

Chaves SSH e Tokens

Por que devemos entender esses dois tópicos? Pois quando vamos jogar o nosso código para o Github, nós teremos que nos autenticar, dizendo a ele que nós somos nós mesmos. Então teremos nosso nome de usuário e nossa senha e isso veio sendo mudado com o tempo no Github e a autenticação de apenas colocar o nickname e a senha acabaram se tornando lentos e obsoletos. Em 2021, o Github anunciou que esse tipo de autenticação seria desligado e não seria mais válido. Hoje já não conseguiremos mais empurrar nossos códigos para lá utilizando apenas o nome e a senha, serão necessários outros processos mais seguros para poder se autenticar e são esses processos que veremos aqui e como configurá-los.

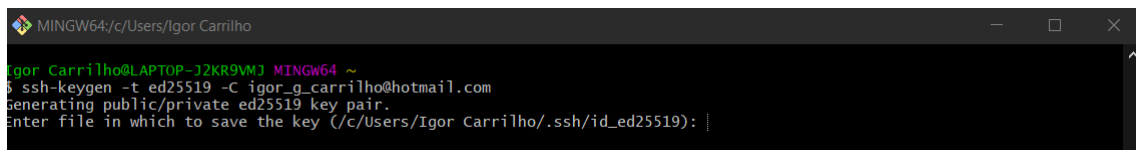
Chave SSH

A grosso modo é uma forma de estabelecer uma conexão segura e encriptada entre duas máquinas, no caso iremos nos conectar com o servidor do Github e configurar a nossa máquina local como uma máquina confiável para o Github, estabelecendo duas chaves, uma pública e uma privada. Com a chave pública, vamos colocar no Github e a partir do momento que fizermos isso o Github vai reconhecer a nossa máquina e aí todos os repositórios que tivermos na nossa máquina por esse processo SSH e formos enviar código para lá, o Github já vai reconhecer a assinatura da nossa máquina, vai ter uma conexão prévia estabelecida e seremos capazes de enviar código sem nem ao mesmo precisarmos colocar senha.

A primeira coisa que vamos fazer é ir até a nossa conta no Github, vamos no ícone da nossa imagem e vamos na opção "Settings". Na página que fomos direcionados, ao lado direito, vamos procurar pela opção "SSH and GPG Keys". Nesse local estará a nossa chave SSH do Github. Clicando nele, teremos que preencher o campo para que possamos ter a chave propriamente no Github. Agora vamos ao CLI para fazer o processo de criar essa chave e assim ela ficar ligada na nossa máquina fazendo com que tenhamos esses dados para alimentar no Github.

Vamos iniciar o Git Bash para iniciar a criação da chave. Vamos aos comandos necessários para que essa chave seja criada.

1. `ssh-keygen -t ed25519 -C igor_g_carrilho@hotmail.com`

A screenshot of a Windows terminal window titled "MINGW64/c/Users/Igor Carrilho". The prompt shows the user is "igor_carrilho@LAPTOP-32KR9VM3 MINGW64 ~". The command entered is "ssh-keygen -t ed25519 -C igor_g_carrilho@hotmail.com". The output shows "generating public/private ed25519 key pair." and "Enter file in which to save the key (C:/Users/Igor Carrilho/.ssh/id_ed25519):". The cursor is at the end of the path.

```
igor_carrilho@LAPTOP-32KR9VM3 MINGW64 ~  
$ ssh-keygen -t ed25519 -C igor_g_carrilho@hotmail.com  
generating public/private ed25519 key pair.  
Enter file in which to save the key (C:/Users/Igor Carrilho/.ssh/id_ed25519): |
```

Todas as pastas que iniciam com ponto, no caso a .ssh, são pastas ocultas e o id_25519 é o id da chave que ele está gerando. Esse é o local especificado onde a chave está sendo gerada. Após isso, daremos um Enter e ele irá pedir para que entremos com uma senha. Após isso, ele irá falar que a nossa senha de autenticação foi salva e a chave pública também foi salva no

mesmo lugar que a privada, porém com o .pub na frente e depois ele dá o fingerprint. E dessa forma, a nossa chave já foi gerada.

```
MINGW64:/c/Users/Igor Carrilho
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/Igor Carrilho/.ssh/id_ed25519):
Created directory '/c/Users/Igor Carrilho/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Igor Carrilho/.ssh/id_ed25519
Your public key has been saved in /c/Users/Igor Carrilho/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:bsEEwH5HHpvtVDG+5/6xoXPqY4X0Xd+V2E+0B/pWFPc igor_g_carrilho@hotmail.com
The key's randomart image is:
+--[ED25519 256]--+
  .... o. |
  . .o . . . . |
  . o.= . . + |
  . .o= o . . E |
  . .So . .+=+o |
  . . . . =*= |
  . o o.=+B |
  . . B+oB |
  . o=*o. |
+-----[SHA256]-----+

Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~
$
```

Vamos navegar até essa pasta agora para vermos essas nossas chaves

2. `cd /c/user/IgorCarrilho/.ssh` e vamos dar o comando “ls” listar os arquivos dentro dessa pasta e nele veremos nossas duas chaves SSH públicas e privadas.

```
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~
$ cd .ssh/

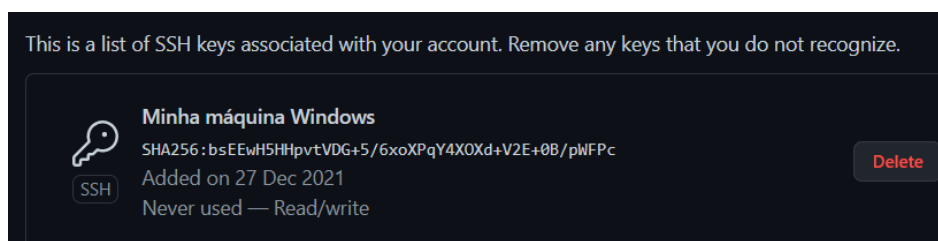
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ ls
id_ed25519 id_ed25519.pub
```

3. Vamos utilizar um comando especial para visualizar o conteúdo de uma dessas chaves e que iremos usar lá no final para colocar no GitHub -> `cat id_ed25519.pub`

Se rodarmos isso, ele irá nos mostrar a chave pública

```
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ ls
id_ed25519 id_ed25519.pub
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINCNO0iukIQ/M27IghGv4ICrP/MmfWpNEQGSNxs5g1T igor_g_carrilho@hotmail.com
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$
```

Podemos copiar essa chave e será isso que iremos usar lá no Github. Vamos dar um nome para a nossa chave, copiar e colocar lá o que foi passado como assinatura da chave para nós no Git Bash. Ele irá pedir a nossa senha para que sejamos autenticados e ele irá gerar para nós a chave.



A chave ainda não está pronta para ser utilizada, ainda precisamos fazer algumas coisas no CLI para validar ela e ela poder funcionar corretamente. Voltando ao Github, vemos que ainda estamos dentro da pasta `.ssh` e se dermos o comando `"pwd"` ele irá nos mostrar o caminho completo dessa pasta.

Agora iremos inicializar o SSH Agent, que é uma entidade que fica responsável por pegar as chaves e lidar com elas. Para isso, iremos utilizar o seguinte comando:

- `eval $(ssh-agent -s)`

Com esse comando ele irá gerar um número que se tivesse startando um projeto para que o SSH possa rodar como plano de fundo e esse é o número do processo no computador.

```
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ eval $(ssh-agent -s)
Agent pid 81

Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ |
```

Agora iremos entregar essa chave para ele digitando: `ssh-add id_ed25519`. Como estamos dentro da própria pasta, podemos colocar direto o ID, em caso contrário, devemos indicar o caminho completo de onde está a chave. Nesse caso iremos passar a chave privada e não a pública. Fazendo isso, ele irá pedir a senha que colocamos ao criar a chave e ele irá nos passar a informação que a chave foi adicionada e inclusive o e-mail em que a chave foi adicionada.

```
Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ eval $(ssh-agent -s)
Agent pid 81

Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ ssh-add id_ed25519
Enter passphrase for id_ed25519:
[Identity added: id_ed25519 (igor_g_carrilho@hotmail.com)]

Igor Carrilho@LAPTOP-J2KR9VMJ MINGW64 ~/.ssh
$ |
```

Token

Vamos até o Github e na mesma aba onde está a nossa chave SSH, vamos descer até a parte onde está `"Developer Settings"`, clicar em `"Personal Access Tokens"` e depois em `"Generate New Token"`. Podemos também configurar uma data de expiração para ele também e vamos deixar marcado a opção `"Repo"` que irá nos atender perfeitamente ao que precisamos. Após isso, vamos clicar em `"Generate Token"`. Esse token iremos salvar em algum arquivo no computador.