Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Estudios de postgrado
Maestría en Ingeniería para la Industria con Especialidad en Telecomunicaciones

Iván Alejandro Esteban Archila
2600 30015 0901
9930 10791

GIT

Git es un sistema de control de versiones, lo que nos permite trabajar sobre un mismo proyecto y crear diferentes cambios y luego mediante comando hacer un merge o unión a la rama principal para unir los cambios. Git trabaja con branches o ramas, se tiene una rama principal que es llamada main o master y a partir de ella se generan nuevas ramas, cada rama suele ser trabajada por uno o varios desarrolladores para una mejora o corrección en un programa, luego de terminar esto, se realiza el merge para aplicar los cambios sobre la rama principal.

Se pueden n cantidad de ramas de forma paralela y la intención es combinar los cambios sobre la principal, resolviendo conflictos y manteniendo un control sobre las diferentes versiones y también se puede regresar a una rama específico o revertir cambios.

Control de versiones

Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

De manera intuitiva muchas personas ya utilizan control de versiones en sus proyectos al renombrar las distintas versiones de un mismo archivo de varias formas como blogScript.js, blogScript_v2.js, blogScript_v3.js, blogScript_final.js, blogScript_definite_final.js, etcétera. Pero esta forma de abordarlo es propenso a errores y inefectivo para proyectos grupales.

Además, con esta forma de abordarlo, rastrear qué cambió, quién lo cambió y porqué se cambió, es un esfuerzo tedioso. Esto resalta la importancia de un sistema de control de versiones confiable y colaborativo como Git.

Estados de un archivo en git

En Git hay tres etapas primarias (condiciones) en las cuales un archivo puede estar: estado modificado**,** estado preparado**,** o estado confirmado**.**

- **Estado modificado:** Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar). En otras palabras, archivos en el estado modificado son archivos que has modificado pero no le has instruido explícitamente a Git que controle.
- **Estado preparado:** Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio .git durante la próxima instantánea de confirmación. Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.
- **Estado confirmado:** Archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente. Por lo tanto un archivo confirmado es un archivo en el cual has registrado su versión preparada en el directorio (carpeta) Git.

Comandos en git

- **git config**

Establece valores de configuración para tu usuario, email, gpg key, algoritmo diff preferido, formatos de archivo y más. Ejemplos:

```
git config --global user.name "Mi nombre"
git config --global user.email usuario@dominio.com
```

- **git init**

Inicializa un repositorio git – crea el directorio .git inicial en un proyecto nuevo o existente. Ejemplo:

```
git init
Initialized empty Git repository in /home/username/GIT/.git/
```

- **git clone**

Crea una copia de repositorio GIT de una fuente externa. También añade la ubicación original como remota de modo que puedas traerlo de nuevo y lanzarlo si tienes permisos. Ejemplo:

```
git clone git@github.com:user/test.git
```

- **git add**

Añade cambios de archivos en tu directorio de ensayo a tu index. Ejemplo:

```
git add .
```

- **git commit**

Toma todos los cambios escritos en el index, crea un nuevo objeto de confirmación que apunta a él y establece la rama para que apunte a esa nueva confirmación. Ejemplos:

```
git commit -m ‘committing added changes’
git commit -a -m ‘committing all changes, equals to git add and git commit’
```

- **git status**

Te muestra el estado de los archivos en el index en comparación con los del directorio de trabajo. Enumerará los archivos que no están rastreados (solo en su directorio de trabajo), modificados (rastreados pero aún no actualizados en tu index), y almacenados (añadidos a tu index y listos para comprometerse). Ejemplo:

```
git status
# On branch master #
# Initial commit #
# Untracked files: #
# (use "git add <file>..." to include in what will be committed) #
README
```

- **git branch**

Para listar las ramas existentes, incluyendo las ramas remotas, si se proporciona ‘-a’. Crea una nueva rama si se proporciona un nombre. Ejemplo:

```
git branch -a * master remotes/origin/master
```

- **git merge**

Fusiona una o más ramas con otra rama activa y crea automáticamente un nuevo commit si no hay conflictos. Ejemplo:

```
git merge newbranchversion
```

- **git reset**

Resetea tu index y directorio de trabajo al último estado comprometido. Ejemplo:

```
git reset --hard HEAD
```

- **git pull**

Obtiene los archivos del repositorio remoto y los combina con el local. Ejemplo:

```
git pull origin
```

- **git push**

Envía todos los objetos modificados localmente al repositorio remoto. Ejemplo:

```
git push origin master
```

- **git remote**

Muestra todas las versiones remotas de tu repositorio. Ejemplo:

```
git remote origin
```

- **git log**

Muestra una lista de confirmaciones en una rama que incluye los detalles correspondientes. Ejemplo:

```
git log commit
84f241e8a0d768fb37ff7ad40e294b61a99a0abe Author: User
<user@domain.com> Date: Mon May 3 09:24:05 2010 +0300 first commit
```