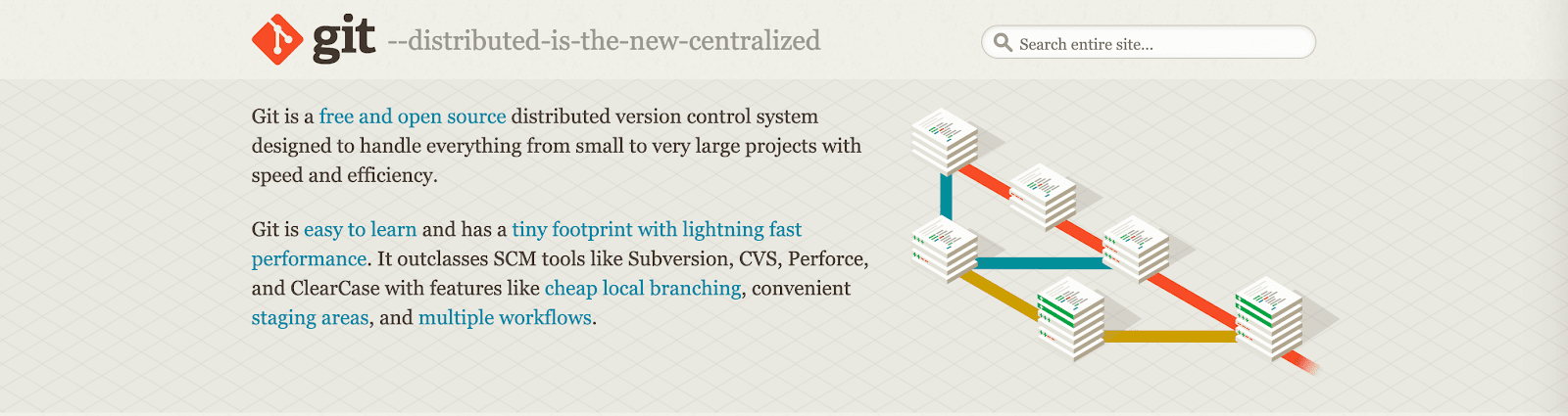
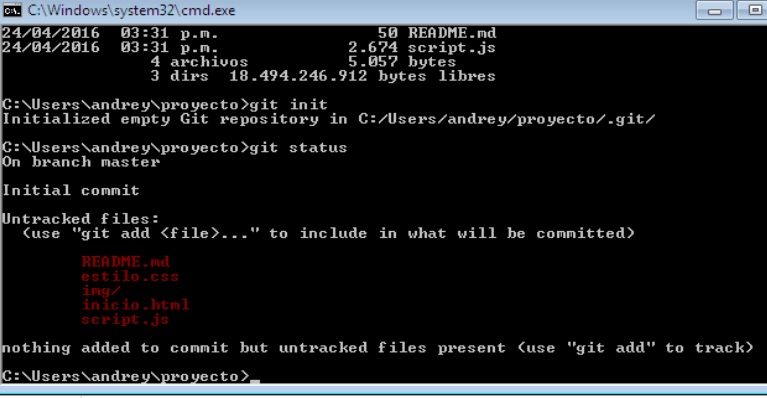
**GIT Y GITHUB**

**GIT**

Es un software de control de versiones creado para aumentar la eficiencia y compatibilidad del mantenimiento de versiones de aplicaciones cuando contienen un gran número de archivos. Su propósito es llevar registro de todos los cambios realizados en los archivos de un proyecto, también permite coordinar el trabajo que diferentes personas realizan sobre los archivos compartidos en un repositorio.





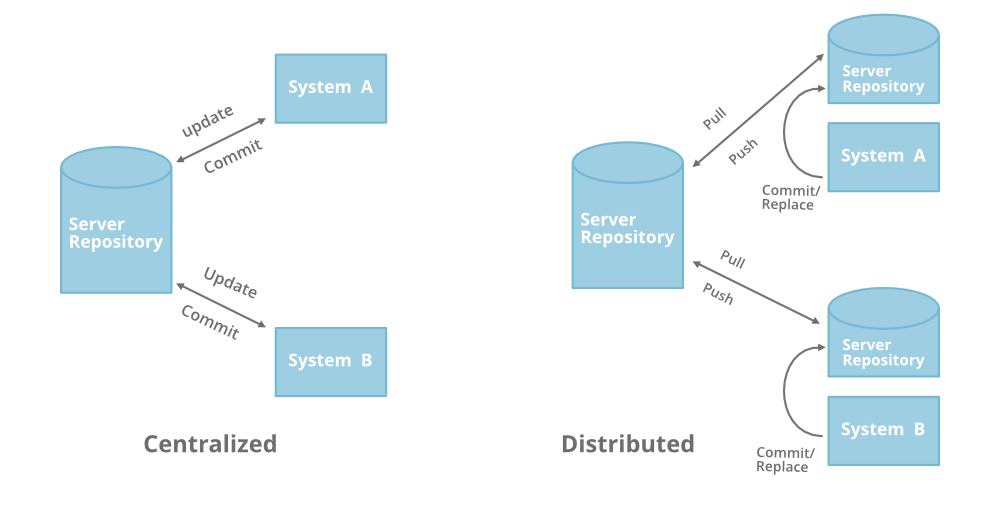
*Control de versiones*: Proceso de guardar diferentes archivos o versiones a lo largo de las diferentes etapas de un proyecto. Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y poder volver a una fase anterior si lo requieren.

El control de versiones es muy útil por las siguientes razones:

* Facilita la corrección de errores que puedan ocurrir durante el desarrollo.
* Permite revisar los cambios hechos en cada versión para ayudar a cualquier miembro del equipo a mantenerse al día sobre lo que se ha avanzado en el código.
* Se caracteriza por almacenar cada versión como una especie de captura de pantalla, permitiendo hacer referencia a versiones anteriores siempre que se necesite.

La mayoría de los sistemas de control de versiones utilizan un servidor centralizado para guardar todos los cambios y actualizaciones. Esta característica permite que personas en diferentes partes del mundo puedan trabajar sobre un mismo proyecto, esto conlleva a que el avance del proyecto dependa estrictamente de la conexión de, de otra manera los cambios no se podrán reflejar en el servidor central.

Git se presenta como un sistema distribuido, en el que todos los nodos manejan la información en su totalidad, por lo tanto, pueden actuar de cliente o servidor en cualquier momento, es decir, se elimina el concepto de sistema centralizado. Todo el avance del proyecto se hace directamente sobre una copia local y cuando se requiera, se suben los archivos modificados al repositorio remoto.

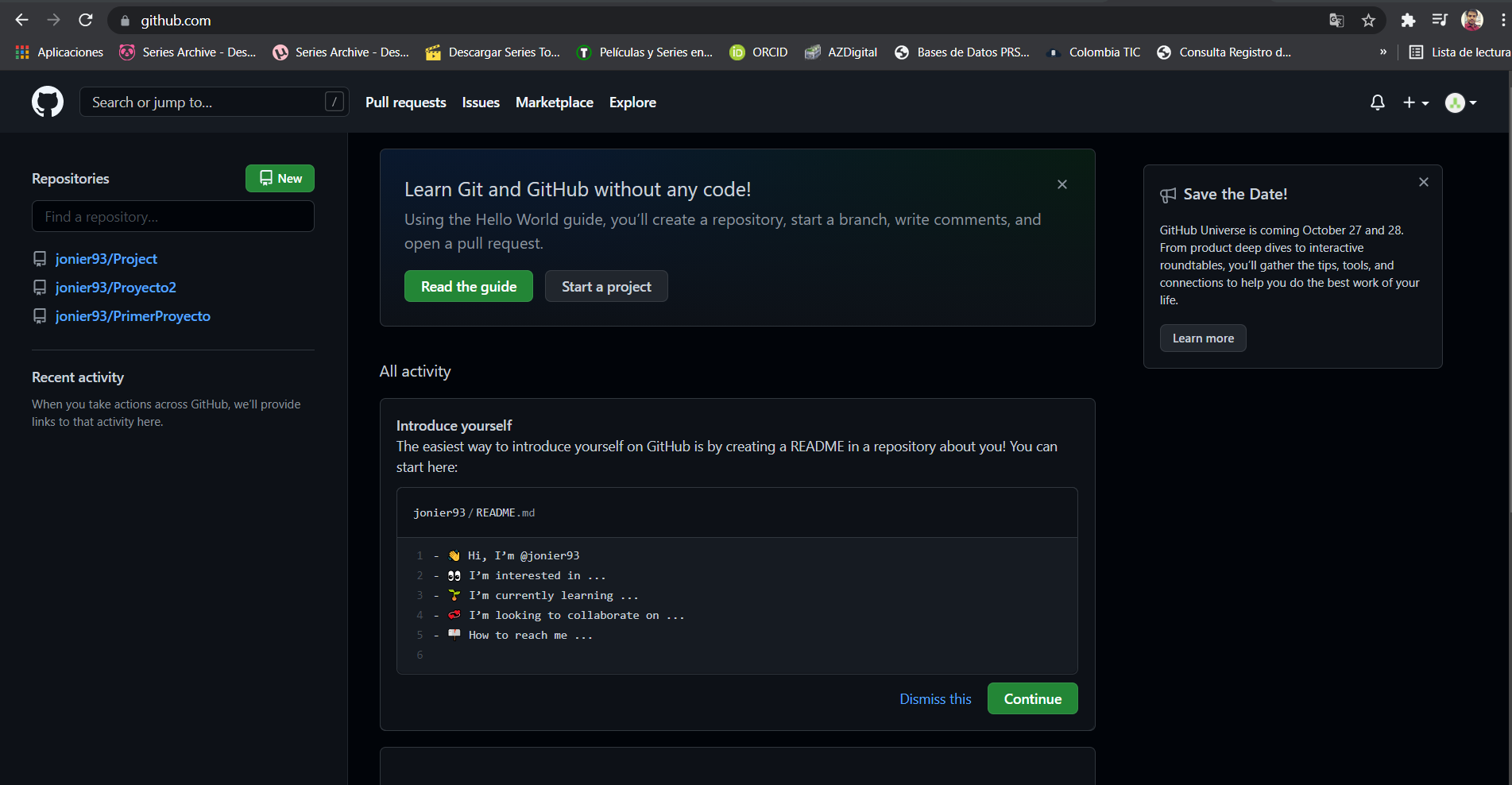


Sin embargo, hay algunas desventajas de tener un sistema distribuido, ya que al ser un software local, los diferentes desarrolladores que se encuentran trabajando en un proyecto simultáneamente, no podrán ver el trabajo del otro en tiempo real.

**GITHUB**

[GitHub](https://github.com/) facilita la colaboración con Git. Es una plataforma compuesta por repositorios de código, utilizados para [almacenarse](https://kinsta.com/es/blog/cuota-de-mercado-de-la-nube/) en la nube (internet), de tal forma que múltiples desarrolladores puedan trabajar en un mismo proyecto simultáneamente y ver las versiones de cada uno. GitHub también incluye funciones de organización y gestión de proyectos, permite asignar tareas, establecer permisos y roles para los colaboradores.

Cabe mencionar que los repositorios de GitHub están disponibles públicamente para todos los desarrolladores del mundo, de esta manera pueden contribuir al código de otros desarrolladores para realizar mejoras.



**comandos de GIT**

***git init***: Permite crear un repositorio local.

***git add:*** Sirve para agregar archivos al entorno de trabajo.

***git status***: Muestra la lista de archivos agregados o modificados en el entorno de trabajo.

***git commit***: Permite guardar un cambio en el repositorio local.

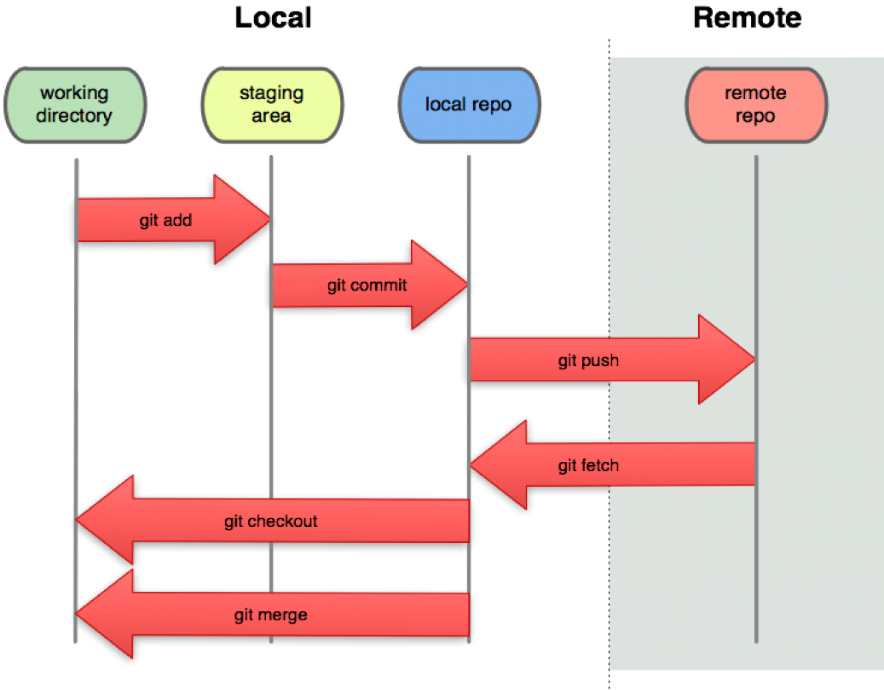
***git config:*** Establece una configuración específica de usuario, tal como email y usuario.

***git remote***: Conexión a repositorio remoto.

***git push***:  Envía el proyecto al repositorio remoto, en nuestro caso a GitHub.

***git clone:*** Permite clonar un repositorio.

***git branch:*** Se utiliza para crear ramas.



**GUÍA**

1. Descargar e instalar Git <http://git-scm.com/download/win>
2. Se crea el proyecto utilizando visual studio code
3. Se ingresa a la carpeta donde se va a localizar el proyecto, utilizando la consola de comandos: *cd ruta\_carpeta*
4. Se inicializa el repositorio git mediante el comando: *git init*
5. Se agregan los archivos al entorno de trabajo de git mediante el comando: *git add Nombre\_Archivo*
6. Para agregar todos los archivos al mismo tiempo se utiliza el comando: *git add .*
7. Se modifican archivos del proyecto y se utiliza el comando: *git status*
8. Para comparar los cambios realizados a un archivo se utiliza el comando*: git diff Nombre\_Archivo*
9. Se guarda en el repositorio local de git la versión actual: *git commit*
10. La primera vez pide un email y un nombre para crear el repositorio local, utilizar los comandos: *git config --global user.email “correo\_electronico”* y *git config --global user.name “nombre”*
11. Alternativamente se puede guardar en el repositorio local utilizando el siguiente comando: *git commit -m “mensaje de version”*
12. Se revisa las versiones creadas mediante el comando: *git log*
13. Se puede Volver a una versión anterior utilizando el comando: *git checkout id\_version*
14. Para crear una versión alternativa se utiliza el comando: *git branch Nombre\_Rama*
15. Para cambiar de rama se utiliza el comando: *git checkout Nombre\_rama*
16. Para guardar el proyecto en el repositorio remoto se utiliza el comando: *git remote add origin Dirección\_web*
17. Si hay problemas con la conexión del repositorio remoto, utilizar el comando: *git remote remove origin* o el comando: *git remote rm origin*
18. Para subir finalmente el proyecto al repositorio se utiliza el comando: *git push -u origin master*
19. La primera vez va a pedir logearse en la ventana emergente GitHub
20. Si hay problemas con actualización del repositorio a una versión actual, se debe eliminar el master con el siguiente comando: *git branch -D master* y luego reasignar al master a la versión actual con*: git checkout -b master*
21. Para descargar un archivo de GitHub utilizar el comando: *git clone dirección\_web*