**Aula 1: Introdução ao Git: Entendendo o que é Git e sua importância**

**Otávio Reis**

Porquê aprender a isso? Porquê é relevante:

Analogia do jornalista criando versões de texto e fazendo vários arquivos → Você cria novas versões de um arquivo, ao invés de ficar salvando-as com vários nomes: confusão.

Git: aplicação criada pelo Linus Torvalds: (Criador do linux)

Sistema que versionasse o trabalho e sustentasse acessos de toda equipe. (é um grandíssimo google drive de informática)

Já tinham sistemas de versionamento, mas o Linus não gostava e criou o Git.

\*Kernel: software com centenas de milhares de códigos; núcleo do sistema operacional.

Softwares são feitos de forma colaborativa.

Git (open source) e Github são diferentes.

Vantagem de aprender os dois juntos: controle de versão; armazenamento em nuvem (usando toda instrutra da microsoft); trabalho em equipe; melhorar seu código (expõe ele pro mundo); reconhecimento (funciona como uma rede social, então visualização gera reconhecimento).

Curso: Desenvolver um aplicação escrevendo um livro de receita, intercalando teoria e prática.

**Navegação via command line interface e instalação:**

**Aula 2: Comandos básicos para um bom desempenho no terminal**

Navegação: diferente do grafic interface; interage por linha de comando (line interface).

Aprender a mudar pastas, listar pastas, criar pastas/arquivos; deletar pastas/arquivos.

Windows: cd; dir; mkdir; del/rmdir (interagem com comandos diferentes; derivados do linux)

Unix: cd; ls; mkdir; rm –rf

As diferenças tem a ver com os terminais.

Terminal do windows – menu – **cmd**:

Windows 10: tem linux rodando dentro do windows o que possibilita rodar comandos pra notar as diferenças.

Listar, mostrar as pastas, se situar, mostra a lista de diretórios: W: **dir**; L: **Ls**.

Navegar entre as pastas: w e l: “**cd /**” → vai levar pra base do diretório C

Se juntar o resultado “C: \>” com “dir” lista as pastas do C (no L tbm?)

Se quiser uma pasta específica: **cd / nomedapasta** ex: cd /windows (É CD OU DIR? cd)

L: Cd etc ls -> todos programas exe e

L e W: Voltar “cd ..” 1 nivel (tipo sai da pasta)

Limpar tela -> w: cls l: clear

L: cd etc ls

L: atalho: ctrl + L= clear

Tab=autocompletar. Ex: W já completa com windows (evita escrever nomes grandes de pastas e possiveis erros)

W: Criar pasta (make directiory): mkdir

W e L: mkdir workspace (se não aconteceu nada é pq deu sucesso, quem cala consente kkk)

\*rodar o dir pra listar e o workspace vai estar lá, ou seja, foi criada.

L: mkdir workspace -> ls (listar)

L: cd workspace

Navegar na pasta recém criada

W: Cd / wo tab (que vai completar com kspace)

W: workspace>echo (printa pro terminal) hello

C:\workspace>echo hello > (simbolo redirecionador de fluxo que pega o output da função echo e joga num arquivo) hello.txt

(vai criar o arquivo)

Dir pra ver o arquivo

L: echo hello > hello.txt

ls

W: cd .. (pra retroceder uma pasta)

Cls (pra limpar)

Dir (pra listar)

Del (delete) SO PRA ARQUIVO w: del workspace -> S

Vai deletar o que tem dentro da pasta

Seta pra cima

Sair da pasta

Pra voltar o arquivo

Remover o repositorio

Rmdir (remove directory)

Rmdir workspace /S /Q (O QUE EIGNIFICA O QUE?)

L: DELETE

cd ..

rm – rf (recurssivo: pra pastas dentro das pastas; f: force pra na aparecer confirma~ção)

rm –rf workspace

ls -> não vai ter mais workspace

Varianças: **flags** que mudam o formato que o formato é devolvido pra gente

No linux tbm é cd

**Aula 3: Realizando a instalação do GIT**

Era pra instalar o core da versão 2.33.0.2

Mas a minha versão veio 2.38

Linux dentro do windows: ubuntu

**Entendendo como o Git funciona por baixo dos panos**

**Aula 4: Tópicos fundamentais para entender o funcionamento do git**

SHA1; Objetos fundamentais; Sistema distribuídos; Segurança.

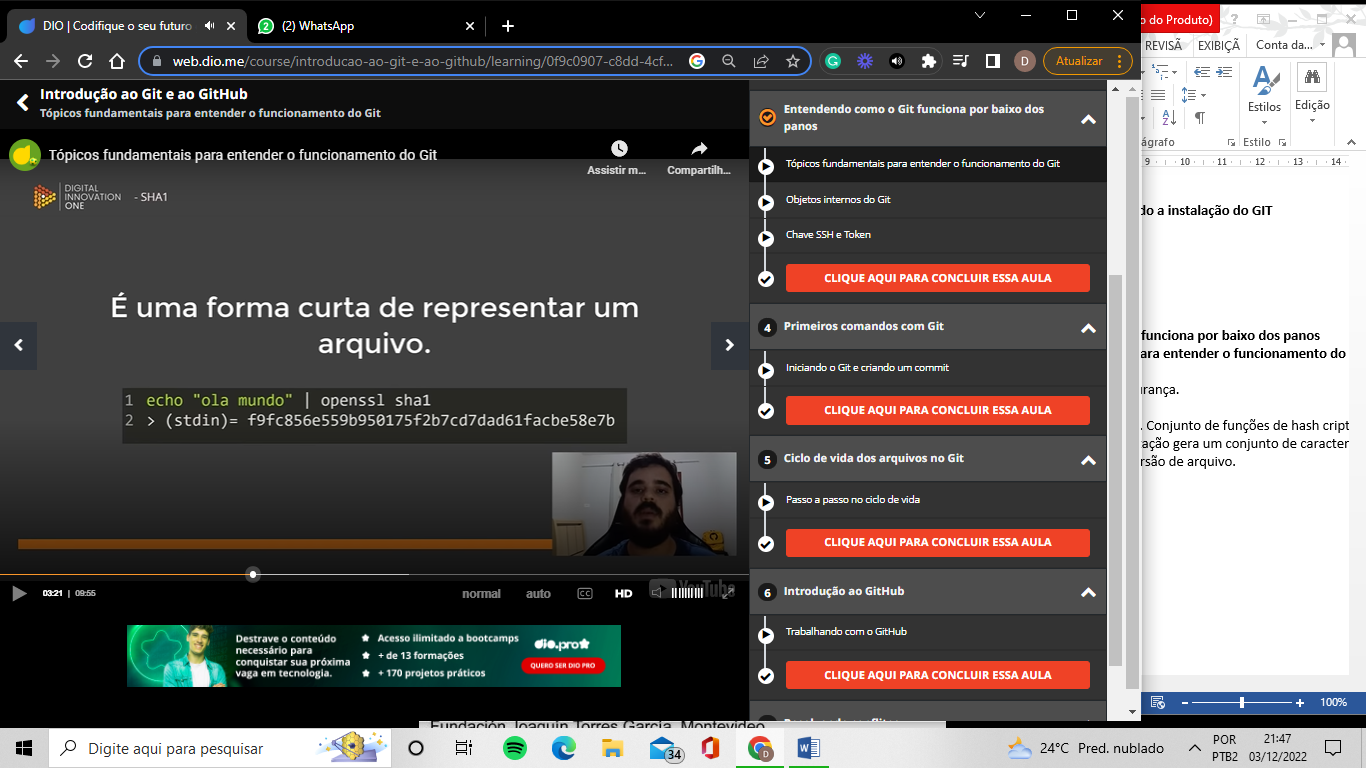
SHA: Secure Hash Algorithm (algorítmo de hash seguro). Conjunto de funções de hash criptografadas projetadas pela NSA (Agencia de segurança nacional dos EUA). A encriptação gera um conjunto de caracteres identificador de 40 dígitos que é único, e serve de identificação pra cada versão de arquivo.

Arq1: adicionei “,” conj1

Arq2: tirei “,” conj2

Arq 3: adicionei no mesmo local “,” conj1

Fora curta de representar um arquivo:



Stdin: dispositivos de entrada padrão

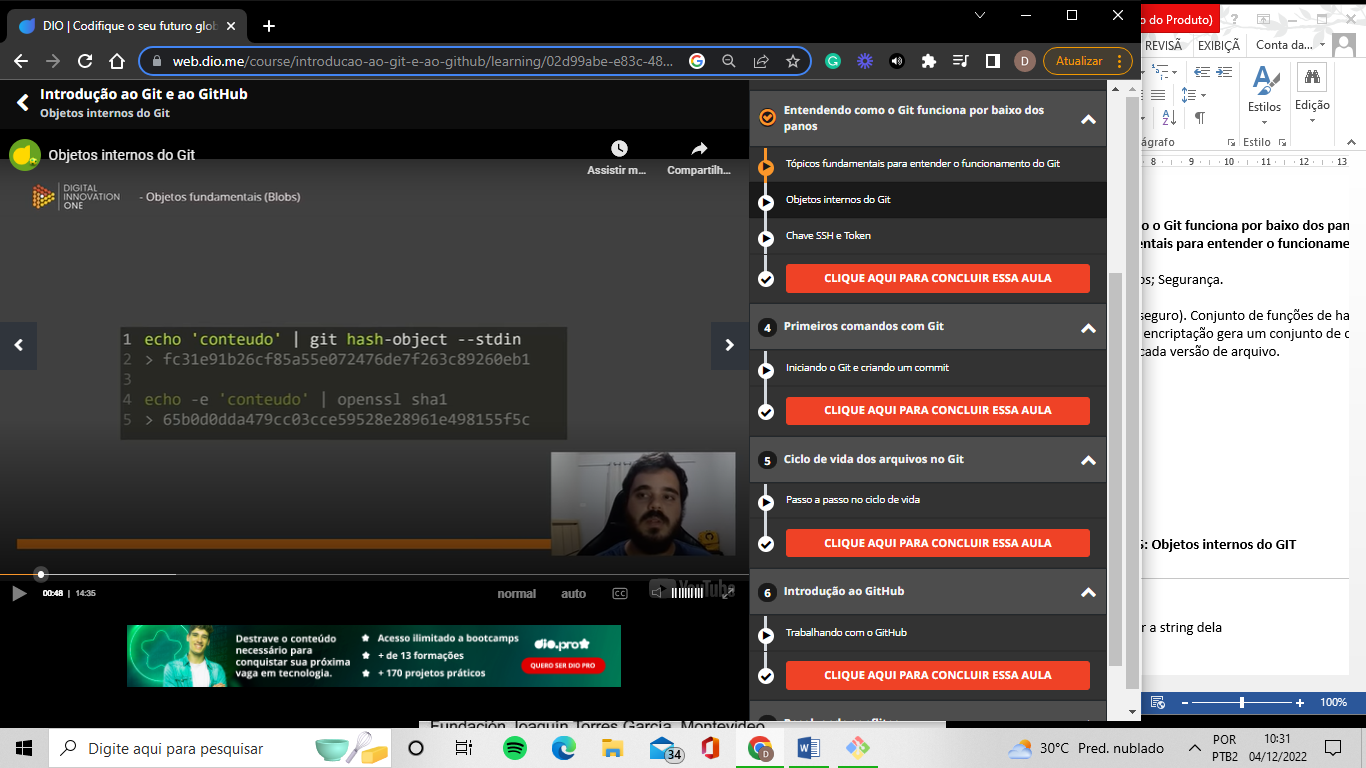
Git bash

**Aula 5: Objetos internos do GIT**

Blobs; trees; commits

Git é distribuido e seguro .

Blobs (objeto binário grande): função echo vai pegar uma output e cuspir a string dela



Função hash object

Stdin (espera receber um arquivo)

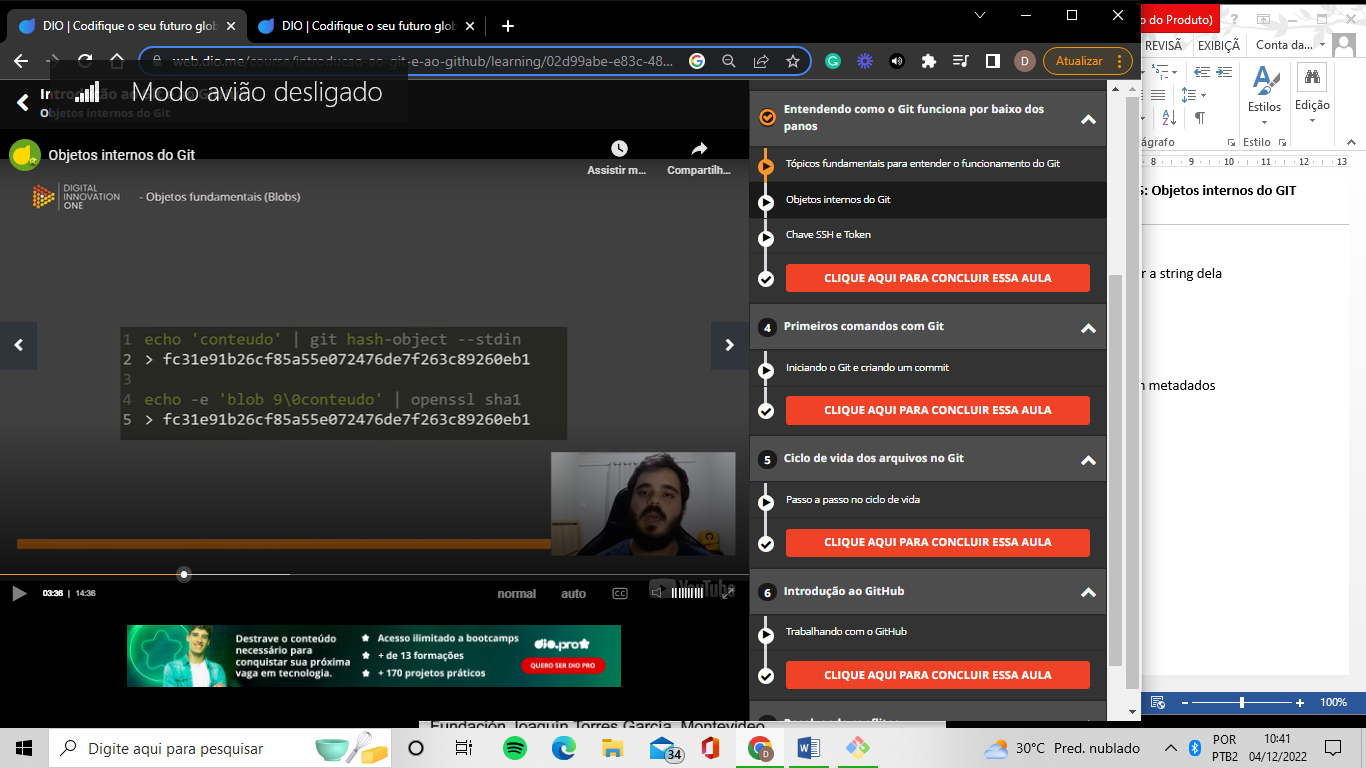
Os arquivos ficam guardados dentro do blob com metadados

Tamanho da string

\0

Estrutura basica do blog

Refez: (botou blob, tamanho 9, \0, conteudo”



Gerou o mesmo sha1

O git guarda arquivos gerando seus respectivos sha para incriptação, armazenando metatadados.

Contem metadados do git (como tipo de objeto, tamanho da string e do arquivo).

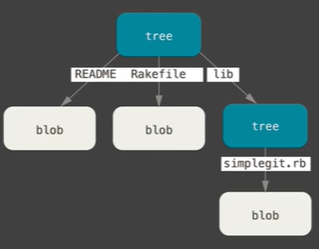
Tree: arvore

Armazena blob, eu é o elemento básico

Também guarda o nome do arquivo, o que o blob não faz

É reponsavel por montar a estrutura de onde esta o arquivo. Podem apontar pra blobs ou outras arvores, já que os diretorios podem conter outros diretorios (seria apontamento recursivo? sim)

Blobs tem o sha1 do arquivo -> arvpres apontam pra esses blobs e tem o sha1 incriptados dos metadados das arvores.



Commit

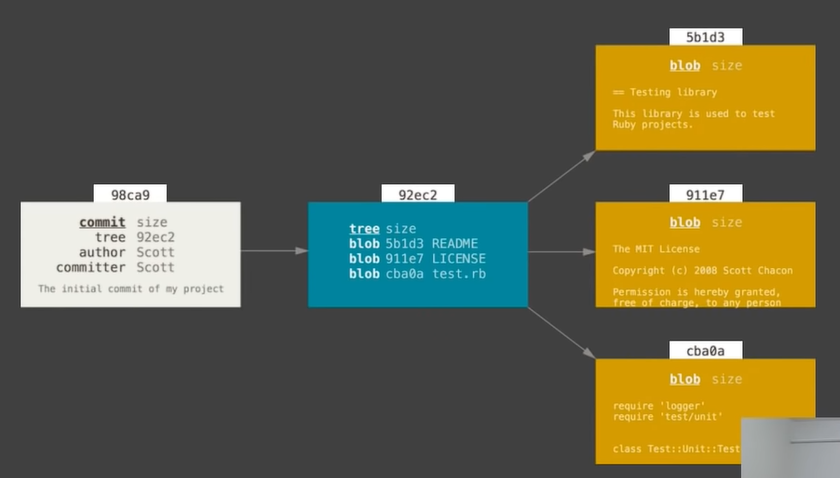
Junta tudo; dá sentido a alteração]aponta para arvore, parente autor ou mensagem. Os dois ultmos dao sentido.

Timestamp data e hora de qd foi criado



Único para cada autor

Estrutura:



Sistema distribuido seguro:

**Aula 6: Chave SSH e token**

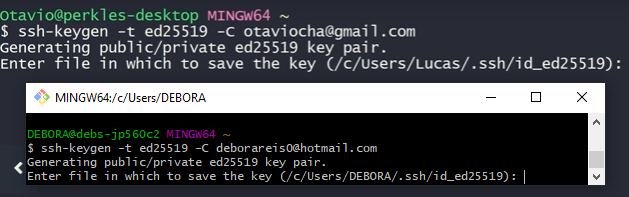
Autenticação mudou 12/08/21

Chave ssh: forma de estabelecer conexão segura e incriptada etre duas maquinas. Chave pública (assinatura da maquina). Eniva codigos sem nem por senha.

Settings → SSH and GPG keys → new SSH key → Gerar chave (parecido no W e L)

Comandos: ssh-keygen –t ed25519(tipodecriptografiadachave) –C deborareis0@hootmail.com

(pastas que começam com “.” São ocultas)



Limpar git bash: ctrl + L

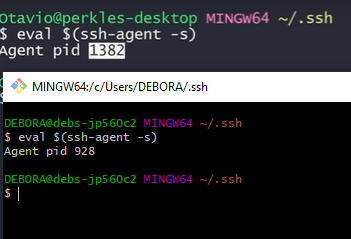
“cat id\_ed25519.pub” mostra a chave publica

Que é a que coloca no github

Pwd

“eval $(ssh-agent –s)” (W e L)

Eval: computa um código javascript representado com uma string; avalia a expressão que a string expressa



Inicializar o ssh agent, que lida com as chaves.

Cria um numero, Starta um projeto, roda em plano de fundo

“ls” mostra as chaves na pasta id\_ed25519



Clonar repositório

Como fazer uma copia pra maquina

Tendo a chave ssh não é só copiar (seria como clonar https)

Copia por ssh

$ git clone git@github.com:deborareis0/DIO-s-Project.git

Confirma, dá ls e aparece o repositório

2ª forma de autenticação segura que o github oferece é pelo token de acesso pessoal.

Options -> developer settings -> personal acess token -> Generate your token

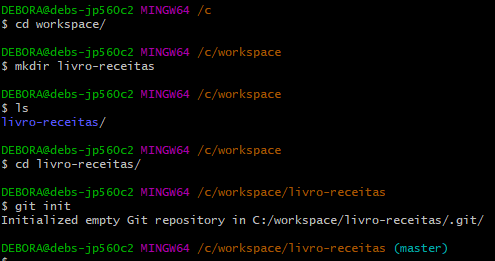
Marcar repo e por nome – gerar – copia e salva – NÃO É MOSTRADO DE NOVO

**Primeiros comandos com GIT**

**Aula 7: Iniciando o Git e criando um commit**

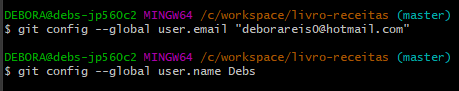
Iniciar git; iniciar o versionamento; criar um commit

Git init (mover); Git add; git commit (criar)



Git init cria um repositório. Pasta oculta .git.

Pra ver a pasta “ls –a” que mostra as ocultas



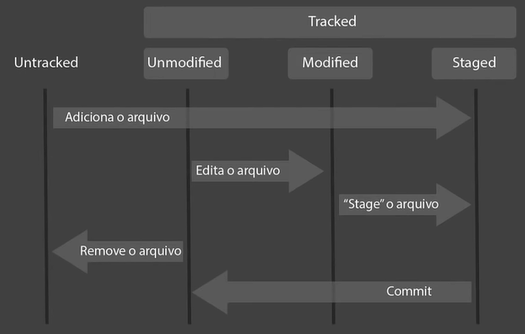


Markdown

**Ciclo de vida dos arquivos no GIT**

**Aula 8: Passo a passo no ciclo de vida**

Git init gera um repositório



Staged: galera da cochia/backstage; etsão se preparando para

Git add: untracked pra staged

Arquivo que ainda não sofreu modificação e é editado, comparando o sha’: unmodified pra modified

Git add de novo vai pra staged

Unmodified pra untracked – não sofreu modificação e foi removido

Commit: retorna ra começar de novo em umodified, cria um snapshot do codigo no commit,

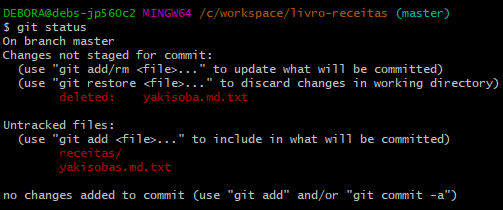


As modificações que faz no codigo da maquina não repercute imediatamente na versao do repositorio remoto

Desen: repositorio de trabalho; varia entre eçe WD e staging area;

Qd faz um commit vai pro repositorio local que pode ser empurrado para remoto

Git status:



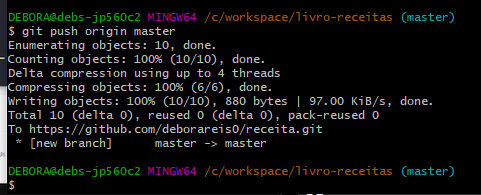
Git add nomeArquivo; Git add\*; Git add .: Movendo, pegando modificados pra staged

Gi commit –m”msg”: pegou os em staged e criou o commit

**Introdução ao GitHub**

**Aula 9: trabalhando com o GitHub**

Repositrio Local para remoto:



Email e name diferente

TENTA BISCAR USUARIO CORRESPONDENTE E NÃO ACHA, não linka a nenhumm perfil

**Resolvendo conflitos:**

**Como os conflitos acontecem no GitHub e como resolvê-los**

**Avaliação:**

