Curso de Git, GitHub y SourceTree
Ejercicio Tema 2

Ejercicio

En este ejercicio veremos la configuración de la identidad en Git y algunos comandos que nos pueden ayudar con la configuración de la herramienta y a conocer más opciones. La entrega se realizará mediante capturas de los procesos y comandos siguientes y subiendo posteriormente a la plataforma un documento de texto con una pequeña explicación del comando junto con su captura.

Lo primero que vamos a hacer cuando instalemos Git es establecer nuestro nombre de usuario y dirección de correo electrónico.

Esto es de vital importancia porque cada commit de Git usa esta información, y está inmutablemente integrada en los commits que comencemos a crear

Para ello hay que ejecutar los siguientes comandos que modificaban el archivo que vimos en la teoría:

git config --global user.name "NombreDeUsuario"

git config --global user.email email@ejemplo.com

```
C:\WINDOWS\system32>git config --global user.name "Raul"

C:\WINDOWS\system32>git config --global user.email raul@imaginagroup.com
```

Debemos hacer esto solo una vez si se pasa la opción --global, porque entonces Git siempre usará esa información para cualquier cosa que hagamos en nuestro sistema.

Si deseamos sobreescribir esto con un nombre o dirección de correo electrónico diferente para proyectos específicos, podemos ejecutar el comando sin la opción **--global** cuando se encuentre en ese proyecto.

Ahora que nuestra identidad está configurada, podemos configurar el editor de texto predeterminado que usaremos cuando Git necesite que escribamos un mensaje. Si no está configurado, Git usa el editor predeterminado de nuestro sistema.

Si deseamos utilizar un editor de texto diferente, como Emacs, podemos hacer lo siguiente:

git config --global core.editor emacs

En un sistema Windows, si deseamos utilizar un editor de texto diferente, debemos especificar la ruta completa a nuestro archivo ejecutable. Esto puede ser diferente dependiendo de cómo esté empaquetado nuestro editor.

En el caso de Notepad ++, un popular editor de código, es probable que queramos utilizar la versión de 32 bits, ya que en el momento de escribir esta versión de 64 bits no es compatible con todos los complementos. Si estamos en un sistema Windows de 32 bits o tenemos un editor de 64 bits en un sistema de 64 bits, introducimos algo como esto:

git config --global core.editor "'C:Ruta de Notepad++.exe' -multiInst -nosession"

Si tenemos un editor de 32 bits en un sistema de 64 bits, el programa se instalará en C: \ Archivos de programa (x86):

$ git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -nosession"

Nota: Vim, Emacs y Notepad ++ son editores de texto populares que los desarrolladores utilizan a menudo en sistemas basados en Unix como Linux y macOS o un sistema Windows.

Podemos verificar nuestras configuraciones, podemos usar el comando git **config --list** para listar todas las configuraciones que Git puede encontrar:

```
C:\WINDOWS\system32>git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.email=raul@imaginagroup.com
user.name=Raul
core.editor='C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession
```

Aquí podemos ver las que acabamos de realizar.

```
user.email=raul@imaginagroup.com
user.name=Raul
core.editor='C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession
```

Puede que veamos claves más de una vez, porque Git lee la misma clave de diferentes archivos (/etc/ gitconfig y ~ / .gitconfig, por ejemplo). En este caso, Git usa el último valor para cada clave única que ve.

También podemos comprobar el valor de una clave específica con el comando:

git config <clave>

Por ejemplo:

```
C:\WINDOWS\system32>git config user.name
Raul
```

A continuación vamos a ver un comando que nos permite conocer más opciones de Git.

Si necesitamos ayuda mientras utilizamos Git, hay una forma equivalente de obtener la ayuda de la herramienta para cualquiera de los comandos de Git:

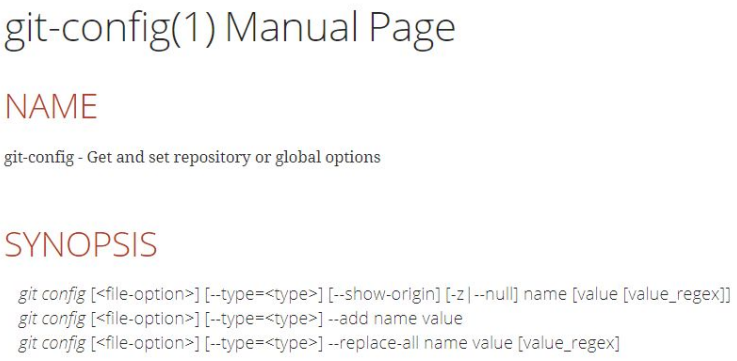
git help

```
C:\WINDOWS\system32>git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --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     Reapply commits on top of another base tip
   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.
```

Por ejemplo, podemos obtener la ayuda de la herramienta para el comando git config ejecutando:

git help config

Esto abrirá una página del manual oficial de git.



git-config(1) Manual Page

NAME

git-config - Get and set repository or global options

SYNOPSIS

git config [<file-option>] [--type=<type>] [--show-origin] [-z|--null] name [value [value_regex]]
git config [<file-option>] [--type=<type>] --add name value
git config [<file-option>] [--type=<type>] --replace-all name value [value_regex]

Estos comandos son muy utiles porque podemos acceder a ellos en cualquier lugar, incluso sin coexión.

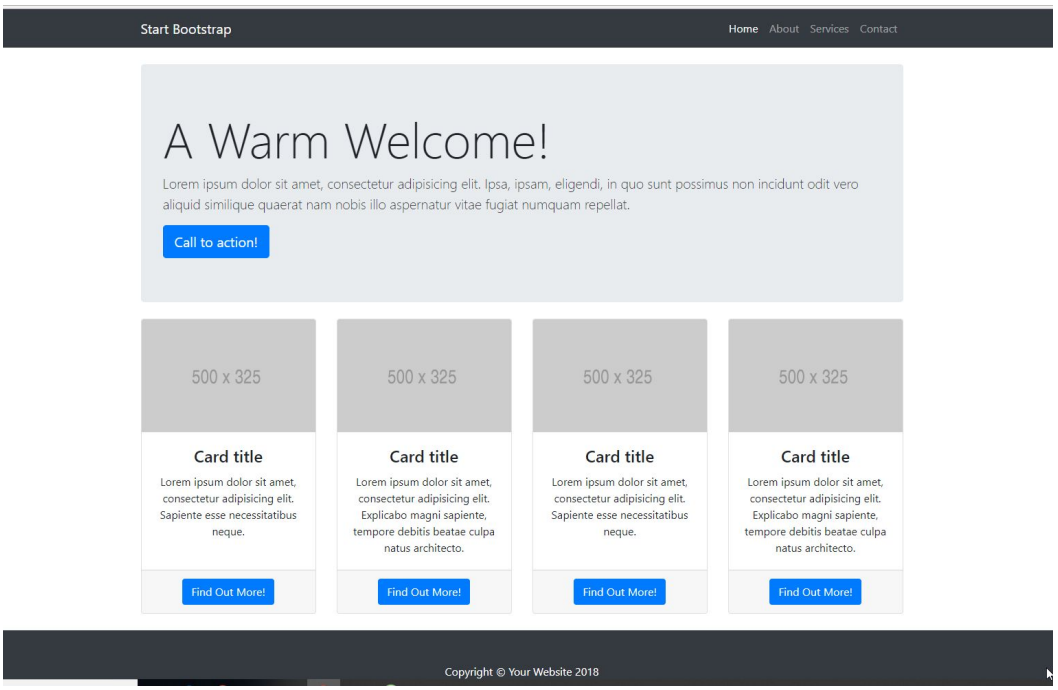
Además, si no necesitamos la ayuda completa, pero solo necesitamos una actualización rápida de las opciones disponibles para un comando de Git, podemos solicitar la salida de ayuda más concisa con -h o --help, como por ejemplo:

```
C:\WINDOWS\system32>git add -h
usage: git add [<options>] [--] <pathspec>...

    -n, --dry-run         dry run
    -v, --verbose         be verbose

    -i, --interactive     interactive picking
    -p, --patch           select hunks interactively
    -e, --edit            edit current diff and apply
    -f, --force           allow adding otherwise ignored files
    -u, --update          update tracked files
    --renormalize         renormalize EOL of tracked files (implies -u)
    -N, --intent-to-add   record only the fact that the path will be added later
    -A, --all             add changes from all tracked and untracked files
    --ignore-removal      ignore paths removed in the working tree (same as --no-all)
    --refresh             don't add, only refresh the index
    --ignore-errors       just skip files which cannot be added because of errors
    --ignore-missing      check if - even missing - files are ignored in dry run
    --chmod <(+/-)x>      override the executable bit of the listed files
```

Finalmente vamos a preparar el entorno sobre el que realizaremos cambios, trabajaremos con repositorios y la herramienta git durante el curso. Utilizaremos una plantilla de bootstrap html que representa la Home de una página web.

No será necesario tener conocimientos de los lenguajes web que se utilizarán ya que se proporcionarán todos los códigos y cambios necesarios durante los próximos ejercicios. La plantilla se encuentra en los recursos del tema en la plataforma.

La plantilla se compone de un archivo html donde se encuentra toda la estructura de la página, y de lo recursos como son los archivos CSS y los archivos necesarios para cargar la librería de estilos bootstrap.


Start Bootstrap
Home About Services Contact
A Warm Welcome!
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, ipsam, eligendi, in quo sunt possimus non incidunt odit vero aliquid similique quaerat nam nobis illo aspernatur vitae fugiat numquam repellat.
Call to action!
500 x 325
Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sapiente esse necessitatibus neque.
Find Out More!
500 x 325
Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo magni sapiente, tempore debitis beatae culpa natus architecto.
Find Out More!
500 x 325
Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sapiente esse necessitatibus neque.
Find Out More!
500 x 325
Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo magni sapiente, tempore debitis beatae culpa natus architecto.
Find Out More!
Copyright © Your Website 2018