Configuración de usuarios en git para grupos de desarrollo en entornos universitarios.

RESUMEN

El empleo de sistemas para el control de versiones del código fuente es una tarea cotidiana en los grupos de desarrollo de software. Este sistema no trae incorporado la gestión de usuarios en grupos de desarrollo, se necesita garantizar seguridad y control de cambios y así minimizar errores y alcanzar la calidad. Los sistemas de control de versiones necesitan de la gestión de usuarios para el trabajo en equipos. Git es una herramienta de control de versiones que se encuentra dentro de los más utilizadas, revelando una popularidad superior al 50% debido a la eficiencia y confiabilidad para quienes lo han implementado en la actualidad. Sistemas como GitHub, GitLab, BitBucket que también llevan a cabo soluciones para cubrir este propósito. De igual forma existen sistemas externos con Gitosis y Gitolite creados con el objetivo de integrarse a Git y garantizar el mismo objetivo. Se identifican las pautas que deben tenerse en cuenta en la selección de propuestas a un determinado grupo de trabajo, el entorno de desarrollo y determinar el escenario. Bajo esta premisa este documento busca dar a conocer una solución que permita identificar según las necesidades, el entorno y vincular el control de usuarios al control de versiones.(Anaisa, 2010)

PALABRAS CLAVE: Control de usuarios; control de versiones; gestión de configuración; git; gitlab; gitosis.

ABSTRACT

The use of systems for source version control code is a daily task in software development groups. This system brings independent user management necessary for development groups, we need to guarantee security and control of changes and errors are minimized and quality is achieved. The version control systems require the user management for teamwork. Git is a version control tool that is located in the most used superior popularity revealing a 50% due to the efficiency and reliability for those who have implemented today. Systems like GitHub, GitLab, BitBucket also performed solutions to serve this purpose. Also, there are external systems with Gitosis and Gitolite created with the objective to integrate Git and guarantee the same goal. A guideline to be taken into account in the selection of proposals to a specific working group was identified. All these points we should consider the development environment and determine the stage. Under this premise, this paper seeks provide a solution that allows recall identifying solutions to the needs, environment and link the user control to version control.

KEYWORDS: Configuration management; git, gitlab; gitosis; user control; version control

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

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

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

Anaisa Hernández González, Ph.D
Instituto Superior Politécnico José Antonio Echeverría (CUJAE - CUBA)
anaisa@ceis.cujae.edu.cu

Emilio Antonio Cedeño Palma, Ing
Instituto Superior Politécnico José Antonio Echeverría (CUJAE - CUBA)
manuel.morejon.85@gmail.com

ARTÍCULO PRESENTADO PARA REVISIÓN: 15 DE ABRIL DE 2015
ARTÍCULO ACEPTADO PARA PUBLICACIÓN: 15 DE MAYO DE 2015

INTRODUCCIÓN

En la actualidad, la mayoría de las empresas que desarrollan software han visto la necesidad de llevar el control de versiones de su código fuente, más aun en ambientes de grupos de desarrolladores, agrupando todas estas necesidades se llega al término global Gestión de Configuración de Software (GCS). En los grupos de desarrolladores, el control debe asegurar la calidad. De aquí nace el término integración, y se hace necesario utilizar una herramienta distribuida y que además incluya otro elemento como la seguridad que empieza por la necesidad de permisos de usuarios (Alicia, 2014).

Existen varias herramientas de GCS de las cuales no incluiremos las que son de licencias pagadas, ya que la objetivo de este artículo va orientado a un entorno universitario en Ecuador donde se promueve la utilización de software libre [Acuerdo-1014, 2008]. Por otro lado va dirigido a un grupo de desarrolladores y como una solución de enseñanza-aprendizaje, por el número de usuarios concurrentes es necesario una herramienta que facilite un entorno distribuido por eso no abordaremos las herramientas cuyo repositorio sea centralizado. (Alicia, 2014)

La importancia de utilizar una herramienta de control de versiones que facilite el almacenamiento de los elementos a gestionar, y la administración de usuarios de forma colaborativa que se adapte al entorno de trabajo y que además defina el escenario nos lleva a analizar la herramienta a utilizar más aun ya en grupos de desarrollo. (Patricia, 2011).

Para el estudio en el entorno universitario se tomó en cuenta criterios de valoración y se definieron dos escenarios. El primero está orientado al grupo de desarrollo de aplicaciones que cubren los requerimientos institucionales para automatizar procesos y el segundo va orientado a los estudiantes con fines académicos de aprendizaje.

Grupo de desarrollo: Gestiona la configuración con todos sus elementos delimitado solo al grupo responsable de cada proyecto; en este escenario conlleva más seguridad ya que se debe gestionar los permisos de acceso a código fuente.

Grupo Estudiantes: orientado al desarrollo de proyectos en clases los cuales serían compartidos en un repositorio en el entorno universitario así poder ir mejorando proyectos desde diferentes aristas y colaborar en el aprendizaje, además de reducir el tiempo en revisión de código por parte de docentes en cualquier momento y cómo van los avances.

En la actualidad empresas desarrolladoras de Software le dan la importancia del control de versiones en los grupos de desarrollo y esto desencadena la necesidad de controlar los permisos de los usuarios al código fuente.

El objetivo de este artículo es dar lineamientos y proponer una herramienta de GCS que se adapte al entorno de trabajo universitario ecuatoriano y en concreto la Universidad Técnica de Manabí como caso de estudio. Particularmente en la configuración de usuarios para tener el control de cambios reducir errores y maximizar la calidad. (Lars, 2013)

GITHUB

Es un servicio distribuido utilizando control de versiones GIT. Utilizado como repositorio online de código fuente para proyectos de código abierto. Podemos encontrar información de códigos fuentes, proyectos, funciones, manuales, framework ya que es público y lo que se necesita es una suscripción sin ningún costo, y además ofrece el servicio privado pero ya tiene un costo. (Web-1)

Desde el punto de vista de una comunidad o de una red social (hacer social nuestro código), se ha convertido en uno de los sitios referentes de albergue público de proyectos GIT de código abierto y unos de los pocos que permite albergue privado con funcionalidades extras en especial la seguridad (permisos y privilegios) (Peter, 2015).

Su principal fin es compartir código libre en donde los suscriptores tienen acceso a cierto código que se encuentra en estado público. Si se desea tener una licencia privada ya tiene costo.

GITLAB

GibLab es un sistema de gestión de proyectos mediante interfaz web que brinda la posibilidad de no depender de la aplicación en la nube (GitHub). Su principal característica es ser privado (local-independiente) que es donde radica la diferencia con GitHub; otra funcionalidad que tiene es que proporciona para migrar repositorios automáticamente. De igual forma incorpora un sistema wikis, la cual ayuda a documentar algunos aspectos de código y algunas funcionalidades extras como:

Login, Diferentes tipos de acceso o permisos, seguimiento de incidencias, revisión de código, Gestión de equipos de grupos, interfaz Web, copias de seguridad (Peter, 2015). Permite gestionar grupos, personas y permisos a los mismos que esté vinculado a algún proyecto en particular, hacer seguimiento del estado actual e histórico del mismo.

Para la gestiona los usuarios la primera regla es no usar el usuario administrador para los proyectos sino que se debe crear un usuario para gestionar los repositorios desde la cuenta administrador. Trabaja como un repositorio remoto para poder compartir los proyectos aunque realmente es un repositorio local.

Cuando existan múltiples proyectos que buscan un mismo fin se pueden agrupar para que facilite su administración, como por ejemplo en el caso de estudio en un entorno universitario los trabajos de investigación de los docentes a sus estudiantes en un mismo periodo.

Los permisos se asignan cuando se crea los usuarios o a su vez cuando se crea un proyecto se puede asignar los permisos con sus diferentes tipos de acceso y podrá hacer sus diferentes tareas asignadas. (Jeroen, 2014).

BITBUCKET

Es un servidor especial, solo miembros de equipos tienen acceso y privilegios en este servidor. Los equipos tienen acceso a este servidor, los cuáles se asignan automáticamente los privilegios al agregarse. El fakehost oculta el tiempo de inactividad (host virtual) que incrementa la privacidad.

GITOSIS

Básicamente es un conjunto de scripts que ayudan a gestionar controles de acceso, controles de permisos (control de usuarios). Vendría a ser un repositorio especial de GIT.

Cabe indicar que GIT no gestiona usuarios ya que se maneja con claves públicas para todos los Usuarios, es decir, que todos los usuarios tendrían acceso a todos los proyectos de lectura y escritura ya que lo maneja en un solo archivo: authorized_keys

La forma cómo gestiona los usuario Gitosis crea directorios por usuario y por proyecto. Cuando se instala en el servidor Gitosis toma el control del archivo authorized_keys y se genera otra carpeta gitosis-admin que donde se manejaría los usuarios y que solo tendría acceso el administrador.

GITOLITE

Funciona como repositorio de GIT y cuando se crean proyectos se generan se los permisos a usuarios en este repositorio que se desean controlar y en consecuencia funcionar como cualquier repositorio de GIT con el nombre de gitolite-admin.(Sitaram, 2014)(Web-3)

PROTOCOLOS DE GIT

Git se puede utilizar cuatro protocolos para transferir datos:

- Local.- Donde el repositorio remoto simplemente otra carpeta en el disco basada en compartir carpeta y archivos. La desventaja es el acceso de distintas ubicaciones a carpetas compartidas por esta razón se hace más lento.
- SSH (Secure Shell).- Este protocolo sería el más habitual para GIT por la disponibilidad en la mayor parte de los servidores. Además, es el único protocolo de red con el que se puede leer y escribir fácilmente. Otra ventaja que es un mecanismo de autenticación segura ya que la información se transfiere de forma encriptada, además los datos los comprime de forma eficiente.
- Git.- Es un protocolo especial que se incorpora al instalar GIT, escucha por un puerto dedicado (9418), el servicio es similar al SSH, pero sin ningún tipo de autenticación, la ventaja que es el más rápido de todos los disponibles utiliza los mismos mecanismos del protocolo SSH pero sin encriptación y autenticación.
- http/https.- Su principal ventaja es la simplicidad de habilitarlo. Suficiente con ubicar los archivos en la raíz de http o https y ejecutar un comando (hook/post-update). Basta con tener acceso al servidor web.

(WEB-4)

Se puede hacer uso de los repositorios de solo lectura con el puerto https siendo las transferencias encriptada pero solo de lectura y para más seguridad se puede incluir certificados SSL

Git cuenta con múltiples protocolos para intercambiar la información. Se debe analizar las posibilidades que tienen estos sistemas para utilizarlos y evaluar las mejores variantes. (ssh, http, https, otros) referirse en forma breve del funcionamiento de cada uno cual es más seguro y en que entorno.

ENTORNOS DE TRABAJOS

Para cada uno de los escenarios descritos a continuación cuanta con características y objetivos sociales distintos (Anaisa, 2010).

Los grupos de trabajo por números de personas pueden ser los siguientes:

- Grupos de menos de 10 personas.
- Grupos de hasta 50.
- Empresas.
- Entornos universitarios.

En el caso de entornos universitarios existen particularidades con otros entornos ya que su fin es académico e investigativo. En los dos escenarios identificados en el equipo de desarrollo el software que se desarrolla es para atender los requerimientos internos en su mayoría procesos académicos. En otro escenario a los estudiantes se lespropone que los trabajos de investigación desarrollados utilicen un repositorio en común (comunidad) y ser compartidos por los estudiantes y docentes.

VALORACIÓN DE SOLUCIÓN A CADA ENTORNO.

En dependencia de cada grupo es recomendable utilizar uno u otro sistema para la gestión de usuarios.

Hay tres tipos de sistemas de control de versiones disponibles. Estos se clasifican en función de su modo de operación:

1. Sistema de control de versiones Local
2. Sistema de control de versiones centralizado
3. Sistema de control de versiones distribuido (Ravishankar, 2013)

De manera general se debe identificar los recursos necesarios para llevar a cabo estas soluciones. En dependencia de esto podrían formar parte o no para incluirlas como solución en los entornos de trabajo.

En el caso que se propone para este entorno universitario la solución iría por un GIT como gestión de configuración ya que esta se monta en un sistema distribuido y Gitolite como herramienta de gestión de usuarios en el caso de un escenario de grupo de desarrollo ya que su fin es atender los requerimientos internos y demanda de más seguridades, aumentar la calidad y minimizar los errores.

Otra variante está el escenario de la interacción docente-estudiante donde se busca es poder compartir código fuente de proyectos de investigación en materias de programación (comunidad de código social), con fines académicos y seria GitLab la solución acorde con las necesidades demandada por el entorno ya que ofrece una interfaz amigable y web en donde cada estudiante puede crear sus proyectos de acuerdo los lineamientos del Docente. GitLab integra la gestión de usuarios de una forma automatizada y facilita la administración de grupos más grandes con menor rigurosidad. Ya que en el entorno universitario y más aún que en Ecuador (Acuerdo-1014, 2008). Firmado el 10 de abril del 2008 donde se promueve la utilización del software libre.

ACTIVIDADES POR PARTE DEL GESTOR DE CONFIGURACIÓN DE SOFTWARE

Es importante incluir en los entornos de trabajo las actividades que debe incorporar el gestor de configuración de software para que los usuarios puedan trabajar: capacitación, control y seguimiento durante el proceso de desarrollo.

Los usuarios del repositorio GIT deben de tener un conocimiento sobre el funcionamiento de la herramienta desde el administrador del repositorio, administrador del proyecto y los usuarios (grupo de desarrollo, docentes, estudiantes del entorno universitario para el caso de estudio).

Tener una guía clara de la funcionalidad y alcance del Gestor de Configuración del software se podrían hacer capacitaciones y ensayos con ejemplos prácticos se podría empezar por tener dos ambientes el de ensayo y el de producción.

En un ensayo se le podría hacer seguimiento para ver en donde tendrían más problemas al momento de utilizar el control de versiones ya que en cada entorno o lugar de implementación puede variar de acuerdo al nivel de conocimiento.

Para el grupo de desarrollo que tiene un nivel superior definir las actividades, responsabilidades en cada proyecto.

CONCLUSIONES

Lo primero que hay que tener claro y bien definido es el entorno de trabajo, los fines de la empresa, para poder determinar la solución más apropiada para poder integrar la gestión de configuración y la gestión de usuarios que vendría hacer una solución que cumpla con las exigencias de calidad y se obtengan los resultados esperados. Se definiría quien tiene acceso y a qué los cuales se definen los niveles de permisos.

Tener en cuenta lo siguiente al momento de elegir las herramientas a utilizar en el control de versiones:

- Tipo de repositorio: Distribuido por ser un entorno de un grupo de desarrollo.
- Modelo de concurrencia: múltiples usuarios de manera simultánea copy-modify-merge (CMM).
- Modelo de almacenamiento: almacena el árbol al antes y después snapshot.
- Alcance del cambio: árbol (no ficheros individuales).
- Identificación de las revisiones: números, funciones, criptografía.(Patricia, 2011)

Brinda de forma consolidada y sencilla las principales sugerencias a tener en cuenta para garantizar una buena gestión de usuarios durante la implantación de Git a un entorno de trabajo. (Erick, 2014)

REFERENCIAS BIBLIOGRÁFICAS.

Acuerdo-1014, Registro Oficial - Ecuador (2008). Utilización de Software Libre en sus Sistemas y Equipamiento Informático. http://www.administracionpublica.gob.ec/software-libre/

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.

Anaisa, H., Martha, D. (2010). Buenas Prácticas en el Desarrollo de Software en Entornos Universitarios.

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

Jeroen van, B. (2014). GitLab Cookbook - Quick answers to common problems.

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.

Peter, B., Brent, B. (2015). Introducing GitHub - O'REILLY®.

Ravishankar, S. (2013). Git: Version Control for Everyone

Sitaram, C. (2014). Gitolite Essentials - Leverage powerful branch and user access control with Git for your own private collaborative repositories.

Web-1, Portal Web Comunidad GitHub. http://github.comhttp://book.git-scm.com/index.html.

Web-2, Portal Web Privilegios BitBucket. http://www.onlinegamesnet.net/devNull.php?action=viewclasses.

Web-3, Portal Web GitHub. https://github.com/sitaramc/gitolite#start http://www.linuxforu.com/developers/gitolite/.

Web-4, Portal Web Git. http://www.kernel.org/pub/software/scm/git/docs/howto/setup-git-server-over-http.txt.