TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco

INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

MATERIA:

HERRAMIENTAS EMERGENTES DE DESARROLLO DE SOFTWARE

NOMBRE DEL TRABAJO:

“INVESTIGACIÓN CONTROL DE VERSIONES CON GIT Y GITHUB”

PRESENTA:

EDUARDO VASQUEZ CONDE

19371076

DOCENTE:

JUAN RAMOS RAMOS

Apizaco, Tlaxcala, a Fecha 26 de Febrero, 2024

¿Qué es Git?

Git es un Sistema de Control de Versiones Distribuido (DVCS) utilizado para guardar diferentes versiones de un archivo (o conjunto de archivos) para que cualquier versión sea recuperable cuando lo desee.

Git también facilita el registro y comparación de diferentes versiones de un archivo. Esto significa que los detalles sobre qué cambió, quién cambió qué, o quién ha iniciado una propuesta, se pueden revisar en cualquier momento.

¿Qué significa "distribuido"?

El término "distribuido" significa que cuando le instruyes a Git que comparta el directorio de un proyecto, Git no sólo comparte la última versión del archivo. En cambio, distribuye cada versión que ha registrado para ese proyecto.

Este sistema "distribuido" tiene un marcado contraste con otros sistemas de control de versiones. Ellos sólo comparten cualquier versión individual que un usuario haya explícitamente extraído desde la base de datos central/local.

¿Qué es un Sistema de Control de Versiones?

Un Sistema de Control de Versiones (VCS) se refiere al método utilizado para guardar las versiones de un archivo para referencia futura.

De manera intuitiva muchas personas ya utilizan control de versiones en sus proyectos al renombrar las distintas versiones de un mismo archivo de varias formas como blogScript.js, blogScript_v2.js, blogScript_v3.js, blogScript_final.js, blogScript_definite_final.js, etcétera. Pero esta forma de abordarlo es propenso a errores y inefectivo para proyectos grupales.

Estados de los archivos en Git

En Git hay tres etapas primarias (condiciones) en las cuales un archivo puede estar: estado modificado, estado preparado, o estado confirmado.

Estado modificado

Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar).

En otras palabras, archivos en el estado modificado son archivos que has modificado pero no le has instruido explícitamente a Git que controle.

Estado preparado

Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio .git durante la próxima instantánea de confirmación.

Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.

Estado confirmado

Archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente.

Por lo tanto un archivo confirmado es un archivo en el cual has registrado su versión preparada en el directorio (carpeta) Git.

Ubicación de archivos

Existen tres lugares principales donde pueden residir las versiones de un archivo cuando se hace control de versiones con Git: el directorio de trabajo, el sector de preparación, o el directorio Git.

Directorio de trabajo

El directorio de trabajo es una carpeta local para los archivos de un proyecto. Esto significa que cualquier carpeta creada en cualquier lugar en un sistema es un directorio de trabajo.

Zona de preparación

La zona de preparación – técnicamente llamado “index” en lenguaje Git – es un archivo normalmente ubicado en el directory .git, que guarda información sobre archivos próximos a ser acometidos en el directorio .git.

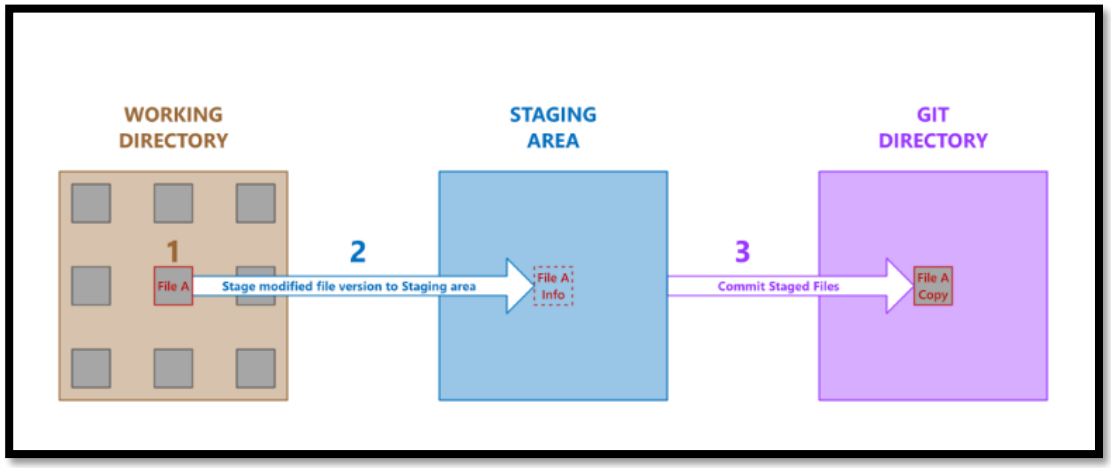
Directorio Git

El directorio.git es la carpeta (también llamada "repositorio") que Git crea dentro del directorio de trabajo que le has instruido para realizar un seguimiento.

La carpeta .git también es donde Git guarda las bases de datos de objetos y metadatos de los archivos que le hayas instruido realizar un monitoreo.

El flujo de trabajo básico de Git

Trabajar con el Sistema de Control de Versiones Git se ve algo así:



1. Modificar archivos en el directorio de trabajo.
   Observe que cualquier archivo que modifiques se convierte en un archivo en el estado modificado.
2. 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.

3. 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.

GitHub Desmitificado

GitHub es una plataforma basada en la web donde los usuarios pueden alojar repositorios Git. Facilita compartir y colaborar fácilmente en proyectos con cualquier persona en cualquier momento.

GitHub también fomenta una participación más amplia en proyectos Código Abierto al proporcionar una manera segura de editar archivos en repositorios de otros usuarios.

Para alojar (o compartir) un repositorio Git en GigHub, siga los siguientes pasos:

Paso 1: Registrar una cuenta de GitHub

El primer paso para comenzar con el alojamiento en GitHub es crear una cuenta personal. Visite la página de registro oficial para registrarse.

Paso 2: Crear un repositorio remoto en GitHub

Después de registrarse, crear un home (un repositorio) en GitHub para el repositorio Git que quieres compartir.

Paso 3: Conectar el directorio Git del proyecto con el repositorio remoto

Una vez que hayas creado un repositorio remoto para tu proyecto, tienes que vincular el directorio .git del proyecto – ubicado localmente en tu sistema – con el repositorio remoto en GitHub.

Para conectar con el repositorio remoto, debes ubicarte en el directorio raíz del proyecto que quieras compartir, utilizando tu terminal local, y ejecuta este comando:

```
git remote add origin https://github.com/tuUsuario/tuRepositorio.git
```

Paso 4: Confirmar la conexión

Una vez que hayas conectado tu directorio Git con el repositorio remoto, verifica si la conexión fue exitosa ejecutando el comando git remote -v.

Paso 5: Subir un repo Git local a un repo remoto

Después de conectar exitosamente tu directorio local con el repositorio remoto puedes comenzar a subir (cargar) tu proyecto local upstream.

Cuando estés listo para compartir tu proyecto en otra parte, en cualquier repo remoto, simplemente le dices a Git que suba todos tus confirmaciones, ramas y archivos en tu directorio .git local al repositorio remoto.

La sintaxis del código utilizado para subir (push) un directorio local Git a un repositorio remoto es git push -u nombreRemoto nombreRama.

Esto es, para subir tu directorio local .git y suponiendo que el nombre corto de la URL remota es “origin”, ejecuta:

```
git push -u origin master
```

Paso 6: Confirmar la subida

Por último, vuelve a tu página de repositorio GitHub para confirmar que Git haya subido exitosamente tu directorio Git local al repositorio remoto.

En resumen

GitHub es una plataforma en línea de alojamiento (o para compartir) repositorios de Git. Nos ayuda crear una avenida para colaborar fácilmente con cualquier persona, en cualquier lugar, en cualquier momento.

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 embarg 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.

Bibliografía

- Castellanos, E. (2021, febrero 14). Git vs GitHub – ¿Qué es el Control de Versiones y Cómo Funciona? freecodecamp.org. https://www.freecodecamp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/