**Estudo Git/ GitHub**

Git Commit grava as alterações no repositório Git Pull - “puxa” as alterações do repositorio remoto para o local (busca e mescla) Git Push - “empurra” as alterações do repositorio local para o remoto

GItHub

git config control l limpa terminal

globol responsavel pela configuracao do usuario

git config —global [user.name](http://user.name) “Ana Carol Wyatt” git config —global [user.email](http://user.email) “ana\_wyatt@hotmail.com”

git config init.defaultBranch main (muda o nome da branch master pra main)

git config — global —list (lista as configuraçoes do ambiente global)

abrir um repositorio novo - atribiu nome, marca add read me e privado ou publico, conforme queira

Para importar esse repositorio para sua maquina local:

via git autenticador app utilizei os passos de autenticação via app, li o token, coloquei o código e o repositório foi clonado automaticamente.

**AUTENTICAÇAO VIA TOKEN**

no repositório no seu github

* vai no botão “code” e copia a url https \*vai para o git bash e comando: \*\*git clone “url copiada” + enter a depender do tipo de conexão seguir os passos (ex, app autenticador)

para gerar o token via github: No GitHub:

* vai no avatar,
* settings,
* developer settings,
* personal acesses token,
* token classic,
* generate new token class - coloca a senha do git, é direcionado para a pagina de configuracao do token \*gera o codigo de token **(atenção q qdo sair da pagina do token ele não ficará guardado)**

Vai no GitBash:

* git clone url
* nome usuario
* na senha coloca o token

\*\*AUTENTICAÇÃO VIA CHAVE SSH

No GitHub:

\*vai no avatar

* Settings
* SSH and GPG keys
* vai no link da documentação e vai na seção - verificar se ha chave ssh
* segui os passos do tutorial \*\* para chave publica\*\* No git bash
* insere o comando para acessar o diretorio que esta o git
* insere codigo ls ~al/.ssh - lista os arquivos Caso não apareça nada é pq nao ha a pasta no seu diretorio

Vai no tutoria de gerar nova chave ssh:

* insere copiado no tutorial com o seu email
* ssh-kergen *(responsavel por gerar as chaves)* -t ed25519 *(tipo de algoritmo a ser usado) -C (adicionar o comentario de pp da chave)*\*\* “endereço de email” + enter

é gerada chave publica e privada. inserir o local para salvar a chave

* clica enter para manter padrão
* insere frase segura a chave publica e privada é gerada

Agora o que vamos fazer é pegar a nossa chave privada e adicionar ela ao ssh agent como esta no tutorial A chave privada equivale a uma senha. vamos adicionar ela no ssh agent para armazenar ela de maneira segura Volta ao tutorial e segue os passos com os comandos

* é gerado o agent pid ai ja startou nosso ssh agent

vamos adicionar nossa chave privada ao ssh agent

add passa endereço coloca frase segura feito isso adicionou nossa chave privada ao ssh agent Agora adicionando nossa chave publica a conta do github

Volta ao github seção ssh new ssh key coloca o titulo q descreve a maquina de onde vc gerou a chave tipo de chave - atutenticação volta ao gitbash vai no diretorio onde estao as chaves cd ~/.ssh ls e ve as chaves queremos ver a publica entao coloca comando cat id\_ed25519.pub a chave é exibida copia a chave e cola no github clica no botao para adicionar

para testar volta no repositorio code ssh copia a url vai no git bash git clone e cola a url yes frase de segura clone feito a pasta esta no diretorio da maqui na.



* mkdir “nome da pasta a ser criada” (cria a pasta dentro da pasta que vc abriu o git bash)
* cd “nome da pasta” (entre na pasta que vc coloca o nome)
* git init (inicializa o git dentro dessa pasta que a gente entrou e transforma a pasta num repositorio git)
* ls (lista as pastas dentro desse repositorio. existem as pastas ocultas .git. se vc abrir pelo explore nao enxerga mas se colocar no endereço do path sim)
* cat config (abre o arquivo configuraçao que esta oculto dentro da pasta que agora é um repositorio git)
* cd .. (volta um nivel da pasta)

Config sem repositorio clonado  
$ cat config

[core]

repositoryformatversion = 0

filemode = false

bare = false

logallrefupdates = true

symlinks = false

ignorecase = true

agora vamos clonar o repositorio na nossa maquina

* git clone (url copiada no github)
* se quiser ja mudar o nome da pasta na sua maquina coloca

git clone (url) “nome da pasta como vc quer”

config com repositorio clonado

$ cat config

[core]

repositoryformatversion = 0

filemode = false

bare = false

logallrefupdates = true

symlinks = false

ignorecase = true

[remote "origin"]

url = https://github.com/CarolWyatt/hello\_data.git

fetch = +refs/heads/\*:refs/remotes/origin/\*

[branch "main"]

remote = origin

merge = refs/heads/main

Origin é o nome padãro que o git coloca

* git remote -v (para verificar em qual repositorio esta vinculado)

Para adicionar o repositorio local com o repositorio remoto:

* git remote add origin “url do repositorio q quer clonar”

o repositorio sera aberto na maquina mas nao traz o readme pq a gente nao fez clone so startou o git init

Salvando alterações no repositorio local

Abrimos uma pasta nova e iniciamos o git nela

* git status (mostra o status do git e se ha algum arquivo nele)
* touch README.md (cria um arquivo readme em markdown)
* arquivo vazio mostra que nao da pra rastrear

MarkDown

[o que coloca aqui fica de titulo de link] (aquicolocao linhk)

botao windows + . abre opção de emoji

## faz titulo

com o arquivo alimentado vamos gravar

agora ele já aparece com alteração se colocar git status aparece as alterações a serem salvas

$git commit -m”nome do commit”

$git log (para saber o status do commit)

$touch (cria arquivo vazio)

$git status (ai reconhece o diretório)

qdo vc nao quer q um arquivo va para o commit é inserir no git ignore

echo resumos/ > .gitignore (adiciona o arquivo em questão no modo ignore 0 texto na frente é o que é colocado no doc e o arquivo vai se chamar gitignore)

para fazer com que um arquivo vazio seja reconhecido tem que colocar:  
aquivo/.gitkeep

Desfazendo alterações no Repositório Local

caso tenha dado git init na pasta errada

rm -rf .git (remove remotamente e a força o repositório git e todo seu conteudo)

git status te confirma que não é mais repositório git

caso vc faça uma alteração no arquivo e queria desfazer

qdo vc fizer a alteração e salvar o documento

$git status

mostrará as mudanças que foram feitas

git restore NOMEDOARQUIVO

vc desfaz as alterações que foram salvas e retorna o documento para versão anterior.

O git reconhece a mudança do arquivo e retorna o arquivo para o status anterior

git status

confirma que vc nao tem mais alterações feitas

ex: apago o conteudo do meu README.md e salvo

dou git status

mostra as alterações no arquvio README

git restore README.md

ele retorna para status anterior

git status

mostra que nao tem alterações a serem salvas

qdo abro meu arquivo readme ele está lá de volta com todo conteudo que eu tinha apagado

qdo utilizar esse comando tenha certeza que nao quer salvar nenhuma das alterações feitas, pq tudo será restaurado.

**Como alterar a mensagem/comentário do último commit**

$git log

mostra os últimos commits

$git commit –amend -m”novo comentário que vc quer atualizar no commit”

$git log

mostrará o comentário atualizado

Ex

git commit –amend -m”Adiciona gitignore e diretório aulas e resumos”

git commit –amend

abre o editor e você consegue editar

$i (vc consegue editar a frase lá em cima)

quando terminar

$esc :wq

vc sairá do editor

$git log

vc consegue verificar as alterações que foram feitas

**COmo desfazer o commit e retornar para o commit anterior**

existem 3 tipos: Soft mixed e hard

soft: Pega os arquivos que estavam no commit e adiciona na area de preparacao (stage area)

$git log

mostra os commits feitos

vc copia o hash do último commit

$git reset –soft hash\_copiado

$git status

você verá os arquivos que estavam dentro do commit agora na area de preparacao para o commit

$git log

você verá apenas o primeiro commit

vc pode fazer o commit direto pois seus arquivos estao na area de preparação

$git commit -m”soft”

o commit sera feito e tera o comentario “soft”

$git log

verifica as alterações

Mixed: ele pega os arquivos do último commit e devolve para arvore de trabalho. você precisará adicionar ele na área de preparação para aí sim fazer o commit novamente.

$git reset –mixed hash\_copiada (é o default entao se nao coloca nada será feito esse)

ele retorna os arquivos do commit para a árvore de trabalho (untracked file)

ai temos que adicionar para área de preparação

$git add .

adiciona todos os arquivos da stage area no commit

$git commit -m”mixed”

o commit sera realizado com o comentario “mixed”

Hard: Ele apagara tudo do commit que foi feito. nao fica nem na area de preparação e nem na arvore de trabalho

#git reset –hard hash\_copiada

e tudo sera desfeito

$git status

vc verifica que nao tem arquivo alterado

$git log

vc verifica que nao tem mais informações sobre o commit

$git reflog

mostrará todo seu histórico de commit

e ai vc consegue ver os que foram apagados

$mkdir resumos

cria de novo a pasta que apagamos no teste do hard

$touch resumos/aula1.md resumos/aula2.md

cria dois arquivos dentro da pasta resumos criada

$git status

te mostrara as alteracoes feitas para serem levadas pra area de preparacao

$git add.

adiciona todos os arquivos na area de preparacao

$git reset resumos/aula1.md

ele tira o arquivo aula1 da area de preparacao e devolve para arvore de trabalho

**Enviando e baixando alterações com o Repositório Local**

Criamos um repositorio novo com o mesmo nome do repositorio local que ja tinhamos criado

Deixamos sem readme

com o tutorial de primeiro acesso ao repositorio aberto e o git bash iniciamos o processo para baixar o que fizemos no github para nossa maquina

com git status verificamos se nossa arvore de trabalho e area de preparação estava limpa

apos isso seguimos o tutorial do github

conectando os dois repositorios:

$git remote add origin urlqueestanoseututorial (comando que conecta os dois repositorios)

$git push (comando responsável por enviar as alterações do remoto para local) -u(setting upStream config a branch main q esta no remoto, como a branch upstream do nosso repositorio local)

se atualizarmos o repositorio remoto ele ja estara com as alteraçoes locais nele

se formos em commit ja aparecem os commits feitos no repositorio local anteriormente

Também podemos alterar remotamente e atualizar no local

no github temos o icone de edicao, podemos editar diretamente na interface do github e depois fazer um commit via botao (commit changes) e é feito o commit

se formos novamente nos commits veremos esse que acabamos de fazer

alteramos tb um arquivo vazio que tinhamos

Outra forma de editar nogithub e utilizando o editor web

dentro do seu repositorio clica na tecla . do seu teclado

sera aberto a pagina do editor online (que parece um visualcode)

vc faz as alterações nessa interface

(clica no arquivo e vai abrir o arquivo de edicao, tem tb como abrir visualização) depois fazemos o commit via botao colocamos o comentario tb.

voltando ao nosso github e se formos em commit verificaremos tb o commit feito pela interface web

Depois disso fomos ao nosso gitbash

$git pull

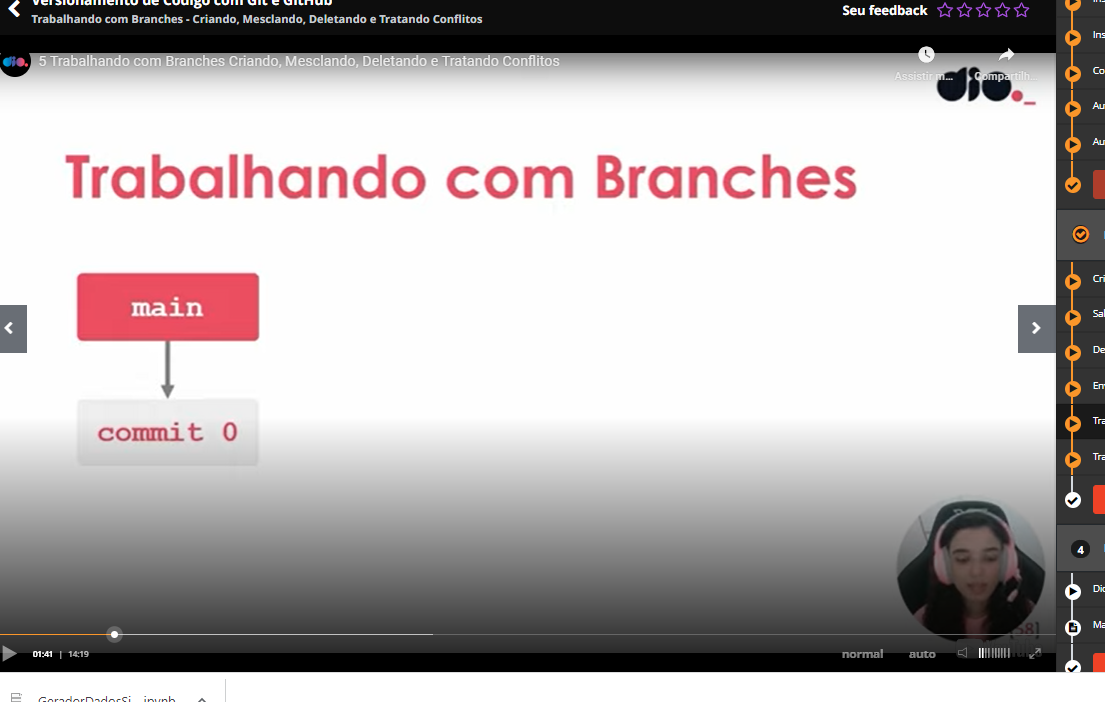
esse comando vai puxar as alterações do repositório remoto e vai mesclar com as alterações que fizemos no repositório local

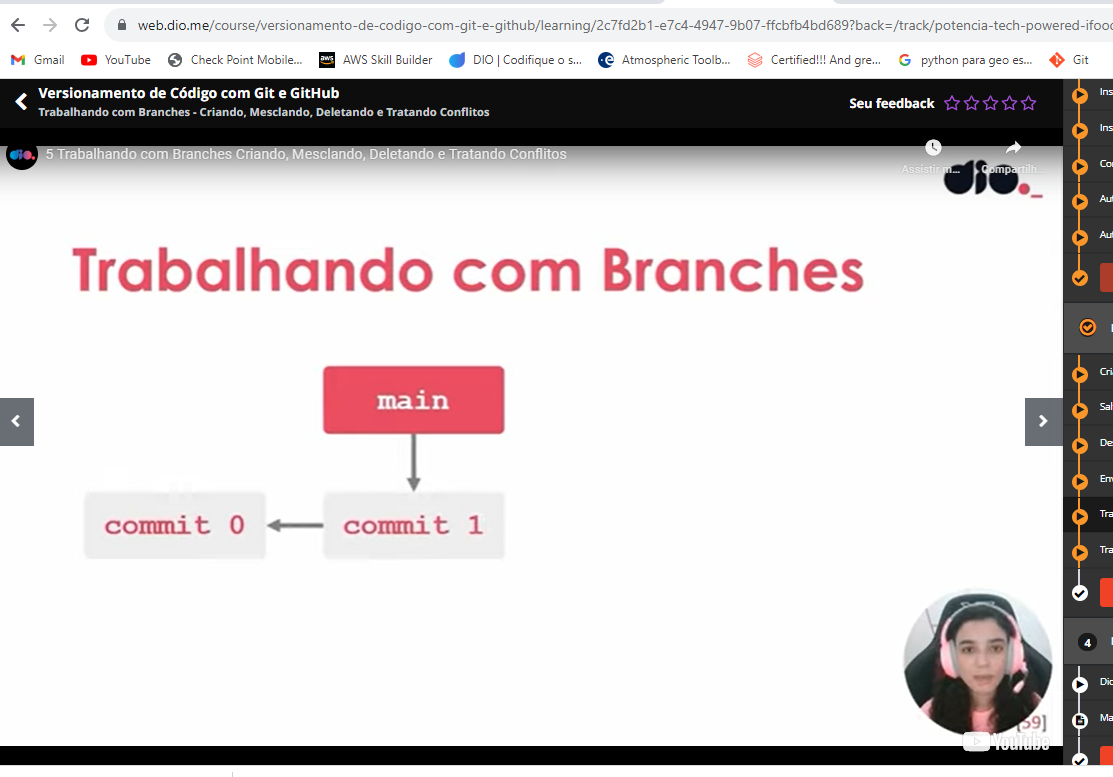
dessa forma os dois repositorios ja estao sincronizados com os mesmos conteúdos.

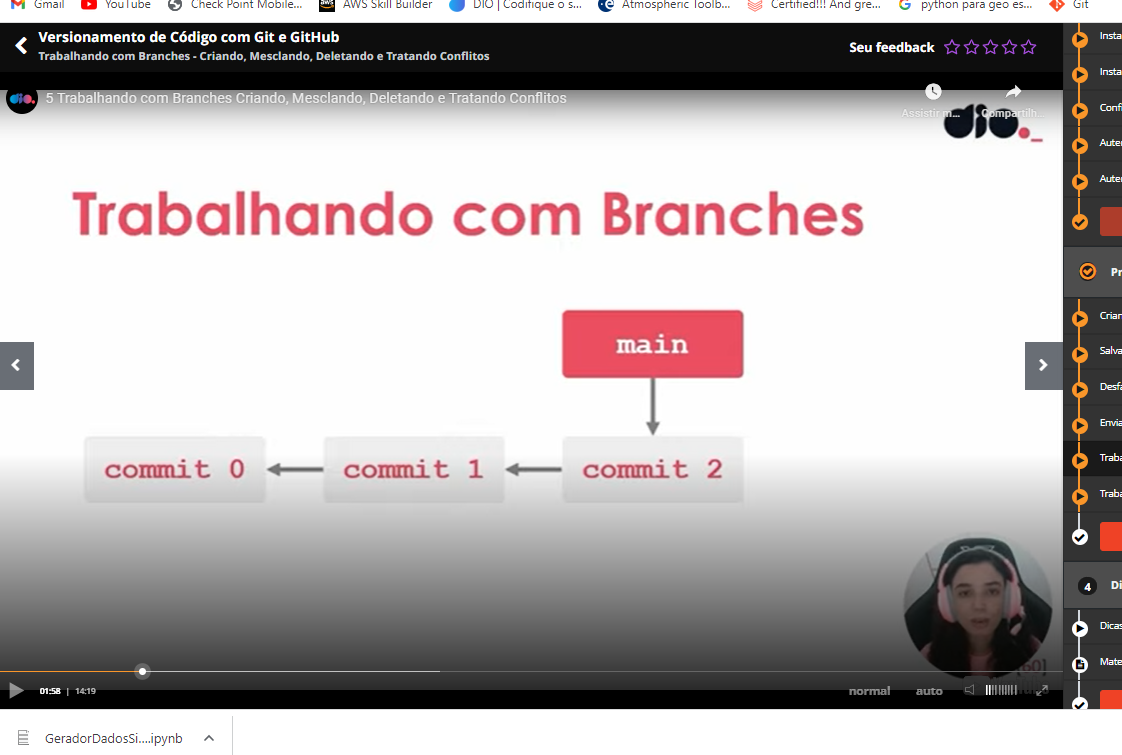
**Trabalhando com Branches**

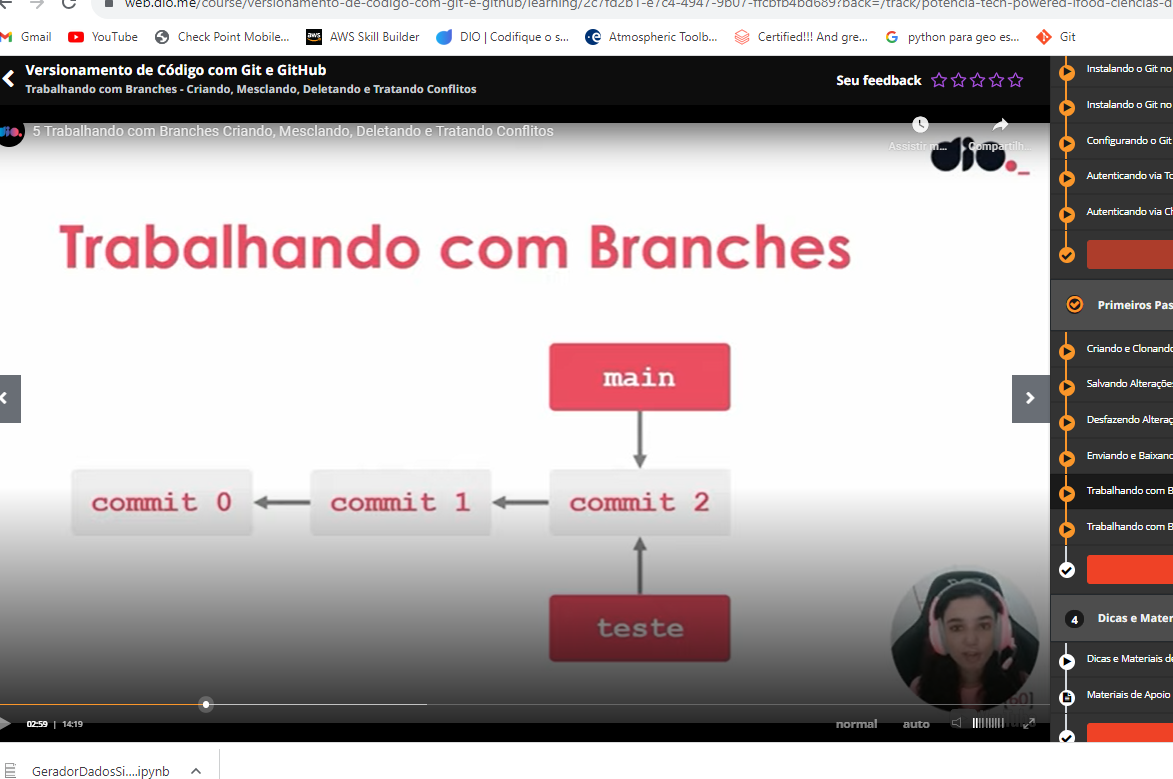
É um ponteiro móvel para um commit no histórico do repositório.

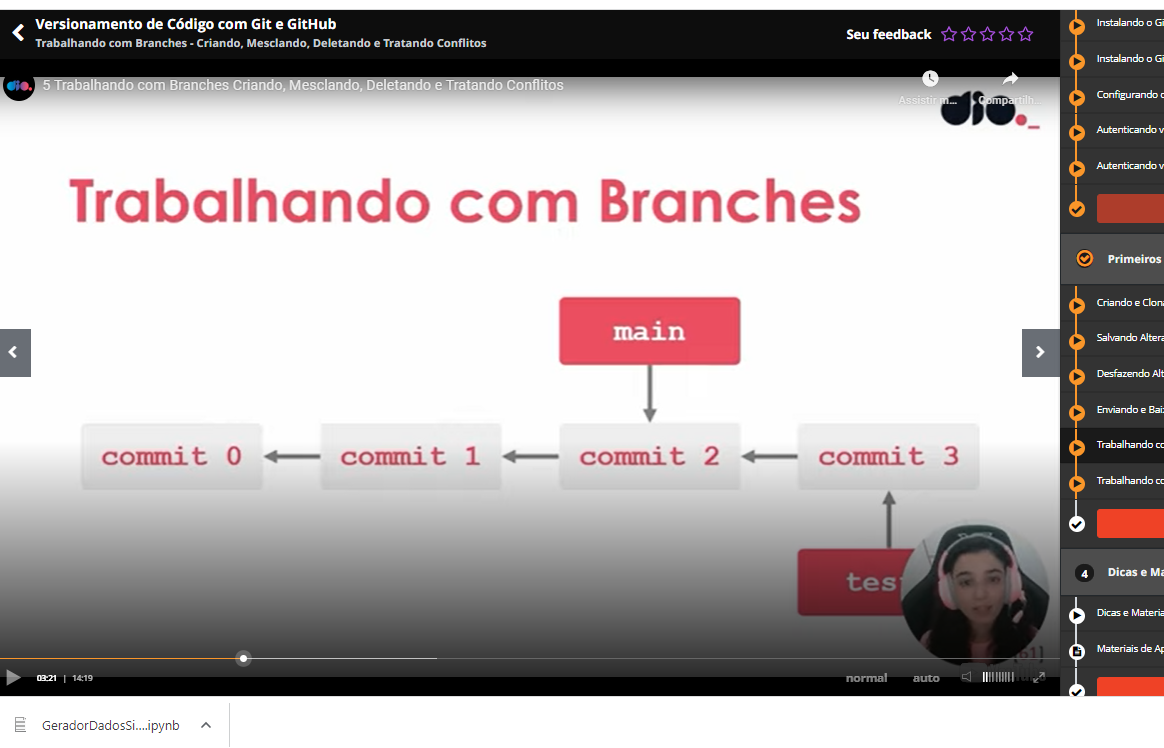
Quando você cria uma branch nova a branch a partir de outra existente, a nova se inicia apontando para o mesmo commit da Branch que estava quando foi criada.











criado um repositorio novo no github

fazemos um commit

fazemos outro commit1

fazemos outro commit2

git log e vemos que a branch main esta apontando para o commit2 e assim vai conforme vamos criando commit

$git checkout -b teste

esse comando vai trocar da branch que estamos e levar para a nova branch que estamos criando “teste”

$git log

mostra que tanto a branch main quanto a branch teste estao apontando para o commit 2

$echo “#commit-3-branch-teste” > commit-3-branch-teste.txt

esse comando cria um arquivo novo (commit-3-branch-teste) dentro da branch teste que criamos

$ git add .

adicionamos o novo arquivo commit-3 para area de preparacao

$git commit -m”commit3”

$git log

percebemos que agora a branch teste esta apontando para a o commit 3 e a branch main esta apontando ainda para o commit 2

Isso é pq as alteracoes que fizemos estao dentro da branch teste e para que elas sejam levadas para a branch principal precisamos realizar o merge, mesclando as informações das duas branchs

$git checkout main

comando que retorna a gente pra branch main

ai se formos no repositório local e olharmos a pasta nao estará mais disponível o arquivo commit 3 que tinhamos no repositório da branch teste

eles funcionam de maneira independente

$git branch -v

mostrará o último commit de cada branch

para sincronizar, mesclar

$git merge teste

e irá ser mesclado e aparecera o arquivo que tínhamos na teste na main

$git branch

listará as branches q temos no repo

$git branch -d teste

excluira a branch teste

$git branch

mostrará que temos agora só uma branch

Conflito de merge

Qando temos alterações concorrentes

Ex. duas pessoas fazem alterações no mesmo documento. Uma localmente e outra remotamente

com um repo com um read me criado fomos ao git bash e fizemos um clone

com a pasta localmente

$git log

mostrara que as duas branches estão apontando para o mesmo commit

depois alteramos o doc localmente depois do clone e salvamos

vamos salvar as alteracoes que fizemos localmente via bash e farems o commit la

fomos ate o repo remoto e alteramos o mesmo doc via github e fizemos um commit

no repo remoto estará apontando apra esse novo commit mas no repo local estara apontando para o commit feito por la.

$git push origin main

para tentar enviar o git recusra pq tem alteraçoes remotas que nao estao na maquina

$git pull

ele retornara o conteúdo remoto mas mostrara um conflito de conteúdo

se abrimos o documento em questão veremos que ela estara com as duas versões. aí escolhemos o que queremos manter e salvamos o documento.

$git status

mostrará que os dois arquivos foram alterados

$ git add.

$ git commit -m”commit feito após o conflito”

$git log mostrará cada repositório apontando ainda para uma branch diferente

$ git push origin main

para estar enviando as informações para o repo remoto

se formos no github e irmos em commit já veremos o commit feito localmente

o doc tb estará na versão mantida localmente

Comando úteis no dia a dia

$git fetch origin main

ele baixa as alterações

$git diff

$git diff main origin/main

para ver as diferenças entre as branches

$git merge origin/main

ele apenas baixa o conteúdo remoto sem mesclar com o conteúdo local

para clonar so uma branch de um repo remoto

vai no repo que quer clonar, pega a url e faz o comando

$git clone urlcopiada –branch nomedabranch – single-branch

se vc nao indicar qual branch quer clonar ele clonara apenas a principal

assim vc clona so uma branch

cd repo

mostrara que foi clonado so a branch que vc indicou2

vc apaga um arquvio

$git status

mostra o arquivo deletado

supondo que vc quer criar uma nova branch dessa agora mas nao quer que essa modificação va junto. o delete desse arquvido.

$git stash

ele vai arquivar a modificação feita

$git stach list

mostra as modificações arquivadas

$git checkout -b teste-2

vc vera que estara num repo com os dois arquivos

$git checkout teste

voltamos pra branch teste

$git stach list

$git stach pop (para trazer as alterações do stach e excluir a modificação feita

$git stach apply

git documentação

referência

git branch

repositorio github da professora elidiana

documentacao git

documentacao github