

GitHub y Bitbucket

Alan Wiler Zarate Chino
Universidad Católica Boliviana “San Pablo”
La Paz, Bolivia
alan.zarate@ucb.edu.bo
10/09/2022

Resumen— En este documento se muestra la herramienta de control de versiones mas popular para el desarrollo de software. Esta herramienta por si misma, carece de una característica muy importante en la actualidad, el trabajo colaborativo. Para complementar esta funcionalidad se crearon los repositorios remotos, los mas populares son GitHub y Bitbucket que son portales para alojar el código de las aplicaciones de cualquier desarrollador o equipo de desarrolladores.

I. INTRODUCCIÓN

Hoy en día, Git es, con diferencia, el sistema de control de versiones moderno más utilizado del mundo. Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto.

GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones llamado Git. Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real. Es decir, nos permite centralizar el contenido del repositorio para poder colaborar con los otros miembros de nuestra organización.

Bitbucket es un servicio de alojamiento basado en web es similar a GitHub, que utiliza Git. En una entrada de blog del 2008, Bruce Eckel hace una comparación favorablemente de Bitbucket frente a Launchpad, que utiliza Bazaar.

II. GIT

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este

paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones. [1]

Conceptos de Git

Cada vez que se guarda el trabajo, Git crea una confirmación. Una confirmación es una instantánea de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo almacenado anteriormente. Este diseño difiere de otros sistemas que almacenan una versión inicial de un archivo y mantienen un registro de deltas a lo largo del tiempo. [2]



Figura 1. Representación de las confirmaciones de los cambios

Las confirmaciones crean enlaces a otras confirmaciones, formando un gráfico del historial de desarrollo. Puede revertir su código a una confirmación anterior, ver cómo cambiaron los archivos de una confirmación a la siguiente, ver información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en Git mediante un hash criptográfico único del contenido de la confirmación. Todo está cifrado, por lo que no puede realizar cambios, perder información o dañar archivos a menos que Git lo detecte. [2]

Ramas

Cada desarrollador guarda sus cambios en su repositorio de código local. Por lo tanto, se pueden realizar diferentes cambios en función del mismo compromiso. Git proporciona herramientas para aislar cambios y fusionarlos más tarde. Ramas, ligeros indicadores de trabajo en curso, gestionan esta separación. Cuando se completa el trabajo creado en una rama, se puede fusionar con la rama maestra (o rama troncal) del equipo. [2]

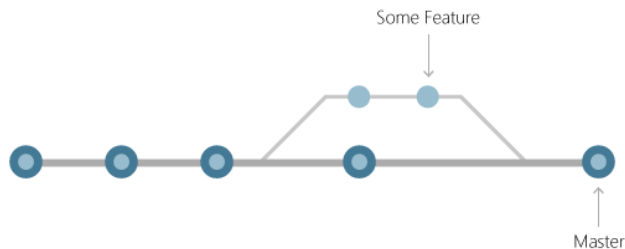


Figura 2. Ramificación del proyecto

Archivos y confirmaciones

Los archivos de Git se encuentran en uno de los tres estados: modificados, almacenados provisionalmente o confirmados. Cuando se modifica un archivo por primera vez, los cambios solo existen en el directorio de trabajo. Todavía no forman parte de una confirmación o del historial de desarrollo. El desarrollador debe almacenar provisionalmente los archivos modificados que se incluirán en la confirmación. El área de almacenamiento provisional contiene todos los cambios que se van a incluir en la siguiente confirmación. Una vez que el desarrollador esté satisfecho con los archivos almacenados provisionalmente, los archivos se empaquetan como confirmación con un mensaje que describe lo que ha cambiado. Esta confirmación forma parte del historial de desarrollo. [2]

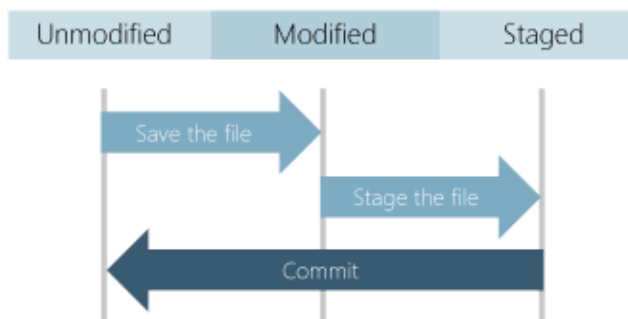


Figura 3. Etapas para confirmar un cambio

El almacenamiento provisional permite a los desarrolladores elegir qué cambios de archivo guardar en una confirmación para desglosar los cambios grandes en una serie de confirmaciones más pequeñas. Al reducir el ámbito de las confirmaciones, es más fácil revisar el historial de confirmaciones para buscar cambios de archivo específicos. [2]

III. INSTALAR GIT

Para instalar git, debe descargar el instalador de la web:
<https://git-scm.com/downloads>

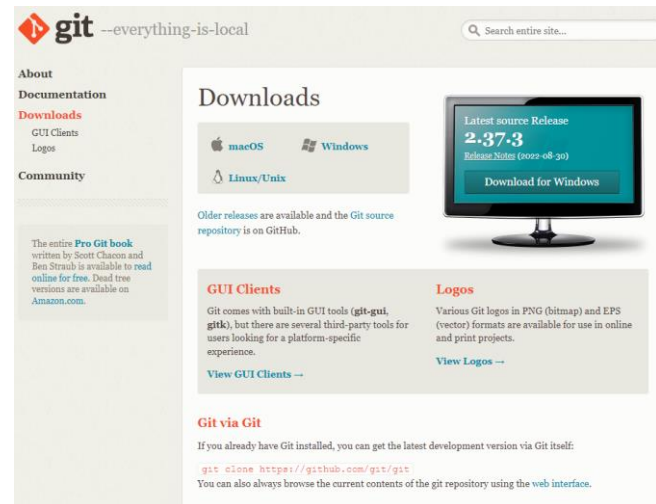


Figura 4. Sitio oficial de git

Una vez descargado el instalador para el sistema operativo se debe dejar las opciones por defecto de todas las características excepto la opción para definir el editor que usara git por defecto. En el caso de Windows se considera que el Notepad sea la mejor opción. En el caso de linux, Vim.

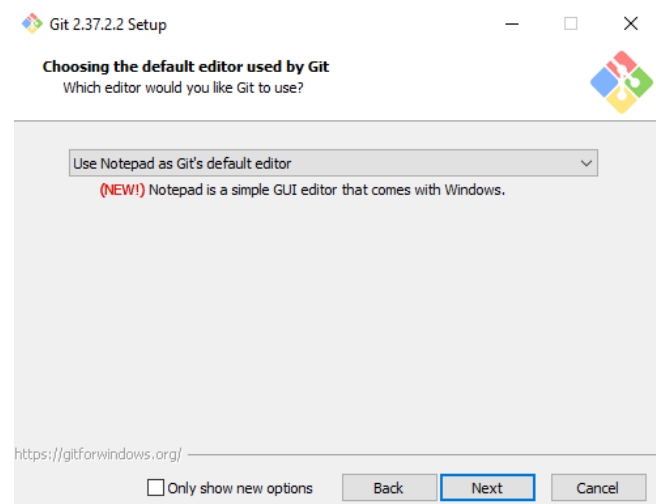


Figura 5. Instalación de git

Comandos Git

git init creará un nuevo repositorio local GIT. El siguiente comando de Git creará un repositorio en el directorio actual:

```
git init
```

Como alternativa, puedes crear un repositorio dentro de un nuevo directorio especificando el nombre del proyecto:

```
git init [nombre del proyecto]
```

git clone se usa para copiar un repositorio. Si el repositorio está en un servidor remoto, usa:

```
git clone nombredeusuario@host:/path/to/repository
```

A la inversa, ejecuta el siguiente comando básico para copiar un repositorio local:

```
git clone /path/to/repository
```

git add se usa para agregar archivos al área de preparación. Por ejemplo, el siguiente comando de Git básico indexará el archivo temp.txt:

```
git add <temp.txt>
```

git commit creará una instantánea de los cambios y la guardará en el directorio git.

```
git commit -m “El mensaje que acompaña al commit va aquí”
```

git config puede ser usado para establecer una configuración específica de usuario, como el email, nombre de usuario y tipo de formato, etc. Por ejemplo, el siguiente comando se usa para establecer un email:

```
git config --global user.email tuemail@ejemplo.com
```

La opción -global le dice a GIT que vas a usar ese correo electrónico para todos los repositorios locales. Si quieres utilizar diferentes correos electrónicos para diferentes repositorios, usa el siguiente comando:

```
git config --local user.email tuemail@ejemplo.com
```

git status muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser preparados o confirmados.

```
git status
```

git push se usa para enviar confirmaciones locales a la rama maestra del repositorio remoto. Aquí está la estructura básica del código:

```
git push origin <master>
```

git checkout crea ramas y te ayuda a navegar entre ellas. Por ejemplo, el siguiente comando crea una nueva y automáticamente se cambia a ella:

```
command git checkout -b <branch-name>
```

Para cambiar de una rama a otra, sólo usa:

```
git checkout <branch-name>
```

git remote te permite ver todos los repositorios remotos. El siguiente comando listará todas las conexiones junto con sus URLs:

```
git remote -v
```

Para conectar el repositorio local a un servidor remoto, usa este comando:

```
git remote add origin <host-or-remoteURL>
```

Por otro lado, el siguiente comando borrará una conexión a un repositorio remoto especificado:

```
git remote <nombre-del-repositorio>
```

git branch se usa para listar, crear o borrar ramas. Por ejemplo, si quieres listar todas las ramas presentes en el repositorio, el comando debería verse así:

```
git branch
```

Si quieres borrar una rama, usa:

```
git branch -d <branch-name>
```

git pull fusiona todos los cambios que se han hecho en el repositorio remoto con el directorio de trabajo local.

```
git pull
```

git merge se usa para fusionar una rama con otra rama activa:

```
git merge <branch-name>
```

git diff se usa para hacer una lista de conflictos. Para poder ver conflictos con respecto al archivo base, usa:

```
git diff --base <file-name>
```

El siguiente comando se usa para ver los conflictos que hay entre ramas antes de fusionarlas:

```
git diff <source-branch> <target-branch>
```

Para ver una lista de todos los conflictos presentes usa:

```
git diff
```

git tag marca commits específicos. Los desarrolladores lo usan para marcar puntos de lanzamiento como v1.0 y v2.0.

```
git tag 1.1.0 <instert-commitID-here>
```

git log se usa para ver el historial del repositorio listando ciertos detalles de la confirmación. Al ejecutar el comando se obtiene una salida como ésta:

```
$root:~ git log
commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw
Author: Alex Hunter <alexh@gmail.com>
Date: Mon Oct 1 12:56:29 2016 -0600
```

git reset sirve para resetear el index y el directorio de trabajo al último estado de confirmación.

```
git reset - -hard HEAD
```

git rm se puede usar para remover archivos del index y del directorio de trabajo.

```
git rm filename.txt
```

git stash guardará momentáneamente los cambios que no están listos para ser confirmados. De esta manera, puedes volver al proyecto más tarde.

```
git stash
```

git show se usa para mostrar información sobre cualquier objeto git.

```
git show
```

git fetch le permite al usuario buscar todos los objetos de un repositorio remoto que actualmente no se encuentran en el directorio de trabajo local.

```
git fetch origin
```

git ls-tree te permite ver un objeto de árbol junto con el nombre y modo de cada ítem, y el valor blob de SHA-1. Si quieres ver el HEAD, usa:

```
git ls-tree HEAD
```

git cat-file se usa para ver la información de tipo y tamaño de un objeto del repositorio. Usa la opción -p junto con el valor SHA-1 del objeto para ver la información de un objeto específico, por ejemplo:

```
git cat-file -p
d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

git grep le permite al usuario buscar frases y palabras específicas en los árboles de confirmación, el directorio de

trabajo y en el área de preparación. Para buscar por www.yyy.com en todos los archivos, usa:

```
git grep www.yyy.com
```

gitk muestra la interfaz gráfica para un repositorio local. Simplemente ejecuta:

```
gitk
```

git instaweb te permite explorar tu repositorio local en la interfaz GitWeb. Por ejemplo:

```
git instaweb --http=webrick
```

git gc limpiará archivos innecesarios y optimizará el repositorio local.

```
git gc
```

git archive le permite al usuario crear archivos zip o tar que contengan los constituyentes de un solo árbol de repositorio. Por ejemplo:

```
git archive - -format=tar master
```

git prune elimina los objetos que no tengan ningún apuntador entrante.

```
git prune
```

git fsck realiza una comprobación de integridad del sistema de archivos git e identifica cualquier objeto corrupto

```
git fsck
```

git rebase se usa para aplicar ciertos cambios de una rama en otra. Por ejemplo:

```
git rebase master
```

IV. GITLAB

GitLab es una herramienta de ciclo de vida y repositorio de Git. Es un completo DevOps plataforma, que permite a los profesionales gestionar y realizar diversas tareas del proyecto. Las tareas incluyen la planificación del proyecto, la gestión del código fuente, el mantenimiento de la seguridad y el seguimiento. [3]

GitLab fue creado en 2011 por desarrolladores ucranianos llamados Dmitry Zaporozhets & Valery Sizov. Actualmente tiene más de 100k usuarios y grandes

empresas como la NASA, IBM, Sony lo utilizan. Es bueno conocer algunas de las siguientes características. [4]

- Pipelines CI / CD potentes y bien definidos
- Contenedores Docker
- Registro incorporado implementado instantáneamente sin configuración.
- Admite servidores de terceros para administrar imágenes de Docker
- Seguimiento de problemas
- Escrito en Ruby and Go

Beneficios de GitLab

- GitLab es fácil de configurar y administrar con sus códigos disponibles gratuitamente.
- La aplicación única es única y crea un flujo de trabajo optimizado con colaboración y eficiencia.
- Las revisiones de código junto con las solicitudes de extracción son fáciles de usar y compactas.
- Al ser una aplicación nativa de la nube con fuertes medidas de seguridad, ofrece características de seguridad como restricciones granulares, autenticación de usuario con Kerberos.
- Integración mínima para reducir el ciclo de vida del desarrollo mientras aumenta la productividad
- Facilita una organización adecuada de Kubernetes y la integración
- Permite una gestión de proyectos amplia y adaptable para acelerar el flujo de trabajo

Diferencias de GitHub

Acciones de GitLab CI / CD y GitHub

Integración continua/ Continuous Delivery (CI / CD) ahorra enormemente a los desarrolladores y proporciona un excelente control de calidad. GitLab ofrece CI / CD gratis , lo que elimina la necesidad de utilizar cualquier solución de CI externa. Y si aún desea utilizar un CI de terceros, puede integrarlo fácilmente con Codeship, Jenkins, etc.

Aparte de eso, ofrece un panel de operaciones conciso para ayudarlo a conocer las dependencias asociadas con sus DevOps y sus esfuerzos de desarrollo. Por lo tanto, GitLab está abordando el problema de los desarrolladores mejor que GitHub en este sentido.

Por otro lado, GitHub lanzó 'Acciones' en 2019 para automatizar su workflows. Pero no tiene una plataforma de implementación, lo que significa que necesita una integración adicional como Heroku o Digital Ocean.

Niveles de autenticación

Con GitLab, tiene la opción de modificar y establecer permisos en función de sus roles. Pero en GitHub, solo puede decidir a quién otorgar acceso de lectura / escritura a su repositorio.

Por ejemplo, puede establecer permiso para que los miembros de su equipo accedan a los rastreadores de problemas sin permitirles acceder al código fuente. Es útil para equipos grandes que involucran a colaboradores basados en roles.

Importar y exportar

GitLab proporciona documentación detallada sobre cómo puede importar datos de diferentes proveedores como Bitbucket, GitHub, etc. Sin embargo, GitHub carece de proporcionar dicha documentación en detalle para la mayoría de los repositorios de git. Pero puede usar el Importador de GitHub para sus códigos en Mercurial, TFS, Subversion, etc.

GitLab le permite exportar cargas de proyectos, wiki, repositorios de proyectos, configuraciones como servicios y webhooks, fragmentos, etiquetas, hitos y más para la exportación de datos.

Pero GitHub es un poco más restrictivo en términos de funciones de exportación para sus repositorios.

V. GITHUB

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo. [3]

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar. [3]

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario. [4]

Así pues, Github es un portal para gestionar las aplicaciones que utilizan el sistema Git. Además de permitirte mirar el

código y descargarte las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación. [4]

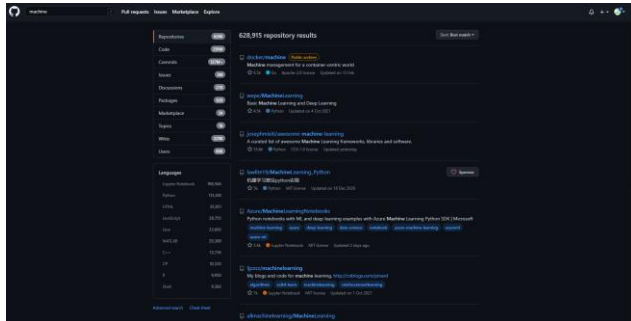
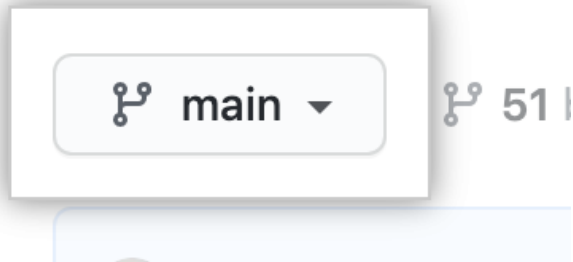


Figura 6. Plataforma oficial de GitHub

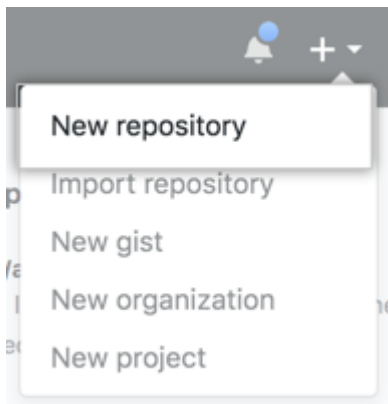
VI. USO DE GITHUB

Para este tutorial no es necesario utilizar la línea de comandos.

Paso 1. Crear un repositorio



- En la esquina superior derecha de cualquier página, utiliza el menú desplegable y selecciona New repository (Nuevo repositorio).



- En el cuadro Nombre del repositorio, escriba hello-world.

- En el cuadro Descripción, escriba una breve descripción.
- Seleccione Agregar un archivo LÉAME.
- Seleccione si el repositorio será Público o Privado.
- Haga clic en Create repository (Crear repositorio).

Owner ⁺ Repository name ⁺

octocat / hello-world ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-system](#)?

Description (optional)

My first repository

☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☒ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

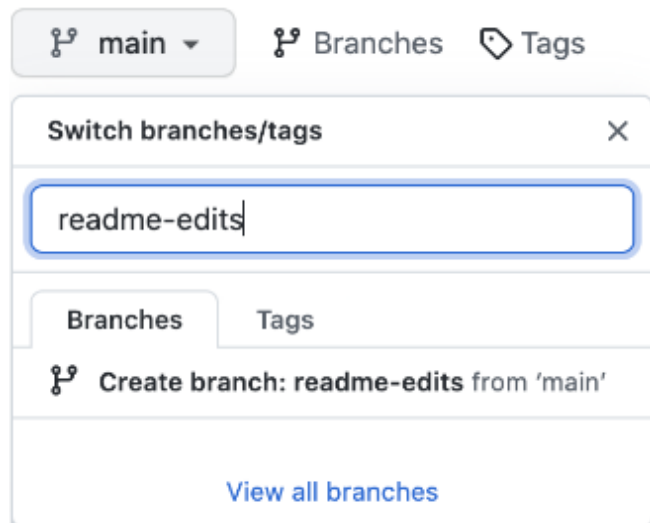
☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

Paso 2. Cómo crear una rama

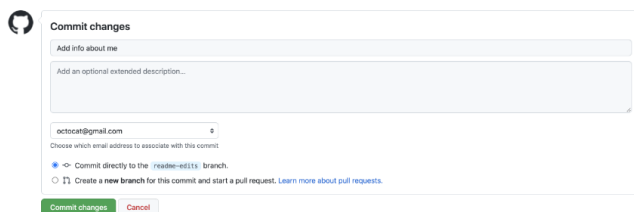
- Haga clic en la pestaña Código del repositorio hello-world.
- Haga clic en la lista desplegable de la parte superior de la lista de archivos que indica main.
- Escriba un nombre de rama, readme-edits, en el cuadro de texto.
- Haga clic en Crear rama: readme-edits desde main.



Ahora se tiene dos ramas, main y readme-edits.

Paso 3. Hacer y confirmar cambios

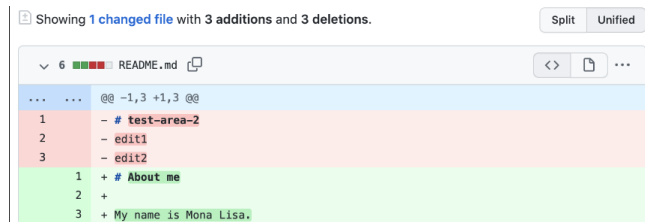
- En la rama readme-edits que ha creado, haga clic en el archivo README.md.
- Haz clic en el icono de lápiz para editar el archivo.
- En el editor, escribe un poco sobre ti mismo. Intenta utilizar elementos de lenguaje de marcado diferentes.
- En el cuadro Confirmar cambios, escriba un mensaje de confirmación que describa los cambios.
- Haga clic en Confirmar cambios.



Paso 4. Abrir una solicitud de extracción

- Haga clic en la pestaña Solicitudes de incorporación de cambios del repositorio hello-world.
- Haga clic en Nueva solicitud de incorporación de cambios.
- En el cuadro Comparaciones de ejemplo, seleccione la rama que ha creado, readme-edits, para compararla con main (la original).

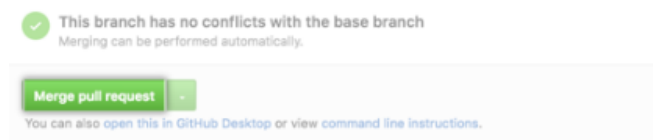
- Mira tus cambios en los diffs en la página de Comparar, asegúrate que son lo que quieres enviar.



- Haga clic en Create pull request (Crear solicitud de incorporación de cambios).
- Dale un título a tu solicitud de cambios y escribe una descripción breve de estos. Puedes incluir emojis y arrastrar y soltar imágenes y gifs.
- Opcionalmente, a la derecha del título y la descripción, haga clic en el icono de engranaje junto a Revisores. Usuarios asignados, Etiquetas, Proyectos o Hito para agregar cualquiera de estas opciones a la solicitud de incorporación de cambios. No necesitas agregar ninguna aún, pero estas opciones ofrecen diversas formas de colaborar utilizando solicitudes de cambio. Para más información, vea "Acerca de las solicitudes de incorporación de cambios".
- Haga clic en Create pull request (Crear solicitud de incorporación de cambios).

Paso 5. Fusionar tu solicitud de cambios

- Haga clic en Combinar solicitud de incorporación de cambios para combinar los cambios en main.



- Haga clic en Confirmar combinación. Recibirás un mensaje indicando que la solicitud se fusionó con éxito y luego se cerró.
- Haga clic en Eliminar rama. Ahora que la solicitud de incorporación de cambios se ha combinado y los cambios están en main, puede eliminar la rama readme-edits de forma segura. Si quieres hacer más cambios a tu proyecto, siempre puedes crear una rama nueva y repetir este proceso.

VII. BITBUCKET

Bitbucket Cloud es una herramienta de alojamiento de código y colaboración basada en Git diseñada para equipos. Las inmejorables integraciones de Bitbucket con Jira y Trello están concebidas para unir a todo el equipo de software con el fin de poner en práctica un proyecto. Ofrecemos un lugar en el que tu equipo pueda colaborar con código desde el concepto hasta la nube, crear código de calidad mediante pruebas automatizadas e implementar código con total seguridad. [5]

Bitbucket es la herramienta ideal para compartir datos, implementar y crear código, así como automatizar pruebas. Se puede acceder a Bitbucket Cloud mediante una URL o también permite el alojamiento local, utilizando infraestructuras on-premises. Bitbucket Server, por otro lado, es compatible con Bamboo, automatizando procesos sin límite de licencia. [5]

Para conocer una solución corporativa alojada en varios servidores agrupados, Bitbucket Data Center es lo ideal. Permite un rendimiento escalable, duplicaciones inteligentes y alta disponibilidad. [5]

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Mercurial y Git. Bitbucket ofrece planes comerciales y gratuitos. Se ofrece cuentas gratuitas con un número ilimitado de repositorios privados (que puede tener hasta cinco usuarios en el caso de cuentas gratuitas) desde septiembre de 2010, los repositorios privados no se muestran en las páginas de perfil - si un usuario sólo tiene depósitos privados, el sitio web dará el mensaje "Este usuario no tiene repositorios". El servicio está escrito en Python. [6]

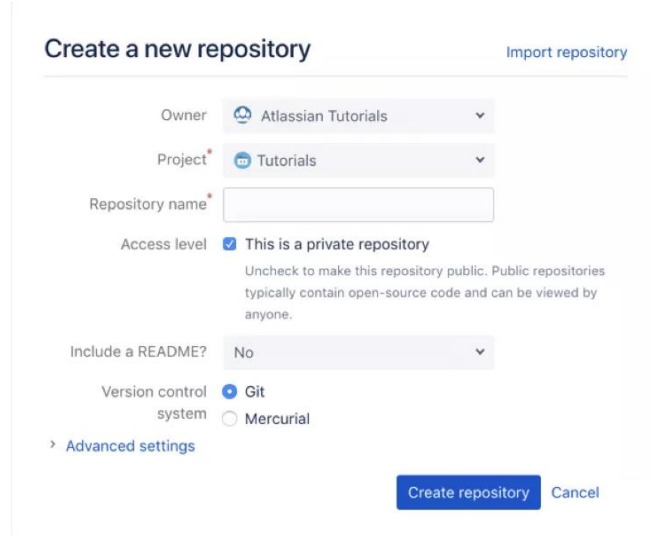
Es similar a GitHub, que utiliza Git. En una entrada de blog del 2008,3 Bruce Eckel hace una comparación favorablemente de Bitbucket frente a Launchpad, que utiliza Bazaar. [7]

VIII. USO DE BITBUCKET

Paso 1: Agregar el código.

Para comenzar, debes crear un repositorio que es donde se almacena el código en Bitbucket.

- En la barra lateral izquierda, haz clic en + y “Crear nuevo”, seleccionando “Repositorio”.
- Escribe el nombre del repositorio que se incluirá en la URL.
- Incluye un archivo README para orientar a los nuevos usuarios.
- Clic en “Crear”.

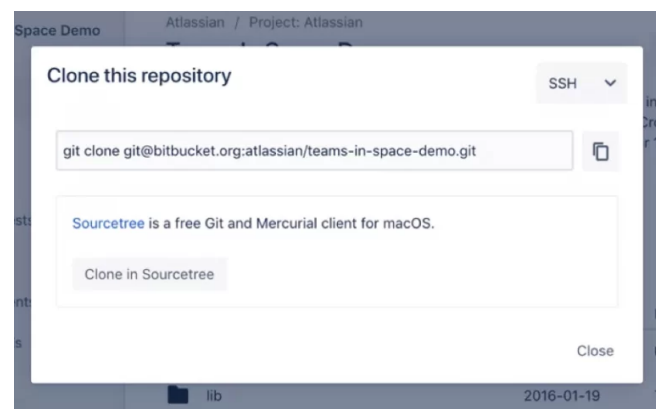


Paso 2. Configurar la cuenta local

Para contribuir o cambiar el código después de haber configurado Git y con tu repositorio en cloud, debes configurar el sistema local.

Después de eso, es necesario clonar el repositorio en el sistema local. Verifica:

- En la barra lateral izquierda, haz clic en +, luego en “Empezar a trabajar” y “Clonar este repositorio”.
- En la esquina superior derecha, si aún no has configurado las claves SSH, selecciona HTTPS.
- Copia el comando clonación.



Configuración de CLI

- Busca el directorio que quieres clonar.
- Toma el comando que copiaste de Bitbucket y pégalo: `git clone https://username@bitbucket.org/teamsinspace/document-ation-test.git`.

Un nuevo subdirectorio con el mismo nombre del repositorio clonado aparecerá en tu disco local. Si se ha clonado en un repositorio vacío, es posible que no haya nada en ese directorio.

Como desarrollador, puedes trabajar en tu código localmente y guardar los cambios al terminar. Después de eso, agrega, confirma y envía a Bitbucket en la interfaz de esta línea de comando. Si estás trabajando con alguien en este proyecto y realiza algún cambio importante en el archivo enviado, puedes verificarlo en tiempo real, simplemente sigue los pasos a continuación para volver a editar:

- Haz clic en el nombre del archivo en el repositorio.
- Luego “Editar”.
- Realiza el cambio deseado.
- Para finalizar haz clic en “Confirmar”.

IX. CONCLUSIONES

La práctica de versionar el código fuente puede abaratar los costos de desarrollo y mantención del software, principalmente reduciendo la ineficiencia que se genera al no tener control sobre cuándo y cómo se realizan los cambios en este.

Cada desarrollador podría hacer sus cambios en varias partes del árbol de archivos. El control de versiones ayuda a los equipos a resolver este tipo de problemas al realizar un seguimiento de todos los cambios individuales de cada colaborador y al contribuir a evitar que el trabajo concurrente entre en conflicto.

Git permite también la gestión distribuida del trabajo. Es decir, trabajar en equipo sin tener que estar físicamente cerca. Cada programador cuenta con su propia copia del repositorio en su equipo. Y cada vez que se realice algún cambio, la modificación se aplica al proyecto en desarrollo.

En ese sentido, los repositorios remotos como GitHub y BitBucket son de gran importancia para complementar la funcionalidad de Git.

X. BIBLIOGRAFIA

- [1] Microsoft (2022, Julio 22) “What is git ?” docs.microsoft.com [Online] disponible: <https://docs.microsoft.com/es-es/devops/develop/git/what-is-git> (accedido el 9 de septiembre de 2022)
- [2] Perveez, S.H. (2022, Julio 12) “What is Git: Features, Command and Workflow in Git” simplilearn.com [Online] disponible: <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git> (accedido el 9 de septiembre de 2022)
- [3] Gaba, I. (2022, Julio 29) “What is GitHub and how to use it ?” simplilearn.com [Online] disponible: <https://www.simplilearn.com/tutorials/git-tutorial/what-is-github> (accedido el 9 de septiembre de 2022)
- [4] Courtemanche, M. (2021, Marzo 1) “Definition GITHUB” techtarget.com [Online] disponible: <https://www.techtarget.com/searchitoperations/definition/GitHub> (accedido el 9 de septiembre de 2022)
- [5] BitBucket “Why bitBucket?” bitbucket.org [Online] disponible: <https://bitbucket.org> (accedido el 9 de septiembre de 2022)
- [6] Kumar, R. (2022, Marzo 15) “What is BitBucket and How it Works?” devopsschool.com [Online] disponible: <https://www.devopsschool.com/blog/what-is-bitbucket-and-how-it-works-an-overview-and-its-use-cases/> (accedido el 9 de septiembre de 2022)
- [7] Tungal, A. T. (2022, Junio 17) “Bitbucket vs Github” upguard.com [Online] disponible: <https://www.upguard.com/blog/bitbucket-vs-github#:~:text=If%20you%20boil%20it%20down,mostly%20enterprise%20and%20business%20users.> (accedido el 9 de septiembre de 2022)