UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE POST GRADOS

NOMBRE	DPI
ELIÚ MIGUEL VELÁSQUEZ BONILLA	2994988240101

Línea de Investigación

Que es GIT:

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos completos repositorios locales facilitan el trabajo sin conexión o de forma remota. Los desarrolladores confirman su trabajo localmente y luego sincronizan su copia en el repositorio con la copia en el servidor.

Git tiene una arquitectura distribuida y es un ejemplo de DVCS (Sistema de control de versiones distribuidas). En lugar de tener una ubicación única para todo el historial de versiones de software, como era el caso de los sistemas de control de versiones que alguna vez fueron populares como CVS o Subversion (también conocido como SVN), Git también tiene la copia de trabajo del código de cada desarrollador en un repositorio que contiene el historial de todos los cambios.

Git es un sistema de control de versiones distribuido, lo que significa que crea una copia completa del repositorio de código base en su computadora (y en la de todos los demás). Usted realiza cambios en su propia copia y luego confirma esos cambios en el servidor, donde un administrador decide si fusionar o no sus cambios en la copia maestra.

Controles de versiones con GIT

Un sistema de control de versiones (VCS) se refiere al método por el cual las versiones de un archivo se almacenan para referencia futura.

El control de versiones, también conocido como "control de código fuente", es el seguimiento y la gestión de cambios en el código del software. Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a administrar los cambios en el código fuente a lo largo del tiempo.

Guarda los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo para que pueda recuperar versiones específicas más adelante.

El software de control de versiones rastrea todos los cambios de código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden retroceder en el tiempo y comparar versiones anteriores del código para corregir el error y minimizar las interrupciones para todos los miembros del equipo.

Estado de un archivo en GIT

Git tiene tres estados principales en los que pueden estar sus archivos: comprometido, modificado y preparado.

- Confirmado: significa que los datos se almacenan de forma segura en su base de datos local.
- Modificado: significa que ha modificado el archivo, pero aún no lo ha agregado a su base de datos.
- Preparado: esto significa que ha marcado un archivo modificado en su versión actual para su próxima confirmación.

Como se configura un repositorio

Git almacena las opciones de configuración en tres archivos distintos, lo que te permite ajustar opciones para repositorios individuales (local), usuarios (global) o todo el sistema (sistema):

- Local: /.git/config —ajustes específicos del repositorio.
- Global: ~/.gitconfig —ajustes específicos del usuario. Aquí es donde se almacenan las opciones configuradas con la marca --global.
- Sistema: $(prefix)/etc/gitconfig —ajustes de todo el sistema.

Define el nombre del autor que se va a usar en todas las confirmaciones del repositorio actual.

```
git config --global user.name <name>
```

Navega al directorio de tu aplicación (o sitio web)

```
cd /home/username/example.com
```

Corre lo siguiente para inicializar el nuevo repositorio

```
git init
```

Agrega todos los archivos de tu aplicación al repositorio de Git. El siguiente comando muestra cómo agregar todos los archivos:

```
git add
```

Corre el comando de estado para confirmar qué archivos están en el área de preparación

```
git status
```

Corre el comando de estado para confirmar qué archivos están en el área de preparación

```
git commit
```

Comandos en GIT

Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo).

```
git clone https://link-con-nombre-del-repositorio
```

Creando una nueva rama:

```
git branch <nombre-de-la-rama>
```

Este comando creará una rama en local. Para enviar (push) la nueva rama al repositorio remoto, necesitarás usar el siguiente comando:

```
git push <nombre-remoto> <nombre-rama>
```

Visualización de ramas:

git branch

git branch --list

Borrar una rama:

```
git branch -d <nombre-de-la-rama>
```

Usaremos git checkout principalmente para cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

```
git checkout <nombre-de-la-rama>
```

El comando de git status nos da toda la información necesaria sobre la rama actual.

```
git status
```

Añadir un único archivo:

```
git add <archivo>
```

Añadir todo de una vez:

```
git add -A
```

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.
También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

```
git commit -m "mensaje de confirmación"
```

Importante: Git commit guarda tus cambios únicamente en local.

Git push envía tus commits al repositorio remoto.

```
git push <nombre-remoto> <nombre-de-tu-rama>
```

El comando git pull se utiliza para recibir actualizaciones del repositorio remoto.

```
git pull <nombre-remoto>
```