01_introduccion_a_git

October 15, 2020

1 Introducción a *Git.*

1.1 Sistemas de gesión de versiones.

Un sistema de gestión de versiones es una herramienta que permite llevar el seguimiento y control de documentos y archivos dentro de un directorio específico, permitiendo identificar las diversas modificaciones que podría haber tenido cada elemento del directorio a lo largo del tiempo.

Los sistemas de gestión de versiones fueron ideados inicialmente para llevar control de código fuente de un proyecto de desarrollo de software, pero también pueden gestionar cualquier tipo de documento de texto e incluso documentos de medios y binarios.

En al siguiente liga es posible consultar un listado de programas para el control de versiones.

https://es.wikipedia.org/wiki/Programas_para_control_de_versiones

Nota: Aún cuando un gestor de versiones puede soportar prácticamente cualquier documento, no se recomienda que sea usado como repositorio de binarios, siendo quen existen otras herramientas, como es el caso de *Jfrog Artifactory*, que han sido diseñadas expresamente para tal fin.

1.2 Artefactos.

El término "artefacto" dentro del ámbito de desarrollo de software se refiere a todos aquellos documentos que sean resultado del proceso de desarrollo de software, como es el caso de:

- Código fuente.
- Documentación
- Bibliotecas de binarios.
- Paquetes de software.
- Imágenes de contenedores o máquinas virtuales.
- Archivos de configuración y especificaciones.
- Datos.
- "Blobs", los cuales correspoden a archivos en formato binario que no son necesariamente ejecutables.

1.3 *Git.*

Git es un sistema de gestión de versiones distribuido creado por Linus Torvalds con el propósito inicial de poder dar seguimiento al desarrollo del kernel de *Linux*. Actualmente *Git* es el gestor de versiones más popular entre la comunidad de desarrolladores.

Las principales caracterísiticas de *Git* son:

- De código abierto.
- Descentralizado.
- Distribuido.
- Rápido.
- Robusto.

1.3.1 Descarga de *Git*.

Git ha sido portado a prácticamente todas plataformas y sistemas operativos modernos, conformando parte de los paquetes básicos de un entorno de desarrollo. Sin embargo, también es posible descargar un instalador desde la siguiente liga:

https://git-scm.com/downloads

1.3.2 Clientes de *Git*.

Git puede ser ejecutado mediante un *cliente*.

El cliente más común de *Git* corresponde al comando `git` ejecutado desde la Interfaz de Línea de Comandos (*CLI*), sin embargo existe una gran cantidad de clientes que aprovechan una interfaz gráfica las cuales pueden ser consultadas en la siguiente liga:

https://git-scm.com/downloads/guis

Nota: Este curso está basado en el uso del comando `git` desde una *CLI* de *GNU/Linux*.

Git GUI. El paquete descargado del sitio oficial de *Git* incluye un cliente con una interfaz gráfica basada en *tcl/tk* llamado *Git GUI*.

La siguiente imagen corresponde a la interfaz de *Git GUI*.

Git Bash. El cliente *Git Bash* se utiliza primordialmente en sistemas basados en *Windows*. Este cliente crea una terminal corriendo la biblioteca *Ming-w64*, la cual es capaz de ejecutar no sólo el comando `git`, sino que crea un entorno que contiene al *shell Bash*. De este modo es posible tener una *CLI* muy similar a la usada en sistemas basados en *GNU/Linux*.

1.4 La documentación de *Git*.

El proyecto *Git* cuenta con una extensa documentacion, la cual puede ser consultada desde:

https://git-scm.com/doc

1.4.1 El libro *Pro Git*.

Entre la documentación del proyecto es posible acceder al libro *Pro Git*, escrito por Scott Chacon y Ben Straub, el cual está disponible en español desde:

https://git-scm.com/book/es/v2

1.5 El comando `git`.

El comando `git` permite realizar muy diversas operaciones con repositorios de *Git* mediante la siguiente sintaxis.

`git <verbo> <opciones y argumentos>`

Donde:

- `<verbo>` es un subcomando de `git`.

Ejemplos:

- El comando `git help` de la siguiente celda desplegará la ayuda de `git`.

[ ]:
```
git help
```

- El comando `git status` de la siguiente celda desplegará el estado del repositorio actual.

Nota: Esta notebook pertenece a un repositorio local clonado desde https://github.com/Cloudevel/cdpx-101.

[ ]:
```
git status
```

- La siguiente celda desplegará la versión de `git` que se tiene instalada en el sistema.

[ ]:
```
git version
```

1.6 Los repositorios de *Git*.

Un repositorio de *Git* no es otra cosa más que un directorio que incluye al subdirectorio `.git`. La información y configuración del repositorio se encuentra en dicho directorio.

Git permite tener repositorios locales, los cuales a su vez pueden acceder y sincronizarse con repositorios remotos.

Ejemplo:

- La siguiente celda mostrará el contenido del subdirectorio `.git` que se encuentra en el directorio de esta notebook.

[ ]:
```
ls -al .git
```

[ ]:
```
tree .git
```

1.7 Configuración de *Git*.

Para conocer o modificar la configuración de *Git* o de un repositorio de *Git* se utiliza el comando `git config`.

Dicho comando puede gestionar la configuración general de *Git* o la configuración de un repositorio específico.

La documentacion oficial de `git config` puede ser consultada en la siguiente liga:

https://git-scm.com/docs/git-config

1.7.1 Despliegue del listado de campos de la configuración.

La opción `--listo -l` se utiliza pare desplegar información, por lo que la siguiente sintaxis regresará un listado de los campos de que conforman la configuración de *Git*, tanto de la informació del usuario *Git*, así como la del repositorio en el que se encuentra.

```
git config --list
```

El resultado es un listado de estructuras `<registro>.<campo>=<valor>`.

Donde:

- `<registro>` corresponde a un registro de la configuración.
- `<campo>` corresponde a un campo específico dentro del registro.
- `<valor>` corresponde al valor asignado al campo indicado.

Ejemplo:

- La siguiente celda mostrará la configuración del usuario actual y del repositorio al que pertence esta notebook.

[ ]:
```
git config --list
```

1.7.2 Despliege de un valor de un campo con la opción `--get`.

Mientras que la opción `--list` despliega todos los datos de configuración, la opción `--get` sólo regresará el valor del campo especificado.

```
git config --get <registro>.<campo>
```

En caso de no encontrar una coincidencia exacta de la estructura `<registro>.<campo>` dentro de la configuración, no se deplegará nada.

Ejemplos:

- La siguiente celda regresará el valor del campo `url` del registro `remote.origin`.

[ ]:
```
git config --get remote.origin.url
```

- La siguiente celda regresará el valor del campo `erroneo` del registro `remote.origin`. En vista de que este campo no existe, no se devolverá nada.

[ ]:
```
git config --get remote.origin.erroneo
```

1.7.3 Ámbitos de configuración de *Git*.

El comando `git config` config puede acceder a:

- La configuración del repositorio que se está gestionando, la cual está ligada al archivo `config` del subdirectorio `.git` del repositorio.
- La configuración del usuario de *Git*, la cual está ligada al archivo `.gitconfig` localizado en el directorio *home* del usuario.
- La configuración de aplicación *Git* dentro del sistema en el que se encuentra, la cual está, ligada al archivo `/etc/gitconfig` del sistema.

Nota: Los archivos de configuración de `Git` están estructurados conforme al formato

La sintaxis para definir el ámbito de `git config` es:

`git config --<opción de ámbito> <opciones>`

Donde:

- `<opción de ámbito>` puede ser:
  - `local` para la configuración del repositorio que se está gestionando.
  - `global` para la configuración global del usuario de *Git*.
  - `system` para la configuración de la aplicación de *Git* en el sistema en el que se encuentra instalada.

En caso de que no se defina un ámbito, el comando `git config` traerá la configuración de los ámbitos `--local` y `--global`.

El archivo `.git/config` del ámbito `--local`. El archivo `.git/config` dentro de un repostiorio de *Git* contiene la configuración de dicho repositorio.

Ejemplo:

- La siguiente celda mostrará el contenido del archivo `.git/config` del repositorio de esta notebook.

```
[ ]: cat .git/config
```

- La siguiente celda desplegará la configuración del archivo `.git/config` del repsitorio actual usando el comando `git config --local list`.

```
[ ]: git config --local --list
```

El archivo `~/.gitconfig` del ámbito `--global`. El archivo de configuración del usuario se llama `.gitconfig` y se encuentra en el directorio *home* del usuario.

Ejemplo:

- La siguiente celda despelgará el contenido del archivo `~/.gitconfig`.

```
[ ]: cat ~/.gitconfig
```

- La siguiente celda desplegará la configuración del archivo `~/.gitconfig`usando el comando `git config --global list`.

```
[ ]: git config --global --list
```

El archivo `/etc/.gitconfig` del ámbito `--system`. La opción `--system` le indica a `git config` que ese comando se aplicará a la configuración de la aplicación *Git* en el sistema. En entornos basados en *GNU/Linux* el archivo de configuración se localiza en `/etc/gitconfig`:

Ejemplo:

- La siguiente celda tratará de desplegar la configuración contenida en el archivo `/etc/gitconfig` usando el comando `git config --system list`.

Nota: Es probable que el archivo `/etc/gitconfig` no exista y entonces se producirá un mensaje de error.

[ ]:
```
git config --system --list
```

1.7.4 Adición o modificación de un campo de la configuración.

Es posible añadir un campo o modificar el valor de un campo de la configuración de *Git* mediante la siguiente sintaxis:

`git config --<opcion de ámbito> <registro>.<campo> <valor>`

Donde:

- `<registro>` indica el nombre del registro al que pertenece el campo a añadir o modificar.
- `<campo>` indica el nombre del campo a añadir o modificar.
- `<valor>` indica el valor del campo a añadir o modificar.
- `<opción de ámbito>` es el ámbito al que se añadirá el campo. En caso de no definirse, se realziará una búsqueda en los ámbitos `--local` y `--global`.

Este comando buscará un registro que coincida y posteriormente un campo que coincida en los archivos de confoguración correspondientes. * En caso de que exista un campo que coincida con la estructura `<registro>.<campo>`, el valor de dicho campo será sustituido por el valor ingresado. * En caso de que no exista un campo que coincida con la estructura `<registro>.<campo>`, el campo será creado con el valor ingresado.

El contenido de `<valor>` siempre será considerado como una cadena de caracteres. En caso de que el valor sea una cadena de caracteres que incluya espacios, es necesario encerrar dicho valor entre comillas `" "`.

Ejemplo de la configuración mínima de un usuario de *Git*. Para poder realizar operaciones de "commit", es recomendable definir los datos básicos del usuario:

- `user.name` para el nombre del usuario.
- `user.email` para la dirección de correo electronico del usuario.

Estos datos pertenecen al ámbito `--global`.

- la siguiente celda asignará el valor `Comunidad Cloudevel` al campo `user.name` en el ámbito `--global`.
- Debido a que el valor es una cadena de caracteres con espacios, es necesario encerrarlo entre comillas.

[ ]:
```
git config --global user.name "Comunidad Cloudevel"
```

- la siguiente celda asignará el valor `contacto@cloudevel.com` al campo `user.email` en el ámbito `--global`. En este caso, las comillas son opcionales.

[ ]:
```
git config --global user.email contacto@cloudevel.com
```

- La siguientes celdas mostrará las modificaciones a la configuración.

[ ]:
```
git config -l
```

[ ]:
```
git config --get user.email
```

[ ]:
```
cat ~/.gitconfig
```

Ejemplo de asignación de un campo inútil a la configuración. Es posible añadir múltiples campos mediante `git config`. El algunos casos estos campos no son útiles para *Git*.

- La siguiente celda creará el campo `error` en le registro `use` el cual se guardará en el ámbito `--global` con el valor `Error`.

[ ]:
```
git config --global use.error Error
```

[ ]:
```
git config --get use.error
```

[ ]:
```
cat ~/.gitconfig
```

1.7.5 Eliminación de un registro dentro de la configuración.

Para eliminar un registro, se utiliza la opción `--remove`.

`git config --remove <registro>`

[ ]:
```
git config --global --remove use
```

[ ]:
```
git config -l
```

1.7.6 Edición de un archivo de configuración de *Git* con la opción `-e`.

Esta opción permite abrir un editor para modificar el archivo de configuración.

`git config <ámbito> -e`

Ejemplo:

En este caso se debe ejecutar desde una terminal:

`git config --global -e`