



DIO - Curso GIT e Github

Chave SSH e Token - como gerar chave no GITBASH

Usar os seguintes comandos: (não precisa estar em um diretório específico)

```
ssh-keygen -t ed25519 -C dai.koblitz@gmail.com
```

ed25519 é o tipo de criptografia da chave

(ou outro email que deve obrigatoriamente estar associado a uma conta do github)

ENTER

aqui é possível colocar uma senha (que não sera visível)

ENTER

```
Otávio@perkles-desktop MINGW64 ~
$ ssh-keygen -t ed25519 -C otaviocha@gmail.com
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/Lucas/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Lucas/.ssh/id_ed25519
Your public key has been saved in /c/Users/Lucas/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:CGDr3Aa2UuoQJ9xslkKTWzGwwJ0Aamh6xnbS9RCZUCo otaviocha@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|@B%O=      oo.|
|+Xo#       . +|
|O+%       .  E|
|+* * o o   .  |
|. X = . S .   |
|++         .   |
+-----[SHA256]-----+
Otávio@perkles-desktop MINGW64 ~
$
```

Para visualizar o local onde estão salvas as chaves basta entrar neste diretório:

```
cd /c/Users/daiko/.ssh
```

e listar os documentos:

```
ls
```

vai aparecer a chave privada e a publica:

```
id_ed25519 id_ed25519.pub
```

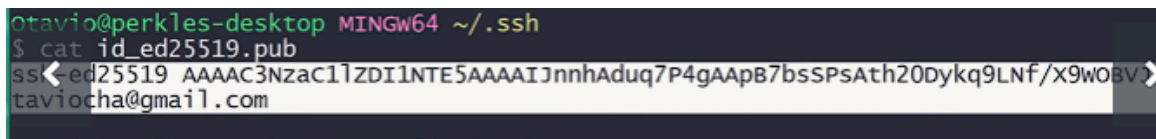
a chave que deve ser utilizada no github é sempre a chave publica.

para pegar a chave publica na sua máquina, digitar

```
cat id_ed25519.pub
```

ENTER

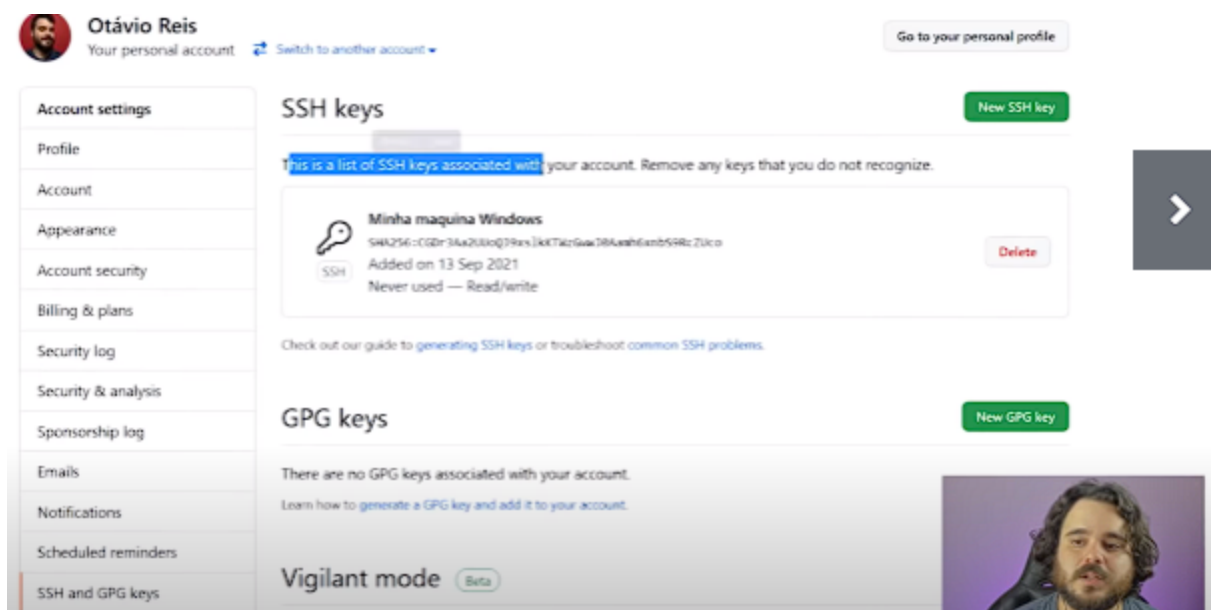
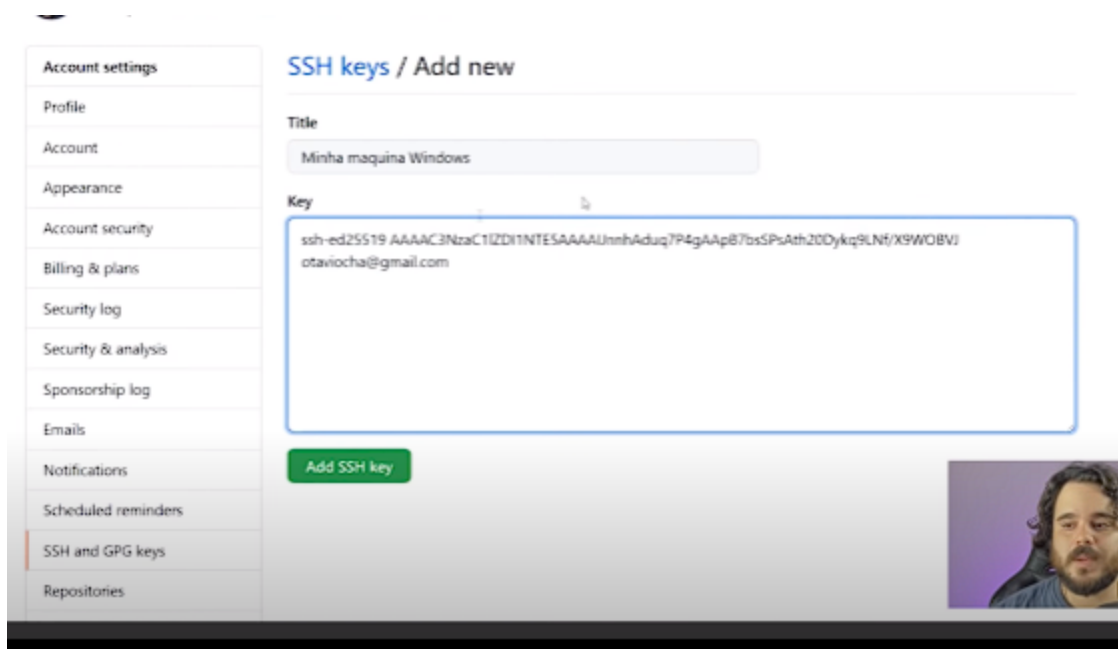
irá retornar uma chave parecida com a da imagem:



```
otavio@perkles-desktop MINGW64 ~/.ssh
$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIjnnhAduq7P4gAapB7bsSPsAth20Dykq9LNf/X9w0BV
taviocha@gmail.com
```

essa chave deve ser copiada para poder colar no github, na sessão SSH and GPG keys

também deve ser adicionado um nome a sua máquina para que você possa identificá-la



em seguida devemos gerar um agente e entregar a chave para ele (privada), e digitar a senha:

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

```
Otavio@perkles-desktop MINGW64 ~/.ssh
$ eval $(ssh-agent -s)
Agent pid 1382

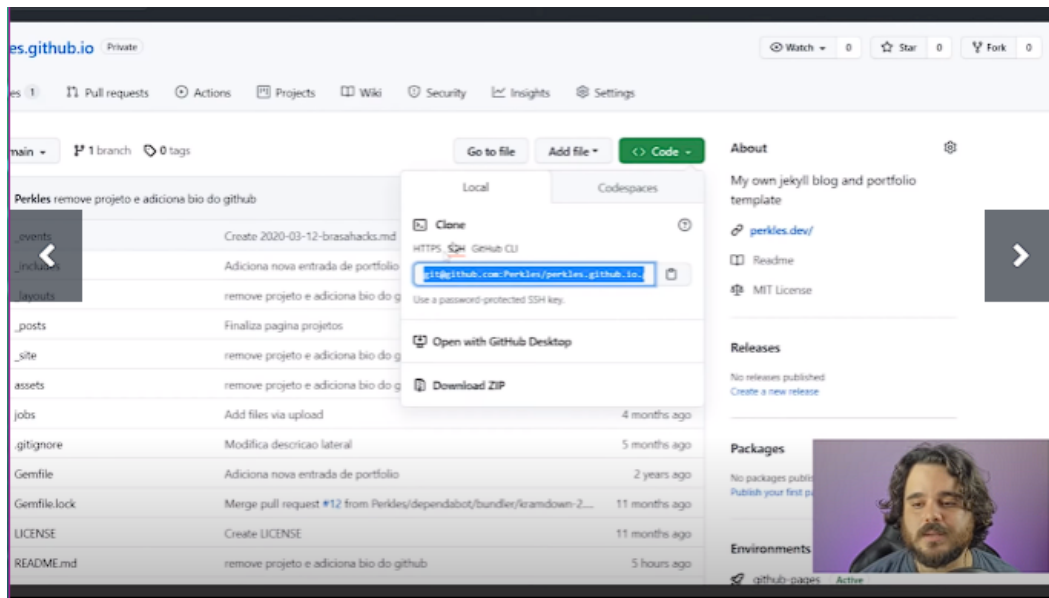
Otavio@perkles-desktop MINGW64 ~/.ssh
$ ls
id_ed25519  id_ed25519.pub

Otavio@perkles-desktop MINGW64 ~/.ssh
$ ssh-add id_ed25519
Enter passphrase for id_ed25519:
Identity added: id_ed25519 (otaviocha@gmail.com)

Otavio@perkles-desktop MINGW64 ~/.ssh
$
```

Para clonar um repositório no Github

```
git clone git@github.com:ILadyLuckI/hello-world.git
```



e dar yes na pergunta no prompt

```

otavio@perkles-desktop MINGW64 /c/workspace/ssh-test
$ git clone git@github.com:Perkles/perkles.github.io.git
Cloning into 'perkles.github.io'...
The authenticity of host 'github.com (20.201.28.151)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXupJWGL7E1IGOCspRomTxdCARLviKw6E5SY8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (RSA) to the list of known hosts.
remote: Enumerating objects: 1278, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 1278 (delta 24), reused 38 (delta 16), pack-reused 1216
Receiving objects: 100% (1278/1278), 21.64 MiB | 7.33 MiB/s, done.
Resolving deltas: 100% (580/580), done.

otavio@perkles-desktop MINGW64 /c/workspace/ssh-test
$

```



para conferir se o repositório foi clonado basta dar ls no diretório

```
ls
```

```

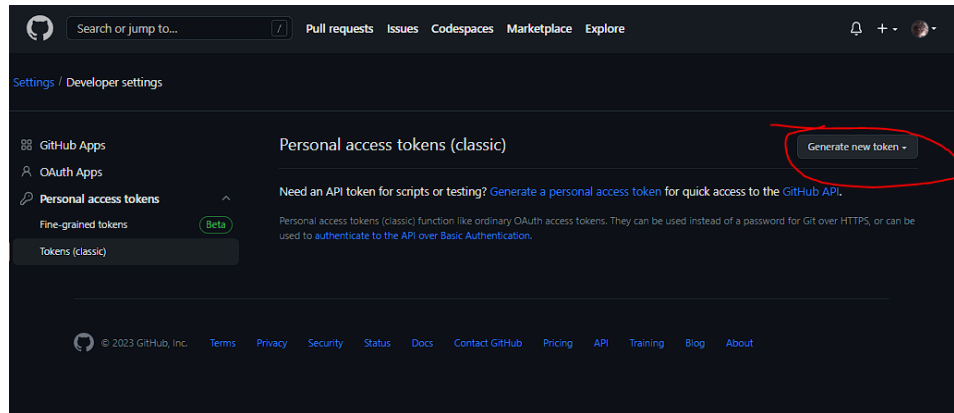
daiko@DESKTOP-S6NOR40 MINGW64 /d/programacao/dio-cursos/workspace/ssh-test
$ ls
hello-world/

daiko@DESKTOP-S6NOR40 MINGW64 /d/programacao/dio-cursos/workspace/ssh-test
$

```

Token de acesso pessoal - 2º forma de autenticação segura

Para configurar o token, deve ir no Github em “Developer settings”, token(classic) e “generate new token”



Marcar ou não uma data de expiração do token, dar um nome para o token e marcar a opção “Repo”

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

Meu token

What's this token for?

Expiration

60 days

The token will expire on Tue, Jun 27, 2023

Select scopes

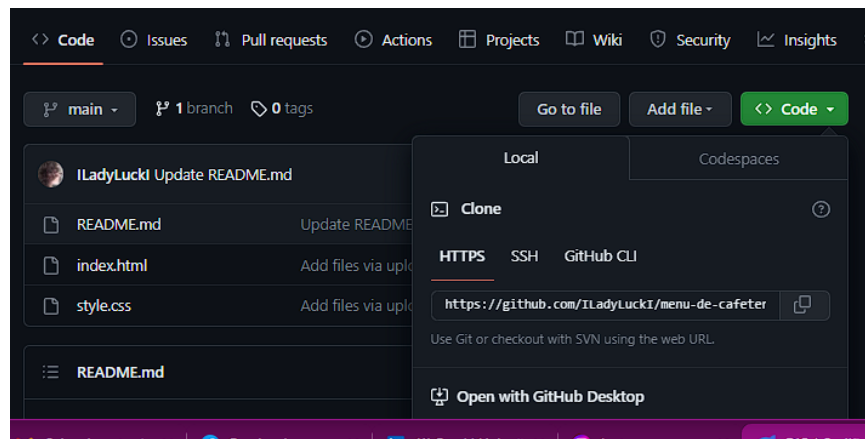
Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repository:write	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> audit_log	Full control of audit log
<input type="checkbox"/> read:audit_log	Read access of audit log
<input type="checkbox"/> codespace	Full control of codespaces
<input type="checkbox"/> codespace:secrets	Ability to create, read, update, and delete codespace secrets
<input type="checkbox"/> project	Full control of projects
<input type="checkbox"/> read:project	Read access of projects
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys
<input type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

Generate tokenCancel

guardar esse token em algum lugar seguro do computador, pois não sera mais possível ve-lo

para fazer um git clone de um repositório usando um token, será preciso copiar o código https e não mais o ssh:



Usando o Git pela primeira vez

O Git precisará de algumas configurações:

*email:

```
git config --global user.email "dai.koblitz@gmail.com"
```

*autor:

```
git config --global user.name Daiana
```

```
daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git config --global user.email "dai.koblitz@gmail.com"

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git config --global user.name Daiana

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$
```


Adicionando um arquivo

No diretório onde se encontra o arquivo a ser adicionado, digite:

```
git add *
```

e em seguida:

```
git commit -m "commit inicial" (ou qualquer outra string)
```

```
daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace
$ ls
livro-receitas/  ssh-test/  token-test/

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace
$ cd livro-receitas

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git add *

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git commit -m "commit inicial"
[master (root-commit) 8f0be27] commit inicial
1 file changed, 51 insertions(+)
create mode 100644 strogonoff.md

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$
```

Modificando o username e o email

usar o seguinte comando:

```
git config --global --unset user.email
```

```
git config --global --unset user.name
```

Adicionando um arquivo ao repositório remoto (GitHub)

primeiro deve se mostrar ao git o caminho do repositório remoto para onde deve ser empurrado o arquivo, usando o seguinte comando:

```
git remote add origin https://github.com/fulanodetal/pasta/arquivo.git
```

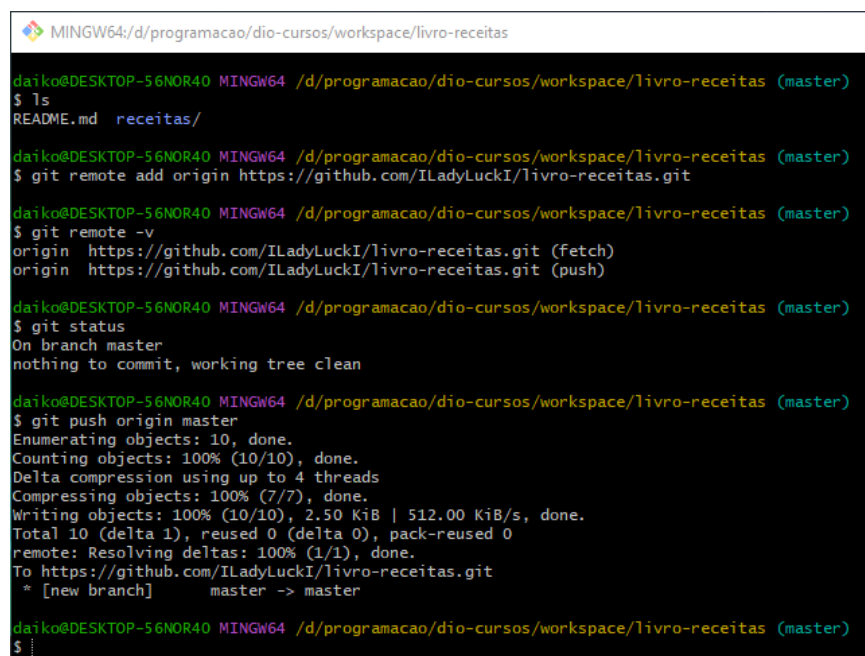
em seguida:

```
git remote -v
```

dar um git status para confirmar se esta tudo ok, em seguida empurrar o arquivo com o comando:

```
git push origin master
```

ficando dessa forma



```
MINGW64; d:/programacao/dio-cursos/workspace/livro-receitas

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ ls
README.md  receitas/

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git remote add origin https://github.com/ILadyLuckI/livro-receitas.git

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git remote -v
origin  https://github.com/ILadyLuckI/livro-receitas.git (fetch)
origin  https://github.com/ILadyLuckI/livro-receitas.git (push)

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git status
On branch master
nothing to commit, working tree clean

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 2.50 KiB | 512.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/ILadyLuckI/livro-receitas.git
 * [new branch]      master -> master

daiko@DESKTOP-56NOR40 MINGW64 /d/programacao/dio-cursos/workspace/livro-receitas (master)
$ |
```

o comando “origin” é apenas um apelido para indicar de onde esta partindo o arquivo, poderia ser qualquer outro nome.

Comandos

ls - exibe uma lista de arquivos dentro de um diretório

cd - muda de diretório, deve estar acompanhado do nome do diretório desejado, para voltar um nível, usar cd ..

pwd - exibe o caminho até o seu diretório atual

clear ou ctrl+l- limpa o prompt

cd .. - voltar um nível na hierarquia de pastas

mkdir - cria um novo diretório

-a - mostra arquivos ocultos (usar ls -a)

-m - passa uma mensagem