Git: É um sistema de versionamento de código distribuído.

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

Navegação via command line interface e instalação

Interface gráfica x Linha de comandos

* Mudar de pastas.
* Listar as pastas.
* Criar pastas/Arquivos.
* Deletar pastas/Arquivos.

Comandos base de navegação de terminal.

## Windows

* **cd**: Mudar de pastas.
  + cd .. : Retroceder 1 nível na navegação.
* **dir**: Listar as pastas
* **mkdir** : Criar pastas.
* **del/rmdir**: Del - deletar arquivos./ rmdir: deletar todo o repositório.
* **cls** : Limpar tela.
* **TAB**: Auto completar os nomes das pastas.
* **echo**: Print no terminal.
* **>** : O símbolo de “maior que” serve para redirecionar o fluxo.
  + Ex.: Ele pega a saída de uma função e joga em um arquivo.

## Unix

* **cd**: Mudar de pastas.
  + **cd ..** : Retroceder 1 nível na navegação.
* **ls**: Listar as pastas.
* **mkdir**: Criar pastas.
* **rm -rf**: Deletar pastas.
* **clear** Ou **Ctrl + L** : Limpar tela.
* **TAB**: Auto completar os nomes das pastas.
* **echo**: Print no terminal.
* **>** : O símbolo de “maior que” serve para redirecionar o fluxo.
  + Ex.: Ele pega a saída de uma função e joga em um arquivo.
* Obs.1: O terminal do windows é derivado do Shell e o terminal do linux é derivado do Bash.
* Obs.2: Se der tudo certo, normalmente o terminal não vai exibir nada (nenhum erro).
* Obs.3: No windows tem diferença entre deletar arquivos e deletar repositórios.

C:\> : Base do diretório Windows.

/ : Base do diretório Linux.

Realizando a Instalação do Git.

Instalação do Linux:

* sudo add-apt-repository ppa:git-core/ppa -y
* sudo apt update;
* sudo apt install git;
* git --version
  + git version 2.33.0.

Tópicos fundamentais para entender o funcionamento do Git

# Entendendo como o Git funciona por baixo dos panos

* SHA1.
* Objetos fundamentais.
* Sistemas distribuídos.
* Segurança.

## SHA1

A sigla SHA significa Secure Hash Algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA).

SHA1 é um algoritmo de encriptação. O algoritmo funciona embaralhando os arquivos de uma forma específica e gera um conjunto de caracteres encriptados de 40 dígitos (único).Este código gerado serve como identificador.

Ex.:

Se executar esse algoritmo em um arquivo de texto, ele vai gerar um conjunto de caracteres de 40 dígitos. Se alguma caractere for alterar e esse arquivo for executado novamente, ele vai gerar outro código, mas se aquele arquivo que foi alterado, voltar o que era antes, o código que é gerado, volta a ser o mesmo que era antes.

o SHA1 nada mais é do que uma forma curta de representar um estado de um arquivo.

## Representação do Algoritmo.

Abra o Git Bash no local onde o arquivo que você quer usar

* Coloque o código “oppenssl sha1 nome\_do\_arquivo.txt”.
* Depois de executar o comando, ele vai gerar um código.
* Depois altere qualquer caracter e execute novamente.
* Vai gerar outro código.
* Depois desfaça a alteração que foi feita e execute o comando novamente.
* Ele vai gerar o mesmo código que foi gerado na primeira execução.

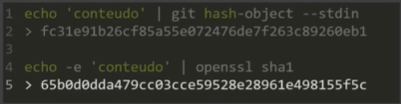
Obs.1: Esse algoritmo não se restringe somente ao arquivo, mas também aos objetos internos no Git.

Objetos internos do Git

Objetos internos do Git:

* BLOBS.
* TREES.
* COMMITS.

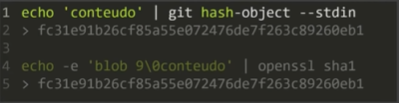
# Blobs



Os arquivos ficam dentro do objetivo chamado Blob e esse objeto contém metadados dentro dele.

O objeto Blob tem: um tipo do objeto, tamanho, \0 e Conteúdo de dentro do arquivo.

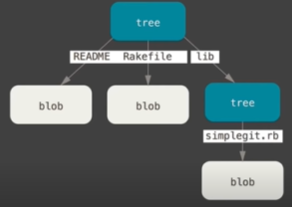
Obs.: o tamanho sempre é o tamanho do nome do arquivo ou string mais o 0.



# Tree

As Tree armazenam os Blobs.

* As árvores também contêm metadados.
  + Os blobs, o código sha1 e o nome do arquivo.
* E aponta para outros Blobs ou outras árvores.
  + Isso acontece porque os diretórios dentro dos sistemas operacionais podem conter outros diretórios. Então no git as árvores podem apontar para outras árvores.
* A árvores vai ser responsável por montar todas as estruturas e localizar os arquivos.
* O Blobs contém os SHA1 e metadados do arquivo, por sua vez a Tree contém os SHA1 dos Blobs e seus metadados. Com isso, se um arquivo for alterado, vão refletir nos outros SHA1.



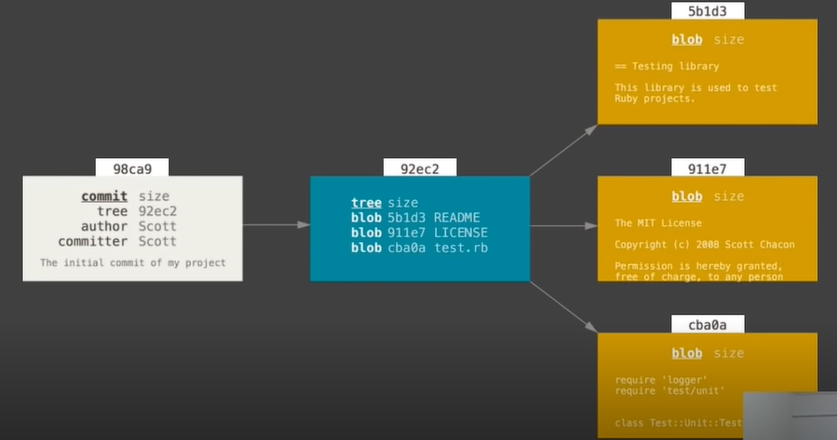
# Commit

O commit é o objeto que vai juntar tudo, é ele que vai dar sentido.

* Commit aponta para uma árvore, aponta para um último commit feito antes dele.
* Commit aponta para um autor e mensagem.
* Os Commit também possuem um SHA1.
* Se uma blobs for alterada, essa alteração vai ser refletida nas tree e no commit por causa do SHA1.

O commit vai dar sentido aos arquivos que estão sendo armazenados (tree e blobs).

O commit também contém um objeto chamado “timestamp” que trás as horas e a data de quando o commit foi criado.



Chave SSH e Tokens

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

## **Path**

Perfil > Settings > SSH and GPG keys > New SSH keys.

## **Configuração Windows**

## Comando para gerar a chave SSH:

* ssh-keygen -t ed25519 -C <email>
* Path de onde a chave vai está armazenada (/c/Users/Sidnei/.ssh/id\_ed25519).
  + O ponto antes do “.ssh” quer dizer que a pasta está oculta.
  + Senha: 19971600
* /c/Users/Sidnei/.ssh/id\_ed25519.pub
* A chave pública está armazenada no mesmo lugar, porém com “.pub”.

Agora é só navegar até a pasta.

Comando:

* cat : Exibe o conteúdo da pasta.
* pwd: Exibe o path completo da pasta atual.
* cat <nome da pasta>

## Ativar o SSH:

Comando:

* eval $(ssh-agent -s).
* ssh-add e o path de onde a chave ssh está.
  + ssh-add id\_ed25519.

# 

# Token de acesso pessoal

Perfil > Settings > Developer settings > Personal access tokens.

* Nomear um nome para a chave.
* Marca a opção “repo”.

Copiar e aguardar o Token.

Quando for usar o github, ele vai pedir para inserir token.

Primeiros comandos com Git.

## Comandos - Git:

* git init: Iniciar um repositório no git.
* git add: Mover arquivo.
* git commit: Cria um commit.

Obs.: Todos os comandos do Git começam com “git”. Significa que está sendo executado comandos específicos do git.

# Criando um repositório.

Configuração do git comandos:

* git config --global user.email “<email>”
* git config --global user.nickname “<nickname>”

## Comandos:

* ls -a : Exibir arquivos ocultos.

# Adicionando um arquivo.

Arquivo Markdown: É uma forma mais “humana” de escrever um arquivo html, sem entender necessariamente o que os elementos html significam.

Comandos:

* git add \*.
* git commit -m ”<mensagem>”.

[master (root-commit) **6628b3e (caracteres do SHA1)**] **commit inicial** (mensagem adicionada)

1 file changed, 15 insertions(+)

create mode 100644 receita.md

Ciclo de vida dos arquivos no Git.

# Tracked ou Untracked.

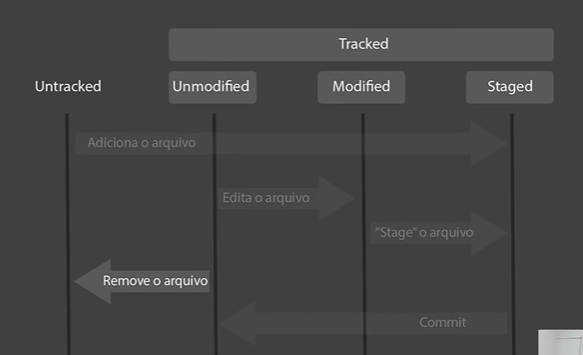
Tracked: São arquivos que o git tem ciência deles.

Dentro dos arquivos que são rastreados pelo git. Podemos dividi-los em três estágios diferentes:

* Unmodified: São os arquivos que ainda não foram modificados.
* Modified:São os arquivos que sofreram algum tipo de modificação.
* Staged: É onde ficam os arquivos que estão se preparando para pode fazer parte de uma outro tipo de agrupamento (Commit).

Untracked: São arquivos que o git ainda não tem ciência deles.

# Ciclo de vida do Git



# Significado dos repositórios

Separação de 2 ambientes:

* Servidor: github.
* Ambiente de desenvolvimento: Representa tudo que está na máquina.
  + Repositório de trabalho.
  + Repositório Local.
  + Staging area.
* Obs.1: Os arquivos sempre vão transitar pelo repositório de trabalho e o Staging Area a medida que os arquivos são alterados ou adicionados.

Pelo fato que o git é um sistema distribuído, ele tem uma versão dos arquivos no servidor e um outra versão dos arquivos na sua máquina.

* Quando é usado um commit, os arquivos que estão na staging area são enviados para o repositório local.

As alterações que são feitas no arquivo na sua máquina, não são alteradas automaticamente no servidor. A não ser, que os envie utilizando os comandos.

Os arquivos do repositório local são enviados para o repositório do git.

## Comandos:

* **mv <nome do arquivo> ./<nome da pasta que você quer mover o arquivos>/ :** Mover o arquivo para alguma pasta.

## Comandos Git:

* **git status:** Mostrar os status dos arquivos.
* **git add <nome do arquivo>:** Adicionar um arquivo.
* **git add \*:** Adicionar todos os arquivos para o modo staging.

Introdução ao Github

## Comandos Git:

* **git config --list:** Vai exibir todas as configurações do seu git.
* **git config --global --unset user.<propriedade que deseja alterar>:** Remover a configuração que estava.
* **git config --global user.<propriedade que deseja>:** Configurar usuário ou email(Ou outras coisas relacionado ao usuário).
* Obs.1: Até agora tudo o que foi feito foi no repositório local. Agora vamos enviar o repositório local para o repositório remoto.

## Comandos para enviar os arquivos para o repositório remoto:

* + **git remote add origin <link do repositório>:** Adicionado a origem, o local onde vão ficar os arquivos.
    - Obs: O “origin” é apenas um apelido, poderia ser qualquer nome.

* + **git remote -v:** Exibir a lista de repositório que estão cadastrados.
  + **git push origin master:** Enviar os arquivos do repositório local para o remoto (Empurrar os arquivos para o <link do repositório> na branch master).

Resolver conflitos

Os conflitos de Merge: Ocorrem quando alterações concorrentes são feitas na mesma linha de um arquivo ou quando uma pessoa edita um arquivo e outra pessoa exclui o mesmo arquivo.

Comando Git:

* git pull: Vai puxar os arquivos do repositório remoto para a máquina.
* git clone <link do repositório que deseja clonar>: Clonar.

## **Excluir um branch REMOTO**

Este é o comando para excluir um branch remoto: git push <remote> --delete <branch>.

* Exemplo: git push origin --delete fix/authentication.

O branch, agora, foi excluído remotamente.

Também é possível usar um comando menor para excluir um branch remotamente: git push <remote> :<branch>

* Exemplo: git push origin :fix/authentication.

Se aparecer para você o erro abaixo, isso quer dizer que outra pessoa já excluiu o branch.

## **Excluir um branch LOCAL**

O Git não deixará que você exclua o branch no qual você está trabalhando no momento. Por isso, certifique-se de fazer o checkout para um branch que você NÃO quer excluir. Por exemplo: git checkout master ou git checkout main.

Exclua um branch com o comando git branch -d <branch>.

* Exemplo: git branch -d fix/authentication.

A opção -d excluirá o branch somente se você já fez o push e o merge com o branch remoto. Use a opção -D em vez disso se quiser forçar a exclusão do branch, mesmo que você ainda não tenha feito o push e o merge com ele.

O branch, agora, foi excluído do local.