Cristian Alexander Medina Mendez
FUNDAMENTOS DE PROGRAMACIÓN Y SCRIPTING

Que es GIT

Git es un sistema avanzado de control de versiones (como el “control de cambios” de Microsoft Word) distribuido (Ram 2013; Blischak et al. 2016). Git permite “rastrear” el progreso de un proyecto a lo largo del tiempo ya que hace “capturas” del mismo a medida que evoluciona y los cambios se van registrando. Esto permite ver qué cambios se hicieron, quién los hizo y por qué, e incluso volver a versiones anteriores

Además, Git facilita el trabajo en paralelo de varios participantes. Mientras que en otros sistemas de control de versiones

GitHub es un servidor de alojamiento en línea o repositorio remoto para albergar proyectos basados en Git que permite la colaboración entre diferentes usuarios o con uno mismo

Un repositorio es un directorio donde desarrollar un proyecto que contiene todos los archivos necesarios para el mismo. Aunque existen distintos repositorios remotos (p. ej. GitLab, https://gitlab.com/, o Bitbucket, https://bitbucket.org/) con funcionalidad similar, GitHub es hoy en día el más utilizado. GitHub registra el desarrollo de los proyectos de manera remota, permite compartir proyectos entre distintos usuarios y proporciona la seguridad de la nube entre otras funciones.

Cuando se trabaja en proyectos colaborativos, la base de la interacción entre Git y GitHub es que todos los colaboradores de un proyecto están de acuerdo en que GitHub contiene la copia principal del proyecto, es decir, GitHub contiene la copia centralizada del control de versiones distribuido o descentralizado

Control de versiones con GIT

Es recomendable utilizar alguna herramienta de control de versiones como Git (https://git-scm.com/) combinada con plataformas en línea para albergar los proyectos (p. ej. GitHub, https://github.com/), facilitando así el seguimiento de los proyectos.

Git es un control de versiones distribuido que permite a todos los usuarios trabajar en el proyecto paralelamente e ir haciendo “capturas” del trabajo de cada uno para luego unirlos.

Estados de un archive en GIT

Sin embargo, **git add** no afecta al repositorio local. Para ver el estado del directorio de trabajo y del área de preparación se utiliza **git status**. Este comando permite ver qué archivos están siendo rastreados por Git, qué cambios han sido añadidos al área de preparación (staged) y qué archivos están siendo registrados por Git.

Para registrar los cambios que nos interesan se utiliza git commit. Al ejecutar git commit se hace una “captura” del estado del proyecto. Junto con el commit se añade un mensaje con una pequeña explicación de los cambios realizados y por qué (p. ej. “incluyo las referencias en el formato de Ecosistemas”). Cada git commit tiene un SHA (Secure Hash Algorithm) que es un código alfanumérico que identifica inequívocamente ese commit (p. ej. 1d21fc3c33cxxc4aeb7823400b9c7c6bc2802be1). Con el SHA siempre se pueden ver los cambios que se hicieron en ese commit y volver a esa versión fácilmente.

Como se configura un repositorio

Para crear un nuevo repositorio, usa el comando git init . git init es un comando que se utiliza una sola vez durante la configuración inicial de un repositorio nuevo. Al ejecutar este comando, se creará un nuevo subdirectorio. git en tu directorio de trabajo actual. También se creará una nueva rama principal.

Primero deberás establecer el directorio de la carpeta raíz del proyecto con el comando cd y luego ejecutar git init.

```
cd /path/to/your/existing/code
git init
```

Apuntar git init a un directorio de proyecto existente hará que se ejecute la misma configuración de inicio mencionada arriba, pero en el ámbito de ese directorio.

```
git init <project directory>
```

Comandos en GIT

git checkout

Además de extraer las confirmaciones y las revisiones de archivos antiguas, git checkout también sirve para navegar por las ramas existentes. Combinado con los comandos básicos de Git, es una forma de trabajar en una línea de desarrollo concreta.

git clean

Elimina los archivos sin seguimiento de tu directorio de trabajo. Es la contraparte lógica de git reset, que normalmente solo funciona en archivos con seguimiento.

git clone

Crea una copia de un repositorio de Git existente. La clonación es la forma más habitual de que los desarrolladores obtengan una copia de trabajo de un repositorio central.

git commit

Confirma la instantánea preparada en el historial del proyecto. En combinación con git add, define el flujo de trabajo básico de todos los usuarios de Git.

git init

Inicializa un nuevo repositorio de Git. Si quieres poner un proyecto bajo un control de revisiones, este es el primer comando que debes aprender.

git pull

Este comando es la versión automatizada de git fetch. Descarga una rama de un repositorio remoto e inmediatamente la fusiona en la rama actual. Este es el equivalente en Git de svn update.

git push

Enviar (push) es lo opuesto a recuperar (fetch), con algunas salvedades. Permite mover una o varias ramas a otro repositorio, lo que es una buena forma de publicar contribuciones. Es como svn commit, pero envía una serie de confirmaciones en lugar de un solo conjunto de cambios.

git status

Muestra el estado del directorio en el que estás trabajando y la instantánea preparada. Lo mejor es que lo ejecutes junto con git add y git commit para ver exactamente qué se va a incluir en la próxima instantánea.