**Git e GitHub**

Por que aprender essa tecnologia?

A tecnologia Git vem para versionamento de códigos.

Um software não é escrito por uma pessoa só e o Git permite que várias pessoas versionem o projeto em conjunto.

Git e GitHub são tecnologias diferentes, mesmo que complementares. Ambos não são únicos no mercado, existem outras opções *(não tão boas).*

Quais os benefícios de utilizar essas tecnologias?

* Controle de Versão
* Armazenamento em Nuvem
* Trabalho em equipe
* Melhorar o código
* Reconhecimento

**Comandos básicos de navegação no terminal:**

O Git tem um design CLI, não possui uma interface gráfica. Interagimos com ele por linha de comando.

Para interagir com o Git é preciso aprender alguns comandos básicos, como mudar de pastas, listar as pastas, criar pastas/arquivos e deletar pastas/arquivos.

Usuários que possuem Windows vão interagir e usar comandos diferentes de usuários que possuem Unix (Linux, MacOS).

Primeiros comandos de navegação:

dir > vai trazer uma lista dos arquivos que estão dentro da pasta.

cd > possibilita caminhar entre os diretórios. O cd / vai caminhar para a pasta C. Como entrar em uma pasta Windows, por exemplo? cd windowns e dir traz todos os arquivos nessa pasta.

cd .. > vai retroceder o caminho já feito.

cls > vai limpar o terminal.

TAB – Tem a função de auto completar.

mkdir > cria uma pasta/diretório.

echo > printa de volta no terminal o que for passado pra ele. Usando esse mesmo comando com > na frente, ele joga a informação de texto em um arquivo. Por exemplo: echo hello > hello.txt

del > deleta o arquivo.

No Windows há uma diferença entre deletar arquivos e deletar repositórios. O comando “del” se restringe a deletar arquivos. Então ele deleta tudo o que há dentro do repositório. Como realmente remover tudo?

rmdir workspace /S /Q > remove o repositório de nome workspace.

Resumindo, os comandos básicos para navegar no terminal do Windows:

* cd
* dir
* mkdir
* del / rmdir

**Como o Git funciona por baixo dos panos?**

**SHA1** – É um algoritmo de encriptação. A sigla SHA significa Secure Hash Algorithm, é um conjunto de funções hash criptográficas projetadas pela NSA. A encriptação gera um conjunto de caracteres de 40 dígitos e esse conjunto é único para cada arquivo e cada versão desse arquivo. É uma forma curta de representar um arquivo. Usamos o “openssl sha1 nomearquivo.text” terminal do Git Bashpara gerar esses caracteres criptografados.

**Objetos internos fundamentais do Git:**

* Blobs – Os “arquivos” manipulados pelo Git ficam dentro de um objeto chamado Blob. Esse objeto tem um tipo, um tamanho e um conteúdo. O Git guarda os arquivos fazendo um sha deles (criptografando) etambém armazenando metadados no objeto.
* Trees – Elas armazenam Blobs. Também contém metadados; aponta para o blob e guarda o nome do arquivo. O blob não guarda o nome do arquivo. A tree é responsável por montar toda a estrutura de onde estão os arquivos. Podem apontar para blobs ou outras árvores. Se mudar qualquer coisa no arquivo blob, por consequência muda a estrutura da tree.
* Commits – O objeto mais importante. Ele junta tudo e da sentido ao que está sendo feito. Ele vai apontar para uma tree, para um parente, um autor e uma mensagem. O sha1 do commit é o hash de toda a informação. Tem também um timestamp, hora, dia de quando foi criado. Ele possui metadados e um hash próprio, se algo for alterado, como uma blob, todos os metadados são alterados.

Por que é um sistema distribuído e por que é tão seguro?

Quando você tem um commit, você garante que nada foi alterado, há uma linha do tempo de tudo que foi realizado. É uma forma segura de dizer que não houve alterações no commit, já que qualquer coisa que for alterada vai alterar o commit como um todo.

**Chaves SSH e Tokens:**

Antigamente, para enviar o projeto para o GitHub bastava informar o login e senha. Houve uma mudança na forma como é autenticada as informações para mais segurança.

A chave SSH é uma maneira de estabelecer uma conexão segura e encriptada entre duas máquinas. Após as configurações inicias, o GitHub já reconhece a assinatura da máquina que está sendo usada e autentica o envio do projeto sem precisar de senha. A máquina fica configurada previamente.

Após ter criado a chave e configurado a máquina SSH no GitHub, não é mais possível clonar um repositório pelo HTTPS. Deve ser usado o link SSH. Para fazer isso no terminal: git clone linkssh.io.git

<https://docs.github.com/pt> > tutoriais e documentação.

Os tokens de acesso pessoal são a segunda forma de autenticação oferecida pelo GitHub, mas ele perde a vantagem de não precisar digitar nada para fazer o commit. Na hora da senha, é utilizado o token. Pode ser configurado uma data de expiração para esse token. Ele precisa ser salvo em um local seguro pois o GitHub não vai mostra-lo novamente.

Para clonar usando o token, é usado o link HTTPS (mesmo processo usando git clone).

Em uma máquina de confiança, é sempre melhor usar a chave SSH.

**Primeiros Comandos com Git:**

* Git init : inicializa/cria o repositório.
* Git add: Mover arquivos e iniciar o versionamento
* Git commit

O que significa um repositório?

Temos um ambiente de desenvolvimento e o servidor. O Git tem as versões na máquina e as versões no servidor. Ao alterar alguma coisa no ambiente de desenvolvimento, ele não altera automaticamente no servidor. Ao adicionar um arquivo Untracked (git add) ele é movido para a área de stage. Ele ficam transitando entre Unmodified, Modified e Staged. Para que eles realmente entrem de fato é necessário o commit. Após o commit, ele volta para o Unmodified ate que haja nova alteração. O repositório local é todo composto por commits.