GIT E GITHUB

GUI = grafical user interface

CLI= comand liner interface (tipo de sistema do git)

Comandos básicos de navegação no terminal: (Windows, Linux)

* Dir ou LS = abre todos os diretórios que constam na pasta selecionada;
* Cls ou clear = limpa terminal; (atalho Linux ctrl + l)
* Tab = auto completa caminhos;
* CD = serve de parâmetro para navegação pelas pastas;

Cd .. = volta um nível da pasta selecionada;

Cd + nome da pasta = acessa pasta selecionada;

* Mkdir = cria pastas;
* Echo = printa na tela um texto passado como parâmetro;

Echo + texto + > nome arquivo = (sinal da maior vai servir como um redirecionador de fluxo, tipo um comparador) esse comando serve para criar arquivo dentro da passa, ele vai comparar se existe arquivo com nome passado dentro da pasta/diretório, caso não exista ele criar arquivo.

* Del + nome da pasta = deleta arquivos dentro da pasta/diretório e não a pasta completa;
* Seta pra cima = mostra todos os comandos executados anteriormente na pasta/diretório;
* Rmdir + nome pasta + /S + /Q = exclui diretório e todos os arquivos que possui;

(Linux) rm + -rf + nome pasta = exclui diretório e todos os arquivos que possui;

Como o Git funciona

Tópicos de funcionamento do GIT:

* SHA1 = (SECURE HASH ALGORITHM) é um conjunto de funções hash criptográficas projetadas pela Agencia de Segurança Nacional dos EUA, esse processo de encriptação gera um conjunto de caracteres identificador de 40 dígitos , esse identificador é único. Todo processo de atualização gera uma chave diferente, salvo alteração de arquivo seja a mesma de uma atualização anterior, nesse caso a chave gerada será a mesma que corresponde ao mesmo conteúdo.
* Objetos internos do GIT (BLOBS, TREES, COMMIT)

- BLOB = possui meta dados do GIT e a estrutura básica de um objeto BLOB vai conter seu tamanho, uma \ com o numeral 0, e o conteúdo desse objeto que pode ser uma string ou etc.

- TREE = as TREES armazenam BLOBS, na sua estrutura básica então vai constar a \ e 0, seu tamanho, a blob com seu sha1 e o nome do arquivo que consta nesse objeto. Uma tree pode apontar para um blob ou para outra tree. Cada arvora possui seu próprio sha1.

- Commit = na estrutura de commit irá conter a tree com seu sha1, o parente com seu sha1 (seria o último commit realizado antes dele), autor (identifica o atual usuário da alteração) , uma mensagem (seria a mensagem identificando o tipo de alteração realizada) e um timestamp (data e horário de criação)... o commit tbm possui um sha1 especifico.

* Sistema de distribuição seguro = face todas essas opções de controle sobre os registros de alterações, se torna um sistema bem seguro de distribuição.

CHAVES DE SSH E TOKENS

- Chave SSH = é uma forma de estabelecer uma conexão segura e encriptada entre duas máquinas.

Processo para gerar uma chave SSH = todos os passos devem ser realizados no GIT BASH. Lembrando que após cada comando é colocado um espaço.

ssh-keygen + –t + ed25519 + –C + e-mail

Depois de dar enter o sistema vai apresentar em que local que arquivo gerado foi salvo e na sequencia vai pedir uma senha. Após enter ele mostra local salvo do arquivo criado e mais algumas informações referente ao processo.

Depois das confirmações, navegar até a pasta que consta o arquivo da chave digitar comando: cat e selecionar a chave pública, copiar e colar na área de ssh do github que é acessada através das configurações. Ai vai colocar um nome pra chave e colar texto copiado do git.

No bash dentro da pasta ssh digitar:

eval + $(ssh-agent –s)

bash vai mostrar um agente pid

após vamos digitar = ssh-add + chave particular (lembrando que no github colocamos a chave pública), após enter vai pedir a senha que foi cadastrada no início. Feito isso processo OK

-TOKENS = é gerado no GITHUB na parte de configurações, developer settings, personal access token, genarete new token. Esse token pode ser definido uma data de validade de uso. Selecionar opção repo.

Token gerado deve ser salvo num local seguro, caso perca esse arquivo é necessário realizar todo o processo para gerar novo token.

- INICIANDO O GIT E GERANDO PRIMEIRO COMMIT –

* Para iniciar um repositório GIT usamos, GIT INIT
* Para mover arquivos ou dar início ao versionamento usamos, GIT ADD
* Para criar um commit usamos, GIT COMMIT

Sempre no início de cada comando vem o nome do programa primeiro, pois é uma característica quando usamos o terminal.

* Flag LS –A = mostra arquivos ocultos dentro da pasta
* Flag git add \* = adicionar todos os arquivos alterados
* Flag git commit –m “” = para inserir um novo commit
* Flag git remote –v = mostra o endereço do repositorio git

Estados de arquivos Git

Untraked = seria pasta vazia pronto para inclusão do arquivo

Unmodified = é um arquivo que já está sendo monitorado mas não possui modificações

Modifed = arquivo existente que contem modificação

Staged = arquivo adicionado pronto para receber o commit

* GIT STATUS = serve para verificar os status dos arquivos na pasta monitorada