CURSO GIT-HUB AZTI 2019

NOTAS DE REFRESCO ANTES DEL CURSO

Para descargarse git:

<https://git-scm.com/downloads>

**Parte 1. Cómo subir un archive a GitHub**

git config --global core.editor "'c:/Users/gboyra/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

alias notepad="c:/Users/gboyra/Notepad++/notepad++.exe"

# Ejercicio: enviar un archivo a GitHub

# se supone que tenemos ya configurada una cuenta GitHub

# en la cual disponemos del repositorio "datasciencecoursera"

touch HelloWorld.md

# abrimos el archivo y escribimos: "## This is a markdown document"

# ahora creamos un repositorio git

# necesitamos hacerlo para poder "empujarlo" al remoto

# (parece que necesitamos empujar repositorios "iniciados")

git init

git add HelloWorld.md

git status

git commit -m "initial version of the project"

# ahora definimos nuestro repositorio remoto:

git remote add origin <https://github.com/gboyra/datasciencecoursera.git>

# le hemos dicho que nuestro repo remoto se llama origin

# y que se encuentra en la dirección ip: <https://github.com/gboyra/datasciencecoursera.git>

# la dirección IP se puede copiar directamente en githup (Copy to clipboard)

# para comprobar si ha funcionado se puede hacer:

git remote -v

# te debería devolver la dirección del repositorio remoto

# Una vez definido el repositorio, tratamos de mandar el repositorio local (llamado master)

# al repositorio remoto (llamado origin)

# si hacemos pull, enviamos de origin >> master

# si hacemos push, enviamos de master >> origin

git push origin master

# te pide el usuario y contraseña

# la primera vez no me funcionó pq los repositorios eran diferentes,

# así que intenté:

git pull origin master

# esto sí me funcionó, así que ahora volví a repetir:

git push origin master

# y vualá!

**Parte 2. Intro rapida a GIT (del Software carpentry 2015)**

Summary: git, init, config, add, commit, status, diff, log

CONFIGURACIÓN

git config --global core.editor "'c:/Users/gboyra/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

alias notepad="c:/Users/gboyra/Notepad++/notepad++.exe"

Git config--global user.email user.name

Git

Git help # generic help

Git help commit # specific help

Repository: a directory under version control

Git init

Hint: don't put repositories inside repositories

Inside the repository, start tracking files

Git add names.txt # to stage the files

(git reset names.txt # to unstage them)

Commit changes (create a new version)

Git commit -m "initial version of the project"

(If you don't put the message, git will open your default text editor so you can type the message)

See all the history of changes

Git log

Edit files and test the changes

...

Stage the files

Git add whatever.txt

Check

Git status

Commit the new version

Git commit -m "I add a blablabla sentence to the names.txt file"

Check

Git status

Ejercicio

Creamos un archivo script.txt

Touch script.txt

Editamos el archivo: wc names.txt | sort -n

Vemos lo que lleva escrito

Cat script.txt

Lo ejecutamos:

Bash script.txt

Añadimos el archivo a stage

Git add script.txt

Check

Git status

New version

Git commit -m "we add a new file script.txt with a script inside"

Git log

(lots of options, check git help log)

VER LAS DIFERENCIAS

Editamos uno de los scritps

Para ver las diferencias entre el editado y el último commit:

Git diff script.txt # IMP: hacerlo antes de git add

Git status

Git add script.txt

Git status

Git commit -m "I add a line at the end of script.txt"

Git status

DESHACER CAMBIOS

Revert (in RStudio) – undo uncommited changes

Go to Git-Tab > History > Revert > Yes

Checkout – undo committed changes  
 git checkout <SHA.number> <file.name>

The SHA number is obtained in the History:

Clich in the previous commit to the one you want to undo and

Copy its SHA number

BRANCHES

Al hacer git init crea una Master Branch

Se puede hacer un test branch

Git branch new.branch

Check the current branch

Git branch # lists all the branches you have

Moverse a la nueva branch

Git checkout new.branch

MERGE BRANCHES

Switch INTO the master branch

Git checkout master

Merge FROM the testing branch

Git merge testing # testing is the branch\_from where you merge into master

If there is a conflict you solve it manually by opening in an editor

Git will have set flags to identify the conflicts

Delete the new branch

Git branch -d new.branch

Git log --oneline

Mirar el archivo galaxies.txt en un estado anterior (definido por el sha.number listado por log)

Git show sha.number galaxies.txt

Entrar en un branch anterior para hacer cambios en él

Git chechout a1b567

SURPRISE

Create a new surprise branch

Create a new file stars.txt

Check the file (ls) in master

Check the file (ls) in surprise

Up to now is visible in all branches

Switch to the master branch

Stage and commit

Now it's only visible in the committed branch

If we now try to delete the surprise branch git won't allow us

COLLABORATION

Branches... or people

Bitbucket.com free private repositories

Add a remote location

Git add origin ip.address

Send data

Git push -u origin master

# it pushes all the files in the repository (I think)

Receive data

Git pull origin master

**Parte 3. Unix shell esquema (del Software carpentry 2015)**

POSIX minimum standard

Ls is a program?

# activar el notepad como editor del shell

alias notepad="c:/Users/gboyra/Notepad++/notepad++.exe"

Shell commands:

Ls # mostrar los archivos y carpetas

Pwd # print working directory

Cd # change directory

Cd .. # directorio superior

Cd . # directorio actual - puede ser util dentro de un script

whoamI # devuelve el usuario

Ls -F # the minus is a "flag"

# pinta una barra cuando es un directorio

Ls -l # proporciona las fechas de modificación de los archivos

# y tb te dice si los archivos se pueden cambiar o leer, y por quién

# mirar en el help

Ls nombre.dir # proporciona los archivos dentro del directorio

Ls -l nombre.dir # combina flag y directorio

Ls -l -F # combina flags

Ls - a # "all" muestra todo el contenido (incluidos ocultos)

TAB-completion (como en R-Studio)

(hay un montón de programas que no muestran los archivos que empiezan con un "." en el nombre)

Mirar algunos comandos: which, open, xdg-open

Mkdir # crear un nuevo directorio

Rmdir # remove directory

Nano # editor de texto (no funciona en windows)

Vim # editor por defecto de git bash

# probarlo

# puto lío

**Open** archivo.txt (mucho más sencillo)

Ayuda:

Touch --help

Mkdir --help

Echo "Hello world"

Cat # muestra el contenido de un archivo de texto en el shell

Clear # limpia el histórico

CREATING THINGS

Cp # copia un archivo en otro

cp /users/myfile.txt . # copia el archivo al directorio actual

Cp -r # (recursive) copia todos los archivos de un directorio

Mv # mueve un archivo en otro (cambia de nombre)

mv

Rm # borra un archivo para siempre

~ se interpreta como el home directory del currrent user

Touch # creates an empty file

Date # prints the date

PIPES AND FILTERS

Wc \*.pdb # "word count" command; cuenta líneas, palabras y caracteres

# por ese orden

# de todos los archivos con ext pdb

Wc -l \*.pdb # solo muestra el número de líneas (-w -c)

Wc -l \*.pdb > length.txt # copia el output en el archivo length.txt

# es un comando destructivo, cada vez que lo haces borra el

# contenido original y lo reemplaza por el nuevo

>> mueve añadiendo en vez de reeplazando

Sort length.txt -n # ordena numéricamente

Sort length.txt -a # ordena alfabéticamente

Sort -k2 length.txt # ordena a partir del segundo carácter

# mirar el help: sort --help

Head length.txt # prints the start of the file

Head length.txt -2 # prints the start of the file until the second line

Tile length.txt # prints the end of the file (no funciona en windows)

Nested commands:

# (el resultado del anterior se usa como input del siguiente…)

Wc -l \*.pdb | sort -n | head -3

Wc < length.txt # the reverse of >

Bash script.txt # ejecuta los scripts que tengas escritos en el archivo

**Explain.shell.com** # explica lo que hace un script de forma gráfica