JUE
29

MEETING 3 · MEETING 43

Git - Workshop

Grabación

Presentación

El equipo docente subió contenido extra de la meeting

Volver al futuro

> ***"My name is Linus Torvalds, and I'm your god." –Linus Torvalds. Linux Expo 1998.***

En esta bitácora aprenderás sobre Git, el control de versiones más utilizado por los programadores. Esta poderosa herramienta te permitirá trabajar en grandes equipos sobre un mismo código. Entre otros beneficios, tendrás un historial de los cambios y podrás crear ramificaciones con distintas versiones.

Control de versiones

Los sistemas de control de versiones pertenecen a una categoría de herramientas de software que ayudan a un equipo a **gestionar los cambios** en el código fuente a través del tiempo.

El software de control de versiones realiza un seguimiento de cada modificación del código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden 'retroceder el tiempo' y comparar versiones anteriores del código para ayudar a solucionar errores y minimizar la interrupción del proceso de desarrollo para los y las miembros del equipo. Además de restaurar un punto en el pasado, pueden ramificar nuevas líneas de tiempo a partir de estos momentos.

Los Data Scientist, Analistas e Ingenieros de datos, así como también todos/as los/as profesionales del ecosistema de datos que trabajan en equipos, escriben continuamente nuevo código y cambian el existente. El código para un proyecto generalmente se organiza en una estructura de carpetas o **"árbol de archivos"**. Una desarrolladora del equipo puede estar trabajando en una nueva característica o pregunta, mientras que otro desarrollador corrige un error no relacionado a lo anterior. Lo ideal es que cada una de estas personas puedan hacer sus cambios en varias partes del árbol de archivos.

El control de versiones ayuda a los equipos a resolver este tipo de problemas, rastreando cada cambio individual de cada contribuyente y ayudando a evitar que el trabajo concurrente entre en conflicto. Los cambios realizados en una parte del código pueden ser incompatibles con los realizados por otra persona que trabaje al mismo tiempo. Este problema debe ser descubierto y resuelto de manera ordenada sin bloquear el trabajo del resto del equipo.

Además, en todo desarrollo de software cualquier cambio puede introducir nuevos errores por sí mismo y no se puede confiar en el nuevo software hasta que se pruebe. Por lo tanto, las pruebas y el desarrollo continúan juntos hasta que una nueva versión esté lista.

Git

El sistema de control de versiones más utilizado en el mundo hoy en día es Git. Git es un proyecto open source desarrollado originalmente en 2005 por Linus Torvalds, el famoso creador del núcleo del sistema operativo Linux. Una asombrosa cantidad de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales, privados y open source.

Git es un sistema distribuido, lo cual significa que cada persona que forme parte del proceso de desarrollo contará con todo el código fuente junto al historial completo de todos los cambios.

Aprender Git requiere práctica de nuestro músculo de la memoria, tanto para entender el esquema histórico de los cambios, como para tipear los comandos en una consola. Si bien existen herramientas visuales que simplifican mucho el trabajo administrativo de Git, en esta bitácora y en nuestro próximo encuentro trabajaremos directamente sobre la consola, para fortalecer el entendimiento de estos conceptos.

Ramificaciones

Git propone el concepto **ramificaciones,** para aislar el desarrollo de cada participante en un equipo. Si bien se está trabajando sobre el mismo código, las personas no estarán trabajando sobre los mismos archivos. Por lo tanto, los branches nos permiten crear nodos paralelos para desarrollar en aislado, y una vez concluida nuestra tareas, volver al flujo inicial, uniendo nuestros cambios.

El gráfico anterior muestra un esquema de ramificaciones en Git, en **azul** está la rama principal, llamada **master,** la cual contiene todo el código que ven las personas usuarias de una aplicación. En **naranja** está la rama **development,** que es una copia directa de **master** y su finalidad es que sea el punto de partida para las y los developers. Este paralelismo entre master y develop existe ya que todo el código nuevo que se vaya haciendo, se une a develop y se prueba; si todo sale bien, se une a master.

De **development** se desprenden las features branch en **violeta**, que son ramas cortas donde se desarrolla un feature en específico.

Comandos

Vamos a describir los comandos básicos de git:

- `git init` : inicializa el registro de actividades dentro de una estructura de carpetas.
- `git add` : permite agregar archivos o directorios al registro histórico
- `git commit` : crea un punto en la historia con las modificaciones hechas hasta ese momento.
- `git checkout` : accede a un punto de la historia (un commit), o visita una ramificación, o branch, del sistema.
- `git merge` : permite unir ramas, o líneas de trabajo, en una sola.

La consola

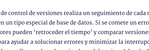
Como mencionamos anteriormente estaremos usando la **consola** -temida por muchos, adorada por otros- para explorar Git. Es clave a la hora de programar, ya que nos permite realizar tareas de forma más directa, e incluso cuenta con algunas funciones exclusivas. En tiempos en los que estamos acostumbrados/as a tener botones, animaciones y transiciones en nuestras interfaces, puede que el entorno resulte poco amigable. Sin embargo, con un poco de práctica y costumbre, la consola se puede transformar en una aliada poderosa, rápida y eficaz a la hora de realizar ciertas tareas.

A grandes rasgos, ofrece una forma de navegar todas las carpetas y archivos que están dentro de tu computadora pero a través de una interfaz de texto. La consola fue la primera "interfaz gráfica" que tuvieron las computadoras, y era la encargada de realizar todas las operaciones (abrir, cerrar, modificar, consultar, listar, entre otras). Ofrece una abstracción del lenguaje en la que se pueden indicar comandos a ejecutar para distintas operaciones

Si eres usuario/a de Windows, puedes presionar la tecla de inicio y tipear "CMD", que significa "COMMAND". Si haces click sobre ella, se abrirá la consola más primitiva que el sistema operativo puede ofrecer.

Si eres usuario/a Mac, puedes invocar el **"Spotlight"** tipeando cmd + barra de espacio, tipear "bash" y, al abrir este programa, verás la consola de este sistema operativo:

La consola comunica por una cadena de texto la ruta de la carpeta sobre la cual estás parado/a, desde el símbolo de tu disco duro (`c:`, `d:`, etc) hasta el recurso o carpetas en el cual estás:



En el ejemplo anterior se muestra a un/a usuario/a dentro de una carpeta llamada "Clase-2", que a su vez está dentro de la carpeta "PAR", contenida en la carpeta "Academia" de la unidad del disco "C:".

Esperamos que hayas seguido el hilo hasta acá. Si no, no te preocupes. Vas a comprender más de que se trata cuando lo implementes.

Git en la vida de un Data Scientist

Quizás te estés preguntando si es necesario saber Git siendo Data Scientist. Después de todo, parece más una herramienta para desarrolladores/as de software (de hecho, el ejemplo que dimos es más apropiado para un desarrollo de software o web). La respuesta corta es que no. Puedes ser un gran data scientist sin manejar Git, pero te estarás perdiendo un mundo grande de oportunidades. Git es mucho más que un software para trabajar en equipo. Incluso para proyectos personales es muy recomendable

Algunos puntos por los cuales creemos que deberías saber Git:

- La programación se volvió una herramienta central en Data Science.
- Muchas empresas lo tienen como estándar en el desarrollo de modelos y sistemas.
- Te permite compartir tus desarrollos con colegas o posibles empleadores - hoy paginas como Github se volvieron casi un complemento del CV, te animamos a que tengas tu propio repositorio[1].
- Git es la herramienta base para todo proyecto open source. Como ya lo experimentaste, la comunidad open source es y será un aliado natural en tu aprendizaje.
- Podrás tener un historial de los cambios en tu proyecto y generar distintas versiones del mismo. Como ya sabes, es clave para nuevos proyectos poder reciclar código ya escrito.

Un último punteo sobre Git:

- Más allá de la terminal, existen entornos de desarrollo que te facilitan la interacción con Git.
- No te preocupes si no recuerdas todos los comandos de Git (nadie lo hace).
- Existen plataformas muy conocidas donde puedes guardar tu propio repositorio e interactuar con la comunidad. Github y Bitbucket son las más conocidas.

¡Prepárate para el próximo encuentro!

Comunidad
Sumérgete en la red donde los profesionales de DS interactúan.

Herramientas
Programas necesarios para facilitar tu experiencia.

Challenge
Te proponemos el siguiente desafío, ¿te animas?

Contenido extra de la meeting

Grabación
Código de acceso para acceder: Academia123*

Contenido extra

Grabación