**Fluxos de Trabalho com Branches**

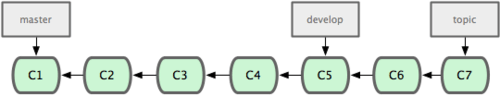
### [Branches de Longa Duração](https://git-scm.com/book/pt-br/v1/Ramifica%C3%A7%C3%A3o-Branching-no-Git-Fluxos-de-Trabalho-com-Branches#Branches-de-Longa-Duração)

A criação e merge de branchs torna possível classificá-los em fluxos de trabalhos. A partir da análise destes fluxos, somos capazes de incorporá-los em ciclos de desenvolvimentos.

O git utiliza um merge de três vias. Fonte, que é o change set do qual se está executando o merge. Destino, o changeset para o qual se está executando o merge. E antecessor, o changeset (ou commit) que é o parente mais semelhante entre fonte e destino. Devido à está característica, é possível realizar o merge de um branch em outro várias vezes durante um período longo de maneira fácil.  Isso possibilita poder usar vários branches sempre abertos em diferentes estágios de um ciclo de desenvolvimento.

Este sistema de trabalho é adotado por muitos desenvolvedores Git. Eles utilizam os branches master somente para códigos já estáveis e aprovados. Em segundo plano, eles possuem outro branch para o desenvolvimento e realização de testes da estabilidade de códigos ainda não liberados. Para isso, ele puxa (pull) branches tópicos (branches de curta duração) quando já estiverem finalizados para passarem por uma avaliação definitiva de estabilidade. Quando os branches de desenvolvimento, comumente chamados de “develop”, chegam a um nível de estabilidade desejado, pode ser feito o merge com os branches master.

Neste tipo de sistema, os branches estáveis estão muito atrás na linha histórica de commits, e os branches de ponta, os que ainda estão sendo desenvolvidos, estão à frente no histórico.



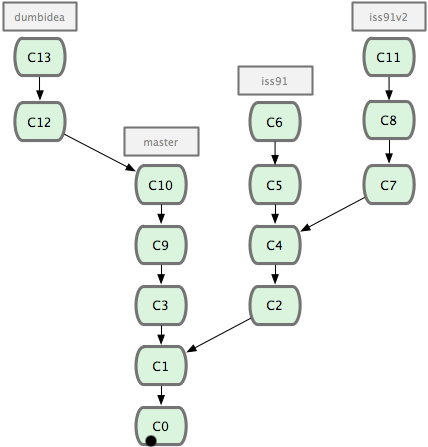
Existem também projetos que, seguindo este mesmo princípio, utilizam vários branches “develop”, de desenvolvimento. Cada um com um nível de estabilidade específico. Quando um destes branches atinge o limite de estabilidade de seu branches, é feito então o merge com seu branch superior. Esse ciclo fica se repetindo até atingir o branch master. Esse método é útil quando se está trabalhando com projetos muito grandes e complexos.

### Branches Tópicos

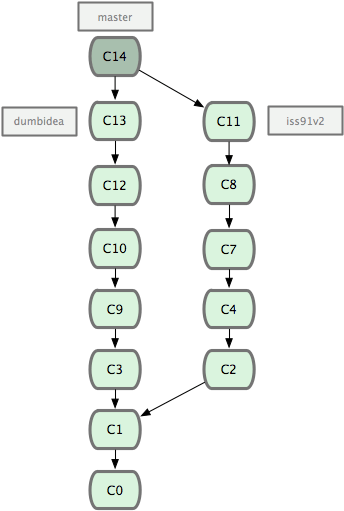
Os branches tópicos, já mencionados antes, são úteis independentemente do tamanho do projeto. Eles são criados para uma funcionalidade específica e por isso são considerados de curta duração.

A maior vantagem destes branches é ter seu trabalho separado em blocos, em contêineres, onde qualquer modificação estará relacionada com ao tópico. Isso torna mais fácil notar erros e visualizar o que acontece durante a revisão do código.

Considere um exemplo onde você está fazendo um trabalho (master), cria um branch para um erro (iss 91), trabalha nele um pouco, cria um segundo branch para testar uma nova maneira de resolver o mesmo problema (iss91 v2), volta ao seu branch principal e trabalha nele por um tempo, e cria um novo branch para trabalhar em algo que você não tem certeza se é uma boa ideia (dumb idea). Seu histórico de commits irá se parecer com a figura a seguir:



Digamos que você decidiu que sua segunda solução é a melhor para resolver o erro (iss91 v2); e você mostrou seu branch dumbidea para seus colegas de trabalho, e ele é genial. Agora você pode jogar fora o branch original iss 91 (perdendo os commits C5 e C6) e fazer o merge dos dois restantes. Seu histórico irá se parecer com a figura a seguir.



REFERÊNCIAS

<https://git-scm.com/book/pt-br/v1/Ramifica%C3%A7%C3%A3o-Branching-no-Git-Fluxos-de-Trabalho-com-Branches>.