2021-10-20

Anton-Bravo, Adolfo

Contents

1 Contenidos

1.1 Git y Github

1.1.1 El repositorio de pontedatos

- Ya tenemos un repositorio en la organización a la que pertenecemos como clase, `https://github.com/pontedatos/uc3m-periodismo-datos`

- Lo que vamos a hacer es o bien descargarlo si no lo hemos hecho o bien actualizarlo si ya lo habíamos descargado.

- Cuando digo "descargarlo" en realidad digo "clonarlo", es decir, voy a la carpeta de mi elección con el comando `cd` y escribo `git clone https://github.com/pontedatos/uc3m-periodismo-datos.git`.

- Esto creará una carpeta por debajo de donde me encuentro con el nombre `uc3m-periodismo-datos`. Esa carpeta es un repositorio git en nuestro ordenador, en `localhost`, que tiene el mismo contenido que el de la dirección desde donde me lo he clonado.

- Si ya hubiera hecho esto alguna vez tendría que ponerme dentro de la carpeta y actualizar con `git pull`.

1.1.2 Nuestro repositorio

- Es probable que tengáis un repositorio vuestro cuyos contenidos solo estén en github.

- Vamos a descargarnos los datos de la misma manera que antes.

- Nos situamos en la carpeta elegida y clonamos con `git clone` seguido de la dirección de nuestro repositorio git en github.

- Luego con `cd` nos situamos dentro del repositorio.

- Ahora con `nano` o nuestro editor favorito cambiamos algo, guardamos, cerramos y seguimos tres pasos para actualizar los cambios en github:

  - `git add nombre-archivo-cambiado`

  - `git commit -m "comentario del cambio"`

  - `git push main origin`

- Si al dar `git push` da error consultamos el error por el foro.

1.1.3 No tenemos un repositorio

- Si no tuviéramos un repositorio podríamos crear uno nuevo.

- Primero creamos una carpeta con el nombre que queramos aunque lo suyo es que sea el nombre del repositorio.

- Luego con `cd` nos situamos dentro de la carpeta.

- Y entonces creamos un repositorio en Github y, cuando llegue el caso, seguimos en la terminal estos pasos que indican:

```
echo "# Proyecto de ..." >> README.md
git init
git add README.md
git commit -m "primer commit"
git remote add origin https://github.com/cuenta/nombre-repositorio
git push -u origin main
```

1.1.4 No nos reconoce

- Comenta una compañera que tiene problemas con `git` porque en el paso `git commit -m` le sale un aviso que no logra interpretar.

- Una recomendación previa y para que lo tengáis en cuenta: conviene leer los mensajes que da el software e intentar interpretarlos adecuadamente. A veces nos "resuelven" el problema. Entiendo que "desconciertan" pero también hay que acostumbrarse a leerlos, a buscar soluciones e intentar resolver problemas/inconvenientes que surjan. En este caso, además, la compañera ha compartido una captura de pantalla del aviso y de esa manera ha sido fácil resolverlo, os lo cuento.

- Spoiler: lo que dice se resuelve en esta sección de este manual que os enlacé con mis notas de git y github:

https://flowsta.github.io/github/#outline-container-org19f011c

- Os lo explico paso a paso. La primera vez que usamos en el ordenador `git` de la manera que os propuse, es decir, no desde la interfaz de Github sino desde vuestro ordenador, a través de la terminal, tenemos que identificarnos.

- En este caso lo que dice es `Author identity unknown`, es decir, se desconoce la identidad de quién eres para `git`, para firmar ese `commit` y luego subirlo al servidor que vas a utilizar.

- Y además añade `*** Por favor, cuéntame quién eres`

- A continuación dice que corras (que ejecutes) dos líneas de instrucciones en la terminal. Tened en cuenta que cuando salen varias líneas en este tipo de avisos, como cuando instalamos `apt-cyg`, no se copian las dos líneas juntas sino que se pone primero una, se ejecuta, y luego la otra.

- En este caso, lo que dice primero es que les digas cuál es tu dirección de correo electrónico. En este caso, como usamos como servidor externo a Github, ponemos el correo electrónico con el que nos hemos registrado en Github: `git config --global user.email "tu-email-en-github"`

- En esta instrucción, la opción `--global` indica a `git` que siempre vas a usarlo en la terminal con este mismo `user.email`. Por tanto, no tendrás que volver a configurar esto.

- La siguiente línea solicita que le digas cuál el nombre de tu cuenta `git config --global user.name "cuenta-en-github"`

- A partir de ahí podréis seguir hasta el próximo desafío.

1.1.5 :new: Tokens

- Para anticiparos el próximo "desafío" con que os vais a encontrar os cuento que Github cambió hace poco la forma de relacionarse con Github y ahora hay que generar una clave para subir los contenidos, para hacer el `git push`.

- Antes había que poner la contraseña de github, ahora hay que generar una clave.

- Esto se hace yendo a `https://github.com/settings/tokens` y generando un nuevo `token`.

- Hay que darle un nombre, elegir una fecha de caducidad (puede ser "nunca" aunque, ya que estáis aprendiendo, mejor ponerle una caducidad para que tengáis que volver a ese paso en algún momento) y seleccionar un ámbito de actuación o "scopes".

- Para lo que vamos a hacer basta con que marquéis "repo" y los que cuelgan de él:

  - repo
  - repo:status
  - repo_deployment
  - public_repo
  - repo:invite
  - security_events

- Una vez marcado esto y creado el token os genera un "hash", un código que conviene que copiéis en alguna parte –:warning: cuidado: no en los apuntes públicos– y que será el que tengáis que poner

cuando hagáis `git push` en vez de la contraseña.

1.2 Enlaces

- Viejo manual de git y Github: `https://flowsta.github.io/github/`
- Recordad mis notas sobre Markdown en `https://flowsta.github.io/markdown`
- Editores para Markdown:
  - MarkDownEditor, W$, `https://github.com/chenguanzhou/MarkDownEditor`
  - Abricotine, W$-OSX-GX, `https://abricotine.brrd.fr/`
  - MarkText, W$-OSX-GX, `https://marktext.app/`
  - Ghostwriter, W$-OSX-GX, `https://wereturtle.github.io/ghostwriter/`
  - Atom, GX, `https://atom.io/`
  - VSCodium, W$-OSX-GX, `https://vscodium.com/`
  - Remarkable, W$-GX, `https://remarkableapp.github.io/`
  - Haroopad, W$-OSX-GX, `http://pad.haroopress.com/user.html`
  - Emacs, W$-OSX-GX, `https://www.gnu.org/software/emacs/`
  - nano, CLI, `https://www.nano-editor.org/`
- Data Science at the Command Line `https://www.datascienceatthecommandline.com/`

1.3 Recordatorios

- Se puede solicitar tutoría para tratar temas del curso.

- Quien tiene problemas con ruby puede probar con la vía propuesta en la sesión anterior y contar cómo ha ido.

- Quien no tenga Cygwin configurado que repase los apuntes. Conviene tener instalado `apt-cyg` y el cambio de la `home`.

- Quien elija `nano` puede visitar su página web por si encuentra información útil: `https://www.nano-editor.org/`.

1.4 ToDo

- Estilo de la terminal

- Estilo de PS1

- Resaltado de sintaxis en `nano`.

- Atajos en `nano`

- Personalización en `nano`

2 Pruebas

- Explica los pasos para clonar en tu ordenador un repositorio de Github.

- Explica los pasos para crear un repositorio en Github y conectarlo con un repositorio local.

- Realiza algún cambio en tu repositorio local y actualízalo también en Github. Explica los pasos.

- Realiza un comentario de una o varias visualizaciones de datos y/o infografías que queráis y justificar las respuestas y la elección.