GIT E GITHUB

* O git foi criado em 2005 por Linus Torvalds - criador do linux;
* surge como um versionador de código distribuído; (monitorar e criar diferente do

nossos códigos dentro da nossa máquina.)

* o git não é uma invenção (já existia versionamento, mas se tornou o melhor e mais utilizado.)
* git e github são complementares, mas diferentes.
* o git possui uma estrutura de compartilhamento, fazendo com que mais de uma pessoa possua a mesma versão do código em diferentes ambientes de trabalho, por isso o git é um sistema distribuído.

**Pontos desenvolvidos com GIT E GITHUB**

* Controle de versão
* Armazenamento em nuvem
* Trabalho em equipe
* Aprimoramento de código
* Reconhecimento

**GUI E CLI**

Gui e Cli são dois tipos de interfaces gráficas utilizadas em sistemas de softwares, os sistemas **CLI** trabalham através de linhas de códigos que são inseridos pelo o usuário para execução de uma tarefa enquanto o **GUI** não requer experiência do usuário em códigos, tornando o uso mais simplificado por meio do visual e a interatividade do mesmo.

**Comandos básicos no terminal**

A maioria dos sistemas operacionais possuem uma interface gráfica, tudo é responsivo, entretanto existem ainda uns programas que não, a exemplo do git e do terminal de comando dos SO.

**commands**

WINDOWS

**(Dir)** lista todos os diretórios (pastas) dentro do local em que o usuário está situado.

**(CD)** Change direction - muda o diretório local

**(CD ..)** Change direction - retrocede ao diretório anterior

**(CLS)** Clear screen - limpa códigos anteriores

**(TAB)** Atalho - Auto completa com o resultado mais próximo

**(MKDIR)** make directory - cria um diretório

**(DEL)** deleta os arquivos de um diretório

**(RMDIR /s /q)** deletar um diretório

git update-git-for-windows - atualiza a versão do google

LINUX/MAC

**(LS)** lista todos os diretórios (pastas) dentro do local em que o usuário está situado.

**(CD)** Change direction - muda o diretório local

**(CD ..)** Change direction - retrocede ao diretório anterior

**(CLEAR)** Clear screen - limpa códigos anteriores

**(CTRL+L)** Atalho - limpa automaticamente

**(TAB)** Atalho - Auto completa com o resultado mais próximo

**(MKDIR)** make directory - cria um diretório

**(DEL)** deleta os arquivos de um diretório

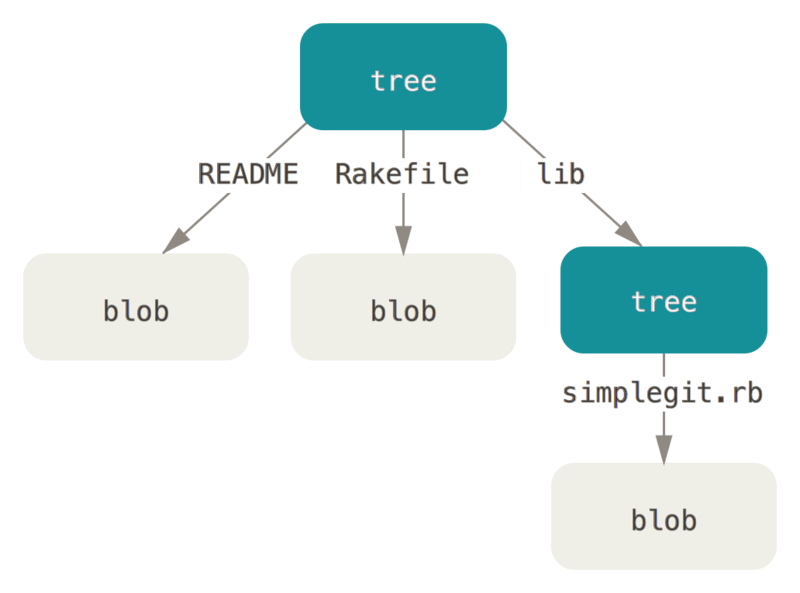
**(RM -rf)** deletar um diretório

**Tópicos fundamentais**

**(SHA1)** é um algoritmo de encriptação (secure hash algorithm). sua encriptação gera um conjunto de caracteres de 40 dígitos único. Ela nada mais é que uma forma curta de representar um arquivo. E o git utiliza este método para gerenciar as alterações de arquivos e seus objetos internos.

**(Objetos internos do git)** São divididos em 3 tópicos: Blobs (arquivos), Trees (pastas) e Commits.

* Blobs: É um tipo de objeto utilizado para armazenar o conteúdo de cada arquivo em um repositório, a exemplo do sha-1 do arquivo que é calculado e armazenado em um blob. A alteração de uma blob (arquivo) reflete na estrutura de tudo, pois o sha1 da tree também é alterada.
* Tree: Nada mais é um armazenamento em grupo de arquivos contendo informações sobre o mesmo, meio que como as entradas do diretório, ela pode apontar para um blob ou outra tree dentro de si mesma.
* Commit é o que junta tudo, aponta para tree, blob, autor, parente (outro commit) mensagem

****

**Chaves SSH e Token**

Chave ssh é uma forma de estabelecer ligação entre duas máquinas de maneira criptografada.

**Comandos Git**

**(Git init)** Inicia o repositório

**(Git add)** Adiciona arquivos à área de stage ( **. = todos modificados)**

(**Git commit) conjunto de objetos do git que carrega os metadados como a mensagem sobre a alteração**

**(Git config)** Configura informações sobre o usuário

**(Git status)** Verifica o estado dos arquivos do repositório

**(Git pull)** Puxa as alterações feitas no repositório remoto (servidor).

**Ciclo de vida dos arquivos**

**Tracked** são arquivos que o git possui ciência/gerência deles, são divididos em modified (modificado), unmodified (não modificado) e staged (onde fica os arquivos prontos para serem commitados.)

**Untracked** são os arquivos que ainda não foram registrados no git, ou seja, o git ainda não possui ciência de sua existência e não o gerencia.