UNIVERSIDAD DE COLIMA

Dirección General de Educación Superior

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN INTELIGENTE

Semestre febrero 2022 – junio 2022

25/05/2022

Nombre: Ronaldo Adán Avalos de la Mora. **Semestre:** 6

Materia: Computo en la nube **Práctica:** Diferencias entre GIT y GitHub Desktop

Diferencias entre GIT y GitHub Desktop.

Git.

¿Que es?

Git es un sistema de control de versiones distribuido. Esto 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 fácilmente. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el 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.

¿Como funciona?

Básicamente, Git funciona del siguiente modo:

1. Crea un "repositorio" (proyecto) con una herramienta de alojamiento de Git (por ejemplo, Bitbucket)
2. Copia (o clona) el repositorio a tu máquina local
3. Añade un archivo a tu repositorio local y confirma ("commit") los cambios
4. Envía ("push") los cambios a la rama principal
5. Haz cambios en tu archivo con una herramienta de alojamiento de Git y confírmalos
6. Extrae ("pull") los cambios a tu máquina local
7. Crea una rama ("branch", versión), haz algún cambio y confírmalo
8. Abre una solicitud de incorporación de cambios ("pull request": propón cambios a la rama principal)
9. Fusiona ("merge") tu rama con la rama principal

Ventajas.

- *Desarrollo simultáneo.*

Todos tienen su propia copia local de código y pueden trabajar simultáneamente en sus propias ramas. Git funciona sin conexión, ya que casi todas las operaciones son locales.

- *Versiones más rápidas.*

Las ramas permiten un desarrollo flexible y simultáneo.

- *Integración.*

Git se integra en la mayoría de las herramientas y productos. Cada IDE principal tiene compatibilidad integrada con Git y muchas herramientas admiten la integración continua. Esta integración simplifica el flujo de trabajo diario.

- *Soporte técnico de la comunidad sólida.*

Git es de código abierto y se ha convertido en el estándar de facto para el control de versiones, por lo que no hay escasez de herramientas y recursos disponibles para que los equipos los aprovechen.

- *Git funciona con cualquier equipo.*

El equipo puede instalarse en herramientas individuales para el control de versiones, el seguimiento de elementos de trabajo y la integración e implementación continuas. O bien, pueden elegir una solución como GitHub o Azure DevOps que admita todas estas tareas en un solo lugar.

- *Solicitudes de incorporación de cambios.*

Use las solicitudes de extracción para analizar los cambios de código con el equipo antes de combinarlos en la rama principal. Las discusiones en las solicitudes de extracción son muy valiosas para garantizar la calidad del código y aumentar el conocimiento en todo el equipo.

- *Directivas de rama.*

Pueden configurar directivas de rama para asegurarse de que las solicitudes de extracción cumplen los requisitos antes de su finalización.

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.

Qué tipo de archivos trabaja.

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.

En los objetos de tipo Blob en Git almacena archivos, cualquier archivo con cualquier extensión es almacenado como Blob.

En los objetos de tipo Tree (árbol) Git almacena directorios, de la misma forma que cualquier otro tipo de archivos, un directorio puede estar vacío o contener archivos u otros directorios. Por lo cual un Tree puede contener un grupo de Blobs y otros Trees.

Los objetos tipo Commit en Git almacenan las versiones del proyecto.

Los objetos tipo Annotated en Git almacenan texto persistente de un Commit.

Qué tipo de lenguajes trabaja.

Los lenguajes centrales para las características de GitHub incluyen a C, C++, C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala y TypeScript.

Como se instala.

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.

Configure el nombre y la dirección de correo electrónico antes de empezar a trabajar con Git. Git asocia esta información a los cambios y permite a otros usuarios identificar qué cambios pertenecen a qué autores.

Ejecute los siguientes comandos desde el símbolo del sistema después de instalar Git para configurar esta información:

> *git config --global user.name "Jamal Hartnett"*

> *git config --global user.email "jamal@fabrikam.com"*

Creación de un repositorio de Git.

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.

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 git init comando 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 .

> *git init*

para crear el repositorio. 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 git clone comando 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/FabrikamProject*

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.

¿Que es?

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 esta 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.

¿Como funciona?

GitHub te permite subir tus repositorios de código para que sean almacenados en la nube a través del sistema de control de versiones de Git y participar también en el desarrollo de proyectos de terceros, lo que significa que cualquiera en el mundo que use GitHub puede encontrar tu código, aprender de él y, por qué no, mejorarlo. Al igual que tú puedes acceder a los repositorios de código de otras personas.

¿Qué es el control de versiones?

Se le llama control de versiones a la administración de los cambios que se realizan sobre los elementos o la configuración de algún proyecto. En otras palabras, el control de versiones sirve para conocer y autorizar los cambios que realicen los colaboradores en tu proyecto, guardando información de qué incluyen los cambios y cuándo se hicieron. Este control comienza con una versión básica del documento y luego va guardando los cambios que se realicen a lo largo del proyecto.

El control de versiones es una herramienta valiosa, pues con ella se puede tener acceso a las versiones anteriores de tu proyecto si es que en algún momento no llega a funcionar de forma correcta.

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.

Los lenguajes centrales para las características de GitHub incluyen a C, C++, C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala y TypeScript.

Como se instala.

Puedes instalar GitHub Desktop en los sistemas operativos compatibles, lo cual incluye acutalmente las macOS 10.12 o posterior y Windows 7 64-bit o posterior.

Debes tener instalado Git antes de utilizar GitHub Desktop. Si aún no tienes Git instalado, puedes descargar e instalar la última versión de Git desde https://git-scm.com/downloads.

Después de que instales Git, necesitarás configurarlo para GitHub Desktop. GitHub Desktop utiliza la dirección de correo electrónico que configuraste en tus ajustes locales de Git para conectar las confirmaciones con tu cuenta en GitHub.

Si las confirmaciones que haces en GitHub Desktop se asocian con la cuenta incorrecta de GitHub, actualiza la dirección de correo electrónico en tu configuración de Git utilizando GitHub Desktop.

Para instalar GitHub Desktop, navega a https://desktop.github.com/ y descarga la versión adecuada de GitHub Desktop para tu sistema operativo. Sigue las instrucciones para completar la instalación.

Si tienes una cuenta en GitHub o en GitHub Enterprise, puedes utilizar GitHub Desktop para intercambiar datos entre tus repositorios locales y remotos.

Si no tienes una cuenta de GitHub aún, consulta la sección "Registrarte para obtener una cuenta nueva de GitHub".

Crear un Repositorio de GitHub.

Lo primero que debes tener es una cuenta creada en GitHub. Registrarse es gratuito. Una vez que tengas la cuenta, inicia sesión con tu usuario y clave. Luego, sigue estos pasos:

En la esquina superior derecha de cualquier página, encontrarás un signo de + que sirve para realizar las acciones de la página. Das clic en el símbolo y creas un nuevo repositorio (new repository).

Una vez realizado eso, debes llenar los datos que se solicitan a continuación. Darle un nombre, que de preferencia debe ser claro, definir si será público o privado y colocar una pequeña descripción sobre tu repositorio. Este campo es opcional, pero te recomiendo que lo

llenes para organizarte mejor y que los demás usuarios tengan una idea sobre lo que trata el repositorio que estás creando.

Activa el checkbox que dice iniciar tu repositorio con un README, este será tu primer archivo, la presentación de tu proyecto.

Presiona el botón de “crear repositorio” y listo.

Bibliografía.

Azure DevOps. (2022, 8 abril). *¿Qué es Git? - Azure DevOps*. Microsoft Docs. https://docs.microsoft.com/es-es/devops/develop/git/what-is-git

Microsoft. (2020). *Documentación del Escritorio de GitHub*. GitHub Docs. https://docs.github.com/es/desktop