Como subir uma branch sem deixar seu Scrum Master louco

1 Acabei meu trabalho, e agora?

Parabéns! Agora está na hora de abrir seu Terminal e acessar a pasta onde esta seu projeto
Você pode fazer isso usando o comando:

```
cd <nome da pasta vô>/<nome da pasta pai>/<nome da pasta filho>/...
```

Quando você acessar a sua pasta, vai ver a branch que você esta. No meu caso, algo assim:

servers git:(master) ✗ //Onde “servers” é minha pasta e “master” minha branch

2 Mas eu NÃO devo subir coisas na master nem develop, né?

Exatamente, seu lindo(a)! Então vamos sair dessa branch e criar uma nova
Você pode fazer isso usando o comando:

```
git checkout -b '<sua branch>'
```

Para trocar de branch Cria branch nova e ir para ela

Depois disso, vai paracer que você esta na sua branch nova. Agora já pode fazer o add e commit!

```
git add .
git commit -m “uma mensagem bonita”
```

3 Uhul! Agora só subir então???

Tá na Disney? Se a gente subir agora, talvez a gente não tenha todas as alterações que estão na nossa branch principal. Então, vamos voltar para nossa branch e pegar tudo que foi alterado.
Para voltar para sua branch principal (vamos assumir que seja a ‘develop’) e pegar o conte que esta nela, usamos os comandos:

```
git checkout develop
git pull
```

Depois do git pull, pode acontecer duas coisas

- **Você não ter nada novo na sua develop**, e receber uma mensagem de “Already up to date.”. Isso significa que a sua develop já esta igual a do git. Agora podemos voltar para nossa branch, subir el e fazer o merge pelo git, pois não vai conflitar. Poc parar o tutorial por aqui!

```
git checkout <sua branch>
git push origin <sua branch>
```

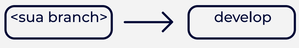
- **Você ter algo de novo na develop**. Se for esse o caso, devemos pegar esse conteúdo e passar ele para nossa branch, e é ai que entra o rebase. Para isso, vamos voltar para nossa branch e falar que queremos fazer o rebase a partir da develop.

```
git checkout <sua branch>
git rebase develop
```

3.1 Ain mas eu sempre subi e fiz o merge, pra que eu vou fazer o rebase?

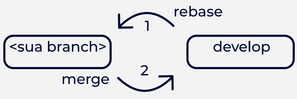
Boa pergunta, jovem gafanhoto! Primeiro vamos entender os dois processos:

Merge sem rebase



Quando fazemos o merge, estamos passando as nossas alterações para a develop, incluindo bugs. Além disso, estamos resolvendo os problemas remotamente e não localmente, então estamos “travando” nossa develop.

Merge com rebase



No caso do rebase, estamos passando o conteúdo da develop para a nossa branch, e já podemos resolver todos os problemas localmente. E caso o app esteja com algum bug depois do rebase, podemos resolve-lo na nossa branch antes de fazer o merge. Depois do rebase com sucesso, podemos subir e fazer o merge para a develop com a certeza de não ter conflitos!

Resumidamente: quando usamos o rebase antes de fazer o merge, estamos “jogando” as alterações da develop na nossa branch, e podemos testar o app antes de fazer o merge na develop. Ou seja, caso tenha algum problema ao juntar as branches, o problema não vai ser nossa branch principal, e isso ainda vai estar apenas localmente. Por tanto, use :)

4 Peraí, e como que eu resolvo os conflitos no rebase?

Primeiramente, é importante saber que nem sempre vai dar confito. Quando não der, vai aparecer uma mensagem como “Successfully rebased [...]”. Se for esse o caso, pode fazer o push da sua branch e o merge dela! Caso tenha conflito, vai mudar o nome da branch que você esta para um id, e vai estar escrito que você esta em estado de conflito. Neste caso, vamos aprender a resolver eles.

1 Temos que achar quais as files que estão dando problema. Para isso, vamos usar o comando:

```
git status
```

Com isso, vai aparecer onde estão os conflitos. Dentro do Xcode, a gente consegue resolver eles, basta entrar nas files que mostraram o conflito. Dentro delas, o conflito fica no seguinte formato:

```
>>>>>>>   Agora é só escolher o código que você quer manter
Codigo 1  e pagar o resto, assim como os '=' e '>'
======
Codigo 2
<<<<<<<<
```

2 Agora que já resolvemos nossos conflitos, podemos continuar no rebase, mas antes disso devemos adicionar as alterações:

```
git add .
git rebase  --continue
```

Importante: É feito um rebase para cada commit novo que existia na develop, logo, pode ocorrer outros conflitos diferentes. É importante resolver todos! Caso queira desistir do rebase e voltar para sua branch como ela estava antes, usamos o comando:

```
git rebase  --abort
```

5 Já deu né, posso subir?

Sim!!! Agora você pode usar o comando de push origin <sua branch>
Qualquer dúvida, pode tirar comigo! Obrigado!

Referências

https://git-scm.com/docs/git-rebase
https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase

Felipe Kaça Petersen

Git