19/3/2019

CLASE 1 : git

* git config --global user.name "Guybrush Threepwood"
* git config --global user.email guybrush@example.com

**-configura el usuario realizando los comandos. sirve porque algunas notificaciones respecto de lo que haga el usuario en los repositorios te las manda el git al email indicado.**

* git init

**crea un repositorio local en la carpeta local en donde la terminal esté ubicada.**

* ssh-keygen

**genera 2 archivos:**

**1- nombre-pub es una clave ssh para vincular la terminal local a un servidor git de forma segura, encriptada, debe agregarse a la cuenta de github menu settings>SSH and GPG keys**

**2- otro archivo de mismo nombre sin extensión que es la *private key***

**éste ultimo debe agregarse al agente ssh local mediante los comandos:**

eval $(ssh-agent -s)

**eso devuelve algo como “>Agent pid 59566” donde el nro será distinto en gral.**

ssh-add ~/.ssh/[nombre archivo sin extensión]

**esto último asocia la private key…si éste archivo está en la carpeta actual, ignorar el “**~/.ssh/”

cd ~/.ssh **cambia la ubicación a la subcarpeta que contiene la clave ssh creada**

cat id rsa .pub **muestra la clave creada**

**eval "$(ssh-agent -s)"**

**ssh-add ~/.ssh/llave**

**This launches the agent and sets the necessary environment variables (****SSH\_AUTH\_SOCK, SSH\_AGENT\_PID).**

**posible error si usamos WSL (Windows subsystem for Linux) y ubicamos .ssh en otro lado y no en el directorio home (“~”):**

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ @ WARNING: UNPROTECTED PRIVATE KEY FILE! @ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ Permissions 0777 for '.ssh/llave' are too open. It is required that your private key files are NOT accessible by others. This private key will be ignored.

**Lo anterior Nos indica que el archivo de llave privada tiene permisos de edición y eso es arriesgado**

**ejecutando lo anterior en el caso de WSL (Windows subsystem for Linux) da error porque no toda carpeta permite alterar los permisos en Linux al haber conflicto con Windows NTFS. Entonces NO ignorar el “**~/.ssh/” **así se crea eso en la raíz del sistema de Linux**.

**Sea el archivo de key privada “llave”**

chmod 600 path//llave …**NO funciona en WSL en general**

**se ejecuta el comando pero no cambia los permisos por el conflicto con NTFS**

chmod 600 ~/.ssh/llave …si funciona

o bien : sudo chmod 600 ~/.ssh/llave

y probar ahora si: ssh-add…

**otro error:**

“Could not open a connection to your authentication agent.”

**Podria ser que el nro de agente creado al hacer** eval "$(ssh-agent -s)" **no coincida con el de la variable SSH\_AUTH\_SOCK**

ssh-agent -k **🡪este commando elimina posibles agentes múltiples en conflicto**

**esto parece pasar porque el PID cuando se crea el agente no coincide con el del la variable antedicha**

**eventualmente cambiamos los permisos del archivo de llave privada y lo agregamos haciendo ssh-add…**

ssh-add ~/.ssh/llave

Enter passphrase for /home/drtandar/.ssh/llave:

**Y escribimos o pegamos la clave que creamos al crear las llaves**

**Si tenemos éxito devuelve**

Identity added:

**al intentar hacer un** git push **con github con la llave ssh y la url para autenticación ssh (no la http) puede que de un error de conexión entonces hacemos**

git remote set-url origin [git@github.com:ethan1987-2/I-C.git](mailto:git@github.com:ethan1987-2/I-C.git)

**si nos equivocamos con la url podemos corregirla con este mismo comando**

**el ultimo argumento es la dirección url de autenticación ssh q da github**

**git remote -v**

**deberia mostrar que quedo la url indicada**

**puede que persista otro error de conexion**

ssh: connect to host github.com port 22: Connection timed out fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists.

**Esto puede tener que ver con una conexión a través de proxy**

**Como es en el caso de la CNEA, usar este sistema de autenticación por ssh-keygen puede traer problemas por el proxy**

**Entonces:**

nano ~/.ssh/config

**y pegar lo siguiente:**

Host github.com

Hostname ssh.github.com

Port 443

User git

IdentityFile ~/.ssh/llave

IdentitiesOnly yes

**Probar haciendo** ssh -T [git@github.com](mailto:git@github.com)

**Probar configurando git un proxy**

git config --global http.proxy http://proxy-cac.cnea.gov.ar:1280

**puede haber el siguiente ridículo error**

Couldn't establish connection to proxy: Success

Connection closed by UNKNOWN port 65535

**Entonces si tampoco funciona probar instalando corkscrew**

Sudo apt install corkscrew

**lo anterior puede que ni así funcione por eso se recomienda usar el otro método de autenticación de github usando PAT personal Access tokens**

**Try HTTPS with a Personal Access Token (if SSH fails)**

**If SSH is blocked and you can’t tunnel through the proxy, switch to HTTPS and use a token:**

git remote set-url origin https://github.com/ethan1987-2/I-C.git

**Then push using:**

**- Username: ethan1987-2**

**- Password: your GitHub Personal Access Token**

**You can generate one at github.com/settings/tokens**

**nos pide una clave que en la cuenta de github nos dan, que sirve para autenticación ssh (no es el pass de la cuenta) esto ultimo a mi me anduvo**

**siempre que hagamos push nos va a pedir la clave por mas que se la hayamos dicho 500 veces. Entonces usando el paquete libsecret (instalar con apt: sudo apt install libsecret-1-0 libsecret-1-dev)**

git config --global credential.helper libsecret

**así podemos hacer que git recuerde nuestra info de login**

git config --global credential.helper store

**After this, Git will prompt for your username and password once, then save them in ~/.git-credentials as plain text**

* git clone [URL]

**copia localmente un repositorio remoto que tenga la URL indicada**

Ejercicio (ejemplo): git clone [git@gitlab.com](mailto:git@gitlab.com):talleres-comcom/taller-git-ejercicio.git

* git add [filename]

**pasa el archivo “filename” al estado *staged* en el repositorio actual. o sea se lo presentamos al git.**

git add -u …agrega todos los archivos modificados

* git ls-tree -r master --name-only

**lista los archivos trackeados.**

* git ls-files -z | xargs -0 rm –f

**elimina los archivos del arbol entero**

Ejemplo: git add archivo.txt

git reset HEAD <filename> quita un archive del add

si <filename>=-- saca TODOS los archivos

* git commit -m [mensaje]

**confirma todos los cambios de los archivos en estado *staged* y los agrega a un “log” o vitácora con el mensaje indicado entre comillas.**

Ejemplo: git commit -m “cambié tal cosa en tales archivos”

* git commit --amend -m [nuevo mensaje]

**arregla un mensaje mal hecho**

* git revert [hash]

**Buscamos el hash del commit a cambiar usando** git log **y usamos este comando**

* git stash

**Se guarda el estado actual de los archivos**

**modificados y nos deja el directorio limpio.**

**Para volver a mostrar los cambios guardados ejecutamos git stash**

**apply.**

* git rm [archivo]

**preborra un archivo y marca esta acción como staged.**

* git mv [archivo] [nuevo nombre/ubicación]

**pre - mueve/renombra archivo y marca como staged**

* git diff

**Para esto tenemos el comando git diff, que muestra las diferencias**

**entre el estado actual de los archivos y la ´ultima vez que hicimos git add**

**(cambios marcados como staged).**

**Si, en cambio, queremos ver las diferencias entre los cambios marcados**

**como staged y los que confirmamos en el ´ultimo commit, podemos usar**

**git diff --staged**

* git log

**muestra el historial de cambios**

* git branch

**crea una rama en el repositorio**

* git checkout [rama]

**cambia a la rama indicada**

* git merge [nombre de la rama a fusionar]

**fusiona la rama actual la indicada. cambia sólo la rama actual**

* git status

**muestra el estado en el que se encuentran los archivos en el repositorio actual**

ADMINISTRACIÓN DE REPOSITORIOS REMOTOS

* git remote add [nombre repositorio] [URL]

**agrega un repositrio remoto y lo vincula al local.**

* git remote show [repositorio remoto]

**Devuelve información del repositorio remoto indicado, si no se pone nada te dice los nombres de los repositorios remotos asociados.**

* git remote -v

**muestra todos los repositorios remotos asociados**

* git push/pull [remoto] [branch]

**envía/recibe al repositorio remoto los cambios *staged* de archivos locales presentes en el repositorio remoto**

**si se queja porque dice que hay cosas en el remoto que no tenemos y quizá queramos le podemos forzar el push con**

git push -f [repo] [rama]

**possible error. Significa que hay que usar un tipo de autenticación ver mas arriba ssh o tokens**

**Password authentication is not supported for Git operations. fatal: Authentication failed for 'https://github.com/ethan1987-2/I-C.git/'**

**GitHub has disabled password-based authentication for Git operations over HTTPS since August 2021. That error means you're trying to push using a username and password, but GitHub now requires either a Personal Access Token (PAT) or SSH key instead.**

ARCHIVO .gitignore para evitar q aparezca la plaga de archivos en git status

**Create** the text **file gitignore**.txt.

Open it in a text editor and **add** your rules, then save and close.

Hold SHIFT, right click the folder you're in, then select Open command **window** here.

Then rename the **file** in the command line, with ren **gitignore**.txt .**gitignore**