UNIVERSIDAD DE COLIMA
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
Ingeniería en Computación Inteligente

Diferencias entre Git y GitHub

Profesor: Carrillo Zepeda Oswaldo.
Alumno: Guerrero Gómez Brandon Yair.

6° B

Fecha de entrega: Martes, 17 de mayo de 2022

En que consiste

GIT

Git, es un software de control de versiones diseñado por Linus Torvalds. Pues bien, se define como control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo, es decir, a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración.

En pocas palabras, un sistema de control de versiones son todas las herramientas que nos permiten hacer todas esas modificaciones antes mencionadas en nuestro código y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida, y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas.

Algunas de las características más importantes de Git son:

- Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida; Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Realmacenamiento periódico en paquetes.

GITHUB

GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones, llamado Git.

Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real. Es decir, nos permite centralizar el contenido del repositorio para poder colaborar con los otros miembros de nuestra organización.

GitHub está basada en el sistema de control de versiones distribuida de Git, por lo que se puede contar con sus funciones y herramientas, aunque GitHub ofrece varias opciones adicionales y su interfaz es mucho más fácil de manejar, por lo que no es absolutamente necesario que las personas que lo usan tengan un gran conocimiento técnico.

GitHub permite llevar un registro y control de los cambios acontecidos en el código fuente de un proyecto. Como es de suponerse, esto conlleva una gran mejora sobre el flujo de trabajo. Por una parte, evita cualquier daño grave en el código fuente principal y, por otra parte, posibilita la coordinación y división del trabajo entre los participantes de un mismo proyecto.

GitHub cuenta con una versión gratuita y con una versión paga. La versión gratuita es bastante popular porque permite gestionar proyectos de una forma sencilla y ordenada. En cuanto a su versión paga, esta resulta muy atractiva para organizaciones que buscan una plataforma que garantice organización del trabajo y seguridad.

Ventajas y desventajas

GIT

Ventajas

- Sistema distribuido, sin un punto central de fallo, que permite el trabajo incluso sin conexión.
- Super rápido y ligero, optimizado para hacer operaciones de control muy rápidas.
- Crear ramas y mezclarlas es rápido y poco propenso a problemas, al contrario que en otros sistemas tradicionales.
- La integridad de la información está asegurada gracias a su modelo de almacenamiento, que permite predecir este tipo de problemas. En sistemas tradicionales este era un problema grave.
- Permite flujos de trabajo muy flexibles.
- El concepto de área de preparación o staging permite versionar los cambios como nos convenga, no todo o nada.
- ¡Es gratis! y de código abierto.

Desventajas

- Es más complejo que los sistemas centralizados tradicionales porque entran en juego más repositorios, más operaciones y más posibilidades para trabajar en equipo, que hay que decidir.
- La curva de aprendizaje es empinada. Lo básico lo aprendes enseguida, pero la realidad te demuestra que no es suficiente "tocar de oído" con él. La documentación es tan compleja que muchas veces no resulta de ayuda.
- Los comandos y algunos conceptos que usa pueden llegar a ser confusos, al igual que algunos mensajes que muestra.
- Por defecto, se lleva mal con archivos binarios muy grandes, como vídeos o documentos gráficos muy pesados. Por suerte existen soluciones para ello (Git LFS).

GitHub

Ventajas

- GitHub permite que alojemos proyectos en repositorios de forma gratuita
- Te brinda la posibilidad de personalizar tu perfil en la plataforma
- Los repositorios son públicos por defecto. Sin embargo, GitHub te permite también alojar tus proyectos de forma privada
- Puedes crear y compartir páginas web estáticas con GitHub Pages
- Facilita compartir tus proyectos de una forma mucho más fácil y crear un portafolio
- Te permite colaborar para mejorar los proyectos de otros y a otros mejorar o aportar a los tuyos
- Ayuda reducir significativamente los errores humanos y escribir tu código más rápido con GitHub Copilot

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.

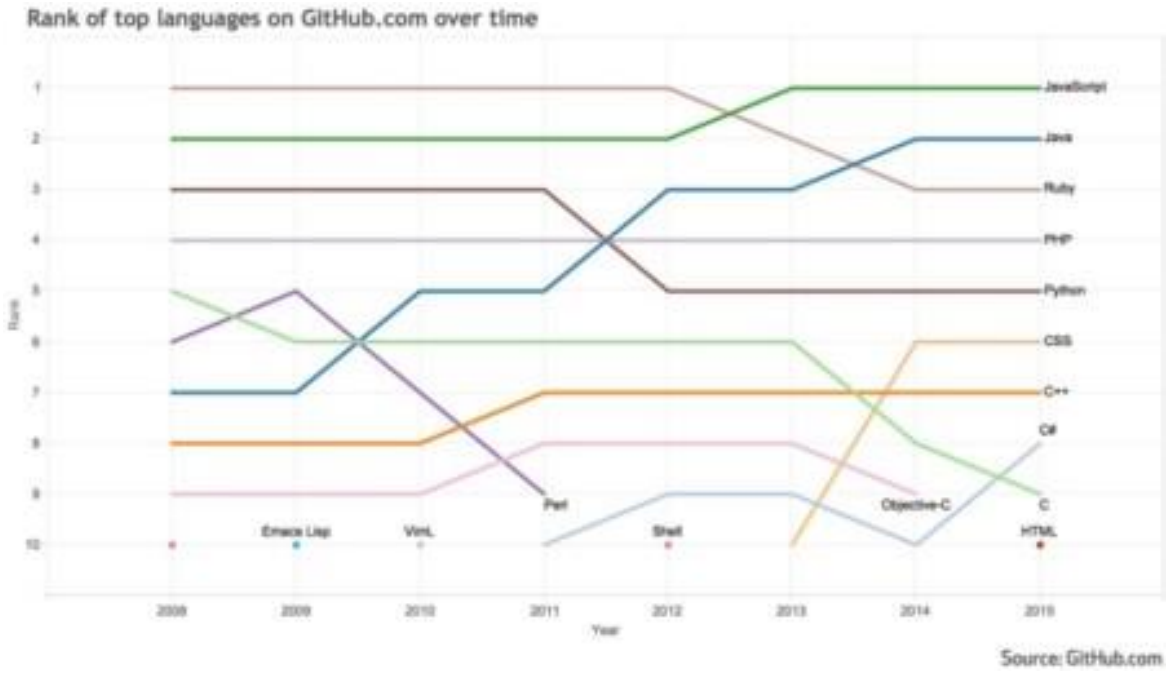
Qué tipo de lenguajes trabaja

En concreto, estos son los preferidos:

1. JavaScript
2. Java
3. Ruby
4. PHP
5. Python
6. CSS
7. C++
8. C#
9. C
10. HTML

Al margen de lo dicho, cabe mencionar el significativo crecimiento de GitHub, que actualmente supera los 10 millones de usuarios, una cifra espectacular si tenemos en cuenta que solo tres años antes contaba con 1,7.

De hecho y, según algunos expertos, esta herramienta se ha convertido en el principal medio de software de código abierto del mundo, pero no solo eso, sino que, según el portal especializado Arstechnica, “el número de empresas que están utilizando el servicio de código privado” va en aumento.



Como se instala y cómo funciona

GIT

Instalación

Descargue e instale Git para Windows. Una vez instalado, Git estará disponible desde el símbolo del sistema o PowerShell. Se recomienda seleccionar los valores predeterminados durante la instalación, a menos que haya una buena razón para cambiarlos.

Git para Windows no se actualiza automáticamente. Para actualizar Git para Windows, descargue la nueva versión del instalador, que actualizará Git para Windows en su lugar y conservará toda la configuración.

Funcionamiento

Un repositorio de Git, o repositorio, es una carpeta en la que Git realiza un seguimiento de los cambios. Puede haber cualquier número de repositorios en un equipo, cada uno almacenado en su propia carpeta. Cada repositorio git de un sistema es independiente, por lo que los cambios guardados en un repositorio de Git no afectan al contenido de otro.

Un repositorio de Git contiene todas las versiones de cada archivo guardado en el repositorio. Esto es diferente de otros sistemas de control de versiones que almacenan solo las diferencias entre los archivos. Git almacena las versiones de archivo en una carpeta .git oculta junto con otra información que necesita para administrar el código. Git guarda estos archivos de forma muy eficaz, por lo que tener un gran número de versiones no significa que use mucho espacio en disco.

Almacenar cada versión de un archivo ayuda a Git a combinar código mejor y facilita el trabajo con varias versiones del código.

Los developers funcionan con Git a través de comandos emitidos mientras trabajan en un repositorio local en el equipo. Incluso al compartir código u obtener actualizaciones del equipo, se realiza desde comandos que actualizan el repositorio local. Este diseño centrado en el entorno local es lo que convierte a Git en un sistema de control de versiones distribuido. Cada repositorio es independiente y el propietario del repositorio es responsable de mantenerlo al día con los cambios de otros.

La mayoría de los equipos usarán un repositorio central hospedado en un servidor al que todos los usuarios puedan acceder para coordinar sus cambios. El repositorio central normalmente se hospeda en una solución de administración de control de código fuente, como GitHub o Azure DevOps, que agrega características que facilitan el trabajo conjunto.

Configurar Git

Una vez instalado nuestro cliente Git, lo primero que tenemos que hacer es configurar nuestro nombre de usuario y dirección de correo electrónico.

Si utilizamos la opción `–global`, esto solo lo tenemos que hacer una vez, puesto que a partir de entonces lo tomará por defecto. Así por ejemplo,

```
git config --global user.name "atareao"
git config --global user.email atareao@atareao.es
```

Si en un proyecto concreto queremos utilizar otro usuario o email, lo que debemos hacer es no añadir la opción `–global`.

Estos dos parámetros son importantes porque las confirmaciones de cambios utilizan esta información.

El resto de parámetros a configurar ya son opcionales. Algunos que te pueden ser de utilidad son:

- El editor de texto. En mi caso particular utilizo nano cuando edito en el emulador de terminal:
  `git config --global core.editor nano`
- Herramienta de diferencias. La herramienta de diferencias es la que se encarga para resolver conflictos de unión:
  `gif config --global merge.tool vimdiff`
- Por último si quieres ver tu configuración, puedes ejecutar la siguiente orden en el emulador de terminal:
  `git config --list`

Creación de un nuevo repositorio de Git

Hay dos opciones para crear un repositorio de Git. Se puede crear a partir del código de una carpeta del equipo, se puede clonar desde un repositorio existente. Decidir qué opción es necesaria es sencilla. Si trabaja con código que está solo en el equipo local, cree un repositorio local con el código de esa carpeta. Sin embargo, la mayoría de las veces el código ya se comparte en un repositorio de Git, por lo que la clonación del repositorio existente en el equipo local es la forma de hacerlo.

Desde código existente
Use el comando `git init` para crear un repositorio a partir de una carpeta existente en el equipo. Desde la línea de comandos, vaya a la carpeta raíz que contiene el código y ejecute el siguiente comando para crear el repositorio.

```
> git init
```

A continuación, agregue los archivos de la carpeta a la primera confirmación mediante los siguientes comandos:

```
> git add --all
> git commit -m "Initial commit"
```

Desde un repositorio remoto
Use el comando `git clone` para copiar el contenido de un repositorio existente en una carpeta del equipo. Desde la línea de comandos, vaya a la carpeta que contiene el repositorio clonado y, a continuación, ejecute:

```
> git clone
https://fabrikam.visualstudio.com/DefaultCollection/Fabrikam/_git/Fa
brikamProject
```

Pase la dirección URL real al repositorio existente en lugar de la dirección URL del marcador de posición anterior. Esta dirección URL, denominada dirección URL de clonación, apunta a un servidor donde el equipo coordina los cambios. Obtenga esta dirección URL del equipo o del botón Clonar del sitio donde se hospeda el repositorio.

No es necesario agregar archivos ni crear una confirmación inicial cuando se clona el repositorio, ya que todo se copió, junto con el historial, del repositorio existente durante la operación de clonación.

GitHub

Instalación

GitHub Desktop es una aplicación de escritorio gráfica que permite agregar archivos y otras operaciones git relacionadas. Esta aplicación asistida por GUI se puede descargar desde `https://desktop.github.com/`

Puedes usar el cliente de GitHub Desktop localmente sin conectarte a un repositorio remoto. La aplicación GitHub Desktop es extremadamente simple, puedes inicializar un repositorio, organizar archivos, confirmar problemas, etc.

Funcionamiento

Aunque la interfaz es fácil de utilizar, GitHub te ofrece una guía para crear y administrar los repositorios, acceder y realizar cambios en códigos, entre otros.

Si te has animado a tener tu espacio para compartir proyectos, ten en cuenta estos pasos:

1. Crea un repositorio
   1.1. Haz clic en la esquina superior derecha junto a tu foto de perfil y luego selecciona `New Repository`.
   1.2. Nombra tu repositorio, por ejemplo: `Prueba`.
   1.3. Escribe una breve descripción del proyecto.
   1.4. Selecciona la opción: `Initialize this repository with a README`.

2. Crea una división (`branch`) o bifurcación
   Por lo general, la rama principal en la que quedará alojado el código fuente original se denomina `master`, mientras que las bifurcaciones o copias que se creen para ser modificadas se llamarán `feature`.
   2.1. Dirígete a tu nuevo repositorio `Prueba`.
   2.2. Haz clic en el menú desplegable llamado `branch: master`.
   2.3. Escribe el nombre de una división en el cuadro de texto, por ejemplo, `readme-edits`.
   2.4. Selecciona la casilla azul `Create branch` o presiona Enter en tu teclado.
3. ¡Realiza cambios y comete errores!
   En GitHub, cada cambio que se realiza es denominado `Commits` y queda asociado a un `Commits message` para explicar el historial de ajustes que se hicieron. Esto permitirá que otros colaboradores comprendan los cambios.
   3.1. Haz clic en el archivo `README`.
   3.2. Haz clic en el ícono de lápiz ubicado en la esquina superior derecha del archivo para editar.
   3.3. En el editor, escribe un poco sobre tu perfil y lo que desees que otros usuarios conozcan de ti.
   3.4. Escribe un mensaje que explique los cambios que has realizado para que los colaboradores comprendan los ajustes en el código.
   3.5. Haz clic en el botón `Commit change`.

4. Abre una solicitud de extracción (si modificaste el archivo de alguien más)
   Una vez que has terminado las modificaciones en un proyecto y estás seguro que funciona adecuadamente, puedes enviar una solicitud de extracción a un usuario, para que tenga en cuenta tu versión, la revise y la agregue a su código fuente o versión master.

4.1. Haz clic en la pestaña `Pull Request`.
4.2. Luego, haz clic en el botón verde `New Pull Request`.
4.3. En el cuadro `Example Comparisons`, selecciona la división que creaste de del archivo `README` para comparar con el código original `master`.
4.4. Revisa tus cambios y asegúrate de que sean los que deseas enviar. En verde aparecerán los elementos agregados y en rojos los eliminados.
4.5. Haz clic en el botón verde `Create Pull Request`.
4.6. Asigna un título a tu solicitud y escribe una breve descripción de tus cambios.

5. Combina tu solicitud de extracción (si alguien más realizó modificaciones de tu proyecto)
   5.1. Haz clic en el botón verde `Merge Pull Request` para combinar los cambios que realizaron a tu código con la versión `master`.
   5.2. Haz clic en `Confirm merge`.
   5.3. Ya que los cambios se han incorporado, elimina la división con el botón `Delete branch` en el cuadro púrpura.

Diferencias que existen entre los dos

Git es un software independiente que necesita ser instalado. La instalación diferirá entre los diferentes sistemas operativos. Lo mejor es verificar qué se debe hacer para que Git funcione en nuestro sistema específico.

Además, Git se usa a menudo a través de la CLI (interfaz de línea de comandos). También están disponibles herramientas GUI (interfaz gráfica de usuario) integradas y de terceros.
Otra forma popular de usar Git es directamente a través de un editor de texto o IDE. Muchos IDE admiten Git de forma nativa. Otros pueden requerir la instalación de un complemento o agregar una extensión.

Cuando se trabaja en un proyecto de forma independiente, usar Git y configurar un repositorio remoto siempre es una buena idea. Es mejor tener un sistema de revisión para usarlo como respaldo, que perder el tiempo reconstruyendo los datos perdidos.

También debemos tener en cuenta si queremos que nuestro proyecto esté disponible públicamente (código abierto) o privado. GitHub ofrece ambas opciones como un medio para compartir su código o mantenerlo confidencial.

Ahora, para responder a la famosa pregunta ¿Cuál es la diferencia entre Git y GitHub? La respuesta es simple, Git es un sistema de control de versiones que nos permite administrar y rastrear el historial de nuestro código fuente. Mientras que GitHub es el medio de almacenamiento basado en la nube que nos permite mantener esos cambios en un repositorio remoto. El uso conjunto de Git y GitHub

es una opción sensata para mantener el desarrollo continuo de una base de código.