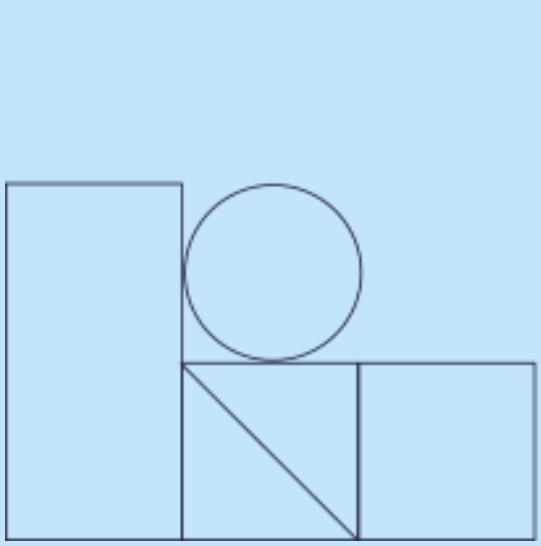
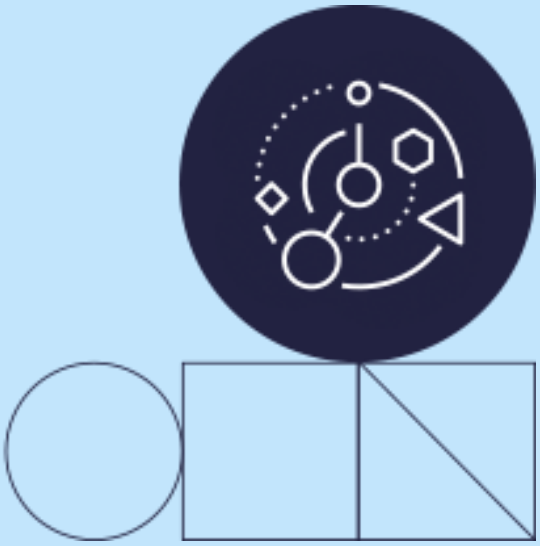
IBM SkillsBuild

IBM

Git y Github

Colaboración en Github





Índice

¿Qué es la colaboración en Git? 3
Paso 1: crear un repositorio 3
Paso 2: agreguar archivos a nuestro proyecto. 3
Paso 3: Agregar a nuestros colaboradores. 3

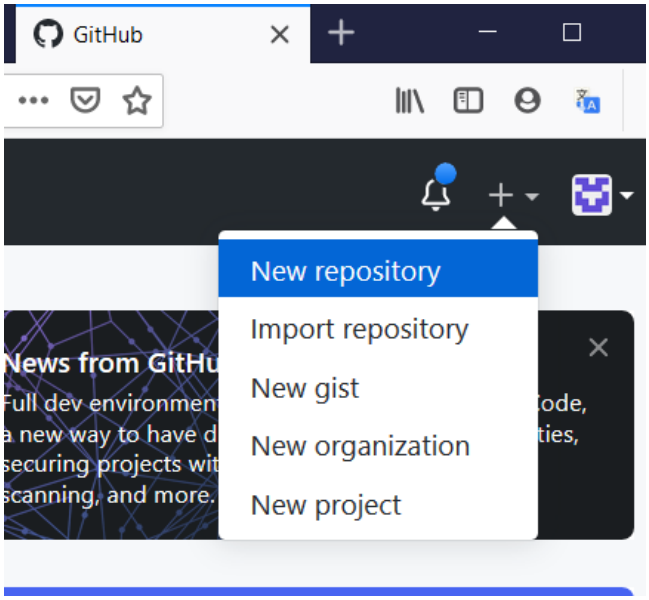
Crear etiquetas firmadas y sin firmar 4
git checkout 4
Otras terminologías relacionadas 4
push 4

¿Qué es la colaboración en Git?

La colaboración es la forma en que diferentes personas pueden trabajar juntas en el mismo proyecto. Es como crear un grupo en *GitHub* al igual que los Grupos en otras redes sociales. Las personas agregadas a la lista de colaboradores pueden hacer push, *merge* y hacer otros tipos de cosas similares en el proyecto.

Para colaborar, debemos seguir los pasos que se indican a continuación:

Paso 1: Crear un repositorio.



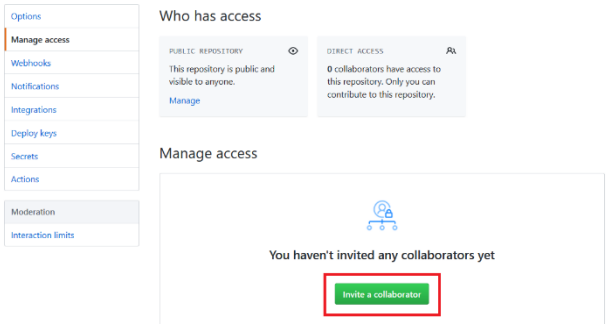
Paso 2: Agregar archivos a nuestro proyecto.

Abriremos GitBash en nuestro directorio de trabajo local donde se guardan los archivos y ejecutaremos los siguientes comandos:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git init

$ git add

$ git commit -m "initial commit"

$ git remote add origin remote
repository URL

$ git remote -v
```

Paso 3: Agregar a nuestros colaboradores.

Haremos clic en configuración y luego seguiremos los siguientes los pasos:



Después de hacer clic en "Invitar a un colaborador", completaremos los detalles requeridos.

Nota: Para trabajar perfectamente en un equipo colaborativo es necesario conocer a continuación las terminologías y sus aplicaciones.

Crear etiquetas firmadas y sin firmar

Primero, surge la pregunta ¿por qué firmamos un código? La razón es que muestra la autoridad del código, quién lo ha escrito y quién debe ser culpado si surge algún error. Para crear una etiqueta simple, escribiremos:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git tag ejemplo
```

Esto crea una etiqueta basada en la versión actual del repositorio.
Para crear una etiqueta anotada sin firmar que contenga un mensaje, simplemente escribiremos:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git tag -a v1 -m “Versión 1”
```

Por último, si deseamos firmar una etiqueta, simplemente escribiremos:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git tag -s Mytag
```

Para verificar su etiqueta firmada, simplemente escribiremos:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git tag -v mytag
```

git checkout

El comando *git checkout* nos permite navegar entre las ramas creadas por la rama *git*. Nos permite verificar archivos en el directorio de trabajo para que coincidan con la versión almacenada en esa rama.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git checkout branch name
```

Este comando se puede confundir fácilmente con *git clone*, pero ambos son diferentes. La diferencia entre los dos comandos es que la clonación funciona para recuperar el código de un repositorio remoto, alternativamente, el *checkout* funciona para cambiar entre versiones de código que ya están en el sistema local.

Otras terminologías relacionadas

push

Este comando se usa para enviar los archivos desde nuestra máquina local al repositorio de GitHub.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git push -f origin master
```

fetch

Es útil cuando interactuamos con un repositorio remoto. Básicamente, este comando se usa para recuperar el trabajo realizado por otras personas y actualizar nuestro proyecto local en consecuencia.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git fetch nombre_remoto
```

pull

Es un método para enviar nuestras contribuciones al proyecto. Ocurre cuando un desarrollador solicita que los cambios comprometidos en un repositorio externo se consideren para su inclusión en el repositorio principal de un proyecto después de la revisión por pares.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git pull nombre_remoto
```

Obtener la copia del control remoto especificado de la rama actual y combinarla inmediatamente en la copia local. Este comando es el mismo que *git fetch* seguido de *git merge*.

Eliminación de cualquier etiqueta

Para eliminar cualquier etiqueta, navegaremos a nuestro repositorio local de GitHub y escribiremos el siguiente comando:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git tag -d tagName

$ git push origin :tagName
```

Buscando código

GitHub ofrece a los usuarios la función de buscar código con un repositorio u organización. También hay algunas restricciones impuestas por *GitHub* al buscar el código. Estas restricciones se imponen debido a la complejidad del código de búsqueda. A continuación, se muestran las búsquedas válidas que puede realizar en *GitHub*.

- Buscar por el contenido del archivo o la ruta del archivo
- Buscar en los repositorios de un usuario o de una organización
- Buscar por ubicación de archivo o tamaño de archivo
- Buscar por idioma
- Buscar por nombre de archivo o extensión de archivo

Verificar los registros anteriores

Para mirar hacia atrás y consultar cuáles son todas los *commits* que se realizaron en el repositorio, simplemente podemos usar:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git log
```

De forma predeterminada, si no se pasa ningún argumento con este comando. Mostrará los registros en orden cronológico inverso.