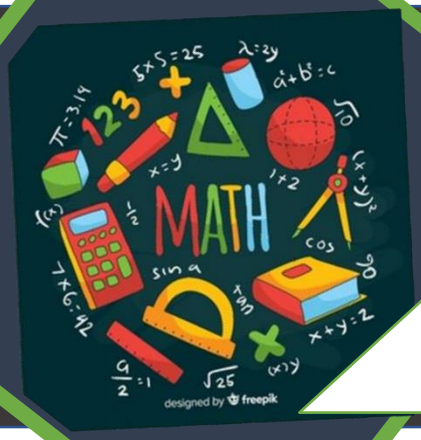
UNIVERSIDAD DE COLIMA



16/05/2022

Computo en la nube

CARLOS ALEJANDRO BALTAZAR PADILLA

6 B

En que consiste GIT

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. 2Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. 3Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores. En cuanto a derechos de autor Git es un software libre distribuible bajo los términos de la versión 2 de la Licencia Pública General de GNU.

En que consiste GITHUB

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública.

El 4 de junio de 2018, Microsoft compró GitHub por la cantidad de 7500 millones de dólares.12 Al inicio, el cambio de propietario generó preocupaciones y la salida de algunos proyectos de este sitio;3 sin embargo, no fueron representativos. GitHub continúa siendo la plataforma más importante de colaboración para proyectos de código abierto.

Ventajas y des ventajas de GIT

Ventajas

- **Compartición selectiva** : El desarrollo de la aplicación serán únicamente nuestro, pudiendo decidir qué parte de nuestro proyecto compartimos y con quién, restringiendo a que sólo pueda verlo, que tenga la posibilidad de añadir notas, comentarios o que pueda añadir cambios. No todo tiene porque ser público (aunque en la compartición encontramos la riqueza y agilidad que

fundamenta el software libre ;).

- **Velocidad** : Muchas empresas deciden implementar GIT como servicio local en su infraestructura física, por lo tanto el control de versiones se realizaría dentro de la propia red con la consiguiente ganancia en velocidad de acceso y escritura, así como eliminando el requisito de contar con una conexión a internet obligatoria. No obstante, contar con el respaldo de un hosting para estos menesteres siempre es un plus de seguridad pues ganaremos la despreocupación para con respecto de la seguridad y accesibilidad de nuestro trabajo.

- **Ramificación** : Ya hablamos de la ramificación **en una entrada anterior** , y como vimos ofrece un amplio abanico de posibilidades a la hora de realizar cambios en la estructura principal, pudiendo crear diferentes ramas sobre las que aplicar nuestras modificaciones en entornos aislados de la línea principal de desarrollo.

- **Convergencia** : Si en la creación de una rama del proyecto encontramos que uno de los cambios incluidos se integra tal y como deseamos, sin presentar conflictos con las diferentes partes de nuestra aplicación, podremos incluir o hacer converger dicha ramificación con el desarrollo principal de forma sencilla y segura, contando así con una nueva versión o revisión de nuestro proyecto lista para ser distribuida, compartida, liberada…

- **Sandbox** : Esta sería una ventaja de una ventaja, ya que las ramificaciones nos preparan un entorno aislado de pruebas sobre el desarrollo de la línea central de nuestra app. Los cambios realizados en una de las ramas del proyecto no tendrán consecuencias para los usuarios que actualmente usen o accedan a la versión sin modificar o principal. Muy útil si lo que vamos a hacer es incluir servicios que antes no existían en nuestra aplicación y esto pudiese poner en peligro la estabilidad de otros componentes.

- **Flujo de trabajo adaptable** : En el sector de los controladores de versiones encontraremos diferentes formas para gestionar el flujo de desarrollo de la aplicación, destacando entre éstos los modelos centralizados y los modelos de libre configuración. Con esto encontramos que trabajemos como trabajemos encontraremos un control de versiones que se adapte a nosotros o nuestra empresa, haciendo uso desde un simple sistema jerárquico hasta un ligeramente más complejo sistema centralizado. En este apartado Git gana puntos sobre la competencia, admitiendo multitud de configuraciones que nos permitirán dentro de su estructura organizar el trabajo tal y como deseemos nosotros, nuestro equipo de desarrollo, etc…

- **Seguridad** : Pero… ¿y si tengo un sistema de control de versiones instalado en local y tengo una caída de la tensión eléctrica que provoca el apagado de la infraestructura? ¿Los datos se habrán corrompido en el proceso de escritura?

La respuesta es que resultaría muy complejo, ya que en su inmensa mayoría, los controladores de versiones cuentan con sistemas de cifrado y otros tipos de medidas de seguridad que se aplicarán para que nuestros datos permanezcan lo más íntegros posible. Por poner un ejemplo, Git hace uso de **sistemas de árbol SHA1** , lo que asegurará que hasta que no se realice la comprobación del cifrado o firma, los cambios no se escribirán en el servidor.

- **Coste** : Obviamente podremos encontrar software de control de versiones que nos ofrezcan las mismas o similares características que nos otorga Git o BitBucket entre otros, pero lo que será complejo es que alguna de estas alternativas sea gratuita. Los sistemas de hosting pueden llegar a presentar algún coste, pero entrará en nuestro juicio contemplar si realizar una pequeña inversión en infraestructura online (asegurando ya de paso un poco más el acceso y seguridad de nuestro proyecto) o adquirir una costosa solución similar a los ejemplos ya citados. Poniéndonos en la piel de un empresario, creo que cuanto más consigamos ahorrar a la empresa mejor para la empresa (y para nosotros demostrando preocupación por los recursos económicos de la misma ;).

Desventaja

- **Aprendizaje** : Es algo por lo que tendremos que pasar cada vez que queramos incluir alguna nueva tecnología en un flujo de trabajo ya establecido. Deberemos formar al equipo de desarrollo o a aquellas personas a las que tendremos que dar acceso a nuestro trabajo en el servicio de control de versiones; para evitar así posibles errores en la realización de cambios (salvables por su puesto al estar ahí el registro de cambios y versiones ^_^).

Ventajas y des ventajas de GITHUB

Ventajas

- Servicio gratuito, aunque también tiene servicios de pago.
- Búsqueda muy rápida en las estructura de los repos.
- Amplia comunidad y fácil encontrar ayuda.
- Ofrece prácticas herramientas de cooperación y buena integración con Git.
- Fácil integrar con otros servicios de terceros.
- Trabaja también con TFS, HG y SVN.

Desventajas

- No es absolutamente abierto.
- Tiene limitaciones de espacio, ya que no puedes exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis.

Qué tipo de archivos trabaja git

En Git existen 4 tipos de archivos: Blob, Tree, Commit y Annotated Tag. Estos cuatro tipos de objetos permiten almacenar archivos y monitorear los cambios en estos.

Qué tipo de archivos trabaja github

Los archivos que agregues a un repositorio mediante un navegador están limitados a 25 MB por archivo. Puedes agregar archivos más grandes, de hasta 100 MB cada uno, mediante la línea de comando. Para obtener más información, consulta "Agregar un archivo a un repositorio mediante la línea de comando".

Trabaja con lenguajes como:

- JavaScript
- Java
- Ruby
- PHP
- Python
- CSS
- C++
- C#
- C
- HTML

Instalación GIT

En Windows, sólo tienes que descargar el instalador y ejecutarlo. Sigue estos sencillos pasos para hacerlo:

1. Descarga el instalador de GIT para Windows.
2. Una vez que hayas descargado el instalador, haz doble clic sobre el ejecutable para que comience el proceso de instalación y sigue las instrucciones que te aparecerán en pantalla. Al igual que cualquier otro programa, tendrás que dar “Next” (siguiente) en varias ocasiones hasta que aparezca la opción “Finish” (terminar) para completar la instalación.\

3. Ahora tienes que abrir el símbolo de sistema y escribir los siguientes comandos en la terminal:

```
git config --global user.name "Tu nombre"


git config --global user.email "ejemplo@email.com"
```

Recuerda que debes de cambiar Tu Nombre y ejemplo@email.com por tu información.

¡Y listo! Ya has instalado GIT en Windows.

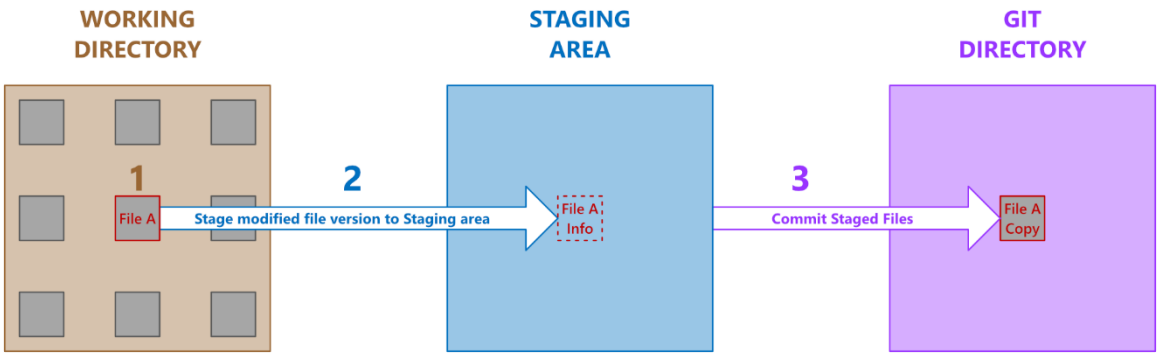
Descargar e instalar GitHub Desktop

Puedes instalar GitHub Desktop en Windows 7 64-bit o posterior.

Advertencia: Debes tener un sistema operativo de 64 bits para ejecutar GitHub Desktop.

1. Visita la página de descargas para GitHub Desktop .
2. Da clic en Descargar para Windows.
3. El botón de Descargar para Windows
4. En la carpeta Download de tu computadora, da doble clic en el archivo de configuración de GitHub Desktop.
5. El archivo de GitHubDesktopSetup
6. GitHub Desktop se lanzará después de que se complete la instalación.

Como funciona GIT



Modificar archivos en el directorio de trabajo.

Observe que cualquier archivo que modifiques se convierte en un archivo en el estado modificado.

Prepare selectivamente los archivos que quieras confirmar al directorio .git.

Observe que cualquier archivo que prepares (agregues) a la zona de preparación se convierte en un archivo en el estado preparado.

También tenga en cuenta que los archivos preparados todavía no están en la base de datos .git.

Preparar significa que la información sobre el archivo preparado se incluye en un archivo (llamado "index") en el repositorio .git.

Confirme el/los archivos que has preparado en el directorio .git. Esto es, guardar de manera permanente una instantánea de los archivos preparados en la base de datos .git.

Observe que cualquier versión del archivo que confirmes al directorio .git se convierte en un archivo en el estado confirmado.

Lo esencial hasta ahora

En resumidas cuentas de todo lo discutido hasta el momento es que Git es un sistema de control de versiones genial para versionado, administración y distribución de archivos.

Pero espera un segundo, si Git nos ayuda en la administración y distribución eficaz de distintas versiones de los archivos de un proyecto, ¿cúal es el propósito de GitHub?

Como funciona GITHUB

¿Qué herramientas proporciona?

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el **trabajo en equipo**. Entre ellas, caben destacar:

- Una **wiki** para el mantenimiento de las distintas versiones de las páginas.
- Un **sistema de seguimiento de problemas** que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una **herramienta de revisión de código**, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un **visor de ramas** donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

Diferencias entre GIT y GITHUB

Diferencia 1: Git vs. GitHub — Función principal

Git es un sistema de control de versiones distribuido que registra las distintas versiones de un archivo (o conjunto de archivos). Le permite a los usuarios acceder, comparar, actualizar, y distribuir cualquiera de las versiones registradas en cualquier momento.

Sin embargo GitHub principalmente es una plataforma de alojamiento para albergar tus repositorios Git en la web. Esto permite a los usuarios mantener sus repositorios remotos privados o abiertos para esfuerzos colaborativos.

Diferencia 2: Git vs. GitHub — Plataforma de operación

Los usuarios instalan y ejecutan Git en sus equipos locales. Esto significa que la mayoría de las operaciones de Git se pueden lograr sin una conexión a internet.

Sin embargo GitHub es un servicio basado en la web que opera solamente en línea. Esto significa que necesitas estar conectado para hacer cualquier cosa en GitHub.

Diferencia 3: Git vs. GitHub — Creadores

Linus Torvalds comenzó Git en Abril del 2005.

Chris Wanstrath, P. J. Hyett, Tom Preston-Werner, y Scott Chacon fundaron GitHub.com en Febrero 2008.

Diferencia 4: Git vs. GitHub — Mantenedores

En Julio 2005, Linus Torvalds entregó el mantenimiento de Git a Junio C. Hamano — quien ha sido el principal mantenedor desde entonces.

En Octubre 2018, Microsoft compró GitHub.

Diferencia 5: Git vs. GitHub — Competidores

Algunas alternativas populares para Git son Mercurial, Team Foundation Version Control (TFVC), Perforce Helix Core, Apache Subversion, y IBM Rational ClearCase.

Los competidores más cercanos a GitHub son GitLab, Bitbucket, SourceForge, Cloud Source Repositories, y AWS CodeCommit.