**Git e Github.**

Git é uma tecnologia de versionamento de código, que em conjunto com o github nos permite coisas como

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O github, diferente de outras plataformas e programas mais comuns, é um CLI (comand line interface) ou seja nós operamos ele através de comandos e não por uma interface gráfica GUI( graphic use interface)

Desenho com traços pretos em fundo branco

Descrição gerada automaticamente com confiança média

Nesse curso vou aprender a

Texto

Descrição gerada automaticamente

Tudo por linha de comando. Adendo, sistemas operacionais diferentes utilizam comandos diferentes.

Texto

Descrição gerada automaticamente

Cd = é o comando que nos permite entrar e sair das pastas, quando utilizamos cd /, ele vai para o c:.

cd + nome da pasta = Para entrar dentro de uma pasta especifica utilizamos o cd + nome da pasta. (o TAB autocompleta o nome das pastas.)

Cd .. = Para retroceter uma vez a navegação (voltar para a pasta anterior

Dir = lista tudo que tem dentro da pasta

Cls = para limpar o terminal.

Mkdir = cria uma pasta (ex: mkdir workspace) (silence on sucess = se n aconteceu nada é pq deu certo)

Echo > = echo devolve o que vc escrever para ele, o > redireciona essa saída do echo e joga, por exemplo em um arquivo que estamos criando, dessa forma para criar um arquivo de texto o comando fica assim: echo nomeArqueivo > nomeArquivo.txt

Del = deleta arquivos, se tentarmos deletar uma pasta com esse comando, tudo dentro da pasta será deletado menos a pasta.

Rmdir = deleta o repositório e tudo que tem dentro (pasta = repositório).

**Entendendo como o git funciona por baixo dos panos.**

tópicos

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média

**SHA1** = (secure hash algorithm), é um conjunto de funções hash criptográficas projetadas pela agência de segurança nacional dos EUA que encriptografa os arquivos gerando um hash (uma representação do seu arquivo), o git faz uso dessa tecnologia para encriptografar os arquivos que colocamos nele, dessa forma os arquivos ficam seguros e mais facilmente localizáveis pelo próprio git.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Para testar essa função de encriptação nos fomos no nosso git hash, abrindo ele clicando com o direito do mouse na área de trabalho para começar a navegação a partir de la, então utilizamos o comando openssl sha1 nomeDoArquivo.FormatoDoArquivo e obtivemos o hash referente ao arquivo

Texto

Descrição gerada automaticamente

**Objetos Blobs, trees e commits.**

Blogs: O git lida com os conteúdos em forma de objeto, esse objeto recebe o nome de blob (bolha), ou seja, para cada arquivo que é colocado no git, é gerado um blob, com uma numeração, o seu conteúdo (metadados) e o seu hash.



**Tree:** na tradução arvores, é uma estrutura que contêm os blobs, e ela é q leva o nome dos arquivos porque ela cria toda a estrutura de onde vão estar os arquivos, e é através destas arvores que navegamos pois uma arvore pode apontar para outra arvore e assim por diante.

É importante lembrar que as arvores carregam seus metadados, e quando mudamos algo nas bolhas q elas carregam, nos mudamos também os metadados das arvores, isso muda a estrutura toda.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Commit:**

é o objeto que junta tudo, ele aponta para uma arvore, autor, mensagem, e para o ultimo commit, ele da sentido as blobs e as trees pq permite colocar os comentários, e as explicação das ações de alteração dos arquivos. Os commits tbm possuem hashs, e a coisa funciona em cascata, pois qndo alteramos uma blob, alteramos o seu hash, que altera o hash da arvore, que altera o hash do commit.

Diagrama

Descrição gerada automaticamente

Tudo isso para dizer que o git é um sistema distribuído seguro.

**Chaves SSH e TOKENS**

Para subir meus códigos para o github hoje não basta apenas ter uma conta com a senha e o email, precisamos de autenticações mais seguras.

Uma delas é a chave SSH, que é uma forma segura de se conectar com outra máquina, no caso a do github, utilizando esta chave pública, que uma vez configurada faz com que o github nos reconheça como uma conexão segura.

Encontramos essa parte de configuração da chave em settings.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Podemos fazer a configuração da nossa chave no nosso github bash através do seguinte comando



Depois de usar o comando, a chave é gerada, dai precisamos colocar uma senha para ter acesso a chave.

Texto

Descrição gerada automaticamente

Depois fomos até a pasta indicada pelo caminho c/users/pedro/.ssh e usamos o comando cat id\_ed25519.pub (esse .pub indica que e a chave publica.)

E assim obtemos a chave publica que colocamos no git para que ele reconheça nossa máquina.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

Ainda precisamos configurar uma outra entidade que gerencia essas chaves (ssh agente) fazemos isso através do comando eval $(ssh-agent -s) depois nós precisamos passar nossa chave privada para o agente da seguinte forma, ssh-add id\_25519.

Agora já podemos fazer coisas como clonar repositórios abertos de outras pessoas, para usso usamos o comando git clone linkDoRepositórioSsh.

Ainda temos a forma de acesso de tokens de acesso pessoal, que ai precisamos gerar um token no git para utilizar na hora da senha. Essa opção poderia ser usada quando utilizamos nosso git em outras máquinas.

Para gerar o token precisamos ir em settings, development settings, personal access tokens, classic, generete a new token, dai escolhemos a data para expirar o token e marcamos a opção repo e depois geramos o token, dai precisamos tomar cuidado com o token pq não conseguimos visualizar o mesmo de novo.

**Git na prática, primeiros comandos:**

Git init : utilizando para fazer o github começar a gerenciar a pasta que será o repositório. O repositório precisa estar previamente criado no pc (cria um repositório).

Depois utilizamos uma flag junto com o comando ls que lista os arquivos, para verificar a pasta git gerencial que foi criada. La dentro vão ter estruturas.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Ainda na pasta iniciada, precisamos fazer uma config com o comando

git config –global user.email “Seu email ”

git config --global user.name pedropaulodev

Depois criamos um arquivo no formato MD (mark down) que é uma espécie de html mais fácil, para colocar uma simples receita.

Depois subimos essa receita para o git através do commit com os comandos

git add \*

git commit -m “nome do commit”

Texto

Descrição gerada automaticamente

**Aprofundando os conceitos**

Quanto utilizamos o git init criamos um repositório

Linha do tempo

Descrição gerada automaticamenteTracked ou untracked: rastreável (visível) ou não rastreável.

Untracked é quando o github ainda não tem ciência do arquivo. Uma vez que nós adicionamos o mesmo no repositório ele se torna tracked, ou seja, o github sabe da existência dele.

Um arquivo que está em um repositório pode ter 3 estados, Unmodified, quando nós só adicionamos, mas não fizemos alterações.

Modified quando já fizemos alterações e Staged quando nós utilizamos um git add e ele passa um estado de espera para fazer parte de outra coisa e essa coisa no caso é um commit.

Quando o staged passa a se tornar o commit ele ganhar uma assinatura com um nome, uma descrição que dá significância ao próprio objeto commit com seus arquivos.

E por último o commit se torna Unmodified para renovar o ciclo. Nesta última etapa o github faz o snapshot, que é uma foto que guarda esta última versão não alterada, trazendo assim a capacidade de entender o que vai ser alterado na próxima modificação com base no estado não modificado anterior.

**Entendendo o que é um repositório do git e do github.**

Um sistema distribuído é aquele que está localizado no ambiente de desenvolvimento e em um externo.

Interface gráfica do usuário, Site

Descrição gerada automaticamenteno computador nós temos o working directory, que é o nosso pc, e a staging área que é a parte que os arquivos modificados esperam para virar commits (atualizações verdadeiras), uma vez que os arquivos se tornam commits eles vão para um repositório local (sua máquina). A questão aqui é que só os arquivos que estão no seu repositório local podem ser movidos para um repositório externo, igual ao do github.

**Aplicando os conceitos teóricos.**

git status : nos mostra qual o status dos arquivos

mv NomeDoArquivo ./ NomeDoNovoRepositório : comando para mover um arquivo para outra pasta.



Nós criamos uma outra pasta e movemos o arquivo strognoff de carne para essa pasta “receitas” e utilizamos o git status para verificar os status dos arquivos, porém como não iniciamos a pasta receitas como um repositório, e nem fizemos estas mudanças para a staging área o git nos devolve informações de que o arquivo foi deletado, mas na verdade só não registramos esses movimentos no processo.

Texto

Descrição gerada automaticamente

Então utilizamos o comando git add e o nome do arquivo que foi movido e o nome da pasta que foi criada, agora os arquivos estão prontos para serem comitados.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamentecomitando as ações.

Verificando se está tudo certo.

Texto

Descrição gerada automaticamente

Index.

No git nos criamos um arquivo que explica o que possui aquele commit e qual a sua função, esse arquivo fica de fora do repositório principal, como uma espécie de capa de boas vindas.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Depois utilizamos o git add \* para tornar todas as ações tracked (staged), e depois utilizamos o git commit – m “ adiciona index” para commitar esse index.

**Utilizando de fato o github.**

É importante que o email e a senha que usamos no git, sejam os mesmos para fazer a conta no github. E para verificar nos utilizamos o comando git config –list

Depois criamos um repositório novo no github, your repositories > new .

Texto

Descrição gerada automaticamentecolocamos o nome, uma descrição, se é publico ou privado, e podemos adc o readme caso não tenhamos criado um para contar a história do repositório.

Uma vez criado o repositório no git, precisamos apontar nosso repositório local para ele, fazemos isso pegando o endereço que é um https.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Depois fomos no nosso repositório local que é a pasta livro-receitas e apontamos ela como origem utilizando o comando

Git remote origin CaminhoDoGit

Git remote -v é utilizado para verificar os repositórios remotos cadastrados.

Git push origin master é utilizado para empurrar os arquivos do repositório local para o remoto.

E pronto, depois de fazer o push e validar as credenciais meus arquivos já estão no repositório remoto

Tela de celular com aplicativo aberto

Descrição gerada automaticamenteo legal de criar um arquivo READM.md é q ele já apresenta a história do repositório em questão.



clicando naquela parte em vermelho é possível ter todas as informações de mudanças que aconteceram nesse commit de forma visual. E na parte azul podemos navegar pelo histórico de commits executados, pois um sempre vai estar ligado a outro anterior



**Resolvendo possíveis conflitos.**

**Conflito de merge:** acontece quando duas pessoas estão colaborando em um projeto, porém as duas pessoas pegam uma versão especifica desse projeta e fazem alterações, a primeira pessoa que fizer o commit não terá problema, porem a segunda pessoa terá problemas caso as alterações sejam na mesma linha de código por exemplo, pois o github vai identificar que a segunda pessoa esta com uma versão desatualizada pois não é a ultima versão que é o commit da primeira pessoa.

Quando essa divergência de arquivos acontece pode aparecer um erro.

Texto

Descrição gerada automaticamente

Quando isso acontece, nos precisamos utilizar o comando git pull origin master para puxar o ultimo commit e resolver a divergência no próprio arquivo.

Depois e só fazer o commit novamente.

**Como baixar um repositório.**

Para clonar um repositório precisamos ir no repositório desejado > code e depois copiar a url

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente



Dai e só ir no bash e usar o comando git clone + a url.



Pronto, já temos um repositório novo que queríamos, porém este já aponta para a sua origem

Pois já vem configurado.

Teremos vários desafios de projeto e código que entregaremos pelo github, criando um repositório e enviando o link dele com uma descrição na plataforma da dio

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

O primeiro é o próprio desafio do modulo de git, vamos criar um repositório para os nossos desafios.

Nós fizemos o processo ao contrario, primeiro criamos um repositório no github, depois clonamos ele para nossa maquina através do git clone.