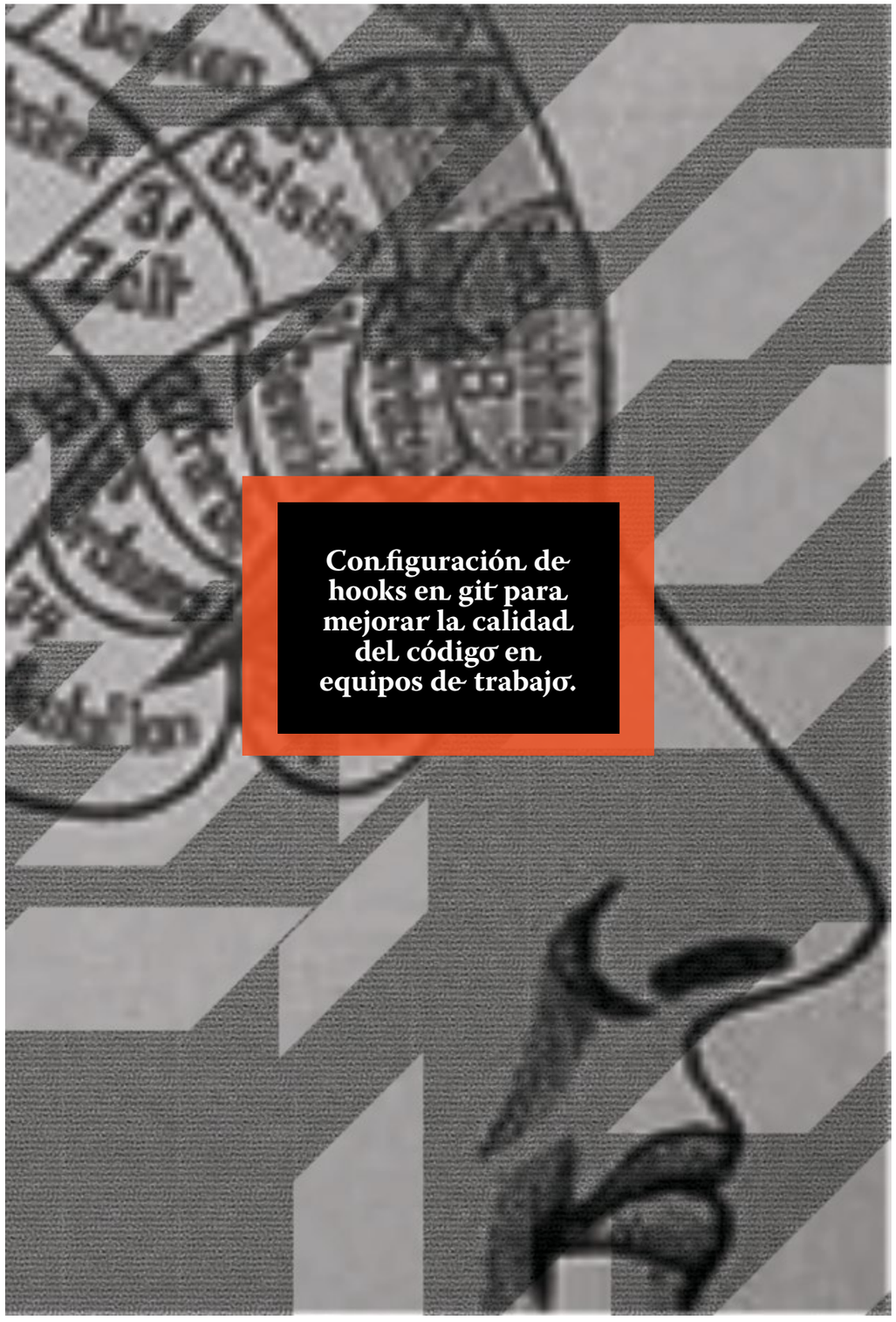

Configuración de hooks en git para mejorar la calidad del código en equipos de trabajo.

RESUMEN

Los hooks en Git son eventos accionados en momentos específicos para ejecutar acciones. Git no trae definido los hooks, simplemente deja abierto el espacio para que puedan ser adaptadas al entorno de desarrollo según las necesidades. El encargado de la Gestión de Configuración tiene la responsabilidad de accionar estos eventos para mejorar la calidad del trabajo en los grupos de desarrollo. Configurar correctamente los hooksgarantiza poder minimizar errores y validar algunas entradas, además de mantener informado de las acciones que se realicen en el repositorio central. Los hooks se agrupan en el cliente (committing y merging) y en el servidor que operan en la red (pushed), por la función que desempeñan, de validación y de información. Todos estos se pueden utilizar antes o después de ejecutar algún comando y se disparan acciones (hooks) y devuelven un estado si son de validación, o mensajes si son de información. Utilizar hooks de forma apropiada colaboran en la calidad del proyecto ya que previenen errores y validan entradas. Otra funcionalidad de los hooks es que se pueden utilizar para registrar eventos que se utilizaran en los informes o auditorias ya que en grupos de trabajos es más propenso cometer.

PALABRAS CLAVE: Clonar; Git; Hook; Linea Base; Línea de Desarrollo; Ramas; Repositorio.

ABSTRACT

"Hooks configuration in git to improve the code quality in workgroups."

The Git hooks are events triggered at specific times to perform actions. Git does not Setting up hooks correctly guarantees to minimize errors and validate some inputs, as well as maintaining informed of the actions taken in the central repository. The hooks are grouped on the client (committing and merging) and server (pushed) operating in the network due to the role they play in validation and information.All of them can be utilized before or after executing any command, and actions are triggered (hooks). They give back a state if they represent a validation or a message if they are of information. The proper use of hooks will collaborate in the quality of the product by preventing mistakes and validating inputs.Another characteristic of the hooks is that you can use them to maintain a record of events necessary for reports of future audits due to in workgroups are more likely to make mistakes and it is needed to know who is responsible for them. Bring in the hooks defined; it simply leaves an open space so they can be adapted to the development environment according to the needs. The individual, who is in charge of Configuration Management, has the responsibility to activate these events to improve the quality of development workgroups.

KEYWORDS: Baseline; Branch; Development; Git; Hook; Merge; Repository.Clone; Development line; Git; Hook; branch; Repository

José Arístides Valencia Ruiz
Universidad Técnica de Manabí Ecuador
jvalencia@utm.edu.ec

Emilio Antonio Cedeño Palma
Universidad Técnica de Manabí Ecuador
eacedeno@utm.edu.ec

José Miguel Loor Intriago
Universidad Técnica de Manabí Ecuador
jmloor@utm.edu.ec

Anaisa Hernández González
Instituto Superior Politécnico José Antonio Echeverría. Cuba
anaisa@ceis.cujae.edu.cu

Manuel Morejón Espinosa
Instituto Superior Politécnico José Antonio Echeverría. Cuba
manuel.morejon.85@gmail.com

ARTÍCULO PRESENTADO PARA REVISIÓN: 22 DE MARZO DE 2016
ARTÍCULO ACEPTADO PARA PUBLICACIÓN: 25 DE ABRIL DE 2016

INTRODUCCIÓN

En un proyecto en donde se aplica GCS se describe la necesidad de hacer algo más que controlar las versiones del código. No basta con registrar los cambios en el Sistema de Control de Versiones. Los commits deben ser realizados con la mayor calidad posible siguiendo las buenas prácticas descritas en los documentos. Git cuenta con la posibilidad de desarrollar una serie de actividades adicionales a partir de eventos específicos que ayudan a garantizar la calidad del código generado en un grupo de programadores. En los sistemas distribuidos como Git y aplicado a equipos de desarrolladores de software es muy importante tener en cuenta las entradas y validación de estas puesto que se puede trabajar sobre el mismo proyecto y al momento de actualizar en el servidor no existan conflictos de código y afecte en el resultado final.

GIT

Git en la actualidad es la plataforma preferida por proyectos grandes como Androi, Rail, Fedora, Twitter, PostgreSQL y muchos más. Siendo el más utilizado en sistemas de control de versiones en servicios web tan populares como GitHub y Bitbucket. (Patricia, 2011)

Hablar de Git como control de versiones distribuido en que todos los nodos manejan información en su totalidad, por esta razón actúan de cliente o de servidor en cualquier momento, esto se logra al sincronizar los cambios con el repositorio remoto de Git guarda una copia entera de los datos con toda la estructura y archivos necesarios y cuando el cliente lo crea necesario enviará los cambios al servidor (actualización).

Git se define como Software de control de versiones distribuido orientado al mantenimiento de versiones,en aplicaciones donde estas tienen gran número de archivos de código fuente e incluso de otros elementos de configuración; existen versiones de Git para diferentes plataformas (Windows, Mac, Linux).(Lars, 2013; Thom, 2014)

Proporciona mecanismos simples y eficaces para la gestión y mezcla de ramas, ya que en el lado del cliente se tiene una copia exacta desde el servidor y es tratado de forma natural.

Una de las variantes de Git es la gestión de ramas que se utilizan como desarrollo alternativo “rama de prueba” y poder clonarla con la rama principal “master” o con otras ramas.

“Los cambios pueden suceder en cualquier momento, son inevitables.”(Bersoff, 1980)

“No hay nada permanente, excepto el cambio”(Heráclito)

HOOKS

Son un conjunto de acciones que se ejecutan antes o después de un comando en particular, permiten la automatización de procesos de calidad que ayudan a no pasar por alto ciertos detalles de forma automática; se ejecutara una acción antes de realizar algún comando y se puede evaluar la respuesta y derivar otras acciones como cancelar el propio comando o enviar alertas, enviar correos.(Ben, 2014)

applypatch-msg(c-v).- Formatear un mensaje para cumpla un estándar deseado del proyecto, toma un solo argumento, Git aborta si el resultado es diferente de cero.

pre-applypatch(c-v).- se utiliza para inspeccionar la rama donde se está trabajando antes de hacer el envío de datos, se puede negar sino pasa cierta prueba cuanto retorna un valor diferente de cero.

post-applypatch(c-i).- se utiliza para enviar notificaciones por la consola, se ejecuta cuando la confirmación se realiza.

pre-commit(c-v).- Encontrar espacios en blanco al final de cada línea. Además se pueden agregar cualquier cantidad de pruebas para asegurar y validar las entradas.

prepare-commit-msg(c-i).- preparar el comentario en un formato deseado

commit-msg(c-i).- funciona con applypatch-msg, toma un parámetro que es el camino a un archivo temporal que contiene el mensaje escrito por el desarrollador, si este scrip devuelve distinto de cero, Git aborta el proceso.

post-commit(c-i).- funciona con post-applypatch pero después del commit, imprime un mensaje de notificación pero actúa una vez que se realizó el commit

pre-rebase(c-v).- previene si una rama es rebasada detiene el proceso si es el valor es diferente de cero.

post-checkout(c-v).- se activa al utilizar checkout y muestra diferencias entre ramas

post-merge (c-v).- se activa por el comando merge se utiliza para salvar o restablecer cualquier metadata asociado a una rama y no se ejecuta si existen conflictos.

pre-receive (s-v).- es activado git-receive-pack en el repositorio remoto. Se encarga de actualizar las referencias de los objetos ocurre cuando se realiza un gitpush, su estado de salida determina el éxito o fracaso de la actualización.

Update (s-v).- parecido al pre-receive con la diferencia de prevenir un gitpush -f (forzado) y toma tres argumentos (nombre rama, imagen del push, nombre usuario).

post-receive (s-i).- muy parecido a update y pre-receive la diferencia actúa del lado del servidor se puede utilizar para enviar correos o verificar alguna información importante como un sistema de tickets.

post-rewrite (c-v).- es invocado por los comandos que sobreescribencommits, se puede utilizar para configurar correctamente el directorio de trabajo de un proyecto.(Ben, 2014)(Kernel.org)

Es importante acotar que los hooks se ejecutan en el cliente y en el servidor, los que se ejecutan en el cliente al momento de clonar un repositorio no se copian al servidor, se necesita implementarlos en el servidor para validaciones.

(s-v) Servidor, Validación	(c-v) Cliente, Validación
(s-i) Servidor, Información	(c-i) Cliente, Información

Son muy útiles en proyectos en donde existen grupos de trabajo y poder prevenir comportamientos y acciones no deseadas.

COMPARACIÓN CON OTROS SISTEMAS

En otros sistemas de GCS como mercurial los hooks son acciones de información y validación muy parecido a GIT y devuelven un valor verdadero si la acción se llevócon éxito o diferente de cero para indicar el fracaso del mismo, tienen una particularidad que se pueden ejecutar a nivel de usuario y dependiendo de los privilegios que tengan por este motivo el usuario que los utilice debe de conocer bien su funcionalidad.

En Subversión son scripts que se guardan en el servidor y se ejecutan cuando ocurren ciertas acciones. Cuando se ejecutan estos hook y todo sale normal la salida es cero;diferente de cero con error, lo hará por medio de un mensaje, estos hook pueden ser editados y modificados para añadir funcionalidades o validaciones.(Fernando)

En Bazaar la mayoría de hooks se ejecutan por parte del cliente, pero algunos en el servidor el modo lo hace por medio de plugin y no por comandos, también existen de dos tipos de información y de error.(Web-2)

En resumen los Hooks como se describe en el párrafo anterior se ejecutan de forma muy similar pero en Git tienen un uso más dinámico y más minucioso ya que se distribuyen en el cliente y en el servidor, en el cliente se ejecutan antes de enviar la información al servidor así minimiza los errores y conflictos con otras ramas y lo hace en la última captura de los ficheros de código y los del servidor se ejecutan cuando existen actualizaciones.

ANTIPATRONES: (WEB-1)

- Evitar la fusión a toda costa, por lo general de un temor a las consecuencias.

- Gastar demasiados tiempo en la fusión de configuración, en lugar de su desarrollo

- Aplazar la fusión hasta el final de las actividades de desarrollo y de intentar fusionarlas simultáneamente.

- Fusiones continuas, siempre queda algo por integrar

- Fusionar una versión de componentes con una versión obsoleta

- Generar ramificaciones sin una versión aparente

- Generar ramificaciones, pero nunca actualizar la línea base

- Detener todas las actividades de desarrollo, mientras que ejecutan ramificaciones, fusiones, o la construcción de nueva línea base.

- Las ramificaciones dividen al equipo de desarrollo, en lugar de dividir el trabajo

En conclusión la mayoría de SGCS utilizan hooks con el mismo fin independientemente de la forma u configuración y con mayor énfasis en los sistemas distribuidos y código abierto como lo hace Git.

SISTEMAS EXTERNOS

Existen sistemas externos que se ven beneficios por los Hooks de Git los cuales se pueden agrupar en dos:(Alicia, 2014)

- Gestores de Proyectos.- es una herramienta o software que permite llevar un control o un registro minucioso de cada uno de los proyectos de una empresa en particular. Incluye todas las partes del trabajo, como planificación, desarrollo, producción y en otros ámbitos como clientes, trabajadores, tareas.

Facilitan las tareas de administración al responsable de los grupos de desarrolladores o de los distintos proyectos. Algunas funciones de los gestores de proyecto como gestión del tiempo de cada usuario Gestionar dependencias de tareas cual se ejecuta primero, cual está pendiente y el avance de cada una, llevar un control de las comunicaciones entre usuarios (Ruby onRails utilizado para desarrollo web)[1] se integra muy bien con Netbeans[2] .

- Integración continua.- forma parte de una metodología de desarrollo basado en test o guiado por pruebas que funcionan creando primero las pruebas antes de generar código fuente. Las pruebas son test unitarios y de integración que prueban las capas de aplicación en diferentes niveles (acceso a datos, procesamiento de datos). Todos los lenguajes modernos incluyen una aplicación que crea un protocolo para llevar a cabo los test e informan de los resultados. Algunos sistemas como Makefile, Maven, en diferentes lenguajes como (Perl, Ruby, etc.). En conclusión consiste en hacer integraciones automáticas (compilación, ejecución de test) de un proyecto lo más a menudo posible y así detectar fallos lo antes posible.

BUENAS PRÁCTICAS APLICADO A LOS HOOKS

Durante el almacenamiento del código en estos sistemas de control de versiones existe la posibilidad de incluir elementos de calidad empleando los hook. Dentro de estos elementos puede estar la presencia de documentación en el código, el respetar formatos para escribir código de forma legible, la ejecución de pruebas, el mezclar de forma automática, entre otros. Estos elementos deben ser descritos y combinados para brindar un proceso de desarrollo con calidad.

Los hooks en Git están organizados de los dos extremos (Cliente-Servidor). Se deben de utilizar del lado del cliente los que se han diseñado para este fin ya que la mayoría al inicio se extrae una copia exacta del servidor del proyecto, pero no se copian los hooks, y tampoco se actualizan en el sentido contrario, tener en cuenta este aspecto contribuye al buen desempeño minimizando errores y más aún en equipos de desarrollo.(Erick, 2014)

CONCLUSIONES

Los hooks en Git se crean para cada proyecto en el servidor y cada cliente, debe de ser configurado cada vez que exista un nuevo proyecto; lo que si se pueden es copiar el scrip al nuevo directorio del proyecto.

1. *Rail es un Framework Web escrito en lenguaje Ruby*
2. *IDE de desarrollo código fuente en JAVA*

Los Hooks en Git, son muy útiles ya que es un sistema distribuido y especialmente para grupos de trabajo, bajo estas características permiten llevar un control histórico y actual de las acciones que se realicen especialmente en el servidor antes y después de ejecutar algún comando, ya sean estos de información y validación, lo interesante del buen uso que se les de, que pueden prevenir errores y en consecuencia mejorar la calidad del Software. En grupos de Desarrollo de Programas por el solo hecho de poder compartir código en proyectos específicos son más propensos a los conflictos en el código fuente lo interesante de esto que se pueden validar formatos respetando un formato.

Por desconocimiento de estas herramientas son poco utilizadas en proyectos más aun donde trabajan muchas personas es donde se puede prevenir comportamientos no deseados. En conclusión los hooks pueden identificarse como elementos que ayudan a la calidad del código siempre que sean utilizados.

GitHub resulta ideal para trabajar con cualquier sistema de integración continua y de hecho hace uso de los hooks como una herramienta que proporciona un control automático en especial del código fuente.

REFERENCIAS BIBLIOGRÁFICAS.

Alicia, S., Patricio, M., Alejandra, B., Natalia, M., Sofia, P., Francisco, C. (2014). La Integración Continua Aplicada en el Desarrollo de Software en el Ámbito Científico – Técnico.

Ben, S., Scott, C. (2014). Pro Git Everything you need to know about Git.

Bersoff et al. (1980).

Erick, P. (2014). Git Best Practices Guide.

Fernando, L. Gestión de Versiones con CVS y Subversion.http://www.fossil-scm.org/fossil/.

Heráclito, 500 a.C.

Kernel.org, Portal GitHooks Manual Page. www.kernel.org/pub/software/scm/git/docs/githooks.html .

Lars, V. (2013). Distributed Version Control with Git.

Patricia, M., Carlos, A., Arcelor, M., Vicente, R., Luis, F. (2011). Análisis y Evaluación de Herrmientas de Control de Versiones en Proyectos de Software.

Thom, P. (2014). Git Fundamentals.www.it-ebooks.info.

Web-1, Portal Gestión de la Configuración. http://www.fing.edu.uy/tecnoinf/mvd/cursos/ingsoft/material/teorico/is11-SCM.pdf Consultado el 20-03-2015.

Web-2, Portal Bazaar la herramienta para el control de versiones de forma distribuida.http://bazaar-vcs.org/ http://bazaar-vcs.org/Documentation Consultado el 20-03-2015.