Tutorial Git e Boas Práticas

by Vinícius Branco

Antes de tudo

- Ter o git instalado na máquina (todos os notes da AGES já estão com git), para quem vai utilizar notes próprios aqui está o link https://git-scm.com/downloads
- Espere que algum dos arquitetos tenham criado o nosso projeto no GitLab da AGES para que possa ser clonado, link do GitLab: http://www.tools.ages.pucrs.br/users/sign_in
- username: <*primeironome.ultimonome*>
- senha: <*matrícula sem o último dígito*>
- Obs: não seja burro e no username substitua “primeironome” pelo seu primeiro nome e “ultimonome” pelo seu último nome, parece óbvio mas vai saber.
- É os guri

Observações

Este tutorial está supondo que você não tem medo de utilizar o terminal :)

Ctrl + Alt + t (para abrir o terminal), caso esteja no windows quando você baixa o git vem um terminal bash, pode utilizar esse que são os mesmo comandos

Obs: Pra quem não é dos guri e não curte derreter no terminal, tem a opção pros Nutella do git usar interfacezinha: https://www.gitkraken.com/

Comandos úteis

Exibir diretórios e arquivos:

ls

Criar uma pasta:

mkdir <nome da pasta>

Acessar uma pasta:

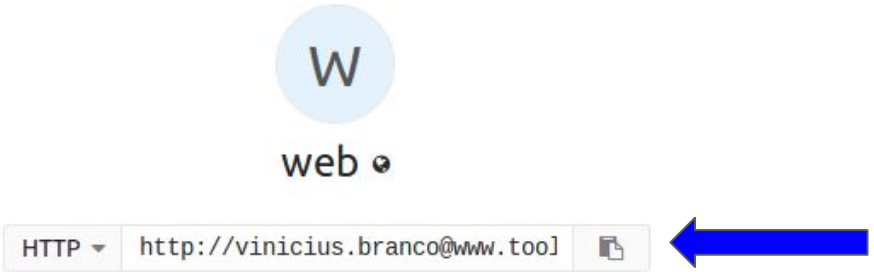
cd <caminho ou nome da pasta>

cd .. (para voltar)

Clonar o Projeto

Para você criar uma cópia do estado atual do projeto no seu repositório local, você deve clonar o projeto. Digite o seguinte comando na pasta em que você deseja guardar o projeto:

- **git clone <*link_do_GitLab*>**
- *link_do_GitLab* = vai ter um baita de um link assim que tu entrar no projeto no Git Lab do lado direito da caixinha vai ter um botão de copiar



E agora?

- Estando dentro da pasta do projeto, a primeira coisa que tu faz é botar o comando ***git branch***, vai te mostrar todas as branchs do projeto e em qual branch tu está no momento
- “Tá meu velho, mas eu nem sei o que é branch”
- Fica firmezinho que no próximo slide tem alguma coisa de teoria sobre o git, se tu já sabe, não julgue o amiguinho que não sabe e pula pro próximo slide, mas eu sugiro tu ler ;-)

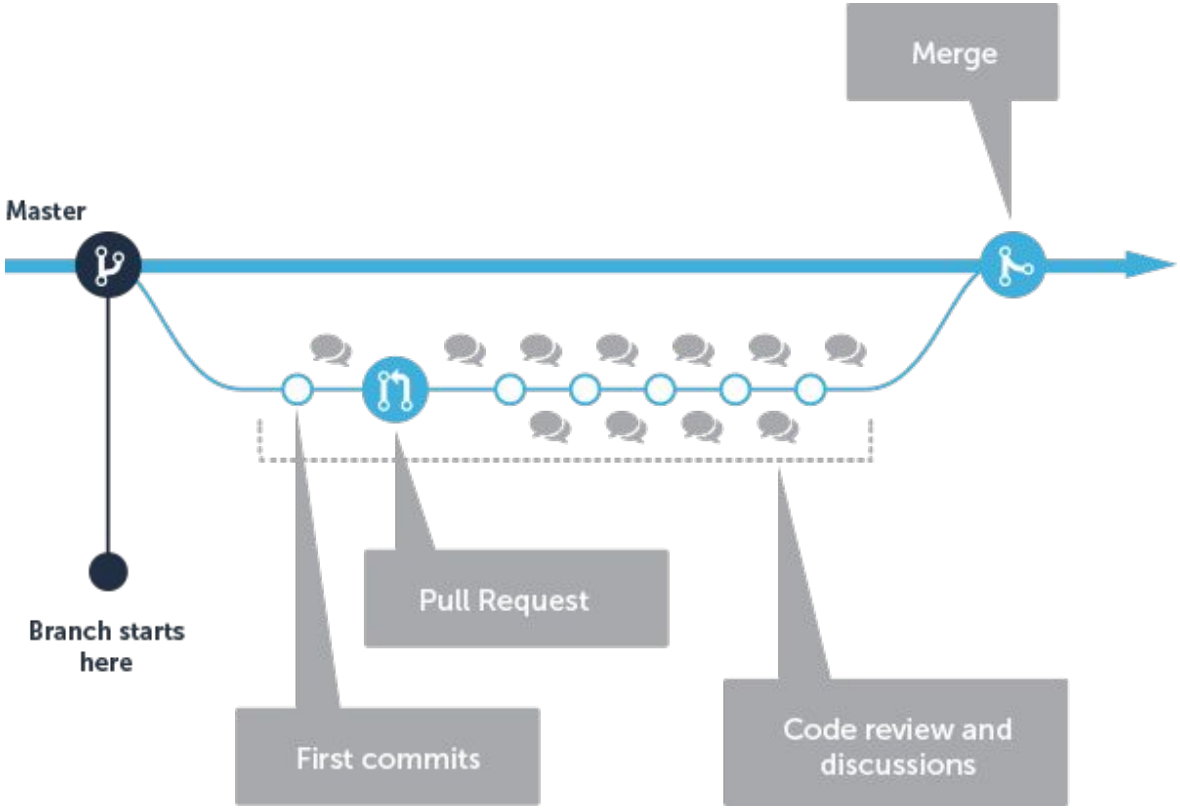
Teoria Básica

Para saber mais: esse é bem simples e bom: https://bit.ly/2r7caue , esse é completo: https://bit.ly/1XhbMEz

Branchs (ramos): São utilizados para desenvolver funcionalidades isoladas umas das outras, por exemplo uma tela de cadastro, então TODAS as tarefas que você pegar no projeto você deve criar um NOVA branch. Após o término da atividade você vai realizar o Merge, que nada mais é do que mesclar a nova funcionalidade que você fez com o estado atual do projeto todo.

Próximo slide tem uma imagem que ilustra bem o funcionamento, a imagem tá fera então deixei um slide só pra ela

Branch e merge ilustrado: ! Importante !



Neste projeto vamos fazer code review, então depois que você jovem realizar a sua tarefa na sua Branch bonitinha, você deverá fazer um Pull Request, para que algum AGES superior ao seu, possa realizar a revisão e fazer o merge.

Próximo slide explica como faz tudo isso

Mão na massa

Para realização da sua tarefa crie uma nova branch a partir da branch *dev* do projeto, está será a nossa branch para desenvolvimento, NUNCA de um commit direto nela (exceto em casos de emergência).

Acessar alguma branch: ***git checkout <nome_da_branch>***

Criar uma nova branch: ***git checkout -b <nome_da_nova_branch>***

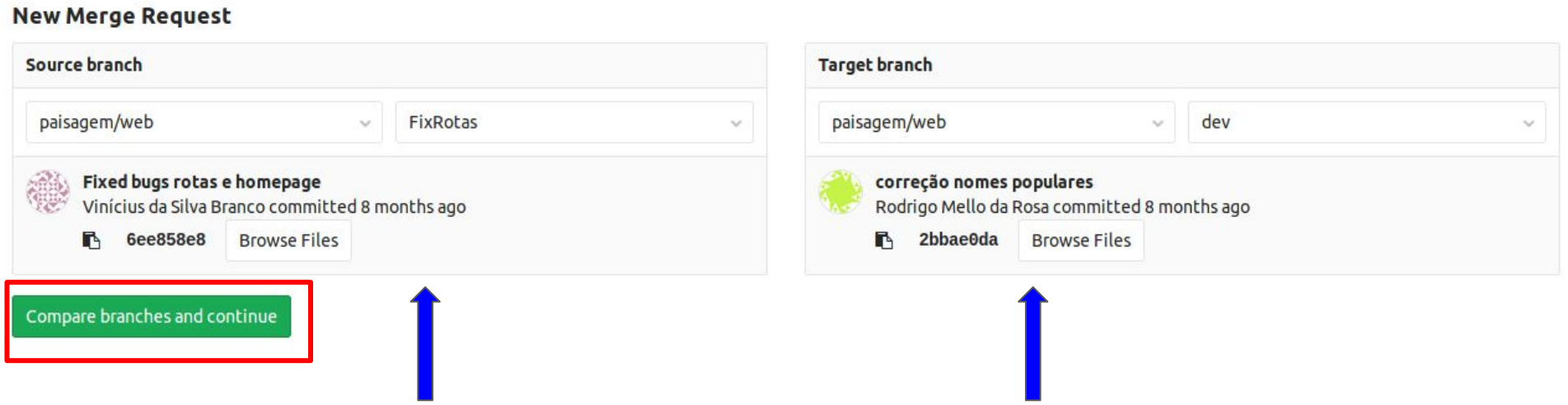
Nome da branch que vocês vão criar é importante, gosto de manter um padrão, por que só de olhar o nome tu já sabe quem está trabalhando nela e fazendo o que, ajuda no code review.

Sugestões de nomes: telaCadastro_ViniciusBranco ou t1h2_ViniciusBranco

Pull Request e Merge

Depois de criado a sua branch e terminado a tarefa, deve ser feito um Pull Request, para que algum AGES superior sem fazer bullying revise o que foi feito e assim realize o merge.

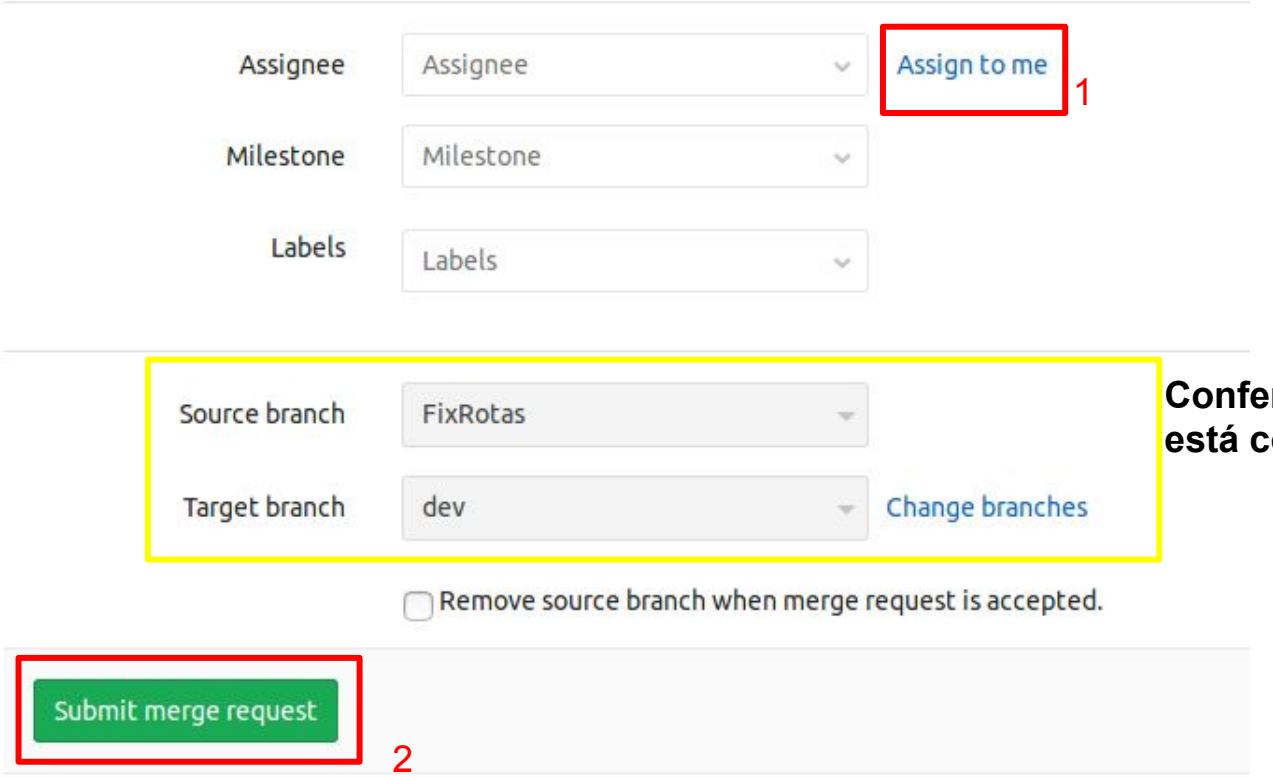
Pull Request e Merge



Branch em que você trabalhou

Branch de destino

Pull Request e Merge



Pull Request e Merge

Após abertura do PR, algum AGES IV/ III ou até II, vai realizar o code review, se necessário, realizar comentários sobre o que foi feito e aceitar ou não o PR, logo após irá realizar o merge.

Algumas vezes se a tarefa se estender um pouco é normal que a sua branch fique um pouco desatualizado em relação a *dev,* então é importante que em tempos vocês atualizem com o seguinte comando:

git pull origin dev

Fluxo commit

Se você está trabalhando em pares na sua branch, antes de começar a trabalhar verifique se há novas atualizações na branch para evitar conflitos, com o seguinte comando:

git pull origin <nome_da_branch>

Fluxo commit

É importante nunca deixar código apenas localmente, sempre que realizar alguma coisa nem que seja bem pouco, subir para o repositório, com os seguintes comandos:

Passo 1:

Adiciona todos arquivos alterados: *git add ** ** (NÃO É UMA BOA PRÁTICA, verifique os arquivos alterados com ***git status* para não mandar nada errado)

ou

Adicionar um determinado arquivo: *git add <arquivo>* (SUGERIDO)

Fluxo commit

Passo 2:

git commit -m “<descrição do commit>”

Por favor, façam uma descrição decente, vocês estão sendo avaliados pelos commits também

Fluxo commit

Passo 3:

Subir para o repositório

git push origin <nome_da_sua_branch>

Agora é só torcer pra não dar merda

Acho que era isso rapaize! Qualquer dúvida, correções, sugestões ou xingamentos é só falar comigo

By Vinícius Branco