**UNIVERSIDAD TECNOLÓGICA CENTRO DE VERACRUZ**

Universidad Tecnológica Centro de Veracruz

INGENIERÍA EN DESARROLLO Y GESTIÓN DEOFTWARE

ALUMNO:

MENDEZ BELLO GLADYS

MATRICULA:

12492

9 IDGS V-S

PROFESOR:

ITI FRANSISCO TRUJILLO ROMERO

LECTURA TÉCNICA

PRIMER PARCIAL

04 de junio 2022.

• Introducción

Esto es importante para mí porque es una buena práctica para los equipos de desarrollo de software y DevOps exitosos. El control de versiones también ayuda a los desarrolladores a moverse más rápido y permite que los equipos de software se mantengan eficientes y ágiles a medida que el equipo se expande para incluir más desarrolladores.

Además que proporciona una gran ventaja para su uso al proporcionar un historial completo de modificaciones a largo plazo para todos los archivos. Tener un historial completo le permite volver a versiones anteriores para ayudar a analizar la causa raíz de los errores y es muy importante cuando necesita solucionar problemas en versiones anteriores de un programa y mantener el proyecto en marcha. En la dirección correcta, acelerar la traducción y solucionar los problemas y errores que existan

Capítulo I.-

* 1. GITHUB

GitHub puede ser el primer nombre en el alojamiento de repositorio de código abierto, pero cuando se trata de servicios de alojamiento de repositorios privados con campanas y silbatos adicionales dirigidos a empresas, el juego no es tan sencillo. Los tres proveedores basan sus plataformas en el sistema de control de versiones distribuido de Git. Cada uno ofrece una solución local para alojar informes dentro de su firewall: GitHub Enterprise, GitLab y Bitbucket Server.

GitHub tiene una gran UI intuitiva, informativa y familiar. Si está instalado en sus propios servidores, la mayoría de los desarrolladores se sentirán como en casa. Presenta integración LDAP, integración con JIRA y muchas otras características importantes para las empresas de hoy. Difícilmente necesitarás algo que GitHub no hace. Además de las "características empresariales", tiene Gists (una forma de compartir fragmentos de código), la capacidad de editar archivos directamente desde el navegador (es útil al tomar notas durante una reunión), funciona con casi cualquier proveedor de la nube, tiene una búsqueda poderosa , y así. Viene con su propio Wiki y seguimiento de problemas. En muchos casos es suficiente para muchas necesidades. Si no es así, se puede conectar fácilmente con Jenkins, JIRA, Confluence, BugZilla, Trello y muchos otros productos. Se puede usar como un paquete completo o en conjunto con otros. En pocas palabras, GitHub es el estándar de oro para repositorios de código.

* 1. GITLAB

GitLab hace casi todo lo que hace GitHub. Las características son más o menos las mismas (con muy pocas excepciones) y la interfaz de usuario es tan buena como la de GitHub. Si va al sitio web de Git Lab, le resultará difícil no pensar que está en GitHub. Uno debería buscar mucho para encontrar algo significativo que GitHub tenga y GitLab no omita lo que distingue a GitLab de GitHub.

GitLab Community Edition es gratuito y de código abierto. Eso en sí mismo le da un gran impulso en comparación con GitHub Enterprise Edition (la única versión que se puede instalar en las instalaciones). Si bien hay algunas funciones disponibles solo en la edición empresarial, es posible que no las necesite e incluso si lo hace, hace que la decisión sea más fácil al saber que puede comenzar con la versión gratuita y actualizar a la versión paga más adelante. No significa que GitHub no tenga una versión de prueba, la tiene. Pero es por un tiempo limitado que podría no ser suficiente. Con la edición comunitaria de GitLab, puede sentirse cómodo hasta el final de los tiempos o puede comenzar con la edición empresarial de inmediato. Lo bueno es que tiene una opción que no caduca.

El precio de la edición empresarial es razonable (varias veces menor que con GitHub). A diferencia de GitHub, GitLab se puede instalar en prácticamente cualquier combinación de hardware / SO / CM. Proveedores en la nube, máquinas virtuales, directamente en el servidor, Linux, Windows, Puppet, Chef, etc. Incluso mantienen imágenes de Docker. Además, la instalación es simple. Muestra el poder del código abierto con infinitas combinaciones que surgió y escribió sobre la comunidad.

* 1. BITBUCKET

Bitbucket es un servicio de alojamiento web para código fuente y proyectos de desarrollo que utilizan sistemas de control de revisión Mercurial (desde su lanzamiento) o Git (desde octubre de 2011) propiedad de Atlassian. Bitbucket ofrece planes comerciales y cuentas gratuitas. Ofrece cuentas gratuitas con un número ilimitado de repositorios privados (que pueden tener hasta cinco usuarios en el caso de cuentas gratuitas) a partir de septiembre de 2010. Bitbucket se integra con otro software de Atlassian como JIRA Software, HipChat, Confluence y Bamboo.

Es similar a GitHub, que utiliza principalmente Git. En septiembre de 2016, Bitbucket anunció que había llegado a 5 millones de desarrolladores y 900,000 equipos en su plataforma. Bitbucket tiene 3 modelos de implementación: Cloud, Bitbucket Server y Data Center.

GitHub vs bitbucket

GitHub admite:

- La importación de Git, SVN, HG, TFS.

GitLab admite:

-La importación de Git.

-Importación fácil desde otros servicios GitHub, Bitbucket, Google Code, Fogbugz.

Bitbucket admite:

- La importación de Git, CodePlex, Google Code, HG, SourceForge, SVN.

Capitulo II.-

* 1. GIT FLOW

Estrategia de branching: GitFlow

La idea principal detrás de GitFlow es aislar el trabajo en diferentes tipos de branches, lo que le permite adaptarse muy bien al proceso colaborativo que necesita un equipo de desarrollo. GitFlow está basado principalmente en dos branches que tienen una vida infinita:

**master**: contiene el código de producción.

**develop**: contiene el código que ha finalizado desarrollo.

Adicional a estos branches principales, durante el desarrollo se crean otros branches de soporte que tienen una vida finita, es decir, existen mientras exista el desarrollo:

**feature:** se crea a partir de develop cuando una nueva funcionalidad necesita ser desarrollada. Al finalizar el desarrollo se hace merge a develop nuevamente.

**release**: se crea a partir de develop para preparar una nueva versión del código que debe ser liberada en producción. Al finalizar el desarrollo se hace merge a develop y a master.

**hotfix**: se crea a partir de master cuando es necesario corregir un error detectado en producción de manera urgente. Al finalizar el desarrollo se hace merge a develop y a master.

* 1. GITHUB FLOW

GitHub Flow, fue creado por GitHub y es conocido en la comunidad de desarrolladores como una alternativa simple y ligera a GitFlow. GitHub Flow se basa en un flujo de trabajo basado en branches que permite a equipos de desarrollo enfocarse principalmente en la entrega continua. A diferencia de Git Flow, no existen los branches de “releases”, ya que está pensado para que la implementación en producción ocurra con frecuencia, incluso varias veces al día si es posible.

En esta estrategia de branching, en el repositorio tenemos dos tipos de branches:

**main (o master):** el branch de código principal, es el que contiene el código que está listo para producción.

**features:** los branches de funcionalidades que permiten el desarrollo en paralelo.

Cuando utilizas GitHub Flow, existen seis principios que debemos seguir para asegurarnos de mantener un buen código listo para producción en todo momento:

* El código que está en \_main debe poder implementarse en producción en cualquier momento.\_
* Cuando se crean nuevos \_feature branches, se deben crear con nombres descriptivos. Por ejemplo, feature/add-new-account-type.\_
* Primero hacer commit en local, para luego hacer push al repositorio remoto.
* Abrir pull requests para solicitar feedback o ayuda, antes de hacer merge en el branch principal.
* Hacer merge del código en el branch \_main, solo después de haber recibido aprobación del pull request.\_
* Una vez hecho el merge del código en \_main, debe implementarse de inmediato, mejor si es en un entorno de producción.

GitHub Flow es ideal cuando necesitas mantener una única versión del código de producción.

* 1. GITLAB FLOW

GitLab Flow se caracteriza por un flujo de trabajo bastante similar a GitHub Flow. La principal diferencia es que no solo permite el uso de release branches, sino que adiciona el uso de environment branches, por ejemplo, QA, Pre-Producción y Producción. Esto debido a que considera los casos donde una nueva funcionalidad no siempre puede implementarse en entornos de producción.

Al igual que GitHub Flow, GitLab Flow propone utilizar un master branch y feature branches. Una vez que terminamos el desarrollo en un feature branch, hacemos merge de nuevo con master.

Con respecto a los branches por entornos, estos se crean a partir de master cuando estamos listos para implementar nuestra aplicación. Tener diferentes ramas por entorno nos permite configurar herramientas de CI y CD para que se despliegue automáticamente los commits realizados en cada una de esas ramas. Si detectamos un problema crítico, lo solucionamos en el feature o en master, una vez corregido, hacemos merge con las ramas de los entornos.

Con respecto a los release branches, seguimos haciendo todo el trabajo en feature branches que luego hacemos merge a master al finalizar. Cuando nos aseguramos que master es lo suficientemente estable, es decir, ya hemos realizado todas las pruebas y corregido todos los errores, creamos un release branch que va a contener nuestra versión de código para desplegar. Si hay un problema crítico, primero lo solucionamos en master, realizamos cherry picking para hacer merge en release, es decir, seleccionamos las porciones de código que deseamos que contenga nuestro release.

* 1. ONE FLOW

OneFlow fue concebido como una alternativa más simple a GitFlow. Sin embargo, “más simple” no quiere decir que permita hacer menos.

Con OneFlow es necesario tener un master branch que tendrá una vida infinita en el repositorio de código, además, cada nueva versión de producción debe basarse en la versión anterior, una condición que la mayoría de proyectos de software cumple. La mayor diferencia entre One Flow y Git Flow es que en OneFlow no existe develop.

Si bien el flujo de trabajo aboga por tener un branch de larga duración (master), eso no significa que no haya otros branches involucrados. Lo que sí es cierto es que cada uno de estos branches de soporte, por ejemplo, feature, hotfix, etc. deben ser de corta duración, se crean a partir de master y deben, una vez finalizado el desarrollo, volver a master. Los branches de soporte, facilitan el intercambio de código entre los desarrolladores y actúan como respaldo. Sin embargo, el historial de cambios de todo el código, siempre se encontrará en el master branch.

• Conclusiones

Aunque se puede desarrollar software sin utilizar ningún control de versiones, hacerlo somete al proyecto a un gran riesgo que ningún equipo profesional debería aceptar. Así que la pregunta no es si utilizar el control de versiones, sino qué sistema de control de versiones usar.

• Bibliografía

[https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow- github-flow/](https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-%20%20%20github-flow/)

<https://apptec.cl/blog/conoce-las-diferencias-claves-entre-gitlab-github-y-bitbucket>