

GIT

É um sistema de versionamento de código distribuído criado pelo **Linus Torvalds**.

Comandos básicos no terminal:

Windows	Unix
- cd	- cd
- dir	- ls
- mkdir	- mkdir
- del / rmdir	- rm -rf

dir-> Listar diretórios no caminho situado

cd-> Entrar em determinada pasta **cd /** irá para o c:

cd ..-> Acessar diretório anterior

cls-> Limpa o terminal

mkdir-> Cria uma pasta deve se passar o nome

echo-> Printa o que for escrito no terminal

```
C:\workspace>echo hello  
hello
```

```
C:\workspace>echo hello > hello.txt
```

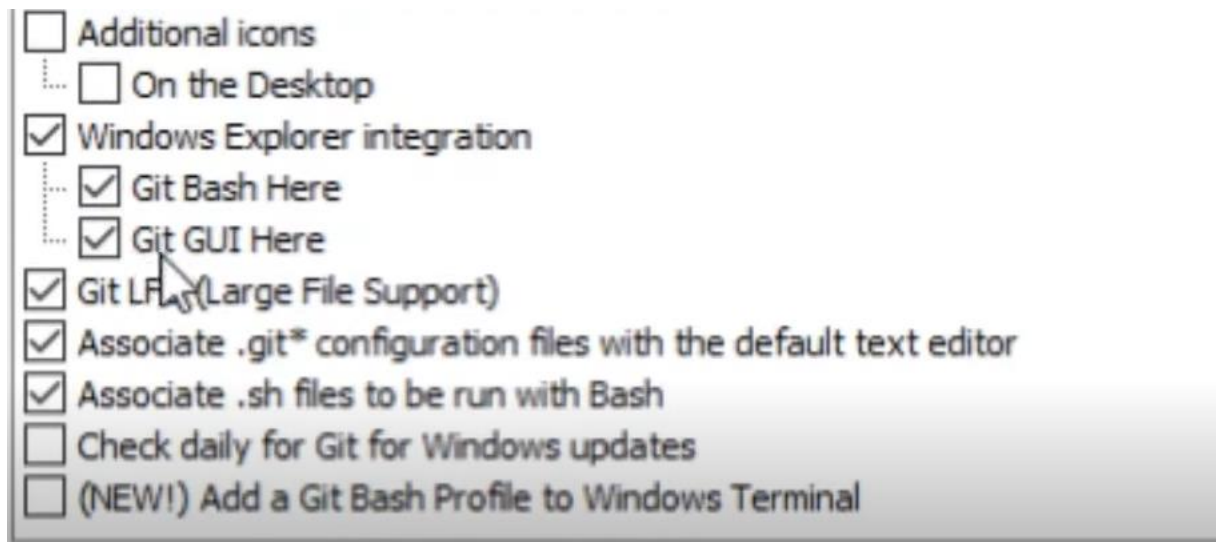
Podemos passar como parâmetro para um determinado arquivo se não existir o arquivo ele será criado.

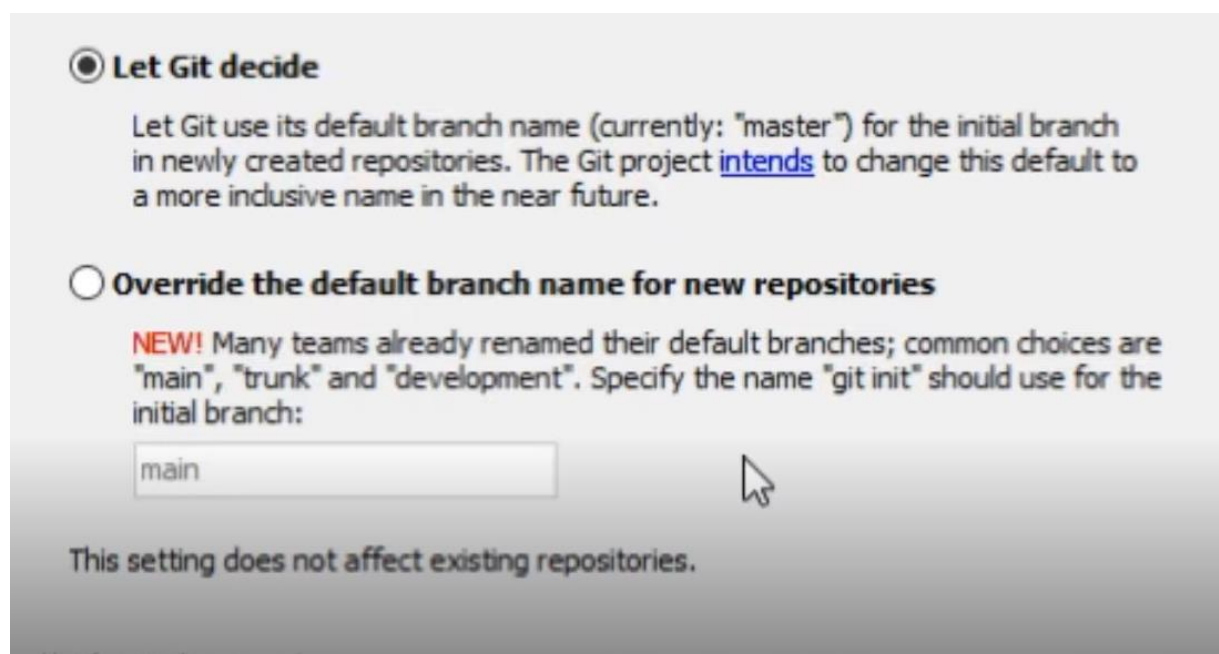
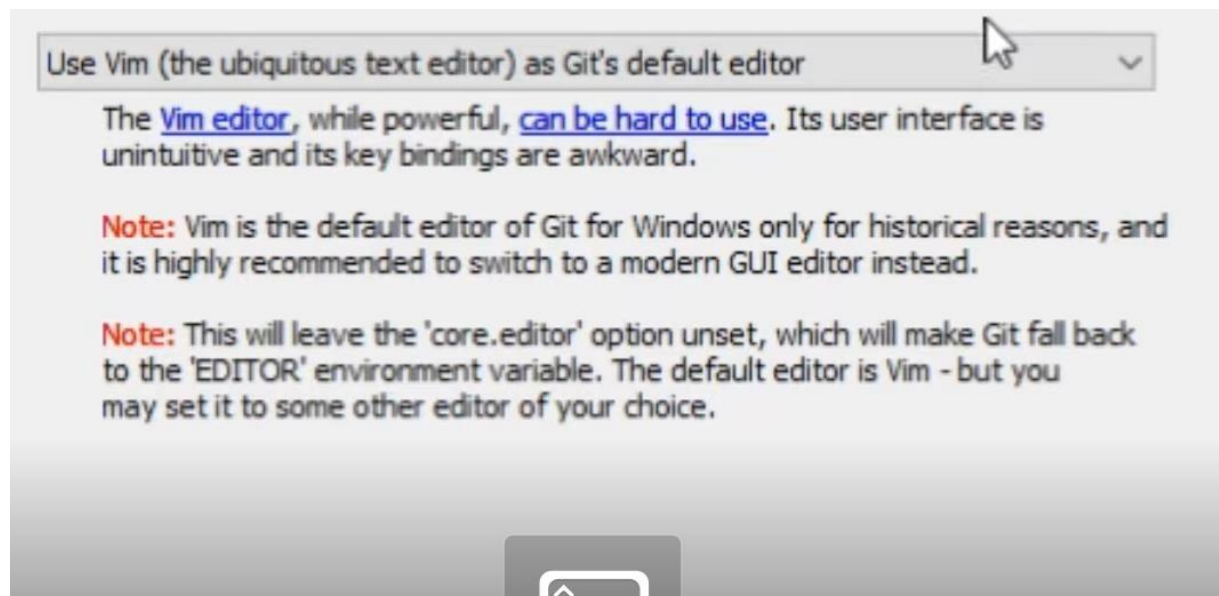
delete ou del-> Deleta um arquivo do diretório deve se passar o nome

rmdir-> deleta um repositório, deve se passar o nome do diretório:

```
C:\>rmdir workspace /S /Q
```

Instalação:





Nessa parte temos que entender que a primeira opção deixa a branch principal com o nome master, no outro podemos escolher o nome tendo como exemplo inicial main.

☐ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ **Git from the command line and also from 3rd-party software**

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

☐ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

☒ **Use bundled OpenSSH**

This uses ssh.exe that comes with Git.

☐ **Use (Tortoise)Plink**

To use PuTTY, specify the path to an existing copy of (Tortoise)Plink.exe:

C:\Program Files\PuTTY\plink.exe



☐ Set ssh.variant for Tortoise Plink

☐ **Use external OpenSSH**

NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

☒ **Use the OpenSSL library**

Server certificates will be validated using the ca-bundle.crt file.

☐ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

☒ **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

☐ **Checkout as-is, commit Unix-style line endings**

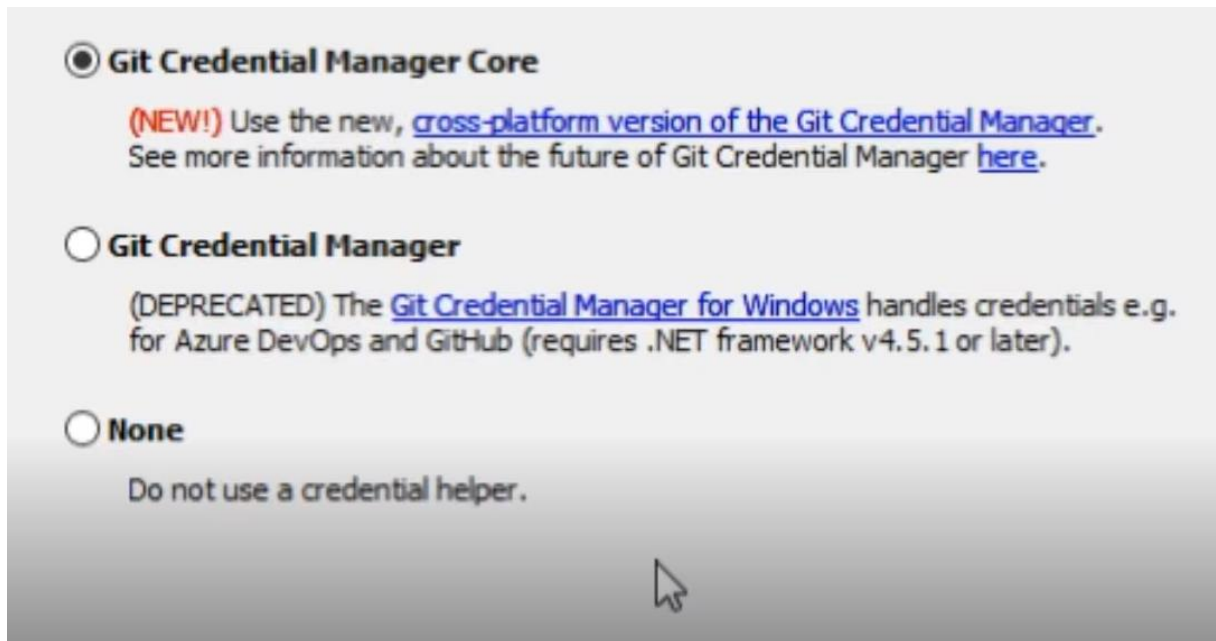
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

☐ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

[s://gitforwindows.org/](https://gitforwindows.org/)

Nessa parte escolhemos o tipo de quebra de linha do sistema operacional.



Aqui é a forma de gerenciador de credenciais, é importante utilizar a nova.

GIT por baixo dos panos:

SHA1->A sigla SHA significa Secure Hash OAlgorithm, é um conjunto de funções hash criptografadas projetadas pela NSA, gerando um conjunto de caracteres identificador de 40 dígitos.

```
$ openssl sha1 texto.txt  
SHA1(texto.txt)= 476ad3f360ced25383c25d13eae57f5  
e30cecb9
```

Objetos fundamentais->

BLOBS:

SHA1 as1s412 ...

Blob

Tamanho 42

\0

Ola Mundo

TREES:

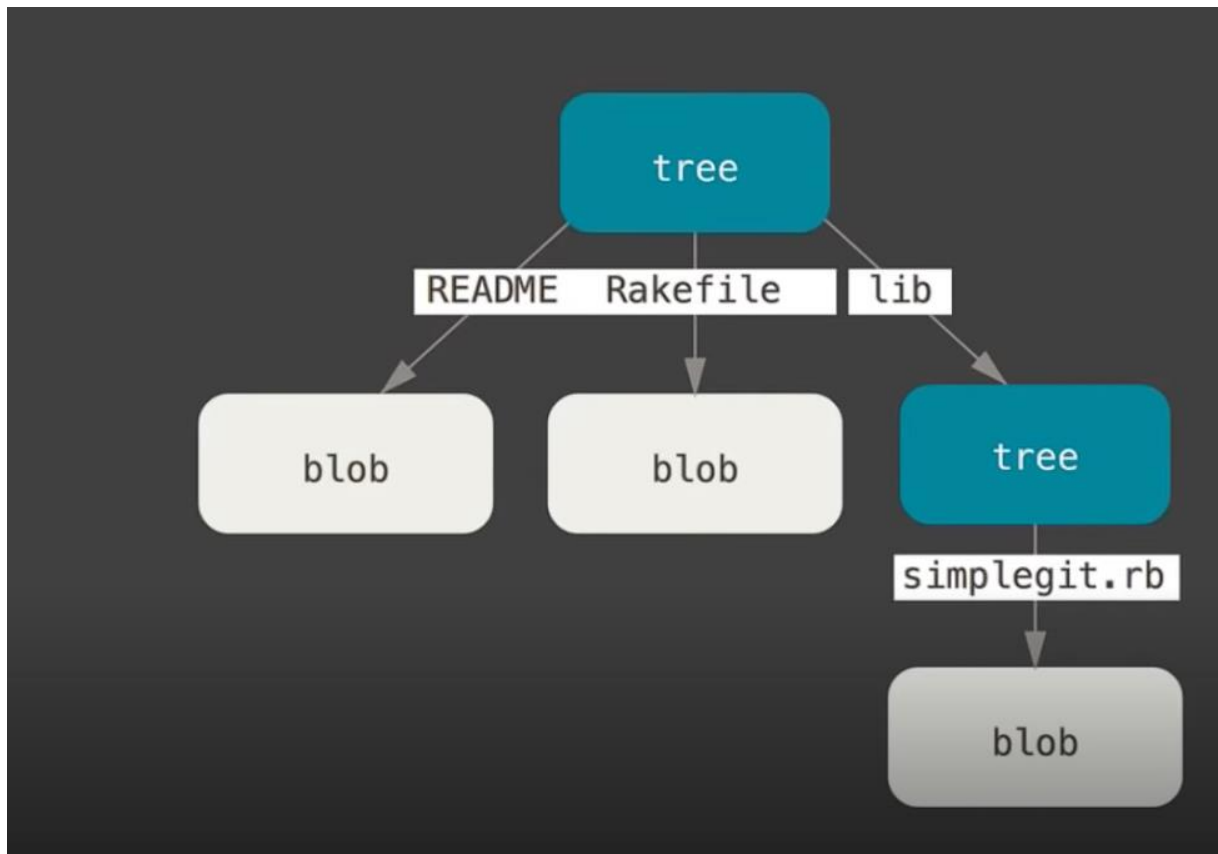
SHA1 hg1a8d...

Tree

<tamanho>

\0

blob sa4d8s texto.txt



COMMITTS:

SHA1 487d4s ...

Commit

<tamanho>

tree s4a5sq1

parente a98acq1

autor perkles

mensagem "inicia ..."

timestamp

O SHA1 desse commit é o hash de toda a essa informação

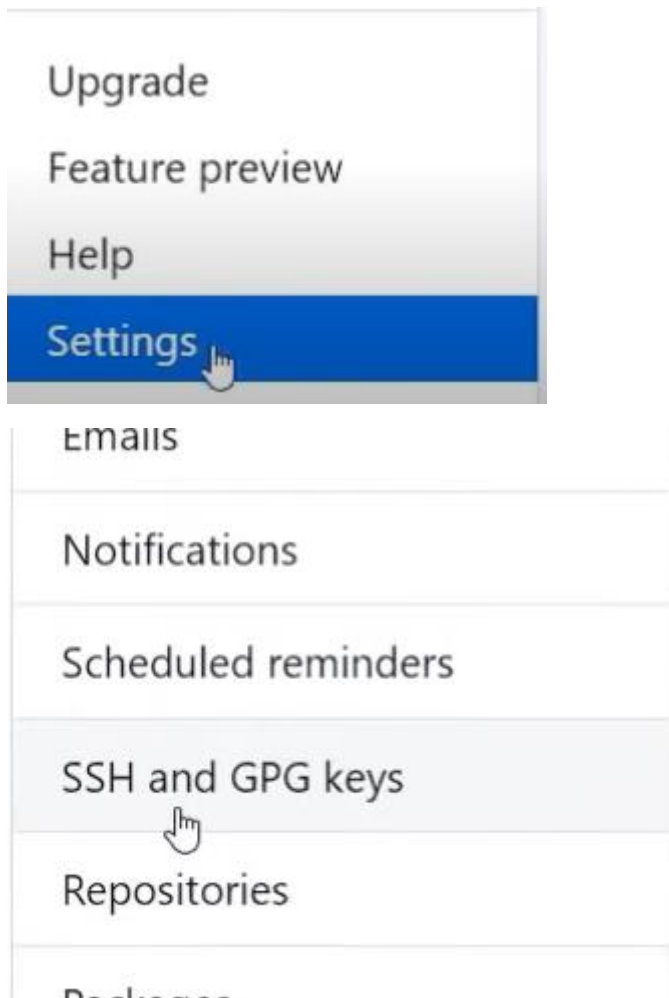
Simplificando um blob é um conjunto de dados encriptados, o blob vai estar em uma árvore que por sua vez é encriptada e vai conter outros blobs ou outras árvores, a árvore por sua vez fará parte de um commit que possuirá a nova árvore, a árvore anterior, o autor, o tamanho, uma mensagem e o tempo.

Se alterarmos qualquer coisa em uma blob sua hash irá mudar, consequentemente a hash da árvore mudará e consequentemente o commit possuirá outra hash.

Autenticação:

Chave SSH->É uma forma de se estabelecer uma conexão segura e encriptada entre duas máquinas, funciona através de duas chaves uma publica e a outra privada.

A chave pública fica no github:



SSH keys

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Title

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Para gerarmos a key devemos utilizar o git Bash:

```
$ ssh-keygen -t ed25519 -C otaviocha@gmail.com
```

Com esse comando iremos gerar uma chave com criptografia ed25519.

```
Enter file in which to save the key (/c/Users/Lucas/.ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:
```

Com isso ele irá passar um local onde a chave ficara guardada (podemos mudar), em seguida irá pedir uma senha.

Lendo uma chave pública :

```
otavio@perkles-desktop MINGW64 ~  
$ cd /c/Users/Lucas/.ssh/  
  
otavio@perkles-desktop MINGW64 ~/.ssh  
$ ls  
id_ed25519  id_ed25519.pub  
  
otavio@perkles-desktop MINGW64 ~/.ssh  
$ cat id_ed25519.pub |
```

Agora voltamos para o github e acrescentamos a chave:

Title

Minha maquina Windows

Key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJnnhAduq7P4gAApB7bsSPsAth20Dykq9LNf/X9WOBVJ
otaviocha@gmail.com
|

Add SSH key

Agora devemos criar um agente para gerenciar nossa chave privada:

```
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)
```

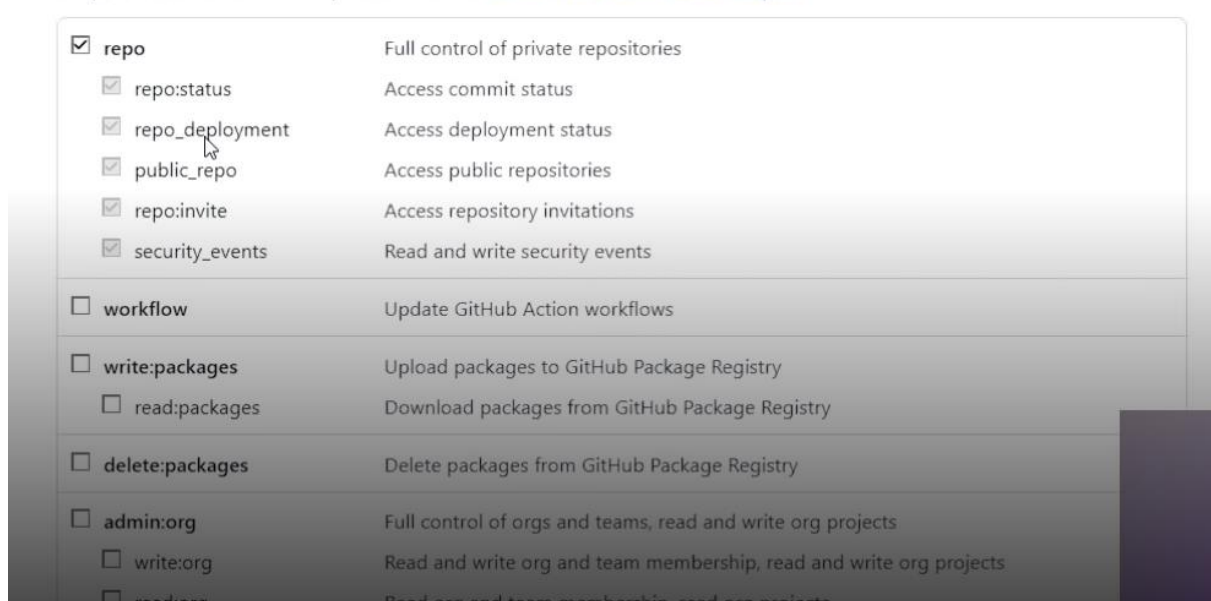
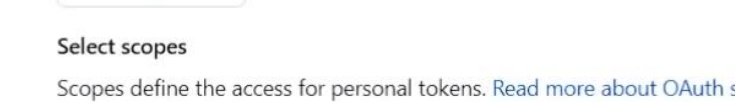
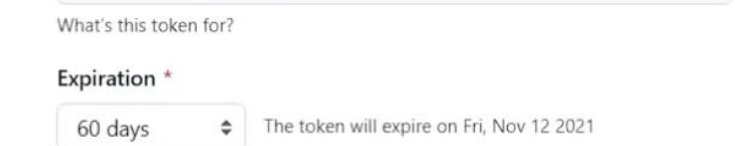
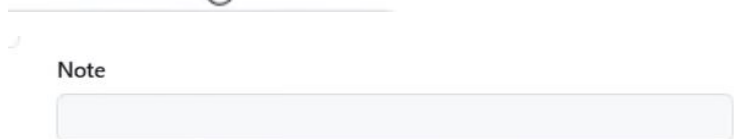
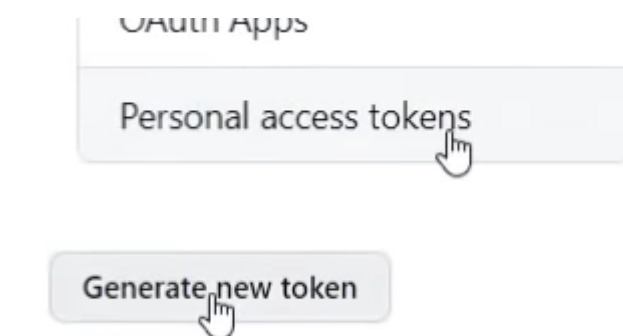
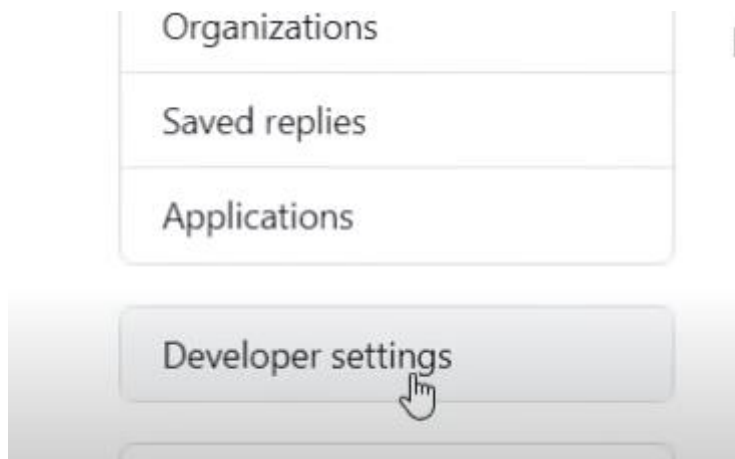
Clonando repositório pela chave ssh:


```
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
$ ls
perkles.github.io/
```

Token de acesso pessoal->é um token adicionado no github que podemos utilizar, para acessar o github pelo token só precisamos adicionar o usuário e no lugar da senha do usuário utilizar o token.

Sua vantagem é a simplicidade, porém todas vez será necessário adicionar usuário e senha.



✓ ghp_VqjFQg5IXjPkXdv1fsEhquaJNsVYx01aFIwH 

Delete

Comandos GIT:

git init-> cria o git em uma pasta possibilitando o versionamento.

```
otavio@perkles-desktop MINGW64 /c/workspace/livro-receitas
$ git init
Initialized empty Git repository in C:/workspace/livro-receitas/.git/
```

git config->cria uma configuração para o git, pode ser global ou especifica do repositório:

```
otavio@perkles-desktop MINGW64 /c/workspace/livro-receitas
$ git config --global user.email "otaviocha@gmail.com"

gabriel@VICTOR@DESKTOP-HK9LH2U MINGW64 ~
$ git config --global user.name Perkles
```

Podemos listar todas as configurações com --list:

```
otavio@perkles-desktop
$ git config --list
```

Para apagar uma configuração usamos --unset:

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

git add-> adiciona um arquivo para ser commitado.

git commit-> faz o commit do repositório:

```
otavio@perkles-desktop MINGW64 /c/workspace/livro-receitas (master)
$ git add *

otavio@perkles-desktop MINGW64 /c/workspace/livro-receitas (master)
$ git commit -m "commit inicial"
[master (root-commit) 94958ac] commit inicial
1 file changed, 21 insertions(+)
create mode 100644 strogonoff.md
```

git status-> diz o status dos arquivos:

```
$ git status
on branch master
nothing to commit, working tree clean
```

```
$ git status
on branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:      strogonoff.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        receitas/

no changes added to commit (use "git add" and/or "git commit -a")
```

Git remote add-> adicionamos a origem do repositório no servidor:

```
git remote add origin https://github.com/Perkles/livro-receitas.git
```

```
$ git remote -v
origin https://github.com/Perkles/livro-receitas.git (fetch)
origin https://github.com/Perkles/livro-receitas.git (push)
```

O origin é apenas um nome para o endereço para que não tenhamos que ficar digitando o link.

git push-> enviamos o repositório local para o repositório remoto:

```
git push origin master
```

git pull-> puxa o repositório do servidor

```
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 701 bytes | 50.00 KiB/s, done.
From https://github.com/Perkles/livro-receitas
 * branch                master      -> FETCH_HEAD
    54c6046..e1dee1e      master      -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

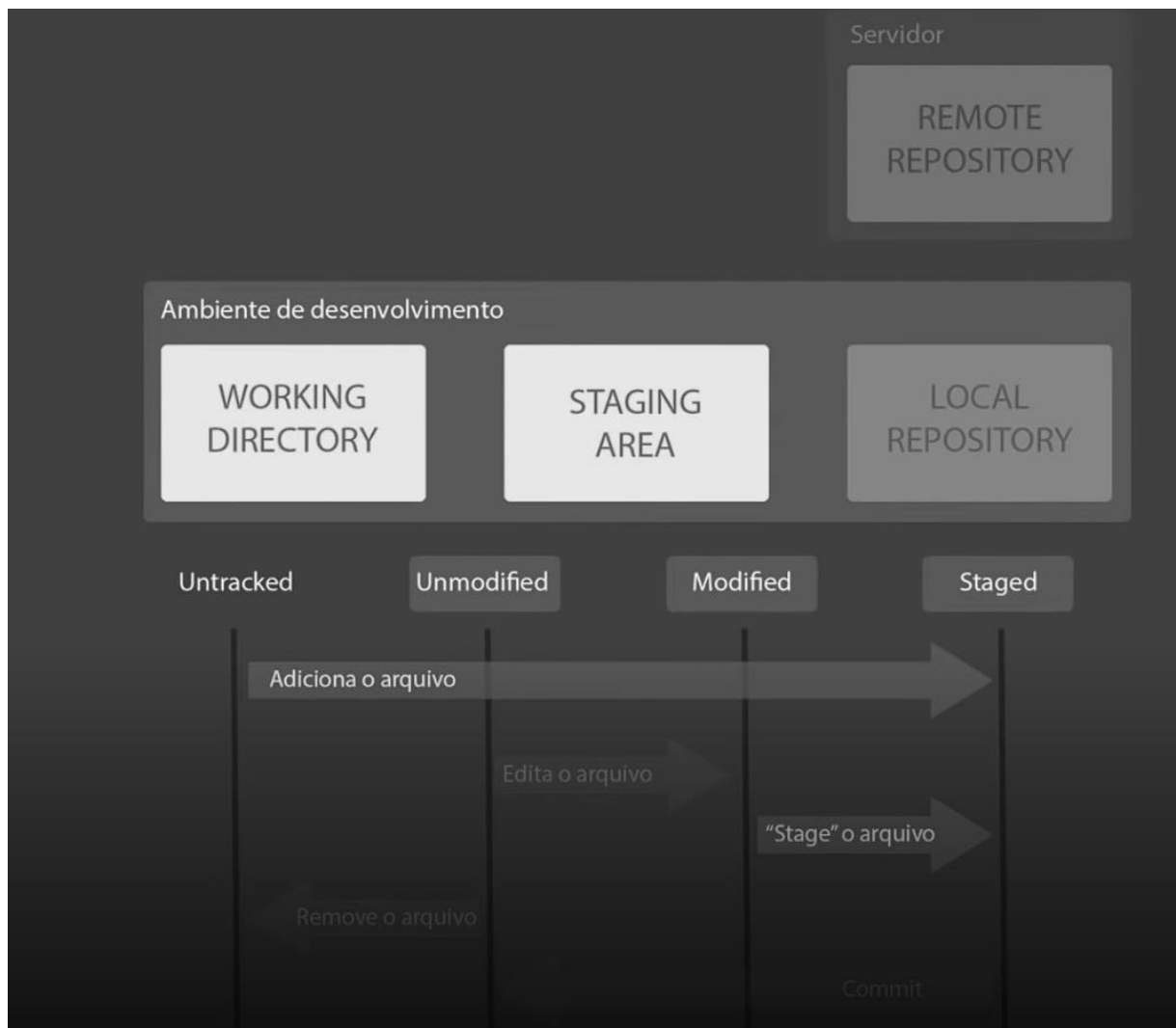
Git clone-> clona um repositório:

```
$ git clone https://github.com/python/cpython.git
```

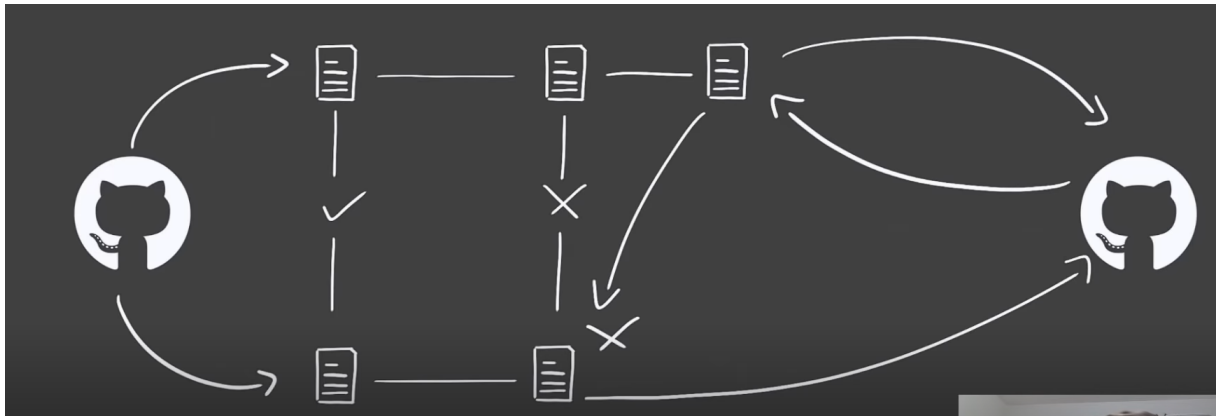
Fazendo isso o repositório já vem com a pasta .git dentro dele.

Ciclo de vida:





Resolvendo conflitos:



Esse conflito ocorre quando dois usuários usando o mesmo repositório alteram a mesma linha do repositório ao mesmo tempo.

```
Octavio@perkles-desktop MINGW64 /C:/workspace/livro-receitas (master)
$ git push origin master
To https://github.com/Perkles/livro-receitas.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Perkles/livro-receitas.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Para resolvermos temos que dar um pull no repositório para identificar a linha com problema, modificarmos ela manualmente e em seguida commitar e dar o push.