INVESTIGACION DE GIT

DAVID ANDRES RODRIGUEZ CHAIN

JAIR LEAÑO

UNIVERSIDAD DE LA COSTA CUC

INGENIERIA DE SISTEMAS

BARRANQUILLA

2017

GIT

Es un sistema de control de versiones distribuido, es decir, es un sistema que sirve para almacenar versiones de nuestro código fuente y múltiples personas pueden trabajar en un mismo proyecto y cada una tendrá una copia el código.

HISTORIA DE GIT

Git comenzó con un poco de gran polémica y destrucción creativa.

El kernel de Linux es un proyecto de software de código abierto con un alcance bastante amplio. Durante la mayor parte del mantenimiento del kernel de Linux (1991-2002), los cambios en el software se realizaban a través de parches y archivos. En el 2002, el proyecto del kernel de Linux empezó a usar un DVCS propietario llamado BitKeeper.

En el 2005, la relación entre la comunidad que desarrollaba el kernel de Linux y la compañía que desarrollaba BitKeeper se vino abajo, y la herramienta dejó de ser ofrecida de manera gratuita. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron mientras usaban BitKeeper. Algunos de los objetivos del nuevo sistema fueron velocidad, diseño sencillo, gran soporte para desarrollo no lineal, completamente distribuido, capaz de manejar grandes proyectos.

Desde su nacimiento en el 2005, Git ha evolucionado y madurado para ser fácil de usar y conservar sus características iniciales. Es tremendamente rápido, muy eficiente con grandes proyectos, y tiene un increíble sistema de ramificación (branching) para desarrollo no lineal.

SISTEMAS DE CONTROL DE VERSIONES

SUBVERSIÓN

Características

* Subversión es un sistema de control de versiones centralizado
* Se trabaja sobre una copia local
* Las operaciones son en el servidor

Ventajas

* Se puede hacer un “checkout” parcial

Desventajas

* Cada commit se hace en el servidor
* Cuando se hace commit hay que tener una versión consistente antes de hacer el commit
* No se puede dejar un cambio por la mitad, ejemplo como se haría en git con branch
* Un branch se usa solamente para congelar una versión y no es posible hacerlo de forma local

BAZAAR

Características

* Sistemas de control de versiones distribuido

Ventajas

* Es fácil de modificar muy intuitivo
* Integrable se puede añadir proyectos existentes
* Se puede utilizar para adaptarse a cualquier tipo de proyecto

Desventajas

* Es lento
* Se trabaja con acceso al servidor remoto

COMANDOS DE GIT

1. Configurar nombre del usuario [$ git config --global user.name “nombre de usuario”]
2. Configurar email [$ git config --global user.email “my\_email@servidor.com"]
3. Ver todas las configuraciones echas [$ git config --global –list]
4. Crear un nuevo repositorio [$ git init]
5. Agregar todos los archivos [$ git add -A]
6. Agregar un archivo [$ git add “archivo”]
7. Quitar un archivo [$ git rm “archivo”]
8. Ver estado de archivos [$ git status]
9. Crear un commit con comentario [$ git commit -m "Comentario"]
10. Viajar entre los commits [$ git checkout "codigo del commit"]
11. Viajar al último commit de la rama [$ git checkout master]
12. Borrar el commit seleccionado sin tocar el codigo (archivo) [$ git reset --soft "codigo del commit"]
13. Borrar el commit seleccionado incluyendo el codigo (archivo o modificaciones que se hicieron en el) [$ git reset --hard "codigo del commit"]
14. Ver todos los commits hechos [$ git log]
15. Crear un archivo de texto (.txt) con todos los commits [$ git log > commits.txt]
16. Crear una rama [$ git branch "nombre de la rama"]
17. Moverse entre ramas [$ git checkout “nombre de la rama”]
18. Moverse y crear rama al mismo tiempo [$ git checkout -b “nombre de la rama”]
19. Fusionar ramas (traer los cambios que tiene la otra rama [$ git merge “rama que tiene los cambios”]
20. Ver ramas [$ git branch]

GITHUB

GitHub es una plataforma que nos permite crear repositorios de nuestros archivos, proyectos, etc, utilizando el sistema de control de versiones Git. Otras personas pueden ver y dar su opinión, modificar el repositorio dependiendo de los permisos que el administrador les otorgue

FUNCIONALIDADES DE GITHUB

PROJECTS

Puedes crear tableros y añadirles tanto Issues como Pull Requests, organizarlos en el tiempo o por prioridades, también puede añadir notas para meter todo aquello que no se ajusta a ninguno de los dos tipos anteriores.

INTEGRACIONES

Han lanzado la API GraphQL, que permite a los desarrolladores acceder a todos los datos que necesiten en una única llamada.

REVIEWS

Tiene la opción de escribir un comentario sin necesidad de llegar a publicar nada, se puede revisar el código que queremos publicar y comentar los problemas que encuentren.

CLONAR

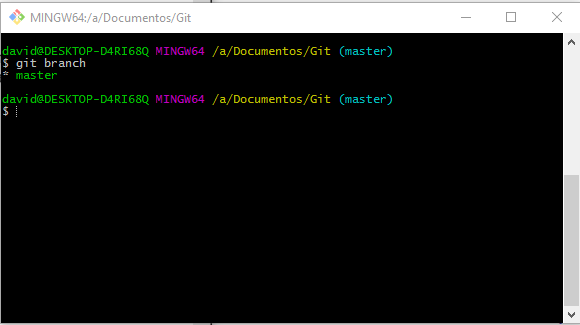
Esta funcionalidad de GitHub sirve para hacer un fork, es decir, puedes clonar un repositorio ajeno para que se copie a tu cuenta.

BRANCH O RAMA

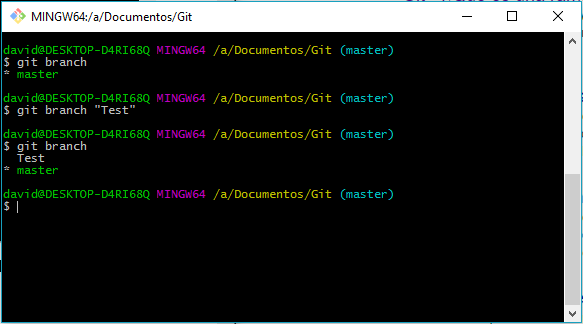
Básicamente las ramas son separaciones de la rama principal o la rama master, son utilizadas para desarrollar funcionalidades aisladas unas de otras. La rama master es la rama por defecto cuando crear un repositorio. Las otras ramas que se creen pueden ser utilizadas para el desarrollo y después ser fusionadas con la rama principal.

Las ramas no sirven para arreglar errores, experimentar, hacer grandes cambios sin tener la necesidad de afectar al proyecto que se esté haciendo.

Si se quiere saber qué ramas tienen creadas se utiliza el comando { git branch }



para crear una nueva rama se utiliza el comando { git branch “rama” }



PULL REQUEST

Un pull request básicamente es una petición para incluir nuestros cambios de codigo a un proyecto. Permiten crear propuestas o cambios se puedan incluirse a un proyecto y también llevar de forma más ordenada las tareas en la etapa del desarrollo.

HOOKS

Son un conjunto de acciones que se pueden ejecutar antes o después de un comando particular de git.