INTRODUÇÃO AO GIT

NAVEGAÇÃO VIA COMMAND LINE INTERFACE E INSTALAÇÃO

Comandos básicos para um bom desempenho no terminal

	Windows	Unix
Mudar de pastas	cd	cd
Listar as pastas	dir	ls
Criar pastas/ arquivos	mkdir	mkdir
Deletar pastas/ arquivos	del/ rmdir	rm -rf

Retornar para pasta anterior “cd..”
Limpar a tela “cls” (Windows) “clear” ou CRTL + L (Linux)
Auto completar tecla “TAB”
Imprimir texto no terminal “echo”
Redirecionador de fluxo “>”

TÓPICOS FUNDAMENTAIS PARA ENTENDER O FUNCIONAMENTO DO GIT

SHA1 (Secure Hash Algorithm/ algoritmo de Hash Seguro)

É um conjunto de funções hash criptografadas projetadas pela NSA (Agência Nacional dos EUA)
A encriptação gera conjunto de caracteres identificador de 40 dígitos.
SHA1 é uma forma curta de representar um arquivo.

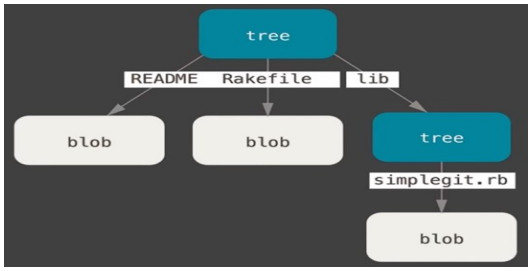
```
1 echo "ola mundo" | openssl sha1
2 > (stdin)= f9fc856e559b950175f2b7cd7dad61facbe58e7b
```

OBJETOS FUNDAMENTAIS

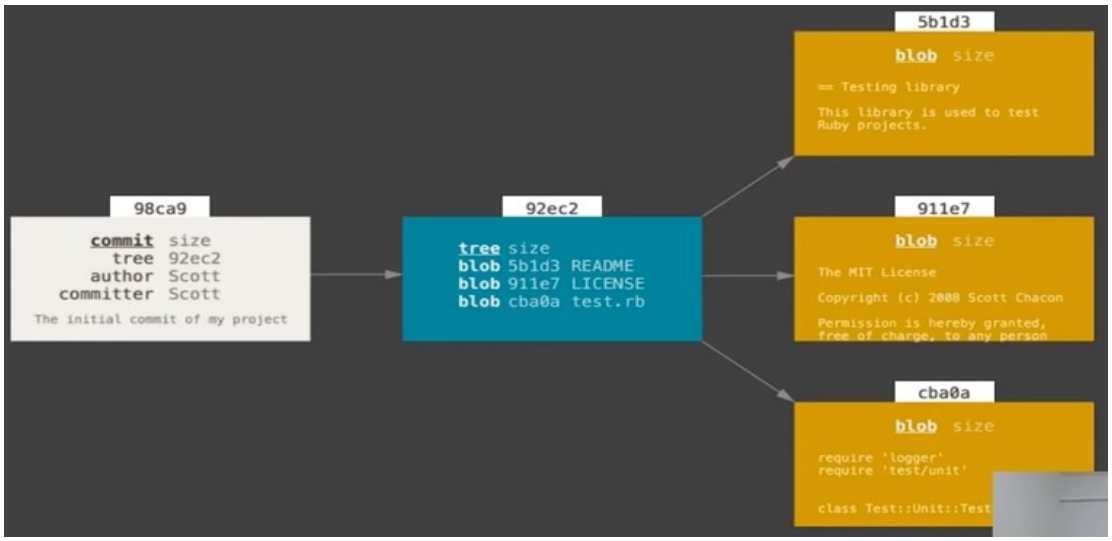
BLOBS – Tem o tipo do objeto, tamanho do arquivo, uma \0 e o conteúdo do arquivo (binário, texto, etc). Ele contem os metadados.

```
1 echo 'conteudo' | git hash-object --stdin
2 > fc31e91b26cf85a55e072476de7f263c89260eb1
3
4 echo -e 'blob 9\0conteudo' | openssl sha1
5 > fc31e91b26cf85a55e072476de7f263c89260eb1
```

TREES – Armazenam BLOBS, ela é responsável por montar toda a estrutura e onde estão localizados os arquivos, ela pode tanto apontar para os BLOBS quanto para outras TREES e também contem uma chave SHA1



COMMITS – É o objeto mais importante, ele é o responsável por juntar tudo e dar sentido para as alterações que são feitas. Ele aponta para uma “árvore”, aponta para o último COMMIT realizado antes dele, aponta para o autor e para a mensagem do autor tem uma timestamp (data e hora que foi criado) e também um SHA1.



Sistema distribuído – para manter a segurança dos arquivos.

CHAVES SSH

É uma forma de estabelecer uma comunicação com encriptação entre duas máquinas.
Criando as chaves:
No Git Bash digite:
$ ssh-keigen -t ed25519 -C email@gmail.com
Obs.: O e-mail é o que foi usado no GitHub

```
Otavio@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:CGDr3Aa2UUoQJ9xslkKTWzGWwJOAamh6xnbS9RcZUco otaviocha@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|@B%O=     oo.    |
|+XO#.    . +     |
|O+% . .   E      |
|+* * o o   .     |
|. X = . S .      |
| + +     .       |
|                 |
|                 |
|                 |
+----[SHA256]-----+
```

Esse comando vai criar duas chaves, uma privada e outra publica e vai pedir para cadastrar uma senha, para conseguir copiar a chave é necessário entrar no diretório pelo Git bash e depois digite:
$ *cat nomedachave.pub*

Ele vai gerar um código que vai ser usado no site GitHub.
No site GitHub entre em configurações > SSH and GPG keys digite o nome para identificar e cole o código na janela “Key”, vai pedir a senha do site e depois estará terminado, voltando ao computador é necessário abrir novamente o Git Bash entrar na pasta onde estão as chaves e então digite:
$ eval $(ssh-agent -s)
Vai iniciar o processo do “agent ssh”, confirme o conteúdo da pasta com “ls”, após isso digite:
$ ssh-add nomedachaveprivada
Entre com a senha que foi pedida na criação das chaves, a saída será “Identity added: nomedachaveprivada (email@gmail.com).
Clonando um repositório no GitHub: digite:
$ git clone endereçoSSH
Para repositórios privados é necessário chave de acesso do proprietário do repositório.

INICIANDO O GIT E CRIANDO UM COMMIT

Iniciar o Git	git init
Iniciar o versionamento	git add
Criar um commit	git commit

Criando um repositório – crie uma pasta com o nome “workspace” no diretório C, dentro da pasta crie uma pasta chamada “livro-receitas”.
A partir do explorer entre dentro no diretório C, utilizando o botão direito do mouse abra o menu e selecione “Git Bash Here”, ele vai abrir o terminal do Git dentro do diretório C.
Utilizando o terminal do Git navegue até a pasta “livro-receitas” utilizando o comando
$ cd livro-receitas/
Agora inicie o Git com o comando
$ git init
Foi criado um repositório vazio no endereço “C:/workspace/livro-receitas/.git” para poder listar a pasta oculta .git é necessário o comando
$ ls -a
Na pasta .git fica os arquivos gerenciais do Git.
Configurando o Git – continuando utilizando o mesmo terminal com o Git inicializado digite:
$ git config --global user.email “seuemail.com”
Configurando o usuário – digite:
$ git config --global user.name seunome
Criando um arquivo markdown – dentro do repositório “livro-receitas” crie um documento de texto do Notepad e modifique sua extensão para .md que é a extensão utilizada pelos leitores de arquivo markdown, não esqueça de instalar um editor de arquivos markdown.
Criando um commit do arquivo – digite:
*$ git add **
$ git commit -m “commit inicial”
Verificando o status do repositório – digite:
$ git status

```
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

Otavio@perkles-desktop MINGW64 /c/workspace/livro-receitas (master)
$
```

CICLO DE VIDA DOS ARQUIVOS DO GIT

Os arquivos git ficam em um ciclo que se inicia na criação do repositório e vão sendo adicionados arquivos, onde eles devem ser reconhecidos pelo git, ou seja, tem de passar de não monitorados (untracked) para monitorados (tracked) e após isso ficam em um estagio de espera para serem modificados (modified) ou sem mudanças (unmodified), esse estagio é chamado de staged, onde eles estão prontos para fecharem um pacote e receber a classificação, data e comentários do autor do pacote (commit), assim que recebem o commit eles retornam para o estagio de monitorados, mas estarão com a classificação de não modificado (unmodified), reiniciando o ciclo para novas modificações.

Criando repositório
$ git init

Tracked – são os arquivos que são reconhecidos pelo git eles podem estar em três estágios no git:
Unmodified – não modificado.
Modified (modificado) - são arquivos monitorados pelo git (tracked) e que foram feitas alterações, modificando a chave de criptografia SHA1.
Staged – são arquivos que estão sendo monitorados pelo git e prontos para fechar o pacote de arquivos recebendo o commit.
Untracked – são os arquivos que não são reconhecidos pelo git.

Converter um arquivo de untracked para tracked
$ git add <nome do arquivo>

Ou converte todos os arquivos do repositório para tracked
*$ git add**

AMBIENTES DE TRABALHO

Temos dois ambientes de trabalho no git um sendo o ambiente de desenvolvimento, ou seja, a máquina local e o servidor onde fica o repositório remoto.
Ambiente de desenvolvimento: Working Directory, Staging Area e Local Repository
Servidor: Remote Repository (GitHub)

TRABALHANDO COM O GITHUB

Para evitar erros no repositório o e-mail e o nickname devem ser iguais aos da plataforma GitHub pois caso necessite modificar os arquivos na plataforma online do GitHub podem ocorrer diferenças por conta do commit.
Para listar as configurações utilizadas no Git digite:
$ git config --list
Apagando as configurações do Git digite:
$ git config --global --unset <configuração que deseja modificar>
Reescrevendo as configurações do Git digite:
$ git config --global <configuração que deseja modificar> "Nome, e-mail ou nickname"

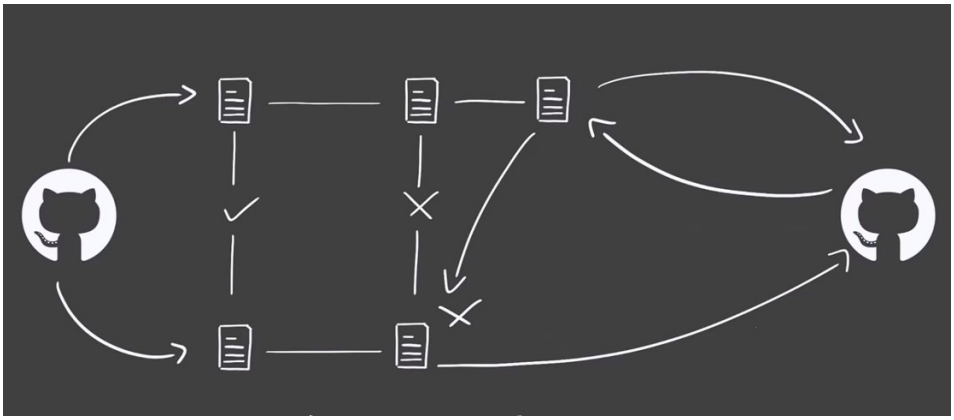
Site do GitHub – Após logar entre em seus repositórios (Your repositories), no botão verde "New", selecione seu usuário e digite o nome do repositório, a descrição é opcional. Selecione se o repositório vai poder ser visto publicamente (Public) ou somente por você (Private).
Após confirmar as informações clique em criar o repositório (Create Repositories), vai ir para a página do GitHub onde ele passa os comandos que vão ser necessários para fazer upload dos arquivos para a plataforma, criar um repositório ou importar código de outro repositório.

Copie a linha que inicia com “git remote add origin https://….”
Agora com o terminal Git Bash aberto cole e execute o comando depois verifique os repositórios cadastrados digitando:
$ git remote -v
Verifique o status do Git digitando:
$ git status
Para enviar o arquivo para a plataforma GitHub digite:
$ git push origin master
Vai pedir para se conectar com a plataforma a partir de uma identificação e após isso vai ser feito o carregamento do repositório para o servidor remoto.

```
$ git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.26 KiB | 429.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/droveio/livro-receitas.git
 * [new branch]      master -> master
```

Para sincronizar os arquivos do repositório digite:
$ git pull origin master

RESOLVENDO CONFLITOS

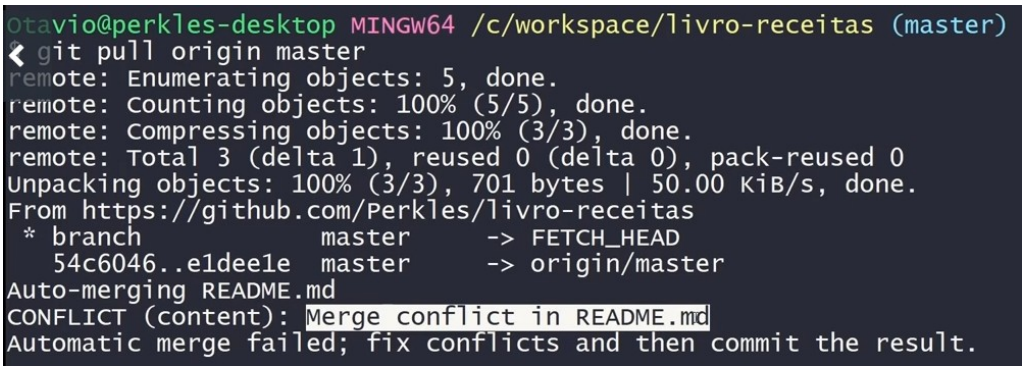


A plataforma do GitHub é colaborativa a partir disso podem acontecer erros ou conflitos por ter várias pessoas fazendo modificações no repositório ao mesmo tempo, um deles é conhecido por “merge”, quando vai ser feito o carregamento do repositório para a plataforma GitHub pode ocorrer um erro assim:

```
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.
```

Na identificação do erro apresenta que o repositório local está desatualizado em comparação com a da plataforma GitHub e também apresenta a forma que deve ser feito a correção onde deve ser feito uma sincronização dos repositórios, ou seja, deve-se baixar os arquivos que estão na plataforma,

porém fazendo isso, vai aparecer outro erro caso o arquivo que esteja trabalhando seja o mesmo que foi modificado na plataforma, criando um conflito. O Git vai tentar sincronizar os arquivos mas caso ocorra um erro, ele vai identificar o arquivo que foi modificado e pedir para ser feito a correção do erro.



Consertando o erro – Baixando os arquivos do GitHub digite:
$ git pull origin master
Identifique o erro no terminal e conserte o conflito (merge). Modificando o arquivo é necessário verificar o status do repositório, digite:
$ git status
Atualize os arquivos que não estão sendo monitorados pelo Git, digite:
*$ git add **
Não havendo pendências e necessário criar outro commit para fechar o pacote, digite:
$ git commit -m "Comentário"
Carregando o repositório para o GitHub, digite:
$ git push origin master