**Fluxos de trabalhos com Branches**

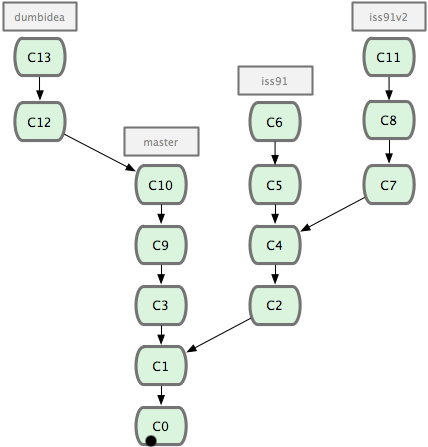
**Branches de longa duração**: levando em conta o Git usa um merge de três vias, fazer o merge de um branch em várias vezes em um período de tempo é geralmente fácil de fazer. Isso significa que você pode ter vários branches que ficam sempre abertos e que são usados em diferentes estágios do seu ciclo de desenvolvimento; você pode regularmente fazer merges de alguns deles em outros.

Muitos desenvolvedores Git tem um fluxo de trabalho que adotam essa abordagem, como ter somente códigos complemente estável em seus branches, possivelmente somente código que já foram ou serão liberados. Eles têm outro branch paralelo chamado develop ou algo parecido que eles trabalham ou usam para testar estabilidade, ele não é necessariamente sempre estável, mas quando ele chega a tal estágio, poderá ser feito o merge com o branch máster. Ele é usado para puxar branches tópicos quando eles estão prontos para ter certeza que eles passam em todos os testes e não acrescentam erros.

Você pode continua fazendo isso vários níveis de estabilidade. Alguns projetos grandes podem ter um branch sugerido “proposed” ou “sugestões atualizadas”, proposed updates que contém outros ideia é que seus branches estejam em vários níveis de estabilidade; quando eles atingem um nível mais estável, e feito o merge no branch acima deles. Repetindo, ter muitos braches de longa duração não é necessário, mas geralmente é útil, especialmente quando você está liberado com projetos muito grandes ou complexos.

**Branches Tópicos (topic):** branches tópicos, entretanto, são úteis em projetos de qualquer tamanho. Um branch tópico é um branch de curta duração que você cria e usa para uma funcionalidade ou trabalho relacionado. Isso é algo que você provavelmente nunca fez com um controle de versão antes porque é geralmente muito custos criar e fazer merge de branches. Mas no Git é comum criar, trabalhar, mesclar e apagar branches muitas vezes ao dia.

Considere um exemplo onde você está fazendo um trabalho no máster, cria um branch para um erro, trabalhando nele um pouco, cria um segundo branch para testar uma nova maneira de resolver o branch para trabalho em algo que você não tem certeza se é uma boa ideia.



Agora, vamos dizer que você decidiu que sua segunda solução é a melhor para resolver o erro, e você mostrou seu branch dumbidea para seus colegas de trabalho, e ele é genial. Agora você pode jogar fora o branch original (perdendo os commits c5 e C6) e fazer o merge dos dois restanres.