

## ProGit Capítulo 2

### 2.5 Fundamentos de Git - Trabajar con Remotos

#### Trabajar con Remotos

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar repositorios remotos. Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más. En esta sección, trataremos algunas de estas habilidades de gestión de remotos.

Ver Tus Remotos

Para ver los remotos que tienes configurados, debes ejecutar el comando `git remote`. Mostrará los nombres de cada uno de los remotos que tienes especificados. Si has clonado tu repositorio, deberías ver al menos `origin` (origen, en inglés) - este es el nombre que por defecto Git le da al servidor del que has clonado:

```
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
$ git remote
origin
```

También puedes pasar la opción `-v`, la cual muestra las URLs que Git ha asociado al nombre y que serán usadas al leer y escribir en ese remoto:

```
$ git remote -v
origin      https://github.com/schacon/ticgit (fetch)
origin      https://github.com/schacon/ticgit (push)
```

Si tienes más de un remoto, el comando los listará todos. Por ejemplo, un repositorio con múltiples remotos para trabajar con distintos colaboradores podría verse de la siguiente manera.

```
$ cd grit
$ git remote -v
bakkdoor https://github.com/bakkdoor/grit (fetch)
bakkdoor https://github.com/bakkdoor/grit (push)
cho45 https://github.com/cho45/grit (fetch)
cho45 https://github.com/cho45/grit (push)
defunkt https://github.com/defunkt/grit (fetch)
defunkt https://github.com/defunkt/grit (push)
koke git://github.com/koke/grit.git (fetch)
koke git://github.com/koke/grit.git (push)
origin git@github.com:mojombo/grit.git (fetch)
origin git@github.com:mojombo/grit.git (push)
```

Esto significa que podemos traer contribuciones de cualquiera de estos usuarios fácilmente. Es posible que también tengamos permisos para enviar datos a algunos, aunque no podemos saberlo desde aquí.

Fíjate que estos remotos usan distintos protocolos; hablaremos sobre ello más adelante, en Configurando Git en un servidor.

## Añadir Repositorios Remotos

En secciones anteriores hemos mencionado y dado alguna demostración de cómo añadir repositorios remotos. Ahora veremos explícitamente cómo hacerlo. Para añadir un remoto nuevo y asociarlo a un nombre que puedas referenciar fácilmente, ejecuta `git remote add [nombre] [url]`:

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb https://github.com/paulboone/ticgit (fetch)
pb https://github.com/paulboone/ticgit (push)
```

A partir de ahora puedes usar el nombre `pb` en la línea de comandos en lugar de la URL entera. Por ejemplo, si quieres traer toda la información que tiene Paul pero tú aún no tienes en tu repositorio, puedes ejecutar `git fetch pb`:

```
$ git fetch pb
```

```
remote: Counting objects: 43, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 43 (delta 10), reused 31 (delta 5)
Unpacking objects: 100% (43/43), done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit     -> pb/ticgit
```

La rama maestra de Paul ahora es accesible localmente con el nombre pb/master - puedes combinarla con alguna de tus ramas, o puedes crear una rama local en ese punto si quieres inspeccionarla. (Hablaremos con más detalle acerca de qué son las ramas y cómo utilizarlas en [ch03-git-branching].)

Traer y Combinar Remotos

Como hemos visto hasta ahora, para obtener datos de tus proyectos remotos puedes ejecutar:

```
$ git fetch [remote-name]
```

El comando irá al proyecto remoto y se traerá todos los datos que aun no tienes de dicho remoto. Luego de hacer esto, tendrás referencias a todas las ramas del remoto, las cuales puedes combinar e inspeccionar cuando quieras.

Si clonas un repositorio, el comando de clonar automáticamente añade ese repositorio remoto con el nombre "origin". Por lo tanto, git fetch origin se trae todo el trabajo nuevo que ha sido enviado a ese servidor desde que lo clonaste (o desde la última vez que trajiste datos). Es importante destacar que el comando git fetch solo trae datos a tu repositorio local - ni lo combina automáticamente con tu trabajo ni modifica el trabajo que llevas hecho. La combinación con tu trabajo debes hacerla manualmente cuando estés listo.

Si has configurado una rama para que rastree una rama remota (más información en la siguiente sección y en [ch03-git-branching]), puedes usar el comando git pull para traer y combinar automáticamente la rama remota con tu rama actual. Es posible que este sea un flujo de trabajo mucho más cómodo y fácil para ti; y por defecto, el comando git clone le indica automáticamente a tu rama maestra local que rastree la rama maestra remota (o como se llame la rama por defecto) del servidor del que has clonado. Generalmente, al ejecutar git pull traerás datos del servidor del que clonaste originalmente y se intentará combinar automáticamente la información con el código en el que estás trabajando.

## Enviar a Tus Remotos

Cuando tienes un proyecto que quieres compartir, debes enviarlo a un servidor. El comando para hacerlo es simple: git push [nombre-remoto] [nombre-rama]. Si quieres enviar tu rama master a tu servidor origin (recuerda, clonar un repositorio establece esos nombres automáticamente), entonces puedes ejecutar el siguiente comando y se enviarán todos los commits que hayas hecho al servidor:

```
$ git push origin master
```

Este comando solo funciona si clonaste de un servidor sobre el que tienes permisos de escritura y si nadie más ha enviado datos por el medio. Si alguien más clona el mismo repositorio que tú y envía información antes que tú, tu envío será rechazado. Tendrás que traerte su trabajo y combinarlo con el tuyo antes de que puedas enviar datos al servidor. Para información más detallada sobre cómo enviar datos a servidores remotos, véase [ch03-git-branching].

## Inspeccionar un Remoto

Si quieres ver más información acerca de un remoto en particular, puedes ejecutar el comando `git remote show [nombre-remoto]`. Si ejecutas el comando con un nombre en particular, como `origin`, verás algo como lo siguiente:

```
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push  URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master                               tracked
    dev-branch                           tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

El comando lista la URL del repositorio remoto y la información del rastreo de ramas. El comando te indica claramente que si estás en la rama maestra y ejecutas el comando `git pull`, automáticamente combinará la rama maestra remota con tu rama local, luego de haber traído toda la información de ella. También lista todas las referencias remotas de las que ha traído datos.

Ejemplos como este son los que te encontrarás normalmente. Sin embargo, si usas Git de forma más avanzada, puede que obtengas mucha más información de un `git remote show`:

```
$ git remote show origin
* remote origin
  URL: https://github.com/my-org/complex-project
  Fetch URL: https://github.com/my-org/complex-project
  Push  URL: https://github.com/my-org/complex-project
  HEAD branch: master
  Remote branches:
```

```
master                tracked
dev-branch            tracked
markdown-strip        tracked
issue-43              new (next fetch will store in remotes/origin)
issue-45              new (next fetch will store in remotes/origin)
refs/remotes/origin/issue-11  stale (use 'git remote prune' to
remove)

Local branches configured for 'git pull':
  dev-branch merges with remote dev-branch
  master     merges with remote master

Local refs configured for 'git push':
  dev-branch pushes to dev-branch          (up to date)
  markdown-strip pushes to markdown-strip  (up to date)
  master     pushes to master              (up to date)
```

Este comando te indica a cuál rama enviarás información automáticamente cada vez que ejecutas git push, dependiendo de la rama en la que estés. También te muestra cuáles ramas remotas no tienes aún, cuáles ramas remotas tienes que han sido eliminadas del servidor, y varias ramas que serán combinadas automáticamente cuando ejecutes git pull.

## Eliminar y Renombrar Remotos

Si quieres cambiar el nombre de la referencia de un remoto puedes ejecutar git remote rename. Por ejemplo, si quieres cambiar el nombre de pb a paul, puedes hacerlo con git remote rename:

```
$ git remote rename pb paul
$ git remote
origin
paul
```

Es importante destacar que al hacer esto también cambias el nombre de las ramas remotas. Por lo tanto, lo que antes estaba referenciado como pb/master ahora lo está como paul/master.

Si por alguna razón quieres eliminar un remoto - has cambiado de servidor o no quieres seguir utilizando un mirror o quizás un colaborador ha dejado de trabajar en el proyecto - puedes usar git remote rm:

```
$ git remote rm paul
$ git remote
origin
```