INVERTIGACION

CARLOS ARCENIO ROCHA BARRIOS

DAVID JOSE PADILLA BARRAZA



UNIVERSIDAD DE LA COSTA CUC

**INGENIERIA DE SISTEMAS**

**BARRANQUILLA**

**2017**

INVESTIGACION

CARLOS ARCENIO ROCHA BARRIOS

DAVID JOSE PADILLA BARRAZA

**TRABAJO INVESTIGATIVO**

**ING. ROBERTO MORALES ORTEGA**



UNIVERSIDAD DE LA COSTA CUC

**FACULTAD DE INGENIERIA**

**INGENIERIA DE SISTEMAS**

**BARRANQUILLA**

**CONTENIDO**

**Pag**

[**1.** INTRODUCCION 4](#_Toc304442591)

[**2.** ¿Qué ES GIT? 5](#_Toc304442592)

[**3.** HISTORIA DE GIT 5](#_Toc304442593)

[**4.** OTROS TIPOS DE SISTEMAS DE CONTROL DE VERSIONES 6](#_Toc304442596)

[**4.1** ACCUREV 6](#_Toc304442597)

[**4.1.1** VENTAJAS 6](#_Toc304442598)

[**4.1.2** DESVENTAJAS 6](#_Toc304442598)

[**4.1** ACCUREV 6](#_Toc304442597)

[**4.1.1** VENTAJAS 6](#_Toc304442598)

[**4.1.2** DESVENTAJAS 6](#_Toc304442598)

[5. 20 COMANDOS DE GIT 15](#_Toc304442600)

[6. QUE ES GITHUB 16](#_Toc304442604)

[7. LAS 5 MEJORES FUNCIONALIDADES DE GITHUB 17](#_Toc304442608)

[7.1 Titulo 1 17](#_Toc304442609)

[7.1.1 Subtitulo 1 17](#_Toc304442610)

[7.1.1.1 Subtitulo 2 17](#_Toc304442611)

[8. QUE UN BRANCH O RAMA 18](#_Toc304442612)

[9. QUE ES UN PULL REQUEST. 19](#_Toc304442613)

[10. QUE ES UN HOOK 20](#_Toc304442614)

[11. LOS 5 MEJORES PROYECTOS OPENSOURCE ALOJADOS EN GITHUB 21](#_Toc304442615)

[11.1 Anexo 1 21](#_Toc304442616)

[11.2 Anexo 2 21](#_Toc304442617)

# INTRODUCCION

En este trabajo de investigación se mostraran la definición de un sistema de control de versiones muy conocido y usado por muchos desarrolladores en el mundo llamado Git. Hablaremos un poco de la historia de Git y resaltaremos las características, ventajas y desventajas de otros sistemas de control de versiones. Dentro de este trabajo también se podrán encontrar 20 comandos Git y su función.

Encontraremos la respuesta a ¿Qué es Github? Y sus cinco funcionalidades más relevantes. También tocaremos el tema de ¿Qué es una rama o branch, para que sirve y como se genera? Hallaremos respuestas a ¿Qué es un pull request?, ¿Qué es un hook?, y finalmente seleccionaremos los 5 mejores proyectos opensource alojados en Github e indicaremos sus funcionalidades, características, imágenes, ventajas y desventajas.

# ¿Qué ES GIT?

Git un software de control de versiones diseñado por Luis Torvalds y fue pensado en la confiabilidad y eficiencia del mantenimiento de versiones de aplicaciones cuando tienen un gran número de archivos de código fuente.

Al hablar de control de versiones, se habla de la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

# historia de git

Como muchas de las grandes cosas en esta vida, Git comenzó con un poco de destrucción creativa y encendida polémica. El núcleo de Linux es un proyecto de software de código abierto con un alcance bastante grande. Durante la mayor parte del mantenimiento del núcleo de Linux (1991-2002), los cambios en el software se pasaron en forma de parches y archivos. En 2002, el proyecto del núcleo de Linux empezó a usar un DVCS propietario llamado BitKeeper.

En 2005, la relación entre la comunidad que desarrollaba el núcleo de Linux y la compañía que desarrollaba BitKeeper se vino abajo, y la herramienta dejó de ser ofrecida gratuitamente. 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 durante el uso de BitKeeper. Algunos de los objetivos del nuevo sistema fueron los siguientes:

* Velocidad
* Diseño sencillo
* Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas)
* Completamente distribuido
* Capaz de manejar grandes proyectos (como el núcleo de Linux) de manera eficiente (velocidad y tamaño de los datos)

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

# OTROS TIPOS DE SISTEMAS DE CONTROL DE VERSIONES

## Accurev

Es una herramienta para la gestión de la configuración de código fuente que integra un gestor de incidencias basado en flujos que maneja de forma eficiente desarrollo paralelo a escala global. También contempla un servidor para replicación. Algunas ventajas y desventajas:

### **VENTAJAS**

* AccuRev captura y controla las relaciones entre bases de código en entornos paralelos utilizando una arquitectura basada en streams. Esto permite a los equipos almacenar el trabajo de forma seguro y probarlo antes de que sea compartido con otros.
* El proceso de [guardado](https://es.wikipedia.org/wiki/Commit) en AccuRev consta de dos pasos. Los usuarios pueden realizar un check-in privado a su espacio de trabajo (workspace) en una operación denominada keep antes de compartirlo con el resto del grupo.[4](https://es.wikipedia.org/wiki/AccuRev#cite_note-4) Posteriormente pueden promover los cambios al stream superior mediante el comando promote.
* AccuRev se integra con diversos [sistemas de gestión de incidencias](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_incidencias) y herramientas de [gestión de proyectos](https://es.wikipedia.org/wiki/Software_de_gesti%C3%B3n_de_proyectos).
* AccuRev permite estructuras de streams remotos y replicación para equipos distribuidos. Los servidores de réplica funcionan como una caché local que acelera las lecturas y escrituras mientras mantiene sincronizada la base de código con el repositorio central.

### **DESVENTAJAS**

* Los workspaces de los desarrolladores no reciben automáticamente los cambios de su stream superior, sino que lo hacen bajo petición del usuario.
* Se pueden definir streams estáticos (snapshots) para mantenimiento de versiones ya liberadas que no deban recibir nuevas actualizaciones de otras líneas de desarrollo.
* Es posible impedir la propagación de cambios que supongan no pasar las [pruebas de integración](https://es.wikipedia.org/wiki/Pruebas_de_integraci%C3%B3n).

## [CA SCM](https://en.wikipedia.org/wiki/CA_Software_Change_Manager)

CA SCM es un [software de](https://en.wikipedia.org/wiki/Computer_software)herramienta para la [gestión de la configuración](https://en.wikipedia.org/wiki/Configuration_management) ( [control de revisiones](https://en.wikipedia.org/wiki/Revision_control) , [SCM](https://en.wikipedia.org/wiki/Software_configuration_management) , etc.) de [código fuente](https://en.wikipedia.org/wiki/Source_code) y otros activos de desarrollo de software.

### **VENTAJAS**

* Cosecha puede proporcionar tanto el control de versiones y la gestión del cambio. El desarrollador hace que los cambios en la cosecha contra un paquete de cambio (la creación de un "conjunto de cambios"). El paquete (s) modificar inicialmente consistirá en una serie de archivos que el desarrollador ha creado o modificado. Este es el componente de control de versiones de la cosecha.
* Una vez que el desarrollador está satisfecho con su / sus cambios, los cambios progresan a través de un ciclo de vida predefinido (es decir, en una serie de etapas secuenciales PRUEBA y finalmente en la producción). En todas estas etapas de este "ciclo de vida", el paquete debe tener la aprobación de los usuarios o grupos de usuarios adecuados.

### **DESVENTAJAS**

* Una rama separada o línea de desarrollo donde los cambios pueden ser aislados (el componente de control de versiones)
* La definición de los procesos y cómo los cambios de progreso a través del ciclo de vida de promoción.

## [AUTODESK VAULT](https://en.wikipedia.org/wiki/Autodesk_Vault)

Herramienta de control de versiones diseñada específicamente para aplicaciones [Autodesk](https://es.wikipedia.org/wiki/Autodesk) que gestionan las relaciones complejas entre ficheros de diseño elaborados por [AutoCAD](https://es.wikipedia.org/wiki/AutoCAD" \o "AutoCAD) y [Autodesk Inventor](https://es.wikipedia.org/wiki/Autodesk_Inventor).

### **VENTAJAS**

* Los usuarios pueden almacenar y buscar tanto los datos CAD (como Autodesk Inventor,[DWG](https://en.wikipedia.org/wiki/.dwg) y[DWF](https://en.wikipedia.org/wiki/DWF) archivos) y los documentos no-CAD (como[Microsoft Word](https://en.wikipedia.org/wiki/Microsoft_Word) y[Microsoft Excel](https://en.wikipedia.org/wiki/Microsoft_Excel) archivos).

## CLEARCASE

Rational ClearCase es una familia de [programas informáticos](https://en.wikipedia.org/wiki/Computer_software) herramientas que soporta [la gestión de configuración de software](https://en.wikipedia.org/wiki/Software_configuration_management) (SCM) de [código fuente](https://en.wikipedia.org/wiki/Source_code) y otros [de desarrollo de software](https://en.wikipedia.org/wiki/Software_development) activos. También es compatible con la gestión del diseño, los datos de los artefactos de diseño electrónico, lo que permite que el hardware y software de co-desarrollo. ClearCase incluye [el control de revisión](https://en.wikipedia.org/wiki/Revision_control) y constituye la base para la gestión de la configuración en las empresas grandes y medianas, con capacidad para proyectos con cientos o miles de desarrolladores. Es desarrollado por [IBM](https://en.wikipedia.org/wiki/IBM) .

### **VENTAJAS**

* Sólo se compilan los scripts cambiados cuando se graba o se reproduce un script, por lo que el rendimiento puede ser más rápido que el de una vista dinámica.
* Los archivos no absorben gran cantidad de espacio de disco en la máquina local porque sólo existirán en la unidad de disco duro local aquellos archivos que extraiga o que cree.
* Los archivos de una vista dinámica siempre son actuales con el VOB. No es necesario que recuerde obtener los últimos archivos del VOB como debe hacerse con una vista de instantánea.

### **DESVENTAJAS**

* Los archivos no se actualizan automáticamente. Debe recordar obtener los últimos archivos del VOB de forma regular pues, de lo contrario, los cambios que otros puedan incorporar podrían romper el script. Una actualización diaria probablemente sería suficiente, en función de las necesidades del equipo en concreto.
* Una vista de instantánea utiliza mucho espacio de disco de la unidad de disco duro local porque una vista de instantánea copia cada archivo del proyecto de Functional Test en dicha unidad de disco duro local.
* En un equipo grande, cuando muchos usuarios efectúen cambios en scripts, es posible que la compilación de todos los cambios se prolongue cuando grabe o reproduzca un script.

# lista de 20 comandos git

|  |  |  |
| --- | --- | --- |
| No | Comando | Función |
| 1 | git config --global user.name "El nombre" | Configura el nombre en git |
| 2 | git config --global user.email "El correo electrónico" | Configura el correo electrónico en git. |
| 3 | git config --global color.ui true | Habilitar los colores de respuestas de git. |
| 4 | git config --global --list | Nos muestra una lista de las configuraciones hechas. |
| 5 | git init | Este comando marca el inicio de nuestro proyecto.  Aquí le decimos a git que empiece a monitorear todos nuestros cambios. |
| 6 | git status | Muestra el estatus en que se encuentra nuestro proyecto. |
| 7 | git add (nombre del archivo) | Con este comando decidimos que archivos están listos para el siguiente paso (crea un commit). |
| 8 | git add -A | Agrega todos los archivos al siguiente nivel. |
| 9 | git commit -m "mensaje" | Guarda los cambios con un mensaje para identificarlos. |
| 10 | git log | Nos da una lista de todos nuestros commits con su respectiva información. |
| 11 | git checkout (aqui pegamos el enlace del commit)---> | Con este comando viajamos atreves del tiempo (los commits) o nuestras ramas. |
| 12 | git checkout master | Nos muestra nuevamente hasta el último commit que hicimos. |
| 13 | git reset | Es como el git checkout pero este ELIMINA los commits. |
| 14 | git reset --soft | El git reset mas simple y no toca nuestro working area (no se mete con nuestro codigo). |
| 15 | git reset --hard | Este git reset borra absolutamente todo lo que hay de commits (borra codigo). |
| 16 | git log > commits.txt | Nos genera un txt con todos nuestros commits, para tenerlo a la mano. |
| 17 | git help | Nos ayuda a entender ¿cómo funciona git?, o alguno de sus comandos (Ej: git help status). |
| 18 | git branch (nombre de la rama que vamos a agregar) | Así creamos una rama. |
| 19 | git branch | Nos muestra las ramas que tenemos. |
| 20 | git checkout (Nombre de la rama) | Nos sirve para movernos entre ramas. |

# ¿Qué ES GITHUB?

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones [Git](http://es.wikipedia.org/wiki/Git).

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

# LAS 5 MEJORES FUNCIONALIDADES DE GITHUB

## Versionar tu codigo

Guardar en determinado momento los cambios efectuados a un archivo o conjunto de archivos, con la oportunidad que tener acceso a ese historial de cambios, ya sea para regresar a una de esas versiones o para hacer comparaciones entre ellas.

## Visor de codigo

GitHub posee un estupendo visor de código mediante el cual, a través del navegador, podremos consultar en cualquier instante el contenido de archivo determinado, con la sintaxis correspondiente a el lenguaje en el que esté escrito. Este navegador es realmente rápido, y gracias a él podremos hacer pequeñas consultas o copiar porciones de código sin necesidad de bajarse todo el repositorio.

## Contribuir

Si luego de copiar un proyecto (hacer Fork) haces ajustes que arreglan bugs o introducen una nueva funcionalidad, puedes proponerle al dueño del proyecto que integre tus cambios en su código. Eso lo puedes hacer enviando un [pull request](https://help.github.com/articles/using-pull-requests/) con todas tus modificaciones o novedades. Mediante un pull request el administrador del repositorio original tendrá la posibilidad de revisar, testear o escribir comentarios sobre alguna líneas de tu propuesta, y finalmente si considera interesante tu contribución podrá realizar la fusión de esta con el código original (a esto se le conoce como hacer [merge](https://help.github.com/articles/merging-a-pull-request/) o en caso contrario rechazar los cambios propuestos.

## Aprender y experimentar

Como muchos desarrolladores tienen de forma pública sus proyectos en Github es posible acceder a su código, leerlo, estudiarlo y aprender de él, e incluso podrías hacer cambios y experimentar sin afectar el código original, a esto último se le como: hacer [fork](https://help.github.com/articles/fork-a-repo/)

## Estar informado

Github tiene un sistema de notificaciones con el cual puedes estar al tanto de las actividades alrededor de un repositorio en el cual estas participando o uno en el cual estés interesado. La notificaciones de los proyectos a los que perteneces o eres dueño, están habilitadas de forma automática, y para aquellos en los cuales no eres participantes pero estas interesado es necesario activarlas haciendo [watch](https://help.github.com/articles%20la%20/watching-repositories/) sobre el repositorio en cuestión. Algunas de novedades que se notifican son: la creación de un pull request o el reporte de un bug, un mención de tu usuario en una conversación o comentario, etc. Estas notificaciones se pueden o consultar a través de correo electrónico o directamente en Github.

# ¿Qué ES UN BRANCH O RAMA?

Para entender realmente cómo ramifica Git, previamente hemos de examinar la forma en que almacena sus datos. Recordando lo citado en el capítulo 1, Git no los almacena de forma incremental (guardando solo diferencias), sino que los almacena como una serie de instantáneas (copias puntuales de los archivos completos, tal y como se encuentran en ese momento).

En cada confirmación de cambios (commit), Git almacena un punto de control que conserva: un apuntador a la copia puntual de los contenidos preparados (staged), unos metadatos con el autor y el mensaje explicativo, y uno o varios apuntadores a las confirmaciones (commit) que sean padres directos de esta (un padre en los casos de confirmación normal, y múltiples padres en los casos de estar confirmando una fusión (merge) de dos o más ramas).

# ¿Qué ES UN PULL REQUEST?

Un pull request es una petición que el propietario de un fork de un repositorio hace al propietario del repositorio original para que este último incorpore los commits que están en el fork. En el caso que nos ocupa, el usuario aprendegit-user1 le enviará la petición a aalbagarcia para que este último incorpore los commits que tiene en su fork.

# ¿QUE ES UN HOOK?

Git tiene una innumerable cantidad de funciones. Los *"hooks"* o ganchos son otras de estas tantas funciones extremadamente útiles pero poco utilizadas por muchos de nosotros. Los *"hooks"* son un conjunto de acciones que se ejecutan antes o después de un comando particular de git, es decir, si estamos utilizando un "pre-commit hook" se ejecutará una acción antes de realizar el "commit", si dicha acción realizada antes del "commit" tiene una respuesta negativa podemos cancelar el "commit", arrojar un error y viceversa. Podemos decir que los *"hooks"* son una capa extra que nos ayudan a no pasar por alto ciertos detalles.

# RECOMENDACIONES Y TRABAJOS FUTUROS

Son observaciones de la parte práctica sobre aspectos por mejorar si alguien decide en un futuro realizar un avance, comparación, rectificación o rebatimiento al tema estudiado, así como una propuesta de los posibles trabajos futuros que complementarían o profundizarían este desarrollo.

# REFERENCIAS BIBLIOGRÁFICAS

Según normas ICONTEC. Los títulos de libros, seminarios, congresos, artículos etc. que son la base para el desarrollo del proyecto de grado. **No se admiten referencias** de portales de internet que no brinden ningún respaldo p.e. [www.wikipedia.org](http://www.wikipedia.org), [www.elrincondelvago.com](http://www.elrincondelvago.com), [www.monografias.com](http://www.monografias.com) o similares. En la medida de lo posible, las referencias bibliográficas no deben exceder de 10 años atrás.

# ANEXOS

Toda la información que sirva de sustento a los cálculos, diseños o desarrollo de la parte práctica del proyecto así como las normas aplicadas o un resumen de ellas como metodología si fuere el caso.

## Anexo 1

## Anexo 2