**PERÍODO ACADÉMICO:**2019-B

**ASIGNATURA:** Aplicaciones en Ambientes Libres

**PROFESOR:** Ing. Julián Galindo

**Práctica No.:** 3

**INTEGRANTES:**

* Jefferson Andrango
* Grace Borja
* Jefferson Collantes
* David Cruz
* Jair Quiñónez
* David Yánez

**FECHA:** 2 de diciembre de 2019

**TEMA: USO DE GIT**

**Objetivos:**

-Conocer el funcionamiento de sistema de control de versiones.

-Aprender el flujo de trabajo de git.

-Ejercicio práctico.

**MARCO TEÓRICO**

1. ***Principales sistemas de control de versiones***

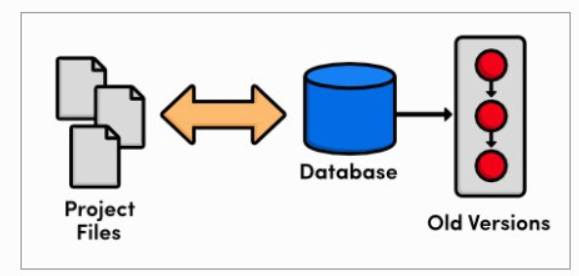
El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo de tal manera que sea posible recuperar versiones especificas más adelante.

Los sistemas de control de versiones han ido evolucionando a lo largo del tiempo y podemos clasificarlos en tres tipos:

**Sistemas de Control de Versiones Locales**

Los sistemas de control de versiones locales como se muestra en la Figura 1 en vez de mantener las versiones como archivos independientes, los almacenaban en una base de datos. Cuando era necesario revisar una versión anterior del proyecto se usaba el sistema de control de versiones en vez de acceder directamente al archivo, de esta manera en cualquier momento solo se tenía una copia del proyecto, eliminando la posibilidad de confundir o eliminar versiones.

En este punto el control de versiones se llevaba a cabo en el computador de cada uno de los desarrolladores y no existía una manera eficiente de compartir el código entre ellos. [1]



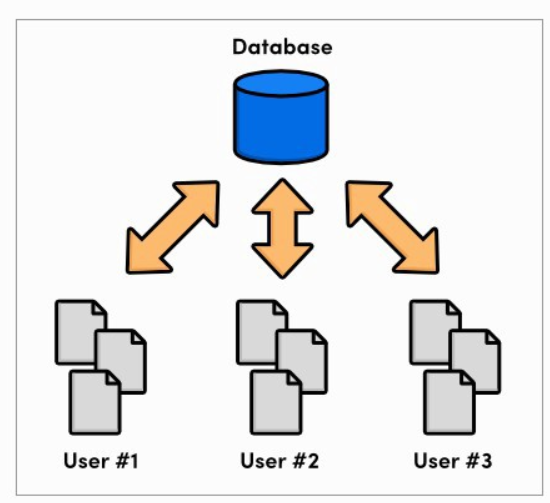
*Figura 1: Sistemas de Control de Versiones Locales*

**Sistemas de Control de Versiones Centralizados**

Para facilitar la colaboración de múltiples desarrolladores en un solo proyecto los sistemas de control de versiones evolucionaron: en vez de almacenar los cambios y versiones en el disco duro de los desarrolladores, estos se almacenaban en un servidor como se muestra en la Figura 2.

Pero existía el problema de que si dos personas editaban el mismo archivo y se presentaba un conflicto alguien debía solucionar este problema de manera manual y el desarrollo no podía continuar hasta que todos los conflictos fueran resueltos y puestos a disposición del resto del equipo.

Esta solución funcionó en proyectos que tenían relativamente pocas actualizaciones y por ende pocos conflictos, pero resulto muy engorroso para proyectos con docenas de contribuyentes activos que realizaban actualizaciones a diario. [1]

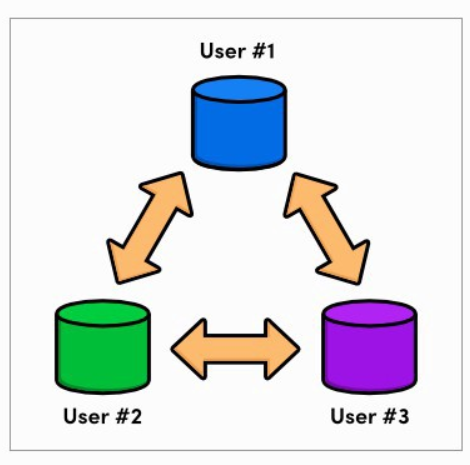


*Figura 2: Sistemas de Control de Versiones Centralizados*

**Sistemas de Control de Versiones Distribuidos**

La siguiente generación de sistemas de control de versiones se alejó de la idea de un solo repositorio centralizado y optó por darle a cada desarrollador una copia local de todo el proyecto como se muestra en la Figura 3, de esta manera se construyó una red distribuida de repositorios, en la que cada desarrollador podía trabajar de manera aislada, pero teniendo un mecanismo de resolución de conflictos mucho más elegante que una su versión anterior.

Al no existir un repositorio central, cada desarrollador puede trabajar a su propio ritmo, almacenar los cambios a nivel local y mezclar los conflictos que se presenten solo cuando se requiera. Cómo cada usuario tiene una copia completa del proyecto el riesgo por una caída del servidor, un repositorio dañado o cualquier otro tipo de perdida de datos es mucho menor que en cualquiera de sus predecesores. [1]



*Figura 3: Sistemas de Control de Versiones Distribuidos*

1. Definición de:
   1. ***Git***

Git, es un software de control de versiones diseñado por Linus Torvalds.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas. [2]

Algunas de las características más importantes de Git son:

* Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
* Gestión distribuida; Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
* Gestión eficiente de proyectos grandes.
* Realmacenamiento periódico en paquetes.
  1. ***Buddy***

Es un software de integración y entrega continúa basado en web y autohospedado para desarrolladores de Git que se puede usar para construir, probar e implementar sitios web y aplicaciones con código de GitHub, Bitbucket y GitLab. Emplea contenedores Docker con lenguajes y marcos preinstalados para compilaciones, junto con DevOps, acciones de monitoreo y notificación. [3]

Algunas de las características que ofrece Buddy son:

* Implementaciones automáticas en push to branch
* Construcciones y pruebas basadas en Docker
* Configuración de 10 minutos del entorno completo
  1. ***Bitbucket***

Bitbucket se sitúa como una de las principales alternativas al conocido y utilizado GitHub. Bitbucket permite a todos los usuarios gratuitos elegir entre utilizar Git o Mercurial como controlador de versiones a la vez que les ofrece una serie de herramientas para el control de errores (como Jira), integración con Google Analytics, HipChat como software de comunicación entre colaboradores y opciones de control de actividad de cada repositorio en cualquier momento para que el desarrollador no tenga que utilizar herramientas externas para esto. [4]

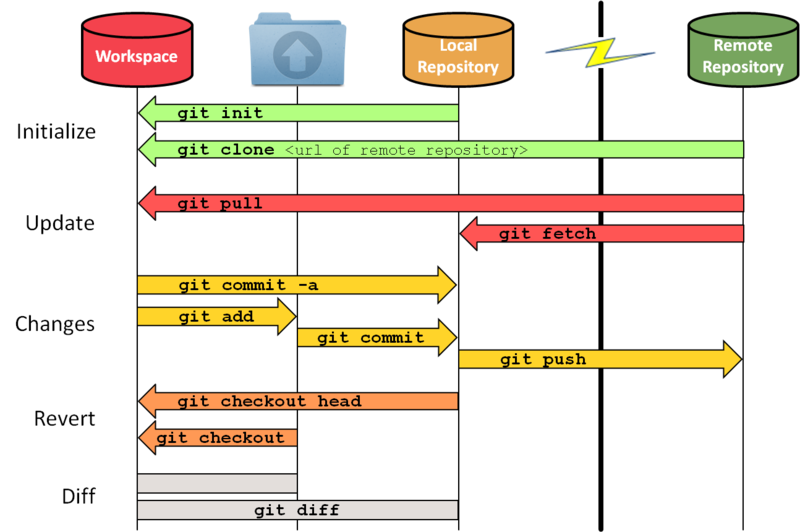
Una de las principales ventajas de Bitbucket de cara a GitHub es que este dispone de un número ilimitado de repositorios privados para los usuarios gratuitos de manera que cualquier desarrollador podrá subir a su plataforma sus códigos, establecerlos como privados y no tener que pagar por ello. A cambio de ello, las versiones gratuitas de Bitbucket limitan a tan solo 5 colaboradores en cada proyecto y los usuarios que necesiten un mayor número de estos tendrán que pasarse a un plan de suscripción. [4]

En GitHub, al contrario, los usuarios disponen de un número ilimitado de colaboradores en su versión gratuita y por lo que pagan los usuarios es por el número de repositorios privados que necesiten.

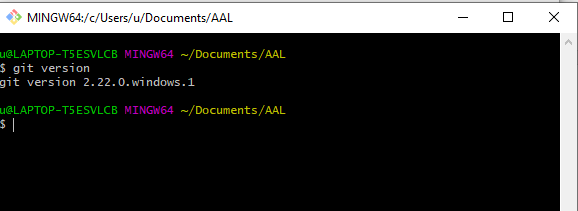
Bitbucket dispone de un cliente oficial para Windows y para Mac, quedando Linux fuera del soporte oficial de esta plataforma. También su plataforma es totalmente accesible desde la web e incluso compatible con el cliente oficial de Git. [4]

* 1. ***Source forge***
  2. ***Gogs***
  3. ***Github***
  4. ***GitLab***

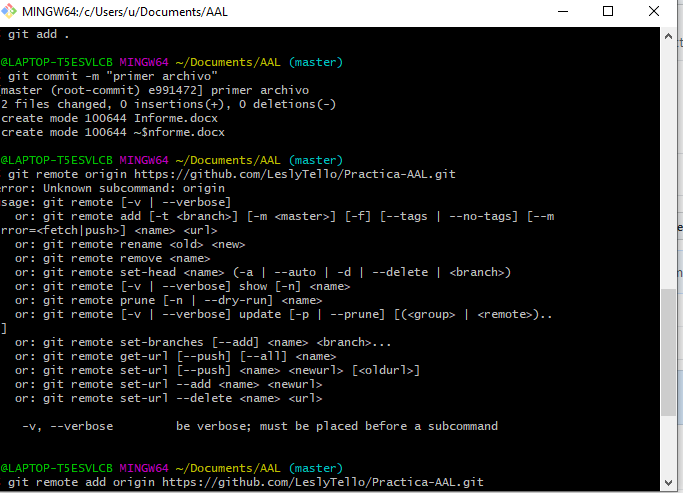
1. Operaciones en un repositorio: commit, branch, pull, push y merge



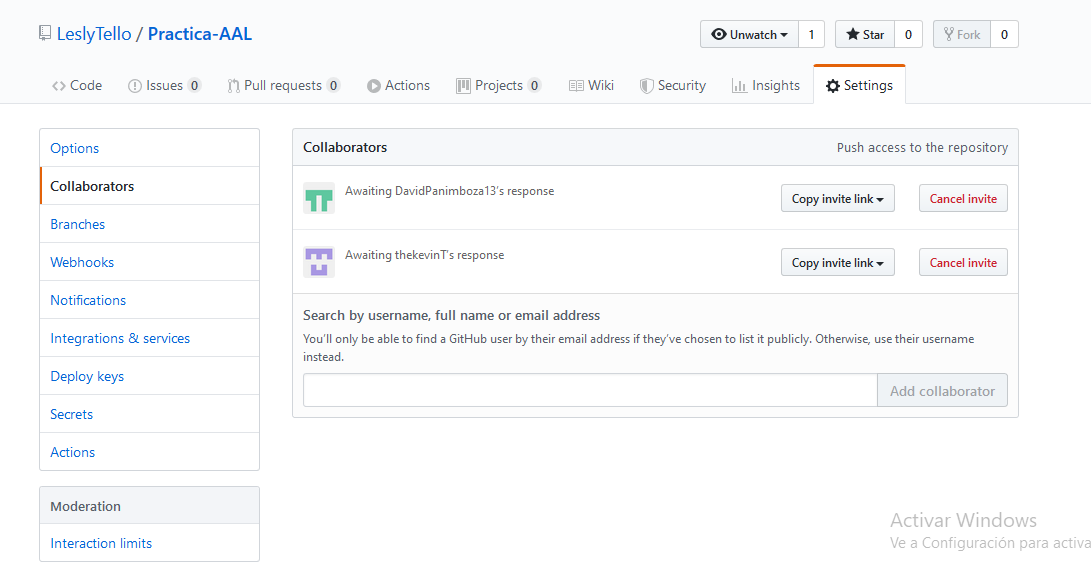
**DESARROLLO DE LA PRÁCTICA**

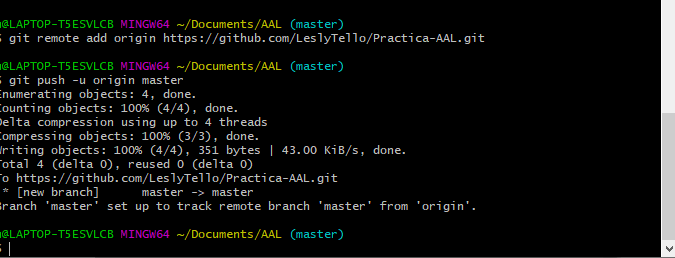
Determinar la versión de Git

Añadir todos los archivos y hacer commit con un mensaje

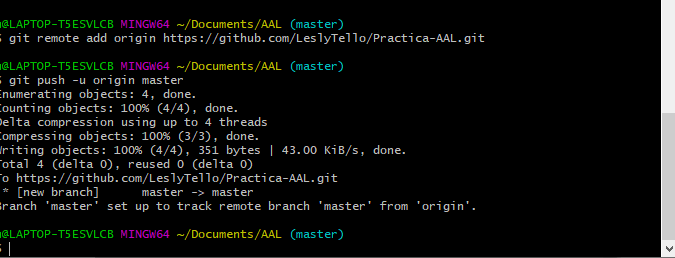


Asociar con la cuenta de github creada





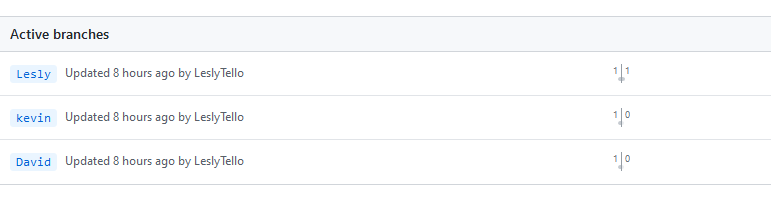
Añadir al repositorio remoto



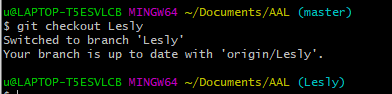
Crear las ramas

Git branch nombreRama

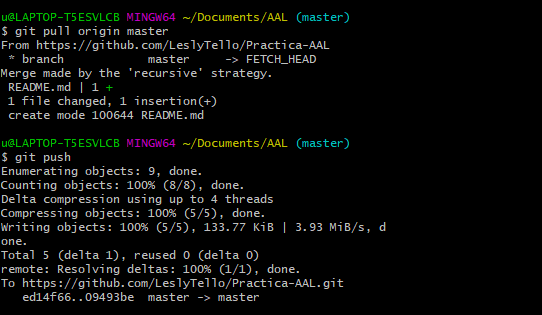
Git push –u origin nombreRama



Para poder cambiar de rama se usa git checkout nombre Rama



Realizar actualizaciones



**CONCLUSIONES**

* Es decisión del usuario decidir qué sistema de control de versiones o la plataforma utilizar, esto dependerá de sus gustos y necesidades teniendo en cuenta si el número de colaboradores será elevado o, por el contrario, se prefiere disponer de repositorios privados en vez de colaboradores de un proyecto.

**BIBLIOGRAFÍA**

[1] Joint, D. (28 enero 2017). “Sistemas de Control de Versiones, qué son y por qué amarlos”. [En línea] Recuperado el: 02 dic 2019. Disponible en: https://medium.com/@jointdeveloper/sistemas-de-control-de-versiones-qué-son-y-por-qué-amarlos-24b6957e716e

[2] Andrés. (16 agosto 2015). “Qué Es Git”. [En línea] Recuperado el: 02 dic 2019. Disponible en: https://codigofacilito.com/articulos/que-es-git

[3] Stackshare. “Buddy vs GitLab: ¿Cuáles son las diferencias?”. [En línea] Recuperado el: 02 dic 2019. Disponible en: https://stackshare.io/stackups/buddy-vs-gitlab

[4] Velasco, R. “Bitbucket, una alternativa al conocido controlador de versiones GitHub” [En línea] Recuperado el: 02 dic 2019. Disponible en: https://www.redeszone.net/2014/05/27/bitbucket-una-alternativa-al-conocido-controlador-de-versiones-github/