El Manual Esencial de Git

Fernando Cardellino

Introducción

Hola! Soy Sanjula, y en esta guía espero poder enseñarte un poco acerca de Git incluyendo:

Qué es Git

Porqué aprender Git

Establecer variables de configuración

Introducción al comando help en Git

Cómo convertir un proyecto existente en un repositorio de Git local

Cosas que hacer antes del primer commit

Cómo agregar archivos al área de preparación (staging area)

Cómo eliminar archivos del área de preparación

Realizar tu primer commit

Cómo clonar un repositorio remoto

Ver información sobre el respositorio remoto

Cómo enviar (push) tus cambios al repositorio remoto

Como crear una rama (branch) para una prestación o problema específico

Enviar la rama al repositorio remoto luego de ejecutar el comando commit

Cómo fusionar (merge) una rama

Cómo eliminar una rama

¡Empecemos!

¿Qué es Git?

En términos sencillos, Git es un **sistema de control de versiones distribuido de código abierto**.

Los sistemas de control de versiones ayudan a cualquier equipo de software a gestionar cambios en el código fuente de un producto o servicio a lo largo del tiempo. Realiza un seguimiento de todas las modificaciones al código fuente en una base de datos. Si se ha cometido un error crítico en el código fuente, los desarrolladores de un equipo de software pueden retrotraer el código fuente a una versión antes de que se realizara el cambio erróneo. Como resultado, los sistemas de control de versiones protegen el código fuente de desastres, errores humanos y consecuencias no deseadas (cuando una corrección de errores rompe otra parte de la aplicación, por ejemplo).

Entonces, ¿Porqué aprender Git?

Git es el sistema de control de versiones más utilizado en el mundo actualmente. Es un proyecto de código abierto maduro y mantenido activamente, desarrollado originalmente por Linus Torvalds.

Una cantidad asombrosa de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto, especialmente utilizando el servicio de alojamiento de repositorios de git, GitHub, que ahora es propiedad de Microsoft. De ahí la importancia de aprender Git.

Prerrequisitos para esta guía

Descarga e instala git desde aquí

Verifica la versión de git

```
git --version
```

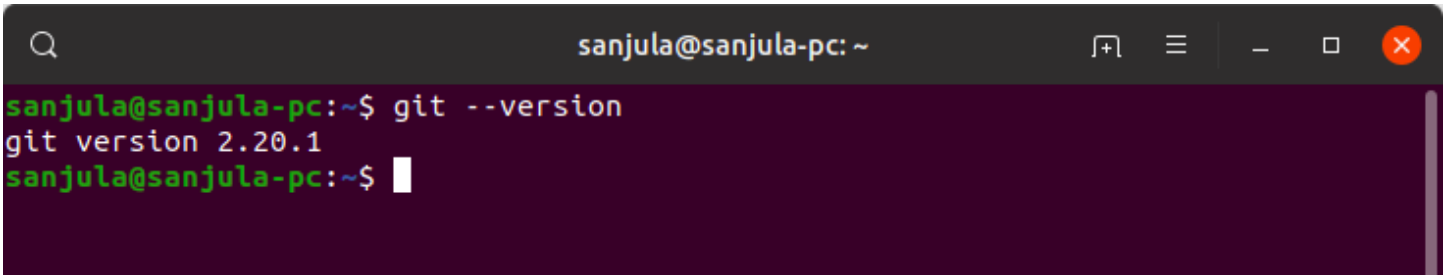

Figure-2: Git version

Si el número de versión es devuelto, entonces significa que git ha sido instalado exitosamente en tu computador.

Estableciendo los valores de configuración

Ahora debemos establecer las variables de configuración global, que son muy importantes, especialmente si estás trabajando con otros desarrolladores. La principal ventaja de esto es que es más fácil averiguar quién ha hecho un commit de determinado bloque de código, por ejemplo.

```
git config --global user.name “Sanjula Madurapperuma”
```

```
git config --global user.email “sanjula@mail.com”
```

```
git config --list
```

Comando help

Como puedes notar, *config* es un verbo que se ha usado con frecuencia hasta ahora en este manual y los verbos también se pueden usar como prefijo o sufijo con el comando help. Podemos usar el mismo ejemplo (el verbo *config*) de arriba para explicar estos comandos.

```
git help config
```

```
git config --help
```

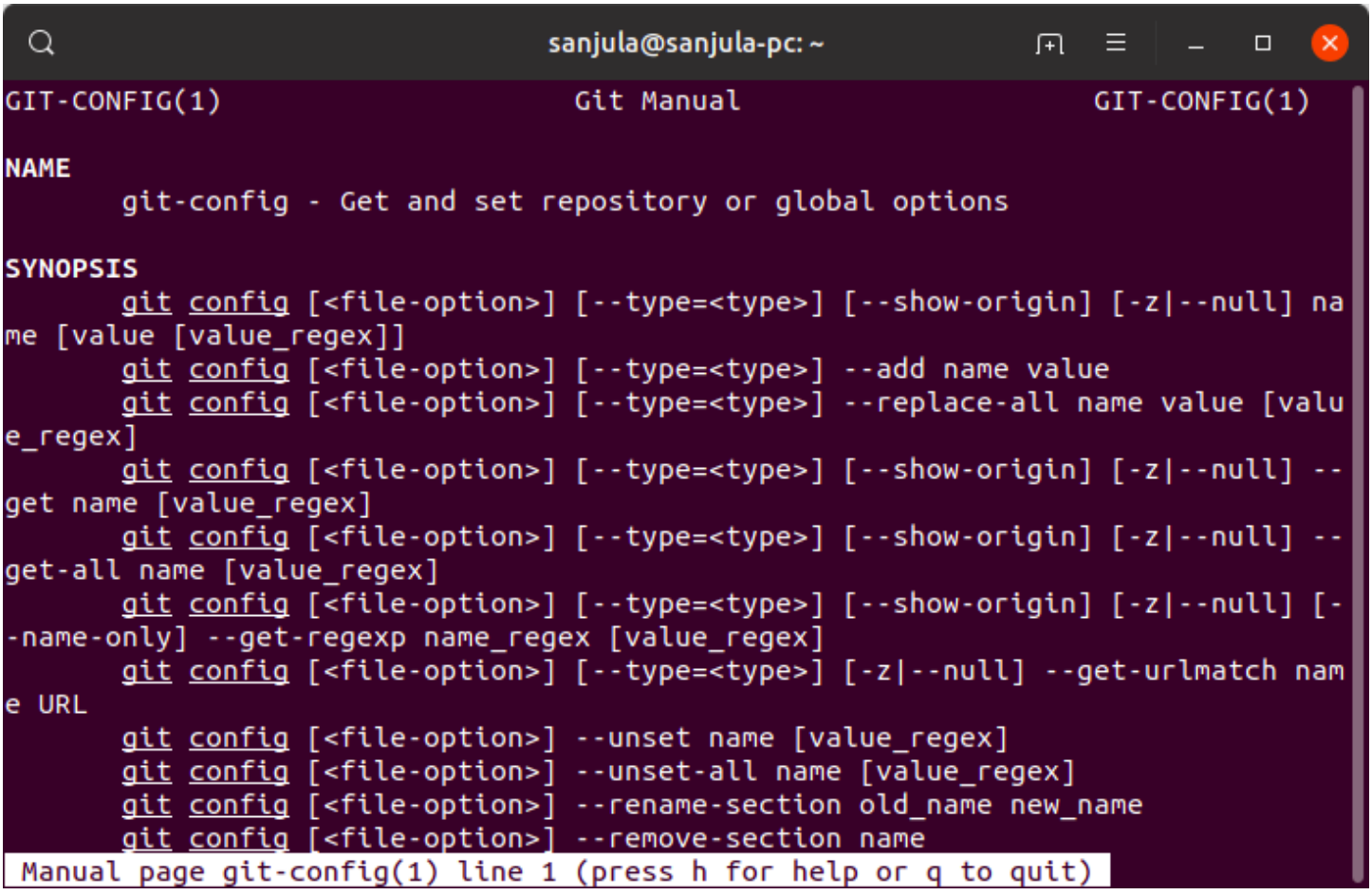

Figure-3: Comando Help

Los dos comandos indicados realizan la misma acción. Muestran la página de manual del verbo especificado. Esto será útil para identificar capacidades más avanzadas de git.

Cómo inicializar un repositorio a partir de código existente

Si tienes un repositorio local que deseas convertir en un proyecto git para comenzar a rastrearlo,

entonces podemos comenzar ejecutando el comando de abajo dentro del directorio del proyecto.

```
git init
```

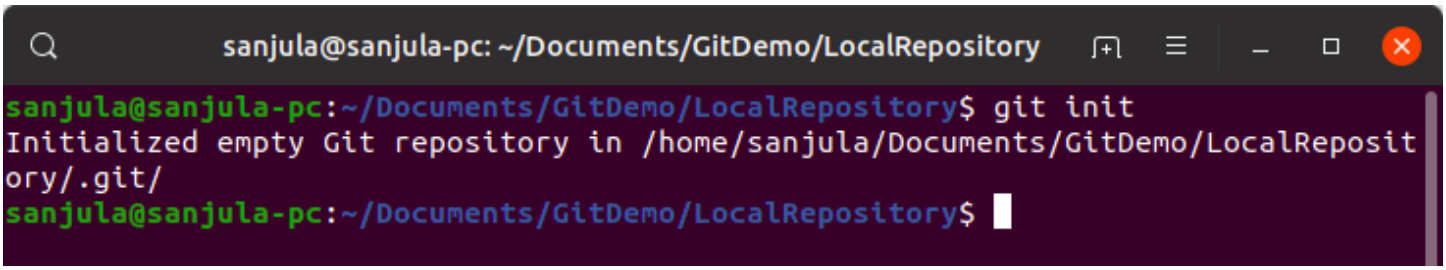

Figure-4: Git init

¡Listo! Así, has convertido tu proyecto en un repositorio local de git. Si abres la carpeta del proyecto, verás que se ha creado un nuevo directorio llamado .git.

Que hacer antes del primer commit

Ingresa el siguiente comando para ver los archivos sin seguimiento (untracked files):

```
git status
```

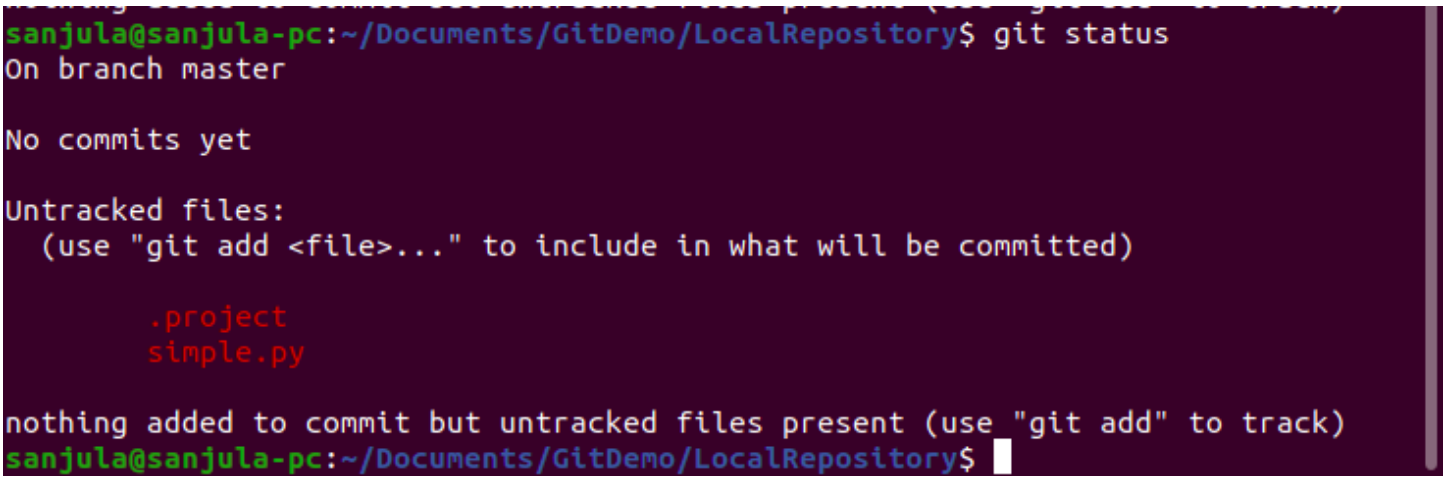

Figure-5: Git status

Si hay archivos que no deseas que otras personas vean en el repositorio, como archivos que contienen preferencias personales o las del IDE, has lo siguiente:

```
touch .gitignore
```

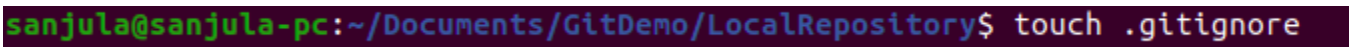

Figure-6: Crear archivo .gitignore

Para especificar qué archivos no se agregarán al repositorio de git, abre en un editor de texto el archivo .gitignore, que se puede editar como un archivo de texto normal. Ahora podemos ingresar lo siguiente en el archivo, por ejemplo:

```
.project
```

```
*.java
```

También se pueden utilizar caracteres comodín. En este caso, se ha utilizado para especificar que no se agreguen todos los archivos que terminan con la extensión .java al repositorio.

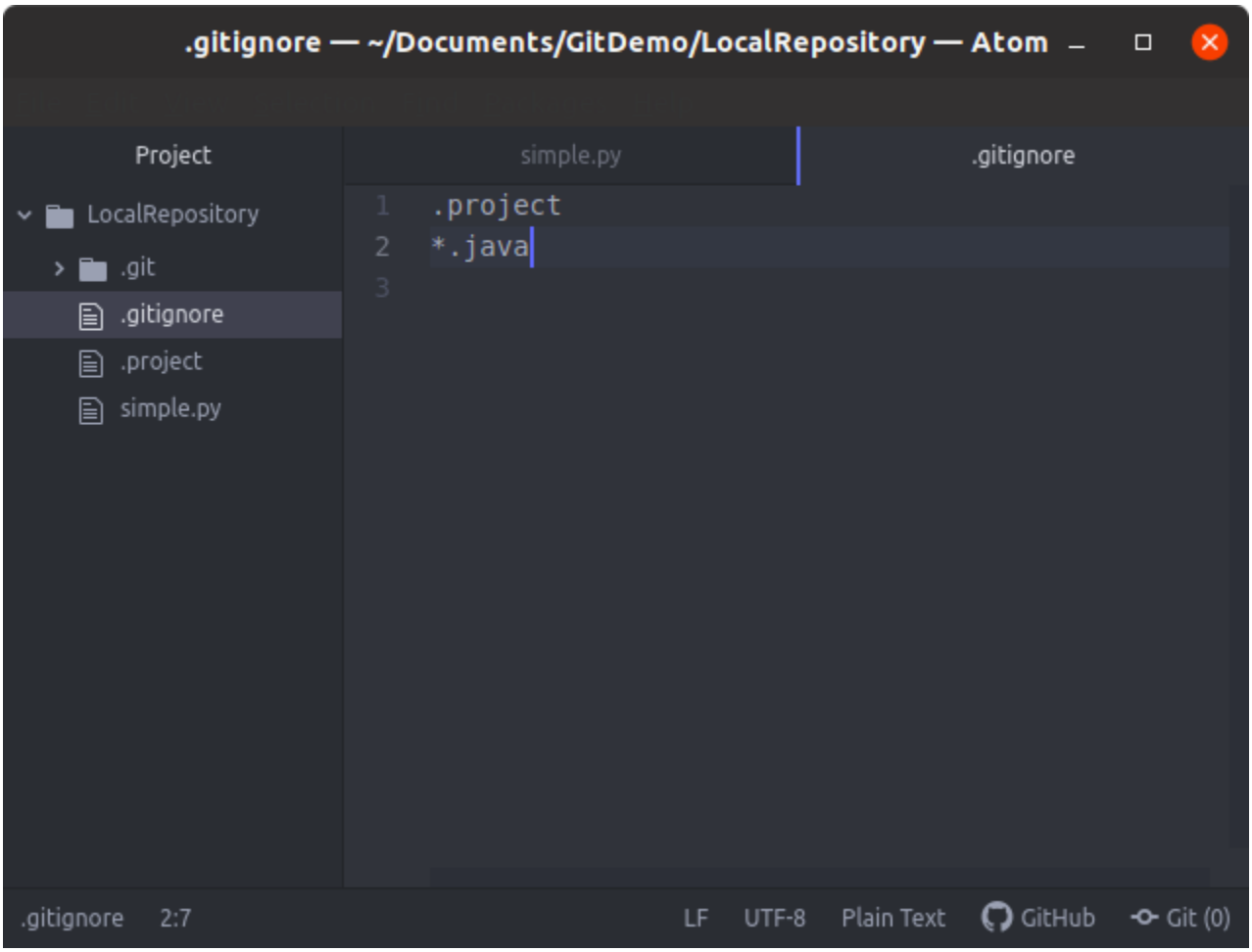

Figure-7: Edición en el editor de texto

Ahora ejecuta nuevamente git status

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        simple.py

nothing added to commit but untracked files present (use "git add" to track)
```

Figure-8:Después de actualizar .gitignore

Ahora puedes ver que los archivos que indicamos en el archivo .gitignore ya no se muestran en la lista de archivos sin seguimiento. El archivo .gitignore debe confirmarse (usando el comando commit) en

el repositorio para mantener las mismas exclusiones en todos los demás lugares.

Agregando archivos al área de preparación (staging area)

Todo este tiempo estuvimos en el directorio de trabajo. El área de preparación es donde organizamos todos los archivos que se rastrean y deben confirmarse antes de enviarlos al repositorio de git. Es un archivo que almacena lo que se debe incluir en la próxima confirmación.

Si deseas agregar todos los archivos que actualmente están sin seguimiento y has cambiado al área de preparación, usa el siguiente comando:

```
git add -A
```

Si deseas agregar archivos individualmente, podemos indicar el nombre del archivo después de git add. Por ejemplo,

```
git add .gitignore
```

Ahora, si escribes git status, verás que el archivo .gitignore está en el área de preparación.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git add -A
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   simple.py
```

Figure-9: area para Staging

Eliminando archivos del área de preparación

Para eliminar archivos del área de preparación de manera individual, escribe lo siguiente (por ejemplo):

```
git reset simple.py
```

Esto eliminará el archivo simple.py del área de preparación. Para ver este cambio, escribe nuevamente el comando git status.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git reset simple.py
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git status
On branch master

No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        simple.py
```

Figure-10: Eliminación del archivo del área de preparación

Si deseas eliminar todos los archivos del área de preparación, entonces ejecuta lo siguiente:

```
git reset
```

Ahora, si escribes git status, veremos que todos los archivos han cambiado a archivos sin seguimiento.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git reset
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        simple.py

nothing added to commit but untracked files present (use "git add" to track)
```

Figure-11: Restablecer todos los archivos

Ejecutando el primer commit

Ahora ejecuta lo siguiente para agregar todos los archivos al área de preparación para ser confirmados.

```
git add -A
```

Si lo deseas, puedes ejecutar git status para ver todos los archivos que serán confirmados.

Para realizar un commit, escribe lo siguiente.

```
git commit -m “Initial Commit”
```

“-m” especifica un mensaje que se debe pasar describiendo la confirmación. Dado que este es nuestro primer commit, escribiremos Initial Commit.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git add -A
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git commit -m "Initial C
ommit"
[master (root-commit) d39d689] Initial Commit
```

```
 2 files changed, 4 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 simple.py
```

Figure-12: Initial Commit

Como puedes ver, el commit se ha ejecutado correctamente.

Si ahora ejecutas git status, verás que se indica que el directorio de trabajo está limpio ya que se han confirmado todos los archivos y no se ha modificado ninguno desde entonces.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git status
On branch master
nothing to commit, working tree clean
```

Figure-13: Árbol de trabajo después del commit

Si ejecutamos el siguiente comando:

```
git log
```

luego podemos ver el commit que habiamos ejcutado, incluyendo el número hash del commit.

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ git log
commit d39d68912a5ddace58df3f1a52156686d078a804 (HEAD -> master)
Author: Sanjula Madurapperuma <sanjula99@gmail.com>
Date:   Fri Apr 19 10:18:29 2019 +0530

    Initial Commit
```

Figure-14: número de hash Commit

¡Ahora estamos rastreando exitosamente el proyecto local con git!

Clonando un repositorio remoto

Si queremos rastrear un proyecto remoto existente con git, entonces tenemos que escribir un comando en el siguiente formato:

```
git clone <url> <directorio donde clonar>
```

A modo de ejemplo, usaré el repositorio de git en este enlace.

Primero me ubicaré en el directorio donde quiero clonar el proyecto, aunque puedes especificar esto tal como se muestra arriba.

Ve al enlace del repositorio indicado antes y has clic en "Code", luego copia el url que figura.

Luego escribe:

```
git clone https://github.com/sanjulamadurapperuma/GitDemoMedium.git
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/LocalRepository$ cd ..
sanjula@sanjula-pc:~/Documents/GitDemo$ cd RemoteRepository
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository$ git clone https://githu
b.com/sanjulamadurapperuma/GitDemoMedium.git
Cloning into 'GitDemoMedium'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
```

Figure-15: Clonación del repositorio remoto

De esta forma hemos clonado el repositorio exitosamente.

Si ingresamos el siguiente comando, veremos todos los archivos que ahora están en el directorio local.

```
ls -la
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository$ cd GitDemoMedium
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ ls -la
total 20
drwxr-xr-x 3 sanjula sanjula 4096 ago  19 10:31 .
drwxr-xr-x 3 sanjula sanjula 4096 ago  19 10:31 ..
drwxr-xr-x 8 sanjula sanjula 4096 ago  19 10:31 .git
-rw-r--r-- 1 sanjula sanjula   16 ago  19 10:31 .gitignore
-rw-r--r-- 1 sanjula sanjula   20 ago  19 10:31 simple.py
```

Figure-16: Listar todos los archivos en el directorio

Viendo información sobre el repositorio remoto

Si escribes el siguiente comando:

```
git remote -v
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git remot
e -v
origin  https://github.com/sanjulamadurapperuma/GitDemoMedium.git (fetch)
origin  https://github.com/sanjulamadurapperuma/GitDemoMedium.git (push)
```

Figure-17: Git remote -v

Este comando enumerará las ubicaciones de donde el repositorio local obtendrá los cambios realizados externamente y a dónde serán enviadas tus confirmaciones o cambios que realices al repositorio remoto.

Si escribes el comando:

```
git branch -a
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branc
h -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

Figure-18: Lista todas las ramas de git

Esto enumerará todas las ramas que se encuentran en el repositorio, tanto local como remotamente.

Para demostrar la actualización del repositorio remoto, haremos algunos cambios en los archivos del repositorio que clonamos.

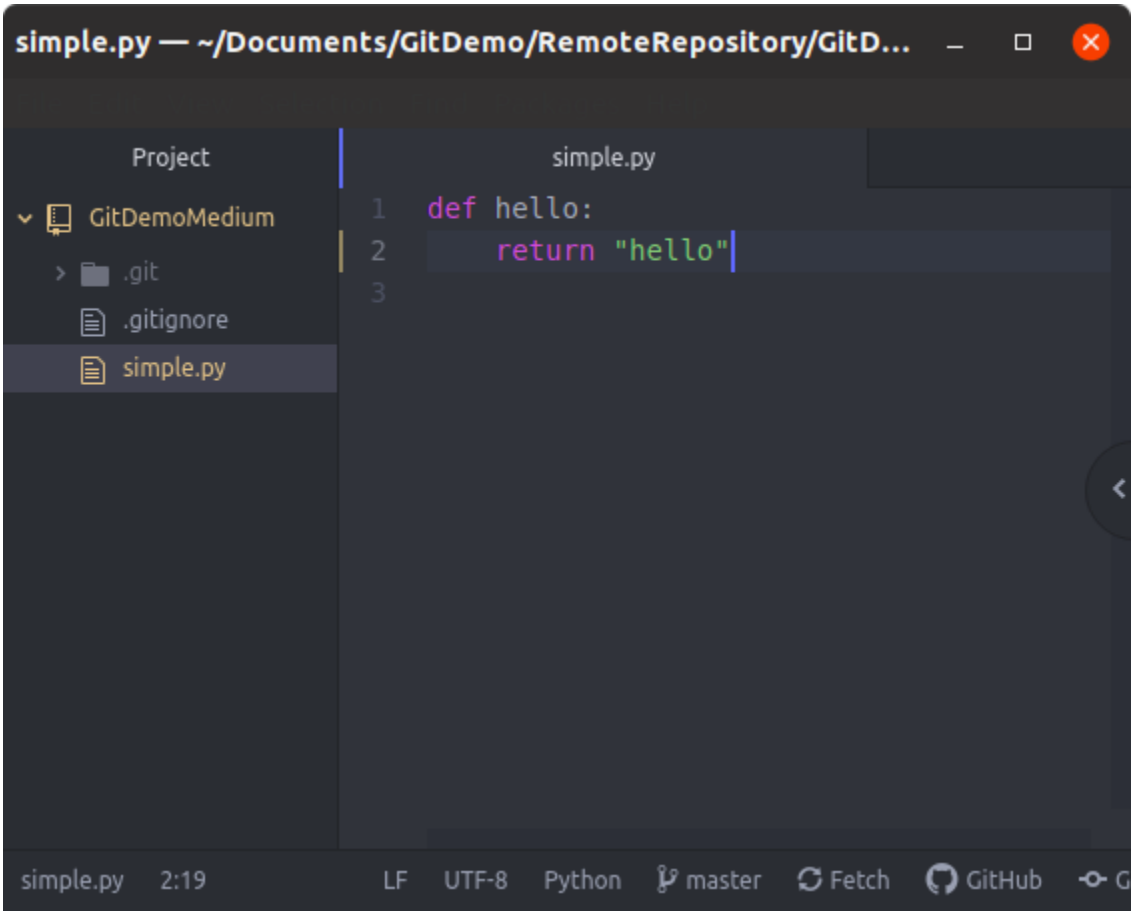

Figure-19: Realizar cambios en simple.py

Ahora que hemos realizado un cambio en nuestro código, la siguiente acción que debemos realizar es enviar estos cambios al repositorio remoto.

Enviando los cambios al repositorio remoto

El siguiente comando mostrará todos los cambios que se han hecho a los archivos.

```
git diff
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git diff
diff --git a/simple.py b/simple.py
index 60e55ba..0df6225 100644
--- a/simple.py
+++ b/simple.py
@@ -1,2 +1,2 @@
 def hello:
-    pass
+    return "hello"
```

Figure-20: Ver los cambios en el archivo

Si ingresamos git status de nuevo, veremos que se han rastreado cambios y que simple.py ha sido modificado.

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git statu
s
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   simple.py

no changes added to commit (use "git add" and/or "git commit -a")
```

Figure-21: Ver archivos modificados

Ahora agrégalos al área de preparación

```
git add -A
```

Ejecuta git status nuevamente

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git add -
A
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git statu
s
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   simple.py
```

Figure-22: Agregar archivos al área de staging

Ahora vemos que simple.py esta listo para ser confirmado.

Luego escribe el comando commit con un mensaje

```
git commit -m “Updated hello function”
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git commit
 -m "Updated hello function"
[master fe013bb] Updated hello function
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Figure-23: mensaje Commit

Ahora debemos enviar los cambios confirmados al repositorio remoto para que otras personas tengan

acceso a ellos.

Dado que lo común es que hay varios desarrolladores trabajando en un solo proyecto, primero tenemos que extraer cualquier cambio que se haya realizado en el repositorio remoto antes de enviar nuestros cambios para evitar conflictos.

Ejecuta el siguiente comando:

```
git pull origin master
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git pull o
rigin master
From https://github.com/sanjulamadurapperuma/GitDemoMedium
 * branch            master     -> FETCH_HEAD
Already up to date.
```

Figure-24: Extraer cambios del repositorio remoto

Como ya estamos actualizados, ahora podemos enviar nuestros cambios al repositorio remoto.

Ahora ejecuta lo siguiente:

```
git push origin master
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git push o
rigin master
Username for 'https://github.com': sanjulamadurapperuma
Password for 'https://sanjulamadurapperuma@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/sanjulamadurapperuma/GitDemoMedium.git
   2de7692..fe013bb  master -> master
```

Figure-25: Subir cambios al repositorio remoto

¡Hemos enviado con éxito nuestros cambios a la rama principal del repositorio remoto!

Creando una rama para una prestación o problema especifico

Hasta ahora hemos estado trabajando en nuestra rama maestra o principal, pero no es así como deberías trabajar en git como desarrollador porque la rama maestra debe ser una versión estable del proyecto en el que estás trabajando. Entonces, para cada prestación o problema, generalmente es la norma crear tu propia rama y luego trabajar sobre esa rama.

El comando para crear una nueva rama llamada simple-greeting es el siguiente:

```
git branch simple-greeting
```

Ahora si ejecutas

```
git branch
```

luego verás todas las ramas del repositorio, y la rama en la que tu estás ubicado se encuentra resaltada con un asterisco del lado izquierdo

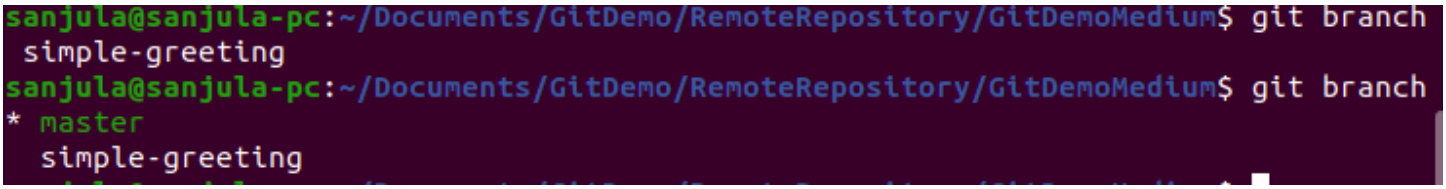

Figure-26: git branch

Si deseas cambiarte a la rama recientemente creada por ti, escribe lo siguiente:

```
git checkout simple-greeting
```

Ahora, si escribes git branch verás que ahora te encuentras en la rama simple-greeting.

Ahora debemos realizar los cambios en el proyecto. Nos dirigimos al archivo y definimos la función greeting.

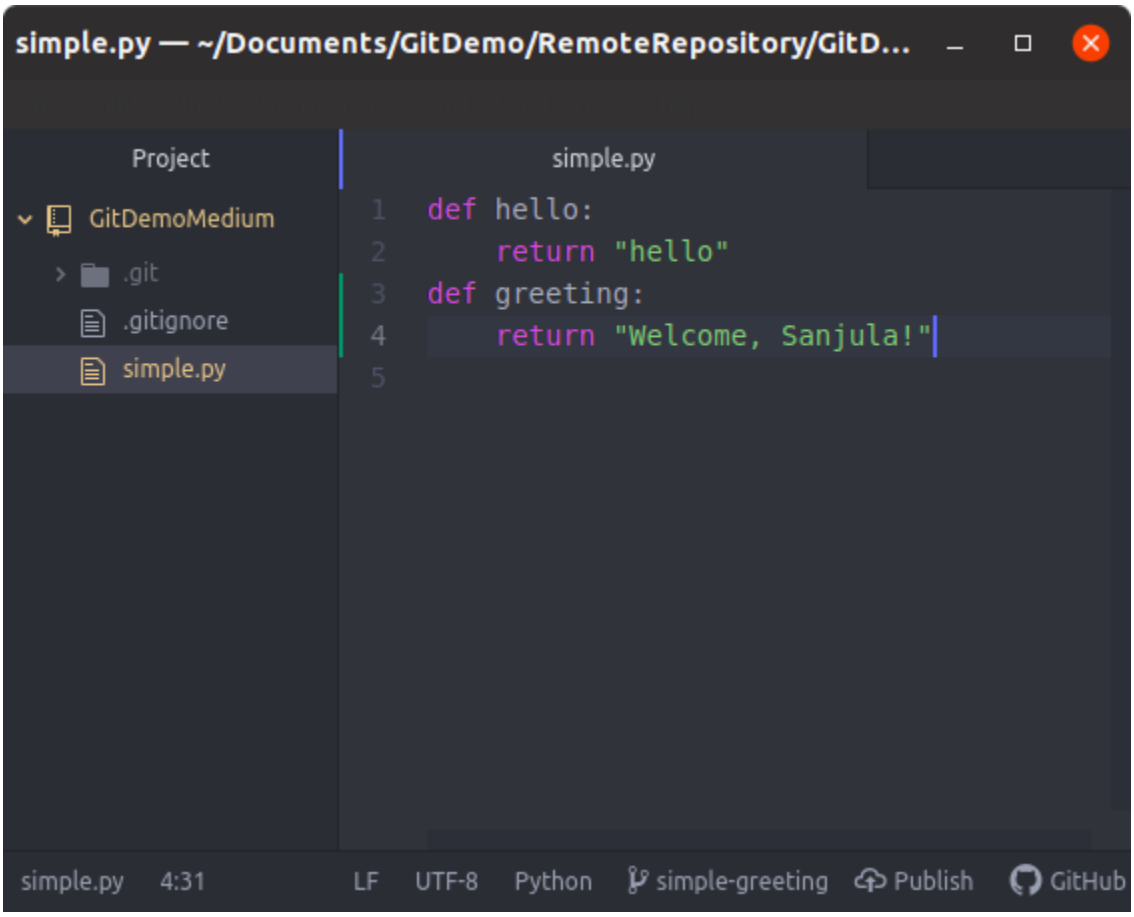

Figure-27: Definir la función de saludo

Ahora repetimos el proceso para confirmar estos cambios:

```
git status
```

```
On branch simple-greeting
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   simple.py

no changes added to commit (use "git add" and/or "git commit -a")
```

Figure-28: Ver los cambios que no son staged

```
git add -A
```

```
git commit -m “Greeting Function”
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git add -A
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git commit
 -m "Greeting Function"
[simple-greeting 7a4eb10] Greeting Function
 1 file changed, 2 insertions(+)
```

Figure-29: mensaje Commit para la función de saludo

Este commit solo cambiará los archivos en la rama simple-greeting local, no habiendo alterado aún la rama master local ni el repositorio remoto.

Enviando la rama al repositorio remoto luego de efectuar una confirmación

Ingresa el siguiente comando:

```
git push -u origin simple-greeting
```

donde origin es el nombre del respositorio y simple-greeting es la rama que le queremos enviar.

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git push -
u origin simple-greeting
Username for 'https://github.com': sanjulamadurapperuma
Password for 'https://sanjulamadurapperuma@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'simple-greeting' on GitHub by visiting:
remote:      https://github.com/sanjulamadurapperuma/GitDemoMedium/pull/new/simpl
e-greeting
remote:
To https://github.com/sanjulamadurapperuma/GitDemoMedium.git
 * [new branch]      simple-greeting -> simple-greeting
Branch 'simple-greeting' set up to track remote branch 'simple-greeting' from 'or
igin'.
```

Figure-30: Sube la rama al repositorio remoto

Ahora hemos enviado la rama simple-greeting al repositorio remoto. Si escribes:

```
git branch -a
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
 -a
  master
* simple-greeting
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/simple-greeting
```

Figure-31: Rama de simple-greeting en el repositorio remoto

Ahora vemos que en nuestro repositorio remoto tenemos la rama simple-greeting. ¿Porqué debemos enviar la rama al repositorio remoto? Porque en algunas empresas es allí donde ejecutan sus pruebas unitarias y en otras para asegurarse de que el código se ejecute bien antes de fusionarse con la rama maestra.

Dado que todas la prueban ha sido exitosas (no entraremos en detalles de eso aquí), ahora podemos fusionar la rama simple-greeting con la rama principal.

Fusionando una rama

Primero, debemos ubicarnos (checkout) en la rama maestra local

```
git checkout master
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git checko
ut master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Figure-32: Navegar a la rama master

Extraemos todos los cambios de la rama maestra remota:

```
git pull origin master
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git pull o
rigin master
From https://github.com/sanjulamadurapperuma/GitDemoMedium
 * branch            master     -> FETCH_HEAD
Already up to date.
```

Figure-33: Extraer cambios del repositorio remoto

Ahora veremos todas las ramas que hemos fusionado hasta ahora:

```
git branch —-merged
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
 --merged
* master
```

Figure-34: Mostrar ramas ligadas

la rama simple-greeting no figurará ya que aún no la hemos fusionado.

Para fusionar simple-greeting con la principal, ingresa:

```
git merge simple-greeting
```

(Ten en cuenta que ahora estamos en la rama maestra)

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git merge
simple-greeting
Updating fe013bb..7a4eb10
Fast-forward
 simple.py | 2 ++
 1 file changed, 2 insertions(+)
```

Figure-35: Ligar la rama simple-greeting

Ahora que ha sido fusionada, podemos enviar los cambios a la rama maestra del repositorio remoto.

```
git push origin master
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git push o
rigin master
Username for 'https://github.com': sanjulamadurapperuma
Password for 'https://sanjulamadurapperuma@github.com':
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/sanjulamadurapperuma/GitDemoMedium.git
   fe013bb..7a4eb10  master -> master
```

Figure-36: Sube a la rama maestra remota

Ahora los cambios han sido enviados a la rama maestra del repositorio remoto.

Eliminando una rama

Dado que la función o la nueva prestación ya se ha implementado, podemos eliminar la rama simple-greeting. Para verificar la fusión realizada en la sección anterior, podemos ejecutar:

```
git branch --merged
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
 --merged
* master
  simple-greeting
```

Figure-37: Mostrar ramas ligadas

Si simple-greeting se muestra aquí, eso significa que hemos fusionado todos los cambios y que la rama ya puede ser descartada.

```
git branch -d simple-greeting
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
```

```
 -d simple-greeting
Deleted branch simple-greeting (was 7a4eb10).
```

Figure-38: Eliminar rama local simple-greeting

Ahora la rama ha sido eliminada localmente.

Pero como la hemos enviado al repositorio remoto, aún continua ahí. Esto puede ser visto ejecutando:

```
git branch -a
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
 -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/simple-greeting
```

Figure-39: Mostrar todas las ramas

Para eliminar la rama del repositorio remoto, escribe:

```
git push origin --delete simple-greeting
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git push o
rigin --delete simple-greeting
Username for 'https://github.com': sanjulamadurapperuma
Password for 'https://sanjulamadurapperuma@github.com':
To https://github.com/sanjulamadurapperuma/GitDemoMedium.git
 - [deleted]         simple-greeting
```

Figure-40: Eliminar rama remota simple-greeting

Si volvemos a ejecutar

```
git branch -a
```

```
sanjula@sanjula-pc:~/Documents/GitDemo/RemoteRepository/GitDemoMedium$ git branch
 -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

Figure-41: Mostrar todas las ramas

Podemos ver que la rama ahora a sido eliminada también del repositorio remoto.

¡¡Felicitaciones!! ¡Ahora eres un maestro en los comandos básicos pero críticos de Git!

Para referencia o uso de este tutorial, aquí está el enlace del repositorio público de GitHub

Traducido del artículo de **Sanjula Madurapperuma** - The Essential Git Handbook