Git e GitHub

Git

-Software de controle de versão

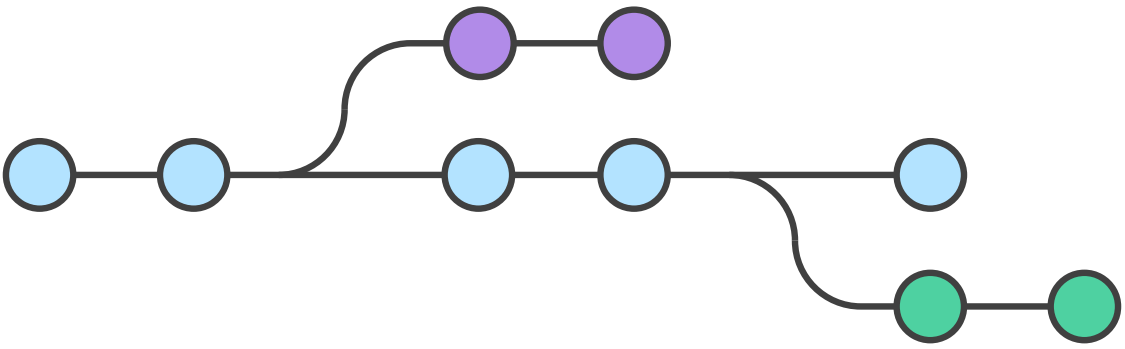
-Versionamento
(estratégias para gerenciar as diferentes versões de um código)

-Repositório local
(você pode jogar para o repositótio remoto com um push)

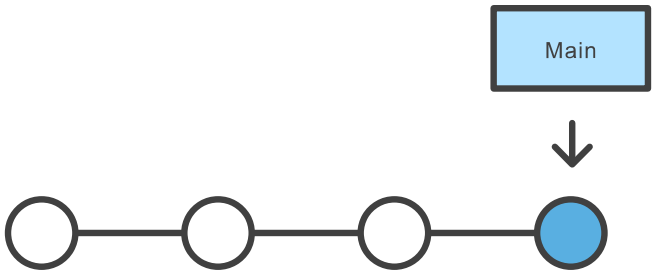
-Commit no rep.local
(Um *commit -comprometer-se* é o ato de enviar e guardar, ou seja, enviar dados ou códigos para armazenamento em um banco de dados ou em um sistema de controle de versão.)
(submeter as últimas alterações do código fonte ao repositório e tornar estas alterações parte da versão principal (head) do repositório)

Principais vantagens

-Controle de Histórico
(te permite 'voltar no tempo',e controlar melhor o código e suas alterações)
-Trabalho em equipe
(facilita execução separada de codigos,e depois a junção dos mesmos em um só projeto)
-Ramificação do Projeto(fork)
(1 progamador front,um back end,e um design,cada um faz seu código remotamente,e fazem um *merge* no projeto final)
-Segurança
(cada um faz seu código,sem interferir inicialmente na versão/parte do coleguinha)
-Organização
(te permite voltar em pontos anteriores do codigo e comitts de maneira muito fácil e organizada)



A **ramificação** funciona como ramos do seu projeto. Isso significa que o seu **projeto principal** fica em uma **branch** (ramo) que atualmente é chamada de branch main.

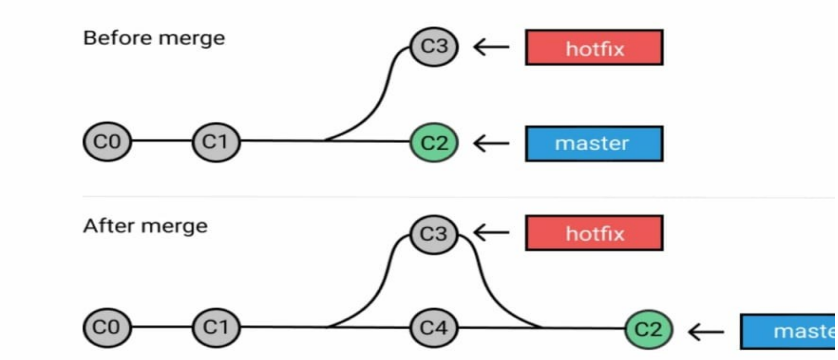


A partir dela você consegue criar outros ramos, ou seja, outras **branches**, que possuem uma **versão do projeto principal** e que você pode realizar modificações sem medo, porque não vai interferir na **branch main**.

Anota aí **:**É de extrema importância ter algo bem claro na hora de realizar um merge, sempre a branch que você está(HEAD)vai ser a que vai "receber" a branch, sendo atualizado para refletir a mesclagem, a branch alvo não sofre nenhum tipo de alteração.

Após cada alteração, é importante que você informe ao **Git** que está na hora dele adicionar essas informações em um **commit**, que é uma mensagem das modificações que você realizou no projeto e que cria um ponto de acesso para essas alterações.

Mesclagem é o jeito do Git de unificar um histórico bifurcado.
O comando git merge permite que você pegue as linhas de desenvolvimento independentes criadas pelo git branch e as integre em uma ramificação única.
O Git merge vai combinar várias sequências de *commits* em um histórico unificado.



git merge é usado sempre depois do *git checkout* para selecionar o branch atual que irá receber e com o *git branch -d* para excluir o branch alvo obsoleto.

GIT

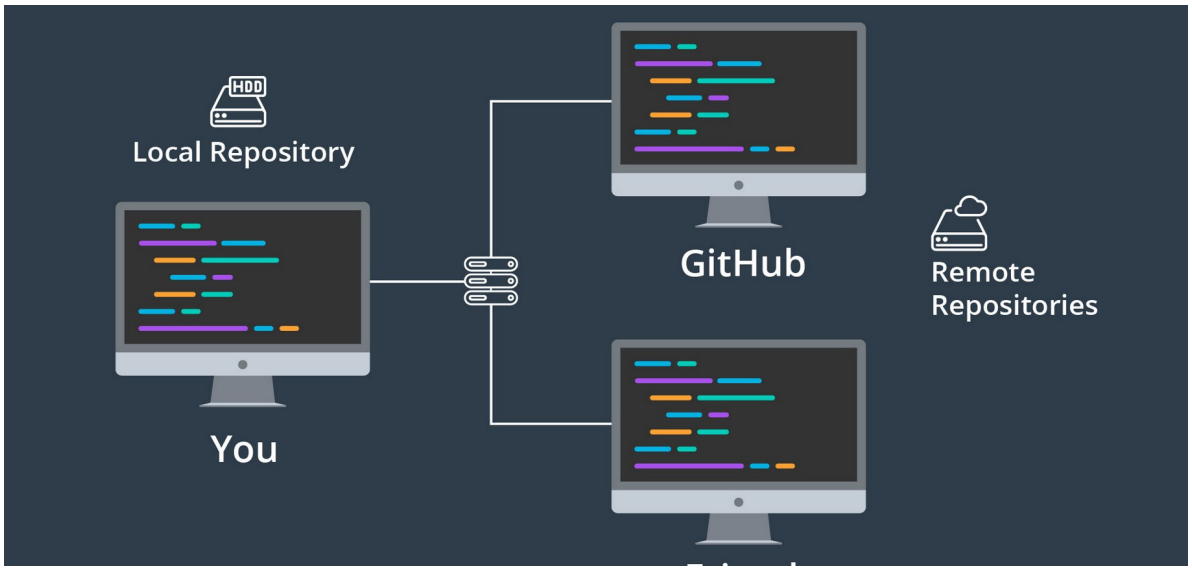
COMANDO	DESCRIÇÃO
git init	Transforma uma pasta em um repositório *git*
git add nome-do-arquivo	Adiciona o arquivo especificado em *staging* (espaço temporário antes do *commit*)
git add .	Adiciona todos os arquivos modificados em *staging* (espaço temporário antes do *commit*)
git status	Verifica o *status* dos arquivos do projeto
git commit -m "Mensagem descrevendo a alteração"	Cria um ponto de acesso para a alteração e possui uma mensagem descrevendo quais alterações foram feitas
git branch	Verifica as *branches* existentes e mostra a *branch* atual
git checkout -b nome-da-branch	Cria uma *branch* nova (a partir da atual) e alterna automaticamente para ela
git checkout nome-da-branch	Alterna para a *branch* especificada
git log	Mostra os registros dos *commits* (alterações realizadas)
git merge nome-da-branch	Realiza a mesclagem das alterações da *branch* especificada

GitHub

-Repositório remoto
(plataforma de hospedagem)

-Rede Social e Portfólio
(acesso ao código de outros desenvolvedores,além de mostrar seus projetos ao mercado)

-Hospedagem de código fonte
(guarda em qualquer linguagem)



Acima vemos um exemplo em que seu repsitório local seria o Git e a hospedagem o Git Hub.

É necessário estabelecer uma **ponte** entre o **Git** (local) e o **GitHub** (remoto), ou seja, você precisa ter uma conexão entre o repositório que está no seu computador e esse mesmo repositório que está remoto.

Duas maneiras de realizar a **autenticação** pra essa ponte:

1-SSH ou ***Secure Shelh*** é um protocolo de criptografia de rede que serve para transferir dados de forma segura mesmo em redes inseguras. Usando o protocolo *SSH*, você pode se conectar ao ***GitHub*** sem precisar digitar seu nome e chave de acesso pessoal a cada comando executado.

2-**HTTPS** ou ***Hypertext Transfer Protocol Secure***: é uma extensão do protocolo de internet *HTTP*, que resumidamente é um protocolo de comunicação entre sistemas, que utiliza certificados digitais para autenticar os dados e permitir que eles sejam criptografados de forma segura.

Criando e enviando Repositórios

Criando um repositório local

informar o diretório em que estão os arquivos que serão gerenciados em um repositório. Isso é feito utilizando o comando ‘cd’ antes do caminho completo.

$ cd /pasta_do_projeto

$ git status(pra conferir)

No GitHub criar novo repositório

-colocar o nome;

-colocar a descrição;

-escolher se será publico ou privado;

-escolher inicializar ou não com Read-me file;

-escolher a licença;

*Abaixo na imagem :como criar um novo repositório com linha de comando

*Acima na imagem: como dar push em um repositório já existente,s partir da linha de comando

Destrinchando o código do PUSH(empurrando do local ao remoto)

$ git remote add (vai add do meu repositório local ao repositório remoto)

o origin (é um nome pro repositório remoto)

o link do caminho do repositório (por SSH OU HTTPS,no exemplo acima git@github.com:andre-noel/meu-app.git

git brain -M main(mudando pra o repositório Main,garantir q estou lá)

git brain (para conferir aonde estou)

git push -u origin main (git push permite que você envie (ou em tradução literal, empurre) os commits de sua branch e repositório Git local para o seu repositório remoto)

Git clone (para clonar (baixar) um repositório do GitHub no seu computador)

-entrar no repositório do gitHub,clicar em code

-selecionar o link SSD OU HTTPS,e copiar

-no terminal entrar na pasta que você quer clonar(cd)

-git clone e o link copiado e enter

-pra conferir entre na pasta(cd) de um ls e confira.

- code . (para abrir o vs code)

Enviando alterações com o git push

-possibilita que as alterações da sua máquina local sejam enviadas para uma máquina remota.

Vamos compreender melhor os 3 comandos tão utilizados

git add= vai preparar as alterações que você fez para serem enviadas, mas aqui, elas ainda não foram enviadas.

git commit= o git commit está “empacotando” as alterações que o comando git add preparou para serem enviadas.

git push = enviará de fato essas alterações.

Na prática:

Para conseguir que as alterações que você tenha feito no seu projeto fiquem acessíveis remotamente, é necessário enviar essas modificações para o repositório remoto.
Para isso:

- Abra o terminal no *VSCode*;
- Adicione o título `# Meu repositório de exercícios` no `README.md` do seu repositório `trybe-exercicios`;
- Utilize os comandos `git add .` e `git commit -m "Mensagem"`;
- Digite o comando `git push -u origin main`.