Carnet: 16002653
Nombre: Alejandra Sierra
Ciencia de Datos en Python

INVESTIGACIÓN No. 1

Qué es git

El diseño de Git se basó en BitKeeper y en Monotone. Originalmente fue diseñado como un motor de sistema de control de versiones de bajo nivel sobre el cual otros podrían codificar interfaces frontales, tales como Cogito o StGIT. Desde ese entonces hasta ahora el núcleo del proyecto Git se ha vuelto un sistema de control de versiones completo, utilizable en forma directa.

Linus Torvalds buscaba un sistema distribuido que pudiera usar en forma semejante a BitKeeper, pero ninguno de los sistemas bajo software libre disponibles cumplía con sus requerimientos, especialmente en cuanto a desempeño. El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por mucha gente, que incide en numerosos detalles de rendimiento, y de la necesidad de rapidez en una primera implementación.

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.

¿Pero si Git es un Sistema de Control de Versiones Distribuido, qué significan exactamente esos términos?

¿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.

Bueno, entonces "distribuido" significa distribuir todas – no solo algunas seleccionadas – versiones de los archivos del proyecto que Git haya registrado. Pero qué es exactamente un sistema de control de versiones?

Carnet: 16002653
Nombre: Alejandra Sierra
Ciencia de Datos en Python

¿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.

Además, con esta forma de abordarlo, rastrear qué cambió, quién lo cambió y porqué se cambió, es un esfuerzo tedioso. Esto resalta la importancia de un sistema de control de versiones confiable y colaborativo como Git.

Sin embargo, para obtener lo mejor de Git, es importante entender cómo Git administra tus archivos.

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.

Nota: El estado de un archivo determina la ubicación donde Git lo colocará.

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.

Nota:
Los archivos en el estado modificado residen dentro del directorio de trabajo.

El directorio de trabajo es distinto al directorio .git. Es decir, tu creas un directorio de trabajo mientras que Git crea un directorio .git.

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.

Nota:
Los archivos en la etapa de preparación residen en la zona de preparación.
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.

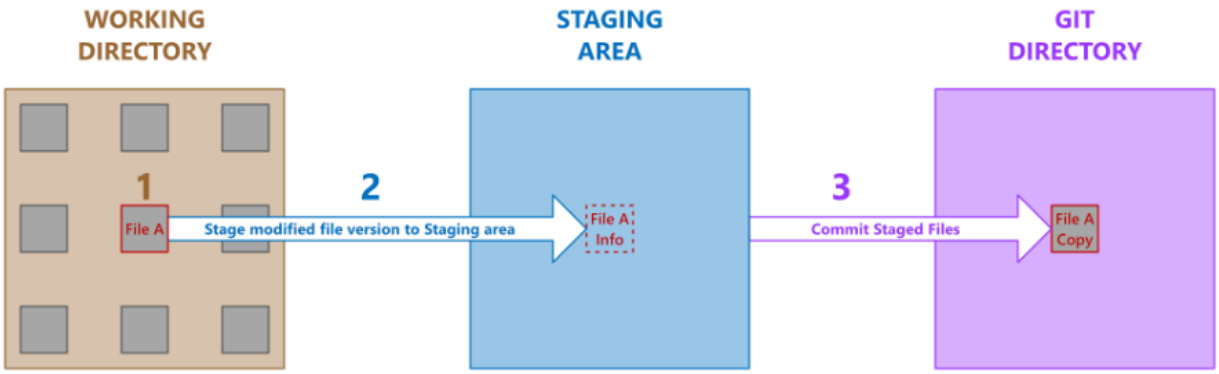
Nota:
El directorio.git es la vida de Git – es el item copiado cuando se clona un repositorio desde otra computadora (o desde una plataforma en línea como GitHub).

Los archivos en el estado acometido residen en el directorio Git.

Carnet: 16002653
Nombre: Alejandra Sierra
Ciencia de Datos en Python

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*.

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?

Carnet: 16002653
Nombre: Alejandra Sierra
Ciencia de Datos en Python

Qué es github

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar.

Así pues, Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario.

Así pues, Github es un portal para gestionar las aplicaciones que utilizan el sistema Git. Además de permitirte mirar el código y descargarte las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.

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
```

Nota:

- Reemplaza tu Usuario en el código de arriba con tu nombre de usuario de GitHub. También reemplaza tu Repositorio con el nombre del repositorio remoto al que te quieres conectar.
- El comando de arriba implica que *git* debe *agregar* al proyecto la *URL* especificada como una referencia remota con la cual el directorio local .git puede interactuar.
- La opción origin en el comando de arriba es el nombre predeterminado (un nombre corto) que Git le otorga al servidor que aloja tu repositorio remoto. Es decir, Git utiliza el nombre corto origin en vez de la URL del servidor.
- No es obligatorio quedarse con el nombre predeterminado del servidor. Si prefieres otro nombre que origin, simplemente sustituya el nombre origin en el comando git remote add de arriba por cualquier nombre que prefieras.
- Siempre recuerda que un nombre corto del servidor (por ejemplo, origin) no es nada especial! Sólo existe – localmente – para ayudarte a referenciar fácilmente la URL del servidor. Por lo tanto, siéntete libre de cambiarlo a un nombre corto que puedas referenciar fácilmente.
- Para renombrar cualquier URL remota que exista, utilice el comando git remote rename de esta manera:

```
git remote rename nombreURLactual nuevoNombreURL
```

- Cada vez que clones (descargues) cualquier repo remota, Git automáticamente le pone nombre origin a la URL de la repo. Sin embargo, le puedes especificar un nombre distinto con el comando git clone -o tuNombrePreferido.
- Para ver la URL exacta guardada para los nombres como origin, ejecuta el comando git remote -v.

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.

Después verifica la impresión para confirmar que la *URL mostrada* sea la misma que la *URL remota* que intentas conectarte.

Nota:
- Ver el artículo "Conectar con SSH" si quieres conectar utilizando la URL SSH en lugar de la URL HTTPS.
- Sin embargo, si no estás seguro de la URL remota que vas a utilizar, echa un vistazo al artículo "¿Qué URL remota debería utilizar?".
- ¿Quieres cambiar tu URL remota? Cambiar la URL de un remoto es una excelente guía.

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
```

Nota:
- El comando de arriba implica que *git* debe *subir* tu rama *master* local a la rama *master* remota ubicada en la URL con nombre *origin*.
- Técnicamente puedes sustituir la opción origin con la URL del repositorio remoto. Recuerda, la opción origin es solo un nombre de la URL que hayas registrado en tu directorio .git local.
- El indicador -u (indicador de referencia upstream/seguimiento) automáticamente vincula la rama local del directorio .git con la rama remota. Esto te permite usar git pull sin ningún argumento.

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.

Nota:
- Tal vez tengas que refrescar la página del repositorio remoto para reflejar los cambios.
- GitHub también tiene un servicio gratuito opcional para convertir tu repositorio remoto en una página web funcional. Abajo vamos a ver "cómo" hacerlo.-

Publica tu página web con GitHub pages
Después de subir tu proyecto al repositorio remoto, fácilmente puedes publicarlo en la web de esta forma:

1. Asegurate que el nombre del archivo principal HTML de tu proyecto sea index.html.
2. En el sitio web de la plataforma GitHub ingresa al repositorio del proyecto que quieres publicar y haz click en la **pestaña settings**.
3. Desplaza hacia la sección **GitHub Pages** y cambia la rama **Source** de **none** a **master**.
4. Después aparecerá una notificación que dice "Your site is published at *https://your-username.github.io/your-github-repo-name/*".
5. Ahora puedes ver – y publicitar – tu proyecto en la URL especificada.

Esta sección solo ha comenzado a tratar el tema de publicar tu proyecto con GitHub. Para aprender más sobre GitHub pages, echa un vistazo a esta documentación "Trabajar con páginas de GitHub Pages".

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.

Todavía en duda?
Todavía estás perplejo por la delgada línea entre Git y GitHub? No te preocupes – Te tengo cubierto. Abajo hay cinco diferencias claves 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 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.

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.

Considerando todo

Git y GitHub son dos entidades diferentes que te ayudan a administrar y alojar archivos. En otras palabras Git sirve para controlar las versiones de los archivos mientras que GitHub es una plataforma para alojar tus repositorios Git.

Markdown y sus comandos

Markdown es una herramienta de conversión de texto a HTML para escritores web. Markdown le permite escribir utilizando un formato de texto sin formato fácil de leer y escribir, y luego convertirlo a XHTML estructuralmente válido (o HTML).

Por lo tanto, "Markdown" es dos cosas: (1) una sintaxis de formato de texto sin formato; y (2) una herramienta de software, escrita en Perl, que convierte el formato de texto sin formato a HTML. Consulte la página Sintaxis para obtener detalles relacionados con la sintaxis de formato de Markdown. Puedes probarlo, ahora mismo, usando el Dingus en línea.

Carnet: 16002653
Nombre: Alejandra Sierra
Ciencia de Datos en Python

El objetivo principal del diseño de la sintaxis de formato de Markdown es hacer que sea lo más legible posible. La idea es que un documento con formato de Markdown se pueda publicar tal cual, como texto sin formato, sin que parezca que ha sido marcado con etiquetas o instrucciones de formato. Si bien la sintaxis de Markdown se ha visto influenciada por varios filtros de texto a HTML existentes, la mayor fuente de inspiración para la sintaxis de Markdown es el formato de correo electrónico de texto sin formato.

La mejor manera de tener una idea de la sintaxis de formato de Markdown es simplemente mirar un documento con formato de Markdown. Por ejemplo, puede ver la fuente de Markdown para el texto del artículo en esta página aquí: http://daringfireball.net/projects/markdown/index.text

(Puede usar este truco del sufijo '.text' para ver la fuente de Markdown para el contenido de cada una de las páginas de esta sección, por ejemplo, las páginas de Sintaxis y Licencia).

Markdown es un software gratuito, disponible bajo una licencia de código abierto estilo BSD. Consulte la página de licencias para obtener más información.

Hay dos maneras de dar formato al código en Markdown. Puedes usar código en línea, colocando comillas invertidas (`) alrededor de partes de una línea, o puedes usar un bloque de código al que algunos renderizadores le aplicarán un resaltado de sintaxis.

Código en línea

Puedes usar el formateo de código en línea para enfatizar un comando o una pieza pequeña de sintaxis dentro de una oración que estás escribiendo.

Por ejemplo, si quisieras mencionar el método Array.protoype.map() de JavaScript. Al utilizar el formato de código en línea, queda claro que se trata de una pieza de código. También podrías usarlo para ilustrar un comando de la terminal como yarn install.

Para utilizar el formato de código en línea, simplemente pon entre comillas invertidas el código al que le quieres dar formato. En un teclado QWERTY estándar de EE.UU., esto se encuentra a la izquierda del ‘1’, y sobre la tecla Tab. Más información sobre la ubicación de las comillas invertidas en teclados internacionales es provista debajo.

Por ejemplo, escribir `Array.prototype.map()` en Markdown se va a renderizar como Array.prototype.map().

Bloques de código
Para escribir fragmentos de código más largos y detallados, habitualmente es mejor colocarlos dentro de un bloque de código. Los bloques de código le permiten utilizar

múltiples líneas, y Markdown las renderizará dentro de su propio cuadro y con un tipo de fuente de código.

Para lograr esto, comience su bloque con una línea de tres comillas invertidas. Esto le señala a Markdown que está creando un bloque de código. Necesitará finalizarlo con otra línea de tres comillas invertidas. Por ejemplo:

```
var sumar2 = function(numero) {
  return numero + 2;
}
```

se renderizará en Markdown como:

```
var sumar2 = function(numero) {
  return numero + 2;
}
```

Resaltado de sintaxis
Aunque no es soportado nativamente por Markdown, varios motores de Markdown, incluyendo el que es utilizado por GitHub, soportarán el resaltado de sintaxis. Esto significa que al indicarle a Markdown el lenguaje que está utilizando dentro del bloque de código, este le agregará colores como lo haría un entorno de desarrollo integrado (IDE).

Puedes hacer esto agregando el nombre del lenguaje en la misma línea de sus tres comillas invertidas de apertura. En el ejemplo de arriba, si en lugar de que la primera línea sea ``` se escribiera ```js, entonces se aplicaría el resaltado de JavaScript al bloque.

```
var sumar2 = function(numero) {
    return numero + 2;
}
```

Sin embargo, el resaltado de sintaxis se puede aplicar a más que solo JavaScript. Puedes utilizar ```html:

```
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <h1>Hola Mundo</h1>
  </div>
</div>
```

```ruby:

```
"Hola Mundo".split('').each do |letter|
  puts letter
end
```

o ```python:

```
a, b = 0, 1
while b < 10:
    print(b)
    a, b = a, a + b
```

Solo recuerda, no todos los motores de Markdown aplicarán el resaltado de sintaxis.

Comillas invertidas en teclados internacionales
La ubicación de la tecla de comillas invertida puede variar en distintos teclados, y si no estás utilizando un teclado QWERTY de distribución estadounidense, puede ser difícil de encontrar. A continuación puedes ver -señalada en rojo- dónde encontrar la tecla de comillas invertida en distintos diseños de teclado.
```