Git é um sistema distribuído e seguro.

**Beneficios do git e github**

* Controle de Versão.
* Armazenamento em nuvem.
* Trabalho em equipe.
* Melhorar seu código.
* Reconhecimento.

**Comandos Básicos**

* **cd** – Windows e Linux – Navega entre as pastas.
* **dir** – Windows - Lista as pastas do diretório que estiver

**ls** – Linux – Lista as pastas do diretório que estiver. Usando a opção “-a” mostra os arquivos ocultos.

* **mkdir** – Windows e Linux - criar uma pasta nova
* **del** – Windows – Deleta apenas os arquivos de uma pasta, não afetando os subdiretórios.

**rm –rf** – Linux - Deleta as pastas, subpastas e arquivos, ou seja, deleta tudo.

* **rmdir** – Windows - Remove todos os diretórios e seus arquivos, usando as opções /S /Q.
* **cls** – Windows – Limpa a tela.

**Clear** – Linux – Limpa a tela.

* **tab** – Windows e Linux - Completa a palavra a partir de um caractere digitado.
* **echo** – Windows e Linux - Ex: echo hello > hello.txt. Criou um arquivo hello.txt.
* **cat** – Linux - Exibe o conteúdo de um arquivo de texto. Exemplo cat arquivo “texto.txt”.
* **pwd** – Windows e Linux - Lista o caminho completo de onde estou naquele momento.
* **mov** – Windows e Linux – Move um arquivo de um local para outro. Exemplo mv texto.txt caminho.

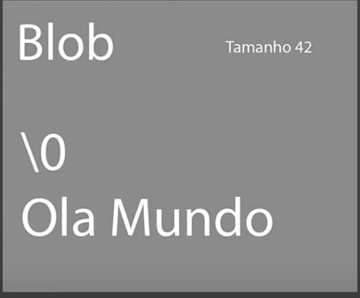
**Entendo o git**

* SHA1 A sigla significa secure hash algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA).

A encriptação gera conjunto de caracteres identificador de 40 dígito.

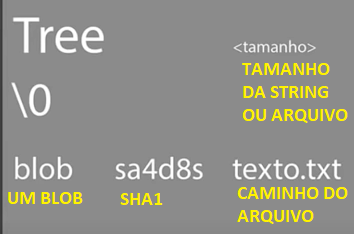
O comando é openssl sha1.

* Objetos fundamentais
  + Blobs ou bolhas: Esse objeto contém metadados, que são o tipo do objeto que é blob, o tamanho do arquivo ou da string, o contéudo que for digitado seja texto ou binário.



* + Trees: São as arvores e elas armazenam os blobs, é um crescente onde o blob o bloco básico de composição e a trees armazenando e apontando para tipos de blobs diferentes.

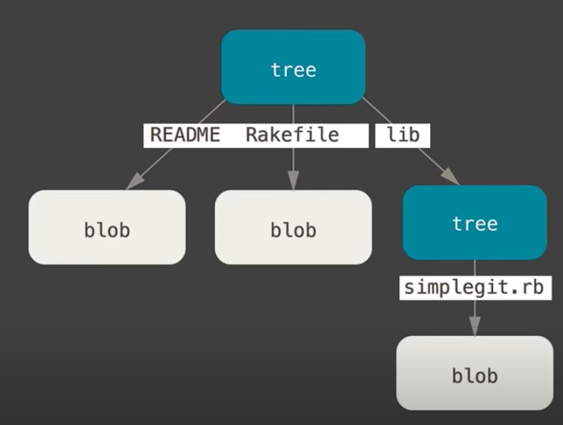
Ela também contém metadados ela tem essa barra e o zero e ela aponta para um blob que por sua vez tem um sha1 e a arvore também guarda o nome do arquivo.



A árvore é responsável por montar toda a estrutura de onde estão os arquivos.

As árvores podem apontar tanto para arquivos blobs ou para outras árvores. Isso porque um diretório pode conter subdiretórios.

As árvores possuem o seu sha1 ou seja se um blob for alterado ou uma árvore que ela aponte for alterada o seu sha1 será alterado também.



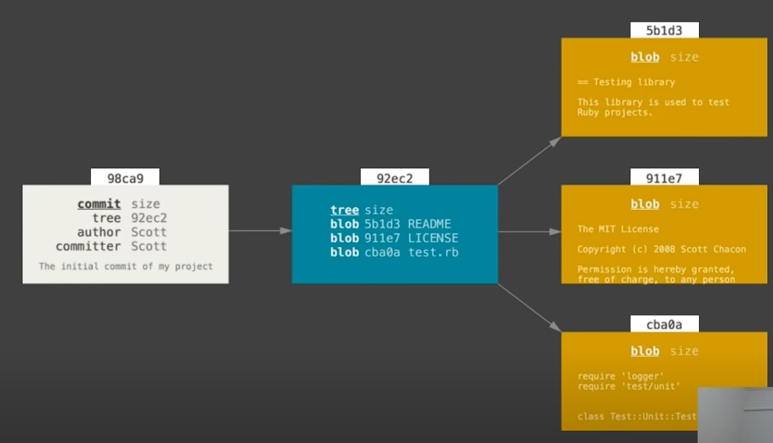
* + Commits: É o objeto que junta tudo, o objeto que dará sentido as alterações que serão feitas.

O commit ele aponta para uma árvore, para um parente que é o último commit realizado antes dele, aponta para o autor que é quem está fazendo aquele commit, a mensagem é uma explicação que justifica os arquivos dentro das pastas essa msg é escrita pelo usuário quando estiver montando o commit. O timestamp é um carimbo de hora e data de criação do commit.

Os commits também possui uma sha1, ou seja se uma árvore for alterada ou um blob for alterado irá alterar o sha1 do commit.

O commits utilizando o parente é possível montar uma linha do tempo, pois sempre terá a rastreabilidade de todas as alterações feitas anterior.





* Sistema Distribuído: É um sistema distribuído, pois seu repositório pode ser compartilhado com diversas pessoas, que podem alterar e modificar os arquivos e com isso cada pessoa que possuir aquele repositório, possuirá versões disponíveis totalmente confiáveis.

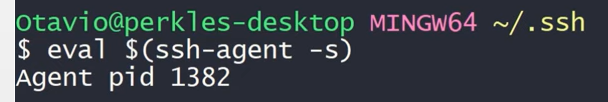
**GIT CHAVES SSH e TOKENS**

* Chaves SSH: É uma forma de estabelecer uma conexão segura e incriptografada entre duas máquinas.

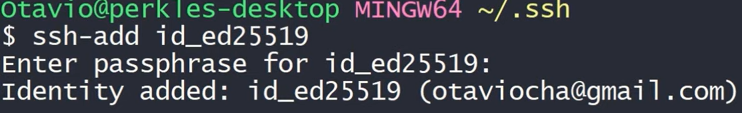
Para estabelecermos uma conexão com o servidor GIT HUB na nuvem iremos ter sempre duas chaves, sendo uma privada e outra pública. I

**Gerando a chave**

1. Gerar a chave, navegar até a pasta e pegar o conteúdo da chave publica
2. Colar no Git Hub
3. Voltar e passar a chave para o agent.
   1. **eval $(ssh-agent –s)** para instalar o serviço de ssh na máquina.



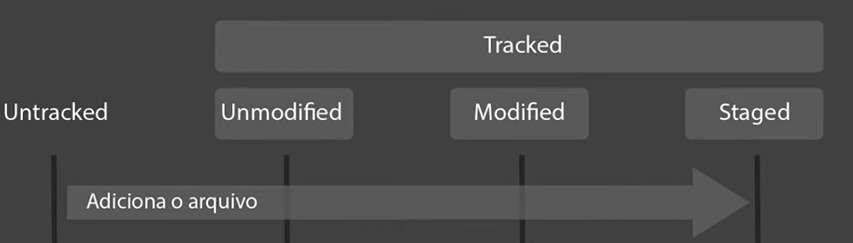
* 1. Navegue até a pasta que está a chave XX.pub
  2. **ssh-add “nome da chave”**, aqui é a chave privada. Será solicitado a senha que você usou no momento que gerou e a confirmação é se exibiu o e-mail cadastrado.



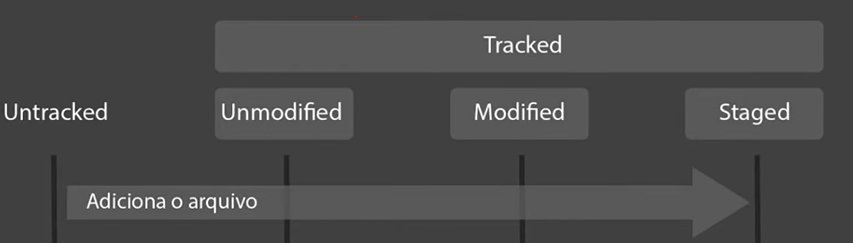
* Tokens: o token é um arquivo que será utilizando quando você acessar o github, passará o login a senha e vai pedir o seu token, para validar o aceso.

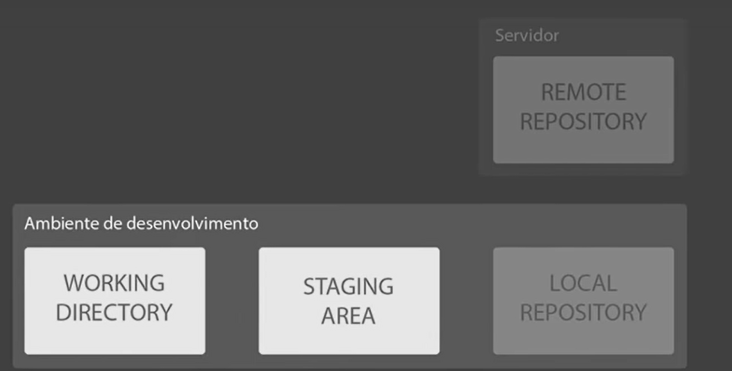
**INICIAR O GIT - TEORIA**

* **Conceito do git init** Quando executamos esse comando é criado a pastinha .git dentro da sua pasta. O que inicia o conceito de repositório.
  + Untracked: Arquivo criado em seu repositório, mas que não passou pelo processo de versionamento, ou seja, não foi comitado. Por isso o git nem sabe ainda que ele existe.



* O comando **git add \*** ao executar esse comando o git irá mover o arquivo do repositório direto para staged. Área em que o arquivo fica esperando para ser realizado alguma ação, mas agora ele é conhecido do git.



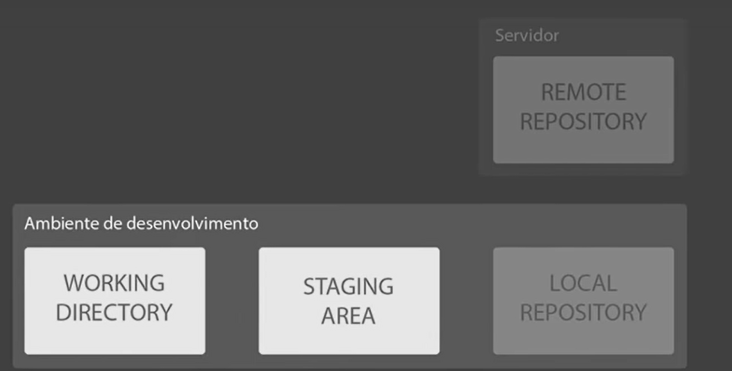


* + Unmodified: O arquivo que não sofreu nenhuma alteração ainda, foi criado, mas nunca modificado.
  + Modfied: Arquivo que foi criado e sofreu a sua primeira modificação. O git compara o sha1 dos arquivos para saber quando ele sofre modificação.



* + Caso seja executado novamente o comando **git add** nesse arquivo modificado, o mesmo fará novamente o processo de mover o arquivo para staged onde ficará esperando uma ação.



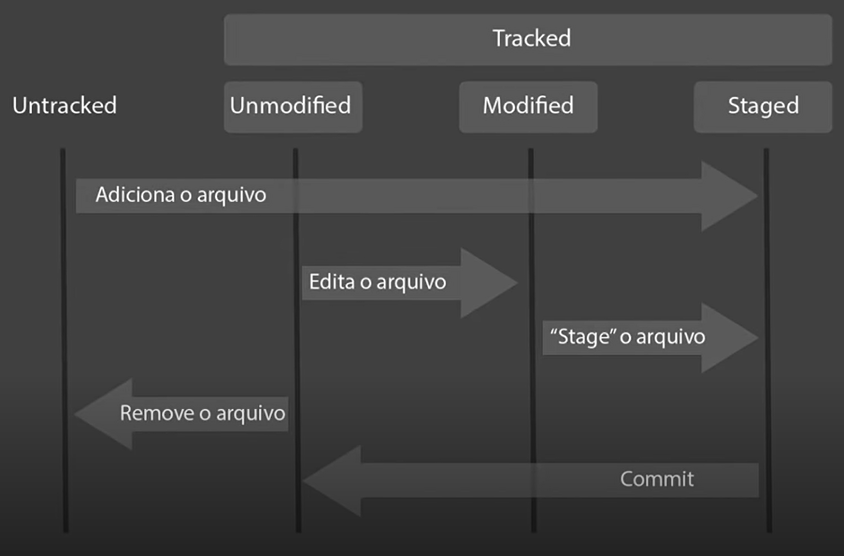


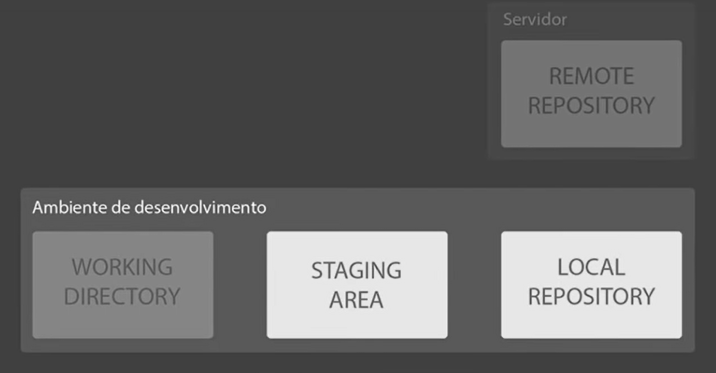
* + Caso o arquivo criado e ainda não modificado seja removido, ou apagado da pasta o arquivo volta para o a situação untracked, ou seja, não conhecido pelo git.



* Uma vez executado o comando **git commit** o arquivo que estava em **staged** volta para o estágio **unmodified**. Esperando que futuramente sejam alterados dando início a um novo ciclo.

Lembrando que o LOCAL REPOSITORY somente mantém o que já foi comitado e pode ser movido para o servidor remoto que fica na nuvem.





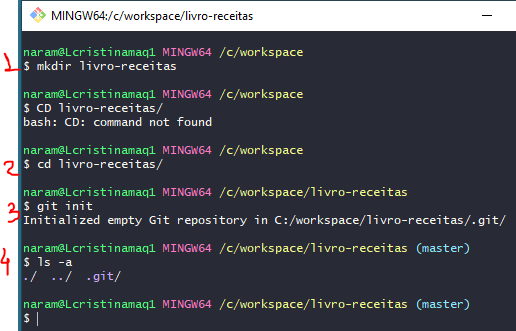
**INICIAR O GIT - PRÁTICA**

Primeiro passo é criar uma pasta para o projeto, no nosso caso criamos a c:\workspace\livro-receitas. Usando o comando **“mkdir livros-receitas”.**

Segundo vamos até a pasta criada para iniciar o git **cd livro-receitas**.

Terceiro **git init.**

Para confirmar a inicialização do serviço o comando **ls –a** irá mostrar a pasta “.git” que fica oculta.

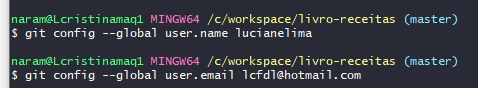


Se for a primeira vez que está usando o git, precisará configurar um nick name e um e-mail. Para isso os comandos:

* **git config –global user.email** [**lcfdl@hotmail.com**](mailto:lcfdl@hotmail.com)

Esse comando “-global” servirá para qualquer projeto, mas é possível criar apenas para um determinado repositório.

* **git config –global user.name lucianelima**



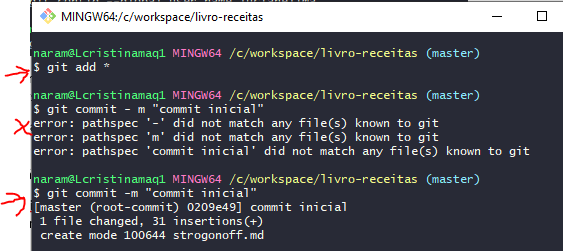
Crie dentro da pasta principal um arquivo, no caso criamos um .md e escrevemos uma receita de strognoff, pode ser criado usando o Windows, não foi necessário comandos.

Depois que o arquivo está finalizado iniciamos o processo de comitar.

**CRIAR UM COMMIT**

Dentro da pasta que está o arquivo de o comando **git add \***

Em seguida o comando **git commit –m “commit inicial”**, onde –m “” é a mensagem que será exibida aos demais usuários que consultarem o seu projeto. Pode ser qualquer mensagem.



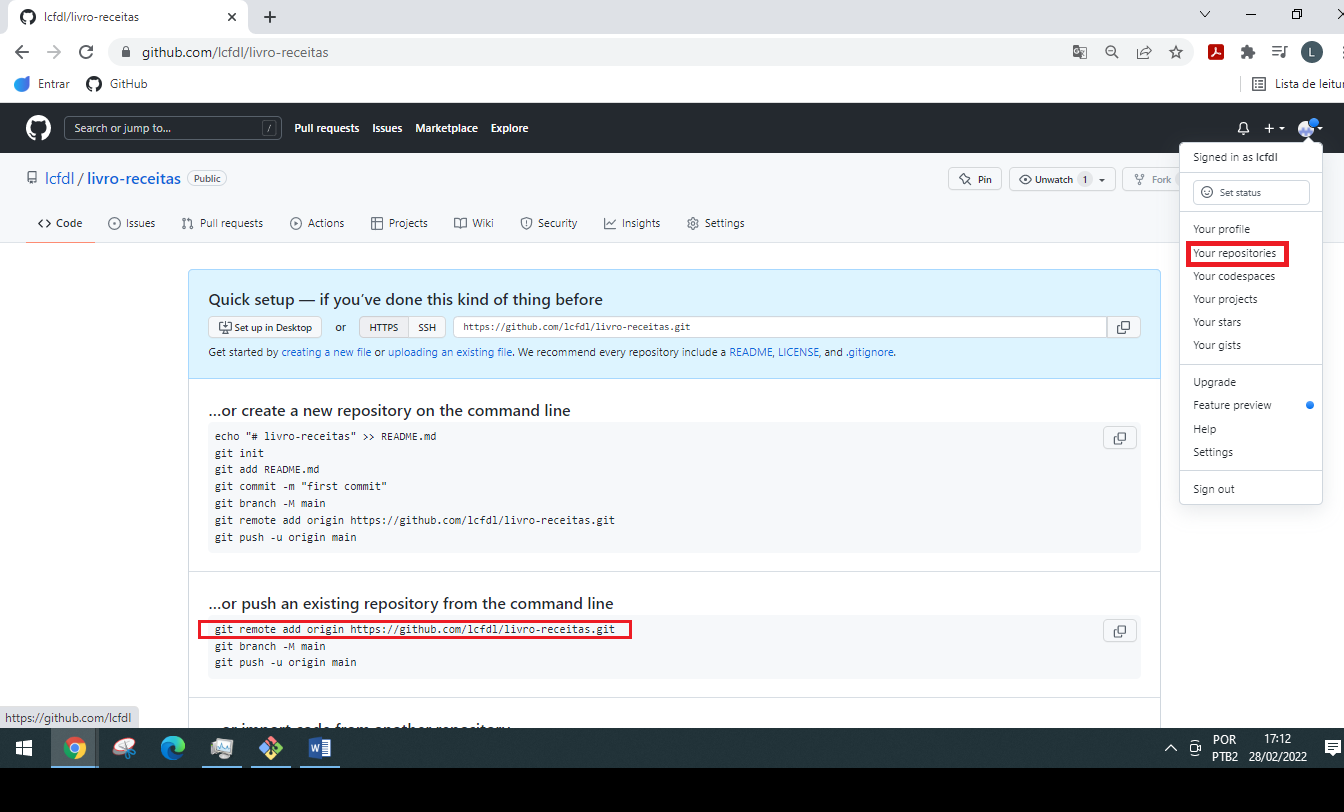
**COMANDOS DO GIT**

git config --list (lista todas as propriedades de configuração do git).

git config --global --unset propriedade (exemplo user.name ou user.mail) – Remove as configurações.

git config --global user.mail (propriedade) “[lcfdl@hotmail.com](mailto:lcfdl@hotmail.com)” – Para inserir as configurações novamente.

git remote add origin <https://github.com/lcfdl/livro-receitas.git> - Esse comando cadastra a url do seu repositório, após ser criado um novo repositório a partir do site.



git remote –v – Mostra todos os repositórios remotos cadastrados.

git push origin máster – Esse comando empurra os seus commit’s para o repositório da nuvem.

git pull origin master – Esse comando puxa os arquivos da nuvem e tenta mesclar os arquivos, porém se ambas as alterações forem na mesma linha o git não fará nada ele manda vc baixar e resolver o conflito. Esses conflitos são chamados de merge conflit.

git clone url – A url vc deverá pegar pelo site da git hub na aba code. Esse comando criará uma cópia do repositório escolhido na sua estação.

