Universidad Tecnológica del Centro de Veracruz.

Programa Educativo: Ingeniería en Desarrollo y Gestión de Software.

7°

Ortega Mezhua Pablo Delfino

20183l301012

Desarrollo web integral

ITI.FRANCISCO TRUJILLO ROMERO

LECTURA TÉCNICA

04 de JUNIO de 2022

Introducción

En todo proyecto de desarrollo de software existen cambios, y estos surgen en la medida en que el desarrollo se va llevando a cabo, nos ayuda a organizar y llevar un control referente a un proyecto donde podemos dar seguimiento y gestionar mejor nuestro proyecto donde podemos tener la facilidad de ver las versiones y procedimiento de nuestro proyecto de esta manera con el versiona miento podemos localizar fácilmente si se presenta un error o en algunos casos llevar el control de actividades de cada miembro.

De esta manera usamos el versiona miento como herramienta de prevención y gestión a mi parecer ya que, si llegáramos a tener un margen de error durante el proyecto, podremos visualizar y localizar rápidamente el origen de dicho error.No solo localizar errores si no también definir actividades repartir roles y trabajar un solo proyecto cómodo y de manera remota.

I.- Realizar una breve descripción de las plataformas más utilizadas para el versiona miento:

**GITHUB**



Apareciendo en el año 2008, estamos ante la que fuera primera plataforma con soporte íntegramente para Git, añadiendo la característica de los DCVS (Distributed Control Version Systems, o sistemas de control de versiones distribuidos) que aportaba flexibilidad a las muchas mejoras que había implantado Git en este sector, tentando enormemente a los desarrolladores a pasar su metodología de trabajo a Git y, claro está, a alojar sus proyectos en sus servidores para contar con mayores ventajas.

Hoy día es la plataforma más usada para la desarrollo de proyectos de código abierto (sacando mucha ventaja al resto hay que decir…). También es una plataforma mantenida por la comunidad y destinada principalmente a alojar proyectos open source, no obstante, en su modalidad de pago ofrece la característica de repositorios privados, pudiendo así contar con todas las ventajas de este software y alojamiento sin tener que compartir código alguno con dicha comunidad de desarrolladores.

El plug-in que ofrece para Eclipse es un punto añadido a todas estas ventajas, pudiendo integrar nuestra cuenta GitHub en el framework citado y subir automáticamente todos los cambios realizados. Aunque si algo hay que destacar de GitHub son sus testeos de código (haciendo uso de software como Jenkins), característica que remata una amplia lista de bondades que lo resaltan sobre la competencia, y hace casi invisible el hecho de que únicamente soporte Git y Mercurial.

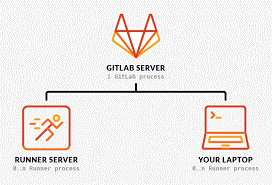
GitHub proporciona también una gran cantidad de características que ayudan a que su elección sea más sencilla como la “bifurcación en un clic” (con la que podremos crear diferentes ramificaciones de nuestro proyecto con tan sólo un clic de ratón), Pull Requests, gestión de incidencias, analizador de código, administración sencilla de comentarios, etc…

**GITLAB**

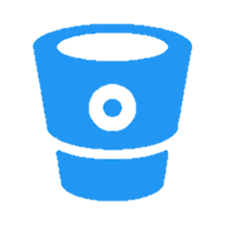


Otra plataforma muy similar a GitHub (prácticamente la toma como base) escrita en Ruby y que cuenta con tres versiones que ofrecer: Community Edition (CE), Enterprise Edition (EE) y su versión de alojamiento en los servidores de la propia GitLab. Y si por ser la última pareciera que es poca cosa, basta nombrar algunas de las entidades que hacen uso de su software (NASA, CERN, Alibaba…) para comprender que no estamos ante un cualquiera.

Sus niveles de permisos, protección de ramificaciones, controles de autenticación, en definitiva la seguridad que ofrece tanto a usuarios como a proyectos hace que destaque sobre sus rivales de sector.



**BITBUCKET**



Y llegamos a BitBucket, un sistema de alojamiento y control de versiones desarrollado por Atlassian y que cuenta con la mayoría de características de GitHub. Con su oferta actual de r epositorios privados ilimitados y un máximo de 5 usuarios presenta hoy día una fuerte competencia a GitHub aunque su uso no esté tan extendido como lo está el sistema antes analizado.

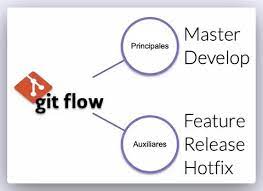
Como características destacadas cuenta con un sistema de soporte a través de notificaciones HipChat, el seguimiento y control de errores con software Confluence y Bamboo, análisis de continuidad e integración de usuarios en diferentes proyectos, etc…

Como decía, el resto de características son muy similares a las ofrecidas por GitHub, incluso en las limitaciones, puesto que comparte el que únicamente sea compatible con Git y Mercurial.



II.- Realizar la descripción de los flujos de trabajo para los controladores de versiones

**GIT FLOW**



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:

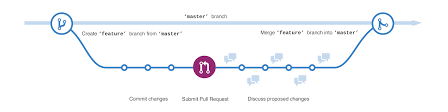
* master: contiene el código de producción.
* develop: contiene el código que ha finalizado desarrollo.

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.

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.



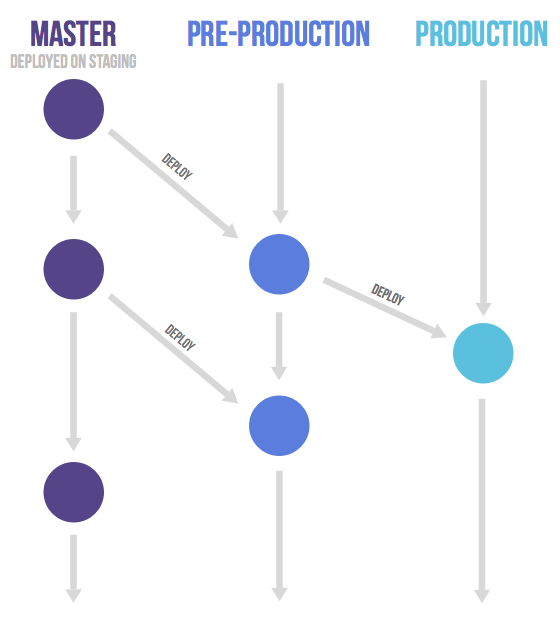
GITLAB FLOW

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.

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.

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 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.

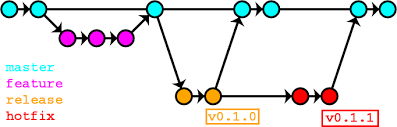


ONE FLOW

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.



**Conclusión**

Para concluir cabe destacar que Utilizar un software de control de versiones es una práctica recomendada para los equipos de software y de DevOps de alto rendimiento

Hoy en día, se puede afirmar que el software de gestión de proyectos es un elemento clave durante el control integral, con eficacia y comodidad. La entidad abarca desde el área de la planificación, desarrollo, producción y relación con los clientes hasta que se lleva a cabo la coordinación de los diferentes proyectos de la organización.Por otro lado, se facilita el registro de diferentes tareas y trabajadores implicados en los mismos dependiendo de las funcionalidades y necesidades del usuario, el control lo lleva a cabo una sola persona, se permite la participación de varias personas o de todo el equipo completo.Su uso nos facilitará el seguimiento y coordinación de trabajos en grupo según todas nuestras necesidades. Los gestores de proyectos, son herramientas muy versátiles y multifuncionales. Se encuentra diseñados para planificar y supervisar tareas, controlar los costos, gestionar los flujos de información, y poder interactuar con todos los sistemas de gestión de las organizaciones.

Bibliografía

<https://www.isotools.org/2018/02/25/necesario-software-la-gestion-proyectos/>

<https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-github-flow/>

<https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>

<https://about.gitlab.com/topics/version-control/what-is-gitlab-flow/>

<https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-github-flow/>