Universidad Nacional Autónoma de México

Facultad de Ciencias

Propedéutico
Git

Semestre: 2025-1

Autor:
Julio Vázquez Alvarez

Historia de Git

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para manejar desde proyectos pequeños hasta muy grandes con rapidez y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux, con el objetivo de reemplazar BitKeeper, un sistema de control de versiones propietario que había sido utilizado por la comunidad de desarrollo de Linux.

¿Qué es Git?

Git es un sistema de control de versiones distribuido, lo que significa que no hay un único repositorio central, sino que cada usuario tiene una copia local del repositorio completo. Esto permite que los usuarios trabajen de manera independiente y sin conexión a internet, y que puedan sincronizar sus cambios con otros usuarios cuando sea necesario.

¿Por qué usar Git?

Git es una herramienta poderosa y flexible que facilita el trabajo colaborativo y la gestión de versiones de un proyecto. Algunas de las ventajas de usar Git son:

- Facilidad para trabajar en paralelo con otros usuarios.
- Historial completo de cambios en el proyecto.
- Posibilidad de deshacer cambios y volver a versiones anteriores.
- Ramificación y fusión de ramas para trabajar en nuevas funcionalidades sin afectar la rama principal.
- Facilidad para colaborar con otros usuarios a través de repositorios remotos.

Conceptos básicos de Git

Git tiene algunos conceptos fundamentales que es importante comprender para poder utilizarlo de manera efectiva.

Repositorio

Llamamos repositorio a la carpeta que contiene todos los archivos y directorios de un proyecto, así como el historial de cambios de cada uno de ellos. Cada repositorio de Git tiene una carpeta oculta llamada `.git` que almacena toda la información necesaria para gestionar los cambios del proyecto.

Staging area

El *staging area* es un área intermedia entre el directorio de trabajo y el repositorio de Git, donde se almacenan los cambios que se van a incluir en el próximo *commit*. Esto permite revisar y preparar los cambios antes de confirmarlos en el repositorio.

Commit

Un *commit* es una instantánea de los cambios realizados en el proyecto en un momento dado. Cada *commit* tiene un mensaje descriptivo que indica los cambios realizados y la razón por la que se realizaron.

Rama

Una rama es una línea de desarrollo independiente en un repositorio de Git. Cada rama tiene su propio historial de cambios y puede ser utilizada para trabajar en nuevas funcionalidades sin afectar la rama principal del proyecto.

Repositorio remoto

Un repositorio remoto es una copia del repositorio de Git en un servidor remoto, que permite a los usuarios colaborar en un proyecto de manera distribuida. Los cambios realizados en un repositorio remoto pueden ser descargados y fusionados en el repositorio local, y viceversa.

Comandos básicos de Git

Git tiene una cantidad considerable de comandos, pero algunos de los más utilizados son:

- `git init`: Inicializa un repositorio de Git en el directorio actual.
- `git add <archivo>`: Añade un archivo al *staging area*.
- `git commit -m «mensaje>"`: Confirma los cambios en el repositorio.
- `git status`: Muestra el estado actual del repositorio.
- `git log`: Muestra el historial de *commits* del proyecto.
- `git branch`: Muestra las ramas del proyecto.
- `git checkout <rama>`: Cambia a la rama especificada.
- `git merge <rama>`: Fusiona la rama especificada con la rama actual.
- `git clone <url>`: Clona un repositorio remoto en el directorio actual.

- `git pull`: Descarga los cambios del repositorio remoto y los fusiona con el repositorio local.
- `git push`: Sube los cambios del repositorio local al repositorio remoto.

Configuración de Git

Antes de empezar a utilizar Git es necesario configurar algunos parámetros básicos, como el nombre de usuario y la dirección de correo electrónico. Esto se puede hacer con los siguientes comandos:

```
git config --global user.name "Nombre de usuario"
git config --global user.email "Correo electronico @ciencias"
```

Una vez teniendo el usuario y el correo configurado, se puede verificar la configuración con el siguiente comando:

```
git config --list
```

Ahora toca configurar el editor de texto que se usará para los mensajes de *commit*. Usaremos el editor `emacs`:

```
git config --global core.editor "emacs"
```

Lo siguiente será configurar una llave SSH para poder autenticarnos con el servidor remoto de Git. Para ello, generamos una llave SSH con el siguiente comando:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Nos pedirá una contraseña, la cual podemos dejar en blanco. Pero esto no es recomendable, por lo que es mejor poner una contraseña segura.

Agregamos la llave SSH al *ssh-agent* con el siguiente comando:

```
eval "$(ssh-agent -s)"
```

Imprimimos en pantalla nuestra llave pública con el siguiente comando:

```
cat ~/.ssh/id_ed25519.pub
```

Copiamos la llave pública y la agregamos a nuestra cuenta de GitHub en la sección de *SSH and GPG keys*.

Existe algo llamado, archivo `config` en la carpeta `.ssh` que se encuentra en el directorio `home` del usuario. En este archivo se pueden agregar configuraciones para diferentes servidores. Por ejemplo, si queremos agregar una configuración para el servidor de GitHub, podemos agregar lo siguiente:

```
Host github.com
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_ed25519
```

Si tuvieramos distintas llaves SSH, podríamos agregar una configuración para cada una de ellas, por ejemplo una personal y otra para el trabajo:

```
Host gh_personal
  HostName github.com
  IdentityFile ~/.ssh/personal_key
  IdentitiesOnly yes

Host gh_work
    HostName github.com
    IdentityFile ~/.ssh/work_key
    IdentitiesOnly yes
```

Si hacemos la segunda solución, tendríamos que cambiar el *remote* de los repositorios de GitHub a `gh_personal` o `gh_work` según sea el caso.

Por ejemplo:

```
git remote set-url origin git@gh_personal:usuario/repo.git
```

Esto también cuando clonamos los repositorios:

```
git clone git@gh_personal:usuario/repo.git
```

1. Ejercicios

1.1. Ejercicio 1: Creación de un Repositorio y Subida de un Documento

En este ejercicio, vamos a crear un repositorio en GitLab, crear una rama y subir un documento que explique por qué decidiste estudiar Ciencias de la Computación. Sigue los pasos a continuación:

Paso 1: Crear un Repositorio en GitLab

1. Ingresa a tu cuenta de GitLab.
2. En la página principal, haz clic en el botón **"New project"**.
3. Selecciona **"Create blank project"**.
4. Rellena el nombre del proyecto, por ejemplo, **"MiRepositorioDeCienciasDeLaComputacion"**.
5. Opcional: Puedes añadir una descripción.
6. Selecciona la visibilidad del proyecto (público, privado, interno).
7. Haz clic en **"Create project"**.

Paso 2: Clonar el Repositorio

1. Clona usando SSH
2. Abre tu terminal o consola.
3. Escribe el siguiente comando:

```
    git clone <SSH-URL-del-repositorio>

```

4. Navega dentro del repositorio clonado:

```
    cd MiRepositorioDeCienciasDeLaComputacion

```

Paso 3: Crear una Nueva Rama

1. Crea una nueva rama llamada **"razones-eleccion-carrera"**:

```
    git checkout -b razones-eleccion-carrera

```

Paso 4: Crear y Subir el Documento

1. Crea un archivo llamado **“por_que_ciencias_de_la_computacion.txt”** en tu editor de texto preferido.
2. Escribe un párrafo explicando por qué decidiste estudiar Ciencias de la Computación.
3. Guarda el archivo en la carpeta del repositorio clonado.
4. Añade el archivo al índice de git:

```
git add por_que_ciencias_de_la_computacion.txt
```

5. Haz un commit de los cambios:

```
git commit -m "Add: Razones para elegir Ciencias de la Computacion"
```

6. Sube la rama al repositorio remoto:

```
git push origin razones-eleccion-carrera
```

Paso 5: Crear un Merge Request

1. Ve a tu repositorio en GitLab.
2. Haz clic en **“Merge Requests”**.
3. Haz clic en **“New merge request”**.
4. Selecciona la rama **“razones-eleccion-carrera”** como la fuente y **“main”** como el destino.
5. Añade un título y una descripción al merge request.
6. Haz clic en **“Create merge request”**.

1.2. Ejercicio 2: Crear un Repositorio y Subir un Documento (Instrucciones)

- Crea un nuevo repositorio en GitLab llamado **“EleccionCienciasDeLaComputacion”**.
- Clona el repositorio en tu máquina local.
- Crea una nueva rama llamada **“mi-rama-ciencias”**.
- Crea un archivo llamado **“porque_ciencias_de_la_computacion.md”** y escribe un párrafo sobre por qué elegiste esta carrera.
- Añade, comitea y sube el archivo a la nueva rama.
- Crea un merge request para fusionar **“mi-rama-ciencias”** con **“main”**.

1.3. Ejercicio 3: Crear un Repositorio y Subir un Documento (Instrucciones)

- Crea un nuevo repositorio en GitLab llamado **“EstudioCienciasComputacion”**.
- Clona el repositorio en tu máquina local.
- Crea una nueva rama llamada **“razones-carrera”**.
- Crea un archivo llamado **“motivos_ciencias_computacion.tex”** y escribe un párrafo explicando por qué te inclinaste por esta carrera.
- Añade, comitea y sube el archivo a la nueva rama.
- Crea un merge request para fusionar **“razones-carrera”** con **“main”**.