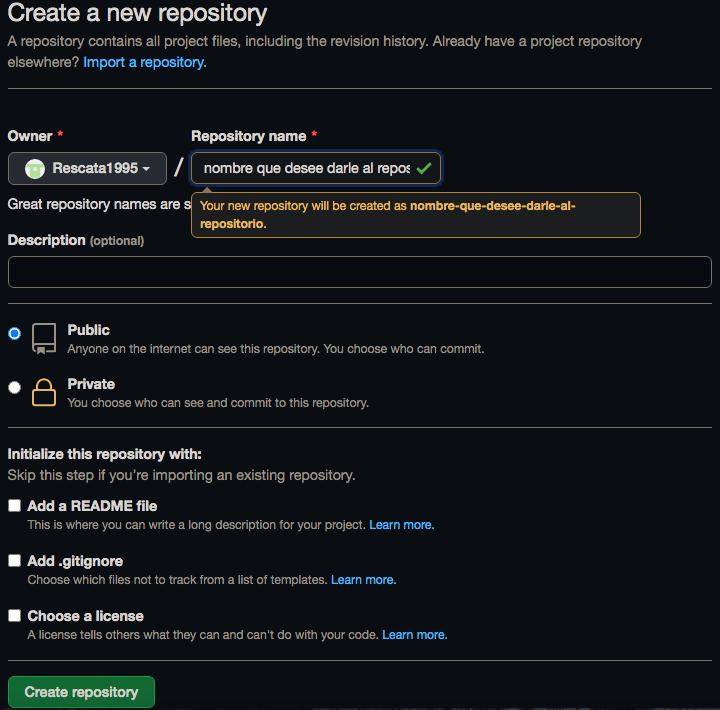
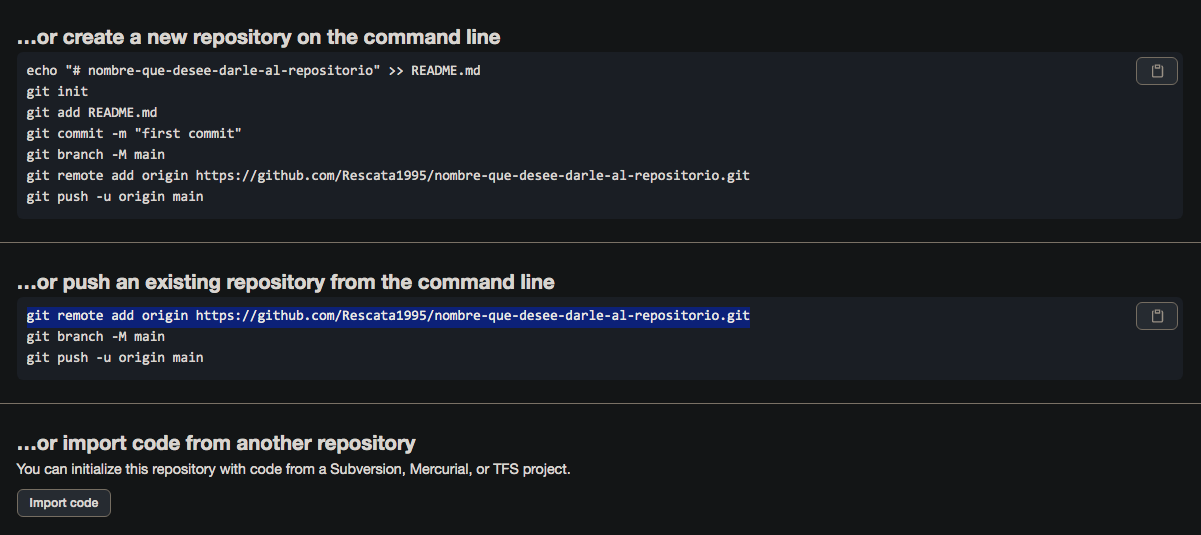
***Tutorial de GITHUB***

Todos los repositorios que hemos creado en GIT ahora nos interesa guardarlos en una nube o servidor externo (GITHUB ocupa GIT en la nube, eso hace). Si en caso dado se nos quema nuestro disco duro ya evitaremos completamente el riesgo de perder absolutamente todo o me encuentro trabajando en un proyecto grupal en el que necesito cooperar con más programadores; pues, todos nuestros repositorios estarán en la nube, el cual podemos retornar nuevamente a nuestra pc en cualquier momento. Esa interacción entre GIT & GITHUB es la que nos interesa. Pasos desde la creación de cuenta en GITHUB hasta la creación de un repositorio remoto vinculado a nuestro editor de texto y terminal:



Le damos un nombre al repositorio remoto que deseemos crear y decidimos si será de caracter público o privado y listo: create repository!

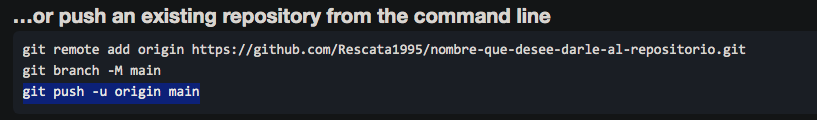


Luego, seleccionamos toda la ruta subrayada con fondo azul, la pegamos en nuestra terminal y presionamos “enter”, tal que así (el comando ***“git remote add origin”***... es el que tiene la función de vincular tu repositorio local con un repositorio remoto; de tal manera que, posteriormente, puedas exportar tus cambios guardados de un repositorio local en un repositorio remoto o, en su defecto, puedas importar los cambios guardados de un repositorio remoto en un repositorio local”):



Esto le dirá a GIT que ya estaremos trabajando de forma remota. Es decir, va a crear un identificador de todos nuestros archivos o de todo el trabajo que hemos hecho con GIT en la URL que hemos importado de GITHUB (nuestro repositorio local ahora estará en una nube).

Entonces, al terminar de registrar o guardar todos los commits correspondientes en nuestro repositorio local; es decir, cuando terminemos la tarea, importaremos la siguiente ruta (subrayada en fondo color azul) a nuestra terminal (y presionamos “enter”):



Esto cargará de forma automatica todos nuestros repositorios locales a GITHUB; de tal manera que, no sólo tengamos repositorios locales, sino que también en la nube (libre de riesgos de pérdida): sólo actualice su página en GITHUB.

Ahora bien, con esto ya tenemos vinculado al GIT local con el GIT remoto; es decir, no es necesario llamar de nuevo las anteriores rutas subrayadas si quisieramos guardar un nuevo commit en nuestro repositorio local y cargarlo en la nube de GITHUB. ¿Entonces qué se hace? pues se ejecuta el siguiente comando: “git push”. Listo, eso es todo. Así hemos cargado en la nube GITHUB nuestro último commit guardado en nuestro repositorio local. En el caso contrario, es decir, que quisieramos importar un “commit” del servidor externo de GITHUB (ya sea porque lo hayamos creado allá y no en nuestro repositorio local); pues, el comando a ejecutar sería: “git pull”. Por último, si deseamos compaginar por igual los “commits” que tenemos en ambos repositorios, local y remoto, debemos ejecutar el siguiente comando: “git fetch”. (la explicación de este último comando puede mejorar).

***Introducción a llaves SSH***

Aunque, en principio, github sea una plataforma o sitio web seguro (*HTTPS: Protocolo seguro de transferencia de hipertexto*) existe cierta vulnerabilidad a la que estamos expuestos: al introducir tu nombre de usuario y contraseña en la página web, éstas se guardan también en nuestro entorno local (digamos, en nuestro computador). Entonces, en el eventual caso de que sufras de un robo de tu dispositivo, estarás completamente expuesto a sufrir lo que se conoce como: ***“Password Cracking”***, estás expuesto a que “crackeen” (descifren) la contraseña de tu repositorio. “Password Cracking” es la ciencia que consiste en descifrar contraseñas o también se refiere al acto o proceso que se lleva a cabo para recuperar contraseñas que se han sido almacenadas en un equipo (cualquiera, desde dispositivos moviles a dispositivos de escritorio). En nuestro caso entonces estamos planteando la posible idea aterradora de que nuestro nombre de usuario y contraseñas de Github se vean expuestas a ser descifradas; seríamos vulnerables a ataques y quedaría en riesgo nuestra información sensible: servidor, proyectos (ya sean propios o de clientes), etc. Esta es la forma en la que sitios web, por ejemplo, son hackeados.

Para evitar todo esto, tenemos que agregarle una capa de seguridad mayor a nuestros sitios web de interés; donde, a veces, colocamos información confidencial o muy sensible y que deseamos proteger: esto se hace por medio de una combinación entre llaves públicas y privadas ***(llaves SSH)***, a este proceso (de cifrar por medio de llaves, especialmente una pública y una privada) más tarde se dará cuenta que se le conoce también como: ***cifradop o criptografia asimetrica.***

***El mecanismo funciona más o menos así:*** en nuestro entorno local (computadora) se debe crear una llave pública y una llave privada; y, una vez creadas, la llave pública debe ser enviada a github (en nuestro caso, nuestro sitio web de interés); pero puntualmente la llave pública es asignada a un repositorio remoto propio de github (al repositorio de nuestro interés *-se supone, el que se quiere cifrar*): todo este proceso se lleva a cabo conectandonos al repositorio por medio de un protocolo nuevo, ya no *HTTPS*; sino, ***SSH (Secure Shell)***. La llave privada sigue intacta en nuestra computadora, no es enviada nunca a ninguna parte.

dato: SSH es el mismo protocolo que se debe usar para conectarnos a servidores remotos, a otras computadoras que queramos controlar remotamente por consola.

Github, a partir de la llave pública que está correlacionada con tu llave privada y que le acabas de enviar; te regresa el favor y te envía, cifrado con tu propia llave pública (la que le mandaste), su propia llave pública de github (es un cifrado hecho, recuerde, a partir de la llave pública que te recibe github; y github, a partir de ahí, con ese cifrado, te genera una llave pública propia de él para ti). Entonces, github le manda su propia llave pública a su entorno local y la conecta: y así ambos quedan, mutuamente, conectados por medio del protocolo SSH, en lo que se conoce como una ***conexión de doble camino***, la cuál es 100% cifrada por SSH.

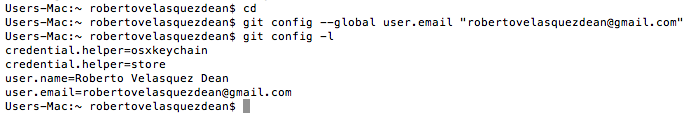
Por otro lado, la llave privada de su entorno local y que nunca es compartida, si desea, le agrega una contraseña encima (sobre ella): esto para tener una llave privada mucho más robusta y poderosa.

***Ahora sí, manos a la obra, veamos paso a paso todos estos procesos en un repositorio real de github.***

Aclaremos algo de entrada, *las llaves SSH no son por repositorio o por proyecto, sino por persona*; es decir, puedes crearlas en cualquier parte donde te situes parcialmente dentro de tu consola. Idealmente se crean en la carpeta home *(cd, cd ~)*.

Posteriormente, verificamos que el correo registrado en nuestra cuenta *github* coincida con el configurado (o potencialmente a configurar) en nuestro entorno local, puntualmente en *git.* El comando de git para registrarle un correo es el siguiente: ***git config --global user.email “inserte correo aquí -que debe coincidir con el registrado en su cuenta de github, recuerde”.***

Para verificar que efectivamente hemos registrado con exito el correo insertado, podemos corroborar esta información con el siguiente comando: ***“git config -l”***, donde podemos revisar también cuál es el nombre propiamente de usuario registrado en git. Hasta ahora el registro de comandos ejecutados dentro de su terminal se debe ver algo más o menos así:



Hecho esto, estando en sintonia nuestro correo electronico registrado en git y, posteriormente, nuestro correo electronico registrado en github; ya podemos crear nuestras llaves SSH (no olvide estar situado en *home*).

Corra el siguiente comando en su terminal: ***ssh-keygen -t rsa -b 4096 -C “robertovelasquezdean@gmail.com”***

*Vamos por partes:*

***rsa*** es un algoritmo más, entre otros, para crear la llave ssh (siendo el más popular de hecho)

***-b + número*** es un elemento que nos indica, a partir de un número, qué tan compleja queremos que sea la llave.

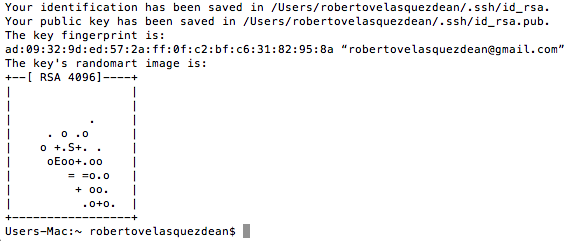
***-C “email...”*** se refiere a definir *qué correo electronico queremos que esté conectado a la llave en cuestión.* Debe intuir que es el mismo correo que hemos estado configurando, y en sintonia, en git y github.

Inmediatamente luego de correr la línea de comando, con el comando para crear llave SSH, se nos mostrará esto:



*Este proceso ya me está generando la llave pública y privada a partir del comando ejecutado.* Posteriormente nos pregunta dónde deberiamos guardar las llaves (y he aquí la importancia de situarnos inicialmente desde el *home*), si usted desea sólo presiona *enter,* ya la consola sobreentiende que usted desea guardar sus llaves en su dirección actual. Es decir, en nuestro caso, las llaves serán guardadas en *home* y, más puntualmente, en una carpeta oculta llamada: ***“.ssh”***

Luego de presionar *enter* la consola pide un ***“passphrase”,*** *passphrase* en realidad es una manera *fancy* de escribir *“passwords con espacios”.* Es básicamente la contraseña, adicional de texto, que le vas a poner *(me imagino que por temas de respaldo)* a tu llave pública y privada; sin embargo, es preciso decir que, esto es opcional, pero lo más recomendable es que sí se lo coloque *(de momento no lo haremos así, presionamos enter dos veces sin decir más).* Debe ver algo más o menos así:



*Expliquemos por parte...*