AEET
ASOCIACIÓN ESPAÑOLA DE ECOLOGÍA TERRESTRE

Ecosistemas 27(2): 140-141 [Mayo-Agosto 2018]
Doi.: 10.7818/ECOS.1604

NOTAS ECOINFORMÁTICAS

ecosistemas
REVISTA CIENTÍFICA DE ECOLOGÍA Y MEDIO AMBIENTE

ISSN 1697-2473 / Open access
disponible en www.revistaecosistemas.net

¿Por qué usar GitHub? Diez pasos para disfrutar de GitHub y no morir en el intento

J. Galeano[1]*

(1) (1) Grupo de Sistemas Complejos, ETSIAAB, Universidad Politécnica de Madrid, España.

* Autor de correspondencia: J. Galeano [javier.galeano@upm.es]

> Recibido el 11 de julio de 2018 - Aceptado el 11 de julio de 2018

Galeano, J. 2017. ¿Por qué usar GitHub? Diez pasos para disfrutar de GitHub y no morir en el intento. *Ecosistemas* 27(2): 140-141. Doi.: 10.7818/ECOS.2017.26-2.08

¿Necesito usar Git y GitHub?

Si para escribir un texto académico o un código de análisis estadístico acabas con varias versiones numeradas v1, *v2, v2*final, v2final2, v2labuena, … puede ser una buena idea usar alguna herramienta de control de versiones. Git (https://git-scm.com/) es una herramienta muy popular para controlar las versiones de archivos (principalmente de texto) en tu ordenador local. Si además colaboras con colegas en proyectos conjuntos, tener un repositorio compartido y unificado en GitHub (https://github.com/) puede hacer tu trabajo más fácil y eficiente.

Pero si todo son ventajas, ¿por qué no se usan más? El principal problema es que la curva inicial de aprendizaje es dura haciendo que mucha gente abandone, sobre todo si se comienza con git. Algunos autores proponen comenzar usando GitHub con un entorno más amigable.

Diez pasos para comenzar a usar GitHub

El paso 0 es abrir una cuenta en GitHub (https://github.com/join). Una vez que ya tenemos esa cuenta creada:

Paso 1. Crear un repositorio. Un repositorio es un directorio donde van a estar los archivos que git puede monitorear. Para crear un repositorio entramos en nuestra cuenta de GitHub y vamos a la pestaña `Repositories` y clicamos en `**New**`. Ponemos el nombre del repositorio, *poema arco iris*, pero a GitHub no le gustan los espacios así que renombrará a *poema-arco-iris*. Añadimos una descripción *Mi poema sobre el arco iris*. El repositorio puede ser público o privado (en este caso activo el botón de *Public*) y además es recomendable que al crear el repositorio se añada un fichero *Readme*, de no hacerlo se complica todo mucho más. En este fichero se copiará el título del 'repo' y la descripción. Es el momento de darle al botón **Create repository**. Tendremos nuestro primer repositorio con el fichero README.md.

Paso 2. Primer fichero, el poema del Arco Iris. Escribamos nuestro primer fichero. Clicamos en el botón, **Create new file** y aparecerá un sencillo editor donde escribiremos nuestro texto. Ponemos el título, *poema_arco_iris.txt*. Nuestro texto es un poema sobre el arco iris y vamos a dejarlo para seguir más tarde. En git y GitHub el equivalente a grabar un fichero se llama **commit**. Si bajamos en la página vemos el botón **Commit new file**. Inicialmente está deshabilitado, pero si escribimos en el campo debajo del título **Commit new file**, el botón se activará. Escribimos un texto explicativo, por ejemplo: `Comienzo el poema` y apretamos el botón. En la carpeta de repositorio aparecerá un nuevo fichero, `poema_arco_iris.txt` y, debajo del título, dos 'commits'. Debajo del botón master, sale el usuario que ha modificado el fichero, el texto del último 'commit' y un montón de letras y números. Git usa un identificador único para cada 'commit', un número hexadecimal de 40 dígitos.

Paso 3. Veamos la historia. Nos volvió la inspiración y editamos el fichero añadiendo los colores del arcoiris. Hacemos un nuevo 'commit' y con el fichero del poema abierto, nos fijamos en el botón **History**: si clicamos aparece la lista de 'commits'. En este caso hay varios 'commits'. Es probable que hasta ahora no nos hayamos dado cuenta de la ventaja de usar GitHub, pero al ver todas las versiones de nuestro fichero recogidas y organizadas en una lista caigamos en la cuenta. Pero esto no es lo mejor, si clicamos en el último 'commit' aparecerá una imagen marcada en verde con lo que hemos añadido con un signo + delante de la línea. En rojo, la línea modificada; en verde claro, lo que ya estaba y en verde más oscuro, lo que añadimos nuevo. Tenemos todos los cambios realizados en esta versión y marcados con colores.

Paso 4. .gitignore y license. Cuando creamos el primer repositorio teníamos dos botones que ignoramos: **Add .gitignore** y **Add a license**. El botón de `license`, nos añade un fichero con la licencia de tipo Creative Commons. El botón de **.gitignore** añade un fichero donde podemos especificarle qué tipo de ficheros debe ignorar.

Paso 5. Creando una nueva rama. Queremos hacer una prueba experimental, es el momento de hacer una nueva rama. Si nos situamos en el repositorio **poema-arco-iris,** podemos ver un botón en el que pone **Branch:master**. Por defecto la rama principal es master, pero si queremos trabajar en una nueva, clicamos en el botón y aparecerá un campo que pone **create or find a new branch**. Voy a introducir animales mitológicos en el poema así que la llamaré `mitos`. Aparecerá un botón **create new branch**, clicamos. Parece que nada ha cambiado, pero en el botón de branch ahora pone **Branch:mitos**. Edito el fichero del `poema_arco_iris.txt` y añado el basajaun y la basandere. Hago el 'commit' y vemos que

está activado el botón **commit directly to the mitos branch**. Estoy añadiendo mi modificación sobre la rama de mitos y no sobre la rama master, que permanece intacta. Si volvemos a la página del repositorio ha aparecido un nuevo botón **compare and pull request**.

Paso 6. Compare and pull request. Veamos qué ha pasado con el repositorio. En la parte superior vemos el botón **insights**, entramos y a la izquierda vemos **network**. Hay una línea horizontal negra, con la etiqueta *master* y debajo una flecha que pone *mitos*. Son las dos ramas de nuestro repositorio vistas gráficamente. En la página principal sigue nuestro botón **compare and pull request** y el botón de branch ha vuelto sobre master, **Branch: master**. Si nos fijamos en el fichero del poema, como estamos en la rama master, no aparecen los mitos. Me gusta el nuevo poema con los mitos vascos y los quiero incorporar a la rama master. Clicamos en **compare and pull request**. Me sale una ventana **open pull request**. Rellenamos títulos, los campos necesarios y le damos a **create pull request**. GitHub comprueba automáticamente si hay algún problema al unir los ficheros, y si no hay ningún conflicto: **This branch has no conflicts with the base branch**. Le damos al botón **merge pull request**, confirmamos y nos da la posibilidad de borrar la rama de mitos. Volvemos a **network** y vemos que la rama *mitos* y la rama *master* se han juntado en un punto.

Paso 7. Fork. Los 'forks' nos sirven para hacer una copia de repositorios ajenos. Busquemos por GitHub algún repositorio que nos interese. En la parte derecha superior veremos un botón de **Fork**. Tras pulsarlo vamos a tener un repositorio igual a éste en nuestra cuenta de GitHub, sobre el que podemos hacer cuantos cambios queramos.

Paso 8. Pull request. Supongamos que el repositorio copiado con 'fork' es un código colaborativo sobre el que estamos trabajando con nuestros colegas. Mejoramos el código y hacemos el cambio en nuestro repositorio. Podría ser interesante enviarlo al repositorio original, pero nosotros no tenemos los permisos para hacerlo, por lo que tenemos que pedir un **Pull request**. El proceso es similar a lo que hicimos en el paso 6. Esta es la base de trabajo en un proyecto colaborativo.

Paso 9. Intro a línea de comandos: git. Normalmente se trabaja en local en nuestro ordenador, y una vez realizados los cambios los subimos a GitHub. Para ello tenemos que hacer 'commits', pero en local, y luego subir los cambios a GitHub. Una vez instalado git en tu ordenador (http://happygitwithr.com/install-git.html), lo primero es conectar nuestro git local con nuestra cuenta de GitHub, para ello escribimos en el terminal estos dos comandos:

```
git config –global user.name "User Name"
git config –global user.email email_usado_en_GitHub
```

Luego escribimos `git config –list` y deberíamos ver nuestro nombre y el email.

Paso 10. Clonar un Repo, push and pull. Queremos trabajar en nuestro repositorio en local para luego subirlo a GitHub. Abrimos nuestro terminal y nos vamos al directorio donde queremos poner nuestro repositorio. Tenemos que clonar el repositorio de GitHub para trabajar en local. En el repositorio de GitHub veremos el botón **clone or download**, hacemos click y aparece la direccción https, copiamos, vamos al terminal y escribimos, `git clone dirección_url`, en mi caso, `git clone https://github.com/galeanojav/poema-arco-iris.git`. Ya podemos abrir desde nuestro ordenador, en el nuevo directorio creado, nuestro poema.txt, lo editamos y grabamos. Si escribimos en el terminal `git status` veremos que nuestro fichero se ha modificado. Para hacer el 'commit', escribimos en el terminal `git commit -a -m "Primer commit en local"` y tendremos nuestro fichero guardado. Si queremos subir los cambios desde nuestro ordenador a GitHub usamos `git push`. Probemos a hacer `git push` y nuestro cambio aparecerá en GitHub con el nuevo 'commit' añadido en local. Para bajar a nuestro ordenador los cambios subidos por otros colaboradores al mismo repo usaremos `git pull`.

En muchos casos será útil integrar Github y Rstudio (ver http://happygitwithr.com). Y puedes solicitar cuentas privadas gratuitas si eres estudiante o investigador (https://education.github.com/). Para saber más puedes consultar https://github.com ecoinfAEET/Reproducibilidad/blob/master/Recursos.Rmd#control-de-versiones-git—github.

Agradecimientos

Me gustaría agradecer al grupo de ecoinformática de la AEET la posibilidad de enviar esta nota y sus comentarios.