1. **¿Qué 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 repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan la copia del repositorio con la del servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.**

1. **¿Cuáles son los comandos más importantes y para qué sirven?**

**Git config:**

**El comando git config sirve para definir valores de configuración de Git a nivel de un proyecto global o local. Estos niveles de configuración se corresponden con archivos de texto con extensión .gitconfig.**

**Git init:**

**Este comando inicializa un nuevo** **repositorio en el repositorio local**.

### Git clone:

**Este comando inicializa un nuevo repositorio en el repositorio local** **clonando íntegramente el contenido de un repositorio remoto que le indiquemos mediante una URL**

### Git add:

**Una vez que hemos realizado los cambios necesarios en nuestra área de trabajo, para comenzar la confirmación de dichos cambios, es necesario pasar todos los archivos que queramos confirmar al área de preparación**

### Git Branch:

**Las ramas** **es uno de los conceptos más interesantes dentro de GIT. Permite el** **trabajo en paralelo** **sobre distintas partes del código base para añadir funcionalidades, corregir errores o realizar versiones de nuestros proyectos.**

### Git commit:

**Una vez que tenemos archivos preparados en el área de preparación, para confirmar** **dichos archivos y crear una confirmación de cambios la sentencia utilizada es git commit**.

1. **¿Cuál es la diferencia entre Git y GitHub?**

**GitHub es una plataforma donde puedes crear y mantener software e interactuar con otros desarrolladores.**

**Git es el programa que utiliza GitHub para hacer que su plataforma sea perfecta y funcional.**

1. **Enumere los comandos para subir cambios a un repositorio.**
2. **Ejecuta git init en la terminal. Esto inicializará la carpeta/repositorio que tienes en tu computador local.**
3. **Ejecuta git add . en la terminal. Esto hará un seguimiento de los cambios realizados en la carpeta de tu sistema desde el último commit. Como es esta la primera vez que haces commit a los contenidos de la carpeta, se añadirán todos.**
4. **Ejecuta git commit -m "inserta Mensaje aquí". Esto preparará los cambios añadidos/rastreados en la carpeta de tu sistema para empujar a Github. Puedes reemplazar *inserta el Mensaje aquí* con cualquier mensaje de confirmación relevante de tu elección.**
5. **Ejecuta git remote add origin https://github.com/nombreDeUsuario/repositorio.git en la terminal. Aquí, *nombreDeUsuario* y *repositorio* serán reemplazados por los valores proporcionados en el enlace copiado. Esto conectará la carpeta existente en tu sistema local al repositorio de Github recién creado.**
6. **Ejecuta git remote -v. Esto hace algo de magia usando *git pull* y *git push* para garantizar que el contenido de tu nuevo repositorio de Github y la carpeta en tu sistema local sean los mismos.**
7. **Finalmente, ejecuta git push origin master para empujar tus archivos a Github. Ten en cuenta que la última palabraen el comando *master*, no es una entrada fija cuando se ejecuta git push, puede ser reemplazada por cualquier “nombre\_de\_rama” relevante.**
8. **¿Qué es una branch?**

**Almacena una rama como referencia a un commit. En este sentido, una rama representa la punta de una serie de commit, no es un contenedor para los commit**. **El historial de una rama se extrapola a través de las relaciones entre commit, que recordamos es la fusión de una instantánea de los contenidos del repositorio en un momento determinado y la referencia al commit anterior.**

**En la práctica diaria, las ramas se utilizan para identificar una línea de desarrollo independiente, ya sea una corrección de errores o una nueva función. Cuando se crea una nueva rama, Git básicamente nos proporciona un nuevo set de working área, staging area e historial.**