INTRODUÇÃO AO GIT

Entendendo o que é o Git e sua importância.

Git - software local (servidor local).
É um sistema de versionamento de código (VCS - Version Control System) distribuído.
Foi desenvolvido Linus Torvalds (2005). “Um software colaborativo não é coisa de um homem só”.

Git e GitHub não são a mesma coisa, são tecnologias complementares, mas diferentes.

Git - (Servidor local). Utilizado para criar e monitorar diferentes versões. É open source.

GitHub - (Servidor Remoto). Repositório online em que armazenamos nosso código. É uma espécie de rede social e pertence à Microsoft. Algumas funções são pagas.

Benefícios de aprender as duas tecnologias juntas:

1. Controle de Versões
2. Armazenamento em nuvem
3. Trabalho em Equipe
4. Melhorar seu código
5. Reconhecimento

NAVEGAÇÃO VIA COMMAND LINE INTERFACE E INSTALAÇÃO

Comando básicos para um bom desempenho do terminal.
Navegação básica no terminal e como instalar o Git

O Git é um CLI - (Command Line Interface) utiliza o terminal por linha de comando - Ele não tem uma interface gráfica (GUI - Graphic User Interface)

O que Vamos Aprender?

Mudar de pastas
Listar Pastas
Criar Pastas / Arquivos
Deletar pastas / Arquivos

Existem comandos diferentes para o sistema operacional Windows ou para o Unix.

No terminal do Windows (Shell)
cd - mudar de diretório (NAVEGAR ENTRE DIRETÓRIOS)
dir - listar diretórios
mkdir - criar uma pasta (criar diretórios)
del / rmdir - deletar diretórios
cls - limpar a tela

TAB - tem a função de autocompletar
echo - retorna o que foi digitado
cd .. - Volta

No terminal UNIX Linux (Bash)

cd - mudar de diretório (NAVEGAR ENTRE DIRETÓRIOS)
ls - listar diretórios
mkdir - criar uma pasta (criar diretórios)
rm-rf - deletar diretórios (Deleta todas as pastas)
clear - limpar a tela (CTRL + L)

No controle de versão (git) um repositório é um local onde os arquivos são armazenados e monitorados, já em uma pasta ou diretório qualquer da sua máquina isso não acontece a menos que você inicialize um no repositório.

REALIZANDO A INSTALAÇÃO DO GIT

Aula para ensinar apenas como instalar o GIT, seguindo exatamente as instruções do professor tudo fica muito fácil**.**

ENTENDENDO COMO O GIT FUNCIONA POR DEBAIXO DOS PANOS

Tópicos Fundamentais para entender o funcionamento do Git

SHA 1
Objetos Fundamentais
Sistema Distribuído
Segurança

SHA 1 - A sigla SHA significa Secure Hash Algorithm (Algoritmo de Hash Seguro) é um conjunto de funções hash criptografadas projetadas pelo NSA (Agência de Segurança Nacional dos EUA).
A encriptação gera conjunto de caracteres identificador de 40 dígitos.
É uma forma curta de representar um arquivo.
É único e serve de identificador.
É uma forma de identificar que os arquivos sofreram alguma alteração.

No Git Bash Here (Terminal do Git nele é possível trabalhar direto na pasta que temos interesse) digitar o comando openssl sha1 (para gerar o conjunto de 40 caracteres.

Objetos Interno do GIT

São responsáveis pelo versionamento do código.

Blobs
Trees
Commits

Objeto Blob - Os arquivos ficam guardados nesse objeto chamado blob, esse objeto contém metadados dentro dele. A blob contém o SHA1 dos arquivos.

```
1 echo 'conteudo' | git hash-object --stdin
2 > fc31e91b26cf85a55e072476de7f263c89260eb1
3
4 echo -e 'conteudo' | openssl sha1
5 > 65b0d0dda479cc03cce59528e28961e498155f5c
```

Obs: Gerou Chaves diferentes, porque o blob contém metadados.

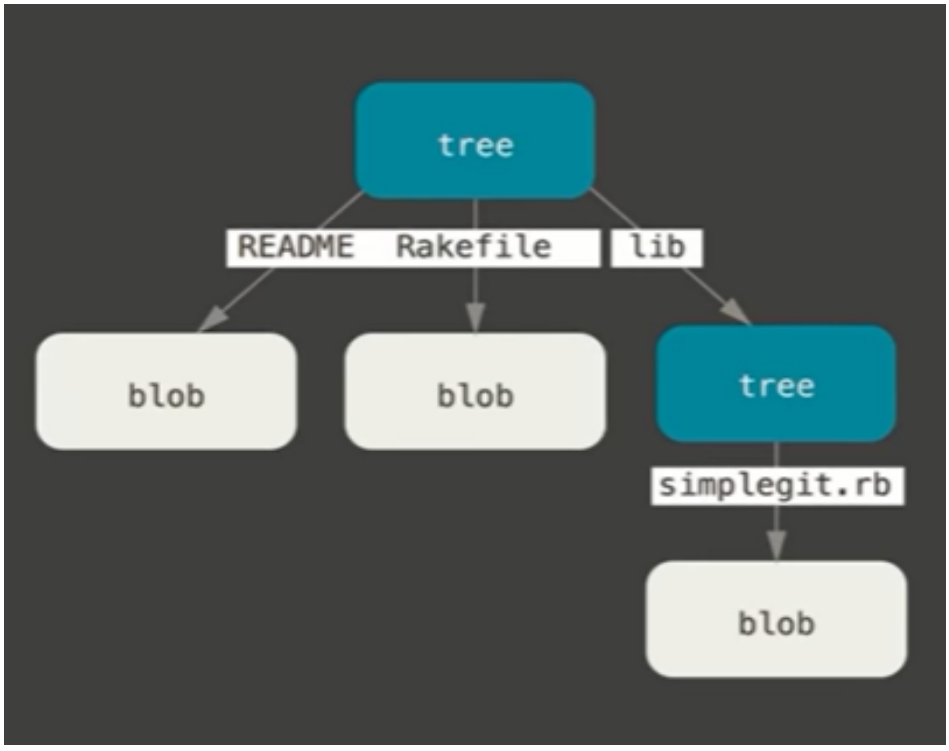
Estrutura básica desse objeto: tipo, tamanho, \0 e o conteúdo (nesse caso Ola Mundo). O blob guarda o SHA1 dos arquivos, não guarda o nome.

Obs: Ao inserirmos no comando o blob, a chave gerada é a mesma.

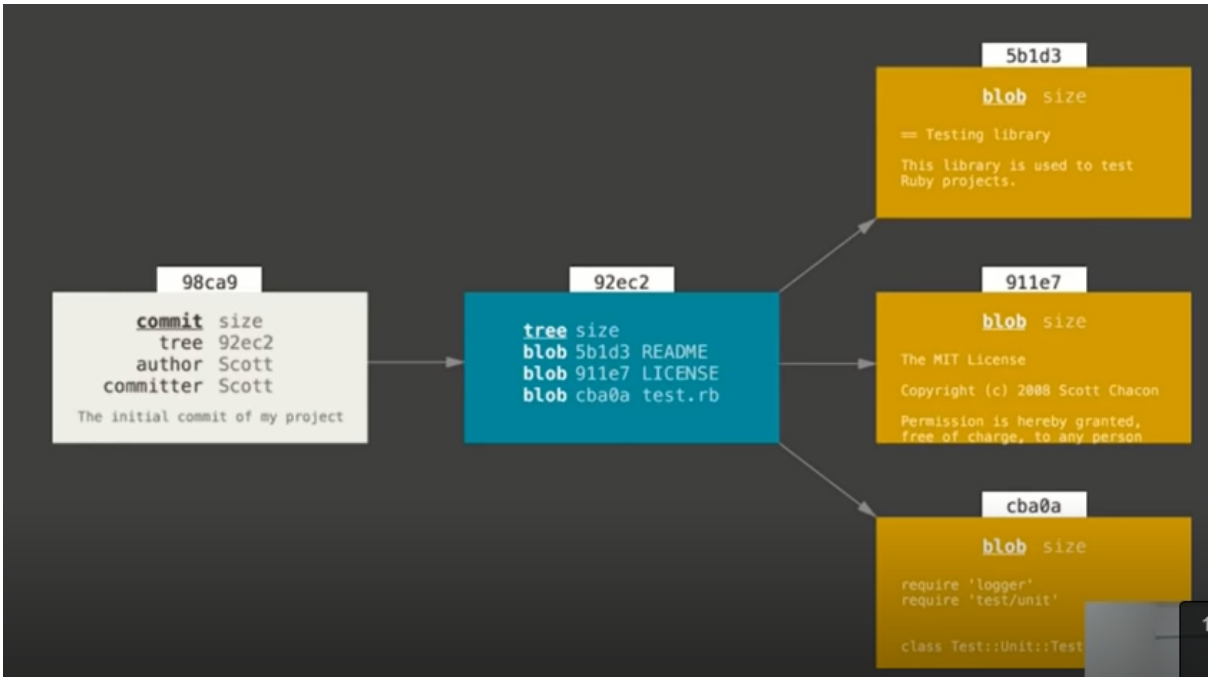
```
1 echo 'conteudo' | git hash-object --stdin
2 > fc31e91b26cf85a55e072476de7f263c89260eb1
3
4 echo -e 'blob 9\0conteudo' | openssl sha1
5 > fc31e91b26cf85a55e072476de7f263c89260eb1
```

Objeto Tree - A tree armazenam blobs (A blob é o bloco básico de composição). A tree armazena e aponta para tipos de blobs diferentes. O blob é um objeto que encapsula esse comportamento de diretórios, ele é usado para apontar para diretórios que contém arquivos. A árvore é responsável por montar toda a estrutura de onde estão localizados os arquivos, apontam para blobs (arquivos) e também para outras árvores, porque os diretórios de um sistema operacional podem conter outros diretórios, sendo assim a tree é um objeto recursivo.A tree também contém metadados e possui o SHA1 desse metadado. A tree guarda o nome do arquivo, já o blob não guarda.

Qualquer alteração no arquivo muda toda a estrutura desse objeto.



Objeto Commit - é o principal objeto, ele que vai juntar tudo, que vai dar sentido para a alteração que você está fazendo. O commit aponta para uma árvore, aponta para um parente (para o último commit realizado antes dele), aponta para o autor e aponta para uma mensagem também. O commit tem também um timestamp que é um carimbo de tempo (Data e hora de quando ele foi criado).O commit também possui um SHA1, se for realizada alguma alteração de dado na blob será gerado um SHA1 daquele blob, essa blob tem uma árvore apontando para ela, portanto vai alterar os metadados daquela árvore, porque aquela árvore aponta para uma blob.
O commit aponta para uma árvore, que por sua vez, aponta para outras árvores, dentro de outras árvores.



O commit é único para cada autor.
O git é um sistema distribuído e seguro, porque qualquer alteração no arquivo muda toda a estrutura.
A versão que está no GitHub é a mais atualizada.

Chave SSH e Token

Chave SSH

A chave SSH é uma forma de estabelecer uma conexão segura e encriptada entre duas máquinas. Estabelecendo essa conexão com 2 chaves, uma pública e uma privada.

Com a chave SSH, não precisamos informar nosso usuário e senha em cada acesso. Basta configurar nossas credenciais uma vez e depois sempre poderemos acessar nosso GitHub sem informá-las, pois o SSH é usado especialmente para realizar a conexão com servidores remotos.

O que precisamos fazer é criar uma chave SSH em nosso computador local.

No Git Bash digitar o comando:

```
$ ssh-keygen -t ed25519
```
Aqui colocar o email que você usa no seu GitHub.

Abaixo os comandos para localizar no computador a pasta que estão as chaves, no caso somente a chave pública (.pub) será utilizada no GitHub.

```
$ cd /c/Users/Lucas/.ssh/
```

```
$ ls
id_ed25519  id_ed25519.pub
```

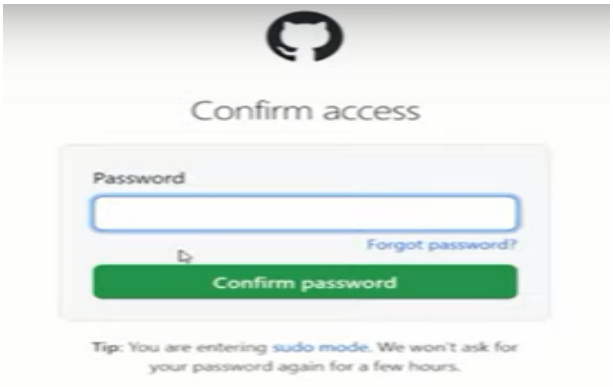
```
$ cat id_ed25519.pub
```
Comando que mostra a chave pública.

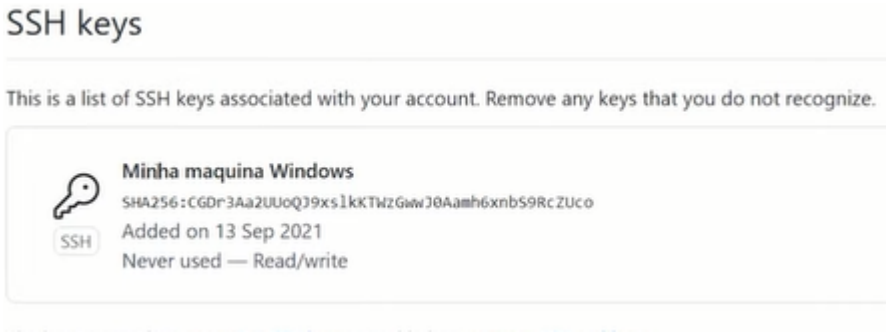
```
ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJnnhAduq7P4gAApB7bsSPsAth20Dykq9LNf/X9WOBV
```

Copiar a chave e adicioná-la no GitHub.

Após adicioná-la ao GitHub

O sistema vai pedir autenticação da credencial.





Essa chave é como uma credencial. Uma chave SSH vai nos permitir acessar o GitHub remotamente.

Para validar a chave no GitHub é necessário seguir os seguintes passos:

No Git Bash entrar na pasta. ssh

Digitar os comando:

```
$ eval $(ssh-agent -s)
Agent pid 1382
```

```
$ ssh-add id_ed25519
```
Temos que informar a chave privada

```
Enter passphrase for id_ed25519:
```
Vai pedir para informar a senha.

No GitHub em Code copiar o caminho da aba SSH.

No Git Bash digitar o comando git clone e inserir o caminho SSH para configurar a chave SSH no computador.

Token

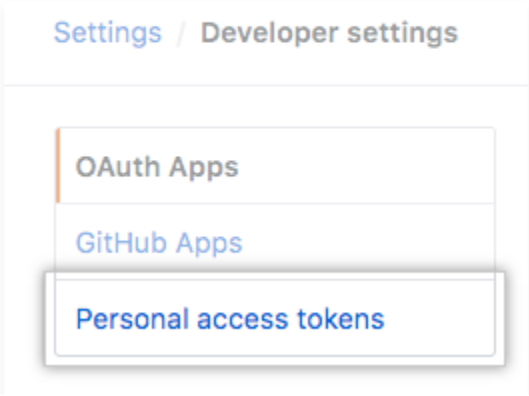
Token de acesso pessoal.

Para gerar o Token no GitHub seguir os seguintes passos:

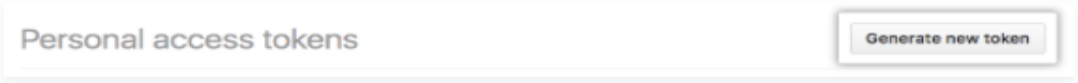
No canto superior direito de qualquer página, clique na sua foto de perfil e, em seguida, clique em Configurações.

Na barra lateral esquerda, clique em Developer settings (Configurações do desenvolvedor).

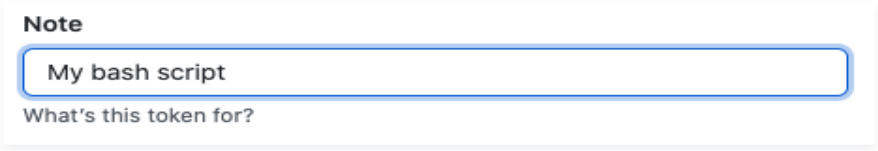
Na barra lateral esquerda, clique em **tokens de acesso pessoal**.



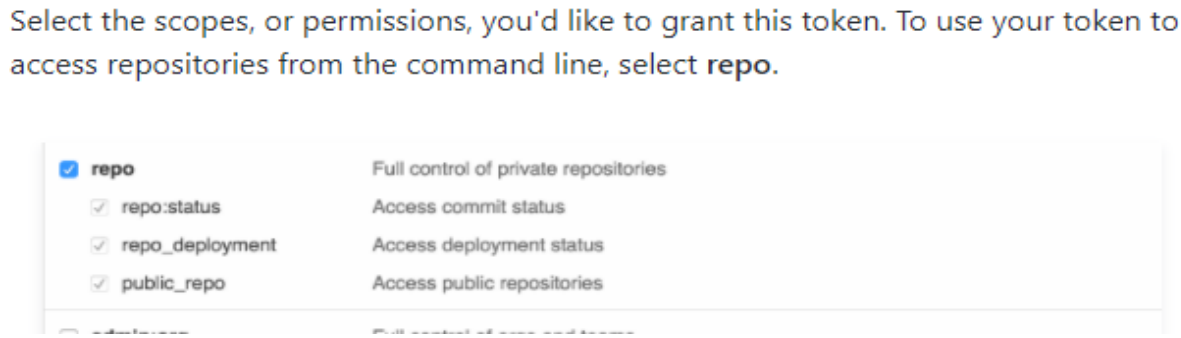
Clique em **Generate new token** (Gerar novo token).



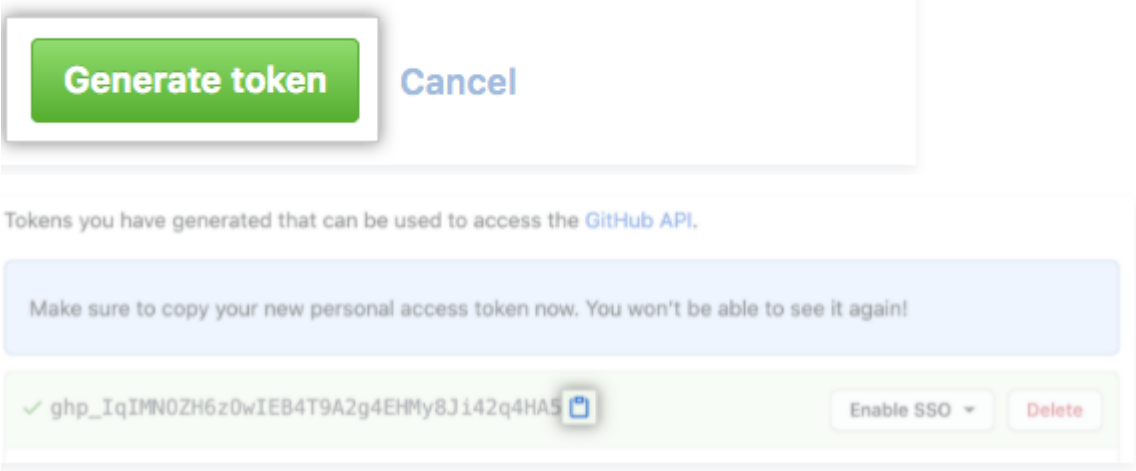
Give your token a descriptive name.



To give your token an expiration, select the **Expiration** drop-down menu, then click a default or use the calendar picker.



Click Generate token.



Copiar o token para inseri-lo no Git Bash. O programa orienta também que o token seja guardado em um local seguro, porque não será possível visualizá-lo mais no GitHub.
Ainda no GitHub copiar em Code o caminho HTTPS

No Git Bash digitar o comando:**git clone** e colar o caminho HTTPS, em seguida vai abrir uma tela para você colocar o token.

PRIMEIROS COMANDOS DO GIT

Iniciando o Git e criando um commit

Iniciar o GIT
Iniciar o Versionamento
Criar um Commit

git init - iniciar um repositório no Git
git add - mover arquivos e versionamento
git commit - criar o commit

git na frente do comando significa que você está chamando pelo terminal o git.

Criando um repositório

Primeiro é necessário ir ao diretório desse projeto utilizando o comando:

cd

Para criar começar a realizar os registros do seu projeto, basta se deslocar via terminal até o diretório onde os arquivos do projeto estão localizados e entrar com o seguinte comando:

git init - inicia um repositório no Git.
Isso cria um novo subdiretório chamado .git que contém todos os seus arquivos de repositório necessários — um esqueleto de repositório Git.

Com o comando **mkdir** é possível criar uma nova pasta.
Com o comando **ls** é possível listar as pastas.
Flag **Ls -a** mostra arquivos ocultos.
Para voltar um nível **cd ..**

Primeira vez que se utiliza o git é necessário configurar o nome do usuário e o email:

$ git config --global user.name "Fulano de Tal"
$ git config --global user.email fulanodetal@exemplo.br

Markdown - é uma linguagem simples de marcação originalmente criada por John Gruber e Aaron Swartz. Markdown converte seu texto em HTML válido. Markdown é frequentemente usado para formatar arquivos README. Possui a extensão .md

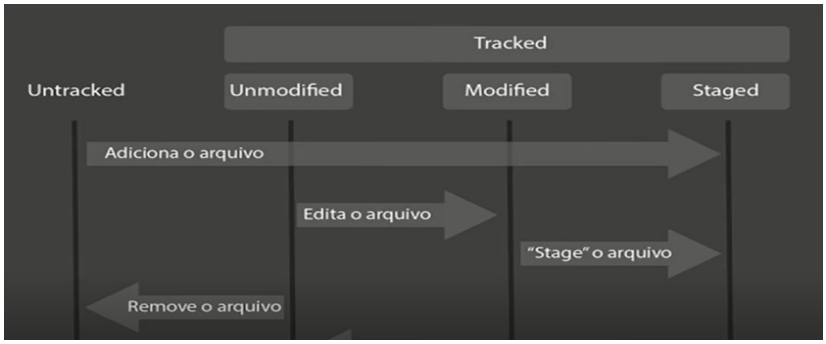
git add * - Adiciona todos os arquivos com uma determinada extensão. Por exemplo, git add *.html (adiciona todos os arquivos que possuem a extensão .html).

git commit -m "Mensagem" - Esse comando cria o commit no repositório do Git, com o comentário que está entre aspas.

CICLO DE VIDA DOS ARQUIVOS NO GIT

Passo a passo no ciclo de vida.

Tracked
Untracked



Untracked é o estado onde o arquivo ainda não foi 'rastreado'.

Tracked

Arquivos que podem ser rastreados e podem ser divididos em 3 estágios diferentes

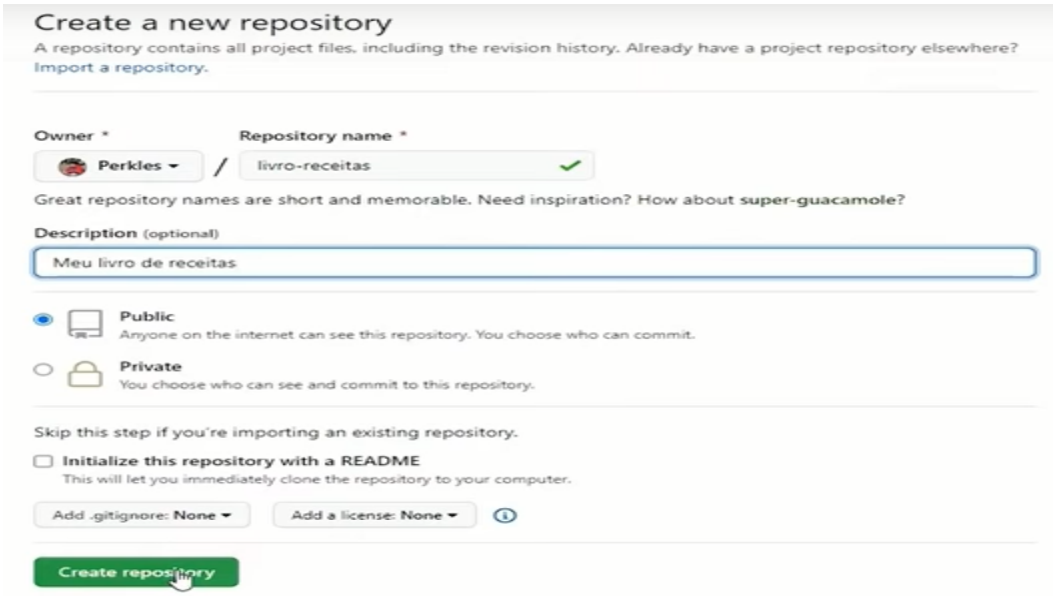
Unmodified é o estado onde o arquivo não sofreu alterações em relação a sua referência.
Modified é o estado onde o arquivo sofreu alterações em relação a sua referência.
Staged é o estado onde o arquivo já foi endereçado e aguarda ser transferido ao repositório.

Git status - O comando lhe mostra em qual branch você se encontra. Vamos dizer que você adicione um novo arquivo em seu projeto, um simples arquivo README. Caso o arquivo não exista e você execute git status, você verá o arquivo não monitorado "Untracked files" na saída do comando status.

INTRODUÇÃO AO GitHub

Trabalhando com o GitHub

No Git Bash verifique se as configurações de usuário estão alinhadas entre o Git e o GitHub, utilizando o comando: git config – list
Em seguida entre na sua conta do GitHub. Em repositórios, clicar em novo.



Ainda no GitHub copiar o caminho HTTPS para ser utilizado no Git Bash.

No Git Bash digitar o seguinte comando:
git remote add origin e colar o link https

Comando para levar (empurrar) nosso repositório local para nosso repositório remoto:
git push origin master

Ao dar enter o terminal vai solicitar as credenciais para empurrar o repositório local Git para o GitHub.

RESOLVENDO CONFLITOS

Como os conflitos acontecem no GitHub e como resolvê-los.

Conflitos são comuns em um sistema de versionamento de código**.**

Nas alterações que ocorrem na mesma linha que acontecem os conflitos, isso ocorre quando duas ou mais pessoas estão trabalhando no mesmo arquivo.

Quando você tentar empurrar o seu arquivo com o comando **git push origin master** o Git vai detectar o conflito.

É preciso puxar o arquivo para a sua máquina para que o Git identifique o conflito utilizando o comando:
git pull origin master
O Git vai tentar juntar esses arquivos e vai identificar o conflito de Merge. O GitHub não vai fazer nada, ele vai esperar que você faça as alterações manualmente, dessa forma que os conflitos são resolvidos.

Como baixar um repositório que está no GitHub:

No GitHub é necessário copiar a URL do repositório.

No Git Bash é necessário digitar o comando **git clone** e colar a URL que foi copiada do GitHub.
É possível verificar para onde seu repositório está apontado utilizando o comando:
git remote -v