El control de versiones con Git

1. ¿Qué es Git?

Git es una herramienta esencial en el desarrollo web, permitiendo el control de versiones del código y la colaboración eficiente entre equipos. Su utilidad radica en diversas características clave que lo hacen una elección popular en el ámbito del desarrollo:

Control de versiones eficiente

Git permite documentar los cambios realizados en el código, asignar responsabilidades y mantener un historial completo de revisiones. Esto facilita la colaboración entre desarrolladores al mantener un registro claro de quién hizo qué y cuándo.

Características destacadas

- **Sistema de control de versiones distribuido**: Con Git, el código existe tanto localmente como en un repositorio remoto, lo que permite un seguimiento completo y descentralizado de los cambios.
- **Trabajo desconectado**: Permite acceder al historial completo del código incluso sin conexión a internet, lo que facilita la continuidad del desarrollo desde cualquier lugar.
- **Suspensión de trabajos**: Git permite pausar el desarrollo para abordar incidencias en producción, lo que proporciona flexibilidad para gestionar tareas críticas.
- **Commits atómicos**: Con la zona de stage, Git permite trabajar con distintos commits, fortaleciendo la capacidad de seleccionar y organizar archivos y partes específicas de los mismos.
- **Flujos de trabajo con ramas**: Posibilita la creación de ramas para aislar partes del desarrollo, permitiendo una gestión eficiente y organizada del flujo total del proyecto.
- **Bajo residuo**: Toda la información se encuentra en una carpeta única .git, lo que facilita un manejo sencillo y modular de la información.

- **Interfaz amigable**: A pesar de ser una herramienta de línea de comandos, Git cuenta con interfaces gráficas que facilitan su uso, adaptándose a usuarios con diferentes niveles de experiencia.
- **Acceso sin cliente**: La capacidad de acceder a cualquier repositorio de Git sin necesidad de tener un cliente específico proporciona flexibilidad y accesibilidad.
- **Multiservidor**: Permite tener varios repositorios, lo que resulta útil para separar proyectos de clientes y desarrollo en un servidor, y aborda problemas legales al rastrear cambios y responsabilidades en el código.

2. Flujos de trabajo en Git

Se describen cuatro enfoques principales:

Sin flujo

Implica trabajar directamente en la rama principal del repositorio, sin utilizar ramas adicionales. Adecuado para proyectos pequeños o individuales, pero puede volverse complicado en proyectos más grandes.

Rama por tarea/defecto

Cada tarea o corrección se realiza en una rama independiente y luego se fusiona en la rama principal. Es útil para grupos pequeños y proyectos que necesitan implementaciones rápidas.

Fork

Cada desarrollador trabaja en su propio repositorio (fork) del proyecto principal y contribuye mediante solicitudes de extracción. Común en proyectos de código abierto, fomenta la colaboración sin conceder acceso directo a la rama principal.

GitFlow

Un modelo más estructurado que utiliza ramas específicas para funciones como nuevas características y correcciones de errores. Proporciona un enfoque claro para el desarrollo y la liberación de software, con reglas específicas para cada tipo de rama. GitFlow maneja muchas ramas, pero las automatiza bastante bien:

- **MASTER**: Es la rama liberada.
- **DEVELOP**: Es nuestra rama de trabajo (¡Cuidado! no se conecta por defecto a esta rama).
- **FEATURED**: Es la tarea que luego volcamos a develop.
- **HOTFIX**: Es la rama de errores.
- **RELEASE**: Es la rama donde testeamos antes de pasar a master.

3. Crear un repositorio de Git

Github

Plataforma de desarrollo colaborativo que utiliza Git para el control de versiones. Tiene una extensa comunidad de desarrolladores.
Proporciona herramientas para la gestión de proyectos, seguimiento de problemas y la posibilidad de colaborar en proyectos de código abierto.

Bitbucket

Bitbucket, propiedad de Atlassian, es una plataforma que ofrece control de versiones Git y Mercurial. Se destaca por su integración con otras herramientas de Atlassian, como Jira y Confluence. Además, ofrece repositorios privados gratuitos.

Gitlab

Proporciona el control de versiones Git y el seguimiento de problemas. Además, incluye herramientas para la integración y el despliegue continuo (CI/CD).
Puede utilizarse tanto como un servicio en la nube o instalarse en un servidor propio. La opción de instalar GitLab en un servidor propio brinda flexibilidad en el control y seguridad del entorno.

4. Comandos de Git

Comandos de Git

Comando	Descripción	Recomendaciones de uso
`git checkout`	Cambia la rama de trabajo actual	• Para cambiar a una rama específica del proyecto. • Para crear una nueva rama y cambiar a ella al mismo tiempo.
`git add`	Agrega archivos al índice de Git	• Para agregar archivos que se van a incluir en el siguiente commit. • Para agregar todos los archivos del directorio de trabajo al índice.
`git commit`	Crea un nuevo commit en el historial de Git	• Para crear un registro de los cambios realizados en el código fuente. • Para preparar los cambios para ser enviados a un repositorio remoto.
`git clone`	Clona un repositorio Git existente	• Para crear una copia local de un repositorio Git existente. • Para trabajar en un proyecto existente sin tener que descargar todo el código fuente.
`git push`	Envía los cambios realizados en el repositorio local a un repositorio remoto	• Para compartir los cambios realizados en el código fuente con otros desarrolladores. • Para almacenar los cambios realizados en un repositorio seguro.
`git pull`	Actualiza el repositorio local con los cambios realizados en el repositorio remoto	• Para actualizar el código fuente local con los cambios realizados por otros desarrolladores. • Para obtener los últimos cambios realizados en un repositorio remoto.
`git init`	Inicia un repositorio local desde cero	• Para crear repositorios personales, como nuestra propia web.
`git branch`	Crea una rama completamente nueva, sin cambiar a ella	• Si hace falta crear varias ramas, por ejemplo para empleados que entran nuevos. IMMUNE TECHNOLOGY INSTITUTE