versiones **Git** para alojar repositorios de código y facilitar la colaboración en proyectos de desarrollo de software.

Es ampliamente utilizado por la comunidad de desarrollo de software para compartir, colaborar y contribuir a proyectos de código abierto y privados.

GitHub se ha convertido en el mayor repositorio de proyectos de software libre. Es de código abierto.

En junio de 2018 Microsoft compró GitHub por 7500 millones de dólares.

GITHUB

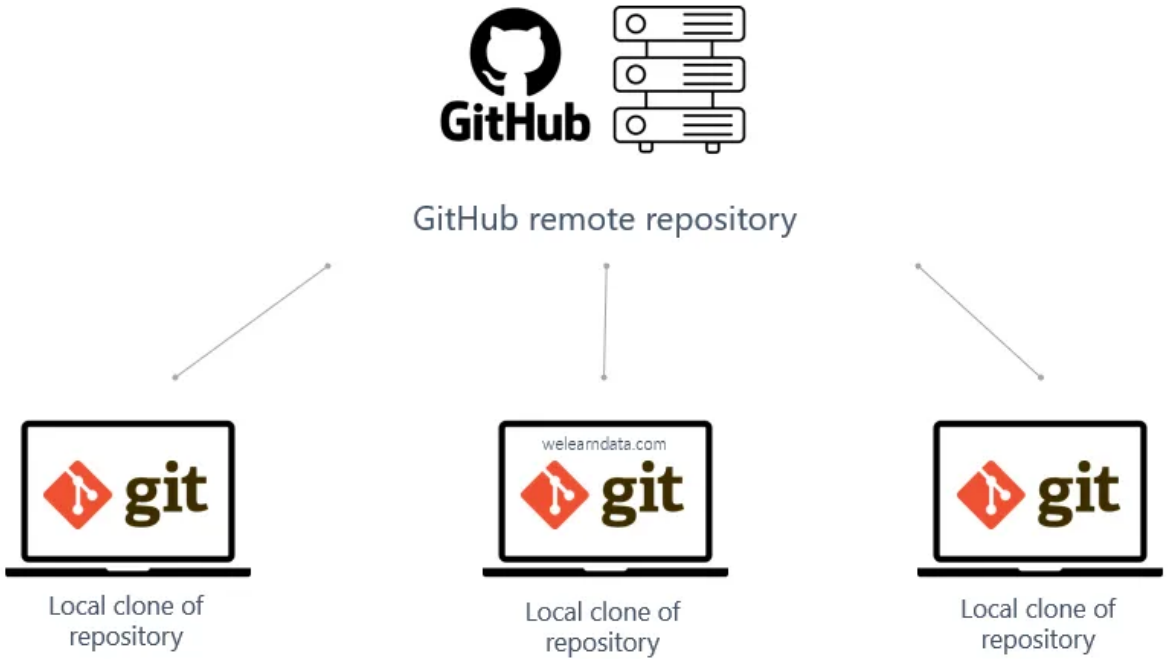
Características principales de GitHub

1. **Repositorios**: GitHub permite a los desarrolladores almacenar y gestionar sus proyectos en repositorios. Cada repositorio contiene todo el historial de cambios, ramas y archivos de código fuente de un proyecto específico.
2. **Control de versiones**: GitHub utiliza Git como su sistema de control de versiones, lo que permite a los desarrolladores realizar un seguimiento de los cambios en el código, crear ramas separadas y fusionar los cambios de diferentes fuentes.
3. **Colaboración**: GitHub fomenta la colaboración y la contribución abierta al permitir a los desarrolladores clonar (hacer una copia local) de los repositorios de otros, proponer cambios a través de "pull requests" (solicitudes de extracción) y revisar y comentar el código de otros.
4. **Issues y seguimiento de proyectos**: GitHub proporciona herramientas para el seguimiento de issues y la gestión de proyectos. Los usuarios pueden abrir los issues para informar errores, solicitar nuevas características o plantear cualquier otra cuestión relacionada con el proyecto.
5. **Integración continua y despliegue continuo (CI/CD)**: GitHub permite la integración con servicios de CI/CD, lo que facilita la automatización de pruebas, compilación y despliegue de aplicaciones. Esto mejora la eficiencia del proceso de desarrollo y garantiza la calidad del código.

GITHUB

Repositorio – Local y remoto

Un repositorio en Git es un lugar donde se almacena y administra el código de un proyecto. Es una estructura de datos que registra todos los cambios realizados en los archivos a lo largo del tiempo, lo que permite rastrear **y controlar el historial de versiones del proyecto**.



GITLAB

¿Qué es GitLab y para qué sirve?

GitLab es una plataforma de repositorios Git de código abierto.

Al igual que otras herramientas similares como GitHub, te permite almacenar bases de código de proyectos en una ubicación centralizada, habilitar el control de versiones para un seguimiento más sencillo y crear ramas para el desarrollo asíncrono.

A diferencia de GitHub, **puedes autoalojar GitLab en tu infraestructura como repositorio interno**. Esto mejora la seguridad, ya que tienes pleno control y propiedad de la herramienta.

GITLAB VS GITHUB

GitHub para poder instalarlo en un servidor propio **se requiere la versión Enterprise de pago**.

Ambos programas cuentan con una versión gratuita y una versión **Enterprise** para empresas.

GitHub no ofrece **herramientas de integración continua propias**. Aquí, GitLab toma la delantera y ofrece integración continua gratuita de fábrica.

Durante mucho tiempo, la gran ventaja de GitLab era que ofrecía infinitos repositorios gratuitos a sus usuarios, pero GitHub vio su desventaja y ahora también ofrece esta característica.

DIRECCIONES WEB DE CONSULTA

- Git: https://git-scm.com/
- GitHub: https://github.com/
- GitLab: https://about.gitlab.com/
- Subversion: https://subversion.apache.org/
- Curso de Git y GitHub: https://welearndata.com/git/
- Git vs GitHub vs GitLab: https://www.tokioschool.com/noticias/git-github-gitlab/
- Comparativa GitHub y GitLab: https://bambu-mobile.com/gitlab-vs-github-diferencias/