Git

August 21, 2016

Git es un sistema de manejo de versiones distribuido creado por Linus Torvalds.

Los sistemas de manejo de versiones permiten tener un control detallado sobre el historial de cambios de un archivo o conjunto de archivos creando instantaneas de estos conjuntos cada que se presenta un cambio. Estos sistemas de control de versiones pueden ser centralizados (ejemplo subversion) o distribuidos (ejemplo git), acorde a si existe un repositorio principal donde se maneja el historial y desde el cual todos los demas repositorios extraen las versiones requeridas, o si todo el historial se distribuye en cada repositorio.

Para el desarrollo de esta sesión se requiere la creación de cuenta en github o bitbucket.

1 Instalación

Para la instalación de git use el siguiente procedimiento.

1.1 Windows

Si usa sistema operativo windows puede descargar los binarios del cliente de *git for windows* y durante la instalación usar la configuración recomendada.

1.2 Ubuntu Linux

En su sistema linux, muy probablemente encontrará el cliente de comandos para instalar desde el repositorio de su distribución.

```
sudo apt install git
```

Si desea realizar la autenticación por medio de SSH requiere instalar los paquetes adicionales a SSH, sin embargo se recomienda la autenticación por medio de HTTP para reducir el número de pasos requeridos, y cuando se desea trabajar en equipos de uso no habitual o que siendo de uso frecuente podría ser usado por un tercero. Igualmente, al usar autenticación por HTTP es posible conectarse a pesar de los bloqueos que por seguridad establecen las politicas de muchas entidades.

1.3 Mac

En mac, puede usar igualmente a linux la instalación por binarios disponibles en la página oficial, *git*.

1.4 Alternativas

Puede usar un cliente multiplataforma y gráfico de fácil instalación como gitkraken o github desktop (este último solo para windows o mac), sin embargo para fines de este curso se exigirá el manejo directo del cliente de comandos de git.

1.5 Uso

Una vez instalado nuestro cliente `git` podemos consultar sus opciones y argumentos.

```
In [1]: %%bash
        git --help
```

```
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Forward-port local commits to the updated upstream head
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
```

```
   pull        Fetch from and integrate with another repository or a local branch
   push        Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

1.6 Configuración

Para una adecuada identificación de nuestros aportes en los repositorios, es necesario configurar nuestro nombre y correo en git.

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

1.7 Creación de repositorios

Una vez creada la cuenta github, debe crear un repositorio git en ella y copiar la dirección de clonado, la cual se usa con el fin de obtener una copia del repositorio localmente.

```
In [2]: %%bash
        git clone https://github.com/cosmoscalibur/herramientas_computacionales.git

Cloning into 'herramientas'...
```

De esta forma crearemos una copia del repositorio git `herramientas_computacionales` en el directorio `./herramientas`. Si no se indica el directorio, se crea un directorio nuevo en la posición actual.

Tras el clonado, tenemos a disposición todo el directorio del repositorio, incluido el directorio `.git` que almacena la información del historial y repositorios (para este momento, heredada del repositorio clonado).

```
In [3]: %%bash
        cd herramientas
        ls -oha

total 1,2M
drwxrwxr-x 3 cosmoscalibur 4,0K ago 21 19:49 .
drwxrwxr-x 5 cosmoscalibur 4,0K ago 21 19:49 ..
drwxrwxr-x 8 cosmoscalibur 4,0K ago 21 19:49 .git
-rw-rw-r-- 1 cosmoscalibur   56 ago 21 19:49 .gitignore
-rw-rw-r-- 1 cosmoscalibur  20K ago 21 19:49 Git.ipynb
-rw-rw-r-- 1 cosmoscalibur  23K ago 21 19:49 Jupyter Notebook Basico.ipynb
-rw-rw-r-- 1 cosmoscalibur 139K ago 21 19:49 Jupyter+Notebook+Basico.pdf
-rw-rw-r-- 1 cosmoscalibur 5,6K ago 21 19:49 Jupyter Notebook Intermedio.ipynb
-rw-rw-r-- 1 cosmoscalibur  92K ago 21 19:49 Jupyter+Notebook+Intermedio.pdf
-rw-rw-r-- 1 cosmoscalibur  13K ago 21 19:49 jupyter.png
```

```
-rw-rw-r-- 1 cosmoscalibur 182K ago 21 19:49 LaTeX_basico.pdf
-rw-rw-r-- 1 cosmoscalibur  18K ago 21 19:49 LaTeX_basico.tex
-rw-rw-r-- 1 cosmoscalibur 1,1K ago 21 19:49 LICENSE
-rw-rw-r-- 1 cosmoscalibur  17K ago 21 19:49 Linux Bash.ipynb
-rw-rw-r-- 1 cosmoscalibur  83K ago 21 19:49 Linux+Bash.pdf
-rw-rw-r-- 1 cosmoscalibur  36K ago 21 19:49 Linux Basico.ipynb
-rw-rw-r-- 1 cosmoscalibur  93K ago 21 19:49 Linux+Basico.pdf
-rw-rw-r-- 1 cosmoscalibur 119K ago 21 19:49 minimo_taller1.pdf
-rw-rw-r-- 1 cosmoscalibur  12K ago 21 19:49 presentacion_herramientas.md
-rw-rw-r-- 1 cosmoscalibur 137K ago 21 19:49 presentacion_herramientas.pdf
-rw-rw-r-- 1 cosmoscalibur 1005 ago 21 19:49 README.md
-rw-rw-r-- 1 cosmoscalibur 7,1K ago 21 19:49 Taller_1.ipynb
-rw-rw-r-- 1 cosmoscalibur 112K ago 21 19:49 Taller_1.pdf
```

Se evidencia que tras el clonado, tenemos configurados los repositorios remotos por defecto.

```
In [4]: %%bash
        cd herramientas
        git remote -v
```

```
origin        https://github.com/cosmoscalibur/herramientas_computacionales.git (fe
origin        https://github.com/cosmoscalibur/herramientas_computacionales.git (pu
```

En caso de ser un repositorio nuevo o desear reinicializar (aceptar nuevas plantillas o mover el directorio), debemos crear el directorio `.git` y agregar los elementos que teniamos en nuestro directorio. Siempre que se agreguen (o se realiza cualquier actualización de los archivos y directorios, se sugiere realizar un `commit` para la indicación de la acción en nuestro historial).

```
In [5]: %%bash
        cd herramientas
        git init
```

```
Reinitialized existing Git repository in /home/cosmoscalibur/Dropbox/UdeM/Herramier
```

1.8 Repositorios remotos

Una vez se inicializa o se clona, podemos agregar nuestro reposotorio remoto, que previamente fue creado en github o bitbucket. Si solo se desea un uso local o si es justo el repositorio clonado no será necesario.

```
In [6]: %%bash
        cd herramientas
        git remote add pruebas https://github.com/cosmoscalibur/pruebas.git
        git remote -v
```

```
origin        https://github.com/cosmoscalibur/herramientas_computacionales.git (fe
origin        https://github.com/cosmoscalibur/herramientas_computacionales.git (pu
pruebas        https://github.com/cosmoscalibur/pruebas.git (fetch)
pruebas        https://github.com/cosmoscalibur/pruebas.git (push)
```

Tambien es posible eliminar el acceso al servidor remoto, con `git remote rm`.

```
In [7]: %%bash
        cd herramientas
        git remote rm origin
        git remote -v
```

```
pruebas        https://github.com/cosmoscalibur/pruebas.git (fetch)
pruebas        https://github.com/cosmoscalibur/pruebas.git (push)
```

1.9 Objetos y referencias

A continuación, y para evitar posibles conflictos de los archivos, es recomendable iniciar con un `fetch` o `pull` para obtener los objetos y referencias del repositorio remoto (esto si creamos el repositorio con los archivos de licencia y `README.md` que nos invita a crear el servicio git), y hacer la unión del contenido existente en el remoto con el local justo en el local.

```
git pull pruebas master
```

El paso anterior se evita si solo usamos el servicio para la creación del repositorio remoto vacio en lugar de poblarlo con los asistentes de archivo de licencia y archivo de descripción. Si en el paso anterior los archivos del repositorio remoto y el local poseen los mismos nombres pero son diferentes, pueden existir conflictos que deban resolverse de forma manual.

Crearemos archivos de prueba la adición al repositorio.

```
In [9]: %%bash
        cd herramientas
        echo "linea 1" > probar_1
        printf "s\n 5" > probar_2
```

Aunque en este caso es claro que nuestros archivos no se encuentran agregados al repositorio, consultaremos por el estado de los archivos frente a su registro en el repositorio.

```
In [10]: %%bash
         cd herramientas
         git status
```

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

        probar_1
```

```
        probar_2

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

Ahora agregamos al registro los archivos de nuestro repositorio local.

```
In [11]: %%bash
         cd herramientas
         git add . # Con la indicación "." agrega todo.
         git commit -m "Agregados archivos de prueba 1 y 2."

[master d193ae6] Agregados archivos de prueba 1 y 2.
 2 files changed, 3 insertions(+)
 create mode 100644 probar_1
 create mode 100644 probar_2
```

Realizaremos una modificación en uno de los archivos y agregaremos directamente a este.

```
In [12]: %%bash
         cd herramientas
         echo "linea 2" >> probar_1
         printf "otro\narchivo\npara probar." > probar_3
         git add probar_1 probar_3
         git commit -m "Actualizado prueba 1 y agregado prueba 3."

[master 412aeb3] Actualizado prueba 1 y agregado prueba 3.
 2 files changed, 4 insertions(+)
 create mode 100644 probar_3
```

Es posible indicar directamente a git la eliminación de archivos mediante

```
git rm archivo
```

o tambien al eliminar el archivo de manera directa, se incluye en git

```
git add -A
```

con el fin de actualizar el arbol de archivos y rastrear incluso los archivos eliminados. Sin embargo, es recomendable que la adición y remoción de archivos sea lo más especifica posible con el fin de tener un registro descriptivo que ayude a la adecuada solución de problemas o la descripción de cambios de una versión a otro de los códigos. Tengase en cuenta que en el caso de directorios, debe indicarse el argumento recursivo para el borrado.

Si se desea eliminar un archivo del repositorio pero no del disco, debe eliminarse de la *cache* del repositorio.

```
git rm --cached archivo
```

Posterior a esto, realizamos el *commit* asociado.

1.10 Actualizar de/desde repositorios remotos

Ahora deseamos actualizar nuestro repositorio remoto con nuestro repositorio local, para lo cual debemos hacer un *push* con el nombre del repositorio remoto y el nombre del *branch* que por defecto si solo hay uno es *master*. Es posible configurar el *push* para que su acción sea la de enviar todos los *branchs* del repositorio pero es preferible un control más fino con el fin de reducir el trafico de datos en la red.

```
In [13]: %%bash
         cd herramientas
         git push pruebas master
```

```
To https://github.com/cosmoscalibur/pruebas.git
 * [new branch]      master -> master
```

Durante la ejecución del *push* se nos solicitará el usuario y contraseña de nuestra cuenta en el servicio git. Al usar el protocolo https por defecto tendremos 3600 segundos tras cada autenticación, o si lo preferimos podemos usar ssh y crear la llave de autenticación con el fin de tener acceso permanente en nuestro equipo privado sin requerir de la autenticación continua.

Para la actualización de nuestro repositorio local con el repositorio remoto reaizamos una acción *pull*, tal cual se indico en la subsección anterior para actualizar el repositorio local con los archivos que posiblemente se crearon con el asistente del servicio git.

1.11 Otras acciones

Se recomienda la lectura de otras acciones de git importantes, las cuales son: `branch` y `merge` para experimentar con modificaciones de un código base y su posterior unión al original, o `log`, `show` para la visualización del historial de modificaciones.
Si particularmente queremos ver el estado del repositorio en un *commit* especifico, usamos `checkout` seguido del *commit*. Esta acción no genera ninguna modificación del repositorio y para continuar en el último estadosolo debemos volver a nuestro estado actual realizando el *checkout* al *master*. Si deseamos conservar este estado consultado realizamos una reversión al *commit* dado con `reverse`.
Si la acción anterior se realiza sobre un archivo, los cambios deben deshacerse con *checkout* a `HEAD` seguido del archivo, de lo contrario el estado del archivo consultado estará como una modificación.
Igualmente existen acciones de interes en caso de usar los servicios git como la acción del `fork` que hace uso de la solicitud de *pull* (acciones no incluidas en git pero que pueden ser usadas con clientes basados en las API de los servicios git).

```
In [14]: %%bash
         cd herramientas
         git log --oneline
```

```
412aeb3 Actualizado prueba 1 y agregado prueba 3.
d193ae6 Agregados archivos de prueba 1 y 2.
261c014 Notas de clase sobre git.
2f875d7 Agregados elementos iniciales.
```

```
c6c3472 Actualización del filtro para exclusión de archivos del repositorio. Error
3d7ddbd Mejora de redacción punto 6. Redacción original implicaba algo que no es po
b8175d2 Removido archivo auxiliar anterior a la adición del filtro en el .gitignore
359315e Corrección nombre de archivo de imagen.
bfb1c4e Agregado pdf de referencia para el taller 1 en el punto de LaTeX.
1880047 Ampliación de notas de LaTeX. Versión mínima completa.
78d2f53 Se inician las notas de clase de introducción a LaTeX.
530f97d Incluido archivo .gitignore para exclusión de archivos auxiliares.
c0c677b Actualización de horario de asesoría
eafa1a1 Corrección de mención de vector a polinomio, y mejor descripción del punto
c7457f7 Agregado resumen de clase de scripting en bash en formato ipynb y pdf.
2c2a162 Merge branch 'master' of https://github.com/cosmoscalibur/herramientas_comp
0983215 Agregada descripción del repositorio.
18ea606 Agregada Licencia MIT sobre el código existente en el repositorio.
a7d666e Agregado primer taller evaluativo sobre Linux, Jupyter y LaTeX.
6fadef3 Agregadas notas de clase sobre Linux básico en formato ipynb y pdf.
6f9da7a Agregadas notas de clase sobre Jupyter Notebooks en formato ipynb y pdf.
6149aaa Agregada presentación del curso.
e4a4be5 Extendido README.md
06cf9e4 Initial commit
```

In [15]:
```
%%bash
cd herramientas
git show d193ae6
```

```
commit d193ae6a32c2b2992a0e7e05b7f8f2f936a654eb
Author: Edward Villegas <cosmoscalibur@gmail.com>
Date:   Sun Aug 21 19:51:56 2016 -0500

    Agregados archivos de prueba 1 y 2.

diff --git a/probar_1 b/probar_1
new file mode 100644
index 0000000..2916432
--- /dev/null
+++ b/probar_1
@@ -0,0 +1 @@
+linea 1
diff --git a/probar_2 b/probar_2
new file mode 100644
index 0000000..198adde
--- /dev/null
+++ b/probar_2
@@ -0,0 +1,2 @@
+s
+ 5
\ No newline at end of file
```

In [16]:
```
%%bash
cd herramientas
echo "Actual"
ls
git checkout e4a4be5
echo "Anterior"
ls
git checkout master
```

```
Actual
Git.ipynb
Jupyter Notebook Basico.ipynb
Jupyter+Notebook+Basico.pdf
Jupyter Notebook Intermedio.ipynb
Jupyter+Notebook+Intermedio.pdf
jupyter.png
LaTeX_basico.pdf
LaTeX_basico.tex
LICENSE
Linux Bash.ipynb
Linux+Bash.pdf
Linux Basico.ipynb
Linux+Basico.pdf
minimo_taller1.pdf
presentacion_herramientas.md
presentacion_herramientas.pdf
probar_1
probar_2
probar_3
README.md
Taller_1.ipynb
Taller_1.pdf
Anterior
LICENSE
README.md


Note: checking out 'e4a4be5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

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

```
HEAD is now at e4a4be5... Extendido README.md
Previous HEAD position was e4a4be5... Extendido README.md
Switched to branch 'master'
```

```
In [17]: %%bash
         cd herramientas
         echo "Actual"
         cat probar_1
         git checkout d193ae6 probar_1
         echo "Anterior"
         cat probar_1
         git checkout HEAD probar_1
```

```
Actual
linea 1
linea 2
Anterior
linea 1
```

2 Referencias

1. Git Downloads. Git. Consultado el 19 de agosto de 2016.
2. BootCamp. Github. Consultado el 20 de agosto de 2016.
3. Git Doc. Git. Consultado el 20 de agosto de 2016.
4. Git Tutorials. Atlassian. Consultado el 20 de agosto de 2016.
5. Version control with git. Software Carpentry. Consultado el 20 de agosto de 2016.
6. Managing remotes. Github. Consultado el 21 de agosto de 2016.